Premessa

"BLENDER 3D tutorials – Volume 2 (versioni 2.5x – no basi, no materials)" è un pdf contenente le versioni testuali di **120 videotutorial realizzati con le versioni 2.6x di Blender 3D** e pubblicati gratuitamente da Francesco Milanese sul suo canale Youtube e sul sito *www.francescomilanese.com* .

L'ebook, che consta di 558 pagine in formato A4, ha un prezzo volutamente basso perché si rivolge a chi ha seguito i videotutorial gratuiti, li ha trovati interessanti e desidera avere un testo in pdf (ricco di screenshots tratti dai videotutorial) da poter consultare facilmente, mediante le funzioni di ricerca o stampandolo... oppure a chi desidera ringraziare in maniera concreta l'autore dei videotutorial, ottenendo comunque una raccolta di tutorial su alcune operazioni effettuabili in Blender 3D.

NOTA: di questa raccolta NON fanno parte gli 80 tutorial del corso base, i 3 tutorial per principianti assoluti e i 17 tutorial sui Materials in Blender; tali tutorial (100, in tutto) sono invece raccolti nel "*Volume 1*", pubblicato precedentemente.

SOMMARIO

1. Superfici liquide con Wave e textures animate 1
2. Bandiera mossa dal vento con Displace, NO CLOTH 7
3. Fumo, fuoco e fiamme con Emitter e Smoke - 1 di 2 13
4. Fumo, fuoco e fiamme con Emitter e Smoke - 2 di 2 19
5. Descrizione scena con effetto SSS per sapone 26
6. Occhio in stile cartoon 31
7. Bolle di sapone con particle emitter e texture 36
8. Animazioni materials (materiali delle mesh), curve IPO e Graph Editor 42
9. Sculpt Mode: descrizione strumenti e applicazioni 48
10. Distruggere le mesh con Explode ed Emitter 52
11. Particelle, Force Fields, collisioni e rimbalzi 57
12. Cloth (tessuti) con vento e collisioni per tende e altro 64
13. Effetti speciali - Ologramma 3D 73
14. Armature, ossa, azioni, animazioni - 1 di 5 78
15. Armature, ossa, azioni, animazioni - 2 di 5 82
16. Armature, ossa, azioni, animazioni - 3 di 5 90
17. Armature, ossa, azioni, animazioni - 4 di 5 96
18. Armature, ossa, azioni, animazioni - 5 di 5 102
19. Modellazione e rendering - Bicchiere di vetro 109
20. I modificatori Array e Build 116
21. Consigli e strumenti per la modellazione di ambienti 3D 121
22. Descrizione scena Lapislazzuli di RedBaron85 126
23. Utilizzi del modificatore Solidify: pareti e tessuti 131
24. Animazioni Follow Path (segui percorso) 137
25. User Preferences e colori Weight Paint personalizzati 144
26. Basi del Motore dei Fluidi - 1: introduzione e Domain 147
27. Basi del Motore dei Fluidi - 2: fluid e obstacle 152
28. Basi del Motore dei Fluidi - 3: inflow e outflow 159
29. Basi del Motore dei Fluidi - 4: particelle, particles 163
30. Utilizzare i Vertex Groups. Il problema del cubo 168
31. Basi della modellazione - Estrusione di curve lungo percorsi (Bevel OB) 175
32. Basi della modellazione - Curve Taper per curve estruse con Bevel 178

33. UV Mapping, le basi - Versione REDUX 182
34. Halo Light - Luci ed ombre volumetriche 188
35. Ombre sfumate con Ray e Buffer Shadow 191
36. Cambiare lo shortcut add mesh in barra spazio 196
37. Ombre in tempo reale nei giochi fatti col Blender GE (Game Engine) 199
38. Utilizzare le texture stencil come maschere coprenti 202
39. Logo con effetto Displacement animato 205
40. Mappare e renderizzare textures di pianeti 210
41. Jupiter Flyby, il sorvolo di Giove: analisi di un'animazione 213
42. Modellazione: ampolle, decanter, bottiglie, vasi con spin e solidify 217
43. Animazioni con constraints e Halo Light: tende alla veneziana 224
44. Ombre sfocate con Compositing dei Nodes 229
45. Traslazioni animate delle immagini con Nodes Compositing 232
46. Animazione ruota panoramica con Parent e Constraints 235
47. Animazione Glare selettivo in Compositing 240
48. Modellazione lampada Liberty con Spin, Bevel, material vetro satinato 245
49. Luci natalizie con Curve Bezier, Material Halo e Texture Magic 253
50. Grattacielo con pochi vertici e una texture immagine su NORMAL 258
51. Rilievo su decorazione natalizia con texture su NORMAL 262
52. Finestre floattanti su due o più monitor 265
53. Usare Spin come un compasso per estrudere ad arco, con rotazione 267
54. Tips and Tricks GUI: snap del 3D cursor con SHIFT S 270
55. Differenze tra Spin Dup e Array con Warp per la rotoduplicazione 272
56. Luci natalizie intermittenti con Halo, IPO e F-Curve Cycles 276
57. Bagliore Glare stile Tron con Materiali Multipli e Nodes Compositing 282
58. Testo inciso su superfici, curve e mesh 286
59. Baking delle textures procedurali, anche per il GE 289
60. Lampi ed effetti luminosi col nodo Normal 293
61. Modifiers animati, animare i modificatori 297
62. La fusoliera del Fokker DR.1 con le NURBS Surfaces 301
63. Effetti speciali: ologrammi virtuali in filmati reali 306
64. Particelle lungo un percorso con il Force Field Curve Guide 311
65. Game Engine - Aprire le porte in un gioco con l'attuatore F-Curve (ex IPO) 314
66. Rendering stereoscopici animati (basi) 318
67. Importare ed esportare oggetti, mesh e modelli 3D nelle scene 324
68. Copy Rotation vs Individual Origins 328
69. Far puntare un oggetto ad una telecamera in un'animazione 332

70. Giochi con immagini 3D stereoscopiche col Game Engine 333
71. Lo script Bridge and Loft di Bart Crouch 336
72. Join vs Boolean Union per i modelli STL e la stampa 3D 340
73. Strumenti di selezione di base ed Edge Loop e Ring 343
74. Bande colorate su oggetti con Textures procedurali 348
75. Indirect Lighting in Blender 2.5, esempi di rendering 352
76. Secondo confronto Blender 3D 2.5 con IL e Yafaray; cucina, luce laterale 358
77. Tips and Tricks modellazione: allargare o restringere una selezione 361
78. L'illuminazione indiretta (Indirect Lighting) in Blender 3D 2.5 363
79. Tips and Tricks: trasformazioni con assi vincolati (locking) 371
80. Lo script Import Images as Planes 373
81. Impostare la Active Camera per rendering con più telecamere 376
82. Quando Blender è più furbo di noi: il caso Lens Flares con Halo 379
83. Magnolia: un rendering con Indirect Light e Glare 382
84. Aggiungere rumore ad animazioni digitali e stills in Compositing 387
85. Un buon motivo per salvare in OpenEXR i rendering realizzati con Blender 3D 2.5 390
86. Oggetti in fiamme con Smoke e Particles Emitter, un esempio pratico 394
87. DupliVerts in Blender 3D 2.5, esempio 1: fori su mesh con Boolean Difference 399
88. Textures con mappatura UV sui due lati di una mesh con i Material Nodes 405
89. L'attuatore Constraint Force Field per gravità o forze repulsive nel BGE 2.5 410
90. Realizzare un Tour Virtuale 3D con il Blender Game Engine 2.58, prima puntata 413
91. Realizzare un Tour Virtuale 3D con il Blender Game Engine 2.58, seconda puntata 417
92. Realizzare un Tour Virtuale 3D con il Blender Game Engine 2.58, terza puntata 422
93. Realizzare un Tour Virtuale 3D con il Blender Game Engine 2.58, quarta puntata 428
94. Realizzare un Tour Virtuale 3D con il Blender Game Engine 2.58, quinta puntata 433
95. Sequenze di Matte Images con le Roto Bezier e il Nodes Compositing 439
96. Noise sulle curve IPO per campi stellati animati ed altro 446
97. Rapporti, rettangoli e triangoli aurei e le Composition Guides 449
98. La grafica dell'App "Newton's AG" con Halo, Smoke e Compositing 453
99. Fuoco e fiamme con Smoke: la color ramp, high resolution e IL 458
100. Burster: plugin per eseguire applicazioni e giochi fatti col Blender GE nei browser 464
101. Creare Normal e Bump Maps dallo Z-Buffer di un rendering 467
102. GUI: le voci del menù Align View per allineare vista e telecamera 471
103. Sfocare il Bake di una mappa Ambient Occlusion col Nodes Editor 476
104. Inserire oggetti nella scena durante il gioco nel BGE 2.5 480
105. Modellazione: scaling a 0 per allineare vertici o appiattire mesh 483
106. Rendering della 3D View con OpenGL Active Viewport 485

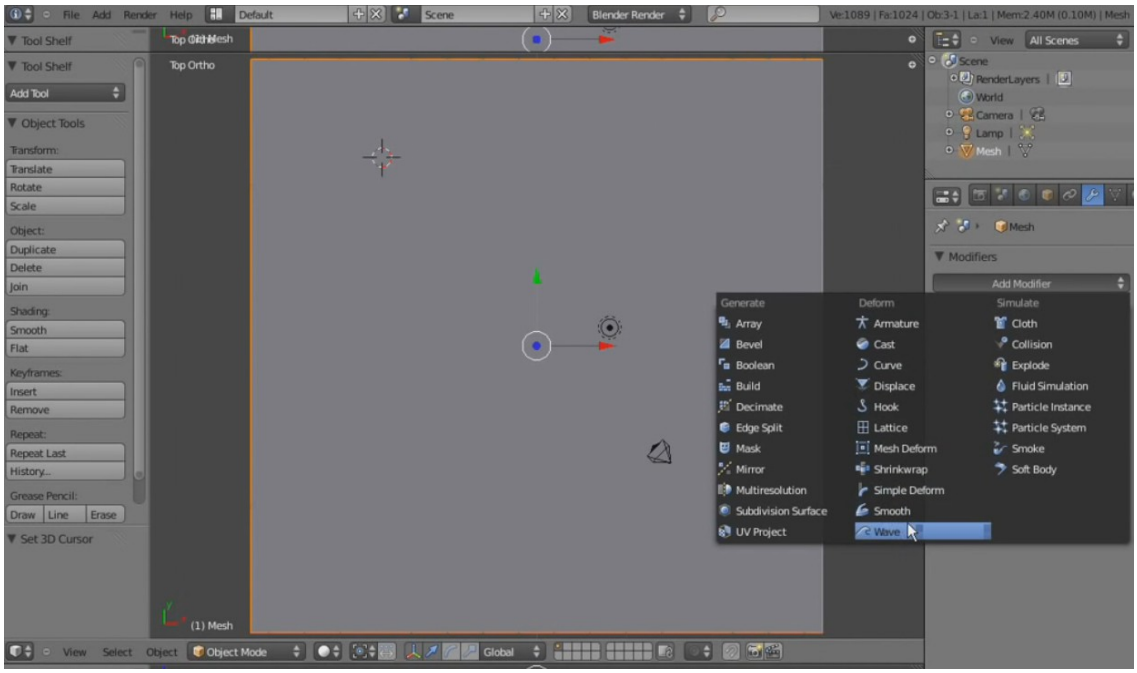
107. Floor per posizionare oggetti su pavimenti o pareti 488
108. Tips and Tricks: le voci di Object Display 491
109. Animare le Layers Relations per far apparire e scomparire le mesh 494
110. Effetti Glow sui riflessi speculari col Compositing 498
111. Modellare filettature e scalanature per viti, bulloni ed altro 501
112. RenderLayers per effetti di Nodes Compositing su Smoke 512
113. Videoclip da sequenza di immagini con il Sequence Editor 517
114. Edges sui rendering delle mesh via Nodes Compositing 520
115. Animazione particelle e scrittura su file con script Python 526
116. Semplice script Python per ancorare le mesh a punti della scena 531
117. Textures animate nel BGE 2.59 con le Sprites (tiled images) 536
118. Modellare un portasapone con Loop Cut e Subsurface 540
119. Immagini HDR su Environment Lighting per oggetti riflettenti 545
120. Modellare e texturizzare l'oculare di un telescopio 548

* * *

Superfici liquide con Wave e Textures animate

Questo tutorial è dedicato all'utilizzo del modificatore *Wave* di Blender 2.5 per simulare la superficie del mare. Non verrà fatto uso del motore dei fluidi, per cui la tecnica qui mostrata non può essere utilizzata per ottenere simulazioni fisiche, quanto piuttosto semplici animazioni: si tratta di un piano leggermente "increspato", ma non vengono create particelle, schizzi e così via, per cui è del tutto inappropriato per simulare realisticamente, ad esempio, il mare in burrasca; per quanto riguarda l'aspetto del materiale, invece, l'effetto finale dipende da come verranno create e mescolate le Textures, per cui si può andare da un effetto cartoon (in questo caso, sì, buono anche per il mare in tempesta) ad un effetto fotorealistico, buono per uno specchio d'acqua calmo.

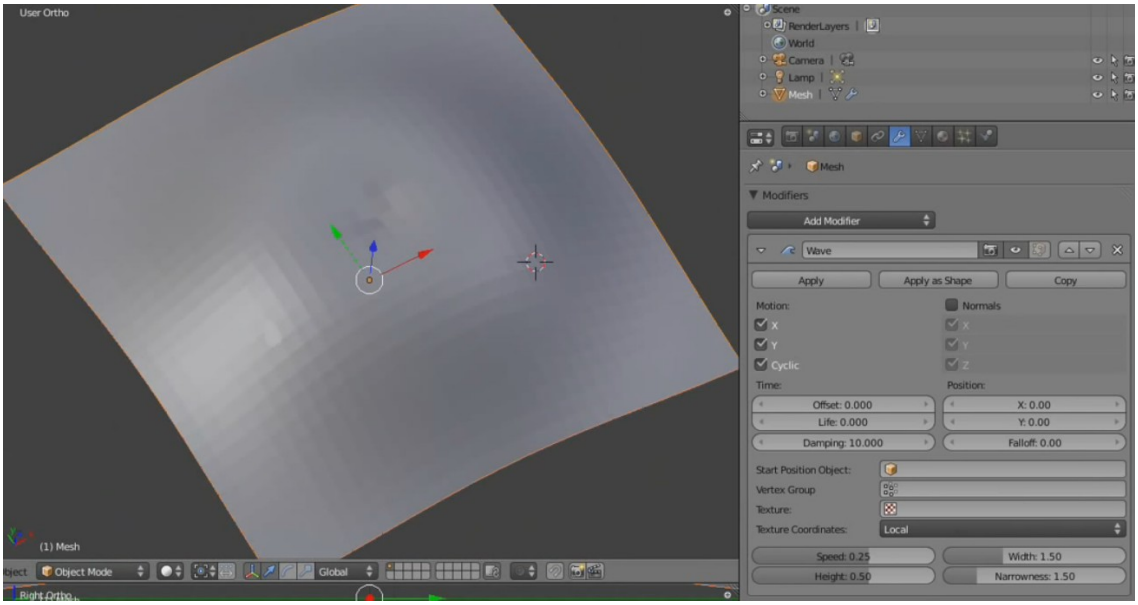
Togliamo il cubo dalla scena di default ed aggiungiamo un oggetto Plane, ingrandiamolo e suddividiamolo un bel po' di volte. Con tale oggetto attivo, apriamo la scheda *Modifiers* nel pannello *Properties*; qui clicchiamo su *Add* e aggiungiamo un primo modificatore *Wave*.



Aggiungere il modificatore Wave al Plane già suddiviso

È possibile vedere l'effetto introdotto ponendo il mouse in una finestra 3D e premendo *ALT A* (anteprima dell'animazione), tuttavia con i valori di default si ha una deformazione alquanto "violenta" della mesh, per cui torniamo alla scheda del modificatore per variare qualche parametro.

In *Motion*, selezioniamo i pulsanti *Y* e *Cyclic*, mentre lasciamo deselezionato il pulsante *Normal*, sulla destra. Impostiamo un valore negativo, come -100 o -200, per l'offset del tempo.
Abbassiamo il valore di *Speed*, portandolo ad esempio a 0.1, nonché i valori di *Height* e *Narrowness*, portandoli rispettivamente, ad esempio, a 0.2 e 0.4, mentre alziamo considerevolmente il valore di *Width*, impostando 4 o 5.



I parametri del modificatore Wave

Per aggiungere un minimo di "rumore" al movimento della mesh, aggiungiamo un secondo modificatore di tipo Wave e modifichiamo alcuni dei suoi parametri, selezionando ad esempio solo *Cyclic* in *Motion* ma attivando, questa volta, *Normal*, e selezionando solo *Z* tra i relativi pulsanti; per *Speed*, *Narrowness* e *Height*; poi, impostiamo come valore 0.1 in tutti e tre i casi, mentre per Width un valore decisamente più alto, come 4 o 5.

Anche in questo caso, impostiamo un valore negativo, come -100 o -200, per l'offset del tempo.

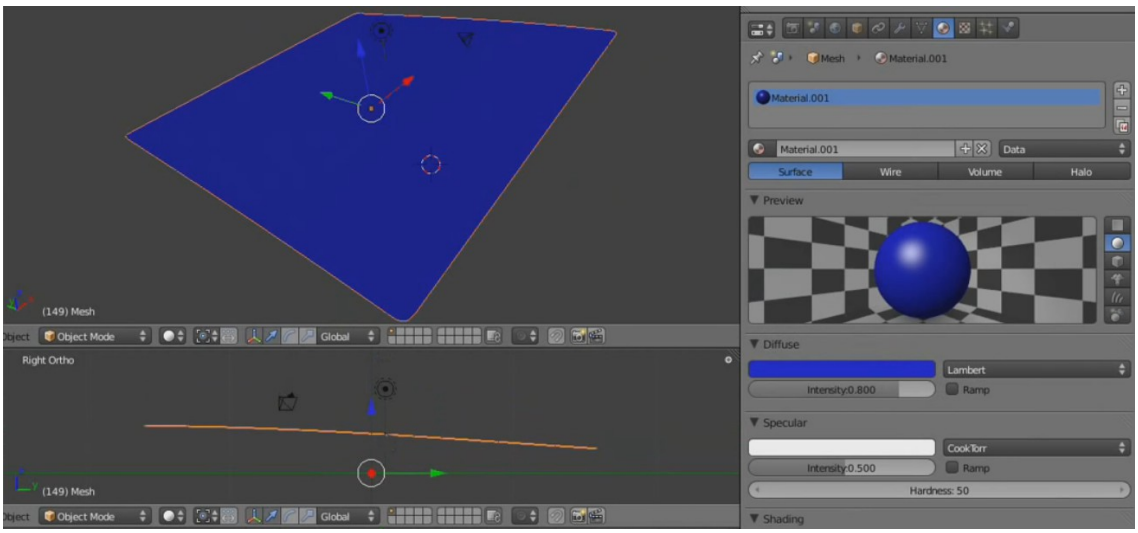
A questo punto possiamo aggiungere anche altri modificatori, ad esempio *Subsurf Smooth*, per "addolcire" un po' la mesh, se vogliamo.

Il secondo modificatore Wave

Ovviamente, i valori mostrati in questo tutorial sono solo d'esempio, non dovete attenervi strettamente ad essi, ed anzi vi consiglio di fare delle prove con valori differenti per valutarne gli effetti; lo stesso discorso vale per il numero e le impostazioni delle Textures da utilizzare, argomento che trattiamo subito.

Con l'oggetto Plane attivo, selezioniamo Material dal pannello Properties sulla destra.

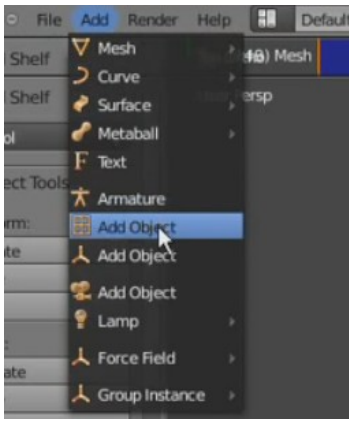
Se non presente di default, aggiungiamo un Material, ed impostiamo una tonalità di blu come colore diffuso... o altri colori, se vogliamo realizzare laghi tossici di mondi alieni!



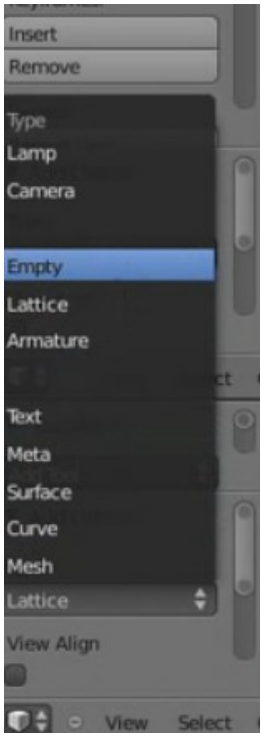
Il Material di base

Prima di aggiungere le Textures, comunque, aggiungiamo un oggetto *Empty* alla scena, cosa che va fatta selezionando, dal menù *Add* in alto nella finestra di Blender, la voce *Add Object*.

Qualora l'oggetto appena creato dovesse essere di un tipo diverso da *Empty*, come ad esempio *Camera* o *Lattice*, clicchiamo sul menù a scomparsa "*Type*" nella *Tool Shelf* per selezionare la voce "*Empty*".



Add Object - Empty



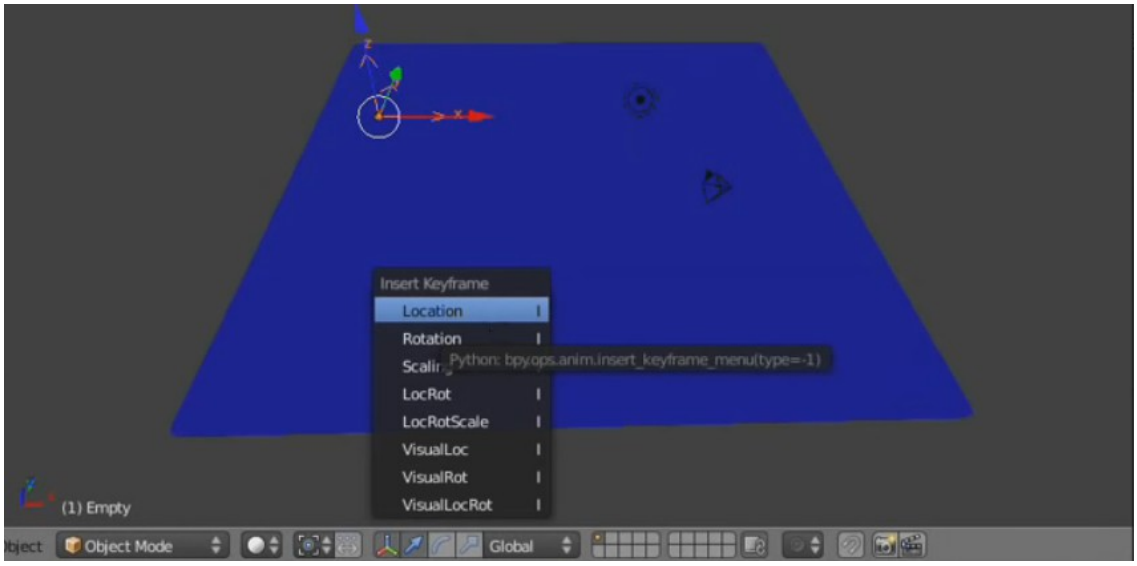
Il tipo di oggetto, nella Tool Shelf

Una *Empty* è un oggetto inconsistente, invisibile, che ci serve per spostare la Texture sulla superficie: mapperemo infatti le Textures con modalità Object alla *Empty*, e lo spirito della *Empty* aleggerà sulle acque per "muovere" le Textures ad essa collegate. Il nome di default della *Empty* è, appunto, "Empty", e visto che in questa scena non ne creeremo altre, possiamo lasciarlo così.

Con la *Empty* selezionata, premiamo ALT R per eliminarne la rotazione iniziale, poi scaliamola un po', ingrandendola ad esempio di 2-3 volte. L'effetto finale dipenderà da dimensioni e posizione della *Empty* per cui, anche in questo caso, conviene fare dei rendering di prova ogni tanto, per vedere l'effetto ottenuto con impostazioni differenti.

Ovviamente dobbiamo animare tale oggetto, spostandolo sul piano XY (ossia, non spostandolo in altezza), altrimenti la Texture resterà immobile; le animazioni della *Empty* vanno realizzate come le animazioni delle mesh, ossia premendo *I* ed inserendo le chiavi *LocRocScale* ai vari KeyframeS (nel nostro caso ne bastano solo 2, uno all'inizio e uno alla fine dell'animazione) per far effettuare l'interpolazione a Blender.

Selezioniamo nuovamente l'oggetto Plane ed accediamo alla scheda Textures nel pannello Properties. Creiamo una prima Texture di tipo *Clouds*.

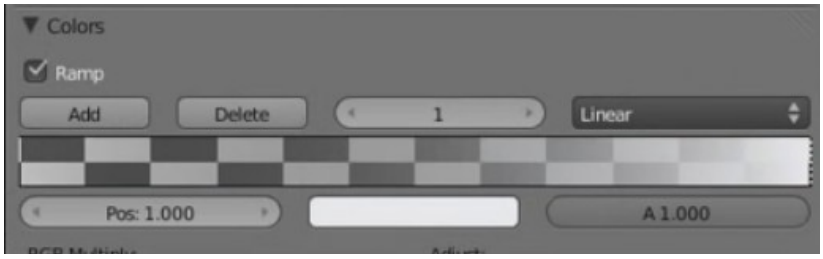


Inserimento dei Keyframes per la Empty

Per rendere la Texture meno "granulosa", impostiamo un valore basso per *Noise Size*, come ad esempio 2, mentre alziamo il valore di *Noise Depth*, portandolo ad esempio a 6 o 8, e lasciamo il tipo di rumore a *Soft*.

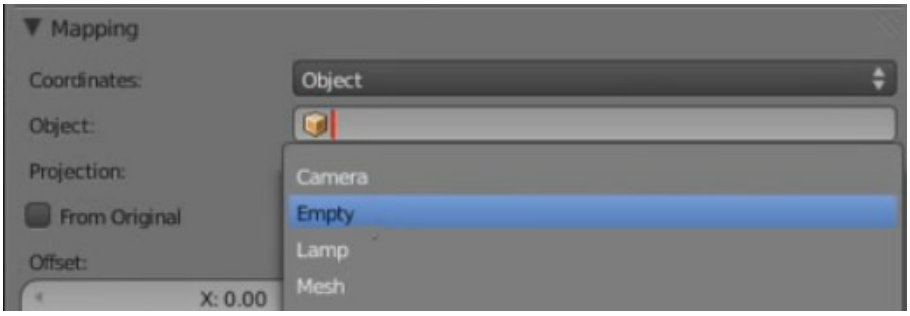
Inserimento e primi parametri della Texture Clouds

Nella scheda Colors della Texture, attiviamo la *Ramp* e cambiamo il colore a destra in bianco.



La Color Ramp (modificata) per la Texture Clouds

Nella scheda *Mapping* scegliamo come voce per *Coordinates* l'opzione *Object* dal menù a scomparsa e, nel campo testuale denominato "OB:" che apparirà sotto il menù, scriviamo il nome della nostra *Empty*, ossia (nel nostro caso) Empty, appunto.



Utilizzare la Empty per le coordinate di mappatura della Texture

Questa Texture non dovrà influenzare il colore della mesh, per cui in *Influence* deselezioniamo *Color* e selezioniamo *Normal* e *Displace*, impostando come valori 0.5 per *Normal* e 0.3 o 0.2 per *Displace*. Questa è solo una *Texture*, ma ovviamente possiamo aggiungerne altre per influenzare ad esempio il solo canale *Normal*, oppure il canale *Color* o ancora *Emit* per simulare degli spruzzi sulla superficie dell'acqua, e così via, legandole all'oggetto, ad un'unica *Empty* o a più *Empty*, per animarle in maniera differente. Niente ci vieta, infine, di aggiungere effetti *Ray* come trasparenze e riflessi o altri effetti, come il Sub Surface Scattering. Già a questo punto possiamo effettuare un rendering di prova per vedere cosa sta succedendo. Va detto che gran parte del lavoro lo fanno le luci, per cui bisognerà lavorarci un po' sopra per ottenere gli effetti desiderati, ma il "meccanismo di base", per così dire, è tutto qui, ed è davvero molto semplice.

Screenshot dell'effetto finale

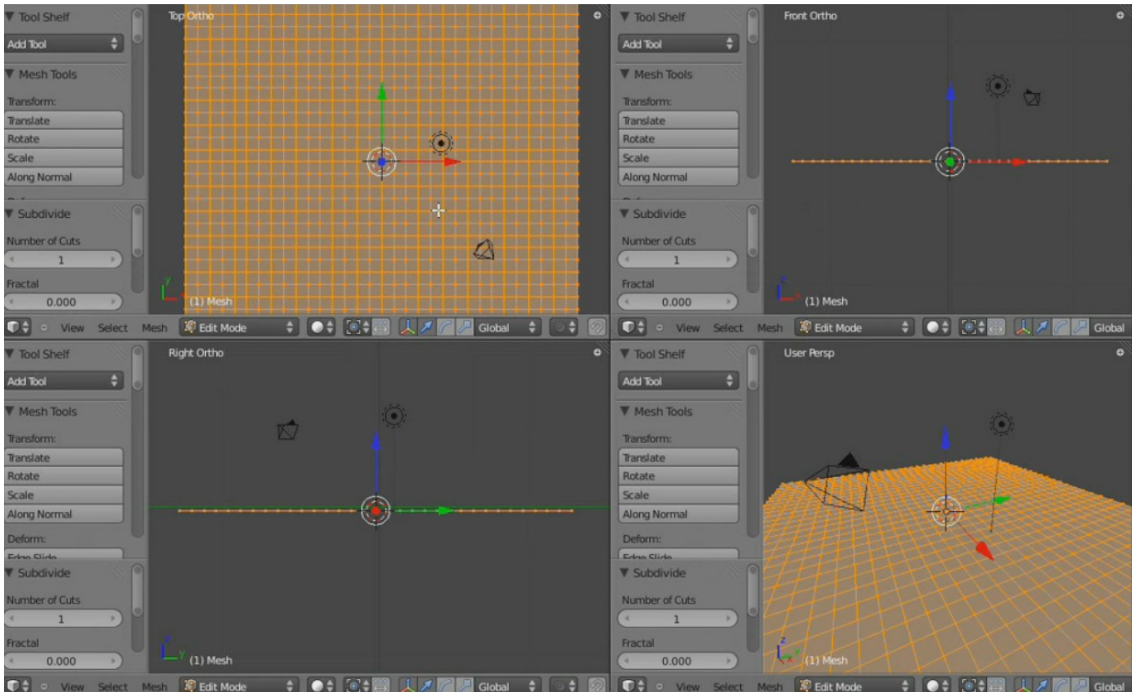
* * *

Bandiera mossa dal vento con Displace, NO CLOTH

In questo tutorial vedremo come realizzare una semplice bandiera mossa dal vento senza fare ricorso al motore fisico dei tessuti (*cloth*) di Blender 2.5.

Non effettueremo quindi una simulazione buona per tutte le stagioni, quanto una soluzione ottima per delle immagini fisse o per animazioni brevi che, tra l'altro, non impegnerà molto il processore, come farebbe invece il simulatore dei tessuti.

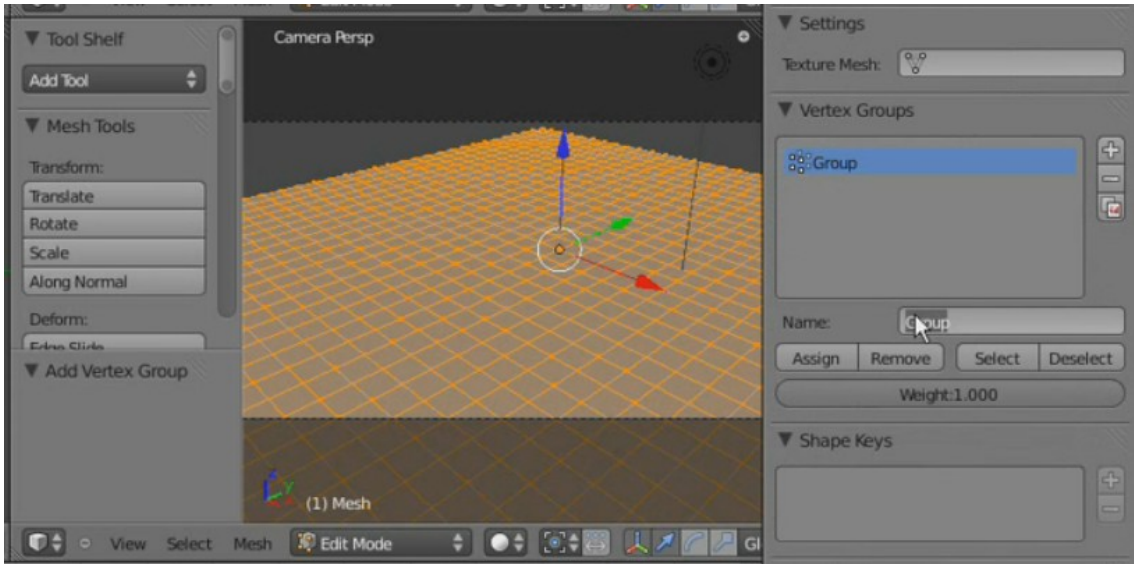
Quello di cui abbiamo bisogno è un Plane suddiviso un bel po' di volte, in modo da aumentarne la risoluzione, poi una Texture per il *Displacement* e una *Empty* per animare la bandiera.



Il Plane di partenza, suddiviso un bel po' di volte

Prima di procedere col *Displacement*, comunque, vediamo come assegnare dei "pesi" al modificatore, per simulare quello che avviene nella realtà, ossia il fatto che le porzioni di tessuto vicine al supporto della bandiera sono meno soggette all'azione del vento rispetto alle estremità, deformandosi in maniera minore o addirittura restando immobili.

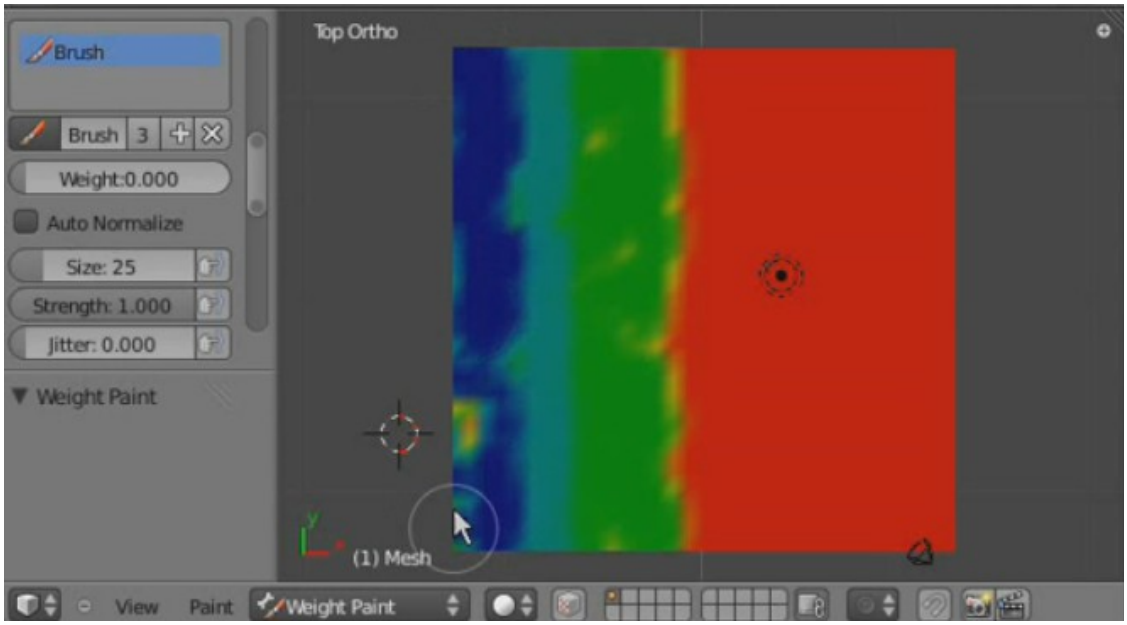
Con il Plane selezionato, passiamo in modo Edit, selezionando tutti i vertici se non sono selezionati di default, dopodiché accediamo alla scheda "Object Data" nel pannello Properties, ed aggiungiamo un Vertex Group all'oggetto. Non c'è bisogno di annotarne il nome perché, come vedremo, la nuova interfaccia di Blender ci suggerirà i nomi di questo ed altri oggetti, quando ci serviranno; possiamo rinominarlo, se vogliamo, ma per questo esempio non ne creeremo altri, per cui non c'è il rischio di fare confusione.



La creazione del Vertex Group per il Plane

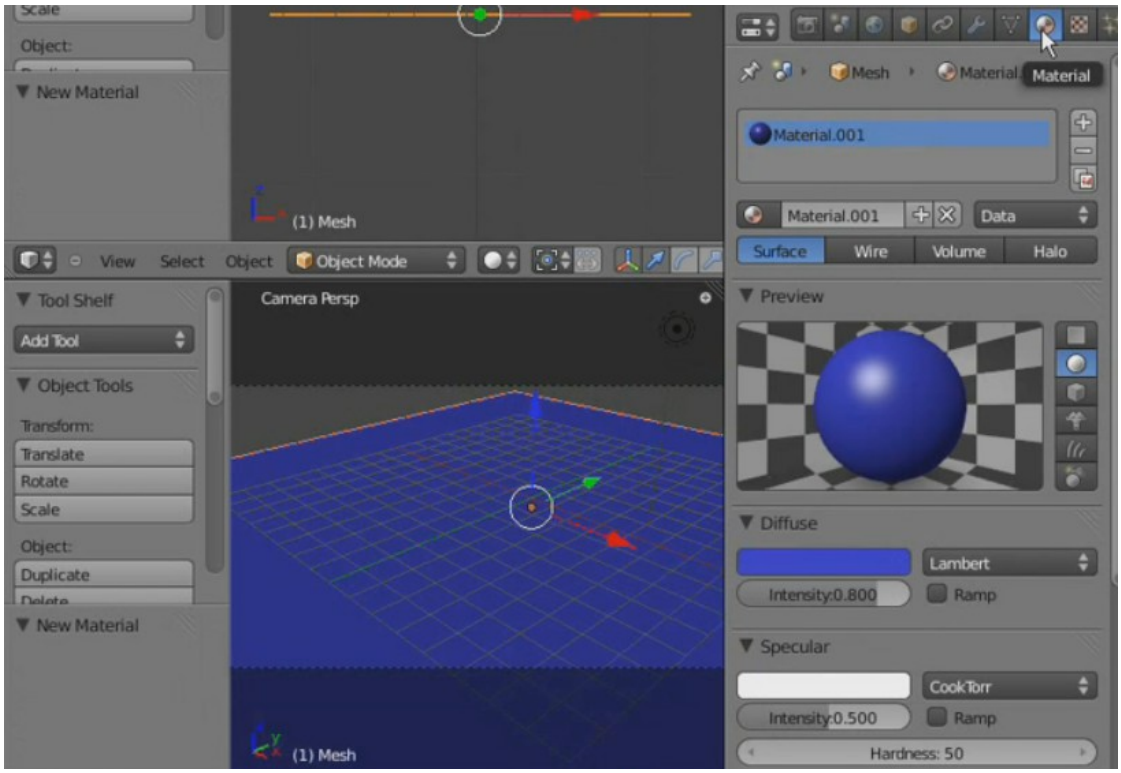
In una finestra 3D passiamo in modalità *Weight Paint* e iniziamo ad assegnare i pesi ai vertici della mesh, operazione che va fatta cliccando e trascinando il mouse, come se stessimo dipingendo l'oggetto. Sulla sinistra, troviamo gli strumenti che ci consentono di definire la dimensione del pennello, l'intensità del tratto e il peso da applicare.

Procediamo "per fasce", associando un peso nullo vicino al supporto della bandiera ed aumentando velocemente via via che andiamo verso l'esterno.



Definizione dei pesi con il Weight Paint: procedere "per fasce"

Terminato questo lavoro, torniamo in modalità oggetto ed accediamo alla scheda Material del pannello Properties, dove aggiungiamo un Material all'oggetto, se non presente di default.
Potete definire come volete l'aspetto dell'oggetto, assegnando colori, riflessi ed altri effetti; qui mi sto concentrando solo sull'effetto della bandiera mossa dal vento tralasciando questi aspetti.



Definizione di un Material di base

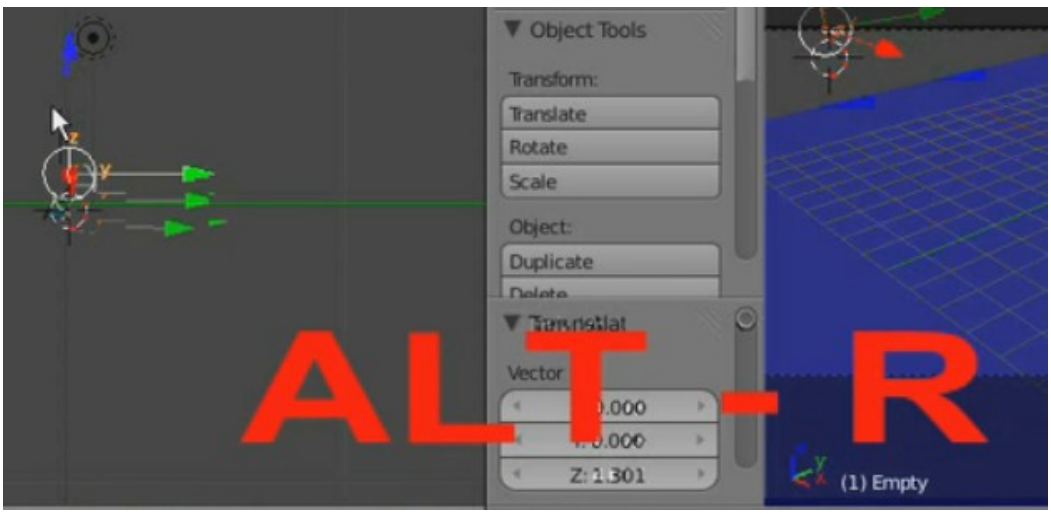
Passiamo subito alla scheda successiva del pannello Properties, ossia Texture, ed aggiungiamo una Texture per l'oggetto, ma deselezioniamola subito nella pila delle Textures (rendendola non attiva). Questa Texture non ci serve, infatti, per definire l'aspetto dell'oggetto, ma solo l'effetto.

Aggiungere una Texture disattivata al Material

Cambiamo il *tipo*, che di default dovrebbe essere *Clouds*, in *Wood*. Selezioniamo *Band Noise* anziché *Bands* e raddoppiamo i valori presenti di default per *Size* e *Turbolence*, portandoli rispettivamente a 0.5 e 10.

Adesso nella scena 3D aggiungiamo un oggetto *Empty*, cliccando su *Add* nel menù in alto e selezionando *Add Object*; qualora l'oggetto creato dovesse essere di tipo diverso da *Empty*, ad esempio *Camera* o *Lattice*, cambiamolo in *Empty* mediante il menù *Type* presente nella Tool Shelf.

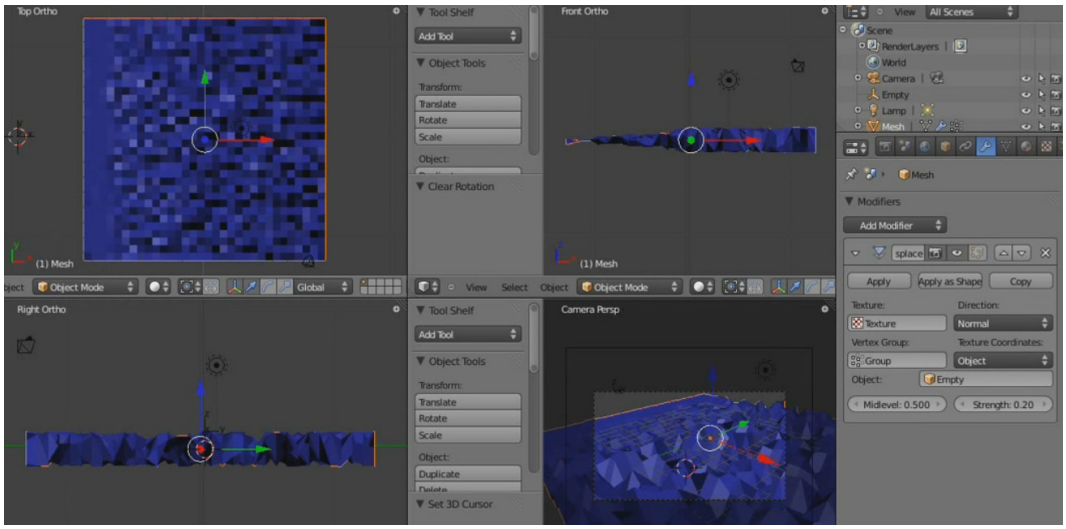
Questo oggetto *Empty* ci servirà per animare la Texture, ma prima disattiviamone la rotazione iniziale premendo *ALT R* e selezioniamo nuovamente il Plane per assemblare il tutto.



Disattivazione della rotazione iniziale della Empty, con ALT R

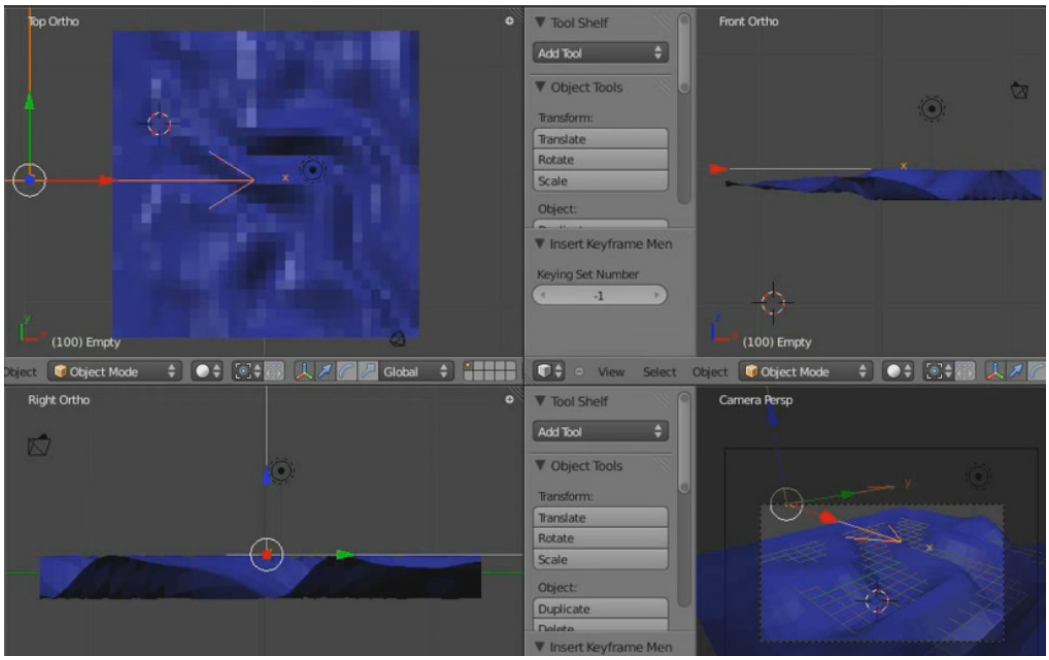
Con il Plane selezionato, quindi, accediamo alla scheda *Modifiers* del pannello Properties, ed aggiungiamo un modificatore di tipo *Displace*.

Qui impostiamo come Texture la Texture *Wood* precedentemente creata, lasciamo *Normal* per la direzione, impostiamo come *Vertex Group* quello creato all'inizio, abbassiamo l'intensità (definita da *Strength*) a 0.2 o anche 0.1, lasciamo inalterato *Midlevel* e cambiamo Texture *Coordinate* da *Map* ad *Object*, inserendo (nel campo testuale che apparirà) il nome della *Empty*.



Effetti sul Plane dopo aver impostato i vari parametri di Displace

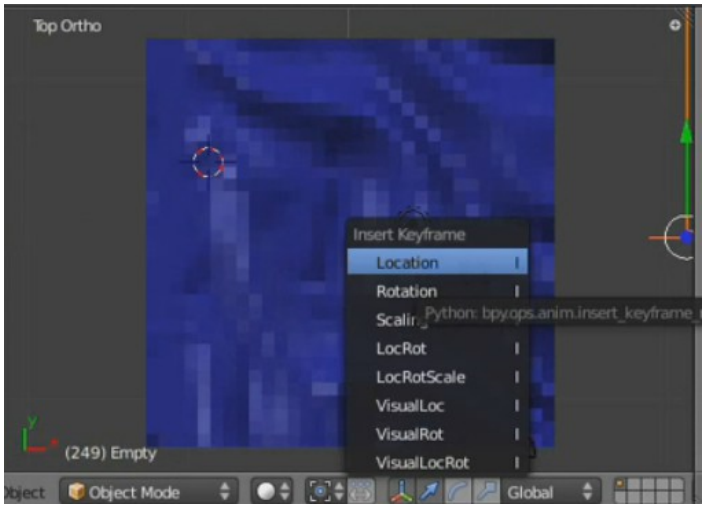
L'aspetto del *Plane* cambierà radicalmente: ciò è dovuto alla dimensione della *Empty*, che influisce su come viene applicata la Texture al modificatore, per cui scaliamo la *Empty*, ingrandendola un po', fino ad ottenere un risultato apprezzabile, visibile già nella finestra di navigazione 3D in quanto tali modifiche vengono applicate in tempo reale.



L'effetto dopo aver scalato (ingrandendola) la Empty

A questo punto, il nostro *Plane* ha assunto l'aspetto di una bandiera mossa dal vento; dobbiamo però animare tale effetto, cosa che va fatta traslando la *Empty*.

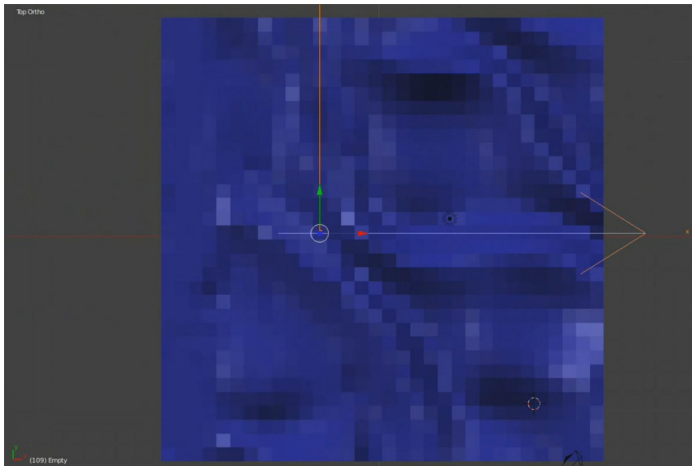
Selezioniamo la *Empty* ed inseriamo un keyframe *LOC* al frame 1; poi passiamo ad un altro frame, a seconda della durata della nostra animazione, spostiamo la *Empty* lungo un asse ed inseriamo un altro keyframe *LOC*.



Inserimento dei Keyframe di posizione (LOC) per la Empty

Come è possibile notare già con un'anteprima dell'animazione, premendo *ALT A* mentre il focus del mouse è in una finestra di navigazione 3D, adesso abbiamo un *Plane* che simula in maniera interessante una bandiera, senza far ricorso al simulatore dei tessuti di Blender.

Ovviamente possiamo aggiungere Textures con immagini per definire l'aspetto finale del tessuto, ma per quanto riguarda l'effetto del tessuto animato, questo è tutto.

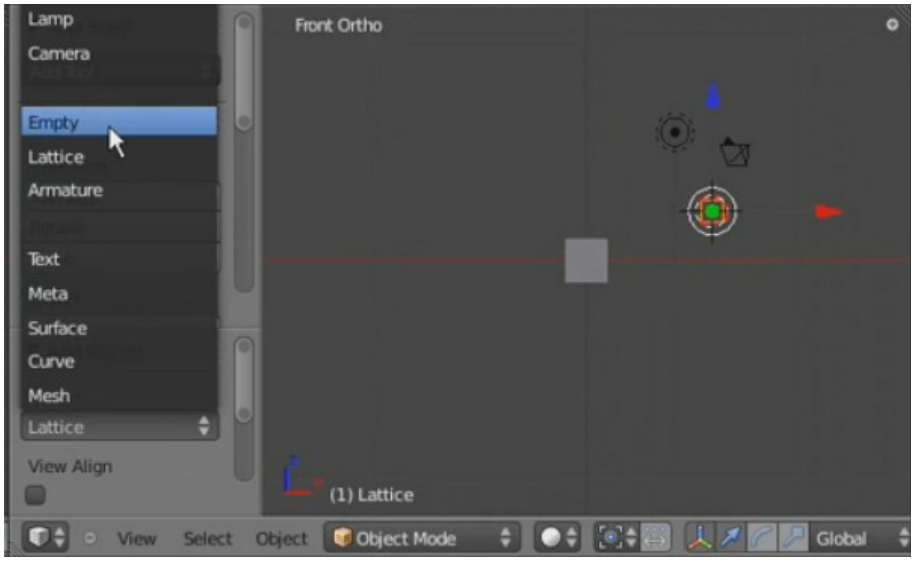

Screenshot dell'animazione finale

* * *

Fumo, fuoco e fiamme con Emitter e Smoke – 1

In questo tutorial vedremo come realizzare fuoco e fiamme con il sistema particellare *Emitter* di Blender 2.5 e come utilizzare il nuovo strumento *Smoke* per creare il fumo.

Prima di procedere, aggiungiamo alla scena un oggetto *Empty*, che ci servirà per animare le Textures delle fiamme. Posizioniamo la *Empty* al centro dell'oggetto che utilizzeremo come emettitore. Annulliamo rotazione e traslazione iniziali di questa *Empty* premendo *ALT R* e *ALT G*.



Creazione della Empty

Come oggetto per generare le fiamme possiamo utilizzare, per questo esempio, il cubo presente di default. Anticipo che avremo bisogno di almeno tre Textures e di un solo sistema particellare *Emitter*, per cui spostiamoci nella scheda *Particles* della Properties Window ed aggiungiamone uno.

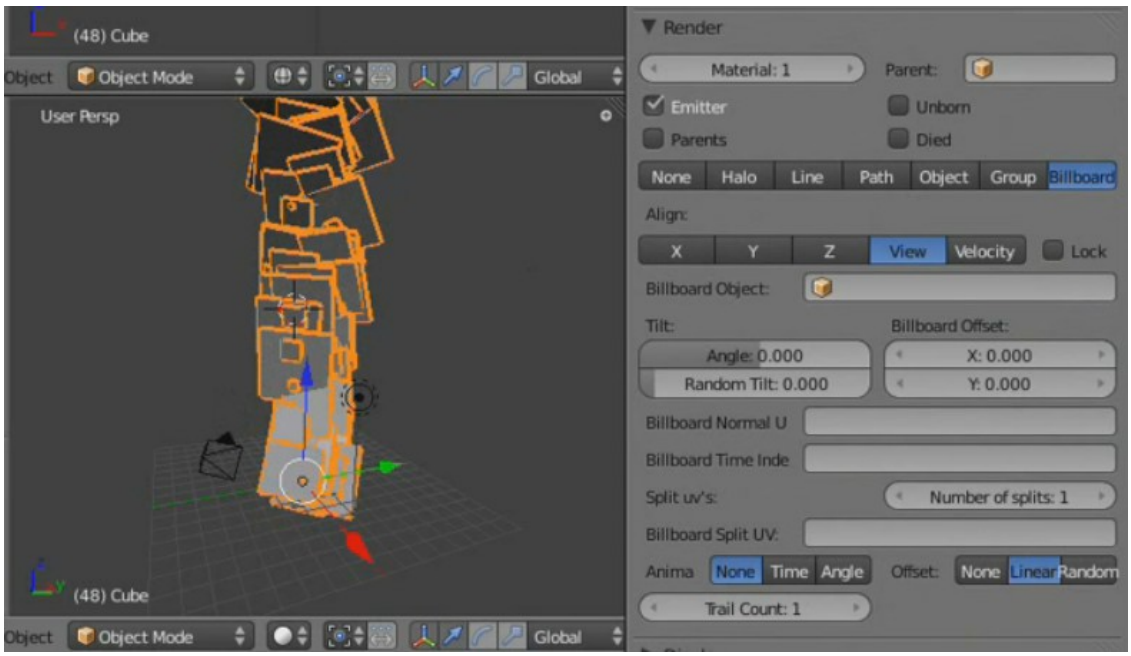
Procediamo con ordine definendo le impostazioni per il sistema particellare Emitter appena creato.

Possiamo lasciare, per questa simulazione, molti valori presenti di default, ma aumentiamo magari la durata della vita (*life*) portandola a 70 o 100, impostiamo come velocità iniziali per *Normal* e *Random* 0.5 e soprattutto poniamo il campo Z di *Emitter Object* a 25.

Creazione del sistema particellare per il cubo di default

Assicuriamoci che, nella sezione della simulazione fisica, sia selezionata l'opzione *Newton*, visto che un comportamento in accordo con la fisica newtoniana, classica, è quello che ci serve. Sempre nella scheda *Physics* del nostro emettitore, impostiamo il valore 2 per *Size* e 1 per *Random Size*.

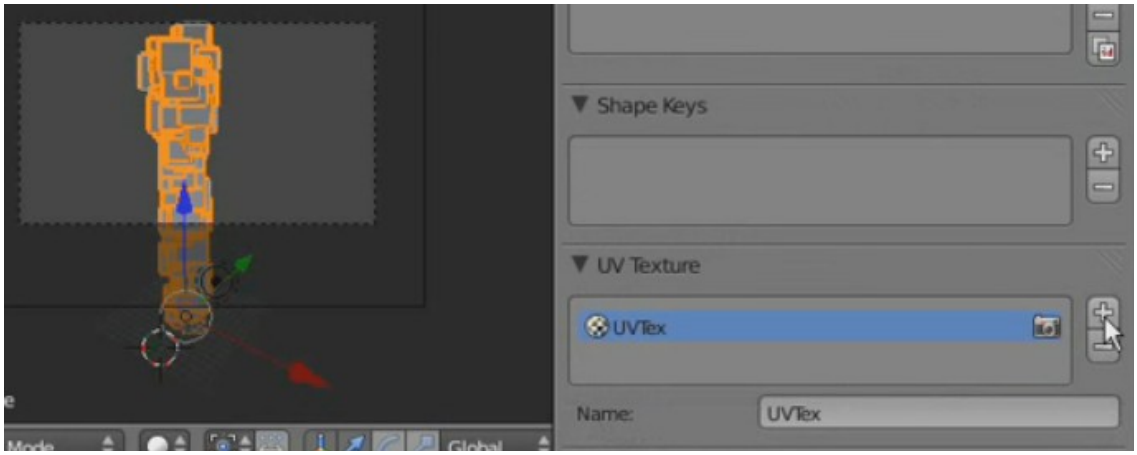
In Render, e questo è molto importante, selezioniamo *Billboard*. Questo si tradurrà nella creazione di tante facce che partiranno dall'oggetto. Un'anteprima della situazione, ottenibile premendo ALT A quando il focus del mouse è in una finestra 3D, produrrà un risultato orripilante, ma non vi preoccupate perché questa situazione è destinata a mutare grazie alle Textures.



L'anteprima delle particelle con la modalità Billboard

Aggiungo che, a seconda della durata di *Life* e della velocità su *Z* dell'Emitter Object, potete creare fuochi da campo, torce o pozzi petroliferi, sia in fiamme (con le Texture che vedremo in questo tutorial) che non (e quest'ultimo risultato è ottenibile addirittura senza Textures, solo con Materials). Allo stesso modo, variando alcuni parametri potrete ottenere fiamme colorate, più luminose, eccetera; questa è solo la procedura di base.

Per quanto riguarda le impostazioni dell'*Emitter*, per questo esempio, questo può bastare.
Prima di procedere con i materiali e le Textures di questo sistema particellare, spostiamoci in *OBJECT DATA* ed aggiungiamo una mappa *UV* di base all'oggetto.



Aggiungere una mappa UV di base all'oggetto

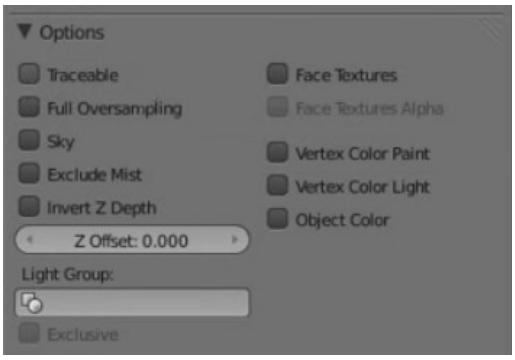
Anche questa mappa, come la *Empty*, ci servirà per le Textures delle fiamme.

Prima delle Textures ci serve, ovviamente, un Material.
Se non presente, aggiungiamo quindi un Material per l'oggetto. Lasciamo gli ombreggiatori (shader) di default per i riflessi diffuso e speculare ma in *Specular* abbassiamo l'intensità a 0 e *Hardness* a 1.

Attiviamo la *Trasparenza*, scegliendo il tipo *Z-Transp*, ponendo però *Alpha* a 0.
L'*Alpha* verrà infatti modulato da una Texture, come vedremo a breve.

Manteniamo deselezionati, come avviene di default, i pulsanti *Shadeless*, *Tangent Shading*, *Cubic Interpolation* e quelli situati in *Options*, ed anzi deselezioniamo *Traceable*, visto che questo tipo di effetti non ci serve per quello che stiamo facendo.

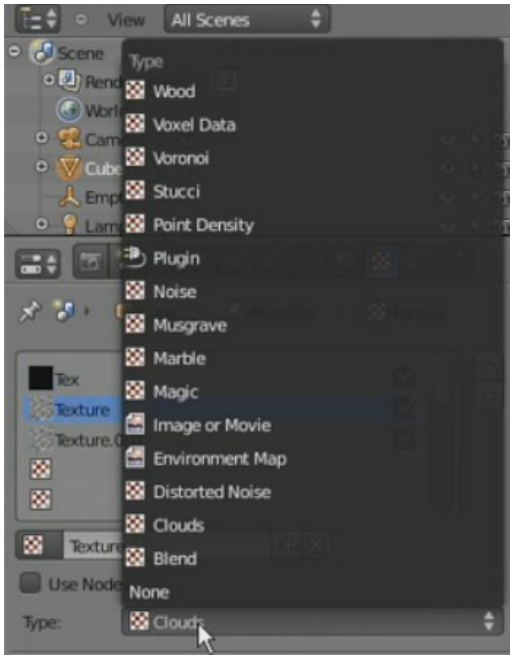
Ci servono tre Textures per il materiale dell'emettitore, per cui iniziamo ad instanziarle tutte e tre nella pila delle Texture ma iniziamo a modificare la seconda, che è una Texture "maschera stencil", ossia che conferirà un alone agli elementi della Texture successiva.



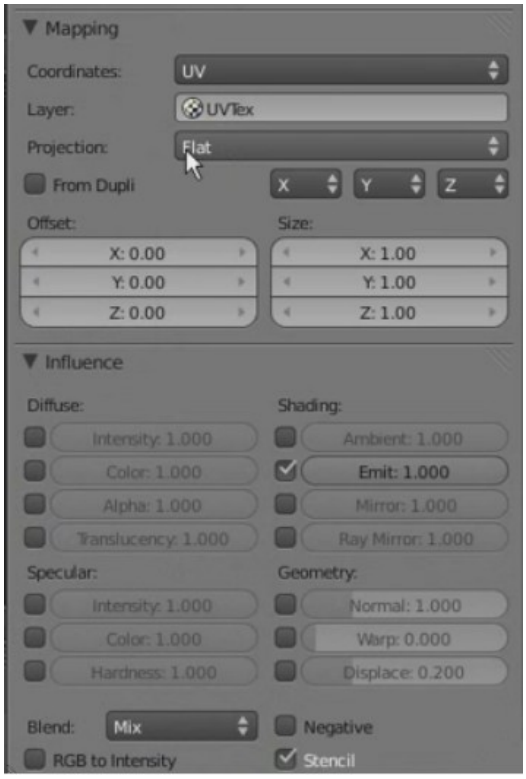
Altre impostazioni del Material

Cambiamone il tipo in *Blend Spherical*, aumentando il contrasto dell'alone bianco, cosa che va fatta aggiungendo un ulteriore passo di bianco puro con *Alpha* a 1 nella *Ramp* del colore, in posizione ad es. 0.4 o 0.3.

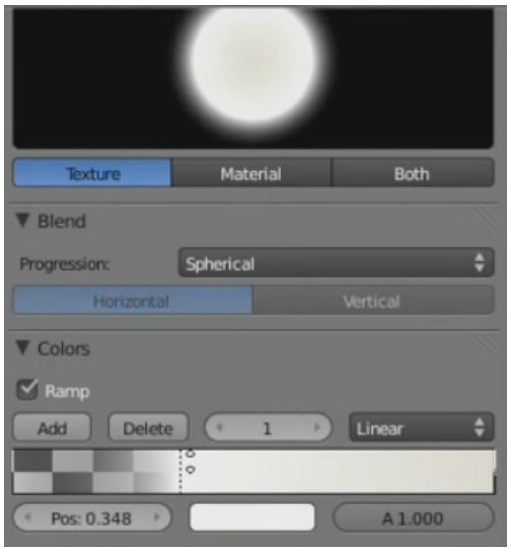
Questa Texture non influenzerà il colore dell'oggetto, per cui in *Influence* deselezioniamo *Col* e selezioniamo *Emit* (per produrre l'alone luminoso) e, più in basso, selezioniamo *Stencil*, ad indicare appunto che l'effetto verrà applicato alle Textures poste nei canali successivi.



Definire le tre Textures e iniziare ad impostare la seconda

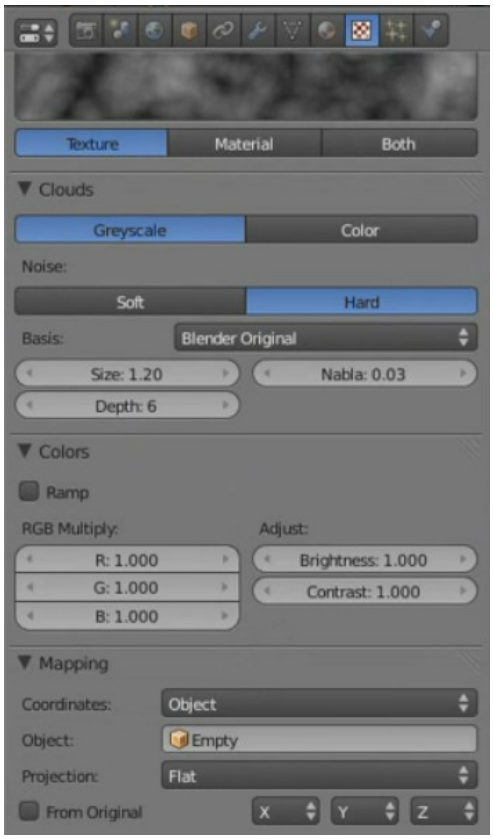


Altre impostazioni della seconda Texture



Le impostazioni Blend Spherical e della Ramp

In *Mapping*, infine, scegliamo per *Coordinates* il valore *UV* e, nel campo testuale con etichetta *Layer* che apparirà sotto, scriviamo il nome della mappa UV precedentemente creata per l'oggetto emettitore, ossia il cubo.



Alcune impostazioni della terza Texture

Passiamo alla terza Texture della pila di Textures di questo materiale.

Impostiamo, come tipo, la Texture *Clouds*, ossia le nuvole, con *Noise Size* 1.2, *Noise Depth* 6 e tipo di rumore *HARD*.

In *Influence*, manteniamo selezionato *Col* ed attiviamo anche *Alpha*. Questa Texture, quindi, conferirà il colore e la mappatura della trasparenza alle particelle.

In *Mapping* scegliamo *Object* e, nel campo di testo di etichetta *Object* che apparirà sotto, scriviamo il nome dell'oggetto *Empty* precedentemente creato. La Texture "seguirà" quindi la *Empty* che, se animata, animerà a sua volta la Texture.

Non basta, infatti, generare particelle animate per conferire un aspetto realistico alle fiamme, ma bisogna muovere anche la Texture "proiettata", per così dire, sulle stesse. In questo modo la Texture appena definita è ancora in bianco e nero: apriamo quindi *Colors* e selezioniamo la casella *Ramp*.

La *Color Band* ha di default due valori, posti alle due estremità, ma nel nostro caso ce ne servono altri due, per cui facciamo click due volte su *ADD*. Al secondo click verrà aggiunta la quarta divisione nello stesso punto della terza, ma possiamo spostare il focus da una fascia all'altra mediante il pulsante numerico con le frecce laterali posto proprio sopra la banda colorata, al centro.

Il valore contenuto in questo pulsante è l'indice della divisione attualmente in uso, con l'estremità a sinistra che ha indice 0.

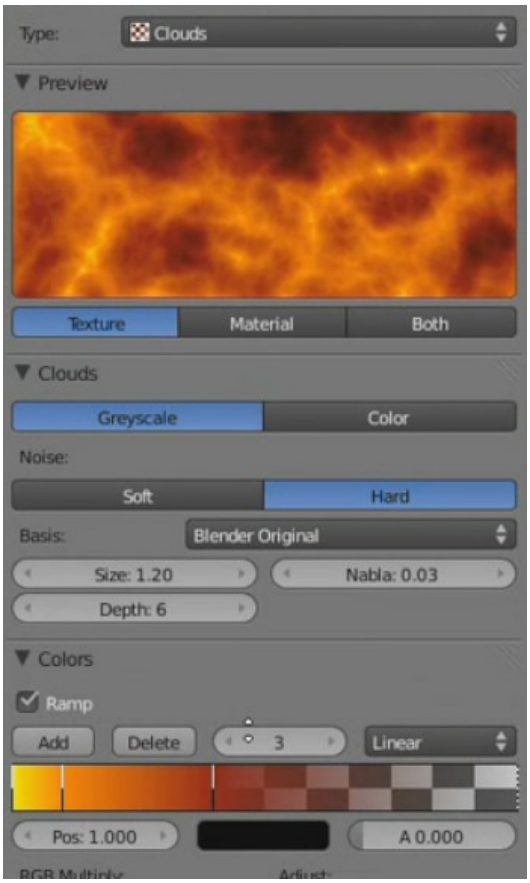
Proprio per l'indice 0, lasciamo la posizione a 0.0, cambiamo il colore in giallo e impostiamo Alpha, che di default è 0 per questa posizione, a 1.

Per l'indice 1, mettiamo come posizione 0.1, quindi vicino all'estremo sinistro, valore di rosso e di Alpha a 1, valore di verde a 0.5 e valore di blu a 0.

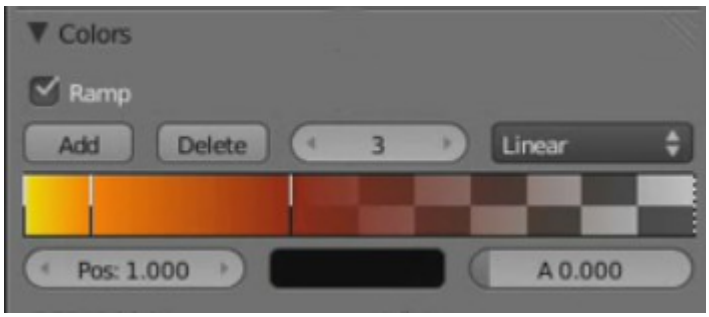
Per l'indice 2, mettiamo come posizione 0.4 o 0.5, colore rosso a 0.5 e Alpha, verde e blu a 0.

Per l'ultimo indice, l'estremo destro, mettiamo tutti i valori RGBA a 0.

Fatto questo, la terza Texture della pila è pronta. Vi ricordo che, dal momento che la seconda Texture della pila era di tipo *Stencil*, i suoi effetti verranno mappati sulla Texture appena creata, per cui le fiamme avranno un po' di luminosità extra lungo i bordi.



Sistemare la Color Band

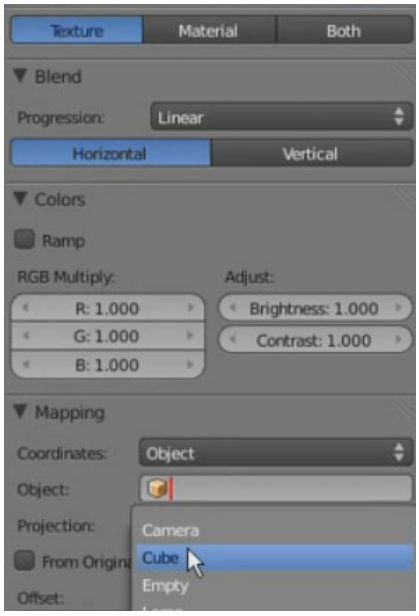


La Color Band con tutte le suddivisioni impostate

* * *

Fumo, fuoco e fiamme con Emitter e Smoke – 2

Passiamo all'ultima Texture che ci resta da definire per l'emettitore, e che poi è la prima della pila di Textures di tale oggetto. Questa Texture deve essere di tipo *Blend Linear*; in *Mapping*, scegliamo per *Coordinates* l'opzione *Object* e, quindi, il nome dell'oggetto emettitore, in questo caso il cubo, mentre nel menù X *Mapping* scegliamo l'opzione Z. In *Influence* lasciamo selezionato solo *Col*, ma abbassando un po' il valore, per limitarne l'effetto, portandolo ad esempio a 0.75.



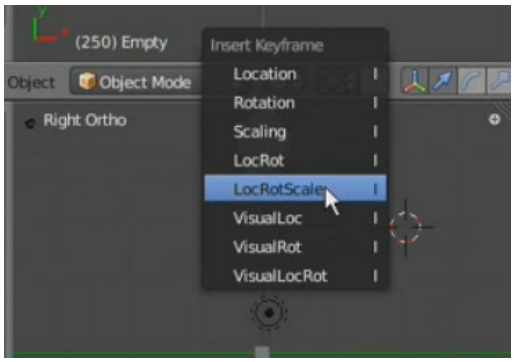
Alcune impostazioni dell'ultima Texture da sistemare (e che in realtà è la prima della lista)

Come per le altre Texture, anche qui le modifiche più importanti riguardano la *Color Band*, per cui apriamo *Colors* e attiviamo la *Ramp*. In questo caso ci serviranno addirittura tre bande interne, oltre cioè alle due poste agli estremi, per cui clicchiamo tre volte su *Add*.

La divisione all'estrema sinistra dovrà essere di un bianco puro e Alpha 1 (d'altronde, tutte le divisioni avranno Alpha a 1) ed anzi anche la seconda divisione, da porre accanto alla prima, ad esempio a 0.05 o 0.1, dovrà essere di un bianco puro, mentre la terza divisione (posta ad un quarto della *Color Band*, quindi in posizione 0.25) dovrà essere arancione; la quarta e la quinta divisione, infine, saranno entrambe rosse e poste rispettivamente a metà della banda e all'estremità destra.

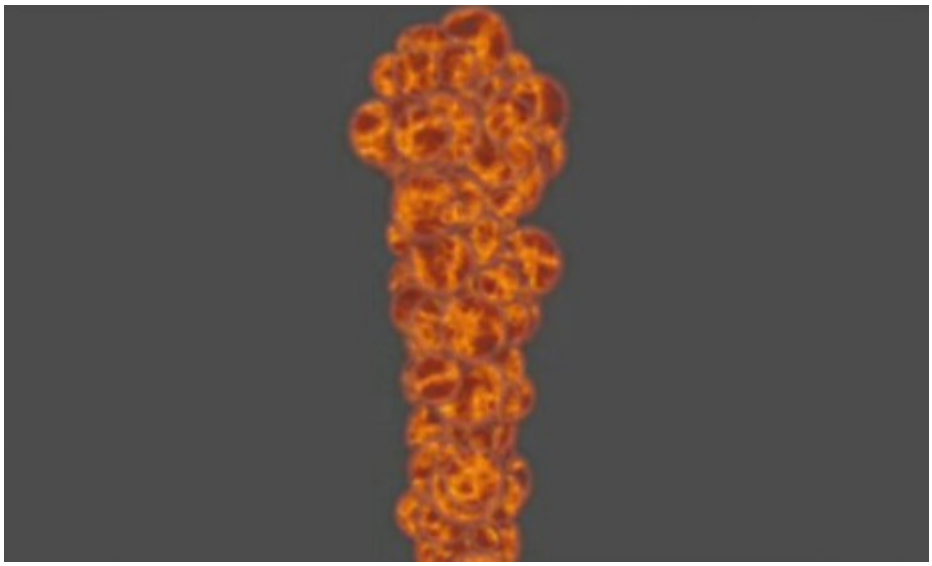
La configurazione finale della Color Ramp

Animiamo la *Empty* creata precedentemente, facendole fare uno spostamento dal basso verso l'alto nel corso dell'animazione. Se ci viene più comodo, ad esempio per creare filmati di durata maggiore o con Texture più turbolente, possiamo animare la *Empty* anche solo per un piccolo tratto e poi prolungarne la *IPO* nel Graph Editor.



Inserimento di Keyframe per animare la Empty, spostandola

L'*Emitter* e le fiamme sono pronte e possiamo finalmente lanciare un rendering di prova per vedere l'effetto ottenuto fino a questo punto.

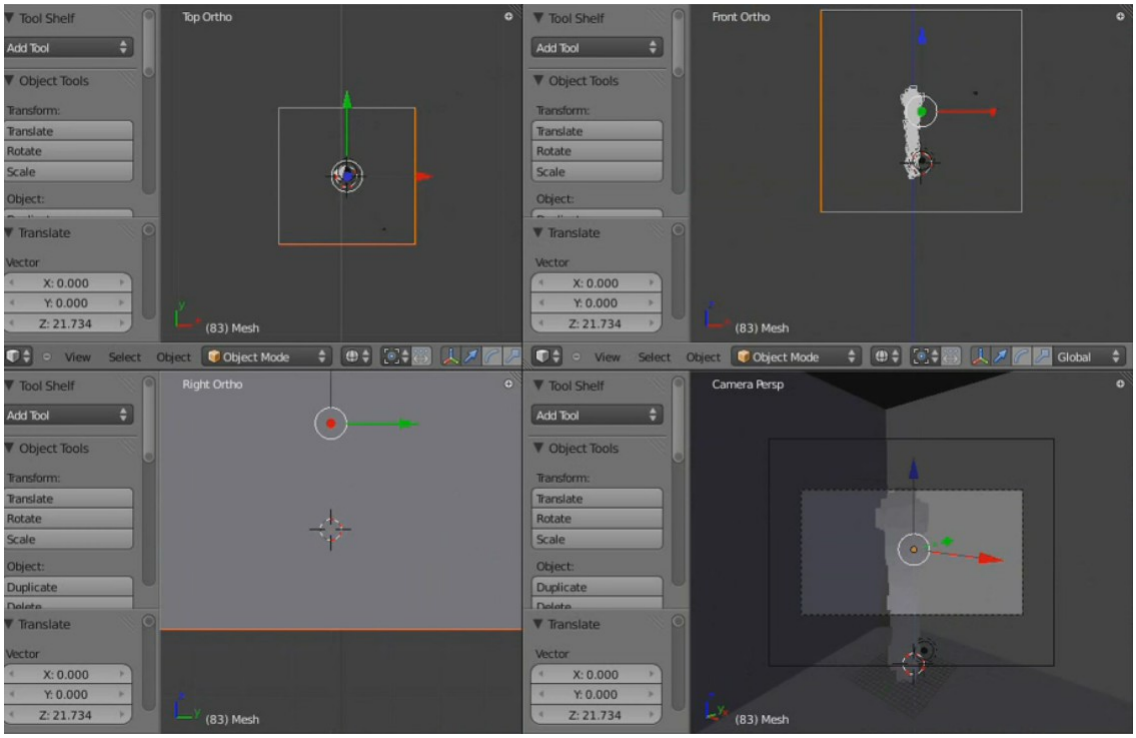


Un primo rendering del risultato ottenuto fino a questo punto

Passiamo all'effetto del fumo. Nelle versioni precedenti di Blender vi era un sistema particellare di tipo *Reactor* che produceva particelle che traevano la loro origine da altre particelle (*Reactor* non stava per "reattore", quanto per "reazione") e tale sistema poteva essere utilizzato, con un buon uso delle Texture, per il fumo emesso dalle fiamme; con la versione 2.5 di Blender, invece, è disponibile (tra gli strumenti di simulazione della fisica) lo strumento *Smoke*, ossia appunto fumo.

Il fumo va creato con lo stesso schema dei fluidi del motore dei fluidi di Blender, ossia creando un contenitore che sarà il *dominio spaziale* della nostra simulazione ed inserendovi, all'interno, gli emettitori *inflow* o i consumatori *outflow.*

Noi abbiamo già un emettitore, cioè il cubo con l'*Emitter*, per cui dobbiamo creare un dominio di applicazione; per farlo, creiamo un nuovo cubo, molto più grande del cubo emettitore e che anzi dovrà contenere quest'ultimo.



Creazione del cubo dominio

Con il nuovo cubo-dominio selezionato, quindi, passiamo all'ultima scheda della Properties Window, ossia *Physics*. Qui aggiungiamo un simulatore *Smoke* e, tra i pulsanti disponibili, scegliamo *Domain*, visto che questo cubo sarà proprio il dominio della simulazione.

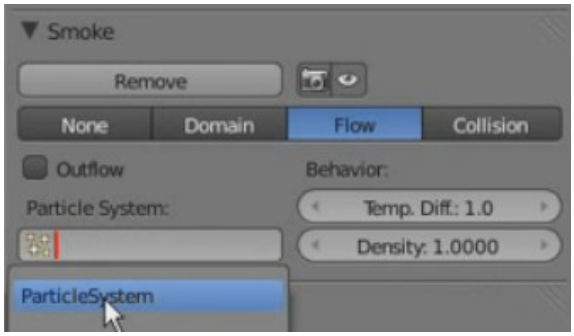
Selezionando *Domain* appariranno nuove schede e una miriade di parametri da impostare, parametri legati più che altro alla simulazione fisica del fumo, ma niente paura: per quanto ci riguarda, le impostazioni presenti di default andranno più che bene... magari aumentiamo solo il numero di *divisioni* in *Resolution* portandolo ad esempio a 50 ed attiviamo *Dissolve*.

Le impostazioni Physics Smoke per il cubo-dominio

Torniamo al cubo emettitore e passiamo anche in questo caso all'ultimo pannello della Properties Window, ossia *Physics*, ed aggiungiamo un simulatore *Smoke*. In questo caso, selezioneremo la voce *Flow.*

Questa volta possiamo impostare solo un paio di parametri di natura "fisica" (ma le impostazioni di default ci vanno bene) mentre possiamo scegliere se rendere l'oggetto emettitore (come è di default, e come lo lasceremo) oppure "consumatore" con *outflow.*

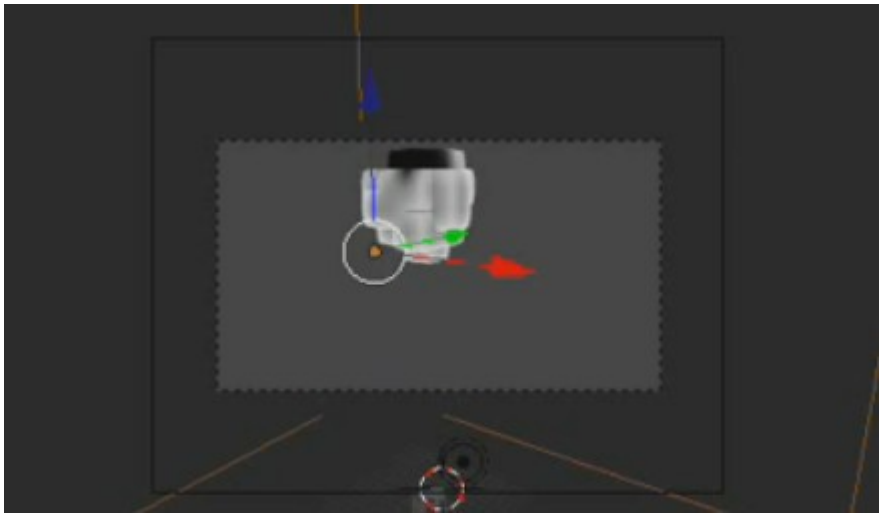
Il fumo ha origine da un sistema particellare, per cui nel campo testuale *Particle System* dobbiamo scrivere il nome del sistema particellare che simula le fiamme che danno origine al fumo.



Impostazioni Physics Smoke Inflow per l'emettitore

A questo punto premendo *ALT A* quando il focus è in una finestra di navigazione 3D dovremmo poter vedere un'anteprima dei "nuvoloni", rappresentati mediante cubi grigi che chiaramente non hanno niente a che fare con l'effetto finale ma che servono a darci un'idea di quello che vedremo.

Non è obbligatorio effettuare l'anteprima animata per vedere le nuvole: basterà portare a inizio e fine corsa la *Timeline* un paio di volte per vedere l'anteprima nelle finestre 3D.

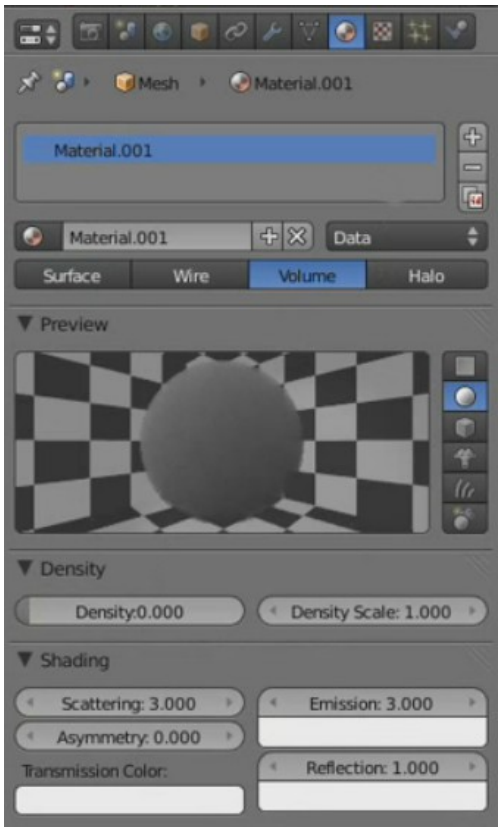

Anteprima dell'effetto in una 3D View

A valori di *Divisions* maggiori per il *Domain* si avrà una qualità migliore nella resa, ma ovviamente ciò peserà moltissimo in termini di tempo di elaborazione, proporzionalmente anche al numero di frames da elaborare. A seconda dei casi, il programma può apparire come bloccato, ma basta aprire la finestra della console per monitorare l'avanzamento del lavoro. Possiamo comunque interrompere l'animazione in qualunque momento premendo nuovamente *ALT A*.

Ad ogni modo, lanciando un rendering *non* vedremo per niente il fumo!
Ciò è dovuto al fatto che bisogna impostare un Material per il rendering dei volumi e un particolare tipo di Texture per il dominio della simulazione.

Selezioniamo quindi il cubo-dominio ed apriamo la scheda Material nel pannello Properties.

Se non presente, aggiungiamo un Material, ma cambiamone il tipo da *Surface* a *Volume*, mediante i pulsanti posti sopra l'anteprima. Portiamo a zero il valore di *Density* e definiamo come preferiamo i colori per *Emission*, *Reflection* e *Trasmission Color*; magari aumentiamo i valori di *Scattering* (visto che vogliamo simulare proprio del fumo) e di *Emission*, portandoli ad esempio a 3.



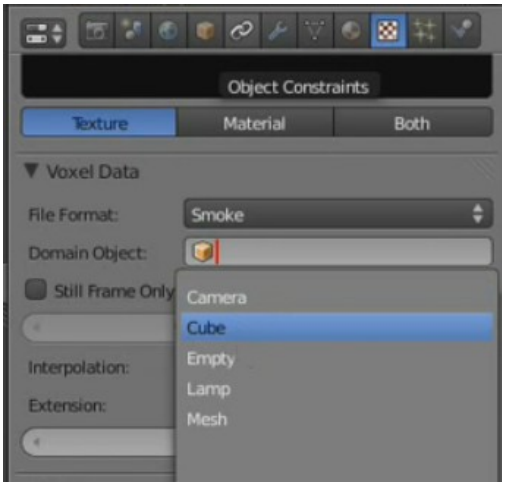
Impostazioni del Material del cubo-dominio

Aggiungiamo ora una Texture per questo Material, scegliendo come tipo *Voxel Data*.

Questo tipo di Texture recupera le informazioni per creare Texture 3D direttamente da oggetti della scena, e da qui deriva infatti il nome *VOXEL DATA*.

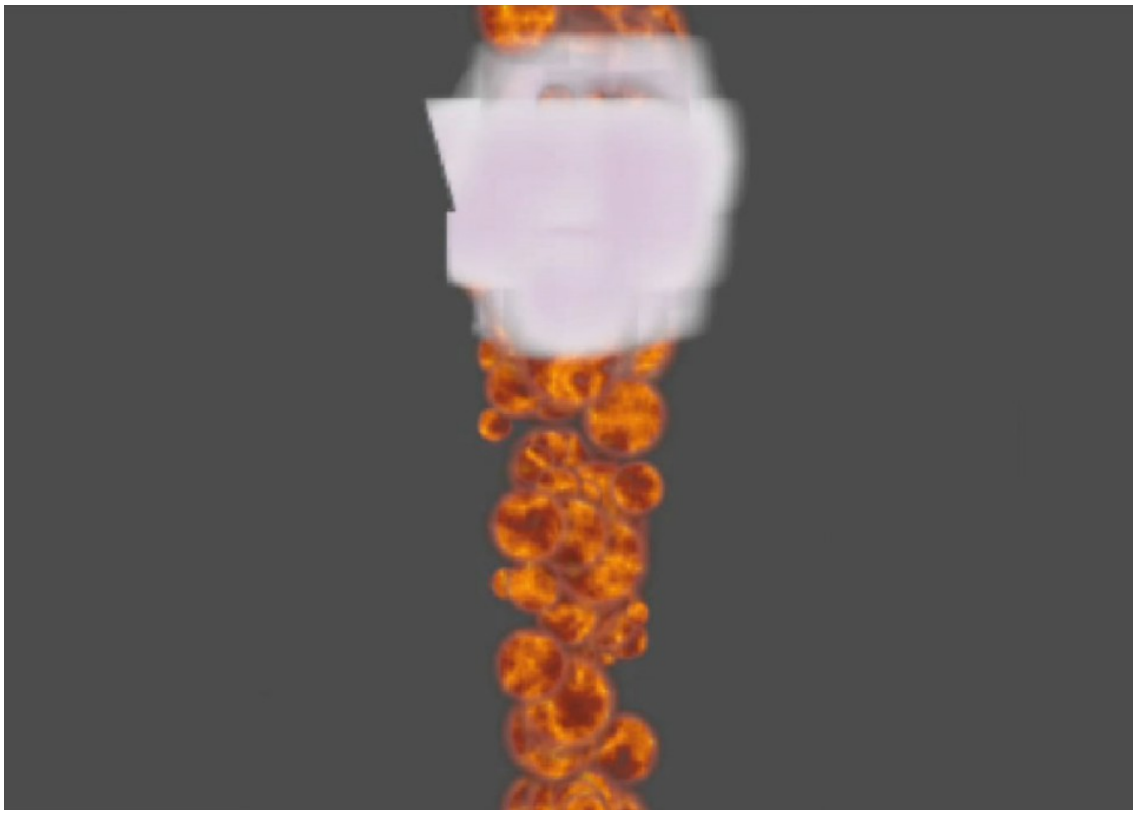
In *File Format*, per questa Texture, dovrebbe esserci selezionata di default la voce *Smoke*, che ci va benissimo. Nel campo testuale *Domain Object*, invece, scriviamo il nome dell'oggetto dominio della simulazione (nel nostro caso, il cubo più grande) dal quale la Texture ricaverà i dati sul volume.

Ancora più in basso, sempre nella scheda della Texture *Voxel Data*, selezioniamo in *Influence* la casella *Density* (aumentandone magari il valore) per cui questa Texture si occuperà di mappare la densità del Material. Verrà attivata automaticamente la casella *Emission Color*: lasciamola selezionata. Anche per questa casella possiamo variare il valore, ed inoltre possiamo attivare anche altre caselle, se vogliamo, a seconda degli aspetti che vogliamo mappare mediante i *Voxel Data*.



Impostazioni della Texture Voxel Data per il cubo-dominio

A questo punto, tutto è pronto e possiamo effettuare un rendering. In realtà, proprio con queste impostazioni, l'effetto sarà deludente, e questo perché i volumi creati dalle particelle sono, come abbiamo visto, simili a dei piani, con visualizzazione *Billboard*, cioè tutti orientati sempre verso l'osservatore.

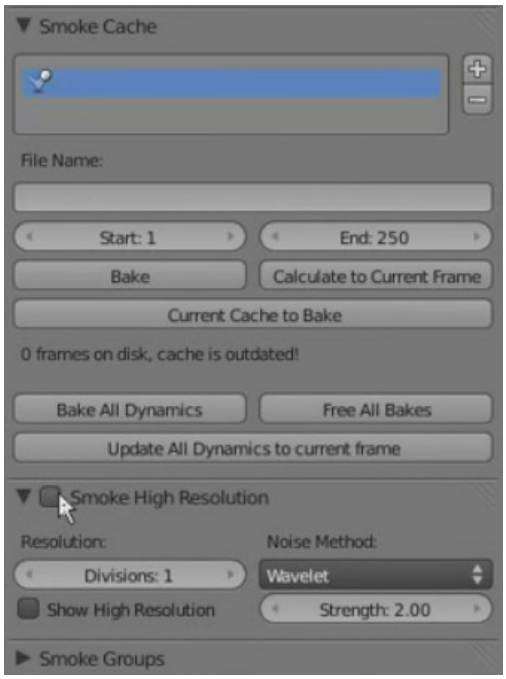


Rendering dell'effetto ottenuto

La Texture *Voxel Data* si basa sui volumi, in questo caso dei piani, e ciò spiega il perché di un risultato inferiore alle aspettative (infatti i piani verranno riempiti in maniera volumetrica e vedremo "quadrati di fumo") ma queste sono solo le indicazioni di partenza per realizzare sistemi particellari con fumo, fuoco e fiamme: basterà variare qualche parametro per ottenere effetti radicalmente diversi, bisogna solo fare diverse prove.

Una soluzione consiste, ad esempio, nel creare un secondo sistema particellare con impostazioni di resa diverse da quelle del primo ed utilizzarne le caratteristiche volumetriche per realizzare il fumo. In questo caso, infine, l'oggetto che dà origine alla simulazione di fiamme e fumo è immobile nella scena, ma per creare il fumo per oggetti in movimento bisognerà fare il *Baking* della simulazione.

Il pulsante di *BAKE* si trova nella scheda *Smoke Cache* per il fumo normale e nella scheda *Smoke High Resolution Cache* per il fumo *High Resolution*, se attivato; entrambe le schede si trovano, ovviamente, nel pannello *Physics* dell'oggetto che farà da dominio spaziale per la simulazione, ossia (nel nostro caso) il cubo più grande.



Strumenti per il BAKING nella scheda Physics Smoke Domain

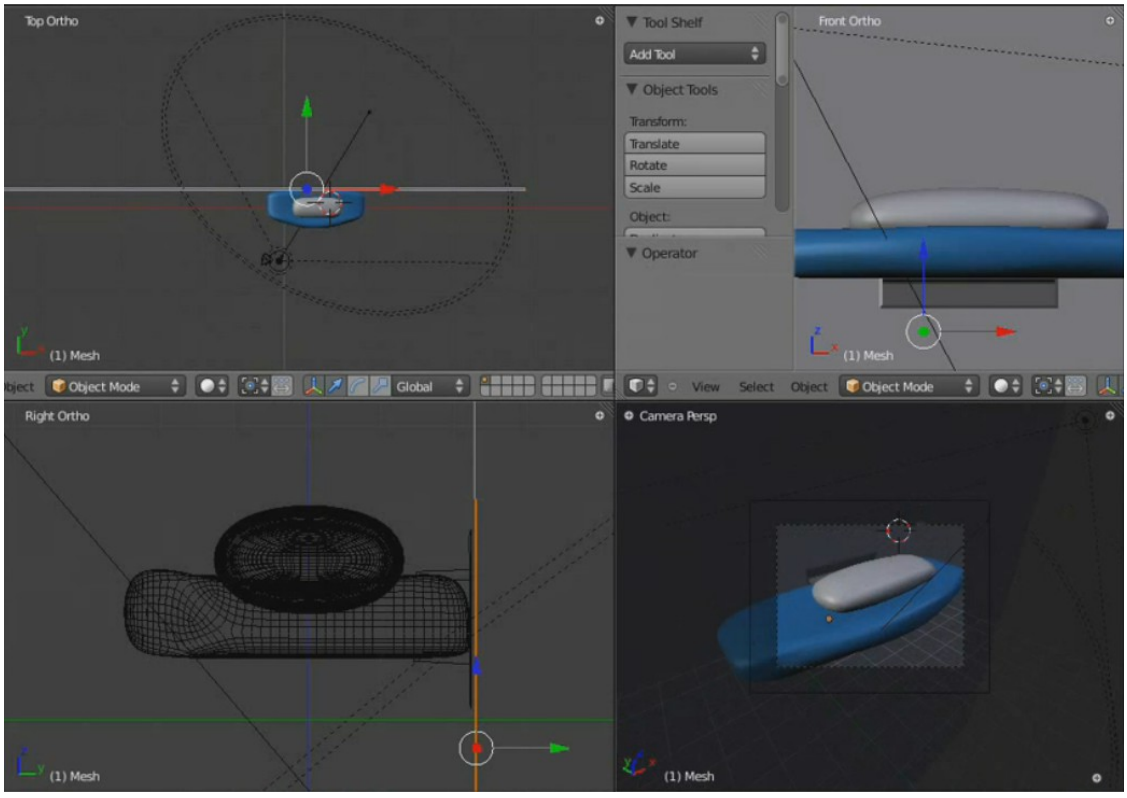
Per effettuare il *BAKE* della simulazione, in entrambi i casi è sufficiente impostare il frame di partenza e il frame finale dell'animazione e cliccare sul pulsante *BAKE*. L'operazione può richiedere anche molto tempo (da minuti a ore), a seconda della complessità della simulazione, della sua durata, della presenza di collisioni e anche in base alle prestazioni hardware del computer.

* * *

Descrizione scena con effetto SSS per sapone

Questo è un tutorial non didattico dove viene mostrato un possibile utilizzo dell'effetto SSS (Sub Surface Scattering) di Blender 3D applicato, in questo caso, ad una saponetta, descrivendo nel mentre altri materiali e impostazioni.

La scena che vedete nell'immagine seguente è composta da un semplice supporto per la saponetta, più la saponetta stessa, un pezzo di parete e una fonte luminosa.



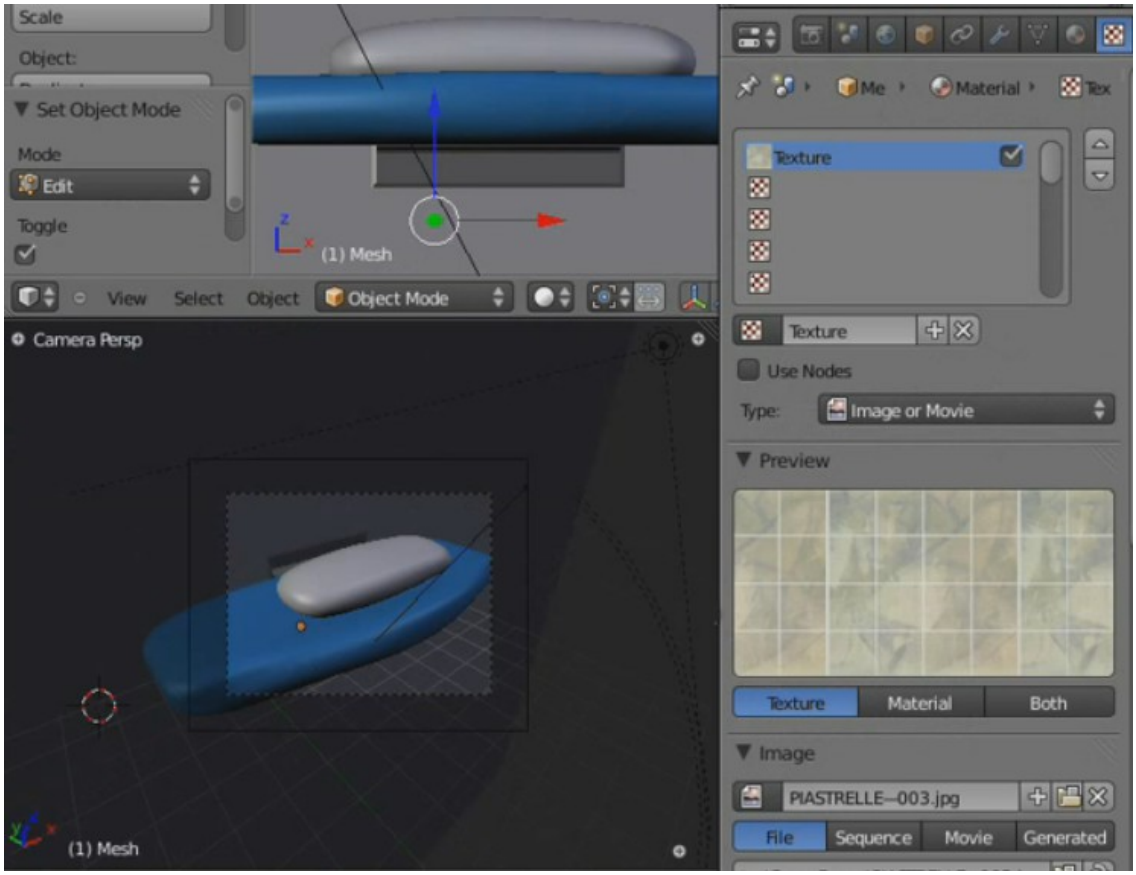
La scena di partenza

Il supporto è modellato in maniera davvero molto semplice: basta infatti suddividere un paio di volte una mesh ed applicare un modificatore *SubSurf (Subdivision Surface)* con 2 o 3 iterazioni; in

effetti, questo è lo stesso procedimento utilizzato per il pezzo di sapone utilizzato in questa scena: come vedremo, la differenza nella resa sta, più che altro, nei materiali.

Un altro Plane, modellato un po', serve da placca per agganciare il supporto alla parete.
La modellazione degli elementi è, quindi, molto banale; passiamo dunque a materiali e Textures.

Al Plane che rappresenta la porzione di parete inquadrata è applicata una Texture immagine raffigurante delle piastrelle. Tale Texture influenzerà sia il colore della mesh che il canale *Normal*, con intensità per tale canale 0.2.



La Texture immagine per la porzione di parete

Il supporto è dotato di un Material con colore diffuso azzurro, colore speculare bianco e ombreggiatori di default, con trasparenza di tipo *Raytrace* attivata.
Nel pannello della trasparenza *Raytrace*, impostiamo come valore di Alpha 0.1, indice di rifrazione IOR 1.05, Fresnel 1 ed entità del Glossiness 1.

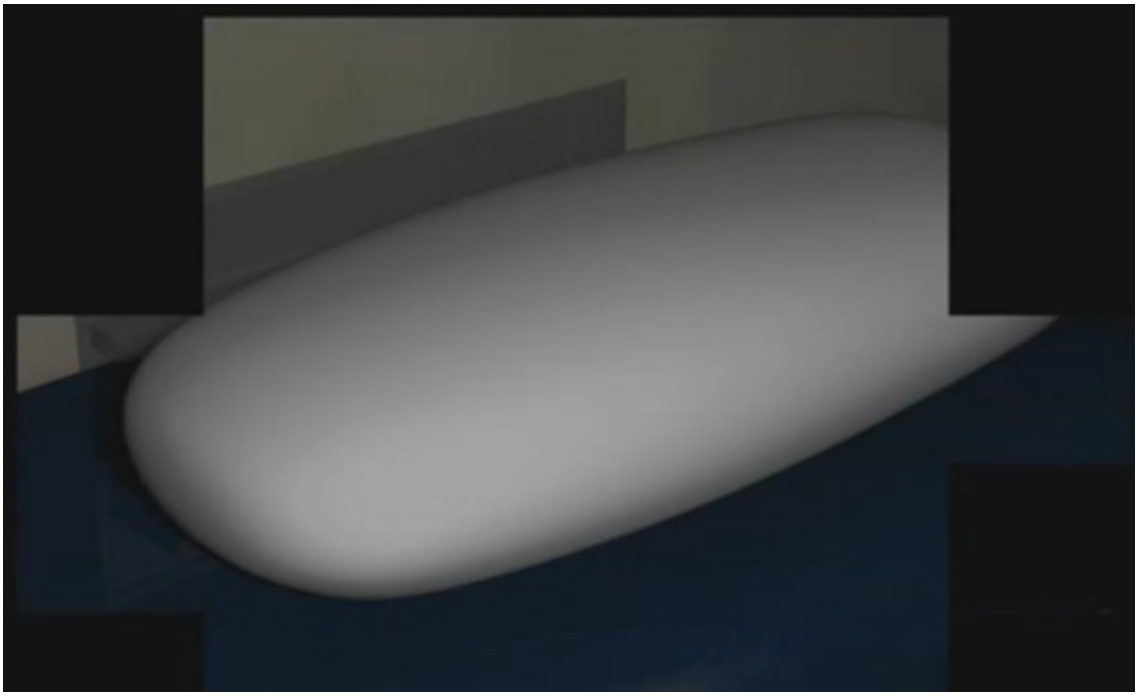
Il Material della placca che serve per agganciare il supporto alla parete avrà invece ombreggiatore diffuso *Oren-Nayar* con colore grigio scuro e ombreggiatore speculare *WardIso* con valore di intensità 1. In basso, nella scheda di questo materiale, andrà poi attivato *Mirror*, con intensità del riflesso 0.05.

La fonte luminosa dovrà essere di tipo *Spot*, in modo da poter creare delle ombre con *Buffer Shadow*, dimensioni del Buffer 4992, valore di Soft 8, Samples 5 e Shadow Sample Buffer 4.

La modellazione della saponetta è, come detto, molto semplice: una semplice mesh suddivisa e scalata un paio di volte alla quale andrà applicato, poi, il modificatore *SubSurf* con 3 iterazioni.

Nella sezione degli ombreggiatori del Material ci basterà dedicarci al colore diffuso, un bianco un po' sporcato oppure un altro colore per creare una saponetta colorata, impostando inoltre come ombreggiatore speculare *Toon* con Smooth 0.6.

Il punto centrale del materiale del sapone è, però, il *Sub Surface Scattering*.
Lanciando un rendering di prova senza effetto SSS, il materiale del sapone non risulterà per niente realistico. Potremmo cercare di applicare Texture, modellare la mesh su più strati o provare altre tecniche, tuttavia il senso di diffusione della luce di certi materiali, come il marmo, il latte, la pelle o, appunto, il sapone, è proprio il campo di applicazione del *Sub Surface Scattering*.



Il rendering (parziale) del sapone senza SSS

Nella scheda del materiale, quindi, attiviamo l'opzione Sub Surface Scattering e scegliamo, dal menù con i materiali pre-impostati, *Whole Milk*.

Di notevole importanza è il valore per *Scale* nel pannello; tale valore dipende dalle dimensioni dell'oggetto reale, ed in particolare impostando scale a 1, una unità della scena di Blender equivarrà ad un millimetro, mentre impostando 0.001, una unità della scena di Blender equivarrà ad un metro, per cui tale valore va regolato in base alle dimensioni dell'oggetto nella scena.

Le impostazioni di SSS per la mesh del sapone

Lanciando un rendering di prova, noteremo che serve più tempo per renderizzare l'immagine; serve, infatti, una "seconda passata" per applicare l'effetto del Sub Surface Scattering ai materiali che ne fanno uso (in questo caso, solo la saponetta) ma la differenza tra la nuova immagine e quella precedentemente ottenuta è evidente.

Il rendering con effetto SSS

Questa è solo una delle possibili applicazioni dell'effetto SSS di Blender, utilissimo per renderizzare materiali organici o con alta dispersione dei raggi luminosi.

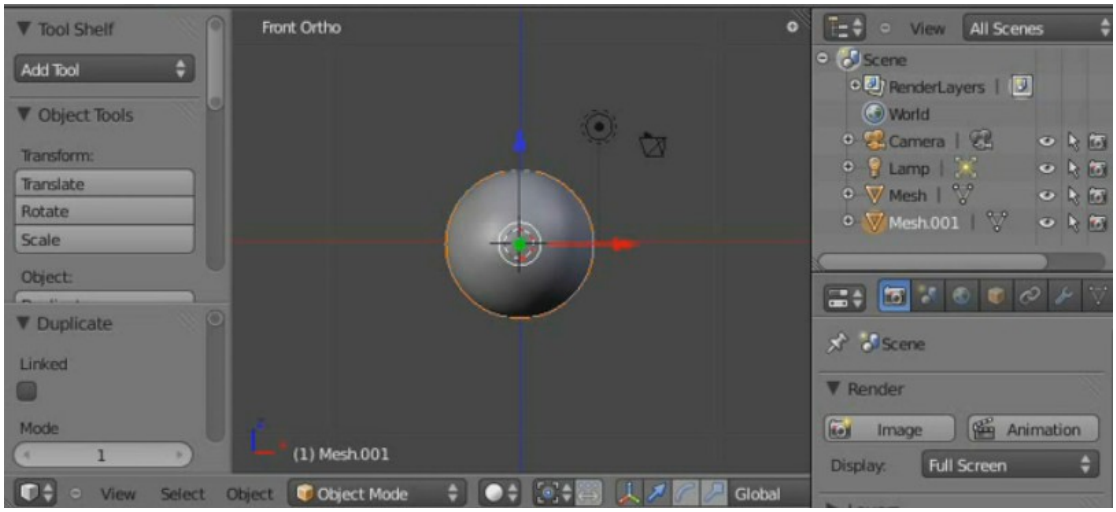
Altre opzioni interessanti per il sapone sono, ad esempio, *Skim Milk* e *Cream*.
È comunque possibile impostare manualmente i valori dei parametri di SSS per creare nuovi effetti. Blender mette a disposizione dei valori predefiniti, ma non siamo obbligati ad utilizzarli *solo* per gli scopi prefissati, anzi è possibile ricorrere a tali preimpostazioni per materiali simili a quelli indicati.

* * *

Occhio in stile cartoon

In questo tutorial vedremo come realizzare un occhio (nel senso del solo bulbo oculare) stile cartoon, in Blender 2.5. La maggior parte del lavoro riguarda le Textures, in quanto il bulbo oculare in sé è fatto da due sfere, una per i riflessi e l'altra per le Textures.

Creiamo quindi una sfera UV, applicando Smooth (il cui pulsante si trova nella Tool Shelf di ogni finestra di navigazione 3D, sotto l'etichetta Shading) e duplichiamola, premendo SHIFT D in Object Mode, per creare un secondo oggetto distinto (che potremo comunque legare al primo con un legame di parentela, per muoverli in maniera uniforme).



Le due sfere UV, con Shading Smooth

Ridimensioniamo leggermente la copia, portandola a circa il 99% delle dimensioni della sfera originale (premendo quindi *S 0.99*, per uno scaling al 99%). Questa seconda sfera (che magari possiamo rinominare in *bulboTextures*, per non confonderci con l'altra) verrà dotata di Textures per definire l'aspetto dell'occhio, mentre la sfera esterna servirà per la superficie.

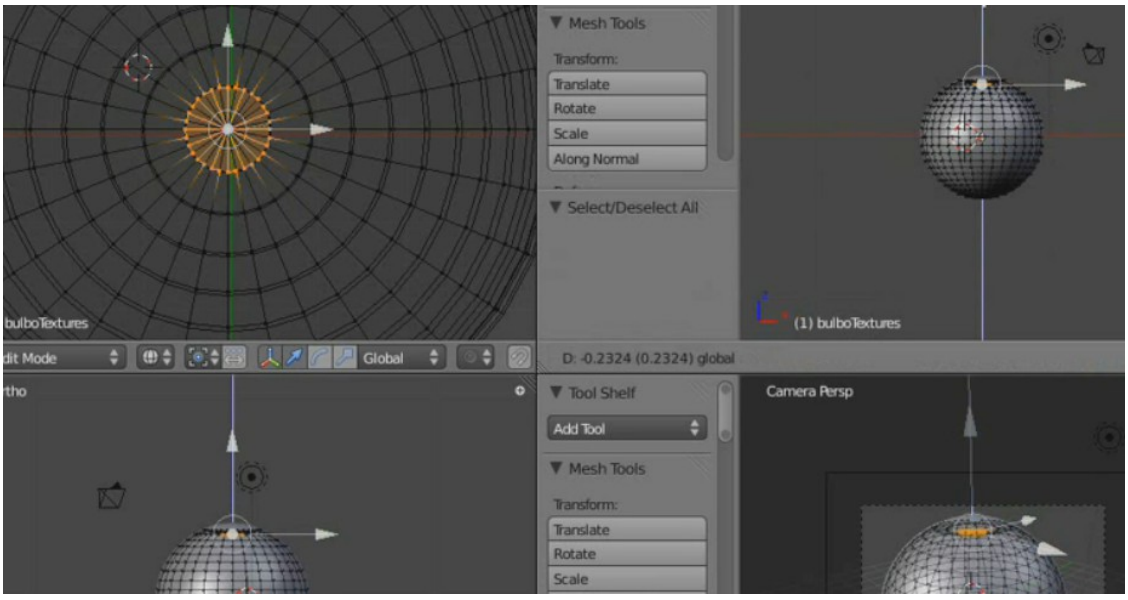
Con la prima sfera (quella più esterna) selezionata, accediamo al pannello dei Materials, aggiungendone uno, se non presente.

Specular del Material della prima sfera (quella più esterna)

Lasciamo l'ombreggiatore diffuso così com'è, ma impostiamo l'intensità del riflesso speculare a 1 e *Hardness* dello stesso a 250 o 275 circa.

Più in basso, nella scheda del materiale, attiviamo la trasparenza Z-Transp, con valore di Alpha 0. Disattiviamo, infine, la voce *Traceable*, che di default è selezionata. Questa superficie simulerà, in pratica, solo i riflessi speculari dovuti all'illuminazione.

Selezioniamo ora la sfera interna, *bulboTextures*; passiamo in Edit Mode e creiamo una piccola rientranza in uno dei poli della sfera, spostando i vertici manualmente, ove dovrà esserci l'iride.



Modificare la sfera "bulboTextures", creando una rientranza

Adesso passiamo al pannello dei materiali per questa mesh e, se non presente, aggiungiamo un nuovo Material. Le impostazioni di default vanno più che bene, più che altro il lavoro riguarda le Textures: dobbiamo crearne almeno tre per definire l'aspetto del bulbo (cioè l'iride, la pupilla e il resto), un minimo di turbolenza e la zona di influenza di questa "turbolenza", che non deve riguardare tutto il bulbo oculare, ma solo l'iride, in modo da rendere meno omogeneo il colore.

Iniziamo dalla prima Texture della pila (aggiungiamola, se non presente); tale Texture dovrà definire l'aspetto di base dell'occhio. Cambiamone il tipo in *Blend Spherical* ed attiviamo la *Color Band*, aprendo la scheda *Colors* e cliccando su *Ramp*.

La divisione all'estrema sinistra ha di default colore nero e *Alpha* a 0, e va bene così, ma dobbiamo aggiungere una nuova divisione, molto vicina alla prima, ad esempio in 0.05, con *Alpha* 1 e colore bianco. Aggiungiamo una terza divisione in posizione 0.25, con colore grigio chiaro e *Alpha* 1.

Aggiungiamo una quarta divisione a circa tre quarti della banda, fornendole *Alpha* 1 e, come colore, il colore che vogliamo dare all'iride, ad esempio verde.

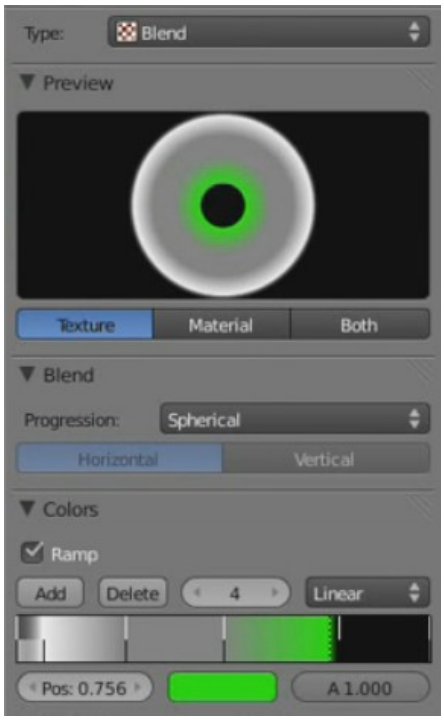
Ovviamente dobbiamo definire la pupilla, per cui aggiungiamo una quinta divisione, con colore nero e *Alpha* 1, appena a destra della divisione col colore dell'iride.

La divisione all'estrema destra dovrà avere colore nero e *Alpha* a 1, come la precedente.

L'aspetto di base è definito, ma dobbiamo mappare correttamente questa Textures sulla mesh.

Le impostazioni di mappatura che definiremo qui andranno poi utilizzate anche per le prossime mesh.

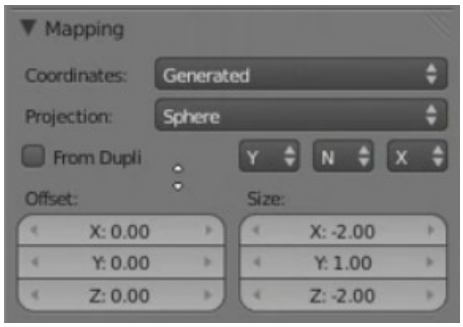
Lasciamo per *Coordinates* la voce di default, ossia *Generated*, e selezioniamo *Sphere*, cioè appunto sfera, per la proiezione.



La configurazione finale della Color Ramp della Texture Blend Spherical

Ora bisogna variare i valori di X, Y e Z *Mapping,* in modo da centrare la pupilla della Texture con la pupilla della mesh, ossia far coincidere i poli della mesh (visto che uno dei due coinciderà con l'iride) con il centro della sfera visibile nell'anteprima della Texture. Ciò va fatto impostando i valori Y, None e X, rispettivamente per i tre pulsanti X,Y e Z.

Cambiamo infine il valore di *Size X* e *Size Z* portandolo ad esempio a -2, -2,5 o -3, per adattare le dimensioni dell'iride della Texture all'iride della mesh (conviene fare un paio di rendering di prova).



Coordinate di mappatura per la Texture sull'oggetto

Con la prima Texture abbiamo finito.
La seconda Texture sarà nuovamente una Texture di tipo *Blend Sferical*, ma la utilizzeremo come maschera *Stencil*, ossia una maschera per definire le aree di influenza, sull'oggetto, delle Textures definite nei canali successivi.

Attiviamo anche per questa Texture la *Color Band*, creando quattro suddivisioni (oltre alle estremità, presenti di default). Tutte le fasce avranno colore bianco, per cui cambieranno solo le posizioni e l'*Alpha* di ciascuna suddivisione.

La divisione all'estrema sinistra e la seconda divisione, posta in posizione 0.1 circa, dovranno avere *Alpha* 1, mentre la terza divisione, posta a circa 0.15 – 0.2, dovrà avere *Alpha* a 1, così come la quarta, posta a metà *Color Band*, mentre la quinta (posta a tre quarti della *Color Band*) e la banda all'estrema destra dovranno avere *Alpha* a 0.

In pratica, abbiamo definito una "corona circolare" leggermente sfumata ai bordi.

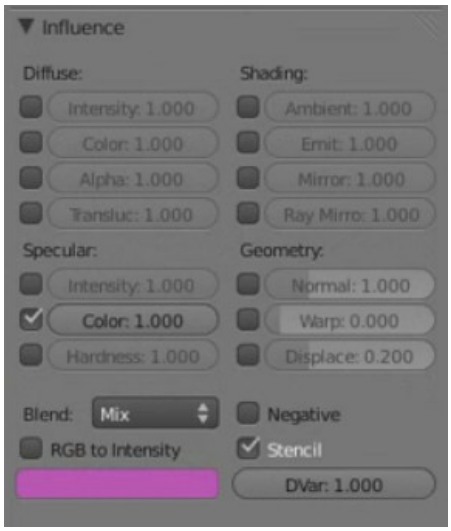
La Color Band impostata anche per la seconda Texture

Bisognerà impostare le coordinate di mappatura come fatto per la prima Texture, ossia *Generated*, *Sferica*, *Y-None-X* e valori di *SizeX* e *SizeZ* adeguati per le dimensioni dell'iride nella mesh.

In Influence, deselezioniamo *Color* da *Diffuse* e selezioniamo *Color* in *Specular*.

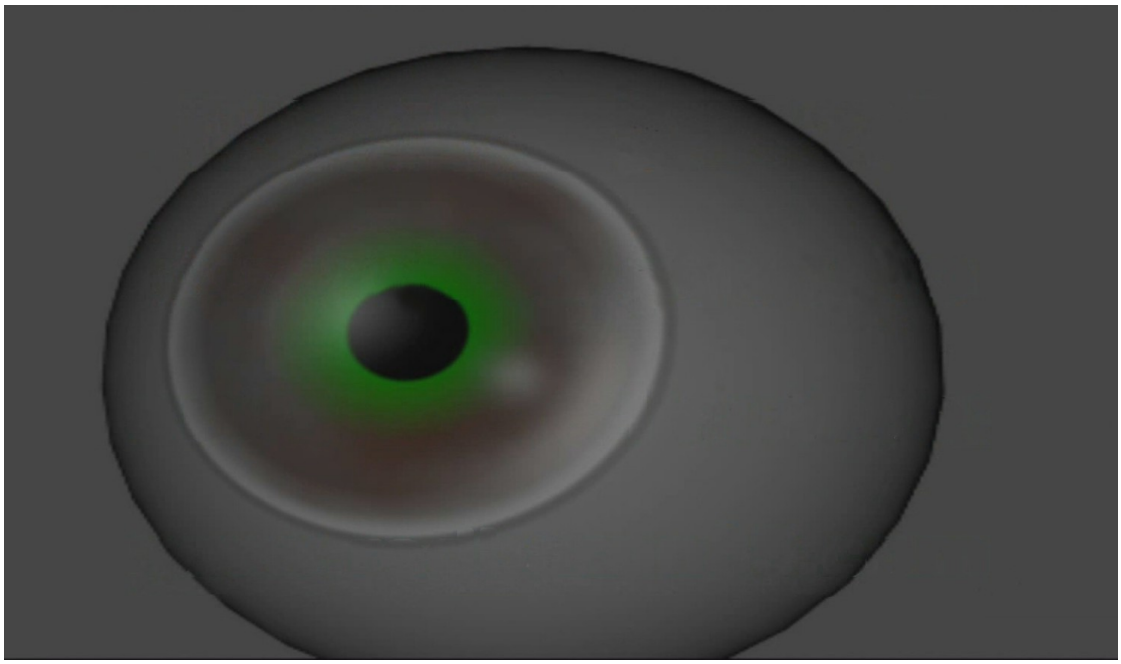
Più in basso, sul fondo della scheda, selezioniamo la casella *Stencil*.

Passiamo alla terza e (almeno nella forma base) ultima Texture. Questa Texture dovrà essere, molto semplicemente, di tipo *Clouds*, ossia nuvole, con *Noise Size* 1.2 e tipo di rumore *Hard*. Anche in questo caso bisognerà impostare le coordinate di mappatura come fatto per la prima Texture, ossia *Generated*, *Sferica*, *Y-None-X* e valori di *SizeX* e *SizeZ* adeguati per le dimensioni dell'iride nella mesh.
Come anticipato, tale Texture serve per aggiungere un po' di rumore, di turbolenza, a certe parti dell'occhio; tali parti vengono delimitate grazie alla Texture al secondo canale, di tipo *Stencil*.



Impostazioni Influence e Stencil per la seconda Texture

Nelle Texture *Stencil*, la parte bianca indica quali parti delle Texture dei canali successivi *non* verranno applicate, per cui con la "corona circolare" abbiamo coperto tutto il bulbo eccetto l'iride.
Il colore di default dell'effetto è viola, che in genere non è proprio adatto allo scopo: meglio rosso, rosso-scuro. L'immagine seguente mostra un rendering del risultato finale ottenuto.


Rendering del risultato finale

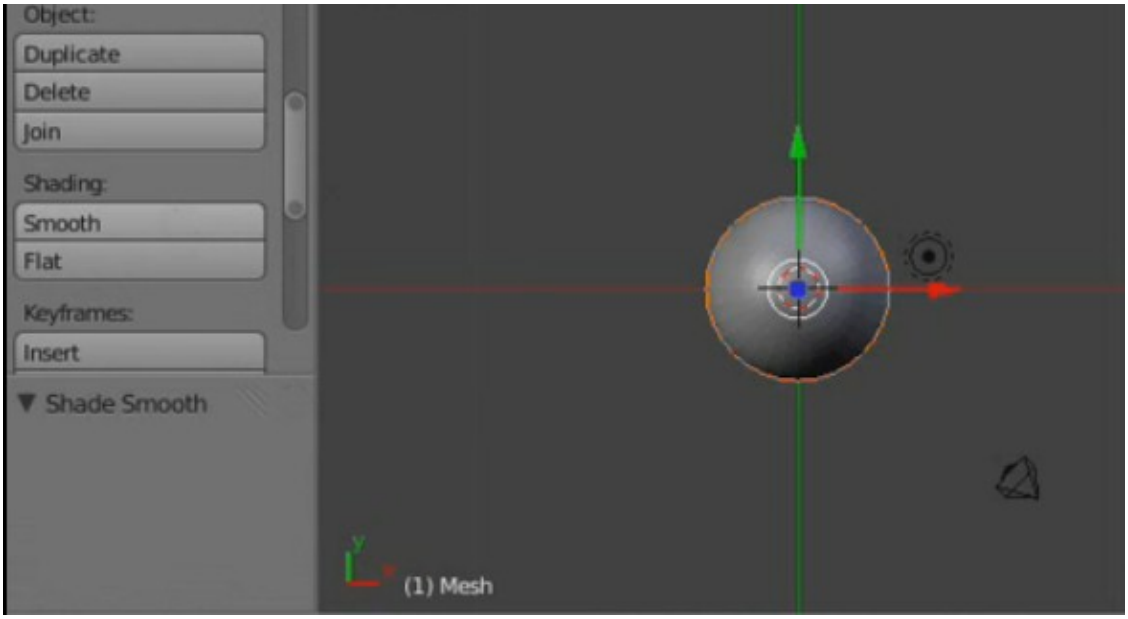
Il bulbo oculare è quindi pronto per essere inserito nella sua orbita, nel vostro personaggio!

* * *

Bolle di sapone con Particle Emitter e Texture

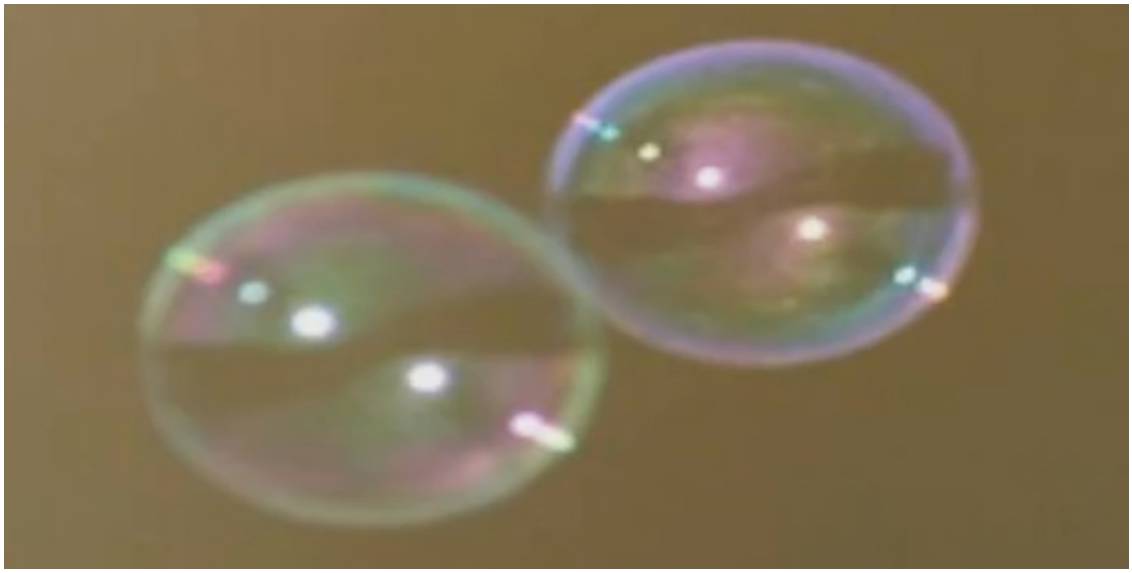
In questo tutorial vedremo come realizzare delle bolle di sapone con Blender.
L'idea può sembrare semplice, ma si tratta di un esercizio utilissimo per comprendere alcuni elementi della mappatura delle Textures e qualche impostazione del sistema particellare *Emitter*.

Avviamo Blender, togliamo dalla scena il cubo presente di default ed aggiungiamo una sfera, cliccando sul pulsante *Smooth* presente alla voce *Shading* nella *Tool Shelf* per levigarla un po'.



La sfera levigata (ombreggiatura Smooth) nella scena 3D

Per capire che tipo di materiale ci serve, osserviamo un paio di immagini reali.
Notiamo che sono presenti, in buona quantità, dei riflessi, ma soprattutto al centro, meno ai bordi, e lo stesso vale per la trasparenza. I riflessi speculari sono concentrati e relativamente intensi.
Le bolle sono particolarmente luminose.



Un esempio di bolle di sapone reali

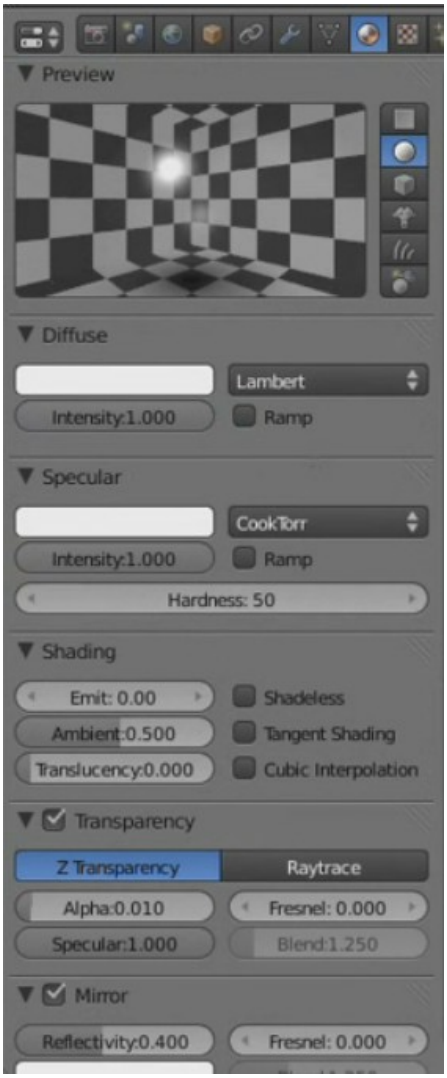
Torniamo quindi a Blender ed aggiungiamo, se non è già presente, un Material di base per la sfera.

Lasciamo il tipo di ombreggiatore diffuso a *Lambert*, cambiando però il colore in bianco puro e portando l'intensità a 1. Per l'ombreggiatore speculare, di tipo *CookTorr*, andranno bene i valori di intensità 1 e *Hardness* 50.

Attiviamo la trasparenza, di tipo *Z-Transp*, impostando come valore per *Alpha* un valore tra 0.01 e 0.05. Attiviamo anche la voce *Mirror*, in basso, impostando come colore bianco puro e intensità dei riflessi 0.4.

Per mappare i riflessi, la luminosità e le trasparenze come descritto in precedenza, dobbiamo utilizzare una Texture, per cui aggiungiamone una per questo materiale.

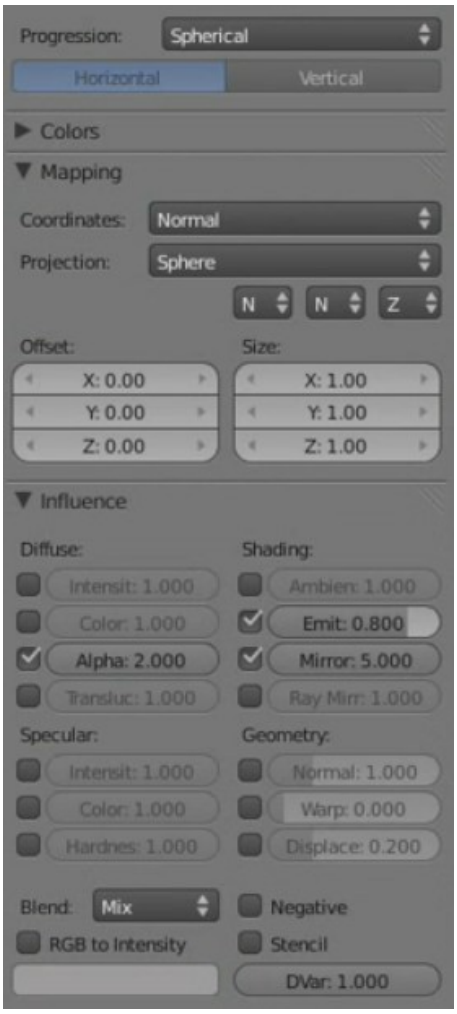
La Texture dovrà essere di tipo *Blend Spherical*, ma non basta: per far sì che gli effetti rimangano uguali ovunque si trovi la sfera, e con qualsiasi orientamento della stessa, dobbiamo



Il Material della bolla di sapone

impostare come voce per *Coordinates*, nella scheda *Mapping*, la voce *Normal*, ossia i vettori normali alle facce, mentre la proiezione dovrà essere di tipo *Sphere*, cioè appunto sfera, e la mappatura dovrà avvenire solo per le coordinate Z, per cui impostiamo Mapping X e Y a *None* e lasciamo Z come Mapping Z. In *Influence* disattiviamo *Color*, attivando invece *Alpha* (portando il valore magari a 2) e *Mirror*, portando il valore di quest'ultimo a 4 o 5.

Per aggiungere un po' di luminosità alla sfera, attiviamo anche il canale *Emit*, ricorrendo così ad un "trucchetto" che fisicamente non è realistico ma che ci aiuta a realizzare un buon effetto per il materiale. Come valore, impostiamo ad esempio 1 o 1.2, comunque dipende anche dal tipo di scena che state realizzando: se la scena non è ambientata all'aperto, in pieno giorno, impostate ad esempio 0.6 o 0.7, e così via.



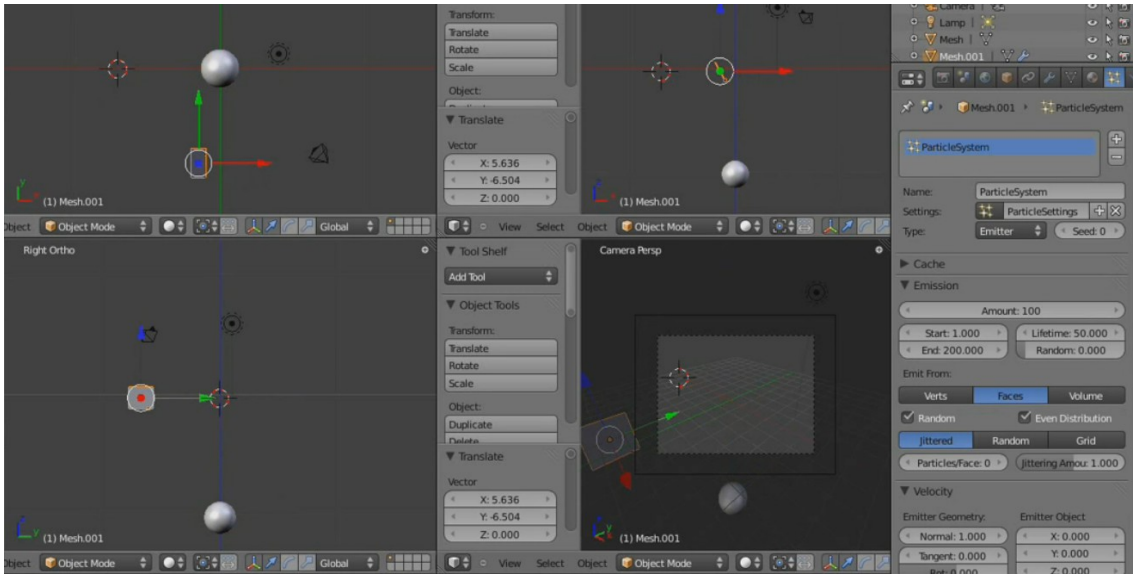
Le impostazioni della Texture della bolla di sapone

Cambiamo infine il *colore* di questa Texture, in basso nel pannello, in bianco o grigio molto chiaro.

In questo caso sto utilizzando la sfumatura di default per *Blend Spherical*, ma potete provare a definire meglio le zone di influenza, ad esempio per confinarle ancora di più ai bordi, attivando la *Ramp* della scheda *Colors* della Texture ed aggiungendo delle divisioni interne, variando poi colore e *Alpha* delle varie bande.

Definito l'aspetto della bolla, vediamo di creare un sistema particellare per realizzare delle animazioni. Creiamo quindi *un oggetto emettitore* e passiamo al pannello delle particelle dello stesso. Aggiungiamo un sistema particellare, che di default sarà *Emitter*: emettitore, proprio quello che serve a noi.

Impostiamo un valore basso per *Amount*, ad esempio 100, ed impostiamo i valori di *Start*, *End* e *Lifetime* come preferiamo, in base alla nostra animazione.

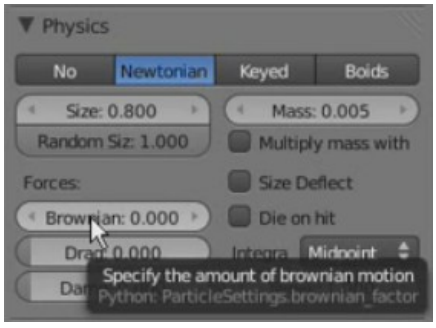


Il sistema Emitter per l'oggetto (mesh) emettitore

La prima cosa da fare è scendere nel pannello per accedere alla scheda *Render* e selezionare la casella *Object*, scrivendo (nel campo testuale di etichetta *Duplication Object*) il nome della sfera o della mesh da utilizzare come riferimento.

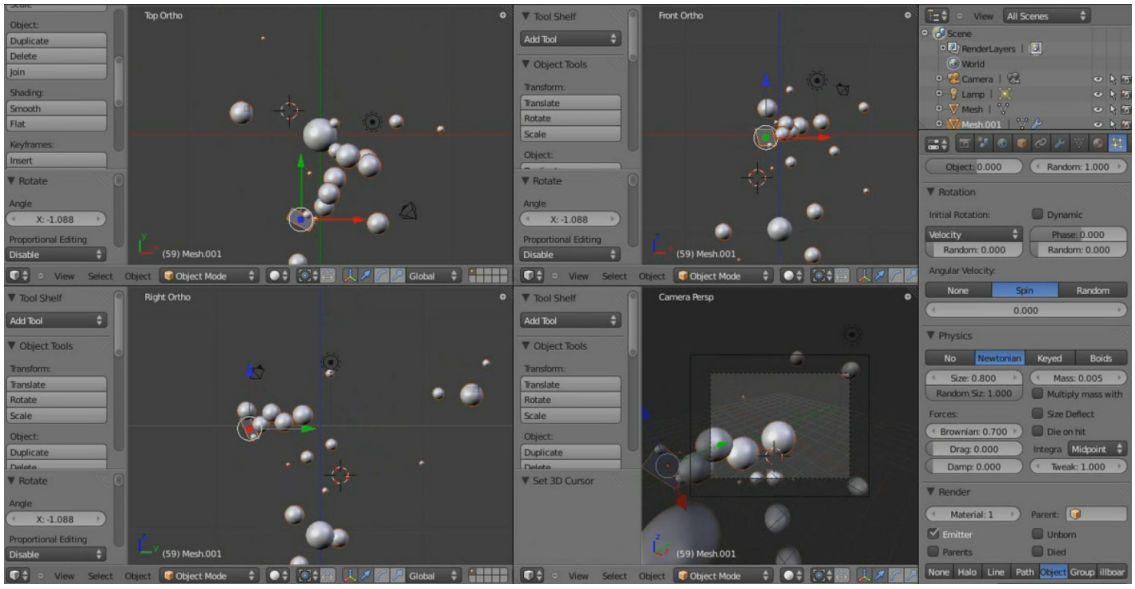
Nella scheda *Velocity* (velocità) portiamo a 5 o 7 il valore di Z per Emitter Object e 1 per Random. Nella scheda *Physics*, definiamo in base alle nostre esigenze, cioè alla scala della scena, il valore per *Size*, la dimensione media delle mesh, dimensione media perché dobbiamo impostare un valore diverso da zero, ad esempio proprio 1, per *Random Size*, in quanto le bolle di sapone non sono proprio tutte uguali. Nel campo *Mass*, la massa dell'oggetto, impostiamo un valore estremamente basso, come ad esempio 0.005.

La vera "chicca" nella simulazione delle bolle di sapone sta però nell'utilizzo del campo *Brownian*, che sta per moto Browniano: un moto caotico che generalmente caratterizza, nella realtà, particelle molto piccole, ma che si presta benissimo anche per le bolle di sapone, almeno nella nostra simulazione, per aggiungere proprio un po' caoticità al loro moto. Impostiamo come valore, ad esempio, 0.7.



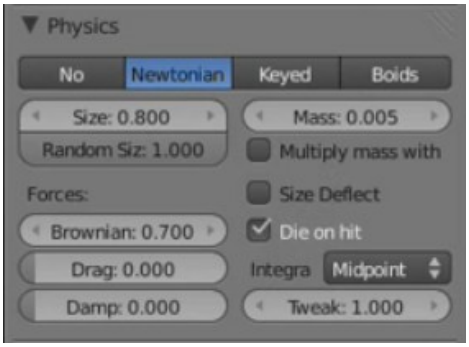
Le impostazioni Brownian per il sistema particellare Emitter

Tutto è pronto per la simulazione e possiamo realizzare, già a questo punto, i nostri rendering.



Anteprima dell'animazione casuale nella 3D View

Una considerazione che riguarda le collisioni con altri oggetti è, comunque, la seguente: per far sparire le bolle di sapone in seguito ad una collisione con un oggetto della scena, selezioniamo la casella *Die On Hit* (cioè letteralmente "muori all'impatto") nella scheda *Physics* del sistema particellare; dovremo poi accedere al pannello *Physics* degli oggetti presenti nella scena e renderli attori "standard", cioè con le impostazioni di default, delle collisioni, cliccando sul pulsante *Add* nella scheda *Collision*.



"Die on hit" per le bolle di sapone

Con questo piccolo accorgimento, potete utilizzare il sistema qui descritto per qualsiasi scena.

Faccio notare infine che, mentre nella realtà siamo vincolati alla forma sferica o quasi sferica, con Blender possiamo realizzare bolle di sapone di qualsiasi forma, passando come oggetto di riferimento nel campo *Duplication Object* del sistema particellare un'altra mesh, per cui potremmo dare sfogo alla fantasia e alla creatività realizzando bolle di sapone con le forme di oggetti o animali per realizzare spot pubblicitari o videoclip.

Lo stesso discorso vale per il colore della bolla o dei suoi bordi, che possiamo variare cambiando o i colori degli ombreggiatori diffuso e speculare del Material oppure il colore della Texture utilizzata.

Esempio di rendering finale

* * *

Animazioni materials (materiali delle mesh), curve IPO e Graph Editor

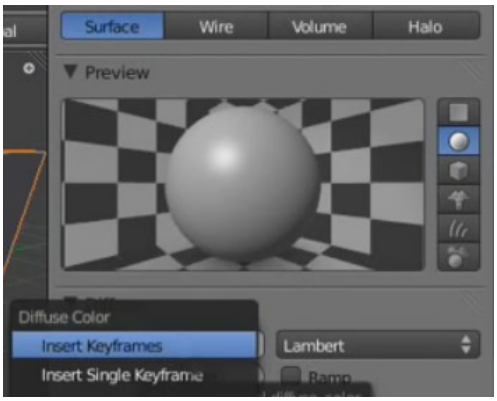
In questo tutorial parleremo delle animazioni dei Materials in Blender; in particolare, vedremo come animare i parametri dei materiali (materials) degli oggetti in Blender (colore diffuso, colore speculare, tipo di ombreggiatore, altri valori), modificando anche le curve IPO nel Graph Editor.

In Blender, si sa, tutto è "animabile", ossia esistono *curve IPO* praticamente per qualsiasi cosa.
Le impostazioni di un Material non fanno eccezione, per cui vediamo come fare.

Selezionato un oggetto, creiamo per lo stesso un Material di base, se non esiste già.

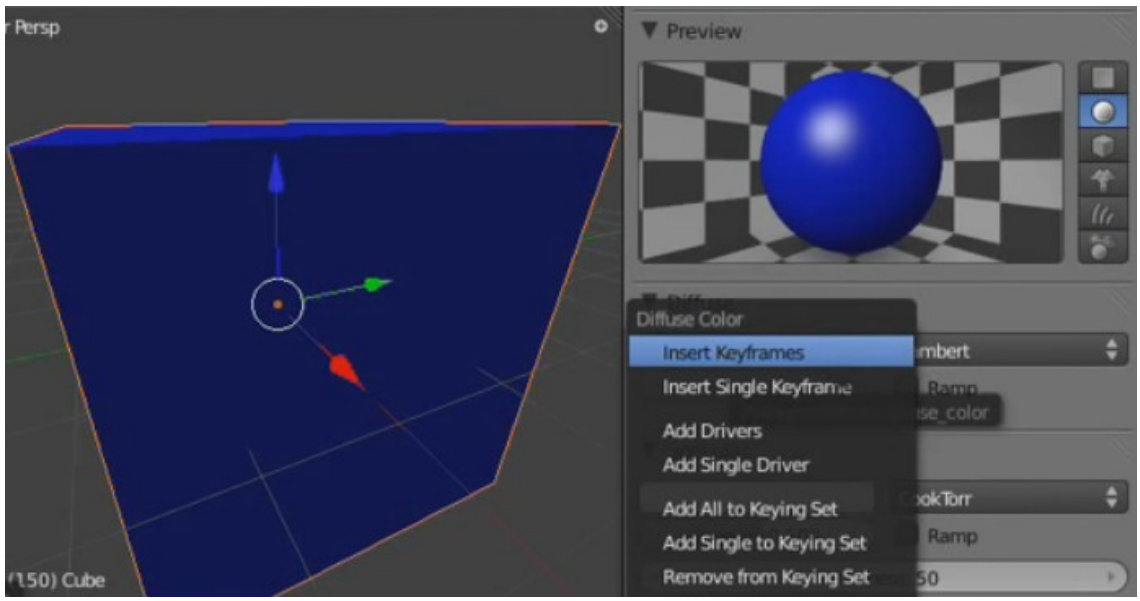
Al frame 1, lasciamo i valori di default per il colore diffuso e l'intensità dello stesso, ma facciamo click col tasto destro del mouse sul rettangolo con il colore diffuso e, dal menù che apparirà a video, selezioniamo la prima voce, ossia "*Insert Keyframes*".

In questo modo stiamo "registrando" il colore diffuso, o meglio i valori delle componenti RGB dello stesso, come vedremo, per il frame 1.



Insert Keyframes per registrare, in questo caso, il colore del canale diffuso di un Material

Adesso spostiamoci, ad esempio, al frame 150, cambiamo il valore dell'intensità dell'ombreggiatore diffuso, portandolo ad esempio a 1, ma *non* inseriamo un keyframe per questo elemento... vedremo presto perché. Sempre al frame 150, variamo il colore diffuso e ripetiamo l'operazione di inserimento del keyframe, facendo click col tasto destro del mouse sul rettangolo del colore e selezionando *Insert Keyframes*.

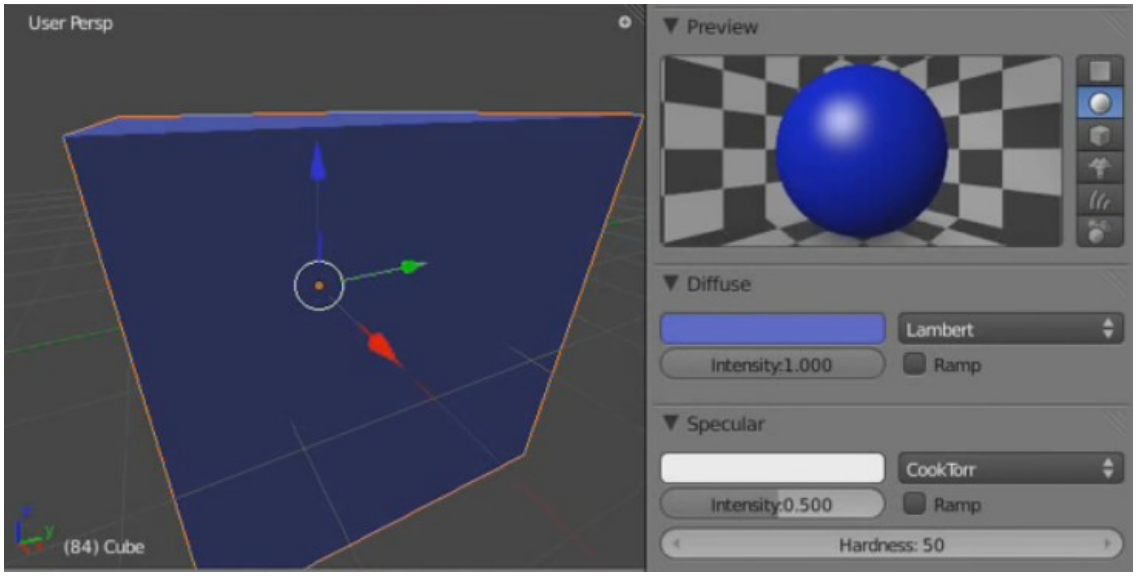


Inserimento del Keyframe, per il colore modificato, al frame 150

Abbiamo realizzato l'interpolazione, ovvero l'animazione del colore diffuso del materiale, e mandando in esecuzione l'anteprima dell'animazione (*ALT A*) vedremo effettivamente l'oggetto cambiare colore. In fase di anteprima non tutte le animazioni possono essere visualizzate; ad esempio, non è possibile osservare eventuali variazioni nel colore speculare dell'oggetto in queste anteprime, ma solo in fase di rendering vero e proprio.

Ogni volta che inseriamo un keyframe per una voce del pannello del materiale, lo inseriamo SOLO per quella voce, non per tutte le impostazioni del pannello.

Per avere una conferma di ciò, osserviamo (in vari momenti dell'animazione) il valore di intensità dell'ombreggiatore diffuso: se lo cambiamo e impostiamo un keyframe per il colore, ma non per il campo intensità, per questo campo Blender utilizzerà l'ultimo valore impostato manualmente (nel mio caso, 1).

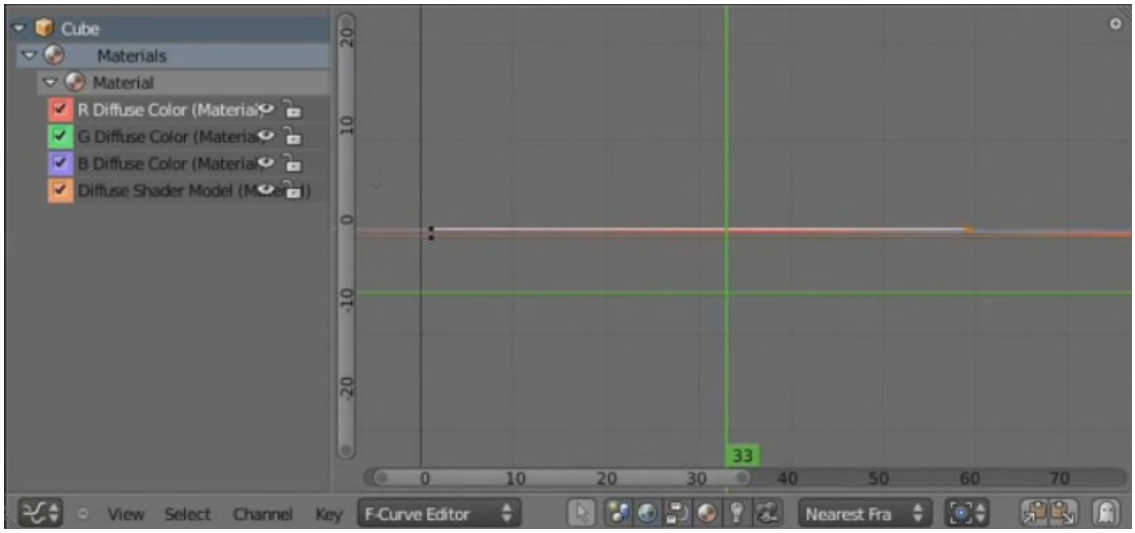


Il campo intensità, non "registrato" con un Keyframe, non viene modificato

Con il colore si ha un'interpolazione “dolce”, nel senso che si passa gradualmente da un set di valori RGB ad un altro nel corso dell'animazione, ma per altre voci non è così: possiamo impostare, ad esempio, un keyframe per il *tipo* di ombreggiatore diffuso (*Oren-Nayar*, *Toon*, *Lambert*, ...), spostarci in un altro frame, cambiare il tipo di ombreggiatore e impostare un keyframe per tale campo, ma in questo caso il passaggio sarà *istantaneo*: non si può cioè passare gradualmente tra due diversi tipi di ombreggiatura, si può avere solo o l'uno o l'altro in un determinato istante di tempo.

Questo discorso vale, in generale, per le voci scelte dai menù combo, come appunto il tipo di ombreggiatore, o per le voci checkbox, cioè le caselle da selezionare o meno, mentre quando si ha a che fare con valori numerici si può effettuare un passaggio graduale.

Diamo ora un'occhiata al grafico delle *curve IPO* del materiale per vedere cosa si può fare lì. Nelle versioni precedenti di Blender vi era la finestra *IPO*, ora invece dobbiamo cambiare una finestra in *Graph Editor*. Aperta una finestra *Graph Editor*, dovremmo poter individuare un menù Materials per l'oggetto, visto che abbiamo già inserito dei keyframe per i materiali. A destra, nel grafico contenente *Timeline* e valori, dovrebbero essere presenti (per lo stesso motivo) delle curve IPO.



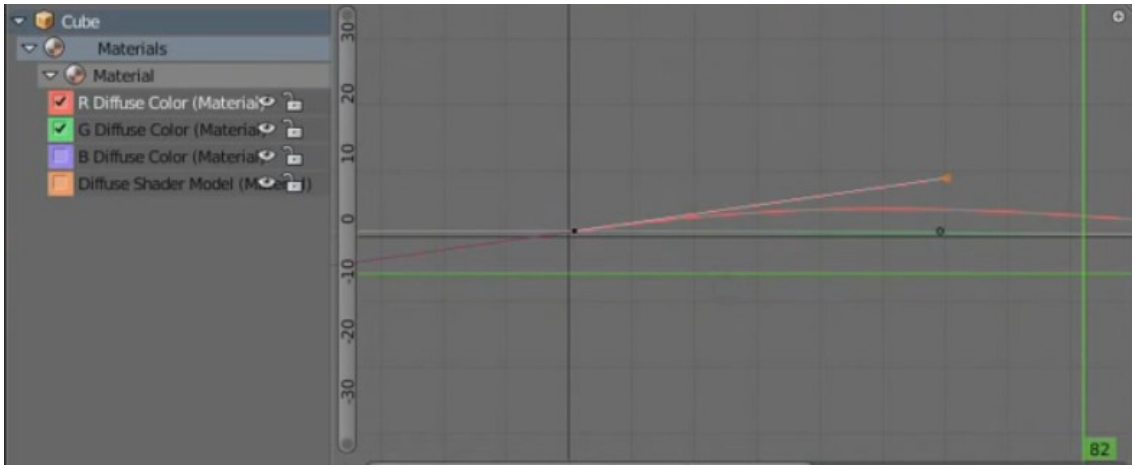
La finestra Graph Editor, con le IPO del Material dell'oggetto selezionato

Espandiamo il menù a sinistra cliccando col tasto sinistro del mouse sulla freccia posta accanto all'etichetta Materials ed espandiamo anche il menù Material. A questo punto avremo un elenco dei campi per i quali esistono delle IPO, ossia nel nostro caso i valori dei canali RGB del colore diffuso e il campo *Diffuse Shader Model* (il tipo di ombreggiatore diffuso)

Possiamo selezionare una curva IPO in questo menù per poi modificarla manualmente nella parte destra della finestra o possiamo renderla invisibile per semplificare il grafico che, in presenza di più curve IPO, anche per altri canali, potrebbe diventare troppo affollato.

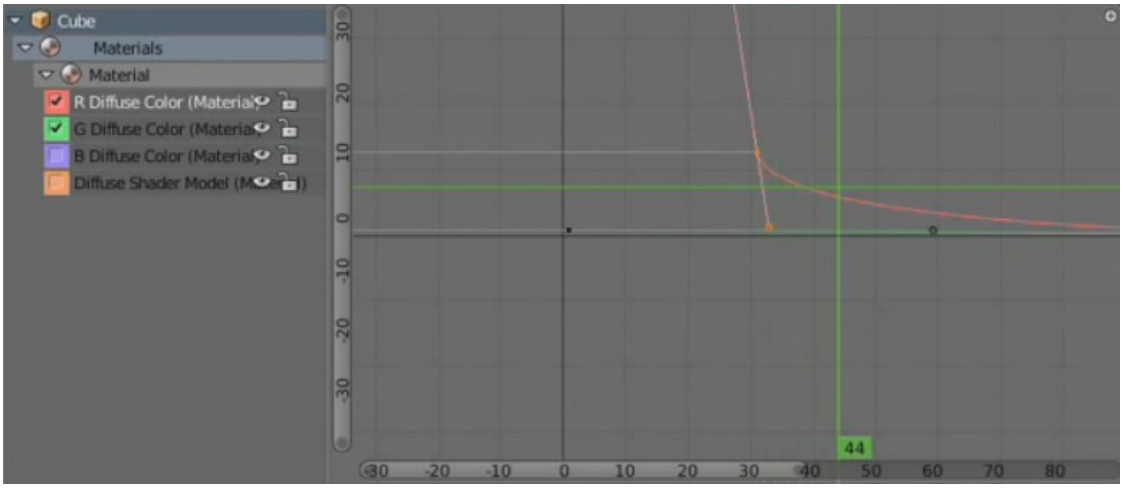
Selezionata una curva, nella parte destra potremo quindi selezionarne i punti di controllo, cliccando su di essi col tasto destro del mouse; tali punti individuano i keyframe della curva, e potremo ad esempio spostarli, non solo nel range di valori ma anche nel tempo, modificando così l'animazione di quella curva.

Possiamo quindi spostare, ruotare e scalare i punti di controllo selezionandoli col tasto destro del mouse (oppure premendo *B* e definendo un'area di selezione col mouse o ancora premendo *A* per selezionarli e deselezionarli tutti) premendo rispettivamente poi *G*, *R* e *S* per effettuare traslazioni, rotazioni e scaling.



Modificare una curva IPO (e, quindi, l'animazione associata)

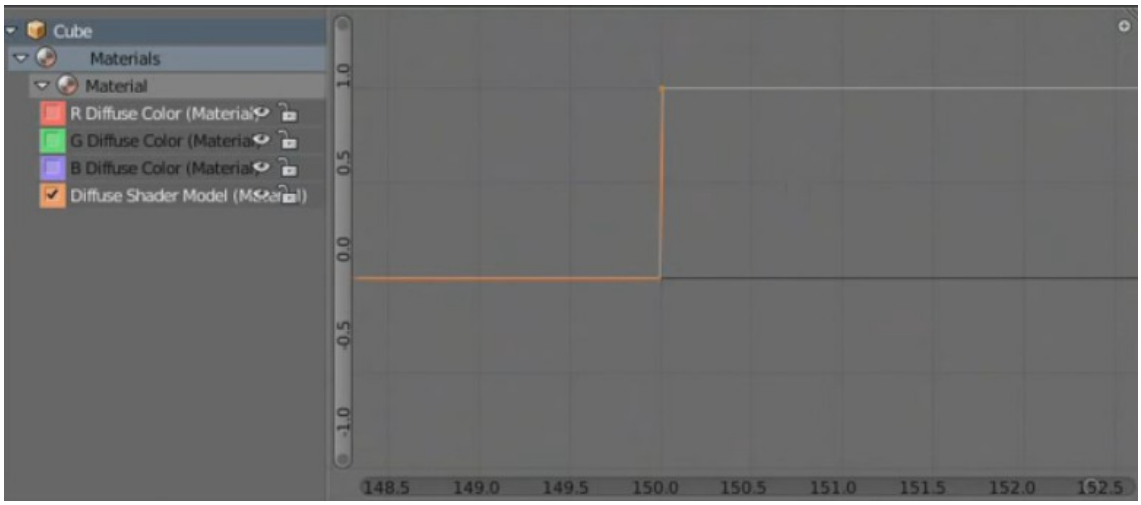
La rotazione dei keyframe ha senso per modificare la curvatura dell'interpolazione, che poi si traduce in cambiamenti più o meno veloci del valore controllato dalla curva (ricordiamoci infatti che questo è un grafico dove l'asse delle x è il tempo e l'asse delle y il valore, per cui la forma della curva indica come varierà il valore nel tempo).
Ridimensionare un punto di controllo significa scalare la lunghezza delle sue maniglie e anche questa operazione ha effetti sulla forma, cioè sull'andamento, della curva IPO.



Scaling di un punto di controllo (modificando così la lunghezza delle sue maniglie e, quindi, l'andamento della curva)

È poi possibile effettuare altre operazioni sulle curve IPO, manipolando i vertici, estendendoli, cambiando il tipo di controllo delle maniglie con *V* (nota: nella versione 2.5.0 lo shortcut era *H*, poi cambiato in *V*) e così via, ma quello della manipolazione delle curve IPO è proprio un argomento diverso che andrebbe trattato a parte, non possiamo parlarne qui.

Come ultima osservazione, verifichiamo quanto detto prima esaminando la curva *IPO* del *Diffuse Shader Model*, ossia del tipo di ombreggiatore diffuso.



La curva IPO di tipo "soglia" del Diffuse Shader Model

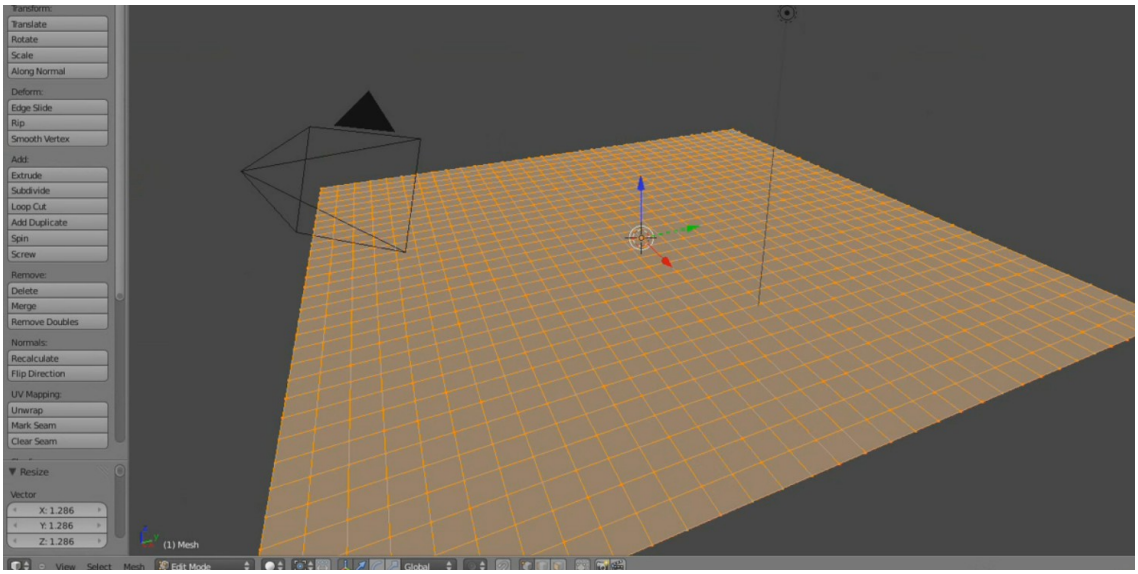
Si tratta di una curva di tipo “soglia”: il valore cambia bruscamente, non è possibile ruotare i punti di controllo, non esistono maniglie e, quindi, non è possibile neanche scalare i keyframe e l'operazione di traslazione è libera lungo l'asse temporale, mentre procede “a scatti” in verticale ma non bisognerebbe MAI traslare in verticale questo tipo di curva perché tale operazione potrebbe portare a risultati imprevedibili.

La manipolazione di una curva IPO del genere avviene quindi proprio in linea con quanto detto precedentemente sull'animazione di questi elementi un po' più particolari.

* * *

Sculpt Mode: descrizione strumenti e applicazioni

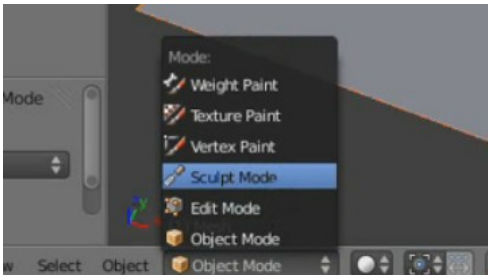
In questo tutorial parlerò brevemente della modalità di modellazione delle mesh denominata *Sculpt Mode* di Blender 2.5. Utilizziamo per le nostre prove un Plane con un buon numero di suddivisioni.



Il Plane, suddiviso un bel po' di volte, da utilizzare per le nostre prove

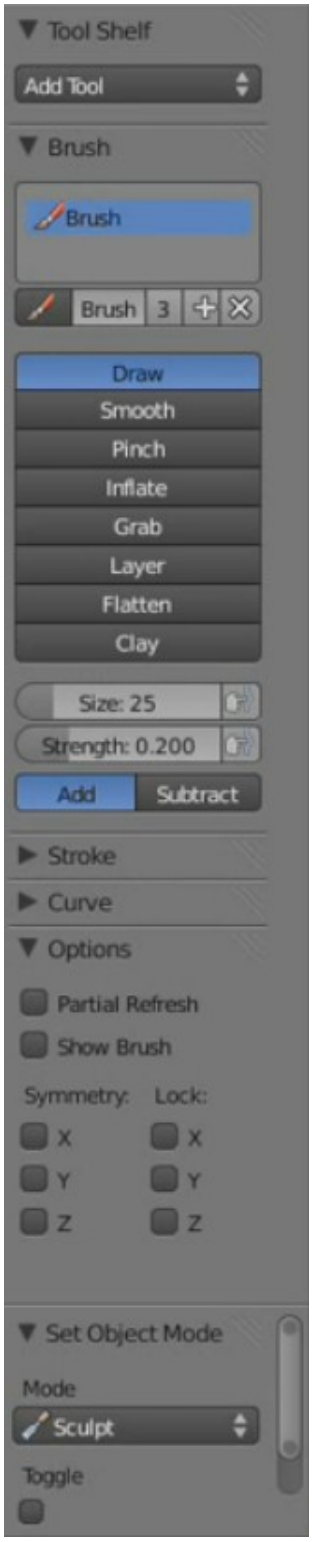
Ovviamente potrete utilizzare lo *Sculpt Mode* per qualsiasi mesh, ma assicuratevi che ci sia un buon numero di vertici, visto che lo "scalpello" dello *Sculpt Mode* lavora proprio su insiemi di vertici, ed in particolare sui vertici che si trovano all'interno della sua area di applicazione.

Si passa in modalità *Sculpt Mode* selezionando proprio questa dicitura nel menù a scomparsa *Mode* presente nell'header di ogni finestra di navigazione 3D.



Passaggio alla modalità Sculpt

Una volta entrati in modalità *Sculpt Mode* avremo a disposizione, nella Tool Shelf, tutti gli strumenti peculiari di questa modalità, a cominciare dagli strumenti raggruppati sotto l'etichetta *Brush*, ossia letteralmente spazzola. Io parlerò dello strumento *Sculpt* chiamandolo, in genere, scalpello, ma non si tratta di una denominazione corretta, in quanto (a seconda della modalità selezionata in *Brush*) sarà possibile, con un click del tasto sinistro del mouse e trascinando, creare dei rilievi nella mesh, trascinare gruppi di vertici e altro ancora.



Gli strumenti della modalità Sculpt, nella Tool Shelf

La cosa bella dello *Sculpt Mode* è che le varie modalità di utilizzo dello "scalpello" hanno nomi che dicono chiaramente a che cosa servono:

- *Draw*, cioè letteralmente "traccia, disegna"; crea una piccola superficie curva sul modello, come una collinetta (se in modalità *Add*, altrimenti una depressione, in modalità negativa o *Subtract*);
- *Smooth* leviga, smussa la superficie; non aggiunge, comunque, nuovi vertici per smussare porzioni di Mesh, ma si limita a deformare quelli esistenti per rendere meno bruschi i passaggi tra le facce;
- *Pinch*, letteralmente "pizzicotto", tira i vertici che rientrano nell'area di applicazione dello strumento verso il centro di tale area, se in modalità *Add*, o li allontana dal centro, se in modalità negativa;
- *Grab* serve semplicemente a traslare (spostare) gruppi di vertici;
- *Inflate* spinge i vertici nella direzione delle loro normali, in positivo se in modalità *Add* o in negativo se in modalità *Subtract*; non si tratta dello stesso effetto del trascinamento perché, come anticipato, la direzione per ciascun vertice è quella della sua Normale, e questa differenza concettuale si traduce, nella pratica, nel fatto che se due vertici con Normali non parallele vengono sottoposti ad *Inflate*, divergeranno o si avvicineranno, a seconda della modalità di applicazione dello scalpello e della direzione delle loro Normali;

- *Layer* crea un effetto simile a *Draw*, ma meno turbolento: disegna un rilievo "dolce" e praticamente uniforme nella parte centrale;
- *Flatten* significa letteralmente "appiattire", ed è proprio quello che fa questo strumento, non c'è molto altro da dire;
- *Clay* significa "argilla" e l'intento quindi è di modellare la mesh, aggiungendo rilievi o creando depressioni, dando l'impressione che si stia modellando l'argilla; è un effetto che, personalmente, apprezzo di più con un valore modesto, come 0.1 o 0.2, di *Strength*.

I "pennelli" disponibili con la modalità Sculpt

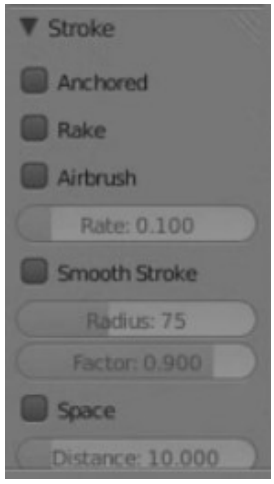
Anche lo scopo dei pulsanti *Size*, *Strength*, *Add* e *Subtract*, più in basso nella scheda, è ovvio: definire dimensione e intensità, nell'applicare l'effetto, dello scalpello, e se l'operazione va effettuata in positivo o in negativo, ad aggiungere o a sottrarre.

A parte Size, comunque, questi pulsanti non sono disponibili per TUTTI gli effetti dello scalpello, ma solo quando hanno senso (ad esempio per *Grab*, che in pratica consente di trascinare, traslandoli, i vertici: in questo caso non ha senso definire l'intensità o una modalità di applicazione "negativa").

Nella sezione *Stroke*, che letteralmente significa "tratto" (nel senso del tipo di tratto di una penna o di un pennino), si trovano alcune opzioni interessanti:

- *Anchored* lega l'effetto alla posizione iniziale, per cui se farete click e trascinerete il mouse l'effetto verrà applicato comunque centrandolo sulla posizione dello scalpello al momento del click;
- *Rake* ruota la direzione di applicazione dell'effetto per farla coincidere con la direzione del click, del tratto;
- *Airbrush* è davvero interessante perché consente di applicare l'effetto per tutto il tempo in cui si tiene premuto il tasto sinistro del mouse, per cui non c'è bisogno di cliccare più volte per ottenere l'effetto desiderato, basta mantenere la pressione; la frequenza di applicazione dell'effetto può essere regolata mediante il campo numerico *Rate*;

- *Smooth Stroke* si occupa, come suggerisce il nome, di rendere più "dolce" il tratto, l'applicazione dell'effetto;
- *Space* limita l'effetto dello strumento ad un'area circolare di raggio pari al valore definito in *Distance*.



Opzioni di Stroke (il tratto)

Le altre opzioni del pannello consentono di modificare ulteriormente la modalità di applicazione impostando ad esempio vincoli sulla direzione di applicazione o forzando la simmetria dell'effetto rispetto a uno o più assi; spesso torna molto utile, poi, la funzione *Partial Refresh*, presente nel gruppo *Options*, che, se attivata, effettuerà l'aggiornamento del modello solo per le parti influenzate direttamente dallo scalpello, tralasciando di aggiornare l'aspetto visivo delle altre, in modo da non rallentare troppo il computer quando si ha a che fare con mesh particolarmente complicate o in presenza di altri modificatori.

Una cosa veramente interessante è che se si dispone di una tavoletta grafica, l'intensità dell'effetto verrà applicato proporzionalmente all'intensità della pressione del pennino sulla tavoletta.

Si tratta di una funzionalità davvero notevole che consente di lavorare in maniera ancora più "naturale" con lo *Sculpt Tool*, perché dà l'impressione di modellare direttamente la mesh variando la pressione del "tratto", anziché mediante un click di intensità costante.

In genere, *Sculpt* viene utilizzato per realizzare corpi umani o parti di essi, come ad esempio la testa (partendo anche da una semplice sfera) o i muscoli delle braccia, dell'addome e delle gambe; va bene, ovviamente, anche per animali, mostri, alieni e altre creature "organiche".

È possibile utilizzare lo strumento *Sculpt*, ad esempio, per modellare dei paesaggi, magari facendo ricorso poi a *SubSurf* (il modificatore *Subdivision Surface*) o Texture per il canale delle Normali e del *Displacement*, se necessario; in questo modo si otterranno paesaggi magari non proprio realistici, ma buoni ad esempio per ambientazioni o illustrazioni fantasy.

* * *

Distruggere le mesh con Explode ed Emitter

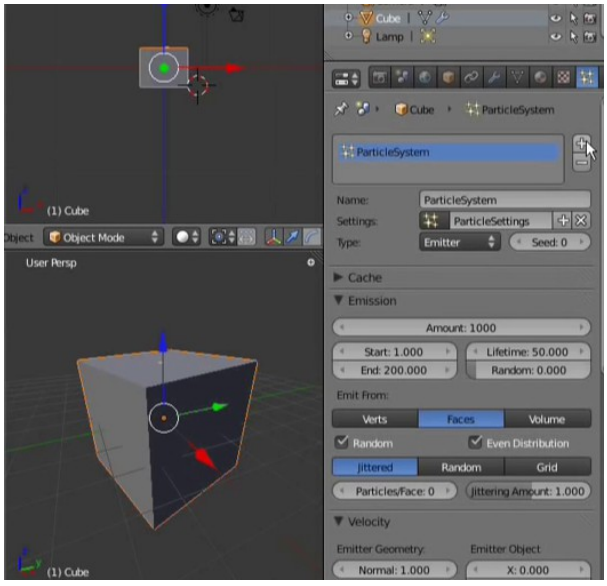
In questo tutorial vedremo come utilizzare il modificatore *Explode* di Blender 2.5 per far "esplodere" e disintegrare, come suggerisce il nome, le mesh.

Explode è un modificatore interessante ma il suo utilizzo non è immediato, infatti inserendolo direttamente per un oggetto non si otterrà alcun risultato.

Il tutto deriva dal fatto che *Explode* "dipende" da un sistema particellare, per cui per prima cosa bisogna aggiungere un sistema particellare alla mesh da disintegrare.

In questo caso, sto utilizzando un cubo per provare *Explode*, ma ovviamente potete utilizzare qualsiasi mesh; consiglio, comunque, di suddividere un bel po' di volte l'oggetto, o almeno di attivare la voce *Split Edges* (che tratteremo tra poco) del modificatore.

Passiamo quindi alla scheda *Particles* dell'oggetto ed aggiungiamo un sistema particellare di tipo *Emitter.*

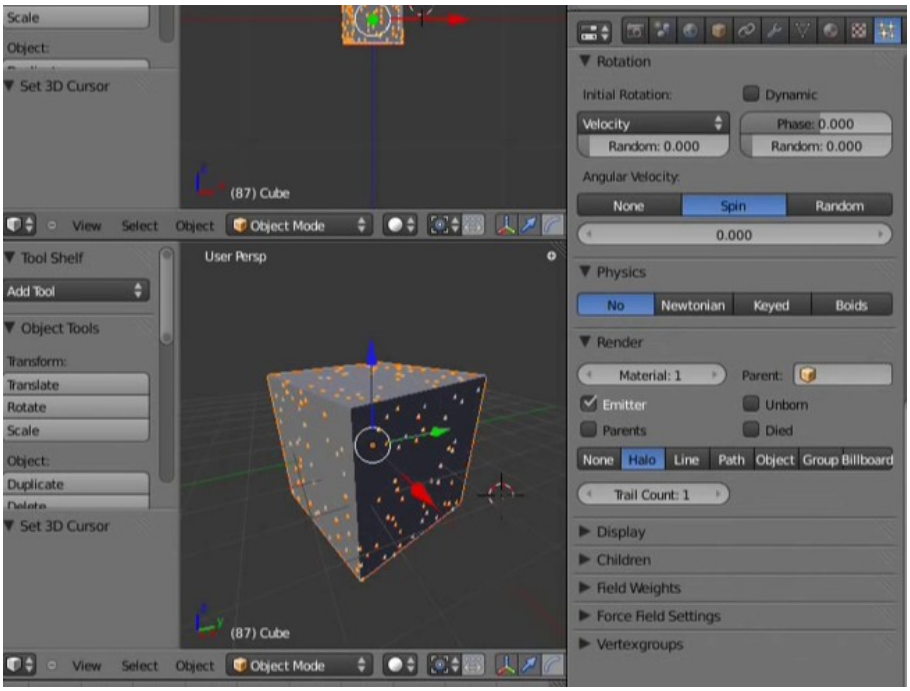


Aggiungere un sistema particellare Emitter all'oggetto

Nelle sezioni *Emission*, *Velocity* e *Rotation* sarà possibile impostare, come suggeriscono le etichette, alcune opzioni riguardanti la quantità di particelle emesse, la durata della loro vita, il frame di inizio e quello di fine emissione, la velocità, la direzione e l'orientamento iniziale delle particelle ed altri parametri. Le particelle sono soggette, di default, alla forza di gravità, ma potete variare ad esempio

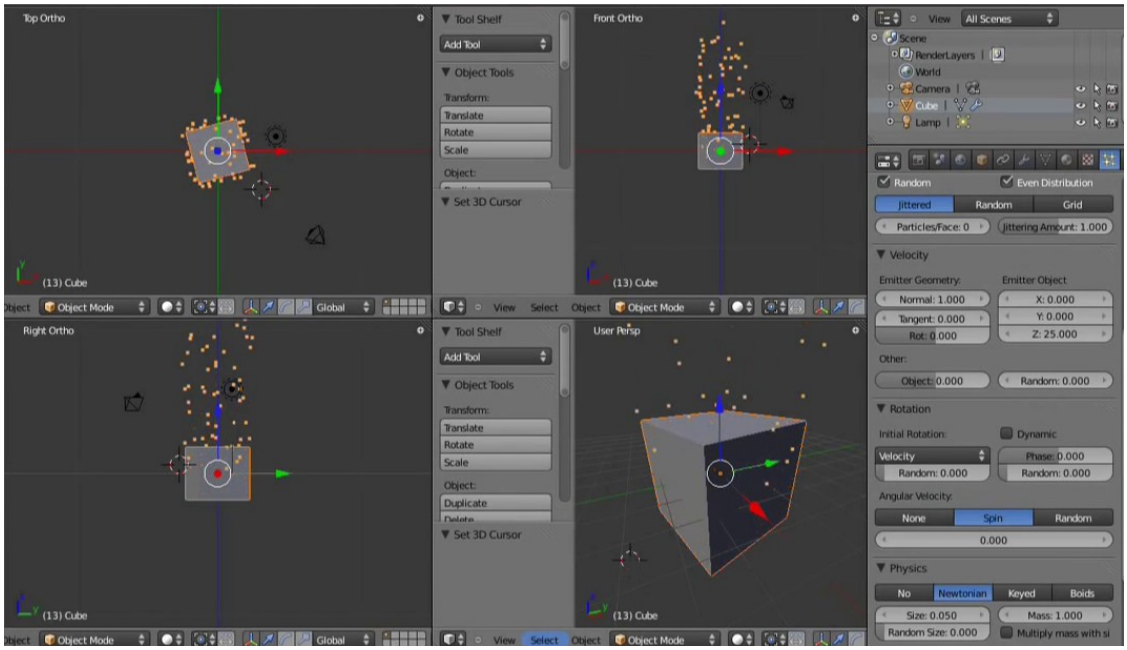
la loro massa, nella sezione *Physics* della scheda *Particles*, agendo sul pulsante numerico *Mass*, oppure potete proprio disattivare la simulazione fisica cliccando sul pulsante *No*.

Con queste impostazioni, oltre ai frammenti di mesh verranno visualizzate anche le particelle che, tra l'altro, hanno di default un aspetto luminescente, *Halo*, che magari per certi effetti sarà pure utile, ma non sempre. Per non renderizzare le particelle, basterà cliccare sul pulsante *None* presente nella sezione *Render* della scheda del sistema particellare; in questo caso, vedremo l'oggetto disintegrarsi senza particelle "extra".



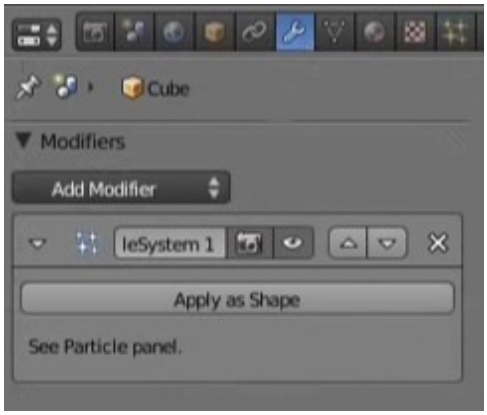
Alcune impostazioni del sistema particellare Emitter

Ad ogni modo, i frammenti della mesh si disperderanno nella direzione (e con la velocità) indicata per le particelle nelle sezioni *Velocity* e *Rotatio*n, per cui è lì che andranno impostati questi aspetti; per proiettare i frammenti verso l'alto, ad esempio, impostate *Physics* su *Newtonian,* quindi impostate 20 o più come valore per *Emitter Object Z* ed osservate, lanciando una simulazione con *ALT A* in una finestra di navigazione 3D, cosa succede (nota: *Physics* dev'essere impostato a *Newton*); vi invito quindi a fare delle prove con le altre impostazioni per provare vari effetti.



Anteprima dell'effetto modificando Emitter Object Z, con Physics Newtonian

Ovviamente è possibile impostare altre opzioni per il sistema particellare e tali opzioni avranno delle ripercussioni su come verrà implementata l'esplosione con *Explode*, tuttavia una descrizione approfondita delle voci e delle impostazioni di *Emitter* non è possibile in questa sede; fortunatamente, i nomi dei vari pulsanti dicono chiaramente a cosa servono le varie voci.



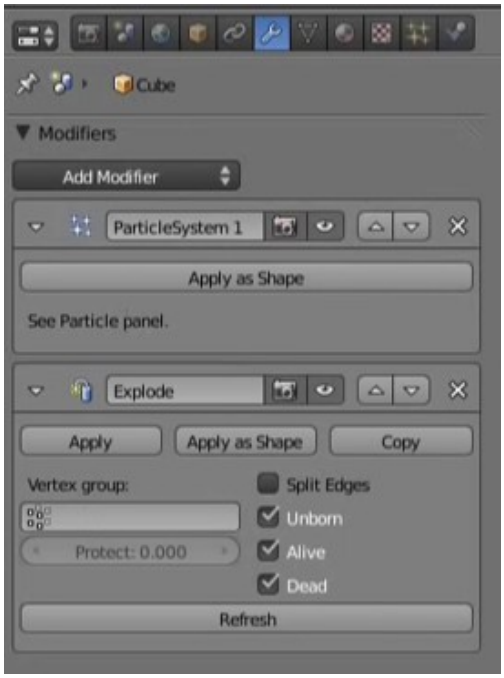
La pila dei modificatori della mesh, con il sistema particellare appena creato

Torniamo alla scheda dei modificatori attivi per la mesh. Dovrebbe essere presente, in cima alla pila di elementi, il sistema particellare appena creato.

Il sistema particellare dovrà trovarsi, nella scheda dei modificatori dell'oggetto, sopra al modificatore *Explode*; l'ordine è importante: creando prima *Explode* e poi il sistema particellare, l'esplosione della mesh NON verrà implementata.

Aggiungiamo quindi un modificatore *Explode*, che verrà posizionato sotto il sistema particellare, proprio come desideriamo. Esaminiamo le varie voci di questo strumento.

Generalmente, *Explode* viene applicato in maniera uniforme a tutte le porzioni di una mesh; impostando un Vertex Group per l'oggetto, ed assegnando vari pesi alle varie porzioni della mesh, invece, sarà possibile determinare in che misura il modificatore dovrà deformare la mesh.



La scheda del modificatore Explode

In particolare, le facce con peso massimo (cioè 1.0, colore rosso) NON verranno modificate da *Explode*, rimanendo intatte, mentre le facce con peso nullo (cioè 0.0, colore blu) verranno distrutte completamente; ovviamente, a valori intermedi tra 0.0 e 1.0 corrispondono applicazioni più o meno marcate dell'effetto.

Il nome del Vertex Group per la gestione di *Explode* andrà quindi scritto nel campo testuale posto sotto l'etichetta *Vertex Group*, appunto, nella scheda del modificatore.

Il campo numerico *Protect*, attivo solo in presenza di Vertex Group, serve a definire il "coefficiente di erosione" degli spigoli; il valore varia da 0.0, con erosione nulla, a 1.0, con erosione massima.

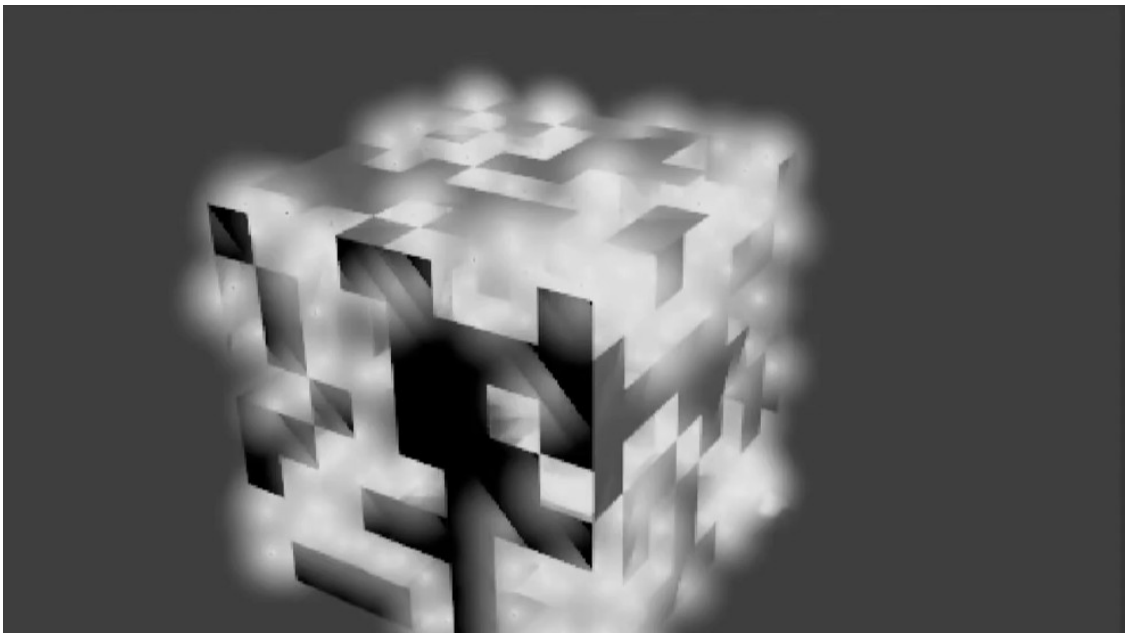
Split Edges significa letteralmente "spezza gli spigoli" ed è proprio quello che fa questa opzione: attivandola, le facce verranno suddivise più e più volte, portando ad un maggior numero di frammenti e, generalmente, ad un risultato migliore.

Unborn consente di rendere visibili o nascondere le facce della mesh quando ancora non sono state create le particelle; deselezionando questa opzione, le facce NON verranno visualizzate; in maniera analoga, *Alive* consente di rendere visibili o nascondere le facce della mesh quando le particelle sono "in vita", attive, mentre *Dead* si occupa di rendere visibili o nascondere le facce quando le particelle ad esse associate sono morte.

Refresh serve semplicemente a forzare l'aggiornamento dell'immagine della mesh a video.

Si tratta di un'operazione necessaria quando, dopo aver eseguito una simulazione di *Explode*, si modificano porzioni della mesh (ad esempio in Edit Mode), o qualche impostazione del sistema particellare; in questi casi, Blender potrebbe non aggiornare "al volo" l'effetto di *Explode* e potrebbe essere necessario, quindi, cliccare su *Refresh*.

È chiaro quindi che tutta la "difficoltà" nell'utilizzare *Explode* sta nel sapere che bisogna creare, prima, un sistema particellare per questo modificatore, e nell'impostare in seguito le varie voci di *Emitter* per ottenere l'effetto voluto.



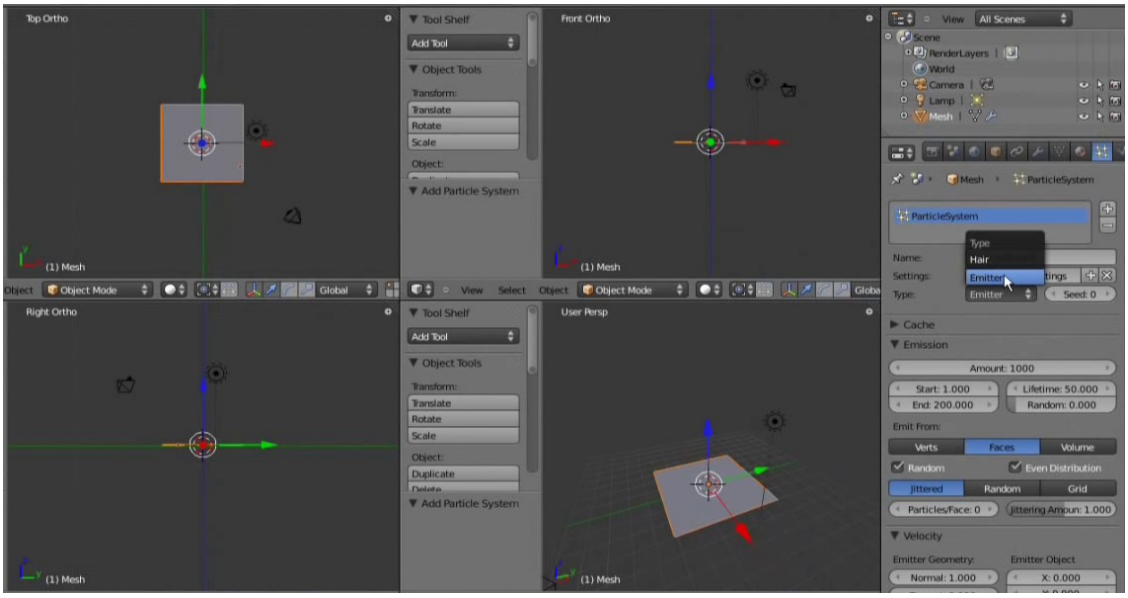
Screenshot dell'effetto finale (fotogramma dell'animazione)

* * *

Particelle, Force Fields, collisioni e rimbalzi

In questo tutorial vedremo come far interagire le particelle emesse da un sistema particellare *Emitter* con campi di forza e deflettori, per simulare collisioni e rimbalzi, in Blender 2.5.

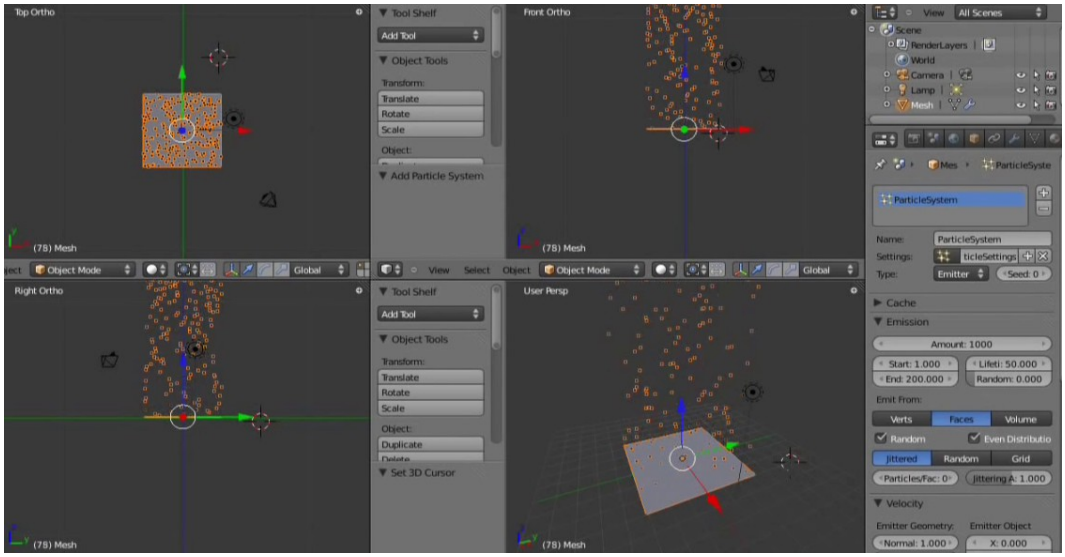
Avviato Blender, selezioniamo un oggetto che dovrà essere l'emettitore di particelle, ad esempio un piano rivolto verso l'alto, quindi apriamo la scheda *Particles* nella *Properties Window* ed aggiungiamo un sistema particellare, che di default sarà proprio un *Emitter* per l'oggetto.



Il Plane con il sistema particellare Emitter

In altri tutorial ho descritto il sistema particellare *Emitter* e qualche esempio di utilizzo, mostrando come impostare le varie voci di tale oggetto per ottenere alcuni effetti, per cui in questa sede non mi dilungherò molto su *Emitter* in sé, anche perché ci vorrebbe troppo tempo e perché utilizzeremo, comunque, un *Emitter* di base, per mostrare le interazioni con i campi di forza e i deflettori.

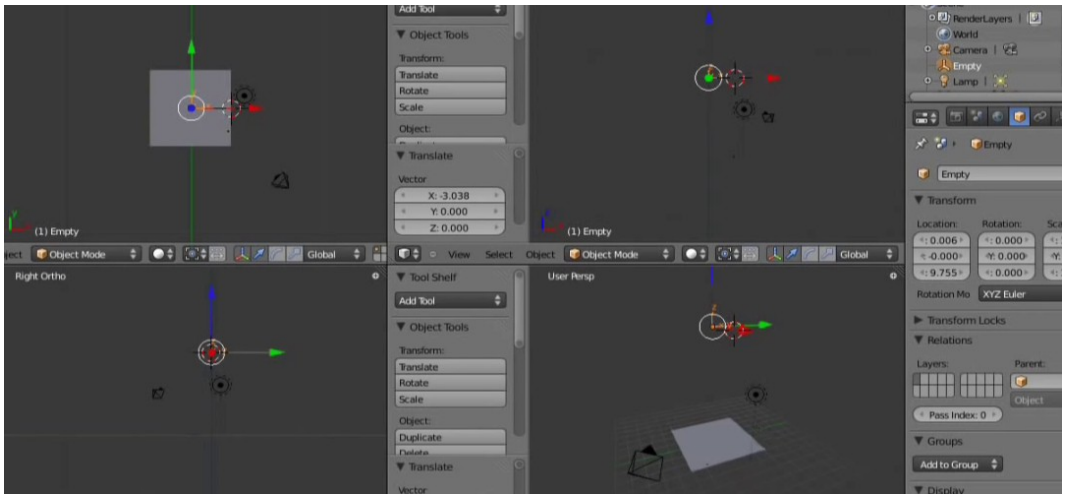
Impostando, nella scheda *Velocity* del pannello di *Emitter*, un valore come 10 o 15 per *Emitter Object Z*, in modo da proiettare le particelle verso l'alto, con il focus del mouse in una finestra di navigazione 3D lanciamo un'anteprima dell'animazione con *ALT A* (anteprima che possiamo bloccare in qualsiasi momento sempre con *ALT A*); dovremmo poter vedere le particelle uscire dal Plane, andare verso l'alto per un po' e poi ricadere a terra.



Screenshot dell'effetto ottenuto a questo punto

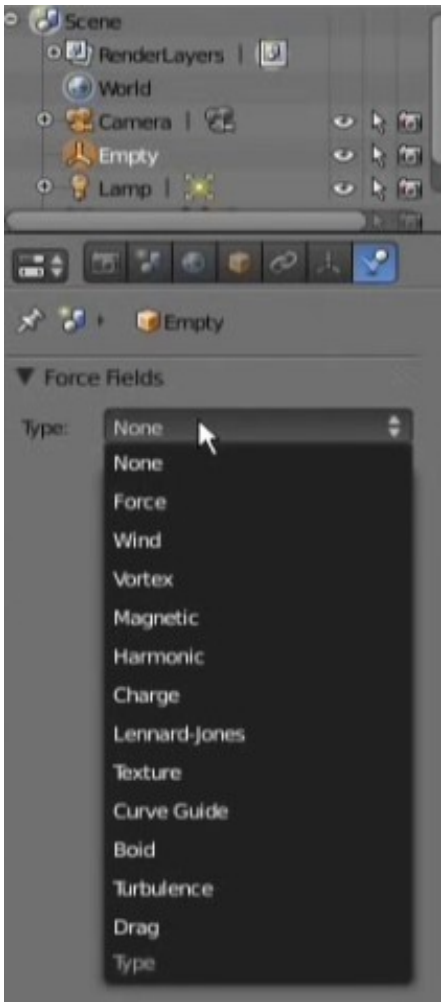
Impostiamo *Lifetime* delle particelle (ossia la durata della loro "vita") a 200 o più, per poter osservare i vari effetti che prenderemo in esame sul lungo termine.

A questo punto aggiungiamo un oggetto, anche una *Empty*, al quale assegneremo un campo di forza, ossia un campo, una zona di spazio, in grado di influenzare le particelle, la loro traiettoria, velocità, eccetera; nel mio caso, sto aggiungendo una *Empty* posizionata proprio sopra il piano ma, come detto, può trattarsi di un qualsiasi oggetto.



La Empty inserita nella scena

Con tale *Empty* selezionata, apriamo la scheda *Physics*, l'ultima della Properties Window.
Qui è facile individuare la scheda con etichetta *Force Fields*, ossia appunto campi di forza; in questa scheda è presente un'etichetta *Type* con accanto un menù a scomparsa, con voce di default *None*, cioè "niente". Cliccando su tale menù, compariranno diverse voci; di queste, noi esamineremo le più interessanti, o comunque quelle utilizzate più frequentemente.



I "tipi" di campi di forza realizzabili

- *TEXTURE* genera delle forze direzionali, nel senso di forze uscenti dalle facce dell'oggetto, con l'intensità di ciascun vettore-forza che dipende dal valore della Texture per ciascuna faccia dell'oggetto; in altre parole, se associamo una Texture all'oggetto, le varie facce avranno un aspetto diverso, con valori RGB differenti, ed il valore del vettore-forza uscente da ogni faccia dipenderà proprio dal valore della Texture per quella faccia. Questa opzione è utile se volete creare degli oggetti, specialmente delle griglie, che esercitino delle forze di intensità variabile per "zone" diverse della mesh; in questo caso, potrete creare delle Texture colorate ad hoc (che non dovranno essere applicate necessariamente ad un canale di mappatura in *Influenc*e, anzi potrete anche disattivarli tutti) da mappare sull'oggetto e selezionare quindi Texture come tipo di *Force Field*, passando a tale modificatore il nome della Texture creata.

In genere, per i *Force Field* degli altri tipi (come *Harmonic* o *Magnetic*) è disponibile anche un menù *Shape* per definire la "forma" del campo di forza, ad esempio se puntiforme o se planare; le varie voci hanno, comunque, significati intuitivi per tutti i casi.

- *HARMONIC* è un po' particolare: la posizione dell'oggetto *Force Field* diventerà il centro di un "ciclo", nel senso che le particelle emesse andranno verso tale punto, superandolo, poi

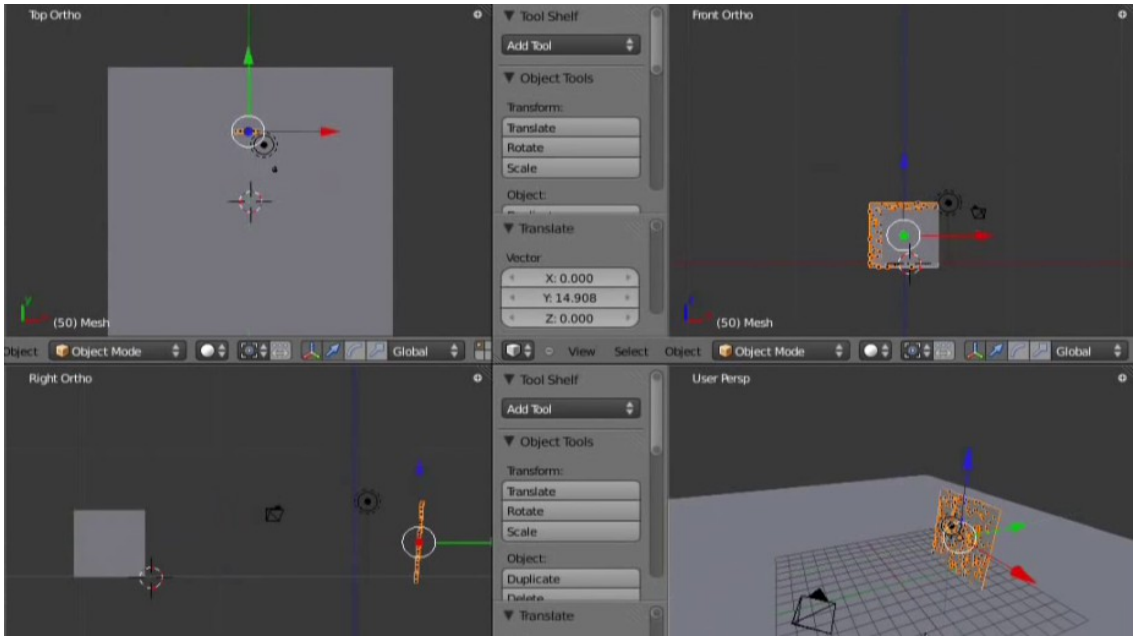
torneranno indietro, doppiando il punto, o almeno questo è quello che dovrebbe succedere *in teoria*... in pratica, siccome le particelle sono soggette, di default, alla forza di gravità, il moto complessivo non sarà proprio perfetto come quello appena descritto, ma una combinazione di più elementi, per cui per provare vari effetti occorre cambiare certe impostazioni del sistema particellare *Emitter*.

- *MAGNETIC* è, intuitivamente, una forza "magnetica", che attrae le particelle; dal momento che, proprio come con *Harmonic*, le particelle sono soggette anche ad altre forze, come ad esempio la gravità, il risultato finale sarà una combinazione di più elementi. Con più *Force Field* di questo tipo è possibile ottenere flussi di particelle davvero particolari, utili non solo per simulazioni realistiche ma anche per composizioni per così dire "astratte". Un generale, tutti i *Force Field* possono essere utilizzati per ottenere risultati "artistici" o comunque particolari, non realistici.

- *VORTEX* farà ruotare le particelle intorno ad un asse dell'oggetto al quale è associato tale *Force Field*, in modo da simulare, appunto, un vortice, un tornado, una tromba d'aria e così via. Si può capire qual è l'asse di rotazione osservando, in una finestra di navigazione 3D, il disegno che verrà associato alla mesh con tale *Force Field*: un disegno stilizzato di una ventola, dove ovviamente l'asse di rotazione è quello perpendicolare al piano del disegno. Nella scheda sono poi presenti delle voci per impostare, in maniera intuitiva, alcuni parametri, come l'intensità o il rumore.

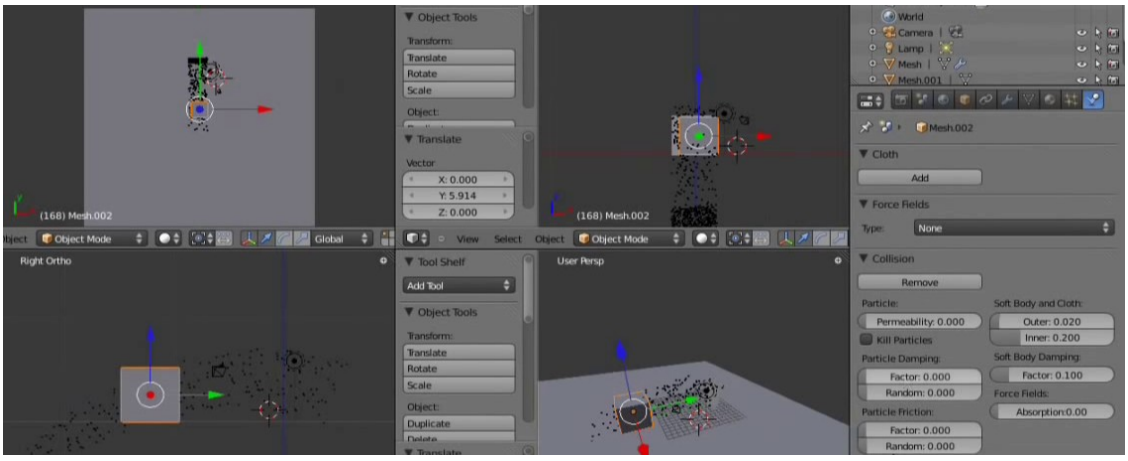
- *WIND* ("vento") è un campo di forza che spinge le particelle secondo una certa direzione, proprio come fa il vento con la polvere, la sabbia o qualsiasi altro elemento. L'effetto si apprezza maggiormente impostando un valore maggiore di 1 per *Strength*, l'intensità.

Questi sono i campi di forza principali. Passiamo ora all'argomento "deflettori", ossia: collisioni e rimbalzi di particelle ed oggetti della scena.
Eliminiamo la *Empty* utilizzata per provare i *Force Field* ed aggiungiamo un piano per il "pavimento" e una mesh, ad esempio un cubo, per provare le collisioni, posizionando tale oggetto sul piano; spostiamo inoltre il piano emettitore del sistema particellare, ponendolo di fronte al cubo ma ad una certa distanza, e ruotiamolo per renderlo perpendicolare al pavimento.



Creazione e posizionamento del Plane e del cubo

Il nostro *Emitter* aveva un valore maggiore di 0 per *Emitter Object Z*, per cui (in questa nuova configurazione) proietterà le particelle verso il cubo. Selezioniamo il cubo e, nella scheda *Physics* della Properties Window di tale oggetto, clicchiamo su *Add* sotto l'etichetta *Collision*, per far sì che le particelle rimbalzino in caso di collisione con tale oggetto.



Il cubo è ora un oggetto Collision

Abbiamo diverse voci in tale scheda, ma possiamo riassumerle come segue, trascurando comunque quelle che riguardano la simulazione dei tessuti:

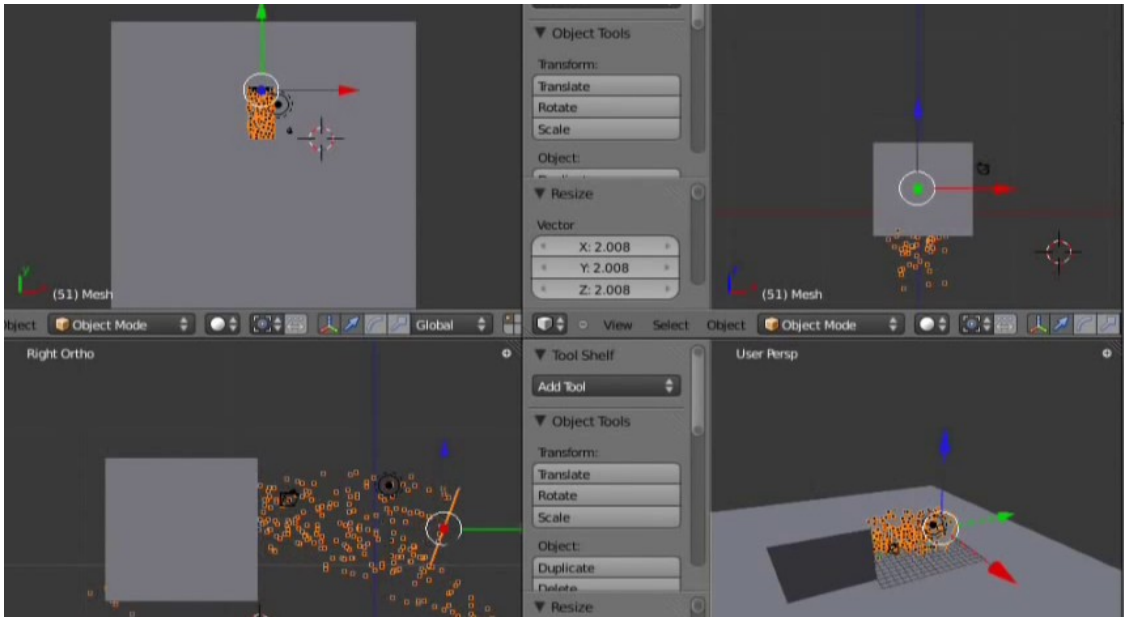
- la *Permeabilità* indica, come suggerisce il nome, la "permeabilità" dell'oggetto, che si traduce qui nella probabilità, da 0 a 1, che una particella riesca ad attraversare la mesh; il valore di default è 0, cioè probabilità nulla, per cui tutte le particelle rimbalzeranno, mentre con probabilità 1 tutte le particelle riusciranno a passare attraverso la mesh (a valori intermedi corrisponderanno, ovviamente, risultati diversi).

- *Kill Particles*, se attivo, distruggerà le particelle nel caso queste dovessero colpire l'oggetto, ad eccezione di quelle che (con Permeabilità maggiore di 0) possono attraversare lo stesso. Con *Permeabilità* a 0 e *Kill Particles* attivo, tutte le particelle verranno distrutte, o meglio scompariranno, in seguito ad un impatto con la mesh.

- *Damping* rappresenta la quantità di energia persa dalle particelle in seguito ad un impatto; questo si traduce in rimbalzi di ampiezza differente, come vedremo;

- *Friction* rappresenta l'attrito, cosa che si traduce, in pratica, in rimbalzi più o meno lunghi.

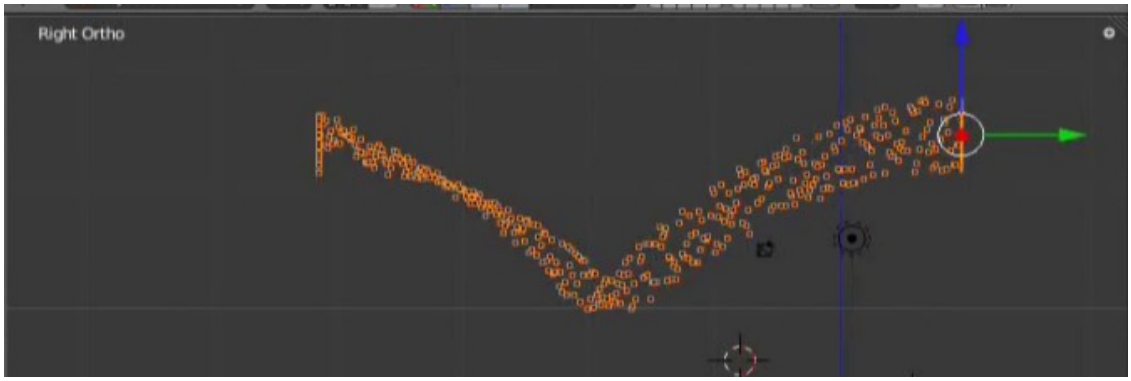
Per i campi *Damping* e *Friction* sono presenti anche dei valori *Random*, cioè casualità, per simulare una superficie non omogenea ed aggiungere quindi un po' di casualità nella lunghezza e nell'altezza dei rimbalzi.



Screenshot dei rimbalzi con il cubo

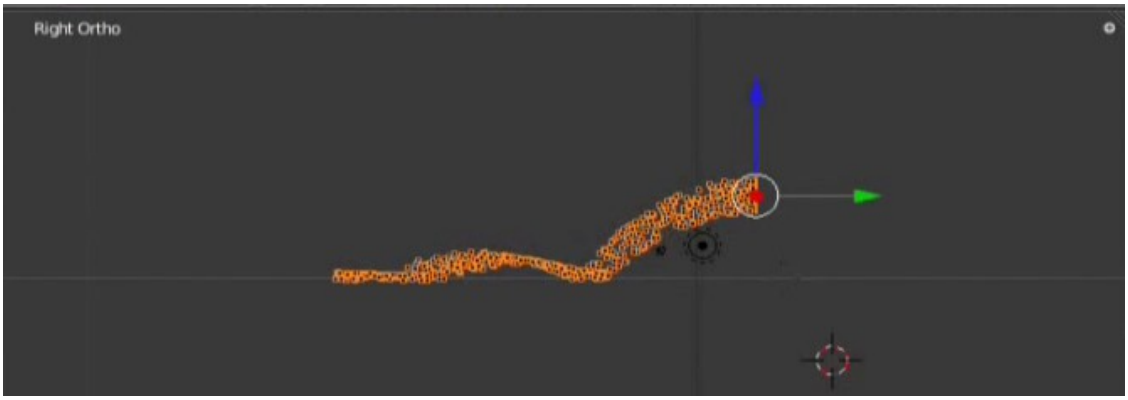
Togliamo il cubo dalla scena e selezioniamo il pavimento per provare *Damping* e *Friction*, per cui aggiungiamo anche per tale piano un simulatore *Collision*, come abbiamo fatto per il cubo.

Spostiamo in alto il piano emettitore delle particelle, in modo da far cadere le stesse sul pavimento e valutare i rimbalzi.
Osserviamo i rimbalzi ottenuti con le impostazioni di default, cioè 0 per *Damping* e per *Friction* del pavimento, premendo *ALT A* mentre il focus del mouse è in una finestra di navigazione 3D.



Screenshot dei rimbalzi con il piano

Selezioniamo il pavimento ed impostiamo *Friction* a 0.8, visualizzando un'anteprima dell'animazione, per valutare gli effetti dell'attrito. Riportiamo *Friction* a 0 per evitare di fare confusione e portiamo *Damping* a 0.5, per valutare gli effetti dovuti solo a questo parametro, lanciando quindi un'anteprima dell'animazione; ovviamente, sarà possibile combinare i vari effetti, impostando valori diversi per i rispettivi parametri... magari dopo averli variati e provati uno alla volta, all'inizio, per valutarli singolarmente.

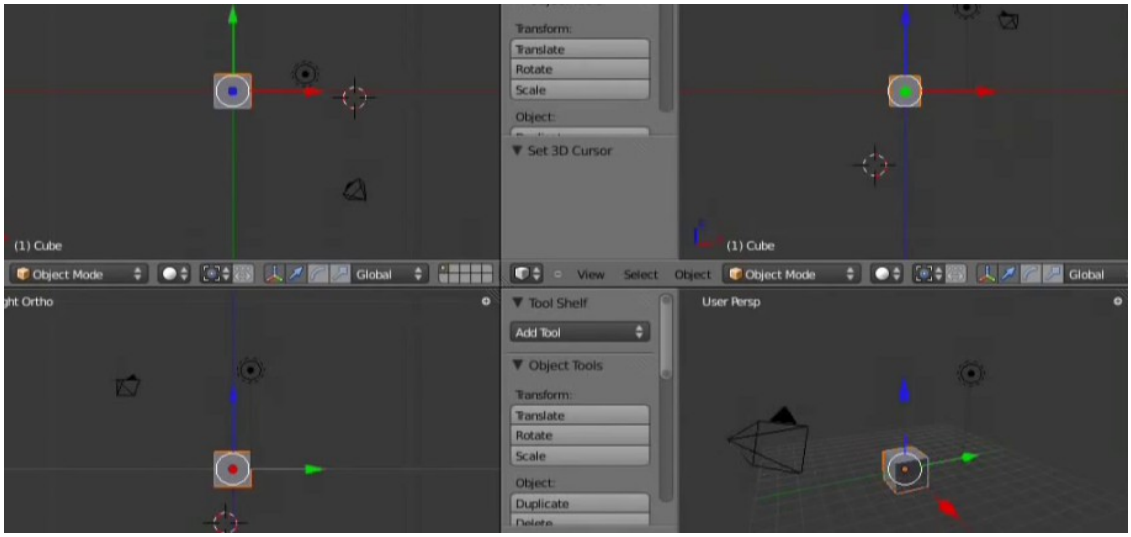


Altro esempio di effetto, variando Friction e Damping

* * *

Cloth (tessuti) con vento e collisioni per tende e altro

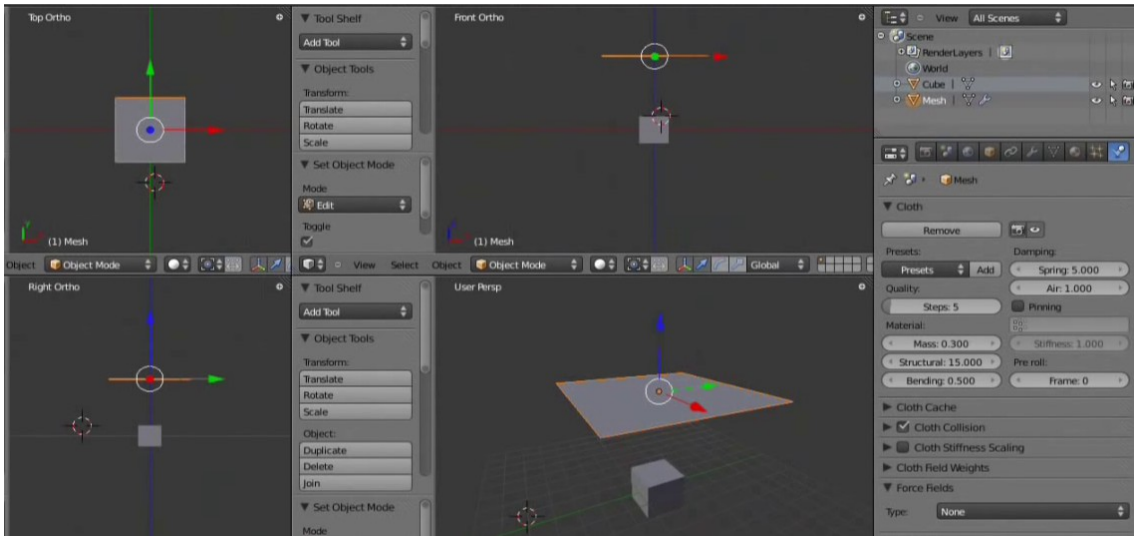
In questo tutorial vedremo le basi della simulazione *Cloth*, i tessuti, in Blender 2.5, con qualche trucchetto per realizzare simulazioni di vele, bandiere, tende mosse dal vento e così via.
Avviamo Blender. Come potete vedere, al centro della scena, nel mio caso, c'è un cubo.



La scena di partenza

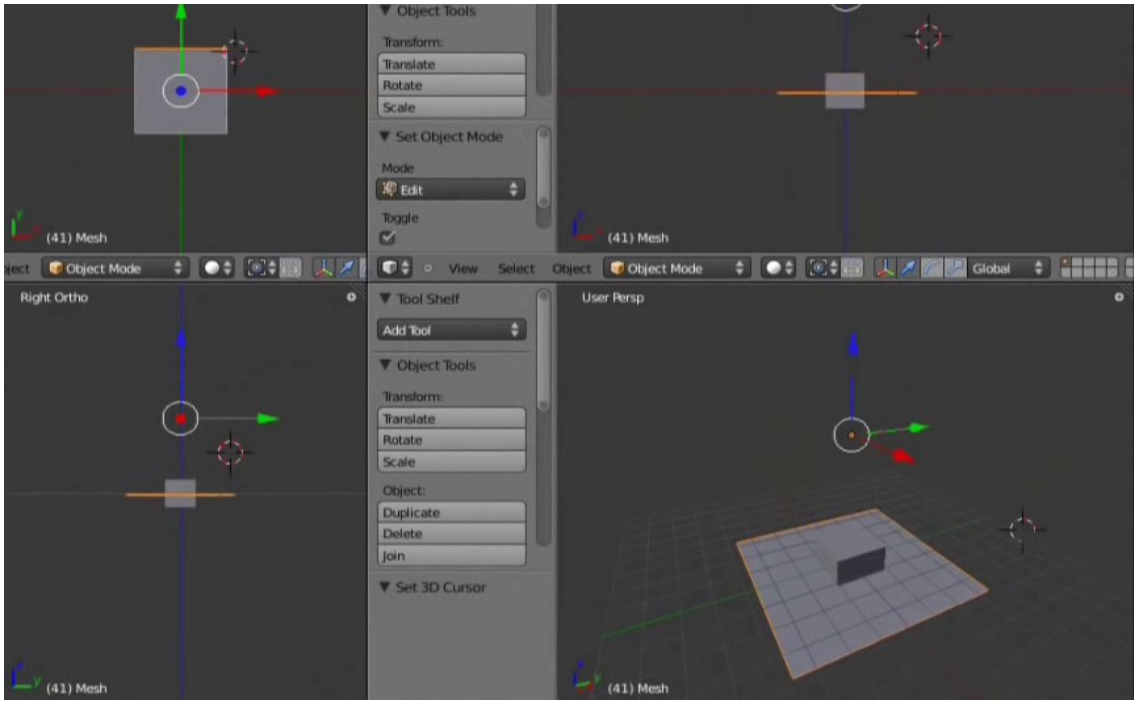
Sopra tale oggetto, ma ad una certa distanza, non proprio "poggiato", aggiungiamo un Plane, più grande del cubo visto che, all'inizio, dovrà avvolgerlo.

Passiamo in Edit Mode e suddividiamo un bel po' di volte questo Plane; attenzione, perché ad un maggior numero di suddivisioni corrisponderà un risultato migliore, durante la simulazione del *Cloth*, ma anche una maggior quantità di tempo per realizzare l'effetto in fase di *Baking*, come vedremo, per cui magari all'inizio fate le prove con poche suddivisioni, poi aumentatele e rifate il *Baking*. Apriamo la scheda *Physics* per questo Plane e clicchiamo su *Add* nella sezione *Cloth*.



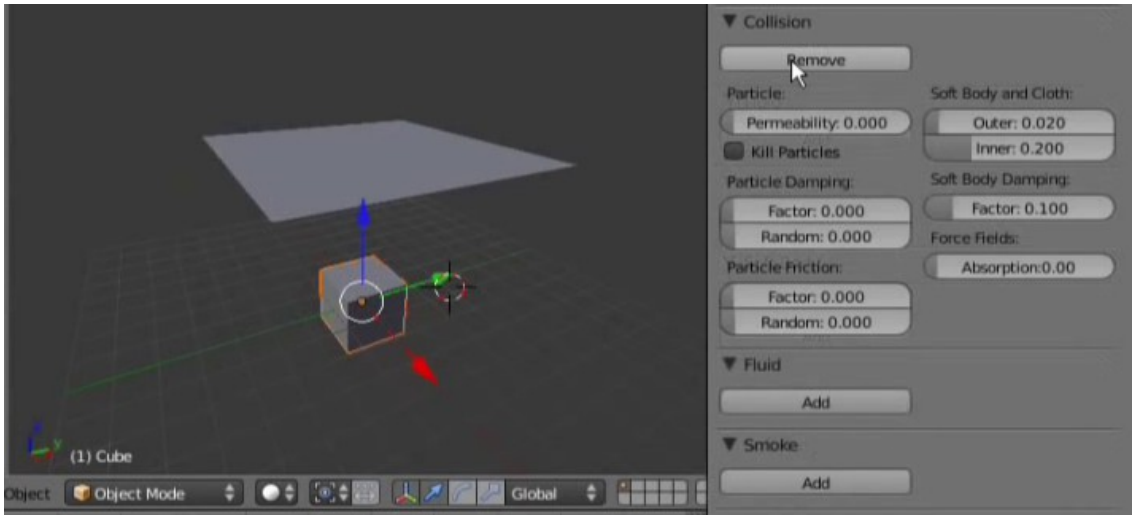
Il Plane, suddiviso, dotato di un sistema Physics Cloth

Già in questo modo abbiamo reso il Plane tessuto ed infatti premendo *ALT A* potremo vedere un'anteprima dell'animazione in una finestra di navigazione 3D; il Plane, però, precipita oltrepassando il cubo, ignorandolo.



Il Plane precipita nel vuoto oltrepassando il cubo

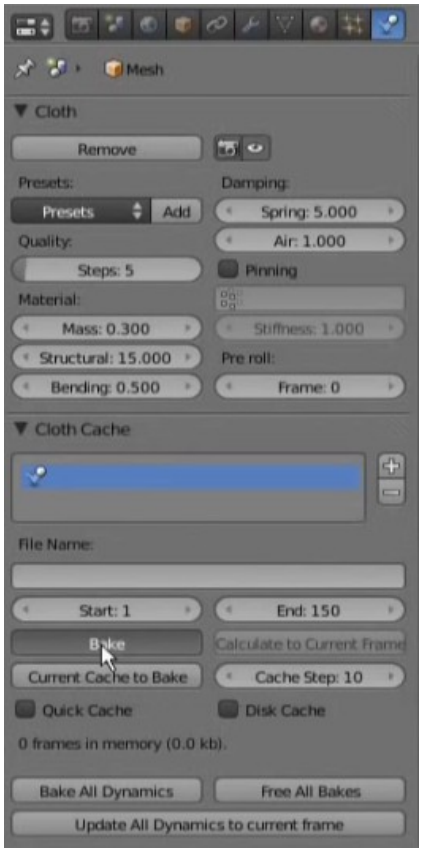
Per far interagire un oggetto con un tessuto, bisogna farlo partecipare alla simulazione delle collisioni, per cui selezioniamo il Cubo e, nella scheda Physics, clicchiamo su *Add* alla voce *Collision*.



Aggiungere la simulazione Physics Collision al cubo

Qui possiamo impostare alcuni parametri riguardanti le collisioni, come la permeabilità o altri effetti di interazione con le particelle, ma a questo argomento ho già dedicato un tutorial ad hoc, inoltre qui dobbiamo parlare dei tessuti, per cui lasciamo le impostazioni di default (che, tra l'altro, vanno più che bene) senza dilungarci troppo sulle impostazioni di *Collision*.

Selezioniamo nuovamente il Plane, ma questa volta effettuiamo un *Baking*, cioè proprio la simulazione fisica, per vedere l'interazione del *Cloth* con altri elementi; in realtà, qui non è davvero necessario ma ne approfitto per dirvi come si fa il *Baking*. Nella finestra *Physics* apriamo la scheda *Cloth Cache*, cioè la "memoria" delle simulazioni di tipo tessuto che riguardano questo oggetto, e clicchiamo su *Bake*.

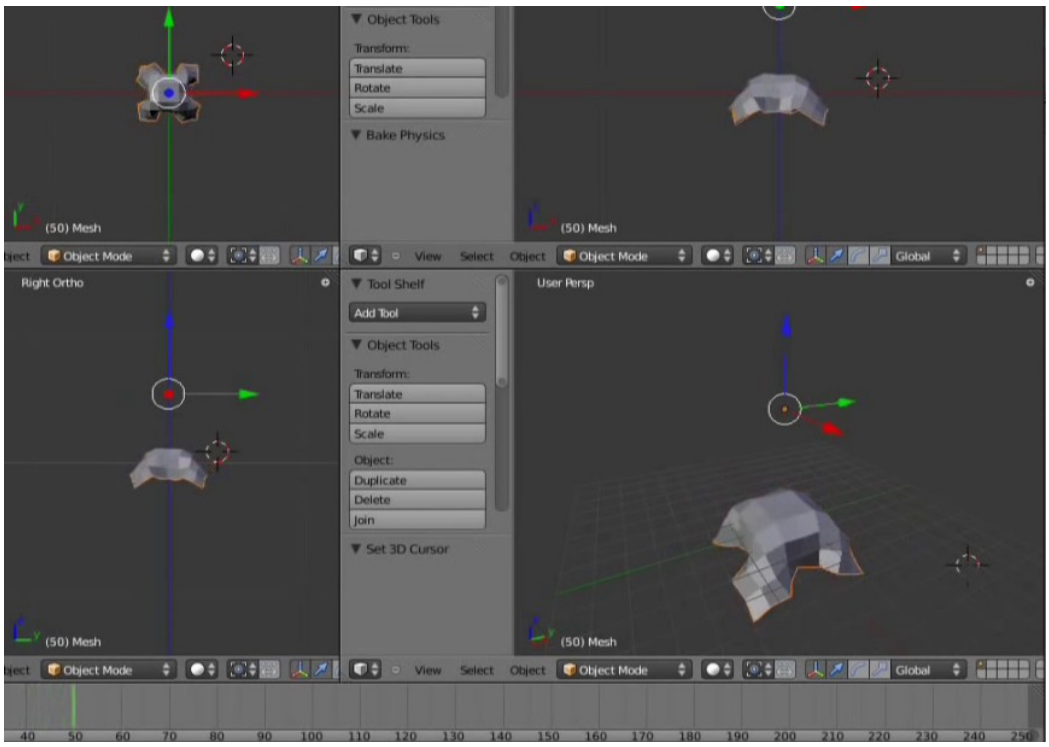


Effettuare il Baking

Quando dovremo fare nuove simulazioni, ad esempio perché abbiamo aumentato le suddivisioni o inserito altri elementi *Collision* o campi di forza, dovremo pulire la memoria cliccando su *Free Bake*, che nel frattempo è apparso al posto di *Bake*, e cliccare nuovamente su *Bake* per avviare una nuova simulazione aggiornata.

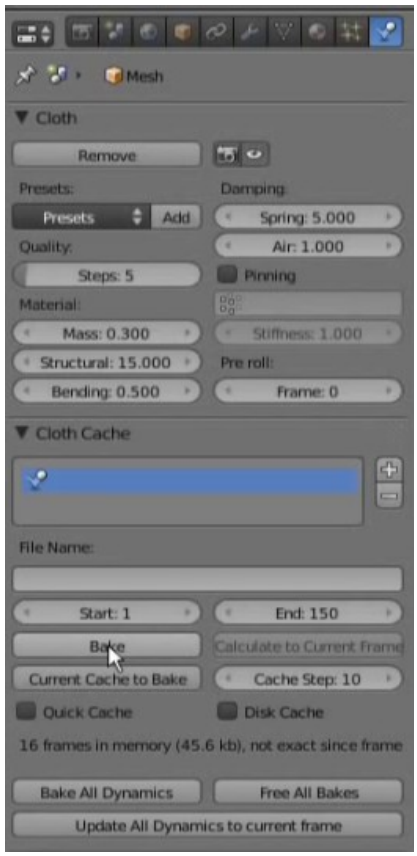
Bene, già così sappiamo fare qualcosina, come possiamo verificare (al termine del baking) premendo *ALT A*, per avviare

un'anteprima dell'animazione in una finestra 3D View.



L'animazione ottenuta mediante Baking (fotogramma dell'anteprima, lanciata con ALT A in una 3D View)

Prima di vedere altri trucchetti, come il *pinning* e l'interazione con i campi di forza, esaminiamo le voci della scheda principale di *Cloth*, per capire a cosa servono.

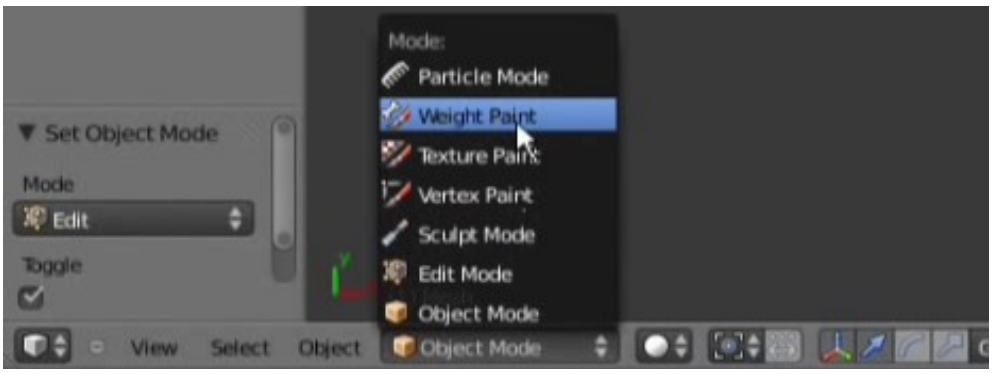


Le voci della scheda CLOTH

- *Presets* contiene, ovviamente, un elenco dei materiali per i quali esistono già dei valori prefissati, come ad esempio *Cotton*, *Denim* (cioè jeans), *Leather* (ossia cuio, pelle), *Rubber* (letteralmente gomma) e *Silk*, la seta. Ovviamente, a materiali diversi corrisponderanno comportamenti diversi, il più possibile realistici. Con *Rubber* e *Denim* è possibile utilizzare mesh con un basso numero di suddivisioni, mentre per *Silk*, la seta, si consiglia un numero di suddivisioni alto (anche se ciò implica più tempo per realizzare la simulazione, ne vale la pena). Possiamo aggiungerne di personalizzati variando i parametri e cliccando su *Add*.

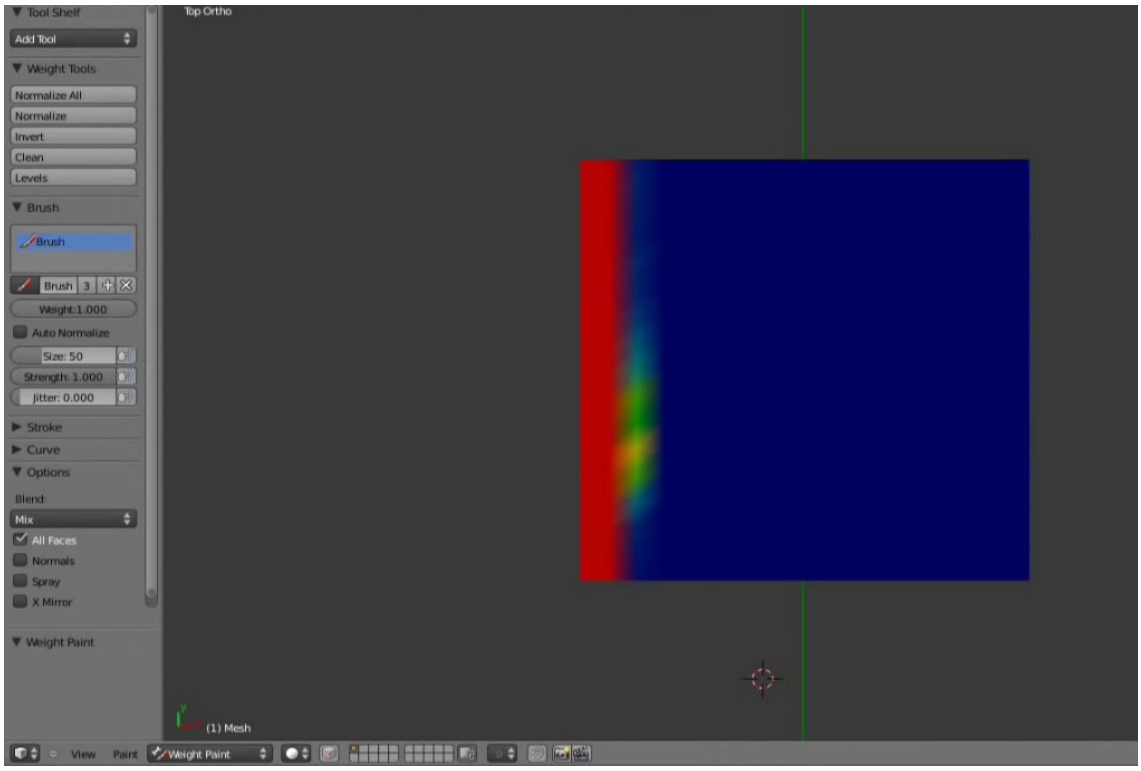
- Le voci di *Damping*, ossia *Spring* ed *Air*, si riferiscono in un certo senso all'attrito: *Spring* indica una "resistenza interna", molto utile perché consente di evitare dei moti ondulatori del materiale poco realistici e indesiderati, mentre *Air Damping* indica la resistenza dell'aria.
- Il campo numerico *Steps* di *Quality* si riferisce ovviamente alla qualità della simulazione, nel senso di numero di esecuzioni dell'algoritmo per frame; di default, si hanno 5 steps, cioè 5 passi di esecuzione, per frame, ed è un valore che in genere va più che bene; a valori più alti corrisponderanno risultati migliori, specialmente in presenza di collisioni, ma la simulazione richiederà più tempo.
- Nella sezione *Material* abbiamo le voci *Mass*, *Structural* e *Bending*. *Mass* è, chiaramente, la massa dell'oggetto, mentre *Structural Stiffness* è un indice della rigidità dello stesso. *Bending* è un valore che, se alto, implica la presenza di un minor numero di grinze piccole, ma un maggior numero di pieghe grandi.
- Parleremo in seguito del *Pinning* e dei campi *Vertex Group* e *Stiffness*, per cui per il momento andiamo oltre.
- *Pre-roll*, come suggerisce il tooltip, consente di avviare la simulazione ad un dato frame.

È giunto il momento di parlare del *pinning*, una tecnica utilissima per simulare ad esempio le tende o le vele, in quanto consente di vincolare certe porzioni di un oggetto *Cloth* mediante il meccanismo dei "pesi" applicati ai vertici dell'oggetto; in pratica, i gruppi di vertici con peso massimo rimarranno fermi (non parteciperanno alla simulazione *Cloth*), mentre quelli con peso nullo la subiranno del tutto; i vertici con pesi tra 0 e 1 verranno influenzati, ovviamente, in maniera proporzionale al loro peso. Selezioniamo quindi il nostro oggetto *Cloth* e passiamo, in una finestra di navigazione 3D, in modalità *Weight Paint*.



Passaggio in modalità Weight Paint per l'oggetto con simulazione CLOTH

Sulla sinistra dovremmo avere gli strumenti per impostare, tra le altre cose, le dimensioni del pennello (e quindi, indirettamente, la quantità di vertici che possiamo influenzare in un colpo solo), il peso (*weight*) da applicare e l'intensità con la quale vogliamo applicarlo, ossia se sono necessarie più passate o una sola per applicare un certo peso. Applichiamo il peso massimo, 1, ad una striscia orizzontale di vertici, quelli che in una tenda corrispondono al supporto, colorandoli di rosso. Con questa semplice operazione, abbiamo vincolato questi vertici ad una posizione fissa.



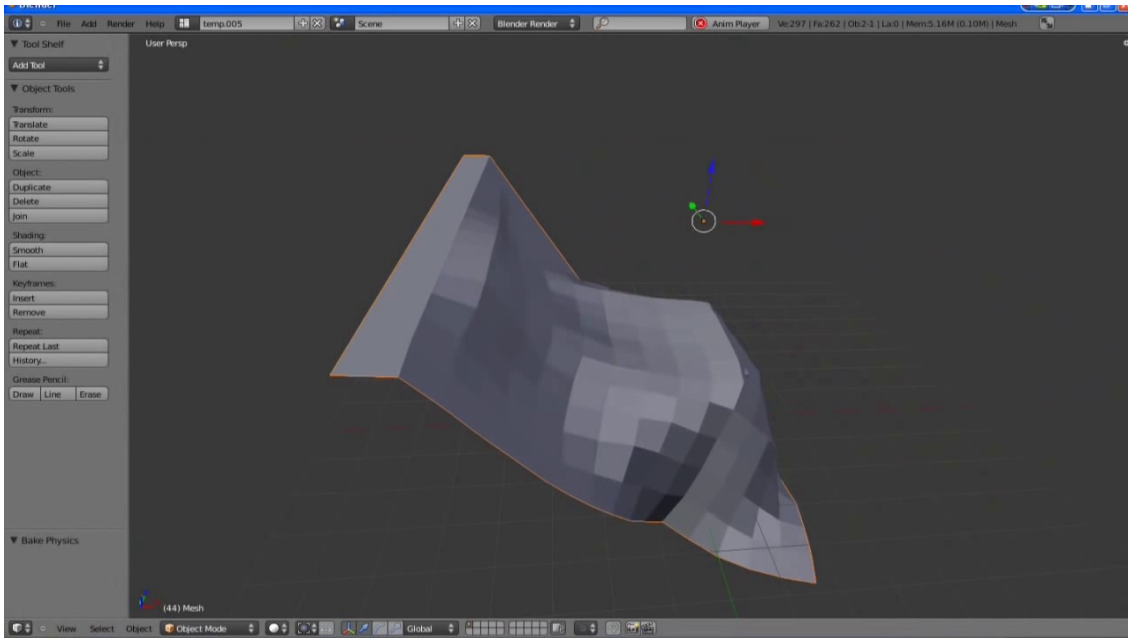
"Colorare" alcuni vertici con peso 1 (massimo), per bloccarli col Pinning

Prima di effettuare il *Baking*, però, dobbiamo attivare l'opzione *pinning* per il tessuto, per cui selezioniamo, nella scheda *Cloth* della mesh, la voce *Pinning*.

Dobbiamo indicare il Vertex Group di riferimento nell'apposita casella sotto *Pinning*, comunque Blender ha creato automaticamente il Vertex Group durante la fase di *Weight Painting*, per cui non dobbiamo fare altro che cliccare sulla casella e selezionare, dal menù che apparirà, l'unica voce presente.

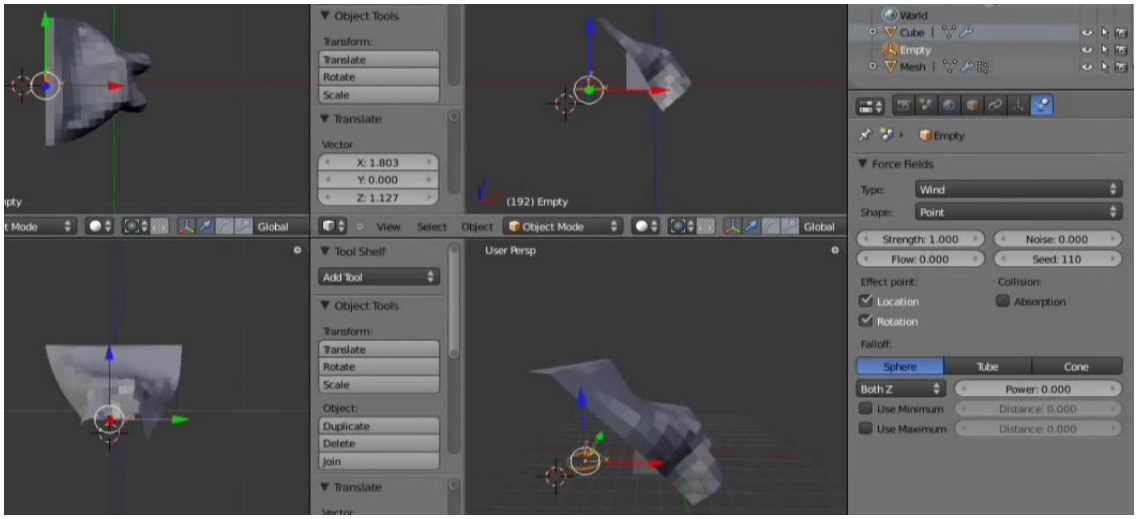
Attivare il Pinning e specificare il VG (creato automaticamente) dei pesi impostati mediante Weight Paint

Effettuiamo quindi il *baking* della simulazione e lanciamo un'anteprima dell'animazione in una finestra di navigazione 3D con *ALT A* per osservare l'effetto ottenuto.



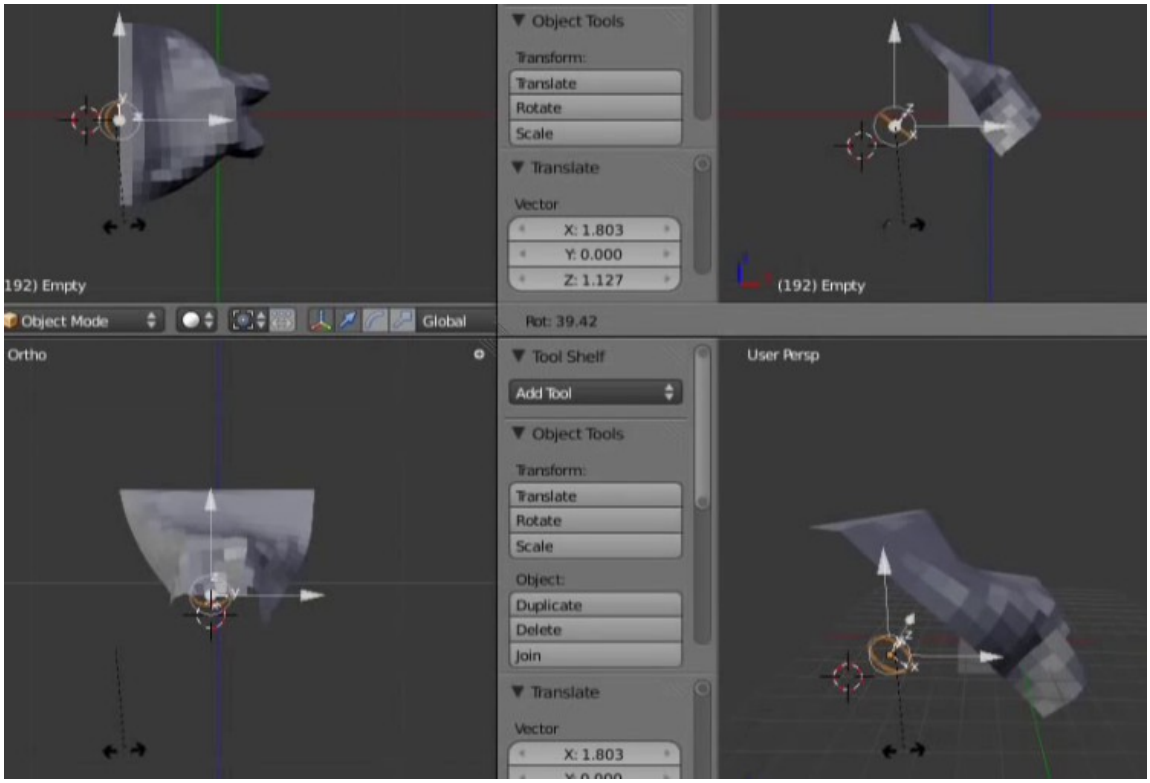
Screenshot del risultato ottenuto fino a questo punto

Vediamo ora come far interagire un oggetto tessuto con un campo di forza come *Wind*, il vento. Aggiungiamo un oggetto *Empty* alla scena, andiamo nel pannello *Physics* della stessa e dal menù *Force Field*, ossia campo di forza, scegliamo *Wind*, il vento.



Il Force Field Wind per la Empty appena creata

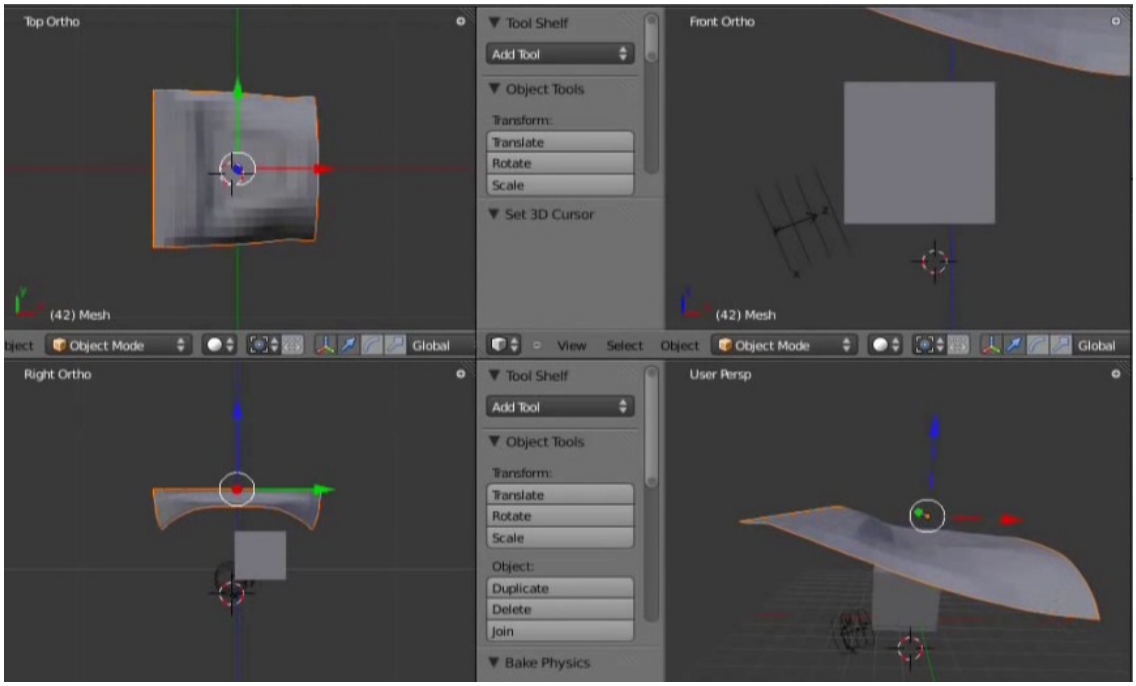
Ai *Force Field* ho già dedicato un tutorial (lo stesso che riguardava le collisioni con *Collision*), per cui qui mi limiterò a dire che questo oggetto simula l'azione del vento; la direzione del vento è visibile osservando, in una finestra di navigazione 3D, la nuova forma assunta dalla *Empty*, per cui orientiamo il campo di forza verso il tessuto, per influenzarlo.



Orientare "il vento" verso il tessuto (CLOTH)

Aumentiamo il valore di *Strength*, l'intensità del vento, portandolo ad esempio a 9 o 10.

Adesso possiamo effettuare nuovamente il *Baking* della simulazione del tessuto (dopo averne selezionato la mesh, ovviamente) e, se abbiamo posizionato il campo di forza in maniera interessante, osservare l'effetto del vento sul tessuto.



Screenshot del risultato ottenuto dopo aver effettuato il Baking con Pinning e Wind

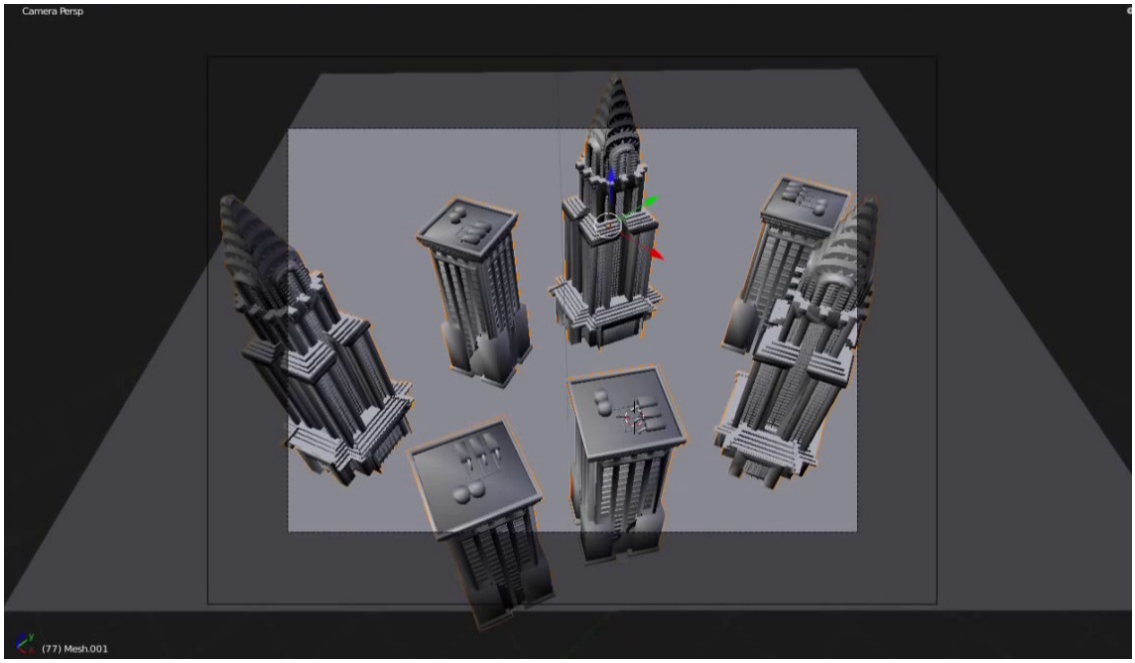
Con gli argomenti trattati in questo tutorial, dovreste conoscere tutto ciò che serve per realizzare delle interessanti simulazioni con i tessuti; ovviamente l'argomento non si esaurisce qui: questa è solo un'introduzione, ma è pur sempre un primo passo.

* * *

Effetti speciali - Ologramma 3D

In questo breve tutorial vedremo come creare un effetto “ologramma tridimensionale” per le mesh di una scena. Questo effetto potrà essere utilizzato, poi, per videoclip realizzati completamente in digitale o per aggiungere un effetto speciale in filmati con scene riprese dal vivo, mediante video compositing o post-produzione.

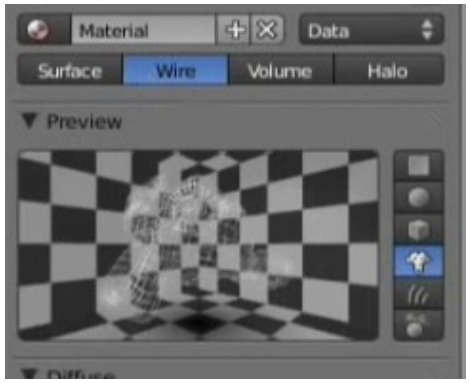
Nella scena visibile nell'immagine qui sotto vi sono delle mesh che riproducono dei grattacieli.
Il tutto sta nel conferire un aspetto particolare a tali oggetti, per cui ci dobbiamo concentrare sul Materiale da utilizzare.



La scena che utilizzeremo per questo esempio

Nella scheda del Material associato a tali mesh, selezioniamo l'opzione *Wire*, ossia “fil di ferro”, reticolo; in questo modo, verranno renderizzati solo gli spigoli delle mesh (la struttura, non le facce o le superfici).

Sempre nella scheda del Material, impostiamo il colore che preferiamo per l'ombreggiatore diffuso: in genere il bianco andrà più che bene, ma potete cambiarlo tranquillamente per adattarlo all'atmosfera della mostra scena; possiamo lasciare le impostazioni di default per le schede *Specular* e *Shading*.



Modalità "Wire" per il Material

Attiviamo la trasparenza, di tipo *Z Transp*, portando il valore di *Alpha* a 0.

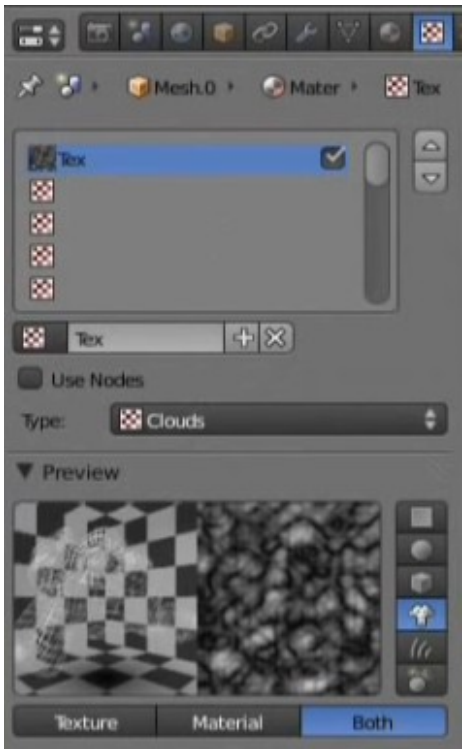
In *Options*, più in basso nella scheda del Material, disattiviamo la voce *Traceable*.

Per variare un po' l'opacità e la luminosità degli spigoli, comunque, dobbiamo ricorrere ad una Texture, per cui aggiungiamone una per questo Material.

Quella che descrivo qui è un'impostazione di base, potete provare con Textures differenti o con altri valori; nel mio caso, sto utilizzando una Texture di tipo *Clouds*, le nuvole, con i valori di default per il rumore, in modo da conferire un aspetto a chiazze agli spigoli.

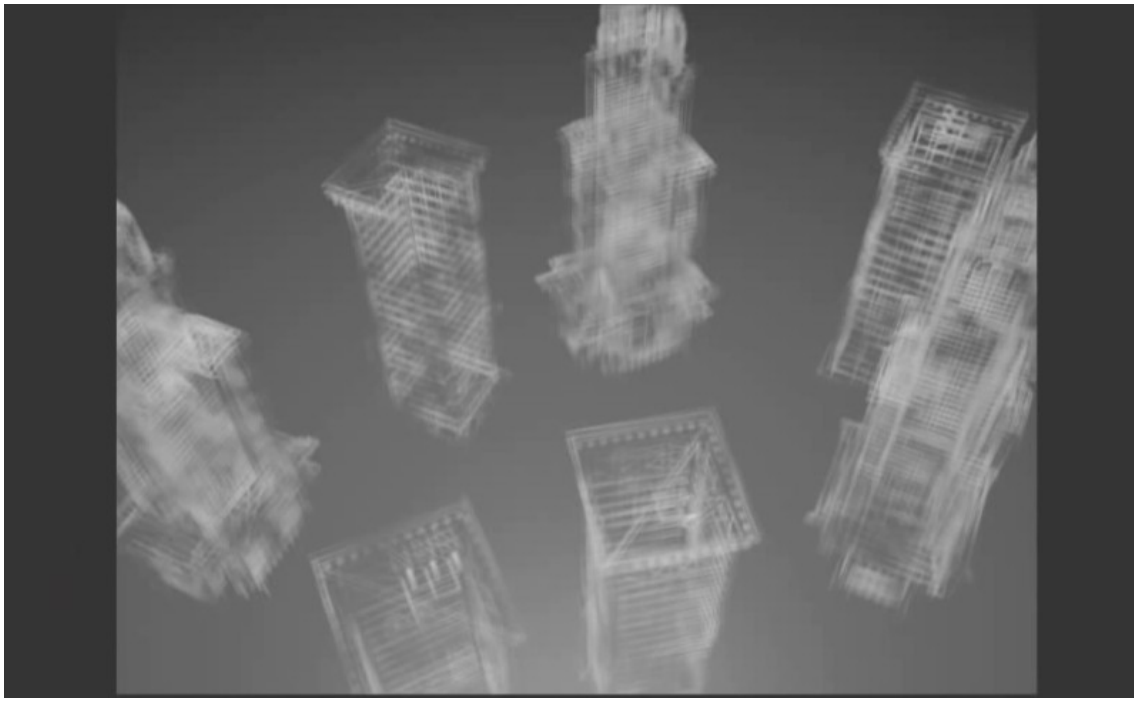
In effetti,trattandosi di spigoli, non di superfici, il tutto si tradurrà in porzioni di segmenti con luminosità e opacità differenti.

Disattiviamo, in *Influence*, il canale *Color*, e selezioniamo invece i canali *Alpha* ed *Emit*, portandoli ad esempio a 0,5 e 1 rispettivamente.



La Texture Clouds sui canali Alpha (trasparenza) ed Emit (emissione luminosa)

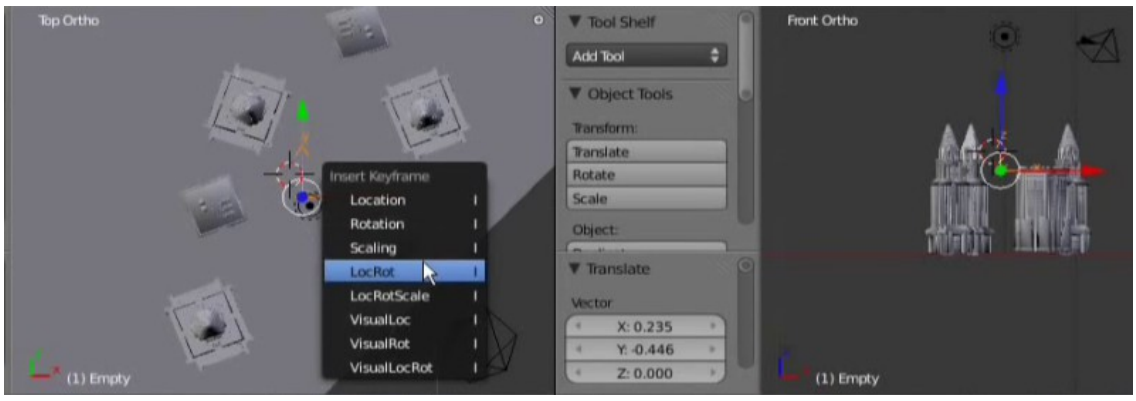
Già a questo punto è possibile lanciare un rendering per osservare il risultato ottenuto e decidere, magari, se cambiare o meno qualche impostazione, per rendere ad esempio gli oggetti più trasparenti o meno luminosi.



Il rendering del risultato ottenuto a questo punto

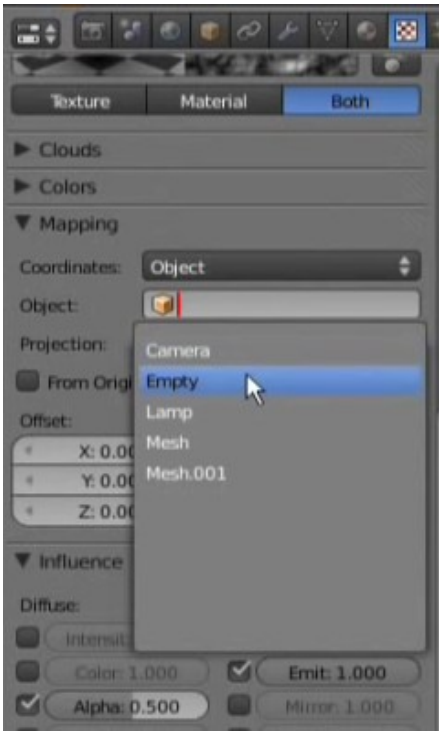
Se vogliamo realizzare delle animazioni o dei videoclip, possiamo animare anche la Texture e, quindi, il modo in cui verranno evidenziate certe porzioni dei vari segmenti, in modo da rendere il tutto meno "statico"; aggiungiamo quindi una *Empty* alla scena e posizioniamola, per comodità, al centro dell'ologramma.

Animiamo la *Empty* fornendole una rotazione intorno ad uno o a più assi, per cui inseriamo un keyframe *Rot* all'inizio dell'animazione, spostiamoci al frame finale, ruotiamo un po' la *Empty* ed inseriamo un secondo keyframe per il canale di rotazione.



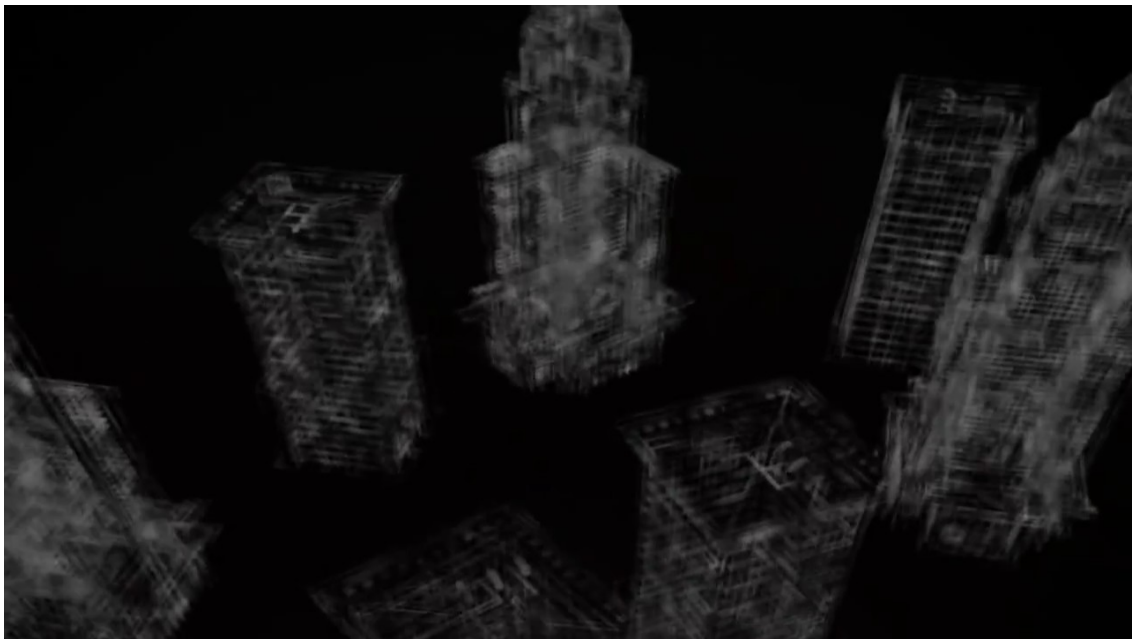
Inserimento dei Keyframe di rotazione per la Empty

Adesso dobbiamo legare la mappatura della Texture al movimento della *Empty*, per cui selezioniamo tale Texture e, nel campo *Mapping*, scegliamo per *Coordinates* la voce *Object*; nel campo testuale che apparirà sotto il menù, inseriamo il nome della nostra *Empty*.



Mapping Coordinates: Object - Empty, per la Texture

Tutto a questo punto è pronto per un'animazione. Dal momento che anche i valori di *Alpha*, *Emit* e, in generale, di tutti i campi dei Material e delle Textures possono cambiare nel tempo, è possibile animare anche questi aspetti, se lo si desidera, per arricchire ulteriormente l'effetto; potete sbizzarrirvi come preferite, queste sono solo le mie impostazioni di base.



Fotogramma dell'animazione finale (il colore di World è stato cambiato in nero)

Un altro effetto interessante consiste, ad esempio, nell'aggiungere una o più fonti di luce di tipo *Spot Light* con luce volumetrica, in modo da simulare i faretti che proiettano l'ologramma.

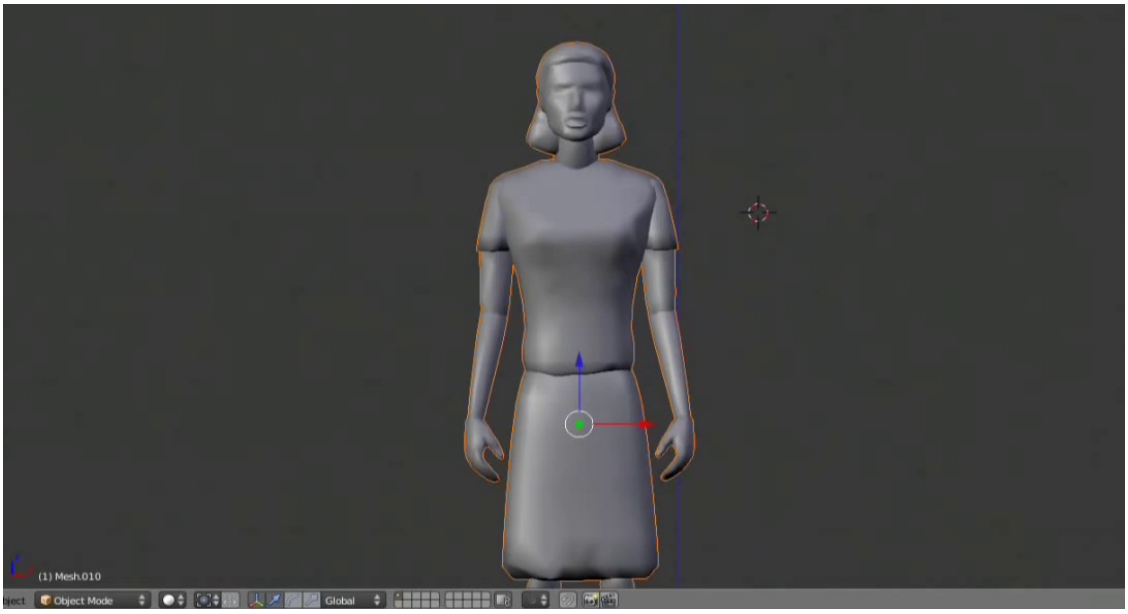
Creiamo innanzitutto una fonte di luce di tipo *Spot*, magari disattivando le ombre in *Shadow* per risparmiare tempo in fase di rendering; selezioniamo quindi, in *Spot Shape*, la voce *Halo*, per implementare la luce volumetrica, abbassando però il valore dell'intensità, portandolo ad es. a 0.1.

Volendo, possiamo aggiungere anche una Texture per la fonte di luce, mappandola magari con le coordinate della *Empty* di prima, in modo da aggiungere un po' di "rumore", per così dire, all'effetto.

* * *

Armature, ossa, azioni, animazioni - 1 di 5

Questo è il primo di cinque tutorial dedicati alle armature e alle loro animazioni in Blender 2.5. Questo tutorial è, più che altro, introduttivo: vedremo cosa sono le armature, a cosa servono, come vengono create eccetera.



La scena iniziale, con il modello che utilizzeremo per i nostri esempi

In Blender, un oggetto *Armature* è un oggetto composto, a sua volta, da altri oggetti, le *ossa* (*Bones*). Un osso è un oggetto a tutti gli effetti: può essere traslato, scalato, ruotato e altro ancora, ma deve appartenere ad una armatura.

Quando si aggiunge quindi un osso, lo si aggiunge ad un'armatura, per cui spostando un'armatura sposteremo tutte le sue ossa, mentre spostando (in Edit Mode o, come vedremo, in *Pose Mode*) un osso, sposteremo solo l'osso ed eventualmente altre ossa collegate.

Un'armatura può contenere anche un solo osso oppure più ossa.

Le armature così definite vengono associate alle mesh per deformarle con semplicità.
La comodità nell'utilizzare le armature piuttosto che nel modellare le mesh sta principalmente nel fatto che con un numero relativamente piccolo di oggetti, le ossa, possiamo "pilotare" più vertici di una mesh, che vengono in effetti *raggruppati ed associati ad un osso*, per cui possiamo animare facilmente mesh con migliaia di vertici agendo su pochi elementi; vi sono, comunque, altri vantaggi: ad esempio, un'armatura e un'animazione ad essa associata possono essere esportate e riciclate, ossia riutilizzate per altri modelli, anche in altri contesti.

La relazione di "parentela" (il collegamento) avviene quindi tra una mesh e un oggetto *Armature*.

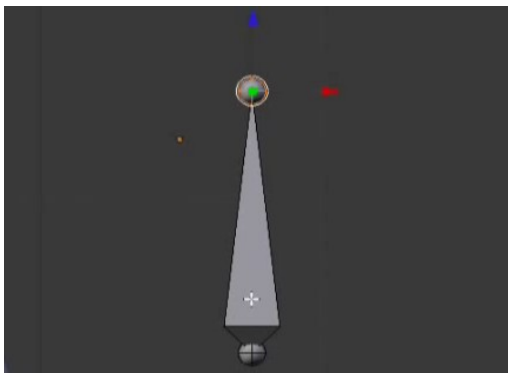
Aggiungiamo un oggetto armatura alla scena con *Add Armature*.

Per aggiungere un osso ad un'armatura possiamo procedere principalmente in due modi:

- per estrusione da un altro osso, premendo *E* in Edit Mode, oppure
- aggiungendo un altro osso con *ADD* e scegliendo, da tale menù, *Armature*.

Entrambe le operazioni vanno svolte, comunque, in *Edit Mode*; in particolare, aggiungendo una *Armature* in Object Mode *non* aggiungeremo un osso all'armatura selezionata, ma proprio un'altra armatura (un nuovo oggetto, distinto).

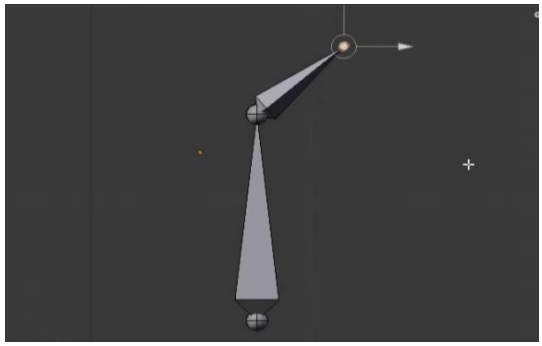
Un osso è composto da tre parti: la *radice*, che è il cerchietto posto in prossimità della piramide più piccola, quella più schiacciata; la *coda*, ossia l'altro cerchietto, e l'osso vero e proprio, il *corpo centrale*, costituito (in questa visualizzazione, nell'immagine seguente) da due piramidi a base quadrata.



Le parti che compongono un osso

È possibile selezionare singolarmente testa e coda, che fungono anche da punti di connessione tra varie ossa collegate tra loro.

Quando si effettua un'estrusione su un osso, il nuovo elemento viene aggiunto a partire dalla coda dell'osso genitore. A partire da un osso possono nascere, per estrusione, più ossa.



Creazione di nuove ossa mediante estrusione da un osso

Selezionando un osso cliccando sul corpo centrale sarà possibile, poi, ruotarlo, scalarlo e traslarlo.

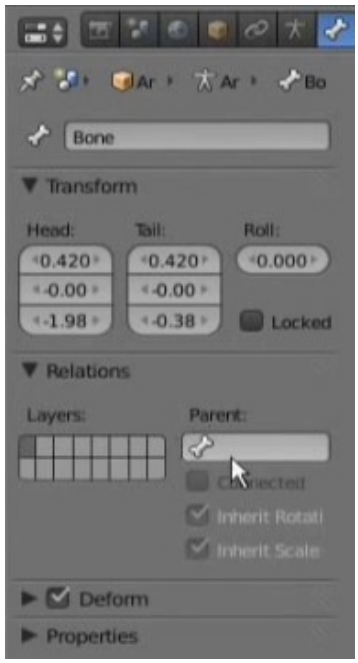
In Edit Mode è possibile aggiungere, rimuovere, scalare, ruotare e traslare le ossa; quella visibile in Edit Mode è, tra l'altro, la *REST POSITION* (posizione di riposo) dell'armatura.

Quando si deve definire un'armatura per una mesh, si dovrebbero definire le ossa che compongono l'armatura quasi "disegnandole" sul modello, come se si trattasse dello scheletro dell'oggetto, e questa operazione iniziale andrebbe fatta in modo *EDIT*, definendo proprio la *REST POSITION*, che in pratica rappresenta la posizione di base, la "forma di partenza" per l'armatura. In seguito, il modello verrà deformato spostando le ossa dell'armatura in *Pose Mode*, (in modo "posa", come vedremo).

Sempre in Edit Mode, è possibile selezionare le ossa col tasto destro del mouse oppure con *A* o con *B*, insomma come con qualsiasi altro oggetto di Blender; è possibile, poi, cancellare l'osso o le ossa selezionate con *X* oppure duplicarle con *SHIFT D*.

Come anticipato, un'armatura viene associata ad una mesh (e vedremo presto come), per cui conviene dare un nome alle armature in base al nome del personaggio o dell'oggetto al quale saranno associate, magari con un prefisso come "*Arm*", per cui avremo ad esempio "*ArmGiocatore1*", "*ArmGiocatore2*" e così via; per le varie ossa di un'armatura, poi, conviene dare dei nomi significativi, come ad esempio "*braccio destro*", "*piede sinistro*" eccetera, per comodità e per evitare di fare confusione in caso di armature complesse o in fase di definizione dei gruppi di vertici, come vedremo nella prossima puntata.

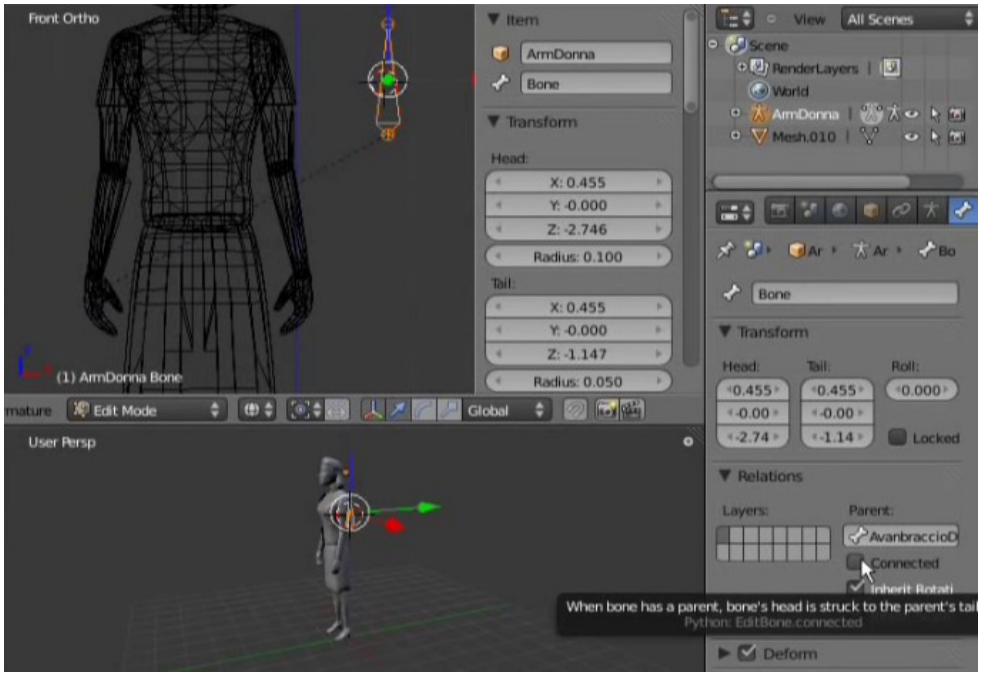
È possibile, ed anzi è di fondamentale importanza, creare delle relazioni di parentela tra le ossa di un'armatura: in Edit Mode, selezionare un osso (cliccando sul corpo centrale dello stesso, non sulla testa o sulla coda) ed andare nella scheda *Bone* (osso, appunto) della Properties Window; tale scheda consente di regolare alcuni parametri degli oggetti di tipo osso, come le relazioni di parentela.



La scheda Bone per l'osso selezionato

Nel campo *Parent* possiamo quindi scegliere, dal menù che apparirà a video e contenente i nomi delle ossa della stessa armatura dell'osso selezionato, l'osso che dovrà diventare genitore di quello selezionato.

Scegliendo un osso, vedremo nelle finestre di navigazione 3D una linea tratteggiata che parte dalla coda del padre e arriva alla testa del figlio; di default, questa operazione non collega "fisicamente" le ossa, cosa che però possiamo fare semplicemente selezionando la casella *Connected*, sotto il campo *Parent*.



La relazione di parentela tra due ossa

Per eliminare un vincolo di parentela, selezionate l'osso figlio e premete *ALT P*, scegliendo poi *Clear Parent* o *Disconnect Bone*, a seconda che vogliate eliminare il vincolo "logico", per così dire, di parentela, oppure scollegare le ossa mantenendo però la relazione di parentela.

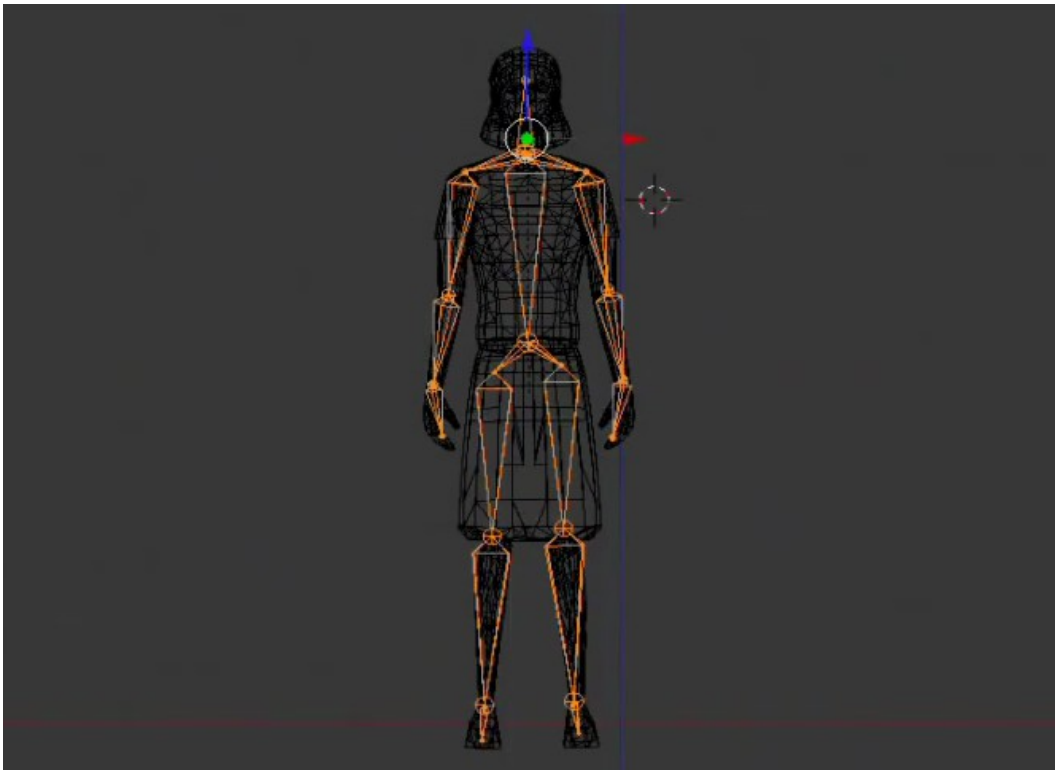
Non è possibile creare relazioni di parentela tra coppie di ossa appartenenti ad armature differenti.

* * *

Armature, ossa, azioni, animazioni - 2 di 5

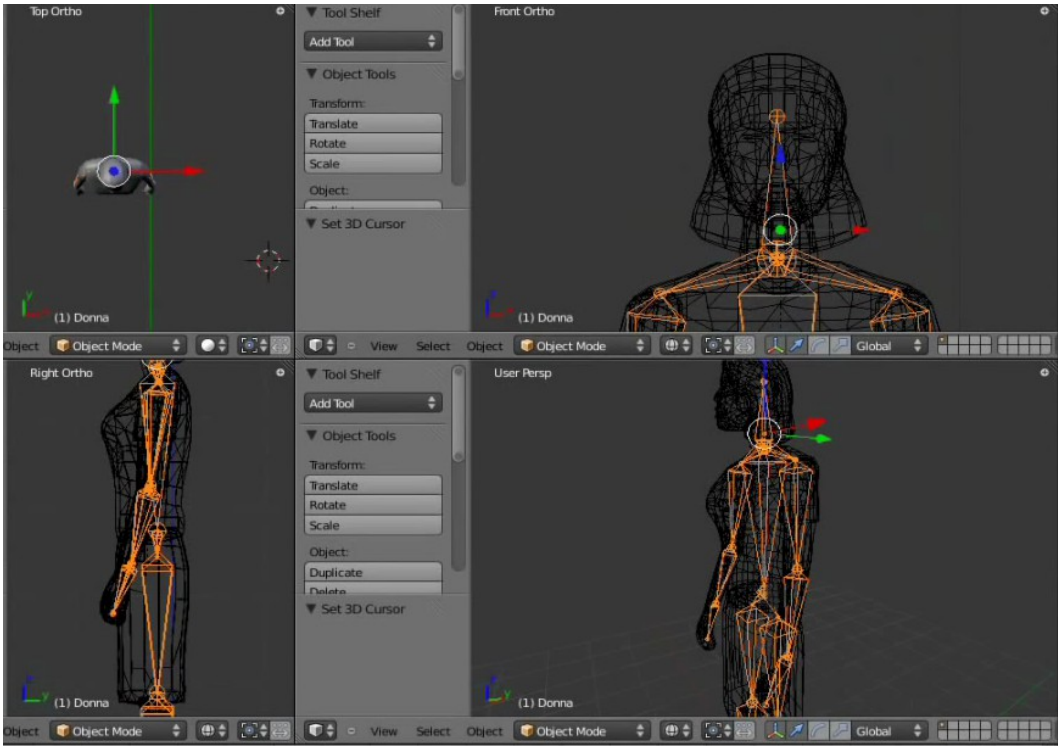
Questo è il secondo di cinque tutorial dedicati alle armature e alle loro animazioni in Blender 2.5. Questa volta vedremo come collegare le armature alle mesh, definendo i gruppi di vertici e, quindi, le "regioni di influenza" di ciascun osso sulle porzioni della mesh.

Nell'immagine qui sotto potete vedere il modello 3D scelto per questi esempi; è presente anche una semplice armatura composta da alcune ossa che faranno da scheletro per il modello.



La scena utilizzata per questo esempio, con una mesh e un'armatura

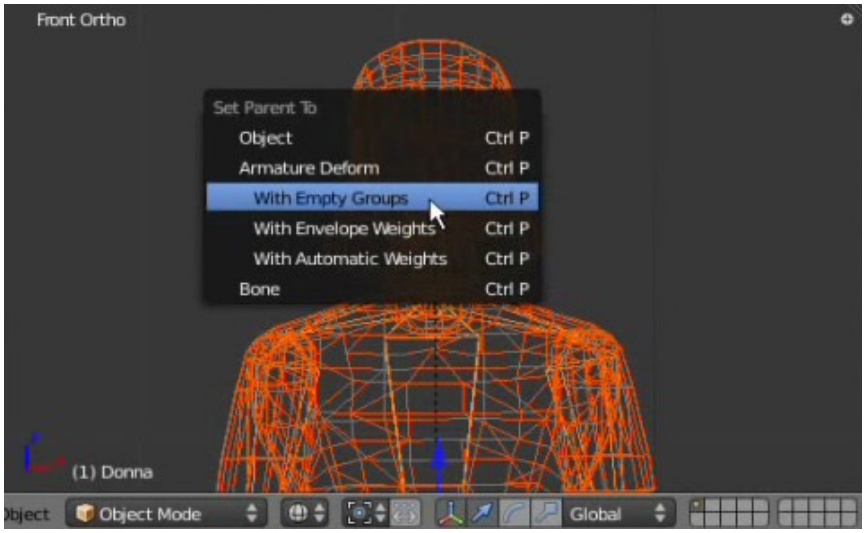
Un consiglio: quando definite un'armatura, se lo fate aggiungendo le varie ossa e posizionandole mentre utilizzate una vista frontale del modello, assicuratevi poi, prima di collegare l'armatura al modello, che anche nelle altre viste l'armatura sia posizionata correttamente, all'interno del modello. Conviene visualizzare la scena in modalità *Wireframe*, per assicurarsi che le ossa siano ben centrate non solo "frontalmente" ma anche in profondità nel modello 3D.



Assicurarsi di aver posizionato correttamente le ossa dell'armatura rispetto al modello

Per collegare un'armatura ad una mesh, in Object Mode dobbiamo selezionare prima la mesh, poi l'armatura (con *SHIFT* e click destro, per fare una selezione multipla) e premere quindi *CTRL P*; apparirà un menù a video, con tre possibili scelte per *Armature Deform*:

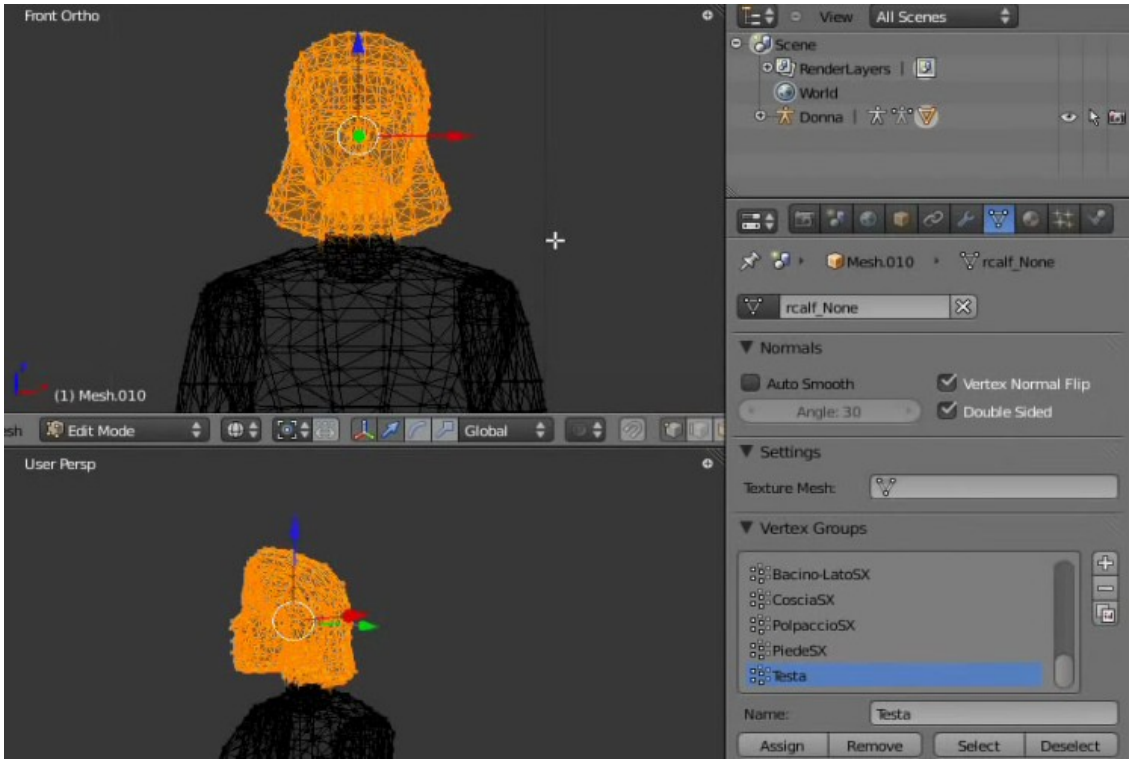
- *with Empty groups*, cioè letteralmente con gruppi vuoti;
- *with envelope weights*, cioè in base a degli "involucri";
- *with automatic weights*, cioè con pesi assegnati automaticamente da Blender.



Creazione della "parentela" tra Armatura e mesh

Andiamo con ordine.

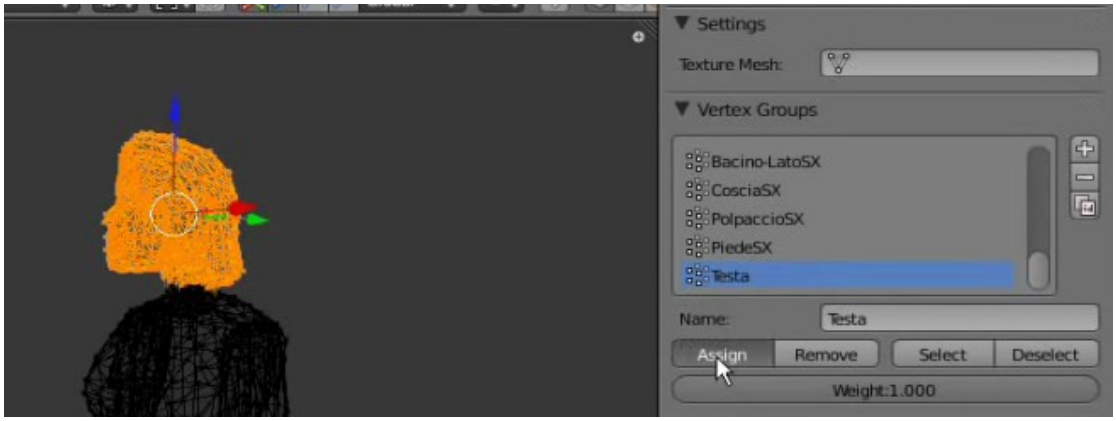
Scegliendo *WITH EMPTY GROUPS*, per ciascun osso dell'armatura verrà creato un Vertex Group, un gruppo di vertici, con lo stesso nome per il modello 3D. Come avrete intuito, l'aver assegnato nomi significativi alle varie ossa ("*braccio destro*", "*piede sinistro*", ...) aiuta molto in questo caso.



I Vertex Group creati con la modalità "With Empty Groups"

Questi Vertex Group saranno, comunque, vuoti: non conterranno alcun vertice al loro interno, per cui toccherà a noi

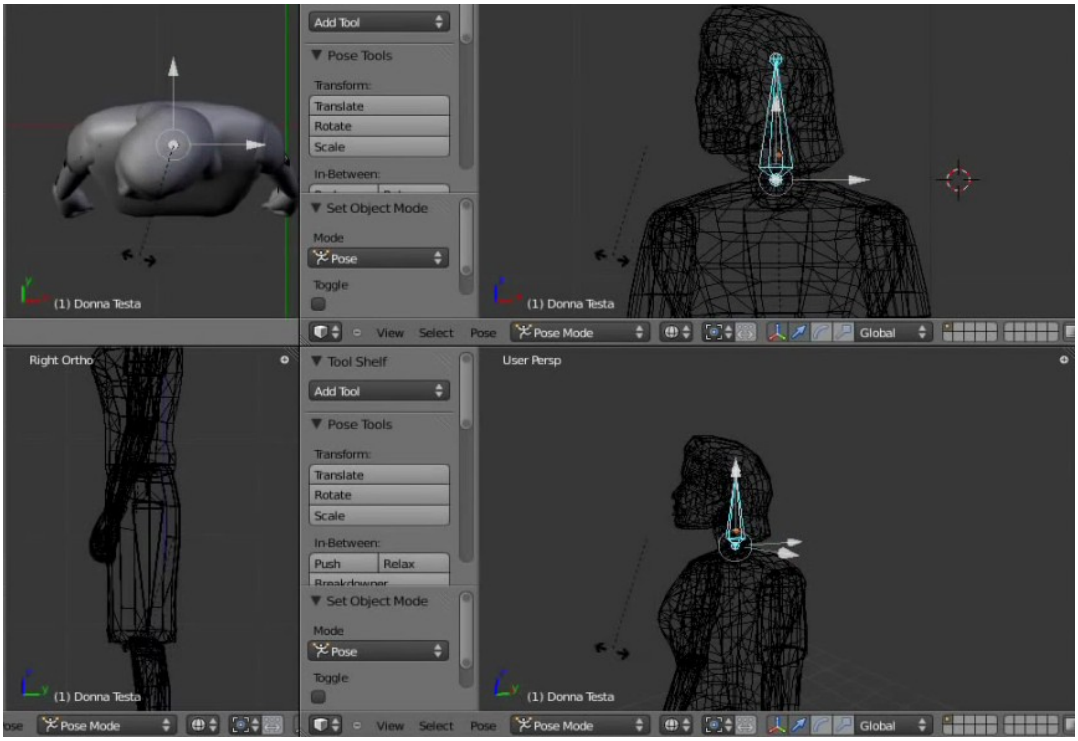
1. selezionare il modello 3D,
2. passare in Edit Mode,
3. selezionare i vertici da associare ad un osso,
4. andare nella scheda *Object Data* della Properties Windows
5. scegliere, dall'elenco dei *Vertex Groups*, il gruppo di vertici (ai quali corrisponderà un osso dell'armatura) al quale associare i vertici selezionati
6. e cliccare su *Assign*, per assegnare i vertici selezionati a quel gruppo.



Assegnazione di una selezione di vertici ad un Vertex Group creato (inizialmente vuoto) automaticamente da Blender

È possibile anche rimuovere uno o più vertici da un Vertex Group selezionandoli e cliccando, in questo stesso pannello, su *Remove*, rimuovi.

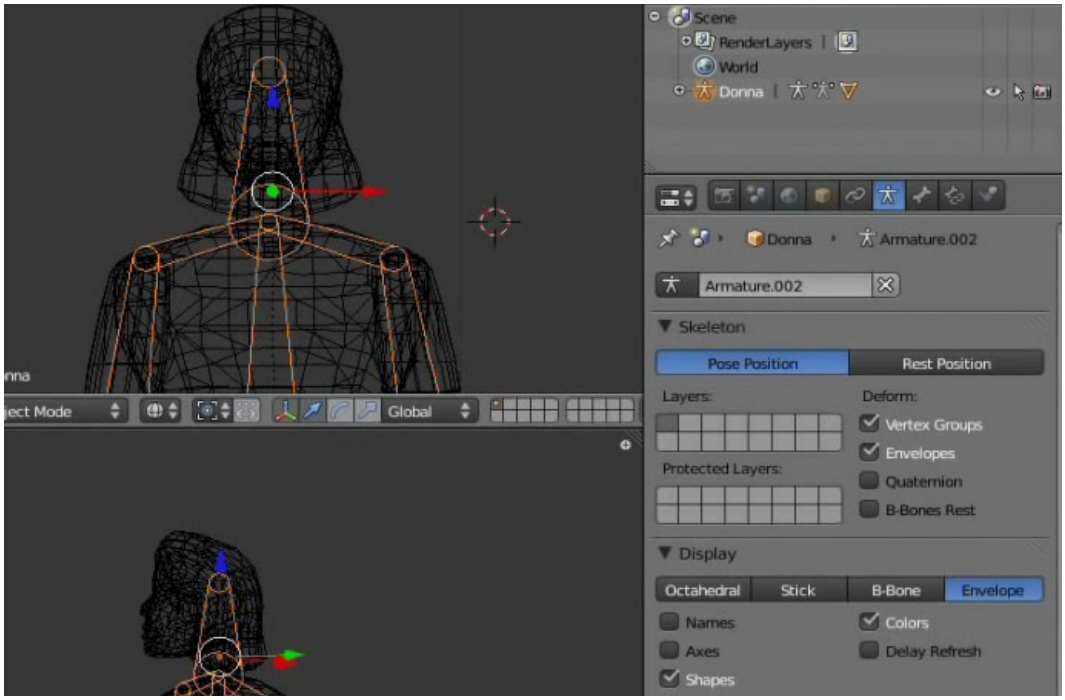
Con l'armatura selezionata, dopo aver assegnato qualche vertice ad un Vertex Group, al quale (come detto precedentemente) è associato automaticamente un osso, passiamo in *Pose Mode* e muoviamo quell'osso dell'armatura per vedere gli effetti sulla mesh.



Muovere un osso e valutare gli effetti di tale operazione sui vertici appartenenti al Vertex Group associato all'osso

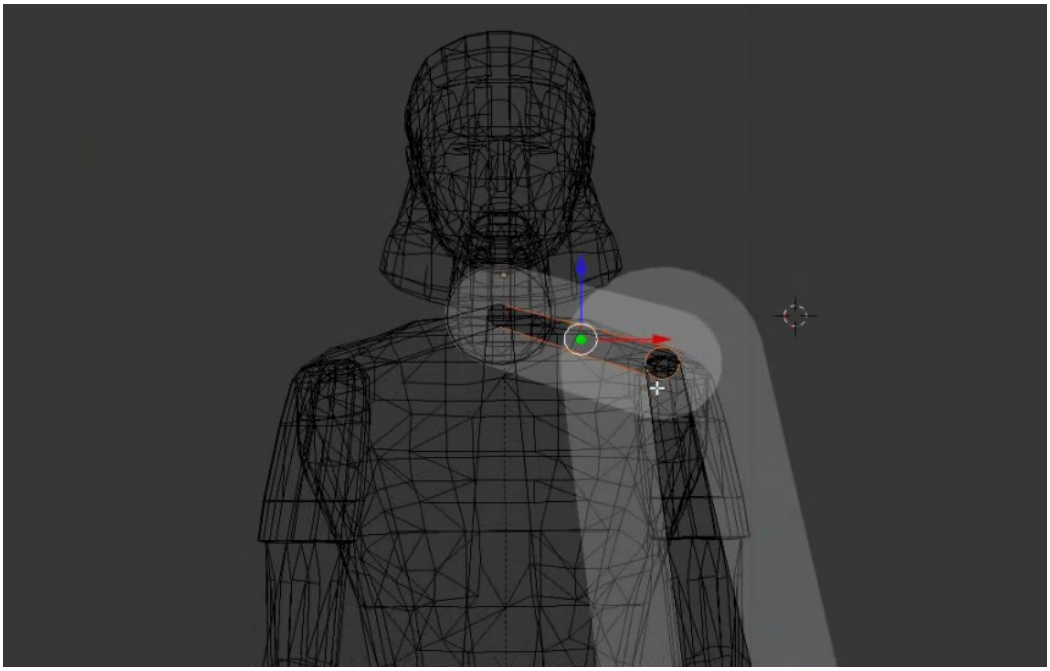
WITH EMPTY GROUPS è quindi una modalità interessante in caso di mesh e di armature non particolarmente complesse, per cui non si rischia di perdere tempo per selezionare correttamente i vertici ed assegnarli ai vari Vertex Groups.

Prima di esaminare la modalità *WITH ENVELOPE WEIGHTS*, una premessa: a ciascun osso è associata una zona di influenza (un involucro) detto *Envelope*; per poter osservare le ossa in questa modalità e, conseguentemente, le dimensioni di ciascun involucro, selezioniamo quindi l'armatura, apriamo la scheda *Object Data* nella Properties Window e clicchiamo sul pulsante *ENVELOPE* nella sezione *Display*.



Display Envelope nella scheda Object Data

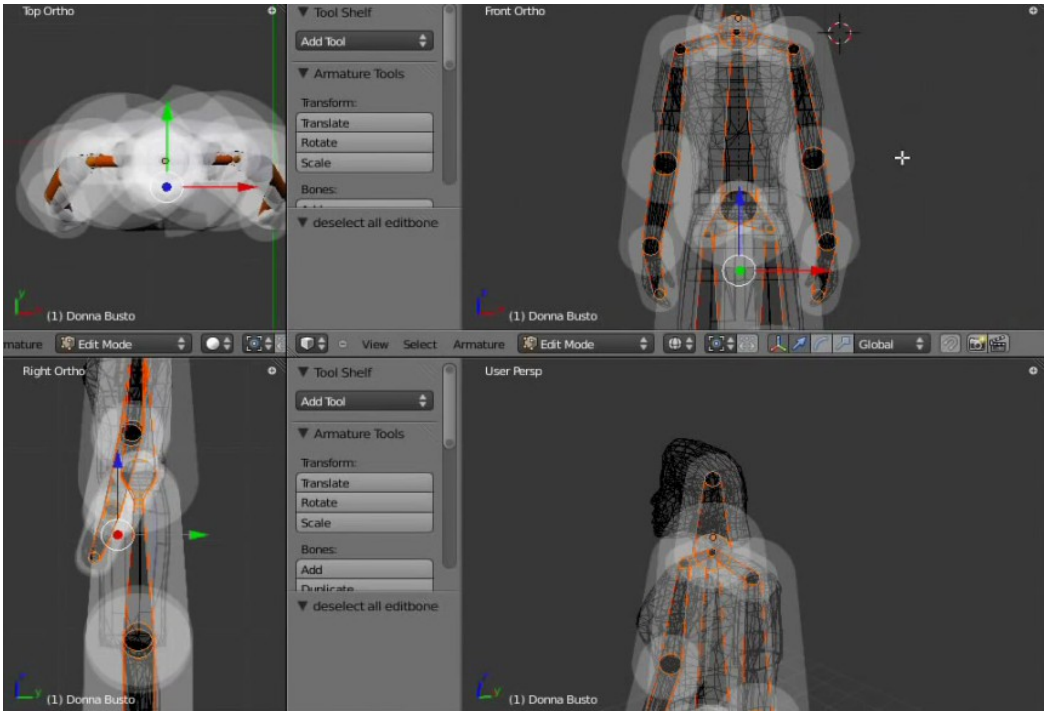
Passiamo in Edit Mode o in Pose Mode: a video dovremo poter vedere l'area dell'involucro; tra l'altro, notiamo anche che ci sono altri modi per visualizzare le ossa di un'armatura, ed anzi può risultare più facile lavorare in modalità di visualizzazione *Stick* e *B-Bone* che in modalità "ottaedro", quella classica.



La visualizzazione dell'involucro nella 3D View

Proviamo quindi a legare l'armatura alla mesh con la modalità *WITH ENVELOPE WEIGHTS*.

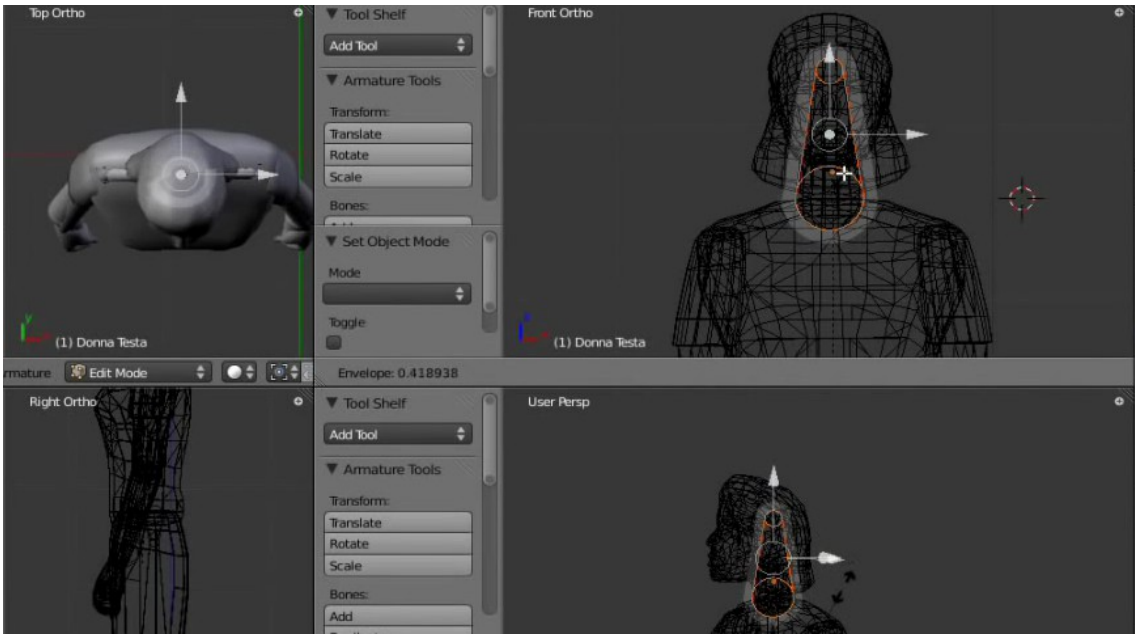
Passando in *Pose Mode* e muovendo o ruotando un osso, noteremo che se l'area di influenza di un osso include anche i vertici di un'altra parte del corpo, o se questo avviene per alcune ossa eventualmente collegate all'osso selezionato, muovendo o ruotando l'osso sposteremo anche quei vertici, e questo è chiaramente un effetto non voluto.



Problemi con la modalità Envelope Weights

È possibile risolvere questo problema scalando la dimensione degli involucri o lavorando con il *Weight Painting*, ma in questo momento vedremo solo la prima soluzione, visto che parleremo del *Weight Painting* nella prossima puntata.

Per scalare la dimensione di un involucro e, quindi, l'area d'azione di un osso, basta semplicemente selezionare l'osso in Edit Mode, in modalità di visualizzazione *Display Envelope*, e premere *ALT S*, dopodiché bisognerà ridimensionare interattivamente con il mouse, cliccando col tasto sinistro per rilasciare a fine lavoro.



Scalare gli involucri

Faccio notare, però, che questo lavoro dovrebbe essere fatto prima di legare un'armatura ad una mesh con *CTRL P*, per cui se intendete utilizzare la modalità *WITH ENVELOPE WEIGHTS* vi conviene definire prima le dimensioni degli involucri come mostrato e poi applicare l'armatura alla mesh. Se avete applicato prima l'armatura, non tutto è perduto: basta selezionare l'armatura in Object Mode, premere *ALT P*, eliminare il vincolo di parentela (selezionando *Clear Parent* dal menù che apparirà a video) e riprendere il lavoro.

La terza modalità è *WITH AUTOMATIC WEIGHTS*.
In questo caso, Blender tenterà, in qualche modo, di associare automaticamente le ossa a gruppi di vertici della mesh, ossia a *Vertex Groups*, creati anch'essi automaticamente.

In certi casi, questa soluzione porta a buoni risultati, ma a volte Blender può associare ad un osso gruppi di vertici che non hanno niente a che vedere con esso; l'esito dipende quindi da come è stata definita l'armatura e, comunque, dalla mesh.

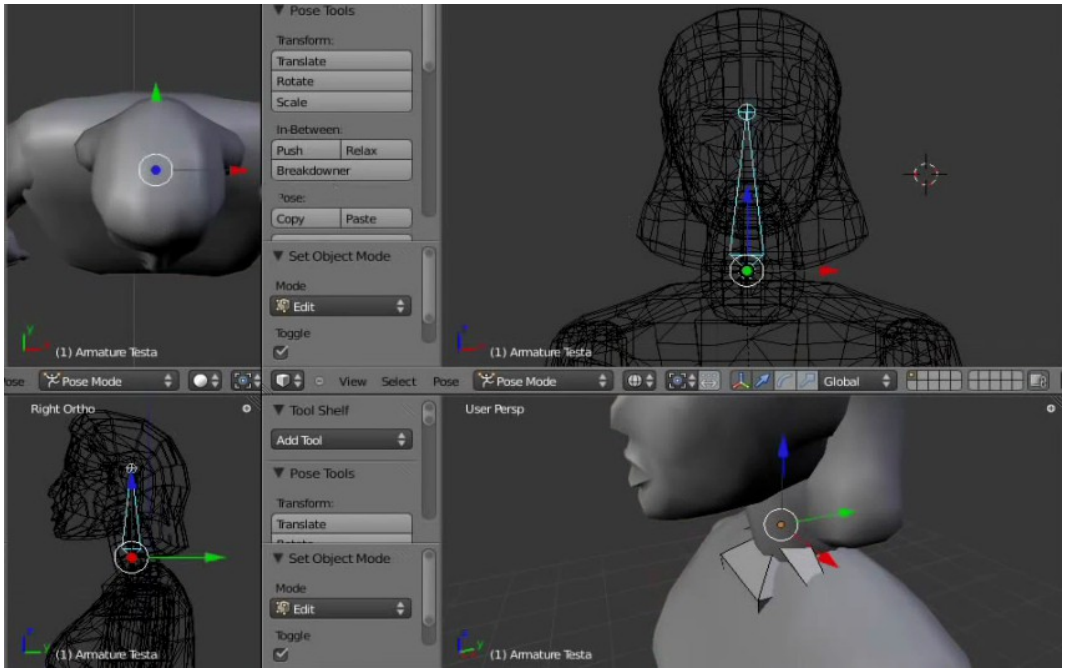
È possibile comunque modificare i *Vertex Group* del modello 3D, aggiungendo o rimuovendo dagli stessi alcuni vertici della mesh in Edit Mode, selezionandoli ed utilizzando i pulsanti *Assign*, *Remove*, *Select* e *Deselect* della scheda *Object Data*, nella Properties Window della mesh.

È possibile definire l'entità della trasformazione da applicare ai vertici quando si muove un osso mediante *Weigth Painting*, ossia letteralmente “dipingendo i pesi”, ma questo è un argomento che vedremo prossimamente.

* * *

Armature, ossa, azioni, animazioni - 3 di 5

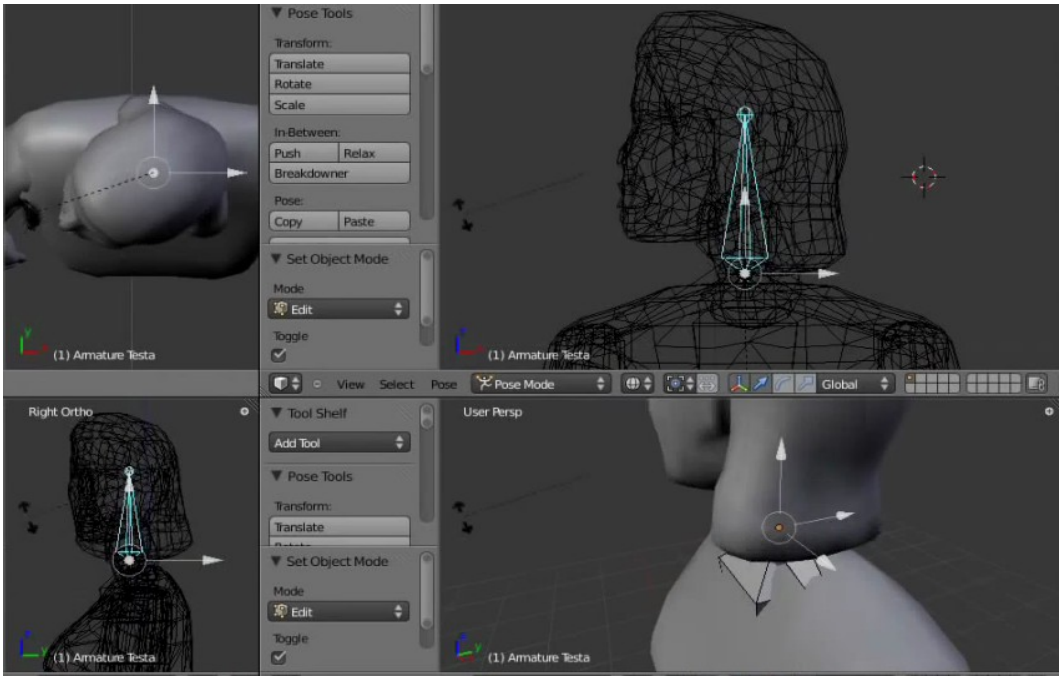
Questa è la terza puntata di cinque tutorial dedicati alle *Armature* e alle loro animazioni in Blender 3D 2.5; questa volta parleremo del *Weight Painting*. Nella scena visibile nell'immagine seguente è presente il modello 3D utilizzato anche nei capitoli precedenti e provvisto di *Armatura* mappata sul modello con modalità *Empty Groups*.



La scena utilizzata in questo tutorial

Ho assegnato vari gruppi di vertici ai relativi *Vertex Group* creati automaticamente con la modalità *Empty Groups*, per cui ora, ad esempio, i vertici che compongono la testa appartengono, appunto, al *Vertex Group* chiamato "*ossotesta*".

L'appartenenza è però "troppo forte", nel senso che in genere, ruotando la testa in *Pose Mode* (che è la modalità da usare per deformare la mesh mediante le *Armature*) dovrebbero essere presenti degli effetti anche sul "collo" del modello; effetti che, in questo caso, non ci sono, perché si tratta di porzioni di mesh appartenenti a due *Vertex Group* (e, quindi, a due ossa) differenti.

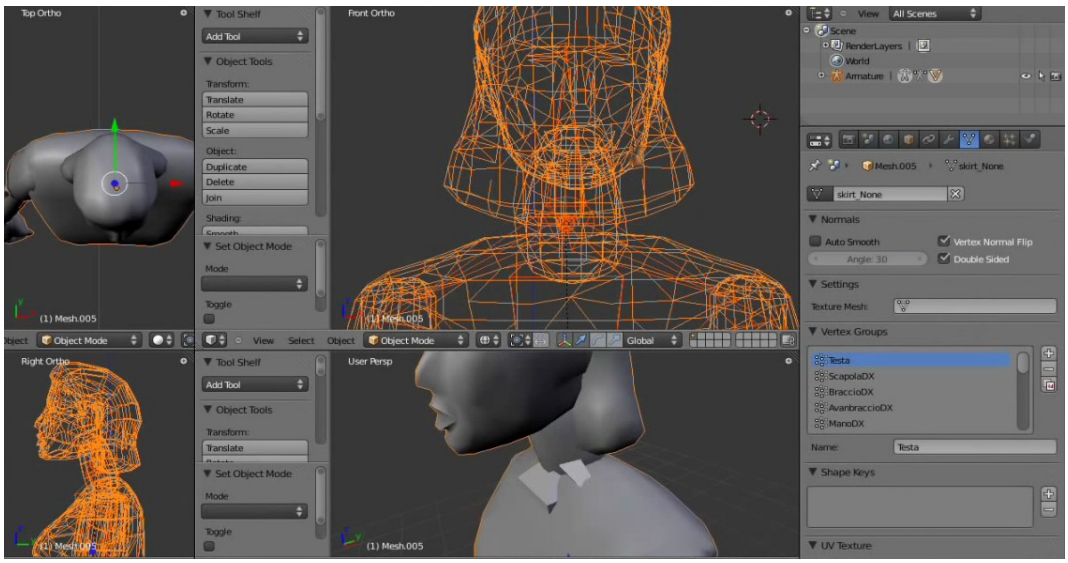


La deformazione realizzata con le impostazioni di base è "troppo forte", nel senso che riguarda solo le parti direttamente associate, senza effetti "periferici" minori

Per risolvere questo problema (ed altri, simili) si ricorre al *Weight Painting*, ossia al "dipingere i pesi" (letteralmente), dove questi "pesi" sono valori tra 0.0 e 1.0 che determinano l'intensità dell'effetto da applicare. In questo modo un osso e le sue trasformazioni avranno effetti su più porzioni differenti, con intensità di tali effetti variabili a seconda del peso assegnato alle varie zone.

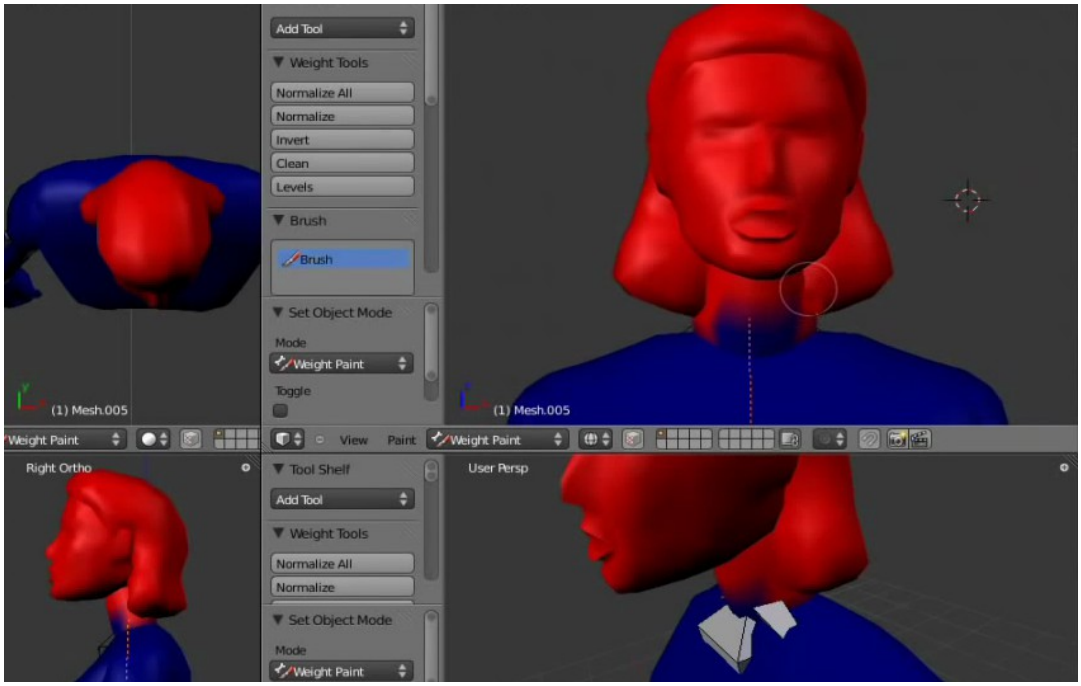
Procediamo con ordine.

In *Pose Mode*, selezioniamo l'osso che ci interessa, poi con *SHIFT* e click del tasto destro del mouse selezioniamo anche la mesh (con una selezione multipla) e passiamo in modalità *Weight Painting*.



Selezione dell'osso e della mesh, con una selezione multipla

Qualora la porzione di mesh corrispondente all'osso selezionato dovesse apparire blu, facciamo click col tasto destro del mouse in prossimità dell'osso: adesso la porzione di mesh (in questo caso, la testa) dovrebbe essere rossa.



La porzione di mesh associata all'osso è rossa

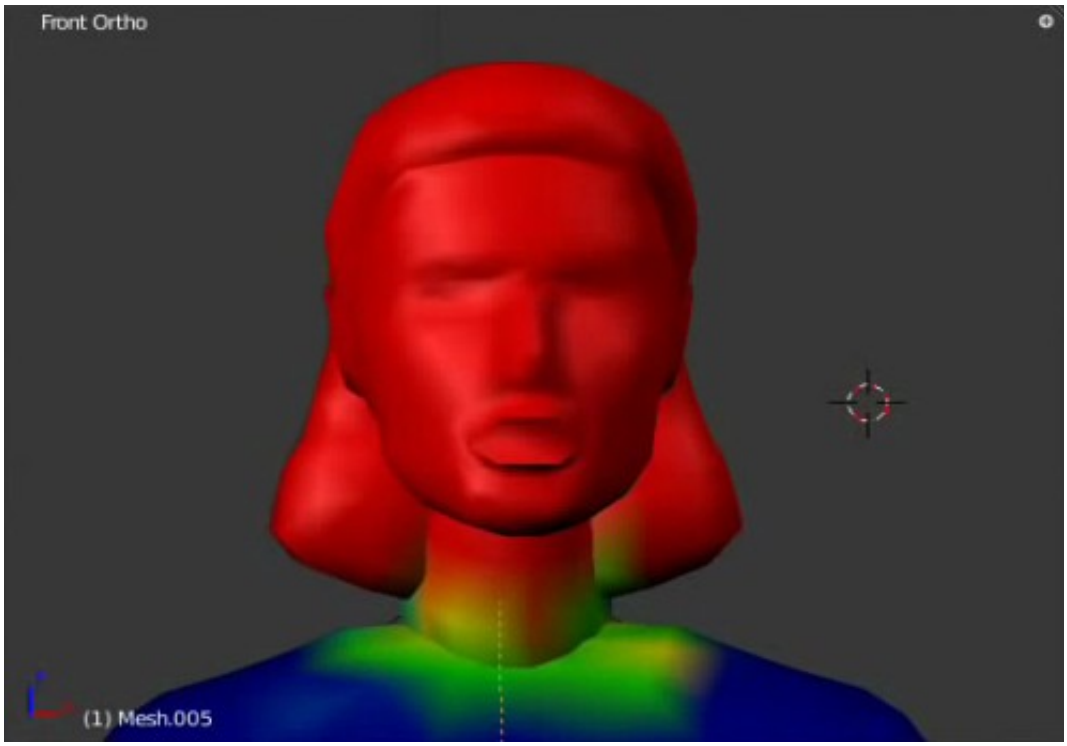
Il rosso è il colore del peso massimo, ovvero dell'effetto massimo associato (per l'osso selezionato) ad una regione della mesh; dal momento che la testa è la regione associata a quell'osso (mediante definizione dei *Vertex Group*), essa è correttamente colorata di rosso, con l'effetto applicato quindi in modalità uniforme a tutte le parti della testa. Il blu è il colore dell'effetto nullo, ed infatti le altre porzioni della mesh sono blu.

Nelle finestre di navigazione 3D, nelle *Tool Shelf* dovrebbero essere ora presenti gli strumenti propri della modalità *Weight Painting*; ci interessano, in particolare, gli strumenti posti nella sezione *Brush* (lett.: "spazzola"), per definire dimensioni ed intensità dell'effetto.

La sezione Brush della Tool Shelf in modalità Weight Painting

Con *Weight* impostiamo il valore del peso che vogliamo assegnare. il valore di default è 1.0, per assegnare il peso massimo, ma possiamo inserire qualsiasi valore tra 0.0 e 1.0 per assegnare valori differenti; ad esempio, è possibile inserire 0.0 per annullare il peso della trasformazione in certe parti della mesh, perché magari non dovrebbero essere influenzate da tale trasformazione ma sono state assegnate per errore.

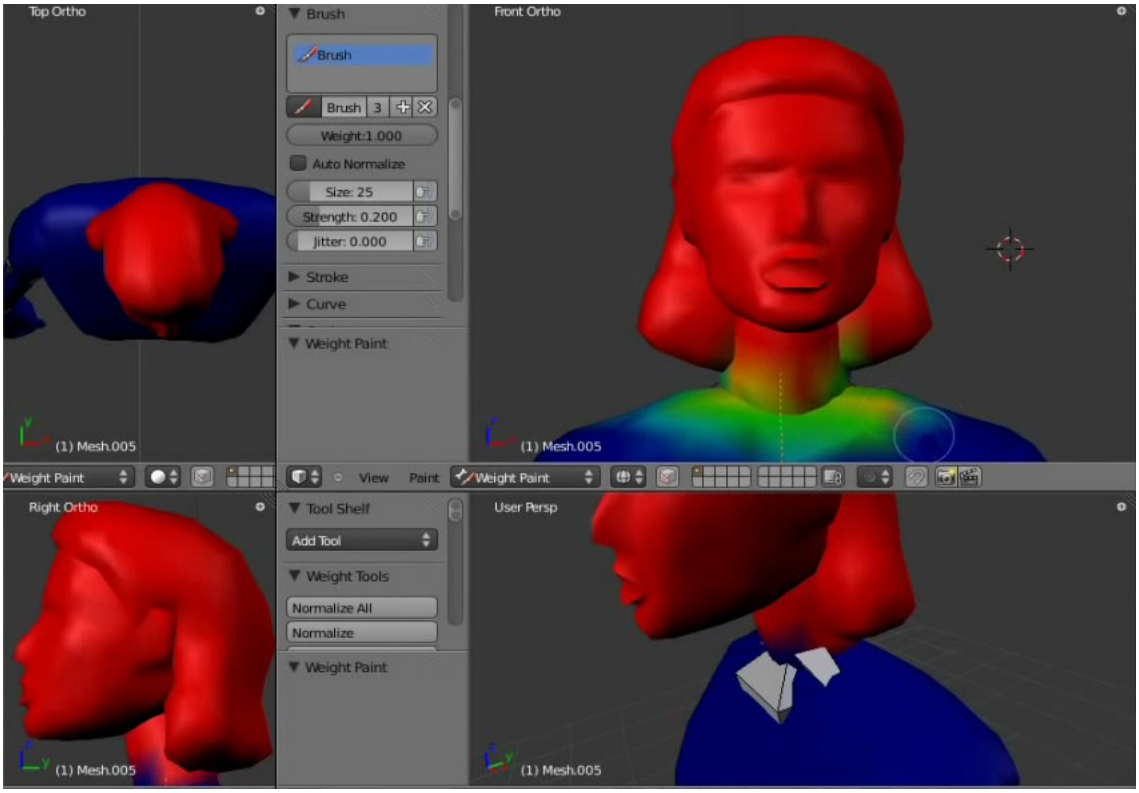
Il peso, comunque, può essere impostato in maniera graduale (per cui serviranno più "passate", ovvero più "click", per ottenere l'effetto voluto), oppure con un solo click del tasto sinistro del mouse sulle varie regioni: il tutto dipende dal valore di *Strength*, "intensità", che di default è 0.2 (per cui servono più passate); impostando ad esempio 1.0 per *Weight* e *Strength* e cliccando su una regione della mesh, il peso massimo verrà applicato in un colpo solo, mentre con altre combinazioni si avranno, intuitivamente, effetti differenti.
Rosso e blu sono, come detto, i colori estremi, associati ai pesi 1.0 e 0.0 rispettivamente, ma per i pesi intermedi (0.2, 0.5, ...) si avranno tonalità di altri colori, ed in particolare ciano, verde e giallo.



I colori intermedi dei pesi in Weight Painting

Con *Size* impostiamo, ovviamente, l'area del pennello, ossia l'area di applicazione dell'effetto, per cui per curare dei dettagli utilizzeremo un valore basso, mentre per colorare zone ampie con poche passate potremo impostare un valore alto.

Con *Weight* a 1.0 e *Strength* a 0.2 clicchiamo un paio di volte sul collo e sulla parte alta delle spalle, del petto e delle scapole. Quest'area di influenza probabilmente risulterà eccessiva, per la sola rotazione della testa, ma così potremo osservare meglio l'effetto, per questo esempio.

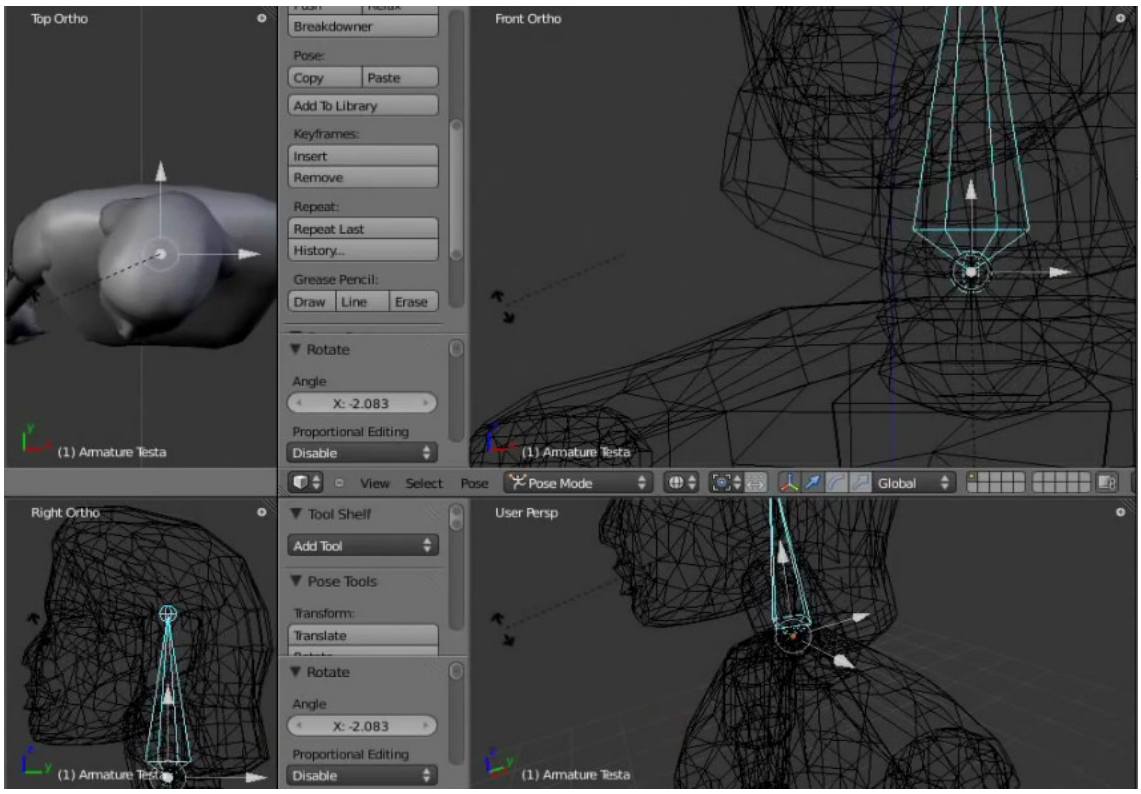


Influenziamo un po' anche il collo, le spalle, il petto e le scapole

Adesso usciamo dalla modalità *Weight Paint* e torniamo in Object Mode.

Selezioniamo l'*Armatura* (e solo questa), passiamo in *Pose Mode* e ruotiamo un po' l'osso associato alla testa del modello: adesso la deformazione dovrebbe interessare anche i vertici e le facce corrispondenti alle aree che abbiamo colorato di verde e giallo.

Non si tratterà, comunque, di un effetto "pesante", visto che non abbiamo inserito il peso massimo, ma di quel tanto che basta per aggiungere un po' di realismo al movimento (l'alternativa sarebbe stata la rotazione della sola testa con uno stacco netto e il resto del corpo e delle spalle del tutto immobili: un'animazione davvero poco realistica).



La deformazione sulle parti "periferiche", grazie ai pesi appena impostati per tali aree

Quanto detto può essere esteso facilmente alle altre ossa dell'*Armatura*, per cui potrete impostare dei pesi differenti per impostare delle trasformazioni “periferiche”, per così dire, per le ossa delle braccia (influenzando ad esempio il torace), per quelle delle gambe (definendo degli effetti sulle ginocchia o sul bacino) e così via.

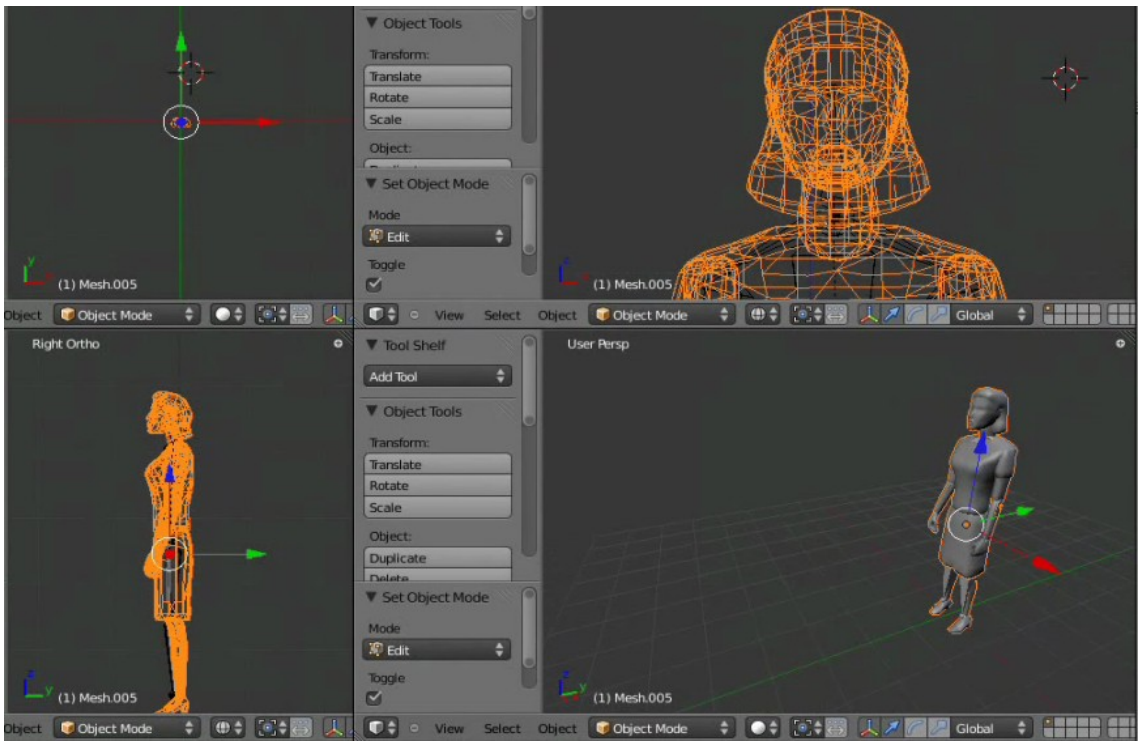
Ci vuole quindi un po' di tempo per definire per bene un'*Armatura* e i pesi di un modello, ma ne vale davvero la pena, specialmente se si devono realizzare delle animazioni.

* * *

Armature, ossa, azioni, animazioni - 4 di 5

Questo è il quarto di cinque tutorial dedicati alle armature e alle loro animazioni in Blender 2.5. Questa volta parleremo delle animazioni delle armature e, quindi, delle mesh, mediante la definizione di *chiavi IP*O in *Pose Mode* e la conseguente definizione delle *Actions*, le azioni.

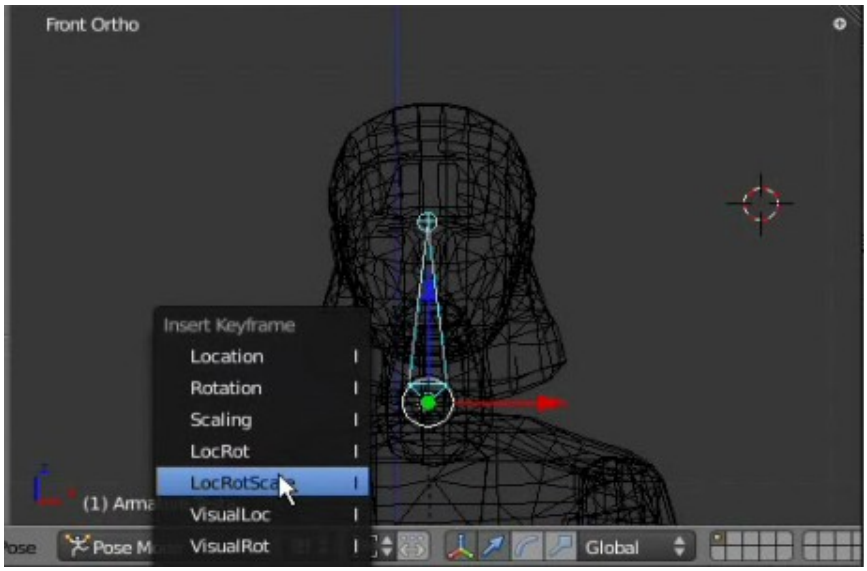
Nella scena visibile nell'immagine qui sotto è presente il modello visto nella scorsa puntata, con i Vertex Group assegnati alle varie parti del corpo e ai quali corrispondono anche delle ossa dell'armatura legata alla mesh; in particolare, abbiamo visto che ruotando l'osso definito come "*OssoTesta*" si ha una rotazione della testa del modello, appunto. Vediamo ora come realizzare un'animazione di tale rotazione.



La scena che utilizzeremo in questo tutorial

Selezioniamo l'armatura, passiamo in *Pose Mode* e selezioniamo l'osso associato alla testa.
Inseriamo un keyframe di tipo *LocRotScale*, ossia Posizione, Rotazione e Dimensioni, premendo il tasto *I* e scegliendo, dal menù che apparirà a video, la voce *LocRotScale*; in questo modo, stiamo memorizzando posizione, orientamento e dimensioni dell'osso per questo frame.

Spostiamoci ora, ad esempio, al frame 50, ruotiamo un po' l'osso associato alla testa, premiamo nuovamente *I* e scegliamo ancora una volta *LocRotScale* per inserire un keyframe al frame 50 per l'osso, memorizzando anche questa volta le caratteristiche spaziali dell'oggetto.



Memorizzazione dei Keyframe in Pose Mode per la rotazione (dell'osso) della testa

La cosa interessante è che, come avviene per le mesh in Object Mode, Blender ha realizzato l'interpolazione delle posizioni di ogni singola faccia del modello, ossia ha creato la trasformazione animata, nel corso dei frame dal frame 1 al cinquantesimo, per cui ora lanciando un'anteprima dell'animazione con *ALT A* mentre il focus è in una finestra di navigazione 3D dovremo poter vedere a video l'animazione. Possiamo bloccare quest'anteprima premendo nuovamente *ALT A*.

Come avviene per le mesh in modalità oggetto, anche per le armature vengono create delle curve *IPO*, ossia *InterPolation Curves* (curve di interpolazione), visibili nelle finestre di tipo *GraphEditor*; ad ogni modo, la cosa interessante è analizzare queste animazioni nelle finestre di tipo *DopeSheet* (che una volta, fino alla versione 2.49 di Blender 3D, si chiamano finestre *Actions*, ossia delle *Azioni*).

Le *Azioni* sono in effetti le interpolazioni create mediante keyframes; la loro visualizzazione nel *DopeSheet* è particolarmente utile perché consente di visualizzare queste animazioni come coppie di rombi disposti solo sulla *timeline*, indicando quindi inizio e fine animazione, senza specificare i valori delle varie trasformazioni (ad esempio, la coordinata X o un fattore di scala), evitando di creare confusione con le curve tipiche della visualizzazione IPO.

La visualizzazione delle azioni nel DopeSheet

Nelle finestre *DopeSheet* è possibile selezionare questi rombi con un click del tasto destro del mouse su di essi, per poi traslarli premendo *G* (per *GRAB*, "trascina"), spostandoli col mouse e fissando la posizione finale con un click del tasto sinistro.

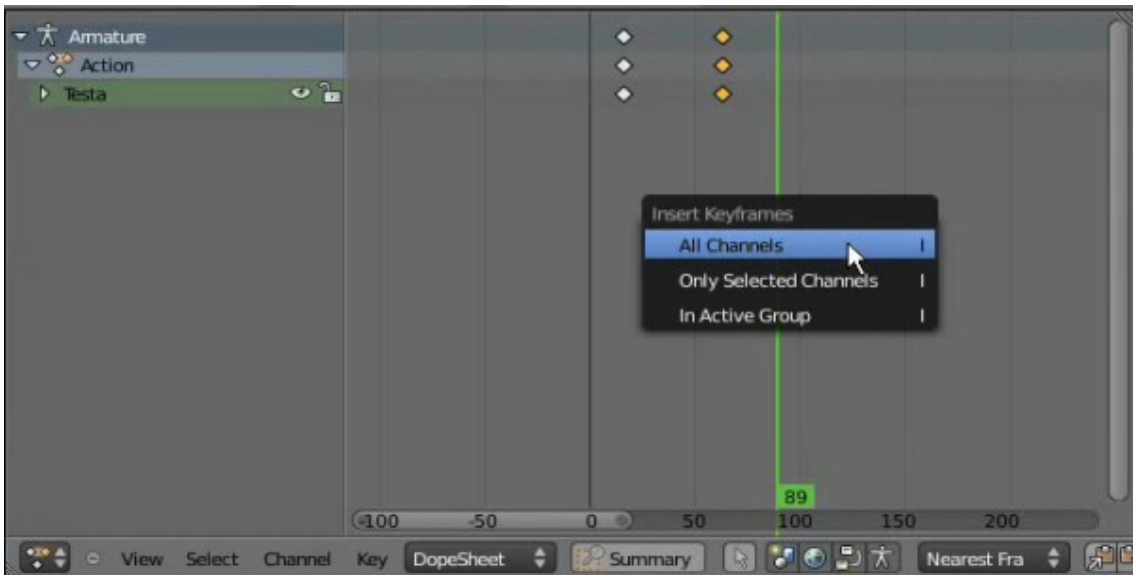
La selezione può essere anche multipla, con *B* (come avviene per le mesh in modo oggetto o in modo Edit) o con *A*, che consente tra l'altro di selezionare o deselezionare tutti i "*marker*", i rombi, in un colpo solo.

È quindi possibile ad esempio ritardare una sequenza animata, spostando una coppia di rombi che magari è stata definita (come nel nostro caso) al frame 1, più in avanti nella *Timeline*, effettuando il *grabbing* dell'intera coppia, oppure aumentare o diminuire la durata di un'animazione (nel nostro caso, la rotazione della testa) spostando solo il *marker* di fine animazione, e così via.

Selezionando una coppia di marker e premendo *SHIFT D*, poi, possiamo duplicare una sequenza animata, spostandola magari più in avanti nella *Timeline*, e questo può tornare utile per far ripetere certe azioni ad una porzione di mesh (ad esempio, per definire un ciclo di camminata di un modello basterà definire una volta sola l'animazione delle gambe e dei piedi e poi duplicare).

Possiamo infine cancellare dei *marker* con *CANC* o con *X*.

I keyframes di un'animazione possono essere inseriti anche nella finestra *DopeSheet,* infatti qui è ovviamente presente una barra del tempo e cliccando in un punto di questa finestra ci posizioneremo in un certo frame, come indicato dalla linea verticale che incrocia appunto la *Timeline* in basso nella finestra; a questo punto, sarà sufficiente premere *I* e scegliere una delle voci del menù che apparirà a video per inserire un keyframe di animazione e creare un *Action marker.*

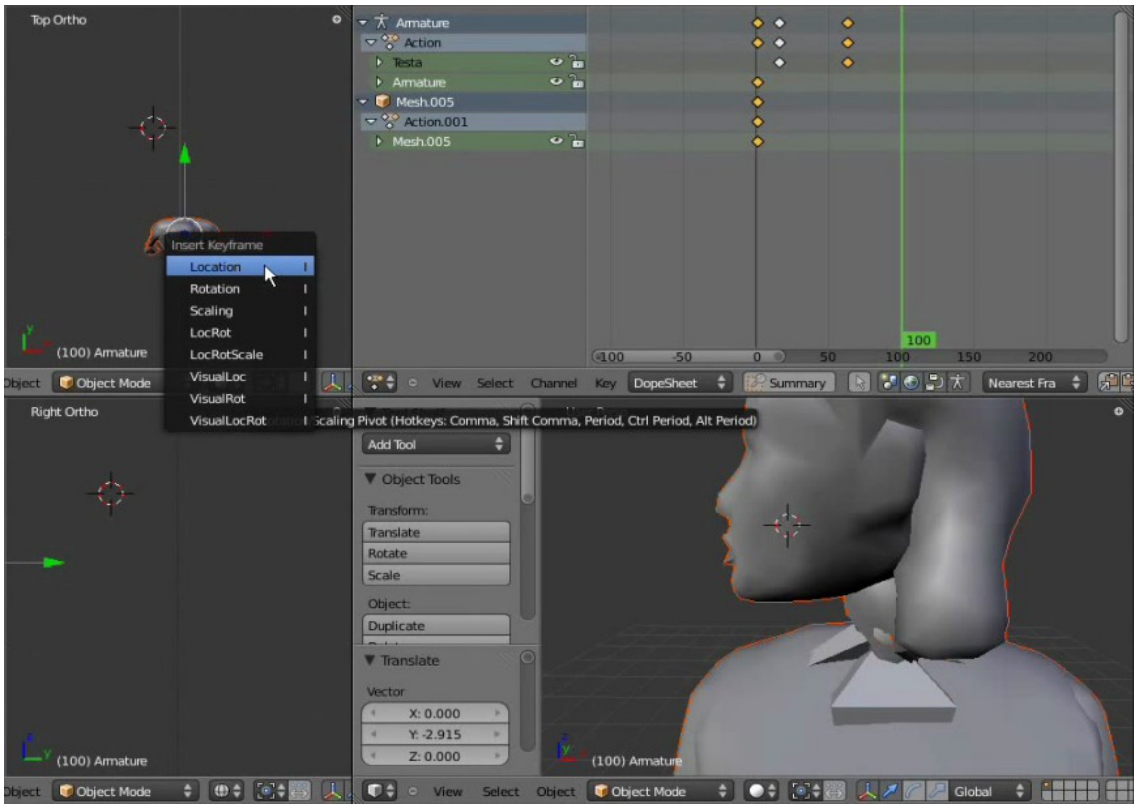


Inserimento di un Keyframe nella Timeline di un DopeSheet

Queste animazioni sono attive a livello di armatura, per cui nel modello vero e proprio, nella mesh, verranno realizzate come trasformazioni per così dire "locali"; ad esempio, nel definire un ciclo di camminata definiremo le animazioni delle armature delle varie parti del corpo "sul posto", ripetendole: non conviene animare l'intera camminata mediante armature.

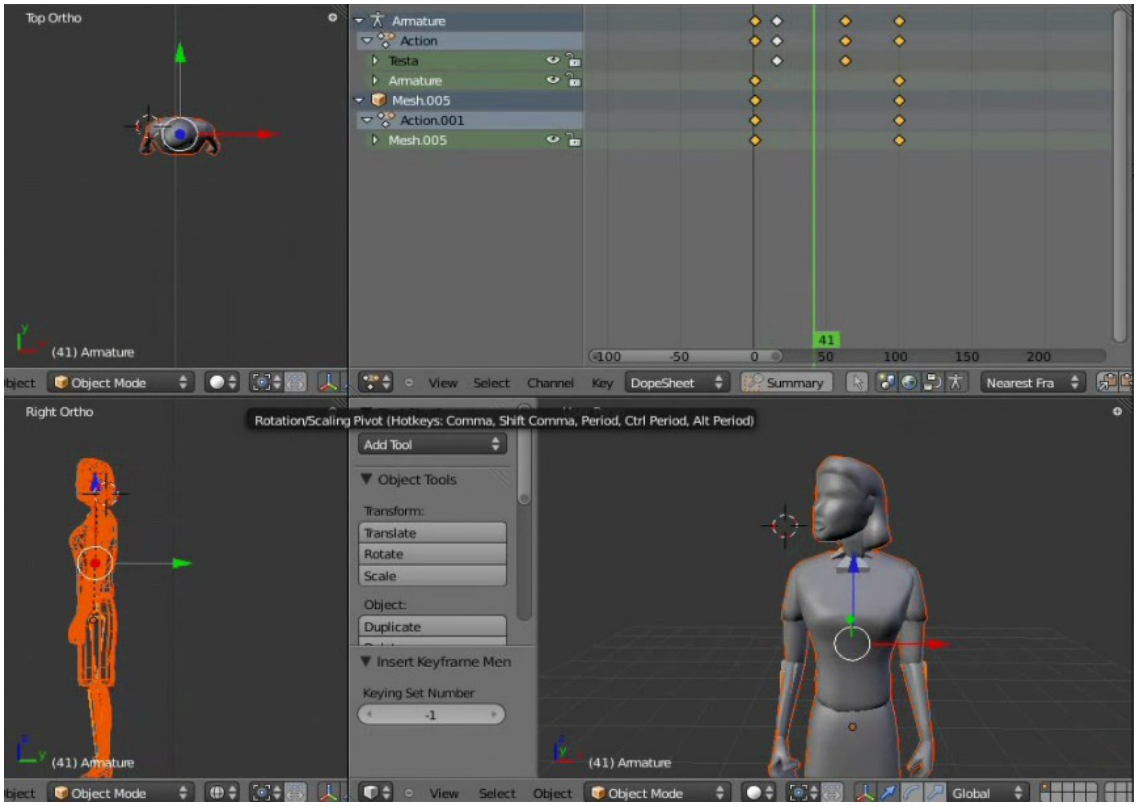
Quello che si fa, in genere, è invece ricorrere alle animazioni "classiche", cioè quelle della mesh in modo oggetto, per trasformare l'oggetto (ad esempio, traslarlo) nella scena 3D; beh, è più facile da

fare che da dire: passiamo in Object Mode, selezioniamo sia la mesh che l'armatura (una selezione multipla, quindi, con *SHIFT* e click sul tasto destro del mouse, oppure mediante *B* e selezione ad area), torniamo al frame 1, premiamo *I* ed inseriamo un keyframe *LOC* (quindi, solo posizione, ad esempio); adesso, spostiamoci al frame 100, ad esempio, spostiamo, traslandola, la selezione (mesh e armatura) ed inseriamo nuovamente un keyframe *LOC*.



Inserimento dei Keyframe LOC sia per l'Armatura che per la mesh (spostata, traslata nella scena)

Abbiamo quindi spostato nello spazio 3D l'insieme, mantenendo comunque la trasformazione creata precedentemente, come è possibile vedere lanciando un'anteprima dell'animazione. Questa è la base per la creazione di animazioni più complesse, come cicli di camminate, salti o altro.



Screenshot dell'animazione ottenuta in questo modo: abbiamo sia l'animazione, in Pose Mode, dell'osso dell'armatura, sia della mesh, in Object Mode, che sposta così anche l'armatura associata

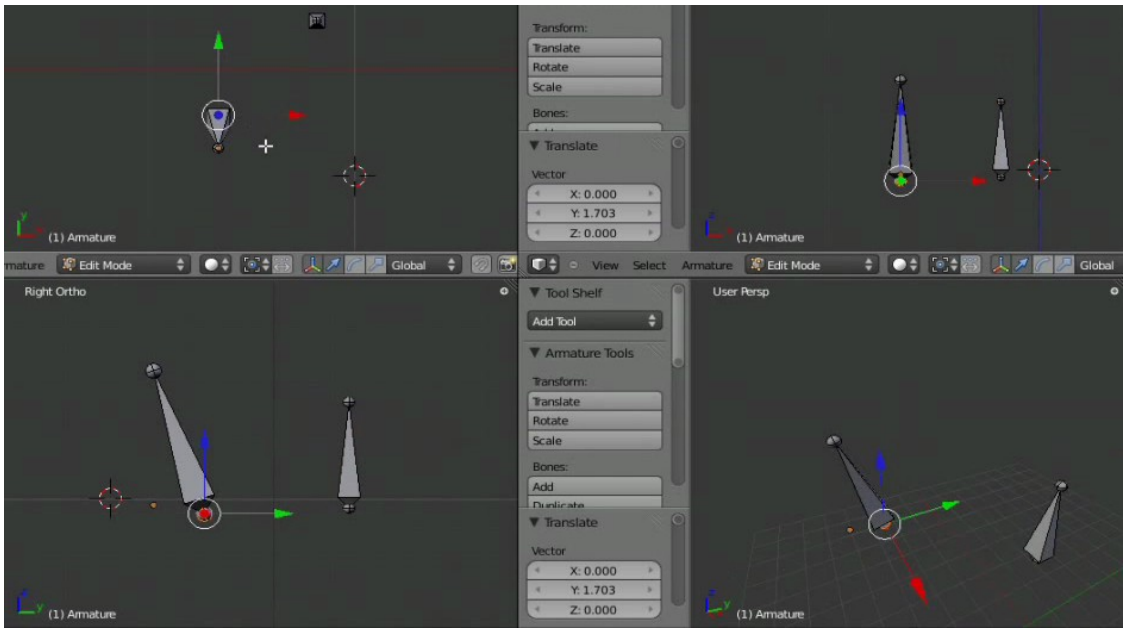
* * *

Armature, ossa, azioni, animazioni - 5 di 5

Questo è il quinto ed ultimo tutorial del ciclo dedicato alle armature e alle loro animazioni in Blender 2.5. In questa puntata parleremo un po' dei *Constraint*s, i vincoli.

I *Constraints* (termine tradotto in italiano con "*vincoli*", di solito) sono caratteristiche degli oggetti, siano essi mesh o ossa. Esistono *Constraint* propri delle mesh, *Constraint* propri delle ossa e *Constraint* comuni ai due tipi.

Per questa panoramica utilizzeremo l'armatura visibile nell'immagine qui sotto, composta da due ossa uguali, non legate tra loro. Per inserire un vincolo *Constraint* per l'osso selezionato in *Pose Mode*, apriamo la scheda *Bone Constraints* nella Properties Windows e clicchiamo sul menù *Add Constraint*. Sono presenti diverse voci, suddivise per categorie; noi prenderemo in esame quelle principali, iniziando con *Copy Location*.

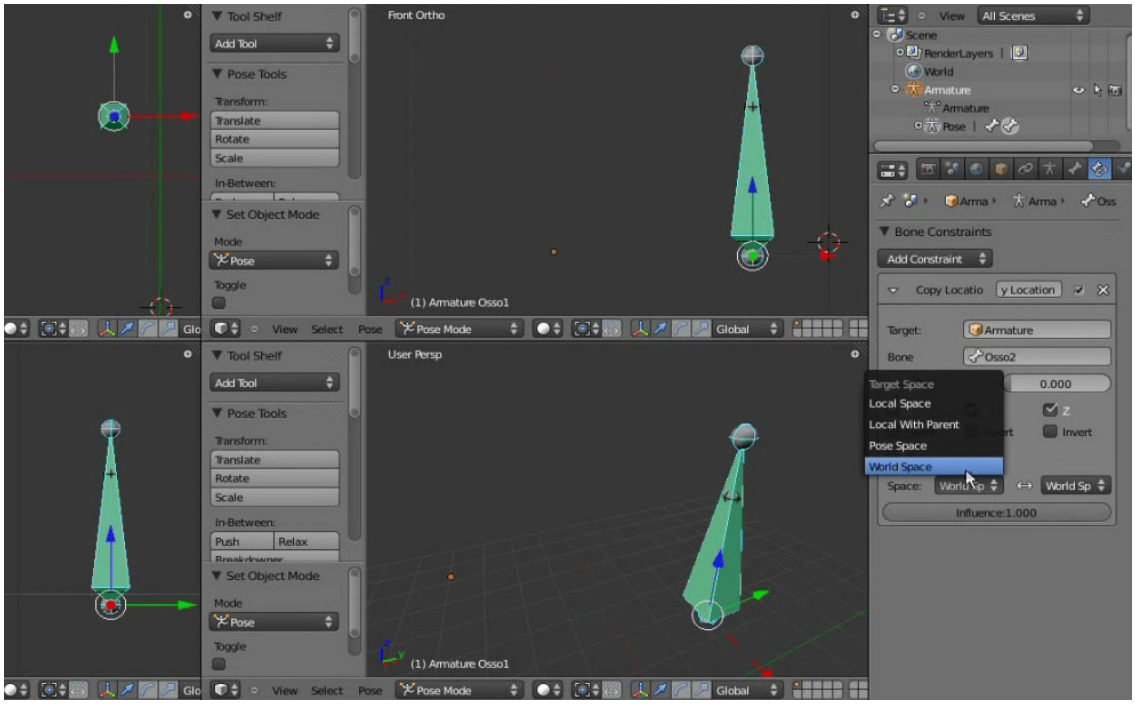


La scena utilizzata in questo esempio

Copy Location serve, intuitivamente, ad assegnare una posizione ad un osso in base alla posizione di un altro osso, detto osso *Target*, bersaglio.

La copia può essere fatta in riferimento sia allo spazio locale che globale del target e dell'osso provvisto di tale *Constraint*, con tutte le possibili combinazioni di tali valori. È possibile scegliere, poi, quale coordinata vincolare, ossia X, Y, Z o una loro combinazione; selezionando inoltre il pulsante col segno -, la copia sarà invertita.

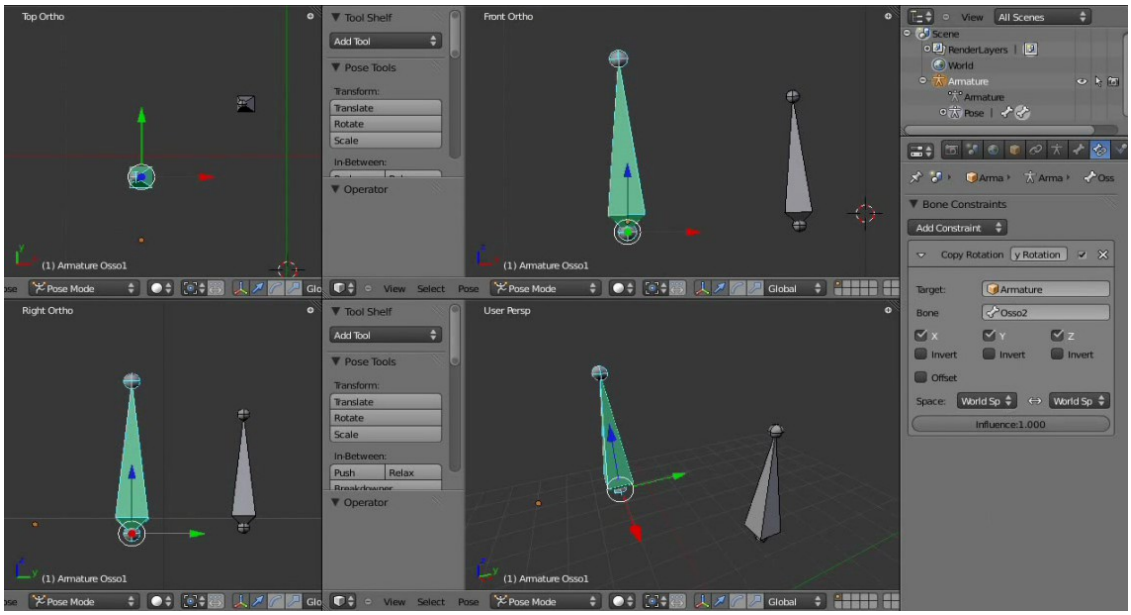
L'osso *Target* non è identificato da un unico punto, ma dalla classica struttura dell'osso, per cui dobbiamo decidere quale punto prendere come riferimento per effettuare la copia: inserendo il valore 0 in *Head-Tail*, prenderemo come punto di riferimento la testa (la "radice" dell'osso), mentre con 1 la coda; a valori intermedi corrisponderanno posizioni intermedie tra testa e coda.



Utilizzo di Copy Location

Copy Rotation fa per le rotazioni delle ossa quello che *Copy Location* fa per le posizioni: in base ad un osso di riferimento, detto osso *Target*, modifica l'orientamento dell'osso provvisto di tale *Constraint*. Anche qui sono presenti XYZ, con lo stesso significato che hanno in *Copy Location*.

Utilizzando l'opzione *Local* anziché *Global*, verrà copiata l'entità della rotazione, non l'orientamento generale dell'osso nella scena 3D.



Utilizzo di Copy Rotation

Copy Scale è come *Copy Location* e *Rotation*, ma per lo scaling (il ridimensionamento di un osso). anche qui abbiamo il concetto di *Target* e *XYZ*, ma non quello di trasformazione locale o globale, perché lo scaling è, in un certo senso, sempre locale; anzi, tutte le ossa hanno sempre la stessa dimensione, ossia 1,1,1, quando sono nella loro *Rest Position*: non importa se un osso è lungo il doppio di un altro, le sue dimensioni in posizione di riposo saranno sempre 1,1,1.

Limit Distance serve a posizionare un osso entro, oltre o ad una distanza esatta da un altro osso, l'osso *Target*. Il valore della distanza va inserito in *Distance*, da considerare come il raggio di una sfera centrata nell'osso *Target*.

Scegliendo per *Clamp Region* la voce *Inside*, l'osso provvisto di tale *Constraint* potrà muoversi soltanto all'interno della regione definita dal vincolo, mentre con *Outside* dovrà mantenersi, per così dire, ad una distanza minima dall'osso target (distanza espressa sempre mediante il campo *Distance*) e con l'opzione *On Surface*, per finire, potrà muoversi solo sulla superficie di questa ipotetica sfera centrata nell'osso target e avente raggio uguale al valore inserito in *Distance*.

Limit Location serve per confinare un osso all'interno di una certa regione di spazio, non necessariamente un parallelepipedo: anche una retta o una superficie o, ancora, un punto limite.
Il tutto dipende da quale casella, tra *Min X*, *Min Y*, *Min Z* e i relativi *Max*, selezioneremo, anche nessuna o tutte, e dai valori che inseriremo per ciascuna casella.

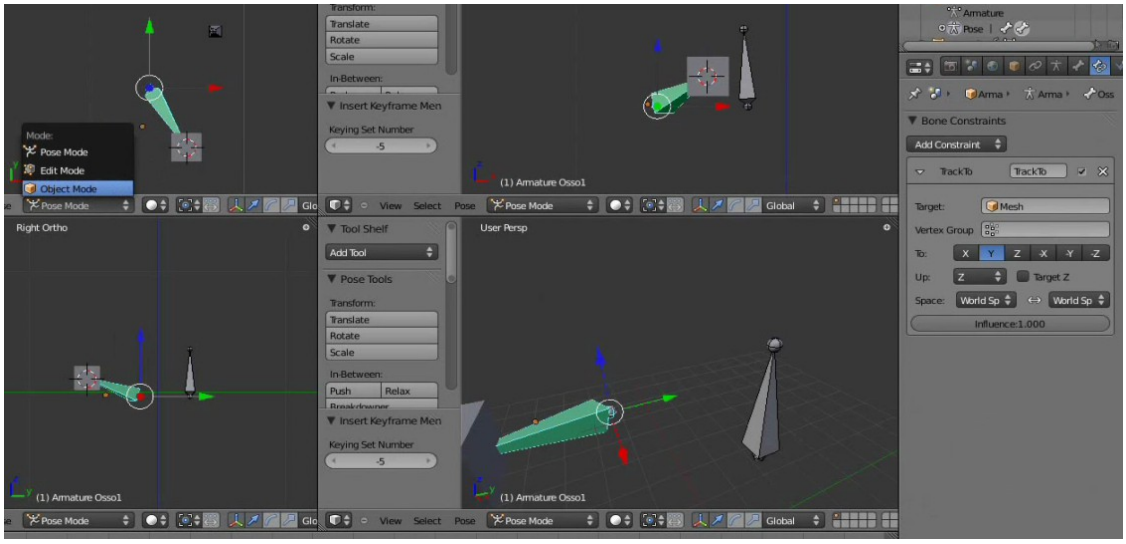
Limit Rotation e *Limit Scale* fanno per le rotazioni e per lo scaling quello che *Limit Location* fa per la posizione: limitano le trasformazioni; nel caso di *Limit Rotation*, è possibile inserire dei valori *Min* e *Max* per la rotazione intorno a ciascuno dei tre assi di rotazione, X, Y e Z, mentre nel caso di *Limit Scale* questo discorso varrà per il ridimensionamento dell'oggetto su ciascun asse.

> È interessante notare che, quando uno dei Constraint è attivo (ad esempio, *Limit Location*, ma il discorso vale in generale), Blender non visualizza l'effetto se andiamo "fuori scala", per così dire, ad esempio se spostiamo un osso al di là dei limiti imposti da *Limit Location*, ma memorizza internamente i valori della trasformazione, trasformazione che verrà applicata istantaneamente se decideremo di eliminare il *Constraint* che la limitava.

TRACK TO consente di far puntare ad un osso un oggetto, sia esso un altro osso dell'armatura che una mesh a parte, ad esempio un cubo che si sposta nello spazio 3D.

In *Track To* dovremo decidere, per ciascuna applicazione del *Constraint*, quale asse influenzare, cioè per quale direzione effettuare il "puntamento", per così dire, e se svolgere il tutto in coordinate globali o locali, sia per il target che per l'osso provvisto di tale *Constraint*.

Qualora decidessimo di far seguire ad un osso una mesh, poi, dovremo anche specificare un Vertex Group della mesh (eventualmente, creandone uno ad hoc, se non presente).



Utilizzo di Track To

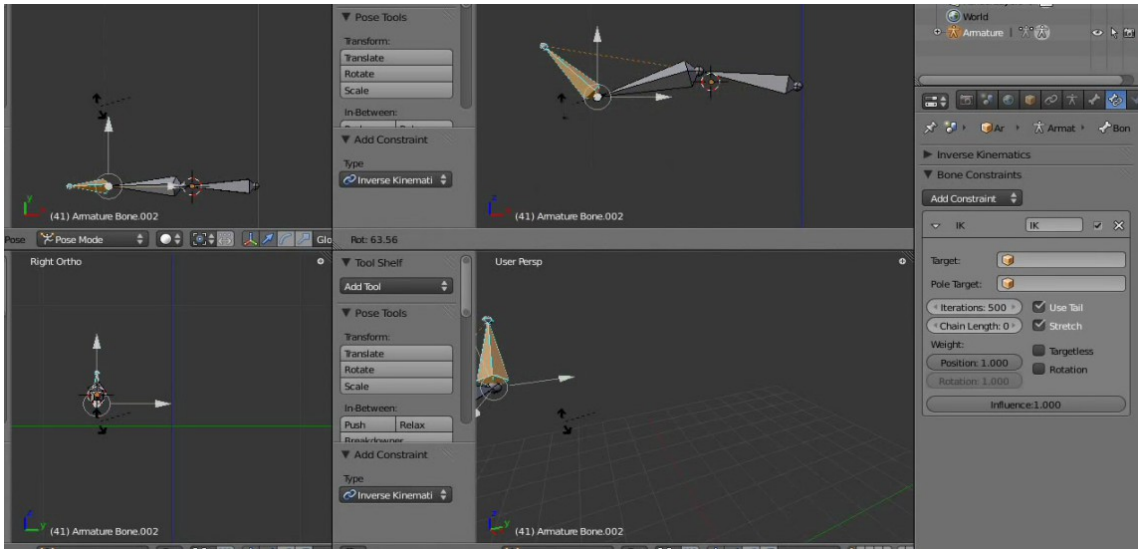
STRETCH TO è molto simile a *Track To*, con la differenza che qui l'osso viene deformato in lunghezza, a seconda della distanza presente, nel corso dell'animazione, tra se stesso e l'oggetto puntato.

ACTION serve per "mappare", per applicare un'azione (definita anche per oggetti esterni) ad un canale di trasformazione (cioè posizione, rotazione e scaling, per ciascun asse X, Y o Z) dell'osso, per un certo lasso di tempo definito mediante i campi *Start* ed *End* di *Action Length*.

CHILD OF lega un osso ad un altro oggetto, anche una mesh a parte (che però dovrà essere provvista di almeno un Vertex Group, da creare eventualmente per questo scopo). Si tratta del *Constraint* da utilizzare per far sì che l'osso si comporti esattamente come l'oggetto *Target*, ossia si sposti, ruoti e si ridimensioni in maniera locale seguendo le trasformazioni di un altro oggetto.

FLOOR serve per considerare degli oggetti mesh di tipo Planes come vincoli, ad esempio per creare soffitti, pavimenti o muri. L'oggetto Plane deve avere almeno un Vertex Group, per cui eventualmente bisognerà crearne uno per l'occasione.
L'opzione *Sticky* è quella che consente di implementare meglio un pavimento, o meglio un ciclo di camminata lungo un pavimento, perché rende l'oggetto immobile quando tocca il piano.

Per analizzare il Constraint Inverse Kinematics, ossia cinematica inversa, cancelliamo questa armatura dalla scena ed aggiungiamone una nuova, creando, a partire dal primo osso, un altro osso per estrusione e da questo un terzo osso, realizzando quindi una catena, come per le ossa del braccio o della gamba, ad esempio.



L'armatura da utilizzare per l'esempio con IK

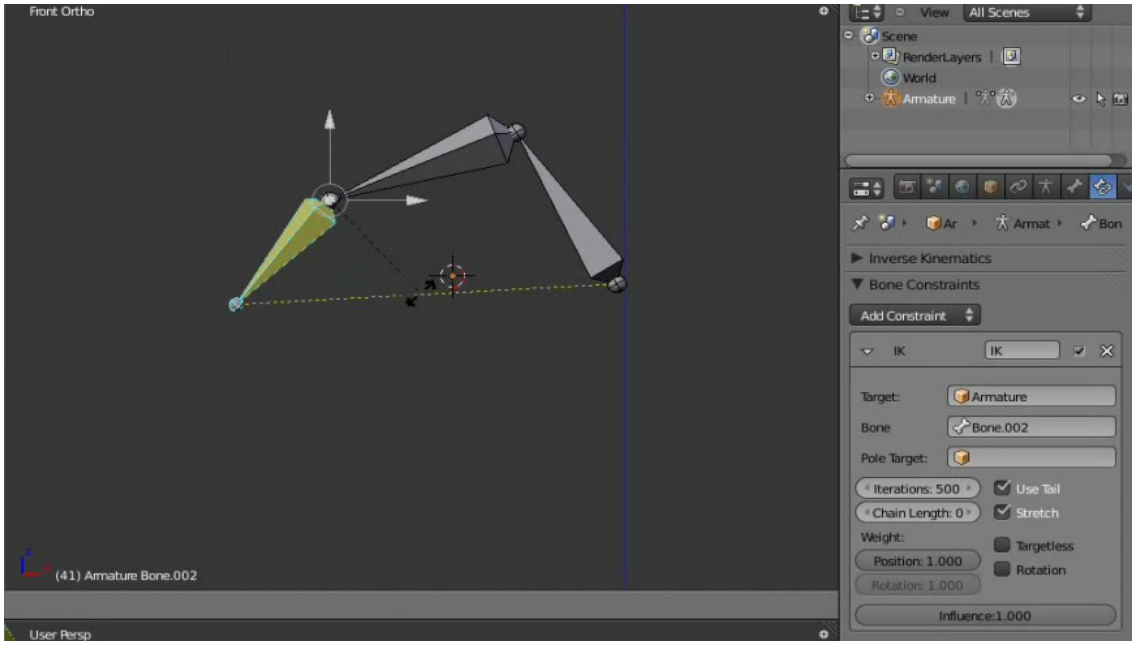
Applichiamo questo *Constraint* all'ultimo osso creato.
Aggiungendo un *Constraint* di questo tipo ad un osso, tale osso (e i suoi genitori, lungo la catena) diventeranno parte della *simulazione IK*, una simulazione dove una trasformazione effettuata su un osso, ad esempio una rotazione, non ha effetti solo sull'osso, ma sulla sua catena.
Si tratta di simulazioni utili per realizzare animazioni, ad esempio, delle gambe.

Come *Target*, impostiamo l'armatura che contiene l'osso; a questo punto dovremo inserire anche il nome dell'osso Target, che generalmente è l'osso stesso, ma può anche essere un altro osso, dipende dai casi;vi consiglio di fare varie prove per valutare i vari effetti.

Chain Length, ossia lunghezza della catena, di default ha valore 0, il che non significa che nessun osso fa parte della catena, anzi: significa che tutta la catena di ossa fa parte della catena *IK*, cosa che non sempre va bene (visto che, ad esempio, se dobbiamo alzare una mano non dobbiamo necessariamente girare la testa), per cui è meglio impostare un valore numerico maggiore di 0.

In genere, l'osso è collegato al capostipite della catena con una linea tratteggiata che parte dalla coda dell'osso, ma possiamo impostare il collegamento per la testa deselezionando la casella Tail.

Selezionando *Rotation*, la simulazione della cinematica inversa terrà conto anche della rotazione delle ossa.



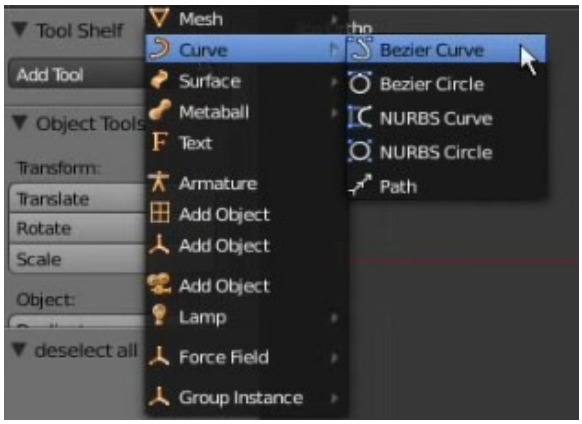
Esempio con IK

* * *

Modellazione e rendering - Bicchiere di vetro

In questo tutorial vedremo come creare un bicchiere di vetro in Blender 2.5, sia dal punto di vista della modellazione che del Material, per definirne l'aspetto.

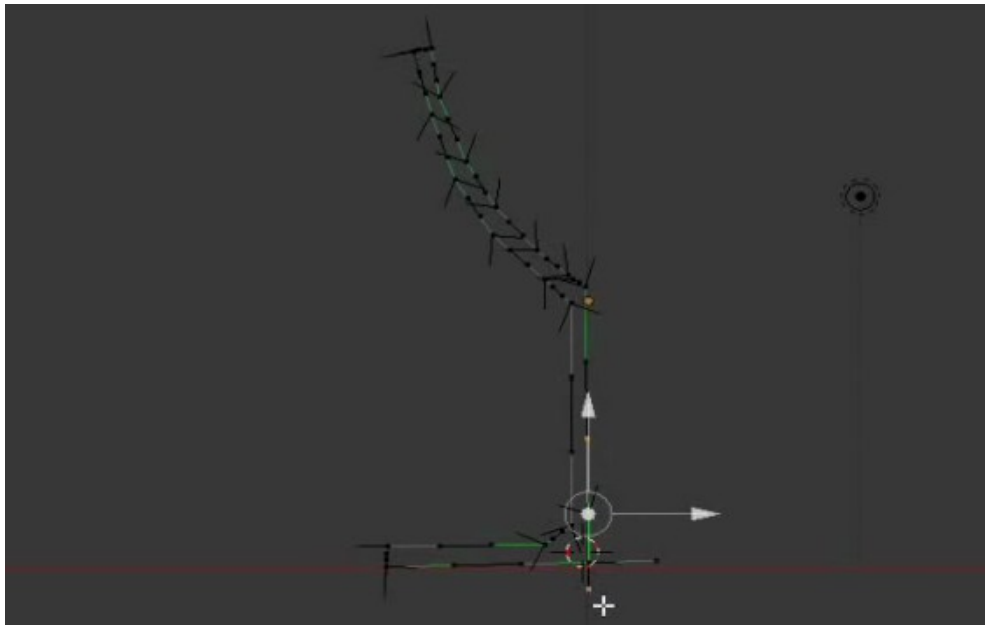
Avviamo Blender e cancelliamo, dalla scena di default, il cubo, mantenendo invece telecamera e fonte luminosa. Aggiungiamo ora una curva Bezier, ossia una curva composta da vertici che, come vedremo, possono essere di varia natura (con maniglie vettoriali o parametriche per la definizione della forma) e che ci servirà per definire il profilo del nostro bicchiere.



Aggiungere la curva Bezier alla scena

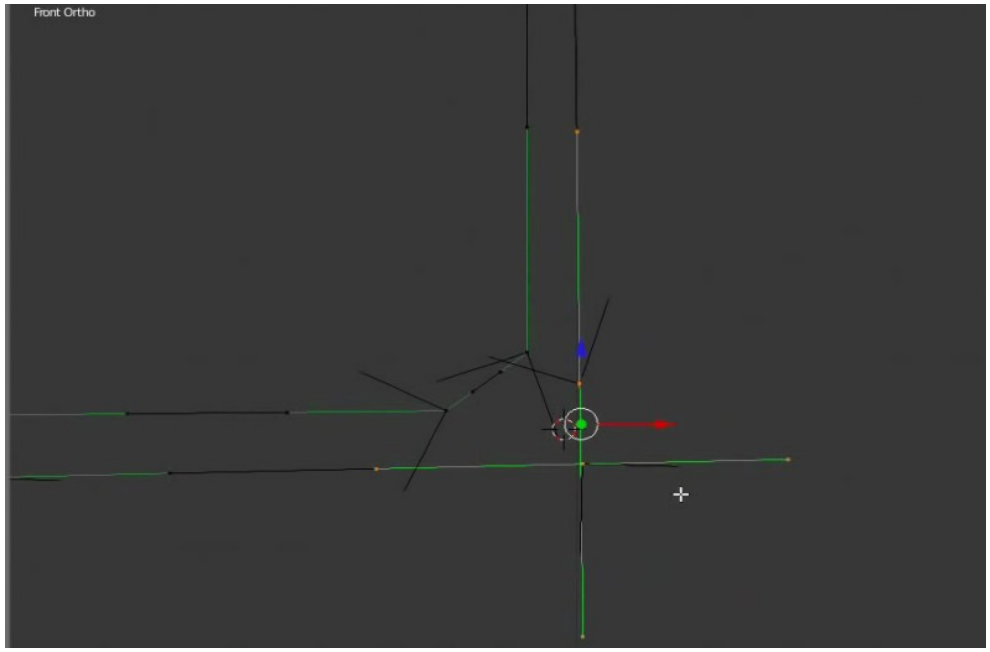
Qualora il *pivot* (il perno delle trasformazioni di scala, traslazione e rotazione) della curva non dovesse essere centrato nella stessa, centriamolo con *CTRL ALT SHIFT C*, scegliendo *Center New* dal menù che apparirà a video.

Disegniamo quindi uno "spicchio" del calice (un trecentosessantesimo, per così dire), partendo dalla forma base della curva Bezier ed estrudendo i vertici (con il tasto *E*) per crearne di nuovi da posizionare nello spazio 3D, oppure suddividendo alcuni segmenti già esistenti (selezionando i vertici che li compongono) con *W* e *Subdivide*.



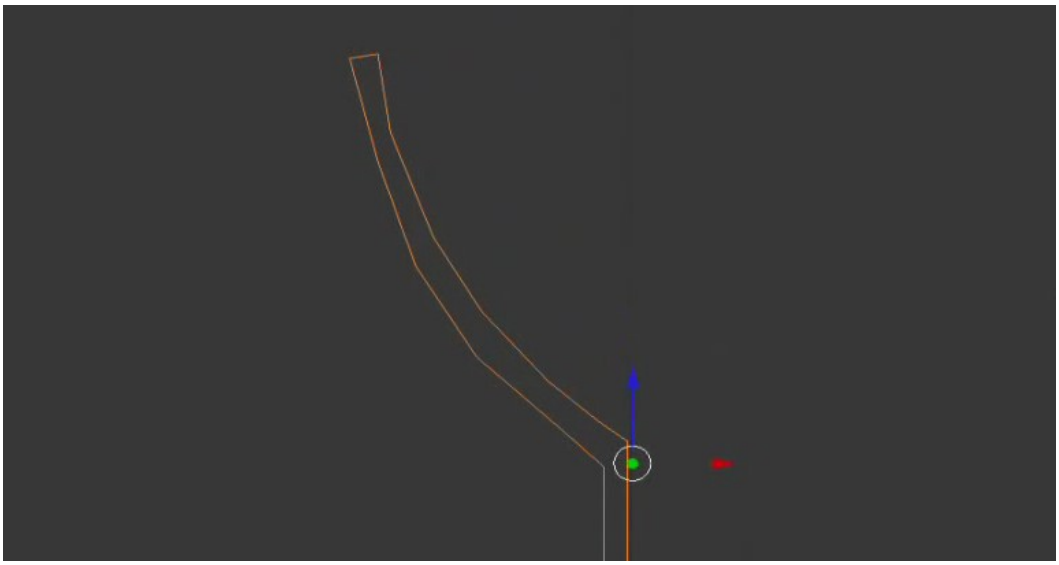
Disegnare lo "spicchio" del bicchiere

Possiamo ruotare, traslare e scalare uno o più vertici selezionandoli ed utilizzando o i tasti del mouse o, meglio ancora, i tasti *R* per Rotation, *G* per Grabbing (la traslazione) e *S* per Scale (il ridimensionamento); attenzione: per creare un calice vuoto dobbiamo dargli un minimo di spessore, non possiamo chiudere il bordo brutalmente collegandolo al vertice della base passando per l'asse centrale, perché tutta l'area racchiusa dalla superficie da noi definita diventerà, per via della rotazione, un volume, come se si trattasse di un birillo da bowling.



Altra fase della modellazione, ancora con tutte le maniglie Bezier

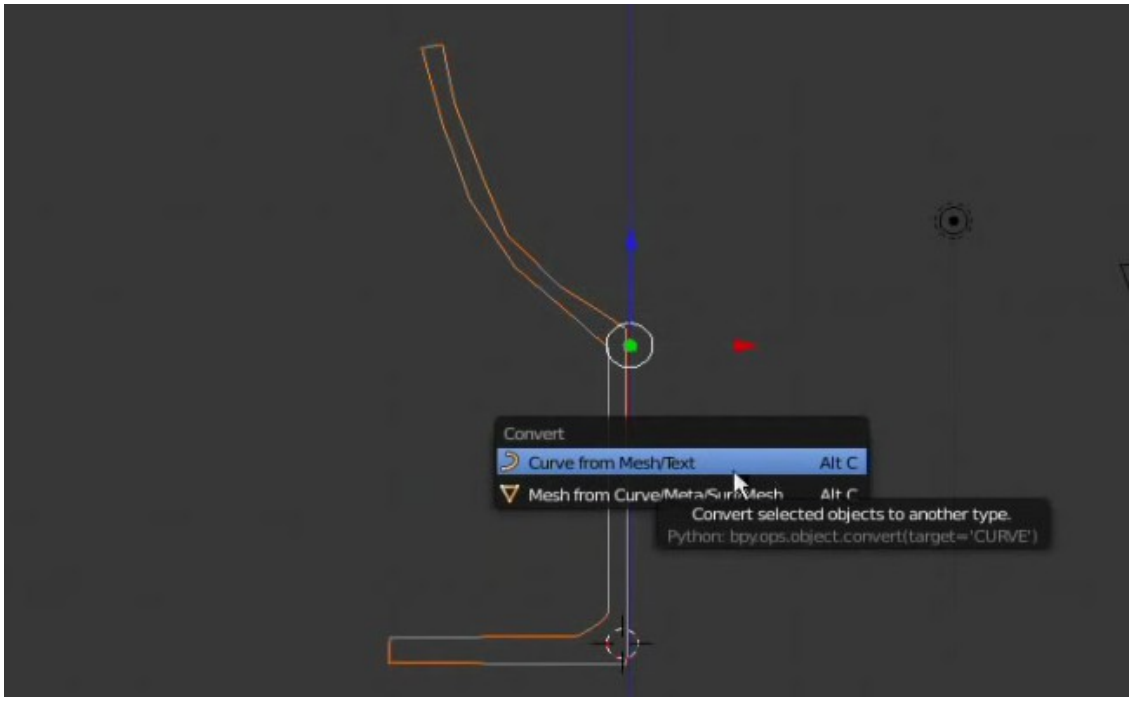
Le maniglie Bezier in genere consentono di definire molto facilmente delle curve, ma non sempre questo effetto è quello voluto, infatti per definire ad esempio la base o lo stelo del bicchiere potrebbero essere necessari vertici ad angolo, con spigoli rigidi, non smussati, anziché punti di curve, per cui potrebbe essere necessario cambiare un vertice in *Vettoriale*, selezionandolo e premendo *V*, oppure riportarlo alle condizioni standard con *H* o *A* [nota: nelle versioni più recenti di Blender, lo shortcut è sempre *V* e bisognerà poi scegliere, dal menù che apparirà a video, la modalità, ad esempio *Vector*, *Free*, *Aligned* o *Auto*].



Utilizzo di maniglie vettoriali

La definizione del profilo andrebbe fatta in una vista frontale o laterale, ma questo è solo un suggerimento per ottenere questo particolare modello dritto, non è una indicazione da seguire sempre. Quando il profilo è praticamente pronto e bisogna solo chiuderlo, selezioniamo i due vertici da unire (facendo attenzione a selezionare proprio i vertici, non alcune delle loro maniglie) e premiamo il tasto *C*, per *Connect*.

A questo punto il profilo è pronto e possiamo convertirlo in una mesh, passando in Object Mode e premendo *ALT C*, scegliendo quindi (dal menù che apparirà a video) la voce che ci consente di creare una mesh a partire da una curva.



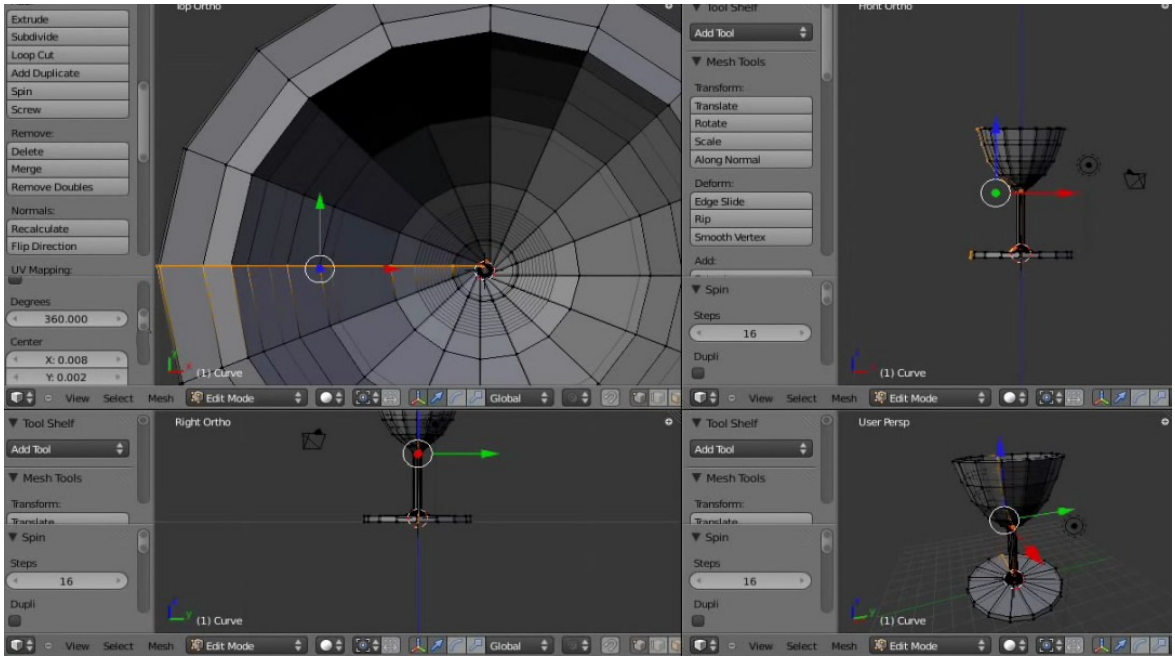
Conversione in mesh, con ALT C in Object Mode

Con tale mesh selezionata, torniamo in Edit Mode e selezioniamo tutti i vertici con *A*.

È giunto il momento di applicare il modificatore di rotoduplicazione, che in alcuni programmi è detto ad esempio *Lathe*, ossia tornio, ma che qui si chiama *Spin*.

Il pulsante che ci consente di utilizzare lo Spin per generare un solido si trova nella Tool Shelf, sulla sinistra in ogni finestra di navigazione 3D, ma dobbiamo scegliere con attenzione la finestra ove premere questo pulsante; se finora abbiamo definito il profilo in una vista frontale o laterale, ora dobbiamo applicare *Spin* in una vista dall'alto, in quanto l'asse perpendicolare al piano visibile nella vista ove viene applicato *Spin* coincide con l'asse di rotazione della trasformazione.

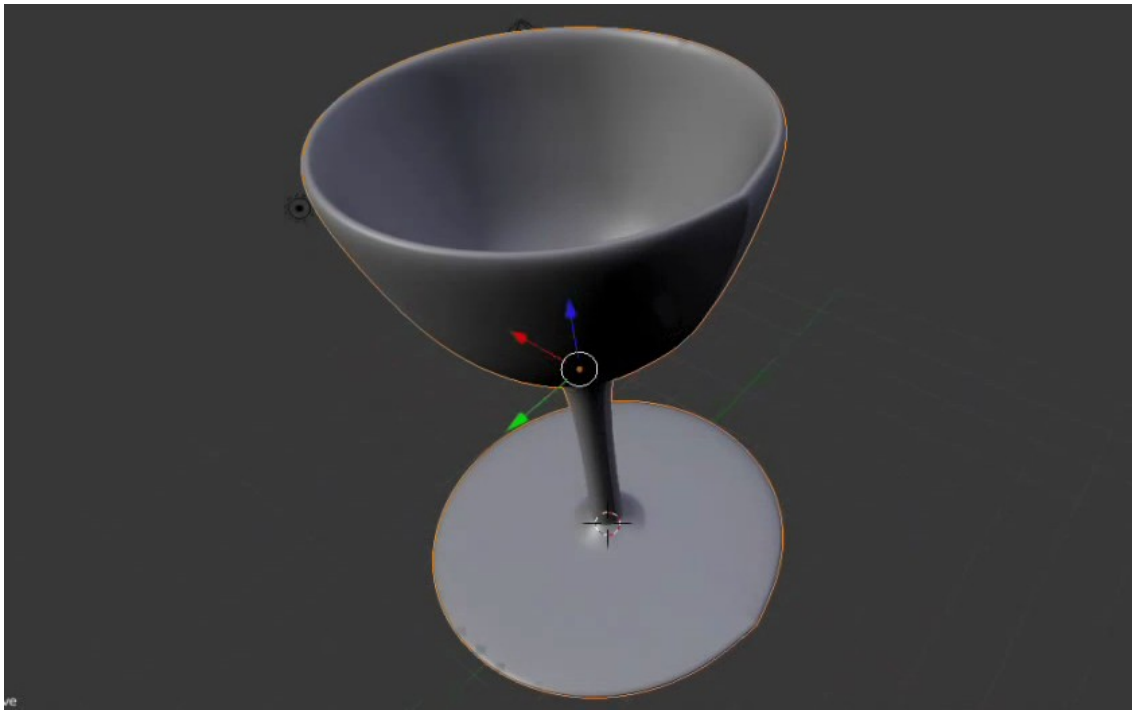
Clicchiamo quindi su *Spin* nella Tool Shelf della vista *TOP*.



Il risultato dopo aver cliccato su Spin, con tutti i vertici dello "spicchio" selezionati in Edit Mode e con l'oggetto inquadrato da una vista "dall'alto" in modalità ORTHO; inoltre, il 3D Cursor deve trovarsi nel centro di rotazione (ad esempio, alla base del profilo, in quello che sarà il punto centrale), altrimenti si avranno risultati non corretti

Inizialmente la trasformazione verrà applicata con valori di default 90 gradi, 9 *Steps* (cioè passi della rotazione, il numero di spicchi creati) e con un punto intorno al quale effettuare la rotazione che magari non coincide con il centro dell'oggetto, per cui mettiamo 360 gradi per fare un giro completo, *Steps* a seconda del grado di dettaglio che vogliamo, ad esempio 12, 16, 24 (le cose miglioreranno ancora, comunque, con *SubSurf*, come vedremo) ed impostiamo le coordinate del punto centrale; a tal proposito, consiglio di mettere il vertice che corrisponderà al centro della base del calice nell'origine del sistema di riferimento cartesiano della finestra di navigazione 3D ed impostare come punto di rotazione 0,0,0 (ossia l'origine, appunto).

Per smussare il modello, in Object Mode apriamo la scheda *Modifiers* nella Properties Window ed aggiungiamo un modificatore *Subdivision Surface* con 3 livelli per modellazione e rendering in modalità *Catmull-Clark*. Nella Tool Shelf di una qualsiasi finestra di vista 3D, in ogni caso, clicchiamo su *Smooth* nella sezione *Shading*.



Il risultato con le nuove impostazioni e SubSurface

È giunto il momento di assegnare un Material a tale oggetto, per simulare il vetro.

Non avremo bisogno di Textures ma di effetti di riflessione, *Mirror*, e trasparenze, in entrambi i casi di tipo *Raytrace*; procediamo comunque con ordine.

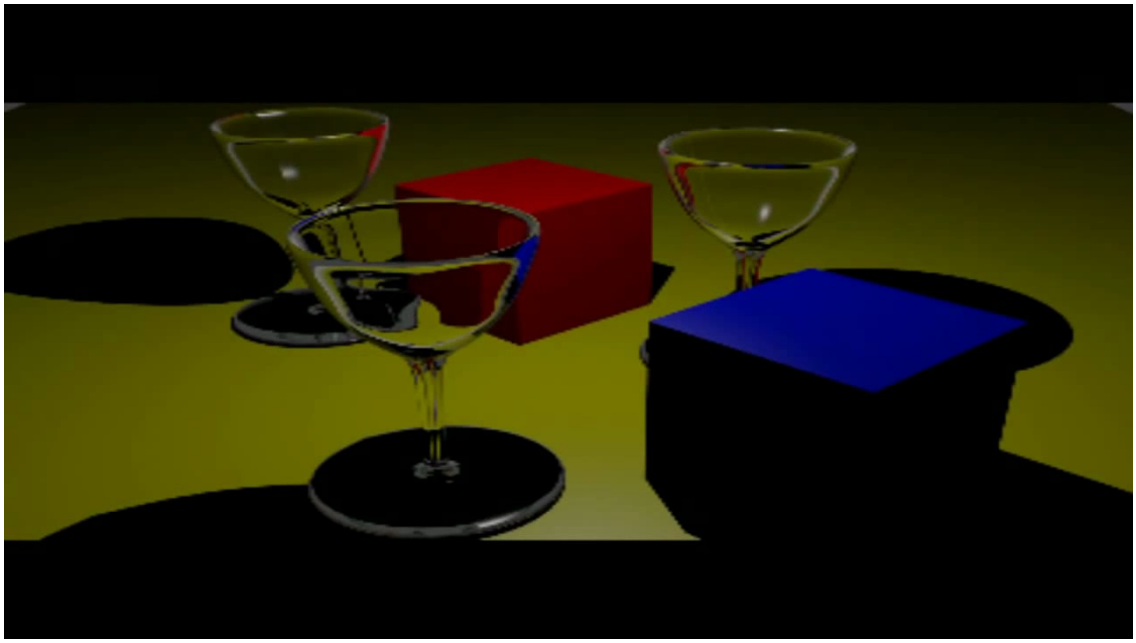
Come ombreggiatore diffuso lasciamo quello di default, ossia *Lambert* con intensità 0.8; possiamo lasciare anche il colore di default o utilizzarne un altro, a seconda dell'effetto che vogliamo ottenere, ad esempio bianco, blu, eccetera.

Come ombreggiatore speculare, andrà benissimo *CookTorr* con intensità 1 e *Hardness* a 475.

Attiaviamo la trasparenza *RayTransp*, impostando *IOR* (ossia l'indice di rifrazione del materiale) a 1.55, la profondità *Depth* a 7, il *Fresnel* a 5 e il valore speculare a 0.8.

Attiviamo anche i riflessi *RayMirror*, con valore di *Reflectivity* 0.15, *Fresnel* a 5 e *Depth* a 2.

Bene, l'oggetto e il Material sono pronti per essere inseriti nelle vostre scene e renderizzati!



Rendering d'esempio

* * *

I modificatori Array e Build

In questo tutorial prenderemo in esame i modificatori *Array* e *Build* per le mesh in Blender 2.5. Iniziamo con *Array*. *Array* consente di creare delle copie di un oggetto, disponendo eventualmente tali copie nello spazio in base a determinati valori di spostamento, detti *Offset*.

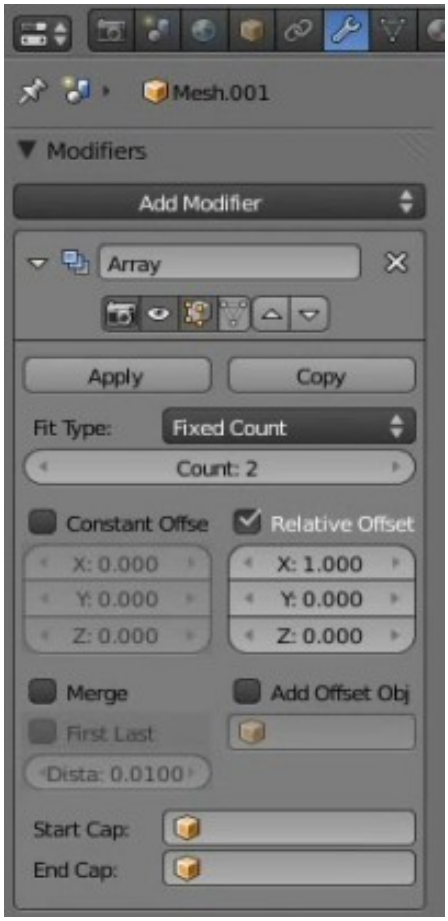
Va detto che, quando il modificatore non è stato ancora applicato (con un click su *Apply*), le copie create sono *istanze*, il che significa che ogni modifica applicata all'oggetto base (di riferimento) verrà applicata a tutte le copie; passando in Edit Mode, poi, vedremo solo l'oggetto base e potremo modificarne la struttura (anche queste modifiche avranno effetti sulle copie, se il modificatore non è stato ancora applicato).

Per creare oggetti dalle forme non banali conviene quindi modellare la mesh di riferimento, creare l'*Array* ed applicare il modificatore.

Una volta applicato, *Array* trasformerà l'originale e le copie in un unico oggetto vero e proprio, per cui passando in Edit Mode vedremo vertici e facce anche delle copie.

Queste caratteristiche rendono *Array* particolarmente utile per creare strutture con ripetizioni, come ad esempio ponti, catene, colonnati e strutture per così dire "frattali".

Esaminiamo la scheda di tale modificatore.



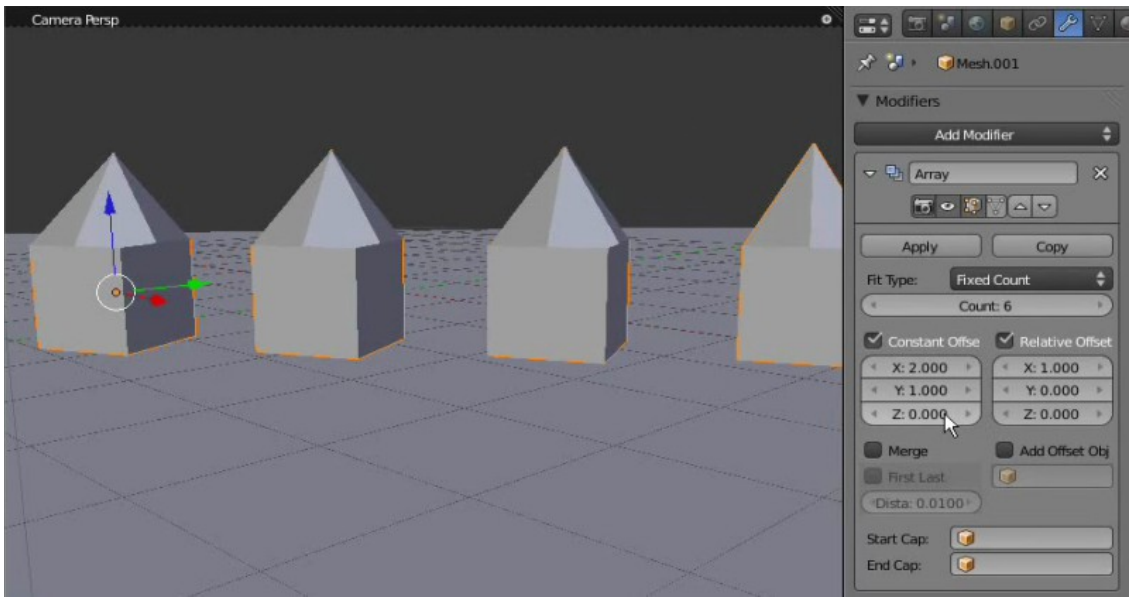
La scheda del modificatore Array

Il primo menù presente nella scheda contiene tre voci:

- *Fit To Curve Length* crea le copie disponendole su un percorso, definito da una curva; scegliendo tale opzione, apparirà la voce *OB:* un campo testuale ove inserire il nome dell'oggetto *Curve* da utilizzare come percorso.
- *Fixed Length* genera un numero di copie necessarie a coprire una determinata distanza; scegliendo tale opzione apparirà la voce *Length* (lunghezza): un campo dove andrà inserito un valore numerico per specificare la distanza da coprire;
- *Fixed Count* genera un determinato numero di copie, a seconda del valore inserito nel campo numerico *Count*.

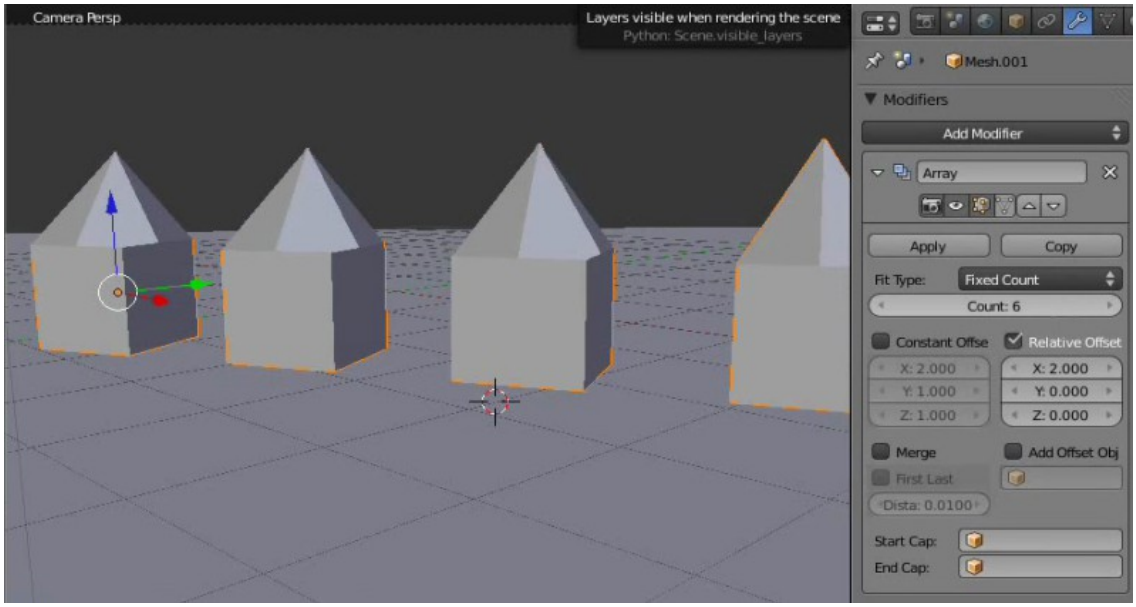
L'*Offset* è un valore che serve a distanziare tra loro le copie. Può essere di tipo *Constant*, *Relative* o *Object*, ma non si tratta di una scelta esclusiva, nel senso che è possibile attivare più opzioni contemporaneamente, sommando quindi gli effetti.

Nel primo caso, *Offset costante*, verranno sommati alla posizione degli oggetti i valori X, Y e Z specificati nei campi posti sotto a *Constant*.



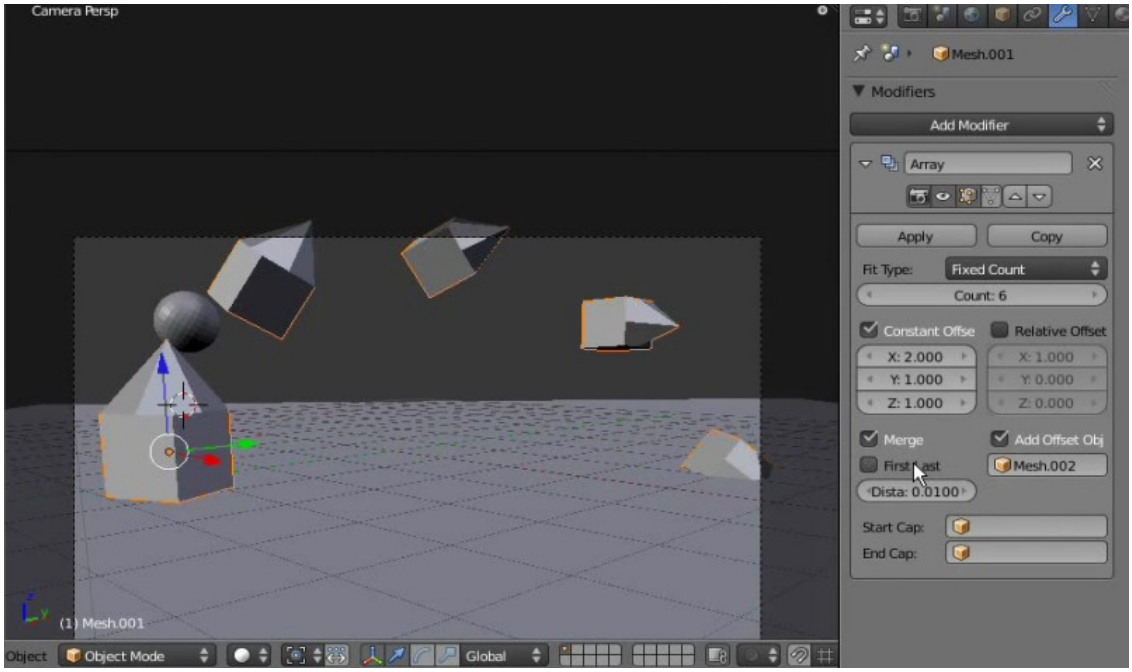
Esempio con Offset costante e relativo

Nel secondo caso, *Offset relativo*, l'*Offset* sarà pari alle dimensioni X,Y e Z del bounding box dell'oggetto (ed ecco perché si dice *relativo*) ma moltiplicate per dei fattori di scala specificati nei campi X, Y e Z.



Esempio col solo Offset relativo

Object Offset applica una trasformazione di offset calcolata a partire da un altro oggetto di riferimento, specificato attraverso il campo *Object*. Si tratta di una opzione particolarmente utile per inserire, ad esempio, offset di rotazione nelle copie da realizzare, infatti inserendo un oggetto, ad esempio una sfera, e ruotandola un po', Blender prenderà in considerazione il valore di orientamento della sfera ed applicherà le trasformazioni alle varie copie.

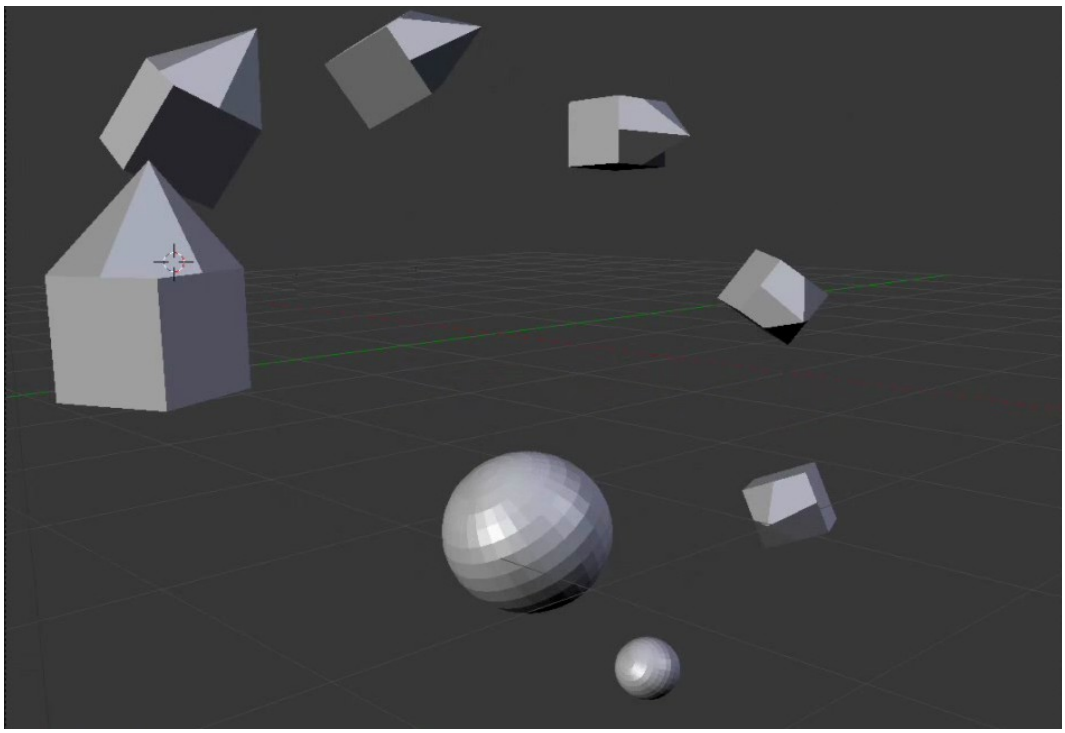


Esempio con Offset Object: la sfera è stata ruotata, per cui anche la disposizione delle varie copie nello spazio è "orientata" (una copia è ruotata rispetto alla precedente, seguendo la rotazione dell'oggetto di riferimento)

Merge, se attivo, imporrà a Blender di fondere insieme i vertici di una copia molto vicini ai vertici della copia successiva, con un valore di distanza specificato nel campo *Distance*, che è quindi una soglia che indica il valore di distanza al di sotto del quale verranno fusi i vertici vicini.

First Last effettuerà la fusione "forzata" tra i vertici della prima copia e quelli dell'ultima. Si tratta di un'opzione utile quando è necessario modellare degli oggetti circolari definendo prima uno "spicchio", una porzione, ed applicando poi *Array* con un offset di rotazione.

Start Cap ed *End Cap* aggiungono delle copie di altri oggetti (da specificare nei due campi) all'inizio e alla fine dell'insieme di oggetti creati da *Array*.
Quando poi verrà applicato il modificatore Array, questi due oggetti (copie di altri oggetti presenti nella scena) diverranno parte del nuovo oggetto creato, infatti passando in Edit Mode sarà possibile selezionarne i vertici e modificarli.



Esempio di utilizzo di Start ed End Cap (utilizzando la sfera per entrambi, per cui abbiamo appunto le sfere all'inizio e alla fine della serie)

Passiamo adesso ad esaminare il *Modifier Build*.
Build è un modificatore un po' particolare: applicato ad una mesh, genera un'animazione nella quale si parte da un ridotto numero di vertici e si ricostruisce faccia per faccia la mesh originale.

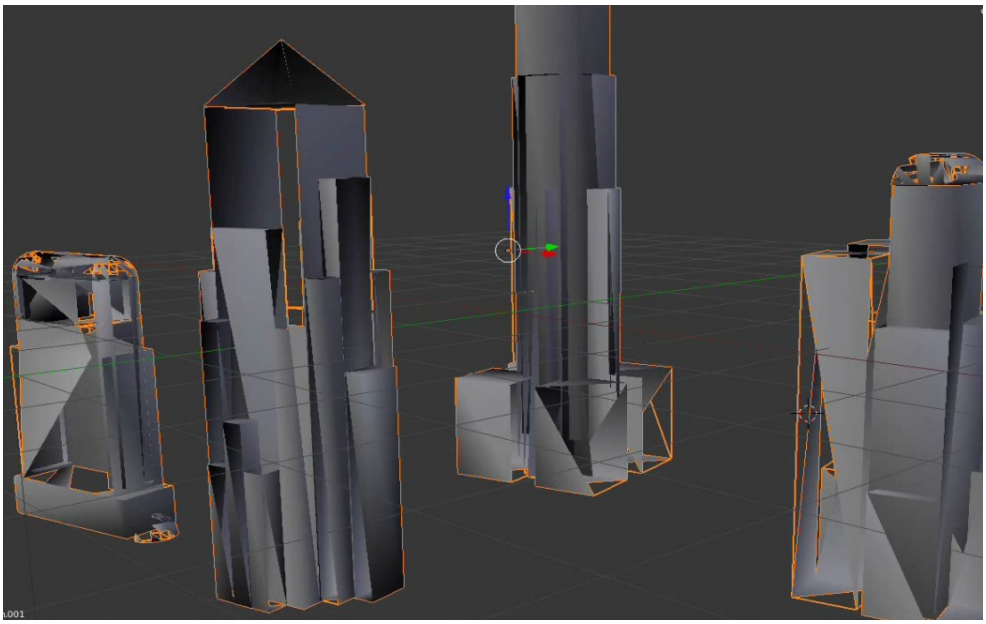
Si tratta di un modificatore che può tornare utile, quindi, per effetti speciali o presentazioni di strutture architettoniche o ingegneristiche.

Le voci presenti nella scheda di tale modificatore sono davvero poche e tutte dal significato intuitivo:

La scheda del modificatore Build

- *Start* indica il frame di inizio animazione, ossia quando applicare l'effetto;
- *Length* indica la durata dell'animazione in numero di frames:
- *Randomize* inserisce un po' di casualità nell'ordine con il quale vengono inserite le varie facce, per rendere il tutto meno ordinato;
- il parametro *Seed*, cioè letteralmente "seme", serve a specificare quale sequenza pseudo-casuale utilizzare; le sequenze generate con *Randomize*, infatti, non sono proprio casuali, ma seguono determinati *pattern* e vengono identificate mediante *Seed*.

È sufficiente premere *ALT A*, dopo aver selezionato tale modificatore nella scheda dei modificatori di una mesh, per vedere un'anteprima dell'esecuzione in una finestra di navigazione 3D, oppure si potrà lanciare direttamente il rendering dell'animazione.


Screenshot dell'animazione realizzata da Build

* * *

Consigli e strumenti per la modellazione di ambienti 3D

Questo più che un tutorial è un articolo con qualche trucchetto e qualche consiglio per affrontare facilmente la modellazione di un ambiente tridimensionale definendo, quindi, muri, pareti (con spazi per porte e finestre), soffitto e pavimento.

LA SCALA DELLA SCENA

Iniziamo dalla scala della scena: quando si lavora, una unità di Blender dovrebbe corrispondere ad un metro nel mondo reale.

Considerare una unità di Blender come equivalente ad un metro non è utile solo per la modellazione, ma anche per il rendering, se fate uso di motori di rendering esterni per Blender come *LuxRender* o *Indigo*, in quanto tali motori tengono in considerazione la scala della scena ed utilizzano, come scala di default, proprio il metro.

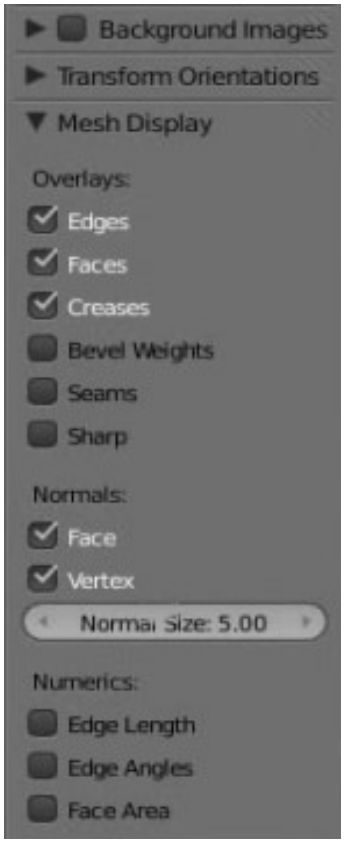
La Tranform Window

MODALITÀ PARAMETRICA e TRANSFORM WINDOW

La modellazione e il posizionamento, per questo tipo di progetti, dovrebbero essere svolti in modalità parametrica, ossia specificando i valori numerici di dimensioni, distanze, eccetera; per questo motivo, la finestra *Transform Window (Panel)*, richiamabile con *N* in una finestra di vista 3D, diventerà la vostra migliore amica: lì troverete, tra le altre cose, le voci *Location*, *Rotation*, *Scale* e *Dimension*, per i vari canali X, Y e Z.

Il significato e l'utilizzo di questi strumenti sono intuitivi.

VOCI DELLA TRANSFORM WINDOW – NORMALS, NUMERICS, GREASE PENCIL



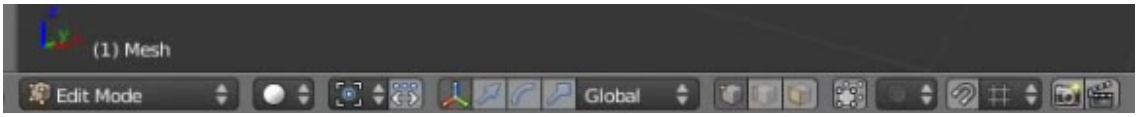
Altre voci presenti nella Transform Window

Nella *Transform Window* sono presenti altre voci davvero utili, disponibili per le mesh in Edit Mode; ad esempio:

- alla voce *Normals*, nella sezione *Mesh Display*, troviamo i pulsanti per la visualizzazione delle Normali di facce (con *Face*) e vertici (con *Vertex*), più un pulsante numerico (*Normal Size*) per definire quanto dovranno essere lunghi i segmenti rappresentanti le Normali;
- sempre in *Mesh Display*, alla voce *Numerics*, troviamo i pulsanti *Edge Length*, *Edge Angles* e *Face Area*: se attivati, mostreranno a video, rispettivamente, la lunghezza degli spigoli, gli angoli formati dagli spigoli e il valore della superficie o area di una faccia;
- in fondo alla scheda, vi è la voce di gestione dei livelli di *Grease Pencil*, la funzionalità che vi consente (cliccando su *Draw*, *Line* ed *Erase* (cioè disegna liberamente, disegna una linea e cancella) nella sezione *Grease Pencil* della Tool Shelf) di disegnare degli elementi letteralmente "sulla vista 3D", ad esempio per annotare qualcosa o mettere qualche indicazione. Si tratta di segni che non vengono renderizzati. Una volta entrati in modalità *Grease Pencil*, appariranno altri strumenti in una apposita sezione della Tool Shelf, in basso, ma questo è un altro argomento.

SELEZIONE VERTICI, SPIGOLI, FACCE

In Edit Mode, di default la selezione avviene a livello di vertici, ma è possibile passare in modalità selezione spigolo o faccia con i pulsanti *Edge Select Mode* o *Face Select Mode*, posti nell'header di ogni finestra di visualizzazione 3D.



L'header della 3D View, in Edit Mode, presenta gli strumenti per la selezione di vertici, spigoli e facce; in più, abbiamo la casella "Limit Selection to Visible" e i pulsanti relativi allo Snap

LIMIT SELECTION TO VISIBLE

Sempre nell'header delle finestre di vista 3D si trova un altro pulsante interessante per la selezione in Edit Mode: il pulsante *Limit Selection to Visible*. Come suggerisce il nome, attivando questa opzione, quando selezioneremo un gruppo di elementi (ad esempio un insieme di vertici con *B* e il mouse, per definire l'area della selezione) selezioneremo solo quelli direttamente visibili, senza selezionare quelli che si trovano dall'altra parte dell'oggetto o che sono, comunque, nascosti.

SNAP TO

Per la progettazione di strutture architettoniche o di ambienti, conviene poi attivare la modalità di *Snap*, per "agganciare" (legare, vincolare) il cursore del mouse a qualche elemento, ad esempio ai punti della griglia o ai vertici dell'oggetto. Lo *Snap* può essere attivato sia in Edit Mode che in Object Mode con *SHIFT TAB* o con il pulsante la cui immagine raffigura una calamita nell'header delle finestre di vista 3D.

È possibile impostare diverse modalità di *Snapping*, ad esempio per effettuare lo *Snap* rispetto alla griglia con *Increment* o ad altri elementi, come ad uno spigolo con *Edge*, oppure scegliere quale target agganciare con il menù *Snap Target*, contenente le voci *CLOSEST*, *CENTER*, *MEDIAN* e *ACTIVE*, tutte dai significati intuitivi.

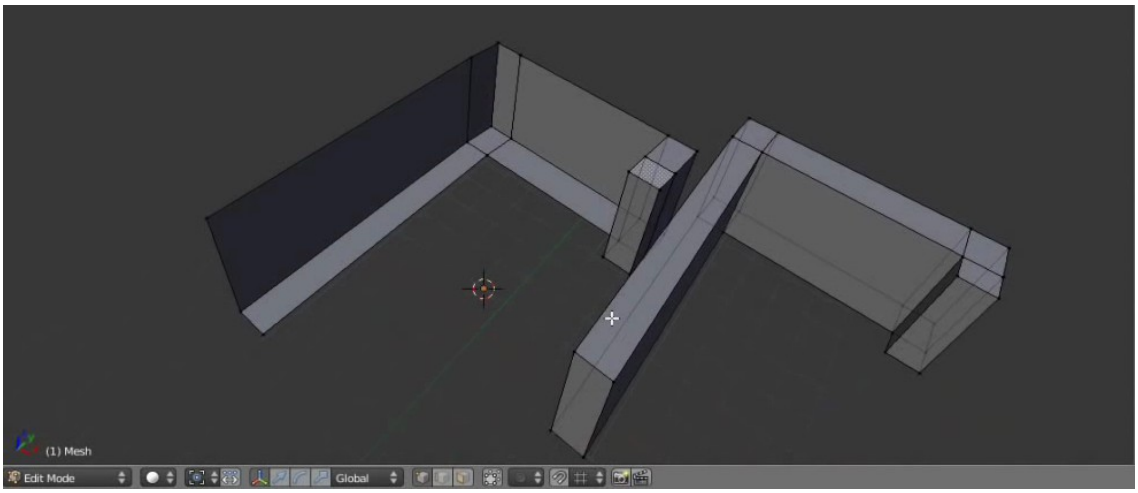
Alcune opzioni dello SNAP

MODELLAZIONE: CONSIGLI

Passiamo ora alla modellazione degli elementi.

Il pavimento è in genere un oggetto Plane, un piano, non c'è molto da dire su questo: può essere ridimensionato spostando i vertici in Edit Mode o, meglio ancora, definendo le sue dimensioni a livello di oggetto mediante le apposite voci presenti nel pannello *Transform Properties*.

Il soffitto viene realizzato, banalmente, duplicando in modo oggetto (per creare due mesh distinte) il pavimento, tuttavia bisognerà poi ribaltarlo (con Mirror, CTRL M e scegliendo l'asse di simmetria) o andare in Edit Mode e cliccare su *Flip Direction*, cioè "inverti direzione", sotto l'etichetta *Normals* nella Tool Shelf. Per definire le pareti, se si ha una pianta della struttura su file (ad esempio JPEG o PNG), si può impostare tale immagine come immagine di sfondo nella finestra di navigazione 3D, aprendo il menù *Transform* con N ed attivando la voce *Background Image*, cioè "immagine di sfondo"; a quel punto, bisognerà caricare l'immagine da disco e posizionarla e ridimensionarla, a seconda della scala della scena, nella finestra di navigazione 3D, mediante i pulsanti messi a disposizione da Background Image. Si consiglia, quindi, di passare alla modalità di visualizzazione wireframe, attraverso la relativa voce nel menù Viewport Shading. Il modo più semplice per definire un muro consiste nel creare un Plane ed effettuare estrusioni degli spigoli con *E* e, se necessario, delle suddivisioni, per definire la pianta, per poi selezionare il tutto ed estrudere verso l'alto (per creare il volume, le pareti vere e proprie), sempre con il tasto *E*, ma in altezza.



Consigli su pavimento, soffitto e pareti

JOIN (UNIONE) DI MESH

È possibile unire più mesh in un unico oggetto selezionandole e premendo *CTRL J*, per *Join*, cioè appunto unione, fusione.

REMOVE DOUBLES, RECALC NORMALS, MERGE, CANC

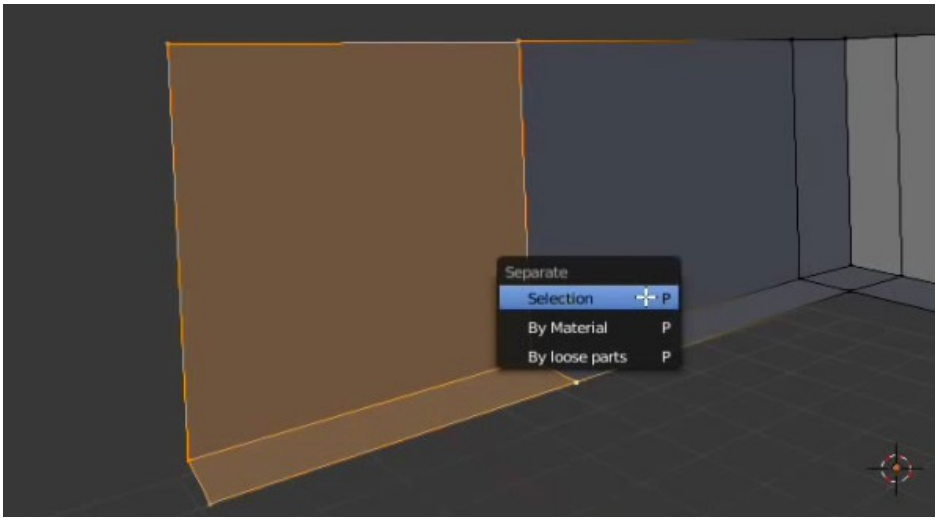
Una volta effettuato un *Join*, potrebbe essere necessario rimuovere alcuni vertici duplicati o troppo vicini (fondendoli) o ricalcolare le Normali; queste operazioni vanno fatte in Edit Mode, selezionando i vertici da considerare (anche tutto il modello, premendo *A*) e cliccando sulle voci *Remove Doubles* (ed in tal caso bisognerà impostare la soglia d'azione) o *Recalculate Normals*, entrambe situate nella Tool Shelf in ogni finestra di navigazione 3D. I vertici selezionati possono essere anche fusi con *Merge* (sempre nella Tool Shelf) o eliminati premendo *CANC* o *X*.

SUDDIVISIONI E TAGLI: SUBDIVIDE, KNIFE, LOOPCUT

Se vogliamo creare una nuova divisione all'interno di una faccia, ad esempio perché dobbiamo inserire lo spazio per una finestra o una porta, possiamo ricorrere a vari strumenti di suddivisione, come il semplice *Subdivide*, il coltello o taglierino *Knife* (premendo il tasto *K*) o *Loopcut* (con *CTRL R*).

MATERIALI MULTIPLI E SPLIT DI MESH

Anche se è possibile assegnare dei materiali multipli ad una mesh, per definire l'aspetto di zone differenti mediante materiali diversi è possibile anche separare fisicamente una selezione da un oggetto, creando un nuovo oggetto distinto, con una operazione detta a volte "mesh splitting" (da *to split*, separare); tale operazione va fatta selezionando, in Edit Mode, i vertici e le facce da staccare dall'oggetto originale e premendo quindi *P*, selezionando infine la voce *Selection* dal menù *Separate* che apparirà a video.



Separare parti di mesh con P - Separate

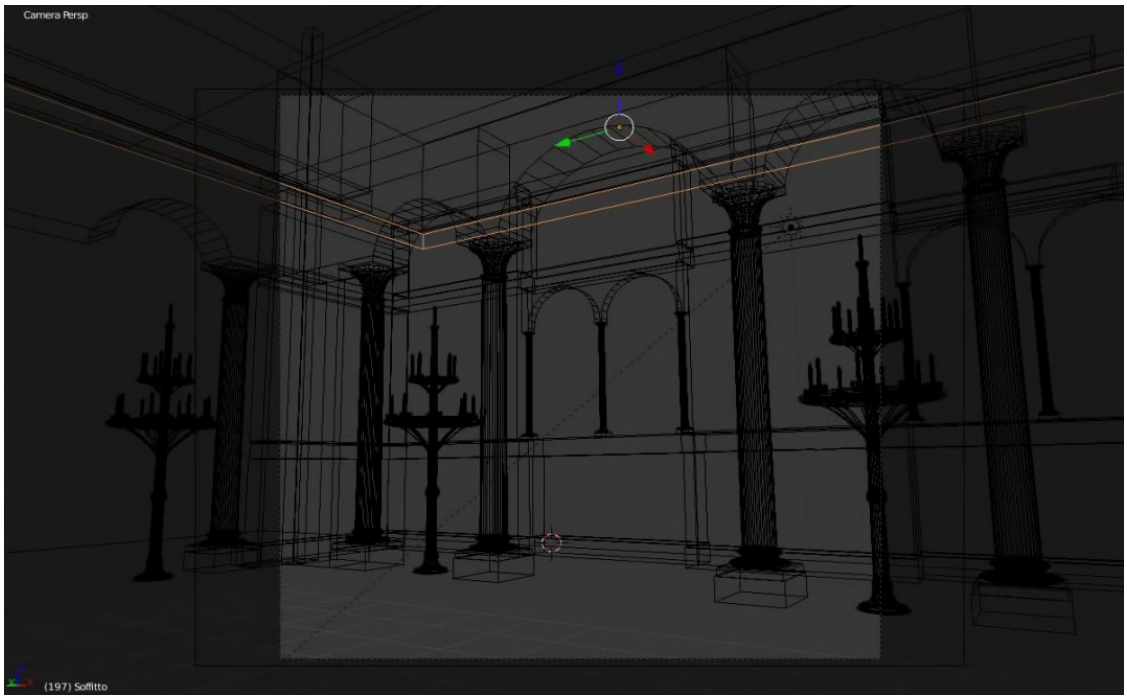
* * *

Descrizione scena Lapislazzuli di RedBaron85

In questo articolo prenderemo in esame una scena da me realizzata tempo fa mediante Blender 2.5, esaminandone le impostazioni di illuminazione, i Materials e le Textures. Il rendering, visibile nell'immagine qui sotto ha richiesto circa 3 ore per una risoluzione in effetti eccessiva, ossia 1800x1800, su un Core 2 Duo. Tutto è giustificato dall'utilizzo di *Ambient Occlusion* per l'illuminazione e, soprattutto, dall'uso intensivo dell'effetto *SSS*, cioè *Sub Surface Scattering*, fondamentale per ottenere effetti fotorealistici ma micidiale a livello di "peso" in fase di rendering.

Il rendering della scena

Andiamo con ordine, parlando per prima cosa delle mesh presenti: in realtà, si tratta più che altro di forme di base, per realizzare pavimento, soffitto, colonne, pareti; i candelabri e le candele sono stati realizzati in file a parte, con pochi passaggi, e poi importati in questa scena. Gli unici oggetti davvero complicati (si fa per dire) sono i capitelli, per il resto si tratta davvero di elementi geometrici di base.



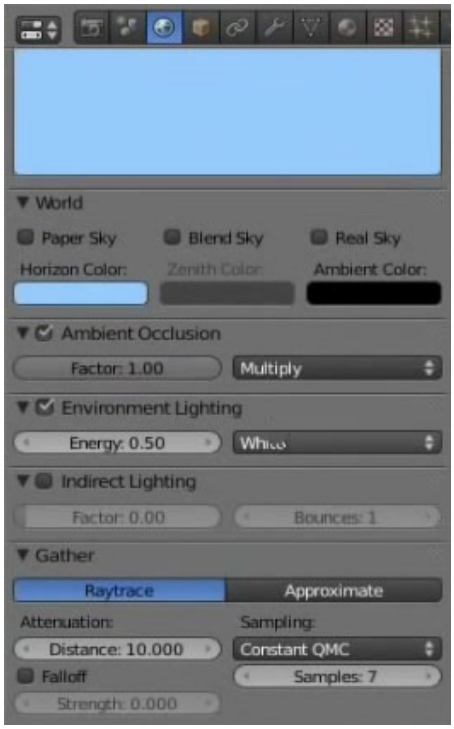
Panoramica degli oggetti presenti nella scena

Passiamo alle impostazioni del mondo, *World*.

Il colore del cielo, visibile dalle finestre, è celeste; non sono state utilizzate Textures per lo sfondo. *Ambient Occlusion* e la *Luce Ambientale* sono attivati; nella sezione *Gather*, abbiamo energia 0.5, modalità *Raytrace* con attenuazione su distanza 10, *Sampling QMC* costante e 7 campionamenti.

È presente una fonte di luce, con i raggi che passano dalla finestra; tale fonte di luce è di tipo *Sun*, il Sole, con colore bianco puro ed energia 2. Il Sole proietta, qui, ombre di tipo *Ray Shadow*, con tecnica di *Sampling* adattiva, 1 campionamento.

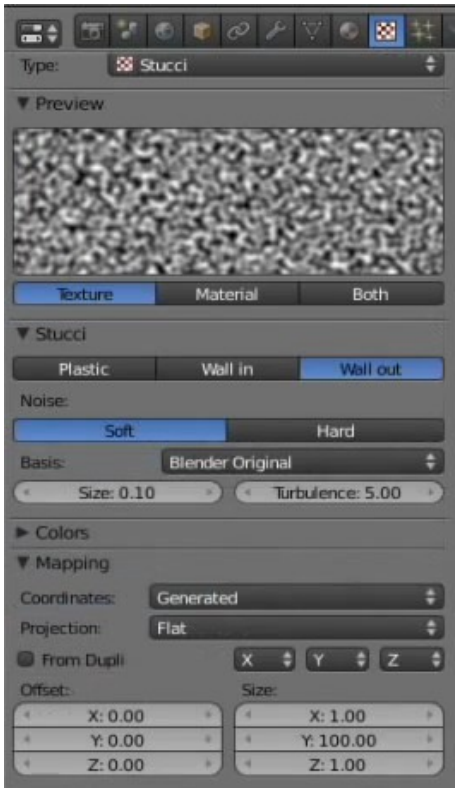
Esaminiamo ora i pochi Materials presenti nella scena.



Impostazioni World

Per i capitelli, i candelabri, il soffitto e certe zone delle pareti è stato fatto uso di un Material creato per raffigurare superfici dorate. Questo Material fa uso dell'effetto *Mirror* (per i riflessi) e di una Texture, di tipo *Stucci*, mappata solo sul canale delle Normali, per creare un minimo di effetto *Bump*; da notare, a proposito di questa Texture, che la dimensione sull'asse Y è 100, mentre per gli altri è 1, il che significa che non verranno prodotti i rilievi puntiformi di *Stucci* ma delle strisce, in quanto stiamo "stirando" la Texture rispetto ad una coordinata.

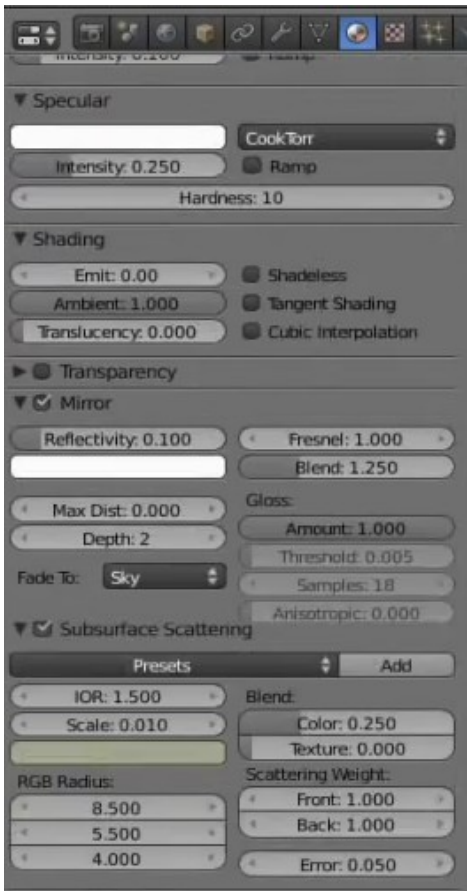
La parte centrale della parate sul fondo fa uso di un Material avorio, realizzato senza ricorrere ad effetti "pesanti" come Trasparenza, *Mirror* o *Sub Surface Scattering*, ma solo alle impostazioni degli ombreggiatori diffuso e speculare (impostando, inoltre, la *Translucentezza* a 1, in *Shading*), più una Texture, di tipo *Stucci* e mappata solo sul canale *Normal*, per creare un minimo di *Bumping*.



La "stiratura" della Texture Stucci

Il "cuore" della scena è rappresentato, comunque, dal Material "granito" utilizzato per il corpo centrale delle colonne, la base delle stesse e il pavimento, dove si fa uso dell'effetto *Mirror* e, soprattutto, di *Sub Surface Scattering*, con indice di rifrazione *IOR* 1.5, colore 0.38, 0.38, 0.26, raggio RGB 8.5, 5.5 e 4 ed infine *Blend Color* a 0.25.

In *Sub Surface Scattering* il valore di *Scale* dipende dalle dimensioni dell'oggetto nella scena; impostando *Scale* 1, ad esempio, 1 unità di Blender equivarrà ad un millimetro nella realtà, mentre impostando *Scale* 0.001 una unità di Blender equivarrà ad un metro nella realtà, e così via; in questo caso, *Scale* vale 0.01.

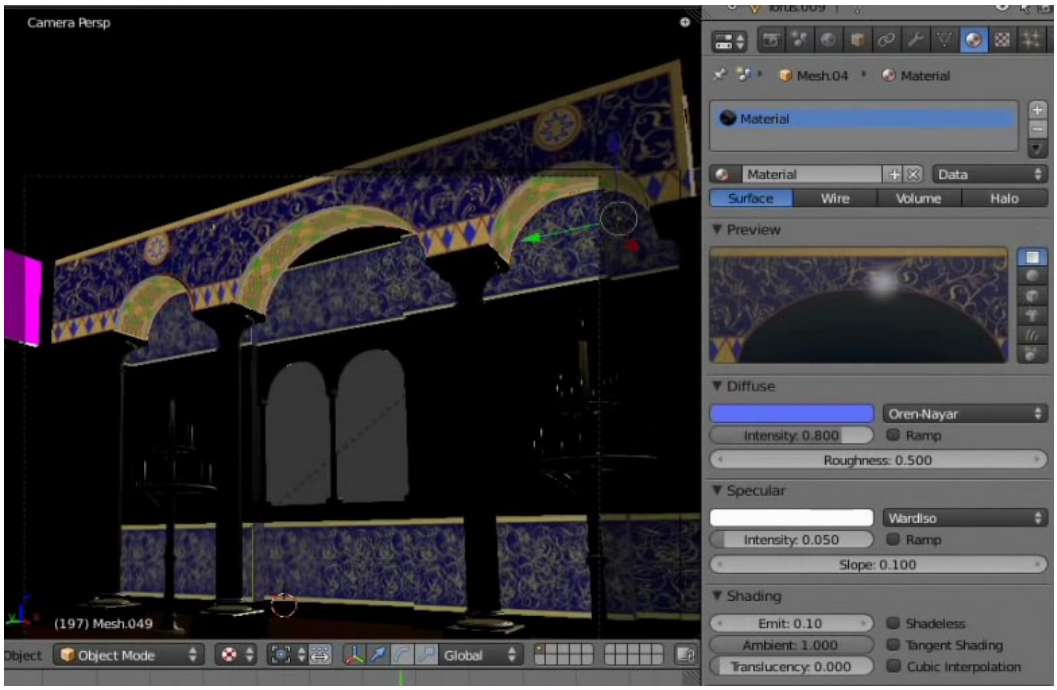


Il Material del granito

Per questo Material sono presenti, inoltre, tre Textures:

- la prima, di tipo *Clouds* (le nuvole), mappata solo sul canale *Color Diffuse*;
- la seconda, di tipo *Distorted Noise*, mappata sul canale *Color Diffuse* e sul canale *Ray Mirror*;
- la terza, di tipo *MusGrave,* mappata solo sul canale del colore *Diffuso*.

Le parti restanti della scena fanno uso di immagini utilizzate come Textures mappate con *UV Mapping*, come è possibile notare passando in modalità di visualizzazione *Textured*.



Le Textures immagini sono mappate con mappatura UV

Per la mappatura *UV* di questi elementi sono state realizzate due Textures ad hoc e le mesh sono state suddivise in *Vertex Groups* per applicare facilmente le immagini alle varie parti.

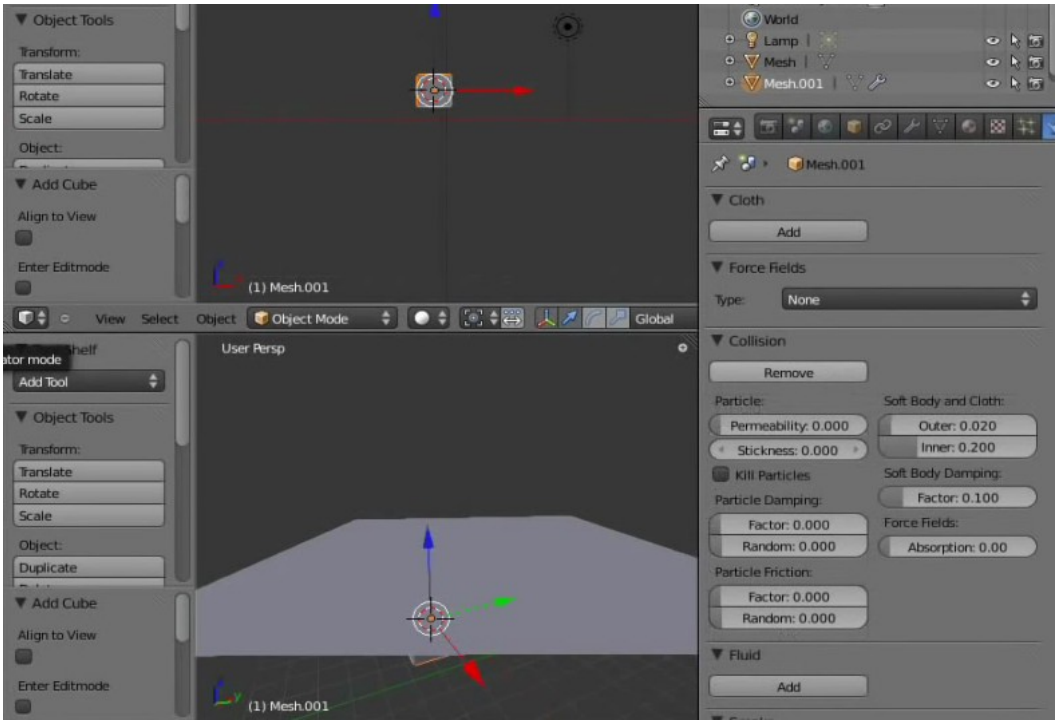
Non c'è solo questo, comunque: è presente anche una Texture di tipo *Stucci* mappata sui canali *Diffuse Color*, con colore RGB 0.07, 0.08 e 1, e *Geometry Normal*, per rendere la superficie meno omogenea; il Material utilizzato, inoltre, ha valore di *Emit* pari a 0.10 (per rendere più luminose le parti in oro, più che altro) ed effetto *Mirror* attivato per i riflessi, con valore di *Reflectivity* pari a 0.04 e *Fresnel* a 1.

* * *

Utilizzi del modificatore Solidify: pareti e tessuti

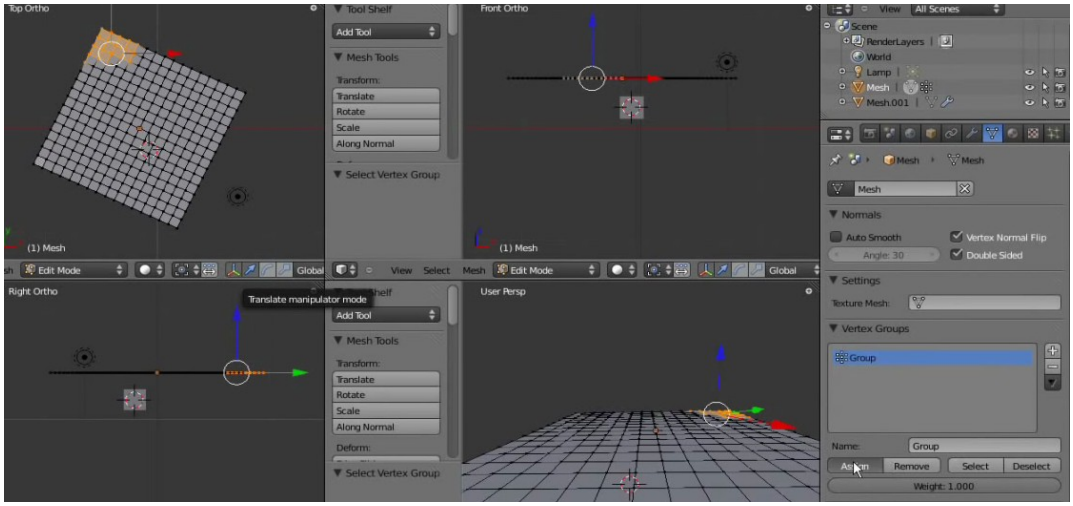
In questo tutorial, realizzato con la versione Alpha 2 di Blender 3D 2.5, vedremo due possibili utilizzi del modificatore *Solidify* di Blender. *Solidify* consente di aggiungere dello spessore alle facce degli oggetti; in pratica, effettua automaticamente delle estrusioni perpendicolarmente alle varie facce. *Solidify* può essere utilizzato in vari modi, ad esempio per aggiungere spessore ai planes utilizzati con il modificatore dei tessuti per realizzare tappeti, tende ed altri elementi che, altrimenti, risulterebbero troppo sottili e non realistici.

In questo tutorial farò uso del simulatore dei tessuti utilizzando anche il *Pinning* e un oggetto per le collisioni; per maggiori informazioni su tali argomenti, rimando ad un mio precedente tutorial (presente in questo volume) dedicato proprio a *CLOTH*. Aggiungiamo quindi un cubo alla scena, sotto il *Plane* già presente, ed attiviamo la simulazione *Collision* per questo oggetto.



Il cubo di default, con simulazione Collision

Selezioniamo il Plane, che deve avere un discreto numero di suddivisioni (anche se qui non ne sto facendo troppe, per non rallentare l'esecuzione), passiamo in Edit Mode e selezioniamo un gruppo di vertici in un angolo, creando un *Vertex Group* ad hoc per questa selezione.

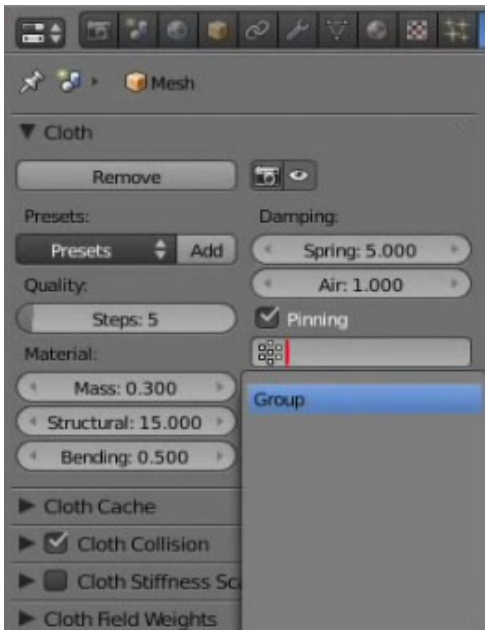


Il Vertex Group appena creato

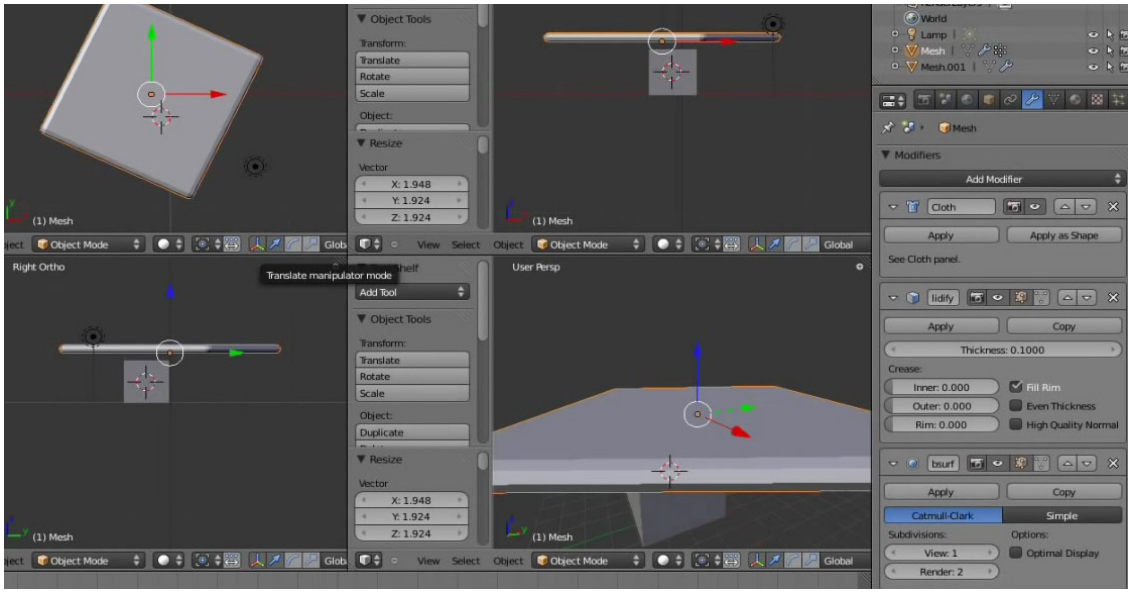
Attiviamo la simulazione *Cloth* per il Plane, selezioniamo la casella *Pinning* e scriviamo, nel campo *Vertex Group*, il nome del gruppo appena creato.

Adesso aggiungiamo il modificatore *Solidify* al Plane, utilizzando come spessore, ad esempio, 0.1 (forse un po' eccessivo, ma ci consentirà di valutare meglio l'effetto).

Prima di lanciare una simulazione definita con questi strumenti, comunque, conviene utilizzare anche il modificatore *Subdivision Surface* con metodo *Catmull-Clark*; ancora una volta, sto limitando le suddivisioni a 1 in fase di visualizzazione e 2 in fase di rendering per non rallentare troppo l'esecuzione, ma conviene utilizzare un valore come 3 o 4 almeno per la fase di rendering.

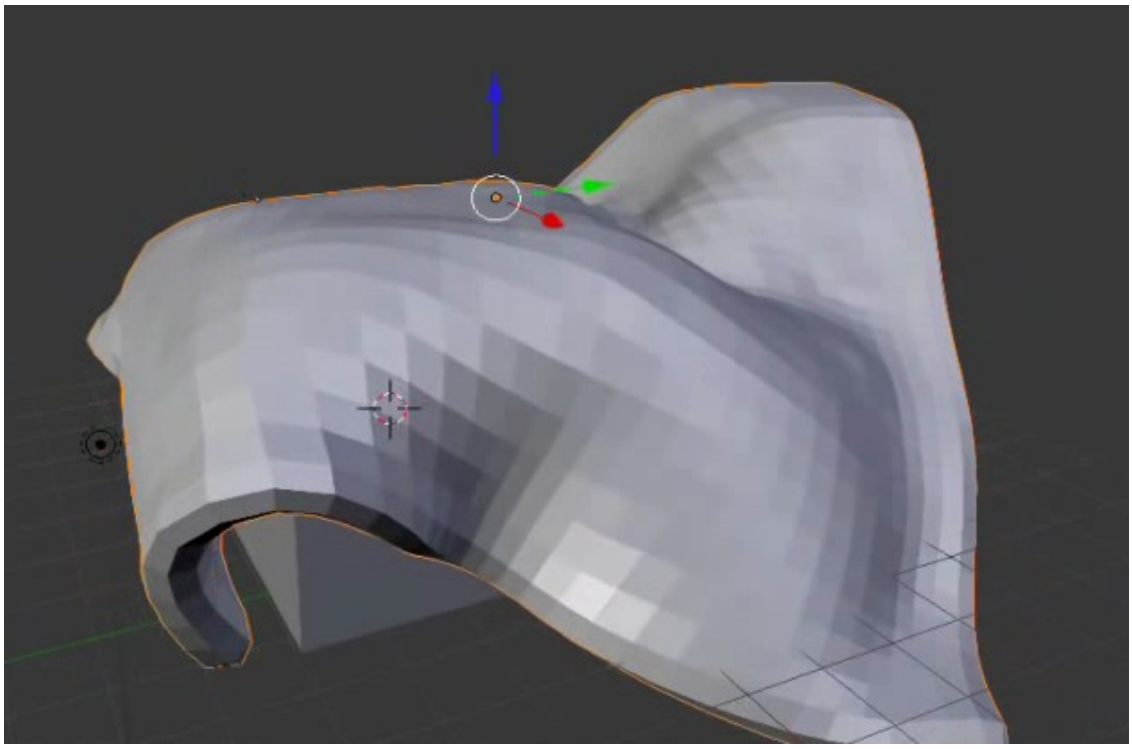


Le impostazioni CLOTH per il Plane



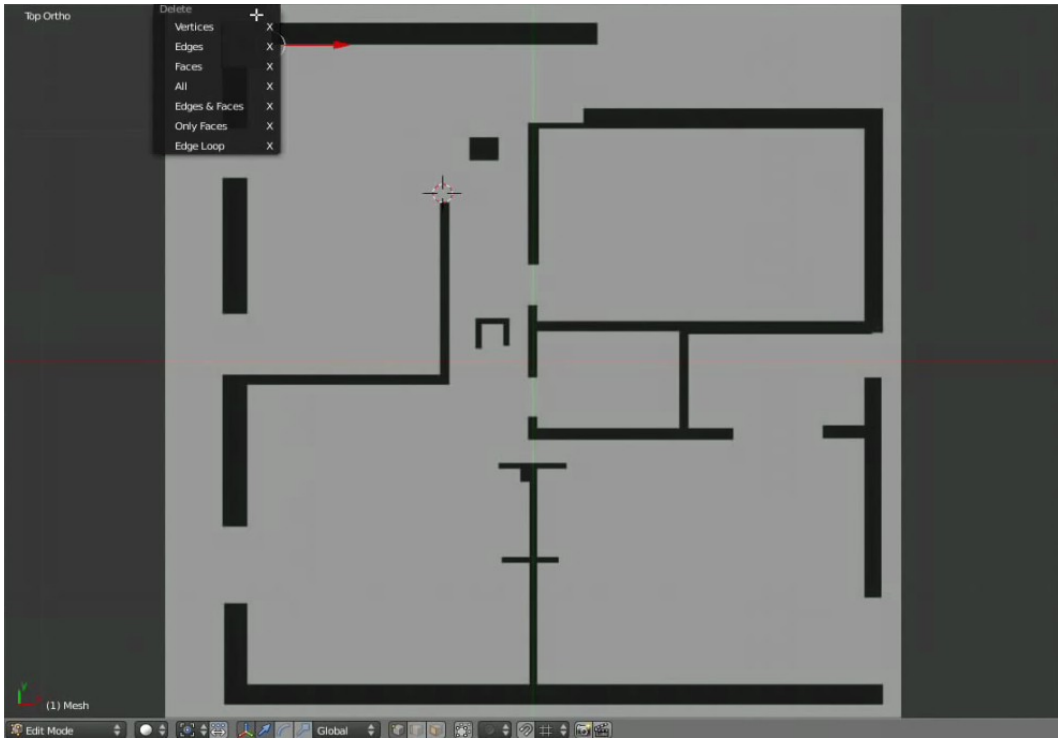
Utilizzo dei modificatori Solidify e SubSurf (Subdivision Surface)

Avviamo una simulazione premendo *ALT A* mentre il focus del mouse è in una finestra di navigazione 3D.



Screenshot del risultato ottenuto (un fotogramma dell'animazione)

Adesso cancelliamo cubo e Plane per analizzare un altro possibile utilizzo di *Solidify*.
In una finestra *3D View* con vista *Top Ortho* ho caricato, come sfondo, una semplice immagine che rappresenta la pianta di un appartamento, anche se in versione estremamente semplificata, per non perdere troppo tempo.



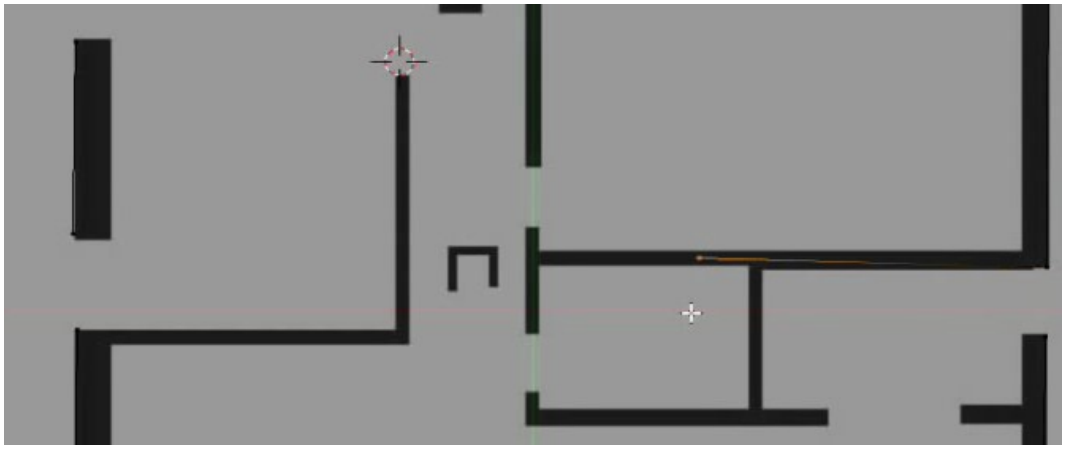
L'immagine di riferimento, vista da una finestra Top Ortho

Per caricare le immagini di sfondo nelle finestre 3D si fa ricorso agli elementi della sezione *Background Images* nella *Transform Window*.

Queste immagini servono solo da supporto per la modellazione, utilizzando magari la modalità di visualizzazione *Wireframe*, per ricreare oggetti o elementi "ricalcandoli", in un certo senso; queste immagini, comunque, non vengono visualizzate in fase di rendering.

Aggiungiamo un Plane alla scena ma eliminiamo uno spigolo, lasciando quindi solo un segmento.

Adesso possiamo procedere alla definizione dei muri, in questo caso utilizzando però degli spigoli senza spessore, estrudendo i vertici, operazione che va fatta in *Edit Mode* premendo *E* e trascinando.

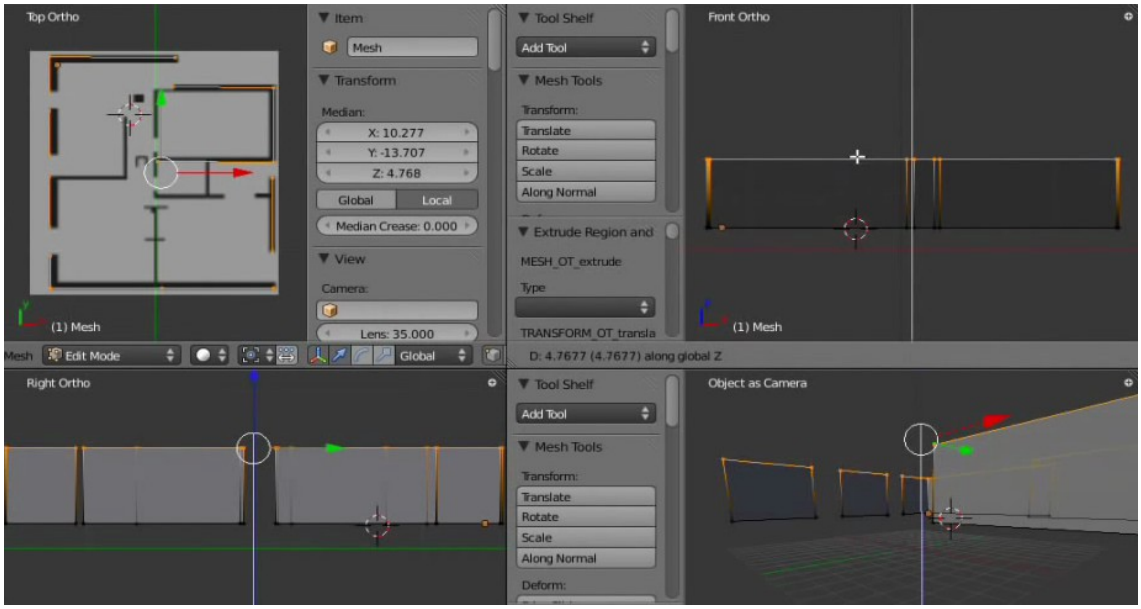


Estrudere i vertici per seguire il disegno di riferimento

Faccio notare che, selezionando un vertice, nella *Transform Window* sarà possibile digitare le coordinate esatte da assegnare al punto, scrivendo i valori per X, Y e Z nelle relative caselle della sezione *Transform Vertex*; questa funzionalità di modellazione parametrica può tornare utile per realizzare i modelli e riprodurre gli oggetti in maniera molto precisa.

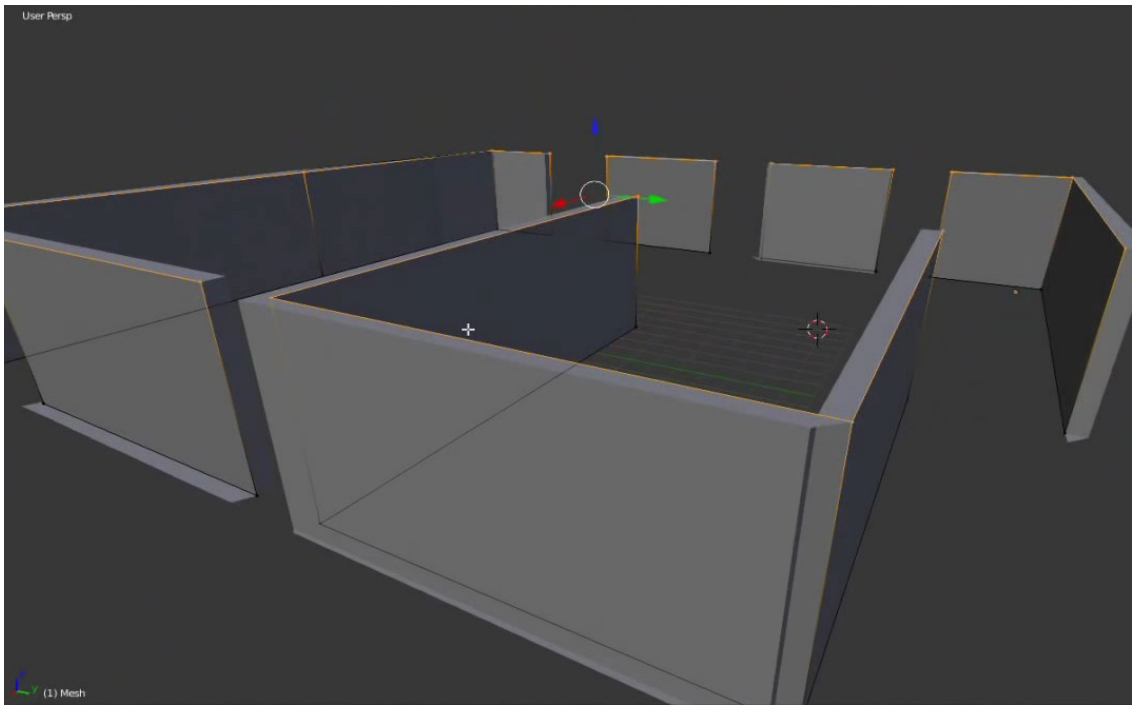
In presenza di muri non connessi ad altri, non dobbiamo necessariamente creare dei Planes a parte: sempre in *Edit Mode*, possiamo selezionare uno spigolo (selezionando due vertici o lo spigolo, a seconda del metodo di selezione utilizzato) e duplicarlo con *SHIFT D*, per poi modificarlo, ad esempio traslandone i vertici, estrudendoli e così via.

Dal momento che la copia è stata effettuata in Edit Mode, la nuova porzione creata farà parte dell'oggetto originale, ossia: passando in Object Mode, potremo trasformare il tutto come un unico oggetto. Prima di applicare *Solidify*, comunque, estrudiamo verso l'alto questi vertici, posizionandoci in una vista *FRONT* e premendo *E Z*; premendo *Z*, infatti, vincoleremo l'estrusione all'asse Z, ossia in verticale, per realizzare delle pareti dritte.



Estrusione verso l'alto dei vertici, per realizzare le pareti

Adesso possiamo applicare il modificatore *Solidify* all'oggetto per aggiungere lo spessore alle pareti. Prima di applicare definitivamente il modificatore, possiamo trasformare ancora l'oggetto in Edit Mode, ad esempio spostando, ruotando o estrudendo vertici; gli effetti di *Solidify* verranno visualizzati in tempo reale.



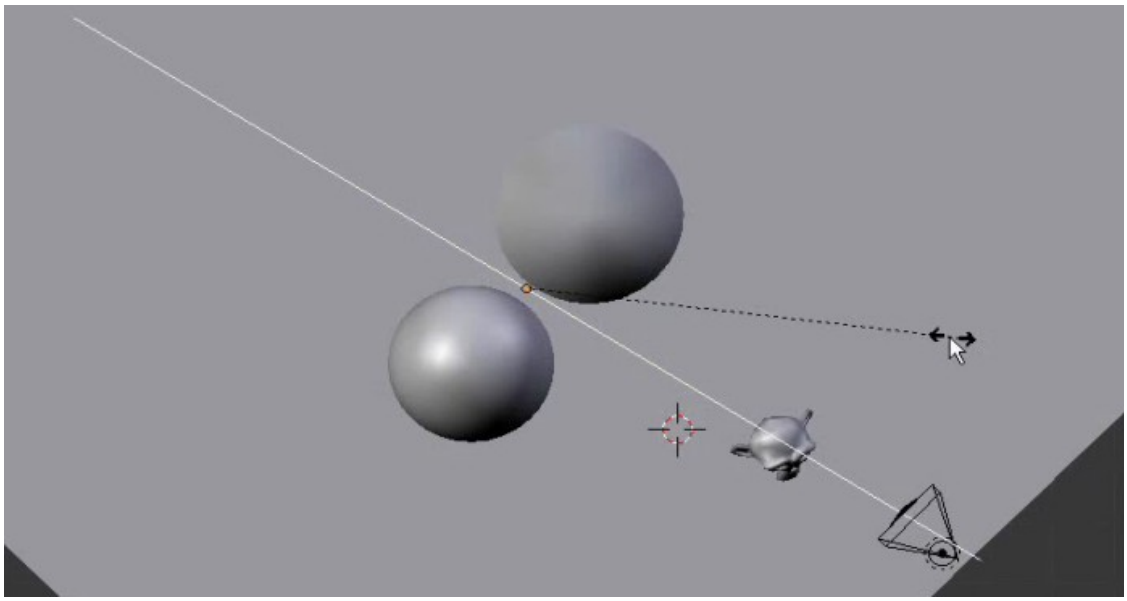
Il risultato ottenuto utilizzando Solidify

* * *

Animazioni Follow Path (segui percorso)

In questo tutorial vedremo come utilizzare una curva *PATH* per definire un percorso di animazione per una telecamera o un altro oggetto (come luci, mesh eccetera), utilizzando una funzionalità nota come *Follow Path*, ossia "segui percorso".

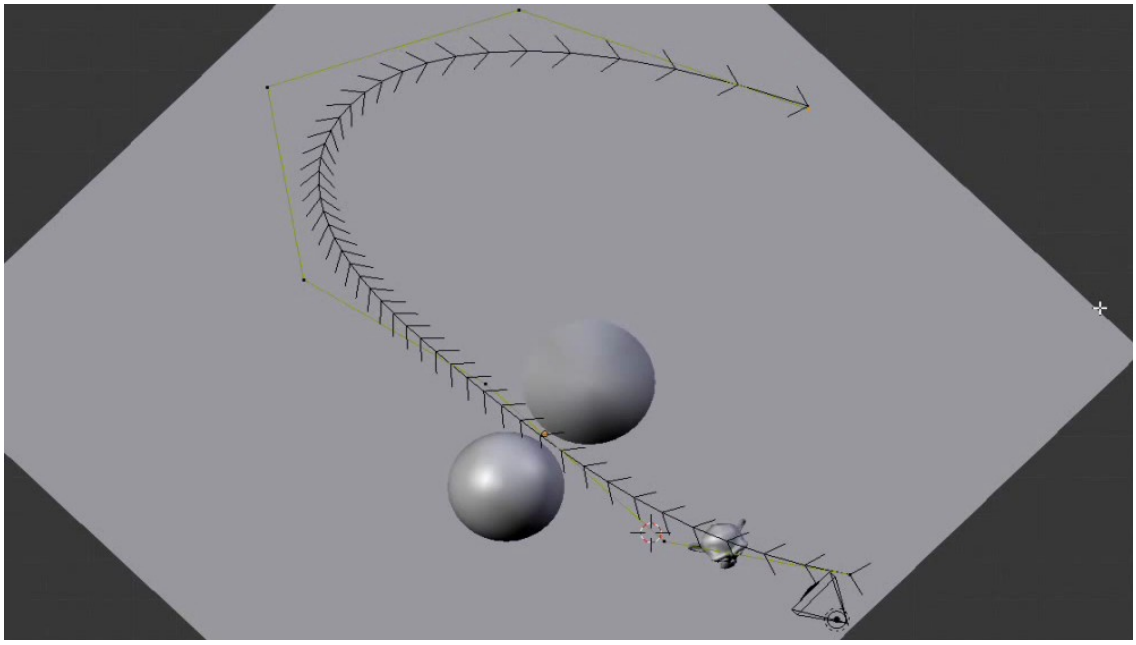
Per prima cosa, inseriamo un oggetto *Path* nella scena con *Add - Curve - Path*.



Una curva PATH inserita nella scena

Path è una curva di tipo *NURBS*, ossia dotata di alcuni punti di controllo che possiamo modificare (traslandoli, ad esempio) passando in Edit Mode e selezionandone uno o più di uno come faremmo con un vertice o un gruppo di vertici di una mesh; premendo *G* e muovendo il mouse, ad esempio, potremo traslare uno o più punti di controllo, ed influenzare, in questo modo, la forma del percorso; spostando il percorso in Object Mode, invece, lo sposteremo nella sua interezza.

Possiamo aggiungere un punto di controllo estrudendone uno già esistente o inserendone uno in mezzo a due già esistenti; nel primo caso, selezioneremo un'estremità della curva e premeremo *E*, mentre nel secondo caso selezioneremo due o più punti di controllo, premeremo *W* per aprire il menù *Specials* e sceglieremo, quindi, *Subdivide*.



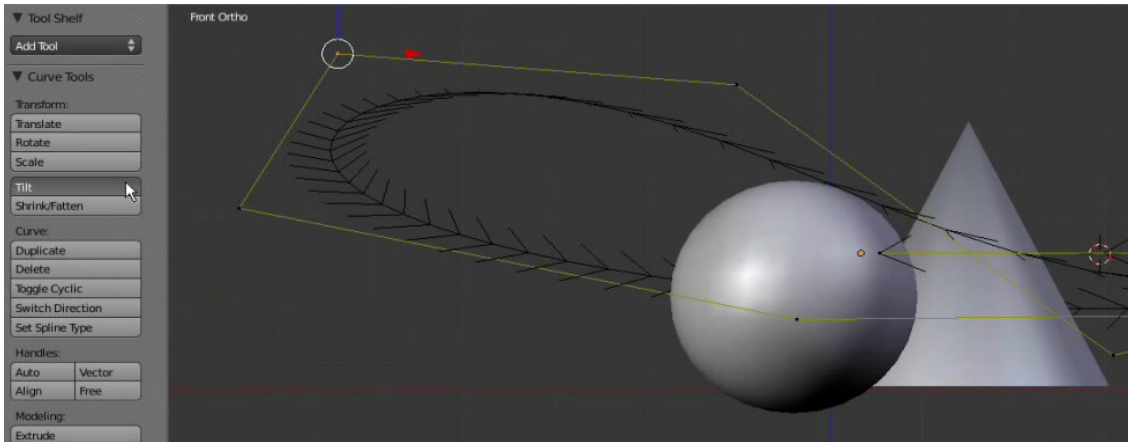
Creazione di punti di controllo nella curva

Possiamo cancellare un punto di controllo premendo *X* o *CANC* e scegliendo *Selected*.

Possiamo chiudere una curva *PATH* premendo *C* in Edit Mode.

Il "verso" del percorso è indicato dalle frecce (visibili in Edit Mode) disposte lungo la curva *Path*; possiamo disattivare la loro visualizzazione o renderle più o meno lunghe utilizzando i pulsanti *Normals* e *Normal Size* presenti nella *Transform Window*, richiamabile con lo shortcut *N* mentre il focus del mouse è in una finestra di navigazione 3D.

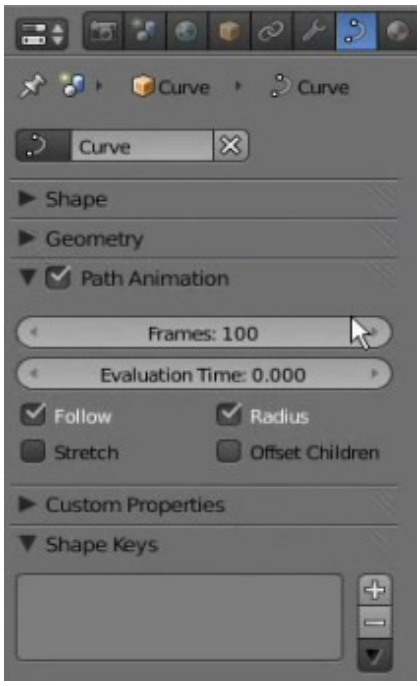
Il percorso è, inizialmente, piatto, ma possiamo muovere i punti di controllo nelle tre dimensioni come preferiamo; per ruotare un punto della curva intorno all'asse che rappresenta la sua direzione, però, dobbiamo selezionare un punto di controllo (in Edit Mode, quindi) e ricorrere al *Tilt*, dal nome del tipo di movimento che vogliamo implementare; dobbiamo quindi cliccare sul pulsante *Tilt*, presente nella *Tool Shelf*, e muovere il mouse per definire l'entità della trasformazione, cliccando col tasto sinistro quando abbiamo finito.



Il Tilting di un punto di controllo

Il *TILT* può essere impostato anche in maniera parametrica, utilizzando le caselle del gruppo *Transform TILT* nella *Tool Shelf*, in basso.

Nella scheda *Object Data* nella *Properties Window* selezioniamo infine *Follow*, ossia letteralmente "segui", nella sezione *Path Animation*, che di default è selezionata.



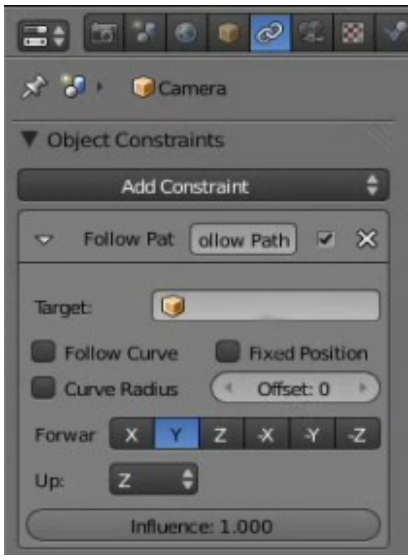
Path Animation Follow

La durata dell'animazione è espressa, in numero di fotogrammi, nel campo *Frames*, che di default ha valore 100. Lasciamo questo valore così com'è, torneremo su questa scheda tra poco; per il momento, però, passiamo alla telecamera.

Per "legare" la telecamera o un altro oggetto (anche una mesh) ad una curva *Path* dobbiamo aggiungere un *Constraint* (letteralmente, un vincolo) per tale oggetto.

Selezioniamo quindi una telecamera, apriamo la scheda *Object Constraint* nella *Properties Window* ed aggiungiamo un *Constraint Follow Path* (letteralmente: segui percorso). Nel campo testuale che apparirà nella scheda dovremo inserire il nome della Curva da utilizzare come percorso, in modo da assegnare tale percorso alla telecamera. Selezioniamo anche la casella *Follow Curve*, per far sì che gli oggetti ruotino mentre percorrono il *path*, seguendo l'orientamento determinato dalle frecce.

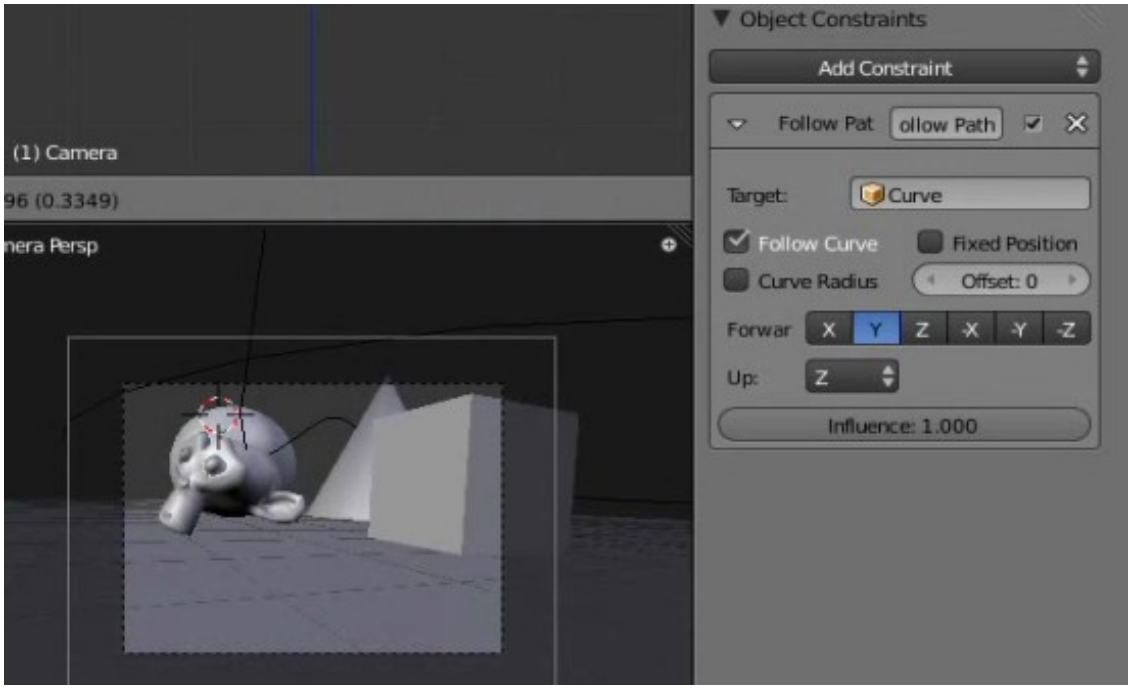
All'inizio Blender potrebbe spostare la telecamera in un altro punto, per cui magari dovremo riposizionarla correttamente, comunque quello che conta è che, a partire da adesso, c'è una relazione tra la telecamera e il percorso, per cui la telecamera (indipendentemente dalla posizione di partenza) si muoverà e ruoterà seguendo quel *path*.



La scheda del Constraint Follow Path per la telecamera

Già in questo modo abbiamo realizzato un'animazione: la telecamera (o, in generale, l'oggetto) seguirà il percorso dal frame 1 al frame 100, come possiamo vedere lanciando un'anteprima dell'animazione, premendo *ALT A* mentre il focus del mouse è in una finestra di navigazione 3D.

Fermiamo quest'anteprima premendo nuovamente *ALT A*.

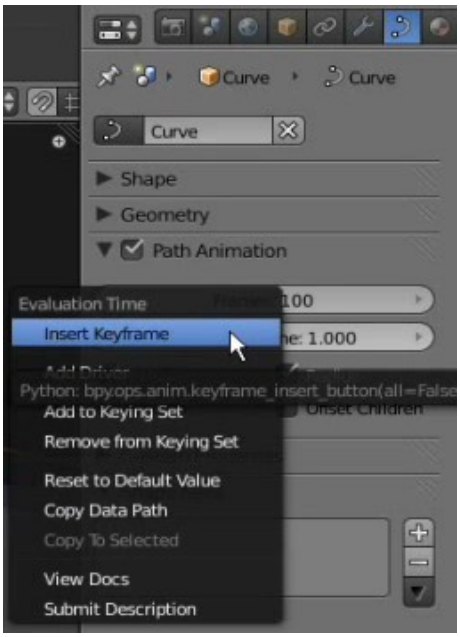


Screenshot dell'animazione realizzata fino a questo punto

L'oggetto (in questo caso, la telecamera) segue il *PATH*, che però può essere ancora modificato, ad esempio traslando un punto di controllo in *Edit Mode*; in questo caso, Blender aggiornerà automaticamente l'animazione della telecamera, che seguirà quindi il nuovo *path*.

Ovviamente, possiamo modificare anche il frame di inizio animazione, la durata dell'animazione e la velocità nelle varie "zone", per creare ad esempio accelerazioni o decelerazioni; ci sono vari modi per farlo ma io consiglio di definire, inizialmente, un'animazione banale, ed utilizzare poi l'editor delle *curve IPO*, noto in Blender 2.5 col nome di *Graph Editor*.

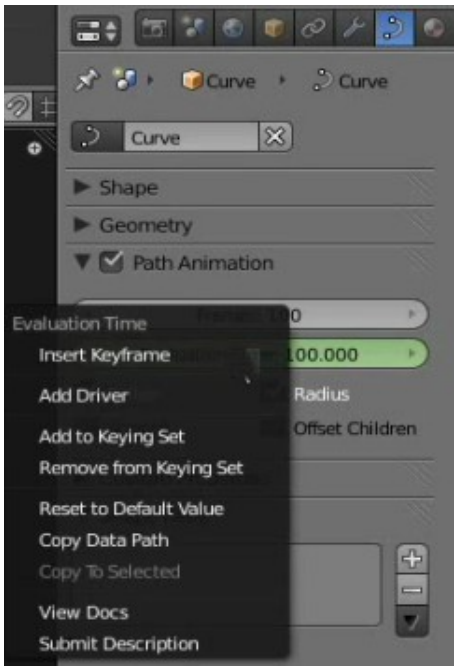
Per definire l'animazione iniziale, in modo da creare una curva di interpolazione o *curva IPO* per il percorso, portiamoci al frame 1, selezioniamo la curva *PATH* ed apriamo la scheda *Object Data* della *Properties Window*; qui, facciamo click col tasto destro del mouse sul pulsante *Evaluation Time*, che dovrebbe avere valore 1 (visto che siamo al frame 1), e scegliamo (dal menù che apparirà a video) la voce *Insert Keyframe*.



Insert Keyframe per Evaluation Time

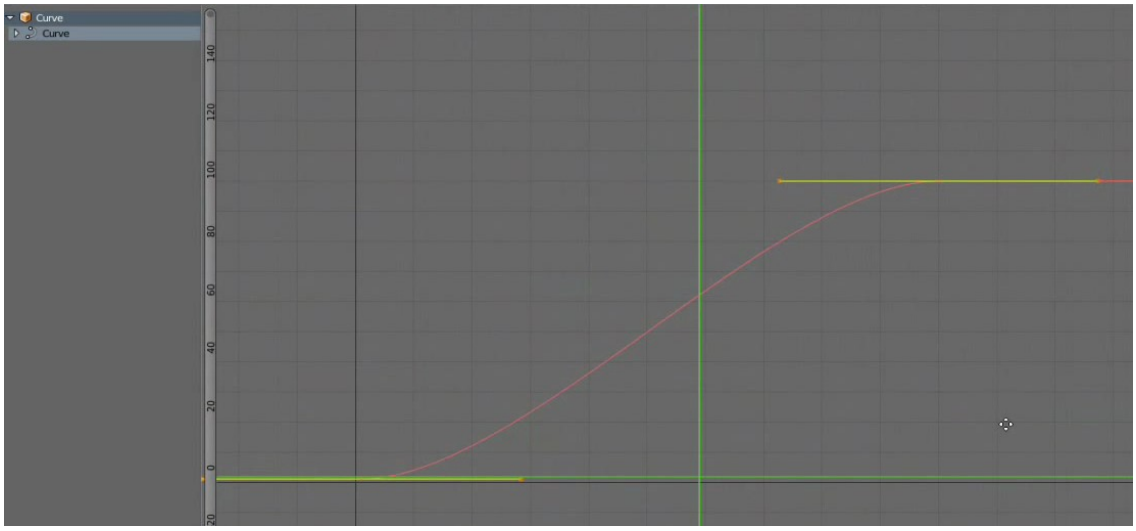
Adesso spostiamoci al frame 100 dell'animazione, utilizzando la *Timeline* in basso; cambiamo manualmente il valore di *Evaluation Time* per la curva in 100 ed inseriamo nuovamente un keyframe facendo click col tasto destro del mouse sulla casella e scegliendo *Insert Keyframe*.

Abbiamo generato nuovamente l'animazione standard, visto che Blender (con i due keyframe inseriti) effettuerà un'animazione della durata di 100 frames (ossia il numero di frames propri del *PATH*, come specificato nel campo *Frames*) dal frame 1 al frame 100; adesso, però, possiamo cambiare una finestra in *Graph Editor* e lavorare sulla curva di interpolazione.



Nuovo Keyframe al 100esimo frame dell'animazione

Con il *PATH* selezionato, infatti, in una finestra *Graph Editor* possiamo manipolare la *curva IPO* associata all'animazione lungo il percorso, come visibile nell'immagine seguente.

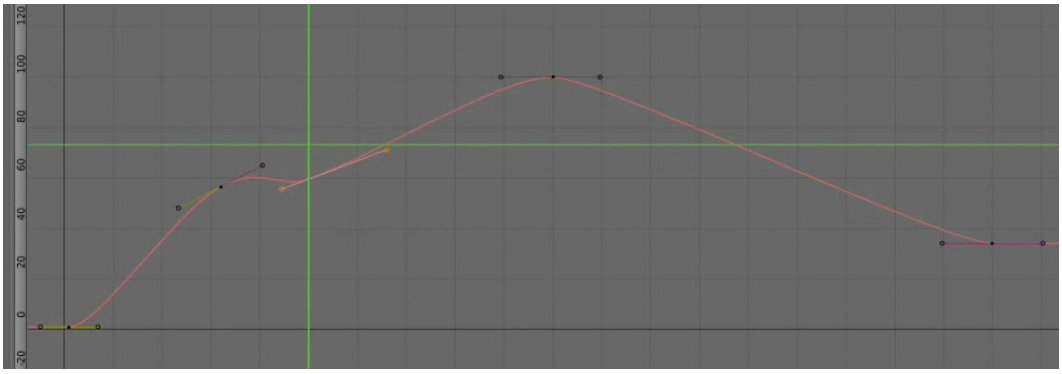


La curva IPO associata all'animazione, nel Graph Editor

L'asse orizzontale rappresenta la *Timeline*, con i numeri dei frame, mentre l'asse verticale rappresenta l'avanzamento dell'animazione dal frame 1 al numero di frame definito nel campo *Frames* per il *PATH*; consiglio di utilizzare, come valore per quel campo, 100, perché così potrete esprimere l'avanzamento in percentuale, ma ovviamente nessuno vi vieta di inserire un valore pari alla durata dell'animazione, a patto di inserire lo stesso valore in *Evaluation Time*.

Torniamo comunque alla nostra *curva IPO*. In genere, l'asse verticale è più "dilatato" rispetto a quello orizzontale, per cui conviene schiacciarlo un po', cosa che va fatta tenendo premuti *CTRL*, il tasto centrale del mouse e muovendo il mouse verso il basso. In questo caso abbiamo solo due punti di controllo, visto che abbiamo inserito due keyframe al frame 1 e al frame 100, ma possiamo aggiungere un punto di controllo selezionandone uno già esistente con un click del tasto destro del mouse e premendo *SHIFT D*, oppure posizionando la barra verde verticale in un punto della finestra (con un click del tasto sinistro del mouse) per poi premere *I* e scegliere per quali canali inserire il keyframe. Questi punti di controllo sono, in effetti, fotogrammi chiave dell'animazione.

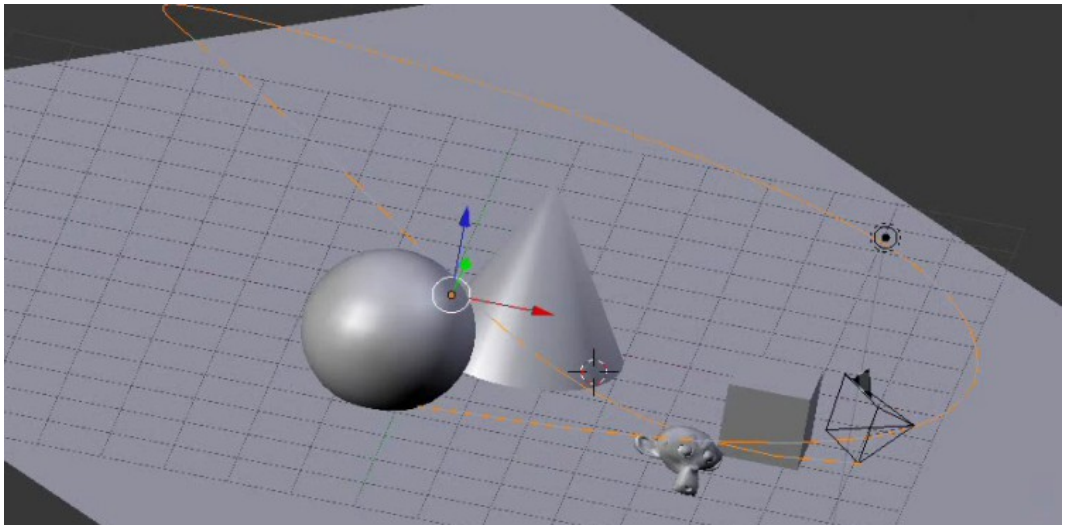
Selezionando le maniglie dei punti di controllo, potremo variare l'andamento della curva in prossimità degli stessi traslando, ruotando o scalando le relative maniglie con *G*, *R* o *S*; possiamo anche variare la forma di una maniglia selezionandola e premendo *V*, per poi scegliere tra le voci presenti quella che fa per noi, ad esempio *Vector* per avere transizioni brusche, oppure free per poter modellare ciascuna maniglia indipendentemente dall'altra.



Modificare le maniglie dei punti di controllo della curva IPO

Frame iniziale e frame finale erano, come anticipato, 1 e 100, ma qui possiamo selezionare uno o più punti di controllo (anche tutti, con *A*) e spostarli lungo la *Timeline* premendo *GX* per variare il frame iniziale, il frame finale e, quindi, la durata dell'animazione.

Dilatando e restringendo le transizioni tra punti di controllo, potremo creare accelerazioni, fasi di stasi e decelerazioni; ad esempio, possiamo animare gli oggetti nel verso opposto facendo assumere alla curva un andamento discendente. Vi consiglio di fare varie prove per valutare i vari effetti possibili. Tornando in una finestra di navigazione 3D, potremo valutare il lavoro fatto lanciando un'anteprima dell'animazione con *ALT A*, come mostrato precedentemente.



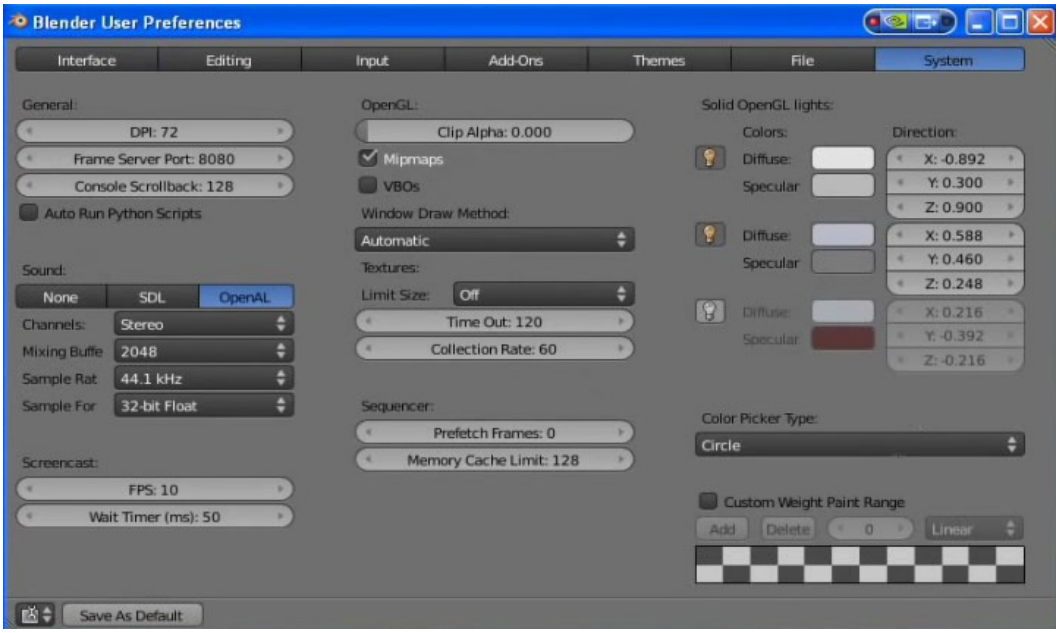
Screenshot del risultato (frame dell'animazione)

La possibilità di far percorrere una curva da noi definita ad un oggetto o ad un gruppo di oggetti, anche di tipi differenti, può essere utile per realizzare animazioni per videoclip (ad esempio, le montagne russe in prima persona oppure i movimenti di pianeti e corpi celesti), oppure effetti speciali, visite in ambienti 3D, visualizzazioni di appartamenti o strutture, musei virtuali eccetera.

* * *

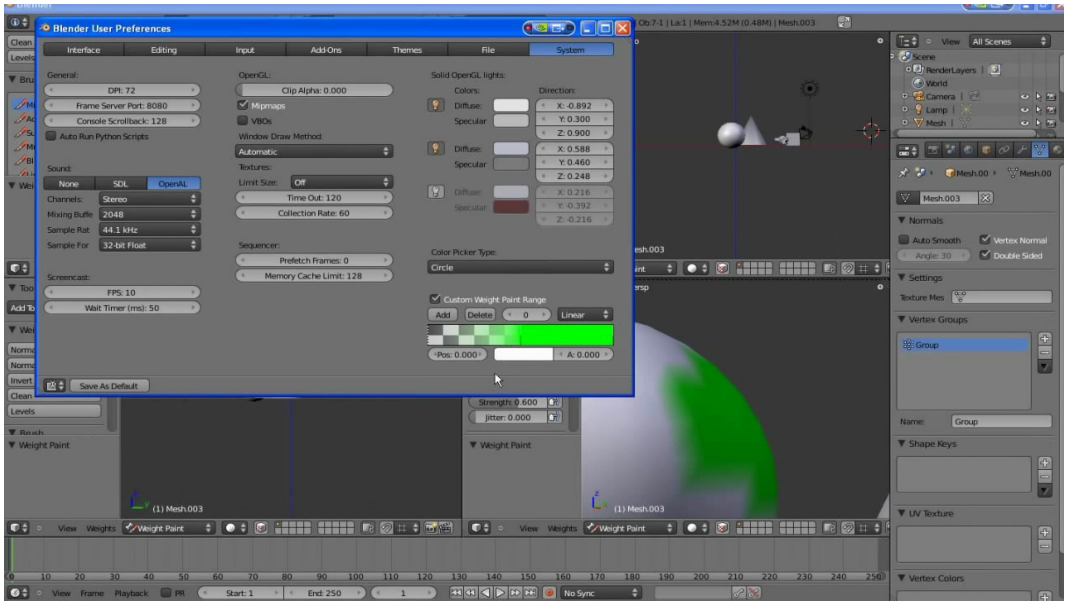
User Preferences e colori Weight Paint personalizzati

In questo brevissimo tutorial vedremo come personalizzare i colori associati ai veri pesi in modalità *Weight Paint* in Blender 2.5. In modalità *Weight Paint*, infatti, la scala dei colori di default associa il blu al peso minimo, il rosso al peso massimo e sfumature di ciano, giallo e verde ai pesi intermedi; è possibile, comunque, cambiare questa scala, ad esempio per utilizzare solo sfumature di grigio dal nero al bianco. Per fare ciò, apriamo il pannello *User Preferences* dal menù *File* e selezioniamo, all'interno di tale finestra, la scheda *System*.



La scheda System nel pannello User Preferences, richiamabile dal menù File

In basso a destra vi è un'etichetta, *Custom Weight Paint Range*, ossia scala dei colori personalizzata per il *Weight Painting*: selezioniamola per attivarla.



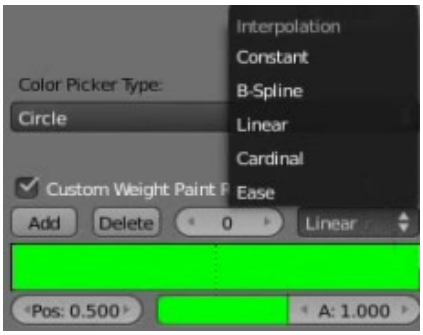
Custom Weight Paint Range (qui già modificata, rispetto all'originale)

Adesso possiamo utilizzare la *Color Ramp* posta sotto l'etichetta per personalizzare la scala dei colori. Possiamo aggiungere una banda, per impostare un colore, cliccando sul pulsante *Add*.

Per spostarci da una banda all'altra possiamo utilizzare il pulsante numerico con un numero al centro e le frecce ai lati, cliccando sulle frecce per passare alla banda precedente o alla successiva; la numerazione inizia con zero ma all'inizio non c'è nessuna banda, neanche la banda zero, appunto, per cui dobbiamo comunque cliccare su *Add* per aggiungerne almeno una.

Per ciascuna banda potremo poi impostare la posizione all'interno della *Color Ramp* (con estremi dell'intervallo 0 e 1); possiamo impostare anche il colore da utilizzare e il livello di trasparenza *Alpha* di tale colore mediante i pulsanti che appariranno sotto la *Color Ramp*.

Il livello di trasparenza *Alpha* del colore di una banda può essere diverso da 1 per consentire una miscelazione un po' più particolare dei colori all'interno della banda.



Il menù a scomparsa del tipo di interpolazione

L'ultimo pulsante da esaminare, che in realtà è un menù a scomparsa, è quello posto sopra la *Color Ramp* a destra: tale strumento ci consente di scegliere il tipo di interpolazione da

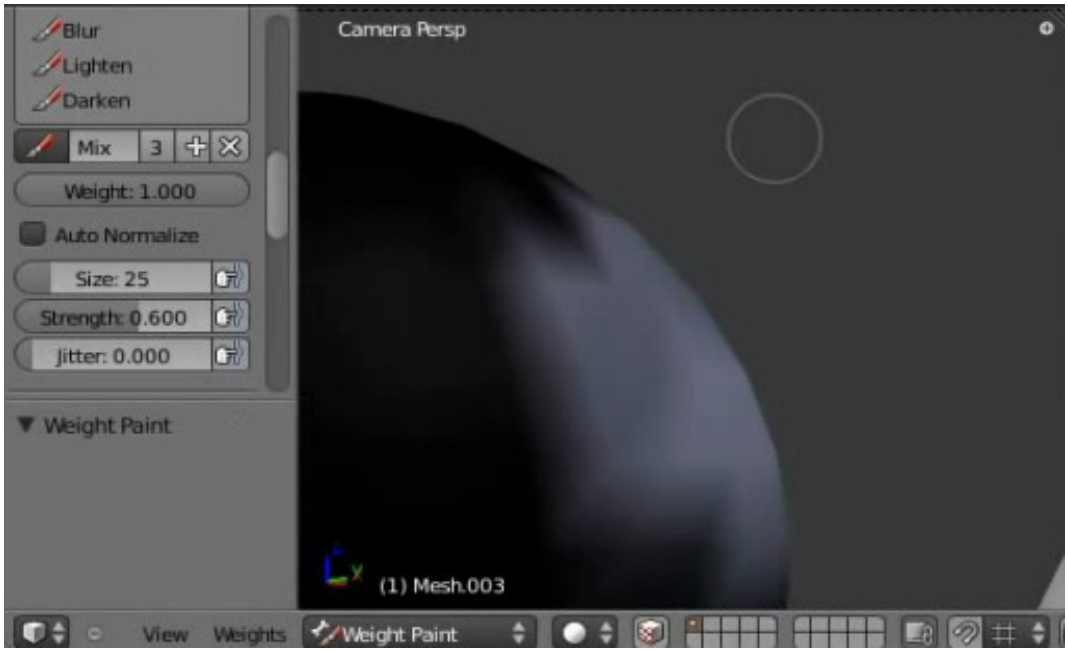
utilizzare per passare, all'interno della *Color Ramp*, dal colore di una banda a quello della precedente o della successiva; in genere, *Linear* va più che bene.

Proviamo quindi a cambiare la scala dei colori originale portandola in scala di grigi.
Dobbiamo utilizzare solo due divisioni, due bande:

- la prima, posta in posizione zero, con colore nero e *Alpha* 1;
- la seconda, posta in posizione uno, con colore bianco e *Alpha* 1.

La nuova Color Ramp

Le modifiche verranno apportate anche ad eventuali oggetti creati precedentemente con scale di colore differenti.



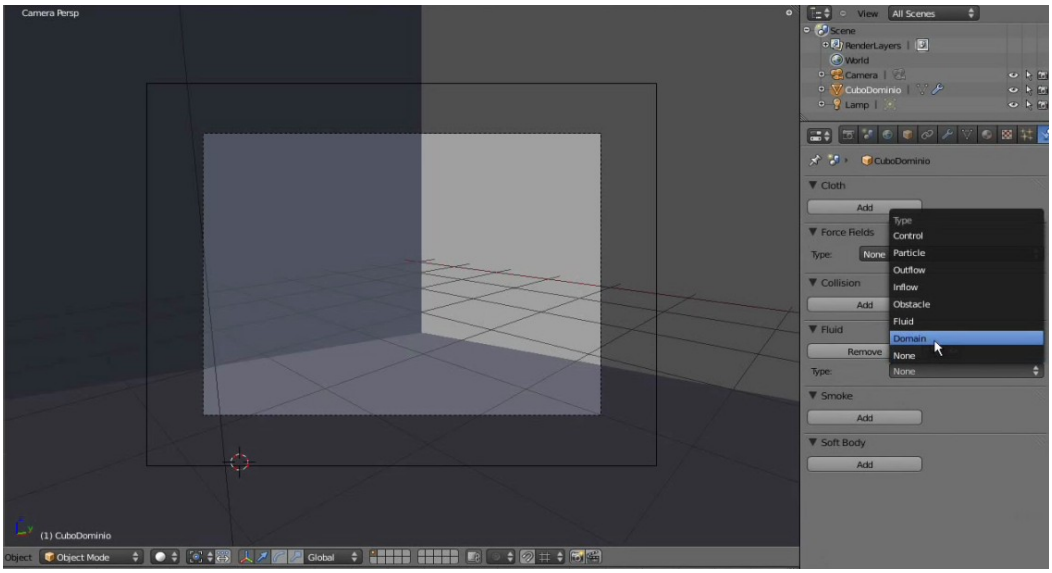
Esempio

* * *

Basi del Motore dei Fluidi - 1: introduzione e Domain

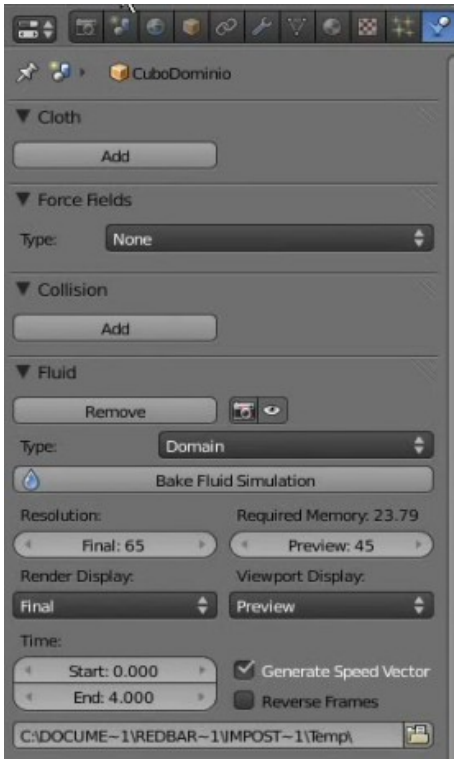
In questo tutorial inizieremo a parlare del motore dei fluidi di Blender 2.5, prendendo in esame i concetti generali e l'oggetto *Domain*; si tratta, quindi, di una puntata "teorica" per comprendere i concetti di base: vedremo il motore dei fluidi in azione nella prossima puntata.
Il motore dei fluidi di Blender genera una simulazione all'interno di uno spazio ben definito, detto *Dominio* della simulazione: tutti gli oggetti di tipo *fluido*, *inflow*, *outflow* e *ostacolo*, che vedremo nelle prossime puntate, dovranno trovarsi all'interno del *Domain*, altrimenti non verranno presi in considerazione nel corso della simulazione.

Per definire il *Dominio*, possiamo utilizzare una qualsiasi mesh chiusa, tuttavia non ha importanza la forma, perché alla fine Blender utilizzerà sempre un cubo, prendendo in considerazione il lato più lungo della mesh da noi utilizzata. Come *Domain* per i nostri esempi, quindi, utilizziamo direttamente un oggetto Cube. Rinominiamo tale oggetto, ad esempio in "*CuboDominio*" (presto vedremo perché conviene effettuare tale operazione). Posizioniamo la telecamera all'interno di tale cubo oppure cambiamo la modalità di visualizzazione di qualche finestra in *Wireframe*, per poter vedere all'interno di tale spazio e posizionare correttamente gli altri oggetti.



Il cubo con la telecamera al suo interno, prima di renderlo "Fluid Domain"

Per rendere il cubo *Domain* della simulazione dobbiamo selezionarlo e aprire, all'interno della *Properties Window*, la scheda *Physics*; qui, clicchiamo su *Add* nella sezione *Fluid* e, dal menù che apparirà, scegliamo la voce *Domain*.



La scheda Fluid Domain nella sezione Physics, per il cubo

Il cubo sparirà, ma si tratta di una modalità di visualizzazione, infatti è ancora presente come oggetto nella scena e per richiamarlo basta selezionare la relativa voce nell'*Outliner* (ecco perché gli abbiamo assegnato un nome distintivo, prima).

Esaminiamo le voci apparse all'interno della scheda *Fluid* per capire come funziona un oggetto *Domain* e quali sono le impostazioni principali.

Il pulsante *Bake Fluid Simulation* è quello che va premuto per ultimo per effettuare la simulazione.

Quando si effettua il *Bake*, per ogni fotogramma vengono create delle mesh per tutti gli oggetti che appartengono alla simulazione dei fluidi.

L'avanzamento del *Bake* viene espresso in percentuale (indipendentemente dal numero di frames della simulazione) con un numero che apparirà al posto del cursore del mouse non appena cliccheremo su *Bake*. Il *Baking* può essere interrotto premendo il tasto *ESC*.

Le mesh prodotte in fase di *Baking* vengono salvate su disco, in una cartella il cui percorso va definito nel campo posto in fondo alla scheda *Fluid*, utilizzando magari la finestra di tipo *File Browser* attivabile cliccando sull'icona in basso a destra.

Quando si effettua una simulazione, quindi, conviene utilizzare una cartella a parte per il progetto corrente, soprattutto per evitare di sovrascrivere simulazioni fatte con altri progetti o per cancellare i file prodotti quando non servono più, visto che possono occupare anche molto spazio su disco.

Sotto *Bake* abbiamo, a destra, un'etichetta che ci dice qual è la quantità di memoria richiesta per la simulazione. Tale quantità dipende fortemente dalla risoluzione adottata, che può essere diversa per la fase di rendering vero e proprio e per la fase di anteprima.

Tali valori di risoluzione vanno impostati nei campi *Final* e *Preview*, sotto l'etichetta della quantità di memoria richiesta. A questo punto però dobbiamo fermarci per capire meglio cos'è la risoluzione, visto che si tratta di un parametro fondamentale, forse il più importante di tutta la simulazione.

Quando si effettua una simulazione dei fluidi, il motore di Blender suddivide l'oggetto *Domain* in cubetti detti *Voxel*, che in pratica rappresentano quello che nelle immagini sono i pixel, ed infatti Voxel sta proprio per per *Volume Element*; ora, come nelle immagini è di fondamentale importanza, ai fini della qualità, la risoluzione in numero di pixel, nella simulazione dei fluidi ad una risoluzione maggiore (ossia, ad numero maggiore di *Voxel*, sempre più piccoli) corrisponderanno risultati migliori. Il problema è che raddoppiando il valore della risoluzione, la quantità di memoria richiesta per la stessa non raddoppierà, ma seguirà in genere *la terza potenza*.

A questo seguiranno, intuitivamente, tempi di *baking* sempre maggiori, a volte anche diverse ORE, per cui nei tutorial che farò per questa serie in genere realizzerò il *baking* con risoluzioni basse e poi mostrerò un rendering effettuato con risoluzioni maggiori, indicando di volta in volta il valore della risoluzione utilizzata.

Passiamo alle voci dei menù *Render Display* e *Viewport Display*: esse ci consentono di scegliere come visualizzare le mesh ottenute in fase di rendering e in vase di modellazione (ossia quella corrente); le possibilità sono: *Geometry* per vedere le mesh originali, *Preview* per visualizzare il risultato ottenuto con la risoluzione impostata in *Preview Resolution* e *Final* per il risultato ottenuto con la risoluzione impostata in *Final Resolution*.

Dobbiamo spendere qualche parola anche per i campi *Time*.

Il motore dei fluidi effettua la simulazione seguendo un suo “orologio interno”, dove il tempo è espresso in secondi. La durata totale della simulazione è data dalla differenza dei valori *End Time* e *Start Time*, di default quindi 4 secondi.

Tale durata verrà poi distribuita su tutta la durata dell'animazione di Blender, che di default è di 10 secondi, in quanto abbiamo 250 frames a 25 frame al secondo. Tutto ciò può servire, ad esempio, per generare animazioni rallentate o accelerate; impostando infatti, ad esempio, 2 secondi come durata della simulazione dei fludi, e 10 secondi come durata dell'animazione di Blender, quei 2 secondi dei fluidi dovranno essere "dilatati" per riempire i 10 secondi di Blender, per cui vedremo un'animazione slow-motion; al contrario, impostando ad esempio 20 secondi come durata della simulazione e 10 secondi come durata dell'animazione di Blender, la simulazione dei fluidi dovrà essere eseguita al doppio della velocità per rientrare nei 10 secondi dell'animazione finale.

Va detto anche che impostare come Start Time un valore maggiore di 0, ad esempio 1, significa applicare al frame 0 dell'animazione in Blender lo "stato generale" della simulazione dei fluidi quando questa era già al primo secondo di simulazione... è più facile a farsi che a dirsi, per cui torneremo su questo argomento nelle prossime puntate, quando vedremo le simulazioni dei fluidi in esecuzione; per il momento, fermiamoci qui.

Le voci presenti nella sezione *Domain World* hanno significati intuitivi: è possibile impostare una forza iniziale, detta qui forza di gravità ma che può agire in qualsiasi direzione e con intensità differenti; è possibile, poi, impostare la viscosità del fluido (anche con valori personalizzati, scegliendo *Manual*) e la comprimibilità dello stesso.

La sezione Domain World nella scheda Fluid Domain

Di fondamentale importanza è il campo *Real World Size*, da utilizzare come fattore di scala per la simulazione, scrivendo a quanto deve corrispondere, in metri, il lato del cubo utilizzato come *Domain*; il valore massimo è 10 metri, il che significa che per simulazioni di effetti in ambienti di dimensioni maggiori, come uno tsunami, bisognerà ricorrere ad effetti speciali e ad insiemi di tecniche, come ad esempio le particelle, le Textures e alcuni modificatori.

Nella sezione *Domain Boundary* possiamo definire il comportamento del liquido sulle pareti dell'oggetto *Dominio*, che si comporta da vero e proprio contenitore per i fluidi.

Le voci della sezione Domain Boundary

Slip Type indica come dovrà "scivolare" il liquido sulle pareti del *Domain*, ossia se le particelle del liquido dovranno rimanere "appiccicate", per così dire, alle pareti, o potranno scivolare liberamente; possiamo scegliere tra *Libero*, *Assente* e *Parziale* (in quest'ultimo caso, potremo impostare manualmente l'entità dell'effetto nel campo *Amount*, con valori compresi tra 0, che corrisponde a *No Slip*, e 1, che corrisponde a *Free Slip*).

Con *Smoothing* possiamo definire quanto dovrà essere levigata la superficie del *Domain*; il valore di default, 1, indica il valore standard, normale.

Le suddivisioni sono, di default, 0, perché servono per la simulazione con le particelle e un valore maggiore di 0 può far crescere anche di molto i tempi di baking.

Parleremo delle voci della sezione *Domain Particles* in seguito.

Come anticipato, il tasto *Bake* va premuto per ultimo, dopo aver definito tutte le impostazioni; cambiare anche una sola impostazione del *Domain*, o la disposizione iniziale degli oggetti *fluid*, *inflow*, *outflow* e *ostacoli*, o le loro caratteristiche, ci costringerà a ripetere nuovamente il *baking*.

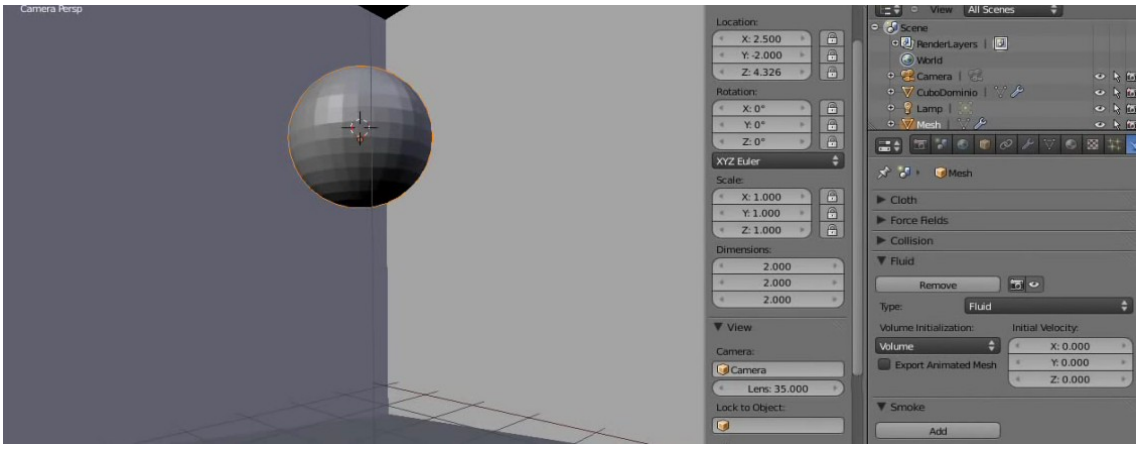
* * *

Basi del Motore dei Fluidi - 2: fluid e obstacle

In questa seconda puntata dedicata alle basi del motore dei fluidi di Blender 2.5 prenderemo in esame gli oggetti *Fluid* e *Obstacle* e vedremo, finalmente, qualche simulazione.
Utilizziamo come *Dominio* della simulazione un cubo qualsiasi, ponendo al suo interno la telecamera, come visto nella puntata precedente.

Il valore di *Real World Size* per tale cubo è 1 metro.

Inseriamo all'interno del cubo un oggetto, ad esempio una *sfera UV* con impostazioni di default, posizionandola più o meno al centro del cubo.



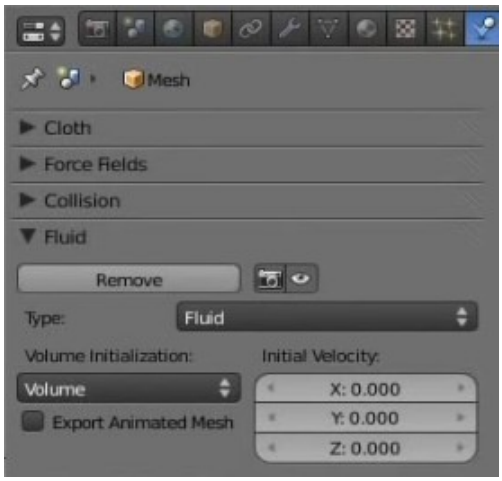
La sfera inserita all'interno del cubo-dominio

Con tale sfera selezionata, apriamo la scheda *Physics* nella *Properties Windows* e clicchiamo su *Add* sotto l'etichetta *Fluid*; nel menù *Type*, scegliamo la voce *Fluid*.

Adesso la sfera è un oggetto fluido, nel senso che la sua mesh si deformerà in base a posizione ed impostazioni del *Domain*; in pratica, dobbiamo immaginare di avere una sfera d'acqua sospesa nel vuoto: appena avvieremo la simulazione, vedremo cosa succederà a questa massa d'acqua.

Esaminiamo le voci presenti nella scheda *Fluid*: *Volume Initialization* è un menù con tre voci, ossia *Shell*, *Volume* e *Both* (letteralmente: guscio, volume ed entrambi); in base alle nostre scelte, verrà creato il volume del fluido partendo:

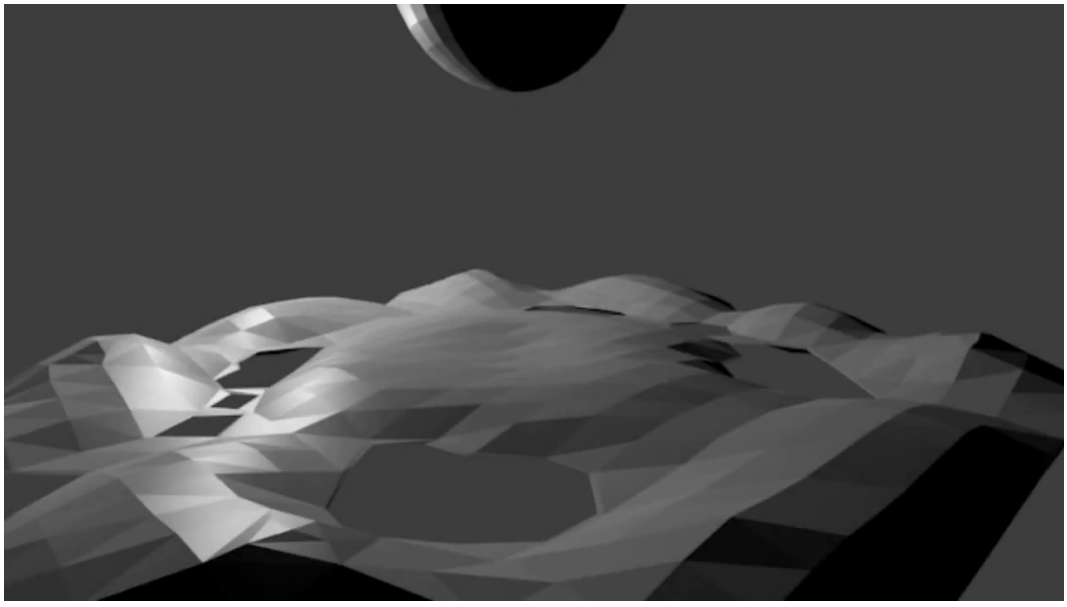
- o dalla sola superficie dell'oggetto *Fluid*, con *Shell*;
- o dal volume dell'oggetto *Fluid*, con *Volume*;
- oppure da entrambi, con *Both*.



Le voci della scheda Fluid

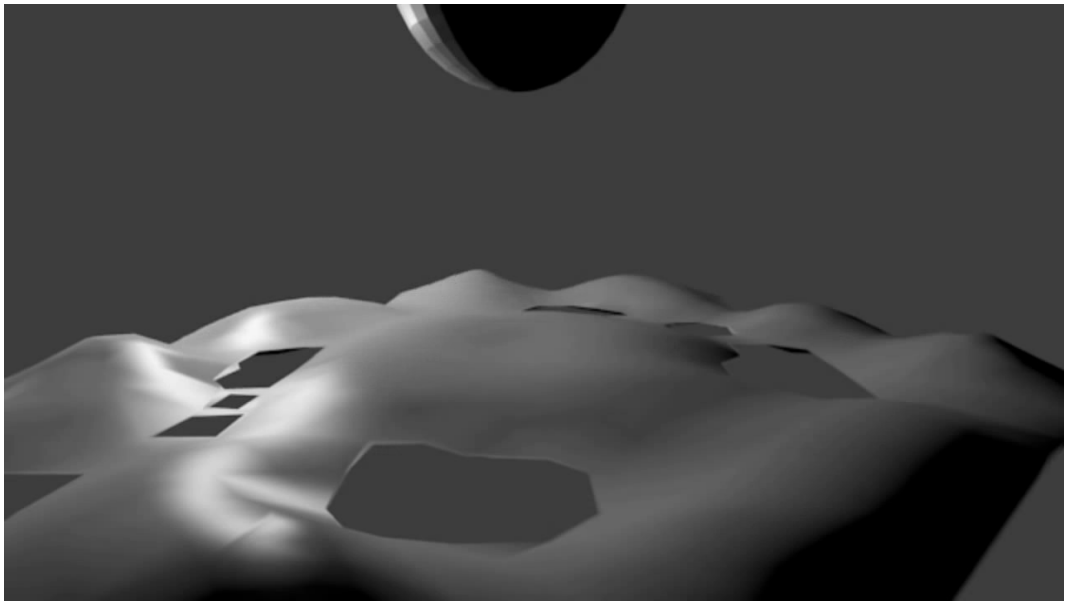
È possibile impostare una velocità iniziale per il fluido utilizzando i campi *X*, *Y* e *Z* e specificando lì il valore di spostamento iniziale nelle varie direzioni, con valori in positivo o in negativo a seconda del verso. Torniamo alla nostra scena e alla simulazione. Dal momento che nel *Domain* è presente, di default, la forza di gravità, impostata come forza di entità 9.8 diretta verso il basso (come indicato nella sezione *Fluid Domain World* della scheda *Physics*), la sfera di fluido creata cadrà verso il basso.

Blender calcolerà automaticamente le collisioni con le pareti del *Domain*.
Nell'immagine qui sotto, potete vedere uno screenshot tratto da una simulazione calcolata su 250 frames a 25 frames al secondo, con *Start Time* 0, *End Time* 10 e risoluzione del *Domain* pari a 30.


Uno screenshot della simulazione realizzata con le impostazioni descritte

Textures e modificatori da applicare ai fluidi della simulazione vanno applicati all'oggetto *Domain*, per cui possiamo smussare un po' la mesh del fluido, ad esempio, selezionando il *Domain* ed applicando, per prima cosa, lo *Shading* di tipo *Smooth*, cliccando su tale pulsante nella *Tool Shelf*.

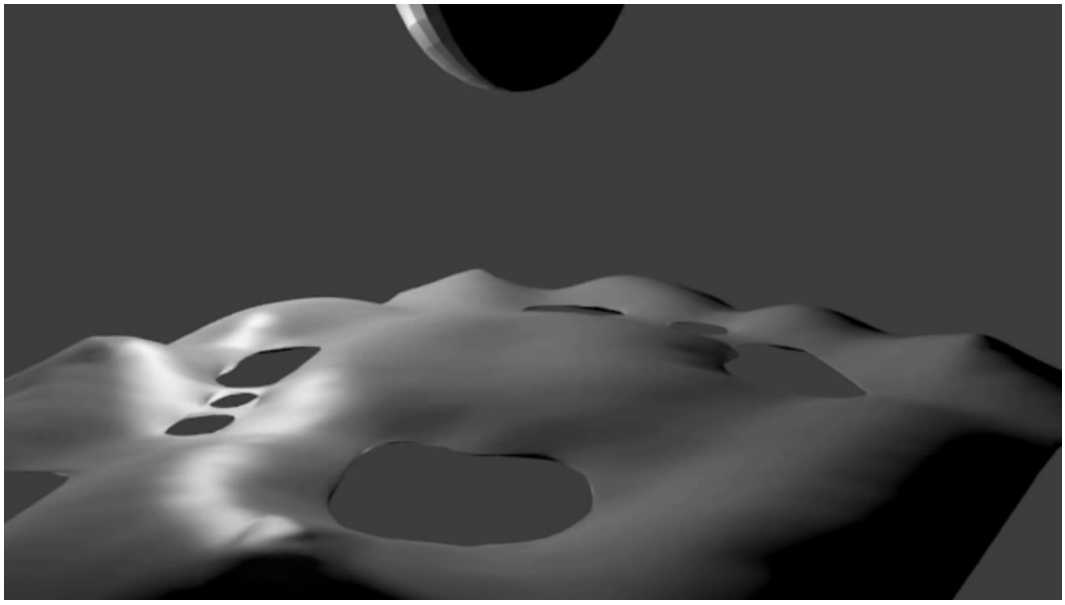
Nell'immagine qui sotto, potete vedere uno screenshot tratto da una animazione ottenuta con *Shading Smooth* e risoluzione 30.



Uno screenshot della simulazione dopo aver applicato lo Shading Smooth al cubo-dominio

Sempre con il *Domain* selezionato, possiamo aggiungere un modificatore di tipo *Subdivision Surface* per migliorare ulteriormente il risultato ottenuto mediante la simulazione; apriamo quindi la scheda *Modifiers* nella *Properties Window* e scegliamo *Subdivision Surface* dal menù presente nella scheda.

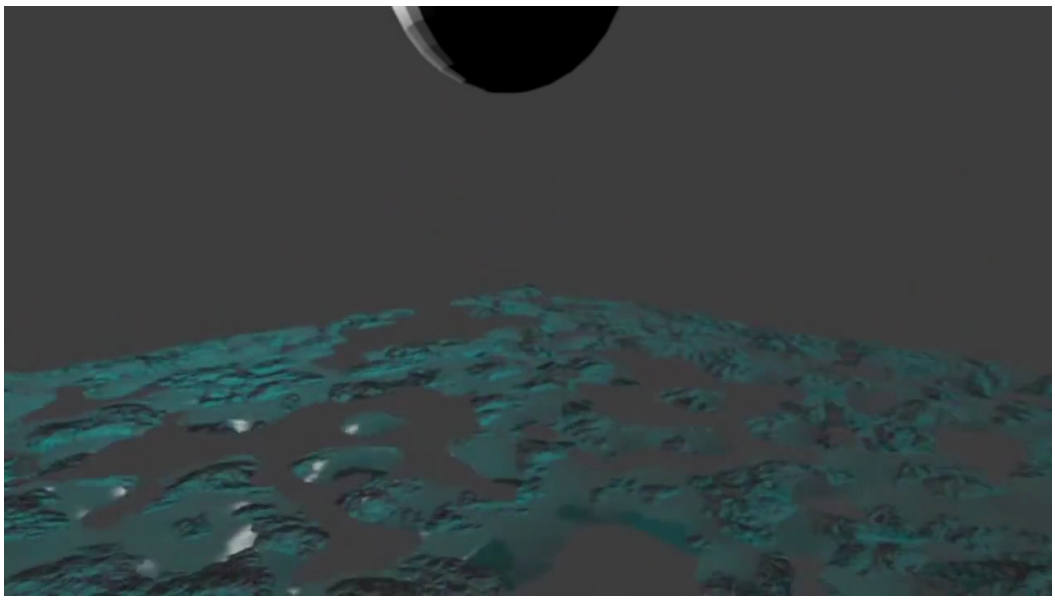
Nell'immagine nella pagina seguente potete vedere uno screenshot tratto dal rendering ottenuto (dopo il *baking*) con risoluzione del *Domain* pari a 30 ma con *Shading Smooth* e *Subdivision Surface* di tipo *Catmull-Clark* con 2 suddivisioni.



Uno screenshot della simulazione dopo aver applicato lo Shading Smooth e un modificatore SubSurf al cubo-dominio

Il Material (eventualmente provvisto di Textures) da associare al fluido, va applicato, ancora una volta, al *Domain* della simulazione.

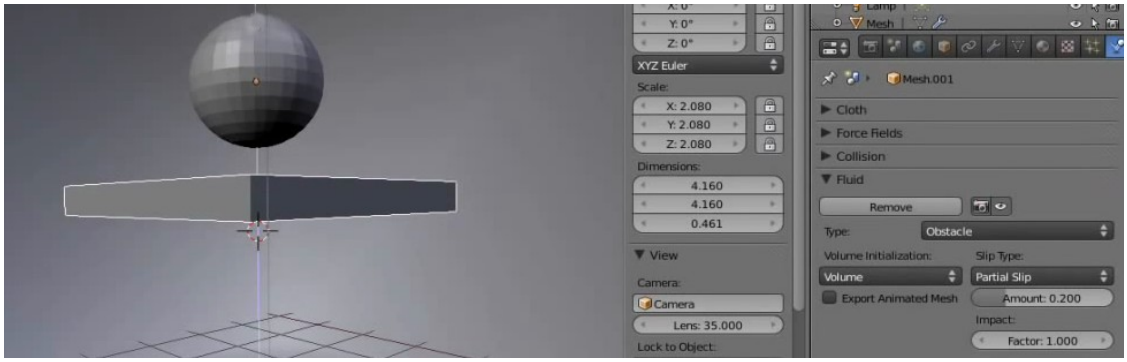
Questa volta lo screenshot riportato nell'immagine qui sotto raffigura il rendering ottenuto (dopo il *baking*) con risoluzione del *Domain* pari a 120, *Shading Smooth*, *Subdivision Surface* di tipo *Catmull-Clark* con 2 suddivisioni e un Material con riflessi *RayMirror.*



Uno screenshot della simulazione dopo aver fornito un Material con riflessi RayMirror al cubo-dominio

Vediamo ora come far interagire un fluido così creato con degli oggetti.
Aggiungiamo, all'interno del cubo *Domain*, a metà altezza tra la sfera *Fluid* e il fondo del *Domain*, un altro cubo, più piccolo, schiacciandolo ed allargandolo un po'.

Con tale oggetto selezionato, apriamo la scheda *Physics* nella *Properties Window*, clicchiamo su *Add* nella sezione *Fluid* ma questa volta scegliamo, dal menù *Type*, la voce *Obstacle*.



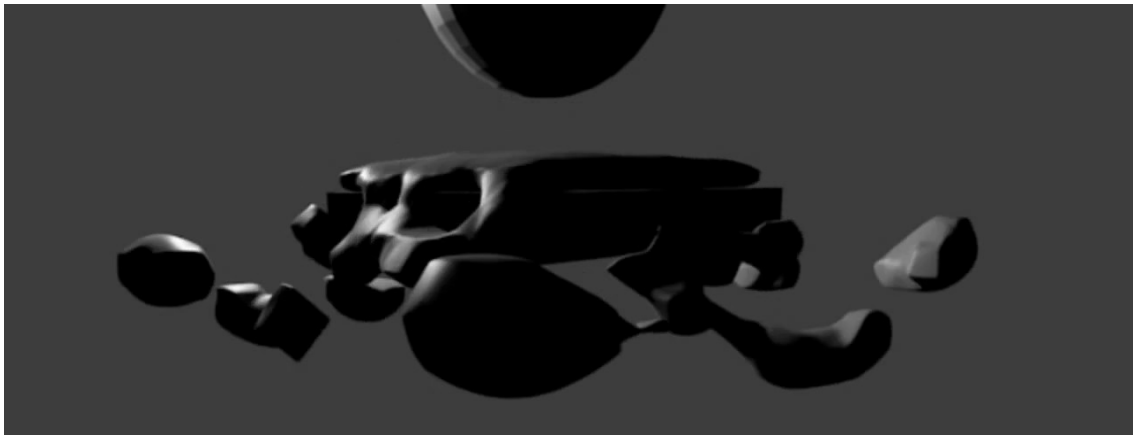
La scheda Fluid Obstacle per il nuovo cubo appena creato

Adesso il parallelepipedo è un ostacolo per gli oggetti liquidi presenti all'interno del *Domain*.

Nella scheda *Fluid* abbiamo ora le voci *Volume Initialization*, *Slip Type* ed *Impact*; a parte le prime due, esaminate in questa puntata e nella precedente (in contesti differenti, ma con significati simili), ci resta da esaminare *Impact Factor*, ossia fattore d'impatto, un parametro che non ha corrispondenza nel mondo reale, infatti il tooltip di Blender ci dice che tale parametro:

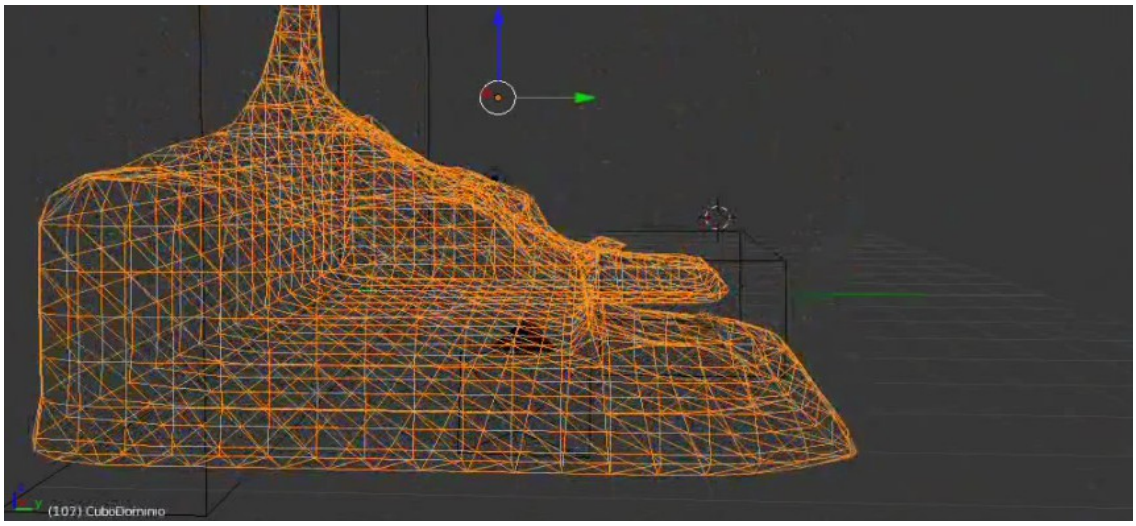
- se uguale a 0, cancellerà il fluido durante l'impatto;
- se uguale a 1, implementerà una collisione standard;
- se maggiore di 1, produrrà reazioni più forti, esagerate.

Nell'immagine nella pagina seguente potete vedere uno screenshot tratto da un rendering di 10 secondi ottenuto (dopo il *baking*) con *Start Time* 0, *End Time* 2 (per renderlo slow-motion), risoluzione del *Domain* pari a 60, *Shading Smoot*h e *SubSurf Catmull-Clark* con 2 suddivisioni.



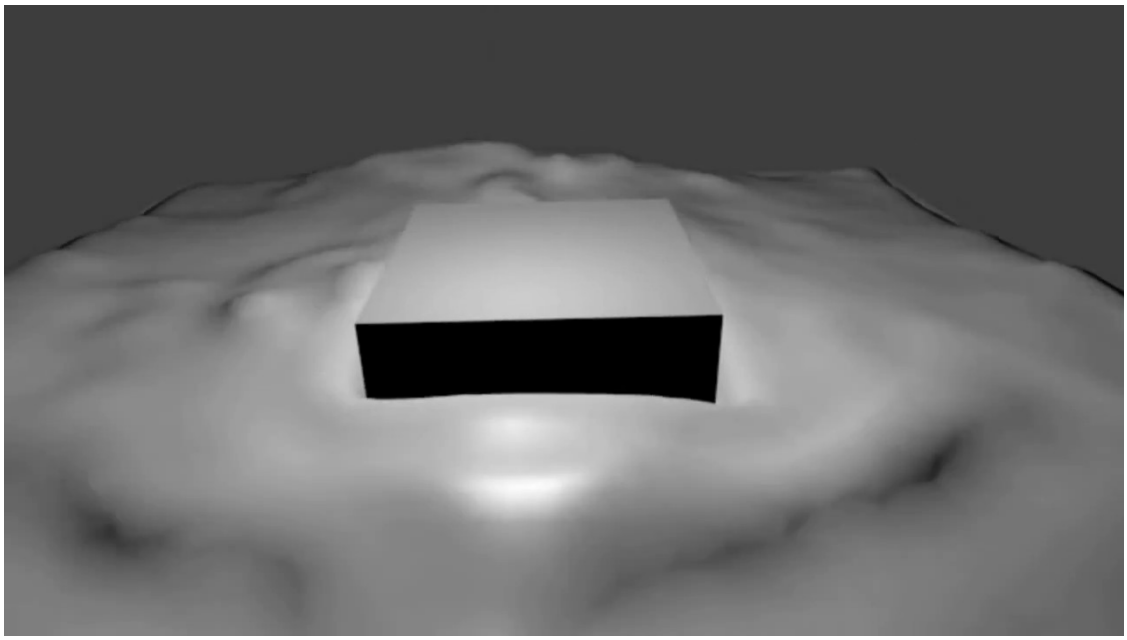
Uno screenshot del rendering relativo al nuovo Baking, questa volta con la presenza di un oggetto Fluid Obstacle

Una considerazione: nel creare simulazioni con oggetti *Obstacle*, conviene utilizzare un valore di risoluzione medio-alto. Ciò è dovuto al fatto che, come detto, il motore dei fluidi suddivide il *Domain* in *Voxel* e calcola le simulazioni secondo la "granularità" dettata dalle dimensioni dei *Voxel*, che sono più piccoli quanto più è alto il valore di *Resolution*; ora, se per caso un *Voxel* dovesse essere più grande, ad esempio, dell'altezza del nostro parallelepipedo, Blender calcolerebbe le interazioni del fluido con le pareti del *Voxel*, per cui vedremmo urtare il fluido contro il nulla o, al contrario, il fluido potrebbe penetrare all'interno dell'oggetto, perché magari questo è più grande dei *Voxel*, anche di parecchie volte, ma parti di esso occupano una piccola porzione in certi *Voxel*, come nel fotogramma dell'immagine seguente (fotogramma tratto da una simulazione realizzata con *Resolution* 20), dove i *Voxel* corrispondenti alla parte alta del cubo ostacolo sono più vuoti che pieni e considerati, quindi, vuoti dal motore dei fluidi di Blender, che li riempie col fluido.



Un fotogramma di una simulazione realizzata con risoluzione 20

Gli oggetti *Obstacle* possono essere anche oggetti animati; il motore dei fluidi, in questo caso, calcolerà automaticamente le collisioni e gli effetti ai vari frames.
Nell'immagine qui sotto potete vedere uno screenshot tratto da una simulazione realizzata utilizzando, come oggetto *Fluid* di base, un parallelepipedo posto sul fondo del cubo *Domain*, per realizzare un fluido di partenza pressoché immobile; è presente, inoltre, un oggetto Cube che è anche *Obstacle* nella simulazione dei fluidi; all'inizio, il Cube si trova a metà altezza all'interno del *Domain*, ma in 48 frames arriva sul fondo dello stesso (con un'animazione realizzata mediante le *chiavi IPO*, nulla di particolarmente complicato).



Un fotogramma di una simulazione realizzata con l'Obstacle in movimento

Realizzata l'animazione della caduta del cubo, si può quindi lanciare il *baking* della scena, effettuato in questo caso con *Start Time* 0, *End Time* 10 e *Risoluzione* 60.
Il cubo è un *Obstacle* con *Init Volume* impostato a *Shell* e *Factor* a 5, per amplificare l'effetto dovuto alla collisione tra il cubo e il fluido.
Una nota importante: Blender dice che l'opzione *Export Mesh* nella scheda *Fluid Obstacle* degli oggetti andrebbe selezionata solo quando gli oggetti vengono animati deformandoli, ad esempio mediante le armature; tuttavia, per ottenere l'effetto visibile a video ho dovuto attivarla anche se l'animazione creata è di tipo *IPO*. Consiglio, quindi, di attivarla comunque, anche se questo porterà a tempi di *baking* maggiori.

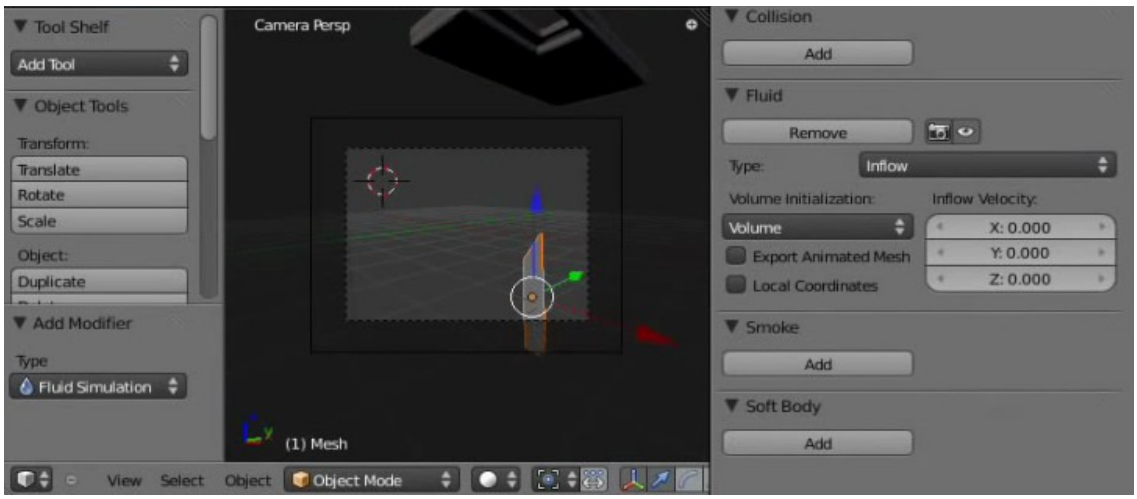
* * *

Basi del Motore dei Fluidi - 3: inflow e outflow

In questa terza puntata dedicata alle basi del motore dei fluidi di Blender 2.5 prenderemo in esame gli oggetti *Inflow* e *Outflow.*

Un oggetto *Inflow* è un oggetto che, come suggerisce il nome, immette un po' di fluido nella scena. Per creare un oggetto del genere dobbiamo inserire una mesh, ad esempio un Plane, all'interno del *Domain* della simulazione, e scegliere *Inflow* dal menù *Add Fluid* nella *Properties Window.*
A questo punto l'oggetto è già un emettitore di fluido.

Il liquido creato resterà all'interno del *Domain*, interagendo con gli altri oggetti.



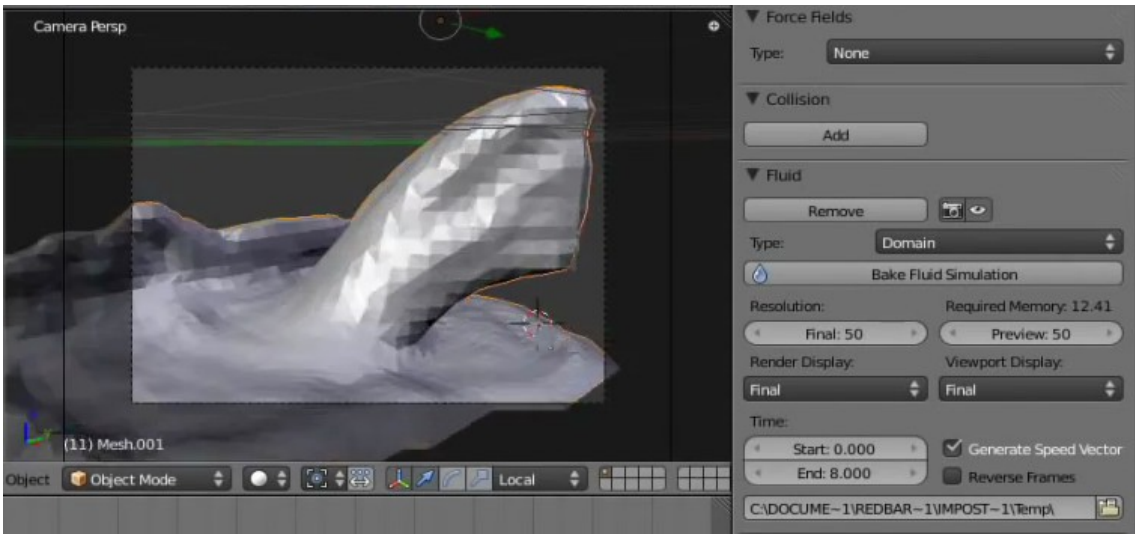
La scheda di un oggetto Fluid Inflow

Anche nella scheda *Fluid Inflow* è presente il menù *Volume Initialization*, trattato nelle puntate precedenti; visto che, in questo caso, stiamo usando un Plane, che quindi non ha un volume, non ha senso utilizzare come *Init* proprio la voce *Volume*, selezionata di default, per cui cambiamola in *Shell*, ossia la superficie definita dal piano.

Nella sezione *Inflow Velocity* possiamo impostare, come suggerisce il nome, una velocità iniziale di fuoriuscita del liquido dalla mesh generatrice, impostando velocità differenti per le varie direzioni.

Definiti i parametri, non ci resta che avviare il *Baking* della simulazione e osservare il risultato.

Nell'immagine qui sotto potete vedere uno screenshot tratto da una simulazione che mostra *Inflow* in azione su di un piano perpendicolare alla griglia XY di Blender; la simulazione è stata effettuata lanciando il *baking* con *Resolution* 50 e velocità iniziale 1 lungo l'asse Y.



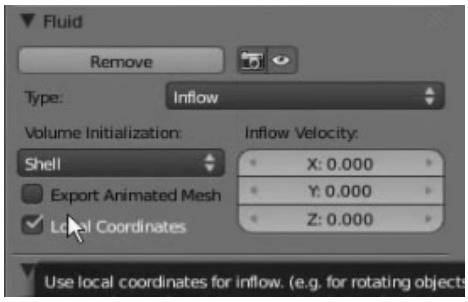
Uno screenshot della prima simulazione realizzata con Inflow

È possibile notare, sul fondo del *Domain*, l'aumentare della quantità di fluido presente nella scena, per via dell'azione di *Inflow.*

I valori espressi in *Inflow Velocity* si riferiscono, di default, agli assi del sistema di riferimento GLOBALE della scena, cosa che può portare ad effetti indesiderati con oggetti *Inflow* ruotati rispetto alla posizione iniziale; per impostare come sistema di riferimento quello LOCALE dell'oggetto, quindi, bisogna selezionare la casella *Local Coordinates* nella scheda *Fluid Inflow.*

Potete verificare la disposizione degli assi locali di una mesh cambiando il menù *Orientation* da *GLOBAL* in *LOCAL* nell'header di ogni finestra di navigazione 3D.

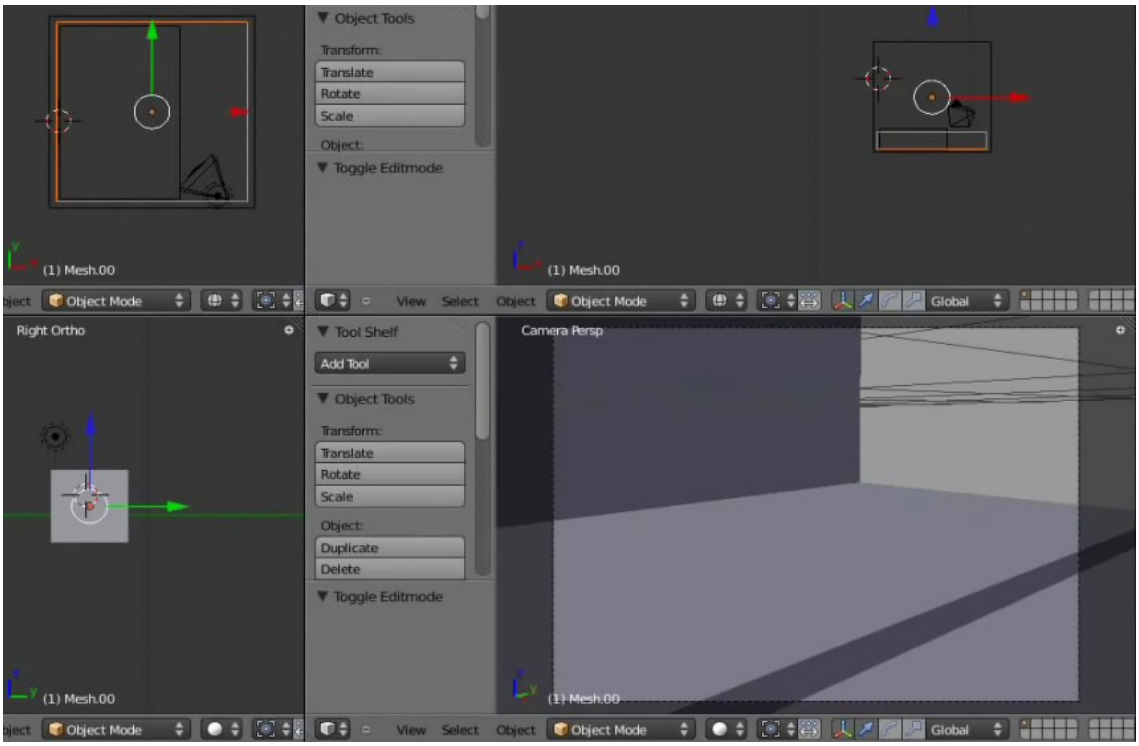
Per la voce *Export Animated Mesh* valgono le considerazioni fatte nella puntata precedente, ossia che andrebbe attivato in caso di animazioni dell'oggetto.



Visualizzare la disposizione degli assi locali di un oggetto

Adesso togliamo il Plane dalla nostra simulazione ed aggiungiamo, all'interno del *Domain*, sul fondo, un parallelepipedo, con la base grande quanto quella del *Domain*. Rendiamo tale parallelepipedo un oggetto *Fluid*, in modo da creare una specie di vasca d'acqua all'interno del *Domain*.

Aggiungiamo inoltre, sul fondo del dominio della simulazione, un altro oggetto, ad esempio un cubo o un parallelepipedo; con tale oggetto selezionato, apriamo la scheda *Physics* nella *Properties Window*, clicchiamo su *Add* in *Fluid* e scegliamo, dal menù *Type*, la voce *Outflow*.



La nuova scena, con il Plane FLUID e il cubo OUTFLOW all'interno del cubo-dominio

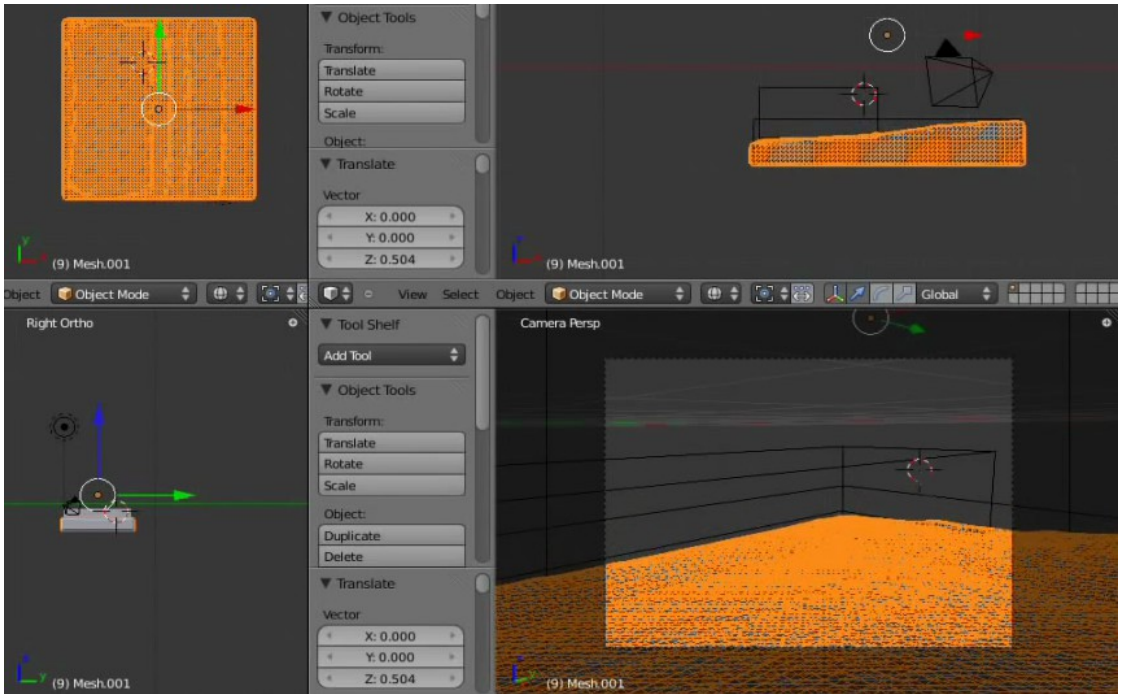
Come suggerisce il nome, un oggetto *Outflow* fa l'esatto opposto di un *Inflow*, ossia toglie del liquido dalla scena, ed in particolare eliminerà tutte le particelle del liquido che, nel corso della simulazione, impatteranno con l'oggetto *Outflow*.

Anche per Outflow abbiamo i parametri *Volume Initialization* e *Export Animated Mesh,* con significati simili a quelli visti in altri contesti nelle puntate precedenti.

I parametri della scheda OUTFLOW

Per *Volume Initialization* conviene lasciare la modalità presente di default, ossia *Volume*.

Lanciamo quindi il *Bake* ed osserviamo la simulazione così ottenuta.
Nell'immagine qui sotto potete vedere uno screenshot tratto da una simulazione che mostra l'effetto di *Outflow*: il livello del liquido all'interno del cubo inizia a diminuire, e ciò è dovuto alla presenza di *Outflow* che, frame per frame, ne elimina un po'.



Uno screenshot della simulazione ottenuta con Outflow

Ovviamente, la dimensione dell'oggetto *Outflow* e la durata della simulazione, espressa come differenza tra *End Time* e *Start Time*, giocano un ruolo fondamentale nel determinare quanto velocemente verrà eliminato il fluido presente nella scena.

* * *

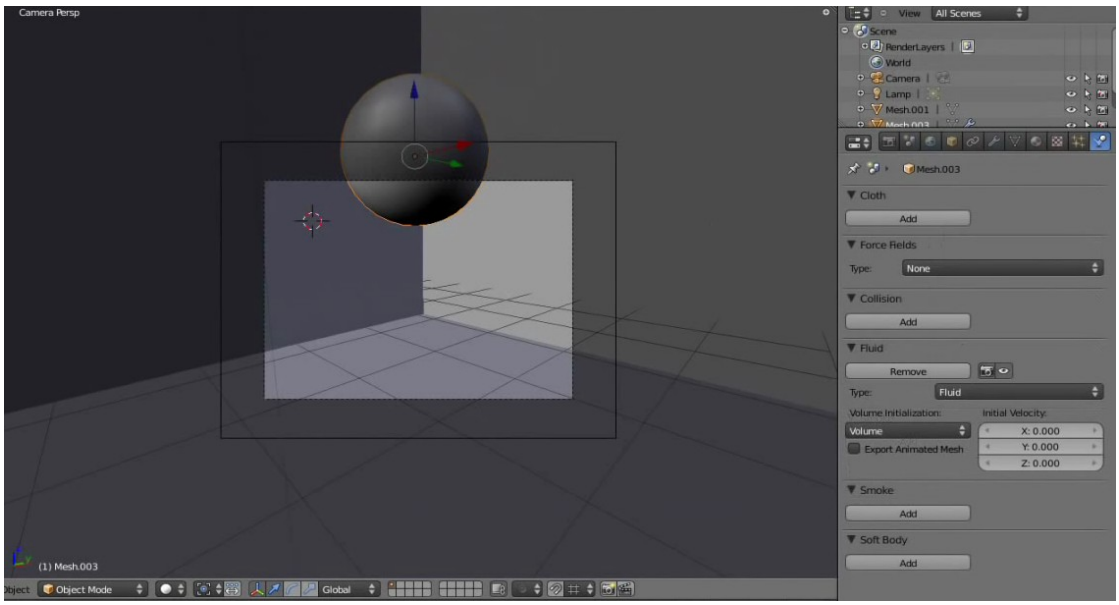
Basi del Motore dei Fluidi - 4: particelle, particles

Eccoci alla quarta ed ultima puntata di un ciclo di tutorial dedicati alle basi del motore dei fluidi in Blender 2.5. Questa volta parleremo delle *particelle* proprie dei fluidi.

È possibile aggiungere, in una scena, dei sistemi particellari "classici", come *Emitter* ed *Hair*, ma per i fluidi sono disponibili tre sistemi particellari propri, detti *Drops*, *Float* e *Tracer*.

Tali sistemi sono legati al *Domain* della simulazione, nel senso che saranno attivati per tutti i fluidi presenti nella scena e le loro impostazioni si trovano sia nella scheda del *Domain* che in una scheda a parte, come vedremo.

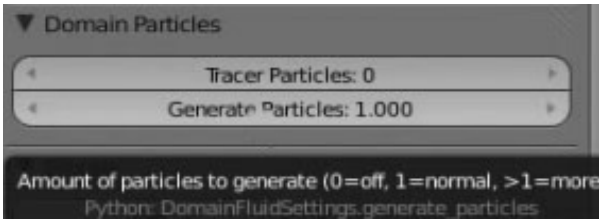
Nella scena visibile qui sotto è presente una sfera *Fluid* posta all'interno di un *Domain* cubico.



La scena utilizzata in questo tutorial: una sfera Fluid posta all'interno di un Domain cubico

Selezioniamo l'oggetto *Domain* e apriamo la scheda *Physics* nella *Properties Window.*

Per aggiungere le particelle, nella sezione *Domain Particles* dobbiamo impostare un valore maggiore di 0 per il campo *Generate Particles*; il valore 1 è un valore considerato "normale", nel senso che genera una buona quantità di particelle, ma ovviamente possiamo impostare anche valori maggiori di 1.



La sezione Domain Particles della scheda Fluid Domain

Altro parametro importante per attivare la simulazione delle particelle del fluido è *Surface Subdivision*, posto all'interno della sezione *Domain Boundary*; in questo caso si consiglia di impostare un valore maggiore o uguale a 2.

Da notare che a valori maggiori corrisponderanno tempi di *baking* più elevati; in genere è sufficiente, comunque, impostare come valori 2, 3, 4 o 5. Va detto inoltre che, in generale, la quantità di particelle generate dipende anche da altri fattori, come ad esempio la velocità iniziale del fluido o l'*Impact Factor* degli oggetti *Fluid Obstacle*, se presenti.

Tecnicamente, già con queste semplici impostazioni abbiamo aggiunto le particelle del motore dei fluidi alla simulazione e potremmo anche avviare un *baking*, ma è meglio studiare questo sistema in dettaglio...

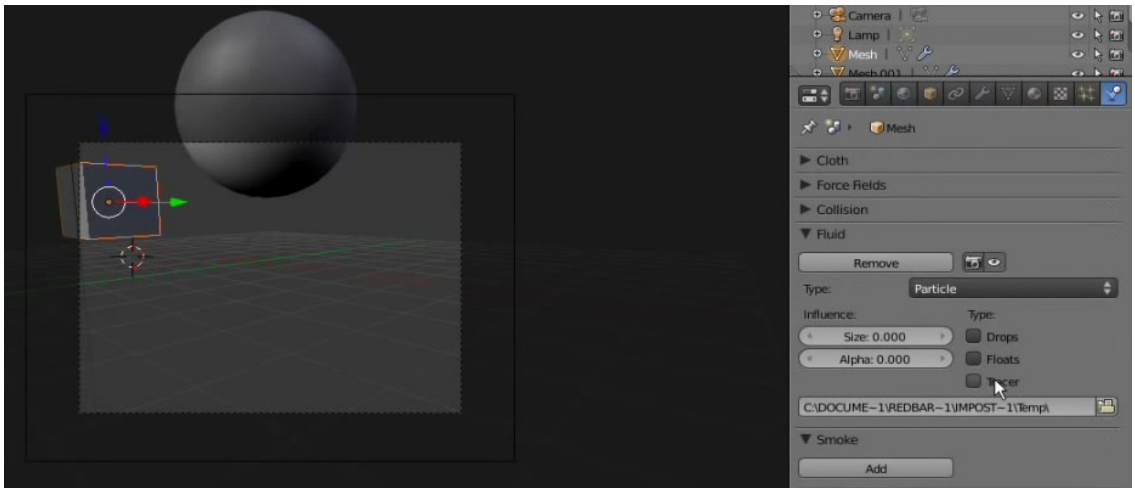
Come anticipato, le particelle generate non sono di un unico tipo, ma appartengono a 3 tipi differenti: *Drops*, *Floats* e *Tracers*.

Le particelle *Drops* sono gli schizzi, ossia le particelle generate e che si allontanano dal fluido in seguito ad una collisione dello stesso con un ostacolo o con le pareti del *Domain*.

Le particelle *Floats* sono piccole particelle che si formano sulla superficie del fluido e che servono a fornire alla mesh un aspetto meno omogeneo.
Le particelle *Tracers* sono particelle che "seguono", in un certo senso, la mesh del fluido; sono utili per simulare, ad esempio, il vapore.

Impostando *Generate Particles* e *Surface Subdivisions* come abbiamo visto precedentemente, le particelle *Drops* e *Floats* vengono create automaticamente; le particelle *Tracers*, invece, vanno attivate manualmente impostando un valore maggiore di 0 nel campo *Tracer Particles*, all'interno della sezione *Domain Particles* nella scheda *Physics* dell'oggetto *Domain*.

La cosa veramente interessante, comunque, è che possiamo definire un aspetto proprio per ciascun sistema particellare. Vediamo come. Aggiungiamo alla nostra scena una mesh qualsiasi, ad esempio un cubo. Tale oggetto può trovarsi anche all'esterno del *Domain*.
Con tale oggetto selezionato, apriamo la scheda *Physics* all'interno della *Properties Window*, clicchiamo su *Add* nella sezione *Fluid* e scegliamo, dal menù *Type*, la voce *Particle*.



La scheda Fluid Particle per il nuovo cubo appena creato

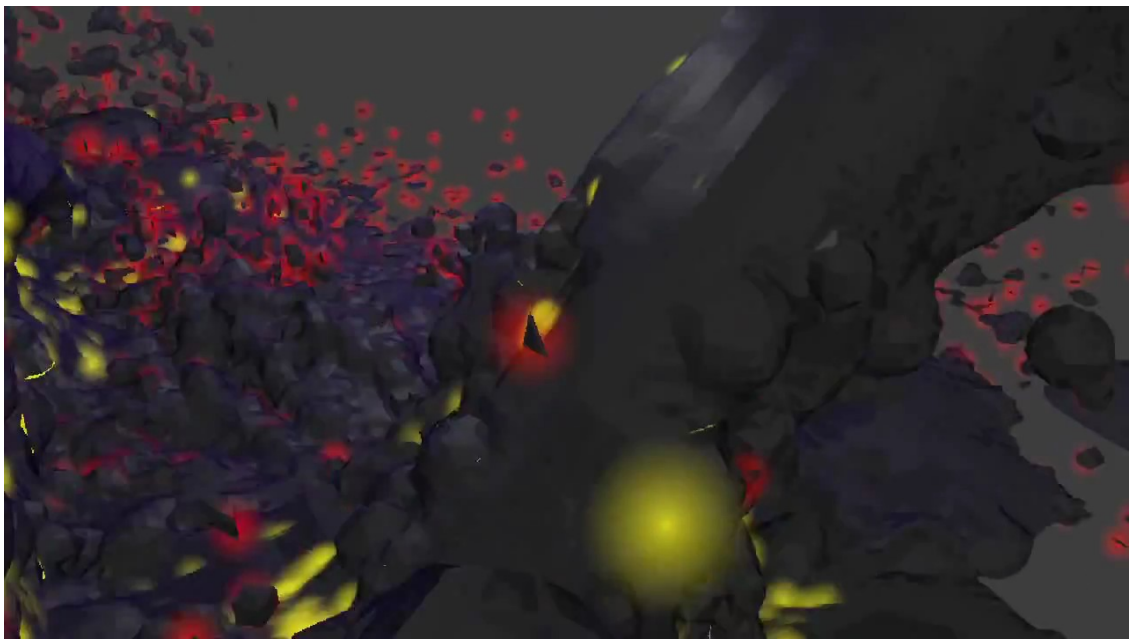
Nella scheda *Fluid Particle* sono presenti le caselle *Drops*, *Floats* e *Tracer*; in pratica, possiamo scegliere se assegnare il Material creato per la mesh selezionata alle particelle di tipo *Drops*, oppure a quelle di tipo *Floats* o a quelle di tipo *Tracer.* La selezione, comunque, non è esclusiva: per tale oggetto possiamo selezionare due voci o anche tutte e tre, e assegnare il Material dell'oggetto ai relativi gruppi di particelle.

Per assegnare Materials differenti ai vari tipi di particelle, sarà sufficiente creare altre Mesh (ciascuna col proprio Material, anche provvisto di Textures se vogliamo), rendere tali mesh oggetti *Fluid Particle* ed assegnare a ciascuna mesh uno o più tipi di particelle *Fluid*.

In genere, i Materials utilizzati per le particelle sono di tipo *Halo*, ma ovviamente potete utilizzare il Material che preferite: questo è solo un suggerimento.

Vediamo ora un paio di simulazioni.
La prima simulazione è stata realizzata con un Plane *Inflow*, *risoluzione* del dominio 65 e *Surface Subdivision* a 2.

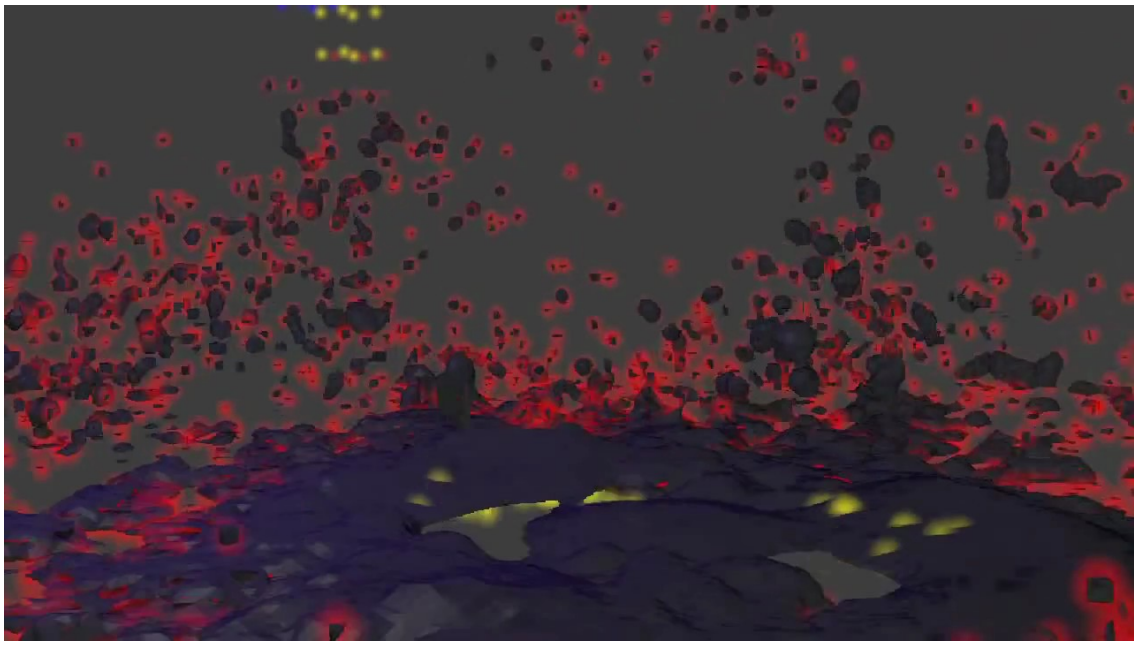
Le particelle *Float* sono gialle, i *Tracer* blu (e mi rendo conto che questa è stata una scelta poco felice, visto il colore del fluido) mentre le *Drops* sono rosse. Tutti e tre i Material sono di tipo *Halo*.
La qualità non è per niente interessante, visto che risoluzione del *Domain* e *Surface Subdivisions* delle particelle avevano valori bassi, ma si tratta di una simulazione dimostrativa senza troppe pretese; anche i colori delle particelle *Drops* e *Floats* sono un po' troppo forti, ma quantomeno le fanno risaltare, mostrando il contributo di particelle portato dalla simulazione *Fluid Particles*.



Screenshot della prima simulazione

La seconda simulazione è stata realizzata con una sfera *Fluid*, risoluzione 65, *Generate Particles* e *Surface Subdivision* a 3. Nel rendering, le "luci" visibili sullo sfondo sono in realtà i vertici dei tre cubi utilizzati come oggetti *Fluid Particles* per impostare i Material *Halo* delle particelle.

Nella parte finale di questo secondo rendering è possibile notare meglio la creazione degli schizzi.



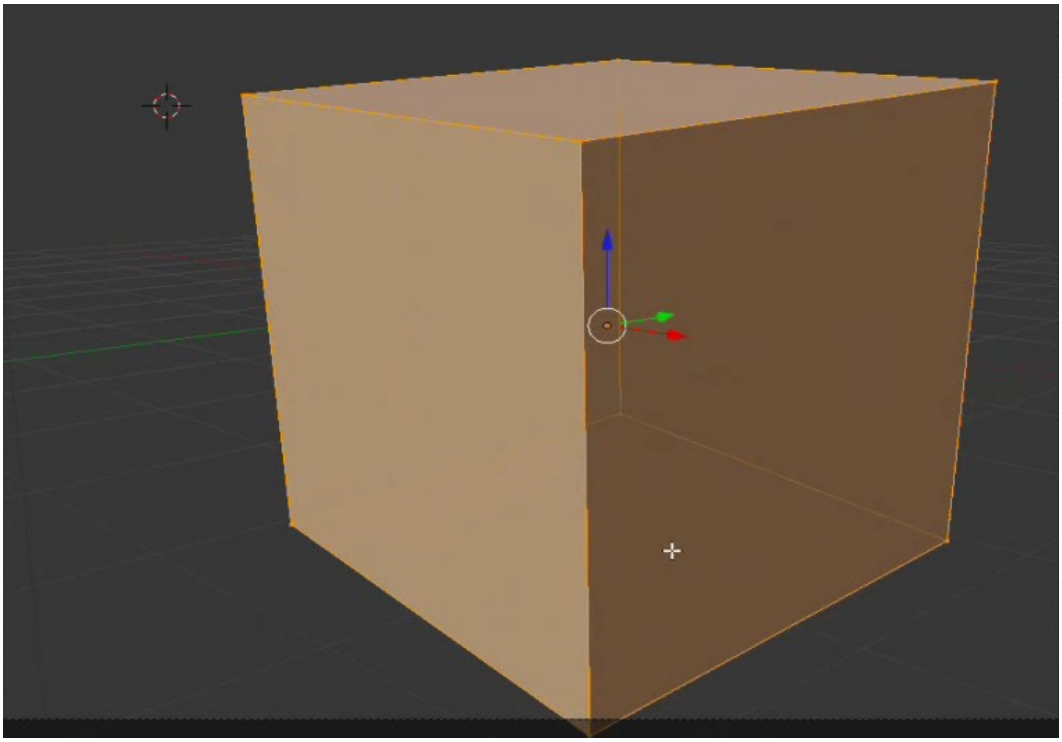
Screenshot della seconda simulazione

* * *

Utilizzare i Vertex Groups. Il problema del cubo

In Blender, i *Vertex Group* sono, come suggerisce il nome, gruppi o raggruppamenti di vertici di un oggetto. Raggruppare un insieme di vertici, dando loro un nome significativo, per poi selezionarli velocemente, può tornare utile in vari campi, ad esempio per associare le ossa di una *Armature* ad una mesh, oppure per realizzare facilmente la *mappatura UV* delle Textures ed altro ancora.
Ecco perché è importante sapere come creare e modificare dei *Vertex Groups*, ed è quello che vedremo in questo breve tutorial.

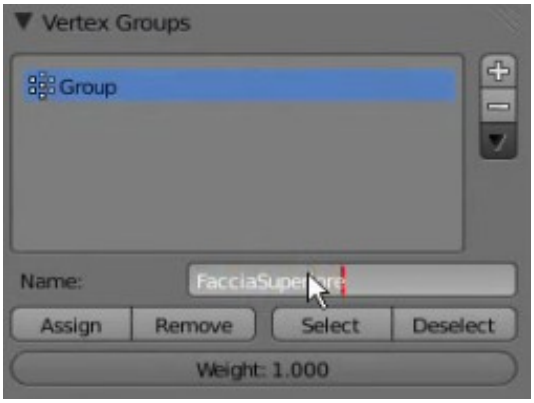
Con una mesh selezionata, ad esempio un cubo, passiamo in Edit Mode.
Se di default ci sono tutti gli spigoli selezionati, deselezioniamoli premendo *A*.



Il cubo, con tutti gli elementi deselezionati, in Edit Mode

Apriamo la scheda *Object Data* all'interno della *Properties Window* e clicchiamo sul pulsante col segno + all'interno della sezione *Vertex Groups*.

Abbiamo creato un gruppo di vertici vuoto per tale mesh, con nome di default *Group*; rinominiamolo in "*FacciaSuperiore*" digitando tale nome nel campo *Name* in basso.

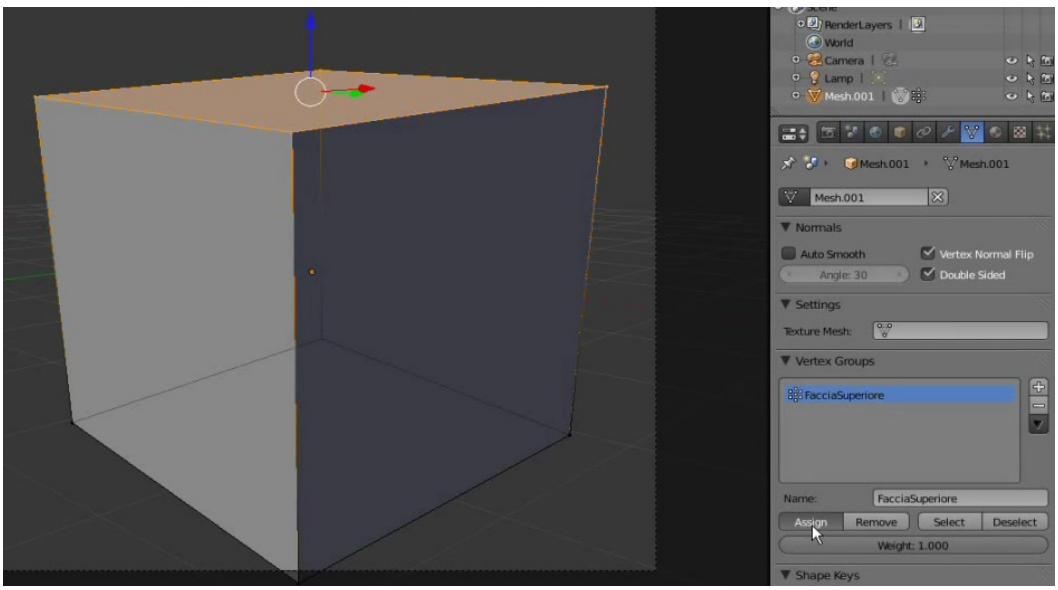


Creazione del Vertex Group "FacciaSuperiore"

Parliamo ora del concetto di selezione: per poter lavorare con un particolare *Vertex Group*, bisogna selezionarlo dall'elenco, cosa che va fatta cliccando sul suo nome col mouse e, subito dopo, sul tasto *SELECT* in basso. In questo modo, il *Vertex Group* scelto è effettivamente selezionato, attivo.

Il gruppo appena creato è selezionato di default ma è, inizialmente, vuoto, per cui spostiamoci per un attimo in una finestra *3D View*.

Selezioniamo i vertici che compongono la faccia superiore del Cubo e torniamo alla *Properties Window* e alla sezione *Vertex Group*; qui, clicchiamo ora su *ASSIGN* per assegnare, appunto, i vertici selezionati (4, nel nostro caso) al *Vertex Group* "*FacciaSuperiore*".

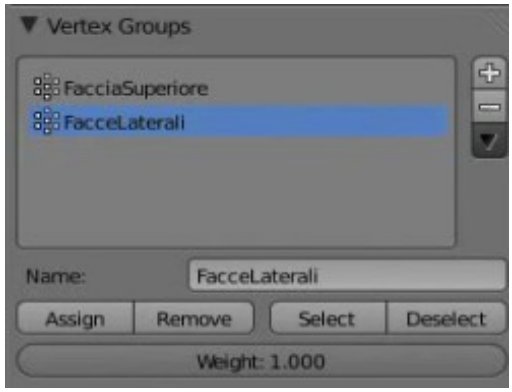


Assegnazione degli elementi selezionati al Vertex Group attivo

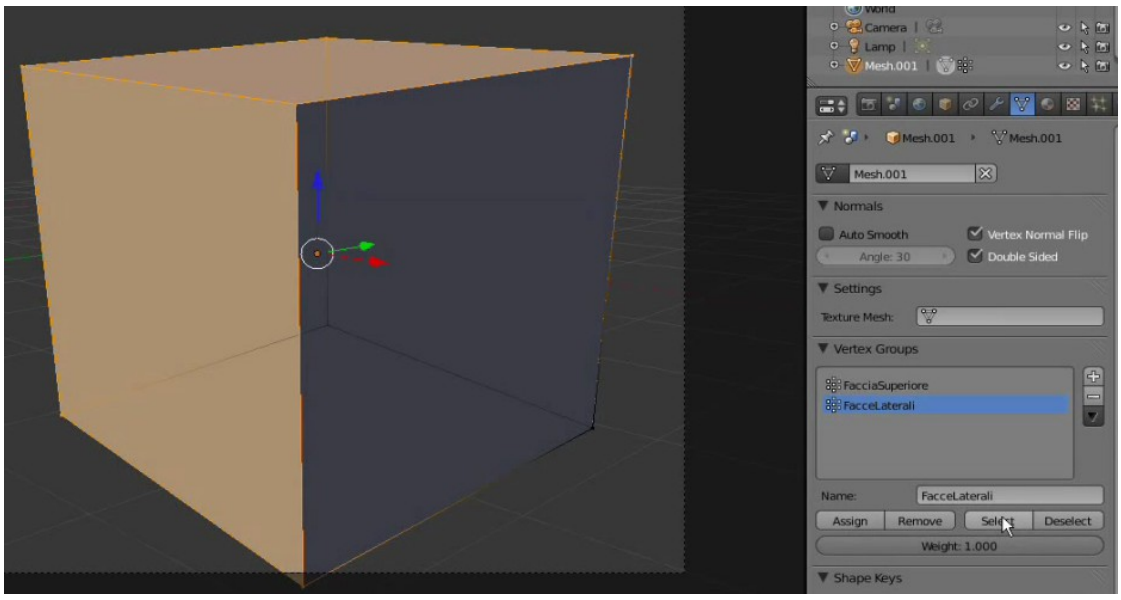
In questo momento stiamo lavorando con il gruppo "*FacciaSuperiore*" selezionato, per cui per poter creare un altro *Vertex Group* deselezioniamolo cliccando su *DESELECT*; potremo comunque selezionarlo in seguito cliccando nuovamente su *SELECT*.

Cliccando su *DESELECT*, verrà deselezionata anche la faccia del cubo all'interno delle finestre *3D View*; questa, volendo, è una conferma del fatto che abbiamo assegnato correttamente il gruppo di vertici della mesh a quel *Vertex Group*.

Creiamo un nuovo *Vertex Group* cliccando nuovamente sul pulsante col segno + e chiamiamo questo nuovo gruppo "*FacceLaterali*". Il gruppo è selezionato di default. In una finestra *3D View* selezioniamo i vertici che compongono la faccia superiore del cubo e altri due vertici, per aggiungere anche una faccia laterale, dopodiché clicchiamo su *ASSIGN* nella finestra *Object Data*. Ho sbagliato di proposito la selezione per mostrare alcune funzionalità della sezione *Vertex Group*.



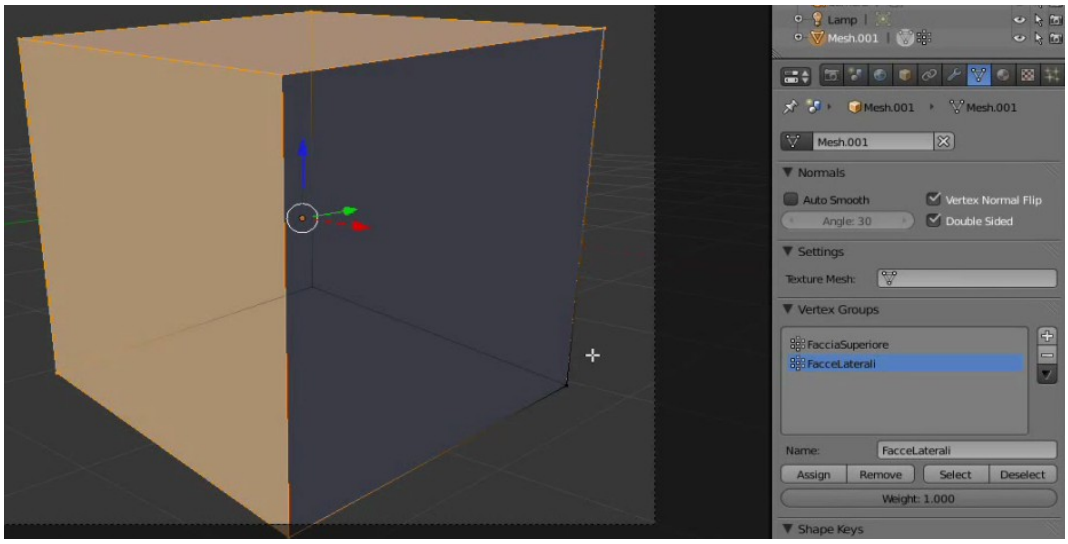
La creazione del Vertex Group "FacceLaterali"



L'assegnazione (con selezione errata) di alcuni vertici al nuovo Vertex Group

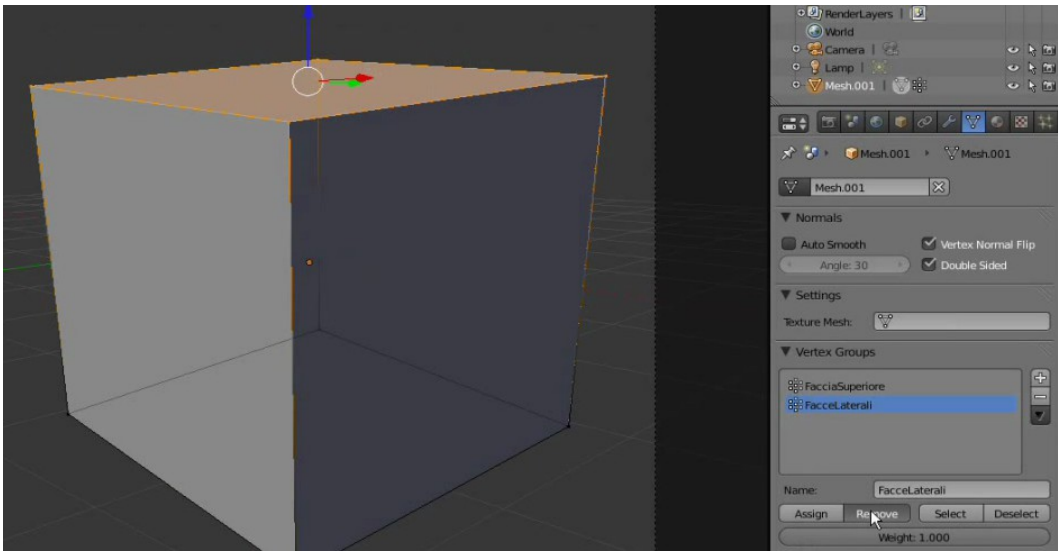
Deselezioniamo anche questo gruppo cliccando su *DESELECT*.

A questo punto possiamo notare, per prima cosa, che nessuno ci vieta di assegnare un vertice a più di un *Vertex Group*, infatti i vertici della faccia superiore sono presenti sia in "*FacciaSuperiore*" che in "*FacceLaterali*", cosa che possiamo verificare selezionando un gruppo nell'elenco e cliccando su *SELECT*, ricordandoci poi di deselezionarlo prima di passare ad un altro *Vertex Group*.
Nel gruppo "*FacceLaterali*", comunque, è presente anche la faccia superiore del cubo, che non dovrebbe stare lì.



Nel gruppo "FacceLaterali" è presente anche la faccia superiore del cubo

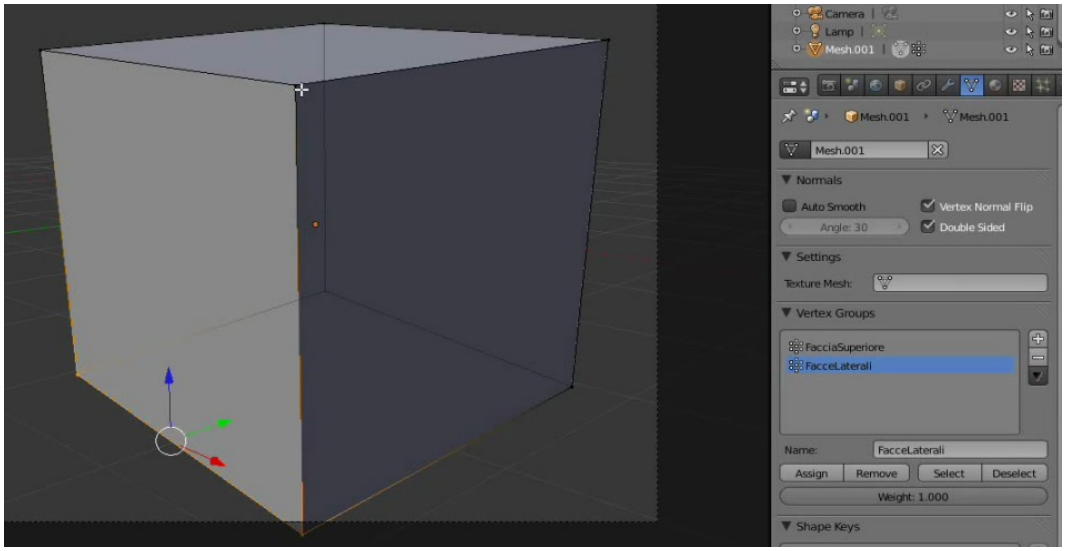
Selezioniamo quindi questo gruppo cliccando su *SELECT*, poi in una finestra *3D View* premiamo *A* per deselezionare tutti i vertici che lo compongono e, a questo punto, selezioniamo solo i vertici che compongono la faccia superiore del cubo e clicchiamo sul pulsante *REMOVE* nella sezione *Vertex Group* della *Properties Window*.



La rimozione della faccia superiore dal gruppo con Remove

La faccia è ancora selezionata: c'è da dire che qui l'aggiornamento di Blender non funziona in maniera eccezionale, comunque ora clicchiamo su *DESELECT* nella sezione *Vertex Group* e deselezioniamo tutti i vertici premendo *A* in una *3D View*.

Per vedere lo stato del nostro *Vertex Group* "*FacceLaterali*", clicchiamo nuovamente su Select.
Notiamo subito un problema: adesso, infatti, c'è solo uno spigolo selezionato, quello in basso; ciò è dovuto al fatto che togliendo dalla selezione i vertici che componevano la faccia superiore, vertici che erano condivisi anche dagli spigoli laterali, è rimasto selezionato solo uno spigolo.

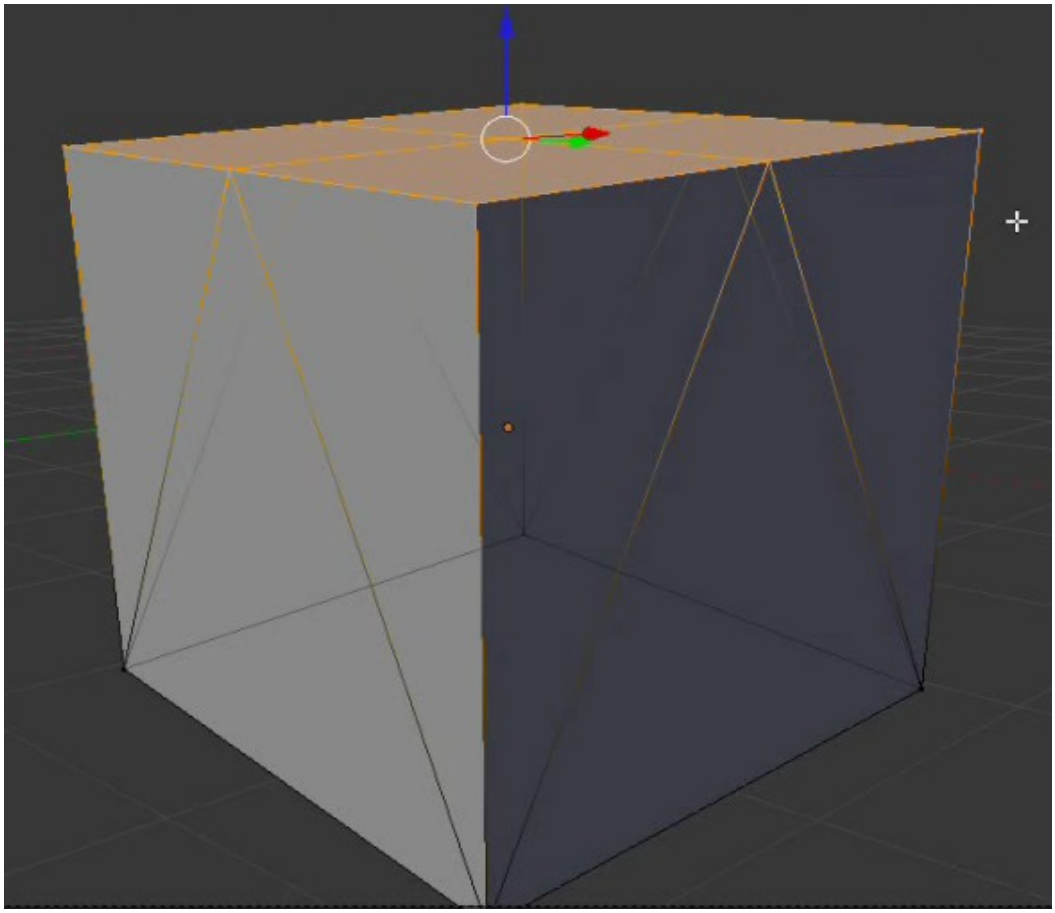


Adesso è "FacceLaterali" a presentare qualche problema...

Questo errore è dovuto alla natura della mesh e va risolto suddividendo la faccia superiore.

Clicchiamo quindi su *DESELECT* per lavorare senza *Vertex Group* attivi, poi in una finestra *3D View* selezioniamo i vertici che compongono la faccia superiore del cubo, premiamo *W* e, dal menù che apparirà a video, scegliamo *Subdivide*.

Abbiamo aggiunto un nuovo vertice, con i relativi spigoli, all'interno della faccia superiore.



La suddivisione della faccia superiore

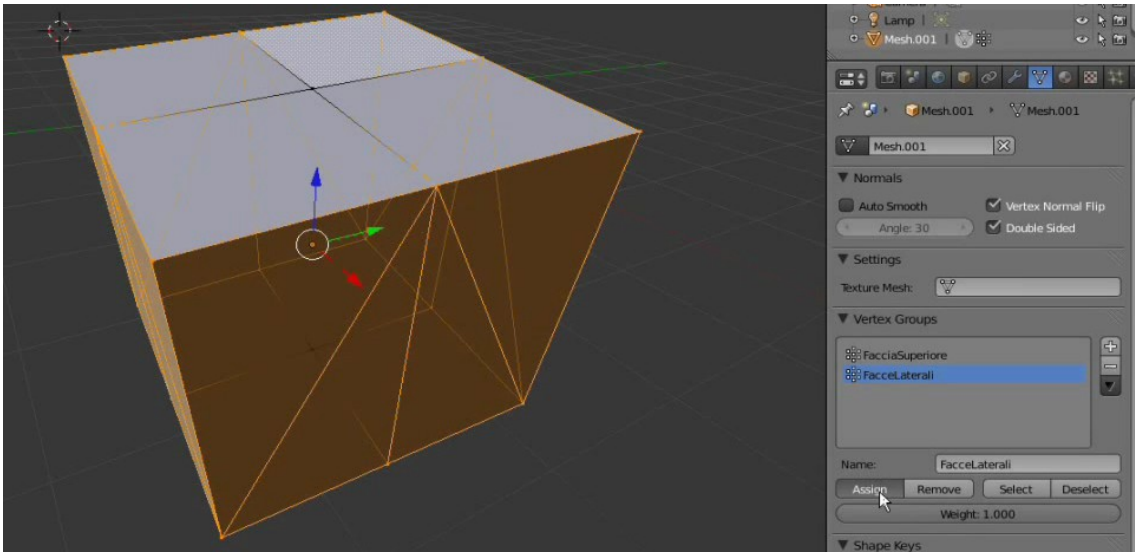
Ripetiamo la stessa operazione di suddivisione per la faccia inferiore del cubo, per evitare di fare lo stesso errore tra poco.

Premiamo *A* per deselezionare tutto.

Nella *Properties Window*, scegliamo il *Vertex Group* "*FacceLaterali*" e clicchiamo su *SELECT*.

Adesso, in una finestra *3D View*, selezioniamo tutti i vertici che compongono le facce laterali del cubo, facendo attenzione ovviamente a non selezionare i nuovi vertici creati al centro delle facce superiore e inferiore.

Effettuata questa selezione, clicchiamo su *ASSIGN* nella *Properties Window* per assegnare i vertici selezionati al *Vertex Group* "*FacceLaterali*", attualmente attivo.



L'assegnazione delle facce laterali del cubo al Vertex Group

Questa volta, se abbiamo fatto tutto come da copione, la selezione sarà corretta; nel mentre, abbiamo visto come risolvere il problema, presentatosi per la faccia superiore del cubo, di selezionare certi spigoli e certe facce evitando però di selezionare anche altre porzioni che condividono gli elementi selezionati: basta creare una suddivisione all'interno di tali facce.

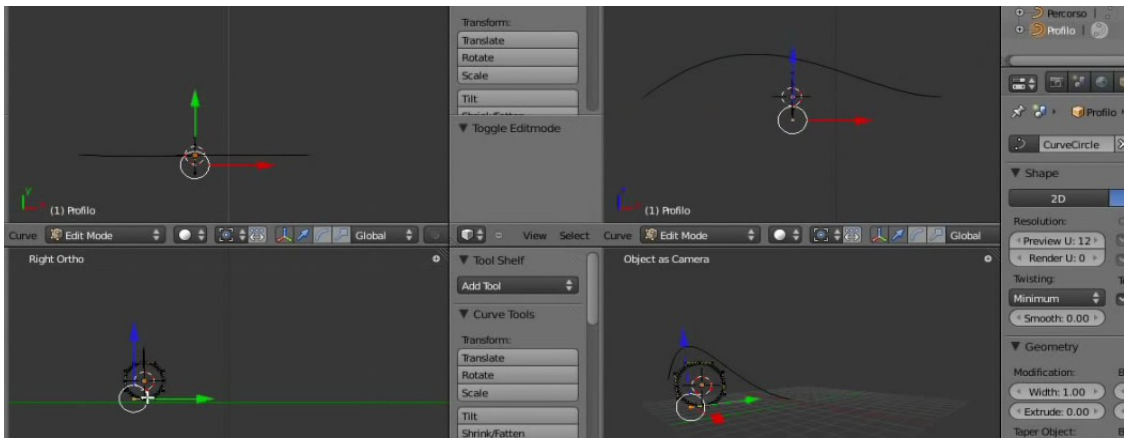
Per questa trattazione, tralasciamo del tutto l'utilizzo del pulsante *Weight* presente sul fondo della sezione *Vertex Group*; brevemente, è possibile assegnare un "peso" ad un *Vertex Group*, da utilizzare con altri strumenti, ad esempio con le *Armature*, per definire con quale intensità dovranno essere applicate certe modifiche ai vertici presenti nel gruppo. Per operazioni di base come il semplice raggruppamento dei vertici appena visto, comunque, non ci serve.

* * *

Basi della modellazione - Estrusione di curve lungo percorsi (Bevel OB)

In questo brevissimo tutorial vedremo come effettuare un'importante operazione di modellazione in Blender 2.5: l'estrusione lungo un percorso, ossia creare un volume a partire da una curva-profilo estrudendola lungo un'altra curva, detta percorso, e non solo lungo un asse (come avviene con l'estrusione semplice). L'estrusione lungo un percorso consente quindi di generare anche forme complesse partendo semplicemente da profilo e traiettoria.

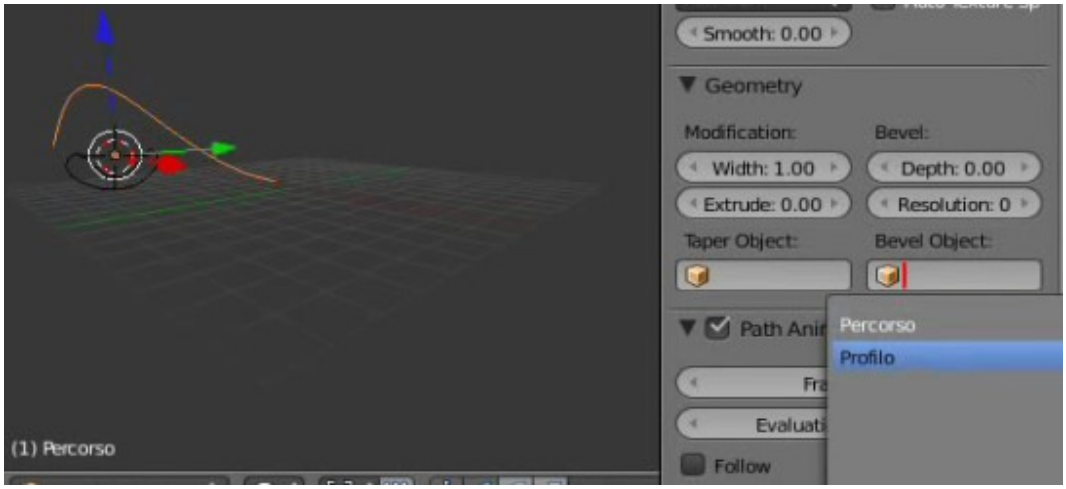
Nella scena di Blender ho inserito una curva di tipo cerchio *Bezier (Bezier Circle)* e una curva *Path*, o *curva percorso*. Ho rinominato il cerchio *Bezier* in "*Profilo*" e la curva *Path* in "*Percorso*".



La scena utilizzata per questo esempio, con la curva Bezier che fa da profilo e la PATH che fa da percorso

Possiamo modificare il cerchio *Bezier* come vogliamo, aggiungendo o spostando vertici.

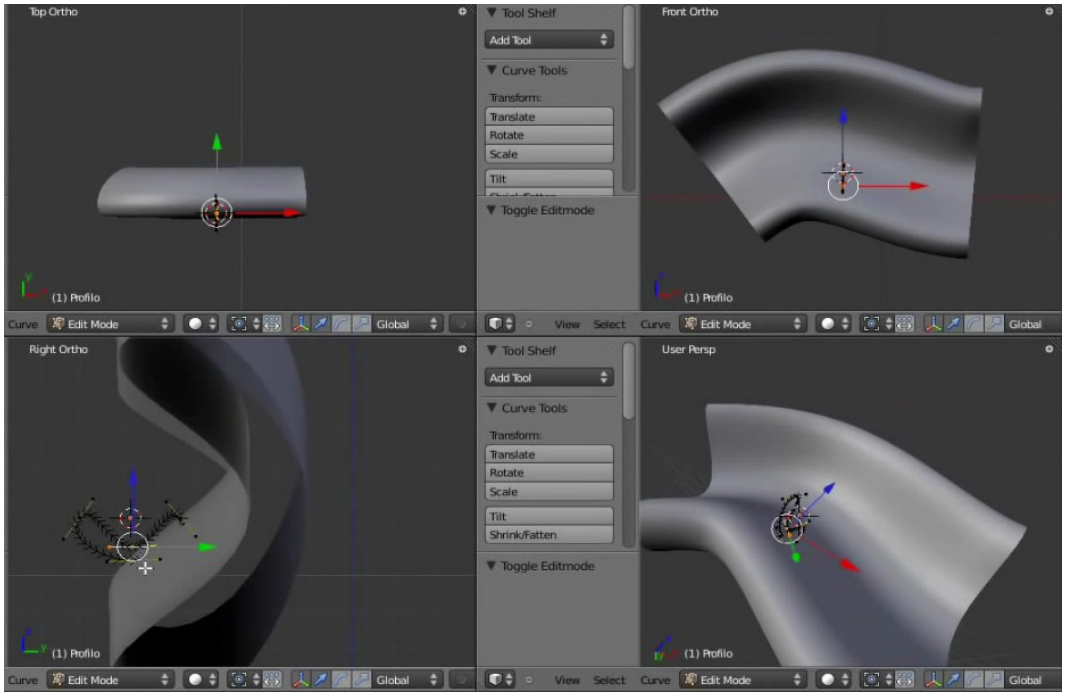
La cosa da notare è che, per estrudere una curva, non dobbiamo selezionare il profilo, ma il percorso. Con la curva percorso selezionata, quindi, apriamo la scheda *Object Data* nella *Properties Window* e scriviamo il nome della curva profilo, nel nostro caso proprio "*Profilo*", all'interno del campo *Bevel Object*, nella sezione *Geometry*.



Scrivere il nome della curva profilo nel campo Bevel Object della curva percorso

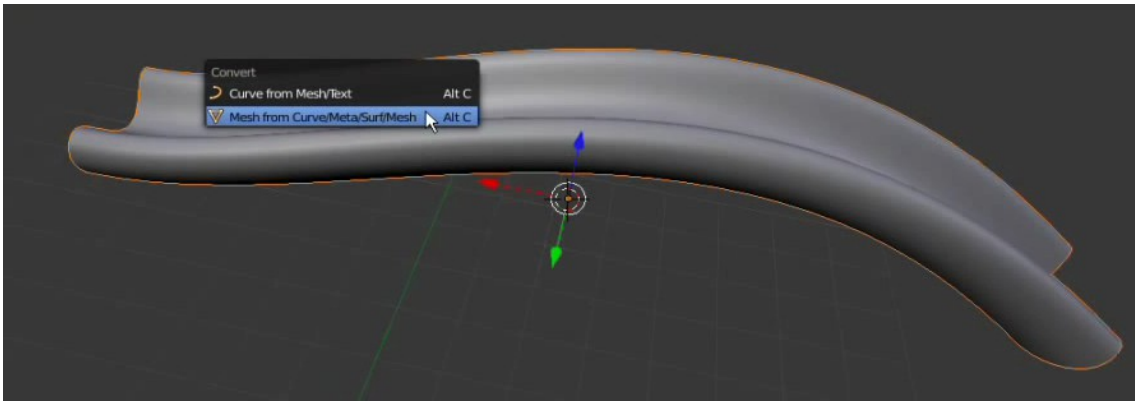
In Blender 2.5, cliccando su tale campo, apparirà un elenco degli oggetti disponibili nella scena, per cui potremo scegliere la curva percorso anche mediante un semplice click: una cosa utile in presenza di molti oggetti e per risparmiare tempo.

Non sempre il risultato sarà quello atteso, o almeno non subito, perché magari l'oggetto profilo era troppo grande; fortunatamente, l'oggetto profilo è ancora disponibile (infatti è l'oggetto percorso che ha subito la trasformazione) ed anzi possiamo selezionare il profilo e modificarlo sia in Edit Mode che in Object Mode, ad esempio spostando qualche vertice o ridimensionandolo: le modifiche verranno applicate in tempo reale anche all'oggetto ottenuto per estrusione lungo un percorso.



Possiamo ancora modificare sia il profilo che il percorso, in maniera interattiva, selezionandoli e passando in Edit Mode

La curva estrusa è, comunque, un oggetto di tipo *Curve*, per cui se vogliamo convertirlo in Mesh, magari per utilizzare qualche modificatore proprio delle Mesh, non disponibile per le curve, dobbiamo premere *ALT C* e scegliere, dal menù che apparirà a video, la voce "*Mesh from Curve/Meta/Surf/Mesh*".



Conversione in mesh dell'oggetto Curve con ALT C e "Mesh from Curve"

Adesso l'oggetto creato è una Mesh modificabile. A questo punto, però, non sarà più possibile modificarlo interattivamente modificando la curva profilo, per cui questa operazione andrebbe eseguita per ultima.

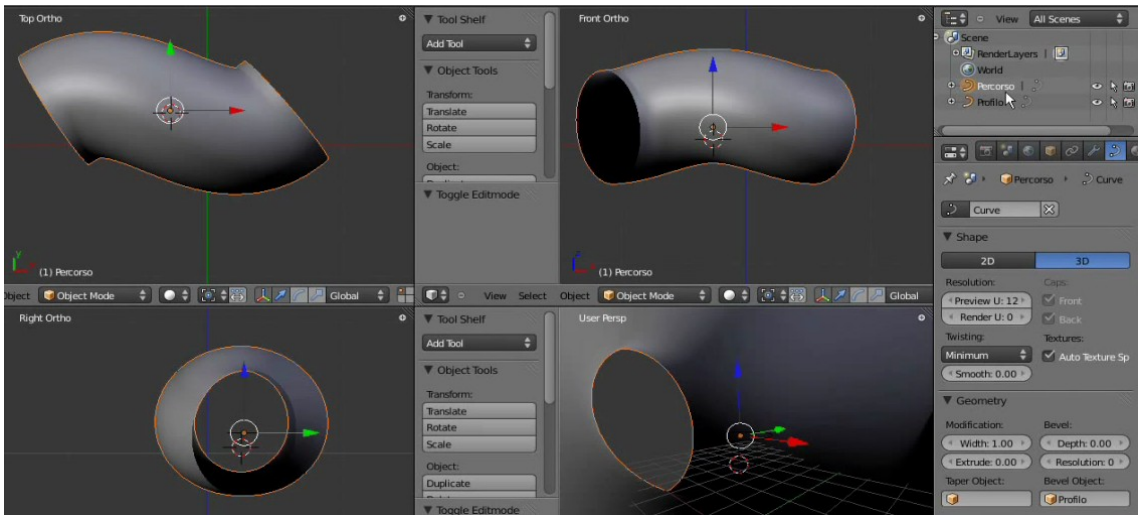
* * *

Basi della modellazione - Curve Taper per curve estruse con Bevel

In questo breve tutorial parleremo dell'argomento *Taper* in Blender 2.5.
In altri pacchetti software tale tecnica è nota, ad esempio, col nome di *loft* o *lofting*.
Taper consente di modificare, mediante una curva di controllo, l'ampiezza dell'estrusione di una curva (estrusione ottenuta mediante *Bevel*, visto in un tutorial precedente).

Procediamo con ordine. Aggiungiamo alla scena una curva *Path*, chiamandola "*Percorso*", e un cerchio Bezier, rinominandolo in "*Profilo*". Possiamo modificare la curva *Path* per definire come preferiamo il percorso di estrusione del cerchio e possiamo modificare anche la forma del cerchio, per definire profili differenti.

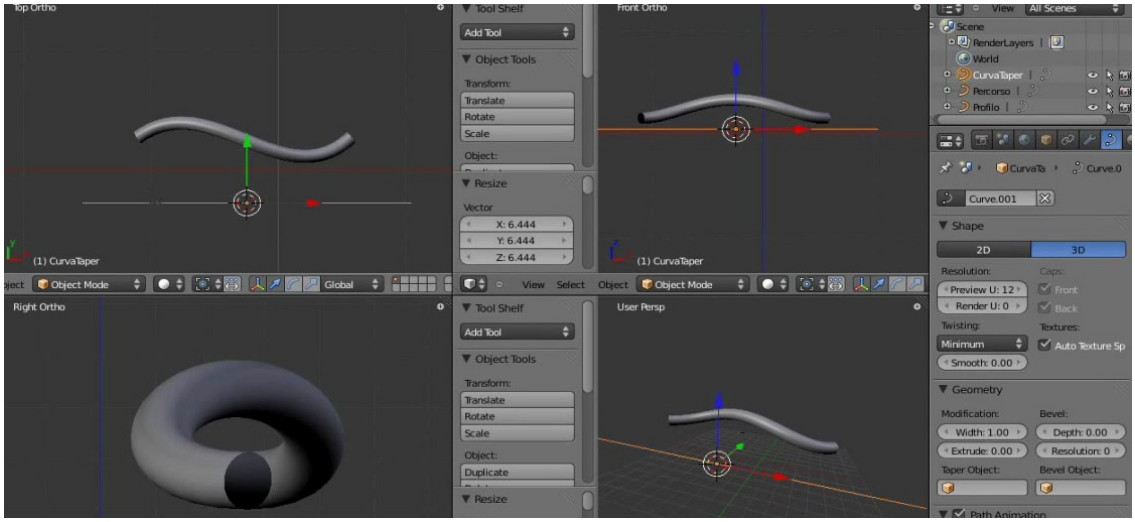
Adesso, con il *Path* selezionato, apriamo la scheda *Object Data* nella *Properties Window* e scriviamo, nel campo *Bevel* all'interno della sezione *Geometry*, il nome del cerchio *Bezier*, ossia, nel nostro caso, "*Profilo*". Abbiamo ottenuto l'estrusione della curva lungo un percorso mediante *Bevel*. L'oggetto è ancora di tipo *Curve* e NON dobbiamo convertirlo in mesh.



La curva profilo utilizzata come Bevel Object per la curva percorso

Ora vediamo finalmente come variare l'ampiezza dell'estrusione nei vari punti della curva.

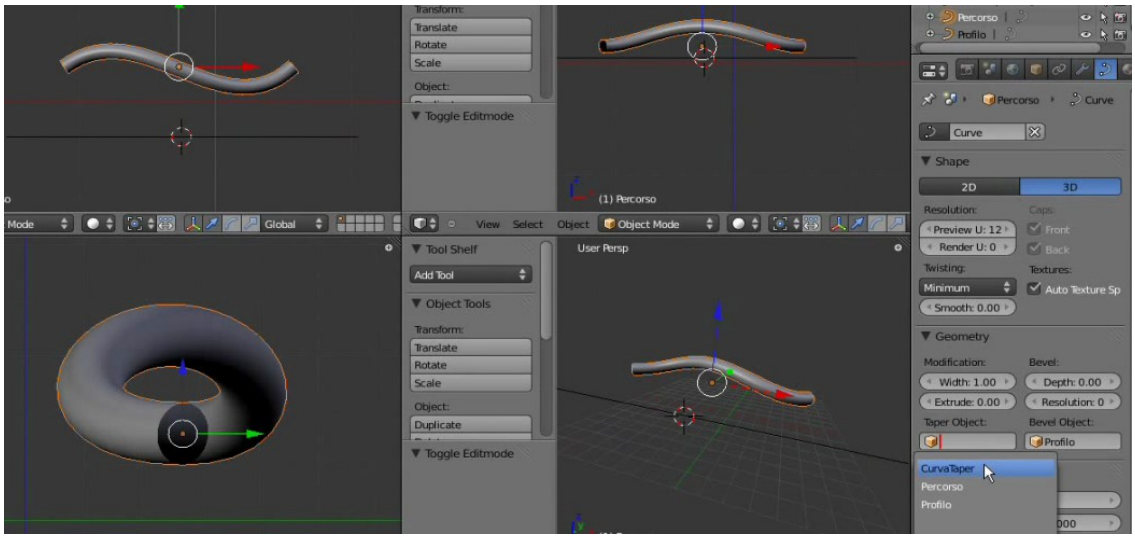
Aggiungiamo una nuova curva *Path* alla scena, rinominandola in "*CurvaTaper*".



La nuova curva PATH, chiamata "CurvaTaper"

Questa curva è una rappresentazione dell'estrusione ottenuta mediante *Bevel*; in pratica, indipendentemente dalla forma della curva estrusa e dalla lunghezza del *Path*, il primo vertice corrisponderà al punto iniziale della curva estrusa, l'ultimo corrisponderà al punto finale e i vertici intermedi alle parti intermedie, ovviamente.

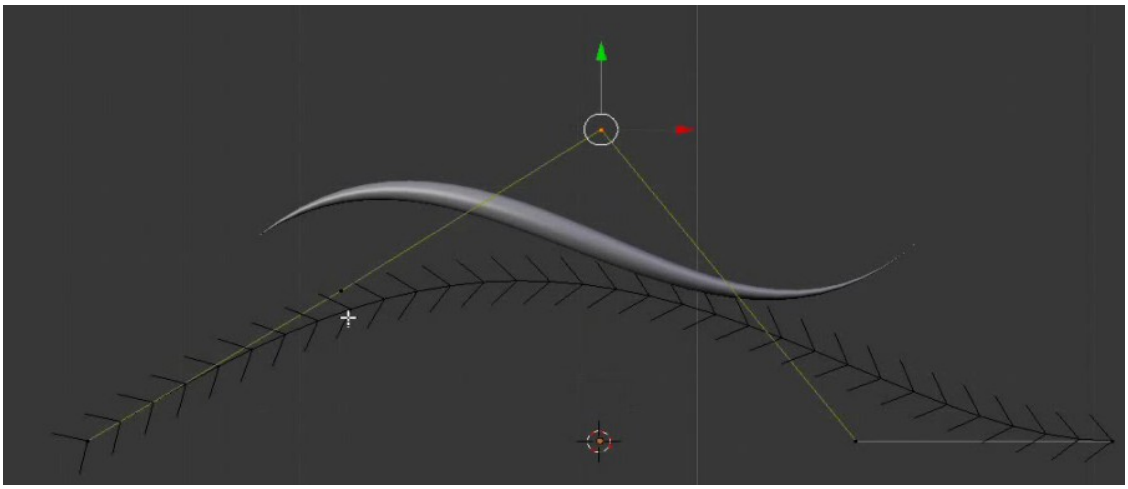
Per associare questo profilo alla curva estrusa, dobbiamo selezionare quest'ultima e scrivere, nel campo *Taper Object* all'interno di *Object Data* (nella *Properties Window*), il nome della curva di riferimento, nel nostro caso "*CurvaTaper*".



Taper Object, per la curva percorso

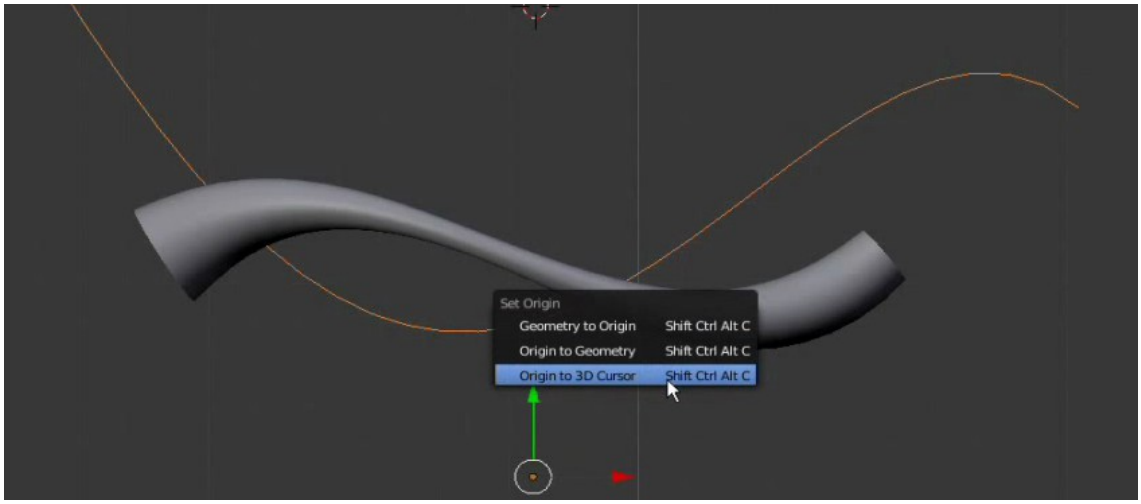
Di default, il *Pivot Point* (il "perno" delle trasformazioni) della curva *Taper* corrisponde al centro della stessa; l'ampiezza della curva estrusa è determinata, punto per punto, dalla distanza sull'asse Y locale tra i punti di controllo della curva e tale *Pivot Point*, per cui per aumentare lo spessore della curva, ad esempio, all'inizio, dobbiamo selezionare il primo punto di controllo della curva *Path* e spostarlo lungo l'asse Y *locale*, allontanandolo dal *Pivot Point*.

In questo modo, operando per i vari punti di controllo della curva *Taper* possiamo definire l'ampiezza della curva estrusa nei vari punti della stessa.



Utilizzo della Curva Taper

Possiamo anche spostare il *Pivot Point* in un altro punto della scena, ad esempio nel punto ove è presente il *3D Cursor*; per farlo, selezioniamo la curva *Taper*, passiamo in *Object Mode*, premiamo *CTRL SHIFT ALT C* e scegliamo *Origin To 3D Cursor.*



Origin to 3D Cursor

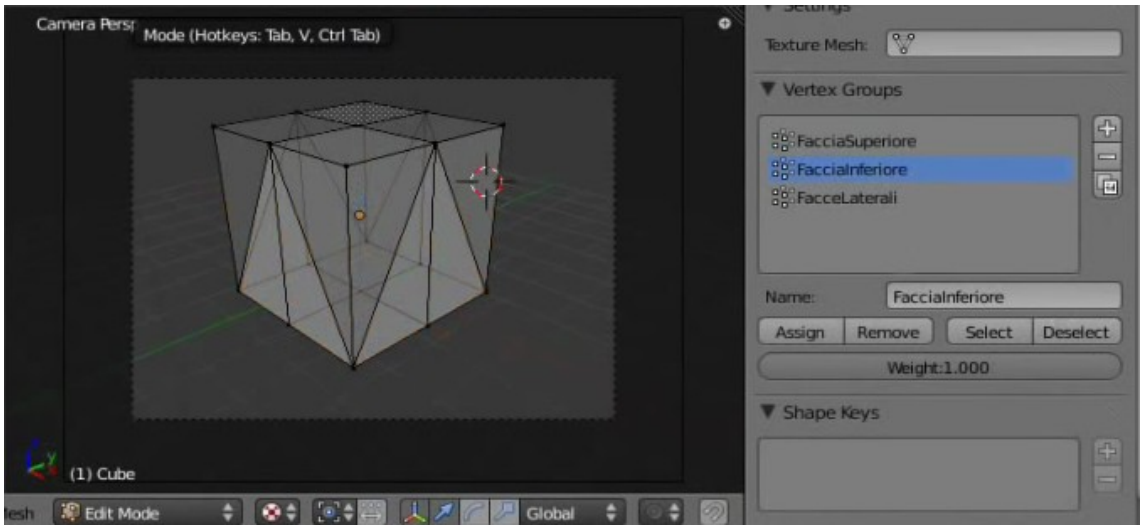
Per posizionare nuovamente il *Pivot Point* nel centro della curva scegliamo l'opzione *Origin To Geometry*. Ovviamente, definite le trasformazioni potremo trasformare il tutto in Mesh, se lo vogliamo, premendo *ALT C* (in Object Mode) e scegliendo *Convert to Mesh* dal menù che apparirà.

* * *

UV Mapping, le basi - Versione REDUX

Questa è la versione riveduta e aggiornata di un mio precedente tutorial dedicato alle basi della mappatura UV in Blender 2.5. Vedremo quindi come applicare una Texture immagine a certe porzioni di un oggetto, selezionando le facce da influenzare e il tipo di proiezione. Vedremo solo una procedura base, mentre per altre modalità e tecniche farò dei tutorial a parte in seguito.

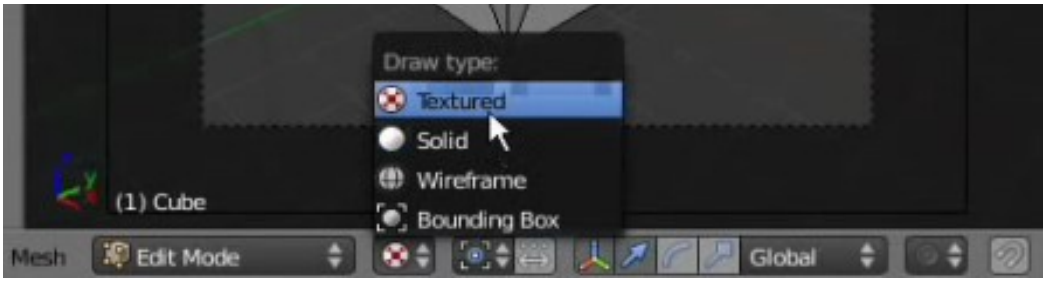
Nella scena visibile nell'immagine qui sotto è presente un cubo dotato di tre *Vertex Groups* per raggruppare i vertici della faccia superiore, della faccia inferiore e delle facce laterali.



La scena utilizzata in questo esempio, con il cubo dotato di tre Vertex Groups

In un tutorial pubblicato precedentemente, ho spiegato come creare i *Vertex Group* correttamente, per risolvere alcuni problemi che si presentano con oggetti come il cubo, appunto. L'utilizzo dei *Vertex Group* non è obbligatorio, ma consigliato, per richiamare facilmente certe facce.

Per vedere in tempo reale l'applicazione della proiezione cambiamo il *Draw Type* in *Textures* in una finestra di navigazione 3D. Passiamo quindi in Edit Mode e selezioniamo tutti i vertici del cubo.



Draw Type (ora Viewport Shading) "Textured"

Cambiamo una finestra in finestra *UV – Editor delle immagini*; qui, mediante la voce *Open* del menù *Image*, possiamo caricare un'immagine in Blender (anche più di una volendo, infatti tali immagini vengono memorizzate da Blender e per richiamarle possiamo cliccare sull'icona posta a sinistra del tasto *New* e sceglierne una tra quelle presenti nell'elenco).

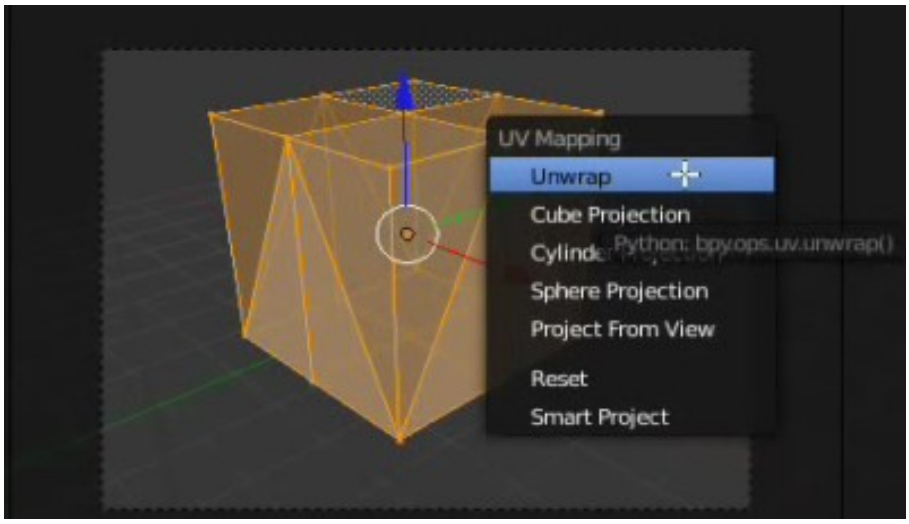
Open Image, in una finestra UV - Editor delle immagini

Nel mio caso ho caricato precedentemente un'immagine da file, comunque per fare questa operazione basta scegliere *Open* da *Image*: si aprirà una finestra stile *Esplora Risorse*, che ci consentirà di scegliere un'immagine presente su disco.

Quando si fa una mappatura *UV* per un oggetto, bisognerebbe creare un file immagine contenente le mappe da applicare alle varie parti dell'oggetto; l'immagine visibile nel mio caso nella finestra *UV* contiene, ad esempio, varie immagini, da applicare alle varie facce del cubo.

Questa è una tecnica utile anche per sfruttare tecniche più avanzate (non discusse in questo tutorial di base) sull'esportazione del *layout UV* per disegnare le mappe con programmi esterni, come *GIMP* o altri, ma ne parleremo un'altra volta.

Torniamo alla finestra 3D e, con tutte le facce della mesh selezionate, premiamo *U*, che sta per *Unwrap*, letteralmente "scuci", in quanto vogliamo appunto "scucire" le facce della mesh e farle corrispondere a certe parti dell'immagine.



U - Unwrap, in una finestra 3D, con tutte le facce della mesh selezionate

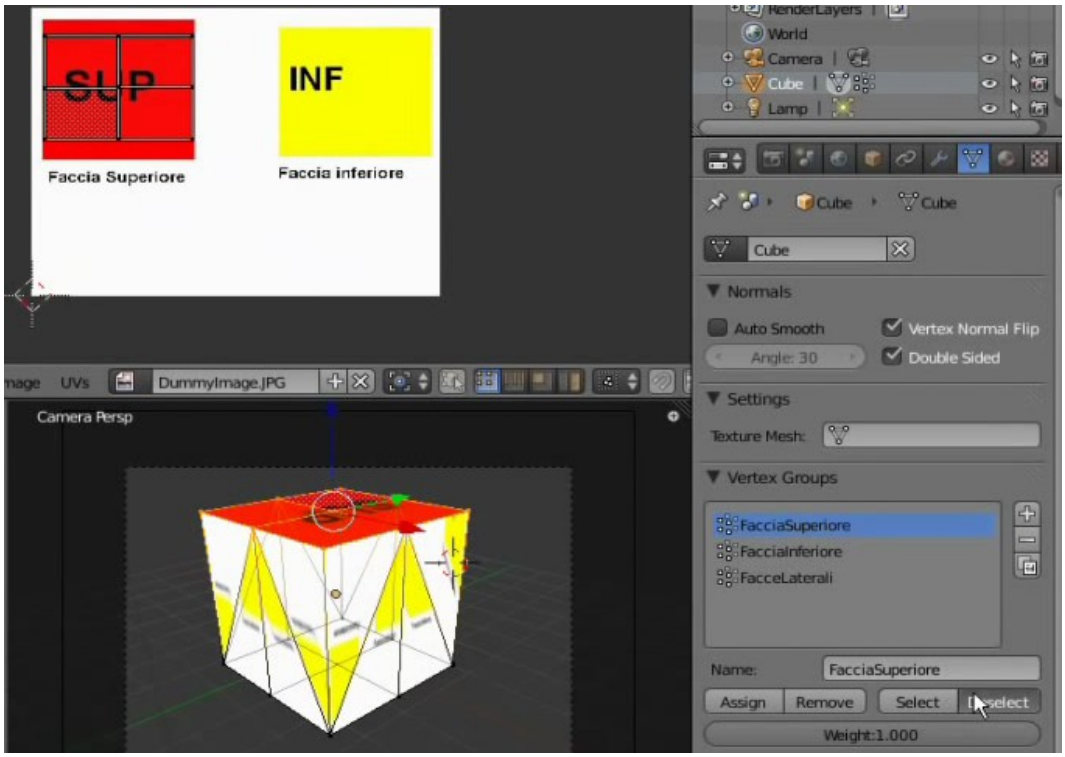
Abbiamo a disposizione diverse modalità per effettuare l'unwrapping, per scegliere in che modo proiettare i vertici selezionati nella finestra *UV - Editor delle immagini*; a seconda della forma della mesh, può essere utile, ad esempio, scegliere la proiezione sferica, cilindrica o cubica, oppure affidarci a *Unwrap* o a *Smart Project*.

Un tipo di proiezione utile quando abbiamo immagini prese non frontalmente, ma in prospettiva, è *Project From View*, che dispone i vertici nella finestra *UV* così come li vediamo nella finestra 3D ove abbiamo premuto *U*.

Bisognerà quindi associare le varie facce della mesh a certe parti dell'immagine presente nella finestra *Editor delle immagini*. Quello che bisogna fare è, quindi, selezionare i vertici nella mesh e premere *U* per effettuare l'unwrap e la mappatura.

L'utilizzo dei *Vertex Group* per raggruppare i vertici torna utile proprio in questo momento, in quanto possiamo selezionare velocemente i gruppi di vertici in Edit Mode per ritrovare i vertici corrispondenti nella finestra *UV* già selezionati e pronti per essere manipolati e posizionati correttamente.

Ovviamente, possiamo selezionare uno o più vertici presenti nella finestra *UV - Editor delle immagini* e traslare, scalare o ruotare la selezione, come se si trattasse di vertici di una mesh.



La mappatura di alcune facce "scucite" nella finestra UV - Editor delle immagini

Quando abbiamo finito di posizionare un insieme di vertici nella finestra *UV*, premiamo *A* per deselezionare tutto, e nel passare ad un altro *Vertex Group* ricordiamoci di cliccare prima su *DESELECT* per disattivare quello corrente.

In questo caso, non stiamo utilizzando l'immagine per realizzare superfici con immagini ripetute, come ad esempio un pavimento con mattonelle o una parete con piastrelle, per cui per non associare certe facce ad una Texture immagine possiamo procedere in due modi: o assegnando le relative facce ad una zona "libera" dell'immagine, ad esempio una zona di colore uniforme, oppure spostando, nella finestra *UV*, i vertici della selezione al di fuori dell'immagine.

Per le facce laterali del cubo, utilizziamo questa seconda opzione; in questo caso, dovremo modificare la mappatura dell'immagine da *Repeat*, che è la voce di default, a *Clip*, nella sezione *Image Mapping* del pannello Texture, come vedremo tra poco.

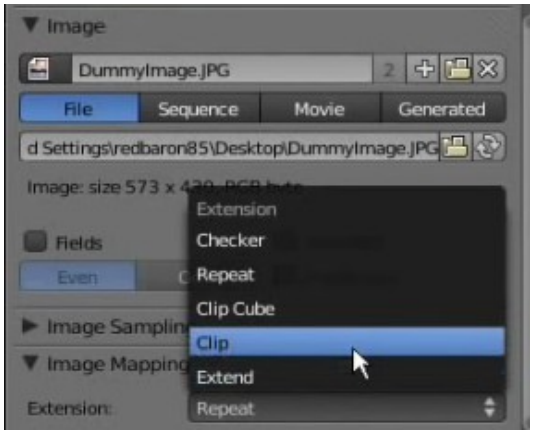
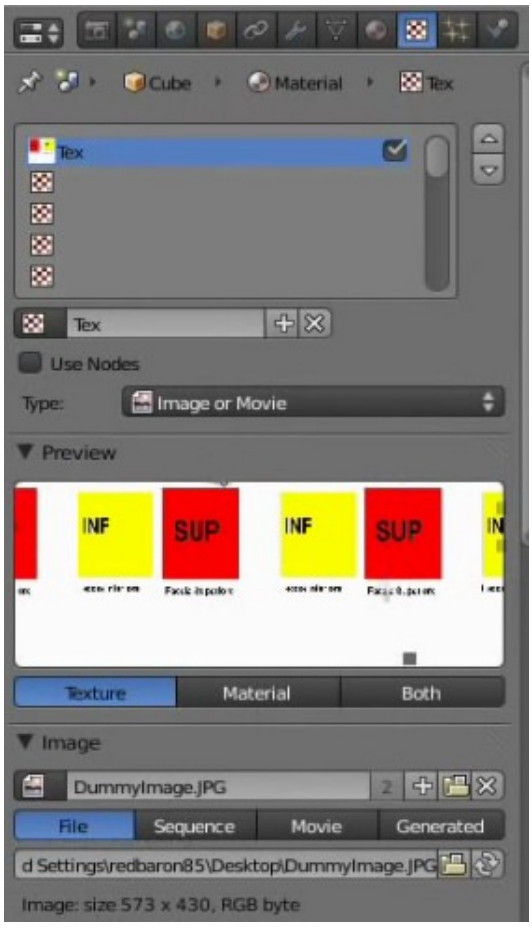


Image Mapping - Clip

Va detto comunque che la proiezione così realizzata NON verrà ancora renderizzata, infatti premendo *F12* per lanciare un rendering non vedremo l'immagine.

Per applicare la Texture *UV* in fase di rendering dobbiamo dotare la mesh di un Material e di una Texture di tipo immagine, per cui creiamone una per questo oggetto; potremo comunque caricare anche altre Textures, per esempio per influenzare altri canali di mappatura: non siamo vincolati solo a questa Textures e ad un solo canale di applicazione.

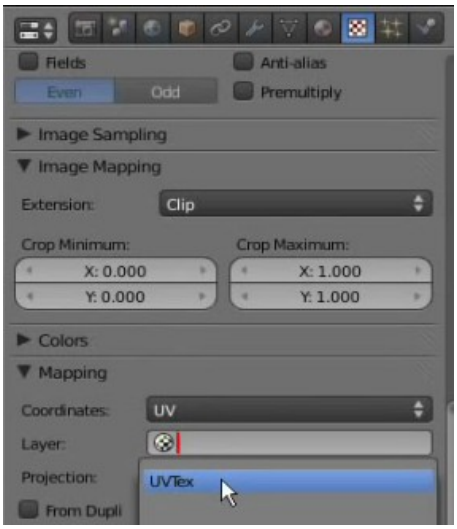


Definizione di una Texture immagine per l'oggetto

La Texture dev'essere, come detto, di tipo *Image*; non c'è bisogno di caricare l'immagine con *Open*, in quanto è già presente nell'elenco delle immagini caricate in Blender, visto che l'abbiamo aperta col menù *UV - Editor delle immagini*, per cui basterà scegliere tale voce dal menù *Browse ID Data*, che si apre cliccando sull'icona posta a sinistra del tasto *New*.

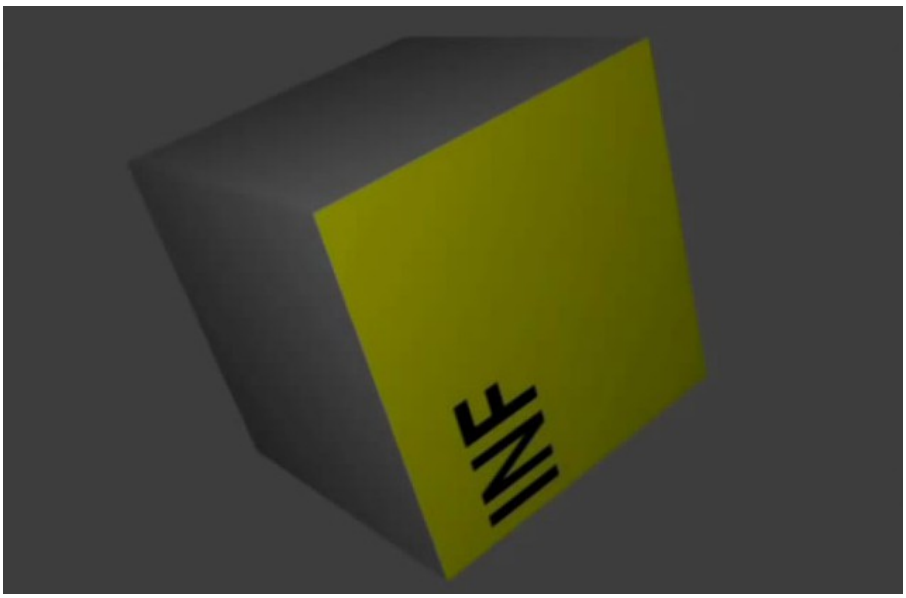
Come anticipato, in questo caso vogliamo che i vertici posti (nella finestra UV) al di fuori dell'immagine non presentino Texture immagine; per fare ciò, nella scheda *Image Mapping* cambiamo la voce *Extension - Repeat* in *Extension - Clip*.

In questo modo l'immagine visibile nella scheda *UV - Editor delle immagini* non verrà replicata anche nelle zone non visibili, per cui i vertici posti in tali zone (come quelli delle facce laterali, nel nostro caso) non avranno Textures immagini. Questa opzione distrugge la possibilità di realizzare con facilità un pavimento o una parete con piastrelle, ma parlerò di come realizzare questo effetto in un tutorial a parte. Torniamo comunque alle voci della scheda Textures: in *Mapping* scegliamo, come voce per *Coordinates*, l'opzione *UV*, e scriviamo il nome del canale *UV* creato automaticamente da Blender; anche in questo caso, facendo click sul campo *Layer*, Blender ci verrà in aiuto suggerendoci i nomi dei campi *UV* creati per l'oggetto (in questo caso uno solo, creato automaticamente). È possibile aggiungere altri livelli *UV* all'oggetto mediante gli strumenti della sezione *UV Textures* nella scheda *Object Data*, all'interno della *Properties Window*.



Mapping Coordinates - UV, nella scheda della Texture

Adesso possiamo lanciare un rendering ed osservare l'effetto ottenuto.



Il rendering del risultato ottenuto

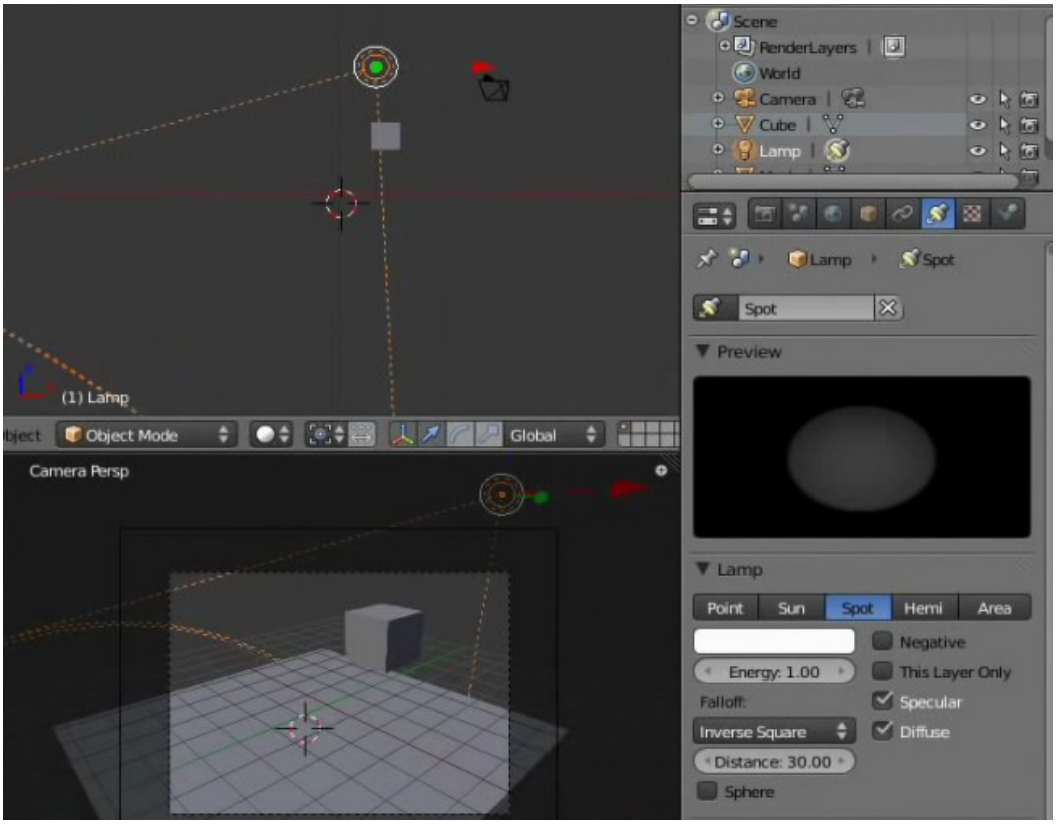
Come anticipato, esistono anche altri strumenti ed altre operazioni più avanzate legate alla mappatura *UV*, ma già con queste basi è possibile ottenere i risultati desiderati.

* * *

Halo Light - Luci ed ombre volumetriche

In questo brevissimo tutorial vedremo come realizzare l'effetto "*luce volumetrica*" in Blender 2.5.
La luce volumetrica è quell'effetto particolare che si ha quando in una scena è presente, ad esempio, del pulviscolo; si tratta di un effetto buono per creare anche atmosfere particolari, tipo "illuminazione divina", infatti i fasci di luce visibili con l'illuminazione volumetrica sono noti come "*God Rays*" (raggi di Dio).

L'effetto è disponibile solo per le fonti di luce di tipo *Spot*, per cui inseriamo nella nostra scena una *Lamp Spot* (o cambiamo una fonte di luce già presente in *Spot*) ed apriamo, all'interno della *Properties Window*, la scheda *Object Data*.



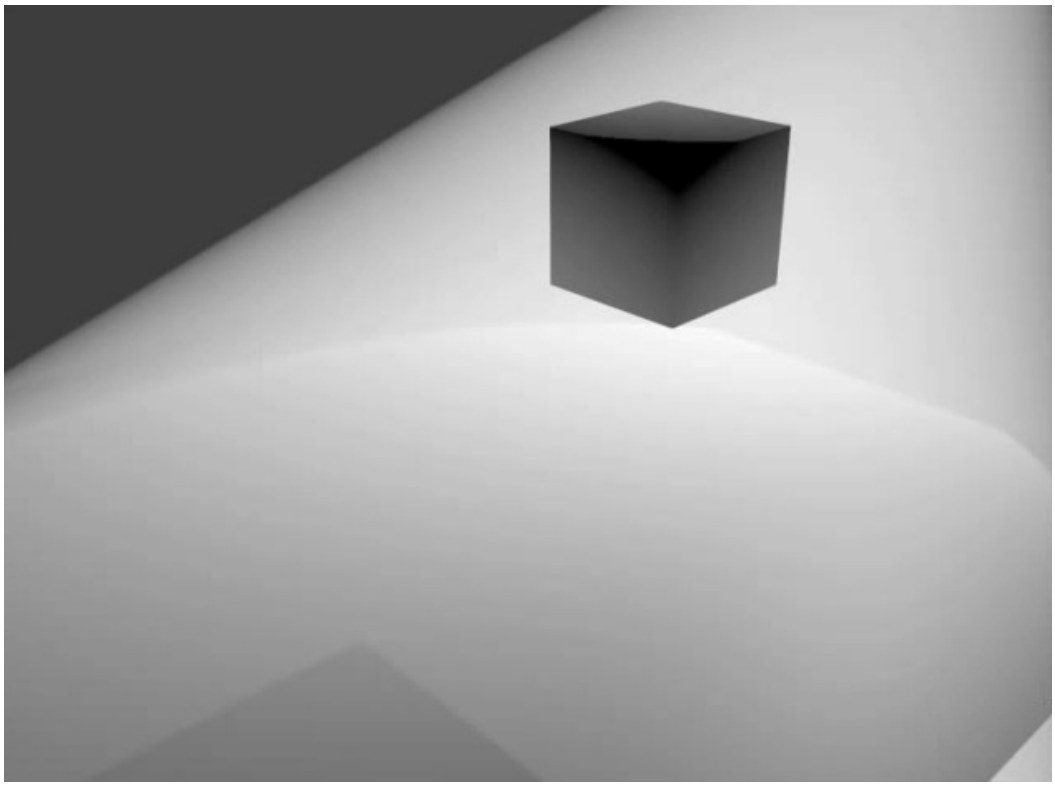
Utilizzare una fonte di luce Spot

Attivare l'effetto luce volumetrica ("*HALO*") è di una facilità disarmante: basta selezionare la voce *Halo* nella sezione *Spot Shape* della scheda, dove è possibile anche impostare le dimensioni del cono di luce, il *Blend Type* (cioè quanto dovranno essere sfocati i bordi del cono di luce, in una scala da 0.0 a 1.0) e cambiare la forma del cono di luce da cono a piramide selezionando "*Square*".

Le opzioni Halo, nella scheda Object Data della fonte di luce

Attivando *Halo* verrà messo a disposizione lo slider "*Intensity*", per regolare l'intensità dell'effetto. Si tratta di un parametro diverso da *Energy* della sezione *Lamp*: *Energy* consente di impostare la quantità di luce emessa in generale dalla fonte luminosa, mentre *Halo Intensity* riguarda l'intensità dell'effetto volumetrico.

Già a questo punto possiamo lanciare un rendering di prova (realizzato, nel mio caso, con *Energy* 0.75 e *Halo Intensity* 3.0), per valutare l'effetto.



Rendering dell'effetto Halo

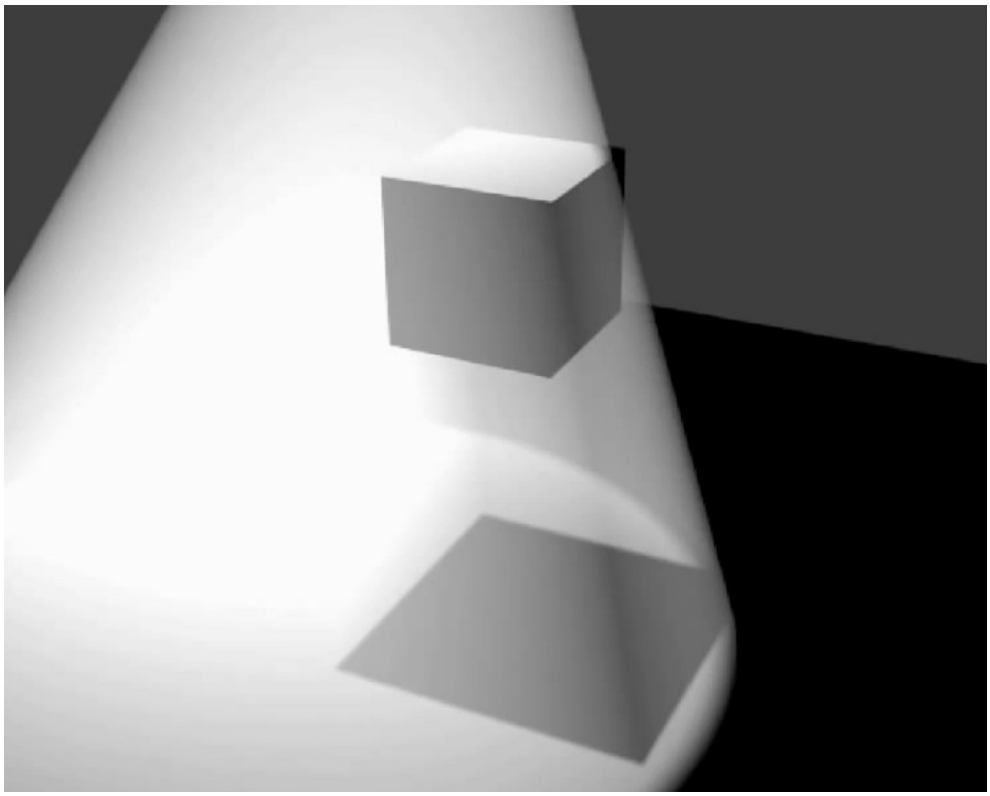
Notiamo subito che non sono presenti ombre volumetriche, che di default non vengono realizzate.

Ombre Buffer Shadow per attivare le ombre volumetriche

Per attivare questo effetto dobbiamo cambiare il tipo di ombre proiettate (di default, *Ray Shadow*) in *Buffer Shadow*, cliccando su tale pulsante nella sezione *Shadow* della scheda *Object Data* dell'oggetto fonte di luce.

A questo punto apparirà, sotto *Halo Intensity*, il pulsante *Step*, il cui valore è direttamente proporzionale alla qualità dell'effetto delle ombre volumetriche.

Possiamo fare delle prove con dei valori in una scala da 0 (effetto nullo) a 12 (massima qualità possibile), ma in generale valori come 8 e 10 dovrebbero andare più che bene; ovviamente, a valori maggiori corrisponderanno anche tempi di rendering più elevati.

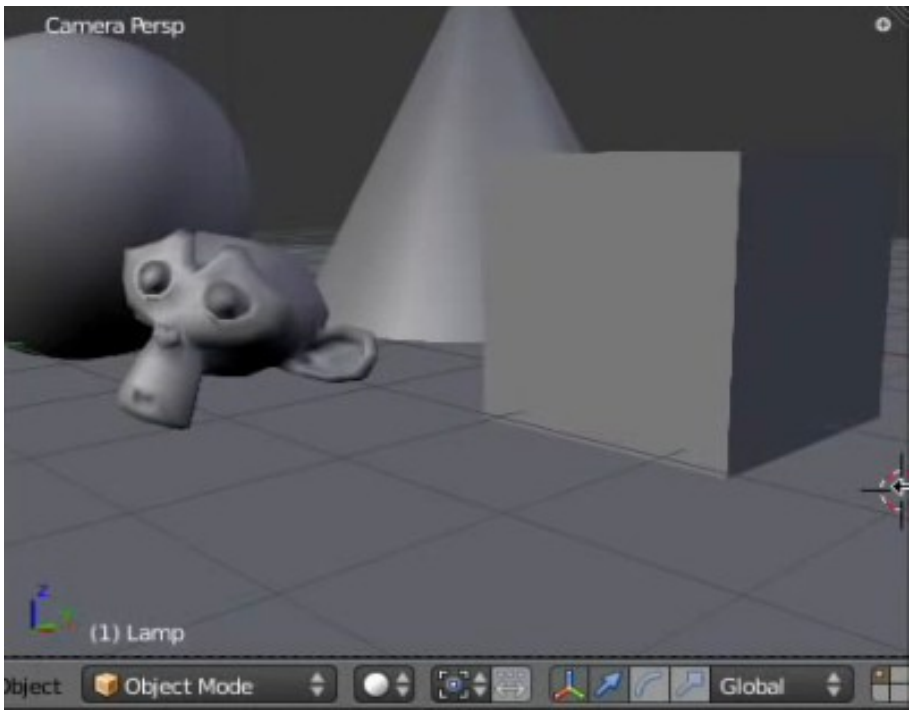

Un rendering effettuato con le ombre volumetriche

* * *

Ombre sfumate con Ray e Buffer Shadow

In questo tutorial vedremo come realizzare le ombre sfumate in Blender 3D 2.5.
Con le impostazioni di default, infatti, le fonti di luce disponibili in Blender proiettano ombre dai bordi netti; ciò avviene principalmente per accelerare i tempi di rendering e può andare bene in fase di preview dei risultati, ma l'effetto è decisamente poco realistico e conviene sfumare le ombre per i rendering finali.

Nella scena visibile nell'immagine sotto sono presenti alcuni oggetti posti sopra un Plane ma ad una certa distanza dallo stesso (non poggiati); più in alto, vi è una fonte di luce di tipo *Point*.



La scena utilizzata per questo esempio

Le fonti di luce di tipo *Point*, *Sun*, *Spot* e *Area* proiettano ombre di tipo *Ray Shadow*; le fonti di luce di tipo *Spot* possono proiettare anche ombre di tipo *Buffer Shadow.*

Fanno eccezione le fonti di tipo *Hemi*, che NON proiettano ombre e, in genere, vista la luce "velata" da loro prodotta, vengono utilizzate per la luce ambientale o scene all'aperto con cielo nuvoloso, dove non vi è illuminazione diretta.

La sezione (Ray) Shadow nella scheda Object Data di una fonte di luce

Iniziamo con le impostazioni delle ombre di tipo *Ray Shadow*. Con la nostra fonte di luce selezionata, apriamo la scheda *Object Data* all'interno della *Properties Window*; in tale scheda, in basso, è ben visibile la sezione *Shadow*, ossia "ombre", appunto.

Le ombre proiettate possono avere un colore diverso da quello di default, che è il nero. Possiamo cambiare il colore mediante l'apposito selettore.

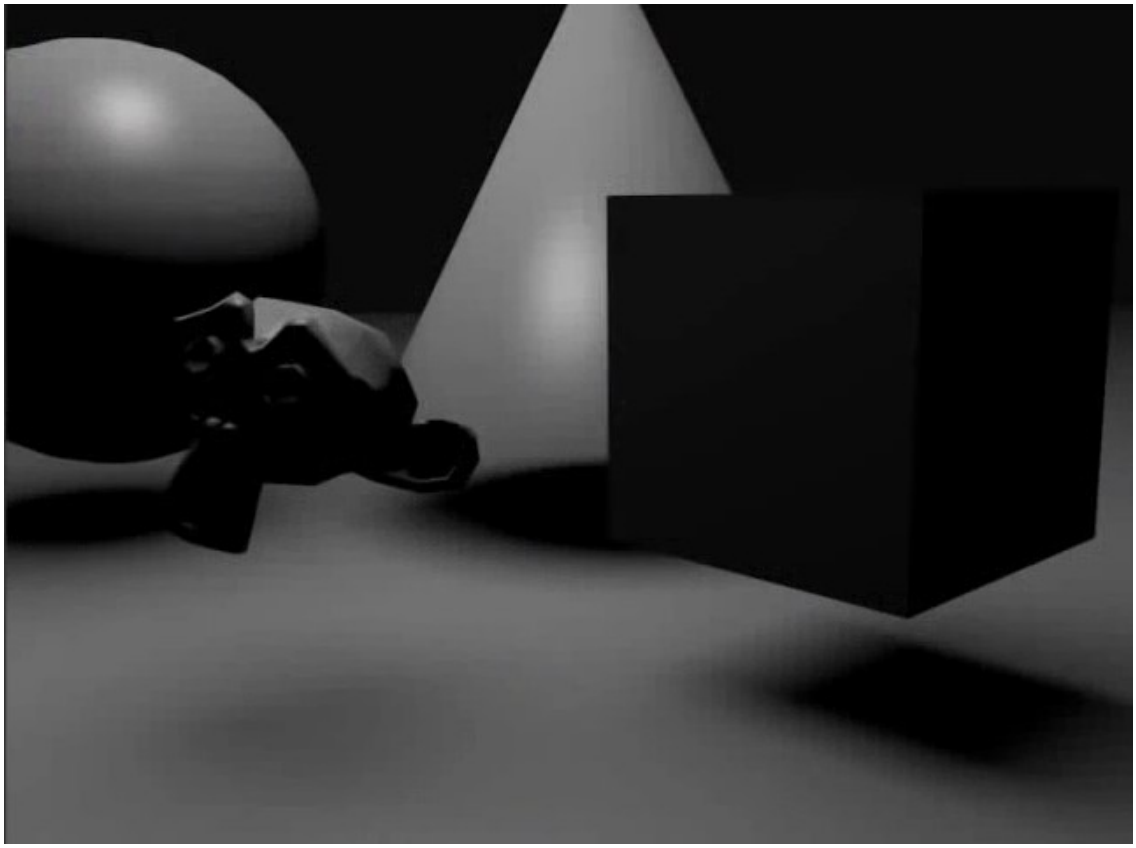
L'opzione "*This Layer Only*", se selezionata, limita la proiezione delle ombre solo agli oggetti presenti nello stesso *Layer* della fonte di luce, mentre l'opzione *Only Shadow* limita l'effetto della fonte luminosa alla sola proiezione delle ombre, senza illuminare gli oggetti.

A questo punto possiamo scegliere tra due diverse modalità di campionamento (e quindi di effetto finale) delle ombre *Ray Shadow*: costante e adattiva.

Il metodo costante produce risultati migliori di quello adattivo, ma con tempi di rendering maggiori.

Per entrambe le modalità sono presenti i parametri *Soft Size*, cioè l'ampiezza del bordo sfumato dell'ombra, e *Samples*, cioè numero di campioni; ad un valore maggiore di *Samples* corrisponderanno risultati migliori, ma con tempi di rendering più elevati. Per la modalità adattiva è disponibile anche il parametro *Threshold*, cioè un valore di soglia per regolare la qualità dell'effetto.

Il rendering nella pagina seguente è stato effettuato con metodo costante, *Soft Size* 8 e 10 *Samples*.



Il rendering con metodo costante, Soft Size 8 e 10 Samples

Come anticipato, la modalità *Ray Shadow* non è l'unica disponibile.

Cambiamo la fonte di luce presente nella scena in una fonte di tipo *Spot Light* (in pratica, un faretto) e selezioniamo la modalità *Buffer Shadow* all'interno della sezione *Shadow* per la scheda *Object Data* dell'oggetto, nella *Properties Window.*

La scheda Buffer Shadow per le ombre della fonte di luce (che ora è di tipo Spot Light)

Abbiamo a disposizione un numero maggiore di parametri rispetto a quelli disponibili per *Ray Shadow.*

Buffer Type ci consente di scegliere il tipo di algoritmo da implementare.

Le possibilità sono *Classical*, *Classic-Halfway*, *Irregular* e *Deep*; in genere, *Classical* andrà più che bene, mentre per risultati migliori (e in presenza di trasparenze) si consiglia di utilizzare *Deep*.

Filter Type ci consente di scegliere il tipo di filtro da utilizzare tra *Box*, *Tent* e *Gauss*; anche qui, l'opzione di default (*Box*) in genere andrà benissimo.

Come nel caso di *Ray Shadow*, il parametro *Soft* serve ad impostare la morbidezza e, quindi, le dimensioni della sfumatura delle ombre; con *Bias*, possiamo aggiungere un po' di distorsione o irregolarità per i bordi delle ombre, per ottenere risultati più interessanti.

Col selettore *Sample Buffer* possiamo decidere quanti *buffer* utilizzare per l'effetto; si consiglia, questa volta, di scegliere 4 o 9.

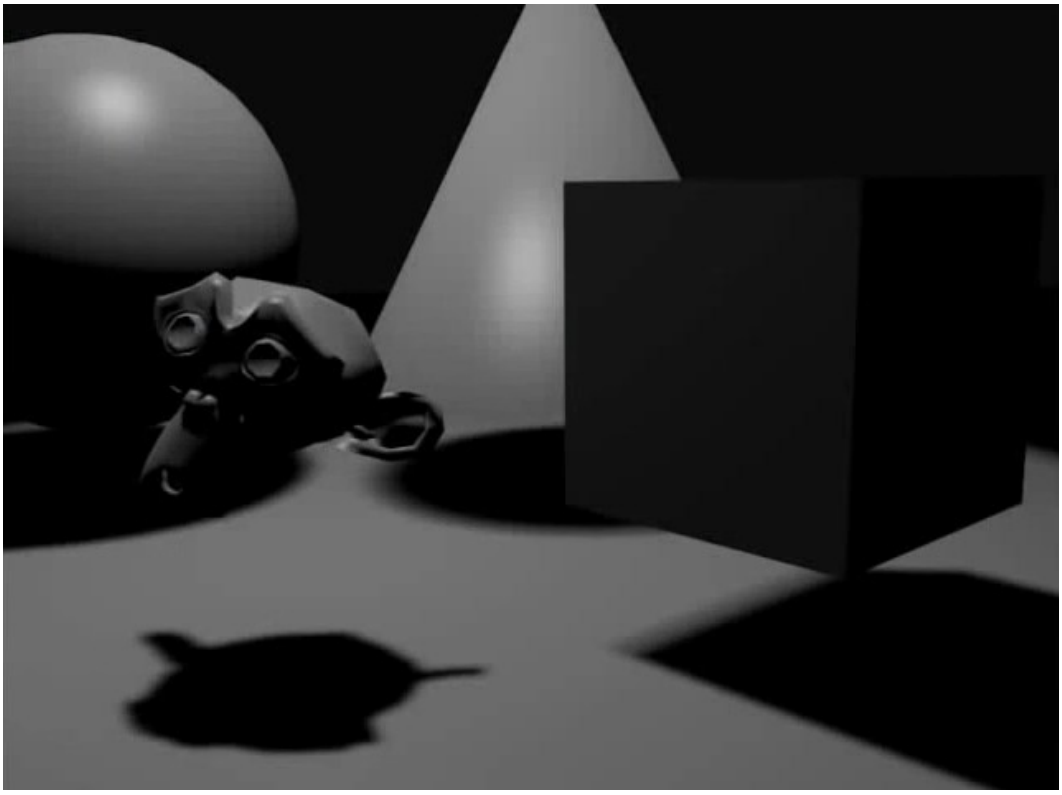
Il parametro *Size* serve a specificare la dimensione del *buffer* delle ombre, con valori di risoluzione che possono andare da 512 a 10240; si consiglia di impostare un valore medio-alto per ottenere buoni risultati, ma ciò porterà inevitabilmente a tempi di rendering più elevati.

Il parametro *Samples* serve per impostare il numero di campioni da utilizzare.

In fondo alla scheda abbiamo i pulsanti relativi al *clipping*.
Con le ombre di tipo *Buffer Shadow*, è possibile limitare gli effetti, ossia la proiezione delle ombre, agli oggetti presenti in certe zone della scena, specificando un punto di inizio ed uno di fine applicazione dell'effetto con *Clip Start* e *Clip End*; con *Autoclip Start* e *Autoclip End* possiamo lasciare a Blender il compito di definire tali punti.

Nell'immagine riportata nella pagina seguente è visibile un rendering effettuato con *Buffer Classical*, filtro di tipo *Gauss*, 9 *Sample Buffers*, *Soft* 8, *Bias* 5, *Size* 7168 e 5 *Samples*.

Va detto infine che la proiezione delle ombre può essere disattivata a livello globale in fase di rendering; ciò può essere utile per realizzare le ombre solo per il rendering finale, non nelle anteprime. Per attivare o disattivare le ombre a livello globale, apriamo la scheda *Render* nella *Properties Window* e, all'interno della sezione *Shading*, selezioniamo o deselezioniamo *Shadows*, a seconda dei casi.



Il rendering ottenuto con ombre Buffer Shadow per una fonte di luce Spot

Disattivando *Shadow* qui, disattiveremo le ombre per qualsiasi fonte di luce della scena; faccio notare però che attivare *Shadows* qui non significa impostare le ombre per tutte le fonti di luce, perché alcune di queste potrebbero avere l'opzione *No Shadow* selezionata nelle loro schede *Object Data*: questo è solo un controllo globale che abilita o disabilita una funzionalità, poi bisogna agire a livello locale (cioè per ogni singola luce) per impostare la modalità di proiezione delle ombre o disattivarla.

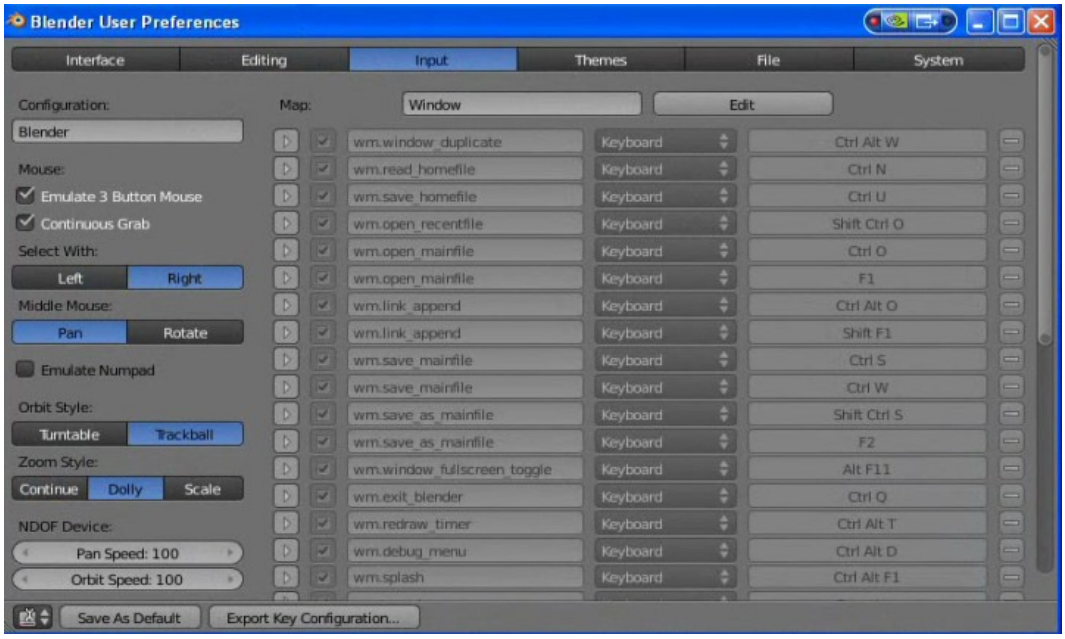
Disattivare le ombre a livello globale, nella scheda Render

* * *

Cambiare lo shortcut add mesh in barra spazio

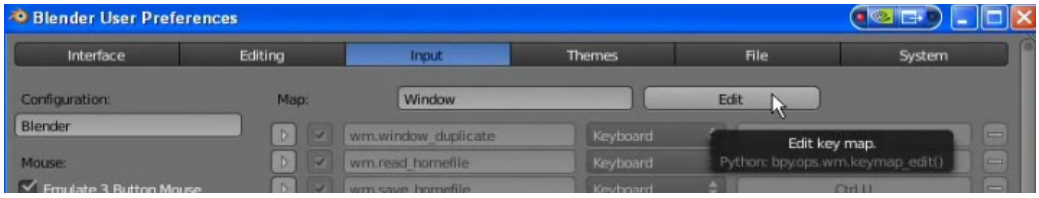
Fino alla versione 2.49 di Blender, bastava premere la barra spazio mentre il focus del mouse era in una finestra di navigazione *3D View* per aggiungere una nuova mesh alla scena; con la versione 2.5 di Blender, la barra spazio serve per aprire la finestra di ricerca del programma, mentre per richiamare la finestra *Add Mesh* bisogna premere *SHIFT A*, ma ovviamente possiamo cambiare questo (e altri) shortcut: vediamo come.

Apriamo il menù *File* e scegliamo la voce *User Preferences*, cioè preferenze, impostazioni personalizzate dell'utente. Nella finestra che apparirà a video, scegliamo la voce *Input*.



La sezione Input della finestra User Preferences, richiamabile dal menù File

Proprio sotto *Input* apparirà un campo testuale di etichetta *Map*, con *Window* come voce inserita di default. I comandi per modificare gli shortcut per tale ambiente sono bloccati, per cui clicchiamo su *Edit* per sbloccarli.



Sbloccare i comandi per modificare gli shortcut

Cerchiamo quindi la voce che ha, come tasto di scelta rapida, la barra spazio, ossia *wm.search_menu* (letteralmente, "Menù di ricerca") e disattiviamola deselezionando la relativa casella, oppure cambiamo lo shortcut con un altro tasto di scelta rapida a nostro piacere: basta cliccare sul pulsante con scritto *Spacebar* e premere un tasto o una combinazione da noi scelta.

Modificare lo shortcut per la wm.search_menu

Adesso, sempre nella scheda Input della finestra *User Preferences*, cancelliamo *Window* nel campo *Map* e scriviamo *Object Mode*. Anche qui, clicchiamo su *Edit* per abilitare la personalizzazione dei tasti di scelta rapida.

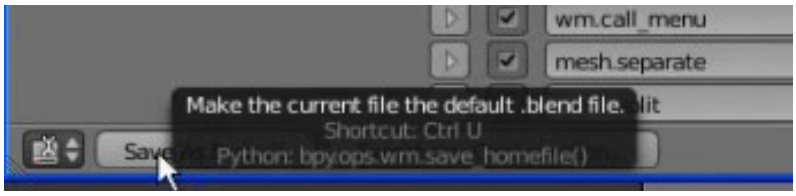
Cerchiamo la voce *wm.call_menu*, che di default ha come shortcut *SHIFT A*, quindi clicchiamo sul pulsante con scritto, appunto, *SHIFT A* e, alla richiesta *Press a key*, premiamo la barra spazio.

Modificare la voce wm.call_menu, impostando la barra spazio

La finestra *Add Mesh* era disponibile anche in Edit Mode per aggiungere nuove forme all'interno di un'unica mesh; anche in questo caso, lo shortcut è stato cambiato in *SHIFT A*.

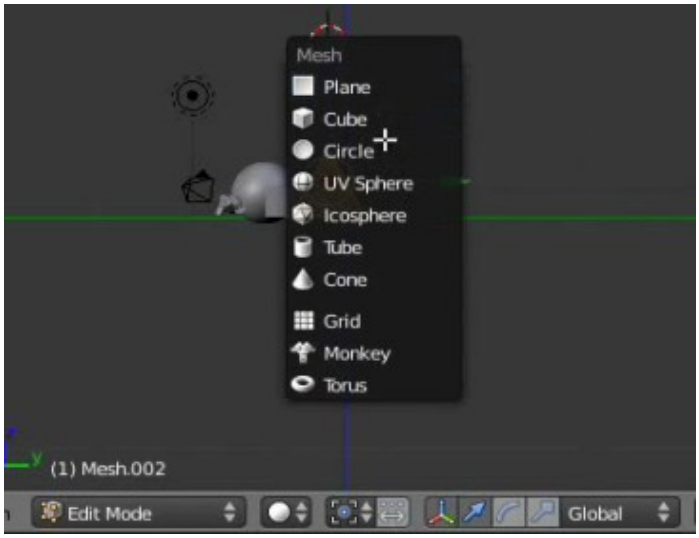
Per impostare anche qui la barra spazio, dobbiamo cambiare il contesto *Map* in *Edit Mesh*, individuare nella finestra la voce *wm.call_menu* e cambiare lo shortcut con la barra spazio, come visto un attimo fa nel contesto Object Mode.

Per salvare queste impostazioni e ritrovarle in futuro, ad ogni successivo avvio di Blender, così come le stiamo lasciando ora, clicchiamo sul pulsante *Save as default*, in basso a sinistra nella finestra *User Preferences*.



Cliccare su "Save As Default" per ritrovare queste impostazioni agli avvii successivi di Blender

Adesso, premendo la barra spazio in una finestra di navigazione 3D, apparirà nuovamente la tanto amata finestra *Add Mesh*, sia in Object che in Edit Mode.



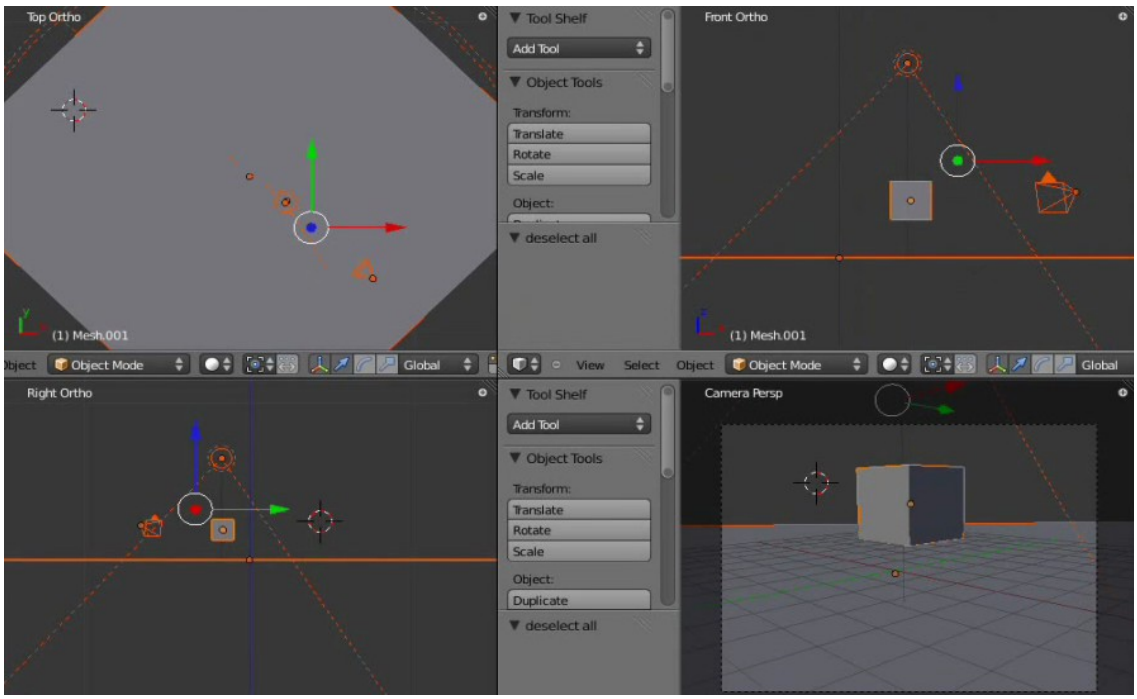
La finestra Add Mesh dopo aver premuto, questa volta, la barra spazio

* * *

Ombre in tempo reale nei giochi fatti col Blender GE (Game Engine)

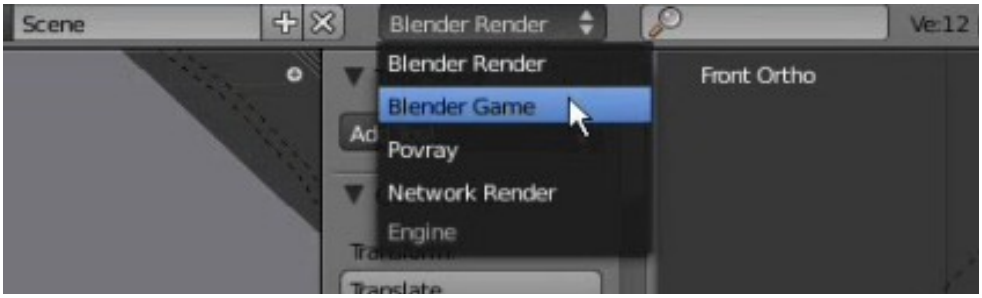
In questo breve tutorial vedremo come inserire le ombre in tempo reale in programmi e giochi fatti col *Game Engine*, il motore di gioco interno di Blender 2.5.
Le ombre in tempo reale possono essere realizzate utilizzando i materials *GLSL* e le fonti di luce di tipo *Spot Light* con ombre *Buffer Shadow.*

Nella scena visibile nell'immagine qui sotto è presente un cubo posto sopra un Plane (non proprio poggiato, ma ad una certa distanza); ancora più in alto vi è una fonte di luce, di tipo *Spot Light* (in pratica il faretto) orientata verso il basso, in modo da illuminare il cubo e il Plane.



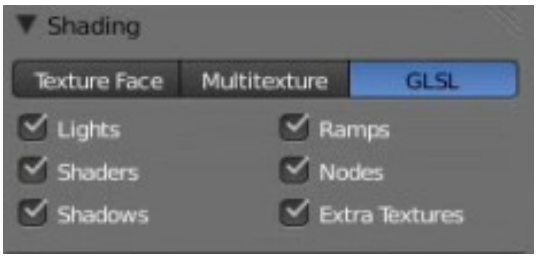
La scena utilizzata in questo tutorial

Per prima cosa, cambiamo il motore da utilizzare da *Blender Render* a *Blender Game* attraverso il selettore *Engine*, situato nella finestra *Info*.



Passaggio al Game Engine, dall'apposito selettore nella scheda Info

Adesso apriamo la scheda *Render* all'interno della *Properties Window* e selezioniamo, in *Shading*, la voce *GLSL*.



Impostare la modalità di ombreggiatura GLSL nella scheda Render

Come anticipato, per poter proiettare le ombre, le fonti di luce devono essere di tipo *Spot Light* con ombre *Buffer Shadow*, per cui selezioniamo il faretto ed apriamo la scheda *Object Data* all'interno della *Properties Window*; qui, nella sezione *Shadow*, scegliamo *Buffer Shadow.*

La sezione Buffer Shadow nella scheda Object Data della fonte di luce Spot Light

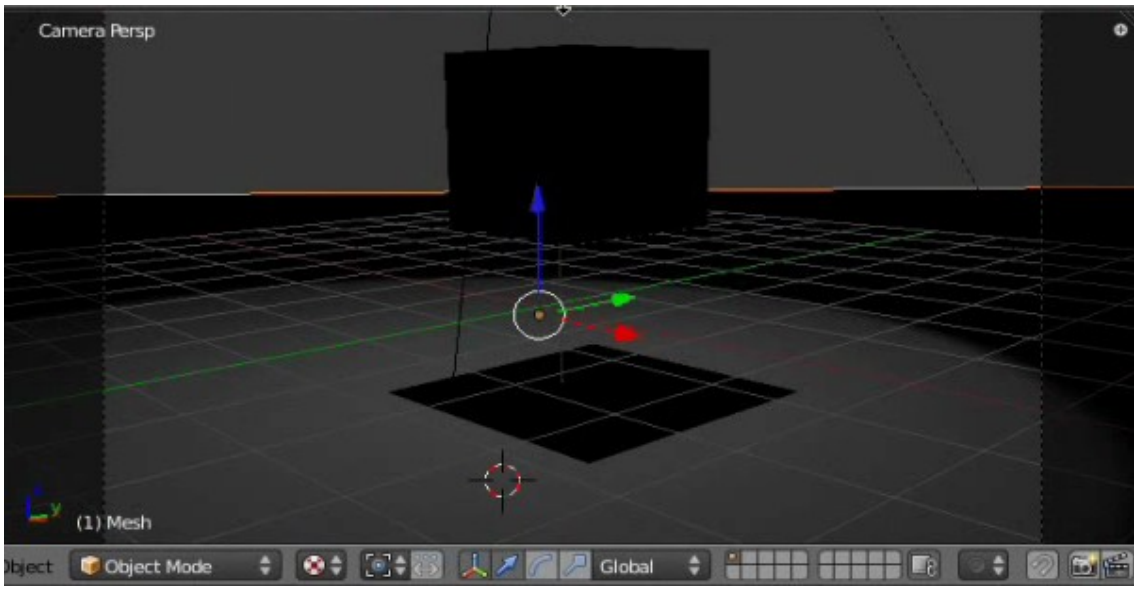
Il parametro che ci interessa, all'interno di *Buffer Shadow*, è *Size*, che serve a specificare la dimensione del buffer delle ombre; a seconda del valore di risoluzione si avranno risultati migliori, ma con un maggior utilizzo delle risorse di sistema. A parte questo parametro possiamo impostare, per il faretto, la dimensione del cono di luce, mediante la voce *Spot Size* nella sezione *Spot Shape* della scheda *Object Data*, nella *Properties Window.*

Conviene comunque non abusare delle ombre *Buffer Shadow* all'interno di un gioco, o almeno utilizzare valori modesti per le impostazioni, per evitare di avere rallentamenti durante l'esecuzione.

Non abbiamo ancora finito.

Cambiamo la modalità di visualizzazione di una finestra *3D View* in *Textured* per vedere quali oggetti sono muniti di Materials, in quanto (per poter implementare le ombre in tempo reale nel *Game Engine*) gli oggetti vanno muniti di Materials; possiamo munirli anche di Textures, se lo desideriamo, ma i Materials ci devono essere per forza.

Adesso possiamo provare l'effetto finale premendo P per avviare il gioco.



Screenshot del risultato ottenuto

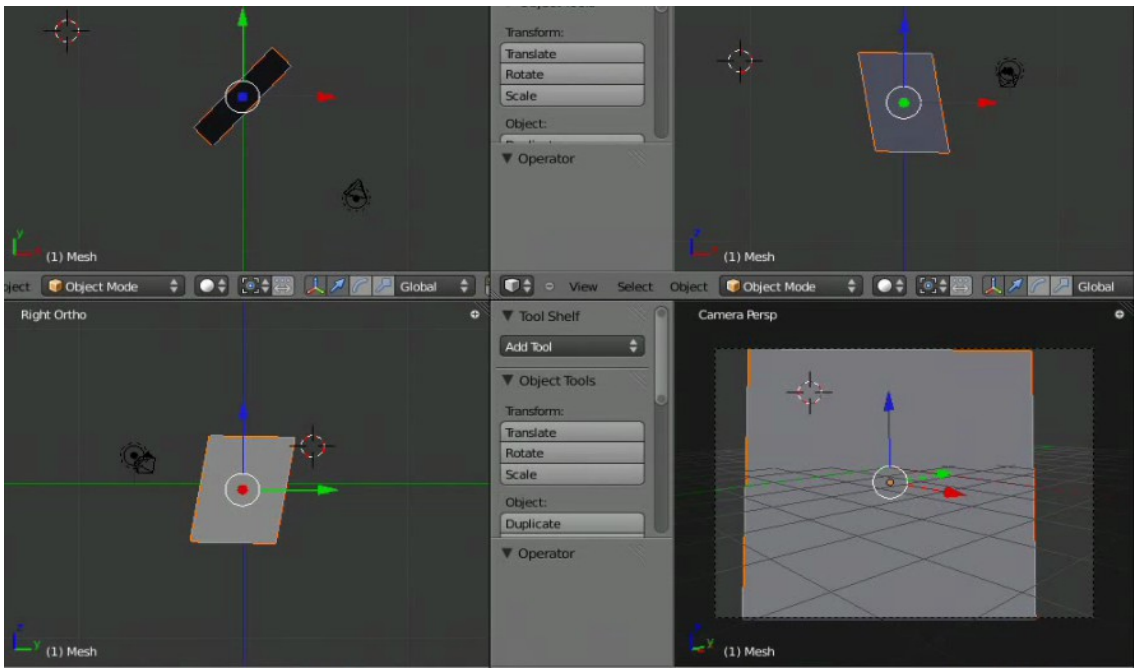
* * *

Utilizzare le Texture Stencil come maschere coprenti

In questo breve tutorial parleremo delle Textures utilizzate in modalità *Stencil* in Blender 2.5.

Le Textures *Stencil* vengono utilizzate come maschere per coprire certe zone delle Textures che stanno sotto di loro nella pila di Textures di un Material, un po' come avviene con certi *Layers* coprenti (o *maschere*, appunto) in programmi di fotoritocco e grafica 2D.

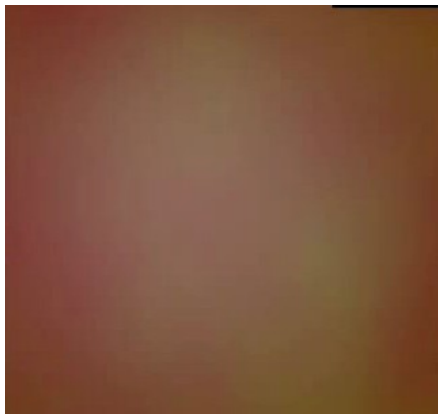
Nella scena visibile nell'immagine qui sotto è presente un Plane posto proprio di fronte a noi.
Tale Plane è dotato di un Material di base.



La scena utilizzata per questo esempio, con il Plane posto di fronte a noi

Aggiungiamo alla pila di Textures del Material, in cima, una Textures di qualsiasi tipo, ad esempio una Textures di tipo *Clouds*, magari colorata (attivando la *Color Ramp*, quindi).

Avviando il rendering, vedremo il Plane texturizzato correttamente con tale Texture.



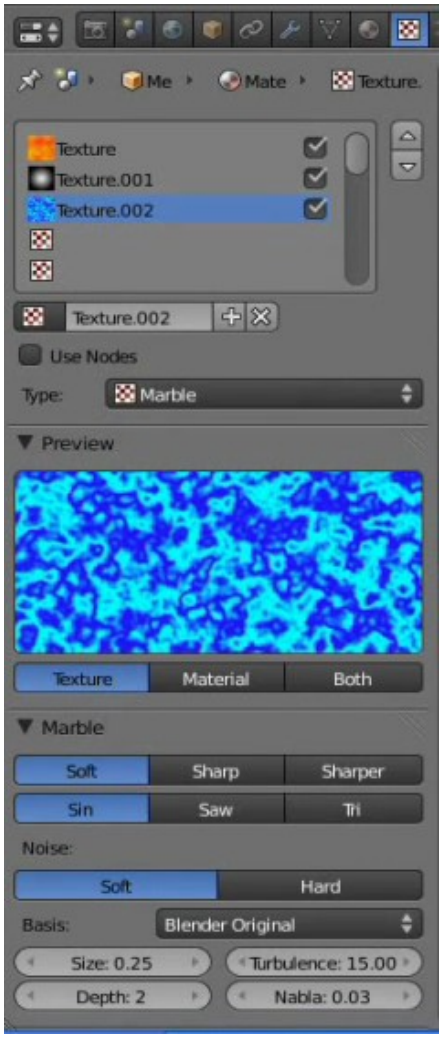
Una porzione del rendering del Plane texturizzato (e colorato) con una Texture Cloud

Aggiungiamo ora una seconda Textures alla pila, scegliendo questa volta (sempre per esempio) una Texture *Blend* di tipo *Spherical*. Nella sezione *Influence* di tale Texture disattiviamo *Color Diffuse* e selezioniamo, in fondo alla scheda, la voce *Stencil*.

A questo punto, la Texture è diventata una Texture maschera, coprente: in pratica, vengono utilizzati i valori della scala di grigi per coprire le Texture sottostanti, ed in particolare le zone nere disattiveranno l'effetto delle relative aree delle Textures sottostanti, per cui (in questo caso, con una Texture *Blend Spherical*) vedremo solo la parte centrale delle Textures che seguiranno, mentre le parti ai bordi, corrispondenti alle zone nere della Texture *Blend Spherical*, non verranno applicate al Material dell'oggetto.

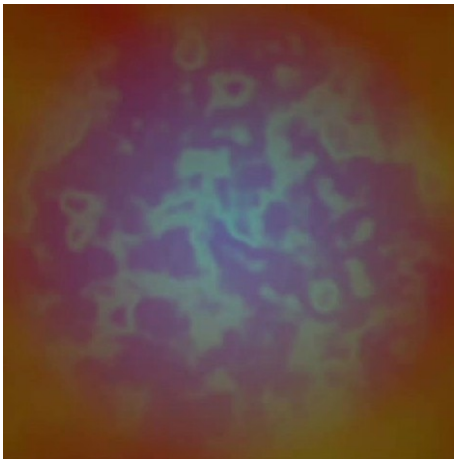
Con le Textures *Stencil*, l'ordine delle Texture nella pila di un Material assume particolare importanza in quanto, come anticipato, le *Stencil* coprono le Texture sottostanti; nel nostro caso, non c'è ancora una Textures sotto la *Stencil* e l'altra Textures presente, *Clouds*, si trova nello slot superiore, per cui non risente degli effetti di *Stencil*.

Aggiungiamo quindi, nel terzo slot, una nuova Texture, ad esempio *Marble*, con un bel po' di turbolenza ed attivando la *Color Band*, magari con colori diversi da quelli utilizzati per *Clouds*, in modo da notare le differenze.



La scheda della terza Texture, di tipo Marble

Per questa Texture, come per la Texture *Clouds*, lasciamo attivato *Color Diffuse*: in questo modo, i colori di *Marble* copriranno quelli della Texture precedente, *Clouds*, ma dal momento che per *Marble* è attivo l'effetto della Texture *Stencil*, posta per così dire "al piano superiore", *Marble* coprirà *Clouds* solo in certi punti, ed in particolare nel nostro caso (cioè con *Blend Spherical* per *Stencil*) solo nella parte centrale. Possiamo verificare quanto detto lanciando un rendering di prova.



Porzione del nuovo rendering ottenuto

Disattivando la Texture *Stencil*, *Marble* coprirà interamente *Clouds*.

Le Textures *Stencil* coprono TUTTE le Textures presenti ai piani inferiori nella pila di Textures di un Material, non solo la Texture successiva, per cui spostando *Blend Spherical* in cima alla pila copriremo, nel nostro caso, sia *Clouds* che *Marble*.

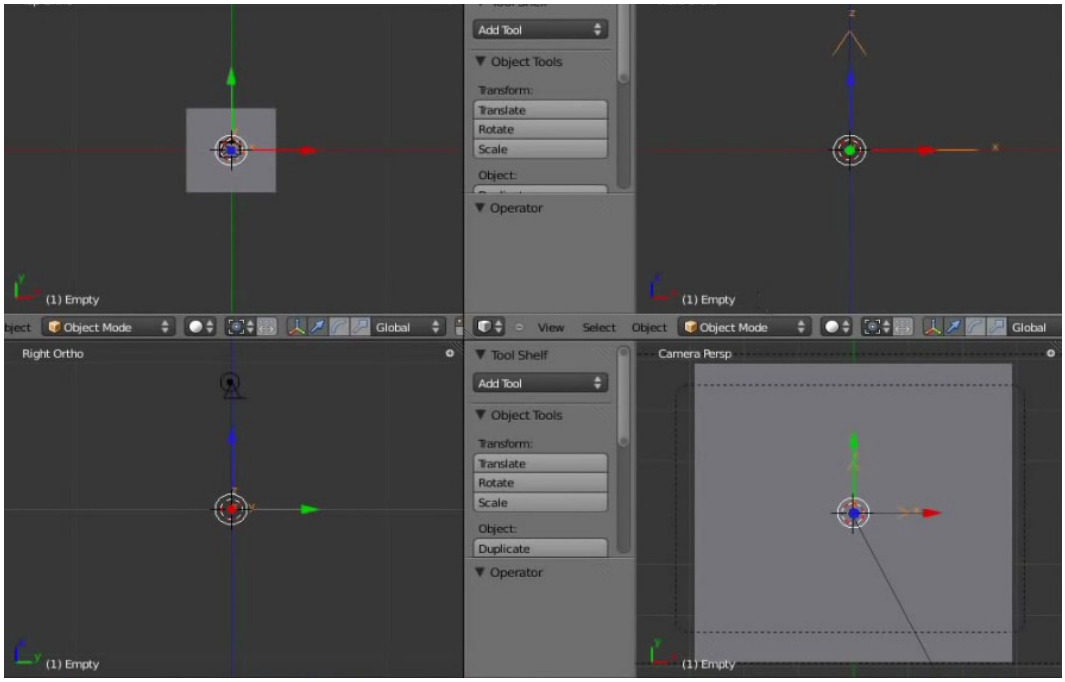
Al posto delle Texture procedurali avremmo potuto utilizzare delle Textures immagini, così come avremmo potuto utilizzare altre Textures per la *Stencil*, ad esempio *Clouds* per simulare un po' di rumore o creare un livello di "sporcizia" per un Material; le applicazioni sono molteplici, questo è solo un esempio, il concetto chiave è che le zone chiare di una Texture *Stencil* lasciano passare le Textures sottostanti, mentre le zone via via più scure, fino al nero, ne disattivano l'effetto.

* * *

Logo con effetto Displacement animato

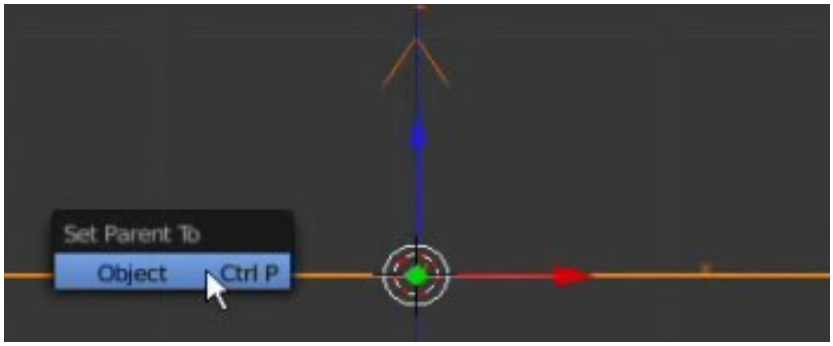
In questo tutorial vedremo come animare l'effetto *Displace* in Blender 2.5.
Displace viene utilizzato, in genere, per creare loghi con rilievi o incisioni, ma non solo; ad esempio, in passato ho mostrato in un altro tutorial come utilizzarlo per creare bandiere mosse dal vento senza dover ricorrere agli effetti *Cloth* e *Wind*.

Nella scena visibile nell'immagine qui di seguito, comunque, è presente un Plane che giace sul piano XY della scena. Proprio sul Plane, con lo stesso orientamento, è presente una *Empty*. Tale *Empty* ci servirà come oggetto di riferimento per ruotare, scalare e traslare la Texture per il *Displacement* sul Plane.



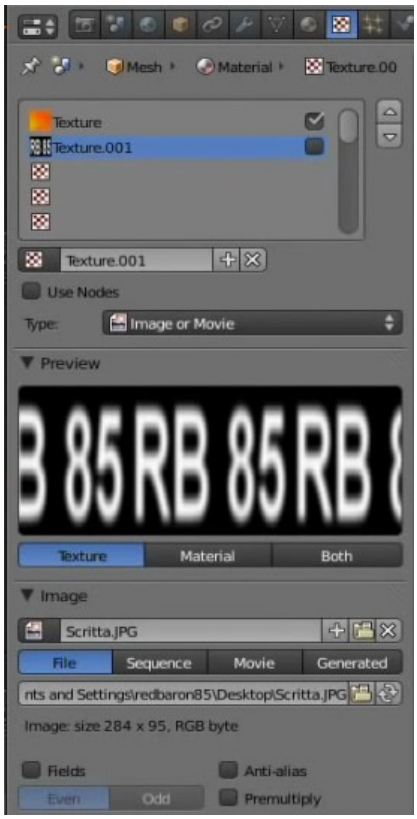
La scena utilizzata per questo tutorial

Leghiamo la *Empty* al Plane con una relazione di parentela, in modo che trasformando il Plane (con rotazioni, traslazione e scaling) verrà trasformata coerentemente anche la *Empty*; per fare ciò, selezioniamo prima la *Empty*, poi il Plane (tenendo premuto *SHIFT* mentre facciamo click col tasto destro del mouse, per fare una selezione multipla), quindi premiamo *CTRL P* e clicchiamo sulla voce "*Set Parent To Object*" nel menù che apparirà a video.



Realizzare la relazione di parentela tra il Plane e la Empty; selezionare prima la Empty (figlia), poi il Plane (padre) con una selezione multipla, quindi premere CTRL P e confermare

Il Plane è provvisto di un Material con una Texture di base che agisce sul canale *Diffuse Color*, giusto per dargli un po' di colore. È presente anche una seconda Textures, un'immagine caricata da file, il cui effetto sul Material è però disattivato, in quanto verrà utilizzata dal modificatore *Displace*, come vedremo.



La Texture immagine (un semplice "logo") caricata nella scena

Questa immagine, che nel mio caso è una semplice scritta, è il nostro "logo" e ci servirà per il *Displacement*. Il *Displacement* mette in risalto le porzioni bianche: ecco perché la scritta è bianca su sfondo nero.

Con il Plane selezionato, passiamo in Edit Mode e suddividiamo l'oggetto un bel po' di volte, in modo da garantire una buona qualità per l'effetto *Displacement*.

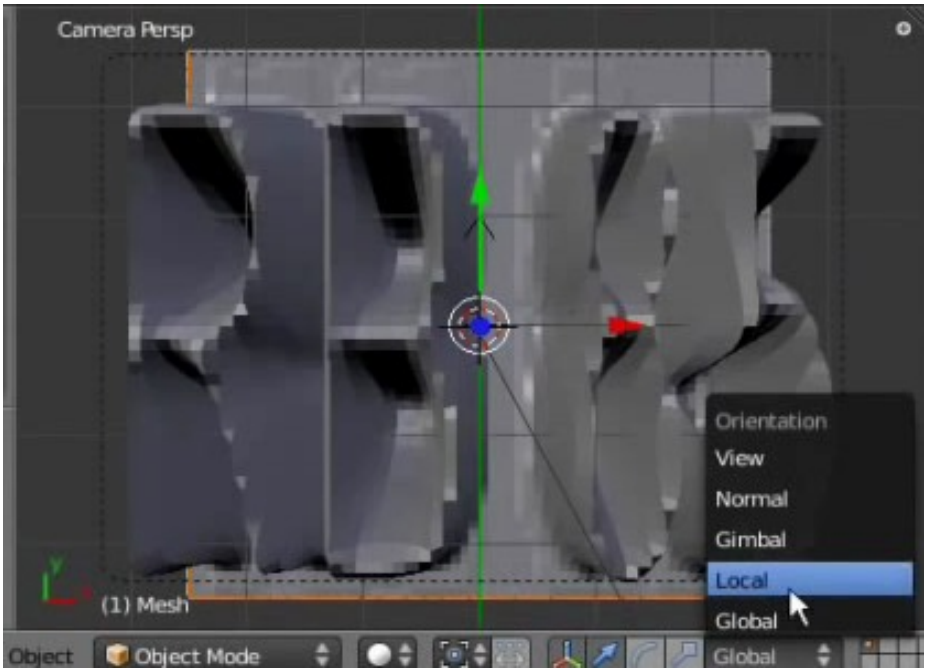
Aggiungiamo quindi un *Vertex Group* per tale oggetto cliccando sul simbolo + nella sezione *Vertex Group*, appunto, nella scheda *Object Data*, all'interno della *Properties Window*.

Il *Vertex Group* ci serve per specificare in quali porzioni della mesh (porzioni raggruppate appunto come insieme di vertici) applicare l'effetto *Displacement*; per questo esempio, impostiamo un *Vertex Group* di default, per cui l'effetto verrà applicato su tutto il Plane.

La scheda (ancora non impostata) del modificatore Displace, per il Plane

Adesso, sempre col Plane selezionato, apriamo la scheda *Modifiers* nella *Properties Window* ed aggiungiamo un modificatore *Displace* per tale oggetto.

Come Texture per tale modificatore impostiamo la Textures immagine precedentemente caricata. Impostiamo, come direzione, *Z*, in quanto la direzione uscente dal Plane, lungo la quale vogliamo applicare l'effetto del rilievo, è appunto Z, come possiamo verificare cambiando la visualizzazione degli assi da *Global* a *Local* mediante il selettore *Orientation* presente nell'header di ogni finestra di navigazione *3D View*.

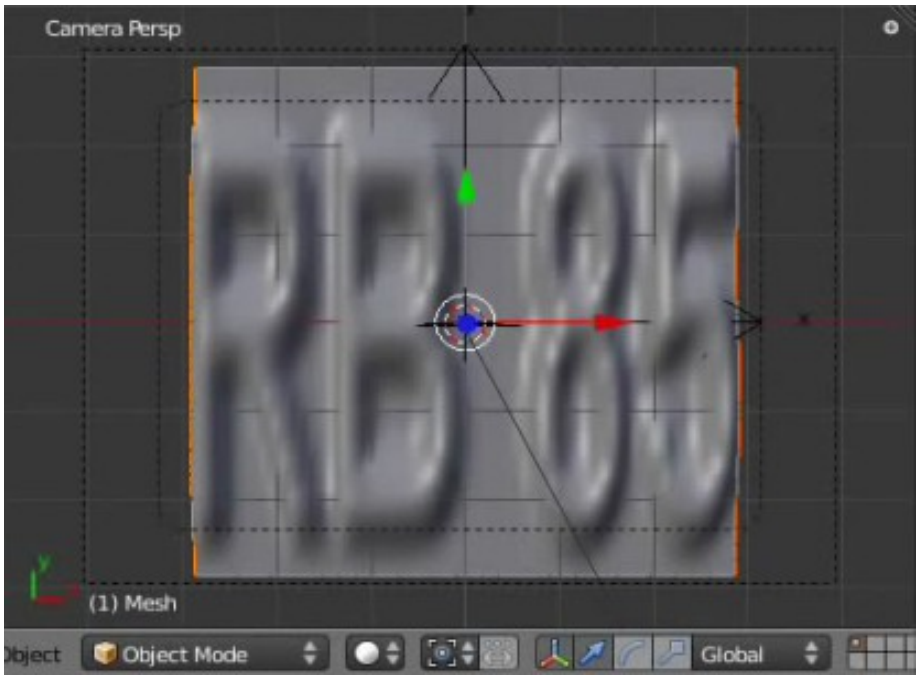


Visualizzazione degli assi locali dell'oggetto

Come *Vertex Group* impostiamo quello creato un attimo fa, che raggruppava tutti i vertici della mesh, per cui l'effetto verrà applicato, appunto, a tutto il Plane.

In Texture Coordinates selezioniamo l'opzione *Object* e, nel campo *Object* che apparirà sotto il selettore, scriviamo il nome dell'oggetto *Empty*.

Per sistemare meglio la scritta, impostiamo *Strength* ad esempio a 0.1 e selezioniamo la *Empty*, ruotandola, scalandola o traslandola per ottenere l'effetto desiderato.



Nuovo screenshot della scheda del modificatore Displace, con tutti i parametri impostati

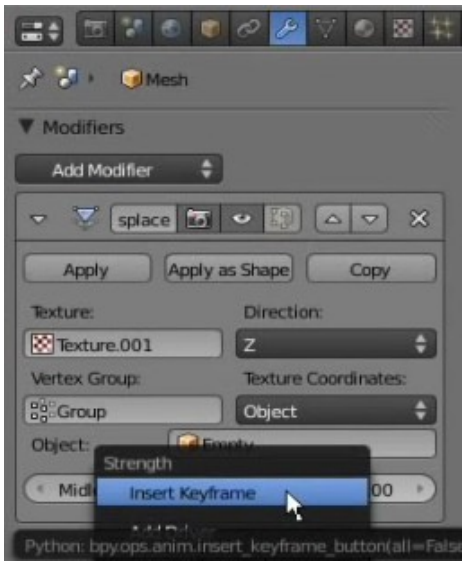
Terminato questo lavoro, selezioniamo nuovamente il Plane, impostando magari lo *Shading* a *Smooth* attraverso la relativa voce presente nella *Tool Shelf* di una qualsiasi finestra di navigazione *3D View*, per rendere più "dolce" l'effetto.

Sempre col Plane selezionato, torniamo nuovamente alla scheda del modificatore *Displace*, nella *Properties Window*.

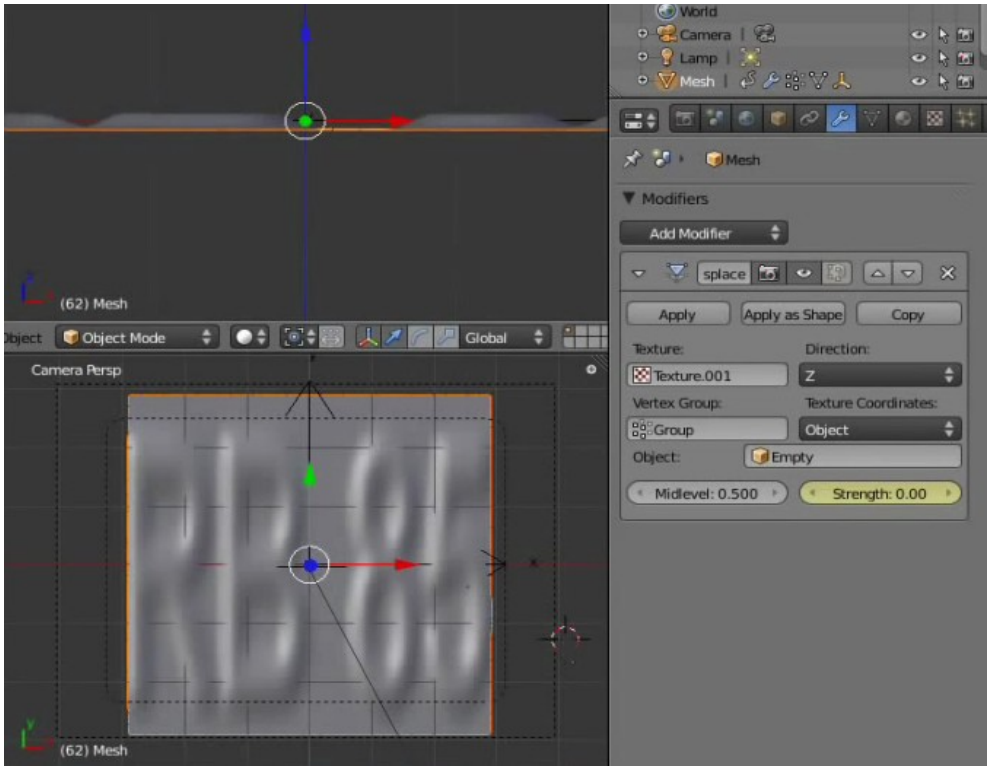
Assicurandoci di essere al frame 1 dell'animazione, portiamo a 0 il valore di *Strength*.
Facciamo click col tasto destro del mouse proprio sul pulsante numerico *Strength* del modificatore *Displace* e scegliamo, dal menù che apparirà a video, la voce *Insert Keyframe*.

Adesso spostiamoci al frame finale dell'animazione, ad esempio 250 (per creare un'animazione di 10 secondi con 25 frames al secondo), impostiamo quindi *Strength* a 0.3 ed inseriamo un keyframe di animazione come fatto poco fa, ossia cliccando col tasto destro del mouse su *Strength* e selezionando *Insert Keyframe* per tale campo.

Blender ha creato per noi l'interpolazione dei valori di *Strength*, ossia l'animazione dell'effetto *Displacement*. Possiamo osservare l'effetto ottenuto premendo *ALT A* mentre il focus del mouse è in una finestra di navigazione *3D View* per avviare l'anteprima dell'animazione, oppure possiamo lanciare direttamente il rendering della stessa.



Inserimento di un Keyframe di animazione per il parametro Strength del modificatore Displace



Fotogramma dell'animazione ottenuta

Da notare che, dal momento che abbiamo legato la *Empty* che controlla l'effetto *Displacement* al *Plane*, possiamo anche ruotare o traslare il *Plane* nello spazio (e generare anche animazioni a livello oggetto sfruttando tali trasformazioni): l'effetto verrà applicato comunque in maniera corretta.

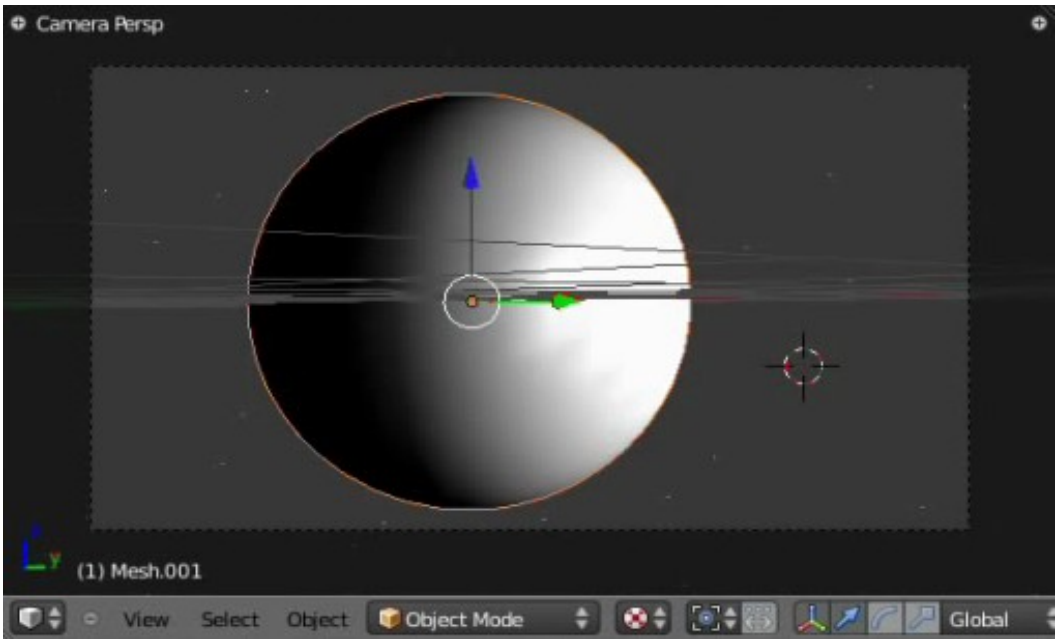
* * *

Mappare e renderizzare Textures di pianeti

In questo tutorial vedremo come mappare e renderizzare correttamente una Texture su di una sfera in Blender 2.5 per realizzare un pianeta, nel nostro caso Giove.

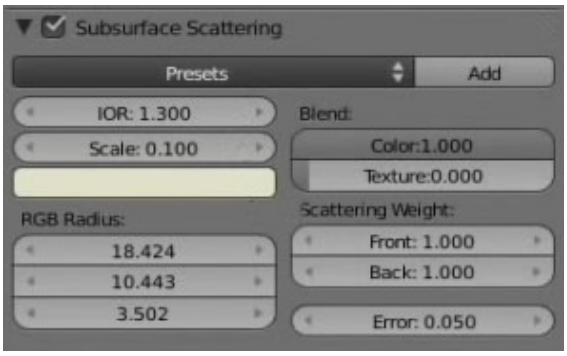
Faremo inoltre uso dell'effetto *Sub Surface Scattering* per simulare la diffusione dei raggi luminosi dovuti alla presenza dell'atmosfera, in modo da evitare uno stacco netto tra le zone in ombra e quelle illuminate.

L'oggetto da utilizzare per il pianeta è, chiaramente, una sfera; possiamo aggiungere una sfera con le impostazioni di default, ma dovremo necessariamente impostare la modalità di ombreggiatura *Smooth* cliccando su tale voce nella sezione *Shading*, nella *Tool Shelf*.



La sfera, con ombreggiatura Shading Smooth, inserita nella scena

Il Material da applicare alla sfera è un Material con ombreggiatore diffuso di default, ombreggiatore speculare disattivato, cioè con intensità a 0 (o sembrerà di osservare una boccia di vetro o plastica, non un pianeta) e soprattutto effetto *Sub Surface Scattering* attivato.



La scheda dell'effetto SSS per il Material

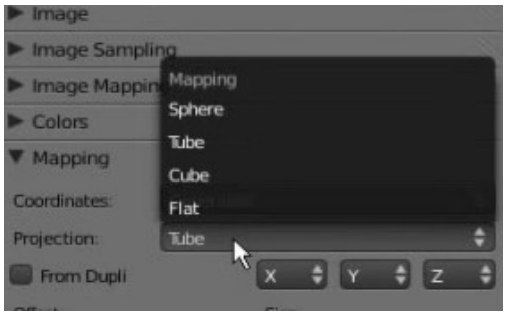
Sub Surface Scattering, o *SSS*, serve a simulare la diffusione dei raggi luminosi sulla superficie di un oggetto; l'ho utilizzato in molti tutorial per realizzare oggetti come perle, marmo, saponette ed altro ancora, ma va benissimo anche per simulare la diffusione dei raggi luminosi dovuti alla presenza dell'atmosfera nei pianeti, o perlomeno nei pianeti provvisti di un'atmosfera degna di questo nome, mentre non va bene ad esempio per Mercurio, la Luna, asteroidi o piccoli corpi celesti rocciosi; ad ogni modo, le impostazioni di default che ho utilizzato nella mia scena sono quelle fornite di serie da Blender per il latte scremato (*skim milk*).

Un parametro molto importante di *SSS* è la scala dell'effetto: senza addentrarmi troppo nei particolari, dico solo che ad un valore maggiore corrisponderà una dispersione maggiore sulla superficie della sfera, soprattutto sulle parti in ombra, mentre noi vogliamo solo sfumare leggermente il confine luce-ombra; nel mio caso, ho impostato la scala di *SSS* a 0.1 con una sfera di dimensione 5.

La Texture immagine utilizzata per l'oggetto

L'unica Texture presente per il Material è una Texture di tipo immagine: in pratica, una mappa fotografica del pianeta in questione, utilizzata proprio per il rendering della scena. Texture del genere sono disponibili su vari siti, ma per immagini in alta risoluzione consiglio le immagini di pubblico dominio della NASA, l'ente spaziale americano.

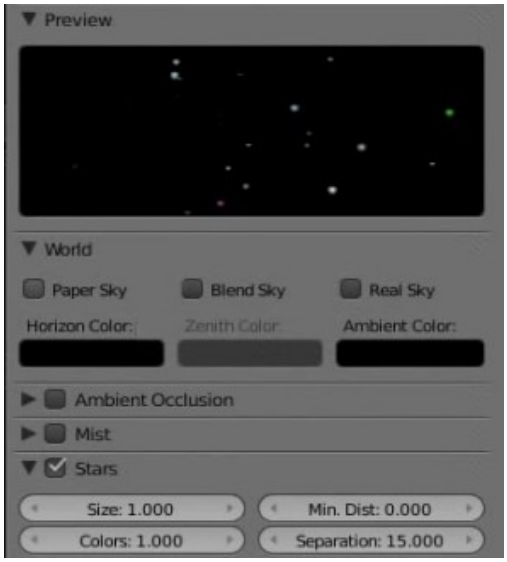
Come anticipato, questa è l'unica Texture presente per il Material: Giove è un pianeta gassoso e, quindi, "liscio", per cui non ci servono mappe di bumping.
Per mappare correttamente una Texture del genere su una sfera, non dobbiamo selezionare (in *Mapping*) la modalità di proiezione sferica, "*Sphere*", né tanto meno quella di default (cioè *Flat*, piatta), ma la modalità *Tube*, cioè cilindrica: in questo modo, l'immagine verrà avvolta intorno alla sfera, adattandosi ad essa.



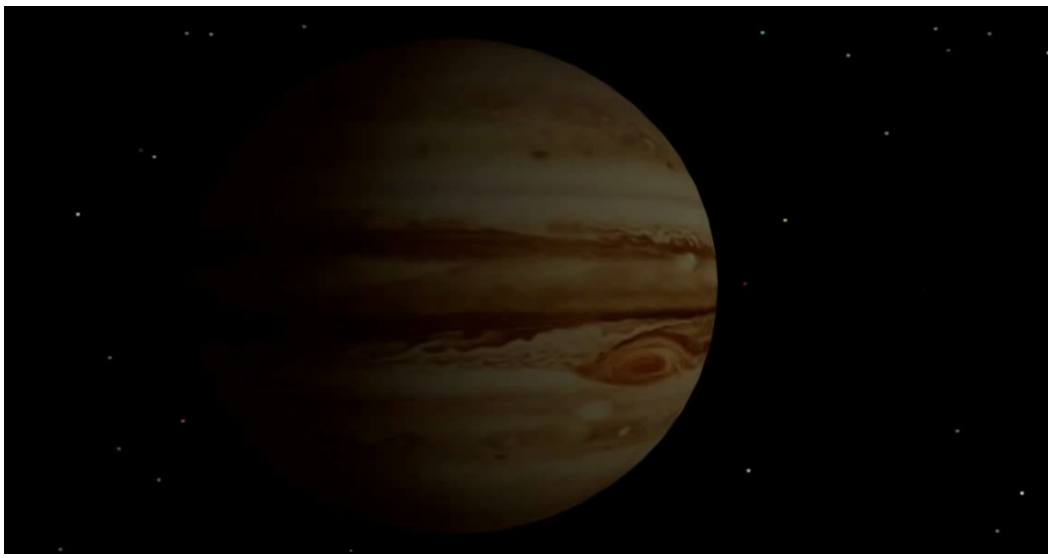
Mappatura Tube per questo particolare tipo di Texture immagine

Nella scena è presente una sola fonte di luce di tipo *Point Light* con intensità 2 e posta ad una buona distanza dalla sfera, in modo da illuminare tutta la porzione che dovrebbe essere, appunto, illuminata.

Completano il tutto, nella sezione *World* della *Properties Window*, le impostazioni ambientali, con colore di sfondo nero ed effetto *Stars* attivato con *Size* e *Colors* a 1.



Le impostazioni di World Stars

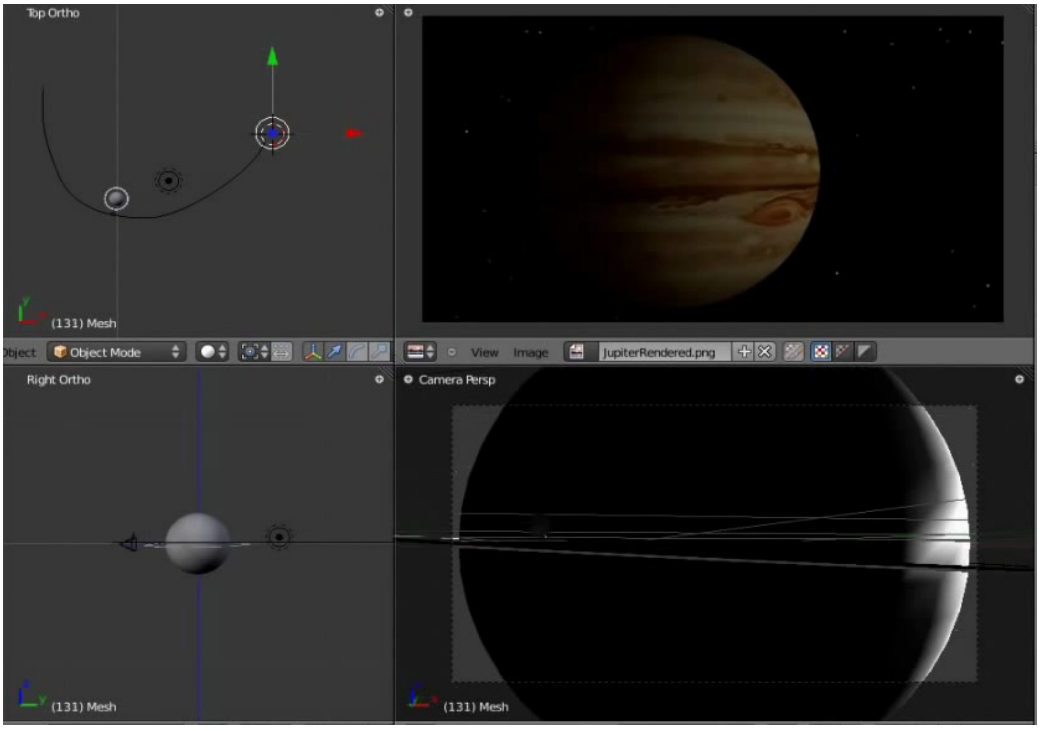

Il rendering finale

* * *

Jupiter Flyby, il sorvolo di Giove: analisi di un'animazione

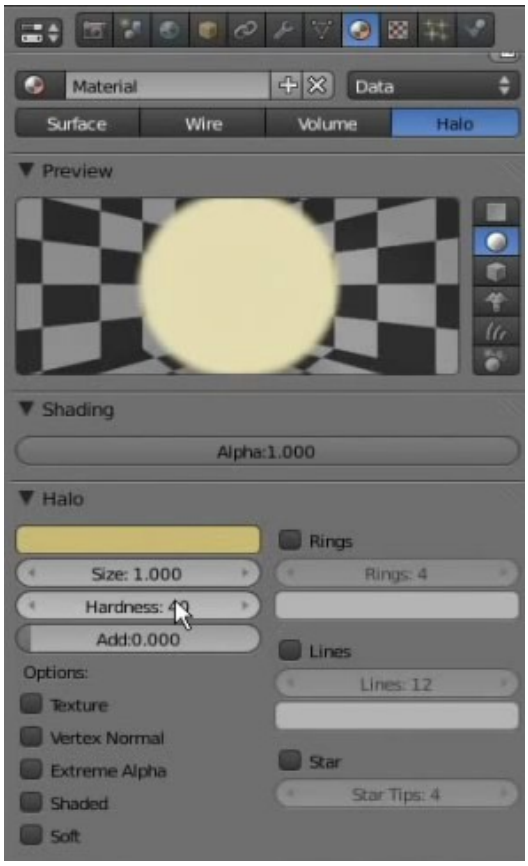
Questo tutorial è l'analisi di un'animazione realizzata con Blender 2.5: si tratta della simulazione del flyby (il sorvolo ravvicinato) del pianeta Giove. In questo tutorial prenderemo in esame concetti come le stelle in Blender, l'effetto *Sub Surface Scattering*, il *Constraint Track To*, l'animazione della telecamera lungo un percorso, l'animazione della trasparenza e l'effetto *Halo* per i Materials.

Nella scena sono presenti: una fonte di luce di tipo *Lamp*, due sfere (che rappresentano Giove e il Sole, ovviamente non in scala), una telecamera, una curva *Bezier* (che è il percorso della telecamera) e una specie di tubo, che raffigura l'anello di Giove (non tutti sanno che Giove è provvisto di un anello sottilissimo, ma niente di spettacolare come i famosi anelli di Saturno; tra l'altro, l'anello di Giove è visibile solo in controluce, con una controluce data dal Sole, infatti è invisibile dalla Terra; in questo tutorial ricreeremo quest'effetto).



La scena di partenza utilizzata in questo tutorial

Alla creazione e texturizzazione del pianeta Giove ho dedicato un tutorial precedente, per cui qui mi limito a dire che si tratta di una semplice sfera con *Shading Smooth*, Texture immagine (una foto del pianeta) mappata con mappatura *Tube* ed effetto *Sub Surface Scattering* attivato con impostazioni *Skim Milk*, ossia latte scremato. L'effetto *Sub Surface Scattering*, o *SSS*, serve a simulare la diffusione dei raggi luminosi dovuta alla presenza dell'atmosfera, in modo da creare stacchi non netti tra le zone in luce e quelle in ombra. La sfera ruota su se stessa nel corso dei 250 frames dell'animazione con una semplice animazione sul canale *ROT* in modo oggetto.



Il Material Halo per la sfera che rappresenta il Sole

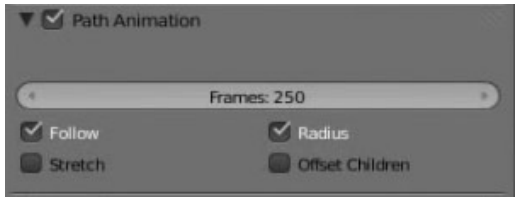
La fonte di luce presente è, come anticipato, una *Lamp*, con intensità 2.

La sfera che rappresenta il Sole è costituita, in realtà, da un solo vertice: ho quindi creato una sfera e, dopo aver fatto delle prove non soddisfacenti con l'effetto *Halo*, ho rimosso tutti i vertici tranne uno; il Material associato all'oggetto è, quindi, un Material *Halo*, con impostazioni di *Hardness* e intensità tali da creare un punto luminoso circondato da un lieve bagliore.

Le stelle di sfondo sono state implementate mediante lo strumento *Stars* delle impostazioni del mondo, *World*, di Blender; è bastato mettere il colore a 1 (in modo da creare stelle di vari colori), un valore basso per la dimensione e uno elevato per la distanza minima dalla telecamera. Sempre in *World*, bisogna impostare come nero il colore di sfondo dell'ambiente.

Le impostazioni della scheda World: stelle e colore dell'ambiente

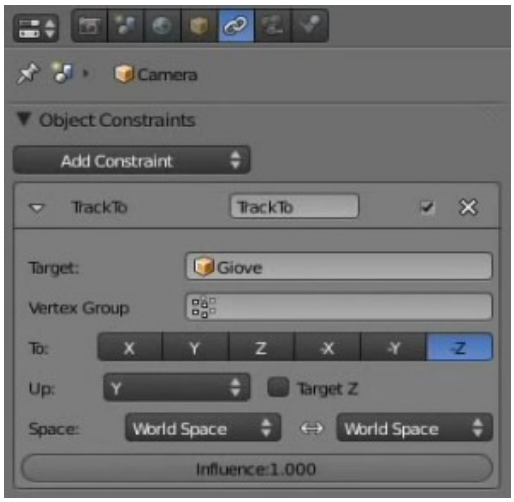
Il percorso è, come anticipato, una semplice curva *Bezier*, opportunamente modellata per conferirle la forma visibile nella scena. Nella scheda *Object Data* della *Properties Window* è necessario attivare, per tale oggetto, *Path Animation*, con 250 frames, e *Follow*.



Le impostazioni Path Animation e Follow per la curva-percorso

La relazione che lega la telecamera a tale curva è una relazione di tipo *Parent*; in particolare, la telecamera è figlia della curva percorso.

Questa relazione fa muovere la telecamera lungo la curva, ma non ci consente di far puntare alla telecamera il pianeta Giove in qualsiasi momento; per far ciò, dobbiamo selezionare la telecamera, aggiungere per la stessa un *Constraint* di tipo *Track To* ed impostare, per questo vincolo, *-Z* per *TO*, *Y* per *UP* e il nome della sfera che rappresenta Giove nel campo *Target*.



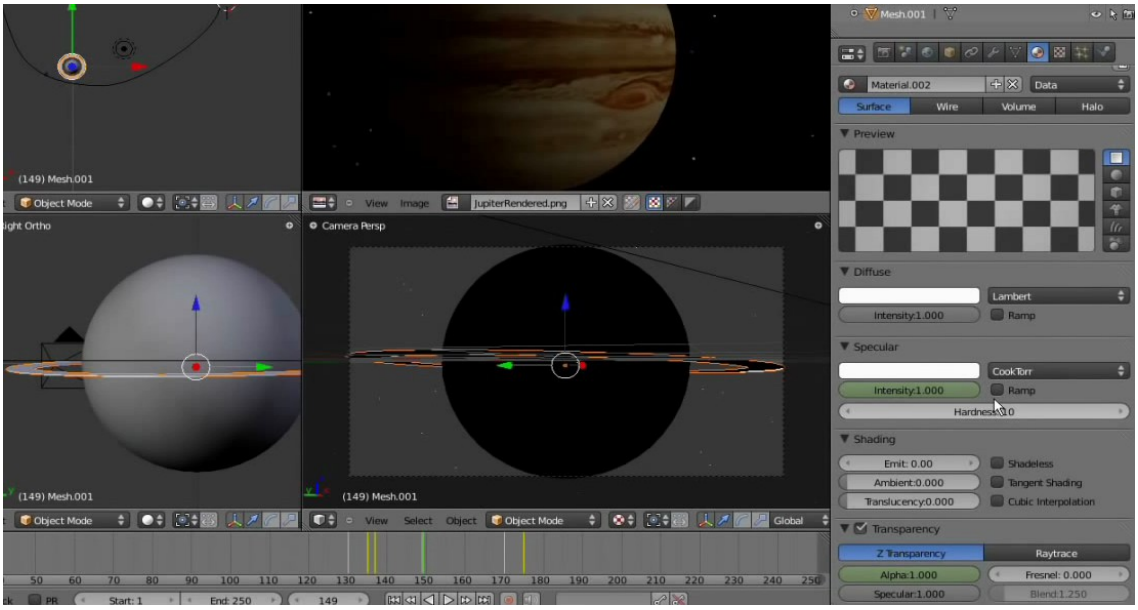
Le impostazioni del Constraint Track To per la telecamera

Passiamo all'anello: esso è, come anticipato, un semplice tubo, sottile, con un Material di base (ma con *Specular* a 1 e *Hardness* 10) e che deve essere visibile solo quando Giove è in controluce; per tale motivo, al frame 0 ho disattivato *Traceable* e impostato l'*Alpha*, il valore di trasparenza in modalità *Z Transp*, e il valore di *Specular* a 0, aggiungendo un keyframe di animazione.

Utilizzando i keyframe dei campi *Traceable*, *Alpha* e *Specular*, ho reso visibile l'anello solo tra i frame 130 e 170, ossia quando è in controluce.

Anche per questo Material è presente *Sub Surface Scattering*, ma con scala 1, più alta di quella del Material associato a Giove, in modo da far disperdere maggiormente la luce sulla superficie della mesh. Al Material è associata una Texture *Clouds* con *Size* 0.05, per aggiungere un po' di granularità all'aspetto dell'oggetto, che agisce sul canale *Color*.

Il colore dell'effetto di tale Texture è bianco puro.



Le impostazioni del Material dell'anello

Il rendering definitivo, realizzato con risoluzione 1280x720, ha richiesto circa 2 ore su un sistema con processore Core 2 Duo, ma va detto che il tempo è richiesto, più che altro, per l'effetto *SSS* del Material del pianeta Giove.

Il rendering finale (screenshot dell'animazione)

* * *

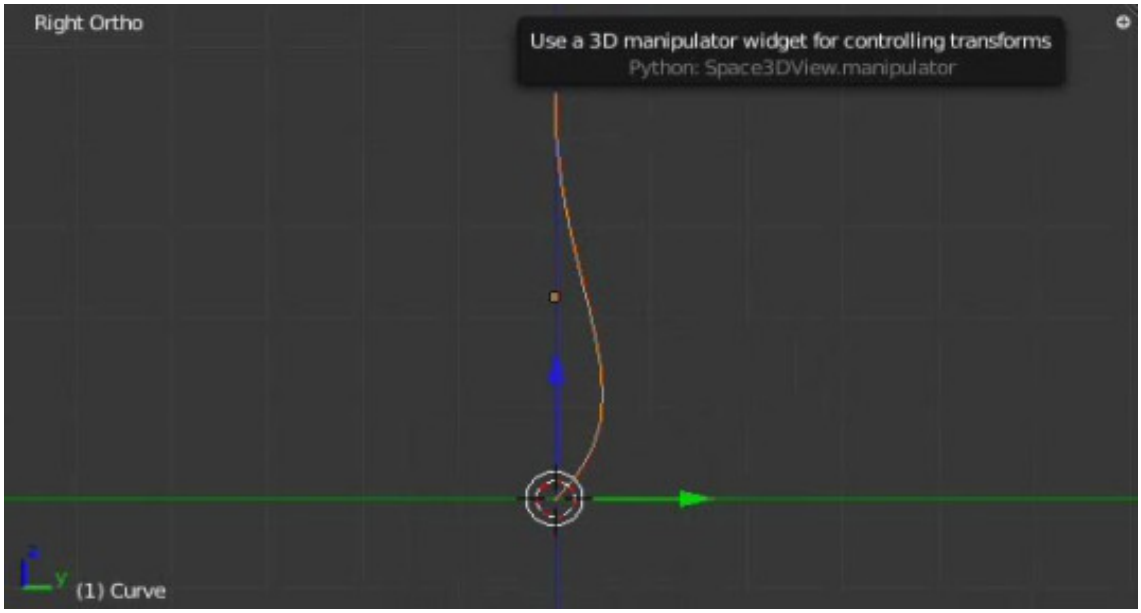
Modellazione: ampolle, decanter, bottiglie, vasi con spin e solidify

In questo tutorial vedremo come creare solidi come vasi, ampolle, bottiglie, bicchieri ed altro ancora con una semplice curva profilo e conferendo loro uno spessore con il nuovo modificatore *Solidify*; quella discussa qui è una delle possibili tecniche, non l'unica, per ottenere tale scopo.

Questo tutorial è stato realizzato con la versione 2.5.2 di Blender, provvista del modificatore *Solidify* per le mesh. Iniziamo.

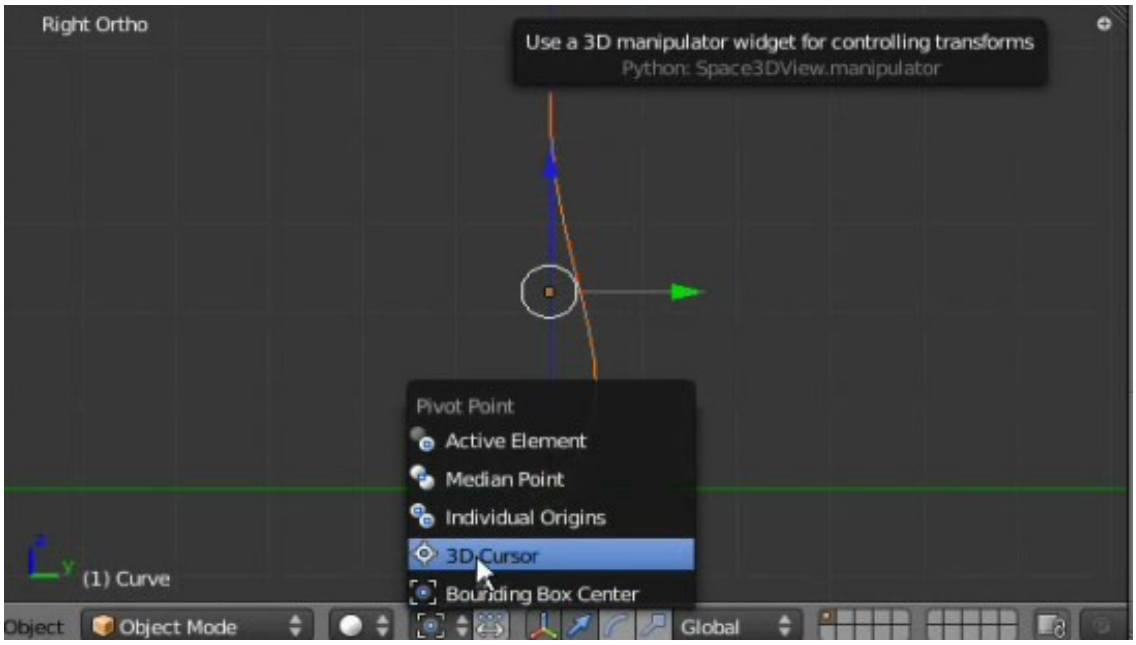
Posizioniamo il cursore del mouse al centro del sistema di riferimento; possiamo eseguire questa operazione premendo *N* per aprire una finestra *Transform Properties* in una finestra di navigazione 3D ed impostando 0 come coordinata *X*, *Y* e *Z* nella sezione *3D Cursor* di tale scheda.

Aggiungiamo quindi una semplice curva *Bezier* alla scena, orientandola in modo da vedere il profilo dritto in una finestra *SIDE* o *FRONT*, ma non in *TOP*.



La curva Bezier inserita nella scena

Conviene cambiare il *Pivot Point*, il perno dell'operazione, in *3D Cursor*, mediante il selettore *Pivot* nell'header di una finestra di navigazione 3D, dal momento che definire la posizione del cursore è un'operazione davvero semplice.



Passaggio al Pivot Point 3D Cursor

Per facilitare le operazioni di rotazione, possiamo utilizzare lo *SNAP*, che consiste nel premere *R* per attivare la rotazione, quindi premere *CTRL* e, mentre si tiene premuto *CTRL*, muovere il mouse per ruotare la selezione: in questo modo, avremo rotazioni di 5 gradi per volta.

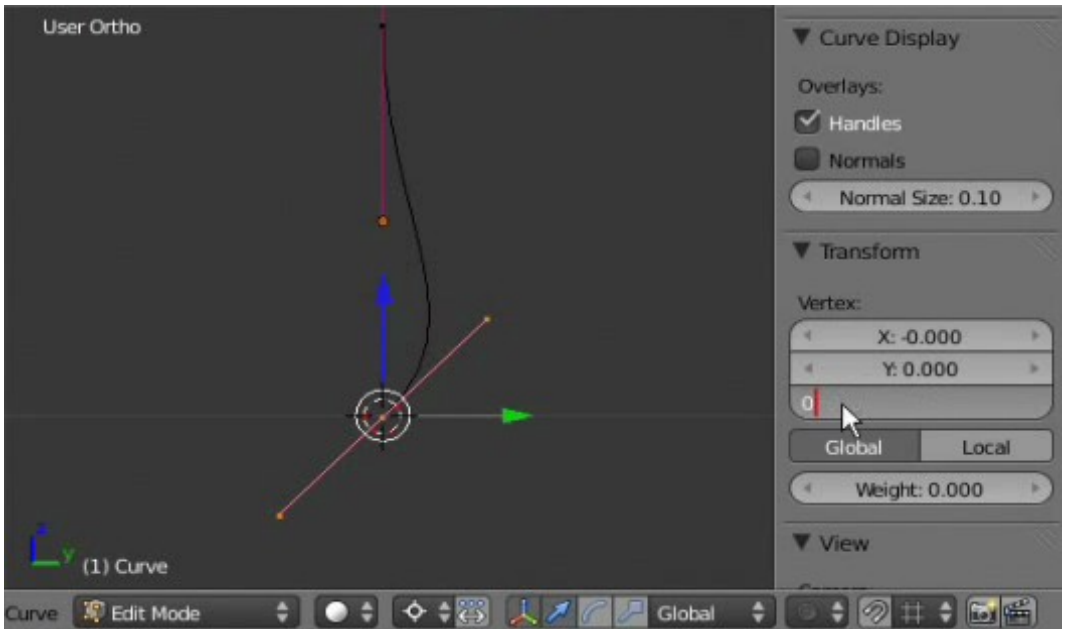
Tale operazione può essere svolta sia in Object Mode, su tutta la curva o su più oggetti selezionati, che in Edit Mode, su un singolo vertice, una maniglia o insiemi di tali elementi.

Adesso, immaginate un asse mediano per la mesh che vogliamo creare, ad esempio un vaso, una bottiglia o un bicchiere; il concetto di asse mediano qui è molto importante per effettuare correttamente le operazioni che seguiranno.

Per comodità, definiamo come asse di simmetria o asse mediano della mesh l'asse *Z* globale di Blender passante per l'origine del sistema di riferimento nella scena, per cui dobbiamo posizionare il vertice centrale della base della mesh proprio nella coordinata 0,0,0.

Per spostare la curva o parti della stessa in maniera precisa, ricorriamo ancora una volta alla finestra *Transform Window*, specificando le coordinate della posizione in *Location*.

Per posizionare poi il vertice centrale della base proprio nel punto di coordinate 0,0,0, passiamo in Edit Mode, selezioniamo tale vertice (e magari disattiviamo la visualizzazione delle Normali deselezionando la casella *Display Normals* nella *Transform Window*), cambiamo la modalità di definizione delle coordinate da *Local* in *Global*, nella sezione *Transform*, e specifichiamo le coordinate per il vertice selezionato.



Utilizzo di Transform Window Location per posizionare con precisione la curva o parti (vertici) di essa

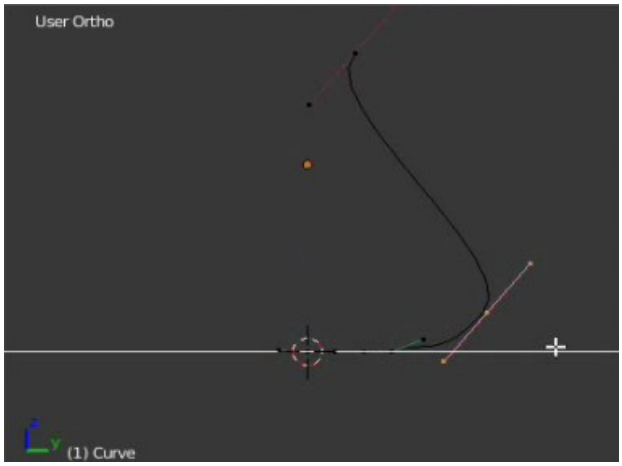
Adesso possiamo definire come vogliamo il profilo della nostra mesh lavorando in Edit Mode; possiamo ad esempio creare un nuovo pezzo di curva a partire da un vertice premendo *E*, per estrusione, oppure possiamo selezionare due vertici della curva e premere *W – Subdivide* per spezzare il segmento, inserendo un nuovo vertice.

Per definire il profilo possiamo scalare o ruotare le maniglie delle curve selezionando i vertici e premendo *S* o *R*, oppure possiamo traslare le singole maniglie selezionandole e premendo *G*; per cambiare i punti di controllo da curvi ad angoli, poi, dobbiamo selezionarli e premere *V* (questa è un'operazione che può tornare utile soprattutto con la base dell'oggetto, dove in genere si hanno segmenti dritti).

Anche nel trasformare le maniglie possiamo far uso della sezione *Transform – Vertex*, magari in modalità *Global*, nella *Transform Window.*

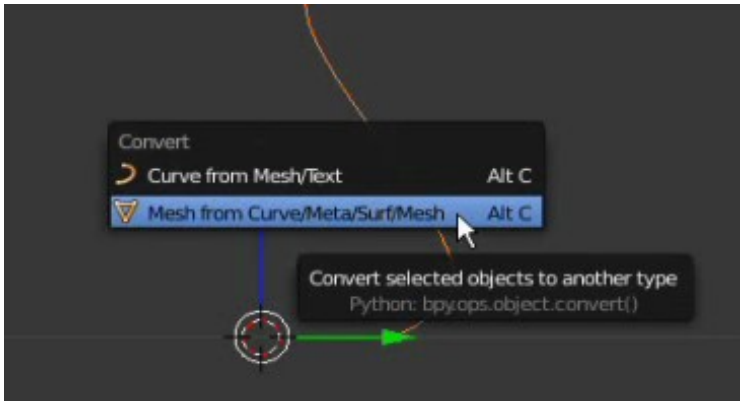
Possiamo anche vincolare una trasformazione (traslazioni, rotazioni e scaling) ad un solo asse: per farlo, basta premere il pulsante della trasformazione, ad esempio *G* per la traslazione, e la lettera dell'asse, che va premuta una volta per vincolare rispetto all'asse globale, quindi ad esempio *GX*, o due volte per vincolare rispetto all'asse locale, quindi ad esempio *RZZ*.

A differenza di quanto visto in altri tutorial (ad esempio un mio tutorial sulla creazione di un bicchiere), non chiudiamo la curva, definendo anche il profilo interno e, quindi lo spessore della stessa, ma definiamo solo il profilo esterno: ci penserà *Solidify* a creare lo spessore.



Il profilo finale dell'oggetto

Definito il profilo, dobbiamo cambiare tale oggetto (che al momento è una *Curve*) in Mesh, per cui passiamo in Object Mode, premiamo *ALT C* e, dal menù che apparirà, scegliamo *Mesh From Curve*.



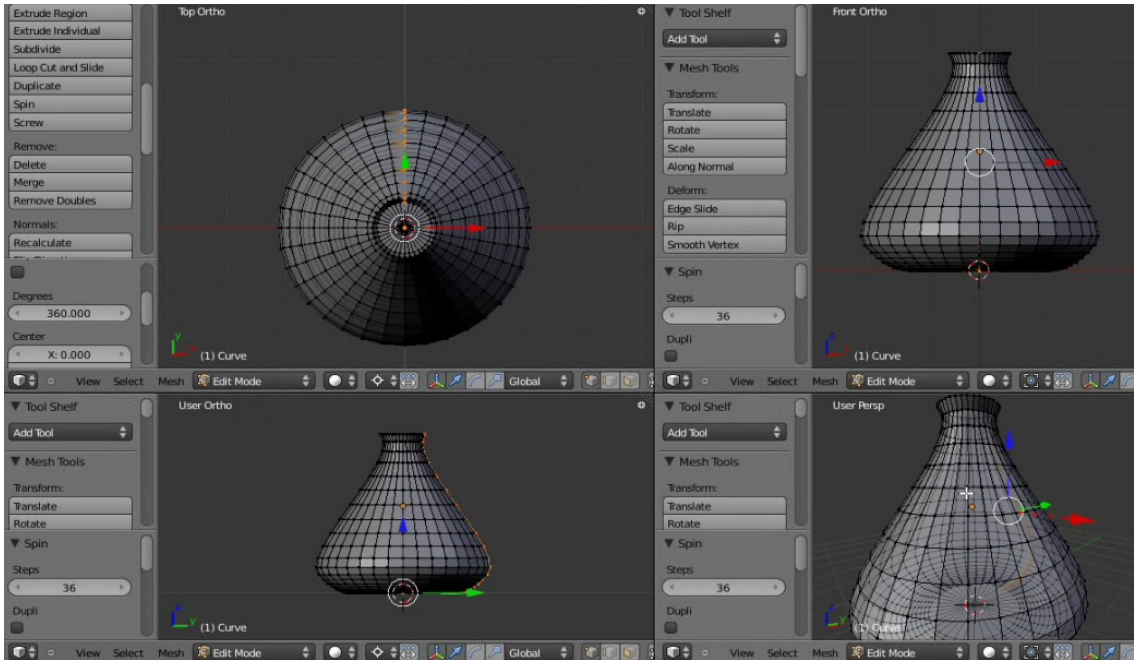
Passaggio da Curve a Mesh

Abbiamo quindi una mesh che rappresenta il profilo dell'oggetto, ma per creare il solido dobbiamo ricorrere al trasformatore *Spin*, che letteralmente significa "Tornio"; in Blender 2.5, *Spin* si trova nella *Tool Shelf*, ma è importante attivarlo nella finestra giusta.

In pratica, per il tornio ci servono un asse di riferimento intorno al quale effettuare la rotazione e un punto centrale; se, come nel mio caso, avete definito il profilo della mesh con la base "poggiata" sul piano XY, cioè col profilo visibile come "dritto" in una finestra *SIDE* o *FRONT*, allora l'asse di rotazione sarà *Z*, che è perpendicolare al piano visibile nella vista *TOP*, per cui *Spin* andrà attivato nella *Tool Shelf* della finestra *TOP*.

Il punto centrale della rotazione corrisponderà alla posizione del cursore 3D, per cui (ed ecco perché era importante posizionare bene il vertice centrale della base della mesh in 0, 0, 0) posizioniamo il *3D Cursor*, come sappiamo fare, in 0,0,0.

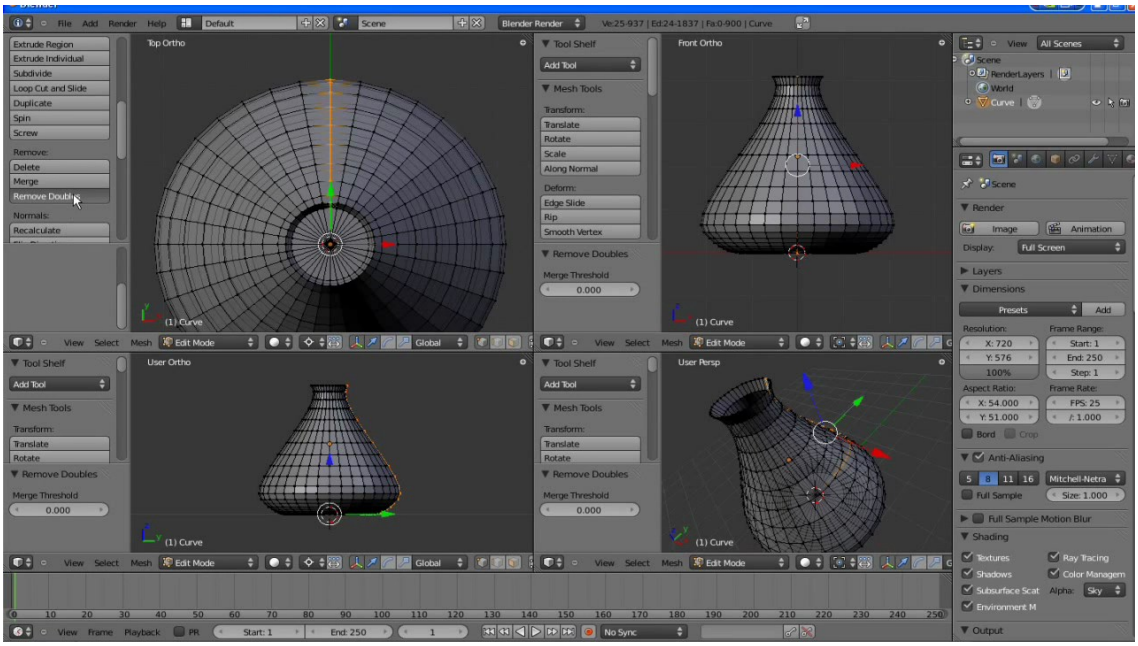
Con la mesh in Edit Mode e tutti i vertici selezionati (premendo *A*), nella *Tool Shelf* scegliamo quindi, nella vista *TOP ORTHO*, la voce *SPIN*, che di default effettua una rotazione di 90 gradi, ma possiamo cambiarla subito in 360 gradi (magari con 12 o più *steps*) mediante le voci relative a *Spin* presenti ora sul fondo della *Tool Shelf*.



Utilizzo di SPIN, in una vista Top Ortho (perpendicolare all'asse mediano dell'oggetto), con il 3D Cursor nel centro di rotazione e tutti i vertici della mesh selezionati in Edit Mode

L'aver fatto coincidere il perno e l'asse di *Spin* con il vertice centrale della base della curva garantirà una base chiusa per la mesh.

Spin tende a creare dei vertici duplicati nel punto di partenza, per cui conviene rimuovere i vertici duplicati in quella zona con *Remove Doubles*, nella *Tool Shelf*, regolando il livello di soglia di tale modificatore con l'apposito pulsante sul fondo della scheda, per evitare delle "spaccature" nella mesh; in casi gravi, si può optare per la cancellazione di metà mesh e la successiva duplicazione e rotazione a specchio, in Edit Mode, della restante metà.



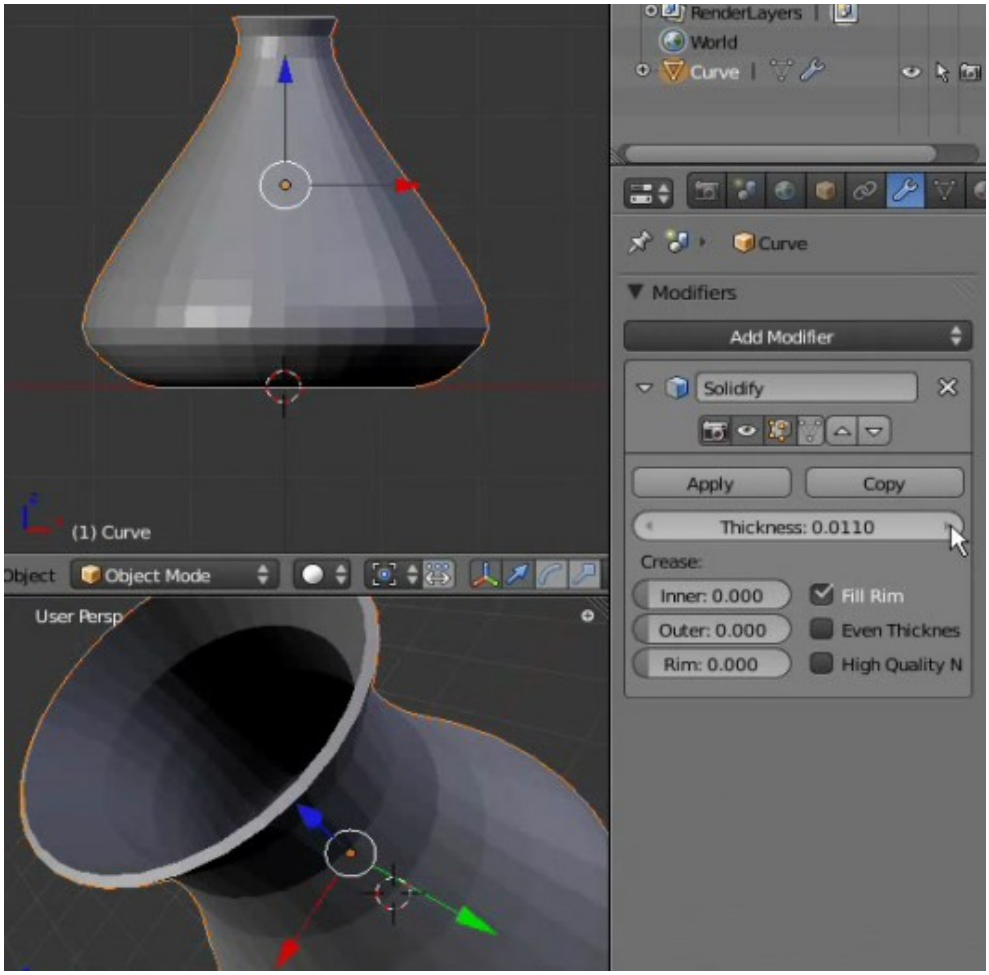
Utilizzo di Remove Doubles per eliminare i vertici duplicati realizzati da SPIN

Abbiamo ottenuto il nostro solido, ma ora dobbiamo dargli spessore.

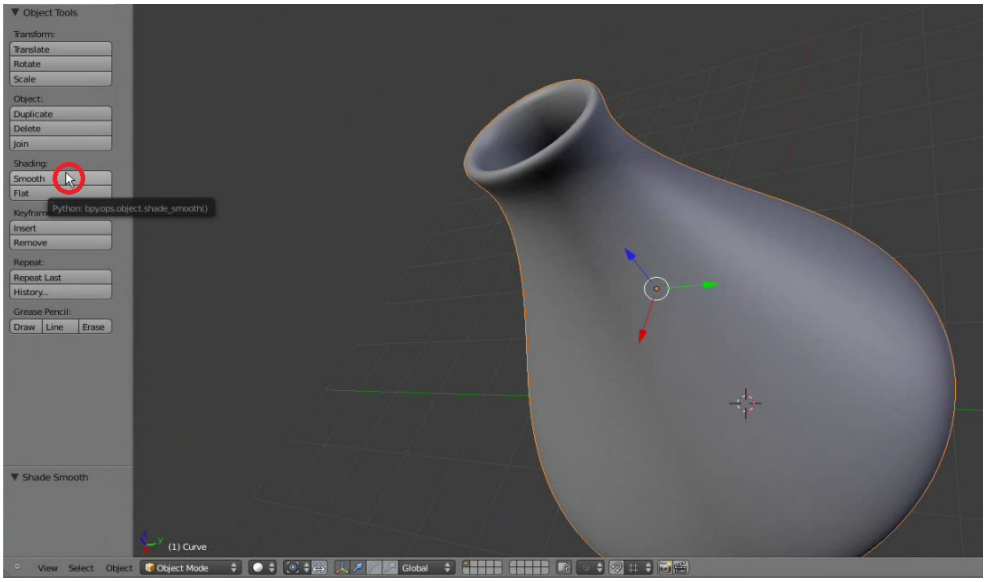
Passiamo in Object Mode e, nella scheda *Modifiers* all'interno della *Properties Window*, aggiungiamo un modificatore di tipo *Solidify* alla mesh. *Solidify* aggiungerà immediatamente un po' di spessore alla mesh, comunque possiamo regolare l'intensità dell'effetto agendo su *Thickness*.

Applichiamo l'effetto cliccando su *Apply*, operazione necessaria per applicare correttamente altri modificatori in seguito, ad esempio *Set Smooth* o *Subdivision Surface*.

Per smussare un po' il tutto, basta aggiungere un modificatore *Subdivision Surface* alla mesh; in genere, basteranno due suddivisioni (conviene più che altro impostare un buon valore, ad esempio 36, per il numero di passi *steps* di *Spin*).



Utilizzo del modificatore Solidify, variando il valore di Thickness



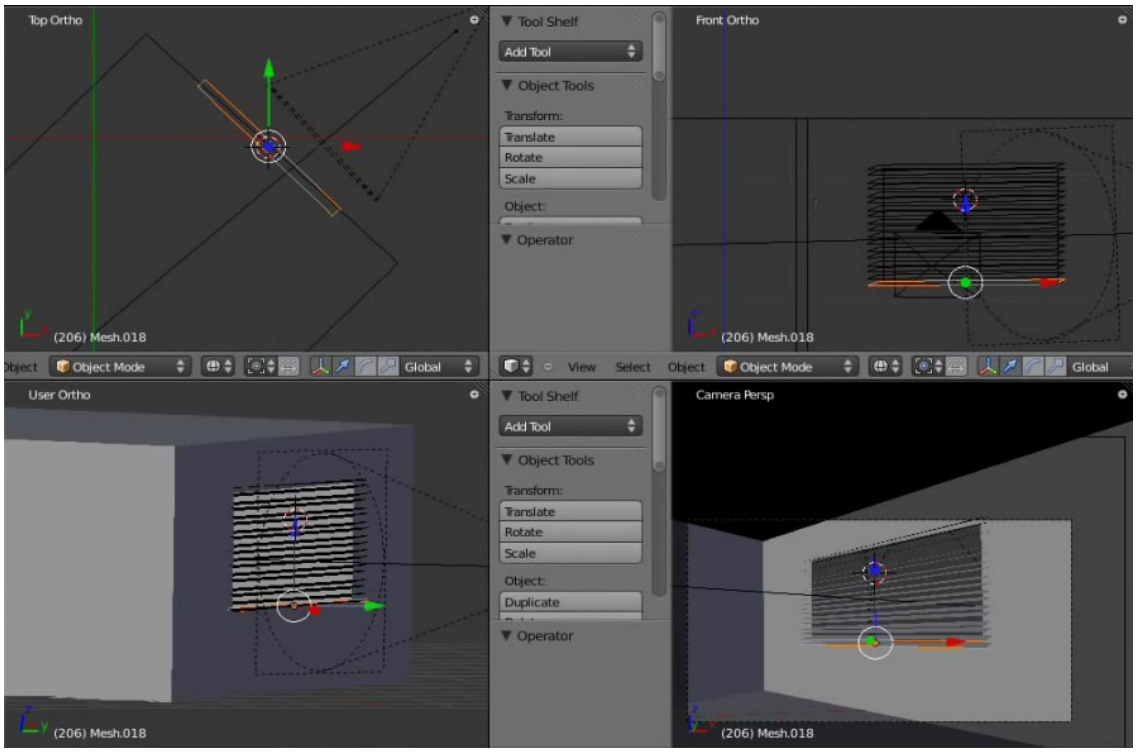
Utilizzo di Shading Smooth, dopo aver applicato un modificatore Subdivision Surface sull'oggetto finale

* * *

Animazioni con Constraint e Halo Light: tende alla veneziana

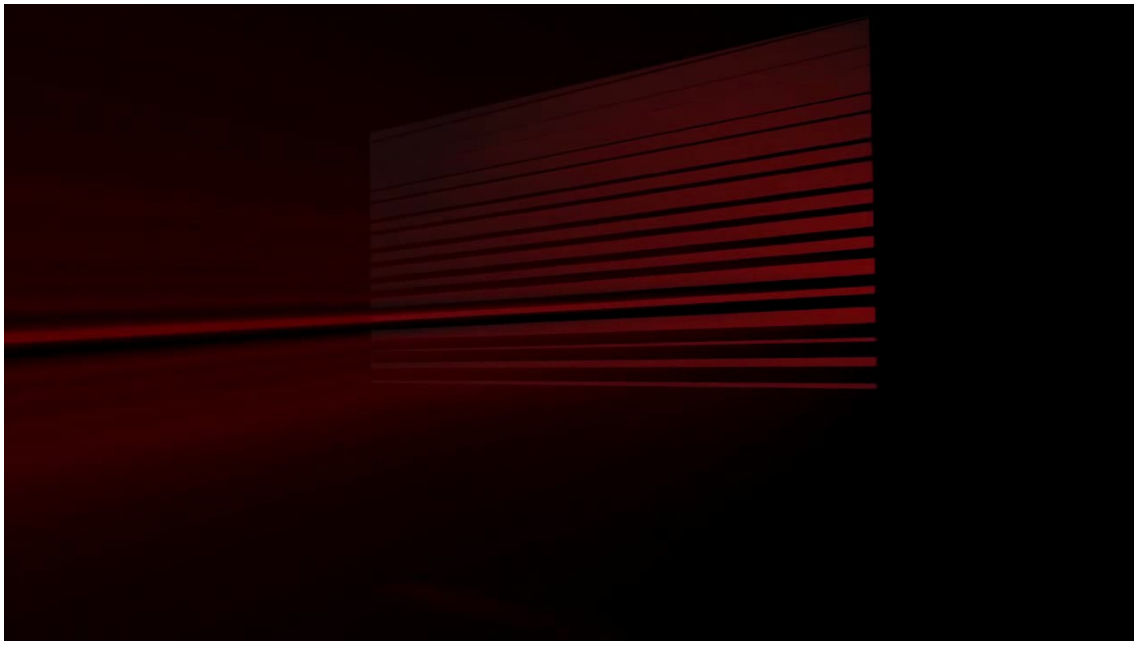
In questo tutorial prenderemo in esame due concetti: l'animazione di una serie di oggetti legati ad un unico elemento di riferimento mediante un *Constraint* (un vincolo) e la corretta illuminazione mediante luce volumetrica (*Halo Light*) in Blender 3D 2.5.

Nella scena è presente una stanza (che può essere modellata, ad esempio, a partire da un Plane mediante estrusioni e cancellazioni, per creare le pareti e lo spazio per la finestra, oppure a partire da un Box mediante suddivisioni e cancellazioni); vi sono, inoltre, una fonte di luce di tipo Spot Light e delle tende alla veneziana con lamelle.



La scena di partenza utilizzata per questo esempio

L'animazione consiste nell'apertura delle lamelle della tenda che lascia quindi entrare la luce volumetrica nella stanza (screenshot dell'animazione qui di seguito).



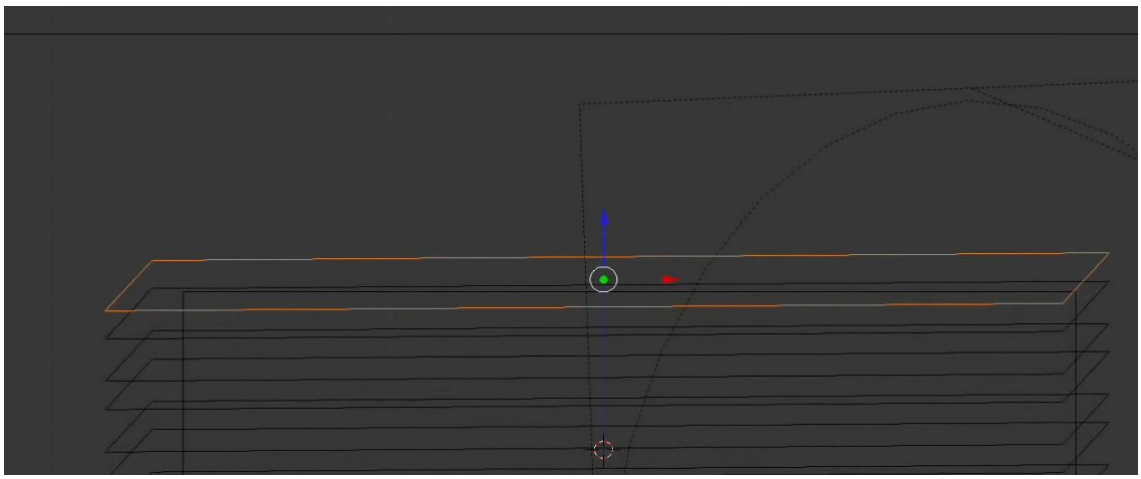
Screenshot dell'animazione delle lamelle

Parliamo per prima cosa dell'animazione delle lamelle; in seguito esamineremo l'illuminazione.

Per poter variare i tempi dell'animazione senza problemi, dopo aver creato i modelli delle varie lamelle, ho deciso di creare la prima lamella, animarla e animare conseguentemente le altre mediante un *Constraint* di tipo *Copy Rotation*, ossia un vincolo che forza un oggetto a seguire l'orientamento di un altro oggetto di riferimento.

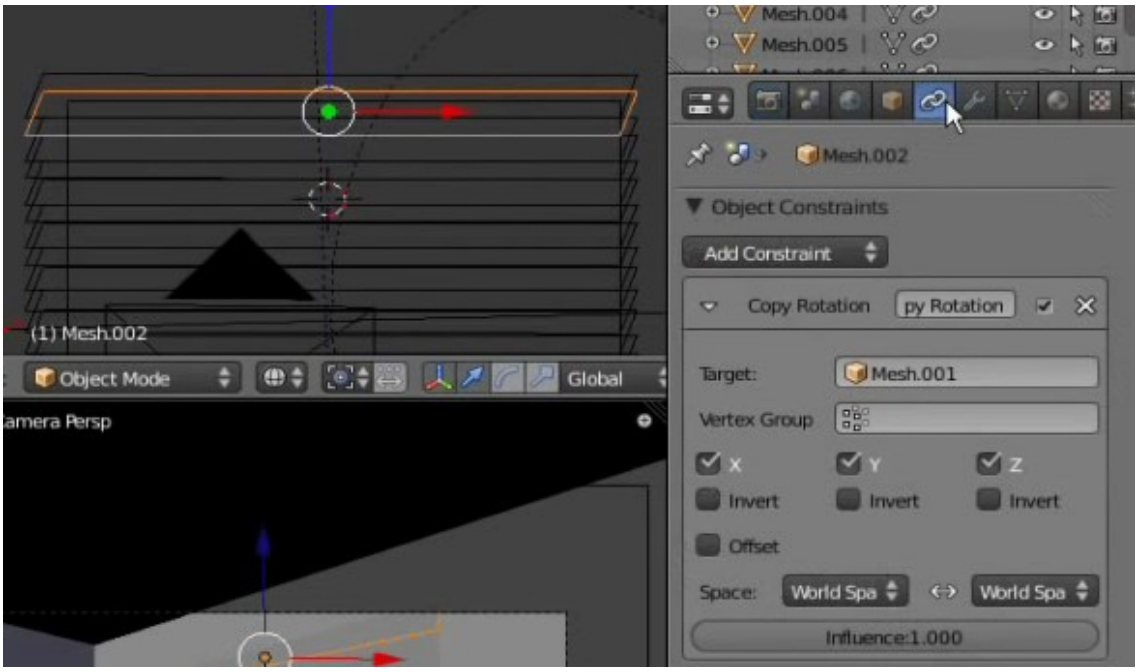
L'attuazione di questo vincolo viene fatta automaticamente da Blender.

Si procede quindi con la modellazione della prima lamella, che nel mio caso è un semplice Plane, per poi creare (con una copia semplice, non linkata, effettuata mediante *SHIFT D* in Object Mode) la seconda lamella.



La modellazione della prima lamella: un semplice Plane, modificato

Si seleziona quindi la seconda lamella e si inserisce per la stessa un *Constraint* di tipo *Copy Rotation*, utilizzando come oggetto di riferimento la prima lamella.

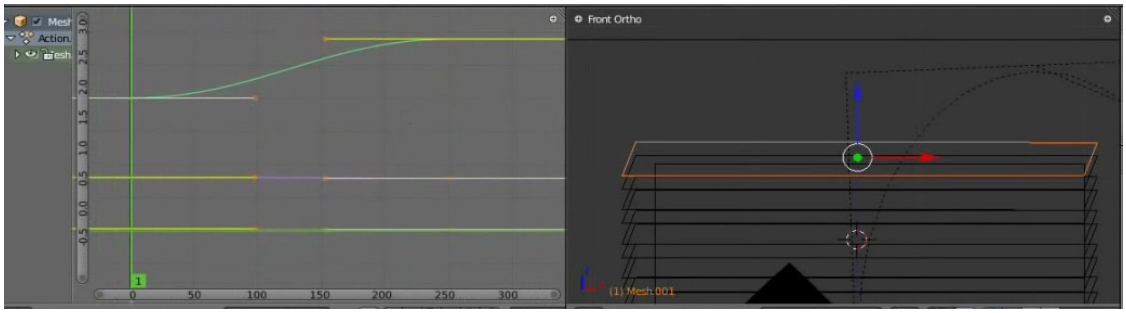


Il Constraint Copy Rotation, per la seconda lamella, con la prima lamella utilizzata come oggetto "TARGET" del vincolo

Segue poi l'operazione, piuttosto noiosa, di creazione delle rimanenti lamelle effettuando copie semplici della seconda lamella; i nuovi oggetti saranno dotati, automaticamente, dello stesso *Constraint* creato per la seconda lamella, quindi ruoteranno in seguito a rotazioni della prima.

Non bisogna utilizzare, per queste operazioni di duplicazione, il modificatore *Array*, altrimenti si avrà la rotazione nello spazio di tutto l'insieme delle copie create considerando la lamella originale come perno.

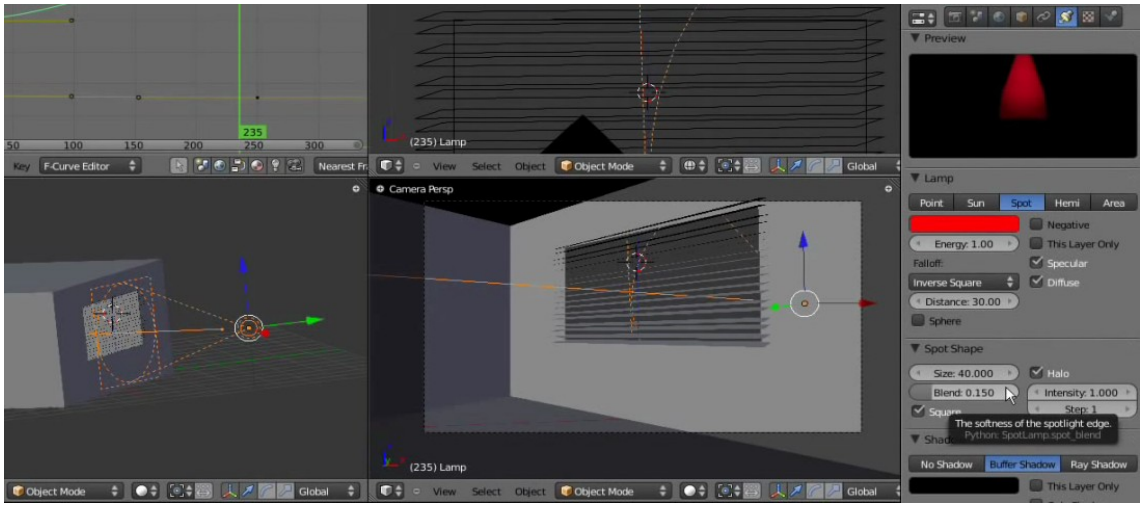
Create e posizionate le copie, si procede all'animazione della prima lamella, semplicemente ruotandola ed inserendo i keyframe a inizio e fine animazione. Le altre lamelle ruoteranno tutte come la prima.



La curva IPO dell'animazione della rotazione della prima lamella

I tempi dell'animazione possono essere regolati in seguito aprendo il *Graph Editor* e, quindi, agendo sulle *curve IPO* della prima lamella.

Passiamo all'illuminazione.
Come anticipato, nella scena vi è una sola fonte di illuminazione, di tipo *Spot Light*, posta al di fuori della stanza e diretta verso la finestra, quindi verso l'interno della stanza.



La fonte di luce Spot Light con effetto HALO presente nella scena

Per ottenere l'effetto di illuminazione visto durante la riproduzione del filmato bisogna ricorrere, ovviamente, alla luce volumetrica, attivando quindi *Halo* nella scheda *Object Data* della fonte di luce, nella *Properties Window*; in questa scheda possiamo impostare parametri come la dimensione del cono luminoso, il colore (rosso, nel mio caso), l'intensità della luce e l'intensità dell'effetto *Halo*.

Halo da solo non basta, perché di default Blender renderizzerebbe il cono luminoso come rosso anche in presenza di oggetti e ostacoli; per ottenere un'illuminazione corretta, quindi, dobbiamo attivare la modalità *Buffer Shadow* per le ombre ed inserire un valore maggiore di 0 per *Step*, nelle caratteristiche dello *Spot Halo*.

La sezione HALO nella scheda Object Data della fonte di luce

In questo modo, il cono di luce terrà conto della presenza degli ostacoli.

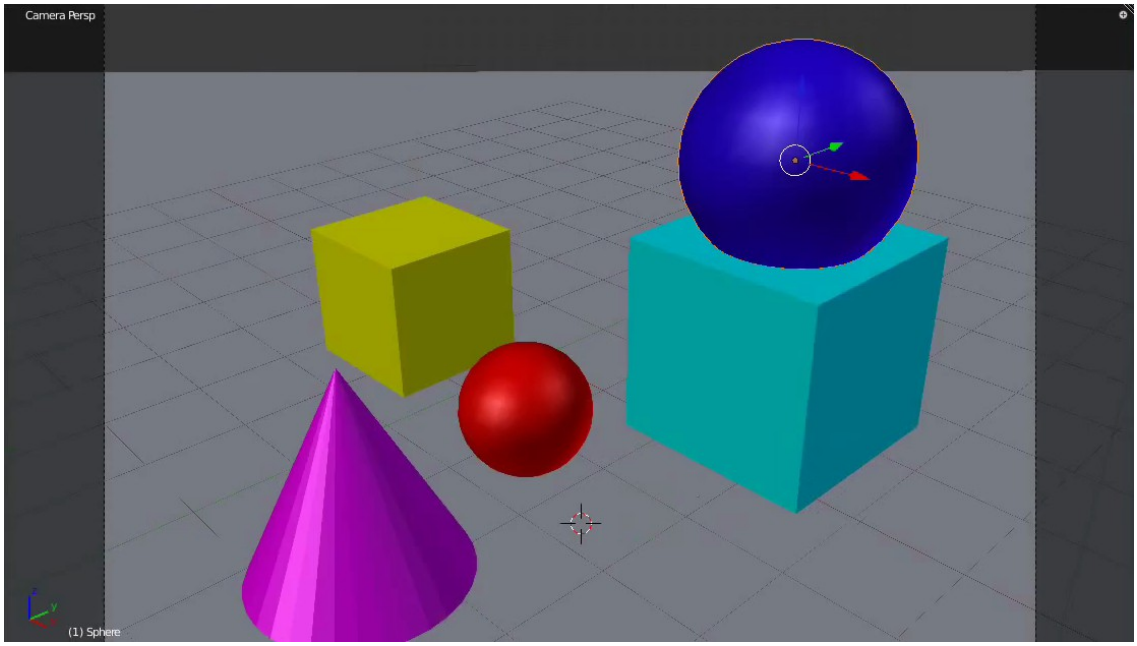
Questo è tutto quello che serve per ottenere l'effetto desiderato.

Buffer Shadow con Step maggiore di 0, per la fonte di luce

* * *

Ombre sfocate con Compositing dei Nodes

Questo tutorial è tratto dal mio *ebook "Il Compositing in Blender 3D 2.5"*, dedicato all'utilizzo dei nodi nella fase di Post-Produzione. In questo tutorial vedremo come sfocare le ombre presenti nei nostri rendering utilizzando i nodi di Compositing.



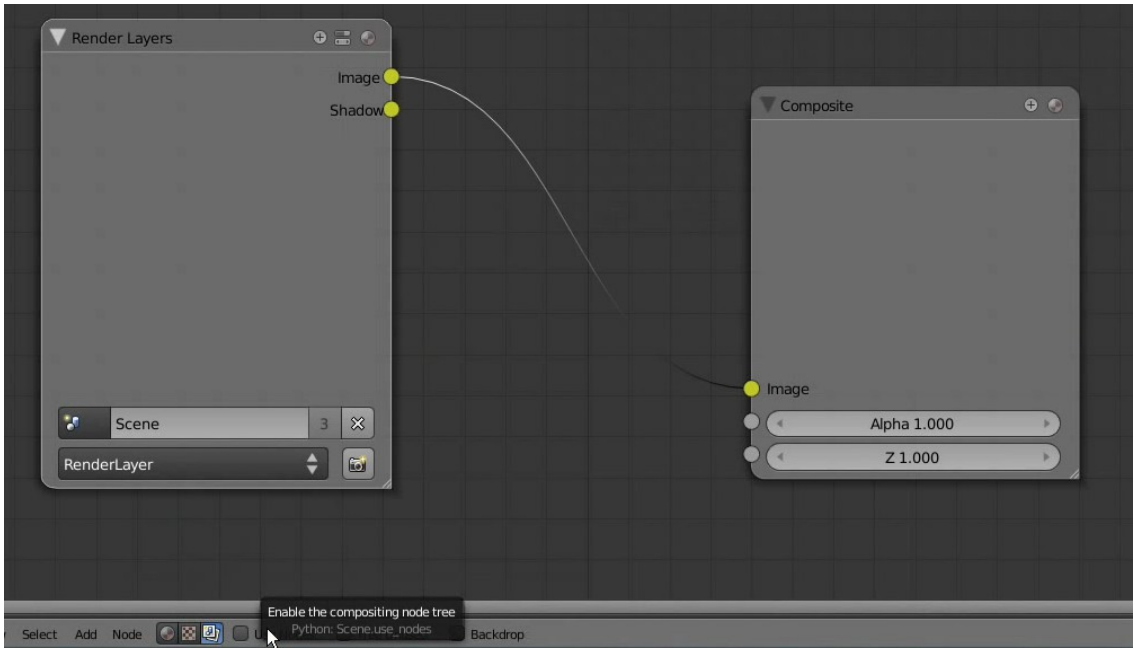
La scena utilizzata in questo tutorial

Con una scena pronta per essere renderizzata apriamo la scheda *Render*, nella *Properties Window*, selezioniamo "*Compositing*" nella sezione Post-Processing e, in *Layers – Passes*, la voce "*Shadows*"; escludiamo inoltre tale canale dal rendering prodotto in uscita da *Render Layers* cliccando sull'icona della macchina fotografica posta accanto a *Shadow*, deselezionandola.

In questo modo non stiamo annullando il rendering delle ombre (che verrà effettuato comunque da Blender), ma lo stiamo escludendo dall'immagine prodotta sul canale Image dal nodo *Render Layers*; tuttavia, potremo accedere a questo canale di informazioni grazie alla porta di output *Shadow* all'interno dello stesso nodo (e abbiamo attivato tale porta proprio selezionando la casella *Shadow* in *Layers Passes*).

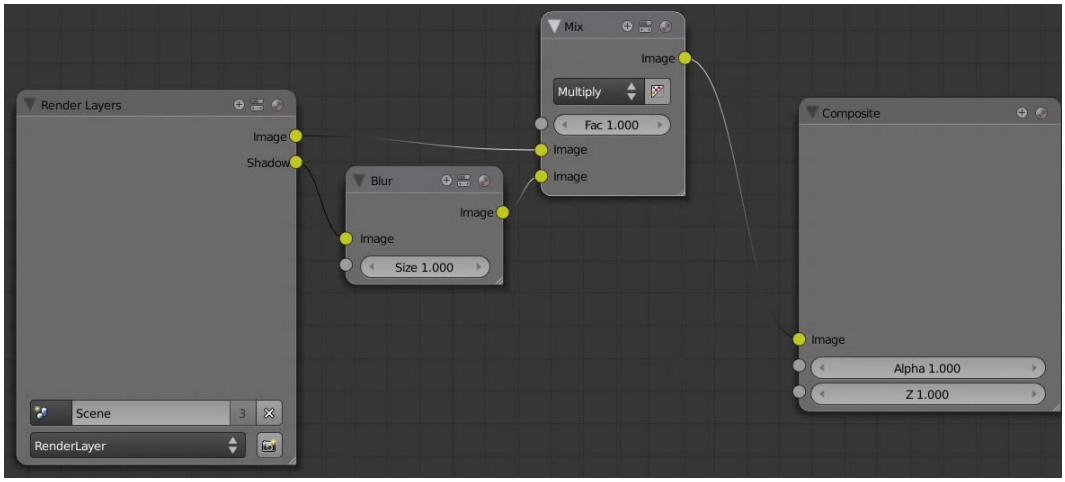
La scheda Render, nella Properties Window

Apriamo ora una finestra di tipo *Editor dei Nodi* e clicchiamo sul pulsante *Compositing Nodes*, selezionandolo; selezioniamo anche il pulsante *Use Nodes*, per attivare l'editing dei nodi per il Compositing.
Per ottenere l'effetto desiderato, poi, iniziamo rimuovendo il collegamento diretto tra *Image* di *Render Layers* e *Image* di *Composite*.



Iniziamo a preparare lo schema nell'Editor dei Nodi

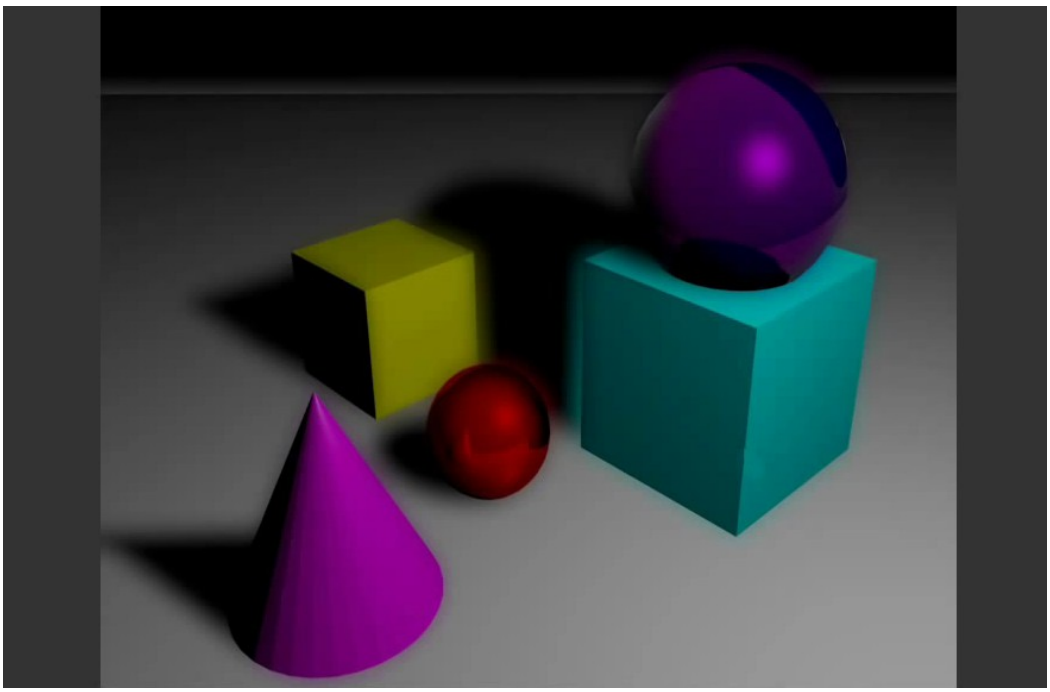
Inseriamo due nuovi nodi: un nodo *Mix* e un nodo *Blur* (la sfocatura).
Colleghiamo il canale di output *Shadow* di *Render Layers* al canale di input *Image* del nodo *Blur* ed impostiamo l'effetto di Blurring come preferiamo; ad esempio, filtro gaussiano con dimensioni X e Y pari a 20. Colleghiamo quindi l'uscita *Image* di *Blur* al secondo canale di input *Image* del nodo *Mix*, collegando poi al primo canale di quest'ultimo nodo l'uscita *Image* di *Render Layers*.



I collegamenti tra i nodi del Nodes Editor

Il nodo *Mix* mescolerà quindi il rendering senza ombre alle ombre sfumate.

Impostiamo *Multiply* (la moltiplicazione) come operazione da effettuare: per aggiungere le ombre ad un'immagine conviene infatti utilizzare la moltiplicazione, visto che il prodotto di due numeri minori o uguali a 1.0 (come sono appunti i valori dei canali-colore dei pixel delle immagini, in Blender) restituisce un valore minore o uguale al minore dei due, cosa che si traduce in un pixel più scuro di quello originale. Possiamo quindi collegare l'output del nodo *Mix* all'input *Image* di *Composite*, avviare il rendering del tutto ed osservare il risultato finale.

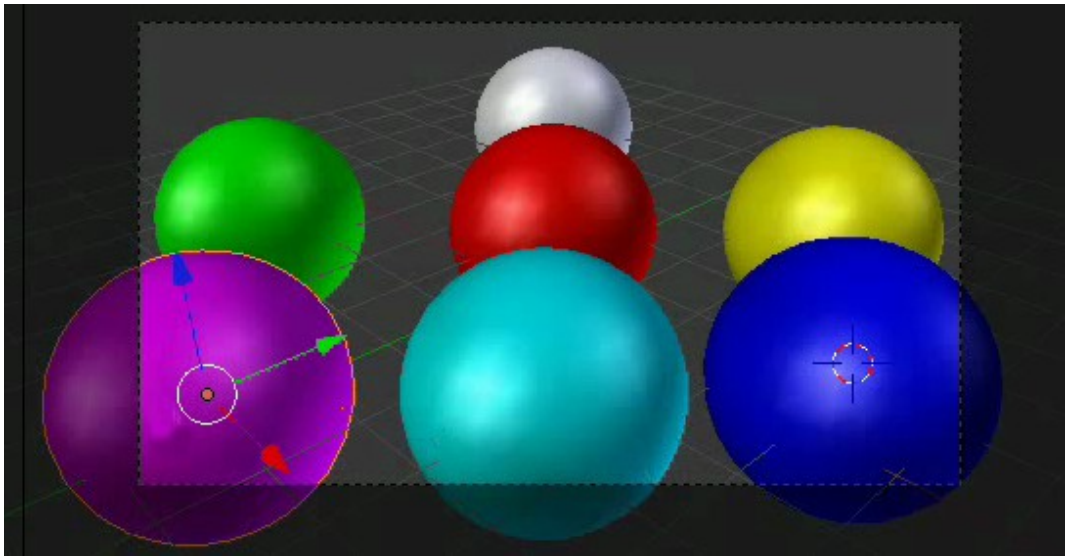


Il rendering finale (dopo il passo di Compositing in Post-Produzione)

* * *

Traslazioni animate delle immagini con Nodes Compositing

Questo tutorial è tratto dal mio *ebook "Il Compositing in Blender 3D 2.5"*, dedicato all'utilizzo dei nodi nella fase di Post Produzione. In questo tutorial vedremo come traslare un'immagine nel corso di un'animazione utilizzando i nodi di Compositing.



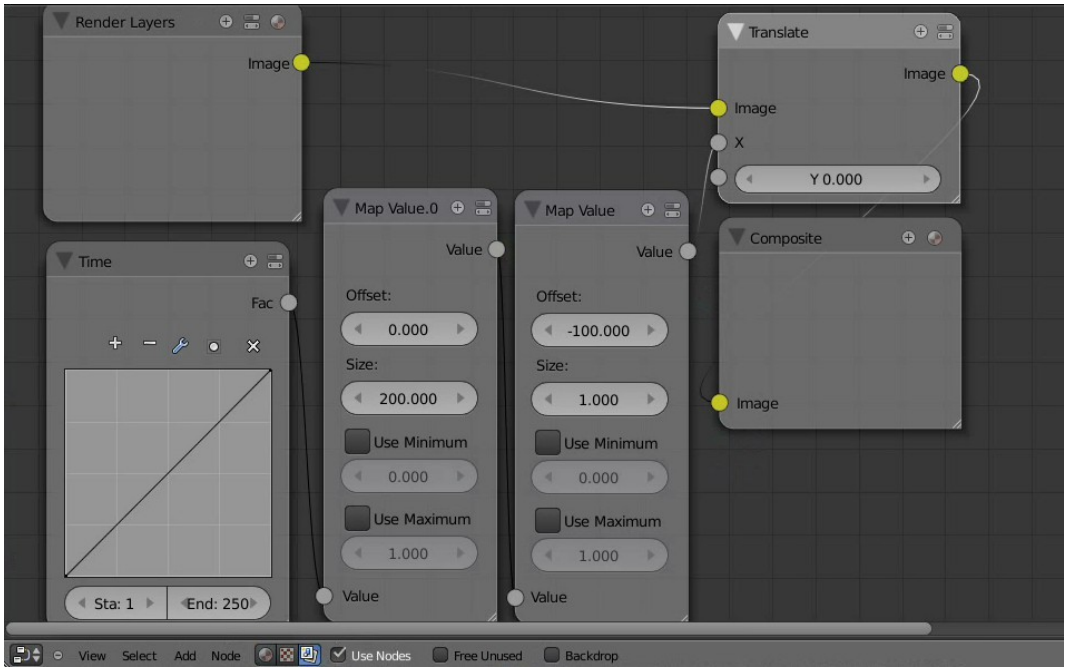
La scena utilizzata in questo tutorial

Apriamo la scheda *Render* nella *Properties Window* e selezioniamo *Compositing* nella sezione *Post Processing*, per abilitare proprio la fase di Compositing finale.

Apriamo quindi una finestra *Nodes Editor* e clicchiamo sul pulsante *Compositing Nodes*, selezionandolo; selezioniamo anche il pulsante *Use Nodes*, per attivare l'editing dei nodi.

Per realizzare l'effetto trattato in questo tutorial ci serve, ovviamente, un'immagine; nel mio caso, sto utilizzando direttamente quella in uscita da *Render Layers*. Il rendering ottenuto dal passo di Compositing coprirà completamente quello in arrivo da *Render Layers*.

Per ottenere l'effetto desiderato rimuoviamo il collegamento diretto tra *Image* di *Render Layers* e *Image* di *Composite*, inserendo poi quattro nuovi nodi: un nodo *Time*, due nodi *Map Value* e un nodo *Translate*.



Lo schema di Nodi da implementare per realizzare l'effetto

I quattro nodi vanno quindi collegati in cascata, collegando in particolare l'uscita *Value* del secondo nodo *Map Value* alla porta di ingresso *X* del nodo *Translate*.
L'immagine verrà quindi traslata orizzontalmente.

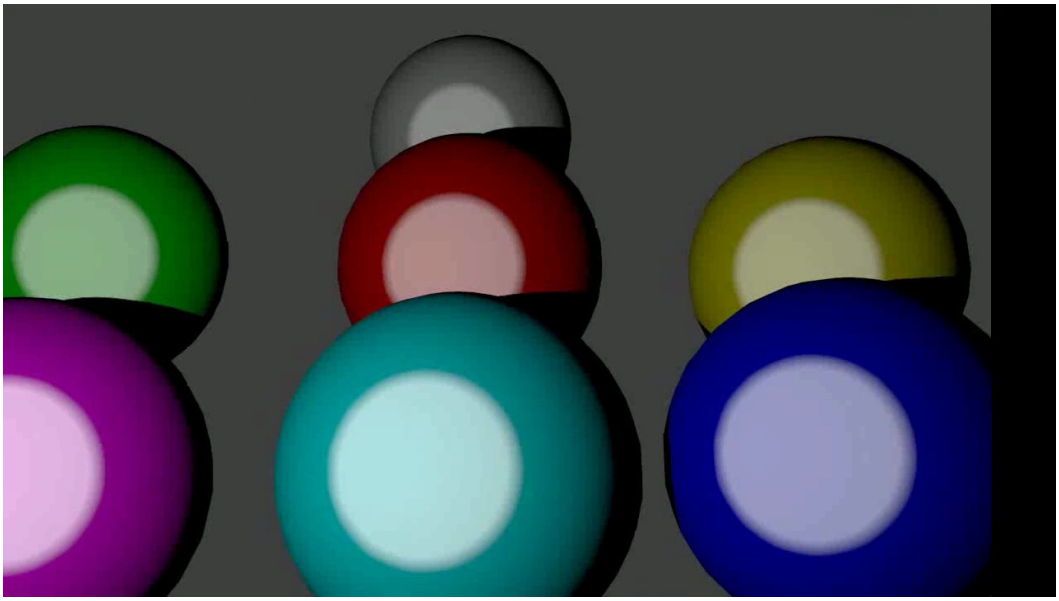
Il nodo *Time* fornisce in output (nel corso dell'animazione di range specificato dai campi *Start* ed *End* del nodo stesso) un numero in virgola mobile tra 0.0 e 1.0, con tale valore che dipende dal valore *Y* della curva presente nel nodo *Time*.

Tale curva può essere ovviamente modificata agendo sui punti di controllo ed inserendone, se necessario, di nuovi; per il nostro scopo, comunque, va benissimo la diagonale di default.

Per il primo nodo *Map Value* impostiamo *Offset* a 0.0 e *Size* a 200.0, mentre per il secondo nodo *Map Value* impostiamo *Offset* a -100.0 e *Size* a 1.0. In questo modo il valore numerico in arrivo da *Time*, che originariamente variava nel tempo da 0.0 a 1.0, adesso varierà da -100.0 a 100.0.

Tale valore verrà inviato, come detto, al canale di input *X* di *Translate*, mentre alla porta di input *Image* dello stesso nodo collegheremo la porta *Image* di *Render Layers*, per cui l'immagine in arrivo da *Render Layers* "scorrerà" da -100 a +100 pixel rispetto all'inquadratura del rendering finale.

Possiamo quindi collegare l'output *Image* di *Translate* all'input *Image* di *Composite* ed avviare il rendering dell'animazione, per visualizzare l'effetto ottenuto.



Screenshot di uno dei primi frame dell'animazione finale; notare lo spazio nero a destra, dovuto al fatto che il rendering è spostato rispetto all'inquadratura del filmato finale (per via di Translate) e, quindi, Blender ha riempito col nero la parte "vuota"

Con pochi accorgimenti è possibile ottenere altri effetti, come ad esempio far muovere un'immagine (magari provvista di trasparenza, come un logo o un testo) su un'immagine di sfondo, utilizzando in quel caso un nodo *Alpha Over*.

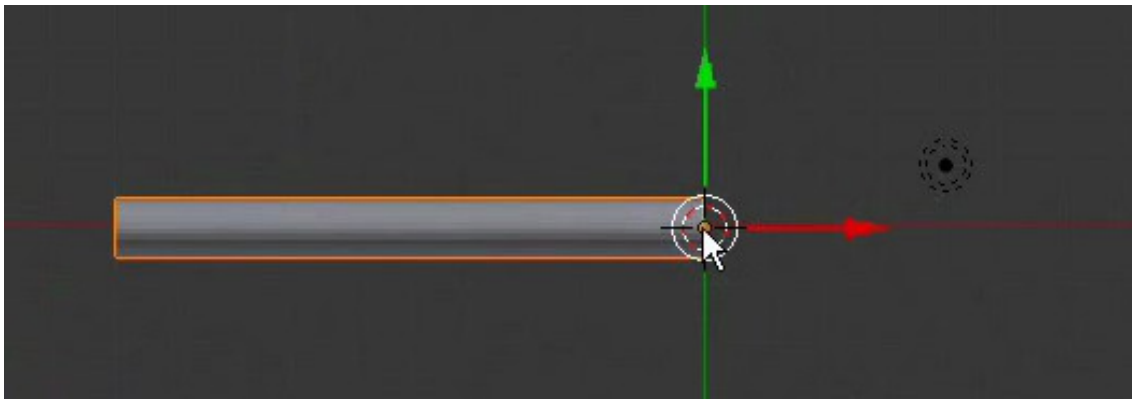
* * *

Animazione ruota panoramica con Parent e Constraints

In questo tutorial vedremo un esempio di utilizzo delle relazioni *Parent* e del *Constraint Limit Rotation*; la prima applicazione che mi viene in mente per l'effetto che vedremo è quella dell'animazione della ruota panoramica, anche se probabilmente questo non è l'unico modo per ottenere quell'effetto e, in ogni caso, qui lavorerò con primitive di Blender, non con modelli complessi, giusto per concentrarci sulla tecnica.

Nel caso della ruota panoramica, la ruota gira intorno ad un asse di rotazione passante per il centro, spostando, nel mentre, le cabine, che però devono restare dritte, a meno di non voler far precipitare nel vuoto i passeggeri durante un giro.

Nella scena di Blender, in una finestra con vista *TOP*, ho inserito un cilindro posto orizzontalmente sul piano della *3D View*, allungandolo un po': questo è uno dei bracci della giostra.

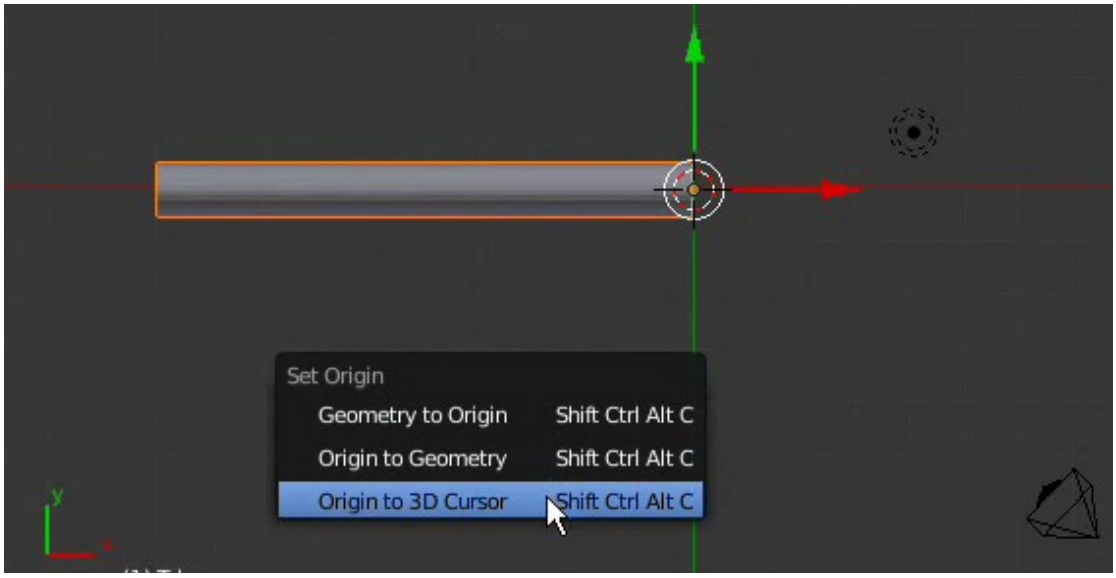


Il primo cilindro / braccio della giostra, da una vista TOP

In questo tutorial, per effettuare correttamente traslazioni e rotazioni, avranno notevole importanza le posizioni dei perni di trasformazione degli oggetti.

Posizioniamo il perno del cilindro al centro della ruota panoramica, centro che coincide (in questo modello estremamente semplificato) con l'estremità destra del cilindro e col centro della scena; per far ciò, possiamo posizionare il cursore 3D nella giusta posizione, magari con il pannello delle trasformazioni parametriche, selezionare il cilindro, premere *CTRL SHIFT ALT C* e scegliere, dal menù che apparirà a video, *Center Origin To 3D Cursor*; dovremo inoltre scegliere, dal menù *Pivot Point* presente nell'header della finestra di navigazione 3D, la voce *3D Cursor*.

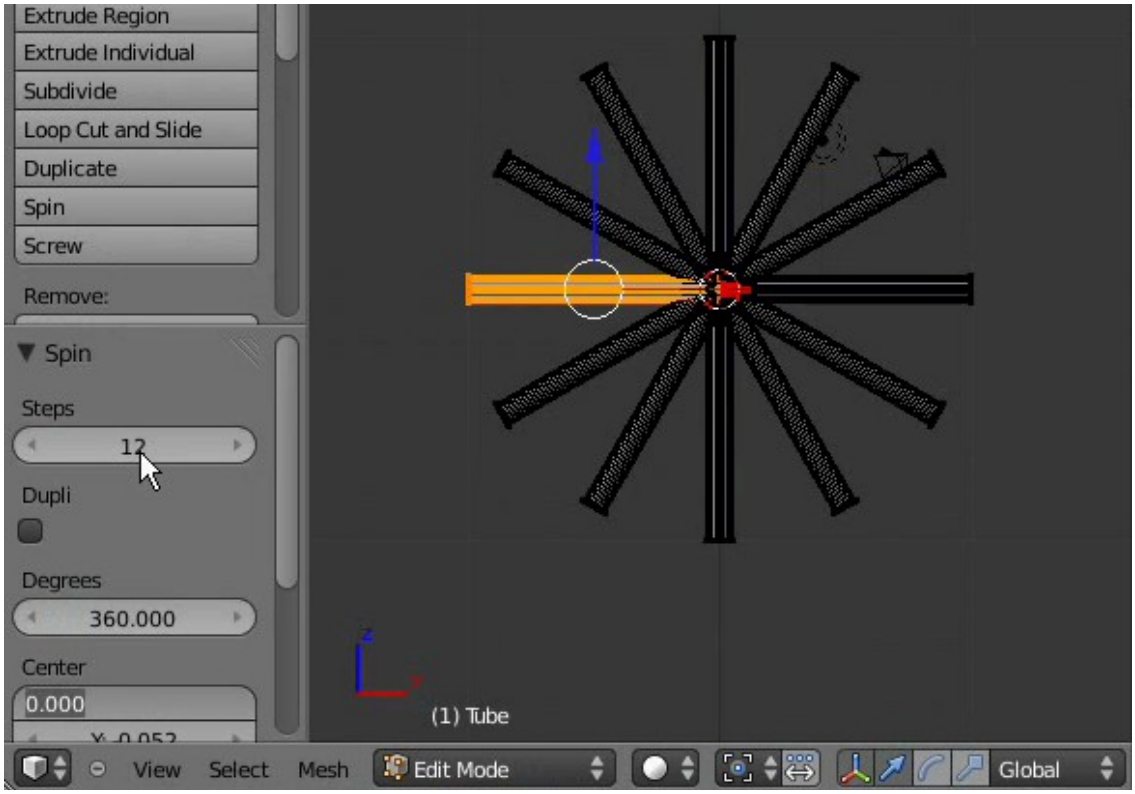
[NOTA: possiamo impostare i valori di posizione del *3D Cursor* mediante le relative voci in basso nella *Transform Window*, richiamabile con lo shortcut *N* in ogni finestra di navigazione 3D]



Set Origin to 3D Cursor

Per ottenere delle copie e, quindi, gli altri bracci della giostra, disponendoli intorno al perno di trasformazione del primo braccio (ossia il centro della giostra), utilizziamo *Spin*: passiamo quindi in Edit Mode, selezioniamo tutti i vertici del cilindro, scegliamo *Spin* dalla *Tool Shelf* della finestra di navigazione *3D View*, nella vista frontale, ed impostiamo 360 gradi e, ad esempio, 12 passi.

Se abbiamo impostato bene il perno dell'oggetto, la rotazione avverrà intorno al centro della ruota. *Spin* va applicato nella finestra con vista frontale per effettuare la trasformazione intorno all'asse di rotazione corretto, che è sempre quello uscente, in maniera perpendicolare, dalla *3D View* utilizzata.



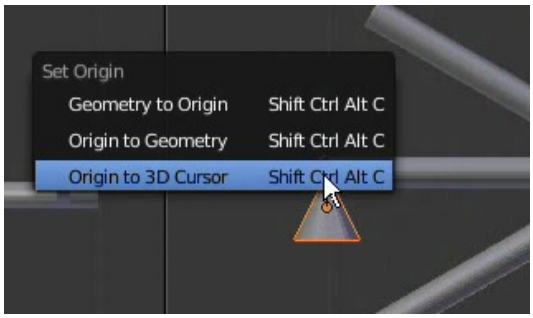
Utilizzo di Spin, in Edit Mode e, questa volta, inquadrando l'oggetto da una vista frontale ortogonale

Adesso passiamo all'oggetto da agganciare al cilindro.

Per osservare l'effetto sull'orientamento dell'oggetto, inseriamo ad esempio un Cono.

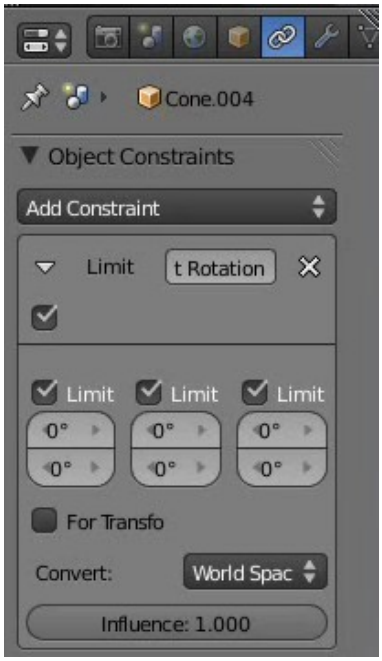
Anche in questo caso, dobbiamo impostare il perno delle trasformazioni in un punto diverso dal baricentro, ed in particolare nella punta del cono, aiutandoci magari con il cursore 3D, come visto precedentemente per il cilindro.

Spostiamo quindi il Cono in modo da posizionarne la punta in corrispondenza dell'estremità sinistra del cilindro originale, posizioniamo in quel punto il cursore 3D ed impostiamo tale punto come perno della trasformazione del Cono.



Il posizionamento dell'Origin nel vertice superiore del cono, posizionato correttamente rispetto ad un braccio della giostra

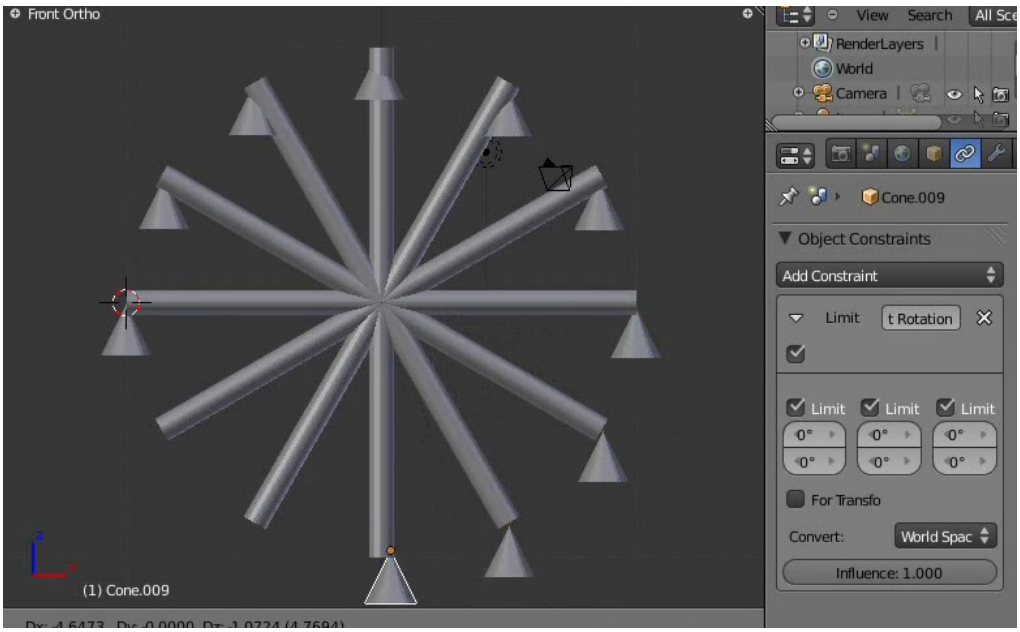
Prima di effettuare le copie del Cono, predisponiamolo per il lavoro che dovrà svolgere. Creiamo una relazione di parentela, rendendo il Cono figlio della ruota, per cui selezioniamo prima il Cono, poi con *SHIFT* e click del tasto destro del mouse, per effettuare una selezione multipla, facendo attenzione all'ordine) selezioniamo la ruota e premiamo *CTRL P*, confermando l'operazione *Set Parent To Object*. A questo punto, dal pannello *Constraints* della *Properties Window*, inseriamo un *Constraint* di tipo *Limit Rotation* per il solo Cono, selezionando tutte e tre le caselle *Limit X*, *Y* e *Z* e lasciando i valori di default nei relativi campi.



Il Constraint Limit Rotation per il cono, che è stato imparentato al cilindro (la giostra vera e propria)

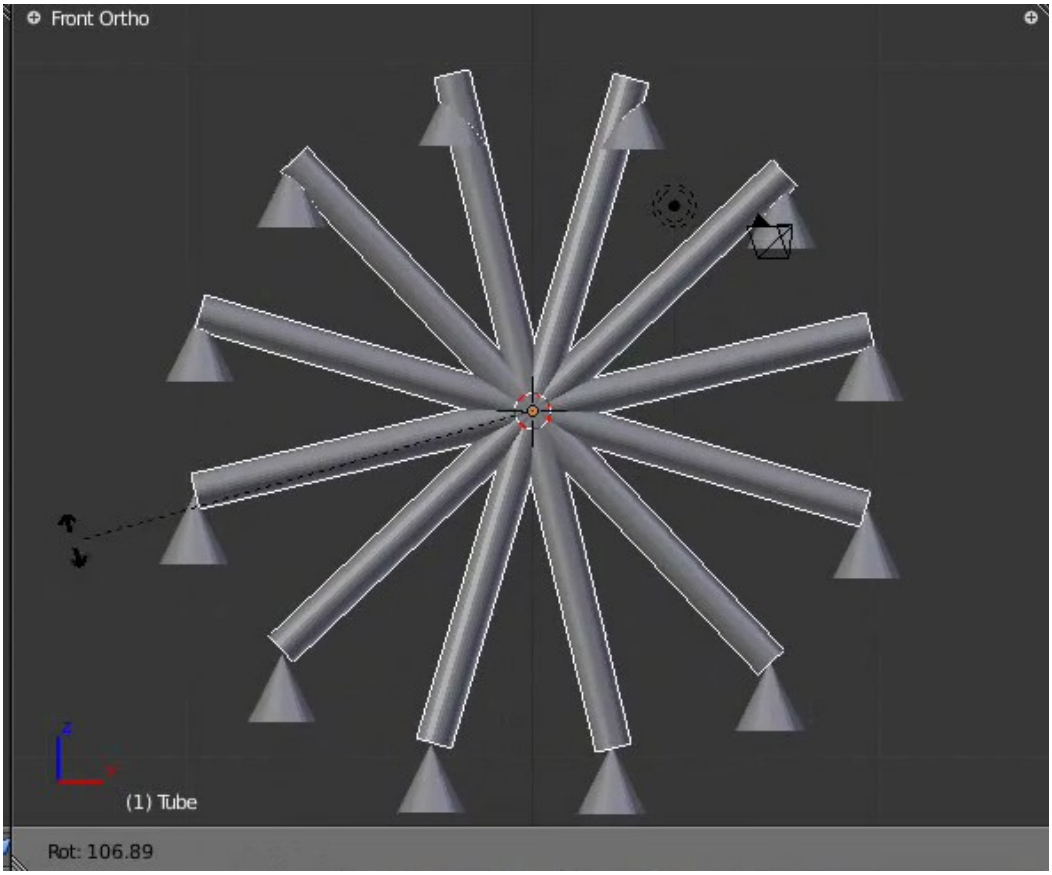
Adesso dobbiamo effettuare le copie dell'oggetto cabina, operazione che va effettuata manualmente a colpi di *SHIFT D* per duplicare il Cono per poi trascinarlo e posizionarlo correttamente, anche se per posizionarlo possiamo affidarci a trucchetti come le trasformazioni parametriche o con *Snap*.

Non possiamo utilizzare, per questa operazione, *Spin*, altrimenti avremo delle copie ruotate, né possiamo unire alla fine i vari coni in un'unica mesh, o avremo altri effetti indesiderati; come anticipato, questo non è l'unico metodo per ottenere una giostra, ma con questo metodo dobbiamo effettuare queste operazioni.



Realizzazione e posizionamento delle copie del cono, in Object Mode

Le copie dei vari coni erediteranno da quello originale la relazione di parentela con la ruota e il *Constraint Limit Rotation*, per cui alla fine, con il *3D Cursor* posizionato al centro della ruota, potremo ruotare la stessa per osservare l'effetto ottenuto: mentre la giostra gira, spostando i coni, questi restano dritti, proprio come vogliamo.



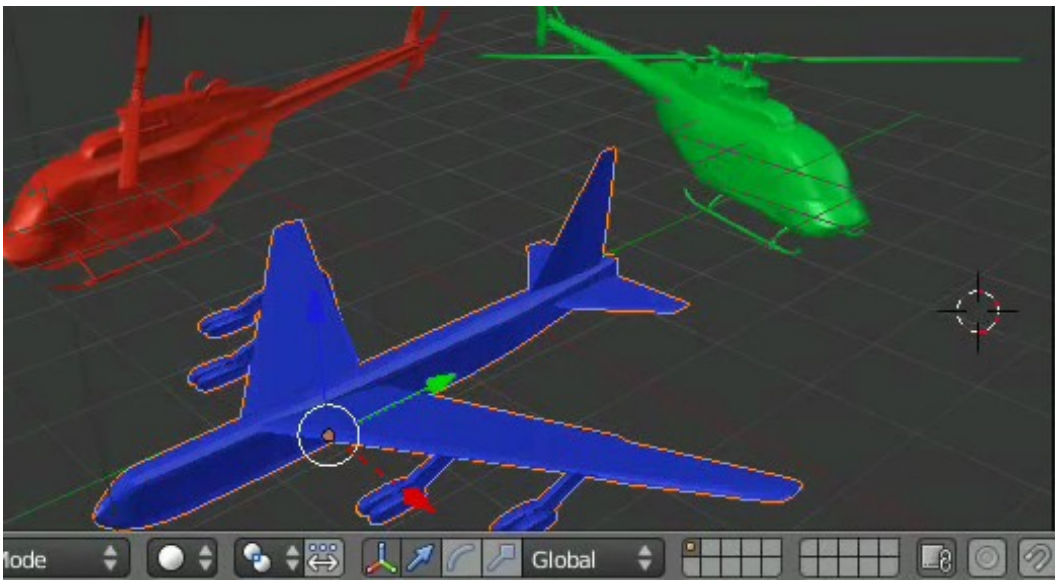
Screenshot del risultato finale (mentre si effettua una rotazione dei bracci della ruota)

* * *

Animazione Glare selettivo in Compositing

Questo tutorial, realizzato con la versione 2.55 Beta di Blender 3D, nasce dal voler creare un'animazione un po' particolare, dove vengono mostrati alcuni oggetti in *Wireframe*, la modalità struttura (detta anche "fil di ferro") e dove solo uno di questi oggetti mostra, ad un certo punto, un bagliore *Glare*, in modo da distinguerlo dagli altri. Per questo esempio sto usando i modelli 3D aerei ed elicotteri, ma potremmo utilizzare questa tecnica per rendering architettonici, visualizzazioni di parti meccaniche ed altro ancora.

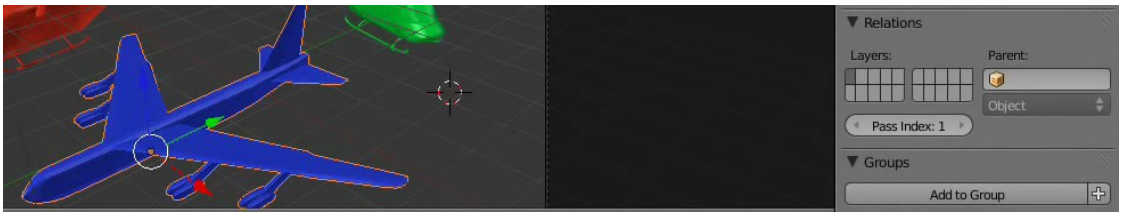
Il tutto si basa principalmente sul *Glare* selettivo mediante *OB Index* in *Compositing*, una tecnica descritta nel mio ebook "*Il compositing in Blender 2.5*", solo che in questo caso sto utilizzando anche un *Nodo Time* per limitare ad un certo range di Frames dell'animazione il *Glare* selettivo; in realtà, l'effetto riguarda più la variazione del mixaggio finale, non l'intensità del bagliore, come vedremo. Nella scena sono quindi presenti alcuni modelli 3D, tutti provvisti di un loro *Material Wire*, in modo da renderizzare solo la struttura degli stessi, non le facce. Assegniamo a ciascun Material, per il relativo oggetto, un colore diffuso proprio.



La scena utilizzata per questo tutorial: abbiamo delle mesh dotate di Material Wireframe con colori diversi

Impostiamo il colore *Horizon Color* del mondo, ad esempio, a nero, e togliamo tutte le luci; i vari Materials avranno infatti *Emit* a 0.2, cosa molto utile per dare un'illuminazione propria, uniforme, a tutti gli spigoli delle mesh.

Adesso, per attivare il *Glare* selettivo, dobbiamo aprire la scheda *Object*, nella *Properties Window*, per l'oggetto che ci interessa ed impostare, nella sezione *Relations*, un valore di *Pass Index* diverso da quello di default, che è 0, mettendo, ad esempio, 1.



Impostare il Pass Index per l'oggetto al quale vogliamo fornire un effetto Glare

Dobbiamo inoltre aprire la scheda *Render*, sempre nella *Properties Window*, ed attivare, nella sezione *Layers*, la voce *Object Index*.

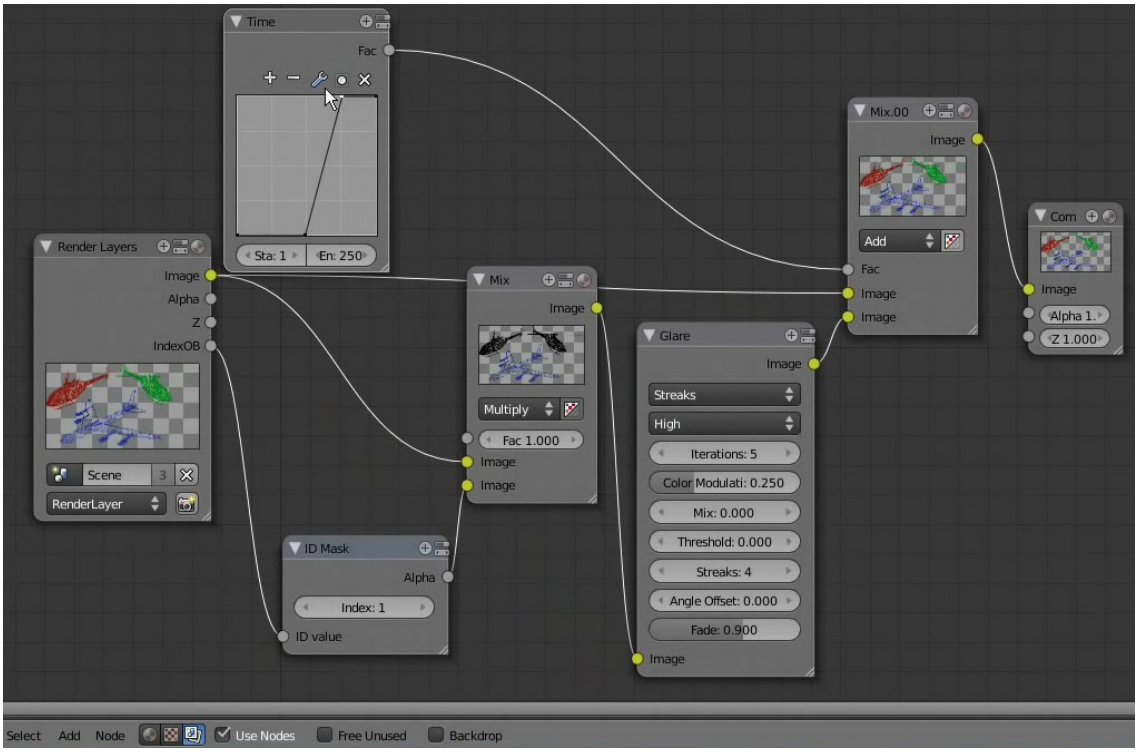
Attivare la casella Object Index nella scheda Layers

Adesso spostiamoci nella scheda del *Compositing*, attivando *Use Nodes* per *Compositing Nodes*.

Eliminiamo il collegamento tra *Render Layers* e *Composite*.

Aggiungiamo al diagramma un *Nodo ID Mask*, del gruppo *Convertor*, e colleghiamo *IndexOB* di *Render Layers* all'input del *Nodo ID Mask*, impostando tra l'altro il valore 1 per il campo *Index* di quest'ultimo.

Aggiungiamo ora un *Nodo Mix* del gruppo *Color*, collegandogli in *Input* l'uscita *Image* di *Render Layer* sul primo canale e l'uscita *Alpha* del Nodo *ID Mask* sul secondo canale; cambiamo il tipo di mixaggio in *Multiply*, con valore *Factor* 1.



La finestra Compositing Nodes, con Use Nodes attivo e il collegamento tra i due nodi iniziali rimosso (cliccare sulla "porta di arrivo" del collegamento col tasto sinistro e, tenendo premuto tale tasto del mouse, rilasciare il collegamento in uno spazio vuoto della finestra)

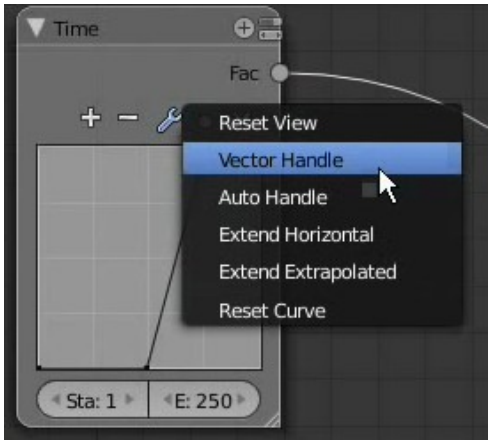
Adesso aggiungiamo il Nodo *Glare*, del gruppo *Filter*, al diagramma, impostando i parametri del Nodo come preferiamo, ad esempio *Streaks* in alta qualità, 5 iterazioni e soglia a 0.

Colleghiamo l'uscita *Image* del Nodo *Mix Multiply* all'input *Image* di *Glare*.

A questo punto dobbiamo effettuare la fusione tra l'immagine originale e quella con *Glare* applicato in maniera selettiva solo all'oggetto con *Index* a 1, per cui aggiungiamo un altro Nodo *Mix* del gruppo *Color* al diagramma e colleghiamogli in input l'uscita *Image* di *Render Layers* e l'uscita *Image* di *Glare*; cambiamo infine la modalità di filtraggio in *Add*, la somma.

Adesso però dobbiamo limitare l'effetto solo ad una parte dell'animazione (che nel mio caso è quella standard di Blender, di 250 frames), ad esempio attivandolo solo a partire dal frame 125 con intensità via via crescente fino al frame 190.

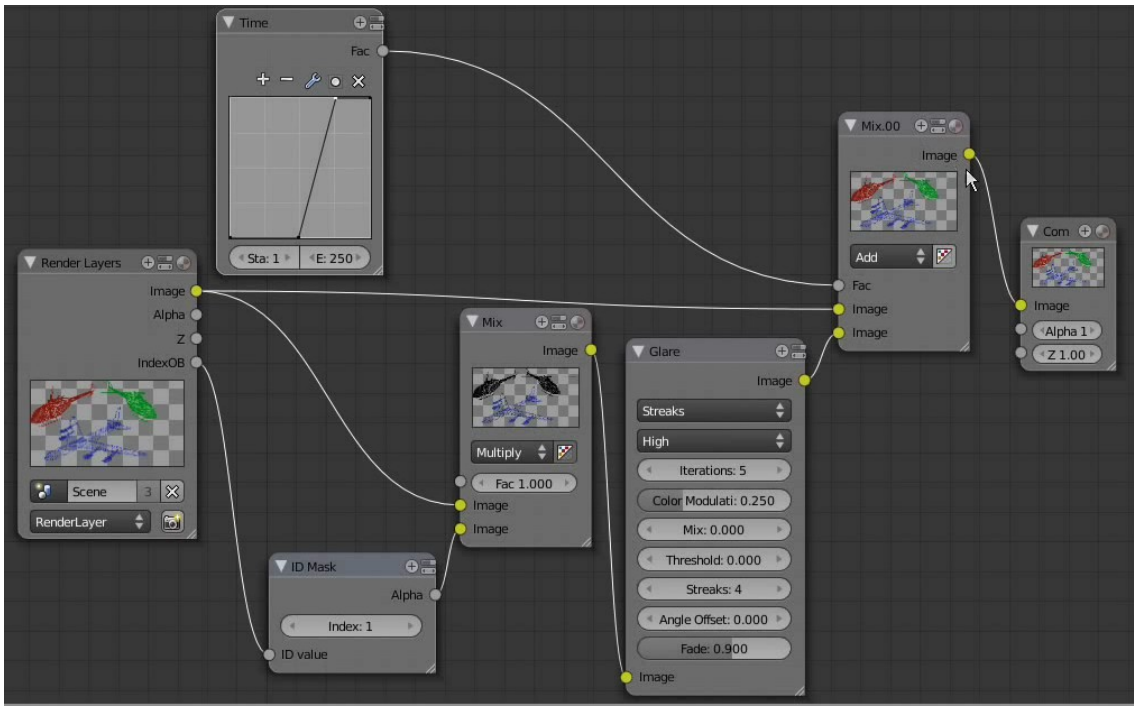
Per far ciò, aggiungiamo un Nodo *Time*, del gruppo *Input* al diagramma, modificando la curva in modo da farla andare da 0 a 1 a partire dal frame 125 dell'animazione fino al frame 190, lasciandola piatta, a 0, nella prima parte, e a 1 nell'ultimo tratto. Per rendere lineare la trasformazione, apriamo il menù degli strumenti per i due punti di controllo creati nel Nodo *Time* e scegliamo la voce *Vector Handle* (la maniglia vettoriale).



La curva del Nodo Time

Colleghiamo l'uscita *Fac* di tale Nodo all'input *Factor* dell'ultimo Nodo *Mix*, quello in modalità *Add* che univa l'immagine originale e l'immagine con *Glare*.

Possiamo quindi collegare l'uscita *Image* di tale Nodo *Mix* all'ingresso *Image* del Nodo *Composite* ed avviare il rendering dell'animazione.

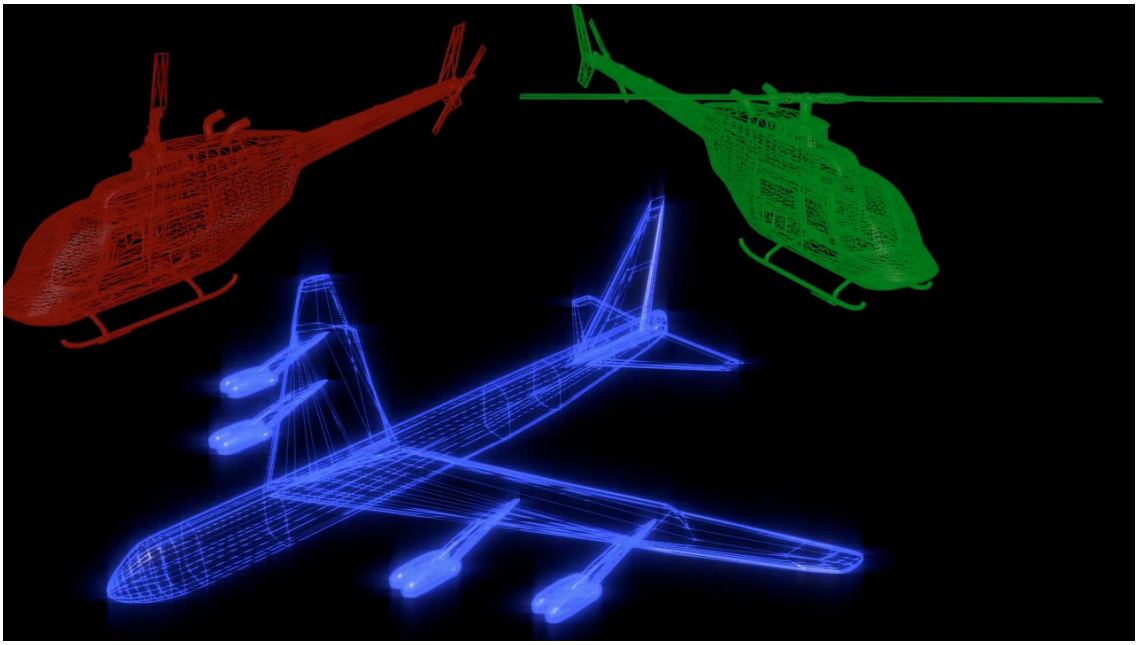


Lo schema di Nodi finale

Tra le altre cose, qui sto facendo variare *Factor* di *Mix Add* da 0 a 1, ma possiamo farlo variare da 0 a 5 utilizzando ad esempio un Nodo *Map Value* (altra tecnica descritta nell'ebook) per accentuare ulteriormente l'effetto del bagliore *Glare*. Gli screenshots che seguono sono tratti da un rendering effettuato proprio con *Map Value* e la variazione da 0 a 5 di *Factor*.

Il Post-Processing con Compositing dei Nodi viene effettuato da Blender solo se la casella Compositing, nella scheda Render all'interno della Properties Window, è selezionata, ma questo avviene di default in ogni nuova scena di Blender 2.5.

Questo è solo un esempio, ovviamente possiamo animare i modelli oppure far variare in maniera diversa il valore del bagliore *Glare* nel corso dell'animazione, magari limitandolo ad un altro range di frames o facendone oscillare l'intensità, ed altro ancora; per questo tutorial, invece, è tutto.



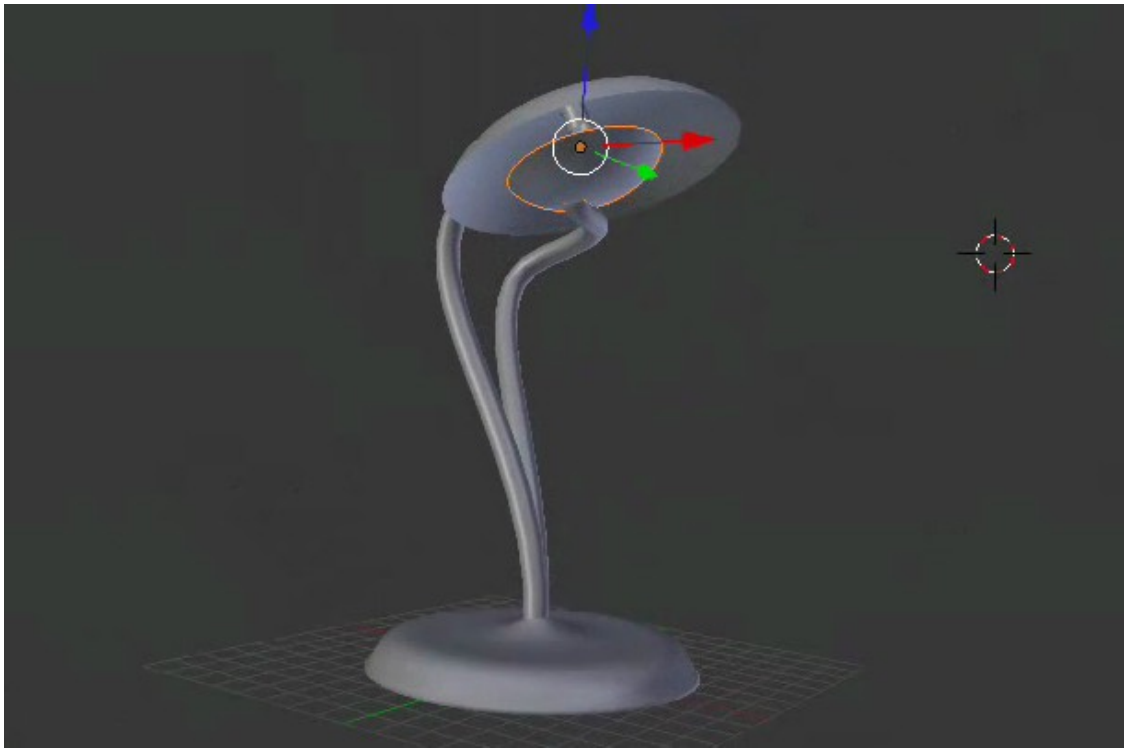
Screenshot del risultato finale

* * *

Modellazione lampada Liberty con Spin, Bevel, Material vetro satinato

In questo tutorial vedremo come modellare una lampada in stile Liberty, trattando argomenti come l'estrusione di un profilo lungo un percorso, l'utilizzo di *Spin* per realizzare solidi di rotazione a partire da una sezione di un oggetto ed altro ancora.

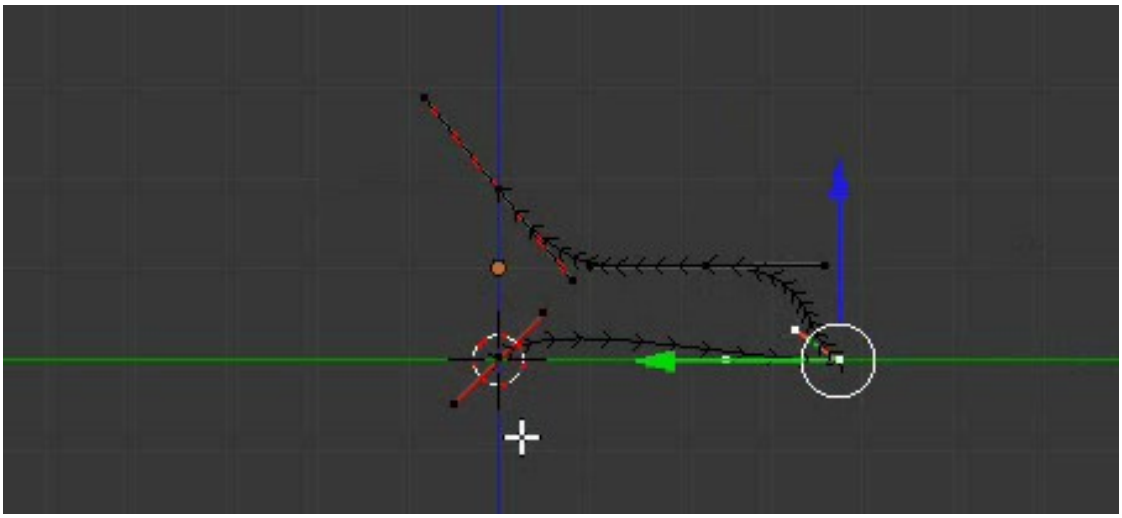
L'oggetto che vogliamo realizzare è quello visibile nell'immagine in basso: come potete vedere, è composto da una base, una struttura curva di collegamento e due piatti che nascondono la lampadina (lampadina che comunque non realizzeremo in questo tutorial, perché non ci serve).



L'oggetto che vogliamo realizzare

Apriamo un nuovo file di Blender e cancelliamo tutti gli elementi presenti.
Iniziamo con la modellazione della base della nostra lampada.
Per modellare questo oggetto, inseriamo nella scena una curva *Bezier* di base.

Utilizzeremo questa curva come profilo della base, per poi ricorrere a *Spin* per ottenere il solido di rotazione. Quello che dobbiamo fare, quindi, è definire il profilo, diciamo un trecentosessantesimo della base, agendo sulle maniglie della curva, eventualmente estrudendo a partire da un punto di controllo per prolungare la curva creando una nuova maniglia o suddividendo un segmento in due o più parti selezionandone le estremità e cliccando su *Subdivide*.



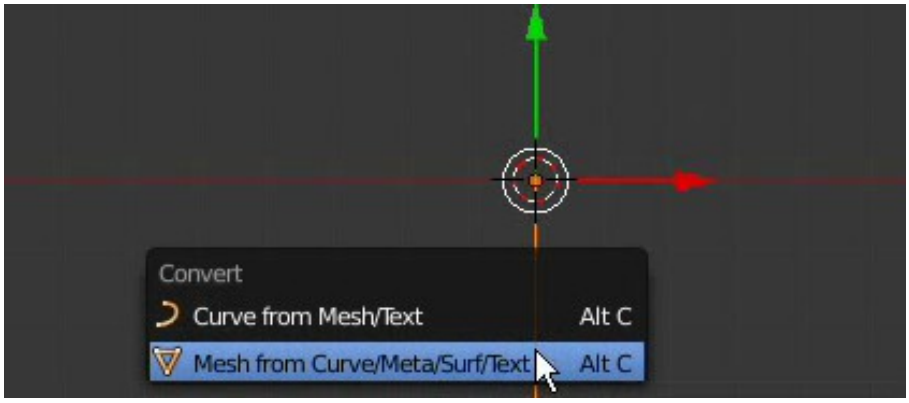
Modellazione del profilo della base con una Bezier Curve

Selezionando un punto di controllo e premendo *V* potremo poi cambiare il tipo di maniglia; per la base consiglio di utilizzare *Vettoriale*, per creare spigoli, traslando, ruotando e scalando come preferiamo le maniglie.

Possono tornare utili anche le voci presenti nella *Transform Window*, richiamabile con *N*, per specificare con precisione le coordinate di ciascun punto di controllo, facendo attenzione a selezionare la modalità *Global*. Non è necessario chiudere il profilo.

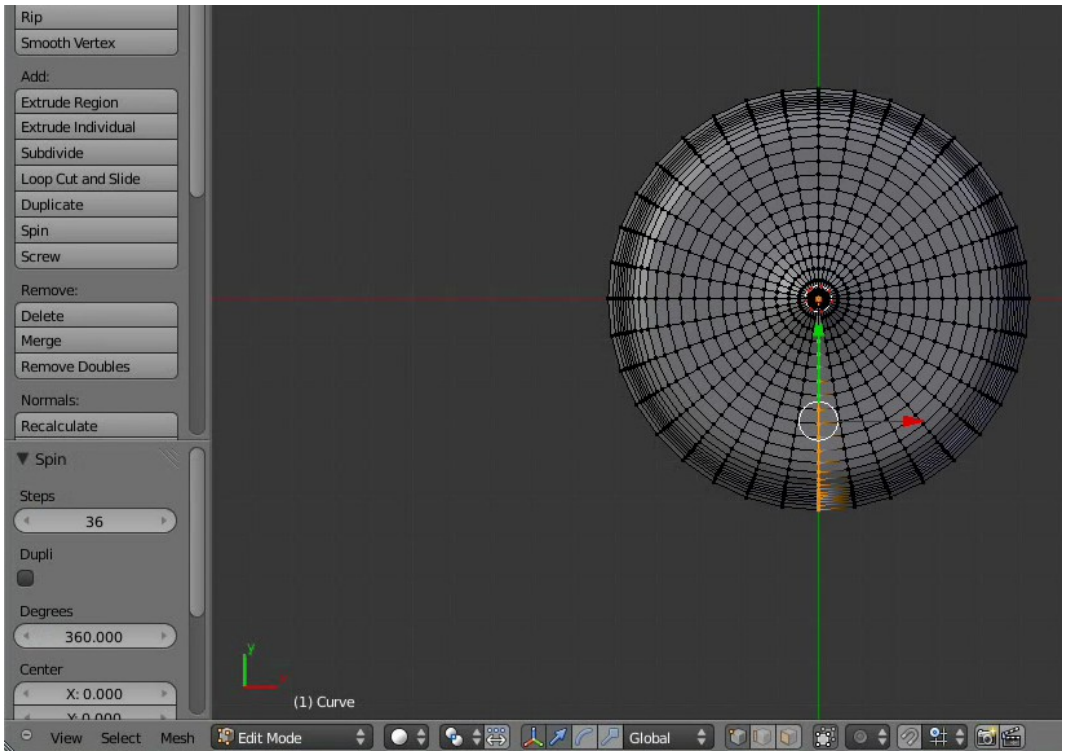
Una volta definito il profilo, centriamo il *3D Cursor* (eventualmente con le voci *3D Cursor Location* nella *Transform Window*) nel punto che farà da centro di rotazione e che coincide con il centro della base della lampada, quindi spostiamoci nella vista *TOP*, in modo da usare come asse di rotazione quello verticale, perpendicolare al piano XY.

Per poter utilizzare *Spin* dobbiamo convertire la curva così definita in mesh, per cui in Object Mode premiamo *ALT C* e scegliamo *Mesh From Curve* dal menù che apparirà a video.



Conversione in mesh della curva profilo, con il 3D Cursor già posizionato nel punto giusto

A questo punto passiamo in Edit Mode, selezioniamo tutti i vertici con *A* e clicchiamo su *Spin* nella *Tool Shelf*, cambiando magari il numero di suddivisioni in 36 e l'angolo di rotazione in 360 gradi.

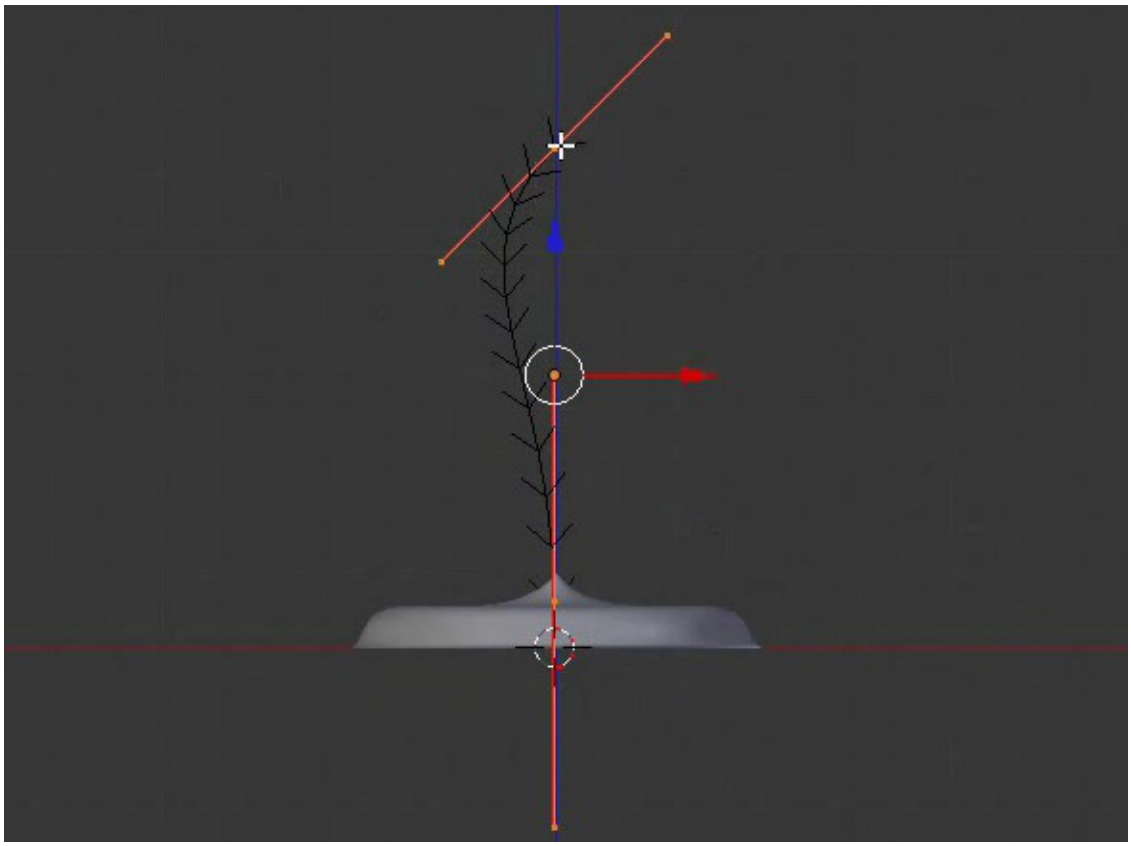


Utilizzo di Spin (vista Top Ortho, con tutti i vertici selezionati in Edit Mode e il 3D Cursor posto nel centro di rotazione)

Possiamo poi cliccare su *Shading Smooth* per levigare l'aspetto della mesh. Con questo abbiamo concluso la base.

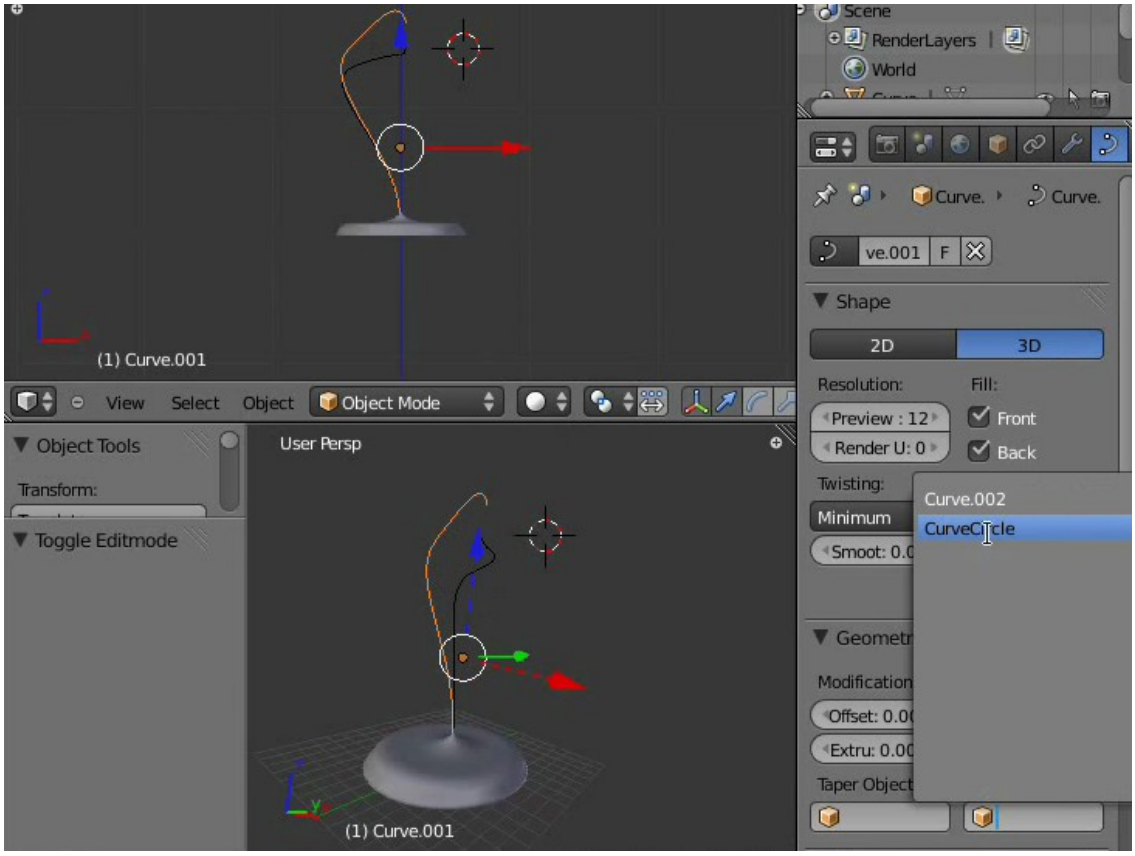
Passiamo ora alla struttura che collega la base alla lampada.
Dobbiamo inserire una curva *Circle* nella scena, posizionandola nel punto dove si trova la base superiore del supporto della lampada. Tale curva servirà da profilo per gli elementi della struttura.

Per definire il percorso di tali elementi, invece, dobbiamo inserire altre due *Curve Bezier* di base nella scena, modellandole come preferiamo, ad esempio intrecciandole o facendo far loro percorsi più o meno strani; per modellarle, possiamo estruderle, suddividerle, cambiare le maniglie e così via, proprio come abbiamo visto con la modellazione della base.



Modellazione delle curve-percorsi (la curva-profilo è già stata inserita nella scena)

Quando abbiamo finito di definire i percorsi, selezioniamo la prima curva *Bezier*, apriamo la scheda *Object Data* nella *Properties Window* e, nel campo *Bevel Object*, scriviamo il nome della curva *Circle* precedentemente creata, e che servirà da profilo per gli elementi da struttura.



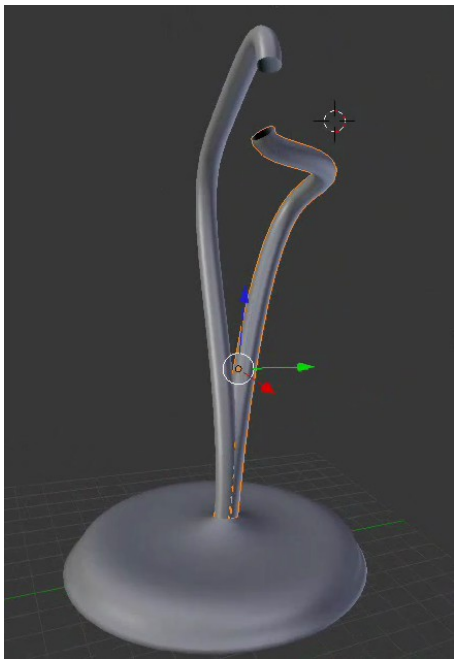
Utilizzo della curva profilo come Bevel Object per la prima curva percorso

Effettuiamo la stessa operazione per la seconda *Curva Bezier* presente nella scena, riutilizzando, in questo caso, lo stesso oggetto *Curve Circle* come profilo.

Possiamo rendere definitivo l'aspetto della struttura convertendo i due elementi in mesh, selezionandoli uno ad uno e premendo *ALT C* in Object Mode per poi scegliere *Mesh From Curve*, altrimenti possiamo continuare ad utilizzarli come curve; in questo caso possiamo modificare l'oggetto *Curve Circle*, ad esempio deformandolo o scalandolo, deformando di conseguenza il profilo degli elementi della struttura.

Alla fine comunque convertiamo le due curve *Bezier* in Mesh, fondiamole selezionando entrambi gli elementi e premendo *CTRL J*, per *Join*, e clicchiamo su *Shading Smooth* per levigare l'aspetto della mesh così ottenuta.

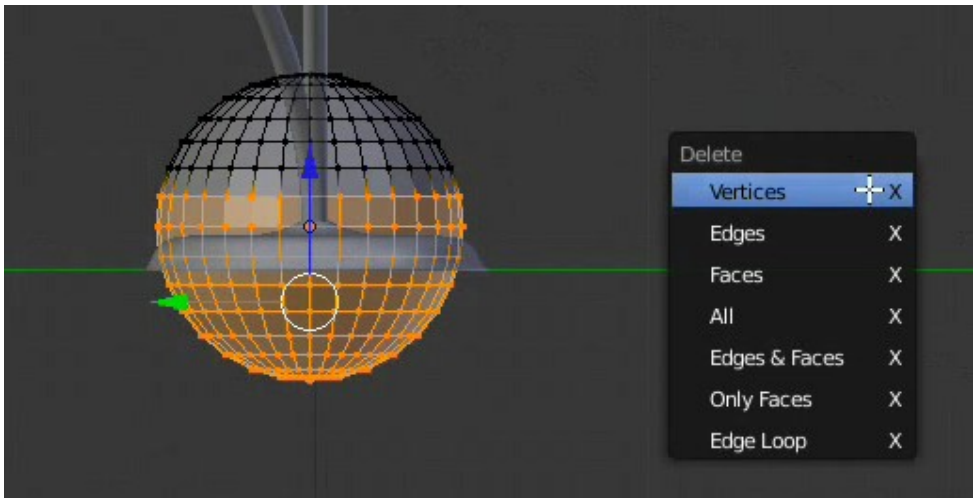
Restano da modellare i due “piatti”che nascondono la lampadina.



Le curve Bezier, dopo la conversione in mesh, la loro fusione con Join e l'applicazione dello Shading Smooth

Per realizzarli, inseriamo nella scena una *sfera UV*, ma cancelliamone metà o più, selezionando in Edit Mode gli elementi da togliere ed eliminandoli.

Per appiattire un po' il piatto, scaliamolo lungo l'asse Z premendo *S Z* e muovendo il mouse, facendo click col tasto sinistro dello stesso per confermare l'operazione quando otteniamo il risultato desiderato.



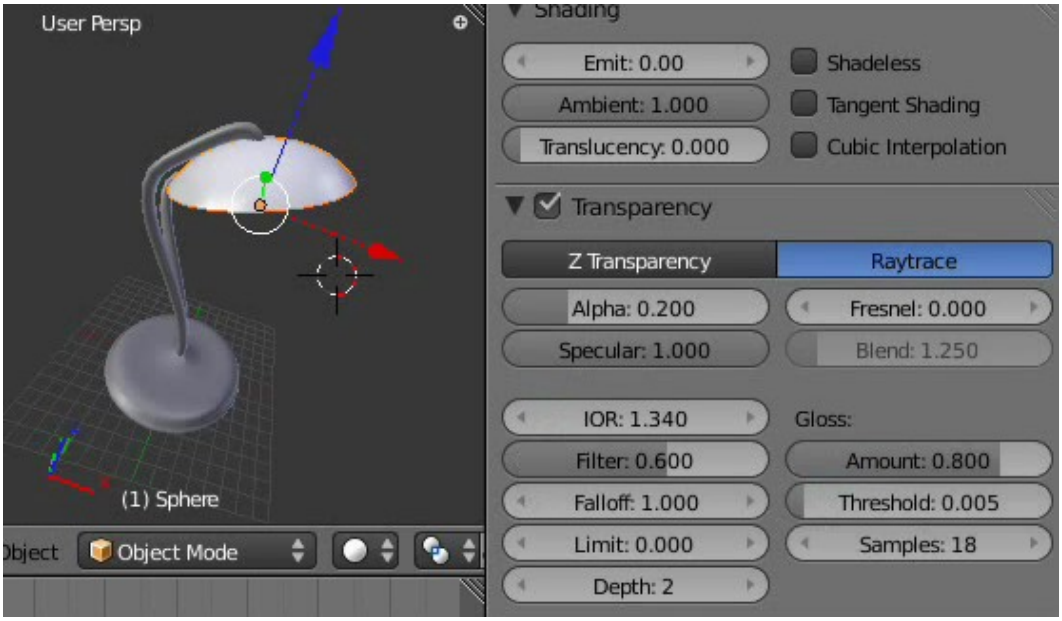
La modellazione del primo "piatto" avviene tagliando una sfera UV e appiattendola lungo l'asse verticale globale con S Z

Posizioniamo il primo piatto e, per ottenere il secondo, duplichiamolo in Object Mode con *SHIFT D*, in modo da ottenere un oggetto distinto, ruotiamolo di 180 gradi intorno all'asse Y globale e posizioniamo anche questo secondo elemento. Infine, per entrambi gli oggetti applichiamo *Shading Smooth* dalla *Tool Shelf*, per levigarne l'aspetto.

Con la modellazione di base della lampada abbiamo praticamente finito, anche se ad esempio avremmo potuto utilizzare lo strumento *Sculpt* per decorare un po' il piatto superiore o quello inferiore, oppure intrecciare di più gli elementi strutturali (la lampada modellata in maniera così semplice, come ho fatto io, sembra più moderna che Liberty, comunque gli strumenti di modellazione sono quelli in entrambi i casi). Fermiamoci qui con la modellazione e vediamo come creare, invece, un Material vetro satinato per i due piatti.

Apriamo la scheda Materials nella *Properties Window*, aggiungiamo un Material di base ed utilizziamo, come Ombreggiatore Diffuso, *Lambert*, con colore bianco puro e intensità 1. Come ombreggiatore speculare utilizziamo *Cook Torr*, colore bianco puro, *Hardness* 125, intensità 1.
Attiviamo la trasparenza *RayTrace*, con *Alpha* 0.2, *Fresnel* 0, *Specular* 1, *IOR* (cioè indice di rifrazione del Material) 1.34, *Filter* 0.6, *Falloff* 0, *Gloss Amount* 0.8, *Limit* 0 e *Depth* 2.

Il Material è praticamente finito e possiamo applicarlo ad entrambi i piatti della lampada.



Il Material "vetro satinato" (tramite parametro Glossiness) per i "piatti" della lampada

Potete definire il Material della struttura come preferite.

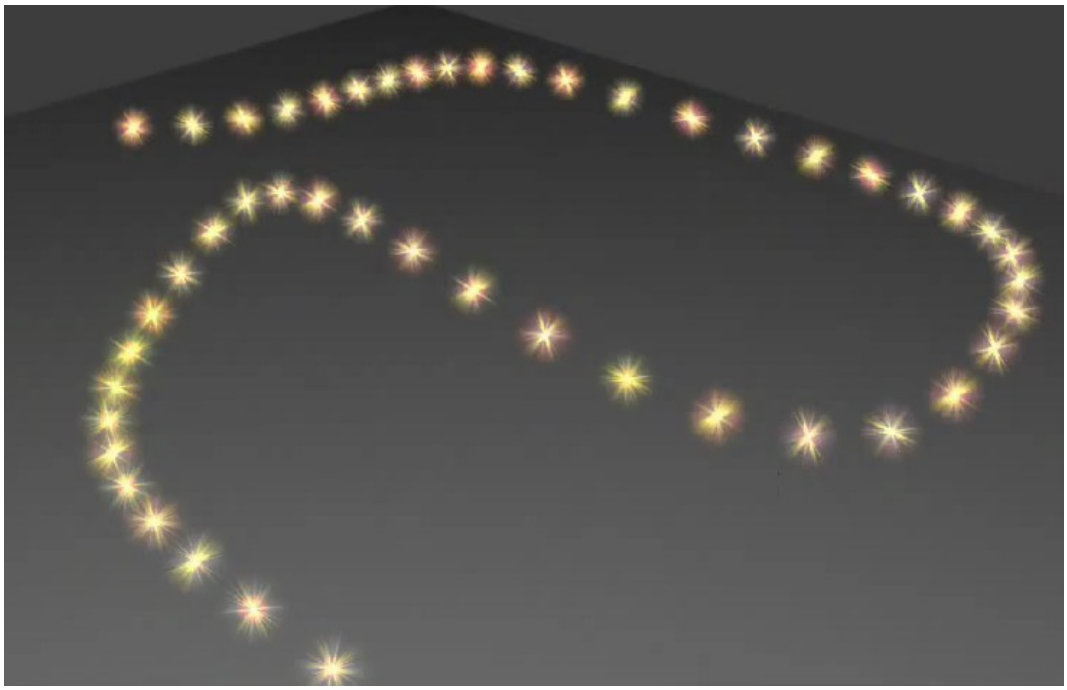
A questo punto il modello è finito e vi ho dato qualche dritta su come realizzare un Material vetro satinato; non vi resta che definire un set ove posizionare la lampada, inserire un paio di fonti di illuminazione e renderizzare la vostra scena.

Il rendering finale

* * *

Luci natalizie con Curve Bezier, Material Halo e Texture Magic

In questo tutorial, realizzato con la versione 2.55 beta di Blender, vedremo come realizzare delle luci colorate con un piccolo effetto *Halo*, elementi utili ad esempio come addobbi natalizi ed altro ancora; per intenderci, l'effetto che vogliamo ottenere è quello visibile nell'immagine seguente.

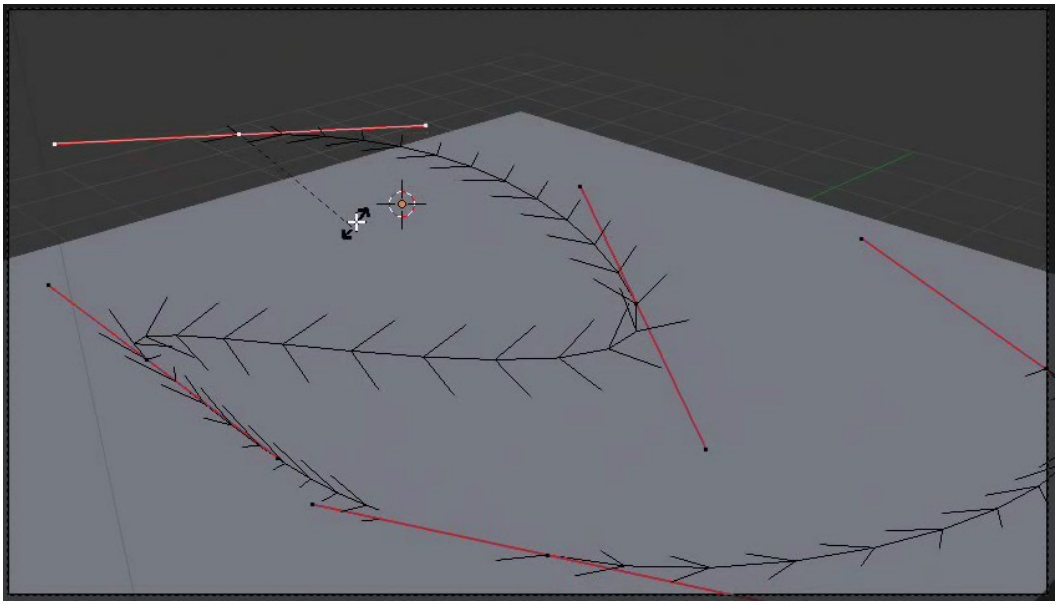


Il risultato che vogliamo ottenere

In realtà, questi elementi NON sono fonti di luce, per cui dovremo comunque ricorrere a delle *Lamp* o ad altri effetti per illuminare la scena; si tratta, come vedremo, di vertici di mesh provvisti di Material *Halo* e Texture *Magic*.

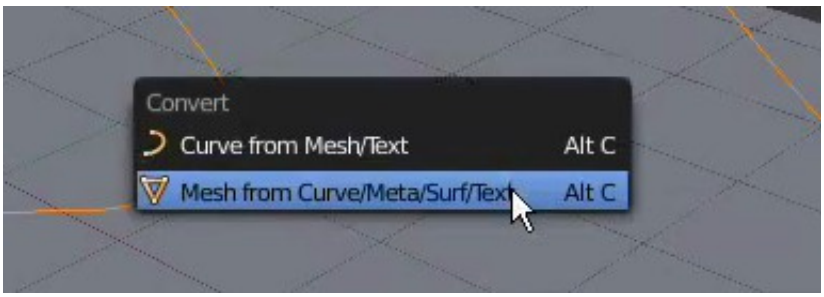
Per definire la forma di questi elementi possiamo procedere principalmente in due modi: con una mesh o con una curva *Bezier* e, da questa, una mesh, mentre materiale e Textures sono identici per entrambe le modalità di modellazione. Per il nostro scopo, comunque, conviene utilizzare una curva *Bezier*, per cui mostrerò solo questa seconda tecnica.

Aggiungiamo proprio una curva *Bezier* alla scena e modelliamola come preferiamo, aggiungendo vertici, variandone il percorso (anche nelle tre dimensioni), modificando le maniglie di controllo e così via; in questa fase, la modellazione del percorso è facile, perché con pochi punti di controllo possiamo variare di molto la forma dell'oggetto, ma in un secondo momento dovremo convertire tutto in mesh e cambiare il percorso diventerà meno facile, per cui conviene definire quest'ultimo bene in questa fase.



La modellazione della curva Bezier

Definita la forma della curva, convertiamola in mesh premendo *ALT C* mentre siamo in *Object Mode* e scegliendo, dal menù che apparirà a video, *Mesh from Curve*. Passando in Edit Mode, possiamo verificare la creazione di vari vertici lungo il percorso della curva.



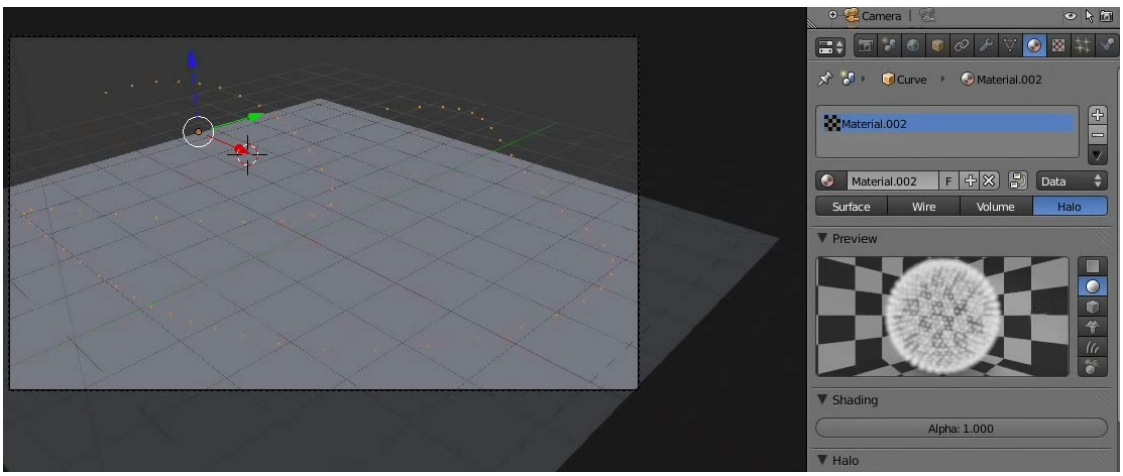
Conversione in mesh della curva, con ALT C in Object Mode

A questo punto, in Object Mode dobbiamo fare una scelta, ossia se visualizzare o meno il filo che collega queste lampadine: visualizzarlo servirà a rendere più realistico l'effetto, ma magari per un albero di Natale o per la decorazione dei muri esterni di un edificio volete evitare di visualizzarlo, quindi potreste optare per il no.

Per visualizzarlo, dobbiamo duplicare (in Object Mode, quindi creando un oggetto distinto) la mesh appena creata e creare un Material a parte, di tipo standard, non *Halo*; in realtà, basterà cambiare il colore diffuso in nero e il gioco sarà fatto.

Per non visualizzarlo... non bisogna fare assolutamente niente e passare direttamente alla definizione del Material per la mesh.

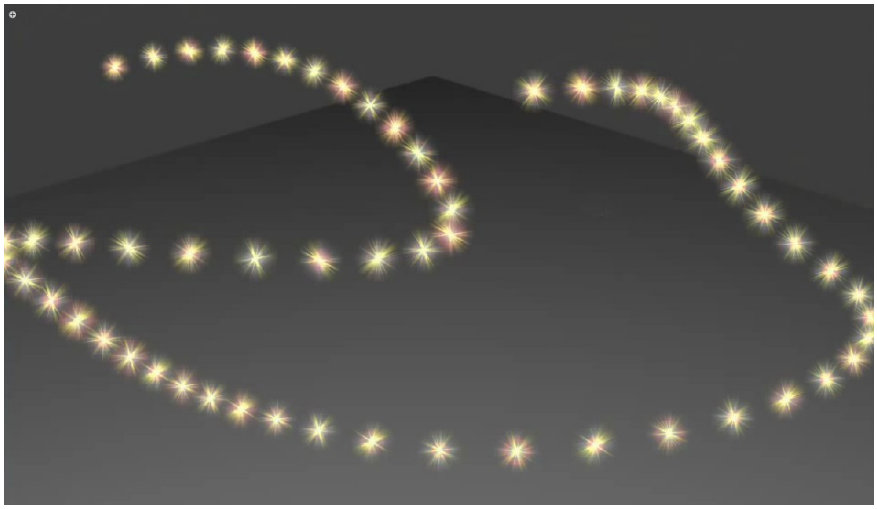
Vediamo quindi proprio questo Material.
Con la mesh selezionata, nella scheda Material, all'interno della *Properties Window*, clicchiamo su *Add* per aggiungere un Material e, in alto, tra le voci che appariranno, clicchiamo sulla voce *Halo* (mentre quella selezionata di default è *Surface*) e, in basso, modifichiamo un po' le nuove voci: il colore non è importante perché lo modificheremo con la Texture, mentre per *Size* consiglio, ad esempio, 0.2, per *Hardness* 50 e un valore basso, ma maggiore di zero, per *Add*, ad esempio 0.2.



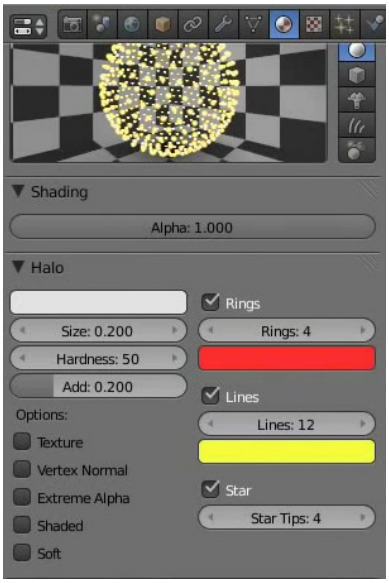
Il Material per l'effetto HALO

A voi la scelta riguardo gli effetti extra, come anelli, trattini e stelline; per mostrare un esempio, attiviamoli tutti, impostando 4 *Rings* con colore rosso, 12 *Lines* con colore giallo e *Star Tips* a 4.

Possiamo già renderizzare la scena, notando però che il colore è lo stesso per tutti i punti luminosi.



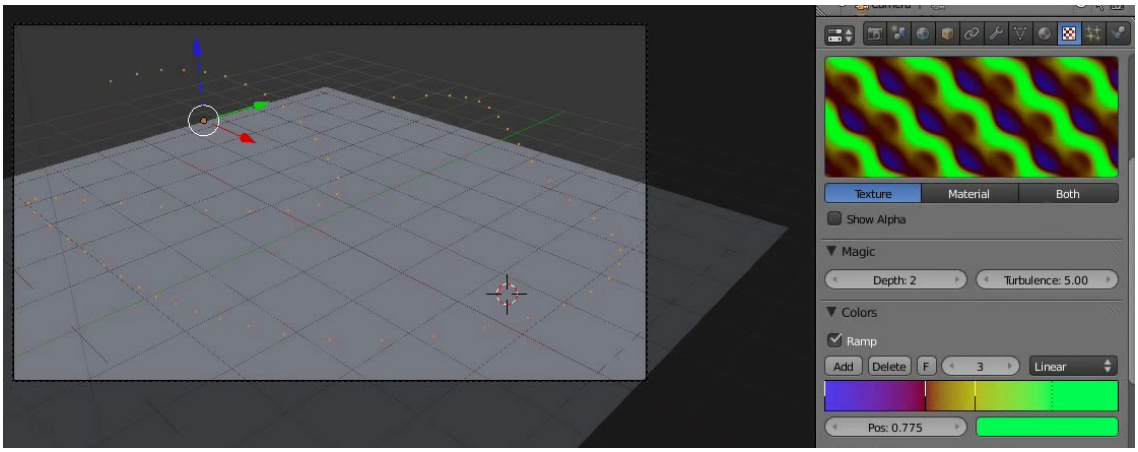
Il primo rendering della scena: tutti i punti hanno lo stesso colore



Gli effetti "extra" per il Material HALO

Per variarlo, ricorriamo ad una Texture di tipo *Magic*, per cui aggiungiamo una Texture di questo tipo ed attiviamo la *Color Ramp* nella scheda *Colors*.

Adesso dobbiamo inserire alcune suddivisioni, tutte con *Alpha* a 1 e con colori diversi, per cui ad esempio potremmo inserire una banda con una tonalità verde, una gialla, una azzurra, una rossa, e così via, come preferiamo; l'importante è, come dicevo, utilizzare *Alpha* a 1 per tutte loro.



La definizione della Color Ramp per la Texture Magic creata per il Material

In genere il metodo di interpolazione e i valori di default di luminosità, contrasto e saturazione andranno più che bene, ma eventualmente potete modificare anche quelli; possiamo lasciare così come sono le altre voci della scheda della Texture ed avviare il rendering per osservare l'effetto ottenuto.

Il rendering finale

A questo punto possiamo definire meglio l'effetto, ad esempio togliendo le stelle o le linee se l'effetto è troppo forte o inverosimile, o modificando ancora i colori della Textures *Magic* o altro ancora; il principio, comunque, è questo.

* * *

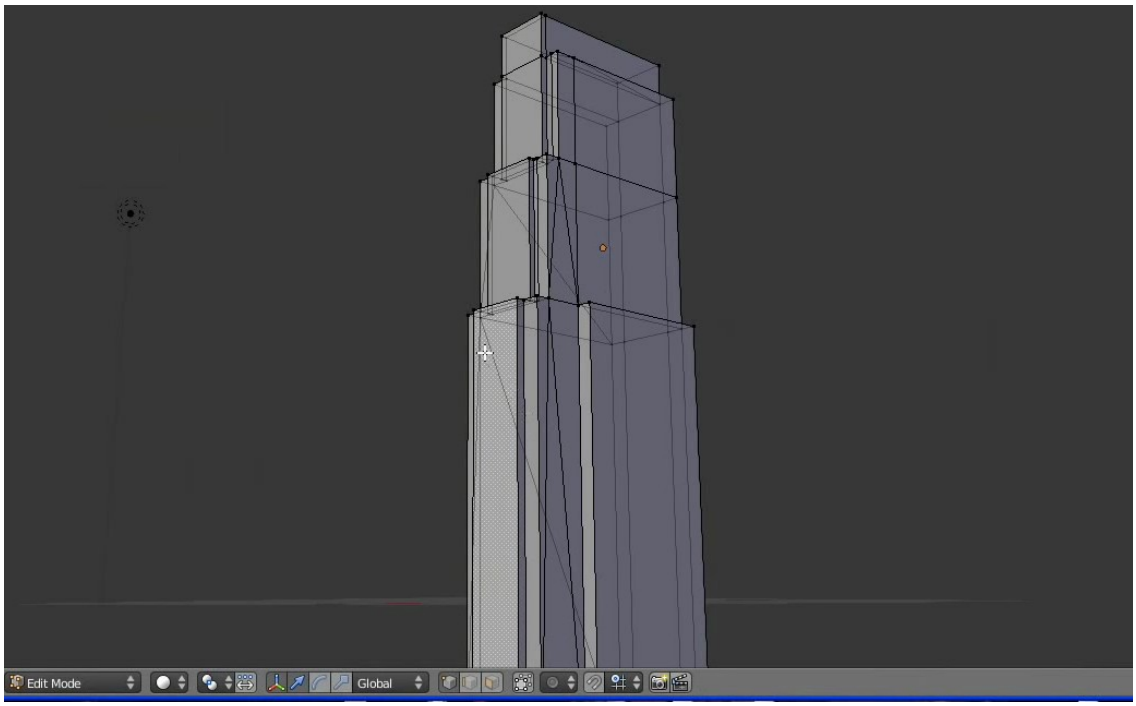
Grattacielo con pochi vertici e una Texture immagine su NORMAL

In questo tutorial, realizzato con la versione 2.55 beta di Blender 3D, parleremo dell'utilizzo di una Texture di tipo immagine da mappare sul canale *Normal* per simulare l'aspetto di un grattacielo simile al GE Building di New York, qui visibile in una foto del 1933, insieme ad un'immagine dell'effetto che otterremo.

Una foto del GE Building e, a destra, il risultato che otterremo al termine di questo tutorial

Per ottenere questo risultato non ho utilizzato una Texture per l'aspetto dell'edificio (che potete colorare come volete per renderizzarlo di giorno o di notte): quello che vedete è l'effetto dovuto alla sola applicazione, sul canale *Normal*, di una Texture immagine.

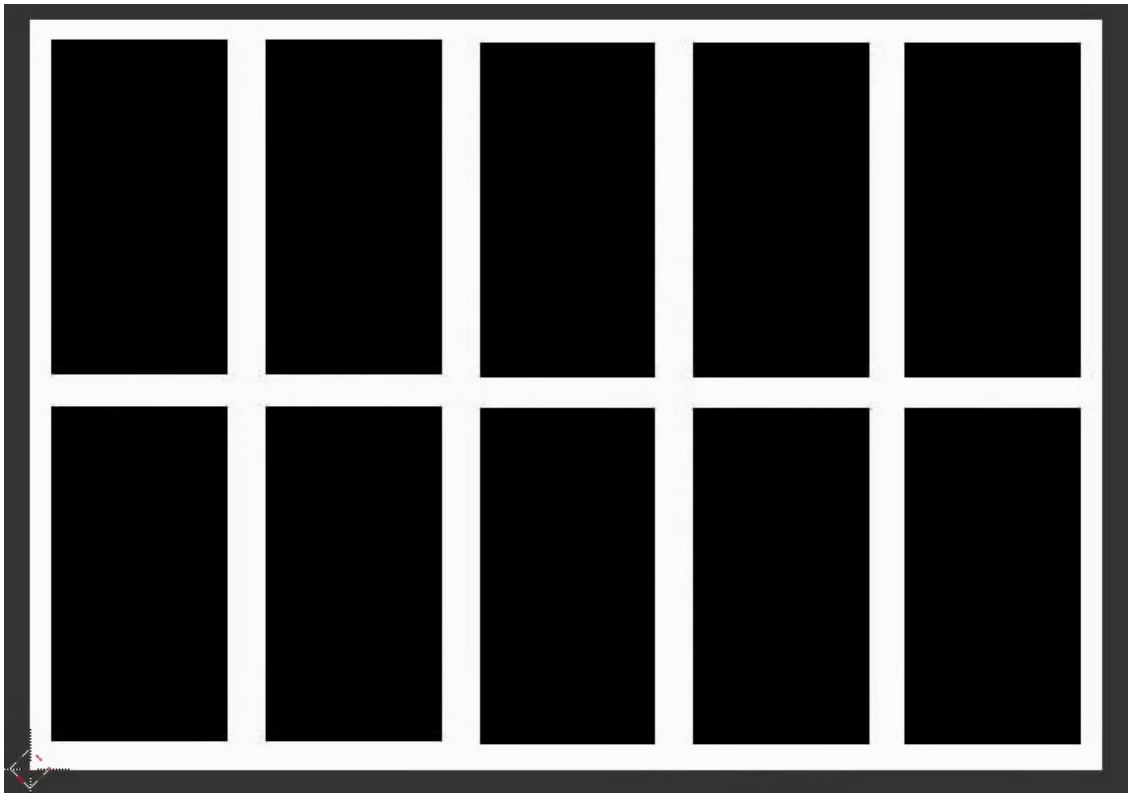
L'effetto è buono per un grattacielo visto da lontano: è sufficiente a conferirgli un aspetto credibile senza dover modellare i singoli elementi ad uno ad uno; anzi, per la verità non dovremo modellare neanche i singoli piani per poi duplicarli: come è possibile notare in Edit Mode, infatti, l'edificio è composto da alcuni *Box* particolarmente alti e allungati un po' lungo la direzione Y globale, ma in tutto sono presenti solo 70 vertici, 145 spigoli e 77 facce!



La mesh che utilizzeremo, in Edit Mode

È importante notare, per quanto riguarda la mesh, che questa dev'essere ottenuta a partire da un solo oggetto modellato, non da più oggetti uniti con *Join* o *Boolean*, o si avranno effetti indesiderati con il texturing; nel mio caso, ho definito la pianta con un semplice Plane, suddividendolo ed estrudendolo sia sul piano XY che verticalmente (asse Z globale), ed aggiungendo le varie suddivisioni che servivano per fare i piani più rientrati o le sporgenze semplicemente con il taglierino *Knife* ed estrudendo ancora.

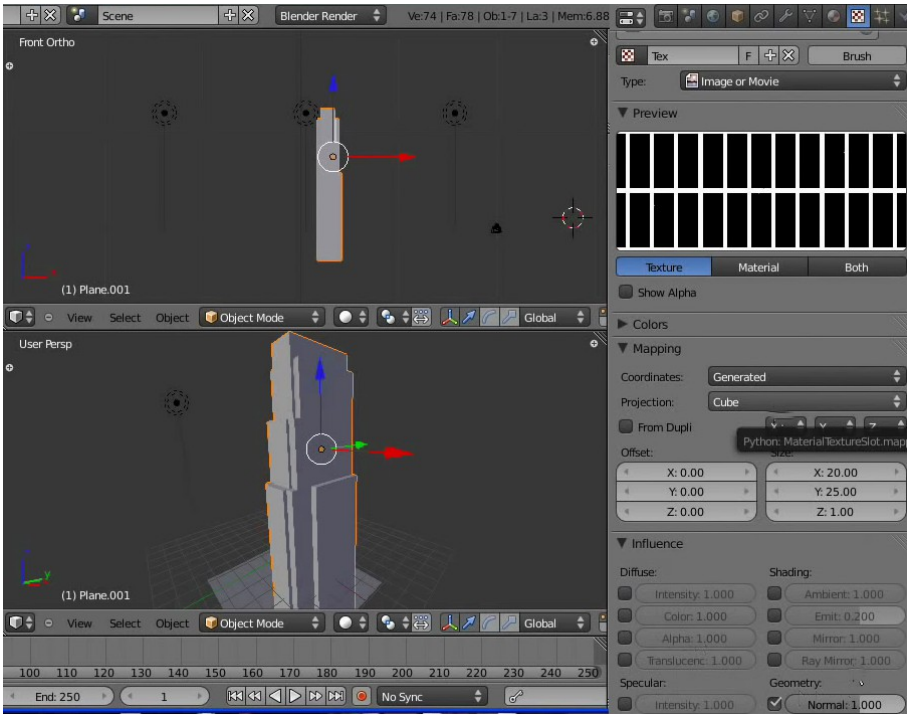
Come Texture, utilizziamo una banalissima immagine in bianco e nero fatta da due file di rettangoli neri su sfondo bianco: il canale *Normal* metterà in rilievo le parti bianche, lasciando indietro quelle nere, che rappresentano la posizione delle finestre.



Un esempio di Texture da utilizzare

Alla mesh va quindi associato un Material di base con Texture immagine e, come immagine, una simile a quella appena mostrata.

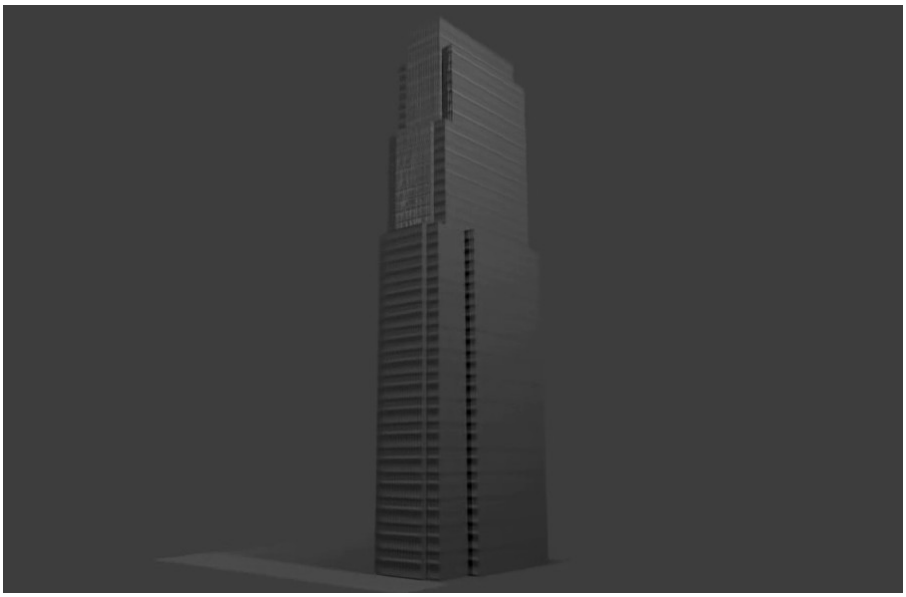
Nella sezione *Mapping* della Texture cambiamo il tipo di proiezione in cubo, in modo da "avvolgere" l'edificio; attenzione, perché qualora dovessimo inquadrare l'edificio dall'alto, dovremo provvedere ad aggiungere una mesh extra, in pratica un semplice piano, con una Texture a parte, per il tetto, o meglio "i tetti" in questo caso, perché il tipo di proiezione e la Texture utilizzata generano risultati sgradevoli nella parte superiore della mesh... per la nostra scena, comunque, questa scelta va bene. Prima di attivare il canale *Normal*, sfruttiamo il canale *Color*, selezionato di default, per valutare le dimensioni delle varie "finestre", ossia le parti nere, così come vengono proiettate di default, con *Size X*, *Y* e *Z* a 1, sulla mesh; sicuramente, visto che stiamo utilizzando una Texture piccola, occorrerà variare i valori di *Size X* e *Y* in *Mapping*, per cui vi consiglio di fare alcuni rendering per scegliere il valore adatto. Nel mio caso, ho messo 20 per *X*, 25 per *Y* e 1 per *Z*.



Le impostazioni di mappatura (Mapping) della Texture immagine

A questo punto, in *Influence* possiamo anche disattivare *Color* e selezionare *Normal*, con valore 1.

Va detto che possiamo comunque aggiungere un'altra Texture immagine creata a partire da quella precedentemente vista ma con colori diversi per gli elementi strutturali e mapparla sul canale *Color*, o su altri canali, ad esempio per mappare gli effetti di riflessione, utilizzando le stesse impostazioni viste per la proiezione e la dimensione dell'immagine: a voi tutte le possibili personalizzazioni.

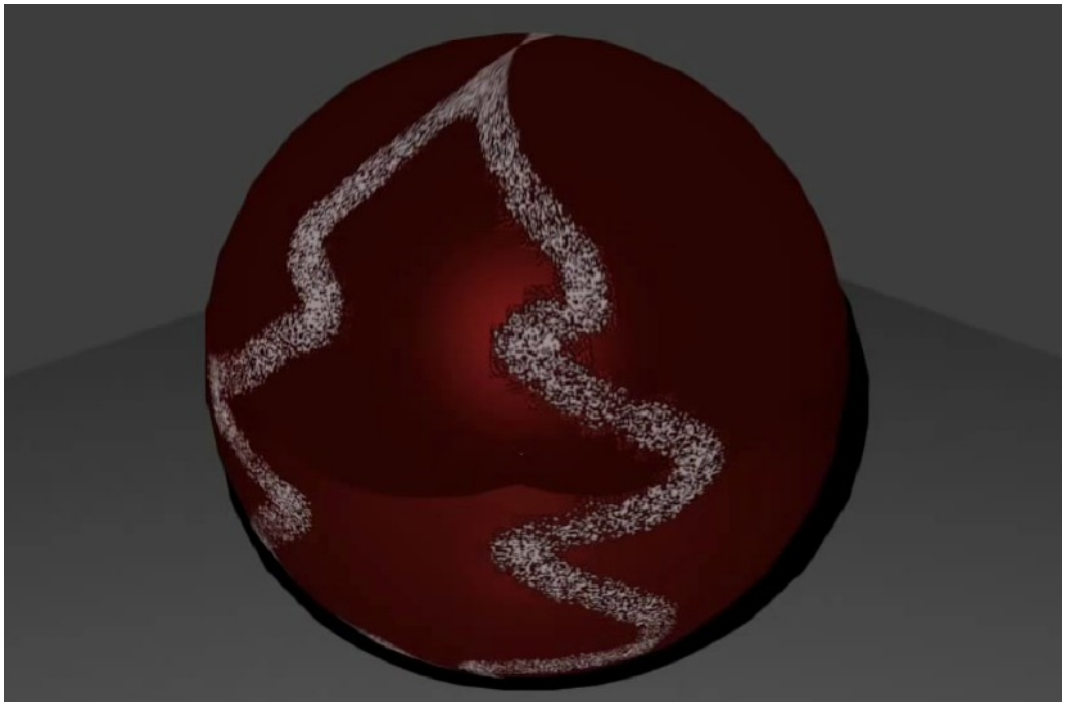


Il rendering finale

* * *

Rilievo su decorazione natalizia con Texture su NORMAL

In questo tutorial vedremo un modo per creare delle parti in rilievo su alcuni oggetti utilizzando solo una Texture; non solo: gli oggetti sono provvisti di un loro colore e le parti in rilievo devono averne un altro e, per ottenere quest'effetto, utilizzeremo una sola Texture senza trasparenza.

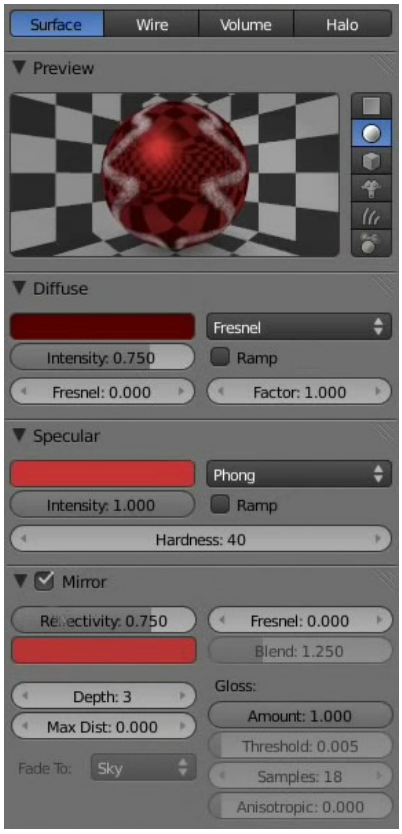


Un esempio dell'effetto che vogliamo ottenere

Iniziamo aggiungendo una semplice *sfera UV* alla scena, suddividiamola un paio di volte e levighiamone l'aspetto cliccando su *Shading Smooth*, nella *Tool Shelf*.
Come Material, realizziamone uno per ricreare della plastica per decorazioni natalizie:

- come colore diffuso, impostiamo rosso scuro (ad esempio, R 0.1, G 0.0 e B 0.0), con ombreggiatore *Fresnel*, intensità 0.75, valore di *Fresnel* 0.0 e *Factor* a 1.0;
- come colore speculare, impostiamo rosso R 0.6, G 0.03 e B 0.03, con ombreggiatore *Phong*, intensità 1 e *Hardness* 40;

- attiviamo inoltre i riflessi, selezionando *Mirror*, con valore di *Reflectivity* 0.75, *Depth* 3, colore rosso come quello speculare, cioè R 0.6, G 0.03 e B 0.03, tutti gli altri valori come di default.



Il Material associato alla sfera UV dell'addobbo

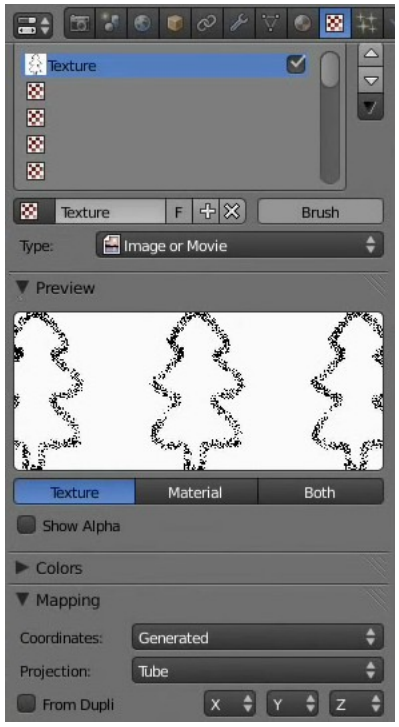
Passiamo ora alla scheda Textures: ci serve una Texture di tipo immagine, anche senza canale trasparenza, per cui possiamo utilizzare immagini PNG, BMP, JPG eccetera.

La Texture verrà mappata, tra gli altri, sul canale *Normal*, per cui utilizziamo un'immagine in bianco e nero, o comunque scala di grigi; nel mio caso, il disegno da mettere in rilievo è stato realizzato in nero su bianco, per cui attiveremo la modalità *Negative*... procediamo comunque con ordine.

Caricata l'immagine, cambiamo la proiezione in *Tube*, in modo da "avvolgere" la sfera, e portiamo a 2 il valore di *Size X*, duplicando in pratica l'immagine orizzontalmente, per cui vedremo il rilievo su entrambe le facce della sfera.

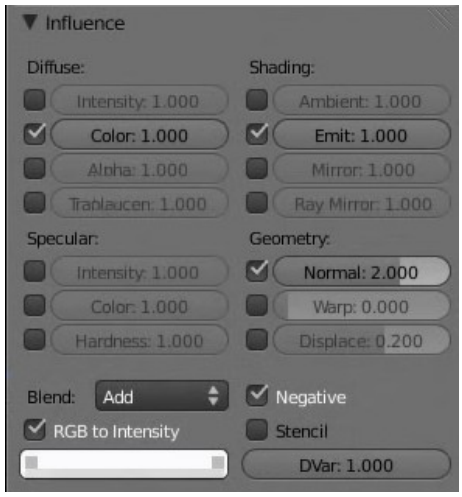
Nella scheda *Influence*, lasciamo selezionato (come avviene di default) *Diffuse Color*, con valore 1, e selezioniamo inoltre *Emit* con valore 1 e *Geometry Normal* con valore, ad esempio, 2 o 3.

Cambiamo la modalità di miscelazione *Blend* in *Add*, selezioniamo la casella *RGB To Intensity* e cambiamo il colore, nella casella sottostante, in bianco puro. Selezioniamo infine *Negative*, in basso a destra nella scheda, e questo perché, come anticipato, l'immagine del rilievo era in nero su bianco, mentre le parti in rilievo devono essere in bianco su nero.



La mappatura della Texture immagine utilizzata

L'oggetto è pronto per essere renderizzato; ricapitolando, i punti principali sono: mappatura *Tube* intorno all'oggetto, Texture immagine, *Blend Type* su *Add* e *Geometry Normal* per la Texture.

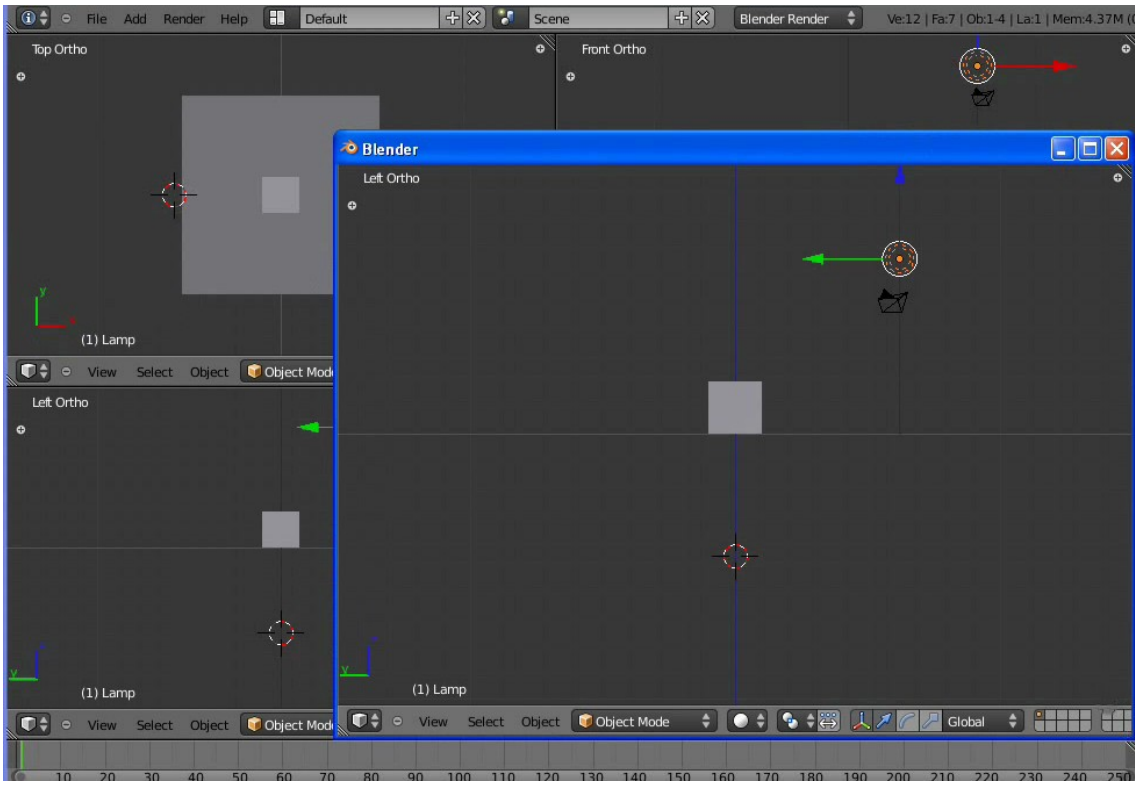


Le impostazioni Influence della Texture

* * *

Finestre floattanti su due o più monitor

Questo più che un tutorial è un "Tips & Tricks" sull'utilizzo delle finestre libere in Blender 3D 2.5. Una caratteristica molto interessante dell'interfaccia utente di Blender 2.5, disponibile già dalla versione Alpha 0, è quella delle finestre "floating", libere, dette a volte "floattanti", ossia finestre che possono essere sganciate da quella principale dell'applicazione, come fossero completamente a parte. Per creare finestre libere a partire da altre è sufficiente posizionarsi con il mouse nell'angolo in alto a destra di una finestra, dove c'è la freccia stilizzata e dove la forma del cursore diventerà un +, premere *SHIFT* e, mentre lo si tiene premuto, fare click col tasto sinistro del mouse sull'angolo.



Una finestra "floattante"

A parte lo *SHIFT*, quindi, l'operazione è identica a quella di splitting di una finestra, ma premendo *SHIFT* si crea la finestra libera.

A questo punto, all'interno della finestra così generata, possiamo modificare gli elementi della scena, allargare la finestra a schermo intero o cambiarne il contenuto.

Tutto ciò può tornare utile, inoltre, per risolvere il famoso problema dell'esecuzione di Blender su due o più monitor: adesso, infatti, è possibile creare una finestra libera, posizionarla nel secondo schermo, allargarla a schermo intero, suddividerla come si preferisce, cambiandone magari i pannelli per avere altri strumenti sotto mano, e così via.

Personalmente, ad esempio, lavoro con due finestre di Blender, una per monitor: nella prima tengo le 3 finestre *TOP RIGHT LEFT* in modalità ortogonale più la finestra *User* o *Camera* in modalità prospettiva, mentre nella seconda finestra dispongo verticalmente 4 o 5 schede *Properties*, ciascuna con selezionata una sotto-scheda diversa, ad esempio *Render*, *World*, *Object Data*, *Material* e *Texture*, per avere tutto sotto controllo.

Salvando un file e riaprendolo in seguito, Blender caricherà anche le finestre floattanti, solo che inizialmente le posizionerà in un solo monitor, per cui poi dovremo spostare quelle da posizionare negli altri monitor, ma è un lavoro che richiede pochi secondi per volta.

La chiusura di una sola finestra float non genera la chiusura di tutto il programma, che può essere terminato o chiudendo in sequenza tutte le finestre grafiche oppure chiudendo la finestra *Console*.

* * *

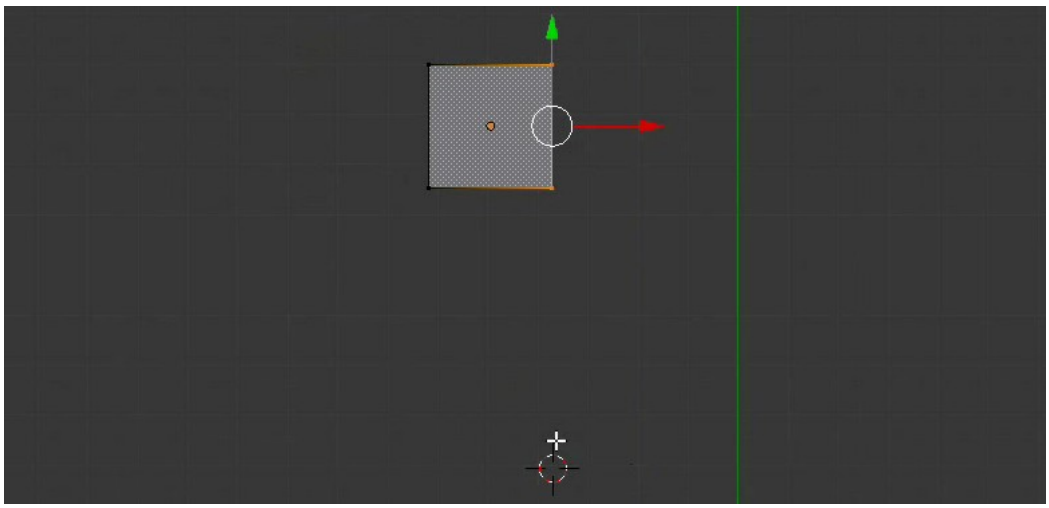
Usare Spin come un compasso per estrudere ad arco, con rotazione

In questo tutorial, realizzato con Blender 2.55 beta, vedremo come utilizzare *Spin* per effettuare un'estrusione con rotazione intorno ad un punto, quindi un'estrusione ad arco, utilizzando *Spin* come un compasso. Tale operazione può essere effettuata sia per un vertice, creando un arco fatto di spigoli, che per spigoli, creando un arco di facce, e può tornare utile per realizzare certe mesh che hanno porzioni curve al loro interno.

In questo tutorial vedremo l'estrusione per spigoli, tuttavia quella per vertice si basa sullo stesso principio, solo che all'inizio dovrete selezionare, anziché uno spigolo (come mostrerò io) un singolo vertice.

Per mostrare un esempio utilizzerò un semplice Plane.
Quello che vogliamo fare è effettuare un'estrusione curva, diciamo di 90 gradi (vedremo comunque presto come variare il valore), partendo da uno spigolo del Plane.

Con il Plane selezionato, passiamo quindi in Edit Mode e selezioniamo lo spigolo da estrudere.

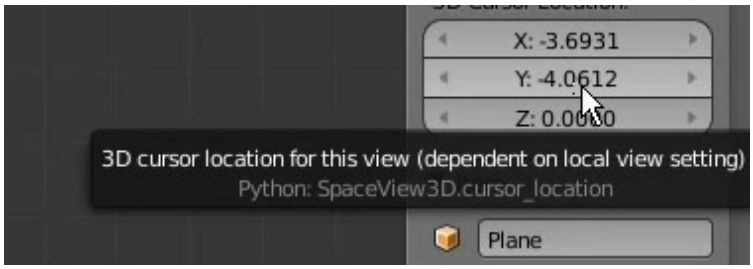


Lo spigolo del Plane, selezionato in Edit Mode, inquadrato da una vista Top Ortogonale, con il 3D Cursor in quello che sarà il centro di rotazione

Se il Plane giace sul piano XY, come nel mio caso, posizioniamoci nella vista *TOP Ortogonale*, in modo da vederlo dall'alto; in generale, per utilizzare *Spin* dobbiamo posizionarci in modo da osservare l'oggetto frontalmente, perché l'asse di rotazione è uscente dallo schermo, perpendicolare al piano osservato.

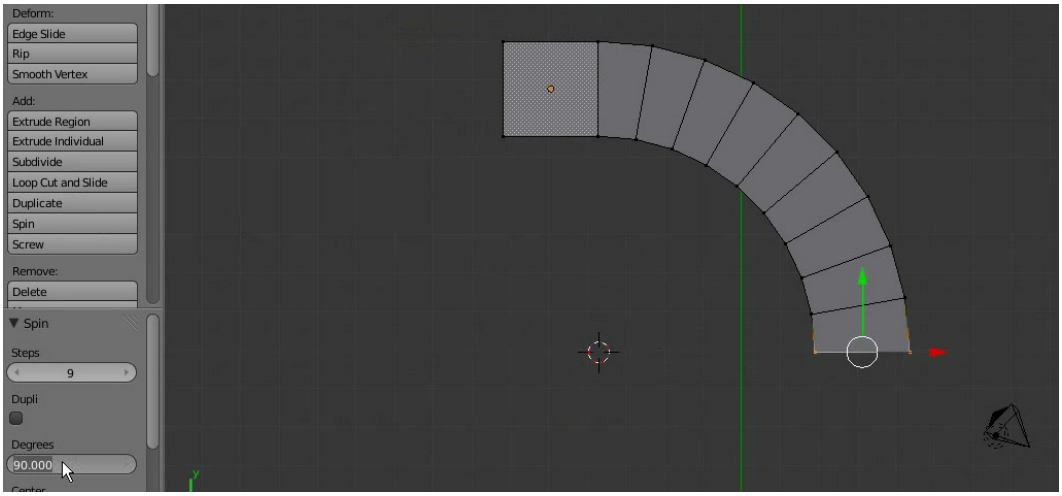
Posizioniamo il cursore del mouse nel punto che, per un compasso, coincide con la punta metallica, cioè il perno della rotazione, per cui sulla retta su cui si trova lo spigolo da estrudere, ma ad una certa distanza, da uno dei due lati, a seconda del percorso che vogliamo dare all'estrusione.

Per posizionare con cura il *3D Cursor* nella scena possiamo ricorrere anche alle voci della sezione *3D Cursor Location*, nella *Transform Window*, richiamabile premendo *N* mentre siamo in una finestra di navigazione *3D View*.



Posizionamento del 3D Cursor

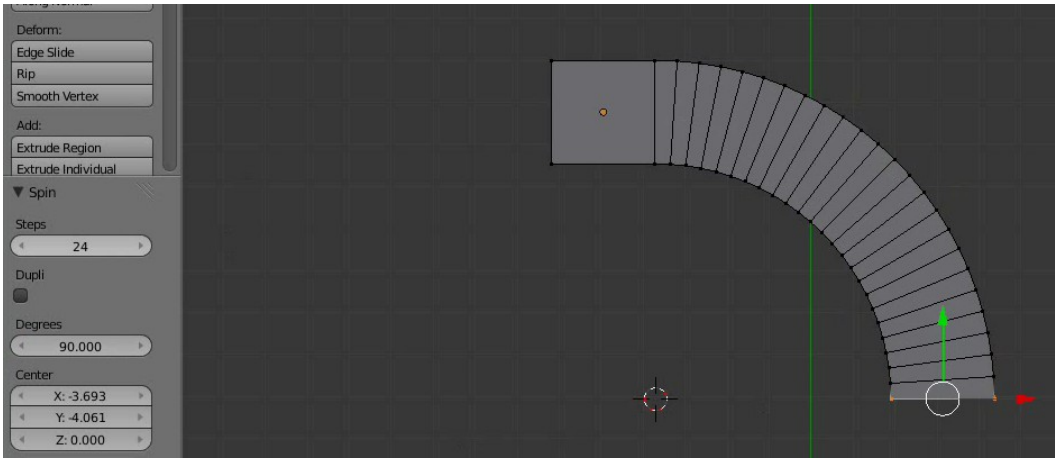
Adesso apriamo la *Tool Shelf* con *T* e scegliamo la voce *Spin*.
Di default, verrà effettuata l'estrusione ad arco per 90 gradi, ma possiamo cambiare questo valore come preferiamo, ad esempio 30 gradi, 180 gradi per effettuare una semicirconferenza, o 270 gradi, e così via; per cambiare il valore, dobbiamo utilizzare il campo *Degrees*, cioè appunto “gradi”, in basso nella *Tool Shelf*, dove ora sono presenti i paramentri del modificatore *Spin*.



Utilizzo di SPIN

Possiamo anche utilizzare un valore negativo, e questo può tornare utile perché certe volte Blender potrebbe effettuare l'estrusione ad arco, sì, ma dall'altra parte, sovrapponendosi alla mesh originale, e questo problema può essere risolto scrivendo appunto un valore negativo, come -30, -90 o -270 nel campo *Degrees* del modificatore.

Il campo *Steps* indica invece il numero di passi di suddivisione da effettuare per realizzare l'estrusione con rotazione; ad un valore maggiore corrisponderanno risultati migliori, specie per un elevato numero di gradi, ma ovviamente ciò comporterà un maggior numero di vertici, spigoli e facce e, quindi, peso nella gestione della mesh.
Potete fare varie prove per trovare un buon compromesso.

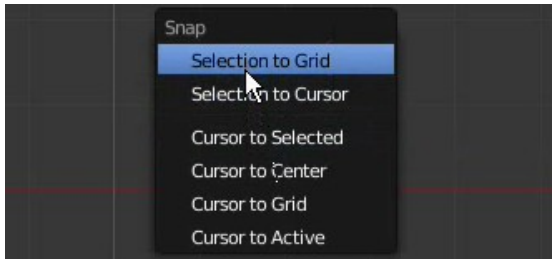


Variare il parametro STEPS

* * *

Tips and Tricks GUI: snap del 3D cursor con SHIFT S

Questo più che un tutorial è un "Tips & Tricks" su alcune funzionalità di *SNAP* del cursore in Blender 3D 2.5. Vediamo subito di che si tratta: in una qualsiasi *3D View*, con un qualsiasi oggetto selezionato, premiamo *SHIFT S* ed osserviamo le voci del menù che apparirà a video.

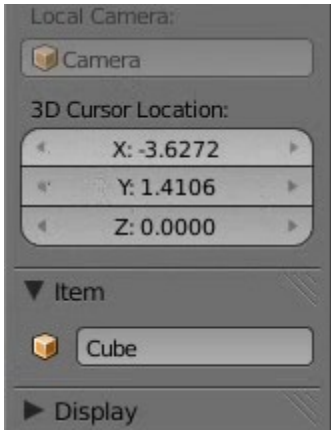


Le voci del menù SNAP (SHIFT S)

Il significato di ciascuna di esse è, infatti, abbastanza chiaro e spiega bene quali effetti è possibile ottenere. Con la prima voce, *Selection To Grid*, possiamo spostare velocemente l'oggetto selezionato nel punto griglia più vicino, detto *Grid Node*, in pratica arrotondando a intero i valori delle sue coordinate X, Y e Z nell'universo virtuale.

Con la seconda voce, *Selection to Cursor*, possiamo invece spostare l'oggetto selezionato nel punto ove è presente il *3D Cursor.*

La posizione del *3D Cursor* può essere impostata in maniera parametrica aprendo la *Transform Window*, con *N* in una finestra di navigazione *3D View*, ed impostando manualmente i valori numerici delle coordinate nei campi X, Y e Z della sezione *3D Cursor Location*.



Impostare le coordinate del 3D Cursor in una Transform Window

Le prime due voci del menù aperto con *SHIFT S* ci consentono quindi di spostare velocemente l'origine dell'oggetto selezionato in un punto griglia o nel punto del *3D Cursor*, mentre le altre voci servono a posizionare velocemente proprio il *3D Cursor* in punti precisi.

Con *Cursor to Selected*, possiamo spostare il *3D Cursor* nel centro dell'oggetto o degli oggetti selezionati, posizionandolo in quest'ultimo caso nel baricentro della selezione.

Con *Cursor to Center* possiamo portare il *3D Cursor* nel centro dell'universo virtuale, ossia alle coordinate 0, 0, 0 (l'origine), cosa che può tornare utile per aggiungere nuovi oggetti proprio nel centro della scena oppure per spostare un oggetto nell'origine del sistema di riferimento, selezionandolo e scegliendo, in seguito, *Selection to Cursor*, sempre dal menù *SHIFT S*.

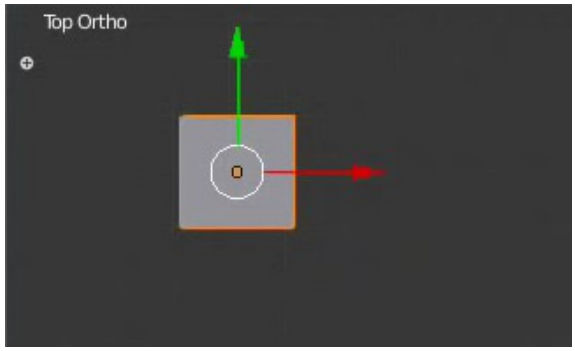
Cursor to Grid fa per il cursore tridimensionale quello che *Selection to Grid* faceva per l'oggetto selezionato: lo sposta nel nodo griglia più vicino.

Cursor to Active, per finire, sposta il cursore 3D nel centro dell'oggetto attivo: quando l'oggetto selezionato è uno solo, esso è anche l'oggetto attivo, mentre quando la selezione è multipla (ad esempio perché effettuata con *SHIFT* e click del tasto destro del mouse o con *A*, *B* o *C*) l'oggetto attivo è uno solo degli oggetti della selezione; in genere, si tratta dell'ultimo oggetto selezionato.

* * *

Differenze tra Spin Dup e Array con Warp per la rotoduplicazione

In questo tutorial, realizzato con Blender 2.55 beta, parleremo di due diversi modi per disporre intorno ad un punto una mesh; vedremo, inoltre, le differenze che si hanno anche in termini di modellazione degli oggetti e di facilità d'uso degli strumenti. Gli strumenti che vedremo sono *Spin Dup* per il primo metodo e *Array* e *Warp* per il secondo. Iniziamo con l'aggiungere, se non presente, un oggetto alla scena.



Inserimento un oggetto qualsiasi nella scena

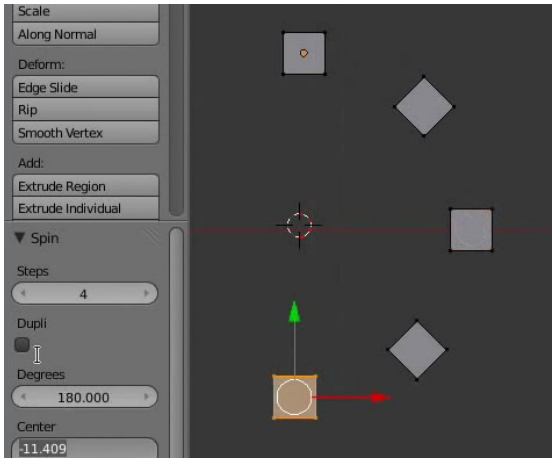
Posizioniamoci in una vista che ci consenta di osservare l'insieme di oggetti che disporremo in cerchio dall'alto, e questo perché per utilizzare *Spin Dup* e *Warp* dobbiamo definire un centro e un asse di rotazione; i modificatori, inoltre, devono essere applicati nella finestra *3D View* in cui l'asse di rotazione è uscente dallo schermo, perpendicolare al piano visualizzato.

Nel mio caso, gli oggetti si troveranno sul piano XY e l'asse di rotazione è quello Z, diretto in alto, per cui mi sto posizionando nella finestra *TOP* in modalità *ortogonale*.

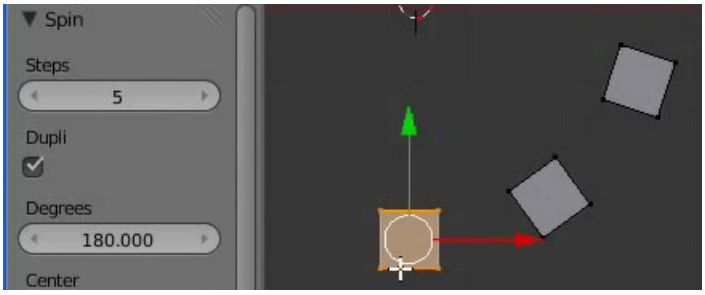
Selezioniamo l'oggetto e posizioniamo il *3D Cursor* nel punto che dovrà fare da centro di roto-duplicazione di *Spin Dup*, utilizzando magari i campi della sezione *3D Cursor Location* nella *Transform Window*, richiamabile con *N* nella *3D View.*

A questo punto passiamo in Edit Mode, selezioniamo tutti i vertici e clicchiamo su *Spin* nella *Tool Shelf*, richiamabile con *T* nella *3D View.*

In basso nella *Tool Shelf* appariranno i comandi propri di *Spin*: selezioniamo la casella *Dupli*; magari con un cubo o altri oggetti non si nota alcuna differenza, ma con singoli spigoli o altri elementi sì: anziché estrudere, infatti, *creeremo delle copie a partire della selezione*.



Utilizzo di SPIN, con l'oggetto inquadrato "dall'alto", gli elementi selezionati in Edit Mode e il 3D Cursor posizionato al centro della rotazione



Il risultato dopo aver selezionato l'opzione "DUPLI" per Spin

A questo punto possiamo personalizzare ancora l'operazione, impostando il numero di *Steps*, che in questo caso indica il numero di copie da fare e da disporre a cerchio, oppure il numero di gradi di rotazione (ad esempio, 90, 180, 270, 360) con *Degrees*, oppure possiamo spostare il centro di rotazione variando i valori dei campi *Center*, in basso nella scheda.

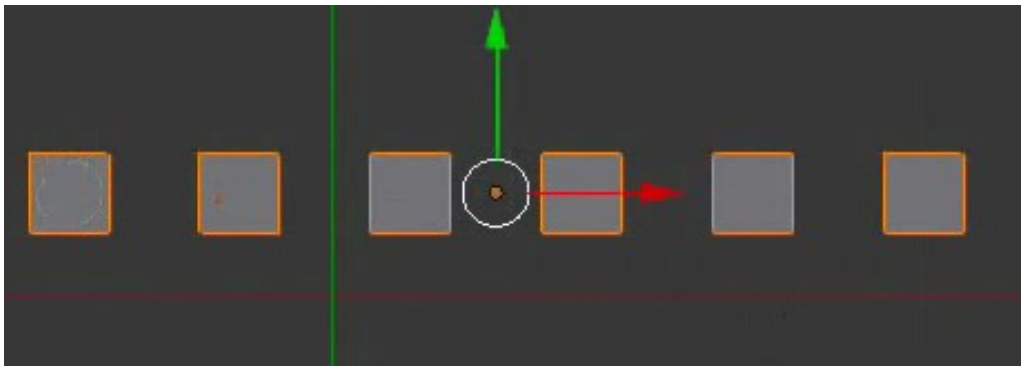
Quello appena descritto è il primo metodo per ottenere la duplicazione SENZA deformazione, disponendo inoltre in cerchio gli oggetti.

Vediamo ora la seconda modalità: *Array* e *Warp*, che introduce invece la deformazione delle mesh. Aggiungiamo un altro oggetto alla scena, per utilizzarlo come oggetto iniziale.

Un nuovo oggetto nella scena, da utilizzare con Array e Warp

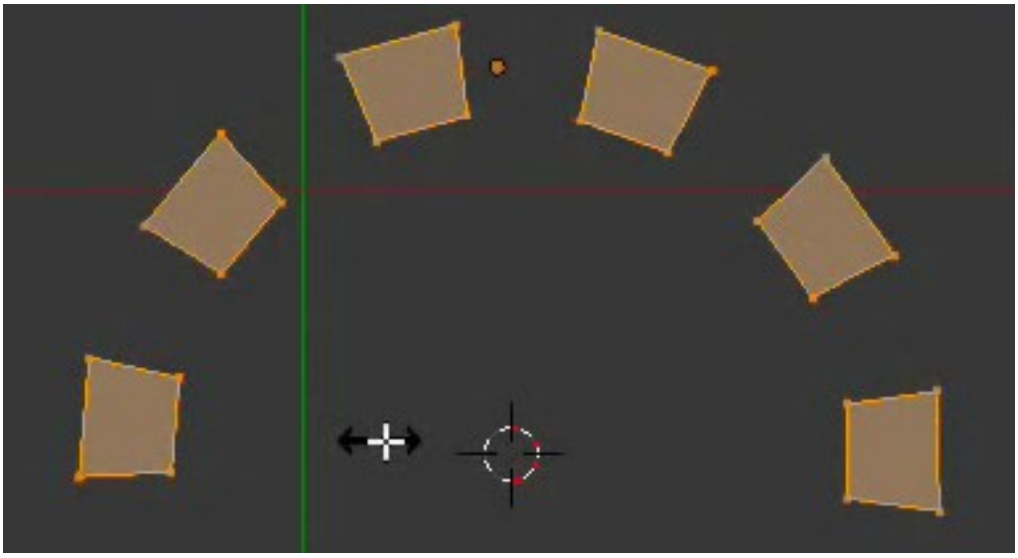
Selezioniamo l'oggetto ed apriamo il pannello *Modifiers* all'interno della *Properties Window* per aggiungere un modificatore *Array*, in modo da creare delle copie dell'oggetto disponendole lungo un asse e distanziandole un po' tra loro.

Applichiamo il modificatore, ottenendo un'unica mesh, e risistemiamo l'origine dell'oggetto con *CTRL SHIFT ALT C* e scegliendo *Origin to Geometry* dal menù che apparirà a video.



Il risultato dopo aver applicato il modificatore Array e centrato l'origine nella nuova mesh così ottenuta

Posizioniamo il *3D Cursor* nel punto che dovrà fare da centro di curvatura di *Warp*.
Adesso, con l'oggetto selezionato, passiamo in Edit Mode, selezioniamo tutti i vertici e premiamo *SHIFT W*, muovendo poi il mouse per modificare interattivamente l'entità della curvatura e facendo click col tasto sinistro dello stesso per rilasciare.



Warping, con SHIFT W in Edit Mode e tutti gli elementi selezionati, dopo aver posizionato il 3D Cursor nel centro della curvatura

A questo punto, le differenze tra i due metodi sono evidenti: con *Spin Dup*, definiamo il numero di copie da fare e per quanti gradi curvare dopo aver applicato *Spin*, mentre con *Array* e *Warp* dobbiamo definire all'inizio il numero di elementi da creare e, in seguito, la curvatura; la differenza più importante è, però, un'altra: con *Warp*, introduciamo anche una deformazione delle mesh.

Tale deformazione è, per così dire, "radiale": parte dal centro di curvatura e va verso l'esterno.
Spin Dup non deforma, invece, le copie create.

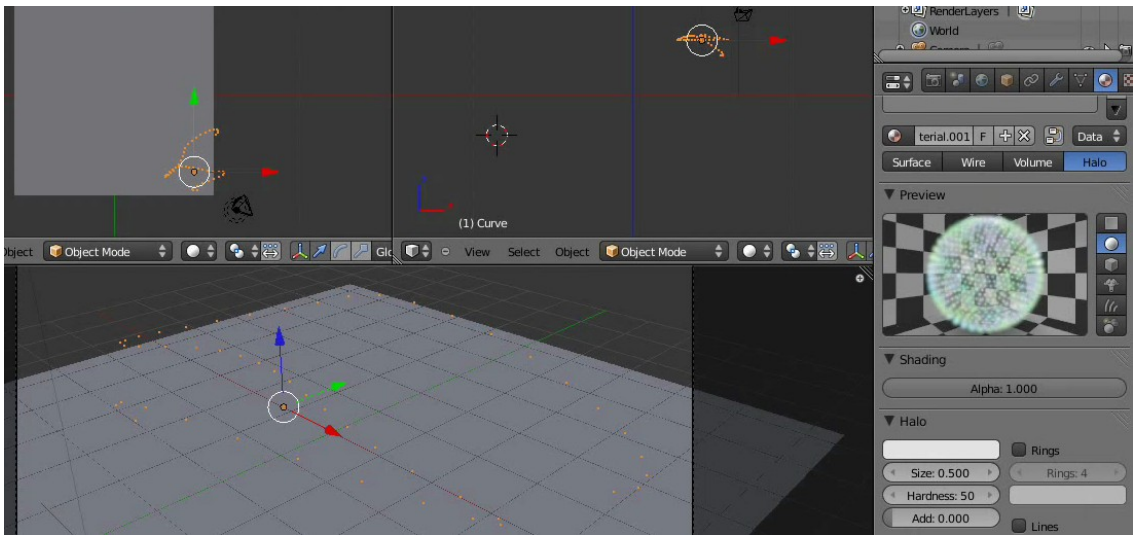
La curvatura con deformazione può tornare utile in certi contesti, per cui ora conosciamo anche un modo semplice per ottenere quest'effetto; a voi la scelta, a seconda delle vostre esigenze, della tecnica da utilizzare.

* * *

Luci natalizie intermittenti con Halo, IPO e F-Curve Cycles

In questo tutorial vedremo come realizzare le luci natalizie intermittenti in Blender 3D 2.5 utilizzando le *curve IPO* e il modificatore *Cycles* (cioè cicli, ripezioni) per ripetere l'effetto per tutta l'animazione.

In un tutorial precedente ho mostrato un modo per realizzare l'effetto luci natalizie mediante curve *Bezier*, *Texture Magic* e *Material Halo* in Blender; ecco un piccolo riassunto: si definisce il profilo della curva *Bezier* a seconda del percorso da far seguire alle luci, poi si trasforma la curva *Bezier* in mesh con *ALT C* e *Convert to Mesh*, in Object Mode, dopodiché si associa un Material di tipo *Halo* all'oggetto, modificando a piacere i parametri di luminosità ed effetti speciali, ed infine, per variare il colore dei punti luce, si associa al Material una Texture di tipo *Magic*, eventualmente modificandone i colori nella *Color Band*, se lo si ritiene opportuno.



La situazione di partenza

Detto questo, selezioniamo la scheda Material all'interno della *Properties Window* e vediamo come animare l'intensità dell'effetto *Halo*, cosa che si traduce nel rendere le luci intermittenti.

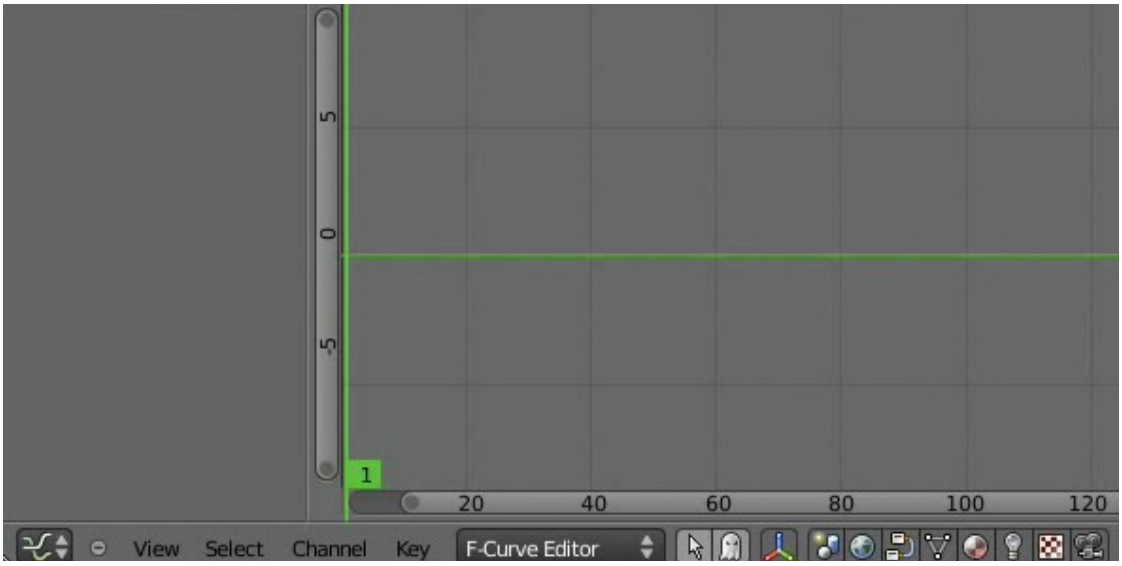
Quello che dobbiamo fare è inserire dei keyframe per il canale *ALPHA*, cioè la trasparenza, del Material *Halo*. Portando *Alpha* a 0, infatti, renderemo gli oggetti invisibili.

Senza scendere troppo in dettaglio con le basi dell'animazione, visto che ciò richiederebbe molto tempo, diciamo qui che un keyframe, o fotogramma chiave, è un particolare fotogramma nell'animazione dove, per un qualche parametro, fissiamo un valore specifico.

Quello che succede, in Blender e in altri programmi di computer grafica sia 2D che 3D, è che, fissato un valore per un parametro in due fotogrammi distinti, il programma realizzerà l'interpolazione dei valori tra i due fotogrammi, per cui, ad esempio, se per il fotogramma 1 fisseremo il valore *Alpha* del Material a 1 e per il fotogramma 25 fisseremo il valore *Alpha* 0, nei fotogrammi intermedi il valore varierà proprio da 1 a 0, realizzando così l'animazione.

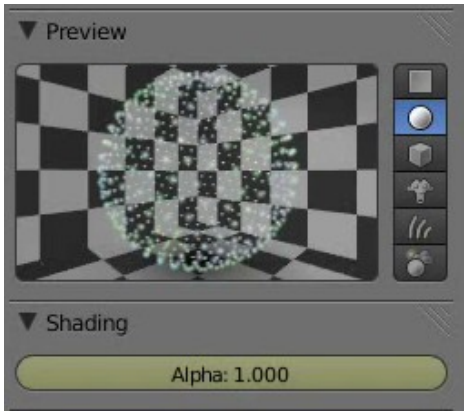
Questo concetto è valido anche per spostare oggetti nella scena, far muovere un arto a un personaggio, e così via.

Tra due o più fotogrammi chiave viene quindi creata una curva, che rappresenta la variazione dei valori nel tempo. Tale curva è detta *curva IPO*, cioè *InterPolation Curve*, curva di interpolazione, ed è mostrata in una particolare finestra di Blender detta *Graph Editor*, per cui, per prima cosa, cambiamo una delle finestre del programma in una finestra *Graph Editor*.



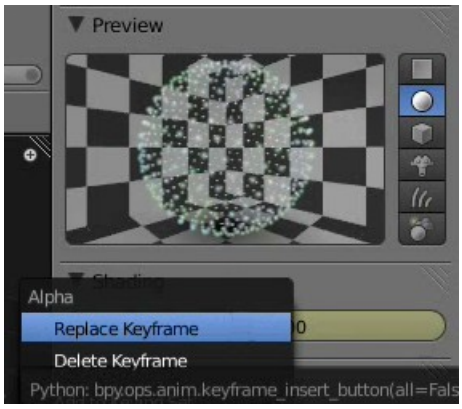
Una finestra Graph Editor

Ora passiamo alla pratica. Al frame 1 dell'animazione, nella barra temporale, posizioniamo il cursore del mouse sulla casella *Alpha* del Material *Halo*, nella *Properties Window* e, senza cliccare col mouse, premiamo il tasto *I*, per *Interpolazione*; abbiamo appena creato un *Keyframe* per questo valore in questo frame, cioè abbiamo "registrato" *Alpha* 1 al frame 1 dell'animazione.



Inserimento di un Keyframe per il canale Alpha del Material HALO

Alternativamente, possiamo fare click col tasto destro del mouse e scegliere la voce *Insert Keyframe*; tra l'altro, nel menù che apparirà a video avremo a disposizione altre voci utili per gestire i keyframes, come *Delete Keyframe* per rimuovere un fotogramma chiave.



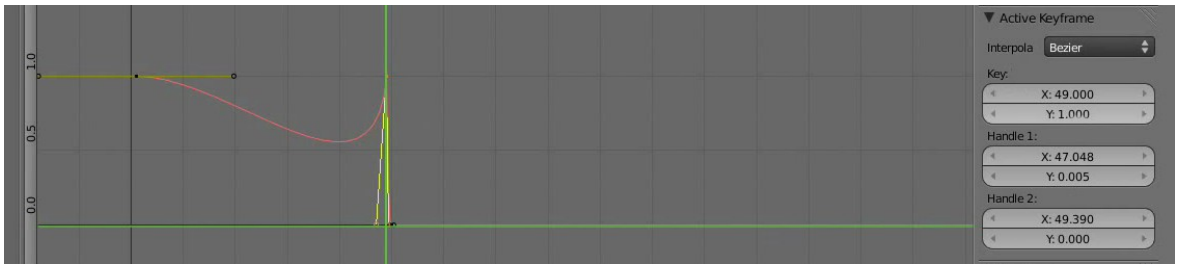
Il menù di gestione dei Keyframe, con click destro sul pavimento Alpha (o qualsiasi altro parametro dell'interfaccia di Blender)

Per creare l'animazione spostiamoci in un altro frame, ad esempio al frame 50 se vogliamo lasciare accese le lampadine per due secondi in un filmato a 25 frame al secondo; portiamo quindi *Alpha* a 0 per spegnerle ed inseriamo un nuovo keyframe. Attenzione, perché abbiamo commesso, volendo, un errore: guardando la *curva IPO* creata automaticamente da Blender nel *Graph Editor*, infatti, possiamo notare che così il valore di *Alpha* varia da 1 a 0 in 50 frames, mentre noi vogliamo tenere accese le lampadine per 2 secondi e poi spegnerle all'istante!

Risolviamo questo problema manipolando direttamente la *curva IPO*: nella finestra *Graph Editor*, con la curva selezionata (come avviene di default), facciamo click col tasto sinistro del mouse in un punto sulla colonna 49, colonna che rappresenta appunto il keyframe, premiamo *I* e scegliamo "*Only Selected Channel*", ossia inserisci un keyframe solo per il canale selezionato.

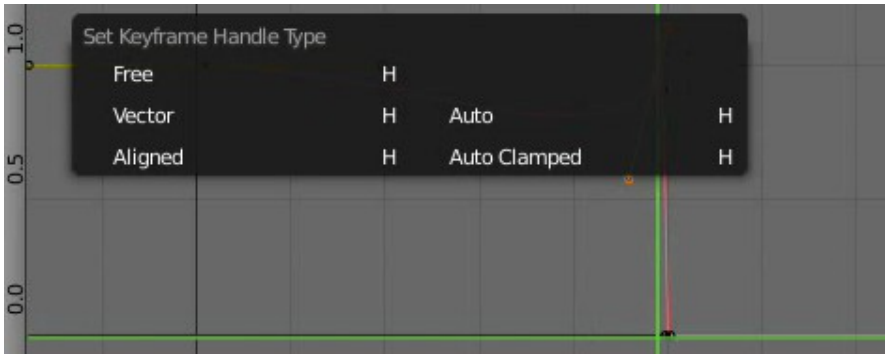
Inserimento di un Keyframe sul solo canale selezionato direttamente in una finestra Graph Editor

Ora dobbiamo spostare questo keyframe al valore 1 sull'asse Y e possiamo farlo premendo *G* e trascinando, facendo poi click col tasto sinistro del mouse per rilasciare, oppure, sempre col nodo selezionato, premendo *N* e digitando 1 nel campo Y all'interno della scheda *Active Keyframe*, che apparirà a video proprio alla pressione del tasto *N*.



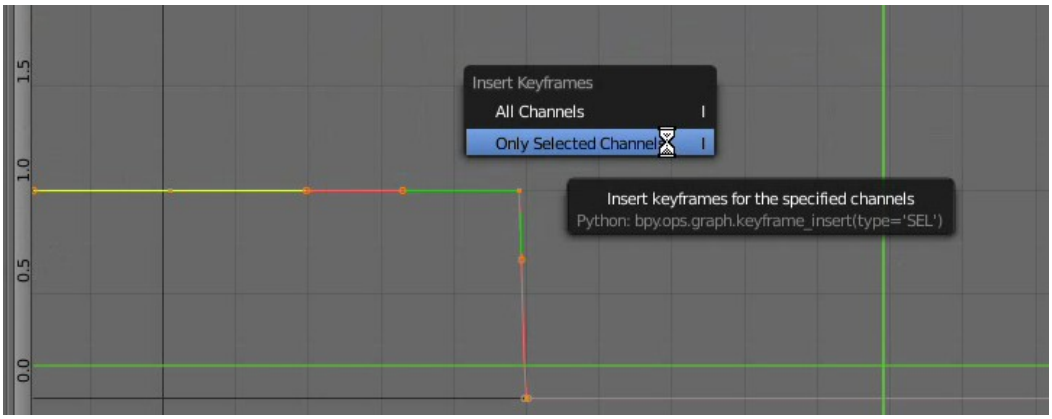
Digitare i valori per l'Active Keyframe nella Transform Window del Graph Editor

Come potete constatare, *N* apre un pannello simile alla *Transform Window* per vertici o oggetti nelle finestre *3D View*, mentre la *curva IPO* è a tutti gli effetti una *Bezier*, con i punti di controllo selezionabili con un click del tasto destro del mouse, anche con selezione multipla, e provvisti di maniglie che possono essere libere o vettoriali, ed anzi per il keyframe creato al punto 49 conviene proprio cambiare la maniglia in vettoriale premendo *V* e scegliendo *Vector* nel menù a video.



Cambiare una maniglia della curva IPO in vettoriale

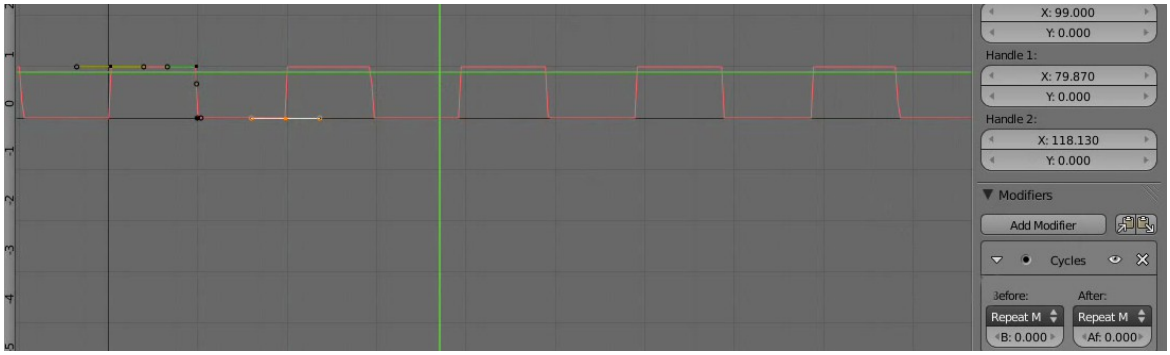
Bene, abbiamo definito la prima parte del ciclo di intermittenza: la fase ON e il passaggio a OFF; ora spostiamoci in un altro keyframe, ad esempio a 99, per inserire un altro fotogramma chiave per il canale *Alpha* del Material *Halo* sempre con valore 0 (non rifacciamo l'errore che abbiamo fatto prima, inserendo il fotogramma successivo direttamente a 1) per definire l'istante in cui finirà il periodo OFF.



Il nuovo Keyframe al frame 99

Questo è un ciclo On/Off, quindi, e per ripeterlo per tutta la nostra animazione (che nel mio caso è quella standard di 250 frames, ma che può durare anche migliaia di frames) non dobbiamo certo ripeterlo manualmente; Blender infatti fornisce dei modificatori ad hoc per le *curve IPO* e a noi interessa un modificatore che ripeta all'infinito i keyframes impostati.

Sempre nel pannello aperto prima con *N*, clicchiamo su *Add Modifier*, cioè aggiungi modificatore, e scegliamo *Cycles,* "cicli": come potete vedere osservando la *curva IPO*, abbiamo appena applicato il ciclo Acceso/Spento, definito una sola volta, a tutta la nostra animazione.



Utilizzo del modificatore "Cycles" per la curva IPO selezionata

Prima ho definito dei passaggi bruschi e nell'arco di un frame, ma ovviamente potete personalizzare i cicli facendo spegnere più lentamente le lampadine e accendendole all'istante, oppure potete utilizzare le maniglie dei singoli punti di controllo per cambiare l'interpolazione; la cosa più interessante è che potete effettuare queste modifiche anche dopo aver inserito il modificatore *Cycles*, perché le trasformazioni verranno applicate automaticamente a tutte le ripetizioni create.

* * *

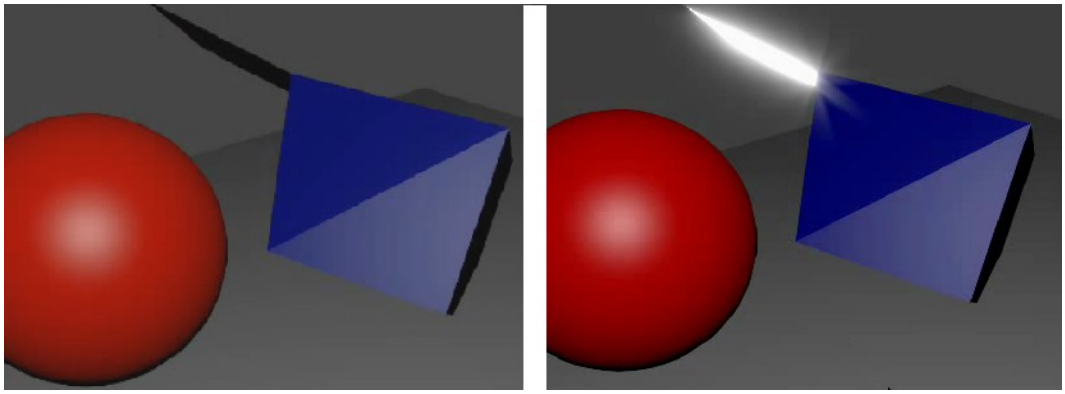
Bagliore Glare stile Tron con Materiali Multipli e Nodes Compositing

Questo tutorial nasce da una richiesta di un utente del forum, *Maxxximus*, che vuole realizzare un effetto bagliore stile *Tron* in Blender 3D 2.5.

La difficoltà, rispetto all'implementazione di un effetto *Glare* selettivo, da me descritto sia nel nono numero di *Blender Magazine Italia* che nel mio *ebook sul Compositing in Blender 2.5*, sta nel fatto che questa volta l'effetto dev'essere applicato ad una sottoparte di un oggetto, non ad una mesh a parte, e alle superfici dell'oggetto, non alla struttura *wireframe*, come spiegato in un mio precedente tutorial (presente in questa raccolta) sul *Glare* selettivo animato.

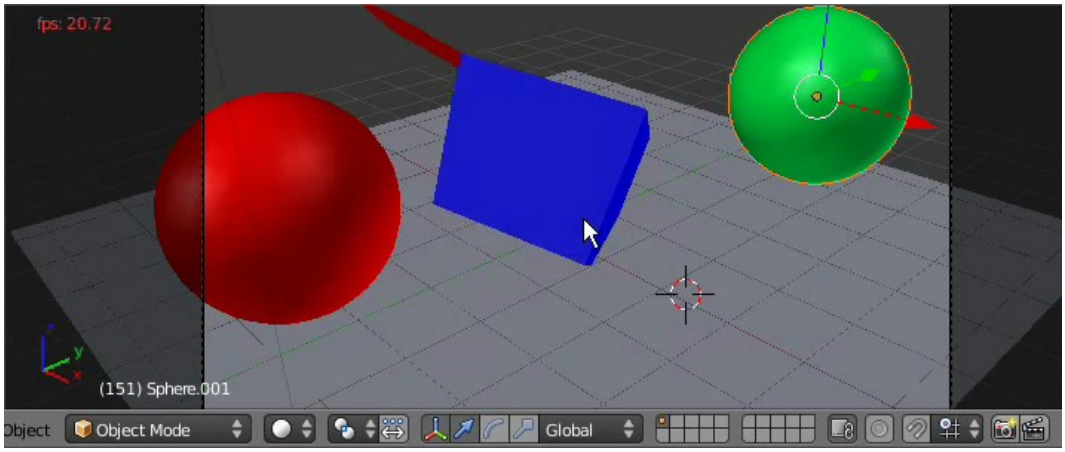
L'effetto è un po' particolare e non rende bene con i *Material Halo* o altre tecniche di base.
Ho dovuto far ricorso, quindi, ai *Materiali Multipli* e a qualche tecnica di *Node Compositing*.

Qui potete vedere, in due screenshot tratti da due filmati, l'effetto che ho ottenuto: nel primo screenshot un fotogramma dell'animazione senza effetto, nel secondo quella con l'effetto, chiaramente applicato ad una sottoparte della mesh, cioè quella che aveva la faccia rossa.



Un fotogramma dell'animazione senza (a sinistra) e con (a destra) l'effetto, applicato alla porzione di mesh con il sottomateriale rosso

La scena che ho realizzato per provare l'effetto è costituita da una fonte di luce, una telecamera, un Plane senza Material, una sfera rossa (colore RGB 1 0 0, *Alpha* 1), una sfera verde con *Reflectivity* 0.5 e un cubo provvisto di un *Materiale Multiplo*: blu per le facce di base, rosso (dello stesso rosso della sfera) per una faccia extra.



La scena utilizzata per questo tutorial

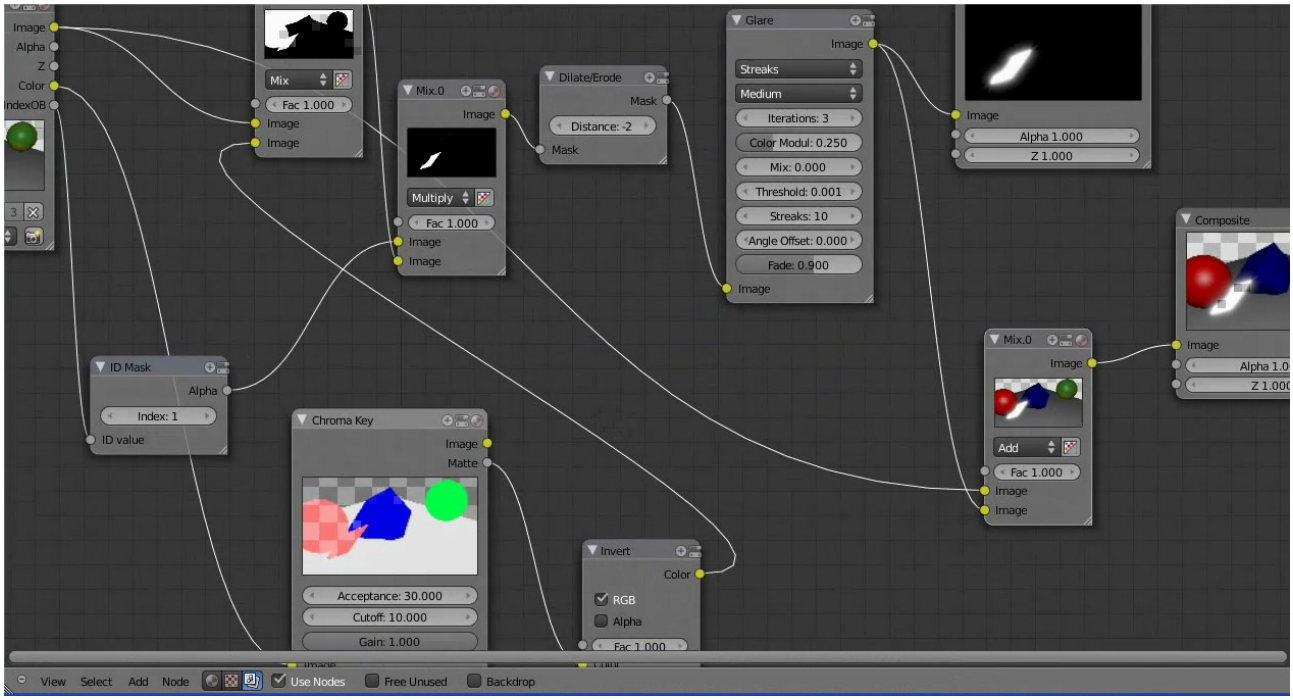
L'obiettivo è quello di applicare l'effetto *Glare* solo alla faccia rossa del cubo; ovviamente, quella mostrata è solo una possibile soluzione, ma potete provarne altre!

Passiamo alla finestra *Node Editor* per analizzare la soluzione trovata.
L'idea è quella di assegnare un particolare colore diffuso alla parte da sottoporre a *Glare*, separarlo dagli altri con delle maschere, applicare *Glare* e ricomporre il tutto; il problema è, come avrete intuito, che altri oggetti nella scena possono avere lo stesso colore e che ruotando l'oggetto, o modificando le fonti di luce, la faccia può cambiare aspetto, con una sfumatura del colore originale o con la presenza di ombre.

Per risolvere il primo problema, ricorriamo alla tecnica *ID Mask*, come nel caso del *Glare* selettivo, mentre per il secondo utilizziamo il canale di *Output DIFFUSE* di *Render Layer*, che restituisce (per ogni pixel dell'immagine renderizzata) il colore diffuso piatto dell'oggetto visualizzato, indipendentemente dalle fonti di luce, dalle ombre e dall'orientamento dell'oggetto.

Qui non posso descrivere in dettaglio i *Nodes* utilizzati e le loro funzioni; per una trattazione completa rimando, come anticipato, al mio *ebook sul Compositing in Blender 2.5*.

Analizziamo quindi il flusso dell'esecuzione.



Lo schema di Nodi di Compositing

L'immagine in uscita dal canale *Color*, che rilascia solo il colore diffuso, va ad un *Nodo Chroma Key*, dove è possibile impostare con precisione i colori R G B da individuare e, quindi, la maschera da applicare; la maschera di *Chroma Key* funziona al contrario, per cui è necessario impostare un *Nodo Invert* sull'immagine maschera generata.

La maschera così ottenuta può essere mescolata con l'immagine originale, ottenendo un'immagine dove solo gli oggetti di quel colore sono bianchi, tutto il resto è nero. Nel nostro caso, anche la sfera fa parte della maschera bianca, perché è dello stesso colore diffuso della sottoparte della mesh, per cui ora dobbiamo ricorrere al *Nodo ID MASK*.

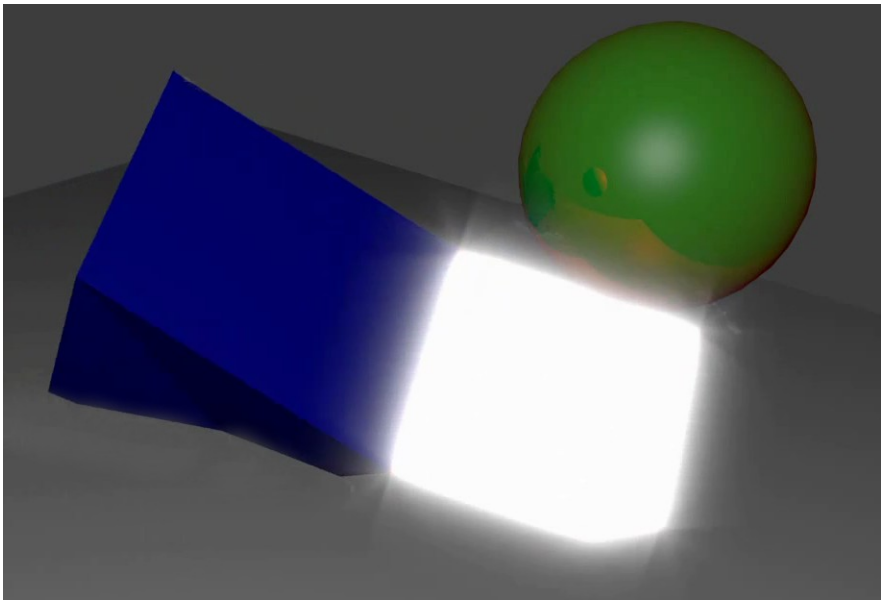
Colleghiamo quindi il canale *Index OB* di *Render Layers* ad un *Nodo ID MASK* con *ID* pari a quello impostato per il cubo, cosa che ci consente, tra l'altro, di escludere lo sfondo (background) dall'effetto *Glare*.

Moltiplichiamo quindi questa maschera con la maschera in arrivo dal processo del *Chroma Key* ed otteniamo la maschera applicata al cubo.

Nel primo rendering ho notato la presenza (anomala, a dire il vero) di uno spigolo rosso in un'altra faccia della mesh; forse tale bordino era dovuto ad una chiusura mal riuscita della mesh o a qualche strano riflesso, comunque per toglierlo è bastato aggiungere un *Nodo di erosione* all'output della maschera ottenuta. A questo punto si passa la maschera ottenuta al canale di *Input* del *Nodo Glare*, si imposta l'effetto desiderato e si miscela l'output (in modalità *Add*, la somma) all'immagine originale in arrivo da *Render Layer*. L'output di questa miscelazione è l'effetto finale.

Maxxximus mi ha chiesto poi di poter colorare un po' il bagliore *Glare*, per fornirgli una sfumatura e non renderlo proprio bianco: si può ottenere facilmente un effetto del genere giocando un po' con gli altri *Nodes*, ad esempio *Input Color* o *Input Textures* e mescolandoli all'output di *Glare*; le potenzialità offerte dal *Nodes Compositing* per la post produzione sono praticamente infinite.

Unica pecca di questa tecnica è che, come potete vedere osservando il filmato con l'effetto, non è possibile ricreare i riflessi negli oggetti con *Mirror* presenti nella scena, e questo perché l'effetto *Glare* è applicato in Post Produzione, non nel rendering della scena.



Assenza di riflessi del Glare negli oggetti Mirror (la sfera verde, nell'immagine)

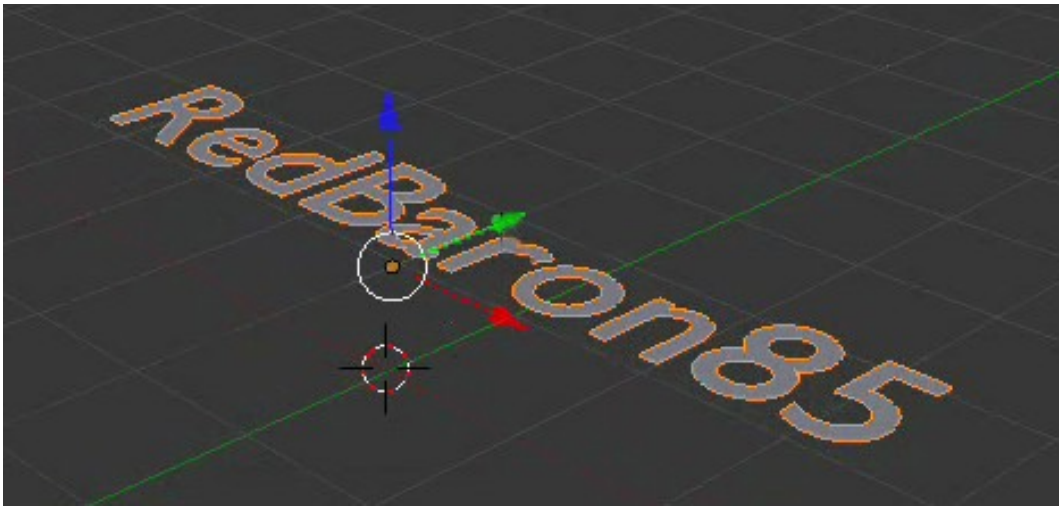
Con un motore di rendering esterno, come *Yafaray*, l'effetto luminescente dei Materiali può essere reso anche in maniera migliore e generando riflessi sugli altri oggetti, facendo ricorso ai Materials *Mesh Light*; per questo tutorial (relativo solo a Blender Internal), invece, è tutto.

* * *

Testo inciso su superfici, curve e mesh

In questo tutorial vedremo un piccolo trucchetto di modellazione in Blender 3D: come realizzare del testo inciso su una mesh senza usare gli operatori booleani che, ogni tanto, danno ancora qualche problema nel generare buone mesh. Il testo dev'essere inciso, almeno inizialmente, su una superficie piana, ma ovviamente poi potremo deformare la stessa, ad esempio mediante *Warp* o gli oggetti *Lattice*, per farle assumere la forma che preferiamo.

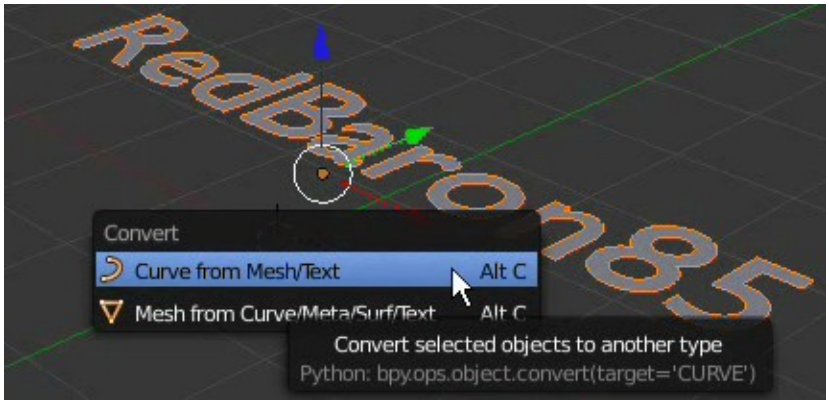
Iniziamo creando un oggetto *Text*, per cui inseriamolo nella scena e digitiamo il testo che preferiamo in Edit Mode. Alle basi dell'utilizzo dell'oggetto *Text* ho dedicato la diciannovesima lezione del mio corso di base su Blender 2.5, trascritto nel PDF "*Blender 3D tutorials – Volume 1*".



La creazione dell'oggetto TEXT

Come detto, l'importante è lasciare il *Text*, almeno in questa fase, bidimensionale.

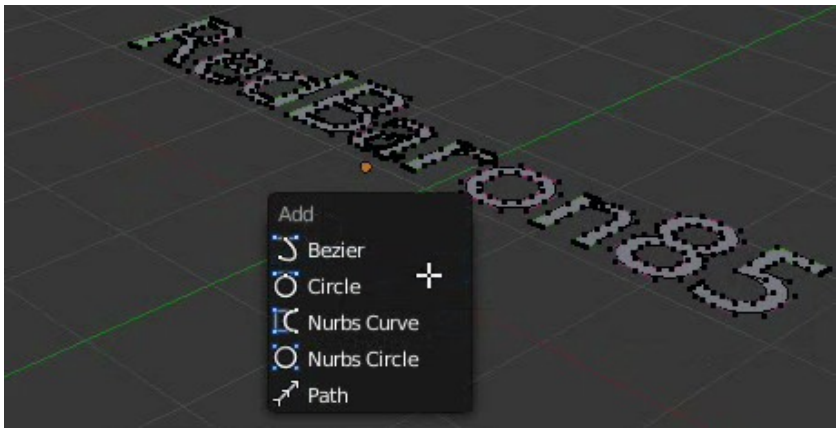
Un oggetto *Text* appena creato non è né una mesh né una Curve, ma proprio una tipologia a parte, "*Text*" appunto. Per convertirlo in *Curve*, premiamo *ALT C* mentre l'oggetto è selezionato in Object Mode e scegliamo *Curve from Mesh/Text*.



Conversione del Text in Curve con ALT C in Object Mode

A questo punto possiamo utilizzare ad esempio un modificatore *Curve*, inserendo una *Curva* o un *Cerchio Bezier*, per modificare ulteriormente l'oggetto, ma sempre mantenendolo *2D*.

Per realizzare l'incisione, passiamo in Edit Mode, con la nuova *Curve* che dovrebbe essere già in modalità *2D*, come possiamo verificare aprendo il pannello *Object Data* nella *Properties Window*, ed inseriamo un oggetto *Curve Bezier*: in questo modo, la curva Bezier verrà aggiunta allo stesso oggetto *Curve* creato a partire dal testo, e ponendo il testo all'interno di tale curva realizzeremo l'incisione.



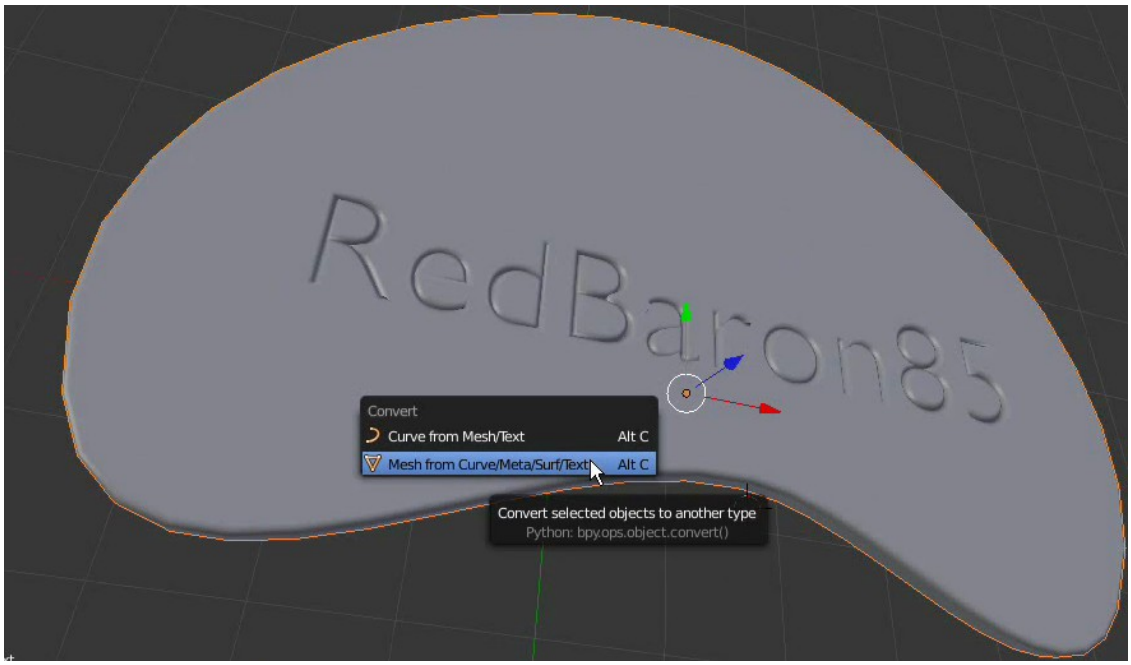
Aggiungere un Curve Bezier, in Edit Mode, all'oggetto curva-testo

Anche in questo caso possiamo divertirci a definire come vogliamo la forma degli oggetti, sia quella del cerchio *Bezier* creato, che quella della figura realizzata complessivamente; possiamo poi estrudere l'oggetto o smussarlo un po' con gli strumenti presenti in *Object Data*; tra l'altro, *Bevel*

può tornarci utile per riempire le fessure prodotte dall'incisione, in modo da non creare proprio dei "solchi" che oltrepassano da una parte all'altra l'oggetto, ma semplici incisioni, poco profonde, sulla sua superficie (anche se visibili dall'altra parte).

Operazioni e trasformazioni a piacere sugli oggetti

Possiamo quindi convertire tutto in Mesh, se preferiamo, sempre con *ALT C* in *Object Mode* ma scegliendo *Convert to Mesh*, ed utilizzare anche i modificatori propri di tale tipologia, se necessario.



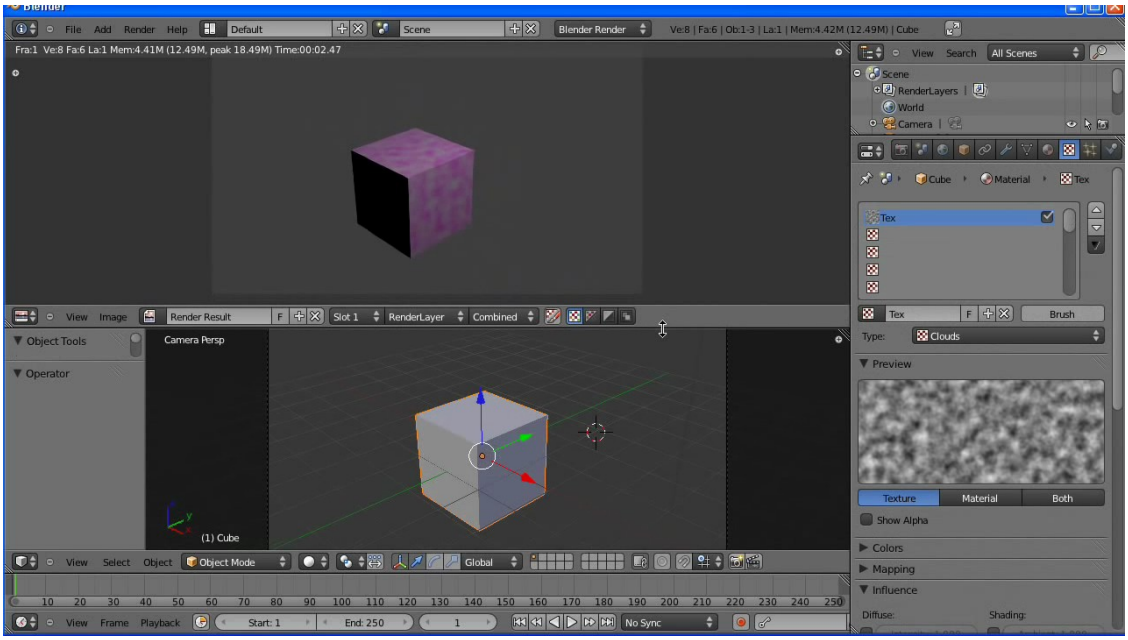
Conversione a mesh, con ALT C in Object Mode

* * *

Baking delle Textures procedurali, anche per il GE

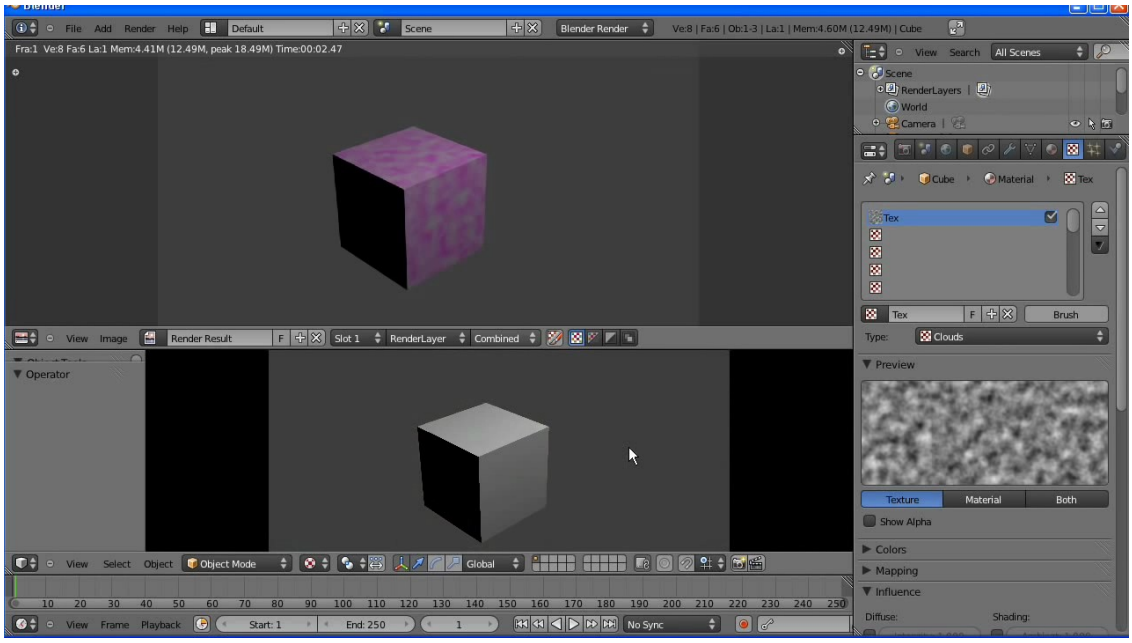
In questo breve tutorial parleremo dell'operazione di *baking* delle Textures di tipo procedurale in Blender 3D 2.5, soprattutto per vedere come evitare l'errore "*No objects or images found to bake to*" (letteralmente, "non sono stati trovati oggetti o immagini per il baking"); nel mentre, daremo anche un'occhiata a quello che succede col *Game Engine*, il motore di gioco di Blender.

Iniziamo con il motore di rendering standard di Blender, utilizzando un semplice cubo provvisto di Material di base per le nostre prove. Assegniamo al cubo una Texture procedurale, ad esempio *Clouds*, anche con le impostazioni di default: questo è solo un esempio. Avviando il rendering della scena col motore *Blender Internal*, con *F12*, notiamo che la Texture viene applicata secondo le impostazioni date.



Il rendering del cubo, provvisto di Texture CLOUDS, con Blender Internal: tutto va come previsto

Per fare le prove col *Game Engine*, passiamo a tale motore (mediante l'apposito selettore nella scheda *Info*), cambiamo la modalità di visualizzazione della *Viewport* in *Textured* e premiamo *P*, per avviare il gioco. Questa volta, come possiamo constatare nell'immagine seguente, le Textures non vengono applicate.

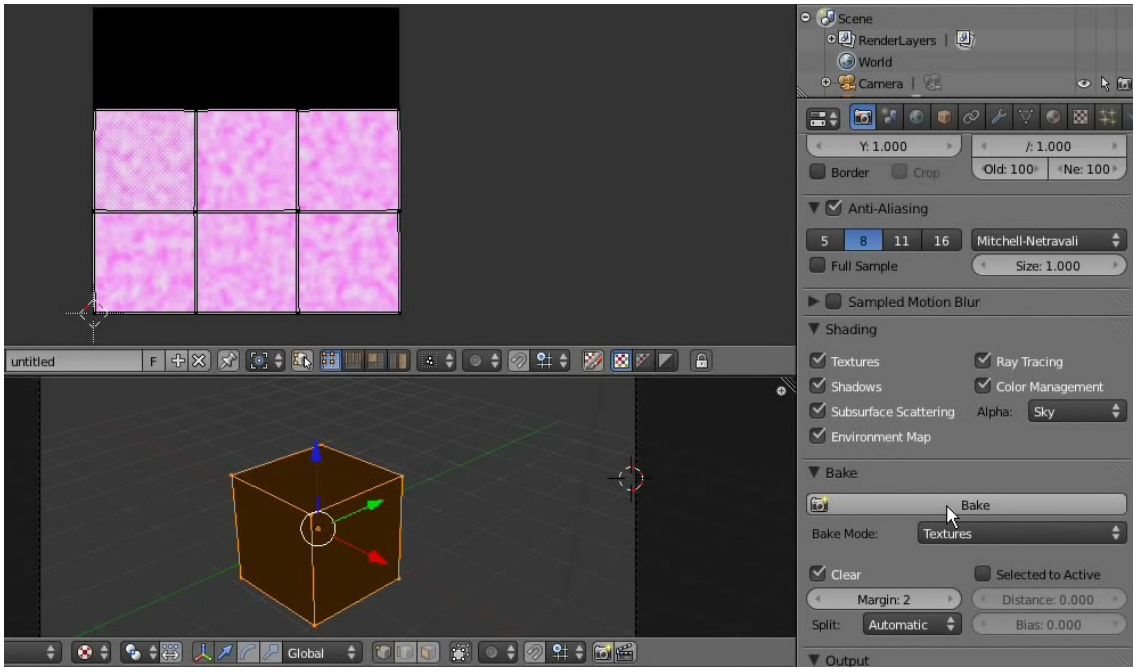


In Blender Game Engine, la Texture procedurale CLOUDS non viene visualizzata durante l'esecuzione

Se non abbiamo la necessità di utilizzare i *Materials GLSL* (per le ombre in tempo reale), possiamo procedere al *baking* delle Textures per il gioco; in ogni caso, il procedimento qui mostrato va bene anche per effettuare il *baking* delle Textures per *Blender Render*, se ad esempio non dobbiamo proiettare ombre di oggetti in movimento o variare l'illuminazione degli oggetti interessati.
Per effettuare il *baking*, seguiamo questa procedura:

- apriamo una finestra *UV - Editor delle immagini*;
- in una finestra *3D View*, con l'oggetto selezionato (nel nostro caso, il cubo) passiamo in *Edit Mode*, premiamo *U* per effettuare l'unwrap e scegliamo una voce, ad esempio *Smart Project*;
- torniamo alla finestra *UV - Editor delle immagini*, dove ora dovrebbe esserci, unwrappato sul nulla, il cubo, ed inseriamo una nuova immagine, con *Image - New*, anche con le impostazioni standard;
- a questo punto, nella scheda *Render* all'interno della *Properties Window*, apriamo la scheda *Bake*, cambiamo la modalità da *Full* a *Texture* (ma ovviamente possiamo provare con *Ambient Occlusion* o altri) e clicchiamo su *Bake*.

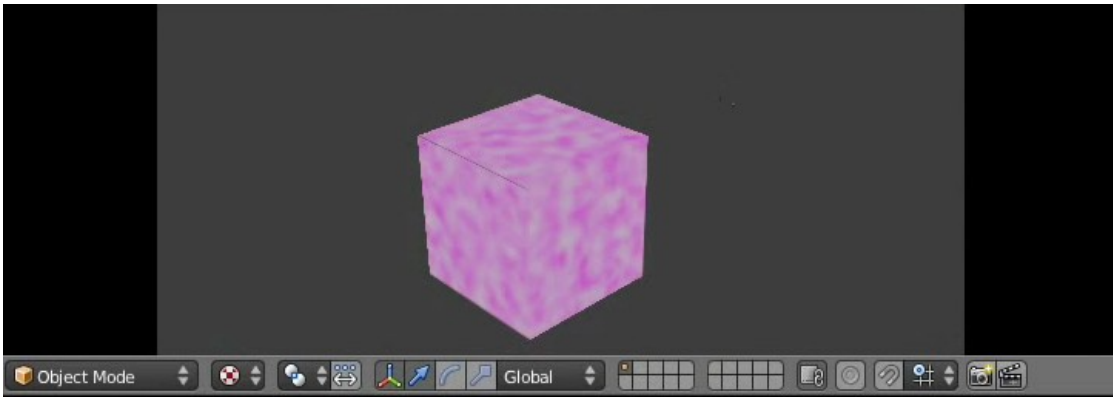
In questo caso, visto che l'oggetto è un semplice cubo con 6 facce ed è presente una sola fonte di luce, l'operazione di *Baking*, visibile in tempo reale nella finestra *UV – Editor delle immagini*, verrà effettuata in poco tempo, ma con progetti più complessi non preoccupatevi se notate dei rallentamenti.



L'operazione di Baking, visibile in tempo reale in una finestra UV

A questo punto possiamo anche avviare il gioco con *P* ed osservare la texturizzazione.

Attenzione, perché questo tipo di lavoro va bene, come anticipato, se l'oggetto non dev'essere messo in ombra e se l'illuminazione non cambia, e questo sia nel *Game Engine* che nel renderer standard; la procedura va bene (faccio un esempio davvero di base) se in un gioco con scenario all'aperto sono presenti oggetti o parti di edifici che non possono essere raggiunti dai giocatori e, in generale, non verranno messi in ombra da altri oggetti: in questo caso, e in casi simili, il *baking* delle Textures farà risparmiare un bel po' di calcoli al computer, rendendo più fluido il gioco.



Adesso la Texture (che non è più procedurale, ma una Texture immagine ottenuta col Baking e mappata con UV-Mapping) è visibile durante l'esecuzione del gioco

In ogni caso, l'aver effettuato il *bake* di una Texture procedurale, creando così un'immagine che possiamo anche salvare, nella finestra *UV – Editor delle immagini*, ci ha risolto, se vogliamo, anche il problema delle Textures procedurali in giochi con ombre in tempo reale, perché a questo punto è sufficiente effettuare la mappatura *UV* dell'immagine appena generata sull'oggetto, come si fa in genere con i giochi con ombreggiatura *GLSL* nel motore di gioco.

* * *

Lampi ed effetti luminosi col nodo Normal

In questo tutorial vedremo un modo per realizzare un effetto di illuminazione un po' particolare, per cercare di simulare effetti come l'illuminazione dovuta ai lampi durante un temporale, utilizzando il *Compositing dei Nodes* in Blender 3D 2.5.

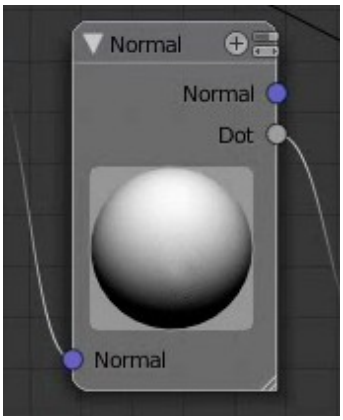
In realtà, l'effetto che realizzeremo non è una modalità vera e propria di illuminazione, in quanto non fa uso di fonti di luce o di sistemi di illuminazione globale e non proietta ombre; procediamo comunque con ordine.

La scena da notte buia e tempestosa visibile nello screenshot seguente, è davvero basilare: non c'è vento, non c'è la simulazione della pioggia e l'inquadratura è fissa, ma per il nostro esempio va più che bene. Nella scena non sono presenti fonti di luce e l'unica illuminazione, debolissima, è dovuta in realtà alla luce ambientale, di colore blu scuro come il colore del mondo, *World*.

La scena utilizzata per questo esempio

Aggiungere una fonte di luce che proietti ombre su tutta la scena è improponibile, perché per il calcolo del *Raytracing* Blender richiederebbe troppo tempo. Per ottenere l'effetto di un'illuminazione uniforme e "impulsiva" (dovuta, durante i temporali, ai lampi) ricorriamo quindi al *Nodo Normal* nel Compositing, in post-produzione.

Il Nodo *Normal* funziona in due modi: o introducendo una Normale o effettuando il prodotto scalare tra il vettore in arrivo e quello definito internamente al Nodo; nel nostro caso, lo utilizzeremo con la seconda modalità.



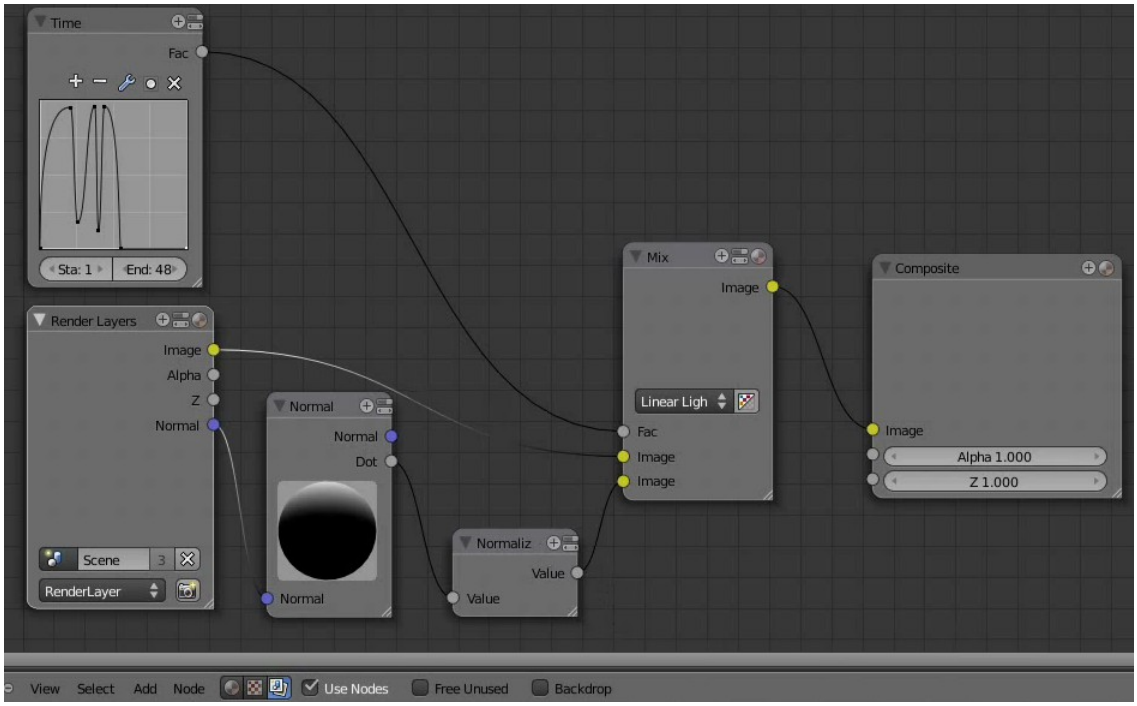
Il Nodo NORMAL, nel Compositing dei Nodes

Il verso del vettore interno va definito cliccando sulla sfera interna al Nodo e muovendo il mouse, rilasciando nel punto desiderato, cercando di capire il verso del vettore guardando la variazione dell'illuminazione della sfera; nel mio caso, il vettore punta dall'alto verso il basso, infatti la parte superiore della sfera è illuminata, il che si tradurrà in un'illuminazione dall'alto per ogni mesh presente nella scena.

In questo caso, l'effetto verrà applicato a tutte le mesh presenti nella scena, ma in alcuni tutorial precedenti e nel mio *ebook sul Compositing dei Nodes* spiego anche come realizzare degli effetti selettivi, utilizzando *ID Mask*.

Esaminiamo quindi l'editor dei Nodi configurato per la nostra scena: la moltiplicazione scalare avviene prelevando il canale di output *Normal* dal Nodo *Render Layers* e moltiplicandolo col vettore definito, come detto, nel Nodo *Normal*.

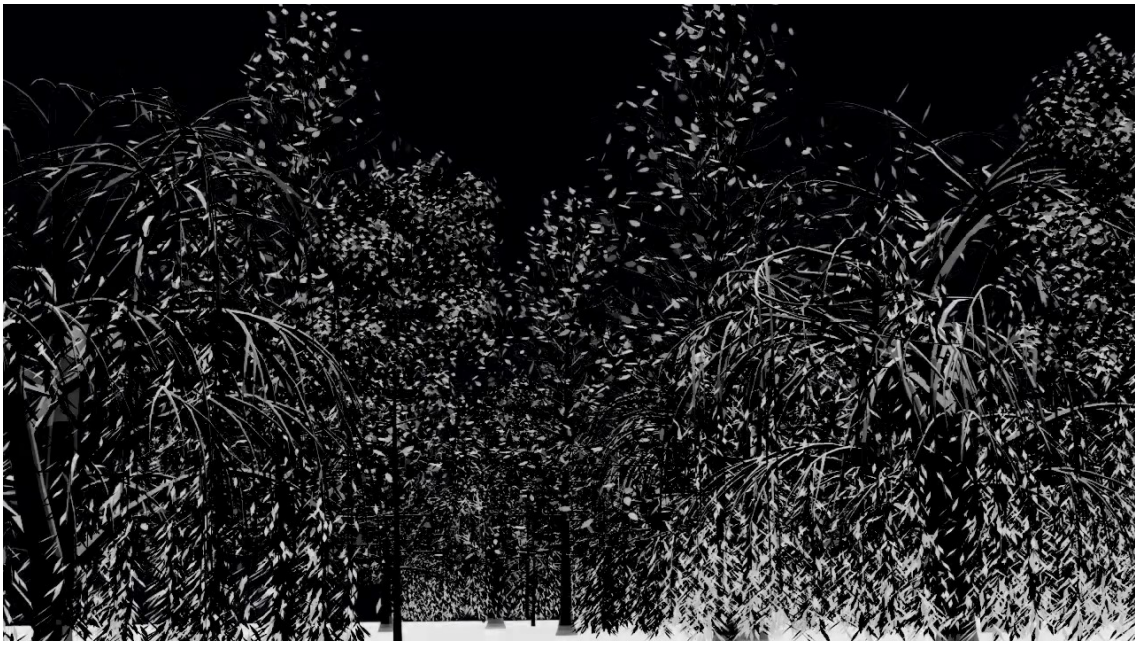
Il valore in uscita viene normalizzato e mescolato, con modalità *Linear Light*, all'immagine in arrivo da *Image* di *Render Layer*, cioè il rendering di base.



Lo schema di Nodi di Compositing implementato

Come potete notare, per realizzare l'animazione dell'intensità dell'effetto ho fatto uso di un Nodo *Time* con curva appositamente modellata, in modo da simulare un paio di lampi luminosi in rapida successione; dal momento che *Normal* non mette a disposizione un canale di ingresso di tipo *Factor* per modulare l'intensità o la normalizzazione del prodotto scalare mediante *Time* o una *Ramp*, ho passato l'output di *Time* direttamente al canale *Factor* del Nodo *Mix*, per modulare l'intensità della miscelazione dell'effetto (in pratica, quando l'intensità è 1, viene applicata la finta illuminazione supplementare dovuta a *Normal*, mentre quando è 0 l'immagine finale che arriva al Nodo *Composite* coincide sostanzialmente con quella in arrivo da *Render Layers*).

L'effetto finale (visibile nell'immagine nella pagina seguente) è effettivamente quello di un'illuminazione un po' spettrale, buona anche per semplici immagini statiche come illustrazioni; per quanto riguarda l'animazione, è una buona base di partenza, alla quale aggiungere ad esempio le foglie in movimento e un paio di effetti sonori.



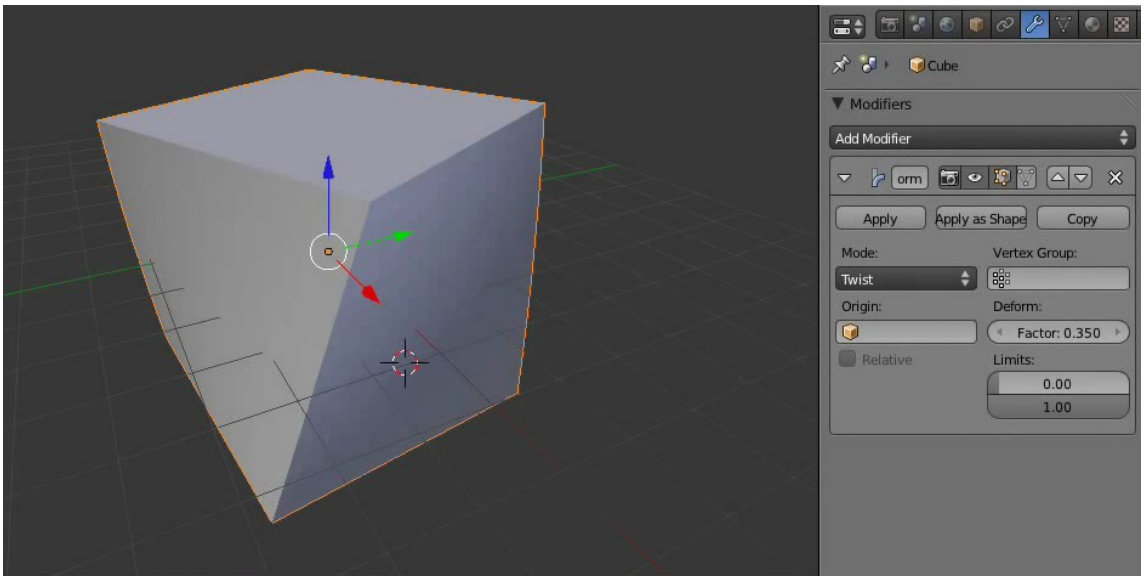
Screenshot dell'effetto finale

* * *

Modifiers animati, animare i modificatori

Questo breve tutorial nasce da una richiesta di un utente del sito, *Wox76*, che mi ha chiesto se sia possibile animare i modificatori in Blender 3D 2.5. La risposta è sì e in questo tutorial vedremo appunto come fare, approfittandone per fare qualche considerazione.

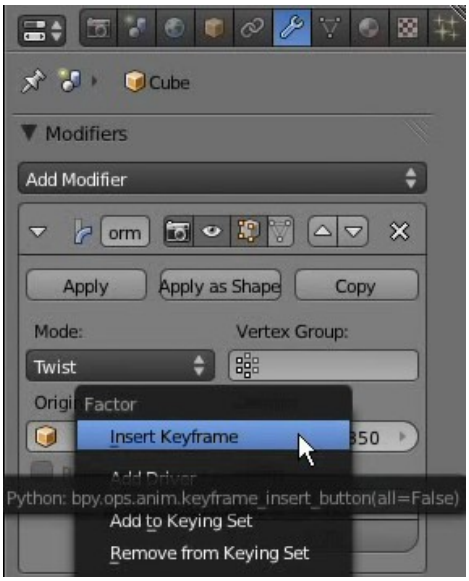
Prendiamo un oggetto, ad esempio un semplice Cube suddiviso un paio di volte (per migliorare la qualità dell'effetto) ed associamogli un *Modifier*, ad esempio *Simple Deform* con effetto *Twist* e le impostazioni di default.



Il cubo che utilizzeremo in questo tutorial, con il Modifier Simple Deform Twist

Per un parametro di un modificatore, come per quasi tutto ciò che c'è in una scena di Blender, è possibile creare più keyframes, letteralmente “fotogrammi chiave”, memorizzando il valore assunto da un parametro in un dato fotogramma dell'animazione, e questo per vari fotogrammi, lasciando a Blender il compito di realizzare automaticamente l'interpolazione dei valori tra due fotogrammi chiave e, quindi, l'animazione nel corso di un certo numero di frames.

Se ci pensate, mettere due fotogrammi per spostare un oggetto da un punto ad un altro significa mettere due keyframes sui parametri *Location X*, *Y* e *Z* dell'oggetto, per cui allo stesso modo possiamo mettere dei keyframes sul parametro *Factor* di *Simple Deform – Twist*.



Inserimento di un Keyframe per il parametro di un modificatore

Per far ciò, possiamo posizionare il cursore del mouse sul pulsante *Factor* nella scheda del *Modifier* e premere *I*, inserendo il keyframe al fotogramma desiderato, oppure fare click col tasto destro del mouse sul pulsante e scegliere "*Insert Keyframe*" dal menù che apparirà a video.

Il pulsante *Factor* cambierà colore, per indicare la presenza di un keyframe.

Un keyframe può essere rimosso facendo click col tasto destro del mouse sul pulsante che lo possiede e scegliendo *Delete Keyframe* dal menù che apparirà a video.

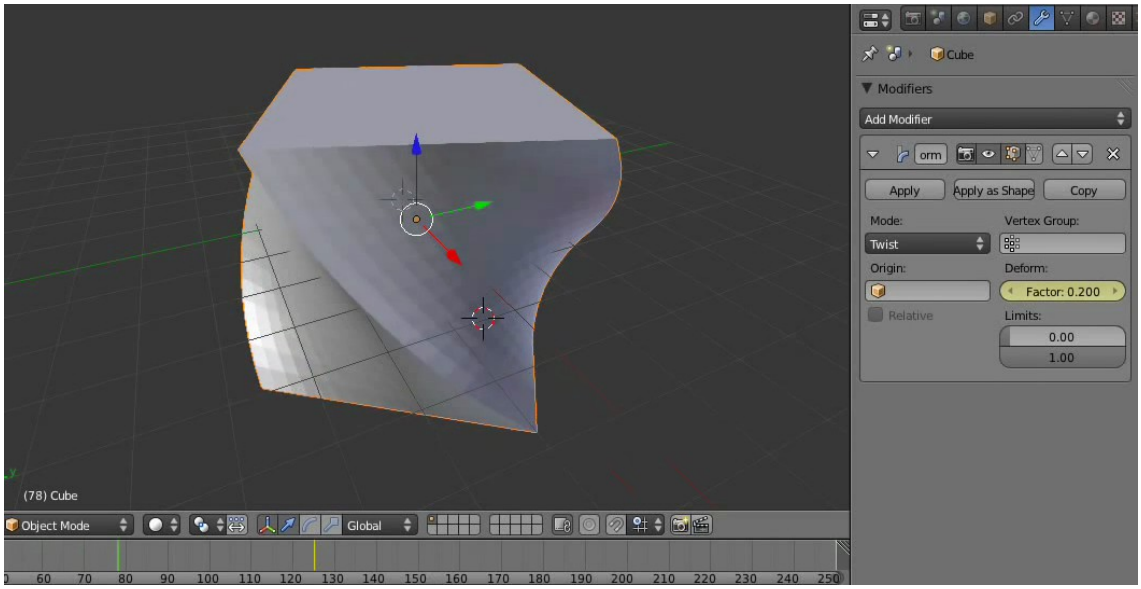
Inserimento di un Keyframe per il Factor di Simple Deform Twist premendo "I" mentre il cursore del mouse si trova su tale parametro

Facciamo una prova con *Twist*.
Posizioniamoci al frame 1, come di default, inseriamo il valore 0.2 per *Factor* e premiamo *I* per inserire un keyframe.

A questo punto andiamo al frame 125, ad esempio, inseriamo -2 per *Factor* e un keyframe con *I*.

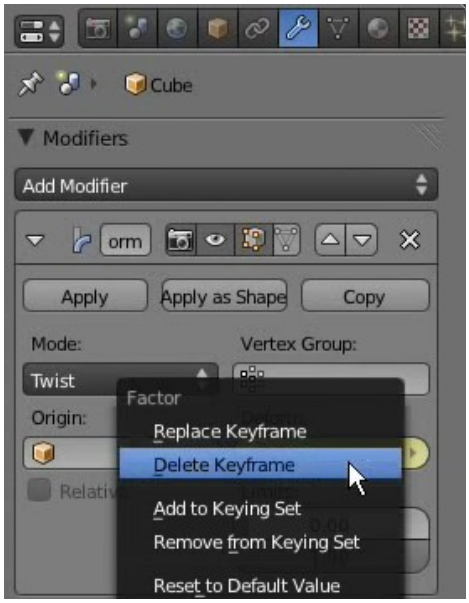
Passiamo al frame 250, inseriamo +2 per *Factor* e un keyframe con *I*.

Ricapitolando, *Twist* passerà da 0.2 a -2 nei primi 125 frames e da -2 a 2 nei successivi 125, "torcendo" il cubo nel verso opposto; possiamo avere una conferma di quanto detto premendo *ALT A* per visualizzare un'anteprima dell'animazione.



Screenshot dell'anteprima dell'animazione

Ho inserito il valore -2 a metà animazione per mostrarvi cosa succede togliendo un keyframe: posizioniamoci quindi al frame 125, facciamo click col tasto destro su *Factor* e scegliamo *Delete Keyframe*. Togliendo questo keyframe, abbiamo lasciato un keyframe a 1, con valore 0.2, e un keyframe a 250, con valore 2.0, per cui Blender ri-calcolerà l'interpolazione dei valori e questa volta Factor passerà da 0.2 a 2.0 nell'arco dei 250 frames.



Rimozione di un Keyframe per un parametro

Questo discorso vale, in generale, quando si toglie un qualsiasi keyframe posto tra altri due: se nella sequenza *A*-*B*-*C* si toglie *B*, viene ricalcolata l'interpolazione tra i frames *A* e *C* considerando, come valori estremi per l'animazione, i valori presenti appunto in *A* e *C*.

In questo caso, con *Twist*, abbiamo utilizzato un parametro, *Factor*, che accetta valori decimali, ottenendo un'interpolazione lineare tra gli estremi registrati nei keyframes, ma altri modificatori, come *Subdivision Surface*, accettano solo valori interi; in questo caso, la transizione da un keyframe all'altro sarà brusca, come un gradino, per cui potremo avere, ad esempio, valore 2 al frame 25 e valore 4 al frame 26, se abbiamo impostato in questo modo i valori dei keyframes.

Da notare che alcuni simulatori, come *Cloth*, sono dei veri e propri *modifiers* animati, solo che i parametri vengono animati automaticamente da Blender in base ai valori restituiti da un algoritmo, per cui noi in quel caso dobbiamo posizionarci in un'apposita scheda, ad esempio *Cloth*, inserire i parametri della simulazione e lasciare a Blender il compito di calcolare i valori dei risultati, dell'interpolazione e, quindi, dell'animazione.

Che succede se ho un modificatore animato e premo *"Apply"* nella scheda dello stesso?
Succede che a quel punto perdiamo la possibilità di variare i parametri del modificatore e l'animazione, applicando tra l'altro la deformazione visibile al fotogramma corrente a TUTTI i fotogrammi dell'animazione.

Questo non è necessariamente un male; io ad esempio creo tende e tovaglie (dove le pieghe a volte sono difficili da fare) con questo sistema, ossia: predispongo la simulazione del tessuto, avvio, osservo l'animazione prodotta (il modo in cui il tessuto si piega e deforma) e, quando sono soddisfatto del risultato, applico il modificatore, in modo da avere quella mesh "congelata" in quella posizione... ma *Cloth* è decisamente un altro argomento.

Bisognerebbe parlare anche delle maniglie nelle *curve IPO*, ad esempio per cambiare la forma dell'interpolazione o ripetere un'animazione, ma a questo argomento ho già dedicato altri tutorials presenti in questa raccolta (ad esempio, quello delle luci natalizie intermittenti con *Halo Light* animato), per cui, per questo tutorial, ci fermiamo qui.

* * *

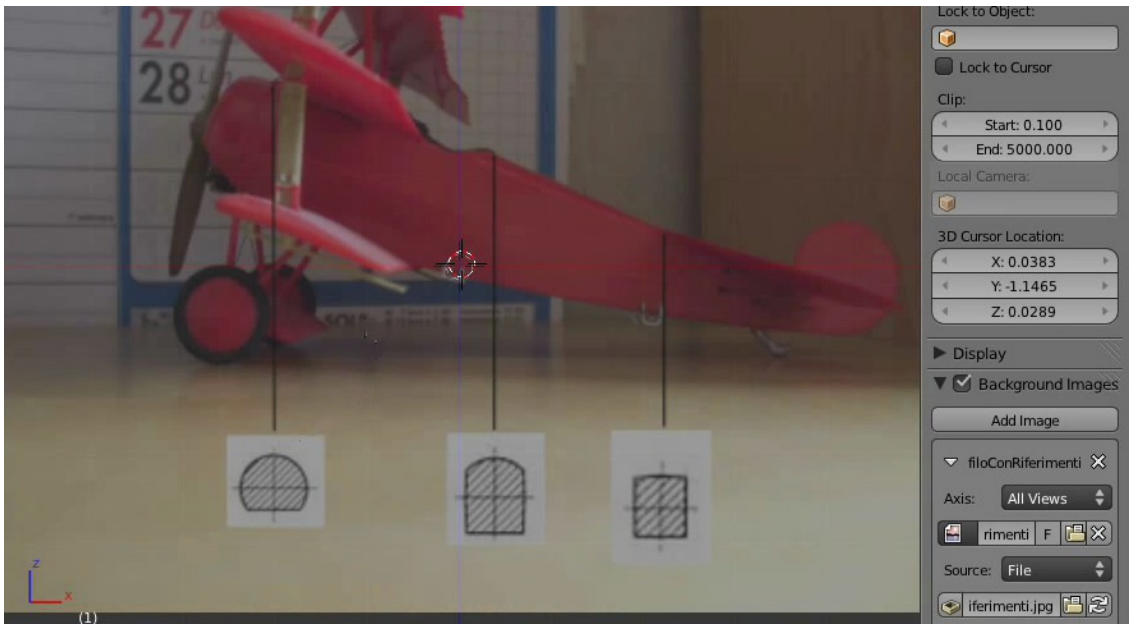
La fusoliera del Fokker DR.1 con le NURBS Surfaces

In questo tutorial vedremo come utilizzare lo *skinning*, ovvero il rivestimento realizzato mediante superfici *NURBS*, per realizzare la fusoliera di un aereo, ed in particolare l'aereo del Barone Rosso, il *Fokker Dr. I*, in Blender 3D 2.5. L'immagine mostra un modellino "reale" di tale aereo.

Un modellino del Fokker DR.I

La particolarità di questa fusoliera (e di altri oggetti, in generale) è che le sezioni variano considerevolmente di forma. Per modellare un oggetto del genere è possibile ricorrere a varie tecniche, ad esempio all'utilizzo di un oggetto *Lattice*, ma personalmente mi trovo bene con le *NURBS Surfaces*.

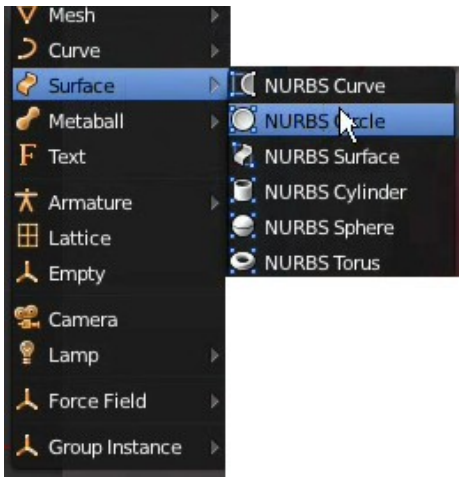
Nella finestra di Blender ho caricato, come *Background Image*, una foto scattata quasi di profilo al mio modellino, aggiungendo poi con un programma di fotoritocco le immagini delle tre sezioni della fusoliera, indicandone le posizioni nel modello.



L'immagine Background Image utilizzata come riferimento

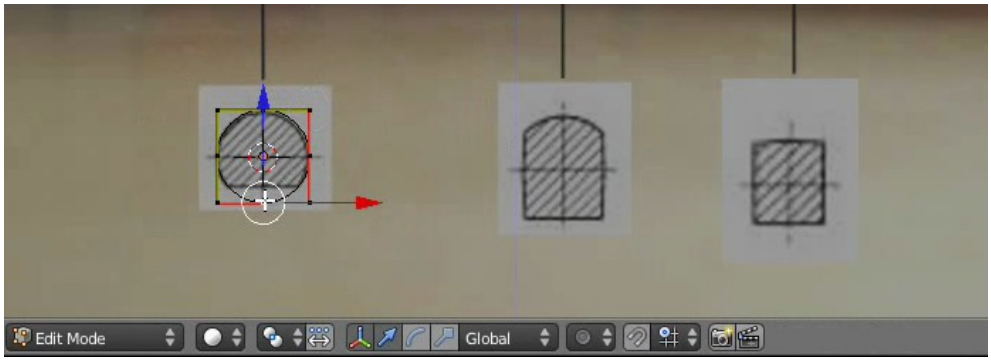
La modellazione visibile in questo tutorial non sarà proprio accurata: non terrò conto delle misure precise del modello perché, più che altro, voglio parlare dello *skinning*.

Aggiungiamo una *NURBS Surface* di tipo *Circle*, e qui facciamo attenzione: deve trattarsi di una *NURBS Surface*, non di una *NURBS Curve*, altrimenti non potremo realizzare lo skinning.



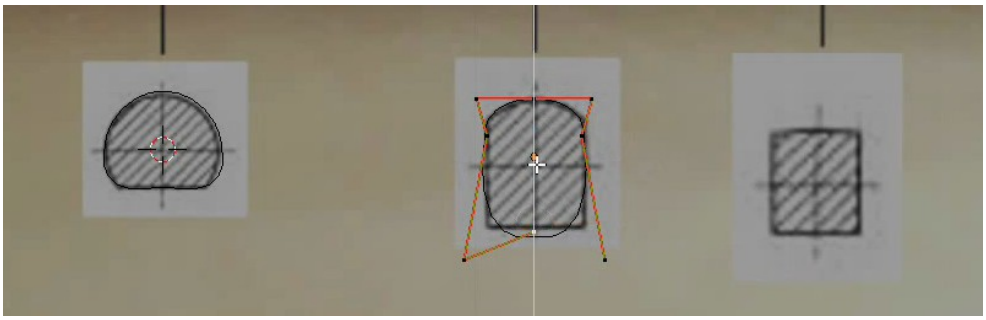
Inserimento di una superficie NURBS (attenzione: deve essere una Surface, non una Curve!)

Deformiamo tale oggetto seguendo l'immagine della sezione. In questo caso non sto dedicando troppo tempo alla modellazione, utilizzando solo i punti di controllo di default ed ottenendo un risultato non proprio eccezionale, ma non è importante.



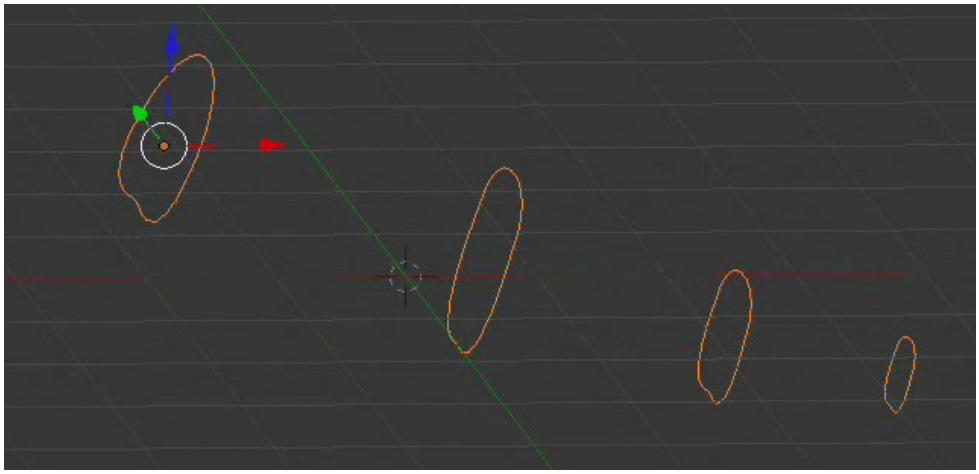
Definizione della prima sezione

Per creare la seconda sagoma, duplichiamo la prima *NURBS Surface* con *SHIFT D* in Object Mode, in modo da ottenere un oggetto distinto dall'originale, dopodiché spostiamo i punti di controllo in Edit Mode per ottenere il profilo desiderato.



Definizione della seconda sezione, partendo da una copia realizzata in Object Mode, in modo da ottenere un oggetto distinto dall'originale

Effettuiamo quest'operazione anche per la terza sagoma e, se vogliamo, anche per una quarta, più piccola e da mettere proprio in coda al modello, per chiuderlo il più possibile. Per realizzare il resto del modellino è necessario fare un bel po' di pezzi, ma in questo tutorial esamineremo solo la fusoliera. A questo punto posizioniamo le 3 superfici nello spazio orientandoci con la foto.

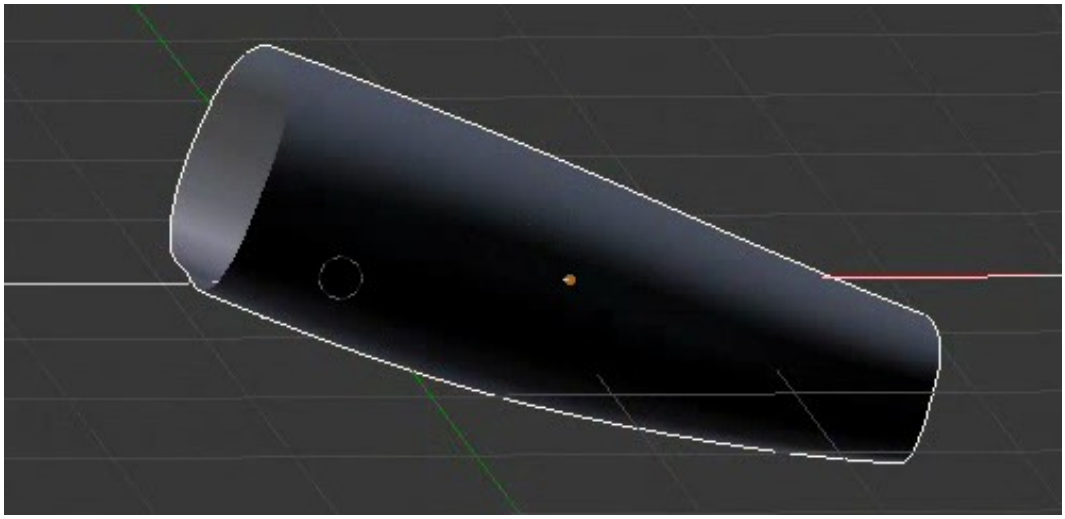


Le tre superfici disposte nello spazio, seguendo la Background Image di riferimento

Per poter implementare lo *skinning*, ogni curva componente deve avere lo stesso numero di punti di controllo delle altre; questo è chiaramente il nostro caso, visto che abbiamo creato le varie *NURBS Surfaces* a partire da quella originale senza aggiungere o togliere punti di controllo, ma è importante fare questa precisazione.

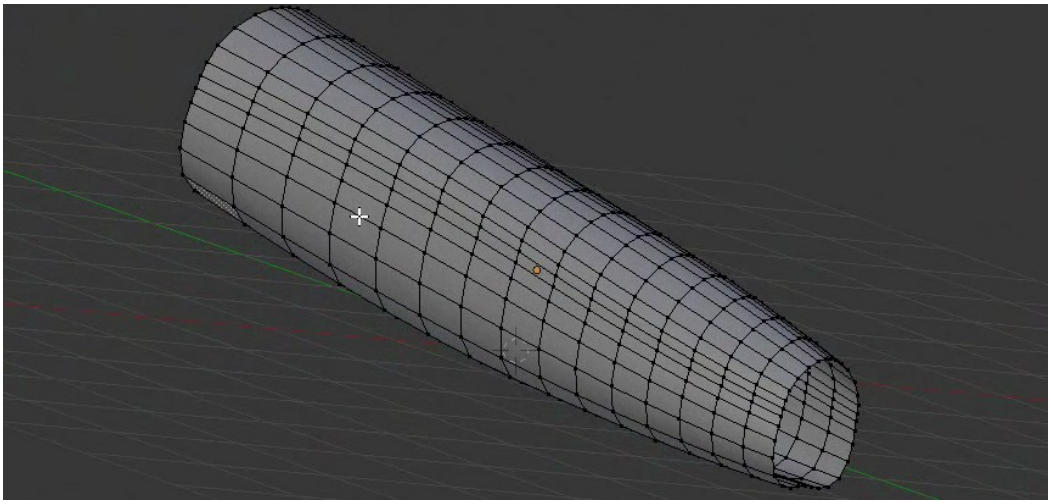
Selezioniamo quindi i tre (o quattro, o più) oggetti contemporaneamente, con una selezione multipla, ed uniamoli in uno solo con *CTRL J*, per *Join* (unione, appunto), dopodiché passiamo in Edit Mode, selezioniamo tutto (con *A*) e premiamo *F*, realizzando così il rivestimento.

Da notare che i punti di controllo sono ancora disponibili e possiamo modificarli, deformando con semplicità l'intero oggetto.



Lo Skinning, ottenuto premendo F (FILL) dopo aver unito le Surfaces in un unico oggetto, con CTRL J, e selezionato tutti i punti di controllo in Edit Mode.
I punti di controllo sono ancora selezionabili e modificabili in Edit Mode.

In Object Mode possiamo poi scalare la superficie *NURBS* rispetto ad uno o più assi, ruotarla e traslarla; se lo preferiamo, possiamo convertire il tutto in mesh con *ALT C* e scegliendo "*Convert To Mesh*" dal menù che apparirà a video; ciò può essere utile, ad esempio, per applicare modificatori propri delle mesh, non disponibili con le *NURBS*, ma attenzione, perché a questo punto non sarà più possibile deformare i punti di controllo come con una *NURBS*, per cui (se questa conversione è proprio necessaria) effettuatela quando avete finito con gli strumenti *NURBS*.



L'oggetto ottenuto dopo la conversione in mesh (con ALT C sull'oggetto NURBS in Object Mode)

* * *

Effetti speciali: ologrammi virtuali in filmati reali

Questo è un piccolo "dietro le quinte" di un videoclip di pochi secondi dove sono presenti delle animazioni realizzate in Blender integrate a riprese per così dire "reali".

Screenshot del videoclip realizzato

In realtà, la realizzazione di questo effetto è davvero semplice (è molto più difficile, ad esempio, animare in maniera credibile un personaggio); in particolare, in questo caso non si ha il passaggio di mani o altri elementi reali sulle immagini prodotte in Blender, per cui non c'è bisogno di creare maschere per l'*Alpha Over* o altri trucchetti, anzi non ho dovuto utilizzare neanche l'editor dei Nodi per il Compositing. Si tratta, più che altro, di un lavoro abbastanza noioso basato sulla sincronizzazione tra quello che avviene a video e le immagini da creare in Blender.

Per prima cosa, bisogna avere ben chiaro in testa cos'è che si vuole ottenere: sono partito dall'idea di giocare con un ologramma 3D, per cui ho definito un piccolo copione che prevedeva l'apparizione del modello 3D, la rotazione e lo scaling dello stesso, il cambio delle Texture e l'animazione finale.

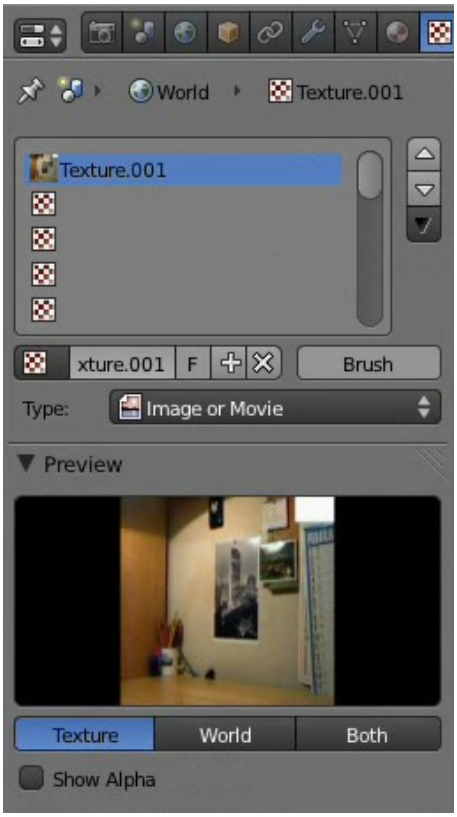
Le *Background Images* di Blender, trattate nell'undicesima puntata del mio videocorso di base, hanno due caratteristiche molto interessanti: possono essere proiettate anche come sfondo per ciò che è inquadrato dalla telecamera virtuale di Blender e, se si tratta di filmati, vengono letteralmente eseguite fotogramma per fotogramma se ci si sposta tra i frame della *Timeline* in Blender.

La prima cosa da fare è, quindi, impostare come *Background Image* della vista telecamera il filmato "grezzo", quello con le sole riprese reali. In questo modo possiamo visualizzare lo sfondo "in diretta", muovendoci anche tra i vari frames dell'animazione osservando i relativi fotogrammi nel filmato. Consiglio a questo punto di "centrare" la griglia di riferimento dell'universo virtuale con un piano o altri oggetti del filmato; nel mio caso, sto utilizzando il tavolino.

Utilizzo dei fotogrammi del filmato come Background Image, da utilizzare come riferimento per inserire e animare gli oggetti nella scena 3D

Altro consiglio è quello di salvare periodicamente il lavoro e di non fare troppi salti nell'animazione, perché la Background Image di tipo filmato "pesa" parecchio e Blender è andato in crash un paio di volte durante la realizzazione del video; conviene quindi scrivere in un foglio i fotogrammi corrispondenti alle varie animazioni, come ad esempio la "pressione" delle sfere e il cambiamento nell'effetto *Halo*, e attivare la *Background Image* solo per effettuare delle verifiche, tenendola disattivata negli altri casi.

Sappiamo benissimo, però, che questa *Background Image* non verrà renderizzata a video… ma non è un problema: tutto quello che dobbiamo fare per far corrispondere quello che vediamo nella *3D View* allo sfondo del rendering finale è caricare una Texture per il mondo, *World*.



Aggiungere una Texture "Image or Movie" (immagine o filmato, appunto) per World, l'universo virtuale della scena

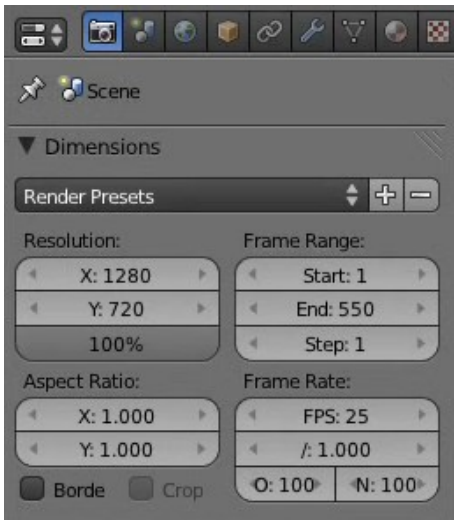
Apriamo quindi *World* nella *Properties Window*, dopodiché spostiamoci in Texture e aggiungiamo una Texture di tipo immagine, che in questo modo verrà associata a *World*.

La Texture, di tipo *Image or Movie*, sarà chiaramente il nostro filmato base, ma per utilizzarlo correttamente come sfondo per la nostra animazione dobbiamo fare un paio di cose:

- attiviamo la casella *Horizon* nella sezione *Influence* della Texture, in modo da coprire il colore del mondo con i fotogrammi del filmato;
- in *Image Mapping*, selezioniamo la modalità *Clip* in *Extension* (in realtà non dovrebbe esserci bisogno di effettuare quest'operazione, perlomeno in questo caso, perché useremo una finestra di rendering grande quanto il filmato, come vedremo tra poco);
- in *Image*, clicchiamo su *Match Movie Length* per far sì che vengano eseguiti tutti i fotogrammi del filmato, non solo i primi 100, come avviene di default;
- torniamo alla scheda *World* e clicchiamo su *World – Paper Sky*, in modo da "appiattire" l'immagine da utilizzare come sfondo.

Per ottenere una corrispondenza completa tra il filmato e l'area renderizzata, apriamo la scheda *Render* nella *Properties Window* ed impostiamo come risoluzione di output la stessa risoluzione del filmato in arrivo, che nel mio caso è 1280x720, facendo attenzione ad impostare 100% nella casella sotto *Y Resolution*. Abbiamo ottenuto una corrispondenza perfetta tra la *Background Image* della telecamera e lo sfondo dell'universo virtuale.

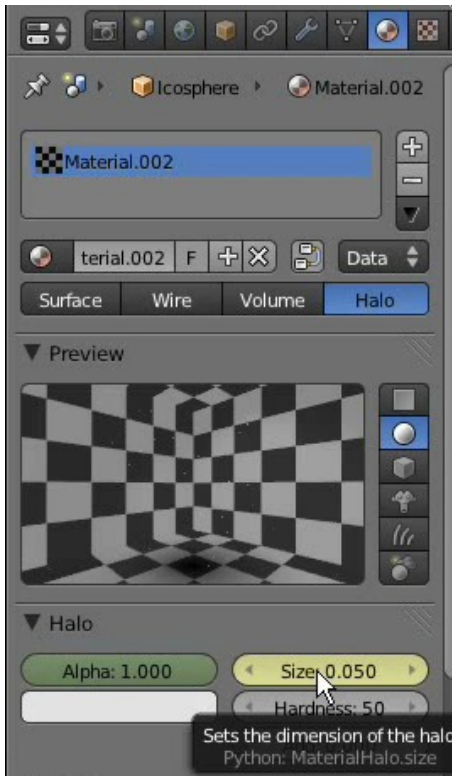
Parliamo ancora dei fotogrammi del filmato: dobbiamo portare la lunghezza dell'animazione da realizzare in Blender al numero di fotogrammi del filmato ed utilizzare il giusto frame rate, il numero di fotogrammi al secondo; per far ciò, sempre in *Render* all'interno della *Properties Window*, portiamo i valori di *End Frame* e *FPS* rispettivamente al numero di fotogrammi e al frame rate del filmato originale: nel mio caso, 550 e 25.



Le voci della scheda Render, nella Properties Window

Adesso la corrispondenza tra i filmati è praticamente perfetta e possiamo dedicarci agli elementi virtuali. Questo lavoro è decisamente noioso e il concetto di fondo è lo stesso per tutti gli elementi, siano essi bagliori *halo* o dimensioni del modello 3D: dobbiamo inserire dei keyframes ogniqualvolta si verifica un cambiamento.

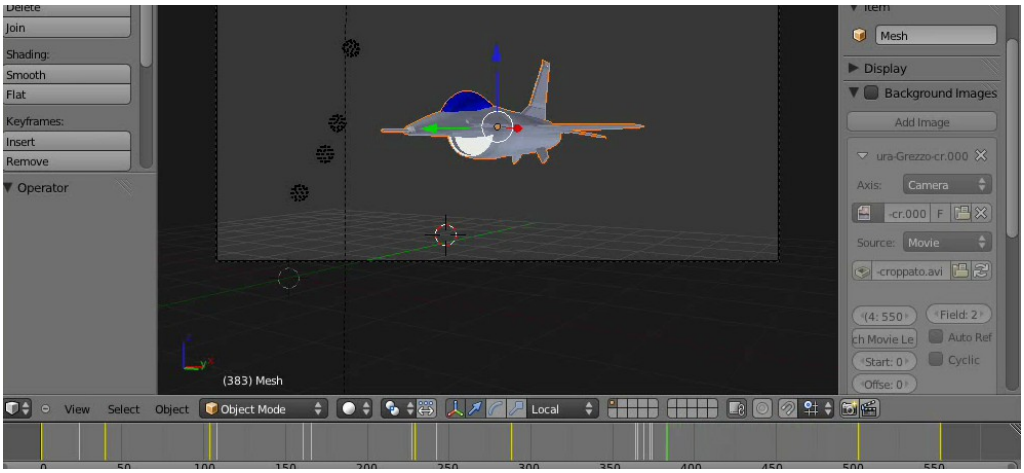
All'inserimento delle *chiavi IPO* per *LocRotScale* o per i Materials ho già dedicato diversi tutorials, per cui non tornerò sulle basi dell'argomento, mostrandovi invece il loro utilizzo in questo particolare caso. Prendiamo ad esempio la prima sfera, dotata di Material di tipo *Halo*: nel filmato, "premo" quel pulsante al fotogramma 37, per cui proprio al fotogramma 37 imposterò un keyframe sul canale *Halo Size* del Material, che ha valore iniziale 0.05, premendo *I* mentre il mouse è su tale pulsante, dopodiché passerò al frame 40, ad esempio, per portare *Halo Size* a 0.1 e inserire un nuovo keyframe. In questo modo, il bagliore Halo sarà meno intenso per i primi 37 frame, passerà da 0.05 a 0.1 in 3 frames (in maniera praticamente istantanea, quindi) e da quel punto in poi resterà a 0.1, a meno di non inserire in seguito altri keyframe.



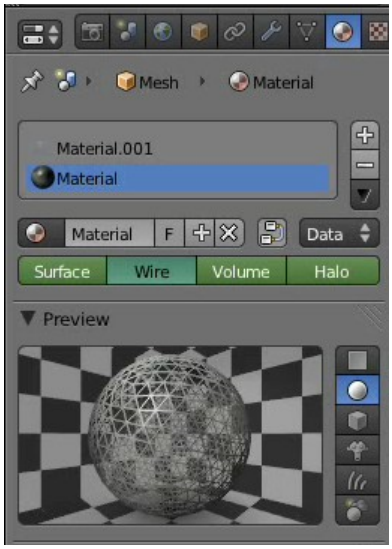
Inserimento di un Keyframe per un parametro di un oggetto

Per far sparire le tre sfere, alla fine del filmato (fotogramma 498, per la precisione) ho utilizzato dei keyframe sul canale *Alpha* dei vari Materials, portandolo da 1.0 a 0.0, rendendo di fatto trasparenti le sfere. Le *IPO* sulla rotazione e lo scaling del modello 3D sono davvero banali: il principio è lo

stesso dell'*Halo*, basta premere *I* e selezionare *LocRotScale* per "registrare" una configurazione, passare ad un altro fotogramma, effettuare una trasformazione (ad esempio lo scaling, con *S*) e premere *I*, scegliendo ancora una volta *LocRotScale*, per registrare la nuova configurazione, e così via: Blender farà per noi le interpolazioni (e, quindi, le animazioni) tra i vari fotogrammi.



Inserimento di un Keyframe Loc-Rot-Scale (posizione, orientamento e dimensione) per un oggetto di scena



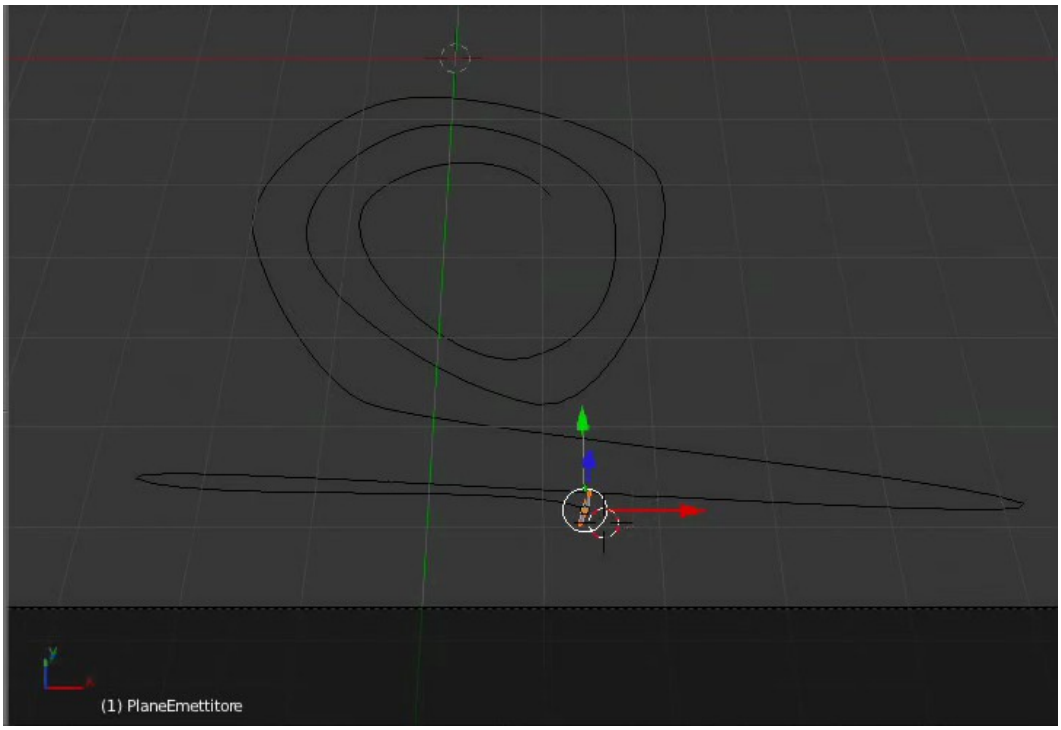
Inserimento di un Keyframe per il passaggio di un Material dalla modalità Wireframe a Surface

Altra cosa interessante è la possibilità di animare la modalità di visualizzazione della Texture, ossia il passaggio da *Wireframe* a *Surface*, che si ottiene al frame 372… no, in realtà non è interessante: semplicemente, si parte con la modalità *Wireframe*, dopodiché al frame 372 si preme *I* mentre il cursore del mouse si trova sulla casella *Wire* per dire a Blender "utilizza Wireframe dal frame 1 a 372"; a questo punto, si passa al frame 373, si seleziona *Surface* e si inserisce un nuovo keyframe per dire a Blender "utilizza Surface dal frame 373 fino a nuovo ordine", nuovo ordine che non c'è, quindi *Surface* verrà utilizzato fino alla fine del filmato. Ho animato anche la trasparenza del tettuccio dell'aereo, inserendo dei keyframes per attivare la trasparenza in sé (quindi con *I* mentre il mouse era sulla casella *Transparency*) e portare il valore di *Alpha* da 1 a 0; il concetto di fondo è sempre lo stesso, come vi dicevo si tratta più che altro di un noioso lavoro di temporizzazione tra la ripresa reale e gli eventi che devono verificarsi nella scena virtuale.

* * *

Particelle lungo un percorso con il Force Field Curve Guide

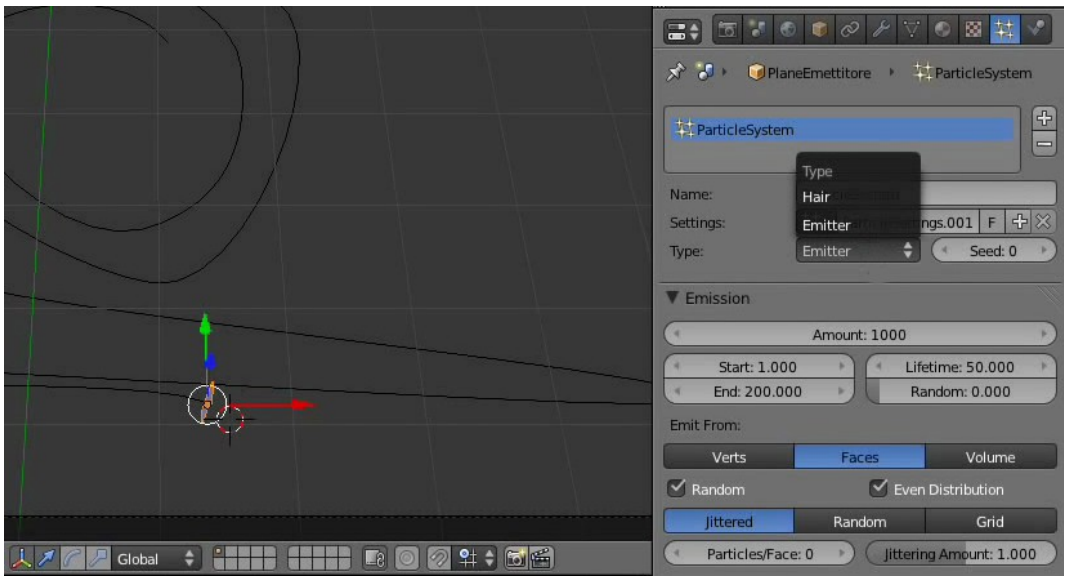
In questo breve tutorial vedremo come far seguire alle particelle emesse con *Emitter* un percorso, definito con una *Curva Bezier*, mediante il campo di forza *Curve Guide* in Blender 3D 2.5; si tratta di una tecnica utile per realizzare, ad esempio, loghi animati o altri effetti.



La scena che utilizzeremo per questo tutorial

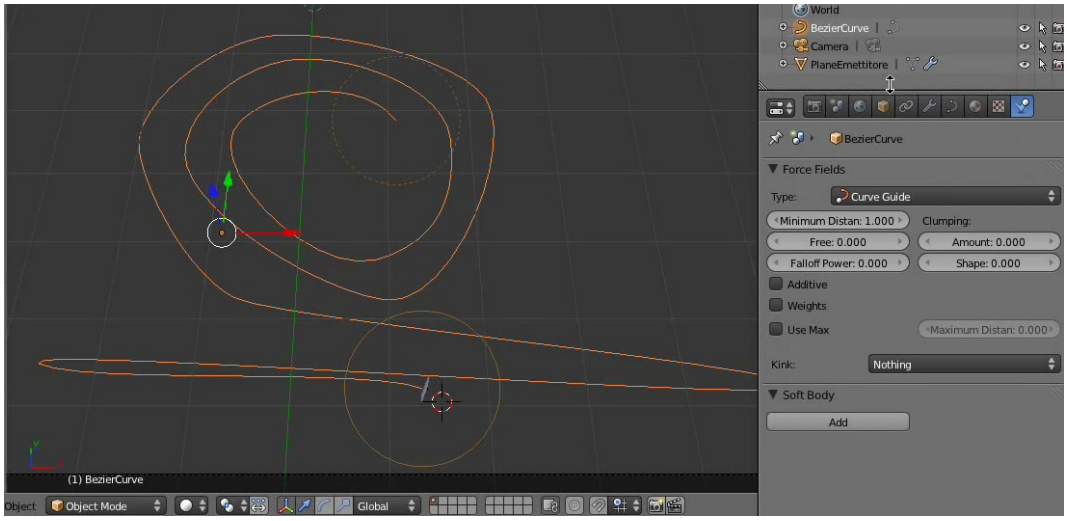
Ho creato due oggetti: una curva, che definisce il percorso da far seguire alle particelle, e un Plane, l'oggetto emettitore; niente di particolarmente complicato: la curva è una semplice *Bezier*, mentre il Plane è, appunto, un Plane, al quale dobbiamo assegnare ora un sistema particellare.

Con tale oggetto selezionato, quindi, apriamo la scheda *Particles* all'interno della *Properties Window* e clicchiamo su + per aggiungere un nuovo sistema particellare; di default, verrà creato un *Emitter*, che ci va più che bene. Impostate la quantità di particelle da creare che preferite con *Amount* e il loro aspetto in *Display* (io sto lasciando le impostazioni di default per concentrarmi su *Curve Guide*).



Il sistema particellare per il Plane emettitore

Di base, le particelle avranno simulazione fisica Newtoniana, per cui avviando ora un'anteprima dell'animazione, con *ALT A* mentre il mouse è su una finestra 3D, vedremo le particelle cadere nel vuoto. Non dobbiamo modificare il sistema particellare, né aggiungere *Constraints* o altro; dobbiamo, invece, selezionare la *Curva Bezier*, spostarci nella scheda *Physics* all'interno della *Properties Window* e aggiungere alla curva un campo di forza di tipo *Curve Guide*, lasciando in questo caso le opzioni di default, che vanno più che bene.



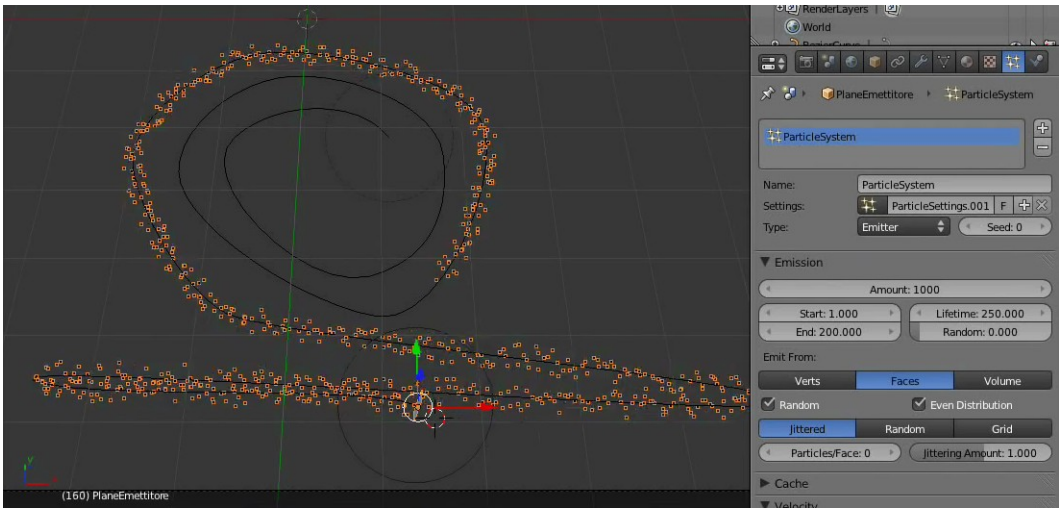
Il Force Field Curve Guide per la curva Bezier

Nella *3D View* appariranno due cerchi, sulla curva: il primo, con linea continua, indica il punto iniziale, mentre il secondo, con contorno tratteggiato, indica il punto finale; in questo caso la curva è aperta, ma l'effetto funziona anche sulle curve chiuse.

Il sistema particellare *Emitter* avrà, di default, valore 1 per il campo *Curve Guide* in *Field Weights* e non dobbiamo vincolarlo ad uno specifico *Force Field*, visto che ce n'è uno solo, per cui teoricamente già ci siamo...

...praticamente, no: avviando infatti l'animazione con *ALT A*, noteremo che le particelle completeranno il loro percorso troppo in fretta.

Quanto in fretta? In circa 50 frames che, guarda caso, è il valore di *Lifetime* del sistema particellare *Emitter*, per cui cambiamo questo valore, ad esempio, in 250 (la durata dell'animazione di default di Blender) e premiamo nuovamente *ALT A* per osservare il risultato: adesso ci siamo.



Screenshot dell'animazione ottenuta (anteprima nella 3D View)

Abbiamo visto quindi come far seguire alle particelle un determinato percorso, impostando anche la durata dell'effetto; la scena presentata è davvero banale, ma l'effetto in sé si presta ad un gran numero di utilizzi.

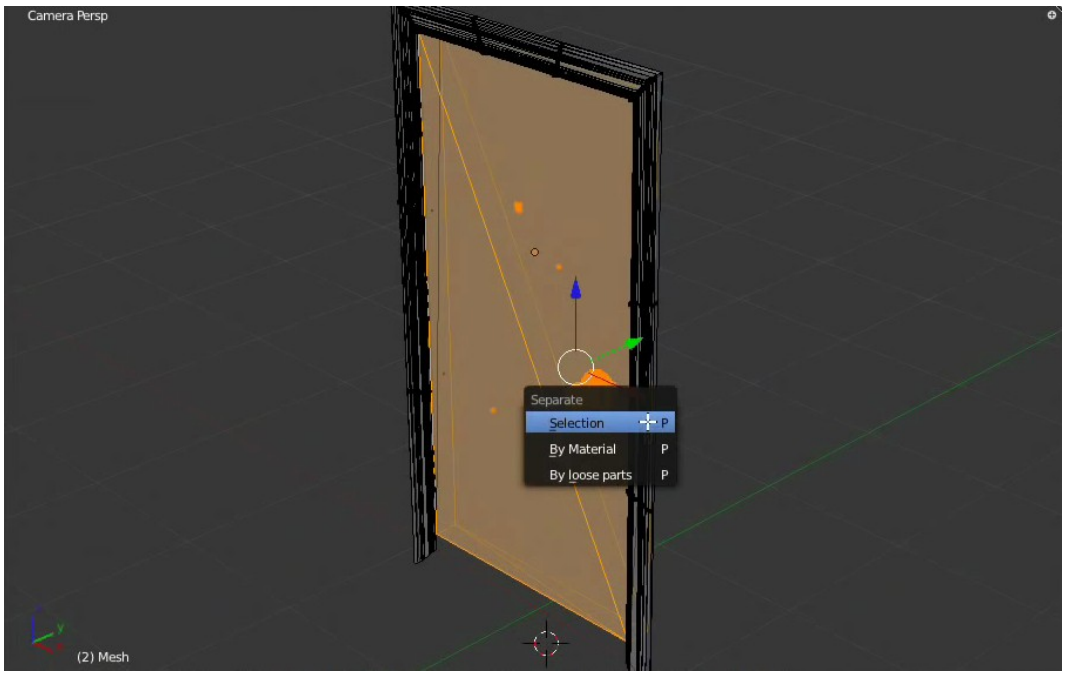
* * *

Game Engine - Aprire le porte in un gioco con l'attuatore F-Curve

In questo breve tutorial vedremo come far eseguire delle animazioni prestabilite agli oggetti del *Game Engine*, il motore di gioco di Blender, mediante l'attuatore *F-Curve*, una volta noto come *IPO Actuator*. La scena che sto utilizzando per questo esempio contiene il modello 3D di una porta; ovviamente, l'animazione prevede l'apertura della porta, solo che il modello 3D è unico, per cui ne approfitto per mostrare l'utilizzo di *Separate* e il posizionamento corretto del *Pivot Point*.

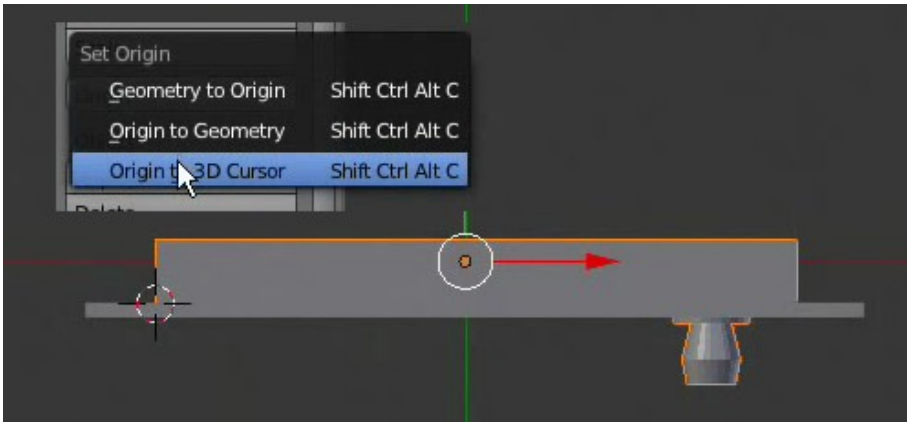
La scena utilizzata in questo tutorial

Selezioniamo l'oggetto, passiamo in Edit Mode e passiamo alla modalità selezione facce.
Assicurandoci di aver attivato il pulsante che ci consente di visualizzare e selezionare anche le facce coperte (pulsante nell'header della finestra 3D), selezioniamo le facce che compongono la parte della porta che dev'essere effettivamente mossa, dopodiché premiamo *P* e scegliamo *Selection* dal menù che apparirà a video.



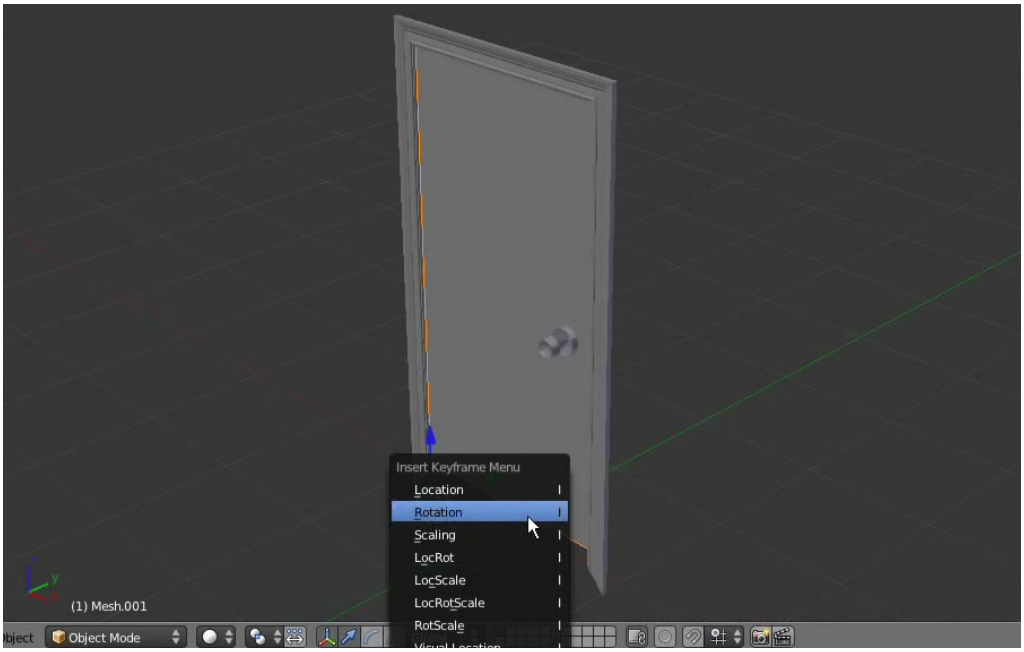
P - Separate, sulla porzione di porta da "staccare" dall'oggetto di partenza, ottenendo un nuovo Object a parte

Adesso dobbiamo impostare correttamente il perno della rotazione per l'oggetto creato, visto che quello posizionato nel baricentro dell'oggetto non va bene, com'è possibile notare premendo *R Z* e muovendo il mouse. Utilizzando possibilmente la vita *TOP* in modalità *Ortogonale*, posizioniamo il *3D Cursor* in corrispondenza di uno dei cardini della porta, dopodiché clicchiamo su Origin nella *Tool Shelf* e selezioniamo "*Origin to 3D Cursor*" nel menù che apparirà a video; adesso, premendo *R Z* e muovendo il mouse, noteremo che il risultato è quello desiderato.



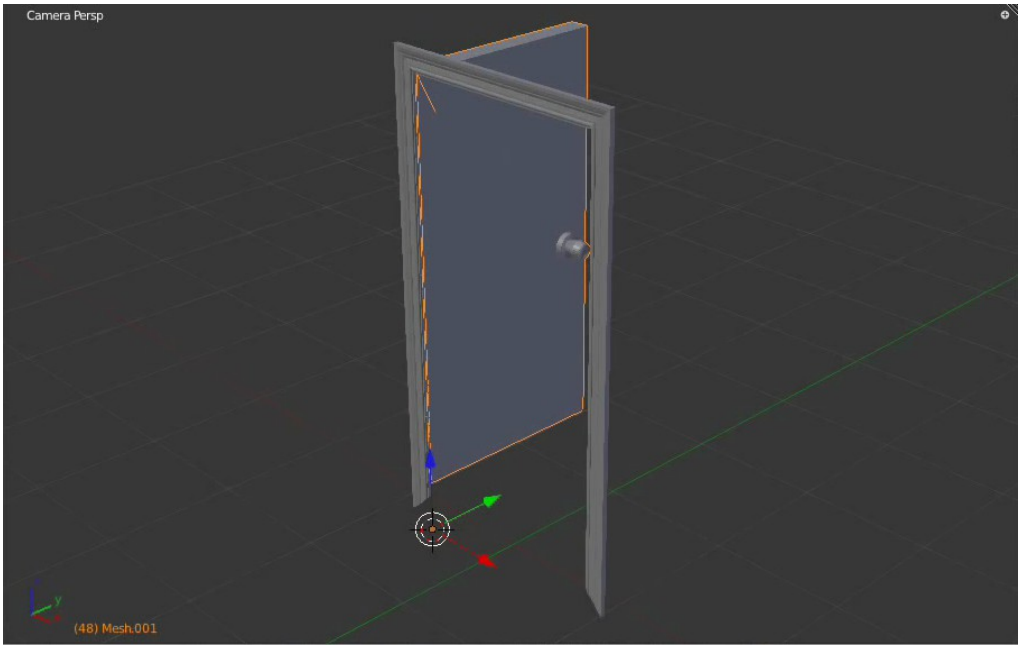
Set Origin to 3D Cursor, per il nuovo oggetto

Dobbiamo realizzare l'animazione base, quella di apertura della porta; quanto deve durare questa animazione? Facciamo ad esempio 2 secondi, per cui inseriamo un keyframe *Rot* al frame 1 premendo *I* e scegliendo *Rotation*, dopodiché andiamo al frame 48 o 50, sei il nostro frame rate è 24 o 25, ruotiamo di 90 gradi la porta con *R Z* 90 ed inseriamo un altro keyframe con *I - Rotation*.



Inserimento di un Keyframe per Rotation (l'orientamento)

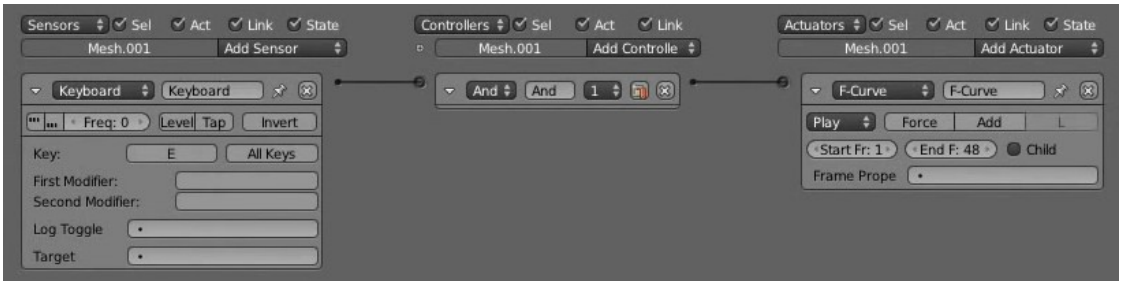
Premendo *ALT A* nella 3D View, avremo una conferma della corretta realizzazione dell'animazione; premendo *P* per entrare in modalità gioco, invece, non vedremo niente.



Fotogramma dell'animazione ottenuta, riprodotta con ALT A; con P (in modalità gioco), invece, non succede niente

Per applicare l'animazione "a comando" dobbiamo infatti assegnare almeno un *sensore*, un *controller* e un *attuatore* alla porta. Con tale oggetto selezionato, quindi, apriamo una finestra *Logic Editor* ed inseriamo un sensore di tipo *Keyboard*, per cui la nostra porta si aprirà quando premeremo

un tasto, ad esempio il tasto *E*; colleghiamo questo sensore ad un *controller AND* e tale *controller* ad un *attuatore* di tipo *F-Curve*. Nell'*attuatore* dobbiamo impostare per *Start* ed *End* i frame di riferimento dell'animazione creata nella *3D View*; nel mio caso, tali frames sono 1 e 48 perché ho creato l'animazione proprio all'inizio del clip. Va da sé che possiamo creare più *IPO* per un oggetto e fargli eseguire diverse animazioni, in base ai tasti premuti o ad altre condizioni, nel *Game Engine*.



Lo schema sensore - controller - attuatore nel Logic Editor, per l'oggetto "porta"

Torniamo quindi alla *3D View*, premiamo *P* per entrare in modalità gioco e premiamo *E* per osservare il risultato ottenuto.

Screenshot del risultato ottenuto in modalità gioco dopo aver premuto il tasto E

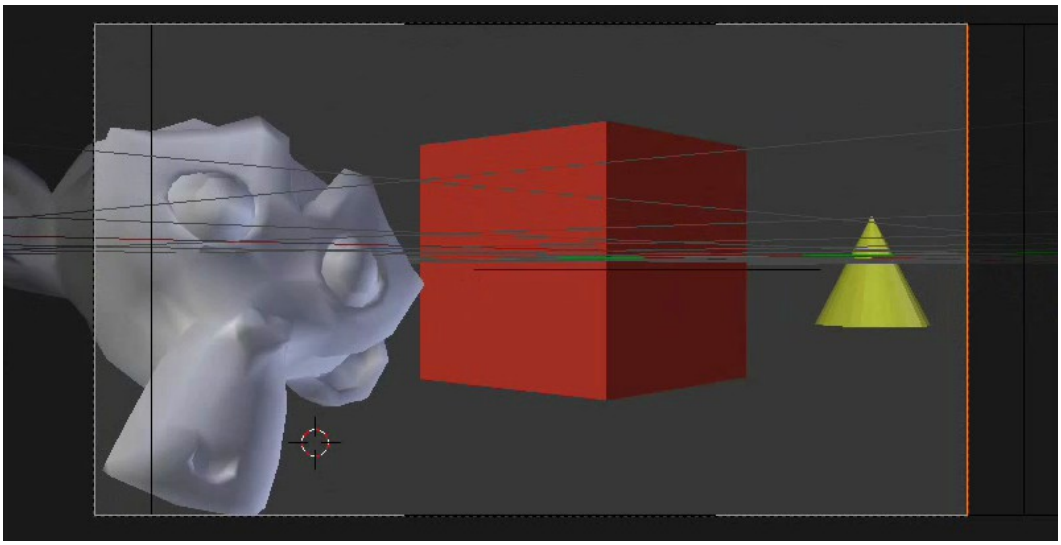
Questo è il caso base, nel senso che premendo *E* la porta si aprirà sempre e comunque, ma non è difficile aggiungere una condizione supplementare, ad esempio una distanza massima tra il giocatore e la porta, oppure un evento successivo, come la chiusura della porta dopo un certo numero di secondi, come avviene in molti giochi, ma di questo parleremo un'altra volta.

* * *

Rendering stereoscopici animati (basi)

Questo tutorial nasce da un paio di richieste sulla realizzazione di rendering stereoscopici in Blender 3D. Tratteremo le *Scene* di Blender, le animazioni, la parentela e il compositing dei nodi per il rendering. Nel mio *ebook sul Compositing dei Nodes* in Blender illustro, tra le altre cose, come configurare i Nodes dell'editor per ottenere proprio quest'effetto, ma dal momento che l'ebook parlava proprio dei *Nodes* tralasciavo la configurazione della scena; in questo tutorial, invece, farò il contrario: parlerò delle telecamere, delle *Scene* e delle animazioni ma non approfondirò l'argomento dei *Nodes*, trattato nell'ebook, anche se ovviamente dovrò utilizzarli per realizzare l'effetto.

Il rendering che produrremo va bene per la visione con i classici occhialini colorati, che in genere hanno una lente rossa e l'altra color ciano; questa tecnica può essere adottata, in generale, se si utilizzano occhialini con lenti colorate, non occhialini con lenti polarizzate o altre tecnologie, per i quali sono richieste invece altre soluzioni. Nella scena che ho preparato sono presenti tre oggetti, dotati di Textures e colori di base, disposti ad una certa distanza dalla telecamera (anzi, dalle telecamere) perché l'effetto stereoscopico dipende fortemente dalla distanza dell'oggetto dall'osservatore, come vedremo con un rendering d'esempio.



La scena utilizzata per questo tutorial

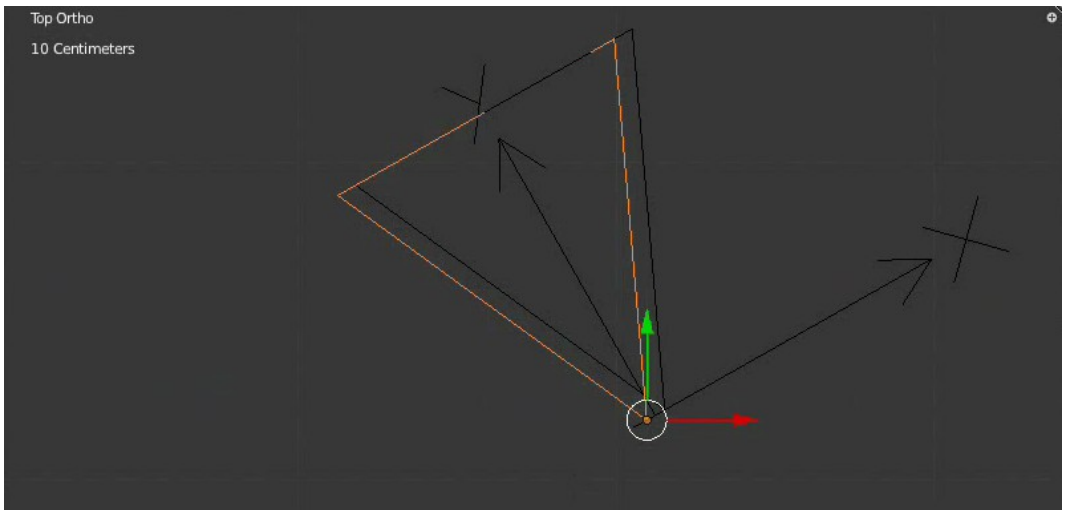
Per realizzare il rendering stereoscopico abbiamo bisogno, in realtà, di due rendering, che poi combineremo mediante l'editor dei nodi di Blender, in post-produzione.

Abbiamo bisogno, quindi, di due telecamere complanari, che simuleranno i nostri occhi. I centri ottici delle telecamere dovranno essere posti ad una certa distanza, tipicamente 6-7 centimetri.

Può essere utile, adesso, cambiare l'unità di misura di riferimento dell'universo virtuale di Blender in metri, mediante l'apposito pulsante nella scheda *Scene*, all'interno della *Properties Window.*

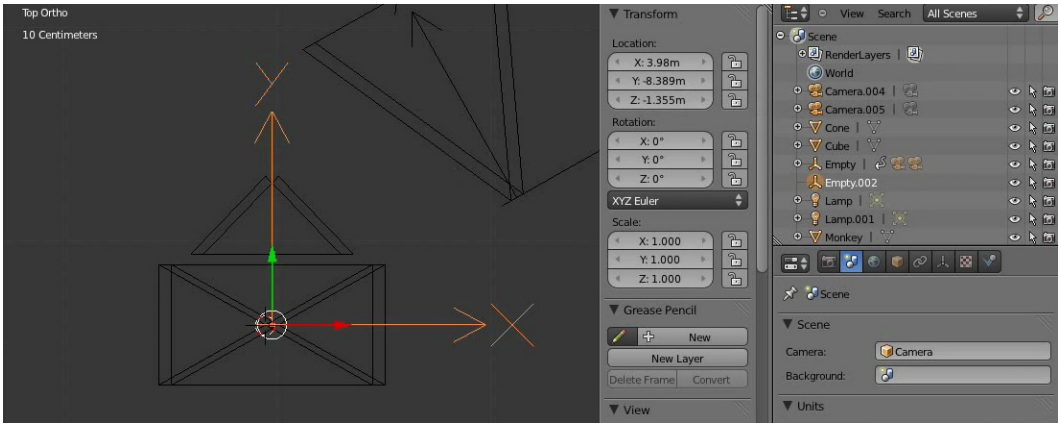
Inseriamo quindi una telecamera, duplichiamola e spostiamo la copia lungo l'asse X locale (cioè con *G X X*) ad esempio di 6 centrimetri, cioè 0.06 metri, per cui quello che dobbiamo fare è: selezionare la prima telecamera, duplicarla con *SHIFT D*, premere *G X X*, digitare 0.06 e premere Invio.

Per animare queste due telecamere in maniera coerente, quando si sposteranno o ruoteranno, inseriamo una *Empty* proprio a metà strada tra i loro centri ottici; utilizziamo magari la *Transform Window*, richiamabile con *N* in una *3D View*, per inserire i valori precisi.



La creazione e il posizionamento della seconda telecamera

A questo punto rendiamo le due telecamere figlie della *Empty*, in modo tale da animare lo spostamento e la rotazione della sola *Empty*, per semplicità; selezioniamo quindi la prima telecamera, poi selezioniamo anche la *Empty* con una selezione multipla, premiamo *CTRL P* per creare la relazione di parentela e confermiamo.



La Empty alla quale imparentare la prima telecamera

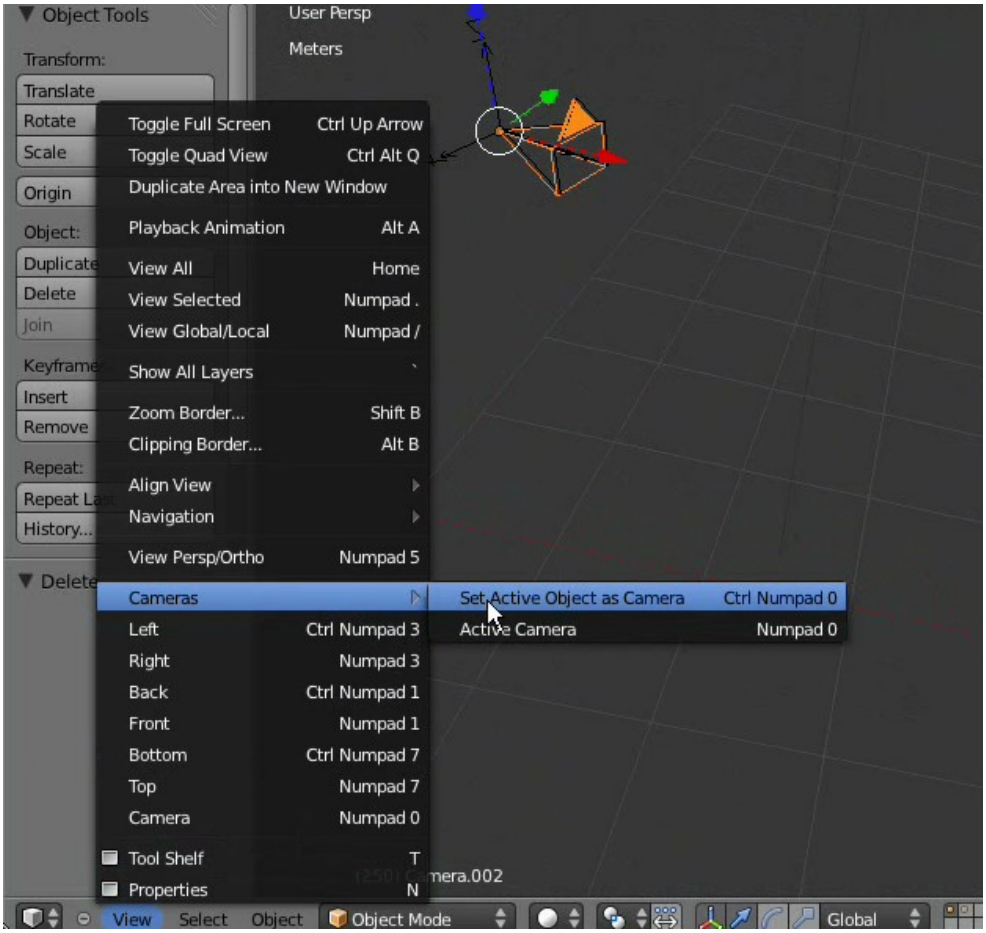
Ripetiamo la stessa operazione con l'altra telecamera, ricordando che per creare la relazione di parentela bisogna selezionare, con selezione multipla, prima l'oggetto figlio e poi l'oggetto padre.

Adesso abbiamo le due telecamere e possiamo anche animare le loro traiettorie con facilità mediante una *Empty*, ma non abbiamo ancora i due rendering.

Per ottenere un altro "*Render Layer*" (un livello di rendering finale) dobbiamo creare un'altra *Scene*, una scena interna al progetto di Blender, identica a quella appena definita ma da renderizzare con la seconda telecamera.

Aggiungiamo quindi una *Scene* cliccando sul relativo pulsante nella finestra *Info* e scegliamo "*Full Copy*" dal menù che apparirà a video; in questo modo, nella nuova scena (che di default avrà il nome della prima seguito dal suffisso *.001*) avremo gli stessi oggetti della prima.

Adesso dobbiamo impostare la telecamera di sinistra come telecamera per il rendering della prima *Scene* e la telecamera di destra per la seconda... beh, visto che abbiamo appena creato *Scene.001* e ci troviamo qui, iniziamo da questa: selezioniamo la telecamera di destra e, in una finestra *3D View*, apriamo il menù *View* e clicchiamo su *Cameras - Set Active Object as Camera*.



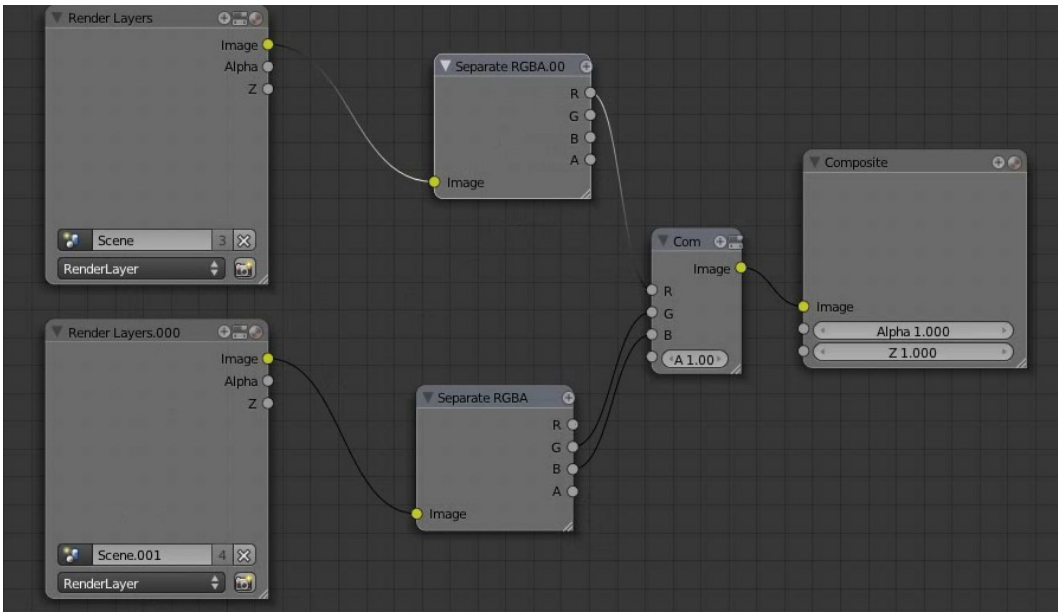
Cameras - Set Active Object as Camera

Torniamo alla Scena iniziale, *Scene*, selezioniamo la telecamera di sinistra ed impostiamola come telecamera per il rendering, quindi di nuovo con *View - Cameras - Set Active Object as Camera*.

Sto lasciando il valore 35, quello di default, come lunghezza focale delle due telecamere, ma chiaramente potete fare le prove anche con altri valori; attenzione, perché in questo caso dovrete modificare la lunghezza focale della telecamera di sinistra nella prima scena e quella della telecamera di destra nella seconda.

A questo punto dobbiamo impostare l'editor dei nodi per effettuare il compositing; come anticipato, non tratterò l'argomento in dettaglio, vedremo solo la configurazione: i due nodi di input *Render Layers*, associati rispettivamente alla prima e alla seconda scena, forniscono le immagini

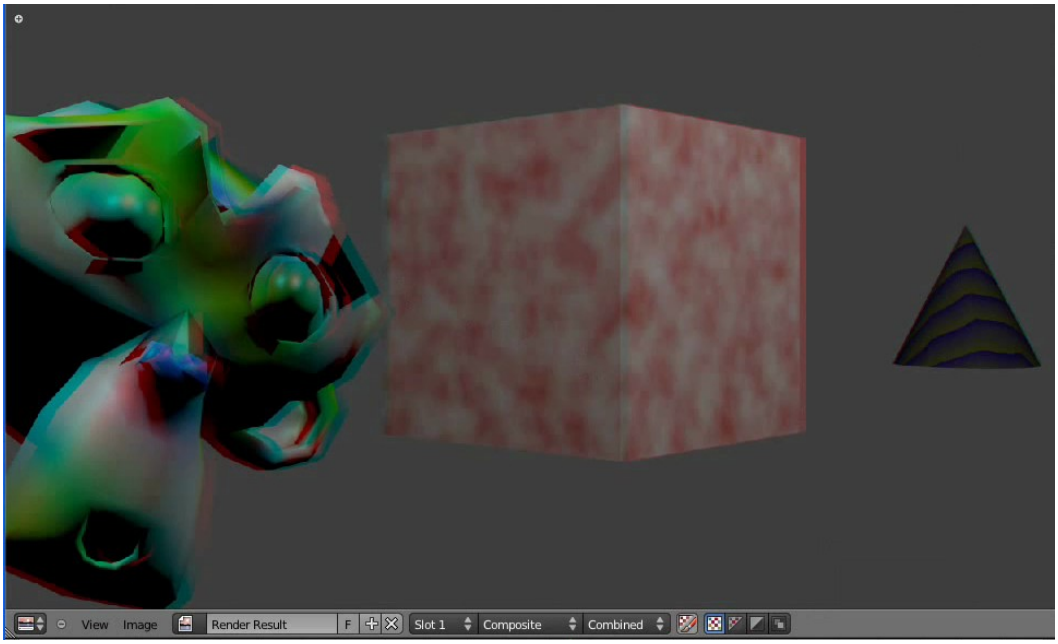
renderizzate frame per frame, ma per visualizzare l'immagine composita con gli occhialini con le lenti rossa e ciano dobbiamo scomporre i canali RGB delle immagini, prendendo solo il rosso dalla prima telecamera e il verde e il blu (che insieme danno appunto il ciano) per la seconda, mediante nodi *Separate RGB*, dopodiché dobbiamo combinare le due immagini mediante un Nodo *Combine RGB* e passare il risultato al Nodo *Composite* per creare l'immagine renderizzata finale.



Lo schema di Nodi di Compositing

Il rendering richiederà più del doppio di un rendering normale perché stiamo renderizzando la stessa scena due volte ed in più stiamo utilizzando il compositing in post-produzione.

Vi mostro (nella pagina seguente) il rendering del frame 135, dove sono visibili tutti e tre gli oggetti della scena, per riprendere il discorso sugli effetti della distanza degli oggetti dalle telecamere durante il rendering: il "doppio bordo" che vedete, soprattutto in *Suzanne*, è dovuto alla sovrapposizione delle due immagini renderizzate, sovrapposizione che è più evidente per gli oggetti vicini (come *Suzanne*, che presenta un bordo largo) che per quelli lontani (come il cono, che presenta un bordo quasi impercettibile): semplice parallasse, niente di particolare.



Rendering del 135esimo frame

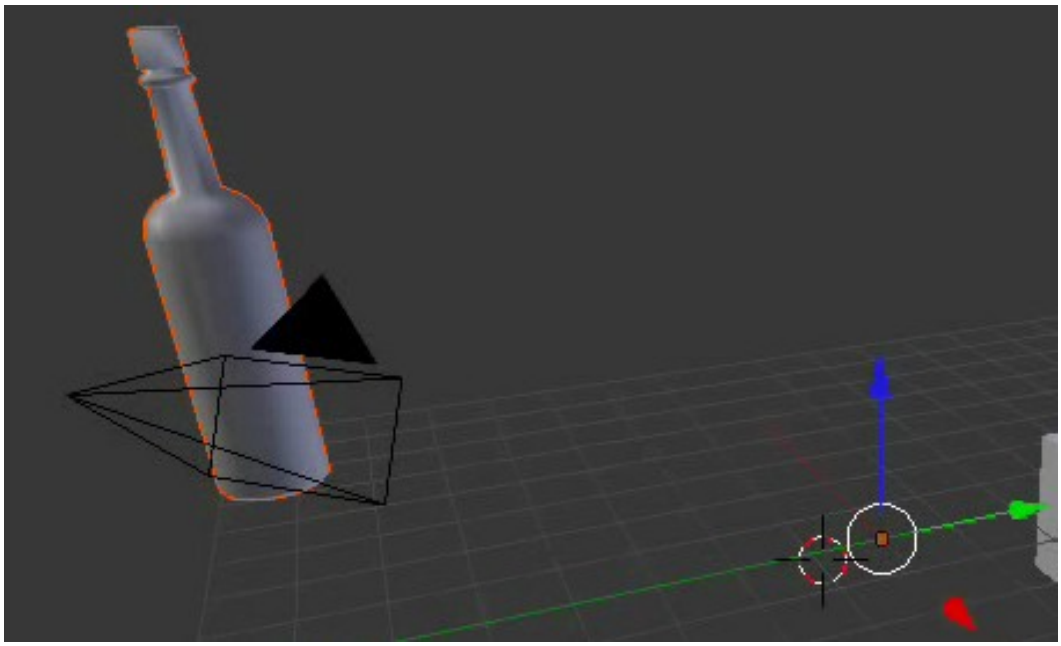
In questo tutorial sto implementando, comunque, un effetto di base, nel senso che ho impostato le due telecamere come complanari, con gli assi ottici paralleli, senza convergenza, con una distanza tra le telecamere fissata "a occhio" in 6 cm, in maniera non proprio scientifica, ma si tratta di una buona base di partenza.

* * *

Importare ed esportare oggetti, mesh e modelli 3D nelle scene

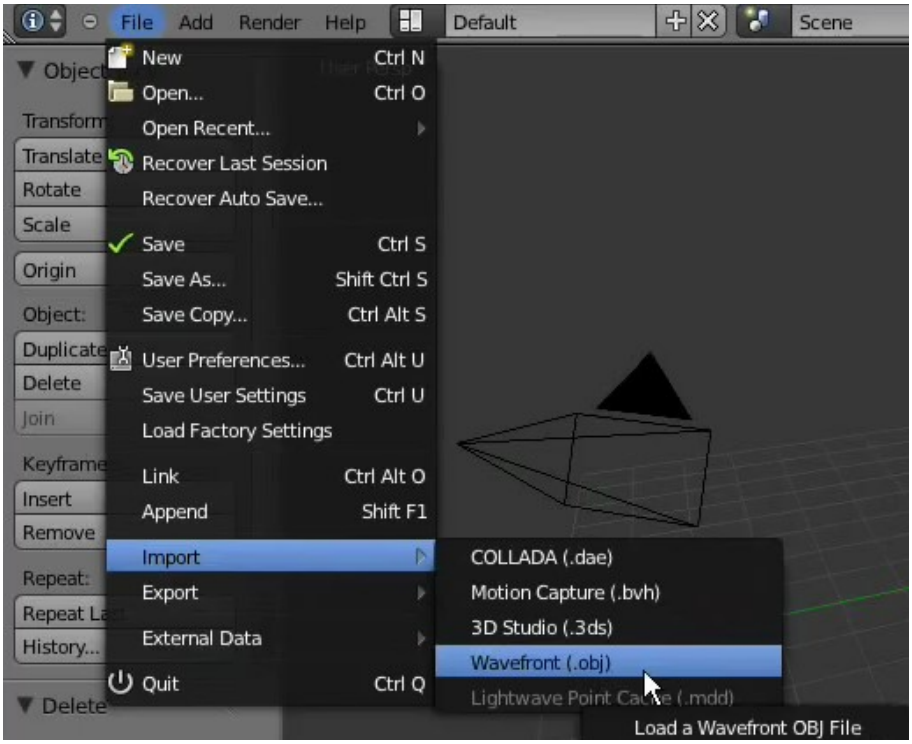
Questo breve tutorial nasce dalle richieste di alcuni utenti del Forum del sito che volevano sapere come poter importare ed esportare mesh ed oggetti 3D nelle scene di Blender, per salvarli come oggetti a parte e riutilizzarli in più scene.

Entrambe le operazioni sono molto semplici, in effetti, e le illustrerò importando un modello in formato *Wavefront OBJ* ed esportandolo (sempre in formato *OBJ*); Blender consente di importare ed esportare modelli in vari formati, come *PLY* o *STL*, mentre per altri formati, come *3DS* per *3D Studio* oppure formati per *Lightwave*, *AutoCAD* o *Maya*, è necessario attivare alcuni script; in ogni caso, consiglio *OBJ* perché è portabile e lo leggono praticamente tutti i programmi di grafica 3D.
Nel modello 3D da me scelto l'origine del sistema di riferimento locale dell'oggetto non coincide col baricentro dello stesso, per cui ne approfitto per mostrarvi come risolvere questo inconveniente.



L'oggetto importato non ha l'Origin nel suo baricentro

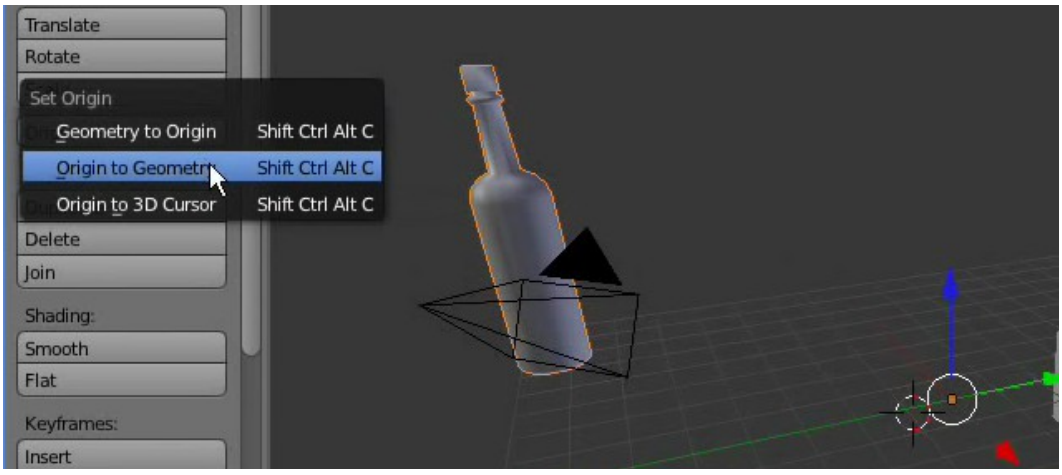
Per importare un modello da disco, apriamo il menù *File* e clicchiamo su *Import*, scegliendo dal menù che apparirà a video il formato del file che vogliamo importare, nel nostro caso *OBJ*.



File - Import - OBJ

Si aprirà una finestra stile esplora risorse, dove potremo selezionare la cartella contenente il file e, quindi, proprio il file *OBJ*. Una volta selezionato tale *File*, clicchiamo su *Import OBJ*.

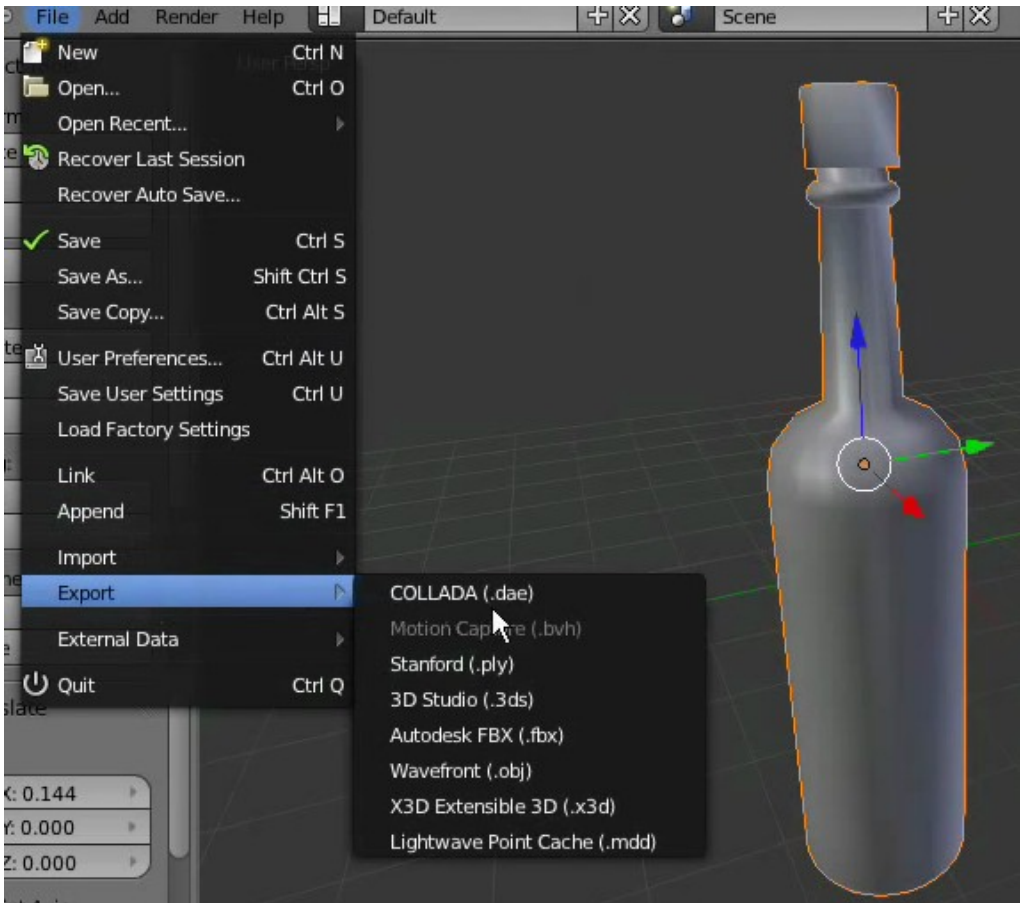
Come potete vedere, l'oggetto non è stato inserito al centro della scena, anche se per Blender la sua "origine" si trova lì, come possiamo constatare osservando la posizione dei manipolatori di traslazione; l'oggetto in realtà non è neanche propriamente selezionato, per cui per prima cosa facciamo click col tasto destro del mouse sulla bottiglia, dopodiché portiamo il suo sistema di riferimento nel baricentro scegliendo *Origin* - *Origin to Geometry*, nella *Tool Shelf*.



Origin to Geometry, per impostare l'Origin nel baricentro dell'oggetto

Quando importerete alcuni modelli, questi potrebbero non apparire a video, anche se la loro presenza è segnalata nell'*Outliner*; in genere, si tratta di modelli o molto distanti dal centro dell'universo virtuale di Blender, per cui sono "fuori vista", oppure particolarmente grandi, così grandi che quando vengono importati ci ritroviamo al loro interno e non li vediamo, per cui conviene zoomare indietro o selezionarli nell'*Outliner* e scalarli con gli strumenti messi a disposizione nella *Transform Window*... non è il caso della bottiglia, ma può capitare, per cui è bene sapere anche questo. Qualora i modelli non dovessero essere elencati nemmeno nell'*Outliner*, invece, significa che non sono stati importati affatto nella scena perché qualcosa è andato male; in questo caso, conviene controllare la *Console* di Blender ed esaminare i messaggi in essa presenti.

Per esportare un modello 3D come file a parte, è sufficiente selezionare *File – Export – formato* ed indicare il percorso di esportazione nella finestra stile esplora risorse che apparirà a video; i vari formati mettono a disposizione diverse opzioni di esportazione, sulla sinistra nella scheda: ad esempio, con *OBJ* possiamo scegliere se esportare anche il file dei materiali (*mtl*) o le informazioni sulla mappatura *UV*, le normali ed altro ancora.



File - Export - Formato di esportazione

Importare ed esportare i modelli 3D è un'operazione di fondamentale importanza in Blender e soprattutto è molto utile per evitare di dover rifare gli oggetti nelle varie scene; è possibile, infatti, costruire intere "librerie" di modelli 3D, da riutilizzare per i vari progetti.

C'è poi un'altra modalità, *Append or Link*, che consente di importare modelli 3d, materiali, luci ed altri elementi da altre scene di Blender, trattando i *blend files*, ma di questo parleremo un'altra volta.

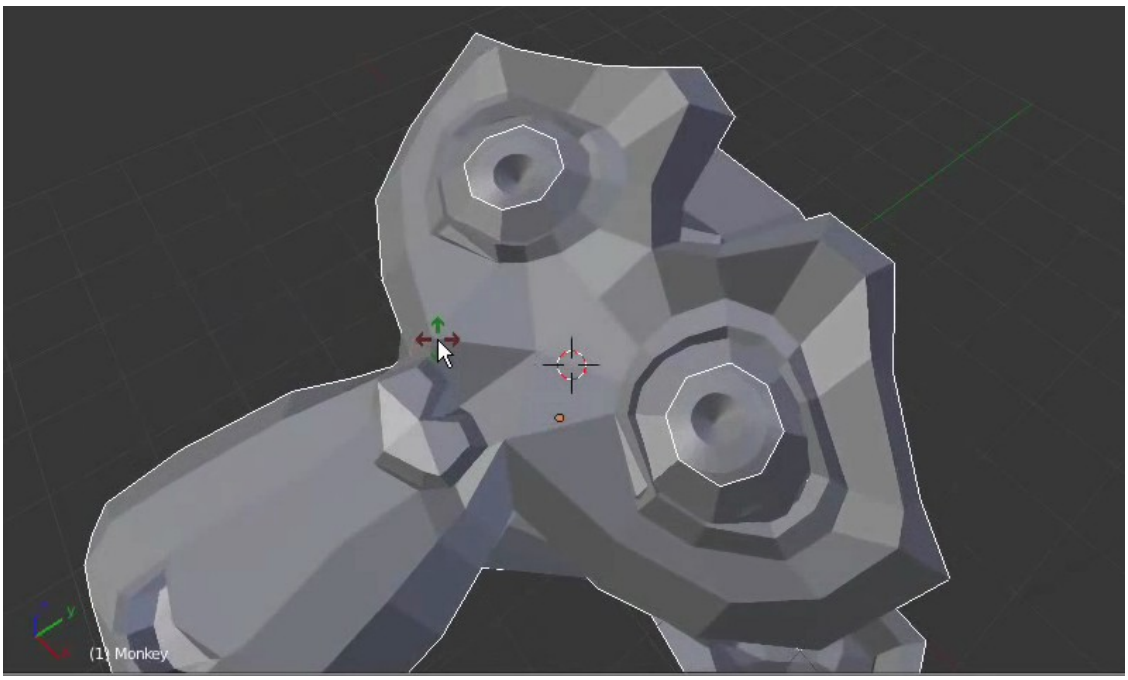
* * *

Copy Rotation vs Individual Origins

In questo brevissimo tutorial metteremo a confronto l'utilizzo del *Pivot Point Individual Origins* e il *Constraint Copy Rotation* per animare la rotazione degli occhi in un personaggio in Blender 3D.

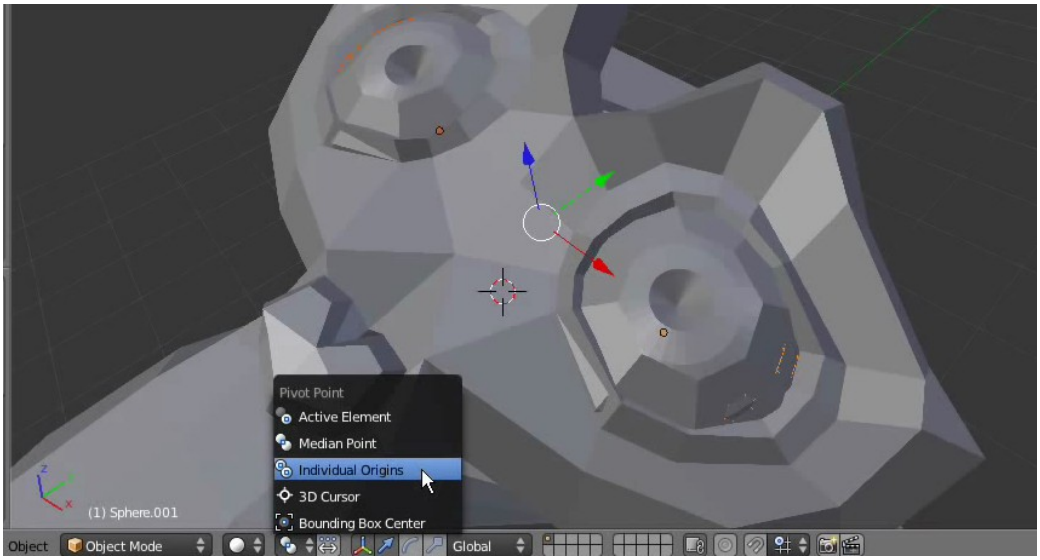
Un'amica mi ha chiesto infatti come poter realizzare quest'effetto per un'animazione; nel suo caso, gli occhi sono stati realizzati con due sfere e devono ruotare, ovviamente, in maniera coerente.
Ci sono almeno due modi per realizzare quest'effetto (senza ricorrere alle *Armature*), per questo caso specifico: *Individual Origin* e *Copy Rotation*, e in questo tutorial li vedremo entrambi.

Nella scena visibile nell'immagine seguente sono presenti *Suzanne* e due sfere. Ho rimosso gli occhi originali di Suzanne e ho posizionato, all'interno delle orbite, le due sfere, modificandole un po' (in pratica, ammaccandole in un polo) per avere un punto di riferimento durante le trasformazioni.



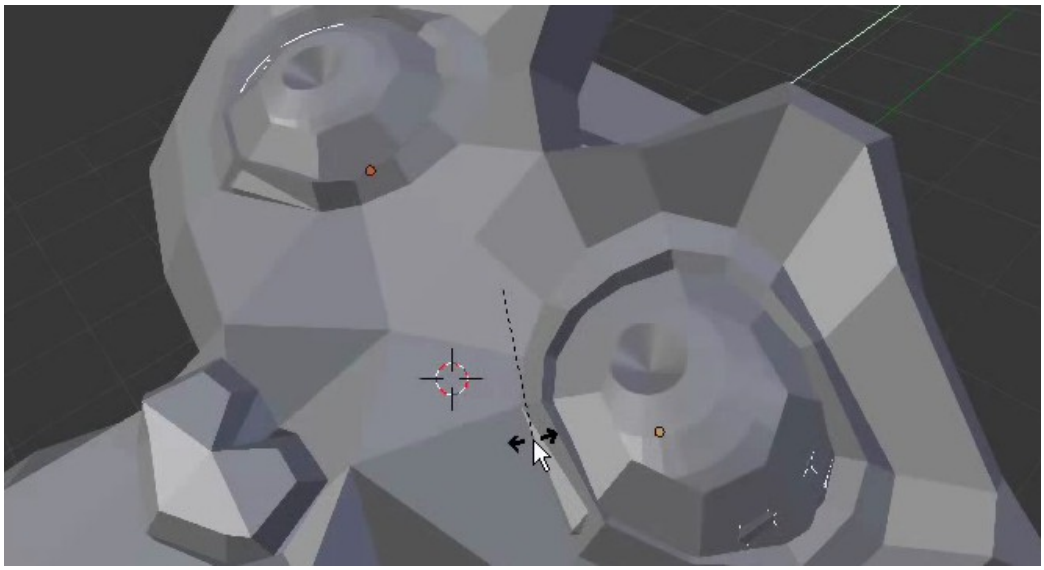
La scena utilizzata per questo tutorial

Le due sfere sono imparentate, come figlie, a *Suzanne*, per cui la seguiranno e ruoteranno in maniera coerente, ed in più potranno avere trasformazioni proprie, come appunto la rotazione.
Iniziamo con *Individual Origins*: in questo caso, dobbiamo selezionare entrambe le sfere, con una selezione multipla, dopodiché dobbiamo passare alla modalità perno "origini individuali" scegliendo appunto *Individual Origins* nel menù *Pivot Point*, nell'header di una finestra *3D View.*

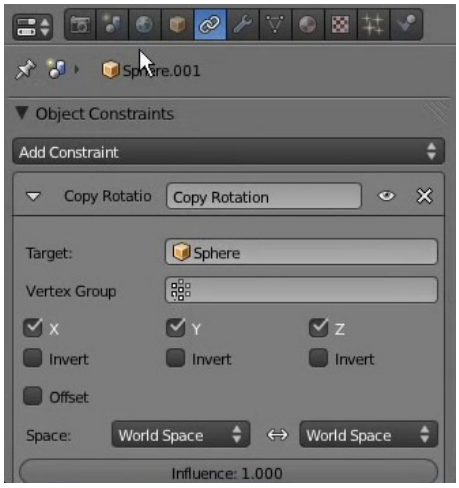


Selezione multipla e Pivot Point "Individual Origins"

In questo modo, anche se la selezione è multipla, per ciascun oggetto selezionato il perno della trasformazione di rotazione, traslazione o scaling sarà il suo baricentro, non il baricentro della selezione, come avviene di default. Possiamo quindi animare la rotazione mantenendo i due oggetti selezionati, ruotandoli con *R* ed inserendo i vari keyframes come desideriamo.

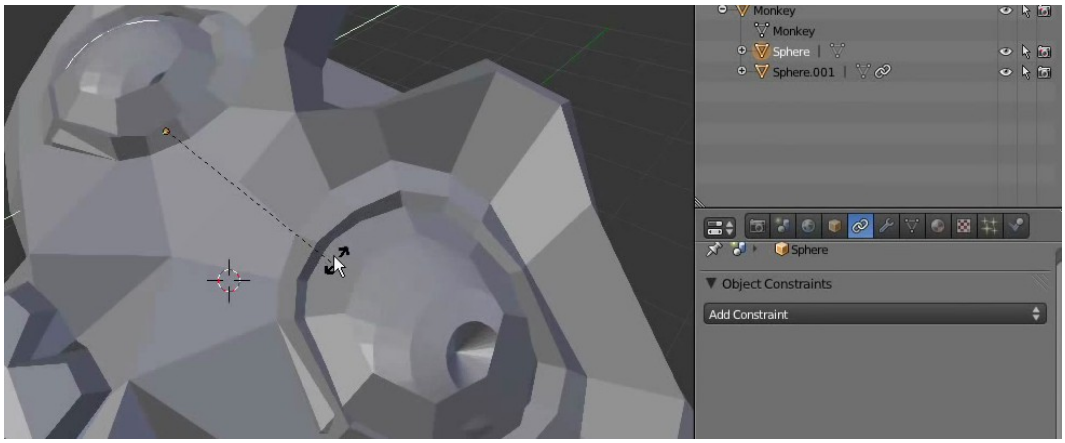

Rotazione sulla selezione multipla con Individual Origins

Esaminiamo adesso la seconda modalità, che fa uso del *Constraint* (letteralmente, "vincolo") *Copy Rotation*. Selezioniamo una sfera (solo una, questa volta), apriamo la scheda *Constraints* all'interno della *Properties Window* e aggiungiamo un vincolo *Copy Rotation*; nel campo *Target* di questo oggetto, digitiamo il nome dell'altra sfera, che utilizzeremo quindi come riferimento.



Utilizzo del Constraint Copy Rotation su una sfera, utilizzando la seconda sfera (che animeremo) come Target del vincolo

La sfera o in generale l'oggetto che possiede il vincolo è quindi quello che seguirà un altro oggetto; d'altronde, è l'oggetto sottoposto al vincolo. Adesso selezioniamo solo la sfera riferimento, con una selezione singola, ruotiamola ed osserviamo l'effetto ottenuto.

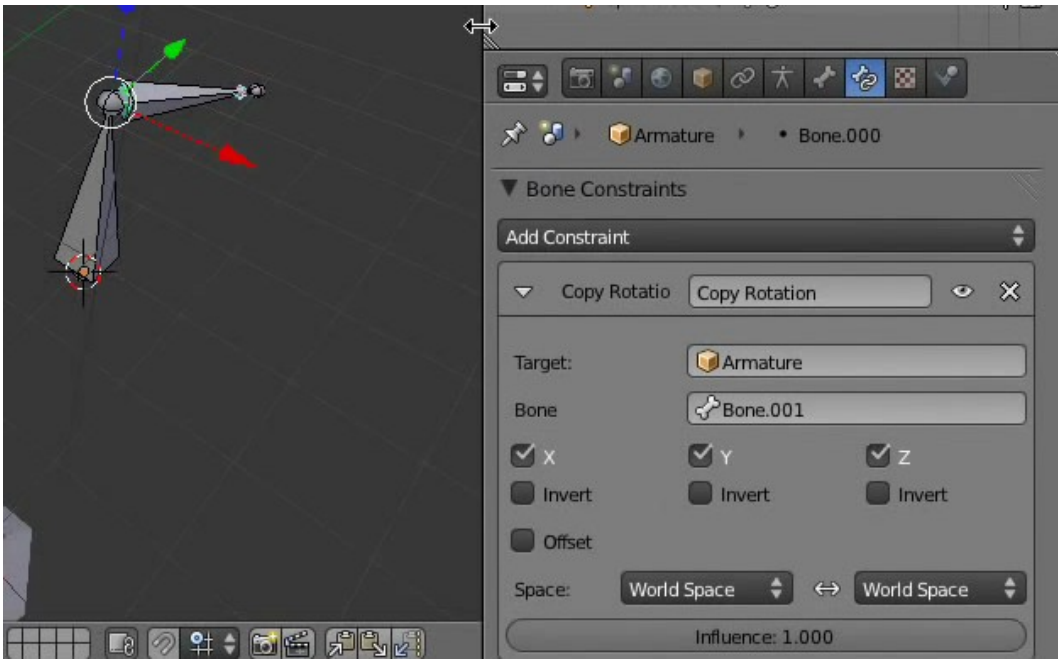


Rotazione della sola sfera utilizzata come Target: anche la sfera provvista di Constraint ruota

In conclusione, il vincolo ci permette di selezionare un solo oggetto e muovere, ruotare o traslare automaticamente tutti gli altri, se questi ovviamente lo utilizzano come riferimento, ma è sempre valido, nel senso che tutte le trasformazioni e le animazioni realizzate con quest'oggetto verranno applicate sempre e comunque agli oggetti collegati; la selezione multipla e l'utilizzo di *Individual Origins* ci lasciano invece una maggiore libertà, nel senso che per ciascun oggetto possiamo inserire anche altri keyframes a parte.

La scelta dell'uno o dell'altro dipende quindi dalla scena in questione.

Aggiungo che questo *Constraint* esiste anche per le *Armature* e ci consente di ruotare un osso in funzione di un altro; proviamolo aggiungendo una semplice armatura alla scena e creando, a partire dall'osso di base, due ossa, dopodiché selezioniamone uno in *Pose Mode* e apriamo la scheda *BONES CONSTRAINTS* nella *Properties Window*: questa scheda contiene, quindi, vincoli propri delle ossa. Aggiungiamo un vincolo di tipo *Copy Rotation* e digitiamo il nome dell'armatura da utilizzare nel campo *Target*; apparirà un altro campo, *Bone*, nel quale dovremo digitare il nome dell'osso da utilizzare come riferimento.



Utilizzo di Copy Rotation per le ossa di un'armatura

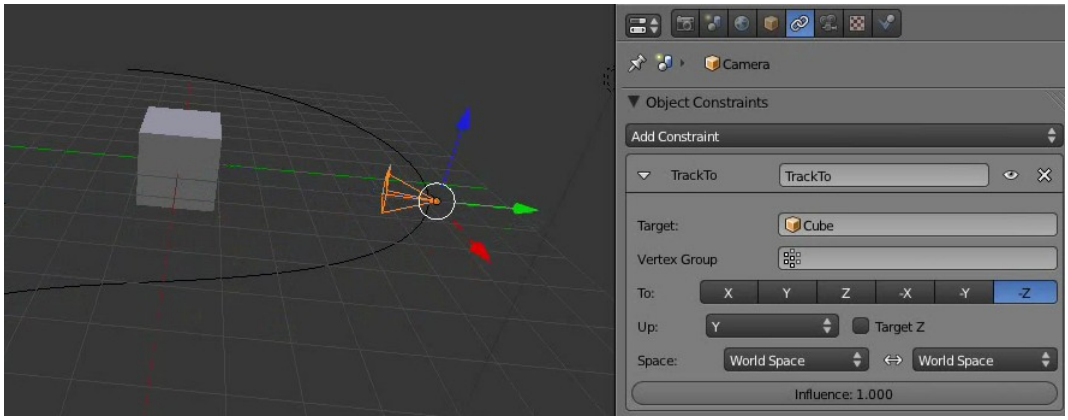
Copy Rotation per le ossa copia, di default, proprio l'orientamento dell'osso *Target*, non solo le trasformazioni, per cui può essere utile attivare l'opzione *Offset*, che mantiene l'orientamento iniziale per l'osso provvisto del vincolo e si limita ad applicare le trasformazioni successive.

* * *

Far puntare un oggetto ad una telecamera in un'animazione

A volte si ha la necessità di far puntare un oggetto ad una telecamera in movimento nella scena di Blender. Blender non mette a disposizione le telecamere "*Target*" presenti in altri programmi come, ad esempio, *3D Studio MAX*, tuttavia si può risolvere il problema utilizzando il *Constraint* (lett.: vincolo) "*Track To*" (da applicare quindi alla telecamera), impostando lo stesso come segue:

- in *Object*, scriviamo il nome dell'oggetto da puntare;
- in *TO*, impostiamo *-Z*;
- in *UP*, impostiamo *Y*.



Il Constraint Track To

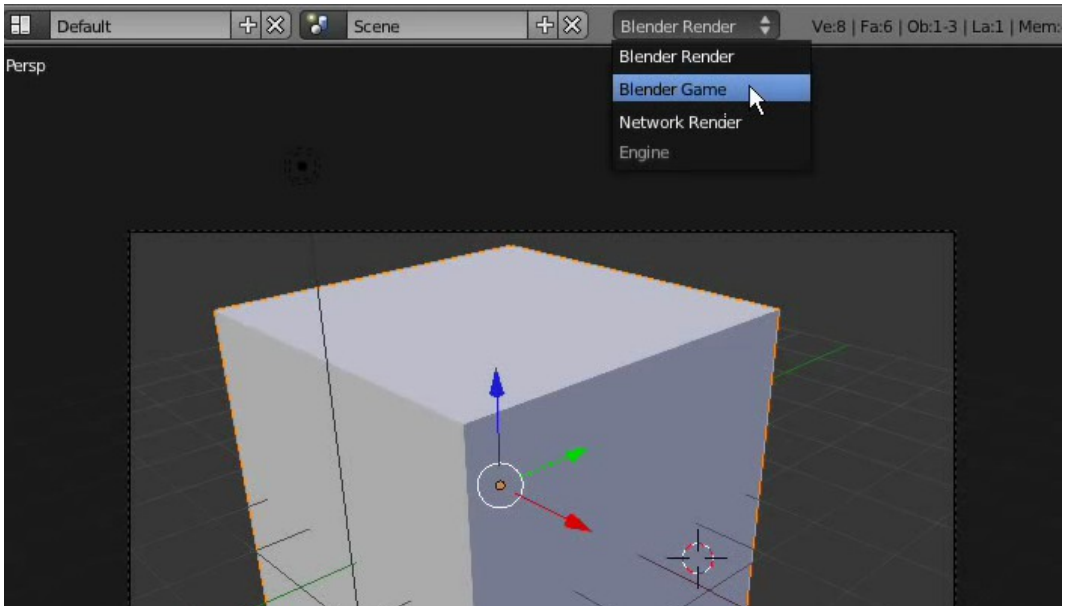
La cosa interessante è che è possibile far puntare alla telecamera un oggetto *Empty*, invisibile, per cui potremo decidere cosa inquadrare nella scena utilizzando un *Empty* che possiamo poi animare (traslandola); in questo modo, potremo muovere la telecamera e, nel mentre, orientarla verso determinati punti con facilità.

* * *

Giochi con immagini 3D stereoscopiche col Game Engine

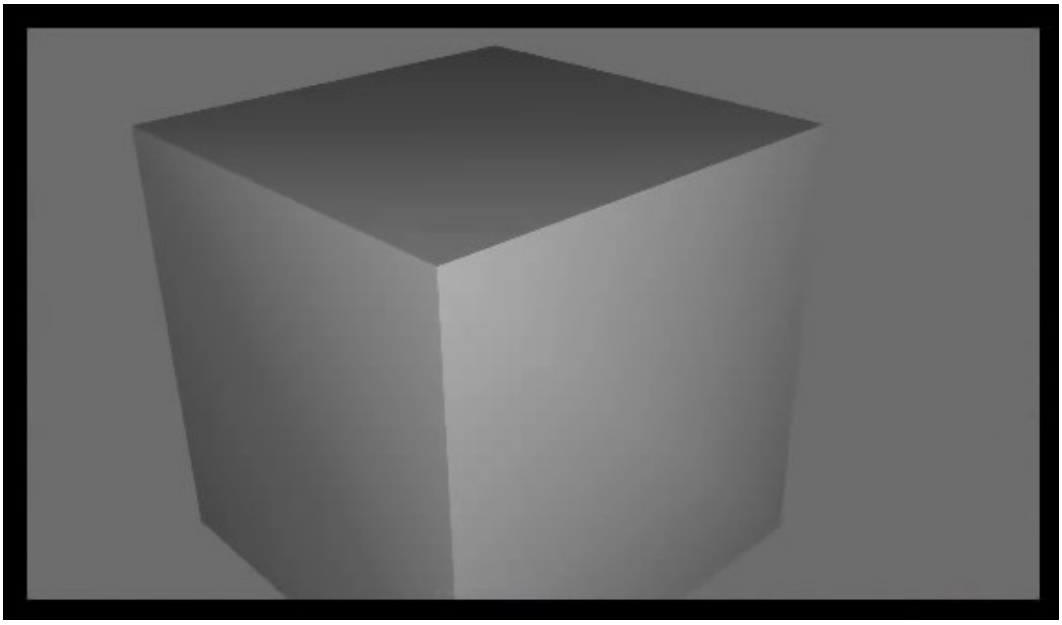
In questo brevissimo tutorial vedremo come visualizzare gli oggetti 3D in modalità stereoscopica, ed in particolare *anaglifo*, nei giochi realizzati col *Game Engine*, il motore di gioco di Blender 3D.
In un tutorial pubblicato precedentemente (presente in questa raccolta) ho mostrato come configurare l'editor dei nodi di Blender per ottenere, mediante il *Compositing*, le famose immagini stereoscopiche dal "doppio bordo", rosso e ciano, da visualizzare con gli occhialini con le lenti colorate. Un'immagine del genere è detta *anaglifo*, mentre gli occhialini, che presentano la lente sinistra di colore rosso e quella destra di colore ciano, sono detti appunto *occhiali per anaglifo*.

Per realizzare la visualizzazione stereografica nei videogiochi prodotti col *Game Engine* di Blender 2.5 è sufficiente attivare una voce presente nel menù *Render*, all'interno della *Properties Window*, quando si lavora appunto col motore di gioco; procediamo con ordine.
Utilizziamo, per questo esempio, una scena di base, ma l'operazione può essere svolta con qualsiasi scena. Dal selettore *Engine*, letteralmente motore (nel nostro caso, di rendering), nella finestra *Info* di Blender, scegliamo la voce *Blender Game*.



La scena utilizzata per questo tutorial, col motore impostato a "Blender Game"

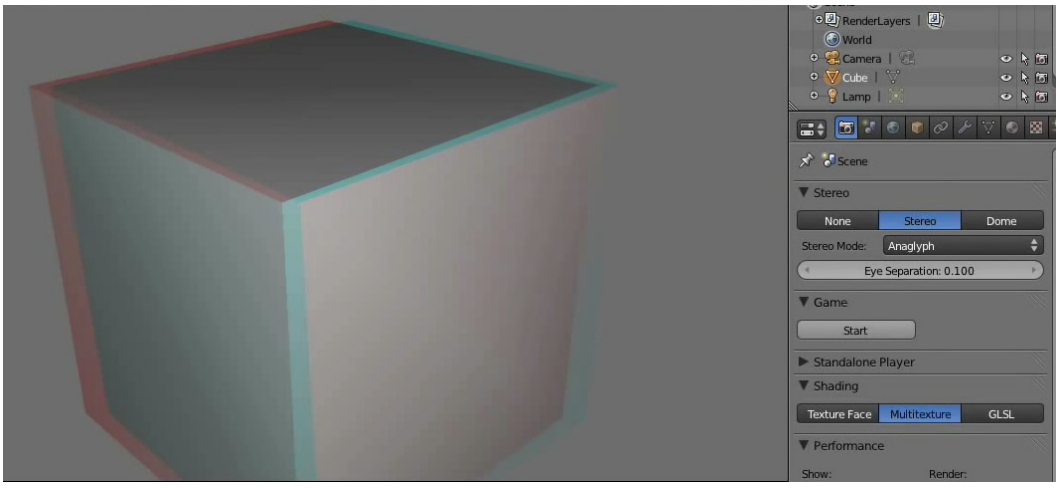
Possiamo avere un'anteprima di come verrà visualizzata la scena durante il gioco premendo *P*, in modo da avviare la simulazione.



Anteprima dell'esecuzione della scena, con P

Premiamo *ESC* per tornare a Blender e selezioniamo il pannello *Render* nella *Properties Window*; qui, alla voce *Stereo* scegliamo appunto *Stereo* e lasciamo le impostazioni di default, cioè *anaglifo* con *eye separation* (una misura della distanza tra le pupille) 0.1.

Avviamo nuovamente la simulazione del gioco con *P*: adesso possiamo notare il famoso "doppio bordo", ciano e rosso, degli oggetti presenti nella scena, quindi tecnicamente questo è tutto quello che serve per realizzare giochi con visualizzazione stereoscopica con il motore di gioco di Blender.



Esecuzione del "gioco" con la modalità Stereo - Anaglifo

Vi conviene provare l'effetto con scene con un maggior numero di elementi, magari posti a distanze diverse dalla telecamera; la scena visibile in questo caso è stata realizzata come semplice esempio per mostrare, più che altro, cosa bisogna fare per attivare la modalità stereo.

Il tooltip di *eye separation* suggerisce di utilizzare, come valore, l'apertura focale della telecamera (che di default è 35, per Blender) diviso 30, ma personalmente apprezzo di più l'effetto proprio col valore di default, che è 0.1, comunque vi consiglio di fare alcune prove per valutare voi stessi.

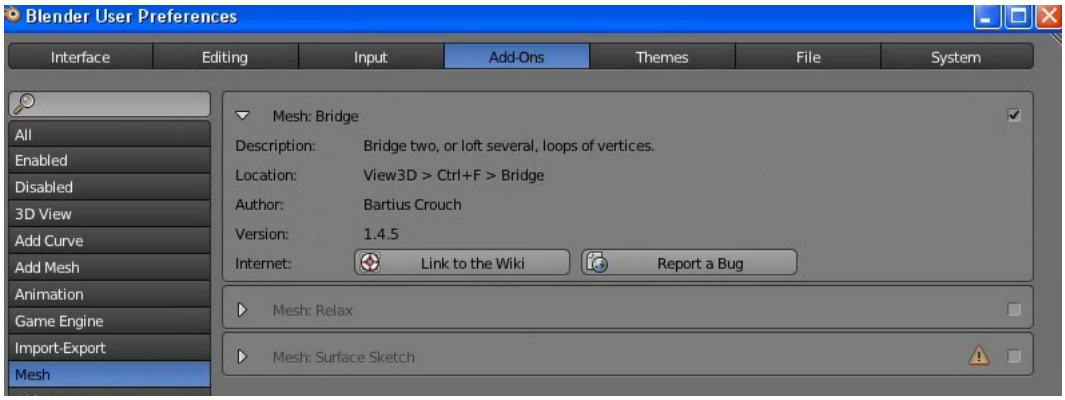
Quanto detto è veramente tutto sull'argomento: non dobbiamo configurare praticamente niente, dobbiamo solo scegliere la voce *Stereo – Anaglyph*; ci sono altre voci, ma (almeno in questo tutorial) non ci riguardano.

* * *

Lo script Bridge and Loft di Bart Crouch

In questo tutorial parleremo dello script *Bridge and Loft* di Bart Crouch per creare facce di collegamento e mesh a partire dai profili in Blender 3D 2.5.

Lo script si trova su *Blender.org* cercando tra le estensioni per la versione 2.5 di Blender oppure all'indirizzo *https://sites.google.com/site/bartiuscrouch/scripts/bridge* ; per installarlo, scaricate il file *.py*, per *python*, nella sottocartella *Script / Addons* della cartella dove avete installato Blender, dopodiché avviate il programma, scegliete *User Preferences* dal menù *File*, quindi la scheda *Add-Ons* nella finestra che apparirà e, qui, *Install Plugin*; si aprirà una finestra stile esplora risorse che vi consentirà di selezionare il file precedentemente scaricato; cliccate su ok per confermare: lo script è ora installato ma va attivato.

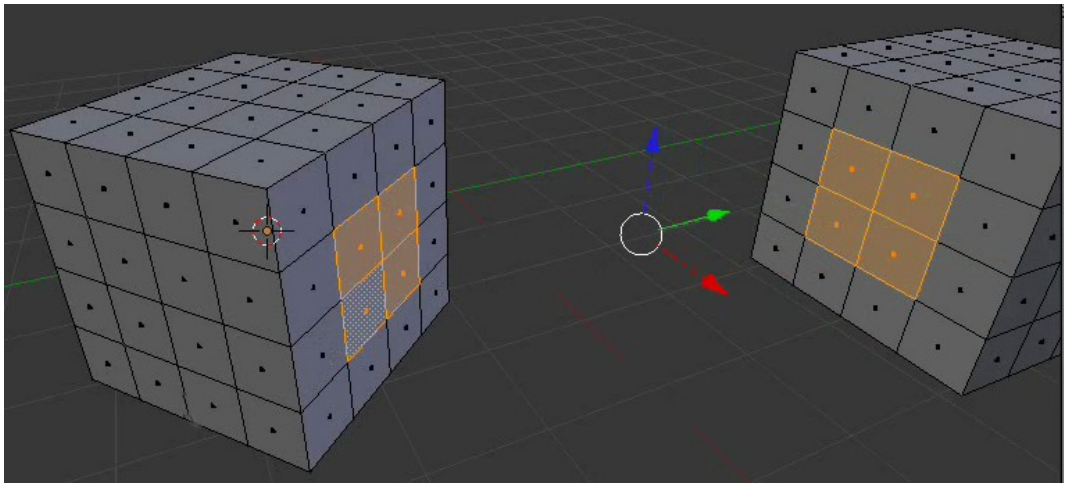


La scheda dello script nella finestra User Preferences - Add-On

Per attivare lo script, scegliete la voce *Mesh* in *Add-Ons* ed individuate *Bridge and Loft*; a questo punto, attivatelo selezionando la checkbox sulla destra.

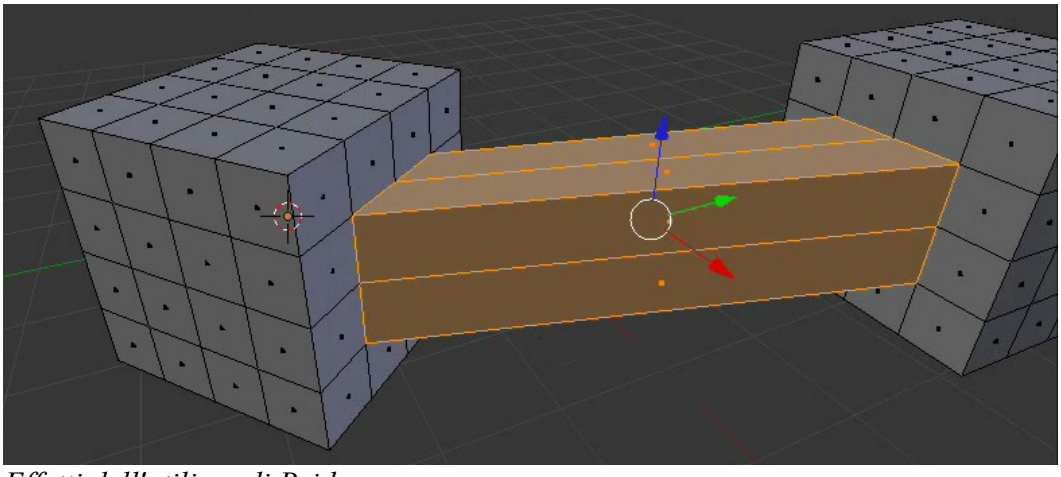
Se volete ritrovare lo script attivato ai prossimi riavvii di Blender, cliccare su *Save as Default* in basso nella scheda, altrimenti chiudetela semplicemente.

Descrivere questo script a parole ha poco senso: molto meglio vedere in pratica cosa consente di fare. Utilizziamo, come prova, un cubo, suddividendolo un paio di volte in modo da creare, come vedremo, collegamenti non banali, dopodiché duplichiamolo, sempre in Edit Mode, in modo da rimanere all'interno dello stesso oggetto; *Bridge and Loft* ci consente infatti di creare in maniera automatica facce per collegare altre facce, ma ovviamente le facce iniziali devono essere all'interno dello stesso oggetto.



La scena utilizzata per questo tutorial

Sempre in Edit Mode, quindi, passiamo in modalità selezione facce e selezioniamo quattro facce dai due cubi. Dal menù *Mesh - Faces* presente nell'header della *3D View* scegliamo *Bridge*, letteralmente "ponte", "collegamento".

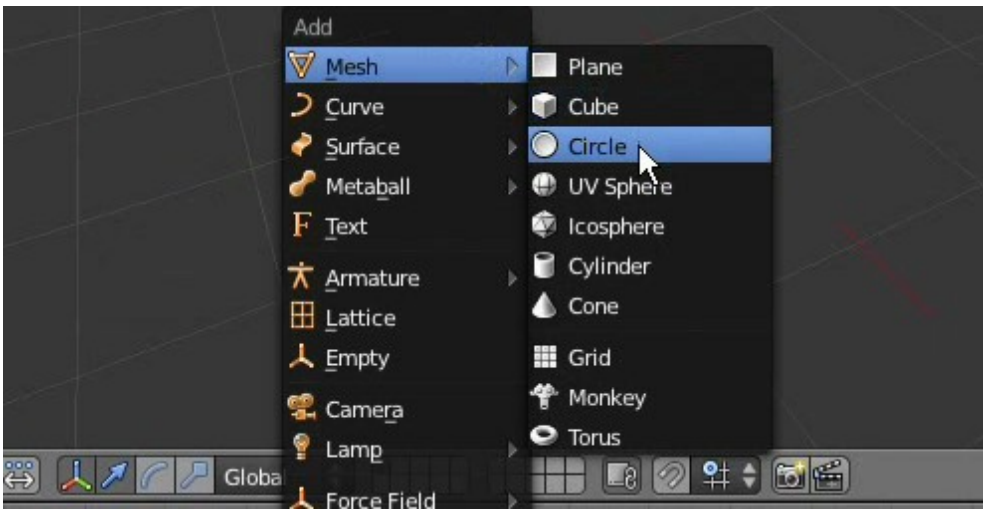


Effetti dell'utilizzo di Bridge

Potete constatare immediatamente l'effetto implementato dallo script; molto probabilmente, i collegamenti non saranno corretti, ma fortunatamente possiamo cambiare la "combinazione" agendo sul pulsante numerico *Twist*, presente sul fondo nella *Tool Shelf*, in una sezione dedicata appunto al nuovo strumento *Bridge*.

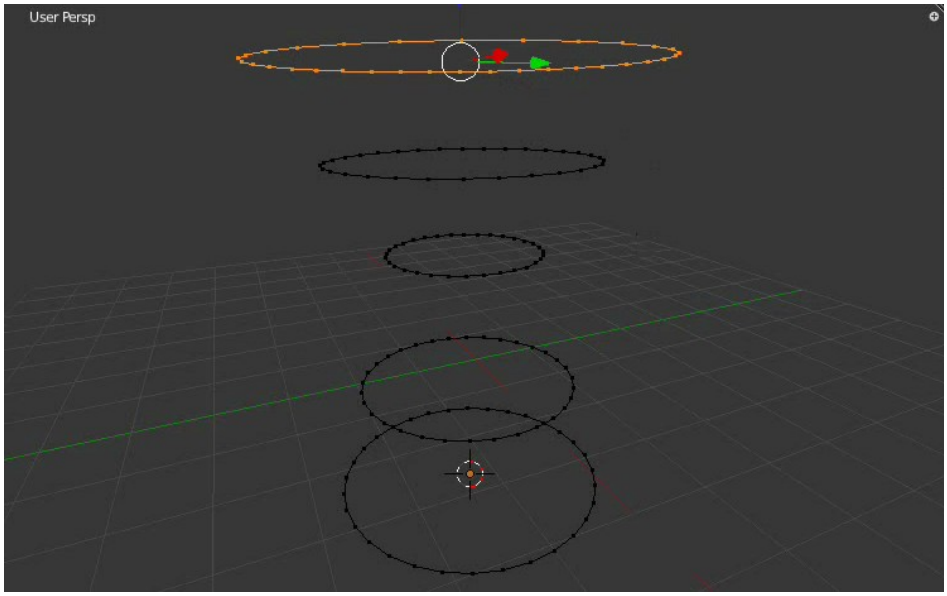
La voce *Segments* indica quanti spigoli bisogna creare per collegare una coppia di vertici delle due facce; scegliendo 1, la connessione sarà diretta, mentre scegliendo un valore maggiore creeremo delle suddivisioni nei collegamenti e potremo optare anche per la modalità di interpolazione cubica mediante l'apposito menù (la modalità per un solo segmento è infatti *Lineare*, mentre con 2 o più segmenti possiamo scegliere). La modalità *cubica* ci consente di deformare le facce create per il collegamento conferendo loro una forma concava o convessa; per far ciò, variamo il valore del parametro *Strength*, provando sia valori positivi che negativi.

Questa è la modalità *Bridge* dello script; l'altra modalità è *Loft*, che consente di collegare alcune facce o spigoli, sempre appartenenti comunque ad uno stesso oggetto, un po' come la tecnica dello *skinning* o rivestimento per le superfici *NURBS*, mostrata in un mio precedente tutorial (presente in questa raccolta), ma questa volta per le mesh. Per provare *Loft*, aggiungiamo ad esempio una mesh Cerchio (potete provare con qualsiasi forma, questo è davvero un esempio di base).



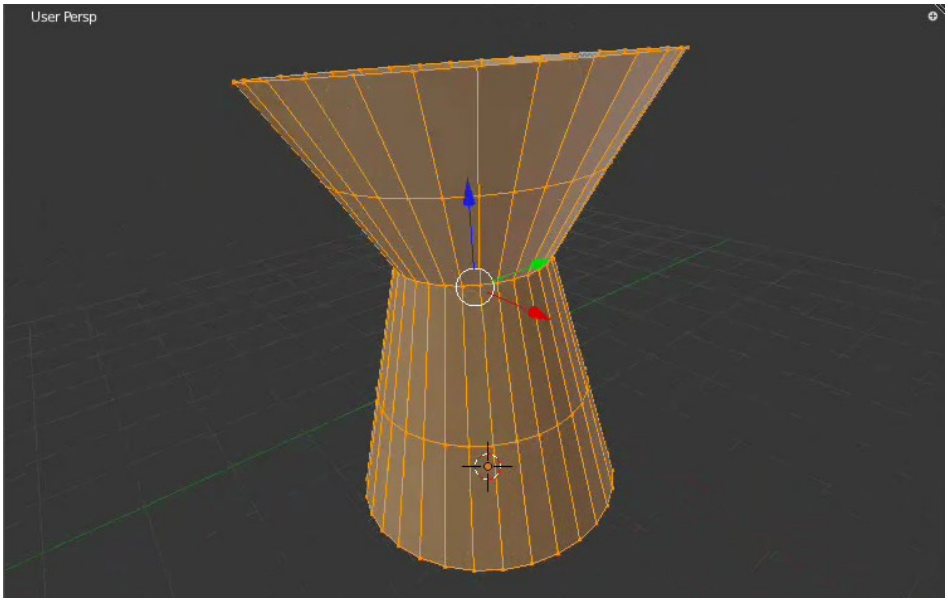
La creazione della mesh Cerchio, per provare LOFT

In Edit Mode duplichiamo la mesh e spostiamola un po', magari scalandola o ruotandola.
Ripetiamo quest'operazione un paio di volte; nel mio caso, sto traslando le copie lungo l'asse Z e le sto scalando un po'. Ricordo che dobbiamo lavorare in Edit Mode per mantenerci all'interno dell'oggetto di partenza.



Creazione delle copie, in Edit Mode (unico oggetto)

A questo punto possiamo aprire il menù *Mesh - Faces* nell'header di una *3D View* e scegliere la voce *Loft*, ed ecco la magia.



Effetti dell'utilizzo di LOFT

Con la voce *Segments*, in basso nella *Tool Shelf*, possiamo aumentare il numero di suddivisioni degli spigoli e delle facce create per collegare i vari profili, mentre con *Loop* possiamo chiedere allo script di chiudere la forma collegando automaticamente la prima faccia con l'ultima, solo che nel mio caso il risultato sarà deludente perché ho traslato i profili sull'asse Z, quindi vi suggerisco di fare delle prove con forme circolari o semicircolari, ad esempio disponendo i profili a "C".

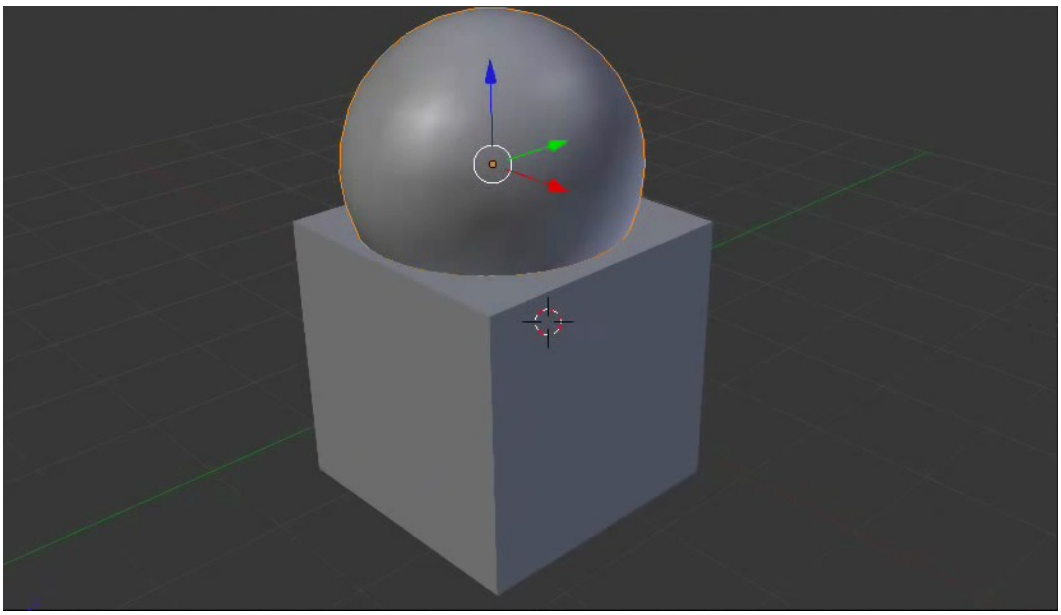
* * *

Join vs Boolean Union per i modelli STL e la stampa 3D

Questo più che un tutorial è un piccolo consiglio che voglio dare a chi fosse interessato a creare modelli *STL* per la prototipazione rapida in Blender 3D; in particolare, vedremo perché è preferibile l'utilizzo di *Boolean Union* al posto di *Join* per fondere le mesh.
Le stampanti 3D accettano in genere il formato *STL*; va detto però che i modelli devono essere il più possibile "puliti", ossia appartenenti ad un'unica mesh e "ben cuciti".

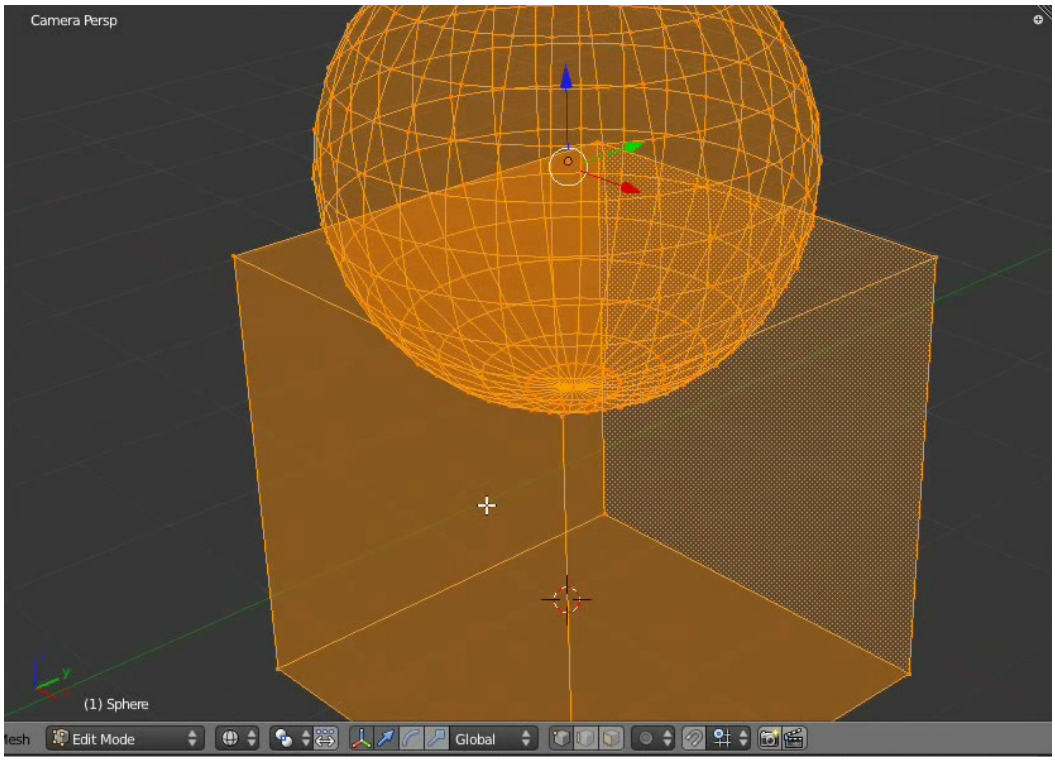
Ovviamente è impensabile realizzare mesh complesse come modellini di aerei, statue ed altro utilizzando un'unica mesh di partenza come un piano o un cubo: è molto più facile creare i vari pezzi, utilizzando magari superfici *NURBS* o metaoggetti quando preferibile, e fondere il tutto.
Vediamo, con un esempio, perché conviene utilizzare il modificatore *Boolean* di Blender piuttosto che il classico *Join*, per effettuare appunto l'operazione di fusione.

Inseriamo nella scena, ad esempio, un cubo e una sfera, posizionando la sfera sopra il cubo con una piccola intersezione con lo stesso.



La scena utilizzata in questo tutorial

Selezionando entrambi gli oggetti e cliccando su *Join*, nella *Tool Shelf*, creeremo una nuova mesh unione delle due di partenza, ma attenzione: come possiamo constatare facilmente passando in *Edit Mode*, Blender non sta creando delle connessioni tra le facce e non sta creando vertici o spigoli per unire gli oggetti, ma si sta limitando ad inserire le informazioni sui vertici e le facce delle due mesh in un unico oggetto, ignorando le intersezioni tra i volumi.



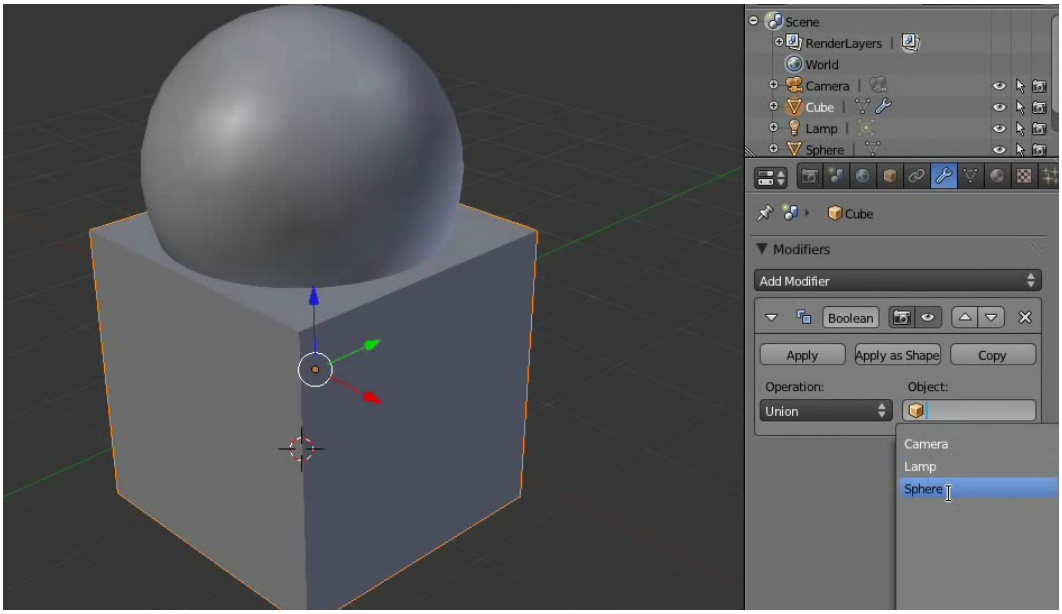
Effetti di JOIN tra le mesh, in Edit Mode o in visualizzazione Wireframe

Questo tipo di fusione a volte non è buono per i file *STL* per la prototipazione rapida.

Annulliamo il *Join* premendo *CTRL Z* ed esaminiamo la seconda modalità.

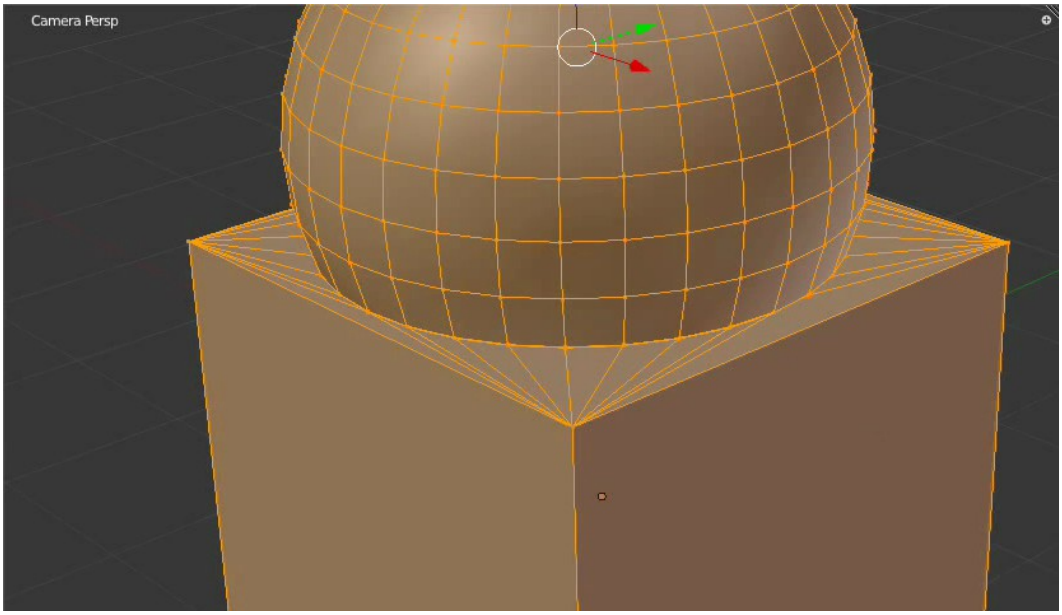
Selezioniamo uno dei due oggetti, ad esempio il cubo, ed apriamo la scheda *Modifiers*, all'interno della *Properties Window.*

Aggiungiamo un modificatore di tipo *Boolean*, cambiando il tipo di operazione (di default, l'intersezione) in unione e selezioniamo come oggetto *Target* della modifica la sfera, dopodiché applichiamo il modificatore.



Utilizzo del modificatore Boolean - Union

Passando adesso in Edit Mode, noteremo che Blender ha generato veramente un unico oggetto, nel senso che dove c'erano le intersezioni tra le mesh ha creato degli spigoli, come se avesse utilizzato il taglierino *knife* in quei punti, ha rimosso le facce superflue e ha creato gli spigoli di collegamento tra i vertici esistenti e i nuovi.



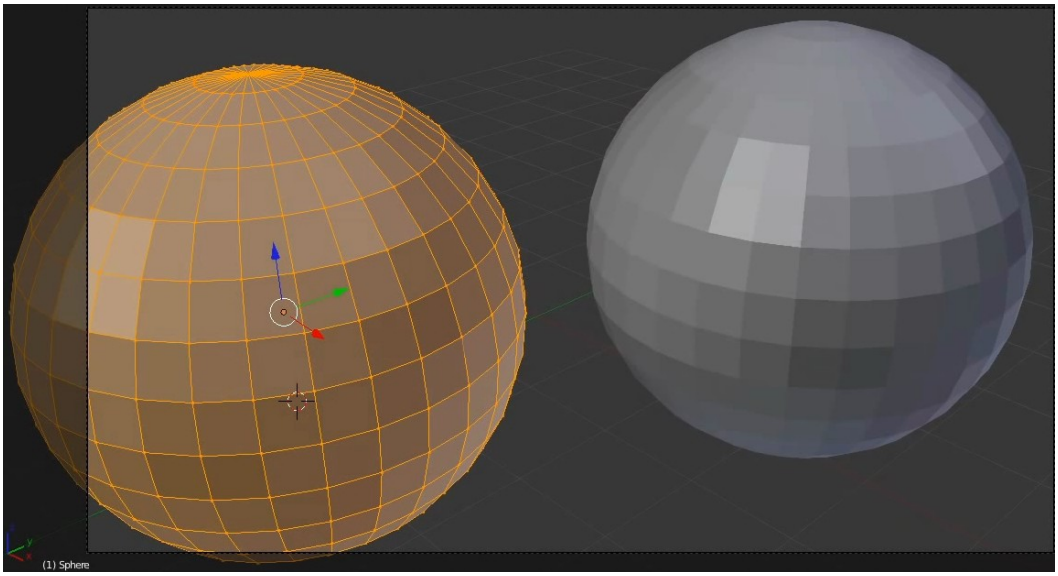
Effetti dell'utilizzo di Boolean - Union (dopo aver applicato il modificatore)

La differenza tra questa tecnica e Join è evidente anche dal punto di vista pratico, adesso; generalmente, come vi dicevo all'inizio, conviene utilizzare l'unione booleana se si ha intenzione di esportare le mesh in *STL* per la stampa 3D, altrimenti *Join* può andare.

* * *

Strumenti di selezione di base ed Edge Loop e Ring

Questo è un tutorial riassuntivo sulle diverse modalità di selezione di oggetti e parti di essi, quindi sia in Object che in Edit Mode, in Blender 3D 2.5; rivedremo quindi strumenti ben noti e altri poco conosciuti. Iniziamo aggiungendo nella scena due oggetti, ad esempio due sfere UV, ben distinte.



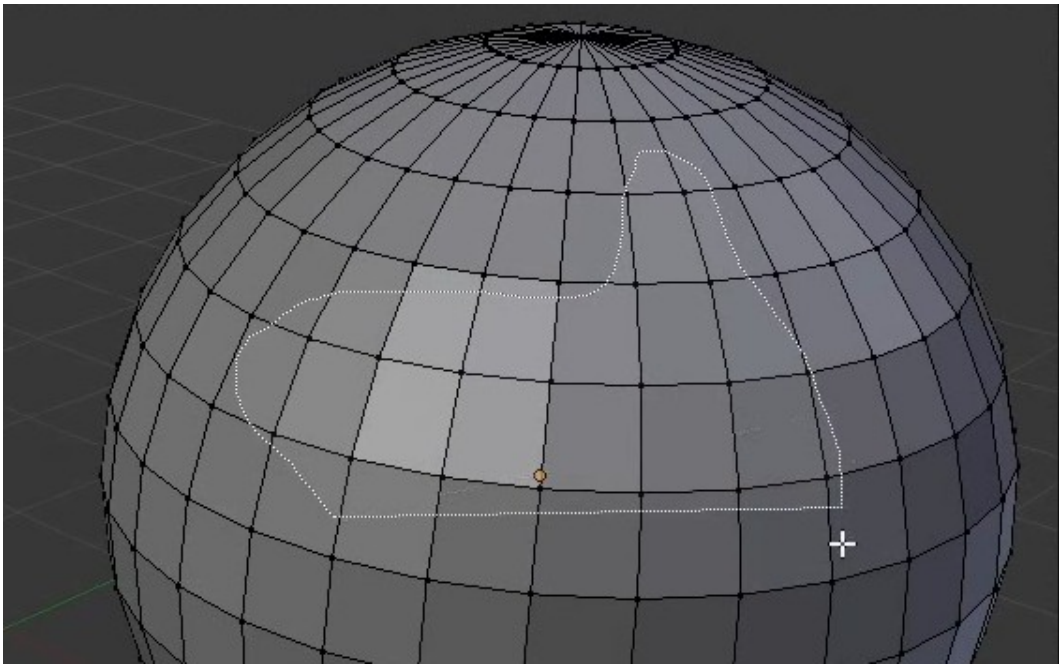
La scena utilizzata in questo tutorial

Possiamo selezionare e deselezionare tutti gli oggetti presenti nella scena premendo, in entrambi i casi, *A*; effettuata in Edit Mode, questa operazione ci consente di selezionare o deselezionare al volo tutti i vertici, gli spigoli e le facce di una mesh, oppure i punti di controllo e le maniglie di una *Curva* o un oggetto *Surface*.

La selezione singola avviene, come sappiamo, con un click del tasto destro del mouse sull'oggetto che vogliamo selezionare, e questo vale sia con oggetti veri e propri in Object Mode che con parti di essi, come vertici, spigoli, facce o punti di controllo in Edit Mode.

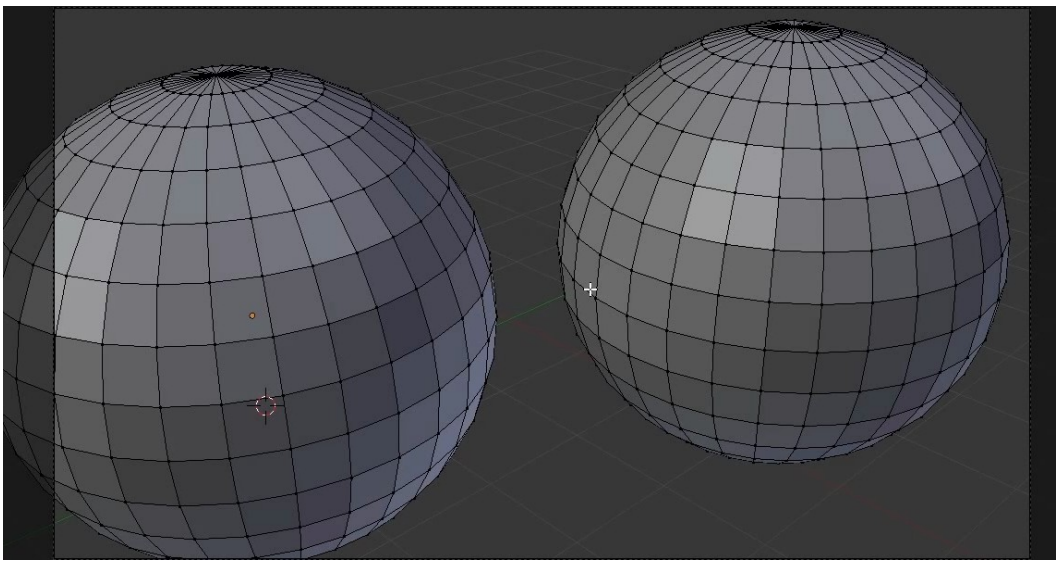
Per effettuare una selezione multipla, ancora una volta sia in Object che in Edit Mode, possiamo procedere in più modi, ossia:

- *SHIFT* e click col tasto destro del mouse sugli elementi da selezionare, uno per uno;
- *B* e definizione, con click del tasto sinistro del mouse e trascinamento, di un'area di selezione rettangolare;
- *C*, movimento della rotellina del mouse (o utilizzo dei tasti + e -) e click del tasto sinistro del mouse per utilizzare un'area di selezione di forma circolare e raggio variabile;
- *CTRL* e clic col tasto sinistro del mouse, con trascinamento, per definire un'area di selezione *Lazo*, cioè la selezione a mano libera per intenderci.



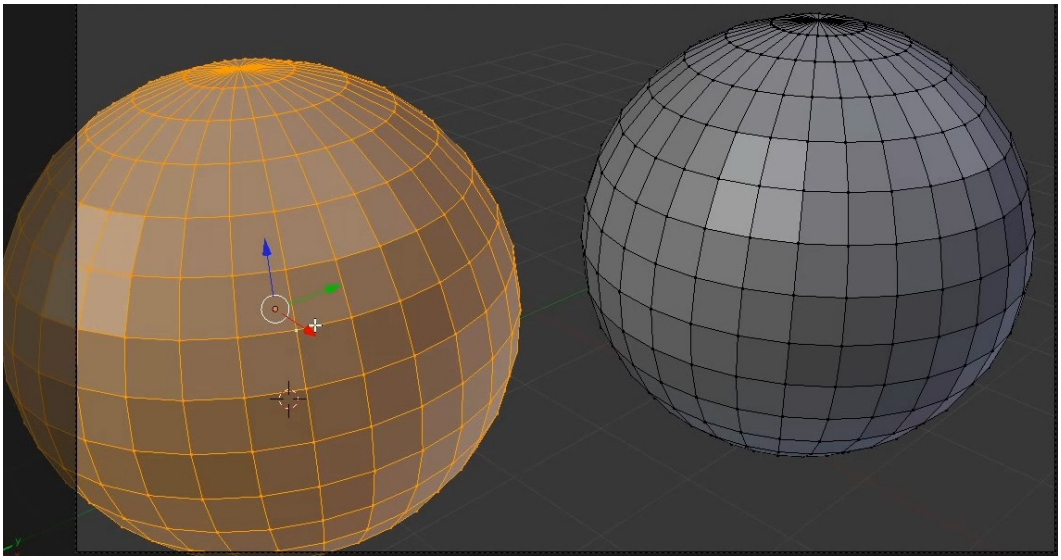
Utilizzo dello strumento di selezione "Lazo", con CTRL e click del tasto sinistro

Esistono poi strumenti di selezione particolarmente utili quando una mesh è composta da più parti distinte e si desidera selezionare al volo una sottoparte; per mostrare questi strumenti, fondiamo le due sfere mediante *CTRL J*, l'unione con *Join*, in modo da lasciarle comunque separate nello spazio, anche se fanno parte di un'unica mesh, e passiamo in Edit Mode.



Le mesh fuse, in Edit Mode

Selezioniamo un vertice di una delle due sfere e premiamo *CTRL L* (nota: solo *L*, nelle versioni più recenti): Blender selezionerà al volo tutti i vertici connessi, direttamente o indirettamente, al vertice o, in generale, alla selezione di partenza; *L* sta infatti per *linked*, cioè connesso, e nel nostro caso ci consente di selezionare tutti i vertici della sfera alla quale appartiene il vertice iniziale.

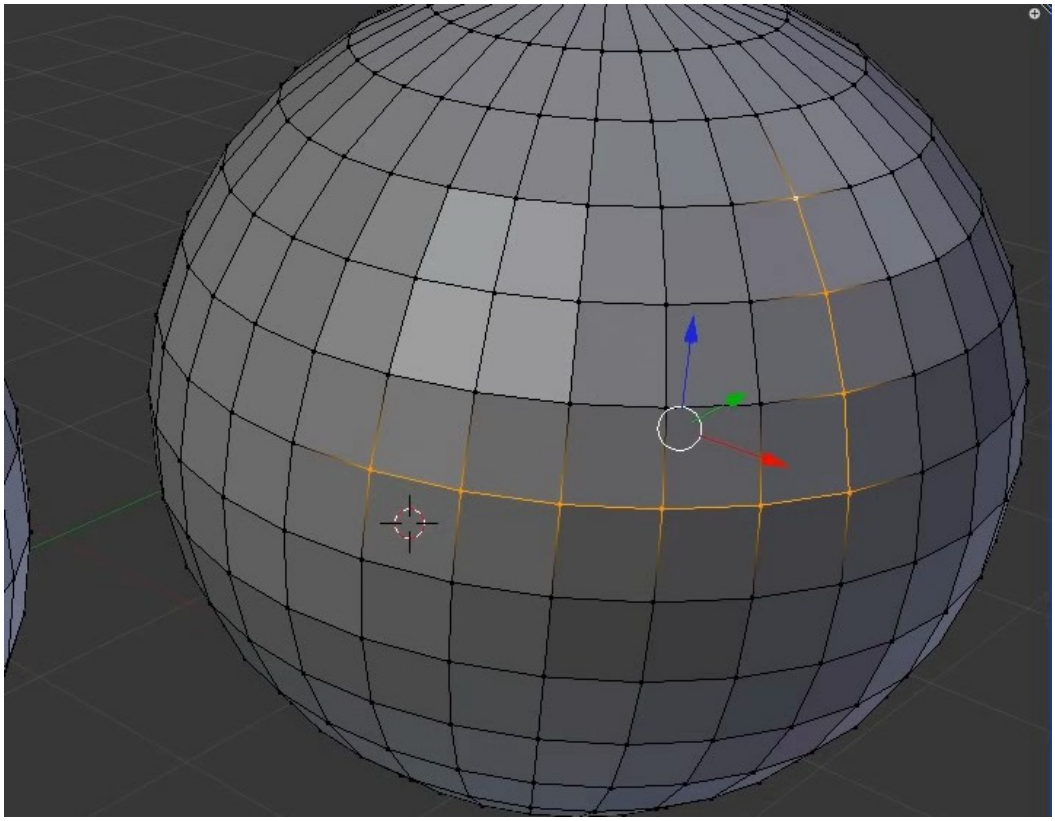


Utilizzo di L (Linked): dopo aver selezionato un vertice, premere L per selezionare in un colpo tutti i vertici connessi (direttamente e indirettamente) a quello selezionato

Un altro strumento di selezione di una sottoparte di una mesh composta da più elementi è dato dall'utilizzo di *L* senza una selezione iniziale, semplicemente posizionandosi sopra l'elemento da selezionare; deselezioniamo quindi tutto con *A*, poniamo il cursore del mouse su un punto qualsiasi di una delle due sfere, in Edit Mode, e premiamo *L*.

L'operazione inversa, la deselezione di tutte le voci che appartengono ad un oggetto collegato, avviene con *SHIFT L*; è utile, ad esempio, se abbiamo selezionato per sbaglio un oggetto e parti di un altro, per cui per togliere queste parti è sufficiente posizionarsi sul secondo elemento e premere *SHIFT L*. Deselezioniamo tutto e rimaniamo in modalità selezione vertici, quella di default, per mostrare l'utilizzo di *Vertex Path*, che seleziona vertici e spigoli per collegare due vertici distanti, definendo quindi un percorso (*Path*, appunto).

Selezioniamo quindi due vertici distanti con una selezione multipla, dopodiché scegliamo la voce *Vertex Path* dal menù *Select* nell'header di una finestra *3D View* ed osserviamo il risultato.

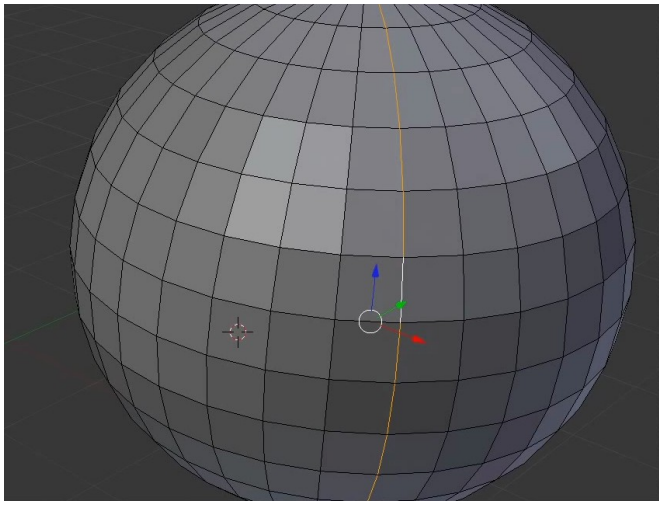


Utilizzo di Vertex Path: Blender selezionerà i vertici del "percorso più breve" tra i due vertici estremi selezionati

Passiamo ora in modalità selezione spigoli per mostrare l'utilizzo di *Edge Loop* ed *Edge Ring*, e qui le sfere ci tornano particolarmente utili per capire il funzionamento di questi particolari strumenti.

Selezioniamo un solo spigolo, di una qualsiasi delle sfere, dopodiché clicchiamo su *Edge Loop* nel menù *Select*, nell'header della *3D View*.

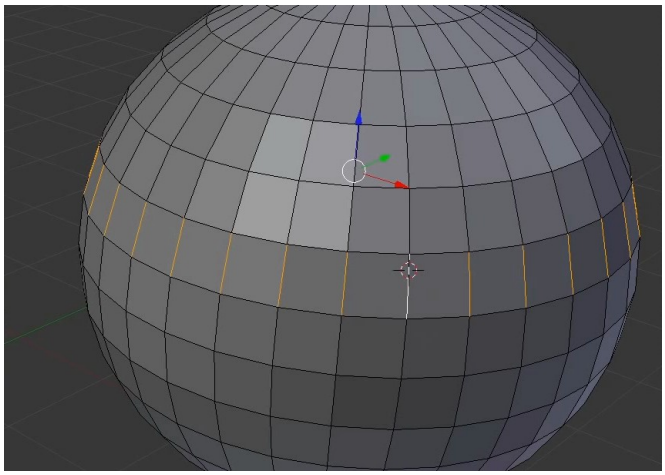
Com'è possibile osservare nell'immagine a lato, *Edge Loop* seleziona gli spigoli direttamente collegati ai vertici dello spigolo originale, in maniera ricorsiva (cioè non si ferma agli spigoli immediatamente vicini); in pratica "segue" il percorso all'interno del quale si trova lo spigolo originale.



Utilizzo di Edge Loop: dopo aver selezionato un solo spigolo (quello bianco, nell'immagine) e scelto Edge Loop, Blender selezionerà tutti gli spigoli del "percorso" al quale appartiene quello selezionato

Deselezioniamo tutto e selezioniamo nuovamente un solo spigolo iniziale, dopodiché clicchiamo su *Edge Ring* nel menù *Select*, nell'header della *3D View*.

Anche in questo caso è più facile capire cosa fa questo strumento osservando il risultato a video piuttosto che descrivendolo a parole: in pratica, consente di selezionare gli spigoli paralleli a quello originale, creando appunto un anello intorno alla mesh.

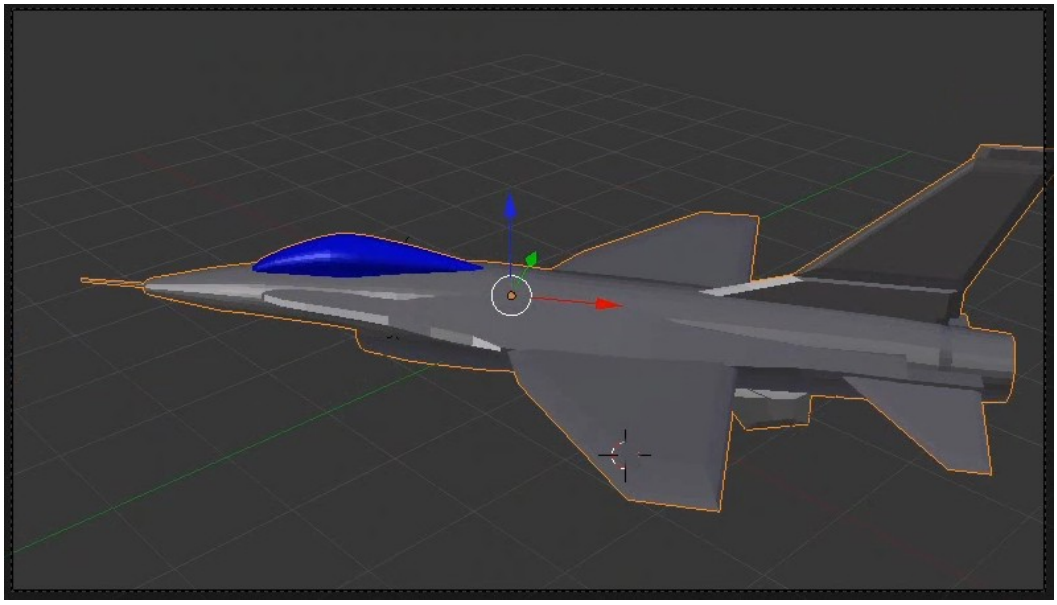


Utilizzo di Edge Ring: dopo aver selezionato un solo spigolo (quello bianco, nell'immagine) e scelto Edge Ring, Blender selezionerà tutti gli spigoli "paralleli"

* * *

Bande colorate su oggetti con Textures procedurali

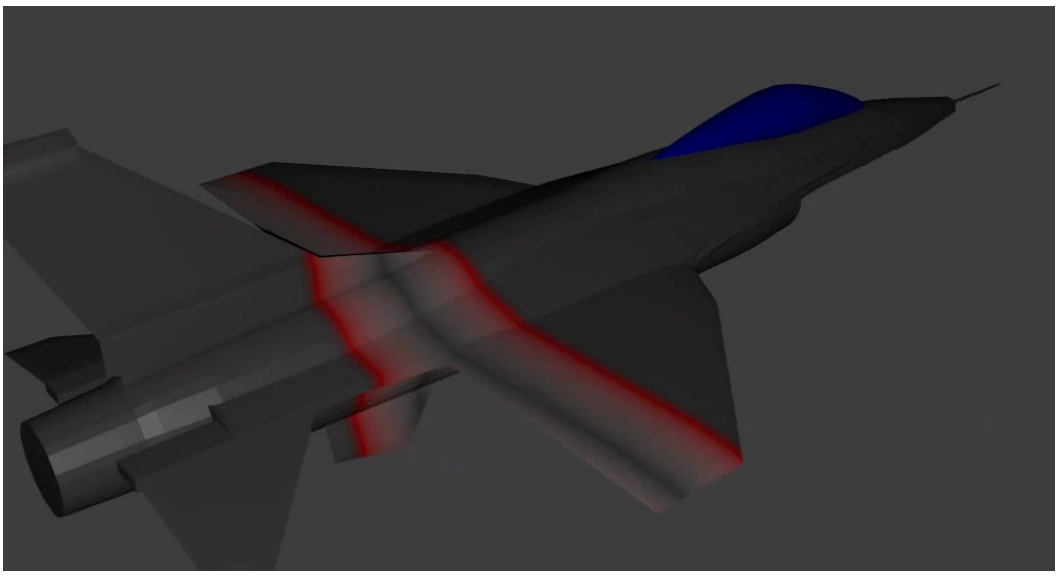
Questo tutorial è tratto dal mio *ebook "Materiali e Textures in Blender 3D 2.5 – Volume 2: le Textures"*; in particolare, in questo tutorial vedremo come applicare delle bande colorate (realizzate mediante Textures procedurali) agli oggetti.



La scena utilizzata in questo tutorial

A volte, come nel caso della realizzazione di loghi o effetti speciali, si ha la necessità di aggiungere, quasi in sovrimpressione, linee o vere e proprie bande colorate (anche in movimento) sugli oggetti. In questi casi non è necessario ricorrere a Textures immagine dotate di trasparenza: la soluzione qui proposta fa uso di una sola Texture procedurale di tipo *Blend Spherical*.

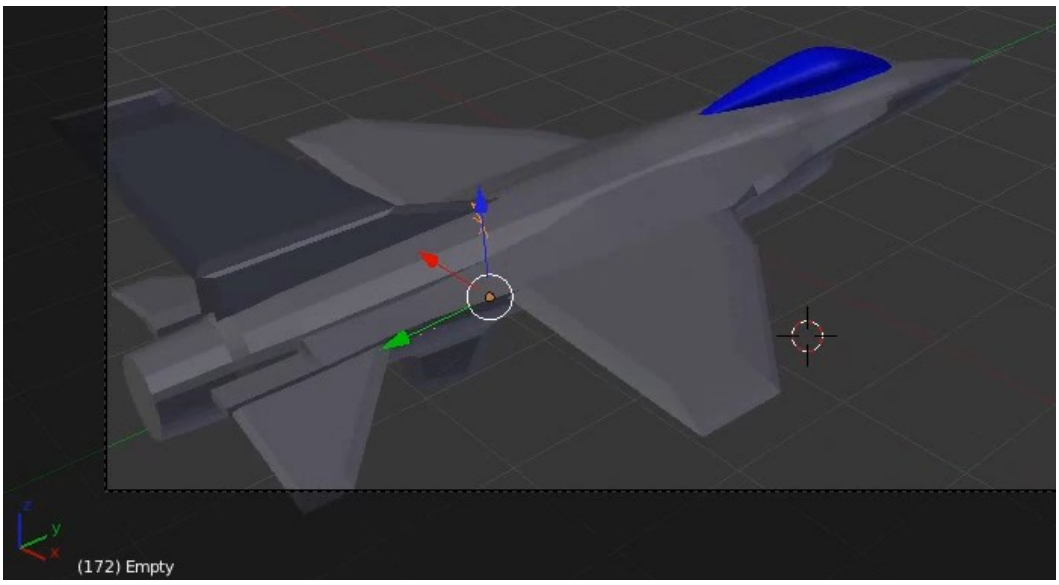
L'immagine seguente mostra un fotogramma dell'animazione che verrà realizzata seguendo questo tutorial: una banda colorata rossa "scorre" lungo un modello 3D. Si tratta di un effetto speciale presente in molti film di azione o fantascienza (dove, ad esempio, i protagonisti esaminano i dati di un particolare oggetto visualizzando quest'ultimo, e l'effetto descritto, su schermi o ologrammi).



Un fotogramma dell'animazione che otterremo

L'animazione dello scorrimento lungo il modello è stata ottenuta mappando le coordinate di proiezione della Texture mediante una *Empty*, con *Coordinates Object* nella scheda *Mapping* della Texture, traslando quindi la *Empty* lungo il modello nel corso dell'animazione: tale operazione ha l'effetto di “trascinare” la banda.

Nella *3D View* possiamo notare (nell'immagine seguente) la presenza del modellino e della *Empty* utilizzata per la mappatura. La *Empty* scorre lungo il modello, trascinando con sé la Texture.



La Empty trasla lungo il suo asse Y Locale, "attraversando" il modellino (e "portando con sè" la Texture, tramite un'opportuna mappatura)

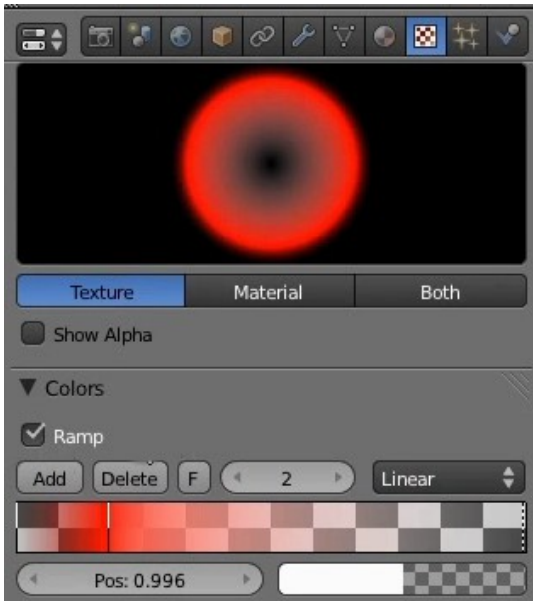
Per animare la rotazione degli oggetti e la traslazione della *Empty* si procede animando prima la traslazione della *Empty* lungo il modello immobile, imparentando poi la *Empty* al modello (come figlia di tale oggetto), animando infine la rotazione del solo modellino: la *Empty*, che effettua già uno spostamento, erediterà anche la rotazione dell'oggetto padre (il modellino), consentendoci di ottenere con semplicità un'animazione "non banale".

Il modellino è provvisto di un Material multiplo, per separare l'aspetto della cabina di pilotaggio da quella del resto dell'aereo (dotato, in questo caso, di un materiale grigio senza riflessi speculari e una Texture *Clouds* con *Size* 2.0 e colore di influenza grigio scuro).

Il cuore della tecnica sta comunque nella mappatura della banda, che come detto è una Texture Blend Spherical, che di default produce un gradiente su tutte e tre le direzioni.

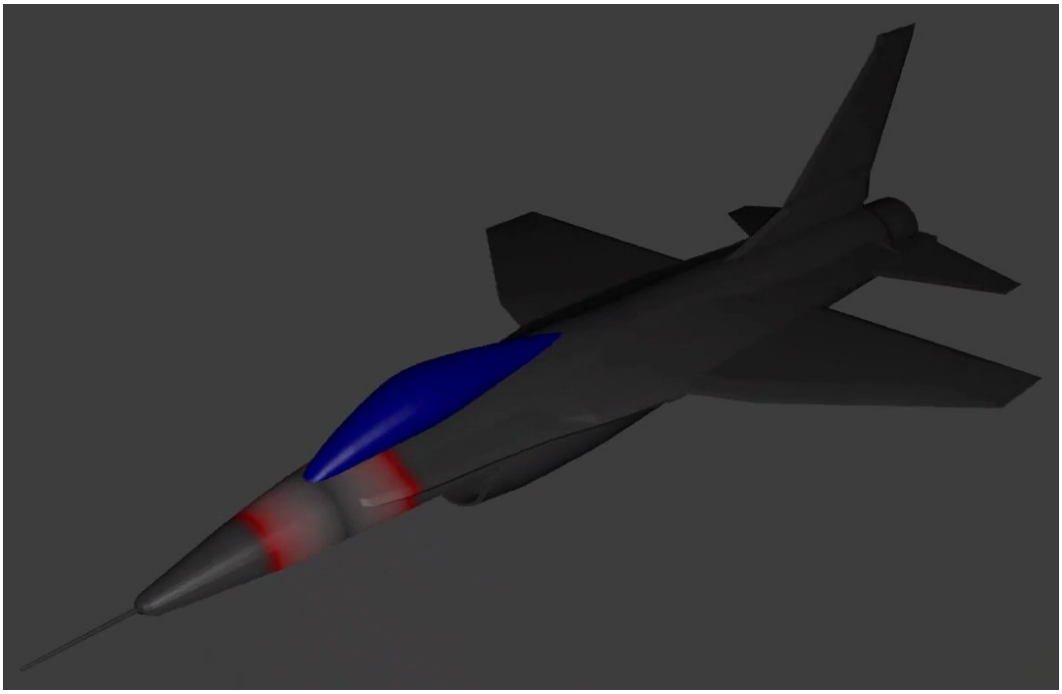
Per ottenere, da tale Texture procedurale, un "anello" da mappare su un oggetto, è necessario effettuare tre operazioni:

1. cambiare la forma del gradiente da cerchio ad anello, definendo nel mentre il colore dell'effetto, utilizzando la *Color Band*, ottenendo un anello completamente trasparente all'interno e opaco nella parte esterna, con una transizione personalizzabile a piacere;
2. nella scheda *Mapping* della Texture, cambiare il tipo di mappatura in *Object* (specificando, nel campo che apparirà, di utilizzare l'oggetto *Empty* definito per l'occasione) e cambiare le coordinate di proiezione in *Sphere*, in modo da coinvolgere tutto l'oggetto;
3. cambiare infine (sempre nella sezione *Mapping* della Texture) le coordinate di proiezione *X Y Z* in *None Y None*, annullandone cioè due e mantenendo quella che rappresenta l'asse locale della *Empty* lungo il quale la stessa "scorrerà" (nella *3D View*, si nota che la *Empty* si sposta lungo il suo asse *Y locale*, mentre attraversa il modello 3D).



La Color Ramp per modificare l'anello (punto 1)

Quelle date sono, ovviamente, solo delle linee guida: l'effetto è ampiamente personalizzabile.



Altro screenshot dell'animazione finale

* * *

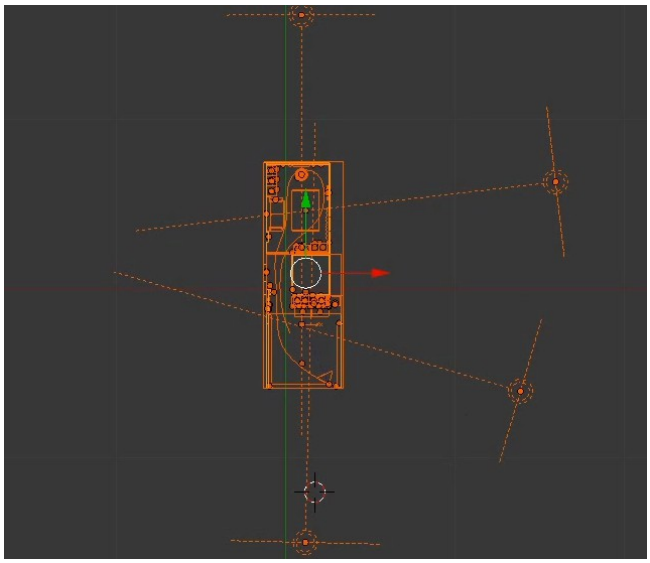
Indirect Lighting in Blender 2.5, esempi di rendering

Questo più che un tutorial è l'analisi di alcuni rendering effettuati in Blender 3D e *Yafaray* utilizzando, in alcuni casi, la nuova funzionalità (per la versione 2.5) *Indirect Lighting*.
Blender, si sa, ha sempre avuto qualche problemino con l'illuminazione indiretta, non integrando un motore di illuminazione globale di base, o almeno di *pseudo-Global Illumination*, fatta eccezione per *Radiosity* e Ambient Occlusion.

Dalla versione 2.5, è presente anche la voce *Indirect Lighting*, che serve a simulare appunto i rimbalzi e la luce indiretta. In questo tutorial non vedremo come utilizzare *IL*, solo qualche rendering d'esempio. Ho utilizzato due scene, una con luce per così dire diurna e una con illuminazione notturna. Per tutti i rendering ho utilizzato un sistema con core i7 870 (quindi 4 cores, 8 thread), 8 GB di RAM e versione 64bit di Blender 2.56, tranne nel caso di *Yafaray* dove ho utilizzato Blender 2.49 e *Yafaray* 0.1 a 32 bit.

La scena utilizzata per questo tutorial

Iniziamo esaminando quella con luce diurna: abbiamo due vani con 4 fonti di luce *Area Lamp* ad avvolgerli per metà (la metà provvista di finestre, per la precisione) e in tutte le finestre e i balconi sono presenti delle tende semitrasparenti che filtrano la luce in arrivo dall'esterno, quindi tutti i materials della scena hanno l'opzione *Receive Transparent* selezionata.



Il set della scena "diurna", con le 4 fonti di luce Area Lamp

Il rendering visibile qui di seguito è il rendering di base, in risoluzione 1280x720, ottenuto con *Blender Internal* utilizzando solo *Area Lamps* senza *Indirect Lighting*; il tempo di rendering è di 19 minuti.

Il rendering in Blender Internal senza Indirect Lighting

Attivando invece *Indirect Lighting* con 1 rimbalzo e 6 passi del motore *Approximate* si ha, invece, il seguente risultato, ottenuto con tempo di rendering di 33 minuti a 1280x720.

Il rendering in Blender Internal con Indirect Lighting

L'importanza del numero di rimbalzi e dei *Samples* utilizzati è notevole, ma non è l'unico parametro da considerare: bisogna tenere conto anche del numero di *Samples* presi per le fonti di luce; di seguito, ecco tre rendering: nel primo caso, 1 rimbalzo e 1 *Sample* per IL, 2x2 *Samples* per *Area Light*, tempo di rendering 6 minuti a 1280x720;

Primo di tre rendering con valori diversi per i Samples

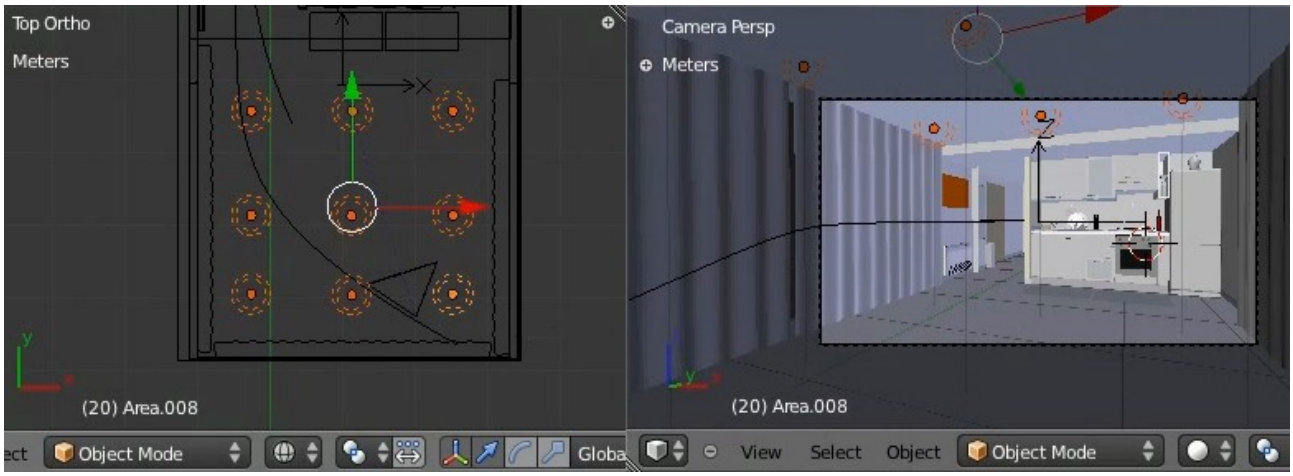
nel secondo caso, 1 rimbalzo e 2 *Samples* per IL, 6x2 *Samples* per *Area Light*, tempo di rendering 8 minuti a 1280x720;

Secondo di tre rendering con valori diversi per i Samples

nel terzo caso, 2 rimbalzi (che contribuiscono a rendere più luminosa la scena) e 4 *Samples*, 12x4 *Samples* per *Area Light*, tempo di rendering 18 minuti a 1280x720.

Terzo di tre rendering con valori diversi per i Samples

Passiamo ora alla scena notturna, ottenuta eliminando le *Area Lamps* ed inserendo 9 fonti di luce Omnidirezionali a reticolo sul soffitto.



La scena notturna

Il primo rendering è stato realizzato con *Blender Internal* senza *Indirect Lighting* a risoluzione 1280x720 con 12 *Samples* per le ombre e ha richiesto 100 minuti.

Il rendering "in notturna" senza Indirect Lighting

Il secondo rendering è stato realizzato con *Blender Internal* a 1280x720 con 12 *Samples* per le ombre delle fonti di luce, quindi come quello precedente, ma utilizzando anche *Indirect Lighting* con un solo rimbalzo e 5 passi del motore *Approximate*; questo rendering ha richiesto 120 minuti, appena 20 minuti in più del precedente.

Il rendering "in notturna" con Indirect Lighting Approximate

I risultati senza illuminazione indiretta sono paragonabili, a volte, a quelli ottenibili con *Yafaray*, solo che con quest'ultimo i tempi di rendering sono decisamente inferiori; l'immagine seguente, ad esempio, è stata realizzata in appena 20 minuti da Yafaray con le solite 9 Lamps sul soffitto (per l'illuminazione notturna) e modalità di rendering *Direct Lighting*, e per di più in risoluzione FULL HD 1920x1080, non 1280x720 come le altre, quindi 20 minuti per una risoluzione decisamente maggiore!

Il rendering realizzato con Yafaray

* * *

Secondo confronto Blender 3D 2.5 con IL e Yafaray; cucina, luce laterale

Ecco qui un altro confronto tra rendering, in Blender 3D 2.5 e *Yafaray*, della scena della cucina, questa volta con un'impostazione leggermente diversa delle luci; in particolare, in questo caso si ha una sola *Area Lamp* sulla destra, per simulare un'illuminazione da luce solare, leggermente di taglio, ed esaminare l'illuminazione derivante dai rimbalzi e dalla luce indiretta.

La scena utilizzata in questo tutorial

Il sistema utilizzato per i rendering è un i7 870 (4 cores, 8 threads) con 8 GB di RAM, Blender 2.56a 64bit per i rendering in Blender con e senza *IL* e Blender 2.49 con *Yafaray* 0.1.1, entrambi a 32 bit, per il rendering in *Yafaray*; la risoluzione dei rendering è 1280x720.

Iniziamo dal primo rendering, effettuato con Blender 3D 2.56a, una sola fonte di luce *Area Lamp* laterale, senza illuminazione indiretta: l'assenza della *Global Illumination* "nativa" è evidente, basta osservare la parete non illuminata vicino al frigorifero o, su quest'ultimo, il vaso, che sembra quasi

metallico. Aumentare l'intensità della fonte di luce non risolve questi problemi, serve solo a rendere più chiare le pareti e gli oggetti illuminati, non ha effetti sulle parti non in luce.

Il primo rendering, senza Indirect Lighting

La situazione cambia radicalmente con il secondo rendering, ottenuto con le stesse impostazioni del primo ad eccezione di *Indirect Lighting*, attivato con 1 solo rimbalzo e 6 passi di *Approximate Gathering*: forse l'intensità della fonte di luce è bassa, ma almeno la scena è tutta illuminata ed è anche priva di rumore (non fate caso a quella strana luce in alto a destra, che deriva da un errore del modello 3D, non della luce).

Il secondo rendering, con Indirect Lighting

Veniamo al terzo rendering, effettuato con *Yafaray*: la quantità di luce è quella giusta, basta osservare la zona del lavabo, ma c'è un problema: il rumore, evidente anche dopo aver fatto uso di 100 milioni di fotoni e 32 passi di *Final Gathering* (ovviamente il tipo di rendering è *Photon Mapping*, da utilizzare proprio per gli interni).

Il terzo rendering, realizzato con Yafaray

Ad una prima occhiata, quindi, il rendering con *Indirect Lighting* sembra il migliore, superiore anche a quello ottenuto con *Yafaray*, ma ci sono due considerazioni da fare: la prima riguarda i tempi di rendering, la seconda i "trucchetti".

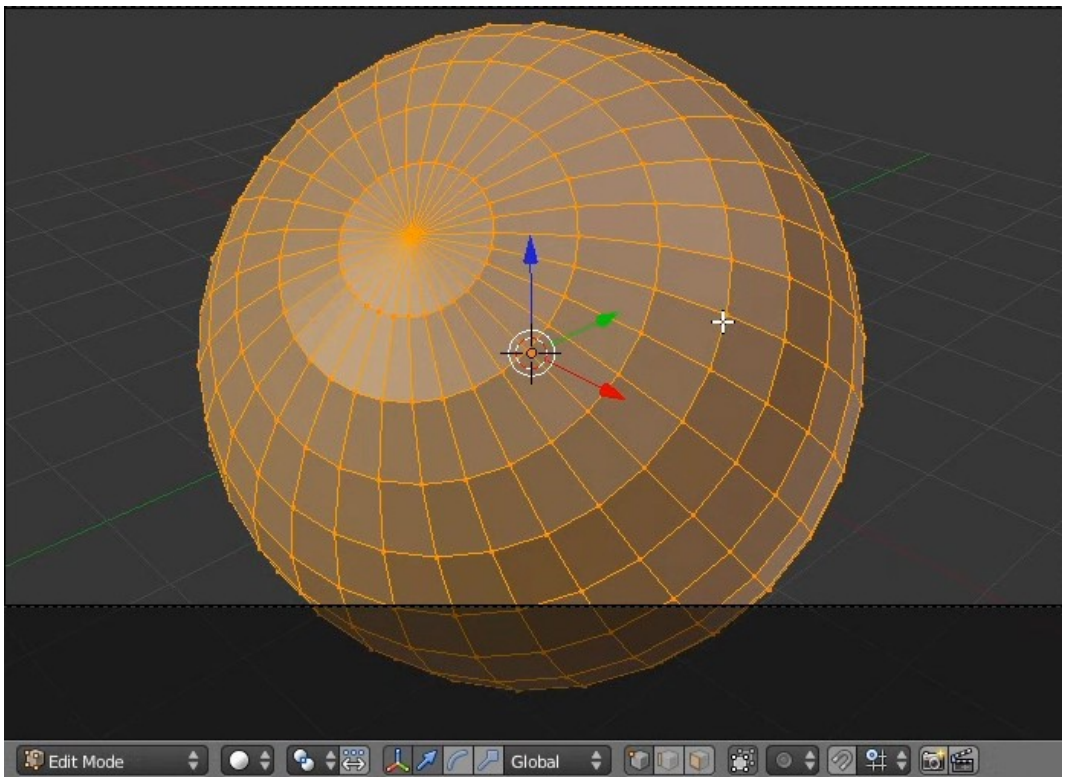
Iniziamo dai tempi: in questo confronto ne esce vincitore *Yafaray*, in quanto il primo rendering (*Blender Internal* senza *IL*) ha richiesto 25 minuti, il secondo (*Blender Internal* con *IL*) 32 minuti e il terzo (*Yafaray*) appena 7 minuti; per un'immagine statica possiamo aspettare 32 minuti, per un breve filmato decisamente no!

La seconda considerazione riguarda i trucchetti: in genere, soprattutto per rendering statici, non è necessario cercare di simulare l'illuminazione "reale" ponendo le fonti di luce così come sono nella realtà (ad esempio, per i rendering in studio è noto l'utilizzo delle tre luci), per cui è possibile ottenere risultati migliori con una o due luci di riempimento o con valori molto bassi di *Emit* per i materiali, allo scopo di ridurre il rumore, soprattutto con *Yafaray*.

* * *

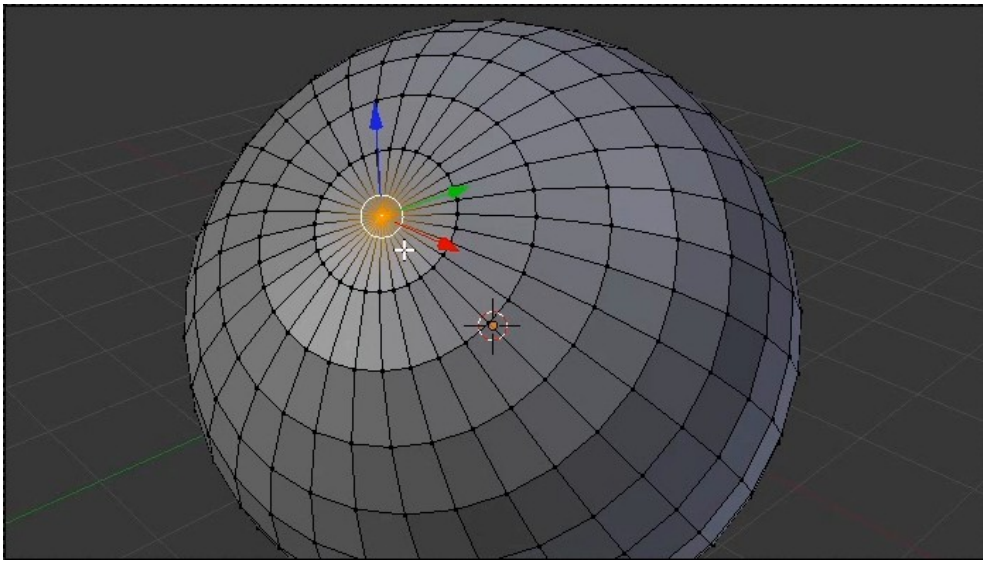
Tips and Tricks modellazione: allargare o restringere una selezione

Questo più che un tutorial è un piccolo "Tips & Tricks" di modellazione in Blender 3D; in particolare, vedremo come selezionare con facilità le facce direttamente collegate a quelle già appartenenti ad una selezione, ed eventualmente come contrarre la selezione eliminando le facce per così dire "periferiche". Aggiungiamo una mesh, ad esempio una *sfera UV*, e passiamo in Edit Mode.



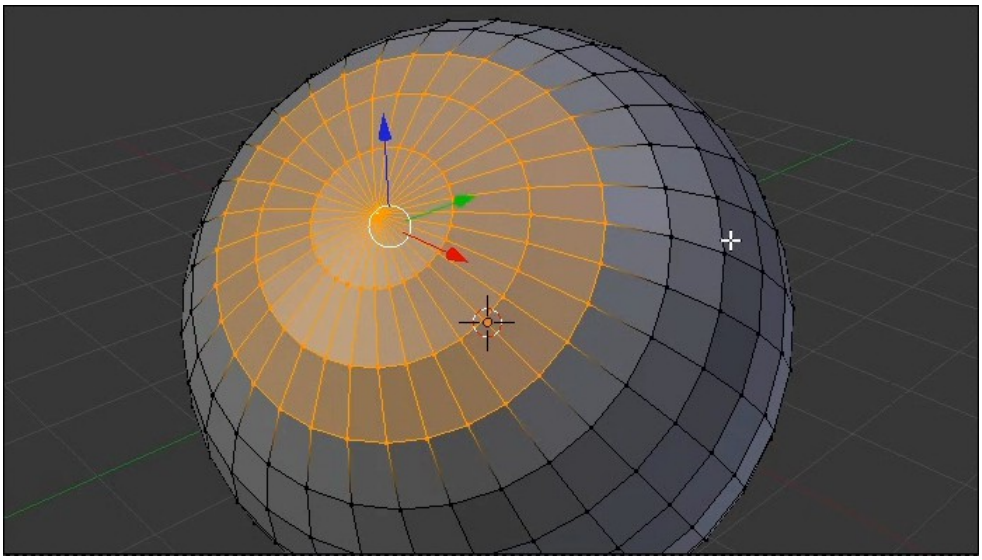
La scena utilizzata per questo esempio

Lo shortcut funziona sia in modalità selezione vertici che spigoli o facce, per cui restiamo anche in modalità selezione vertici e selezioniamo, ad esempio, uno dei poli della sfera, mediante click del tasto destro del mouse sull'elemento.



La selezione di un vertice, in Edit Mode

A questo punto premiamo la combinazione di tasti *CTRL* + : come è possibile notare a video, Blender aggiungerà alla selezione gli elementi (vertici, spigoli e facce) direttamente collegati a quelli già presenti (nel nostro caso, solo un vertice); l'operazione può essere ripetuta, per cui possiamo ampliare la selezione seguendo lo stesso criterio premendo ancora *CTRL* +.



Effetti di "CTRL +" sulla selezione originale

Ovviamente, nulla ci vieta di aggiungere altri elementi in maniera libera con *SHIFT* e click del tasto destro del mouse oppure *B* o *C*.

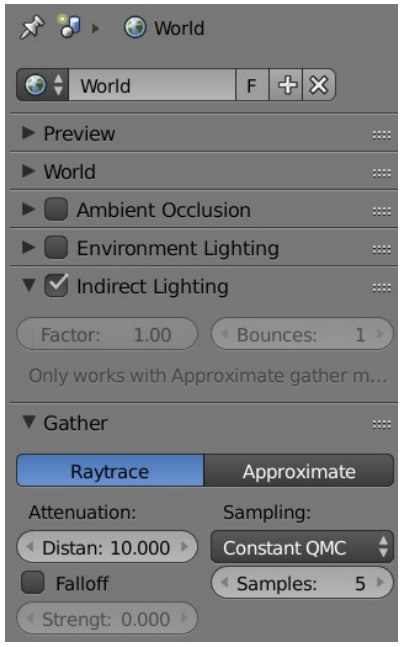
È possibile inoltre eliminare gli elementi periferici di una selezione con un'operazione per così dire di erosione; neanche a dirlo, la combinazione di tasti per effettuare quest'operazione è *CTRL* -.

* * *

L'illuminazione indiretta (Indirect Lighting) in Blender 3D 2.5

Questo tutorial è dedicato alla funzionalità *Indirect Lighting* di Blender 3D 2.5; ho già pubblicato degli articoli contenenti anteprime dei risultati ottenibili con questo strumento ma non l'ho descritto a dovere, mentre questo è effettivamente un tutorial.

Indirect Lighting calcola i rimbalzi dei raggi luminosi consentendo di ottenere, come suggerisce il nome, la luce indiretta; questa è una caratteristica base dei motori di rendering con *Raytracing* e illuminazione globale (come *Yafaray*, *Octane*, *V-Ray*, *LuxRender*, ecc...) che finora mancava in Blender 3D, anche se nelle versioni precedenti del programma era presente *Radiosity*.

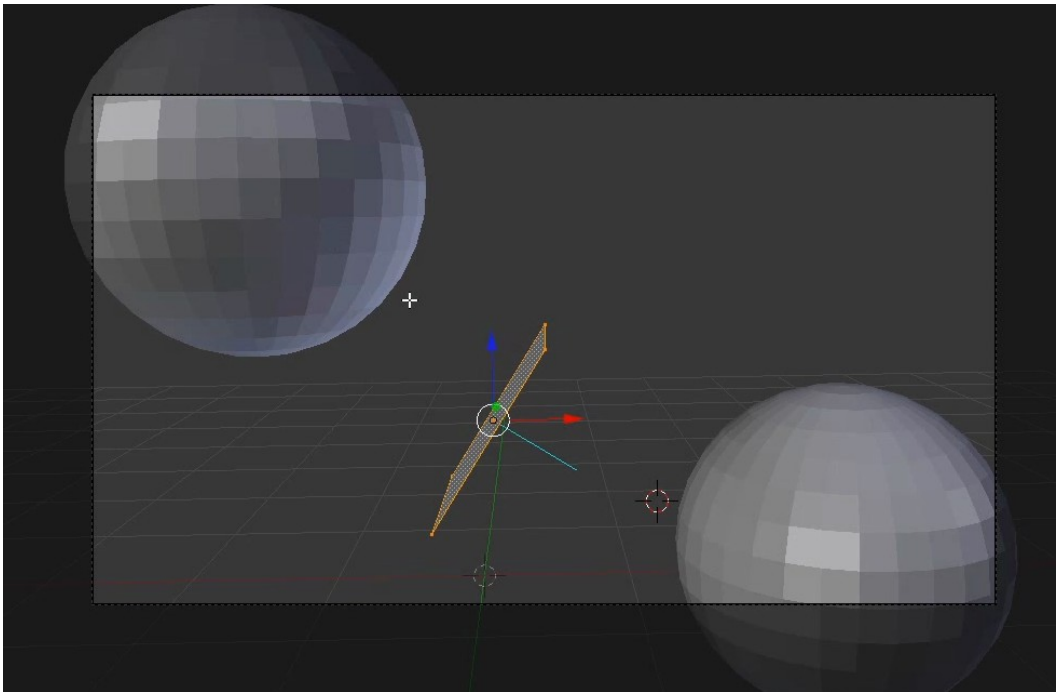


Le voci di Indirect Lighting nella scheda World della Properties Window

Indirect Lighting si appoggia al motore utilizzato per *Ambient Occlusion*, per cui la sua scheda si trova nella scheda *World* all'interno della *Properties Window* e, almeno fino alla versione 2.56a del programma, l'unico *Gathering Method* consentito è *Approximate*: *Raytrace* non è disponibile.

Indirect Lighting funziona principalmente in due modi: calcola la luce di rimbalzo per le fonti di luce e rende davvero emettitori i Materials con *Emit* maggiore di 0; in questo tutorial vedremo questi due aspetti iniziando da una scena di base.

Per prima cosa, eliminiamo le fonti di luce dalla scena ed aggiungiamo un paio di oggetti, ed in particolare un Plane inclinato, una sfera posta dal lato del Plane con la normale positiva uscente e un'altra sfera posta esattamente dall'altra parte (capirete presto il perché di tale configurazione).



La scena utilizzata in questo tutorial

Di default, *Ambient Occlusion* e *Indirect Lighting* sono disattivati, quindi lanciando un rendering ora non vedremo nulla. Selezioniamo il Plane ed associamogli un Material di base, con un colore qualsiasi, portando però *Emit* a 1.0.

Adesso, avviando un rendering, vedremo il Plane colorato, e questo perché *Emit* è maggiore di 0, ma come ben sappiamo questa caratteristica da sola non è sufficiente a rendere un oggetto un emettitore "reale", infatti non vediamo effetti di illuminazione sulle sfere.

Il rendering con il Plane EMIT, ma senza Indirect Lighting

Passiamo quindi alla scheda *World* nella *Properties Window*, selezioniamo la casella *Indirect Lighting* e scegliamo, come *Gathering Method*, *Approximate*, visto che *Raytrace* non è disponibile.

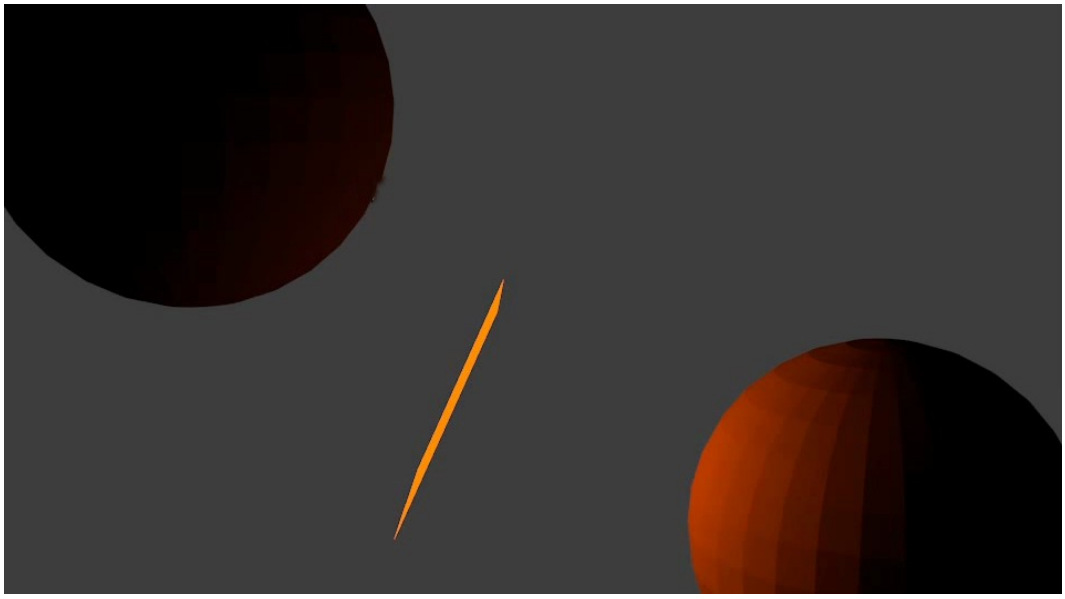
Nella sezione *Indirect Lighting* possiamo impostare l'intensità dell'effetto e il numero di rimbalzi da calcolare per ogni raggio luminoso; neanche a dirlo, all'aumentare del numero di rimbalzi avremo scene più credibili ma anche tempi di rendering sensibilmente maggiori, e il motore *Ambient Occlusion* è già abbastanza avido di risorse.

Nella sezione *Approximate* possiamo impostare la qualità del rendering agendo sui *Passes*, le iterazioni dell'algoritmo (portandone il valore almeno a 1), e il valore di *Error*, la soglia di errore.

Avviando un rendering adesso, con l'impostazione di default di un solo rimbalzo, vedremo gli effetti dell'illuminazione sulla sfera posta dal lato del Plane con la normale positiva uscente.

Il rendering con un rimbalzo di Indirect Lighting

Impostando 2 rimbalzi, vedremo gli effetti di illuminazione anche sulla seconda sfera, e questo perché i raggi luminosi partiti dal Plane colpiscono la prima sfera (bounce 1), rimbalzano e colpiscono la seconda (bounce 2) e a questo punto si fermano.



Il rendering con due rimbalzi di Indirect Lighting

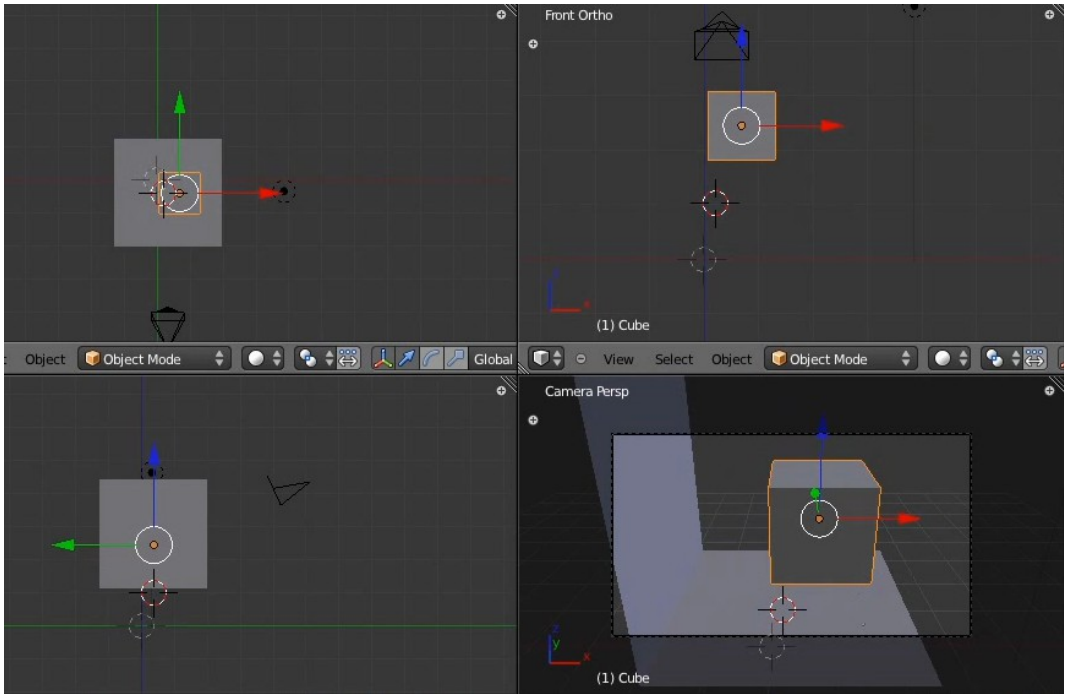
L'illuminazione è ovviamene soggetta ad attenuazione sul rimbalzo e sulla distanza.

Il valore di *Emit* del Material determina l'intensità della fonte luminosa, per cui portandolo ad esempio a 5 illumineremo maggiormente gli altri oggetti presenti nella scena.

Utilizzando *Ambient Occlusion* o *Indirect Lighting* noterete, ad ogni rendering, una prima fase con schermo nero; tale fase è detta "*Occlusion Preprocessing*" e a volte richiede molto più tempo del rendering vero e proprio, che parte subito dopo.

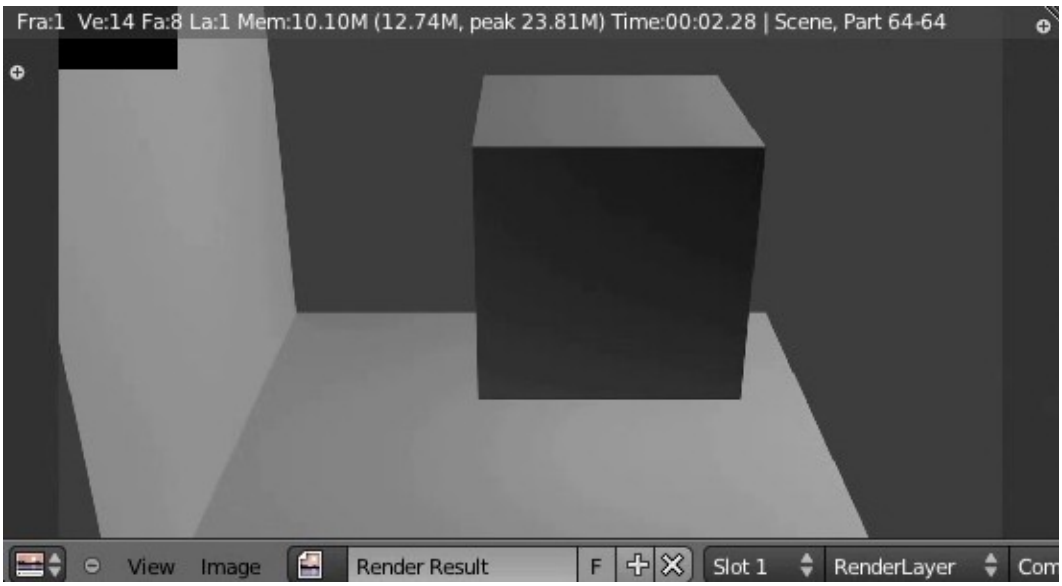
L'utilizzo di mesh emettitrici è ottimo per realizzare rendering di interni, lampade e insegne al neon ed altri effetti. Cancelliamo tutto per passare al secondo esempio. Deselezioniamo *Indirect Lighting*, in modo da avere una scena completamente buia, priva di fonti di illuminazione.

In questo caso, componiamo la scena come segue: aggiungiamo un Plane ed estrudiamo verso l'alto uno degli spigoli, come una parete; aggiungiamo quindi una fonte di luce posta all'interno del volume, di fronte alla parete, e un altro oggetto, ad esempio un cubo, al centro del volume, tra la parete e la fonte di luce, non poggiato sul Plane di partenza ma leggermente più in alto.



La scena utilizzata per il secondo esempio

Avviando un rendering adesso, vedremo illuminate solo le facce colpite direttamente dai raggi luminosi. Attiviamo quindi *Indirect Lighting Approximate* con 1 o 2 rimbalzi e almeno 1 passo di iterazione ed avviamo un altro rendering: adesso, anche le facce non direttamente colpite dai raggi luminosi sono illuminate.



Il rendering della scena con Indirect Lighting attivato

Indirect Lighting è di fondamentale importanza nella realizzazione di rendering di interni ma, a seconda dei casi, può tornare utile anche per rendering di esterni; dipende dalla qualità che volete ottenere: non serve per un concept, è indispensabile per il fotorealismo.

Un problema di *Indirect Lighting* (e in generale del motore *Ambient Occlusion* di Blender) è il tempo necessario per la fase di *Occlusion Preprocessing*. Vediamo un esempio pratico.

L'immagine visibile qui di seguito è stata realizzata ponendo una fonte di luce *Area Lamp* a destra, per simulare la luce in arrivo dall'esterno, senza *Indirect Lighting*.

Rendering ottenuto con Blender Internal senza Indirect Lighting

L'assenza di un motore di illuminazione globale è evidente: basta osservare le zone non direttamente illuminate, qui completamente nere, e il rumore nelle zone di confine.

Il rendering di questa immagine ha richiesto circa 25 minuti su un sistema 64 bit con processore i7 870 e 8GB di memoria RAM, con risoluzione 1280x720.

L'immagine seguente è stata ottenuta, invece, con *Yafaray*, utilizzando *Photon Mapping* con 100 milioni di fotoni sulla stessa architettura e alla stessa risoluzione; anche qui ci sono delle imperfezioni e una notevole quantità di rumore, ma è già qualcosa, considerando che il rendering ha richiesto appena 7 minuti.

Rendering ottenuto con Yafaray - Photon Mapping

La terza immagine, ottenuta con *Blender Internal* con 1 solo rimbalzo e 6 passi di *Gathering*, è decisamente interessante: questo rendering non presenta rumore, ombreggia in maniera più convincente superfici come, ad esempio, il forno, ed illumina correttamente tutte le aree; il problema è che, alla stessa risoluzione e sulla stessa architettura degli altri, ha richiesto ben 40 minuti, un tempo da 5 a 10 volte superiore rispetto a quello richiesto da altri motori esterni come *Yafaray* o *Octane*; ad ogni modo, si tratta di una funzionalità nuova e in evoluzione, confidiamo negli sviluppatori!

Un altro rendering realizzato con Indirect Lighting

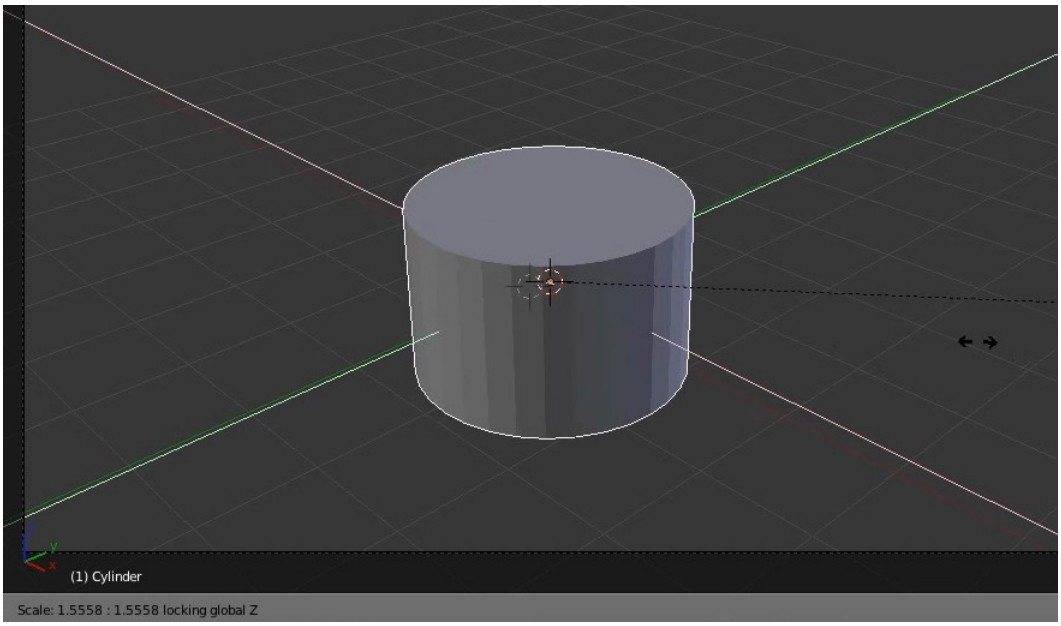
Come suggerito in un tutorial precedente, inoltre, in genere, soprattutto per rendering statici, non è necessario cercare di simulare l'illuminazione "reale" ponendo le fonti di luce così come sono nella realtà, per cui è possibile ottenere risultati migliori con una o due luci di riempimento, attive magari solo su certi gruppi di oggetti della scena, o con valori molto bassi di *Emit* per i materiali, allo scopo di ridurre il rumore ed evitare di far ricorso a *Indirect Lighting*, ove possibile.

* * *

Tips and Tricks: trasformazioni con assi vincolati (locking)

Questo più che un tutorial è un piccolo "Tips and Tricks" sulla modellazione e l'editing in Blender 3D 2.5; in particolare, vedremo lo shortcut per vincolare un asse mentre si effettua una trasformazione di traslazione, rotazione libera o scaling.

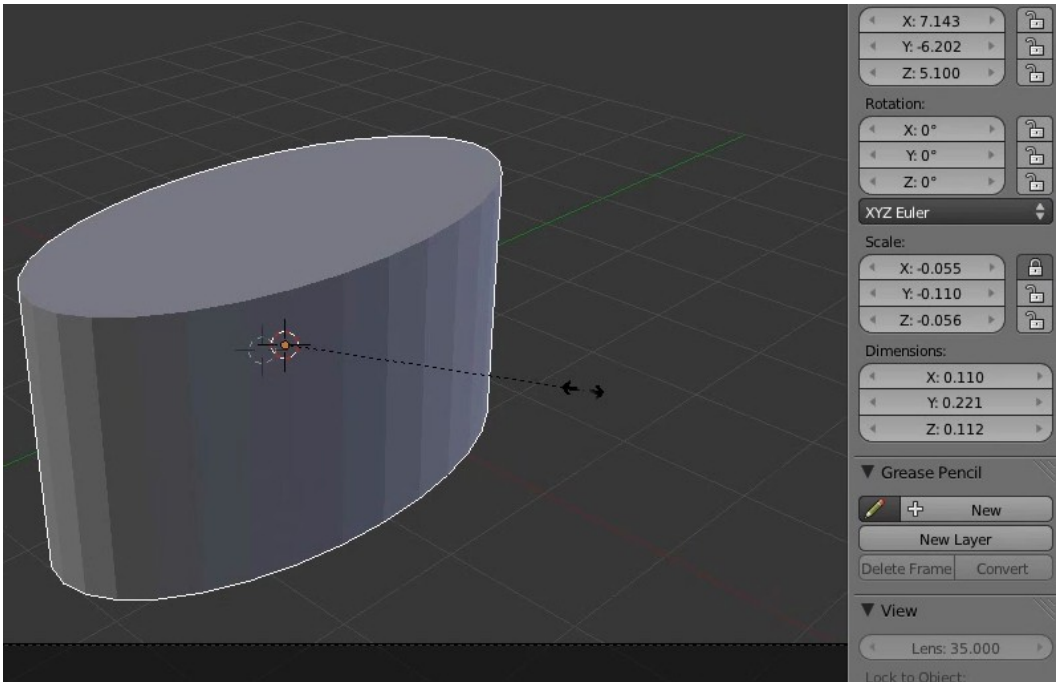
Capita infatti di dover, ad esempio, scalare una selezione in Object Mode (coinvolgendo quindi l'intero oggetto) o in Edit Mode (coinvolgendo gruppi di vertici, spigoli e facce) ma di dover mantenere inalterata, con un vincolo, una dimensione, ad esempio l'asse Z; in questo caso, dopo aver premuto il comando per la trasformazione (ad esempio *S*, per lo scaling) bisogna premere *SHIFT* e la lettera dell'asse da vincolare, ad esempio *SHIFT Z* per mantenere inalterata l'altezza della selezione.



Vincolare, bloccandolo (locking), l'asse Z, con SHIFT Z

Come avviene con le trasformazioni senza vincoli, digitando una sola volta la lettera relativa all'asse da bloccare definiremo un vincolo su quell'asse globale, mentre digitandola due volte, ad esempio *SHIFT Z Z*, definiremo il vincolo sull'asse locale, come ci informa d'altronde l'header della *3D View*, che riporta "*locking global*" o "*locking local*", a seconda dei casi, seguito dal nome dell'asse.

Se il vincolo dev'essere applicato solo una volta, lo shortcut va più che bene, altrimenti (per più operazioni, per trasformazioni parametriche o semplicemente per comodità) si può ricorrere al lucchetto nella *Transform Window*, richiamabile con *N* in una *3D View*; clicchiamo ad esempio sul lucchetto posto accanto a *Scale X*, in modo da creare un vincolo fisso per lo scaling su quest'asse, poi nella *3D View* digitiamo *S 2*: in questo modo raddoppieremo le dimensioni dell'oggetto in maniera parametrica ma solo sugli assi liberi, in questo caso Y e Z.



Utilizzo del "lucchetto" nella Transform Window

L'utilizzo della *Transform Window* ci consente inoltre di vincolare anche due assi per volta, limitando le trasformazioni ad un solo asse: è sufficiente selezionare due lucchetti per volta.

* * *

Lo script Import Images as Planes

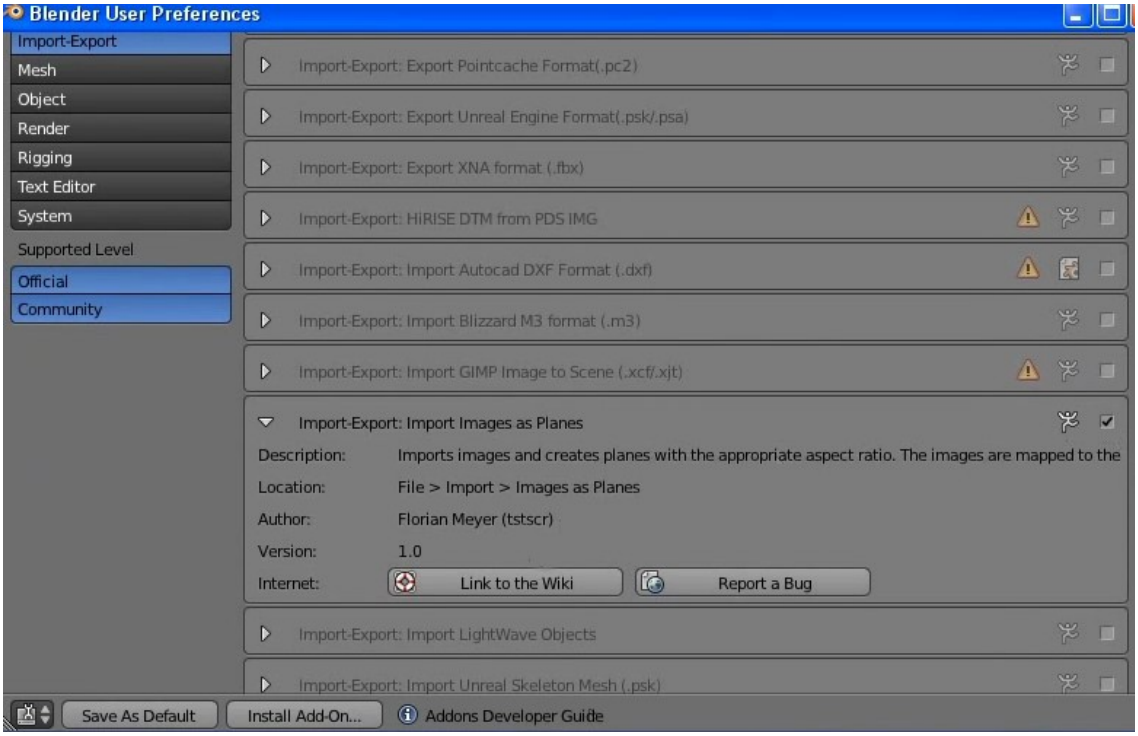
Questa è la versione italiana di un tutorial pubblicato da Andrew Price sul suo sito web, *blenderguru.com*; Andrew mi ha autorizzato a tradurre i suoi tutorials, a patto ovviamente di menzionarlo. In questo breve tutorial, realizzato con la versione 2.57a di Blender 3D, vedremo come utilizzare lo script "*Import Images as Planes*" che consente di caricare delle immagini e applicarle direttamente a dei Planes ridimensionati automaticamente in base alle dimensioni e all'*Aspect Ratio* delle Textures immagine.

Capita infatti di dover applicare un'immagine in maniera diretta ad un Plane, ad esempio per realizzare un poster, un quadro, una parete, un'immagine su un monitor, ecc...; un'operazione che volendo è semplicissima ma che a volte richiede un po' di tempo perché è necessario scalare il Plane per visualizzare bene l'immagine.

La "procedura standard", per così dire, è la seguente: si aggiunge un Plane, si fornisce un Material *Shadeless* con trasparenza *Z Transp* e *Alpha* 0.0 al Plane (*Alpha* 0.0 perché verrà modulata dalla Texture), quindi si collega una Texture immagine con *Premultiply* attivo e *Influence Alpha* 1.0 al Material e si ridimensiona il Plane in modo da visualizzare la Texture con il giusto *Aspect Ratio*.

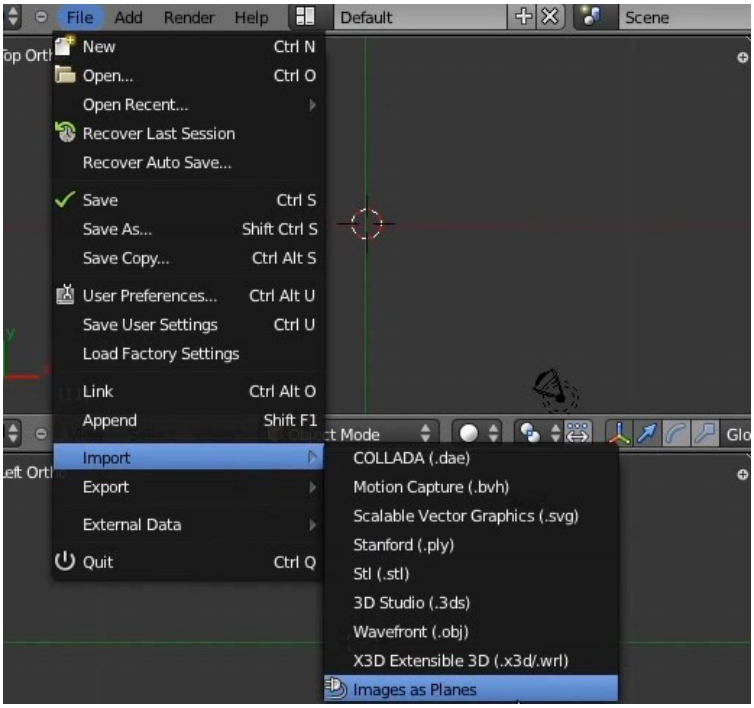
Tutte queste operazioni vengono effettuate dallo script "*Import Images as Planes*", che si trova nella sezione *Import/Export* degli *Add-Ons*, per cui apriamo la finestra *User Preferences* dal menù *File*, scegliamo la scheda *Add-ons*, sezione *Import/Export*, individuiamo "*Import Images as Planes*" e selezioniamo la checkbox a destra per attivarlo.

Per ritrovare lo script attivo nelle sessioni successive di Blender, clicchiamo su *Save As Default*, altrimenti chiudiamo semplicemente la finestra *User Preferences*.



Attivazione dello script Import Images as Planes nella finestra User Preferences - Add-Ons

A questo punto, lo script funziona scegliendo *File* - *Import* - *Images as Planes* e scegliendo il file immagine dalla finestra File Browser che apparirà a video.



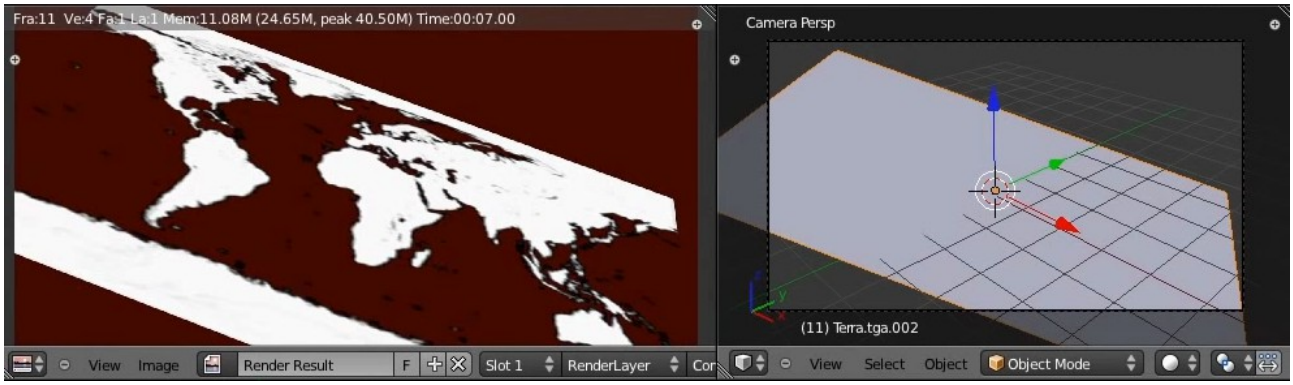
Richiamare lo script da File - Import

In questa finestra, sulla sinistra, troviamo alcune funzionalità extra, ed in particolare *Shadeless*, *Use Alpha* e *Premultiply*: selezioniamole tutte per ottenere l'effetto che volevamo, ossia un'immagine dotata di canale di trasparenza mappata su un Plane e che utilizzi le informazioni native per la trasparenza dell'oggetto; infine, in basso a sinistra, clicchiamo su "*Use Images Dimensions*" per ridimensionare automaticamente il Plane in base alle dimensioni dell'immagine.

Le impostazioni di importazione dell'immagine

Il Plane, con la relativa Texture immagine mappata correttamente, verrà inserito in maniera automatica nella

scena 3D.



Il risultato finale, con il Plane ridimensionato e con la Texture immagine correttamente mappata

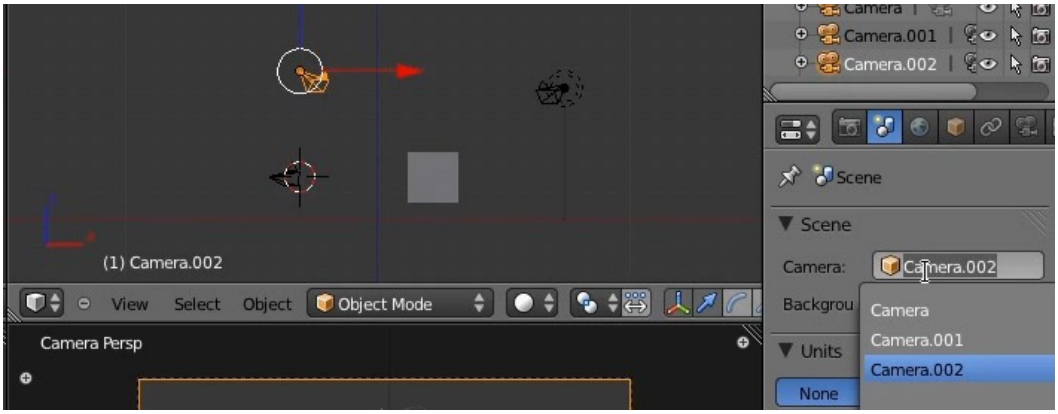
* * *

Impostare la Active Camera per rendering con più telecamere

Questo più che un tutorial è un piccolo "Tips & Tricks" di risposta ad una domanda di un utente che chiedeva come poter cambiare la telecamera da utilizzare in fase di rendering quando ci sono più telecamere in una scena.

Capita a volte di dover renderizzare, in Blender 3D, la stessa scena da vari punti di vista, a volte sempre gli stessi (tipo i punti cardinali), per cui le telecamere hanno posizioni e orientamenti standard (magari impostati con *User Preferences*) e a cambiare è la scena o parti di essa, perché ad esempio dopo il rendering dalla telecamera 2 decidiamo di cambiare qualche impostazione.

La telecamera "attiva", da utilizzare per il rendering, è quella specificata nel campo *Camera* nella sezione *Scene* della *Properties Window*, per cui dopo aver effettuato un rendering con la prima telecamera è sufficiente aprire tale scheda, cliccare sulla casella di testo accanto all'etichetta *Camera*, scegliere la telecamera da utilizzare per il nuovo rendering dal menù che apparirà a video dopo aver fatto click col mouse ed avviare il rendering, sia esso un'immagine statica del frame corrente (con *F12*) che un'animazione (con *CTRL F12*) nel range specificato nella scheda *Render*, nella *Properties Window.*



Il campo Camera nella scheda Scene della Properties Window

Quanto detto funziona sia per telecamere statiche che per telecamere in movimento, provviste ad esempio di animazione lungo un percorso per realizzare una camminata o visita virtuale all'interno di un ambiente; alle animazioni *Follow Path* (segui percorso) con *Constraint Track To*, per specificare cosa far puntare alla telecamera, ho dedicato la sessantanovesima e la settantesima puntata del videocorso sulle basi di Blender 3D.

* * *

Quando Blender è più furbo di noi: il caso Lens Flares con Halo

Questo è un piccolo tributo agli sviluppatori di Blender, che hanno pensato proprio a tutto, anche ad un effetto lenticolare, riprodotto con i Flares nei materiali di tipo Halo, che all'inizio può sembrare sbagliato ma che, come vedremo, è corretto.

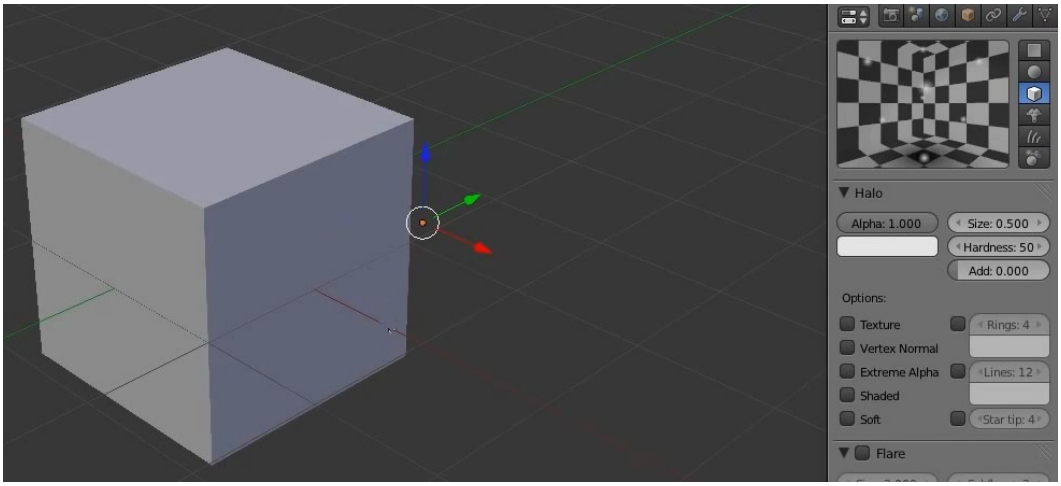
Come visto nella 45esima puntata del corso di base su Blender 3D e come discusso nel mio ebook dedicato ai *Materials in Blender 2.5*, i Materials di tipo *Halo* consentono di realizzare effetti noti come i bagliori lenticolari (*Lens Flare* in altri pacchetti software) di varie forme, dimensioni e colori, intorno ai vertici *Halo*.

Nel mondo reale, questi effetti lenticolari sono prodotti dalla lente o dalle lenti del sistema ottico di una macchina fotografica o telecamera, sia essa classica (cioè con pellicola) che digitale; il bagliore in sé è dovuto alla luce che sovraespone la porzione di pellicola o di sensore, ma la formazione del bagliore, così come la sua forma e la sua dimensione, sono dovuti alle lenti; si tratta di un effetto quasi ineliminabile, anche con le lenti migliori, che copre gli altri oggetti della scena, anche se la fonte di luce che l'ha prodotto è parzialmente coperta da altri oggetti.

Quest'ultimo punto può sembrare strano, ma se ci si pensa un attimo è ovvio: il bagliore si forma nelle lenti, quindi non arriva da dietro l'oggetto che copre la fonte di luce, ma in un certo senso tra questo e la pellicola o sensore.

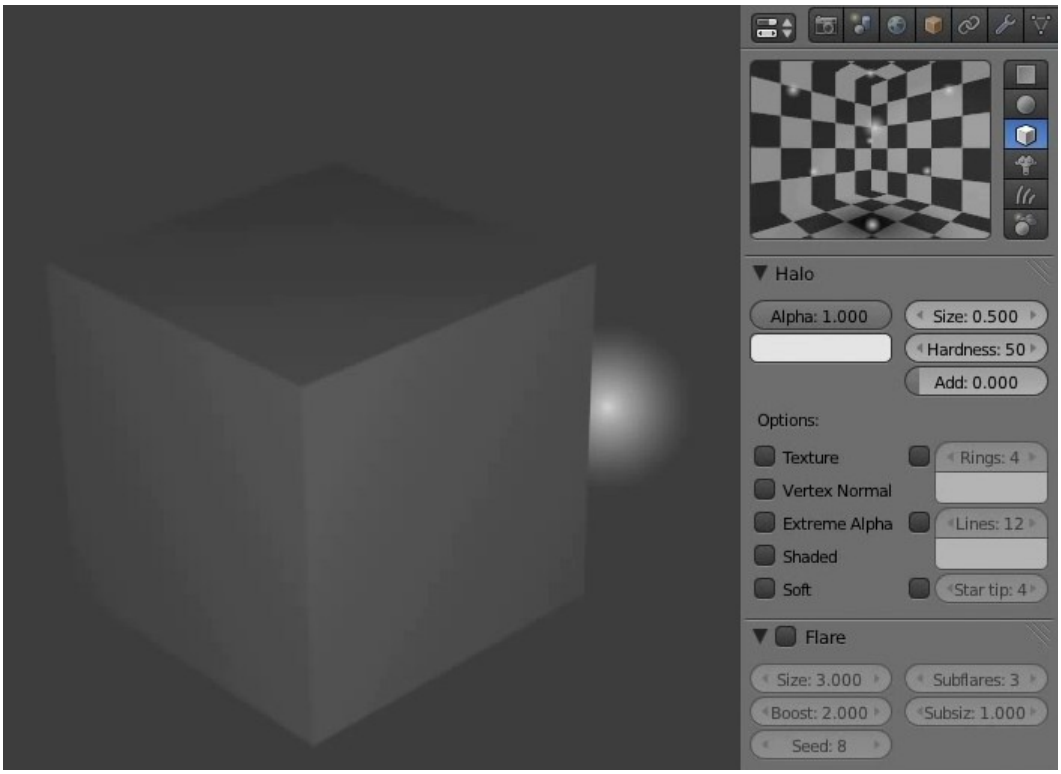
... e Blender, come gestisce quest'effetto?

La scena utilizzata per questo esempio contiene un oggetto (il classico cubo) e un vertice posto molto vicino al cubo; questo vertice è dotato di un Material *Halo*, con effetto *Flare* disattivato.



La scena utilizzata per questo tutorial

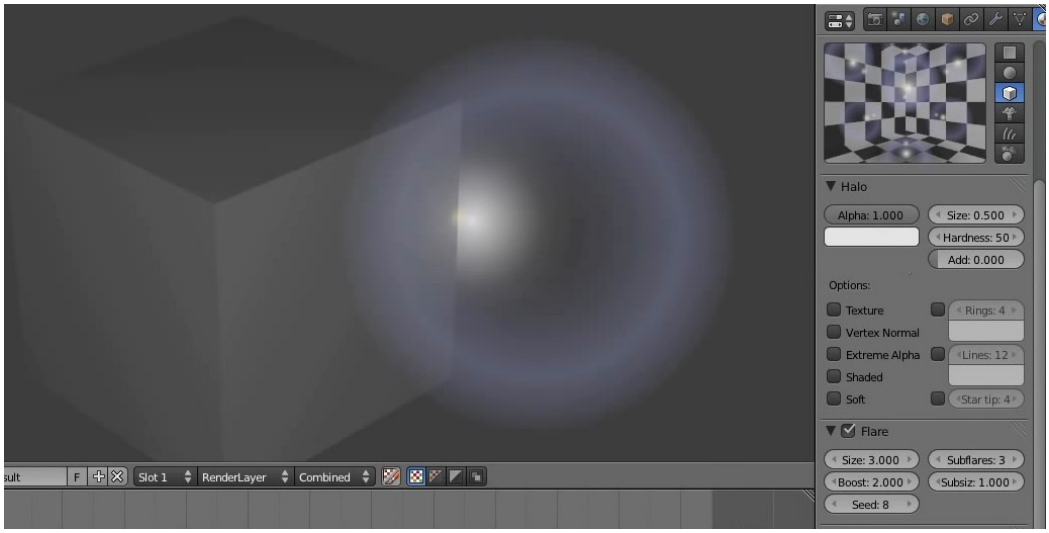
Avviando un rendering, notiamo che parte dell'effetto *Halo* è coperta dal cubo, quindi non è visibile, e questo è corretto: i raggi luminosi dell'*Halo* vengono bloccati dal solido e non sono visibili.



Il primo rendering

Attiviamo adesso l'effetto *Flare*, che è un effetto di compositing se vogliamo; abbiamo *Size* 3, *Boost* 2, *Seed* 8 e *Subflares* 3; sto utilizzando queste impostazioni solo per accentuare l'effetto dei bagliori lenticolari, ma ovviamente noterete la caratteristica qui discussa anche con altri parametri.

Avviando adesso un rendering, che verrà effettuato in due passaggi (il secondo, che riguarda proprio il *Flare*, è in effetti un passo di compositing), noteremo la presenza dell'effetto *Flare* che copre il cubo: questo effetto, anche se può sembrare strano, è fisicamente corretto, per i motivi esposti precedentemente; ancora una volta, gli sviluppatori di Blender hanno pensato proprio a tutto!



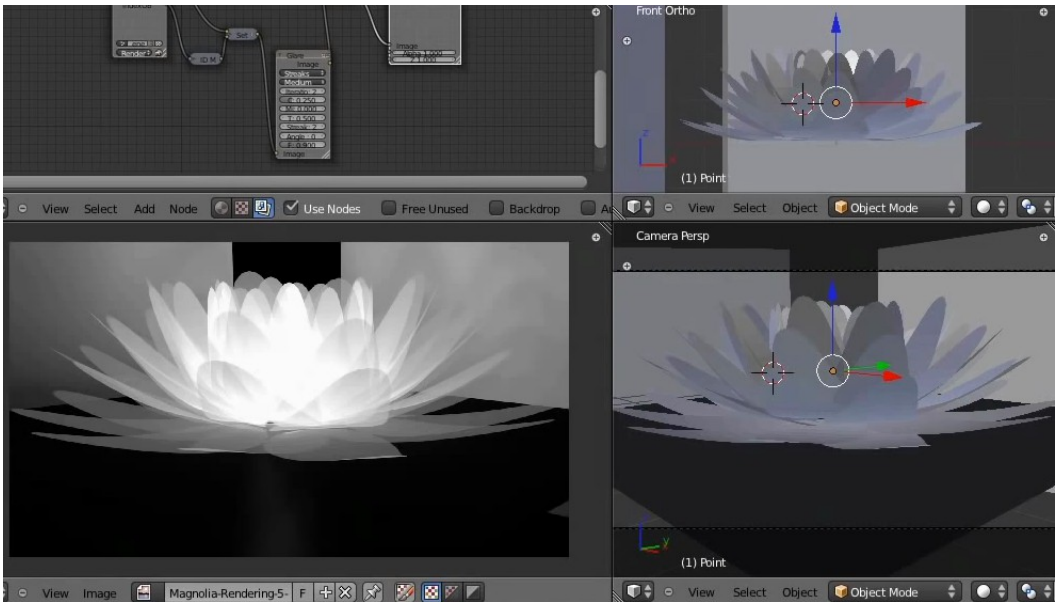
Il nuovo rendering, con l'effetto FLARE

L'opzione *Flare* può essere attivata o disattivata anche in maniera animata, ossia si tratta di un parametro provvisto di *curva IPO*, come discusso nella 66esima puntata del corso di base su Blender; questa caratteristica ci consente di implementare un *Lens Flare* in certi istanti di un'animazione ponendo, come in questo caso, un vertice con *Halo* e *Flare* nella scena, ad esempio se la telecamera sta cambiando orientamento e nel mentre inquadra una qualsiasi fonte di luce, come il Sole, una candela, una lampadina: in questi casi, possiamo far passare il vertice (e con esso il bagliore lenticolare) da una parte all'altra della scena, ottenendo un effetto simile a quello che si ha nella realtà.

* * *

Magnolia: un rendering con Indirect Light e Glare

Questo più che un tutorial è l'analisi di un rendering realizzato mediante *Indirect Light* e l'effetto *Glare* in Blender 3D 2.5. Si tratta di una lampada un po' particolare, dal design "a petali" con luce diffusa; ho chiamato questa scena "*Magnolia*" per via della forma della mesh, che potete vedere nell'immagine seguente.



La scena utilizzata in questo tutorial

In realtà, esamineremo diversi rendering, da quello di base fino a quello definitivo, per mostrare le differenze introdotte dalle varie opzioni attivate di volta in volta.

Iniziamo dalla mesh, anzi proprio dalla scena: per ottenere il primo rendering, non utilizziamo fonti di luce standard come le *Lamps*, bensì la caratteristica *Emit* (cioè l'emissione luminosa) del Material associato alla mesh *Magnolia*.

Diamo un'occhiata a questo Material: per l'ombreggiatore Diffuso *Lambert* è presente la *Ramp* di base (da nero trasparente a bianco opaco) su *Input Normal*, per variare l'illuminazione in base all'orientamento delle facce, mentre l'ombreggiatore *Speculare CookTorr* ha intensità 0.5 e *Hardness* 50; in *Shading*, *Emit* ha valore 0.05, mentre la *Translucentezza* vale 0.5; la trasparenza è attiva in modalità *Raytrace*, con *Alpha* 0.5, tutte le altre impostazioni come di default (se lo si desidera, è possibile cambiare l'indice di rifrazione *IOR* per provare qualche effetto particolare).
Non sono presenti Textures.

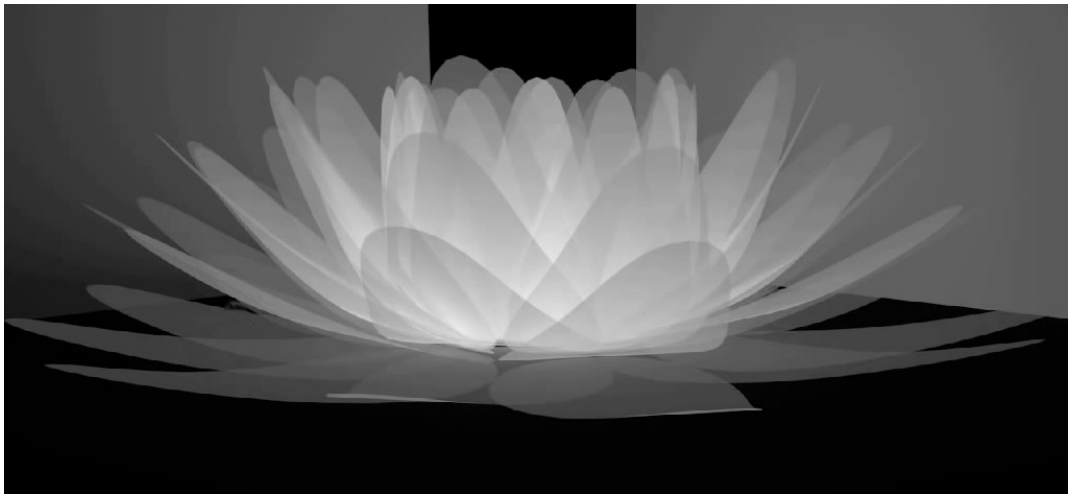
Avviando un rendering, comunque, avremo un risultato davvero deludente: il Material è visibile, visto che *Emit* ha un valore maggiore di zero, ma in realtà la mesh non sta emettendo luce.

Il primo rendering

Per rendere la mesh davvero "emettitrice" dobbiamo attivare l'opzione *Indirect Lighting*, per la luce indiretta (alla quale ho già dedicato qualche tutorial in passato), caratteristica nuova della release 2.5 di Blender.

Le caratteristiche di *Indirect Light* nella sezione *World* della *Properties Windows* sono 3 rimbalzi, 4 passi di *Approximate* e tutte le altre voci come di default.

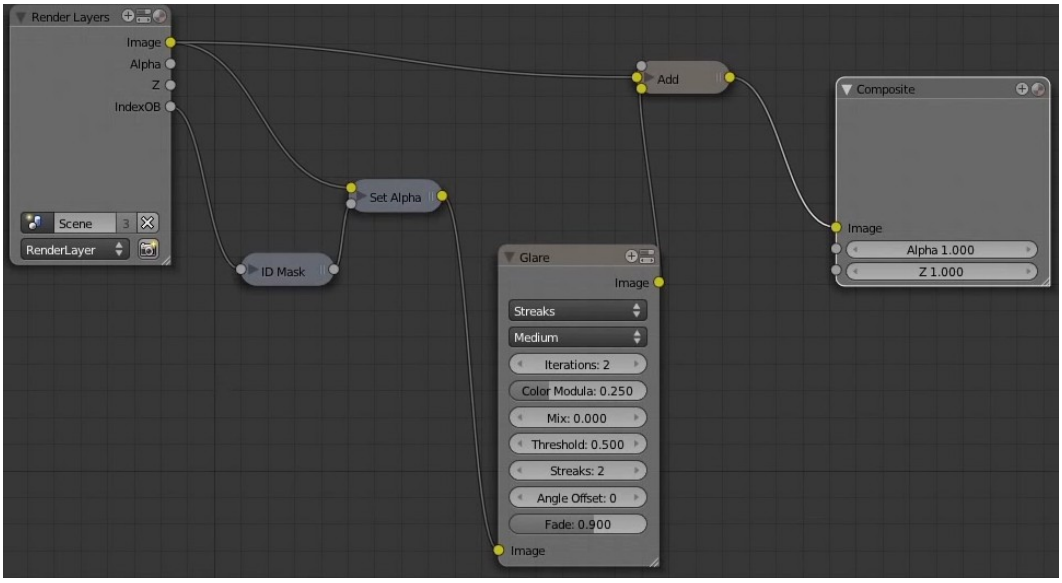
Ecco il rendering ottenuto con *Indirect Light* così impostata.



Il secondo rendering, questa volta con Indirect Lighting

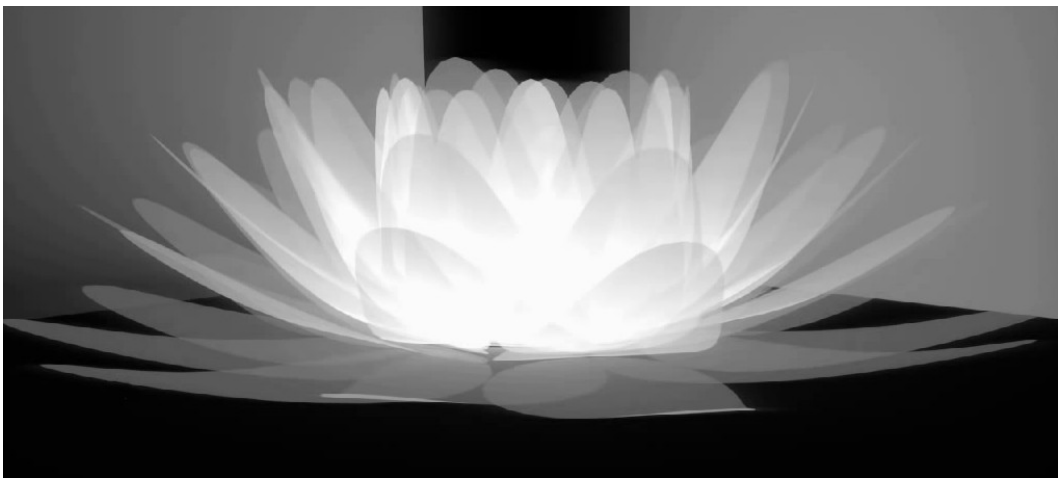
Ancora non ci siamo, comunque: la luminosità è meno uniforme, è vero, ma manca "quel non so che" in grado di rendere interessante la scena... ma noi sappiamo benissimo di che si tratta: del bagliore *Glare*, ottenuto mediante il *Nodes Compositing* di Blender.

All'argomento *Compositing dei Nodes in Blender 2.5* ho dedicato un ebook, per cui non tratteremo l'argomento in dettaglio, limitandoci ad esaminare la configurazione adottata in questo caso: 3 iterazioni di *Streaks Medium* con 2 *Streaks* e *soglia* 0.0, le altre impostazioni come di default, in maniera selettiva sulla mesh mediante Nodo *ID Mask*.



Lo schema di Nodi di Compositing

Ecco quindi il rendering ottenuto attivando anche questo effetto, oltre a *Emit* e *Indirect Light*.



Il nuovo rendering

C'è da dire che questi effetti non proiettano ombre interessanti sul resto della scena, ma per questo non c'è altro modo se non l'utilizzo di una Lamp posta al centro della mesh, che ha Material semitrasparente, come sappiamo; non basta, comunque: dobbiamo attivare l'opzione *Receive Transparent* per il Material della mesh e per tutti i Materials che dovranno ricevere le ombre proiettate. Per sfumare le ombre ed evitare bordi netti, poco credibili, dobbiamo invece modificare i *Samples* e la dimensione del bordo sfumato della Lamp; in particolare, utilizziamo *Size* 0.5 con 8 *Samples*. Ecco il rendering ottenuto impostando anche queste ultime caratteristiche, oltre a quelle elencate precedentemente.

Il rendering con la fonte di luce e le ombre proiettate

I tempi per ottenere i vari rendering variano sensibilmente ad ogni nuovo effetto implementato, infatti *Indirect Light* e le ombre sfumate sono effetti che anche singolarmente pesano parecchio in termini di tempo e risorse di rendering, a maggior ragione tutti insieme; in particolare, sulla stessa architettura hardware e software, sempre a risoluzione 1280x720:

- il primo rendering, con solo *Emit*, ha richiesto 3 secondi;
- il secondo rendering, con Emit e *Indirect Light*, ha richiesto 300 secondi;
- il terzo rendering, come il secondo più *Glare*, ha richiesto 303 secondi, solo 3 secondi in più per la post-produzione;
- il quarto rendering, come il terzo più 8 *Samples* e *Soft Size* 0.5 per la *Lamp*, ha richiesto 435 secondi.

In questo modo abbiamo esaminato varie opzioni, comunque nel quarto rendering la luminosità mi è sembrata eccessiva, per cui ne ho realizzato un quinto cambiando alcuni parametri:

- 2 rimbalzi e *Factor* 0.8 per *Indirect Lighting*;
- 2 iterazioni e *soglia* 0.5 per l'effetto *Glare*;
- intensità 0.75 per la *Lamp*.

Quest'ultimo rendering ha richiesto 420 secondi.

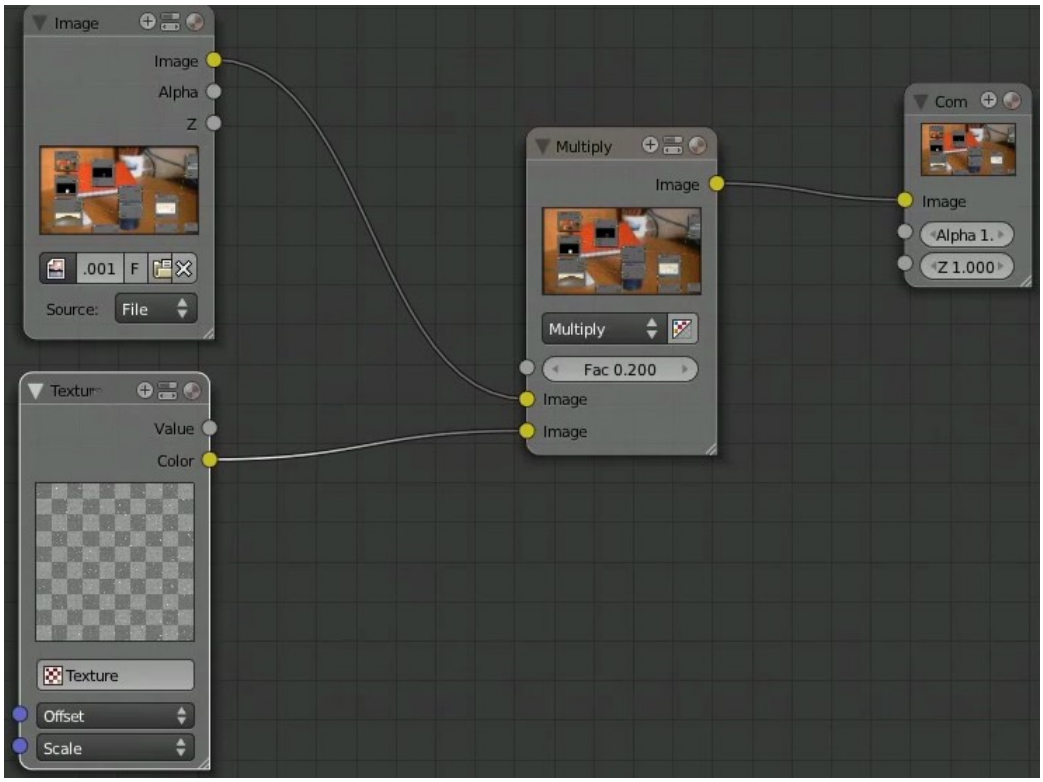
L'ultimo rendering, con l'illuminazione ridotta per la Lamp

* * *

Aggiungere rumore ad animazioni digitali e stills in Compositing

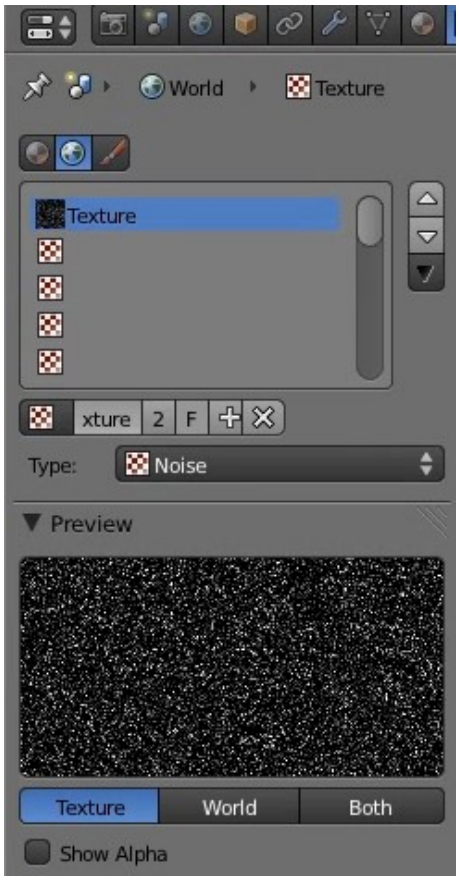
In questo breve tutorial vedremo un piccolo trucco di *Compositing*, ossia come aggiungere del rumore (della "granulosità") alle immagini; la cosa può sembrare strana, visto che in genere si cerca di togliere il rumore, che però è ineliminabile nelle riprese "reali" e quando si cerca di fondere immagini digitali su riprese reali, la differenza salta agli occhi. Può sembrare strano, ma è più facile notare l'assenza di questi "granelli di rumore", tipici delle riprese reali (ad esempio per via della compressione o del rumore elettronico dei sensori) che notarne la presenza, ecco perché se intendete aggiungere un oggetto realizzato al computer in una ripresa vi consiglio di fare quanto segue.

Utilizziamo il *Nodes Editor* di Blender 3D 2.5; non spiegherò in dettaglio il funzionamento dei Nodi, trattato ampiamente nel mio *ebook sul Compositing in Blender 2.5*: qui ci limiteremo a studiare la configurazione adottata.



Lo schema di Nodi di Compositing

La prima immagine di input è quella in arrivo da *Render Layers*, eventualmente filtrata mediante una maschera per isolare solo l'oggetto che ci interessa e sovrapporlo in seguito, mediante *Alpha Over*, alla ripresa reale.



Creazione di una Texture NOISE, non applicata a World

L'altro Nodo che ci interessa è di tipo *Input Texture*, infatti dobbiamo utilizzare una Texture di tipo *Noise* per produrre la granulosità. Una cosa poco nota, infatti, è che la Texture *Noise* è intrinsecamente animata, nel senso che i granelli di rumore che la compongono sono generati in maniera casuale e variano da frame a frame, per cui se applicherete tale Texture ad un oggetto vedrete cambiare il suo aspetto durante l'animazione.

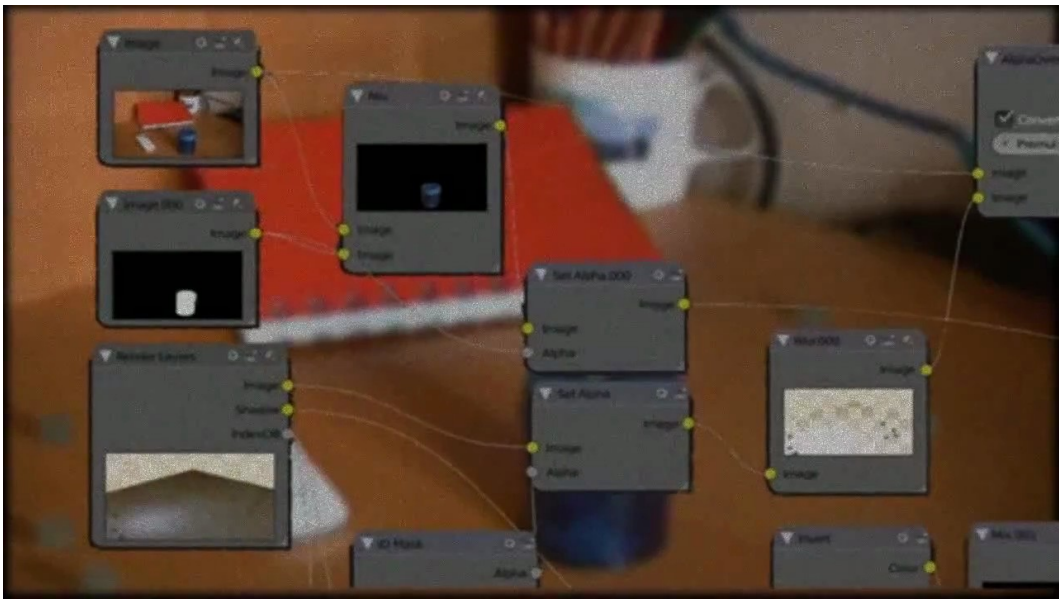
Selezioniamo quindi *World* e aggiungiamo una Texture *Noise* nel primo slot di *Texture* dell'elemento, avendo cura di deselezionare tutte le caselle della sezione *Influence*, infatti questa Texture non dev'essere applicata a *World*.

Passiamo ora nella finestra *Nodes Editor*, con il compositing dei Nodi attivato, ed aggiungiamo un Nodo *Input Texture* scegliendo, come Texture, proprio il rumore appena creato.

Inseriamo un Nodo di tipo *Color Mix* con operazione *Multiply*.
Nel mio caso, non sto utilizzando *Render Layers* ma una fotografia (cioè una singola immagine, priva del rumore animato tipico delle animazioni) ma il concetto è lo stesso.

Colleghiamo il canale di output *Image* della fotografia al primo canale di input del Nodo *Multiply*, mentre sul secondo colleghiamo, ovviamente, l'output *Image* in arrivo dal Nodo *Texture*.
Così facendo, moltiplicheremo i pixel "sale e pepe" della Texture per i pixel dell'immagine; dal momento che il nero vale 0.0 e il bianco 1.0, dobbiamo impostare correttamente il valore di *Factor* perché con *Factor* elevato o massimo potremmo porre a neri (e in generale scurire sensibilmente) i pixel dell'immagine originale, un effetto decisamente eccessivo.

In maniera del tutto empirica, sono arrivato alla conclusione che un valore nel range 0.1 – 0.2 va più che bene per ottenere un buon effetto, che c'è ma che non disturba l'osservatore (bisogna proprio andare ad esaminare con attenzione un dettaglio del fotogramma per notare il rumore).



Screenshot del risultato ottenuto (ma tale immagine è soggetta anche agli effetti della compressione dello screenshot!)

Come anticipato, non abbiamo impostato nessun tipo di parametro per animare la Texture: il rumore di *Noise* è animato per natura.

C'è da dire che alcuni riproduttori non sempre riescono a riprodurre in maniera fluida questo tipo di rumore, perché puntiforme e che varia da frame a frame, quindi se notate dei cambiamenti “a scatti” del rumore, il motivo dovrebbe essere questo.

* * *

Un buon motivo per salvare in OpenEXR i rendering realizzati con Blender 3D 2.5

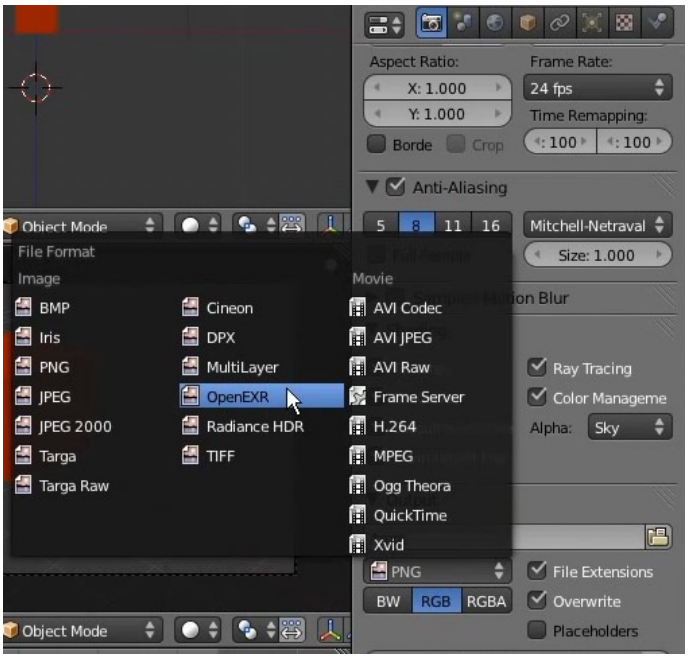
Questo è un tutorial teorico-pratico sull'utilizzo dei file *EXR* soprattutto per il Compositing dei rendering realizzati con Blender 3D 2.5.

Il formato EXR, sviluppato e reso pubblico dalla *Industrial Light and Magic*, consente di memorizzare in un unico file più canali immagine (ad esempio le mappe *RGB*, *alpha*, *ambient*, *mirror*, eccetera) e ad una profondità dati (*bit depth*) maggiore del solito, ad esempio a 16 o 32 bit per canale, consentendo tra l'altro di memorizzare le immagini *HDR*, *High Dynamic Range*.

Blender consente di salvare i rendering in formato *EXR* e di sfruttare la possibilità, offerta da questo formato, di memorizzare anche il *canale Z*, ossia le informazioni sulla profondità degli oggetti, insieme al rendering vero e proprio.

Questa informazione può tornarci molto utile per implementare effetti basati sulla distanza in fase di compositing, ad esempio la nebbia o la foschia per gli oggetti distanti o, come vedremo in questo tutorial, l'utilizzo del Nodo *Z Combine*.

Per prima cosa, cambiamo il formato di *Output* dei file renderizzati in immagine *OpenEXR* nella sezione *Output* della scheda *Render,* all'interno della *Properties Window.*

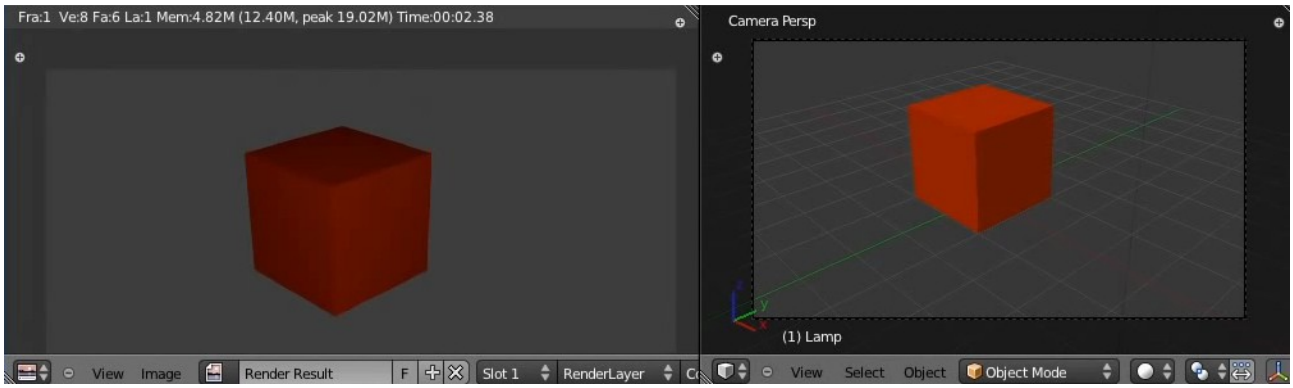


Output - EXR

Tra le opzioni offerte, notiamo *HALF*, che consente di salvare 16 bit per canale anziché i 32 bit standard del formato *EXR*, e *Z-Buf*, che consente di salvare le informazioni sulla distanza e che lavora a 32 bit; selezioniamo *Z-Buf*, lasciando inoltre l'opzione *ZIP lossless* (cioè compressione senza perdita di dati) nel menù *Codec*.

Di default, vengono memorizzati i canali R, G, B con 32 bit per canale, ma possiamo cambiare in scala di grigi col pulsante *BW* o aggiungere il canale dell'opacità (o trasparenza) *Alpha*.

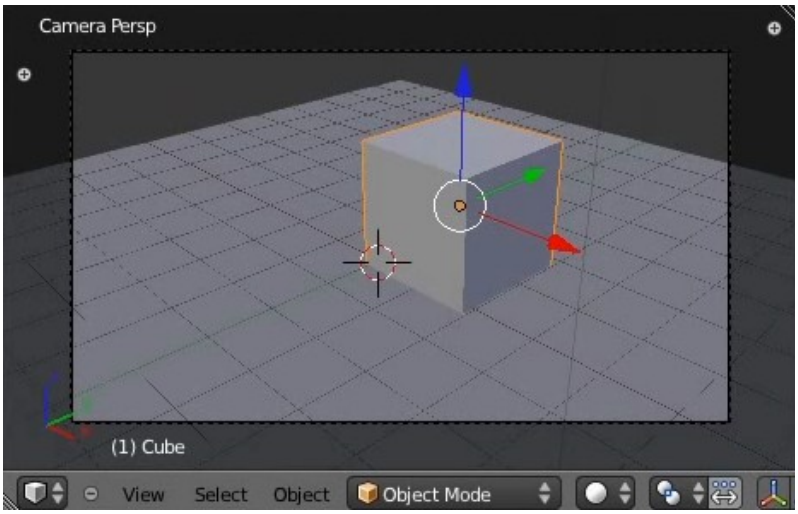
Cambiamo magari il colore di un oggetto nella scena standard di Blender, in modo da distinguerlo bene dal prossimo rendering, e lanciamo un rendering con *F12*, salvando il file *EXR* con *F3* e specificando il percorso nella finestra stile esplora risorse che apparirà a video.



Il primo rendering, memorizzato come File EXR

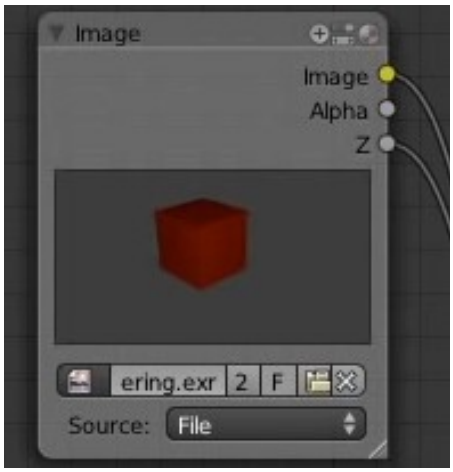
Abbiamo appena creato e salvato un'immagine in *EXR*. Tale immagine è provvista, in questo caso, dei canali colore e delle informazioni *Z-Buffer*.

Adesso apriamo una nuova scena di Blender, spostando un po' l'oggetto presente di default (non troppo, perché vi anticipo che vogliamo sovrapporlo al primo utilizzando solo il Compositing).



La nuova scena

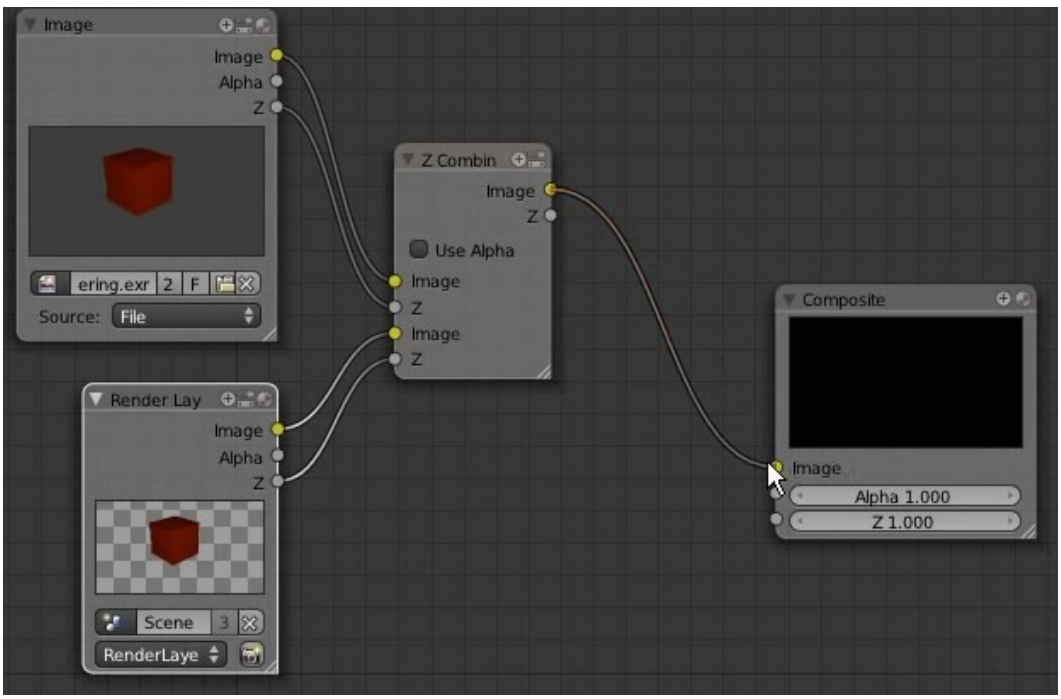
La corretta interpretazione del *Buffer Z* (che, passato da *Render Layer* a *Viewer*, restituisce un'immagine bianca, che può trarre in inganno) e l'utilizzo del Nodo *Z-Combine* sono spiegati nel mio *ebook sul Compositing dei Nodes in Blender 3D*; qui esaminiamo la configurazione dei Nodi.



Il nodo Input Image, con l'immagine renderizzata caricata, provvisto di canale Z-Buffer

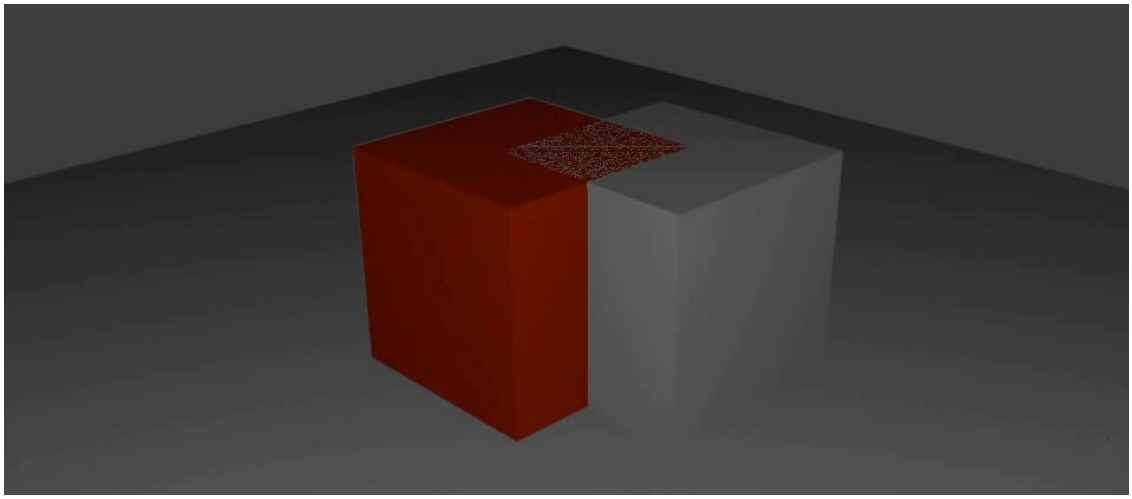
Inseriamo un Nodo di tipo *Input Image*, caricando in tale nodo il rendering realizzato poco fa. Noterete, tra le socket di uscita del Nodo, il canale *Z-Buffer*: l'immagine è dotata di questo tipo di informazione.

Inseriamo un Node di tipo *Color – Z-Combine*, passando i canali *Image* e *Z* del rendering salvato sulle prime due socket di *Z-Combine* e, sulle altre due, i canali *Image* e *Z* del Nodo *Render Layers*. Passiamo l'informazione *Image* (e, se lo vogliamo, il canale *Z*) in uscita da *Z-Combine* a *Composite*, dopodiché avviamo il rendering della scena corrente: nel primo passaggio, vedremo il rendering della scena, e nel secondo l'effetto del Compositing, con sovrapposizione basata sulla distanza di un oggetto che non è presente nella scena corrente.



Lo schema finale dei Nodi di Compositing

Faccio notare che il passo di Compositing di *Z-Combine* non dev'essere per forza l'ultimo della catena; ad esempio, possiamo sovrapporre due oggetti con *Z-Combine* e poi filtrare le parti di un oggetto che emergono ed applicare ad esse altri effetti come cambiamenti di colore, saturazione, contrasto, *Glare*, eccetera.



Il rendering ottenuto

Spero che questo breve tutorial possa servirvi da spunto per ottimizzare il lavoro, assemblando con più facilità le scene, coordinarvi meglio con altre persone o, in generale, provare nuovi effetti.

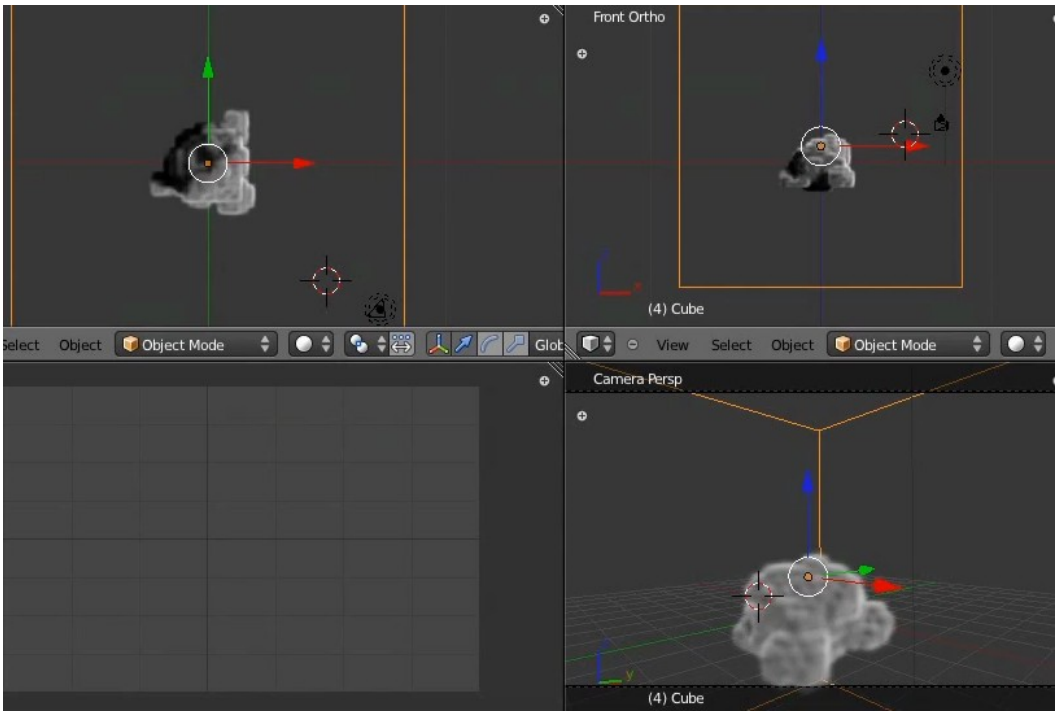
* * *

Oggetti in fiamme con Smoke e Particles Emitter, un esempio pratico

In questo tutorial vedremo un esempio di impostazioni da utilizzare con *Smoke*, *Particles Emitter*, Materiali e Textures *Voxel Data* per dar fuoco ad un oggetto in Blender 3D 2.57b (e successive).

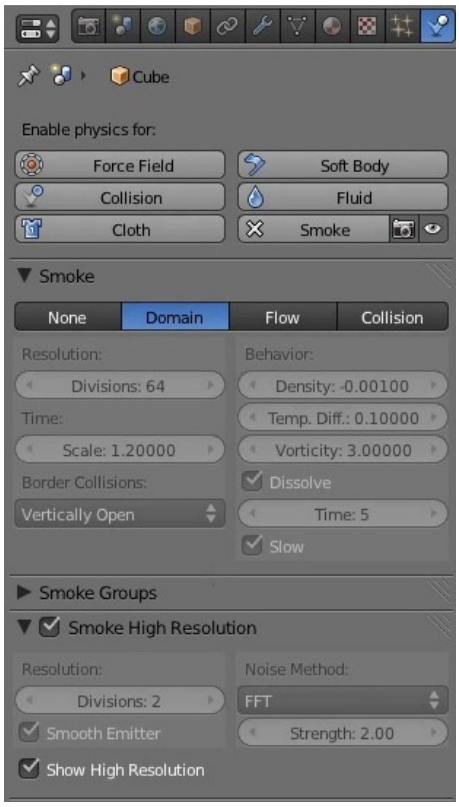
Al simulatore *Smoke* ho già dedicato un paio di tutorial più teorici; qui preferisco concentrarmi su un esempio pratico di simulazione, esaminando i parametri utilizzati piuttosto che il significato delle varie voci.

Scaliamo il cubo presente di default in modo da posizionare la telecamera al suo interno; inseriamo qui anche l'oggetto da incendiare (in questo esempio, *Suzanne*) e un paio di fonti di luce; utilizzeremo il cubo come *Domain* della simulazione.



La scena che utilizzeremo in questo tutorial

Con il cubo-dominio selezionato, apriamo la scheda *Physics* nella *Properties Window* ed aggiungiamo un simulatore *Smoke* di tipo *Domain*.



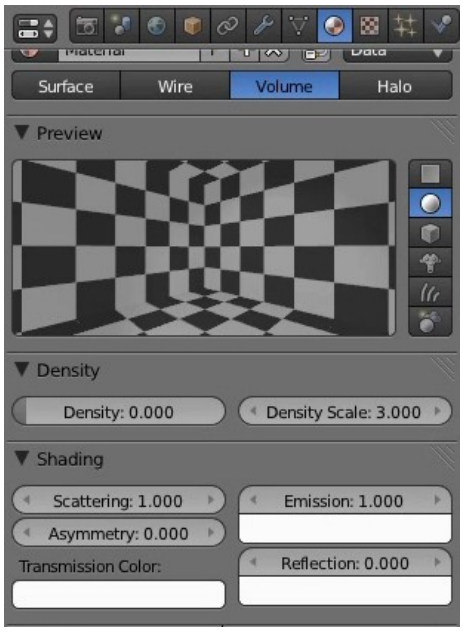
Le impostazioni Physics Smoke - Domain per il cubo-dominio

I valori che indicherò sono solo d'esempio, potete personalizzarli come volete:

- 64 o più *divisions* per aumentare la qualità, ma occhio alla RAM;
- *Slow* e *Dissolve* selezionati; *Time* 5;
- *Densità* 0.001, differenza di *temperatura* 0.1;
- *turbolenza* 3.0 per rendere le fiamme meno lineari;
- *Time Scale* 1.2 per accelerare un po' l'animazione;
- *Border Collisions Vertically Open*;
- nella sezione *Smoke High Resolution*, da attivare, impostiamo due divisioni, rumore *FFT* o *Wavelet* (è indifferente), con intensità 2, e selezioniamo *Smoke Emitter* e *Show High Res*.

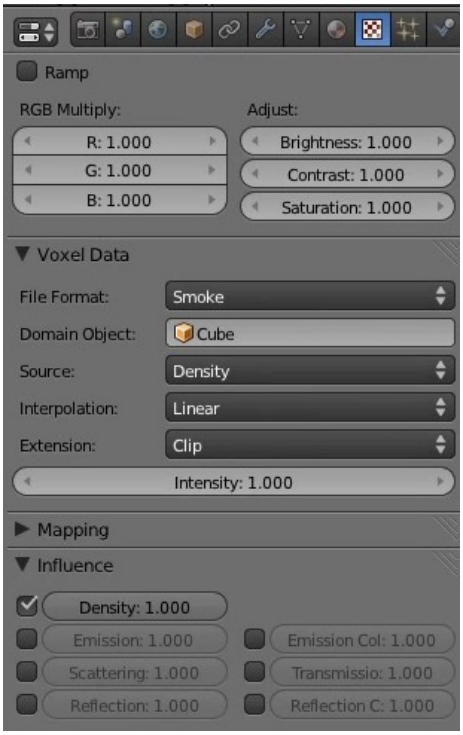
Il Material delle fiamme riguarda, anziché l'oggetto emettitore in sé, il dominio della simulazione.

Forniamo quindi un Material di tipo *Volume* al cubo, con *Density* e *Reflection* 0.0, *Density Scale* ad esempio 3, *Scattering* ed *Emission* 1, altri colori ed impostazioni come di default; in *Integration*, *Step Calculation Randomized* con *Step Size* 0.02 e *Depth Cutoff* 0.01; maggiori informazioni sui significati di questi parametri negli *ebook su Materials e Textures in Blender 3D 2.5*.



Il Material delle fiamme, da associare in realtà al cubo-dominio

Associamo a questo Material una prima Texture di tipo *Voxel Data*; dico "prima" perché le Textures *Voxel Data* sono in realtà due: una per il canale *Diffuse*, l'altra per il colore di emissione.



Le impostazioni della prima Texture Voxel Data

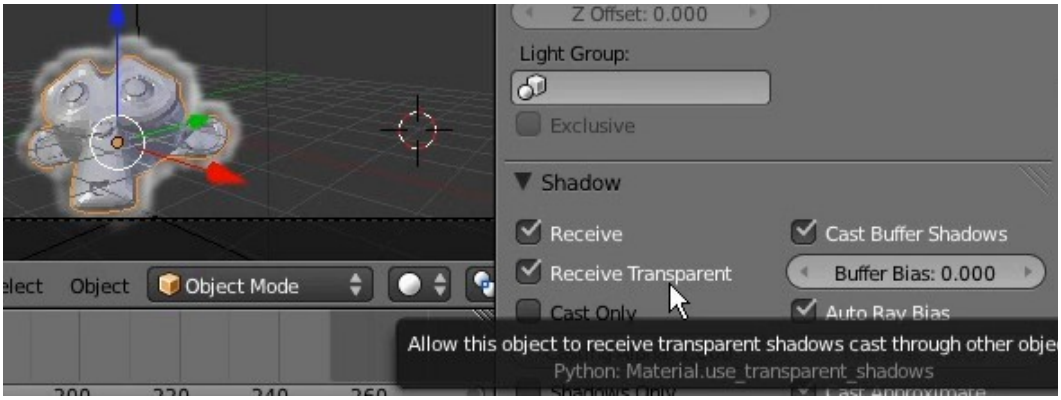
Nella prima *Voxel Data* impostiamo *File Format Smoke*, il cubo utilizzato come *Domain Object*, *Source Density*, interpolazione *Cubic B-Spline*; in *Influence*, lasciamo *Density* con valore 1.0 e modalità di blending *Mix*.

Nella seconda Texture *Voxel Data* attiviamo la *Ramp* fornendo una colorazione rossa e trasparente ai bordi della rampa e gialla e opaca al centro, con una buona sfumatura sia di colore che di livelli di trasparenza.

Nella sezione *Voxel Data*, impostiamo *File Format Smoke*, il cubo utilizzato come *Domain Object*, *Source Density*, interpolazione *Cubic B-Spline*; in *Influence*, disattiviamo *Density* e selezioniamo invece *Emission Color* con valore 1.0 e modalità di blending *Multiply*.

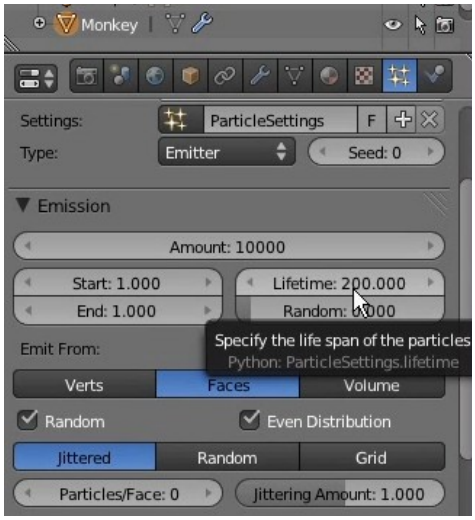
Le impostazioni della seconda Texture Voxel Data

Il Material di *Suzanne* (e, per estensione, dell'oggetto emettitore) non è importante: è distinto da quello delle fiamme, potete utilizzare qualsiasi Material e qualsiasi Texture ma dovete attivare l'opzione "*Receive Transparent*" nella sezione *Shadows* del Material, in modo da disegnare le ombre proiettate dalle fiamme come effettivamente trasparenti, non nere e completamente opache.



L'opzione "Receive Transparent" per le ombre del Material di Suzanne

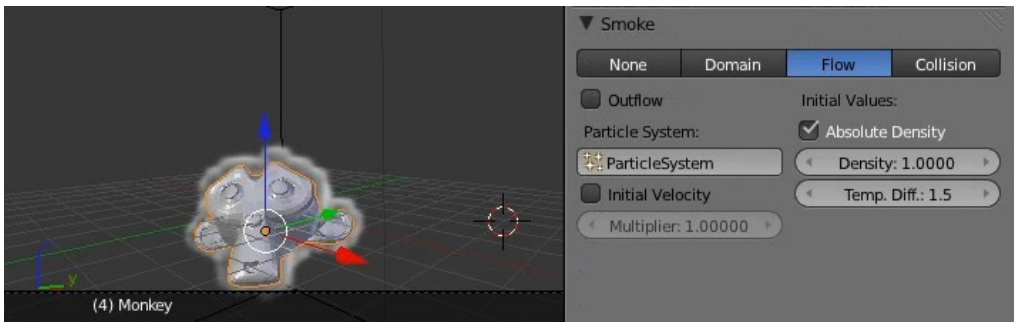
Aggiungiamo un sistema particellare *Emitter* per *Suzanne*; come alcuni di voi sapranno da qualche tutorial precedente, l'Emitter per *Smoke* è, in un certo senso, un "finto" sistema particellare: *Start* ed *End* devono valere infatti 1, mentre *Emitter Geometry Normal* deve valere 0 (anziché 1, valore di default) e *Particles Lifetime* deve avere la durata dell'animazione del fuoco, ad esempio 200 frames.



Le impostazioni del "finto" sistema Emitter per Suzanne

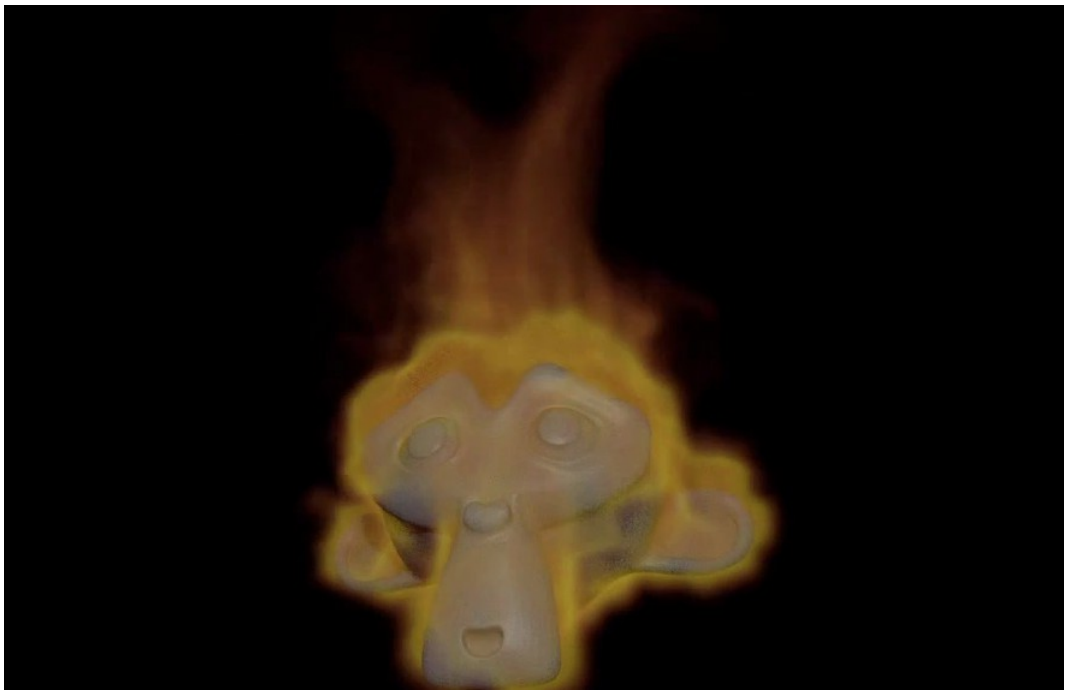
Nella sezione Render del sistema particellare, impostiamo a *None* la modalità (che di default è *Halo*). Impostiamo l'*Amount* a piacere, ad esempio 10.000 particelle, e disattiviamo la simulazione fisica, che di default è *Newtoniana*, lasciando le altre impostazioni come di default.

Sempre con *Suzanne* selezionata passiamo alla scheda *Physics* nella *Properties Window* e aggiungiamo un simulatore *Smoke* di tipo *Flow* con sistema particellare appena creato; se vogliamo far salire velocemente le fiamme, aumentiamo la differenza di temperatura portandola ad esempio a 1.5 o 2, lasciando il resto come di default.



Le impostazioni della simulazione Smoke Flow di Suzanne

Prima di effettuare il *bake* di una simulazione fisica, è necessario salvare il file della scena su disco.
A questo punto conviene effettuare un primo *Bake* della simulazione, premendo *ALT A* mentre il cursore è in una finestra *3D View* oppure, meglio, selezionando la scheda *Smoke Domain* del Cubo e cliccando su *Bake* in *Smoke Cache*; a seconda delle impostazioni della simulazione (ad esempio, il numero di suddivisioni del *Domain* o la durata dell'animazione) e dell'hardware, questa operazione può richiedere un bel po' di tempo.
Alla fine, dopo aver fatto qualche rendering di prova, è possibile modificare ad esempio il contrasto o la saturazione della Texture *Voxel Data*, oppure parametri come la turbolenza o la densità o altro ancora. Nel mio rendering il colore di *World* è nero, ma solo per rendere l'effetto più interessante rispetto a quello ottenibile col grigio di default.



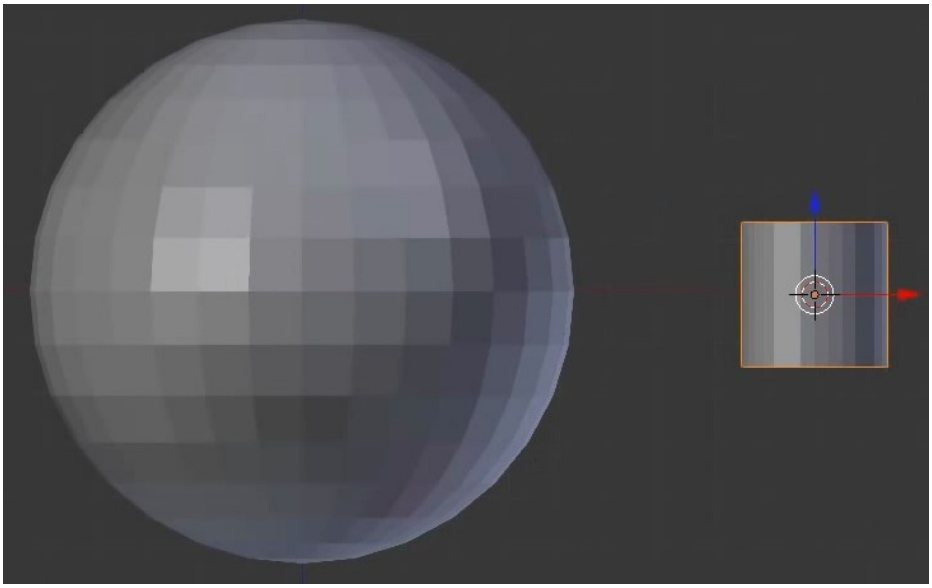
Uno screenshot del rendering finale dell'animazione

* * *

DupliVerts in Blender 3D 2.5, esempio 1: fori su mesh con Boolean Difference

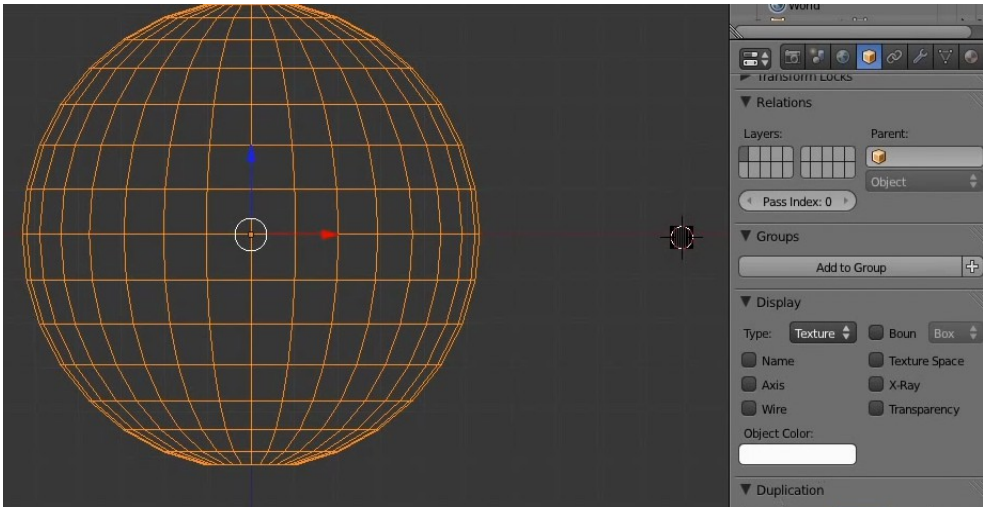
In questo tutorial vedremo come utilizzare i *DupliVerts* di Blender 3D per posizionare copie di un oggetto in corrispondenza dei vertici di un altro e realizzare fori sulla superficie dell'oggetto padre con le forme date mediante il modificatore *Boolean*.
Il tutorial è stato realizzato con la versione 2.57 stabile di Blender 3D.

Per realizzare quanto detto, ci servono due oggetti di partenza: l'oggetto padre, che è l'oggetto sul quale andremo a disporre gli elementi, e un oggetto figlio, che verrà duplicato e posizionato sul padre. Come oggetto padre possiamo utilizzare una mesh qualsiasi, sia primitiva che modellata a piacere, mentre come figlio possiamo utilizzare mesh o altri elementi, ad esempio anche fonti di luce, per disporre dei faretti lungo un percorso o altro ancora. Per questo esempio, utilizziamo una *sfera UV* come oggetto padre e un cilindro come oggetto figlio. Scaliamo in maniera opportuna gli oggetti; l'idea è di far corrispondere le copie del cilindro ai vertici della sfera, quindi rendiamo il cilindro di partenza abbastanza piccolo per non far “sconfinare” le copie sugli altri vertici.



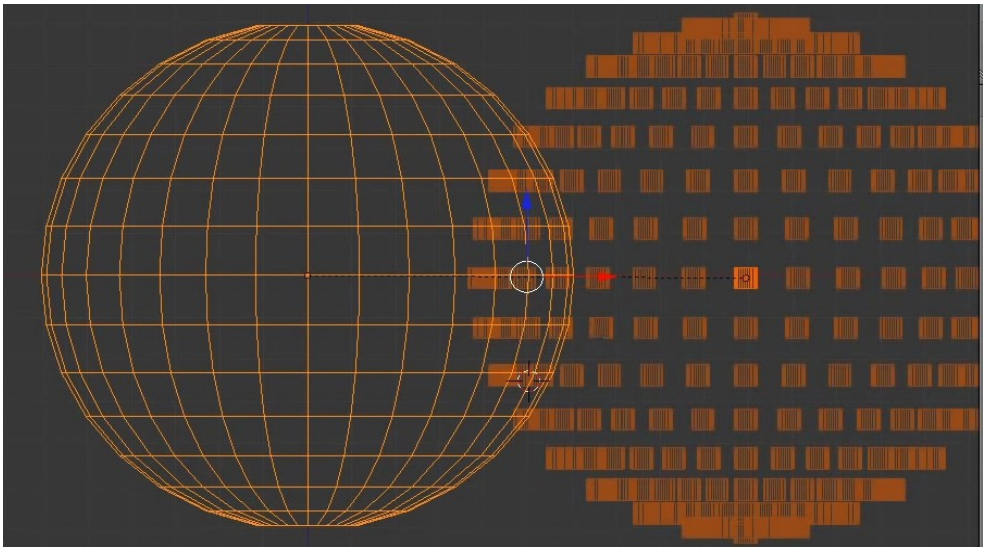
La scena utilizzata per questo tutorial

Adesso selezioniamo l'oggetto padre, apriamo la scheda *Object* nella *Properties Window* e sezioniamo l'opzione *Verts* nel pannello *Dupli* per indicare a Blender che vogliamo utilizzare la sfera come superficie sulla quale disporre gli oggetti, utilizzando come punti di riferimento i vertici della sfera. Questa è solo un'indicazione che stiamo dando a Blender, non stiamo specificando quali oggetti disporre sulla sfera, né troviamo campi del genere nella scheda Object; in realtà, sulla sfera verranno disposti gli oggetti figli della stessa, per cui dobbiamo realizzare una relazione di parentela rendendo il cilindro figlio della sfera.



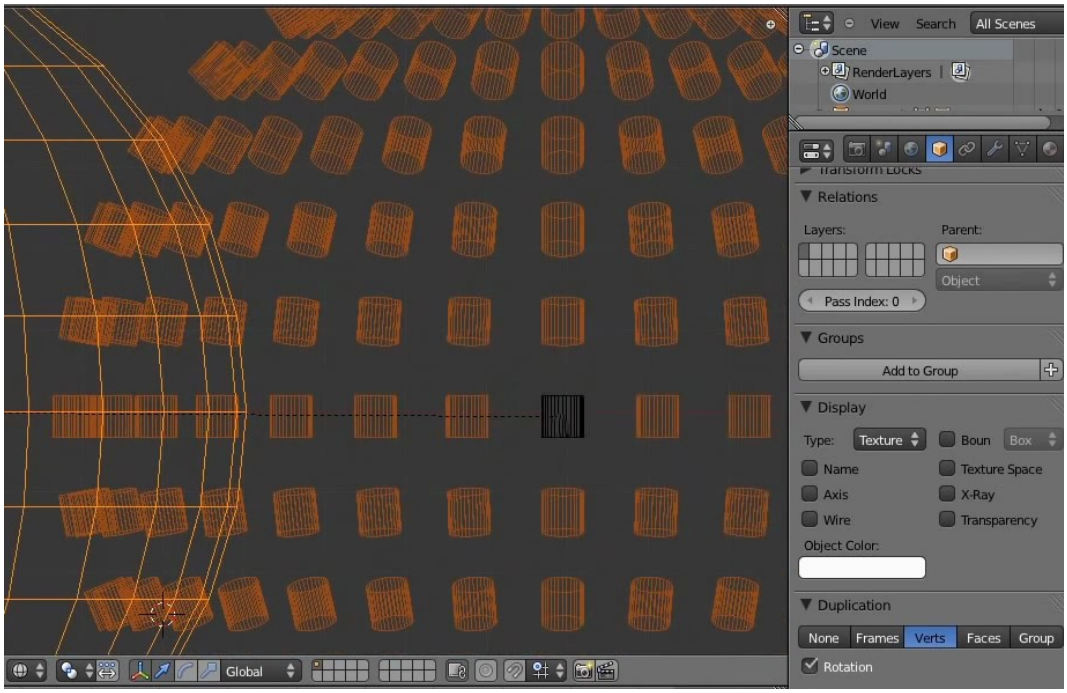
Dupli Verts, nella scheda Object

Selezioniamo quindi il cilindro, in quanto prima bisogna selezionare l'oggetto da rendere figlio, poi il padre, quindi selezioniamo la sfera con una selezione multipla (*SHIFT* e click del tasto destro del mouse sulla sfera), premiamo *CTRL P* e, nel menù che apparirà a video, confermiamo l'operazione. A video vedremo subito le copie disposte seguendo lo schema dei vertici.



Le copie realizzate e disposte automaticamente sui vertici

Di default le copie sono solo traslate, non sono orientate secondo le normali dei vertici, per cui selezioniamo la sfera e attiviamo l'opzione *Rotate*, deselezionata di default, nella sezione *Dupli*, all'interno della scheda *Object*.

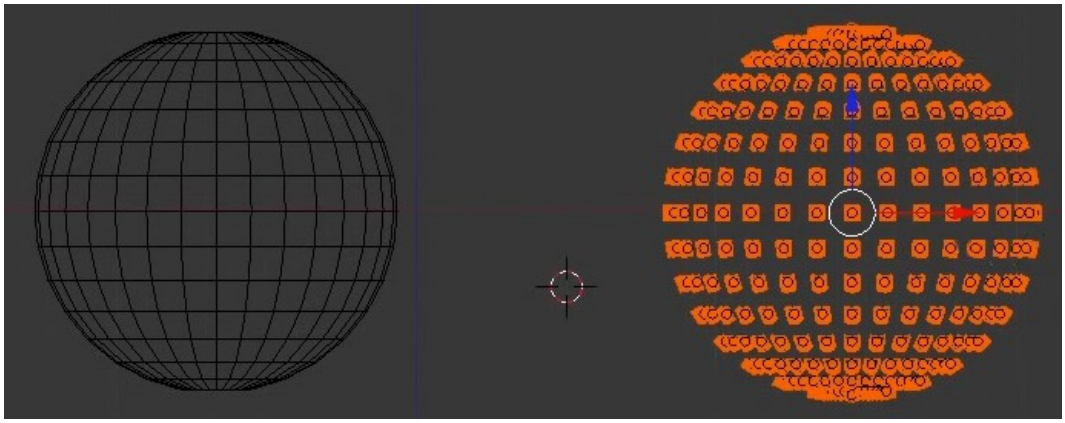


Opzione Rotation in Dupli Object

In questa fase è ancora possibile scalare e modellare l'oggetto figlio, nel nostro caso il cilindro, applicando le modifiche in tempo reale alle copie, quindi se vediamo ad esempio che gli oggetti sono troppo piccoli o troppo grandi possiamo scalarli.

Prima di procedere, dobbiamo effettuare un'operazione: rimuovere le facce dei cilindri interne alla sfera; fortunatamente, siccome le copie sono ancora istanze possiamo procedere rimuovendo le relative facce del cilindro originale, lasciando a Blender il compito di applicare le trasformazioni alle altre. Questa operazione è necessaria perché altrimenti *Boolean* sottrarrà il volume dei cilindri alla sfera, ma chiuderà comunque la geometria, creando pareti e fondo per ogni buco.

Attenzione, perché le copie non sono oggetti a parte, ma legati al padre, quindi cancellando ad esempio la sfera cancelleremo anche i vari cilindri; per creare nuovi oggetti a partire da *DupliVerts*, premiamo *CTRL SHIFT A*.

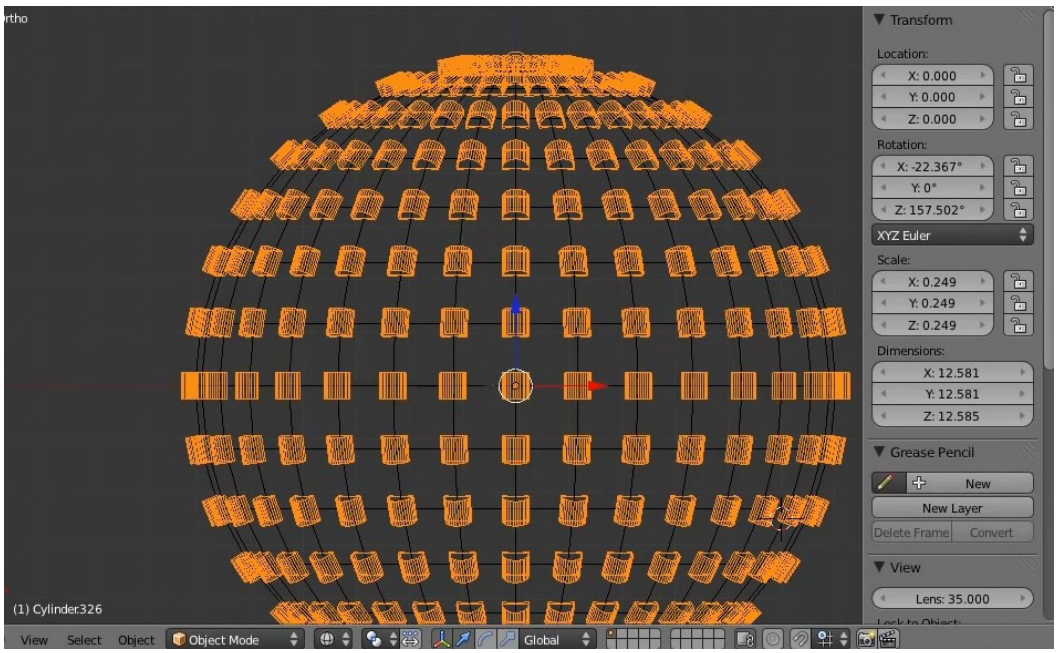


Creare nuovi oggetti dall'originale: CTRL SHIFT A

A questo punto avremo la sfera originale, il cilindro di riferimento (che a questo punto possiamo cancellare) e tante copie del cilindro originale, disposte e ruotate nello spazio 3D.

Blender renderà ogni singola copia un oggetto a parte; ciò è utile se vogliamo togliere alcuni elementi dalla selezione o applicare singole modifiche agli elementi; se ad esempio volessimo creare dei fori con forme diverse a seconda dei paralleli, potremmo applicare *DupliVerts* in due fasi, con due figli diversi, ed eliminare poi le varie "strisce".

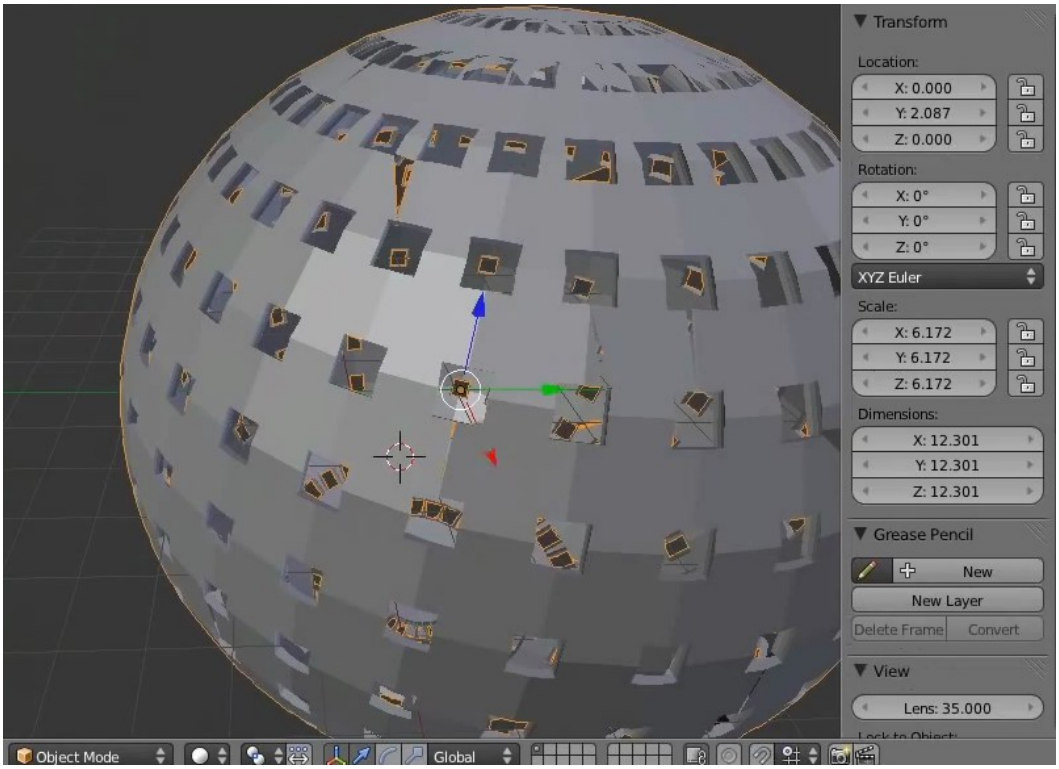
Dal momento che vogliamo applicare *Boolean* per sottrarre questi volumi alla sfera, selezioniamo tutti i cilindri e fondiamoli in un unico oggetto con *CTRL J*, per *Join*.



Le copie "fuse" in un unico Object con CTRL J (Join, unione)

Di default, questo nuovo oggetto non sarà centrato sulla sfera, ma utilizzando gli *SNAP* possiamo risolvere questo problema, ad esempio ponendo sia la sfera che il nuovo elemento nel centro della scena 3D (attenzione a sistemare l'*Origin* degli oggetti nel loro baricentro reale).

A questo punto selezioniamo la sfera (il primo operatore della sottrazione booleana), aggiungiamo un modificatore *Boolean* con operazione *differenza*, specifichiamo come secondo operatore l'insieme di cilindri appena fusi in un'unica mesh e... aspettiamo! A seconda del numero di elementi, dei vertici coinvolti e dell'hardware utilizzato, l'operazione *Boolean* può richiedere molto tempo.



Utilizzo del modificatore Boolean

Quando il cursore del mouse tornerà normale, *applicate subito il modificatore* (con un click su *APPLY*, nella scheda di *Boolean*) senza spostare ulteriormente le mesh, altrimenti ad ogni minimo aggiornamento perderete altri minuti perché Blender dovrà calcolare in tempo reale le operazioni.

Purtroppo i risultati non sono sempre buoni, alcuni vertici vengono fatti bene, altri danno luogo a forme strane; il modificatore *Boolean* è una delle cose da migliorare, in Blender 3D.

Attenzione anche a quanto fate penetrare i lati degli oggetti figli all'interno della sfera, quando rimuoverete una faccia del cilindro base e sposterete i suoi vertici, perché quelle porzioni di "pareti interne" verranno create per tutti i vertici.

Qui ho mostrato l'utilizzo di *Boolean* per la sottrazione ma va da sé che potete utilizzare l'unione per disporre i nuovi elementi sul primo.

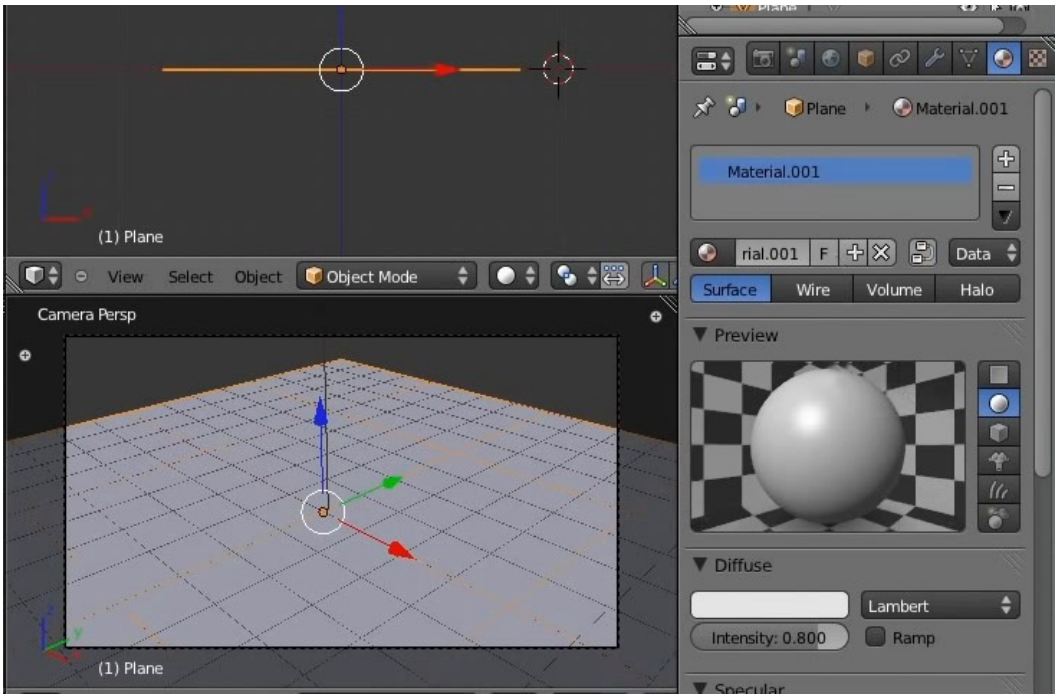
Se il modello creato non vi serve per la stampa 3D, allora anziché *Boolean Union* potete utilizzare *Join*, che applica le modifiche al volo.

* * *

Textures con mappatura UV sui due lati di una mesh con i Material Nodes

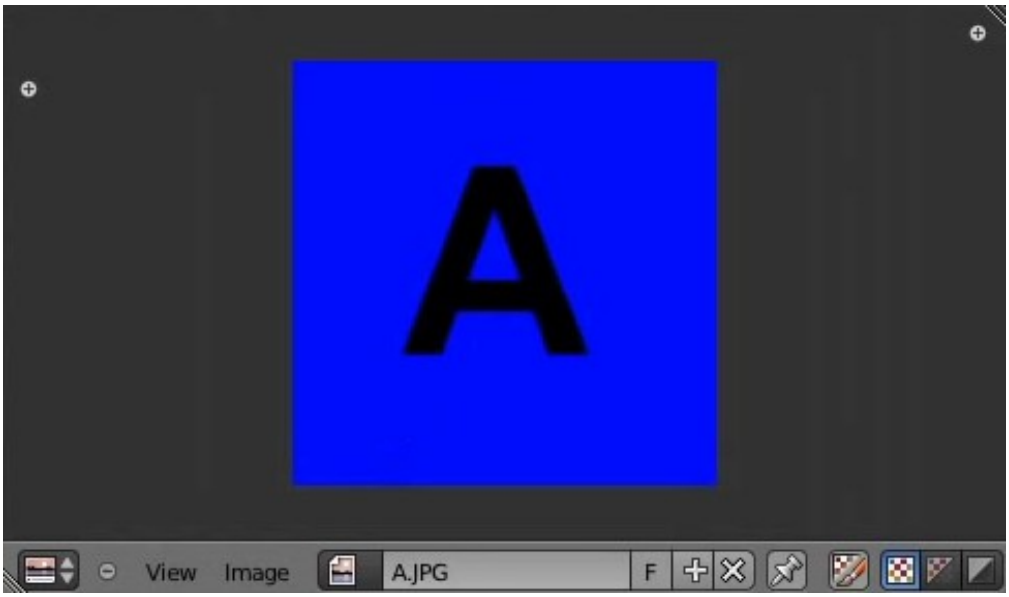
In questo tutorial vedremo come utilizzare l'editor dei *Nodes dei Materials* in Blender 3D 2.5 per applicare due Textures diverse ai due lati delle facce di un oggetto.
Il tutorial è stato realizzato con la versione 2.57 stabile di Blender 3D.

Per questo esempio prendiamo un semplice Plane, magari suddividendolo un paio di volte, dopodiché diamogli un Material (che possiamo personalizzare come vogliamo, con effetti di riflessione, trasparenza, ecc...).



La scena utilizzata in questo tutorial, con il Plane suddiviso e dotato di un semplice Material

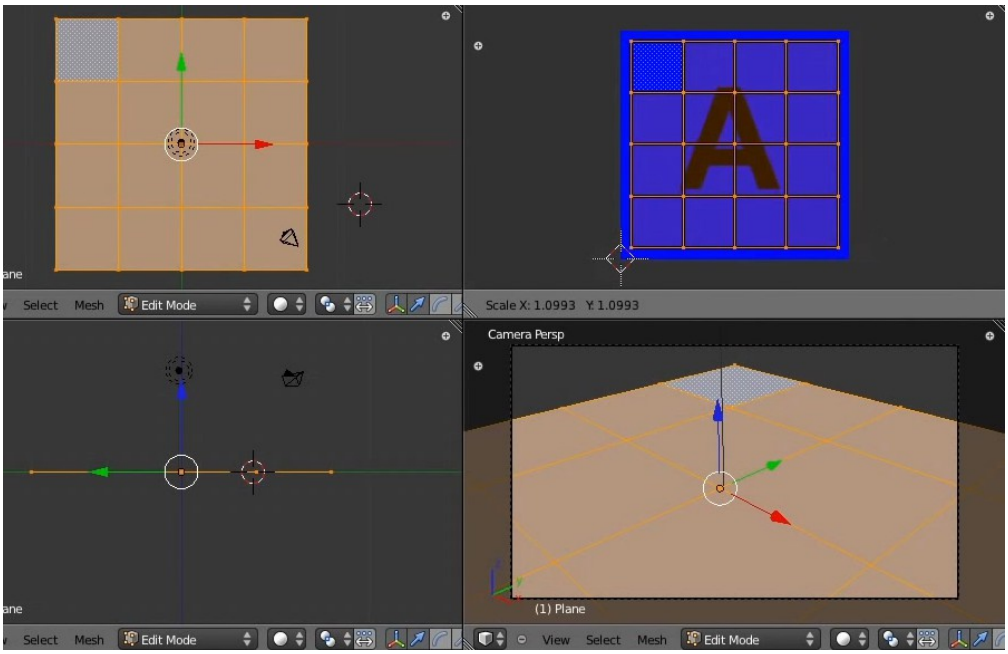
Cambiamo una finestra in *UV - Editor delle immagini* per procedere all'*unwrap* e alla creazione dei due *Layer UV* che ci servono. Carichiamo nella finestra *UV* l'immagine della Texture sul lato A dell'oggetto.



Caricamento della prima Texture immagine in una finestra UV

In una *3D View* dove possiamo inquadrare frontalmente il Plane, ad esempio la vista *TOP*, selezioniamo l'oggetto, passiamo in Edit Mode, selezioniamo tutti i vertici, premiamo *U* e scegliamo la scucitura *Project From View*.

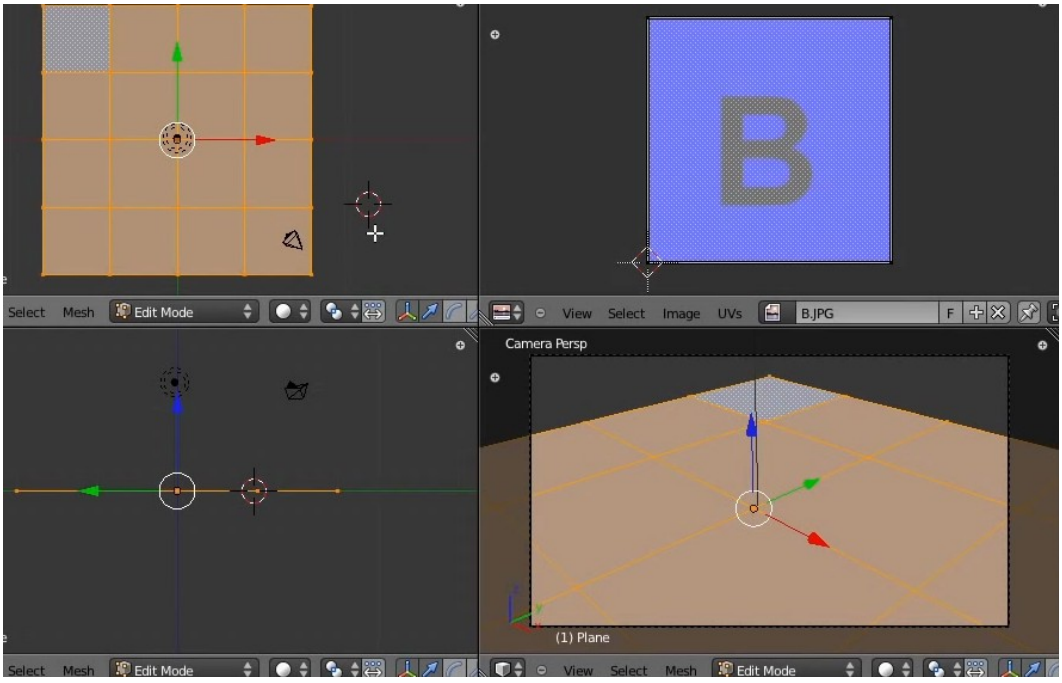
Posizioniamo la scucitura in maniera corretta nella finestra *UV - Editor delle immagini*.



UV-Mapping della prima Texture immagine, dopo aver effettuato la scucitura UV in una 3D View in Edit Mode

Blender ha associato queste informazioni al *Layer UV* di default, *UVTex*; nella sezione *UV Texture* della scheda *Object Data*, nella *Properties Window*, facciamo click sul pulsante + per aggiungere un altro *Layer UV* e selezioniamolo.

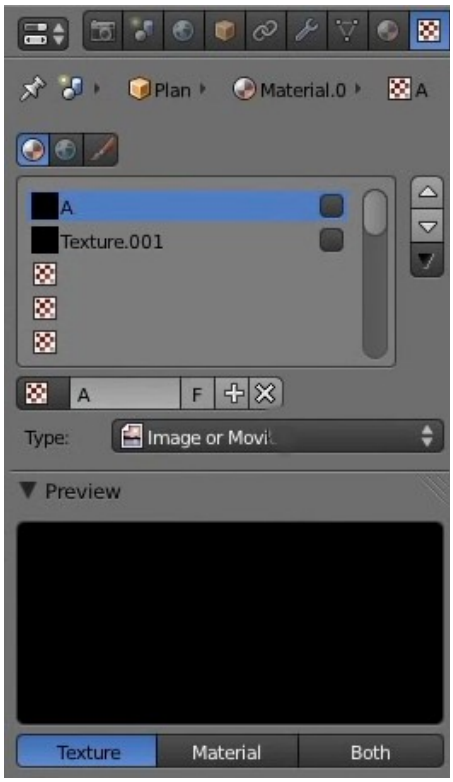
A questo punto, cambiamo l'immagine nella finestra *UV - Editor delle immagini* scegliendo quella del lato B e, nella 3D View, facciamo una nuova scucitura, associando le facce dell'oggetto alla nuova immagine. In questo modo la mappatura della prima immagine sarà immagazzinata nel primo *UV Layer*, la seconda in quello appena creato.



UV-Mapping della seconda Texture, con il nuovo Layer UV appena creato

Dobbiamo comunque inserire le Texture nella scena di Blender, visto che quelle appena definite sono solo le informazioni di mappatura; con l'oggetto selezionato, quindi, apriamo la scheda *Texture* nella *Properties Window*, creando due Texture di tipo *Image or Movie* ma deselezionandole nella pila di Textures, infatti non vogliamo applicarle direttamente, ma solo crearle per applicarle in seguito con l'editor dei *Nodes*.

Carichiamo le due immagini dei lati nelle due Texture ed impostiamo la mappatura di tipo *UV*, specificando il primo *Layer UV* per il lato A e il secondo per il lato B, nelle schede *Mapping* delle due Textures.



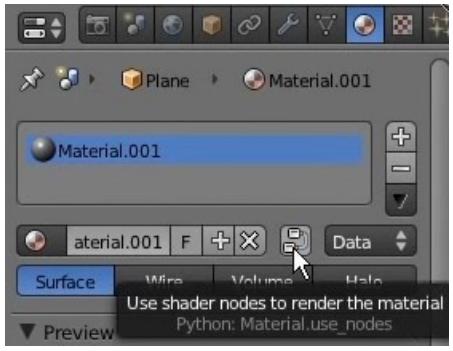
La creazione delle due Textures "disattivate"

Fatte queste operazioni di base nella scena, nella scheda Material dell'oggetto selezioniamo la casella "*Use Nodes*", dopodiché cambiamo una finestra in *Nodes Editor*, lasciamo selezionata l'icona "*Material Nodes*" e selezioniamo l'opzione "*Use Nodes*" a destra, nell'header.

Adesso dobbiamo personalizzare (e parecchio, anche) la configurazione dei *Nodes* per questo Material. Iniziamo aggiungendo due nodi di tipo *Input Texture*, da associare alle due immagini che rappresentano il lato A e il lato B.

Aggiungiamo inoltre due nodes di tipo *Input Geometry*, collegando il canale *UV* del primo al canale di input *Vector* della prima Texture; stessa operazione per la seconda coppia *Geometry – Texture*.

Nel primo nodo *Geometry*, impostiamo il nome del *Layer UV* creato per il lato A nel campo *UV Layer*, idem per la seconda coppia.

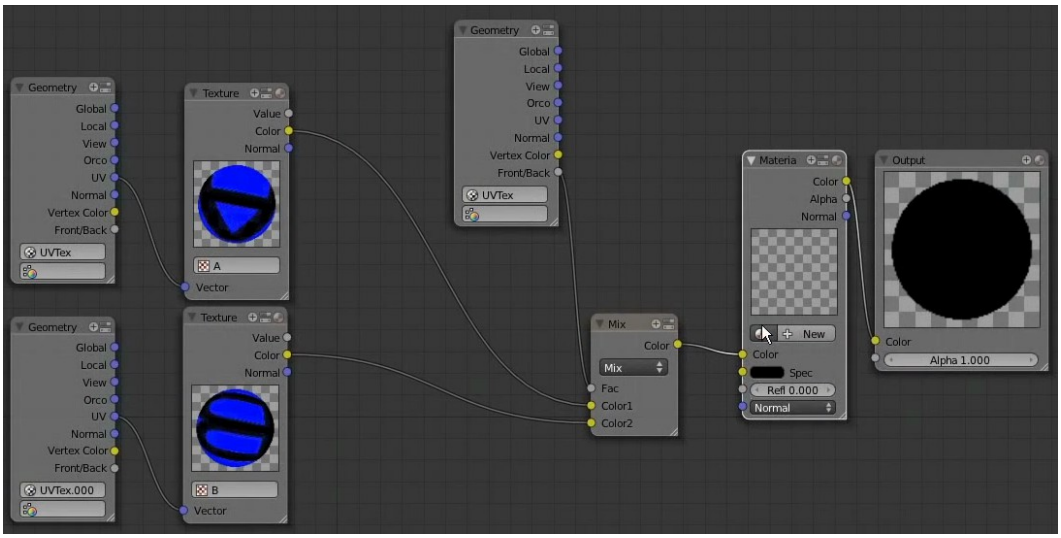


"Use Nodes" nella scheda del Material

Quello che stiamo facendo è chiaro: abbiamo inserito le due Textures nell'editor del Material e a ciascuna stiamo dando le informazioni di mappatura definite in fase di *unwrap*; tuttavia, dobbiamo ancora fondere in un unico Material le Texture e specificare che la prima appartiene a un lato, l'altra al secondo. Inseriamo quindi un nodo di tipo *Color Mix*, collegando i canali di output *Color* dei due nodi *Textures* ai relativi canali di input di *Mix*.

Per quanto riguarda il canale di input *Factor*, inseriamo un terzo nodo di tipo *Input Geometry* e colleghiamo il canale di output *Front/Back* (ossia: informazioni basate sul lato di ciascuna faccia che compone un oggetto) al canale di input *Factor* di *Mix*; in questo modo, Blender miscelerà le Textures applicando la prima sul lato *Front* delle facce, la seconda sul lato *Back*.

A questo punto, passiamo il canale output *Color* di *Mix* al canale input *Color* di Material e colleghiamo il Material al nodo *Output*, come avviene di default.



Lo schema di Nodi di Compositing per il Material

Abbiamo mappato correttamente le due Textures, con *mappatura UV*, sui due lati delle facce di un oggetto, come possiamo constatare facilmente (ad esempio, lanciando due rendering, ruotando di 180° il Plane tra un rendering e l'altro).

* * *

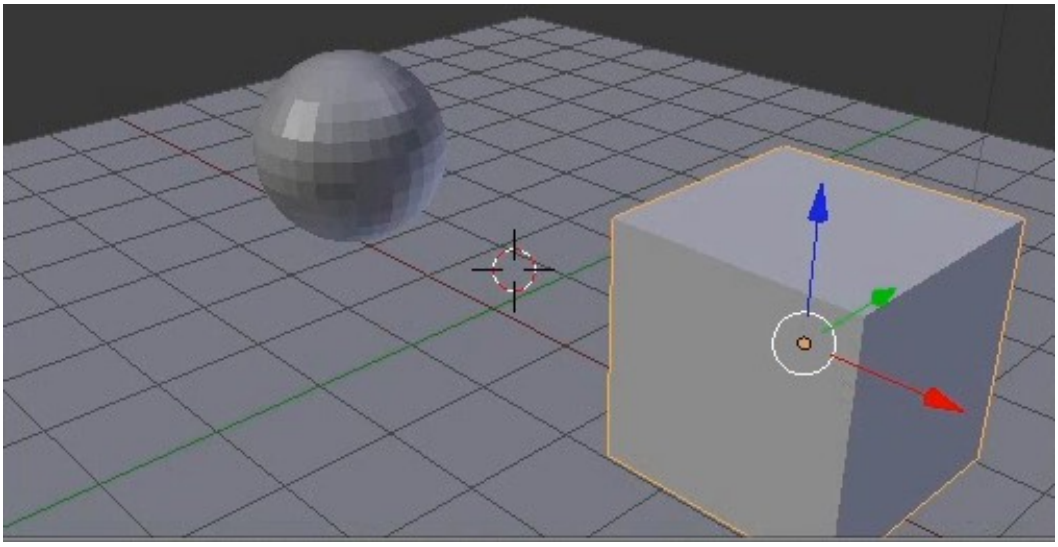
L'attuatore Constraint Force Field per gravità o forze repulsive nel BGE

In questo mini-tutorial parleremo dell'attuatore *Constraint Force Field* del Blender *Game Engine* 2.5, il motore di gioco di Blender 3D. *Force Field* permette di realizzare campi di forza attrattivi (come la forza di gravità) o repulsivi.

Il funzionamento è il seguente: l'oggetto che deve essere attratto o respinto deve possedere un attuatore di tipo *Constraint* con modalità *Force Field*, dove andremo a specificare il tipo di forza alla quale sarà soggetto quando si avvicinerà all'emettitore.

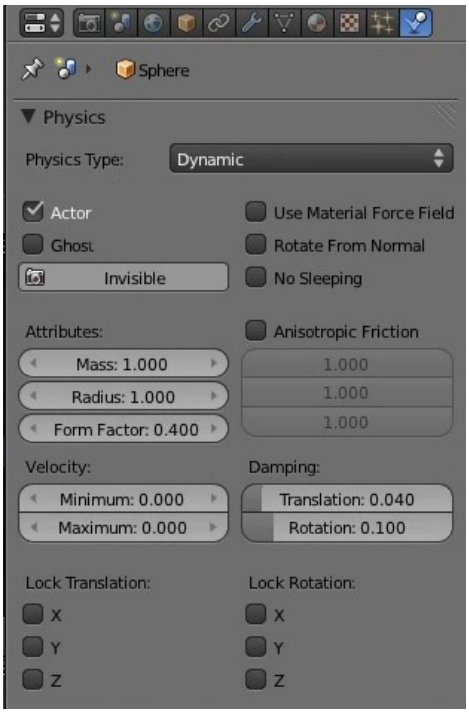
Identificheremo l'emettitore o mediante una *Property* o mediante un *Material*; nell'esempio che vedremo tra pochissimo, utilizzeremo questa seconda opzione.

Inseriamo tre oggetti nella scena: un Plane, che servirà da pavimento, un cubo (che sarà l'emettitore del campo di forza, ossia l'oggetto che attrarrà o respingerà gli altri) e una sfera, che invece risentirà del campo di forza.



La scena che utilizzeremo per questo tutorial

La sfera dovrà spostarsi per via del campo di forza, per cui dobbiamo impostarla a *Dynamic Actor* nella scheda *Physics*, all'interno della *Properties Window* (il motore è chiaramente *Blender Game*).



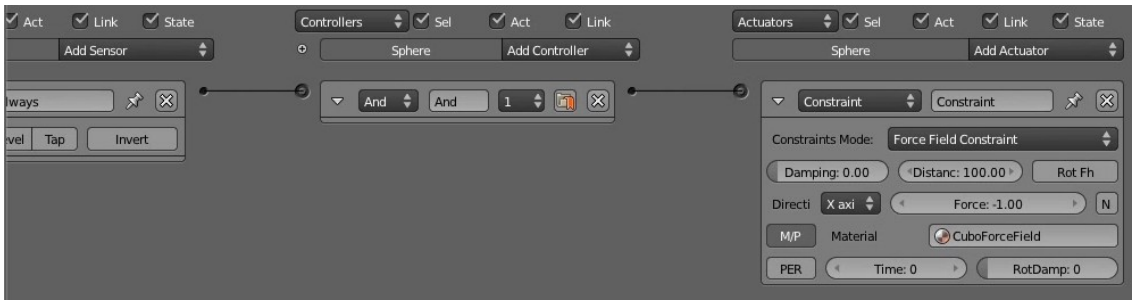
Le impostazioni Physics della sfera

Il Plane serve a non far cadere la sfera all'inizio; è, quindi, un oggetto *Static* non *Ghost*.

Il cubo è un oggetto *Static* di base.

Dotiamo il cubo di un Material a piacere e teniamo a mente il nome di tale Material, che chiameremo ad esempio "*CuboForceField*"; utilizzeremo questo nome nell'attuatore della sfera.

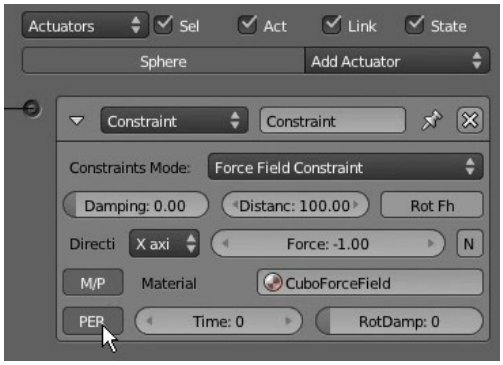
Selezioniamo appunto la sfera e forniamole un sensore *Always*, un controller *AND* e un attuatore *Constraint* di tipo *Force Field*. Colleghiamo sensore, controller e attuatore.



Sensori, controller e attuatori per la sfera

In questo caso sto utilizzando *Always*, quindi l'attrazione sarà sempre attiva, ma ovviamente possiamo utilizzare altri sensori di attivazione (i cosiddetti *Triggering Events*), ad esempio per attivare la gravità se si verificano certe condizioni date da *Properties* degli oggetti, se un giocatore preme un tasto particolare, e così via.

Nella scheda dell'attuatore impostiamo:

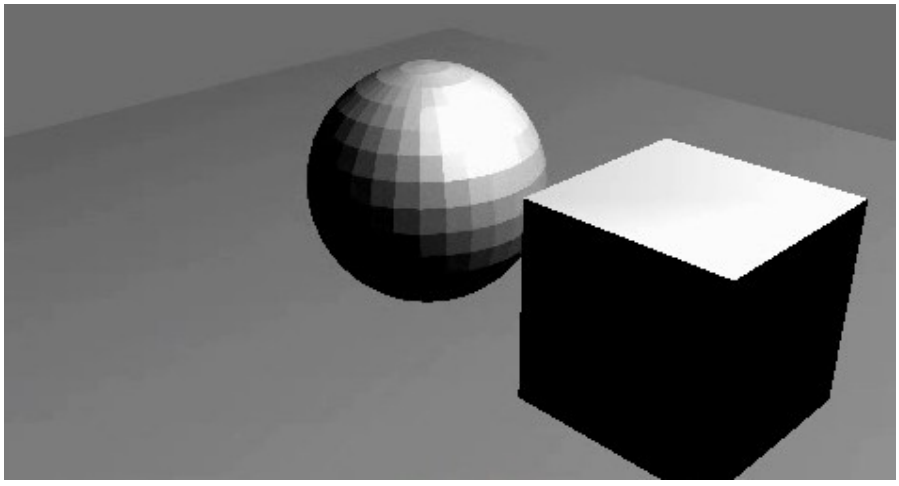


Le impostazioni dell'attuatore Constraint

- in *Distance*, la distanza entro la quale il campo di forza dovrà essere attivo;
- in *Force*, l'entità dell'attrazione (con valori negativi di *Force*) o della repulsione (con valori positivi);
- in *Axis*, l'asse lungo il quale dovrà farsi sentire questo attuatore; per applicare il *Force Field* in più direzioni, utilizzare più attuatori;
- in *Properties* o *Material*, la proprietà o il Material caratteristici dell'oggetto emettitore del campo di forza; nel nostro caso, il cubo ha un Material con nome "*CuboForceField*", per cui scegliamo la *M* e scriviamo tale materiale nel campo *Material*; l'attuatore *Force Field* della sfera sposterà quindi l'oggetto con forza e direzione dati da *Force* qualora tale oggetto si trovi a distanza inferiore a *Distance* dall'emettitore provvisto di Material "*CuboForceField*".

In realtà, in questo modo l'attuatore non sarà sempre attivo: dobbiamo o attivare *TRUE Level Triggering Pulse Mode* in *Always* oppure il pulsante *Permanent* nella scheda dell'attuatore. Scegliamo questa seconda opzione.

Nella scena ho inserito la sfera, provvista di *Constraint Actuator*, allineata sull'asse X col cubo in direzione negativa, per cui *Axis* va impostato a X positivo, perché il cubo si trova in posizione X positiva rispetto alla sfera. Adesso possiamo premere *P* per osservare il risultato ottenuto.

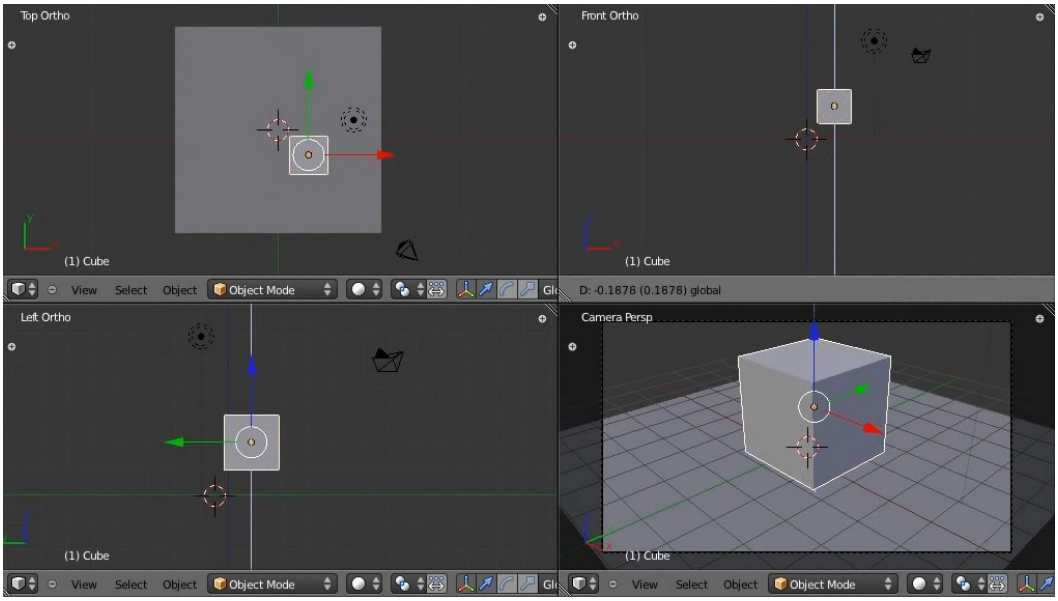

Screenshot dell'esecuzione: la sfera si avvicina automaticamente al cubo, fino a toccarlo

* * *

Realizzare un Tour Virtuale 3D con il Blender Game Engine 2.58, prima puntata

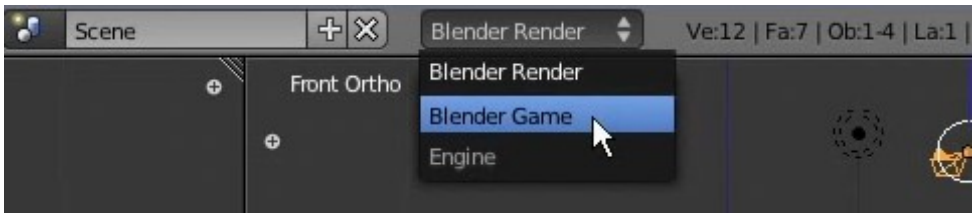
Questa è la prima di una piccola serie di puntate dedicate alla realizzazione di un'applicazione 3D, un tour virtuale all'interno delle stanze di un museo, con il Blender *Game Engine*, il motore di gioco di Blender 3D. Vedremo come impostare luci, ombre, Materials e Textures degli ambienti, come spostarci nelle "sale", interagire con gli elementi, ad esempio per ottenere informazioni sugli oggetti visualizzati, ed infine distribuire la nostra applicazione come eseguibile a parte, quindi non da avviare per forza all'interno di Blender 3D come simulazione. In questa puntata daremo un'occhiata alle impostazioni di base della *3D View* e dei Materials degli oggetti nel motore di gioco.

Avviamo Blender 3D 2.58. La mia area di lavoro presenta 4 viste *3D View*, organizzate con *TOP*, *FRONT* e *LEFT* in modalità ortogonale e la vista *Camera* in modalità prospettiva; la vista *Camera* è il nostro punto di osservazione all'interno del Game: andremo a spostarla con i tasti freccia per muoverci all'interno dell'universo virtuale.



L'area di lavoro

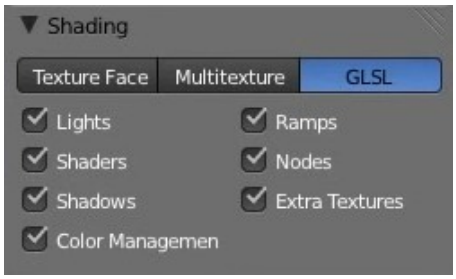
La prima cosa da fare è cambiare il motore di rendering utilizzato, che di default è *Blender Render*, in *Blender Game*, mediante l'apposito selettore *Engine* (motore, appunto) nell'header della finestra *Info*. Cambieranno alcune voci nelle schede all'interno della *Properties Window*; ne vedremo alcune, non tutte perché certe voci (come quelle riguardanti la fisica dei Materials) non ci interessano in questo tipo di progetto.



Passaggio a Blender Game Engine

In questo tutorial, diamo una prima occhiata alle proprietà dei Materials.
Il motore di gioco permette di implementare l'ombreggiatura, quindi l'illuminazione e il texturing degli oggetti, in più modi; a noi interessa la modalità *GLSL*, che tra l'altro sfrutta l'accelerazione hardware della scheda video, se l'hardware fornisce tale possibilità.

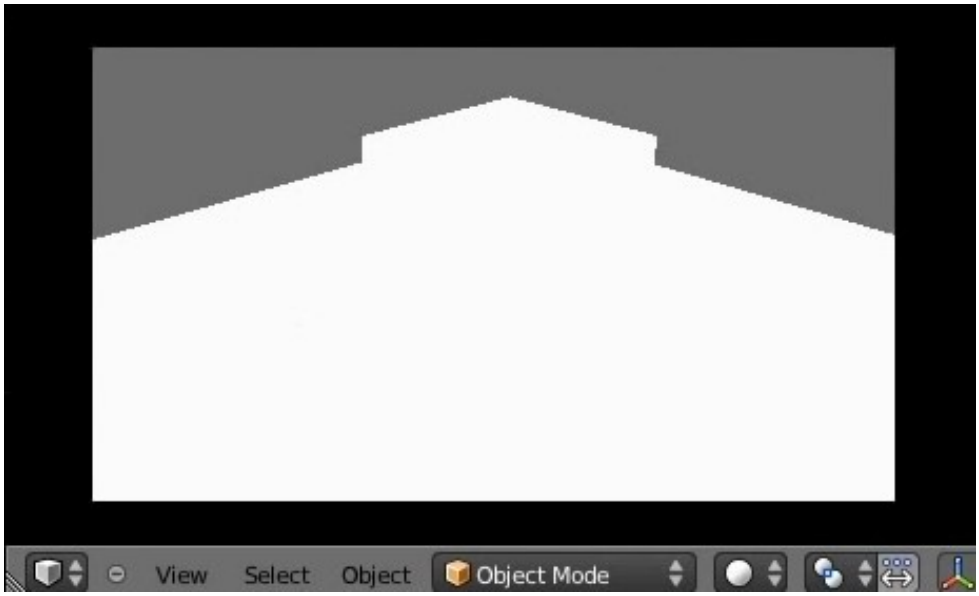
Per abilitare lo *shading GLSL*, apriamo la scheda *Render* all'interno della *Properties Window* e, nella sezione *Shading*, selezioniamo *GLSL*. Appariranno diverse caselle relative a fasi della pipeline di rendering; in maniera intuitiva, stanno ad indicare funzionalità da implementare quando si avvia il gioco all'interno di Blender, per fare delle prove.



Shading GLSL

Le caselle consentono ad esempio di disattivare gli effetti di illuminazione (*Lights*), il calcolo delle ombre (*Shadows*), l'utilizzo dei Nodes (*Nodes*) ed altre funzionalità; questa possibilità è utile quando dobbiamo fare dei test e, per accelerare la generazione delle anteprime, possiamo evitare di provare certi effetti. Lasciamo tutte queste voci selezionate, comunque.

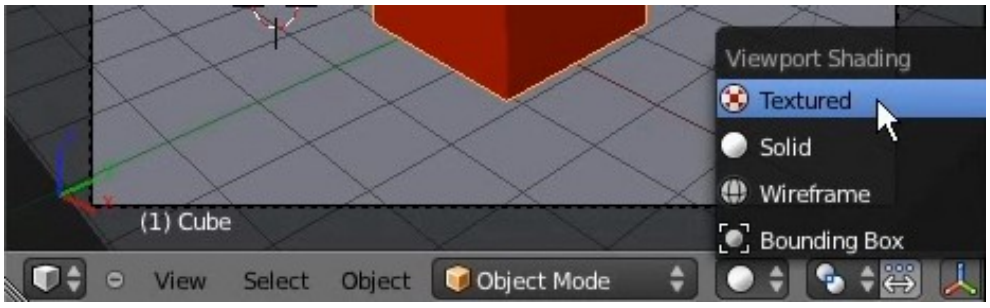
A questo punto vediamo come avviare la nostra applicazione, ancora all'interno di Blender; semplicemente, premiamo *P* quando il mouse si trova in una *3D View* o clicchiamo su *Game Start* nella scheda *Render* all'interno della *Properties Window.*



Esecuzione dell'applicazione, con P

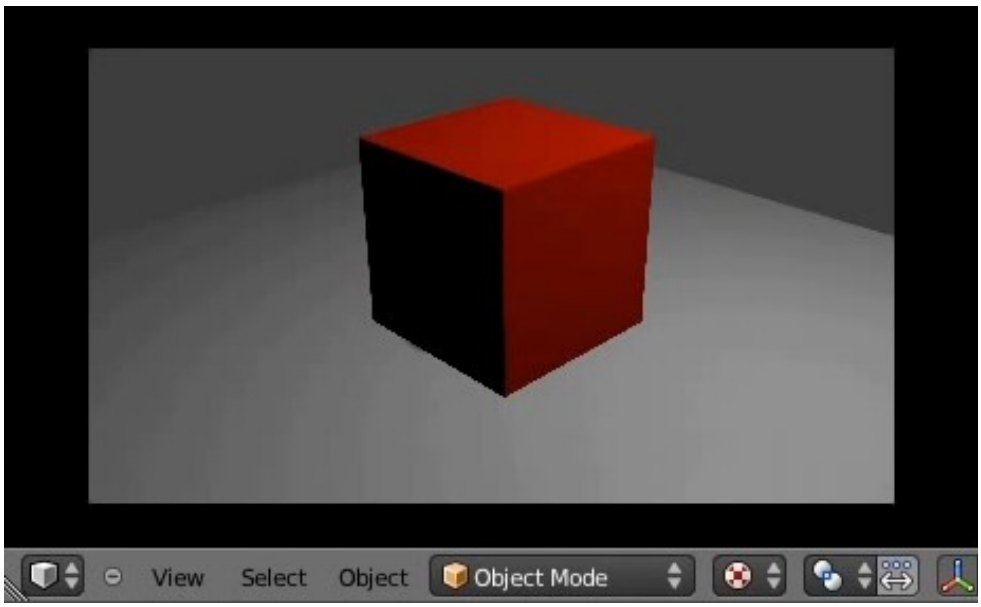
Con la scena di default, contenente un cubo e una fonte di luce (oltre alla telecamera) vedremo il cubo bianco, senza ombreggiatura. Per terminare la simulazione, premiamo *ESC*.

La situazione non cambia fornendo un Material colorato al cubo. Quello che dobbiamo fare è cambiare la modalità di visualizzazione della *3D View* (anche solo per la vista Telecamera) in *Textured*, mediante il selettore *Viewport Shading* nell'header della *3D View.*



Viewport Shading Textured, nella 3D View della vista Camera

Coloriamo il cubo di rosso, dandogli un Material con questo colore diffuso, poi avviamo la simulazione premendo *P*: questa volta, vedremo il cubo ombreggiato in maniera corretta.



La nuova esecuzione del gioco, con il cubo rosso e ombreggiato, per via di Textured in modo GLSL

Visto che abbiamo creato un Material per il cubo, diamo un'occhiata a tale scheda nella *Properties Window*: notiamo subito che molti pannelli sono identici a quelli visti nel caso dei Materials per *Blender Render*, inoltre abbiamo la sezione *Physics* per impostare proprietà fisiche di rimbalzo e scivolamento per gli oggetti che avranno tale Material.
Notiamo l'assenza di *Mirror*, e qui c'è una brutta sorpresa: i riflessi non sono abilitati di default, anzi non abbiamo proprio gli effetti di *RayTracing*, come vedremo presto con la trasparenza; ad ogni modo, è possibile risolvere questi problemi con alcuni "trucchetti".

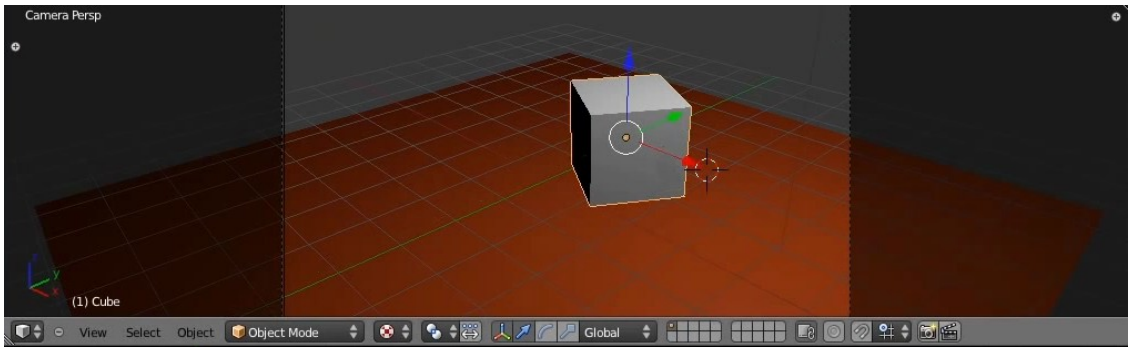
Come anticipato, la trasparenza *RayTrace* non è supportata, infatti attivando la trasparenza e selezionando tale modalità non appariranno i controlli tipici di *RayTrace* come *Glossiness* o *IOR*, l'indice di rifrazione del Material, solo il valore di opacità *Alpha*; passando alla modalità *Z-Transp*, invece, vedremo la trasparenza già nella *3D View*, in tempo reale, e ovviamente in modalità *Game Engine* con *P*. Tutte queste limitazioni sono dovute a motivi di performance del sistema.
Come vedremo nella prossima puntata, anche le ombre hanno un comportamento particolare; ad esempio, le *Spot Light* in modalità *Buffer Shadow* possono proiettare ombre dinamiche, le altre fonti di luce no, ma per le ombre statiche è possibile ricorrere a Textures *UV-Mapped*; si tratta quindi di trucchetti per aggirare limitazioni date da esigenze pratiche.

* * *

Realizzare un Tour Virtuale 3D con il Blender Game Engine 2.58, seconda puntata

Questa è la seconda di una piccola serie di puntate dedicate alla realizzazione di un'applicazione 3D, un tour virtuale all'interno delle stanze di un museo, con il Blender *Game Engine*, il motore di gioco di Blender 3D. In questa puntata, parleremo di luci e ombre nel *Game Engine*, trattando anche il *baking* delle *Shadows* e la loro applicazione come Textures con mappatura *UV Mapping*.

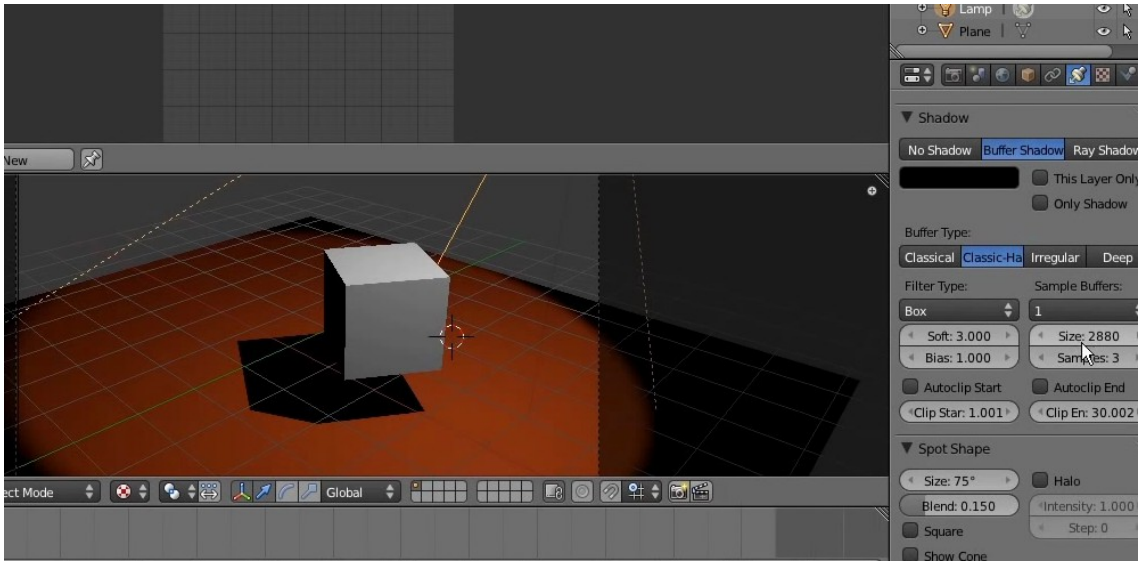
Impostiamo la scena con motore *Game Engine*, ombreggiatura *GLSL* e visualizzazione *Textured* nella *3D View* con vista *Camera*, come discusso nella puntata precedente.



La scena di partenza per questo tutorial

Il motore di gioco cerca di mediare tra buoni risultati visivi ed esigenze pratiche, considerando le limitazioni date dall'hardware e dal fatto che tutto va eseguito in tempo reale e possibilmente con un buon frame rate; mentre nel rendering classico possiamo anche aspettare ore per un singolo frame, qui ci servono diversi frame al secondo, per cui certi effetti non possono essere (ancora, almeno) implementati in maniera realistica ma bisogna ricorrere a dei trucchetti.

Le ombre non fanno eccezione. Nel *Game Engine* l'unica fonte di luce che può proiettare ombre in tempo reale è *Spot Light*, il faretto, tra l'altro con modalità *Buffer Shadow.*



Le impostazioni di una Spot Light con ombre Buffer Shadow

Tra le varie impostazioni, quella che ci interessa è *Shadow Buffer Size*, la dimensione dell'area di memoria da riservare al calcolo delle ombre; intuitivamente, a valori maggiori corrisponderanno risultati migliori, ma non tutte le schede video riescono a supportare valori alti e, in questi casi, Blender semplicemente non mostrerà l'ombra (ad esempio la GeForce Go 7300 di un portatile non supporta il valore 4992, mentre una GTX della serie 200 o successive non si fa di questi problemi).

Ho parlato di schede video perché i giochi realizzati con ombreggiatura *GLSL* si appoggiano proprio sull'accelerazione hardware delle schede; ecco qui un caso nel quale ad essere importante è la scheda video, mentre nel rendering con *Blender Render* conta invece la CPU.

Con questo tipo di illuminazione possiamo proiettare ombre in tempo reale che interagiscono con gli elementi della scena e li seguono nei movimenti.

Piccolo problema: le ombre trasparenti non vengono proiettate, neanche variando *Shadow Casting Alpha* per il Material con modalità *Irregolare* o *Deep* per la luce *Spot Light* (queste impostazioni servono a proiettare ombre trasparenti in fase di rendering con *Blender Render* ma non risolvono il problema con il *Game Engine*).

A questo punto potremmo pensare che per avere le ombre nel nostro videogioco dobbiamo utilizzare sempre delle *Spot Light* con *Buffer Shadow*... beh, questo è vero solo per le ombre in

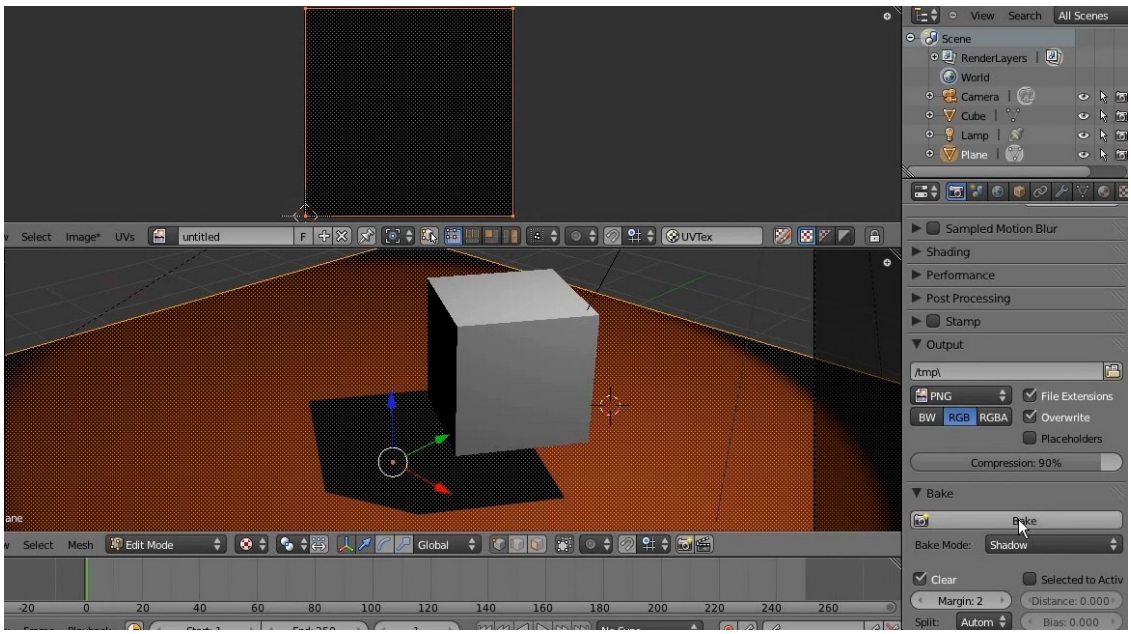
tempo reale, ossia quando la fonte di luce si sposta, quando l'oggetto si sposta e quando c'è interazione tra le ombre proiettate da un oggetto e altri elementi.

Se non si verificano queste condizioni (e lo scenario del museo virtuale è il caso ideale), allora possiamo ricorrere ad un trucchetto, utile anche per ottenere ombre da altre fonti di luce, non solo *Spot Light*: l'applicazione di ombre pre-calcolate mediante *baking* e Textures con *blending Multiply*.

È un trucco utilizzato nello sviluppo di molte applicazioni 3D, videogiochi e non, e questo in generale, non solo col Blender *Game Engine*.

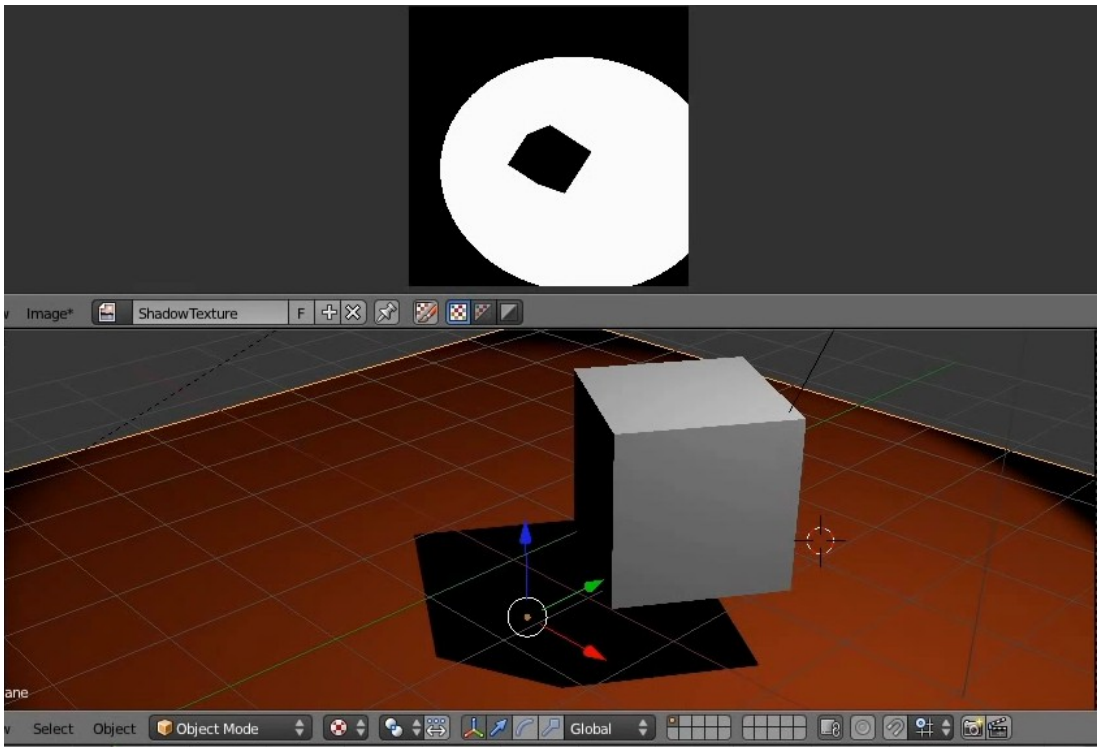
Ecco cosa bisogna fare: passare al motore *Blender Render*, impostare la scena, con luci e oggetti nei punti giusti, avviare il *baking* delle ombre per gli oggetti da mettere in ombra, ottenendo una *Texture "shadow"* per l'oggetto, tornare nel *Game Engine* ed applicare tale immagine al Material con modalità *Multiply*.

Per effettuare il *baking* delle ombre, bisogna selezionare l'oggetto che riceverà le ombre (ad esempio, il Plane) farne l'*unwrap* con un metodo di scucitura a piacere, passare alla finestra *UV - Editor delle immagini* per aggiungere una nuova Texture di default, andare nella scheda *Render* all'interno della Properties Window e fare il *Bake* delle *Shadows*.



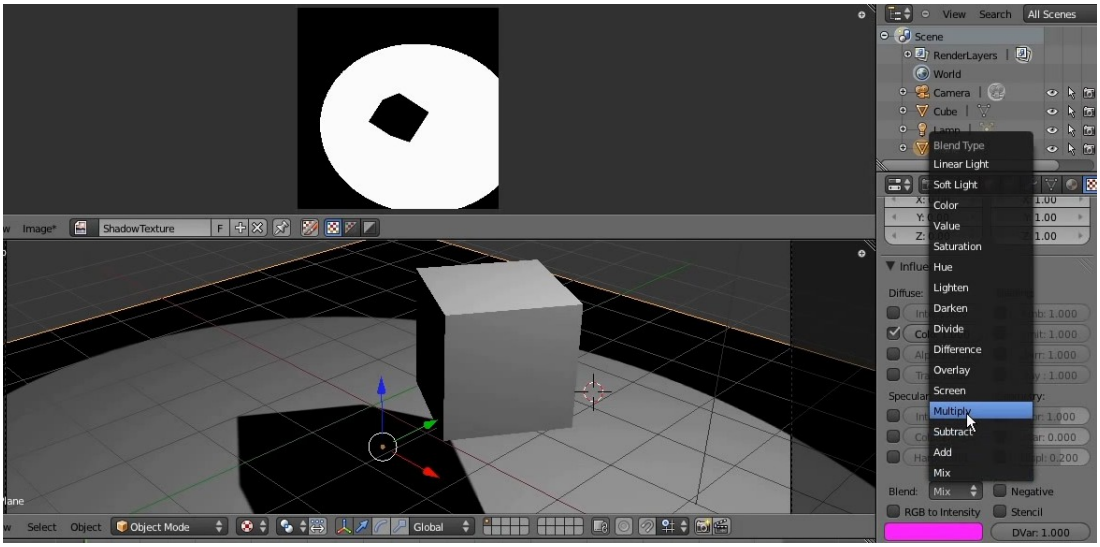
Effettuare il Baking

A seconda della complessità della scena, tale operazione può richiedere un bel po' di tempo; ad ogni modo, alla fine vedremo la Texture delle ombre nella finestra *UV - Editor delle immagini* (tale risultato sostituirà infatti l'immagine nera di default creata un attimo fa con *New*).



L'immagine delle ombre proiettate sul Plane

Ovviamente, l'immagine ottenuta (che può essere salvata su disco) andrà ri-mappata sul Plane, ancora una volta con *mappatura UV*, per far combaciare l'ombra con l'oggetto che la sta proiettando.



Mappatura dell'immagine Baked come Texture UV sul Plane, in modalità Multiply (le ombre vanno infatti moltiplicate al colore)

In questo modo, durante il gioco l'ombra non verrà calcolata istante per istante, sprecando risorse, ma sarà in un certo senso "statica", calcolata una volta e applicata sempre, infatti in questo momento sto usando una *Omni Light*, che non proietta ombre: l'ombra che vediamo è una Texture!

È ovvio che, come anticipato, non ci devono essere interazioni tra questi oggetti ed altri, altrimenti si scoprirebbe il "trucco". Tecnicamente, gli oggetti e la fonte di luce non devono muoversi, perché l'ombra resterebbe immobile, tuttavia se i movimenti previsti sono ben definiti, fissati una volta per tutte e non interattivi, è possibile effettuare il *bake* delle ombre frame per frame ed utilizzare come *Texture UV-Mappata* una sequenza di file immagine anziché un singolo file. Per realizzare quest'ultima funzionalità è necessario, però, ricorrere a script in *Python* per il *Game Engine*, quindi non ne parleremo in questa puntata.

Mediante un programma di fotoritocco è possibile simulare anche le ombre trasparenti semplicemente cambiando il nero della *Texture Shadow* in un valore di grigio: l'operazione *Multiply* ha infatti lo scopo di rendere neri i pixel corrispondenti a parti nere della *Texture Shadow*, lasciare inalterati quelli corrispondenti a pixel bianchi e rendere più o meno scuri tutti gli altri.

Prima di chiudere questa puntata, una considerazione: se nel nostro set le fonti di luce sono fisse e ci sono oggetti statici e altri animati in maniera interattiva, possiamo ottimizzare il calcolo delle ombre utilizzando una Texture pre-calcolata per gli oggetti statici e giocare un po' con i *Light Groups*, trattati nella trentanovesima puntata del mio corso sulle basi di Blender 3D 2.5, in modo da utilizzare ad esempio una seconda *Spot Light* con *NO SHADOWS* oppure una *Omni* (che non proietta ombre in tempo reale) per gli oggetti statici, dotati di ombre pre-calcolate, e la *Spot Light* per quelli dinamici, risparmiando così risorse in fase di esecuzione in real-time.

Come potete intuire, si tratta più che altro di trucchi per aggirare certi ostacoli e sprecare meno risorse a runtime.

* * *

Realizzare un Tour Virtuale 3D con il Blender Game Engine 2.58, terza puntata

Questa è la terza puntata di una piccola serie dedicata alla realizzazione di un'applicazione 3D, un tour virtuale all'interno delle stanze di un museo, con il Blender *Game Engine*, il motore di gioco.

In questa puntata daremo una prima occhiata a Sensori, Controller, Attuatori e Proprietà. Parlerò di questi elementi in generale, non solo per quanto riguarda il progetto di tour virtuale, in modo da dare una visione d'insieme e suggerirvi spunti per realizzare altri progetti con altre caratteristiche.

Il concetto di base è che ogni elemento presente nella nostra applicazione può avere delle proprietà, alle quali possiamo associare dei valori: un giocatore può avere un nome (quindi una proprietà stringa con valore il nome del giocatore), un determinato punteggio (quindi una proprietà numerica), un pulsante può essere premuto o meno (una proprietà booleana, zero o uno) e così via.

Non solo: a seconda del verificarsi o meno di certi eventi o condizioni, possono essere effettuate delle operazioni; ad esempio, premendo la freccia in avanti la telecamera può spostarsi, oppure se un oggetto si avvicina troppo ad un altro può essere attratto dalla sua gravità o rimbalzare in caso di contatto, e così via.

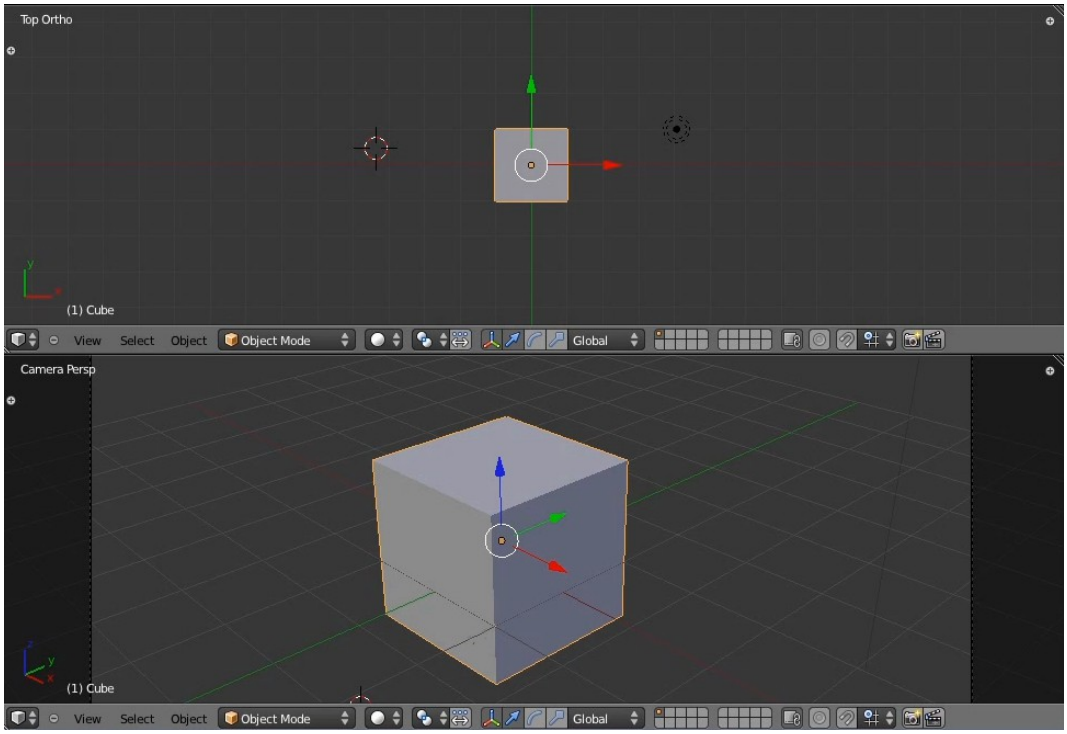
In questo caso, si parla di sensori associati agli elementi; quando un sensore viene attivato, vengono effettuate una o più operazioni mediante degli attuatori. Per effettuare un'azione, può essere necessario attivare un solo sensore, due o più sensori insieme o un sensore tra un insieme determinato.

Ovviamente, è possibile cambiare il valore di una proprietà mediante un attuatore, mentre il cambiamento di valore di una proprietà può attivare un sensore; esempio: premiamo un tasto per

aumentare la potenza dei motori, quindi il valore della proprietà "velocità" aumenterà, ma a tale valore abbiamo associato un sensore che, quando il valore supera una certa soglia, mediante un attuatore fa apparire un messaggio di pericolo a video.

Le potenzialità sono tantissime, comunque ora torniamo al nostro tour virtuale 3D!
Intuitivamente, abbiamo bisogno di sensori, controller e attuatori almeno per spostarci tra le sale, ma non ci limiteremo a questo; vedremo, ad esempio, come far apparire o scomparire pannelli informativi ed altri elementi in base a determinate condizioni di attivazione.

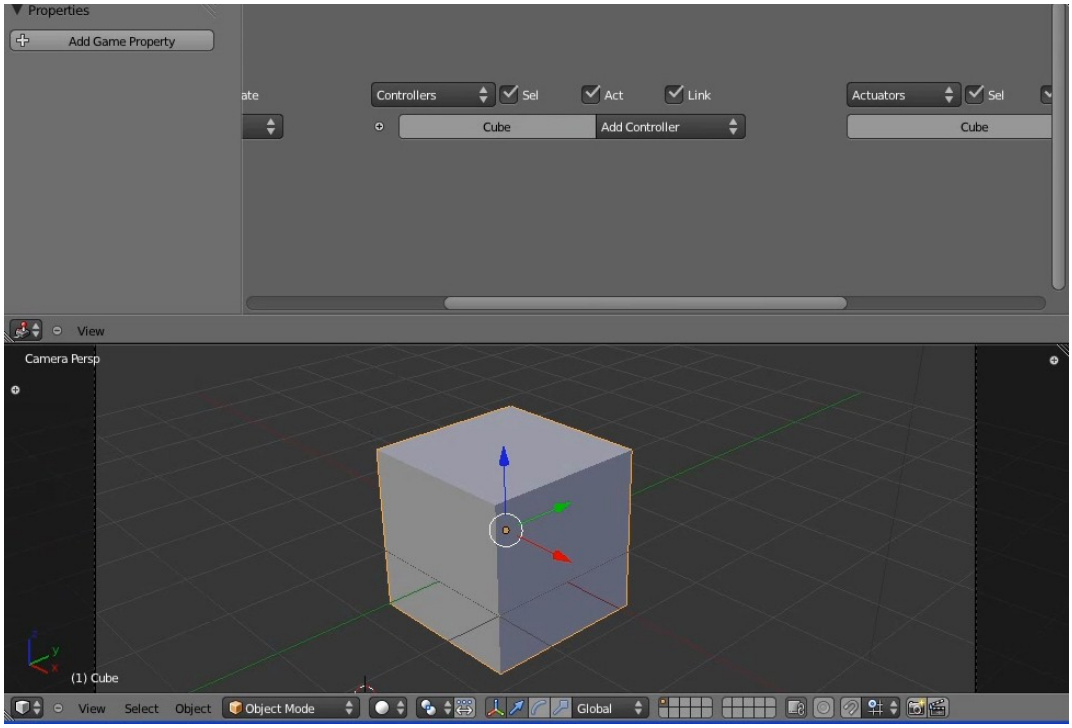
Finora abbiamo parlato di sensori, controller, attuatori e proprietà a livello teorico, ora vediamo dove si trovano tutte queste informazioni nella nostra scena 3D.



La scena utilizzata per questo tutorial

Per prima cosa, selezioniamo un oggetto: effettuando questa operazione, avremo modo di definire proprietà e condizioni solo per l'oggetto selezionato.

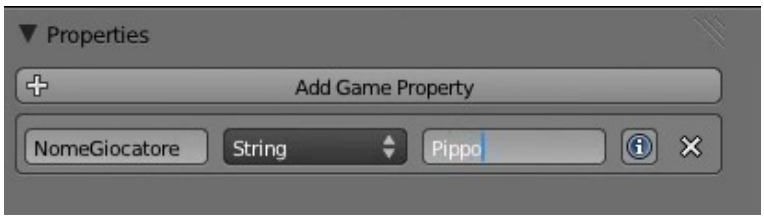
A questo punto cambiamo una finestra *3D View* in finestra *Logic Editor*.



Una finestra Logic Editor, per l'oggetto selezionato

Nella parte sinistra della finestra abbiamo l'area dedicata alle *Properties*, le caratteristiche dell'oggetto selezionato.

Aggiungere una proprietà è semplice: clicchiamo su *Add Game Property*, cambiamo il tipo dell'elemento selezionato in base a quello che ci interessa (ad esempio, *String* per parole o frasi), specifichiamo nella casella di sinistra il nome della proprietà e a destra il valore da associarle; ad esempio, chiamiamo la proprietà "*NomeGiocatore*", ovviamente di tipo *String*, con valore "*Pippo*".

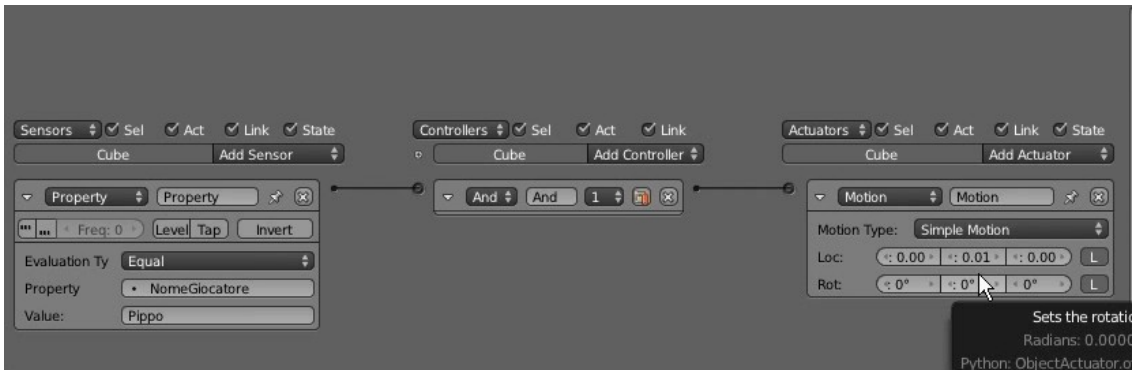


La proprietà "NomeGiocatore", di tipo String, con valore "Pippo"

Nella parte destra della scheda troviamo la catena sensore-controller-attuatore: sotto ciascuna pila possiamo aggiungere una nuova voce e, mediante le porte di collegamento, realizzare appunto dei collegamenti con click del tasto sinistro del mouse e trascinando, come vedremo tra un attimo.

Aggiungiamo ad esempio un sensore *Property*, ossia un sensore che, frame per frame, verifica il valore di una proprietà dell'oggetto e, se tale proprietà ha il valore specificato nel campo *Value* del sensore, emette un segnale positivo sulla porta di uscita; scriviamo quindi "*NomeGiocatore*" come *Property* e "*Pippo*" come valore da verificare, lasciando la modalità *Equal*: nel nostro caso, tale proprietà è sempre verificata.

Aggiungiamo un controller *AND* e un attuatore *Motion*, collegando la porta di output del sensore *Property* a quella di input di *AND* e quella di output di *AND* a quella di input di *Motion*.

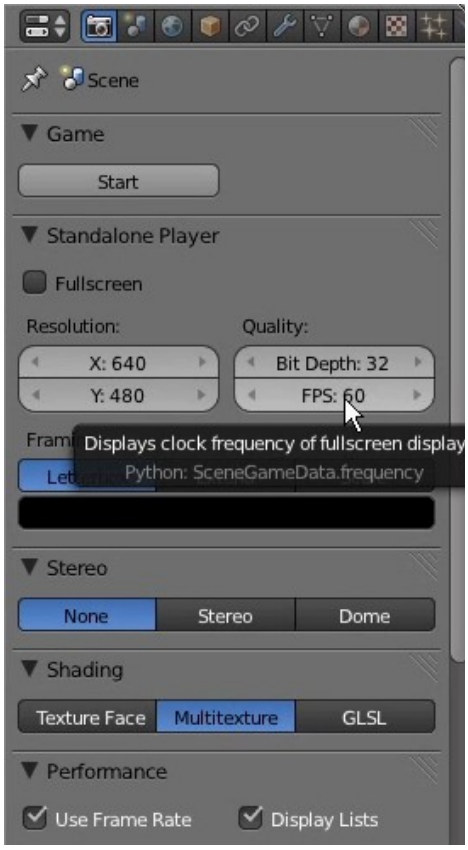


Sensore, controller e attuatore collegati nel Logic Editor dell'oggetto

In questo modo, il "segnale" emesso dal sensore quando la condizione è verificata passerà in *AND* e, non dovendosi confrontare con altri segnali, passerà indisturbato fino all'attuatore *Motion*, attivandolo. *Motion* realizza, appunto, un movimento, per cui dobbiamo impostare un qualche valore di spostamento per vedere degli effetti; mettiamo ad esempio 0.01 in *LocY*: ciò significa che, ad ogni frame, l'oggetto che possiede questa proprietà e questa catena sensore-controller-attuatore, si sposterà di 0.01 unità di Blender lungo l'asse Y locale (infatti la casella *L* è selezionata) positivo.

Prima di passare all'esecuzione, trattiamo l'argomento *fps*: è noto che i videogiochi vengono eseguiti con un certo frame rate, ossia un certo numero di aggiornamenti dello stato e della scena al secondo; di default, Blender esegue l'applicazione a 60 fps, come è possibile constatare nella scheda *Render* all'interno della *Properties Window* (ovviamente, il motore dev'essere *Blender Game*).

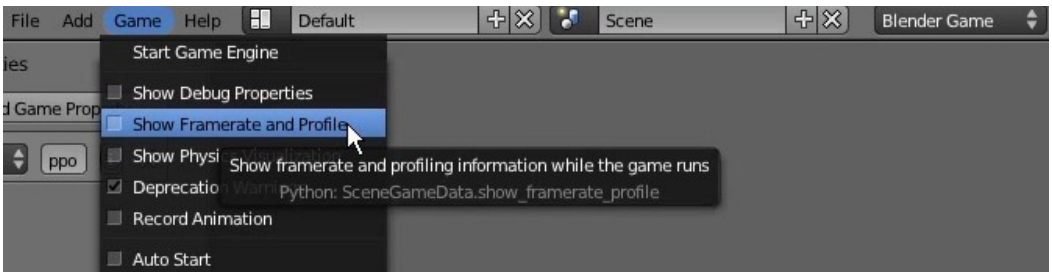
In tale scheda possiamo anche variare il valore, se ad esempio vogliamo eseguire l'applicazione a 24 frames al secondo, più altri parametri, come ad esempio la risoluzione di output (che di default è 640x480) e altro ancora.



La sezione Render nella Properties Window

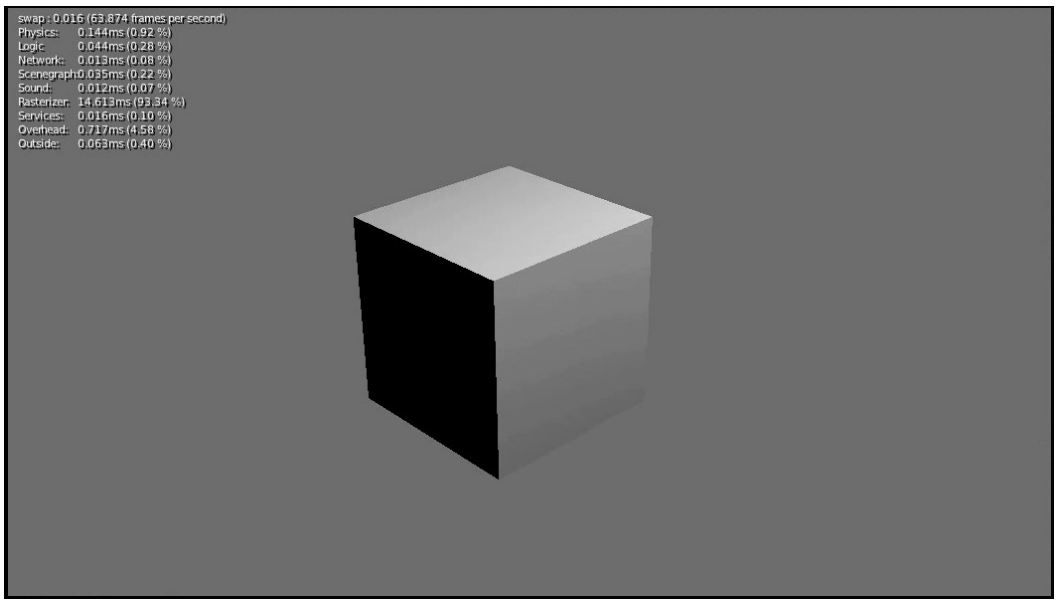
In fase di test può essere utile voler leggere a video i valori di certe caratteristiche dell'esecuzione, ad esempio il frame rate implementato effettivamente, cioè in tempo reale; per far ciò, selezioniamo la casella *Show Framerate and Profile* nel menù *Game*, all'interno della finestra *Info*.

Nel menù ci sono altre voci simili, dai significati intuitivi.



Show Framerate and Profile, nel menù Game

Torniamo alla nostra *3D View* e premiamo *P*: vedremo l'oggetto spostarsi lentamente (a passi di 0.01 unità per frame, quindi 0.6 unità di Blender al secondo, nel nostro caso) lungo l'asse Y locale positivo. In alto a sinistra nella schermata possiamo leggere le informazioni sul framerate e il profilo dell'applicazione, visto che abbiamo attivato la relativa casella nel menù *Game*.



Il progetto in esecuzione

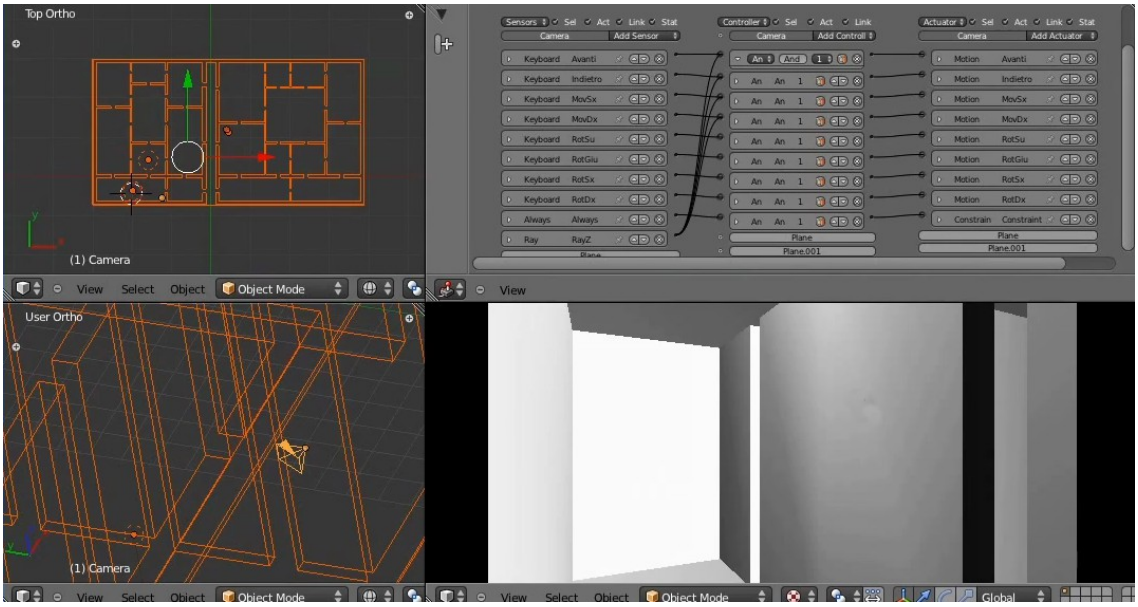
Ricapitolando: possiamo associare proprietà, eventi ed azioni agli elementi della scena secondo il paradigma sensori-controller-attuatori e la definizione di *Properties* di vario tipo.

Gli oggetti possono avere anche proprietà fisiche, reagire alla gravità o agli urti ed altro ancora; in queste puntate comunque non vedremo in maniera approfondita queste opzioni, limitandoci a quelle che possono interessarci per il tour virtuale.

* * *

Realizzare un Tour Virtuale 3D con il Blender Game Engine 2.58, quarta puntata

Questa è la quarta puntata di una piccola serie dedicata alla realizzazione di un'applicazione 3D, un tour virtuale all'interno delle stanze di un museo, con il Blender *Game Engine*, il motore di gioco di Blender 3D. In questa puntata vedremo come muoverci all'interno della "sala", realizzando in maniera corretta spostamenti e rotazioni ed evitando di passare attraverso muri ed oggetti. Faremo uso del *Constraint Location* e del sensore *Ray*... prima, però, qualche informazione sul movimento del nostro oggetto telecamera.



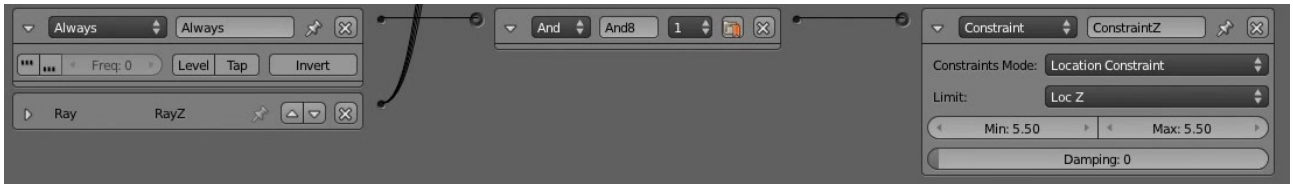
La scena utilizzata per questo tutorial

Anche se l'ambiente si sviluppa su un solo piano, vogliamo fornire all'utente la possibilità di ruotare la vista. Utilizzeremo i tasti freccia per ruotare la telecamera e i tasti *WASD* per spostarla, ma dobbiamo far attenzione ad una cosa: a volte dovremo utilizzare gli assi globali anziché quelli locali, selezionati di default nell'attuatore *Motion* (infatti la casella *L* appare come "premuta").

Ecco in particolare qual è il problema: ruotando verso l'alto la telecamera, ossia intorno al suo asse X locale, e premendo poi *W* per spostarla in avanti con l'attuatore *Motion*, che agisce sull'asse Z locale della telecamera, sposteremo la telecamera in alto o in basso nella scena 3D.

Per risolvere questo problema, inseriamo un sensore *Always*, che come suggerisce il nome invia continuamente segnali positivi, il solito controller *AND* obbligatorio e un attuatore di tipo *Constraint Location*, impostando il vincolo sull'asse Z (quello globale della scena) con valori *min* e *max* dati dall'altezza che vogliamo dare alla telecamera; nella mia scena, la telecamera si trova a 5.5 unità dal terreno, quindi sto impostando questo valore nell'attuatore.

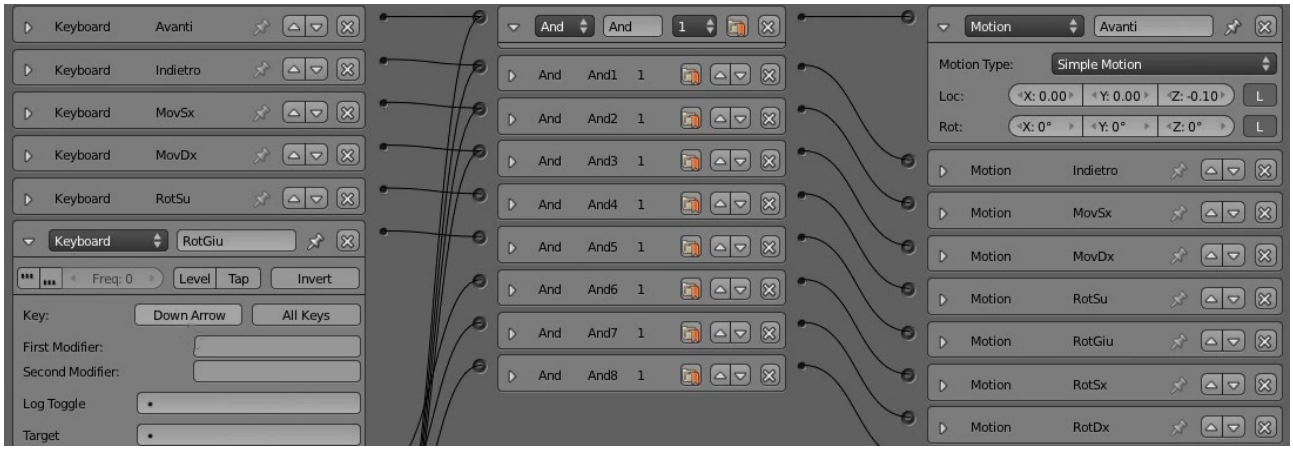
In questo modo, anche ruotando verso l'alto o verso il basso la telecamera e spostandola, il movimento avverrà mantenendo l'oggetto all'altezza corretta.



Il sensore ALWAYS collegato in AND ad un attuatore CONSTRAINT LOCATION sull'asse Z globale, per mantenere la telecamera ad una certa quota dal suolo

Il secondo problema da risolvere riguarda invece gli assi: la rotazione verso destra e verso sinistra deve avvenire intorno all'asse Z, ok, ma tale asse dev'essere quello *GLOBALE* della scena, non quello locale della telecamera (che per inciso è quello che ci fa spostare avanti o indietro la telecamera), altrimenti avremo degli effetti indesiderati ruotando l'oggetto se questo è stato precedentemente ruotato verso l'alto o verso il basso... provare per credere.

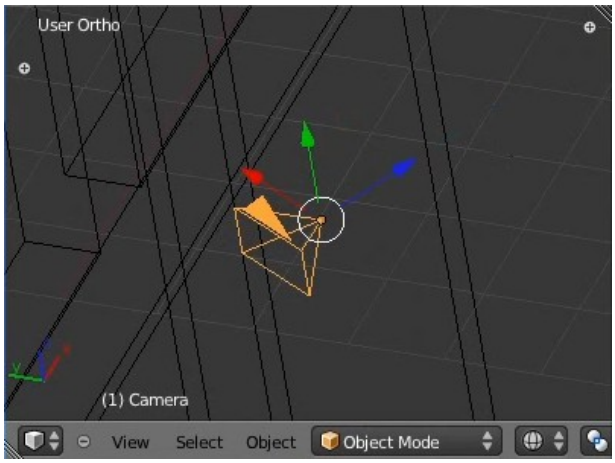
Diamo quindi un'occhiata ai sensori, controller ed attuatori che si occupano del movimento e della rotazione del punto di vista dell'osservatore nella scena: 8 sensori *Keyboard*, 1 sensore *Always*, 9 controller *AND*, 8 attuatori *Motion* e un attuatore *Constraint*.



Lo schema di sensori, controller e attuatori per il movimento e la rotazione del punto di vista

I sensori *Keyboard* si limitano a ricevere gli stimoli in arrivo mediante la tastiera.
I comandi sono: le frecce su, giù, destra e sinistra per le rotazioni nelle relative direzioni; *W* per lo spostamento in avanti, *A* per lo spostamento laterale a sinistra, *D* per lo spostamento laterale a destra, *S* per andare indietro. Il sensore *Always* non ha comandi particolari: invia in continuazione un impulso positivo.

Gli attuatori *Motion* utilizzati per spostare la telecamera agiscono sugli assi locali dell'oggetto.



Gli assi locali della telecamera

Com'è possibile osservare in una *3D View* selezionando la telecamera e passando alla modalità *LOCAL*, l'asse Z è quello perpendicolare al piano dell'immagine ed ha valori positivi verso dietro, cioè verso l'utente, mentre i valori negativi si hanno andando verso lo schermo, "in avanti", in un certo senso; inoltre, lo spostamento laterale (e quindi la rotazione verso l'alto o verso il basso) si ha sull'asse X locale dell'oggetto, con valori positivi "a destra".

Gli attuatori *Motion* hanno quindi valori -0.1 e 0.1 sull'asse Z locale per andare rispettivamente avanti e indietro e valori, -0.1 su X locale per lo spostamento laterale a sinistra e 0.1 su X locale per lo spostamento laterale a destra.

La rotazione verso l'alto o verso il basso avviene, come anticipato, intorno all'asse X locale dell'oggetto, quindi l'attuatore *Motion* per la rotazione verso l'alto avrà valore 1 grado su *Loc* X, mentre quello verso il basso avrà valore -1° su *Loc* X; ovviamente, i valori che sto suggerendo per definire l'entità degli spostamenti e delle rotazioni sono solo indicativi e potete variarli come preferite. Gli attuatori di rotazione verso sinistra e verso destra avranno invece valori 1° su Global Z e -1° su Global Z, quindi con la casella *L* deselezionata per i motivi precedentemente detti.

Come anticipato, l'attuatore *Constraint*, di tipo *Location*, agisce sull'asse Z globale vincolando costantemente la posizione della telecamera a 5.5 unità di altezza dal pavimento.

Detto questo, passiamo alle collisioni, o meglio: a fermare la telecamera prima che possa toccare elementi della scena, in questo caso le pareti.

Esiste un sensore, *RAY* (lett.: "raggio") che proietta un raggio lungo un determinato asse locale dell'oggetto che lo possiede e che manda degli impulsi positivi quando il raggio, proiettato per una certa distanza massima, tocca un elemento dotato di una certa proprietà.

Il sensore RAY

La distanza percorsa dal raggio può essere regolata ovviamente a piacere mediante la casella *Range* del sensore; lasciando bianco il campo *Property*, il sensore si attiverà in prossimità di qualsiasi oggetto della scena, e questo è il nostro caso perché vogliamo evitare l'impatto con qualsiasi tipo di elemento presente, non solo le pareti.

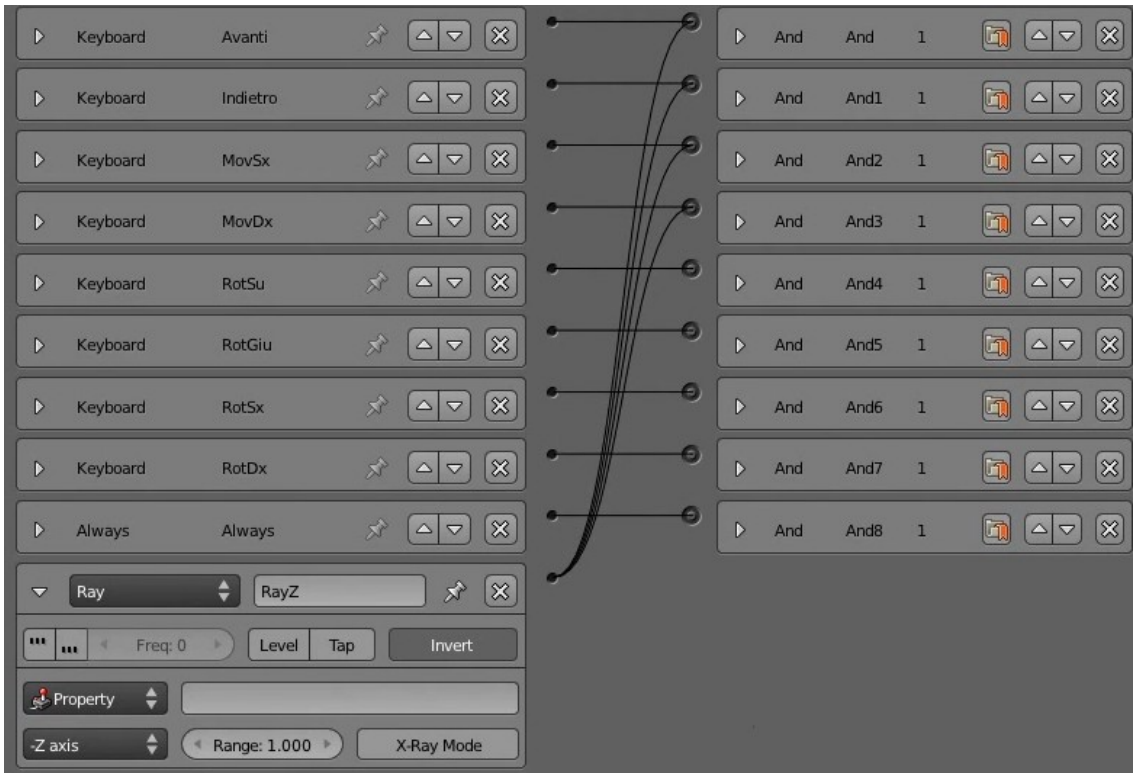
Neanche a dirlo, l'asse sul quale andrà proiettato il raggio è quello *Z NEGATIVO* della telecamera, cioè quello che va "in avanti".

Problema: *RAY* invia un impulso positivo quando la telecamera è vicina all'oggetto, entro una certa distanza. Collegando quindi *RAY* in *AND* con i vari sensori di movimento (solo quelli di spostamento, non rotazione sul posto), avremo l'effetto opposto a quello desiderato: *AND* restituisce infatti 1 se e solo se tutti i segnali in ingresso sono a 1, quindi potremmo muoverci solo se molto

vicini agli oggetti mentre resteremmo fermi oltre una determinata distanza, e noi vogliamo l'esatto contrario! Fortunatamente, il sensore *RAY* mette a disposizione il comando *INVERT* che, se selezionato, inverte l'output del sensore.

In questo modo, *RAY* restituirà *TRUE* in assenza di oggetti sul suo percorso (fino a distanza *Range*), per cui in *AND* con il sensore di movimento ci permetterà di muoverci, mentre restituirà *FALSE* quando il raggio intersecherà un qualsiasi elemento della scena, bloccando il movimento.

Nel mio caso, ho fissato *Range* a 1; ovviamente, dovremo porre il sensore *Ray* in *AND* con tutti e 4 i sensori di movimento, ma fortunatamente non dobbiamo creare 4 copie di *Ray*: ne basta una.

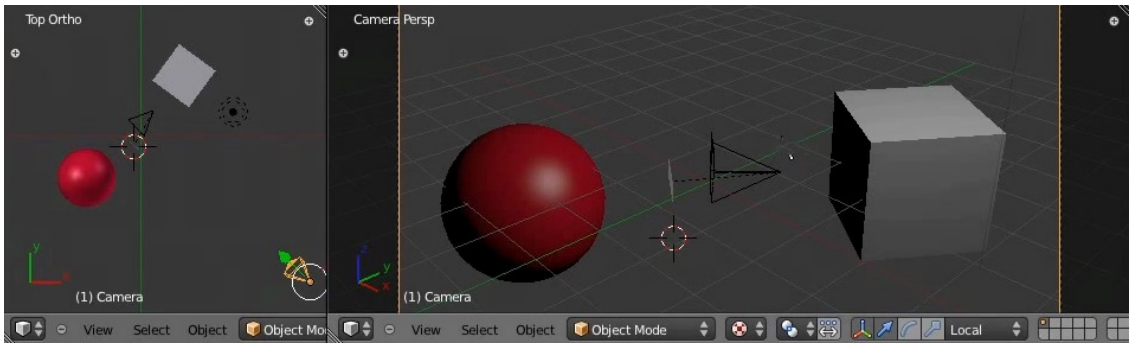


Il sensore RAY messo in AND con i sensori KEYBOARD (per il movimento)

* * *

Realizzare un Tour Virtuale 3D con il Blender Game Engine 2.58, quinta puntata

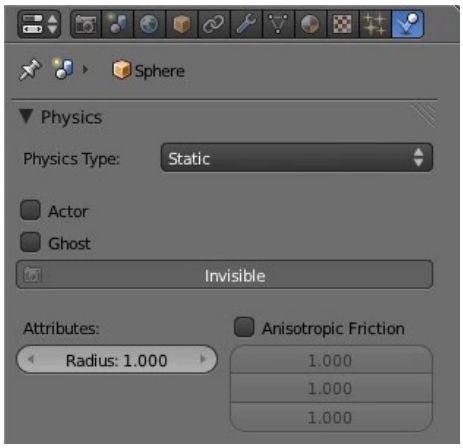
Questa è la quinta puntata di una piccola serie dedicata alla realizzazione di un'applicazione 3D, un tour virtuale all'interno delle stanze di un museo, con il Blender *Game Engine*, il motore di gioco di Blender 3D. In questa puntata vedremo come far apparire pannelli informativi e generare altri eventi quando ci avviciniamo a certi oggetti. Dal momento che, per trattare questi argomenti, devo fare alcune considerazioni sugli *Actor* e i vari tipi di elementi, non utilizzerò (come potete notare) la scena del museo virtuale, ma una piccola scena costruita appositamente per questo scopo; i concetti verranno poi applicati nella scena del museo 3D.



La scena utilizzata in questo tutorial

Parliamo dell'apparizione degli oggetti, iniziando dal concetto di visibilità: all'avvio del gioco, un oggetto della scena può essere impostato a visibile o invisibile mediante un'apposita voce nella scheda *Physics* dell'oggetto, nella *Properties Window*; tale voce è disponibile se il motore utilizzato è *Blender Game*, ovviamente.

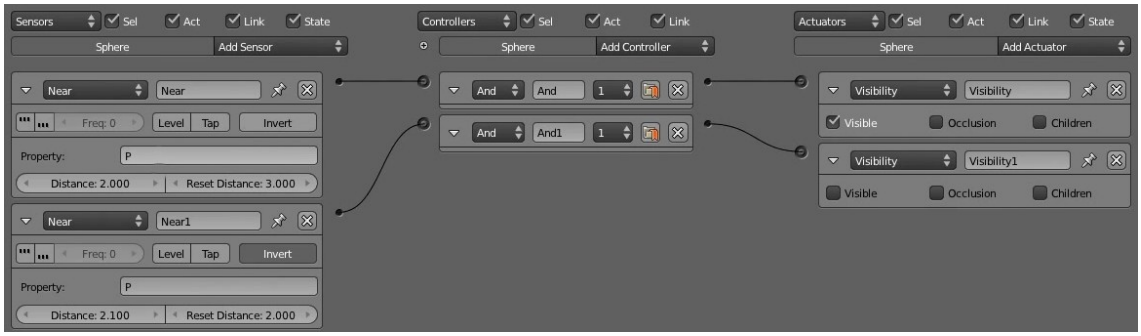
Nel nostro caso, vogliamo che la sfera sia inizialmente invisibile, per apparire solo quando le si avvicina la telecamera; l'idea è quella di creare pannelli informativi o altri elementi da affiancare agli oggetti nel museo 3D e farli apparire solo quando la telecamera è in zona.



Le impostazioni di visibilità nella scheda Physics di un oggetto

Per far ciò, dobbiamo selezionare la casella *Invisibile* nella scheda *Physics* della sfera.

Per gestire l'apparizione e la sparizione dell'oggetto, spostiamoci nell'editor *Logic* ed inseriamo due sensori *Near* (lett.: accanto, vicino, prossimo), due controller *AND* e due attuatori *Visibility*.



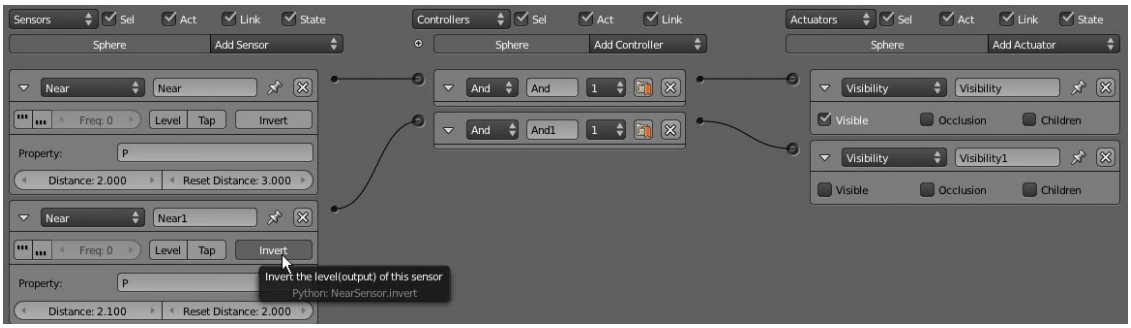
I sensori NEAR, i controller AND e gli attuatori VISIBILITY

Il primo sensore *Near* si attiverà quando un oggetto dotato di una *Property* specificata nell'apposito campo (da lasciare bianco per far attivare il sensore da qualsiasi oggetto, ma non è il nostro caso) si avvicinerà entro distanza *Distance*.

Anticipiamo la creazione della *Property* per l'oggetto che dovrà attivare il sensore scrivendo il nome della *Property*, nel mio caso "*P*", nel campo.

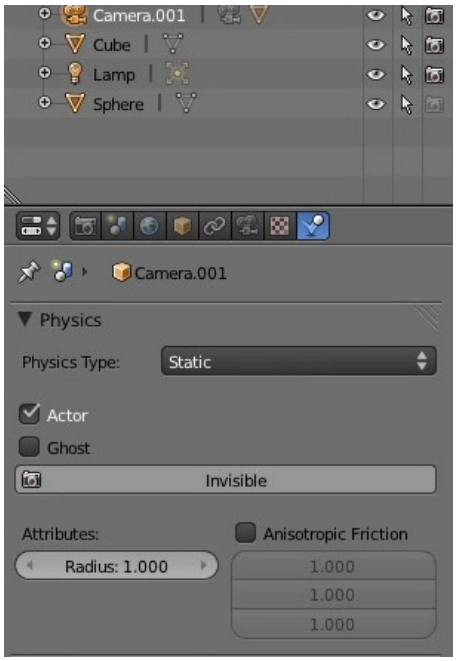
Specificare la Property "P" (non ancora creata, in effetti) nel campo Property di NEAR

Colleghiamo il sensore al controller *AND* e tale controller al primo attuatore *Visibility*, con la casella visibilità selezionata: in questo modo, quando un oggetto con la proprietà "*P*" si avvicinerà alla sfera, la renderà visibile... ma la lascerà tale allontanandosi, quindi utilizziamo il secondo sensore *Near* in modalità *Invert* con *Distance* leggermente maggiore di quella impostata per il primo sensore e colleghiamolo al secondo attuatore *Visibility*, che avrà la casella visibilità deselezionata.



Sensori, controller e attuatori impostati e collegati

A questo punto potremmo pensare che sia sufficiente muovere la telecamera spostandola vicino alla sfera per farla apparire... nel mio caso, c'è una telecamera posizionata con l'asse Z negativo locale verso la sfera, dotata di una *Property Float* di nome "*P*" e di attuatori di movimento che le consentono di spostarsi avanti e indietro mediante i tasti *W* e *S* (tra parentesi: *Near* funziona in tutte le direzioni, sto usando solo l'asse Z perché per questo esempio è sufficiente)... ed ecco la prima sorpresa: avvicinando la telecamera alla sfera, non succede niente!

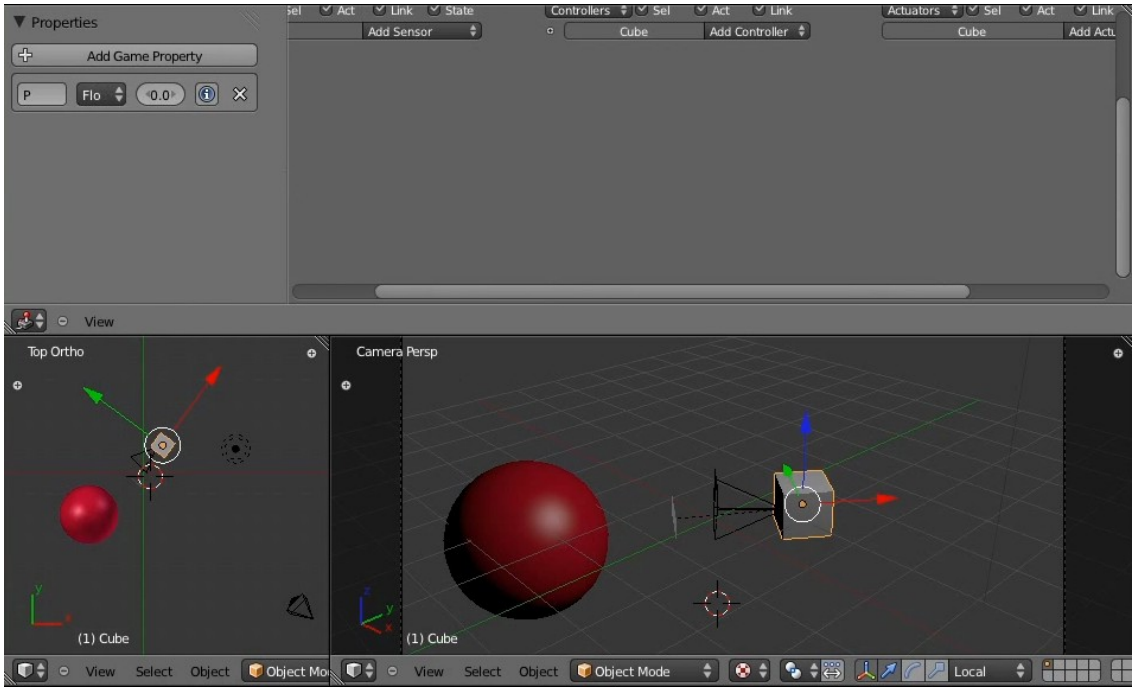


Render "Actor" la telecamera

Le specifiche di *Near* dicono che l'attore che deve attivare il sensore dev'essere, appunto, un *Actor*, per cui selezioniamo tale casella nella scheda *Physics* della telecamera e riproviamo... ancora una volta, non succede niente!

Il motivo è che l'*Actor* che deve attivare *Near* non dev'essere, per così dire, "inconsistente", per cui non possiamo utilizzare la telecamera o una *Empty*: ci serve un oggetto, ed ecco spiegata la presenza del cubo (sto usando un cubo ma potrei usare anche un singolo vertice ottenuto ad esempio da un *Plane* o da un'altra mesh).

Dotiamo tale cubo di una *Property Float* con nome "*P*" e rendiamolo figlio della telecamera con una relazione di parentela, per cui prima selezioniamo il cubo, poi (con una selezione multipla, *SHIFT* e click destro del mouse) la telecamera, premiamo *CTRL P* e confermiamo l'operazione di parenting nel menù che apparirà a video.



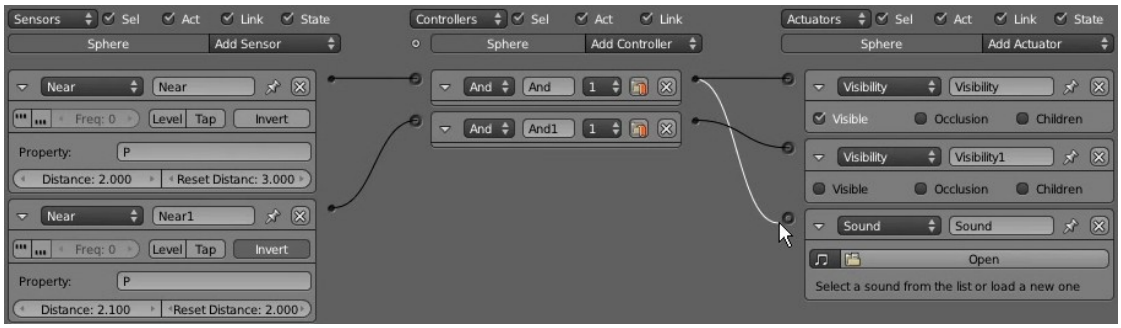
Il cubo, provvisto di una Property "P", reso figlio della telecamera virtuale

Ovviamente, il cubo (o in generale la mesh) deve avere la casella *Actor* selezionata, nella scheda *Physic*. Muovendo la telecamera con *W* e *S*, muoveremo anche la mesh figlia, che attiverà i sensori *Near* avvicinandosi o allontanandosi dalla sfera.

La presenza di una mesh imparentata alla telecamera può andare bene in uno sparatutto, dove vogliamo visualizzare l'arma ed anzi farla "reagire" ai diversi tipi di illuminazione presenti nella scena, ma per il museo virtuale questa soluzione non va bene.

Il problema viene risolto selezionando la casella *Invisibile* del cubo nella scheda *Physics* dello stesso, rendendo appunto tale oggetto invisibile in fase di gioco.

Così come abbiamo implementato l'apparizione di un elemento, possiamo generare altri eventi in seguito all'attivazione di *Near*, ad esempio far eseguire un file audio contenente una descrizione dell'oggetto; per far ciò, dal controller *AND* associato al primo sensore *Near* (quello di attivazione entro una certa distanza) facciamo uscire un collegamento diretto ad un attuatore *Sound*, nel quale caricheremo il file audio da riprodurre e selezioneremo la modalità di riproduzione.

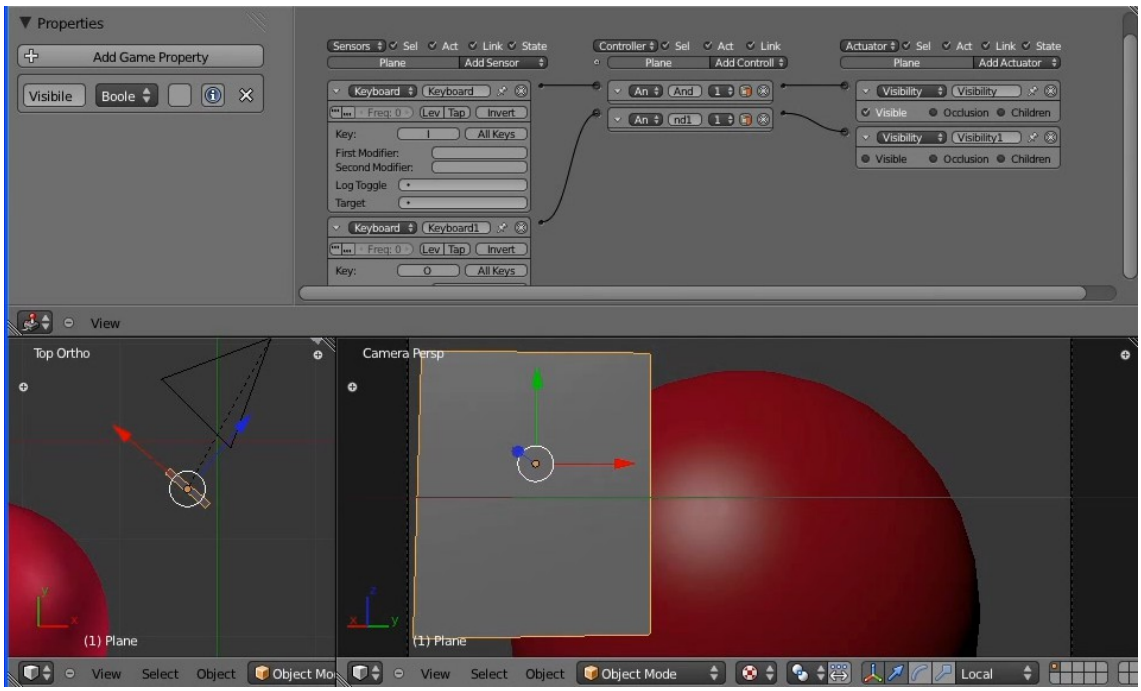


Utilizzo dell'attuatore SOUND

A proposito di quest'ultimo punto, le opzioni che ci interessano sono due: *Play End* e *Play Stop*.

Play End, quando attivato, esegue il file audio una volta fino alla fine del pezzo, anche se il sensore che l'ha attivato nel frattempo non è più attivo, ossia se la telecamera nel frattempo si è allontanata dalla sfera; *Play Stop*, di contro, esegue il file audio solo quando il sensore di attivazione resta attivo, per cui nel nostro caso la riproduzione si interromperà bruscamente allontanando la telecamera dalla sfera.

Nella scena è presente anche un Plane posto di fronte alla telecamera, sulla sinistra: rappresenta un possibile pannello informativo "generico", ad esempio con la descrizione dell'ambiente e dei tasti da utilizzare; tale pannello è inizialmente invisibile (sempre mediante *Invisibile* in *Physics*) ma può essere attivato o disattivato mediante la pressione dei tasti *I* e *O*; così come abbiamo imparentato il cubo alla telecamera, possiamo imparentare il pannello in questa posizione relativa rispetto alla telecamera, così la seguirà sempre e in qualsiasi punto della scena l'utente potrà visualizzare tale pannello, che magari avrà per Texture un testo con trasparenza o altro, premendo *I*.



Il Plane "informativo" posto di fronte alla telecamera

Il file *.blend* della scena appena vista, senza relazione di parentela tra il cubo e la telecamera e senza attuatori *Sound*, è scaricabile gratuitamente, per poterlo analizzare o personalizzare con calma, all'indirizzo *www.redbaron85.com/museo3dnear.zip* .

[NOTA: la serie del "tour virtuale" non è stata completata; non ci sono altre puntate]

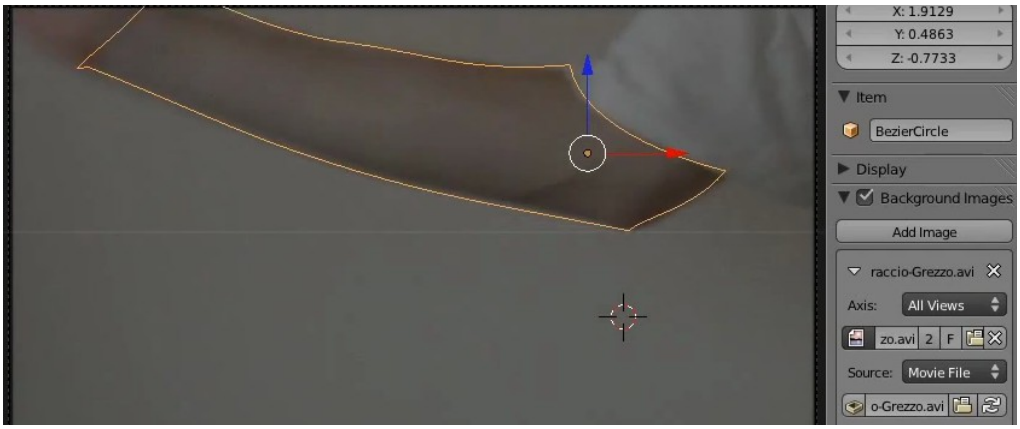
* * *

Noise sulle curve IPO per campi stellati animati ed altro

In questo tutorial vedremo come generare sequenze di *matte images*, ovvero immagini in scala di grigi utilizzate in fase di compositing per miscelare elementi digitali o reali in un'unica immagine, mediante *RotoBezier* in Blender 3D 2.5.
Il tutorial, realizzato con la versione 2.58 stabile di Blender, si divide in tre parti: la generazione della *RotoBezier*, la creazione della sequenza di *matte images* e il compositing finale.

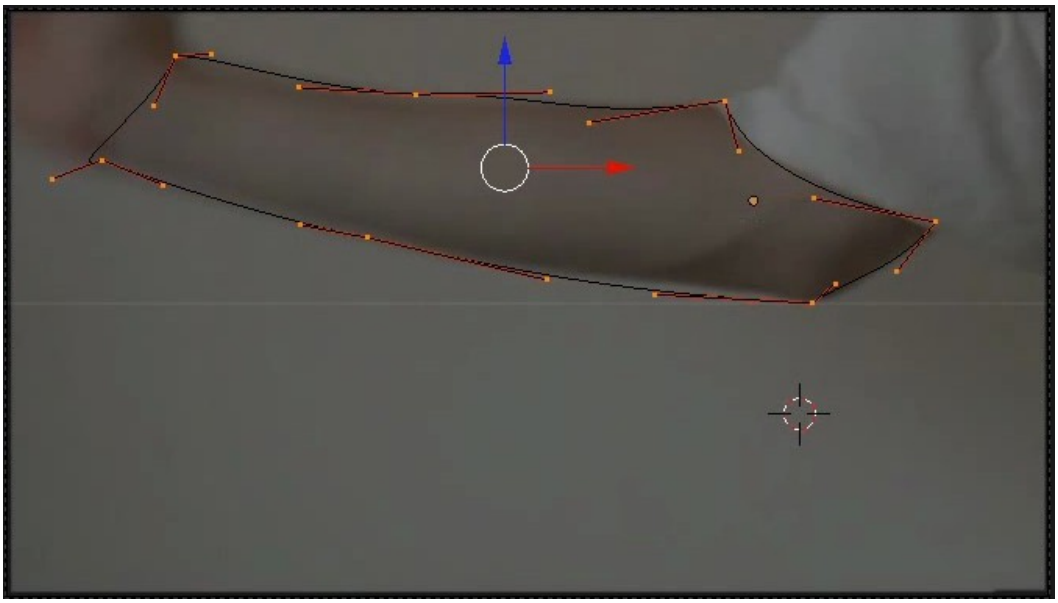
Gli argomenti non verranno trattati in dettaglio, o meglio: non mostrerò passo per passo le varie fasi, fornendo invece una panoramica delle operazioni; chi volesse approfondire l'argomento *Nodes Compositing*, inoltre, può dare un'occhiata al relativo ebook, sul mio sito web.
Iniziamo dalla prima fase: l'animazione della *RotoBezier*.

Il filmato utilizzato per l'esempio mostra un braccio che si muove ad una certa distanza da una parete. Supponiamo di voler "proiettare", per così dire, un logo, un modello 3D o un'animazione sulla parete: dobbiamo trovare il modo di oscurare tale immagine al passaggio del braccio... Blender può aiutarci nella creazione della cosiddetta maschera matte, un'immagine che identificherà il movimento del braccio, come vedremo.



Screenshot del filmato utilizzato per questo tutorial

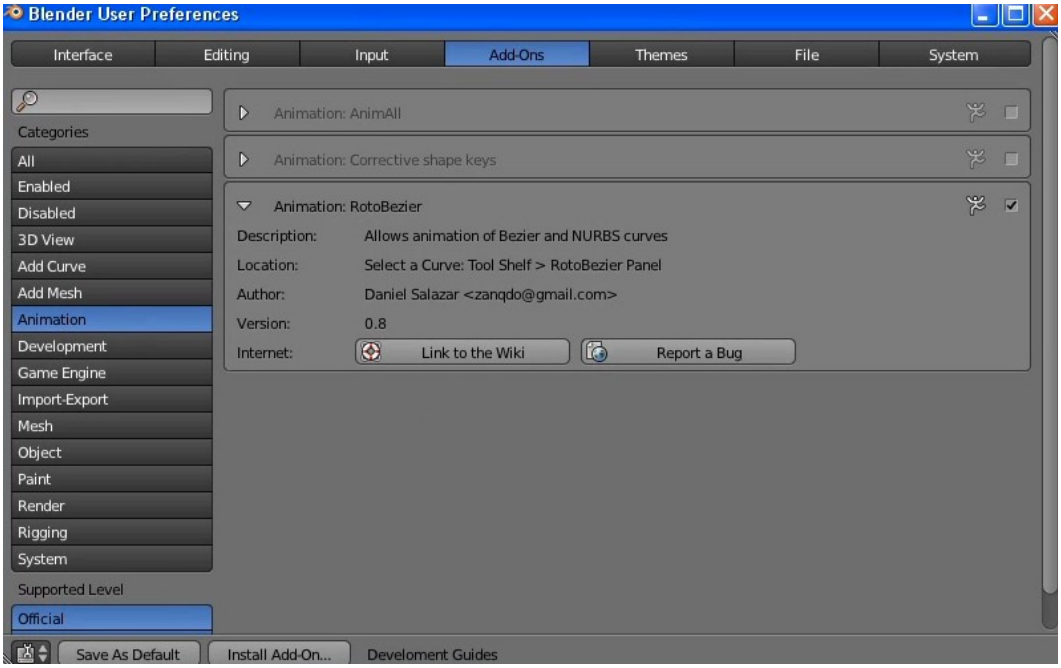
Quello che dobbiamo fare è caricare il filmato con il movimento del braccio come *Background Image* della vista telecamera, facendo attenzione a sincronizzare frame rate, durata del filmato e risoluzione. Aggiungiamo un *Bezier Circle*, quindi una curva chiusa, alla scena, posizionando tale oggetto in modo da vederlo frontalmente, infatti rappresenterà la sagoma del braccio; cambiamo la modalità in 2D per ottenere una forma chiusa, estrudiamolo un po' (altrimenti non verrà renderizzato) con la voce *Extrude* della scheda *Object Data*, nella *Properties Window*, e, se lo preferiamo, cambiamo la modalità di visualizzazione della *3D View* in *Wireframe*.



La modellazione della curva (o cerchio) Bezier, seguendo l'immagine di riferimento

È consigliabile, per questa operazione, cambiare la modalità *Persp* della telecamera in *Ortho*, mediante il relativo pulsante nella scheda *Object Data* della telecamera, all'interno della *Properties Window*. Il filmato NON deve essere utilizzato in fase di rendering: ci serve solo da riferimento in questa fase per animare la *Bezier* che rappresenta il braccio, quindi non c'è bisogno di impostarlo come *Texture* di *World*; anzi: come vedremo, *World* dev'essere nero e la *Bezier Shadeless*!

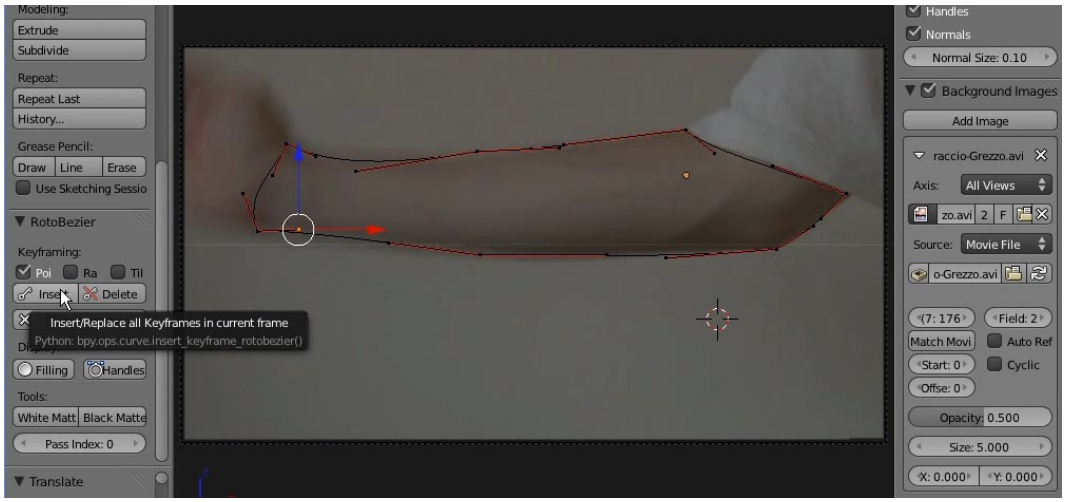
Il plugin *RotoBezier* è disponibile di default con Blender 2.58 ma è disattivato, per cui dobbiamo aprire la scheda *User Preferences*, pannello *Add-Ons*, e selezionare *RotoBezier* nella sezione *Animation*. I comandi del plugin appariranno nelle *Tool Shelf* delle *3D Views*.



Attivare l'Add-On delle Rotobezier nella finestra User Preferences

Selezioniamo la nostra *Bezier* e passiamo in Edit Mode.

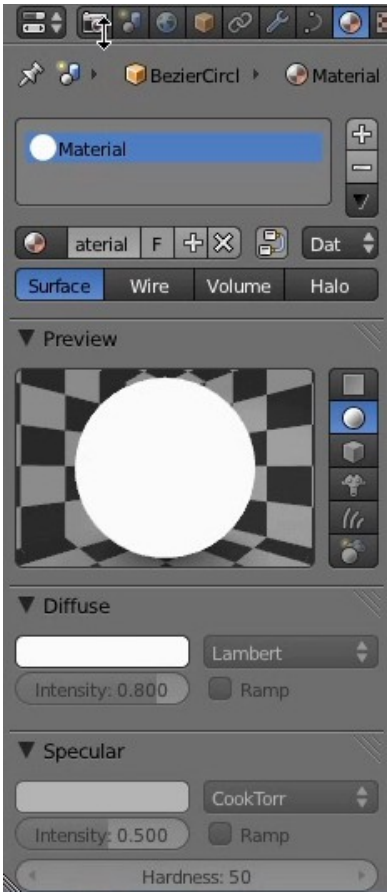
Adesso dobbiamo deformare, nel corso dell'animazione, la nostra Bezier, che suddivideremo un certo numero di volte posizionando i vari punti di controllo in altrettanti punti di riferimento dell'oggetto da tracciare; ovviamente, inizieremo individuando dei fotogrammi chiave, ad esempio i fotogrammi nei quali il braccio si ferma: dovremo quindi spostare i punti di controllo per seguire il braccio, selezionarli ed inserire un keyframe per la sagoma con il relativo pulsante nella sezione *RotoBezier* della *Tool Shelf*.



Esempio di inserimento di Keyframe per la Rotobezier

Altro esempio di deformazione e inserimento di Keyframe per la Rotobezier

Purtroppo, non basta individuare i keyframes e lasciare a Blender il compito di effettuare l'interpolazione: i movimenti reali non sono mai lineari, ad esempio perché il braccio rallenta o accelera durante l'animazione, né le forme restano immutate, visto che nel 3D possono esserci anche spostamenti in profondità, cambi di prospettiva o piccole rotazioni che sembrano allungare, restringere e in generale deformare gli elementi, per cui (spostandoci tra un frame e l'altro con le frecce) dovremo individuare alcuni fotogrammi intermedi dove la forma della *Bezier* si discosta eccessivamente dall'immagine del braccio e deformare la curva, inserendo altri keyframes.



Cambiare il Material della Bezier (leggermente estrusa) in bianco puro Shadeless

Terminato questo lavoro, dobbiamo renderizzare la *Bezier*.

Le *Matte Images* sono in scala di grigi, con il bianco che rappresenta la parte di maschera da considerare per coprire l'immagine sottostante, i grigi elementi semitrasparenti (ad esempio la sfocatura dovuta al movimento, o *Motion Blur*) e il nero le parti da non considerare; cambiamo quindi il colore di sfondo dell'universo virtuale, *World Horizon*, in nero puro, ed

impostiamo il Material della *Bezier* (leggermente estrusa, per renderla renderizzabile) in bianco puro e *Shadeless*.

Nella scheda *Render*, impostiamo il *Motion Blurring* per implementare anche un po' di sfocatura da movimento nel corso dell'animazione (una possibile alternativa è costituita dall'utilizzo del Nodo *Blur* in fase di Compositing, come descritto nell'*ebook sul Compositing dei Nodes in Blender 3D*).

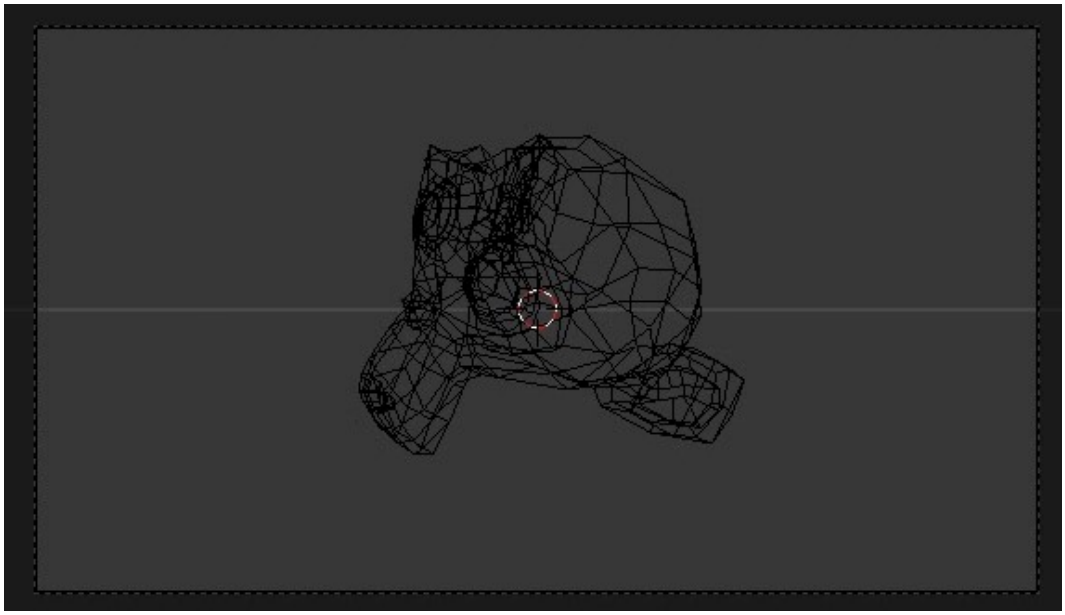
Motion Blur

Avviamo quindi il rendering per ottenere l'animazione della maschera *Matte*.

Screenshot dell'animazione ottenuta

A questo punto apriamo il *Nodes Editor* per realizzare il compositing.

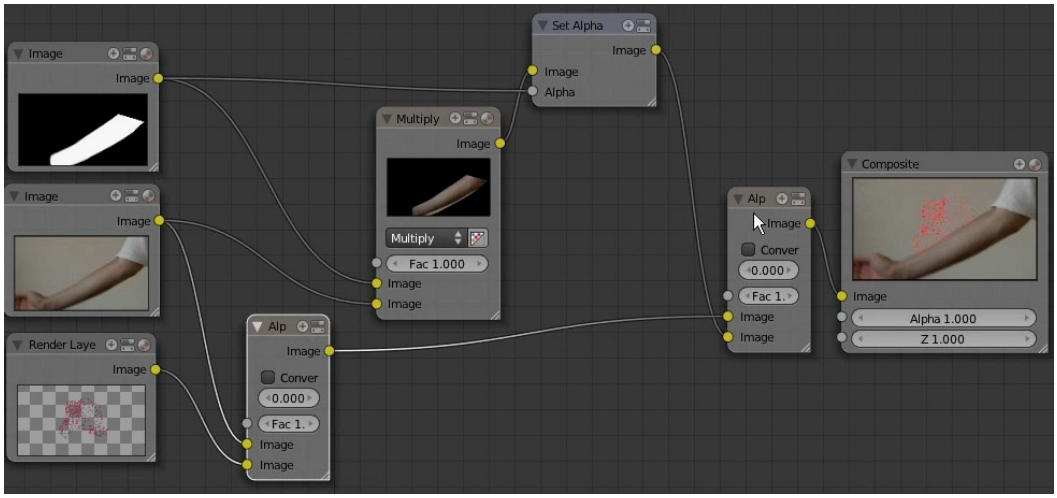
Come immagine da proiettare sulla parete sto utilizzando, per questo esempio, il rendering (fatto in tempo reale nella scena di Blender) di *Suzanne* in *Wireframe*... è solo un esempio.



La nuova scena, con Suzanne (giusto per fare un esempio)

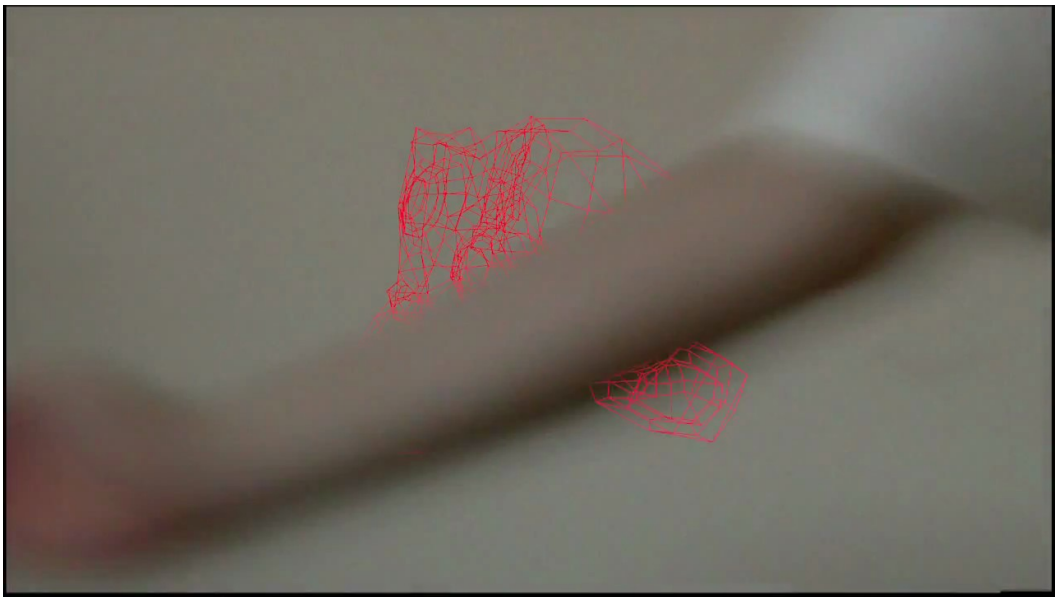
Configuriamo l'Editor come segue: moltiplichiamo la maschera per l'animazione del braccio, in modo da ottenere solo quest'ultimo e sovrapporlo, in seguito, all'animazione con *Suzanne*; dobbiamo però fornire un canale *Alpha* a tale elemento, per cui utilizziamo un Nodo *Set Alpha* passandogli come immagine il braccio isolato e, come *Alpha*, la maschera *Matte*.

Applichiamo il rendering di Suzanne sull'animazione del braccio con un semplice *AlphaOver*, dopodiché prendiamo l'output di quest'ultimo nodo e del *Set Alpha* precedentemente visto e, tramite un secondo *Alpha Over*, ricaviamo il rendering finale.



Schemi di Nodi di Compositing

Non ho realizzato la maschera *Matte* con molta cura, come si può notare osservando soprattutto la parte inferiore del braccio nel rendering finale, ma il principio è questo, per il resto si tratta di avere tempo e pazienza per ricavare una buona sagoma *Bezier* ed implementare un buon *blurring*.



Screenshot del risultato finale (renderizzato con Compositing)

* * *

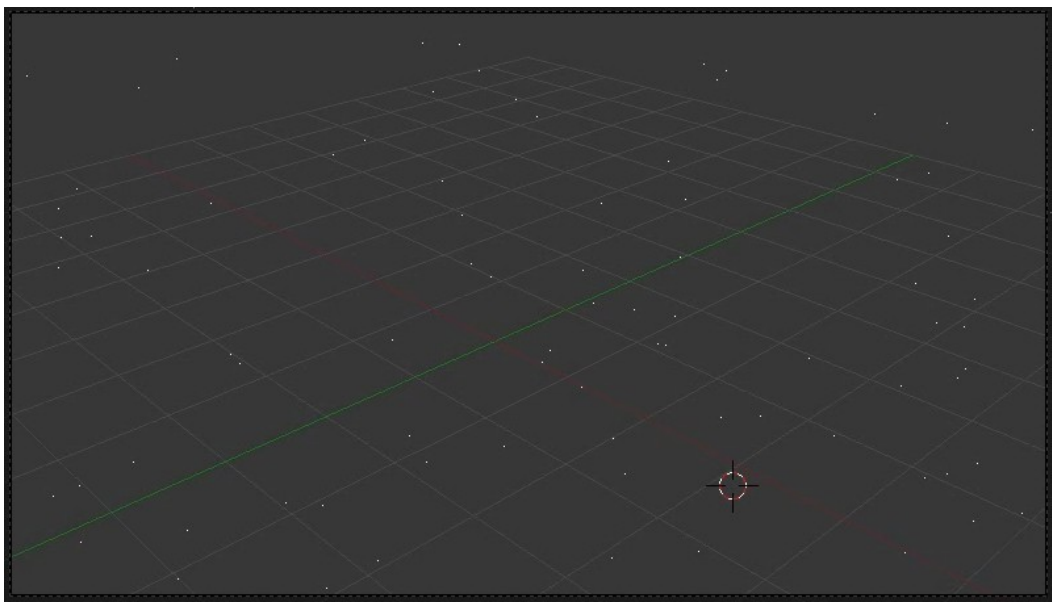
Noise sulle curve IPO per campi stellati animati ed altro

In questo tutorial vedremo come applicare il modificatore *Noise* alla curva *IPO Size* di *Star* in Blender 3D 2.5 per ottenere un campo stellato animato; recentemente infatti mi è stato chiesto di realizzare questo fondale per un'animazione e l'utilizzo di *Noise* mi è sembrato un buon metodo per animare in maniera casuale le stelle.

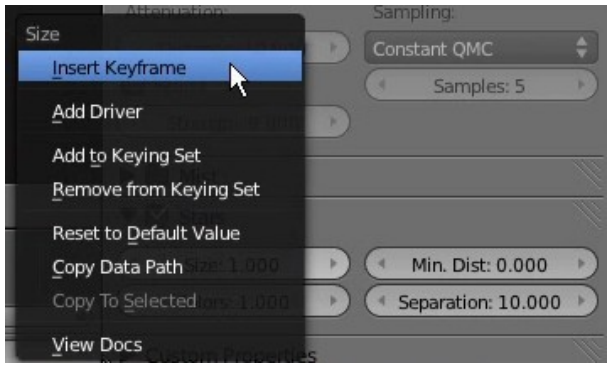
La sezione Stars di World, nella Properties Window

Rendiamo nero lo sfondo dell'universo virtuale impostando tale colore nel campo *Horizon Color*, dopodiché attiviamo *Star*, sempre nella scheda *World*, all'interno della *Properties Window*.

In *Stars*, *Separation* indica la distanza media tra una stella e l'altra ed è, quindi, una misura della densità del campo stellato; impostiamo, ad esempio, 10, per avere un buon numero di stelle.
Size indica la grandezza media degli elementi creati, Blender poi renderà più o meno grandi le stelle in maniera casuale; impostiamo, ad esempio, 1. Portiamo a 1 anche *Color*, per colorare le stelle.

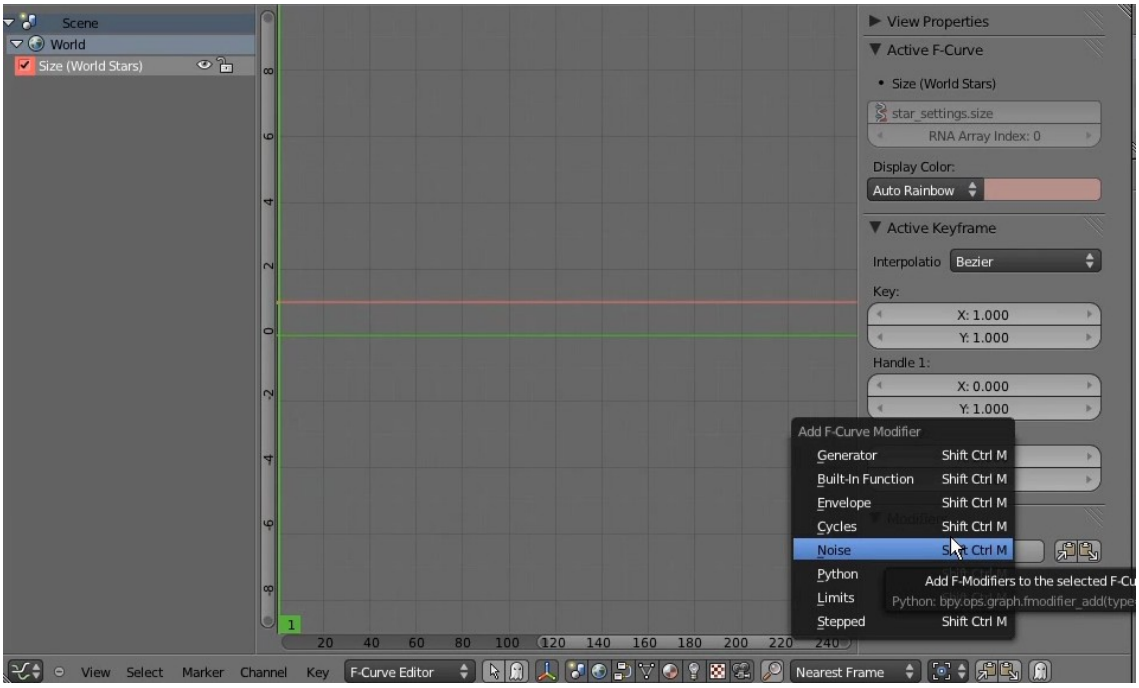

La Preview del campo stellato in una 3D View

Animeremo il pulsante *Size*: la variazione di dimensione verrà percepita come una variazione di luminosità. Inseriamo quindi un keyframe sul campo *Size* premendo il tasto *I* mentre il cursore del mouse è su tale voce, oppure con click destro e "*Insert KeyFrame*": il keyframe iniziale ci serve solo per generare la curva *IPO*, alla quale applicheremo il modificatore *Noise*.



Inserimento di un Keyframe (ce ne serve solo uno, in questo caso, giusto per creare la curva IPO) per il campo SIZE di STARS

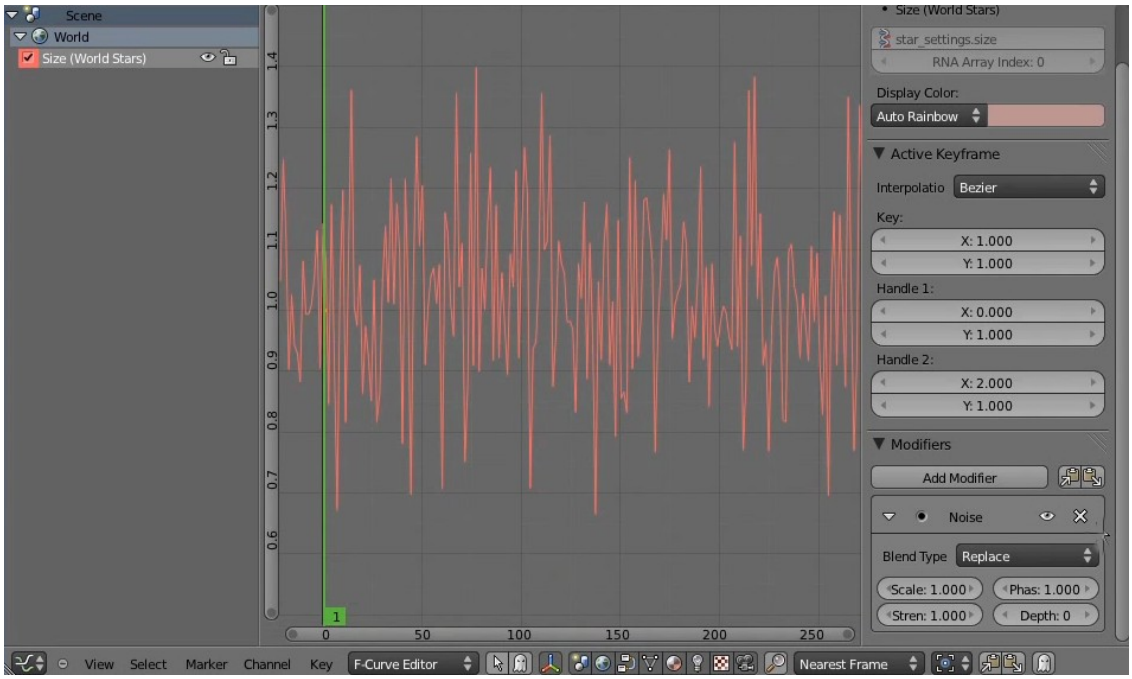
Spostiamoci quindi nel *Graph Editor* e selezioniamo la *curva IPO* appena creata per *World Stars Size*, dopodiché apriamo la *Transform Window* di *Graph Editor* con *N* ed inseriamo, in basso, un modificatore *Noise*.



Aggiungere un modificatore IPO NOISE alla curva IPO del campo SIZE di STARS

Vediamo come si comporta *Noise*: esso altera la forma della curva sottostante, con un'ampiezza massima della deformazione (rispetto all'andamento originale) definita nel campo *Strength*, mentre con *Scale* possiamo dilatare o restringere la curva, il che si traduce in variazioni più o meno veloci; inoltre, con *Depth* possiamo aumentare il grado di suddivisioni tra un punto e l'altro, per realizzare interpolazioni meno brusche. Possiamo ignorare il campo *Phase*.

Per accentuare l'effetto, rendendolo davvero esagerato per osservarlo meglio, impostiamo, ad esempio, *Scale* 3 con *Strength* 3 e *Depth* 1 ed avviamo il rendering.



La scheda del modificatore IPO NOISE

Noise può essere utilizzato per aggiungere del rumore a... praticamente tutto ciò che è animabile in Blender: possiamo variare il grado di distorsione di una Texture *Distorted Noise*; l'aspetto di un oggetto *Halo* con effetti *Lines*, *Stars* e *Flares*; la luminosità di una fonte di luce (ad esempio per realizzare un'illuminazione a lume di candela); la posizione della telecamera (ad esempio per una webcam che osserva una città durante un terremoto) ed altro ancora.

L'esempio visto è davvero la punta dell'iceberg: ho voluto spiegarvi i significati di *Strength*, *Size* e *Depth* proprio per consentirvi di variare tali parametri con criterio e provare altri effetti.

* * *

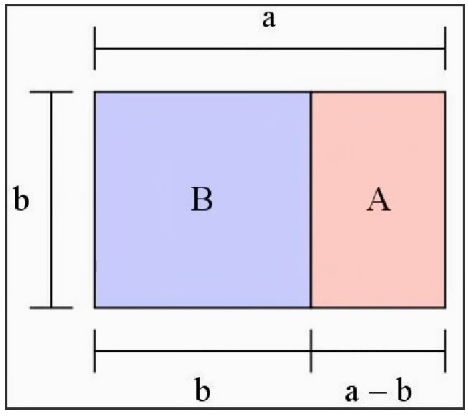
Rapporti, rettangoli e triangoli aurei e le Composition Guides

In questo tutorial vedremo come utilizzare la funzionalità "*Composition Guides*" della telecamera di Blender 3D 2.58, una voce che ci viene in aiuto nel migliorare la composizione della scena; ne approfitterò per dire, anche se molto velocemente e senza pretese di completezza, qualche parolina sulle linee guida della composizione fotografica.

Si tratta, in effetti, di tecniche in arrivo dal mondo della fotografia e prese in prestito dalla Computer Grafica, visto che alla fin fine i rendering statici sono delle immagini.

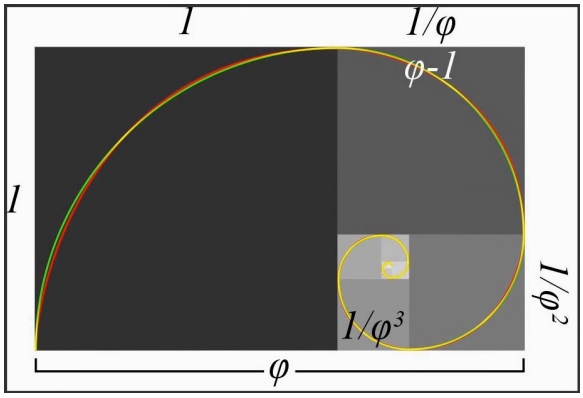
Iniziamo dalla teoria.
Contrariamente a quanto si possa pensare, il punto più interessante di un'immagine non è il centro della stessa; in realtà, è stato dimostrato che ci sono più punti di interesse, ben localizzati, secondo una regola detta "*regola dei terzi*": si suddivide l'immagine in 9 aree, tracciando due linee verticali e due orizzontali equidistanti, individuando 4 "nodi" o "punti di interesse" all'interno della griglia così costruita, cercando di porre i punti salienti dell'immagine in prossimità di tali nodi.

Non solo: le linee orizzontali e verticali possono essere utilizzate come linee guida per porre oggetti con tali forme; ad esempio, l'orizzonte può essere posto lungo la linea inferiore o superiore, per mettere in risalto il cielo o il mare, creando un effetto decisamente più interessante di quello ottenuto piazzando la linea dell'orizzonte esattamente a metà; idem per una parete, un palo, una colonna o altri elementi verticali, che possono essere messi in prossimità di una delle due linee verticali, ed ovviamente le cosiddette "cornici naturali" possono essere piazzate seguendo, ad esempio, la linea verticale destra e la linea orizzontale superiore, e così via.



Rettangolo aureo

La regola dei terzi non è l'unica, anzi: partendo dal concetto di *sezione aurea* è possibile creare dei rettangoli dove il rapporto tra il lato maggiore e il minore è uguale a quello ottenuto tra il lato minore e la differenza tra il lato maggiore e il minore.



Spirale di Fibonacci

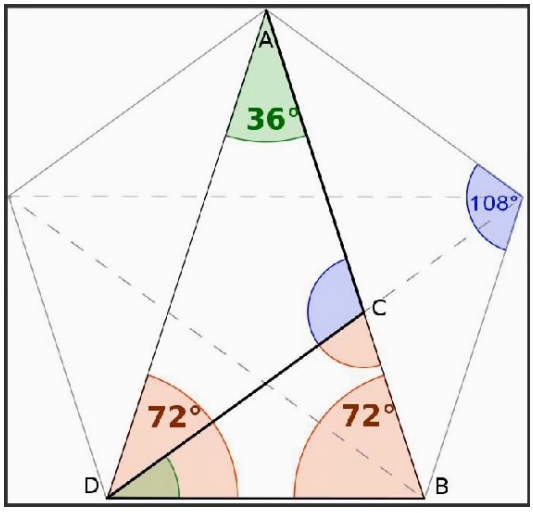
Tali rettangoli sono detti rettangoli aurei e la cosa più interessante è che la regola può essere applicata ai sottorettangoli generati, seguendo la cosiddetta *spirale di Fibonacci*.

Ovviamente, sfruttando questa suddivisione "armoniosa" degli spazi è possibile porre gli elementi in parti dell'immagine con una collocazione che li farà risaltare in maniera particolare, piuttosto che ponendoli, ad esempio, al centro dell'inquadratura.

Una considerazione che vale anche per le altre regole: non preoccupatevi se vi sembra di lasciare spazi “vuoti”, perché contribuisce a “bilanciare” la composizione, fidatevi.

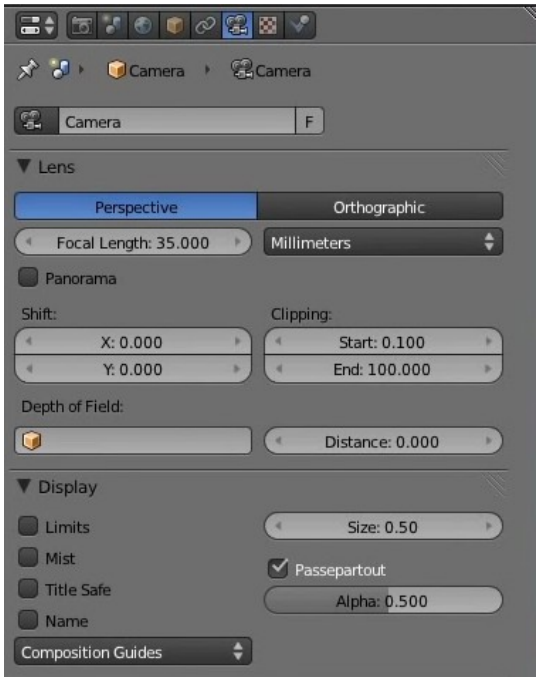
Altra figura geometrica molto utilizzata in ambito fotografico è il *triangolo aureo*, o meglio le possibili suddivisioni degli spazi seguendo le costruzioni dei triangoli aurei.

Tali suddivisioni sono molto interessanti perché introducono linee oblique all'interno dell'inquadratura, ottime se nell'immagine sono presenti elementi diagonali o se dobbiamo confinare degli oggetti in alcuni angoli; ad esempio, una persona seduta a gambe incrociate può essere racchiusa nel triangolo isoscele aureo (che ha i lati uguali in rapporto aureo con la base e ha quindi angoli di 72, 72 e 36 gradi); oppure, se una persona sta raccogliendo un oggetto da terra, questo può essere racchiuso nel triangolo in basso a destra o a sinistra con un'altra configurazione delle diagonali, e così via.



Triangolo aureo

Fatte queste premesse, vediamo quali sono gli strumenti messi a disposizione da Blender 3D 2.58 in tal senso. Nella scena 3D, selezioniamo l'oggetto telecamera e apriamo la scheda *Object Data*, nella *Properties Window*; qui, cerchiamo il menù "*Composition Guides*".

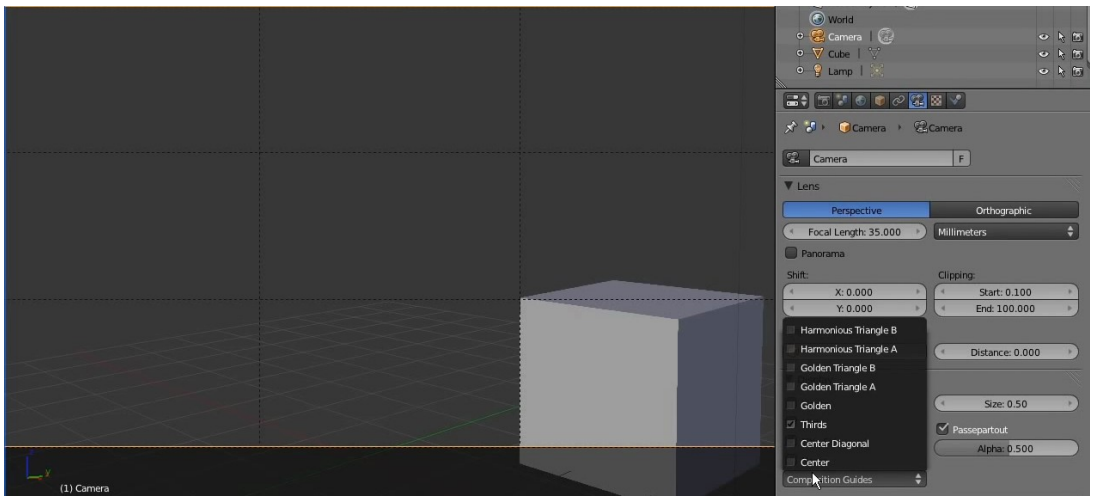


La scheda Object Data di un oggetto telecamera

Aprendo tale menù noteremo che alle voci sono associate delle checkbox (caselle a scelta multipla), quindi è possibile aggiungere più tipi di linee guida all'interno della stessa finestra di anteprima.

Ovviamente le linee guida saranno disponibili solo per la *View* della telecamera selezionata e non verranno renderizzate.

Ecco quindi che *Center* ci consente di individuare il centro dell'inquadratura con un segmento orizzontale e uno verticale, mentre *Center Diagonal* mostra il centro sfruttando le diagonali del frame; *Thirds* implementa la regola dei terzi, permettendoci di individuare i nodi dell'immagine, cosa che avviene anche con *Golden* (regola "aurea"), mentre le restanti voci si riferiscono tutte a configurazioni di triangoli di composizione.



Esempio di Composition Guides (con le relative linee tratteggiate nella 3D View con vista Camera)

Volendo impostare la risoluzione dell'immagine di rendering in modo tale da ottenere un rettangolo aureo, possiamo partire dal valore della risoluzione verticale che vogliamo per l'immagine finale e moltiplicare tale valore Y per 1,618; si tratta comunque di un valore approssimato alla terza cifra decimale, visto che il rapporto o numero aureo è un numero irrazionale.

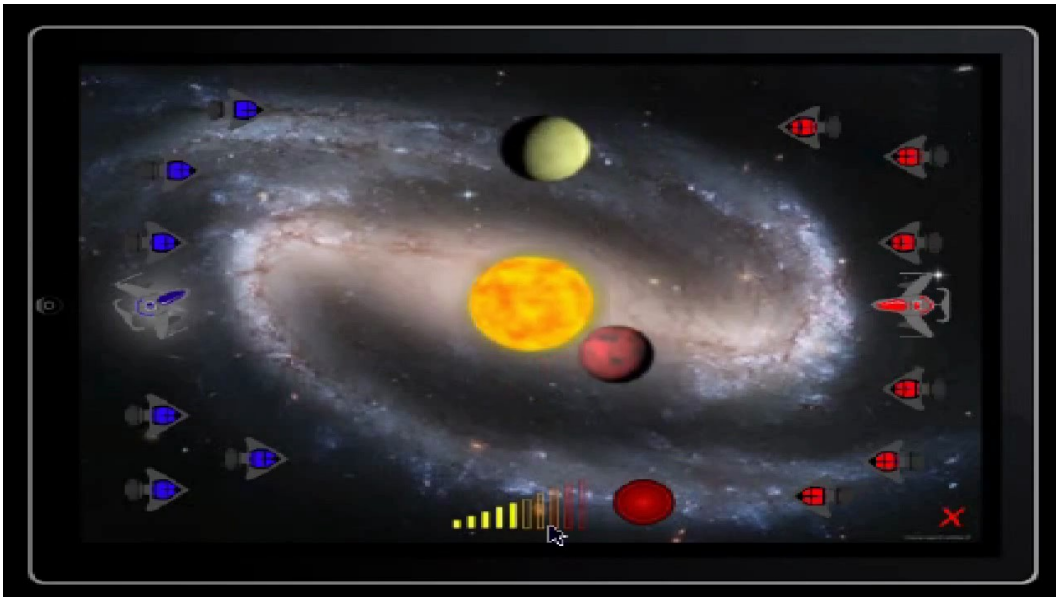
Allo stesso modo, se si parte dalla risoluzione orizzontale X, possiamo ricavare Y moltiplicando X per 0,618, che poi è 1 / 1,618; è sufficiente ricordarsi la sequenza 618 per non sbagliare!

Esempi di possibili... "risoluzioni auree": 1280x791, 1165x720, 1920x1186 e 1747x1080.

* * *

La grafica dell'App "Newton's AG" con Halo, Smoke e Compositing

In questo tutorial parleremo della realizzazione della grafica dell'app "*Newton's AG*", un artillery game sviluppato con la collaborazione di Minix84, disponibile per iPad e iPad2 nell'Apple Store; vedremo, in particolare, come sono state realizzate le animazioni delle stelle, dei pianeti e soprattutto l'esplosione del proiettile mediante il simulatore *Smoke* e il *Compositing dei Nodes*.
Il tutto è stato realizzato con la versione 2.58 stabile di Blender 3D.

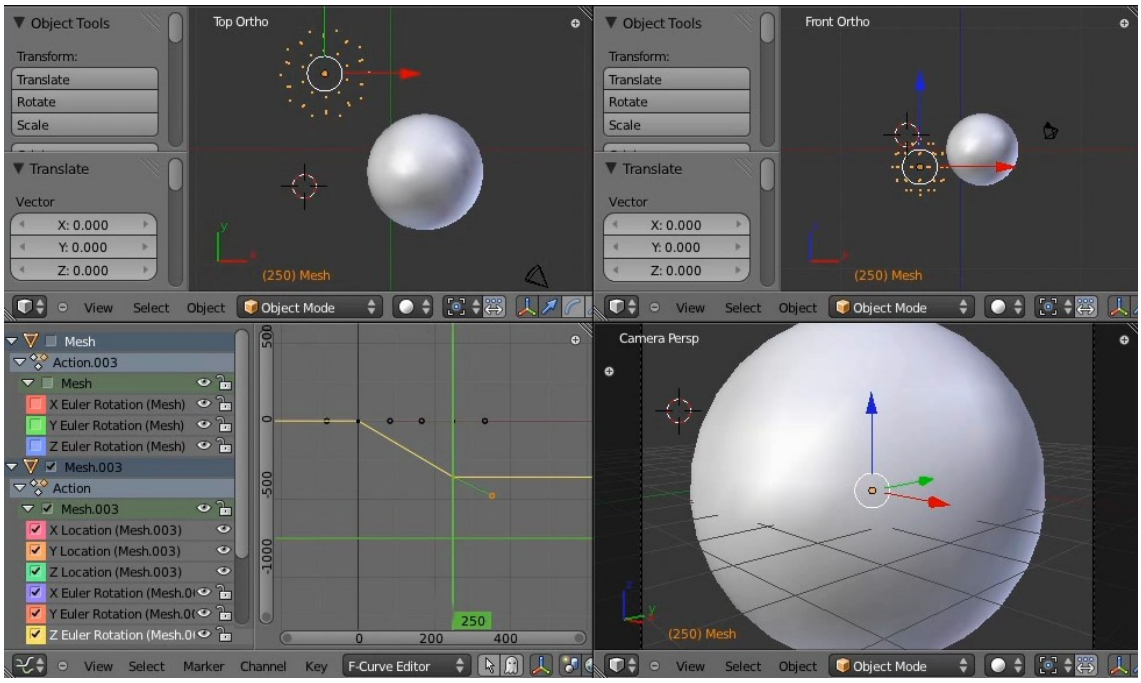


Screenshot dell'App "Newton's AG"; è visibile un'esplosione, realizzata in Blender con Smoke

Nel gioco, due utenti possono ruotare la propria astronave madre, impostare la potenza del tiro e sparare per distruggere le navicelle avversarie; prima di poter distruggere l'astronave madre nemica e vincere la partita, è necessario distruggere le 6 astronavi figlie nemiche.
Nella prima versione del gioco sono presenti 15 livelli con diversi fondali e disposizioni di astronavi, stelle e pianeti; in particolare, i pianeti sono ostacoli in movimento che distruggono il proiettile in caso di impatto, mentre le stelle sono oggetti fermi provvisti però di un campo gravitazionale in grado di deviare (e parecchio, anche) la traiettoria dei proiettili.

Le stelle sono delle semplici sfere con Texture *Clouds* colorata che ruotano intorno all'asse verticale. Per realizzare il bagliore, è stata inserita una icosfera con pochi vertici e Material *Halo*; l'icosfera ha infatti forma irregolare e, ruotando, produrrà un bagliore *halo* "turbolento" intorno alla sfera. Le sequenze animate occupano un bel po' di spazio e appesantiscono le *App*; inizialmente, abbiamo sviluppato le rotazioni in 250 frames, poi ne abbiamo utilizzati solo 125 impostando Step 2 nella scheda *Render*, all'interno della *Properties Window*.

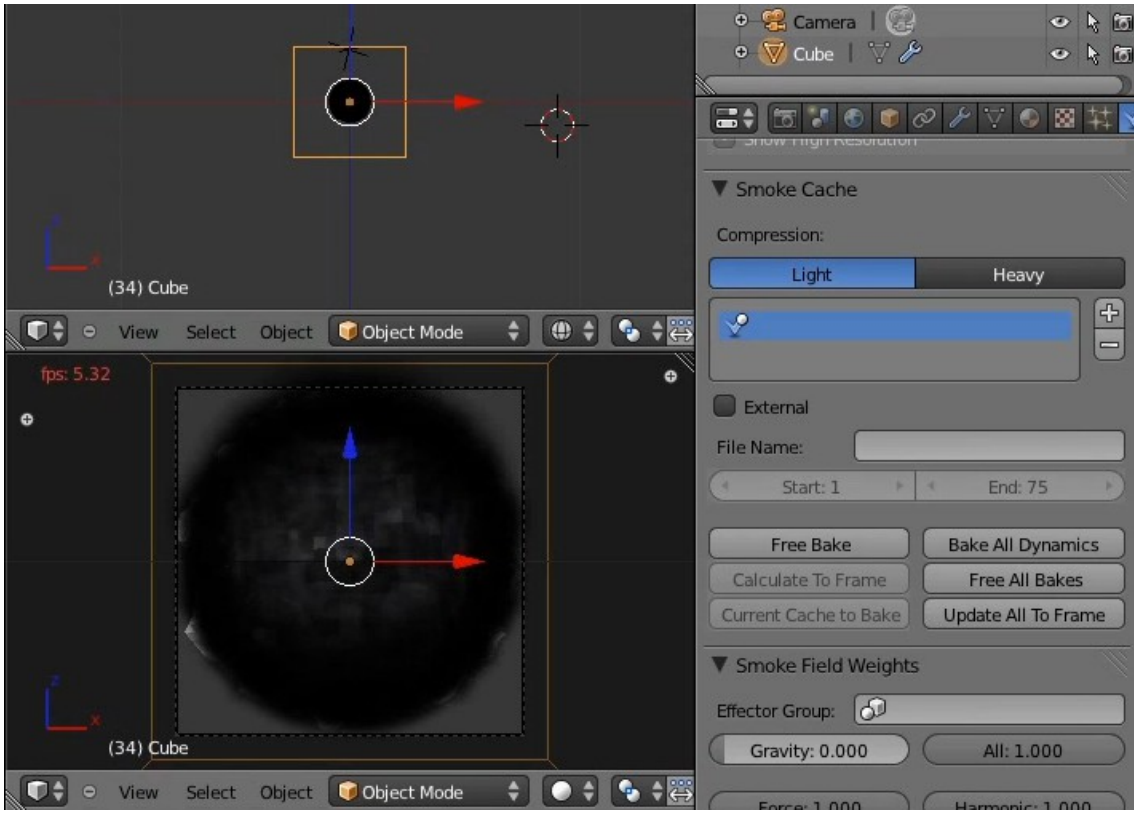
Per renderizzare lo sfondo dell'universo virtuale come trasparente, abbiamo salvato le *PNG* (provviste di canale *Alpha*) in modalità *RGBA* e *Alpha Sky*.



Screenshot delle componenti di una "stella": l'icosfera con bagliore HALO servirà a fornire l'effetto GLOW animato, turbolento

I pianeti sono stati realizzati come le stelle, ossia con Textures Procedurali colorate, tranne per il fatto che non hanno alone *Halo* né *Emit* maggiore di 0, anzi sono ombreggiati da una *Point Lamp*.

La parte più interessante di questo tutorial riguarda, comunque, l'animazione dell'esplosione del proiettile: l'animazione dura appena 75 frames a 30 frame al secondo ma alla fine abbiamo utilizzato solo 0,6 secondi di animazione; il tutto nasce da una sfera provvista di simulatore *Smoke Flow* con impostazioni di default e sistema particellare con 10.000 particelle, *Start* ed *End Frame* a 1, *Lifetime* 50, emissione dai vertici, *Normal Velocity* 0.0, niente fisica e render *None*.



Screenshot della simulazione SMOKE

Il cubo *Domain* utilizzato ha *Dissolve* attivo con valore 5, 64 suddivisioni, *Gravity Field Weigth* 0.0, tutte le altre voci come di default; *Smoke High Resolution* è disattivato.

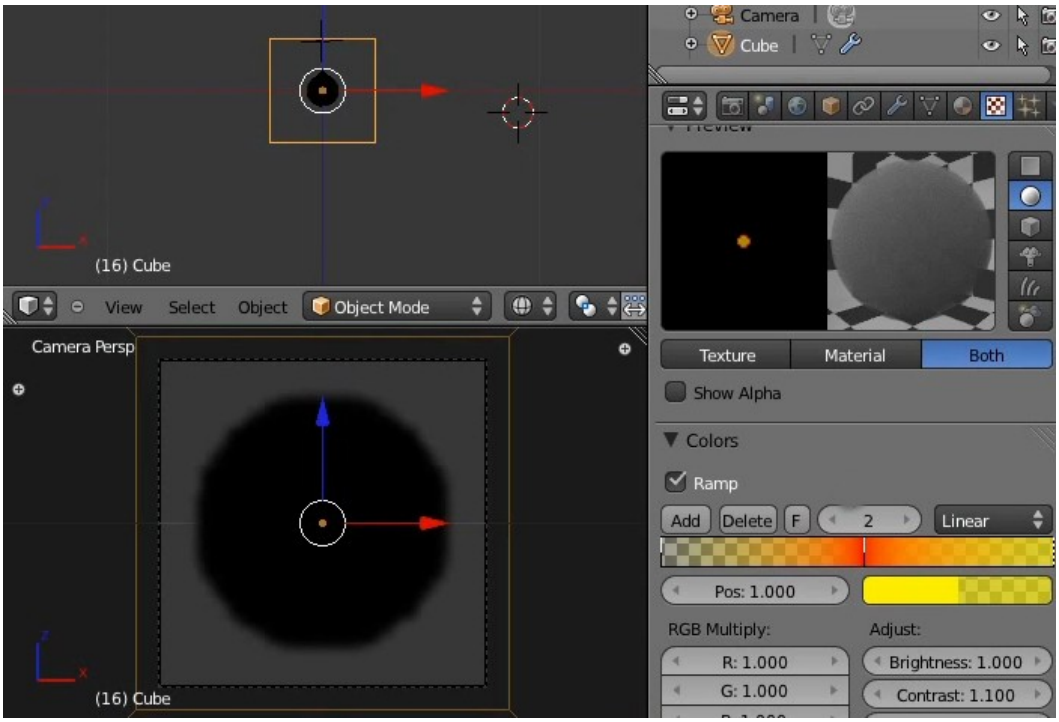
Impostazioni Smoke Domain

Il Material del cubo dominio è chiaramente di tipo *Volume* con *Density*, *Density Scale* e *Scattering* 1.0, mentre *Emission* ha valore 5.0. Al Material sono associate due Textures di tipo *Voxel Data*, entrambe con il cubo stesso come *Domain Object*.

La prima Texture *Voxel Data* ha le impostazioni di default tranne per *Influence Density* 1.0.

La seconda Texture *Voxel Data* agisce invece su *Emission Color* con intensità 1.0 in modalità *Multiply*; qui è

importante la *Color Ramp*, impostata come segue: estremità sinistra, in posizione 0.0 e corrispondente alle zone di densità nulla del fumo, di colore giallo puro con *Alpha* 0.0, quindi completamente trasparente; banda centrale, in posizione 0.5, di colore rosso con *Alpha* 1.0, quindi completamente opaca; estremità destra, in posizione 1.0 e corrispondente alle zone di densità massima del fumo, di colore giallo con valore di opacità *Alpha* 0.6.



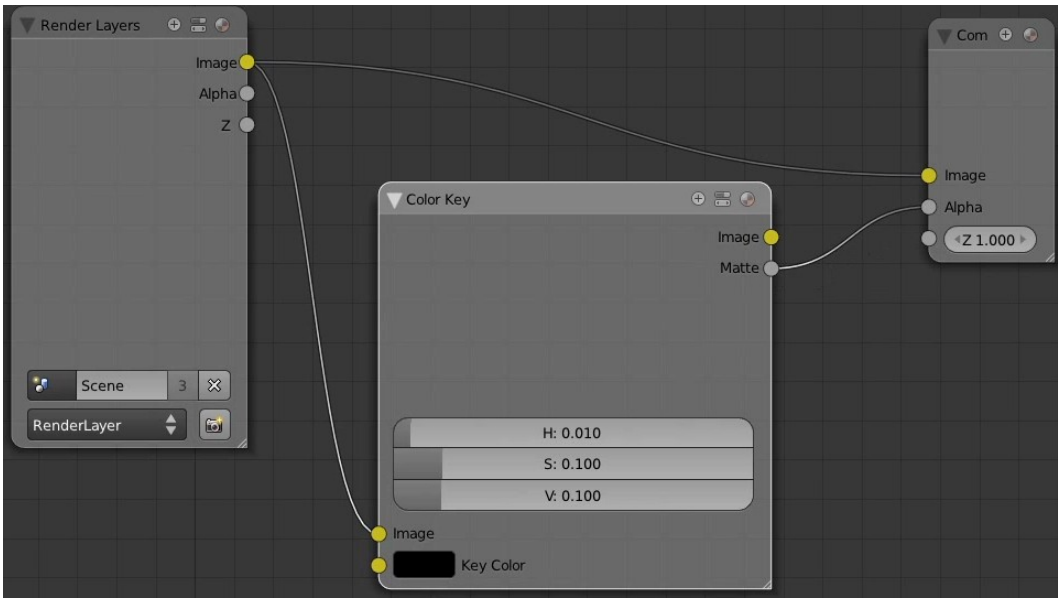
La Color Ramp della seconda Texture Voxel Data, per il Domain

Il rendering è stato fatto ovviamente in *PNG* in modalità *RGBA* con *Alpha Sky*, come con le stelle, solo che in questo caso il nero del fumo veniva renderizzato opaco, per cui a video veniva mostrato un orrendo quadrato nero (il bounding box dell'esplosione)...

Per rendere trasparente questa porzione di immagine, è stato utilizzato il *Nodes Editor* di Blender 3D; per una descrizione approfondita di tale strumento di post-produzione, rimando al mio ebook "*Il Compositing dei Nodes in Blender 3D 2.5*".

In questo caso, per isolare il nero e renderlo trasparente ho aggiunto un Nodo *Color Key* che, presa in input l'immagine in arrivo da *Render Layer*, crea una maschera *Matte Image* rimuovendo dalla

prima immagine il colore specificato nel secondo canale di input, ossia appunto il nero, dopodiché la *Matte Image* così ottenuta viene passata al Nodo *Alpha* di *Composite*, quindi le parti nere verranno renderizzate come trasparenti, il resto opaco, ed ecco la sequenza dell'esplosione.



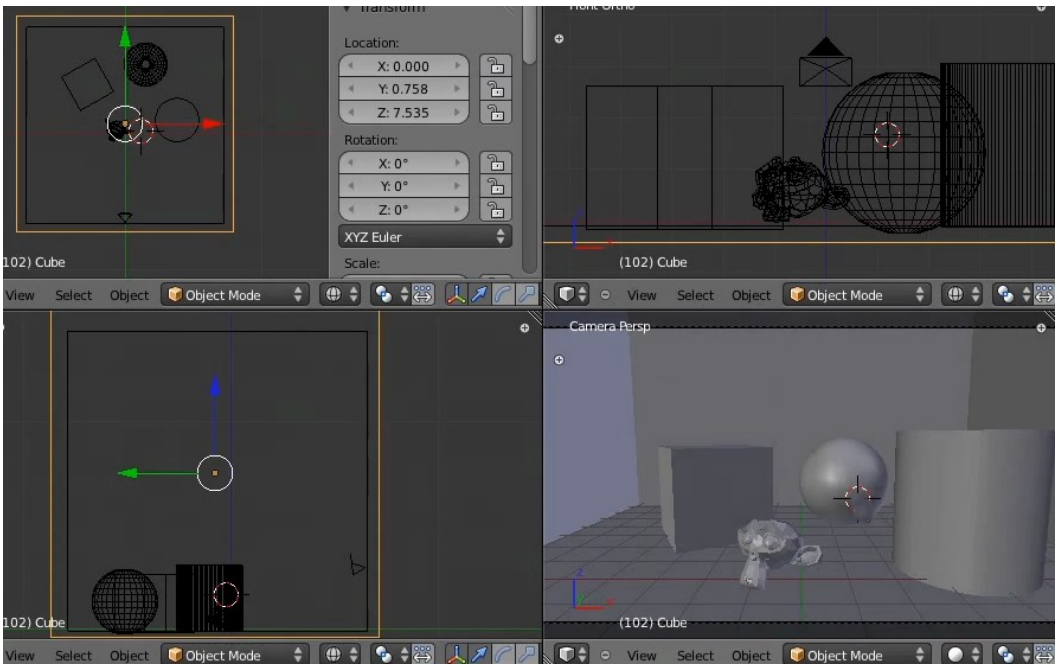
Schema di Nodi di Compositing

* * *

Fuoco e fiamme con Smoke: la color ramp, high resolution e IL

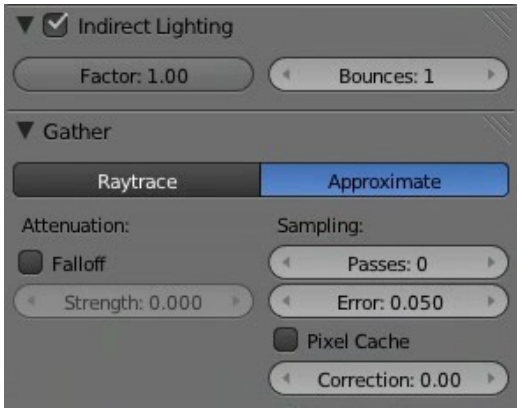
Ecco un altro tutorial sulla realizzazione di fuoco e fiamme con il simulatore *Smoke* di Blender 3D 2.5; questa volta, definiremo meglio l'utilizzo della *Color Ramp* per colorare le fiamme, renderemo lo sviluppo delle stesse più veloce con la differenza di temperatura, analizzeremo l'illuminazione prodotta con *Indirect Lighting* ed infine confronteremo due rendering ottenuti con e senza *Smoke High Resolution*. Il tutorial è stato realizzato con la versione 2.58 stabile di Blender 3D.

La scena utilizzata è composta da una telecamera ed alcuni oggetti posti all'interno di un cubo al quale sono state flippate le *Normali*, rivolte quindi verso l'interno; tra gli oggetti, abbiamo in particolare *Suzanne*, che farà da emettitore, mentre non sono presenti fonti di luce, infatti avviando un rendering al frame 1 vedremo tutto nero. Tutto il set è contenuto all'interno di un altro cubo, che rappresenta invece il dominio della simulazione *Smoke*.



La scena utilizzata per questo tutorial

L'illuminazione deriverà infatti da *Emission* delle fiamme che si svilupperanno da *Suzanne*; *Emission* di per sé non illumina, per cui dobbiamo attivare *Indirect Lighting Approximate*, anche con le impostazioni di base, nella scheda *World*, all'interno della *Properties Window*, proprio come si fa per rendere i classici Materials con *Emit* maggiore di 0.0 dei "veri" emettitori luminosi.

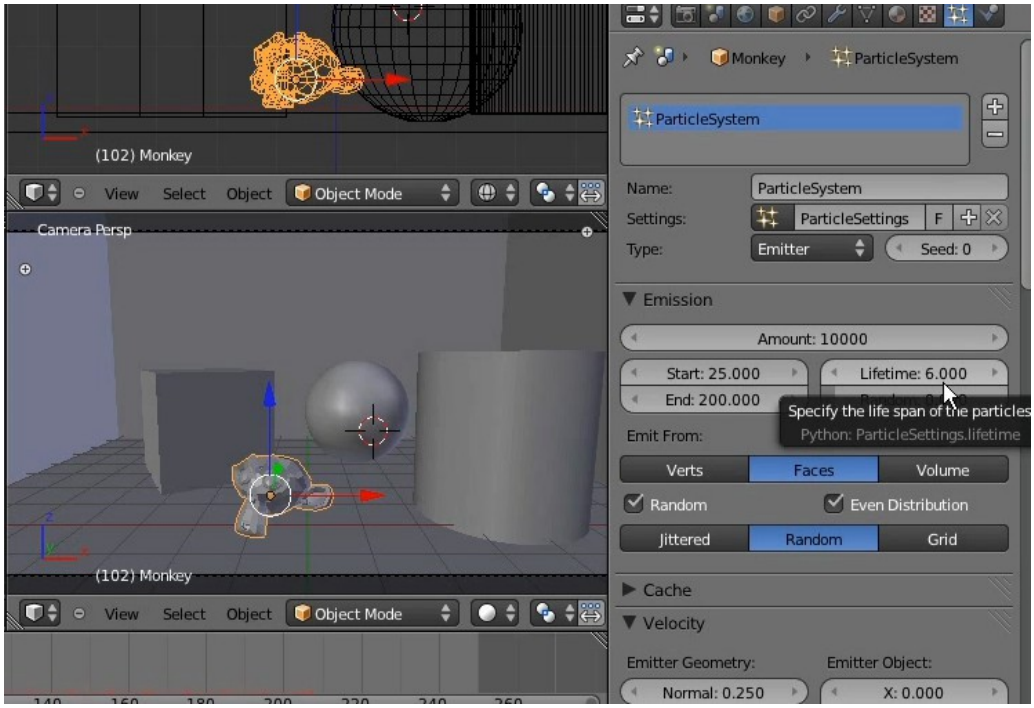


Indirect Light Approximate

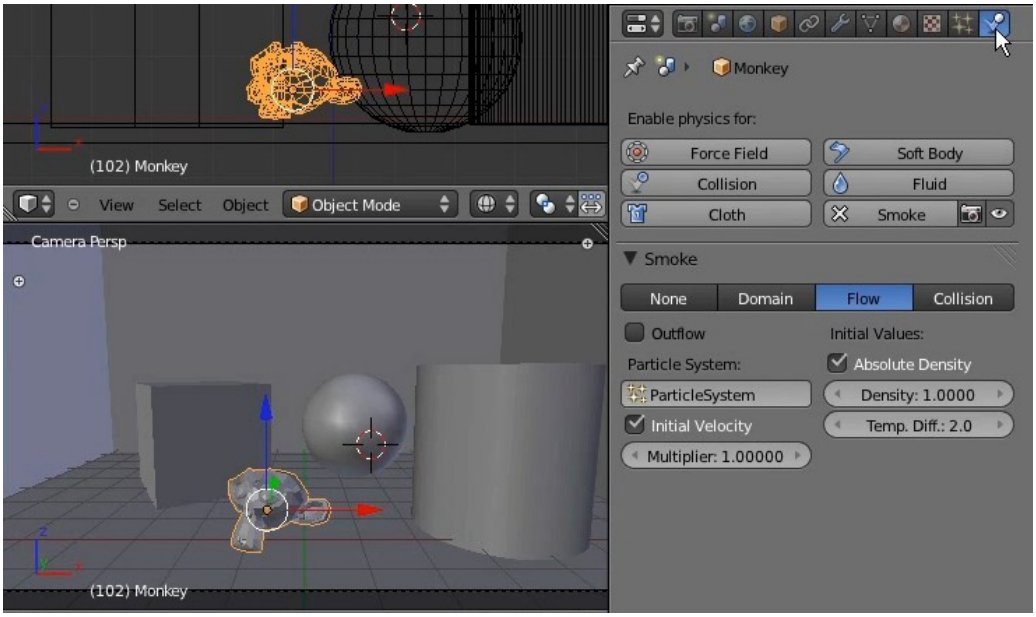
Iniziamo dalle impostazioni del sistema particellare associato a *Suzanne*: questa volta, anziché creare un *Emitter* con end Frame 1, impostiamo i valori come segue:

- 10.000 particelle o più, con *Lifetime* 5 o 6 (quindi molte particelle di breve durata); *Start* ed *End Frame* a piacere, ad esempio nell'animazione con *Smoke HiRes*, che vedremo a breve, ho messo rispettivamente 1 e 250, mentre in quella senza *Smoke HiRes* ho messo 25 e 200 per mostrare l'accensione e l'estinzione delle fiamme;
- cambiare *Emission* da *Jittered* a *Random*;
- abbassare la velocità orientata lungo la Normale delle facce aumentando invece la turbolenza, cosa che va fatta portando *Velocity Normal* a 0.25 e *Velocity Random* a 0.125, i valori ovviamente sono solo indicativi;
- togliamo la gravità impostando *Gravity* 0.0 in *Field Weights*, per cui solo questo sistema particellare non risentirà della gravità di default, che non stiamo disattivando a livello di scena;
- disattivare il rendering delle particelle impostando *Render None*.

Per quanto riguarda il sistema particellare di *Suzanne*, è tutto; dobbiamo impostare, adesso, l'emissione delle particelle di fumo, quindi aggiungiamo un simulatore *Smoke Flow* per la mesh portando la temperatura a 2.0, cosa che farà sviluppare le fiamme più velocemente; per provare un effetto rallentato (come ad esempio il fuoco di un caminetto o un fuoco da campo), abbassare questo valore, portandolo al di sotto di quello di default, che è 1.0.
Selezioniamo infine "*Initial Velocity*".



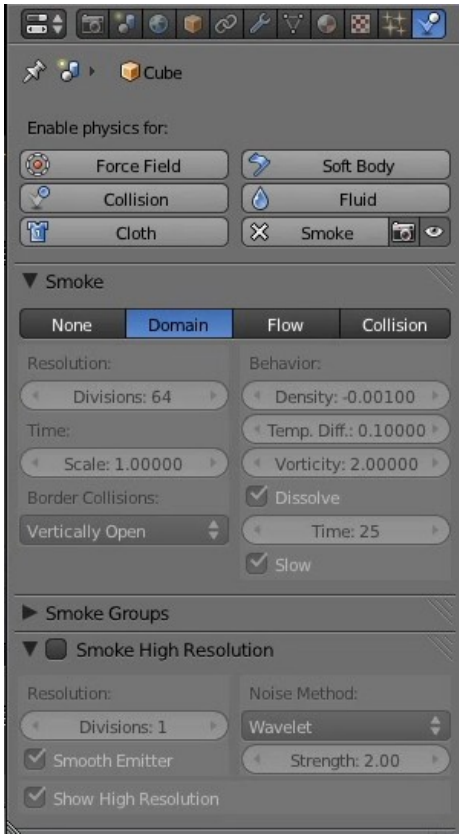
Alcune impostazioni del sistema particellare di Suzanne



Le impostazioni Smoke Flow, per Suzanne

Passiamo al cubo dominio della simulazione: dobbiamo impostare, ovviamente, il simulatore *Smoke*, il Material e la Textures; ricordo infatti che, nel caso delle simulazioni *Smoke*, Material e Texture non vanno impostati per gli oggetti *Flow*, ma per il *Domain*. Una considerazione: il cubo-dominio va ridimensionato sempre in Object Mode; mai fare editing e scaling in Edit Mode.

Iniziamo dalla simulazione *Smoke*, scegliendo ovviamente *Domain*, con un discreto numero di suddivisioni (ad esempio 64; non esagerare se si intende utilizzare *Smoke High Resolution*); selezioniamo *Dissolve* per far sparire lentamente le fiamme, impostando ad esempio *Time* 25 e mantenendo *Slow* selezionato, come di default quando si attiva *Dissolve*.

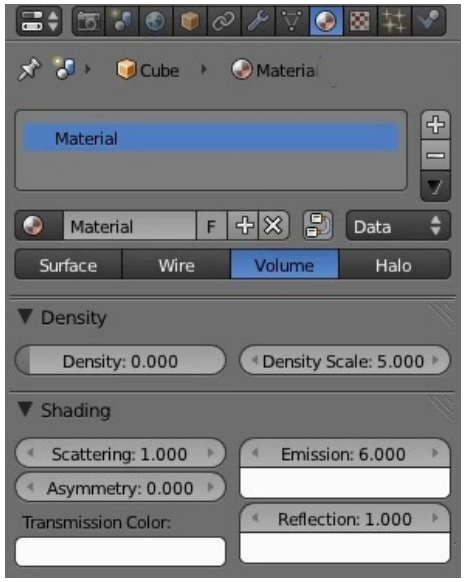


Impostazioni Smoke Domain, per il cubo-dominio

A questo punto, possiamo scegliere se attivare o meno *Smoke High Resolution*.
Smoke HiRes introdurrà delle suddivisioni nelle fiamme, rendendole più fini ma anche più turbolente; richiederà inoltre molto, MOLTO più tempo per il *baking*, anche con una sola suddivisione aggiuntiva, sia con modalità *FFT* che *Wavelet*.

Passiamo al Material, ovviamente di tipo *Volume*, da associare al cubo-dominio.

Impostiamo *Density* a 0.0, *Density Scale* a 4.0 o 5.0, *Scattering* 0.6 ed *Emission* tra 2.0 e 10.0, tenendo conto che tale valore rappresenta la luminosità delle fiamme.

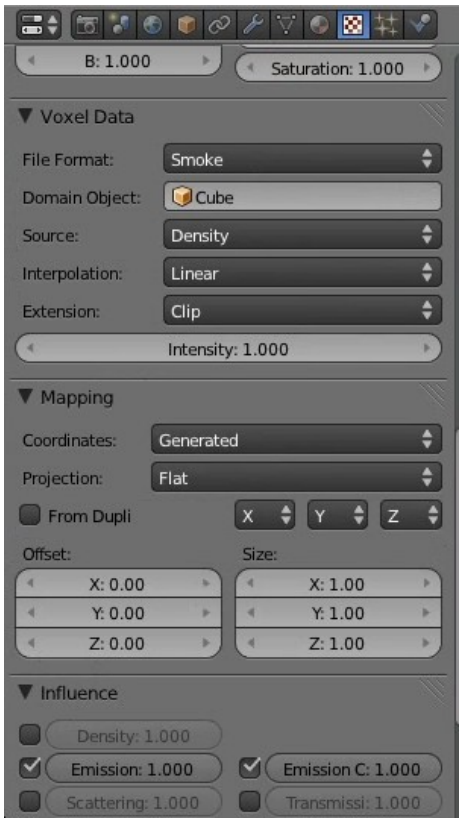


Impostazioni del Material Volume per il cubo-dominio

Densità, intensità e colore dell'emissione verranno comunque regolate dalle Textures *Voxel Data*; per maggiori informazioni sul significato dei vari parametri, rimando ai miei due ebook su *Materials e Textures in Blender 3D 2.5*.

Sempre nel Material, impostiamo uno *Step Size* più basso di quello di default per *Integration*, ad esempio 0.1 o, se il computer "regge", ancora più basso, per migliorare la qualità della resa.

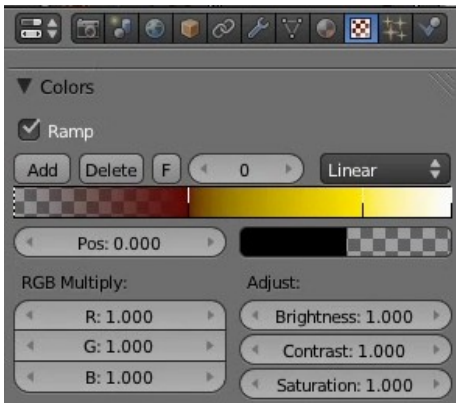
Dobbiamo associare due Textures di tipo *Voxel Data* a tale Material. La prima Texture dovrà influenzare il canale *Density* del Material con intensità 1.0; a parte impostare il cubo stesso come oggetto *Domain* della simulazione, non dobbiamo fare altro.



Le impostazioni della seconda Texture Voxel Data per il cubo-dominio

La seconda Texture dovrà influenzare invece i canali *Emission* ed *Emission Color*, in entrambi i casi con intensità 1.0, e dovrà avere modalità di *Blending Multiply*; ovviamente, l'oggetto *Domain* della simulazione sarà il cubo.
A questo punto dobbiamo personalizzare la *Ramp* per definire il colore delle fiamme; vediamo quindi qual è il criterio utilizzato per colorarle.

Nella *Ramp*, l'estremità sinistra (ossia la posizione 0.0) verrà associata alle zone di densità nulla del fumo, mentre quella all'estrema destra colorerà le zone di densità massima; per questo motivo, definiamo almeno 4 bande, con la banda all'estrema sinistra nera trasparente, la seconda banda color rosso scuro con *Alpha* 1.0 (quindi completamente opaca), la terza gialla con *Alpha* 1.0 e la quarta, all'estrema destra, bianca, con *Alpha* 1.0.

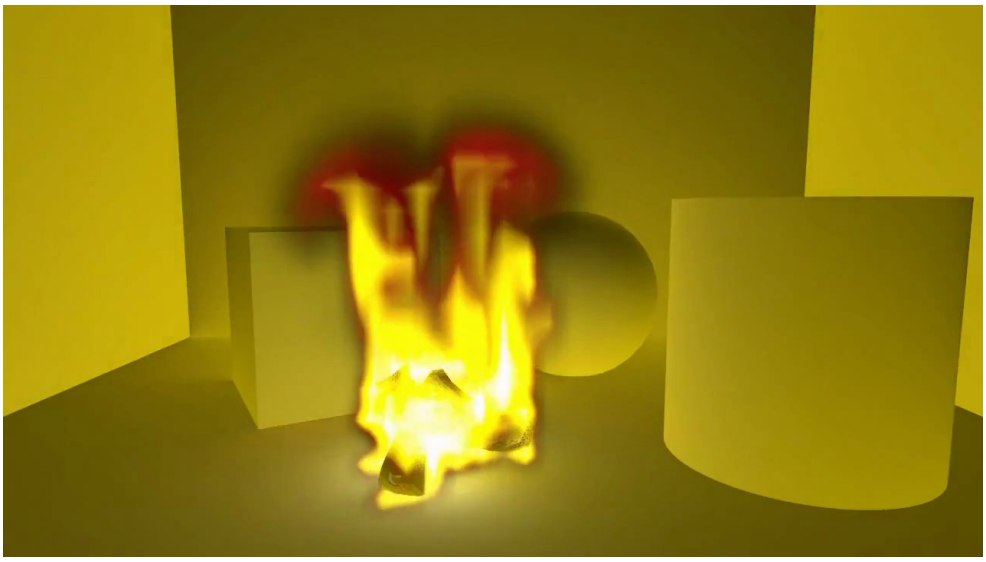


Le impostazioni della Color Ramp

Dove posizionare le bande centrali, ossia la seconda e la terza? Dipende dall'effetto che volete ottenere: tenete conto che spostando la seconda banda verso il centro otterrete una maggiore quantità di fumo, in quanto aumenterà il range di densità associato al nero trasparente, mentre spostando la terza banda verso il centro lascerete più spazio, forse troppo, al bianco; nel mio caso, ho posto le bande in 0.4 e 0.8.

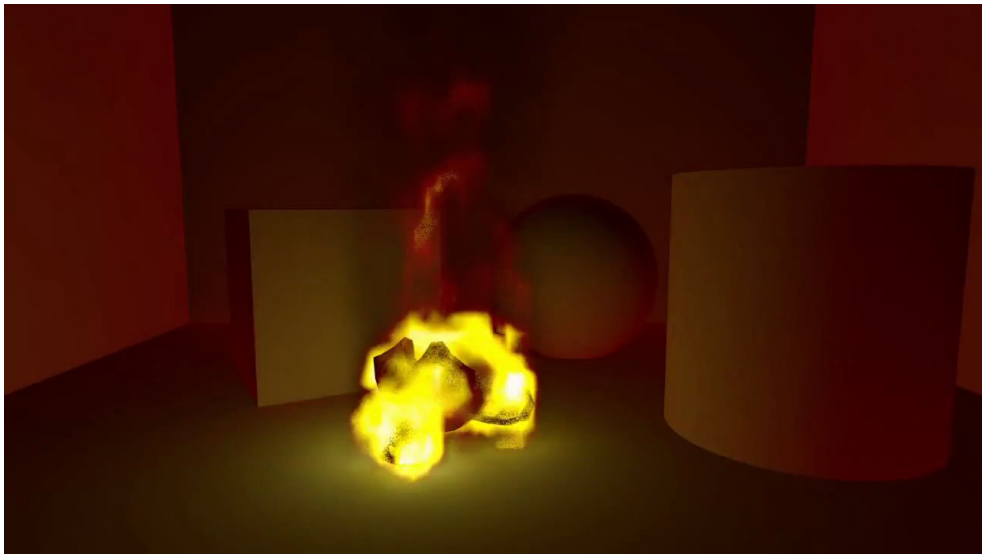
Per quanto riguarda la preparazione della scena, questo è tutto: facciamo quindi il *Bake* della simulazione (cliccando su tale pulsante nella scheda *Smoke* del cubo-dominio), avviamo il

rendering ed osserviamo il risultato; ovviamente, dovremo fare *bake* e rendering due volte: la prima volta senza *Smoke High Resolution*, la seconda con.
Ecco uno screenshot del rendering senza *Smoke High Resolution*:



Screenshot del rendering senza Smoke High Resolution

ed ecco invece il rendering con *Smoke High Resolution*:



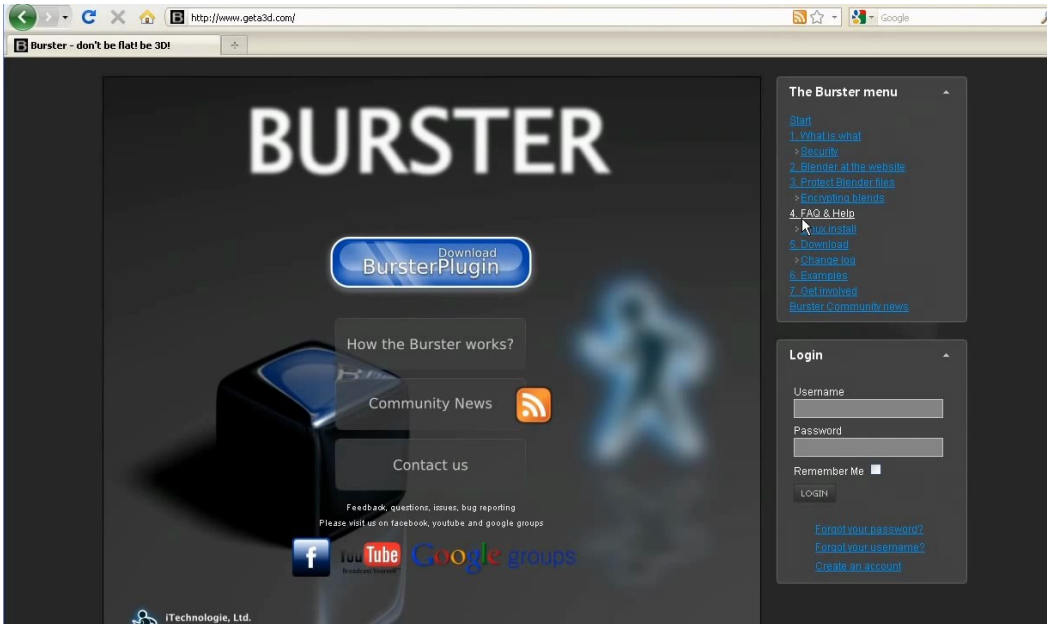
Screenshot del rendering con Smoke High Resolution

Come anticipato, attivando *Smoke High Resolution* avremo fiamme più "fini" ma anche più turbolente, come si può notare anche dall'illuminazione "scoppiettante" prodotta dalla simulazione; senza *Smoke HiRes* avremo fiamme più grossolane (cosa risolvibile comunque aumentando le *Divisions* del *Domain*) ma meno turbolente.

* * *

Burster: plugin per eseguire applicazioni e giochi fatti col Blender GE nei browser

In questo breve articolo voglio parlarvi di *Burster*, un plugin per browser web che consente di caricare, all'interno della pagina web di un sito, progetti realizzati con il Blender Game Engine (il motore di gioco di Blender), consentendo ai visitatori del sito di eseguire giochi o altre applicazioni 3D, come cataloghi o tour virtuali, in tempo reale.
Per quanto riguarda gli utenti, è sufficiente scaricare il plugin per il browser web; attualmente, sul sito web ufficiale di *Burster* (www.geta3d.com) sono presenti vari plugin per browser web per le varie piattaforme Window e Linux.



Il sito ufficiale del Plugin BURSTER

Per quanto riguarda gli sviluppatori, è sufficiente inserire una porzione di codice all'interno della pagina, caricando il progetto *.blend* come se si trattasse di un componente FLASH; la porzione di codice da utilizzare viene mostrata su *geta3d.com*: è sufficiente andare nella sezione "*Blender at the website*", aprire il topic "*Embedding blends*" (incorpora file *.blend*, appunto) ed osservare il codice.

```
<OBJECT classid="CLSID:8318DE8B-B213-426b-B1B6-0A2589859898"
        width="800"
        height="600">
        <PARAM name="type" value="application/x-burster"/>
        <PARAM name="src" value="yourfile.blend" />
        <embed type="application/x-burster"
                src = "yourfile.blend"
                width = "800"
                height = "600">
        </embed>
</OBJECT>
```

La porzione di codice da utilizzare per incorporare un progetto BLEND in una pagina web

All'interno del tag *OBJECT* è possibile specificare le dimensioni della finestra all'interno della quale andrà eseguito il gioco, con *WIDTH*, *HEIGHT*, l'*url* del file *.blend* e altri parametri con voci standard come "*type*".

Ovviamente, caricare il file *.blend* "puro" sul sito non è il massimo (se desideriamo proteggere il sorgente dell'applicazione rendendolo eseguibile ma non distribuibile via browser web). *Burster* consente quindi di criptare il file *.blend* del nostro progetto e caricare ed utilizzare il file protetto; per sfruttare questa opzione (ottenendo i certificati di crittografia) è necessario registrarsi al sito.

Su *geta3d.com* sono disponibili anche degli esempi pratici; ad esempio, in *Examples* abbiamo una versione 3D di *Space Invaders*, detta "*Space Busters*". Aprendo la pagina troveremo l'applicazione incorporata nella finestra del browser e, dopo aver digitato il nostro nickname, potremo avviare la partita e giocare (dopo il caricamento del gioco e il controllo dei certificati).

Scegliere un gioco nella sezione EXAMPLES e attendere il caricamento e il controllo dei certificati

In particolare, a "*Space Busters*" si gioca muovendo il mouse e facendo click col tasto sinistro per sparare, mentre con *ESC* si sospende il gioco.

Screenshot dell'esecuzione di Space Busters

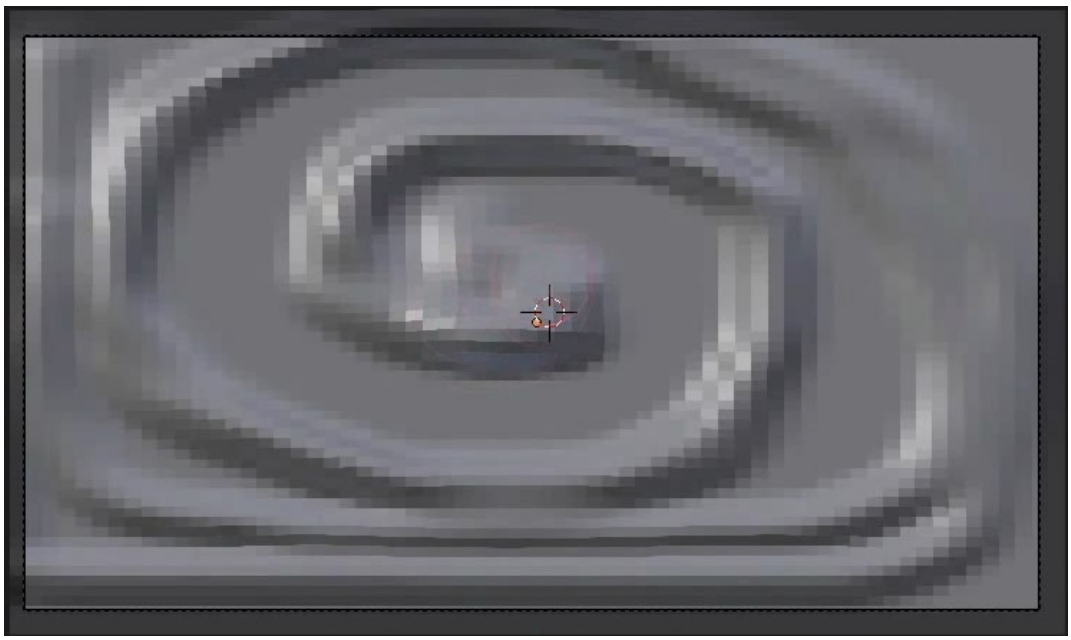
* * *

Creare Normal e Bump Maps dallo Z-Buffer di un rendering

In questo tutorial vedremo come ricavare una *Normal Map* dal rendering di un oggetto provvisto di rilievi e depressioni e inquadrato frontalmente in Blender 3D 2.5.

La tecnica è utile per reperire, da un oggetto con un elevato numero di poligoni, le informazioni sulla superficie ed applicarle come Texture sul canale *Geometry Normal* di un oggetto in bassa risoluzione. Il tutorial è stato realizzato con la versione 2.58 stabile di Blender 3D.

Iniziamo da un semplice Plane sul quale creeremo dei rilievi o delle depressioni mediante lo strumento *Sculpt* di Blender. Poniamo la telecamera di fronte al Plane e cambiamo la modalità di rendering della telecamera da *Prospettiva*, la modalità selezionata di default, a *Ortogonale*, scegliendo appunto la voce *Ortho* nella scheda *Object Data* della telecamera.



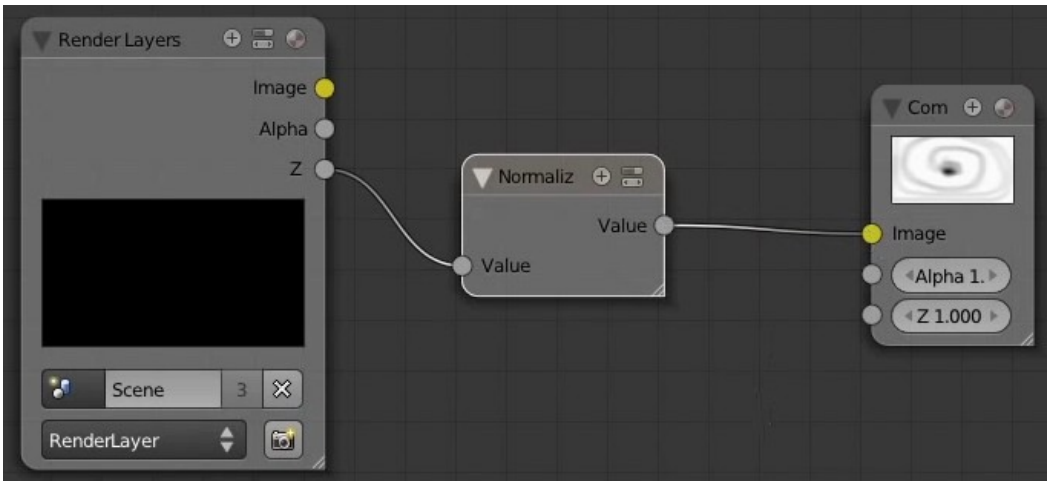
La scena utilizzata per questo tutorial, con il Plane suddiviso e "modellato" mediante Sculpt

Selezioniamo il Plane e suddividiamolo un buon numero di volte; per poter ottenere buoni risultati con *Sculpt*, infatti, è necessario lavorare su mesh con una buona risoluzione, ossia con un elevato numero di poligoni, magari levigati ulteriormente con *Set Smooth* o *SubSurf*. Utilizziamo quindi lo strumento *Sculpt Mode* per modellare un po' il Plane; va da sé che potete utilizzare qualsiasi tecnica o strumento di modellazione: *Displace*, editing di vertici, spigoli e facce, eccetera.

A questo punto, cambiamo una finestra in Editor dei Nodi, attiviamo il *Nodes Compositing* scegliendo la terza icona del gruppo di tre nell'header e, subito dopo, selezionando la casella "*Use Nodes*", che apparirà nell'header accanto alle icone.

Tagliamo il collegamento tra *Image Output* di *Render Layer* e *Composite* e colleghiamo l'output *Z* di *Render Layers* al canale di input *Image* di *Composite*... renderizzando, però, avremo una brutta sorpresa: l'immagine in output è completamente bianca.

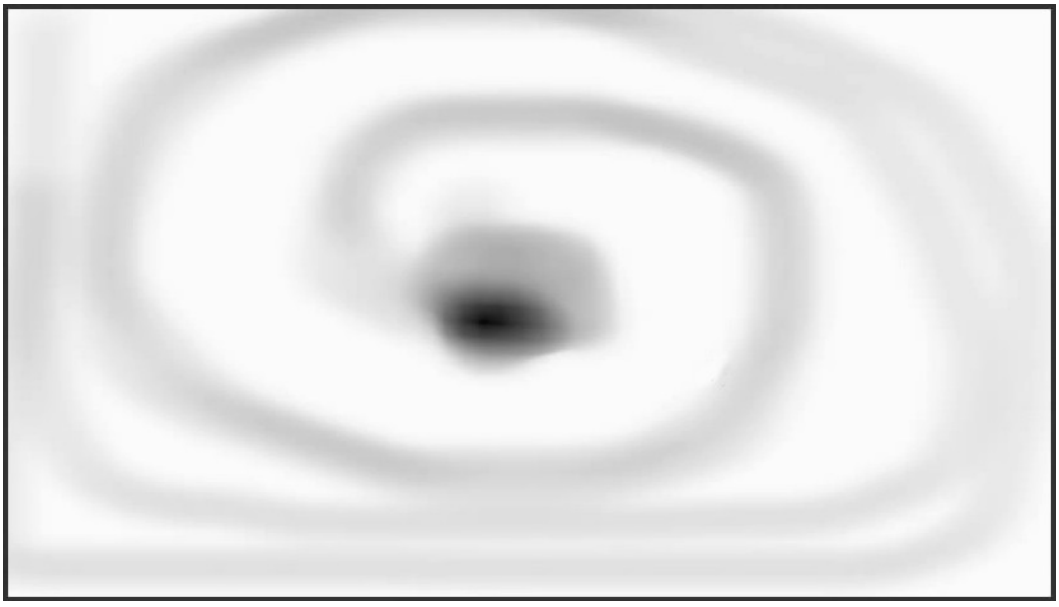
Come specificato nel mio *ebook sul Compositing*, infatti, l'output del canale *Z Buffer* è a 32 bit, un po' come le informazioni contenute nei file *EXR*; per normalizzare i dati nel range "standard", dobbiamo utilizzare il Nodo *Normalize*, ponendolo tra *Z* di *Render Layers* e *Image* di *Composite*.



Lo schema di Nodi di Compositing

A questo punto, dopo aver centrato il Plane nell'area inquadrata dalla telecamera (eventualmente cambiando la risoluzione del rendering), possiamo effettuare il rendering e salvare l'immagine.

Tale immagine può essere ovviamente modificata con programmi di fotoritocco, ad esempio aumentando il contrasto, la luminosità, la sfocatura gaussiana ed altro, tenendo conto del fatto che le modifiche si tradurranno in effetti sul canale *Geometry Normal*.

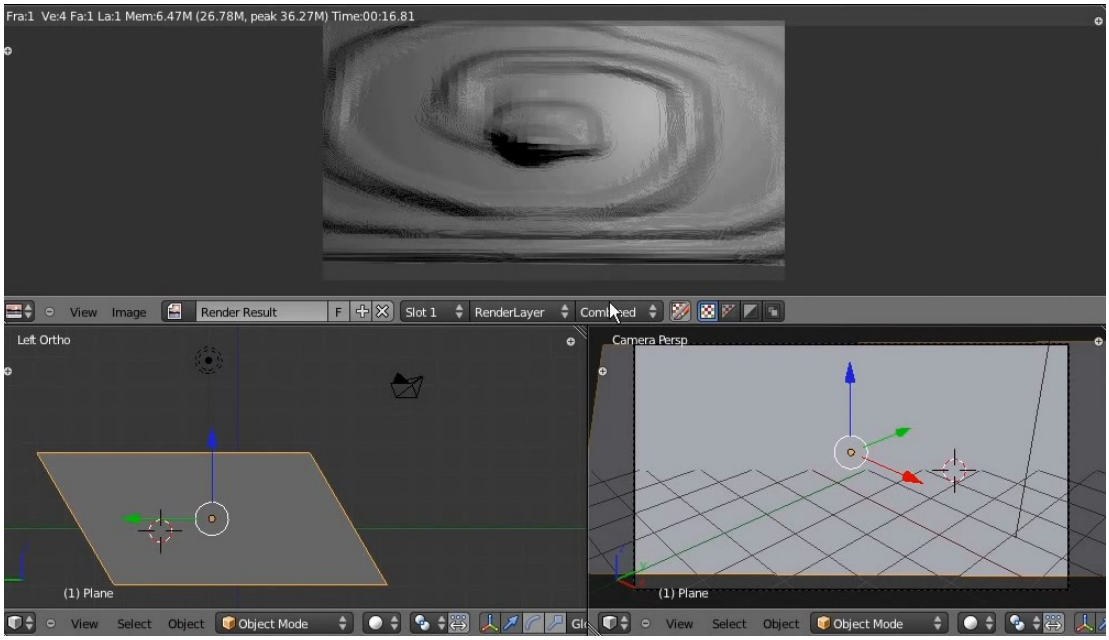


L'immagine ottenuta

La prima fase del lavoro è pronta; adesso, apriamo una nuova scena, utilizzando un nuovo *Plane* o comunque una mesh con un basso numero di poligoni, quindi associamo un Material a tale mesh e una Texture di tipo immagine.

Come immagine caricheremo ovviamente quella generata poco fa, mappandola sull'oggetto come preferiamo (ad esempio, con mappatura *Flat* per un nuovo Plane o *UV* per oggetti più complessi), quindi deselezioniamo *Color Diffuse* in *Influence* e selezioniamo invece *Geometry Normal*, effettuando alcuni rendering di prova per scegliere il valore adatto per l'intensità dell'effetto.

Il concetto è il seguente: il canale *Z* del rendering restituisce informazioni sulla distanza degli elementi inquadrati dalla telecamera, con nero per i vertici più vicini e bianco per quelli più lontani; se renderizziamo un elemento inquadrato frontalmente con modalità *Ortho*, possiamo usare tali dati come informazioni sulla geometria della mesh.



Il risultato dell'utilizzo della Texture immagine appena creata sul canale Geometry Normal di un Plane

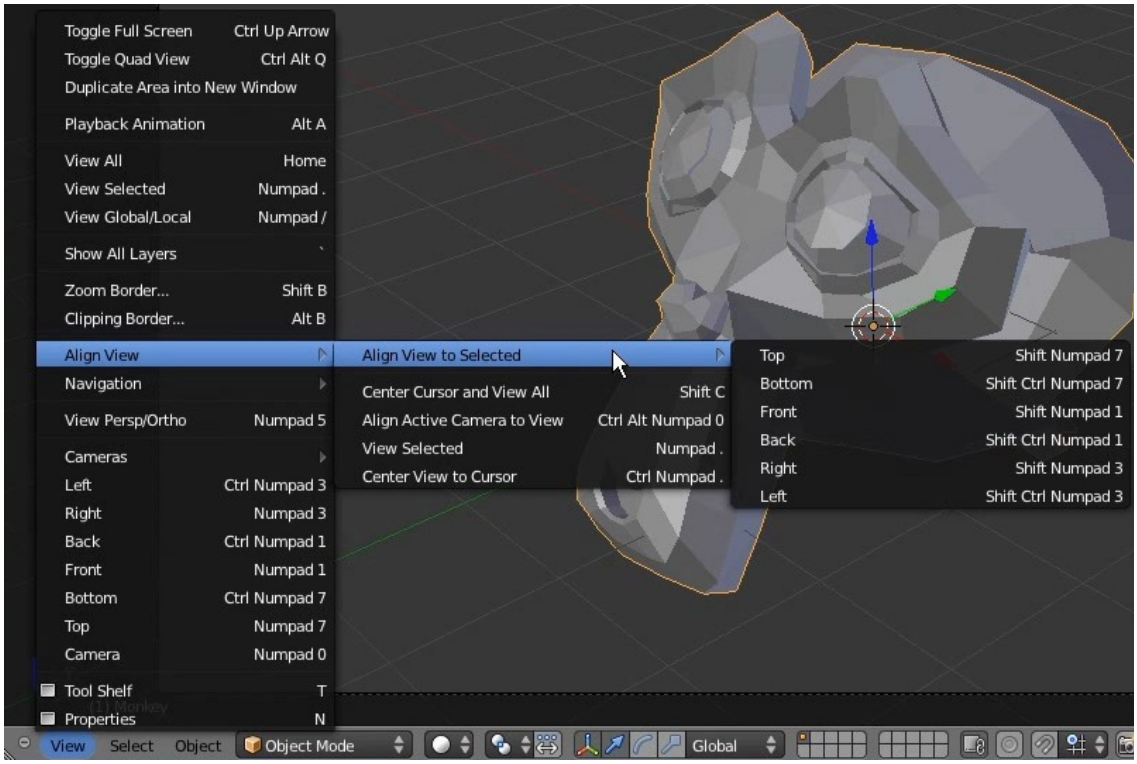
Geometry Normal non modifica la geometria in tempo reale: le operazioni vengono svolte in fase di rendering, modificando i valori dei pixel durante il processo di rasterizzazione, infatti *Geometry Normal simula* la presenza di rilievi, depressioni ed altri elementi sulle facce ma senza deformare la geometria delle stesse (operazione ben diversa da *Displace* che invece deforma fisicamente le mesh prima di avviare il rendering e che, per questo motivo, dev'essere utilizzato su mesh con una buona risoluzione, ossia un elevato numero di poligoni).

Le Textures così generate vengono utilizzate soprattutto nei videogiochi, infatti per simulare la profondità e i rilievi degli oggetti non c'è bisogno di utilizzare mesh complesse e modellate ma si può applicare una Texture *Normal Map* a mesh low poly, lasciando alla scheda video il compito di calcolare la resa in tempo reale.

* * *

GUI: le voci del menù Align View per allineare vista e telecamera

In questo brevissimo "Tips & Tricks" sulla GUI e la navigazione all'interno dell'universo virtuale di Blender 3D 2.5 parleremo delle voci della sezione "*Align View*", utilissime per allineare il punto di vista rispetto ad un oggetto o al *3D Cursor*, oppure per posizionare la telecamera in modo da inquadrare la vista corrente. Per accedere a questo menù, cliccare su *View - Align View* nell'header di una qualsiasi finestra di navigazione *3D View.*



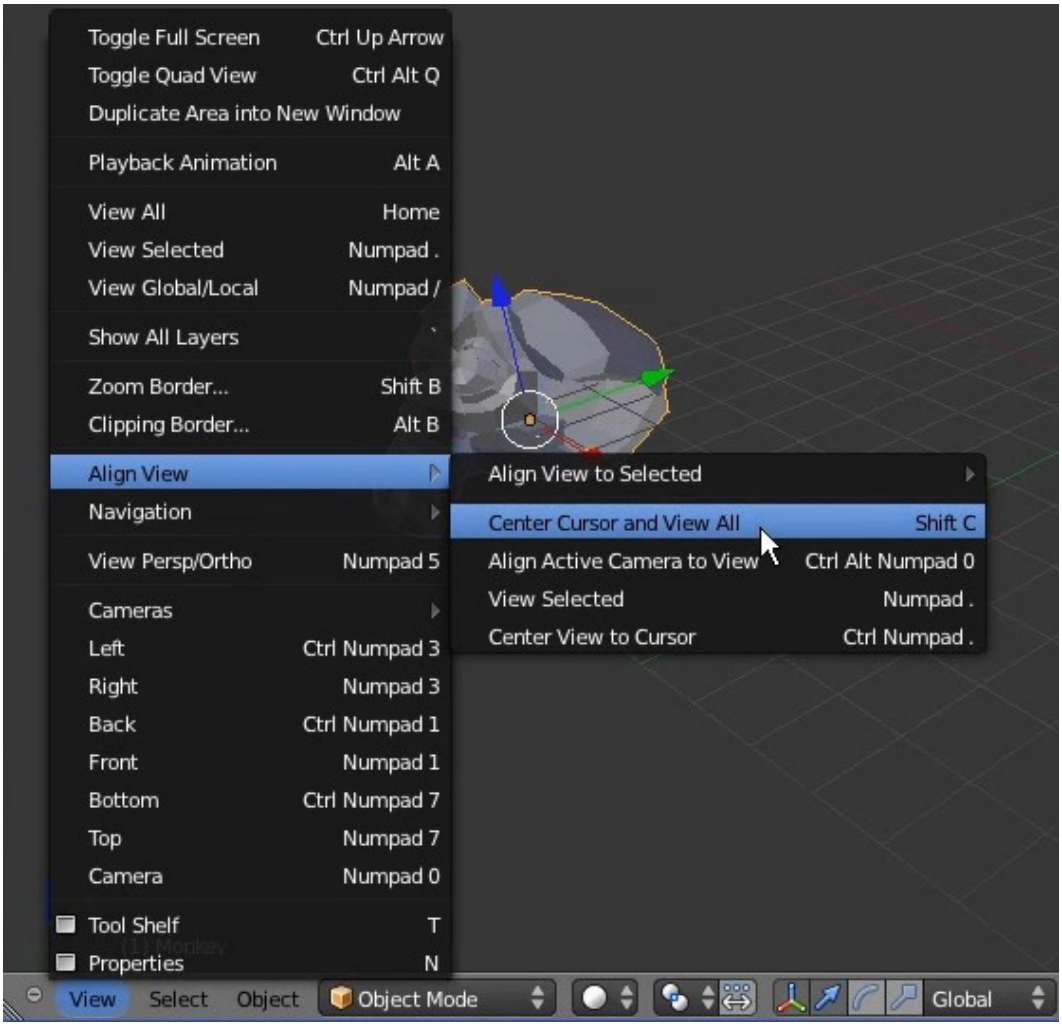
View - Align View

La prima voce è, in realtà, un altro sottomenù: "*Align View to Selected*", ossia "allinea la vista sull'elemento selezionato"; intuitivamente, un oggetto può essere visto da 6 lati, se si immagina di racchiuderlo all'interno di un *bouding box*, per cui, dopo aver selezionato un elemento, si dovrà scegliere il punto di vista tra le voci *Top*, *Bottom*, *Front*, *Back*, *Left*, *Right* (cioè da sopra, da sotto, di fronte, da dietro, da sinistra, da destra) nel sottomenù *Align View to Selected*.

A cosa corrispondono queste voci? Per "sopra" e "sotto" ci si regola con l'asse Z, mentre per destra e sinistra con l'asse X e per avanti e dietro con l'asse Y; per capire meglio questi orientamenti, conviene utilizzare una primitiva come Suzanne, anziché coni, sfere o cubi.

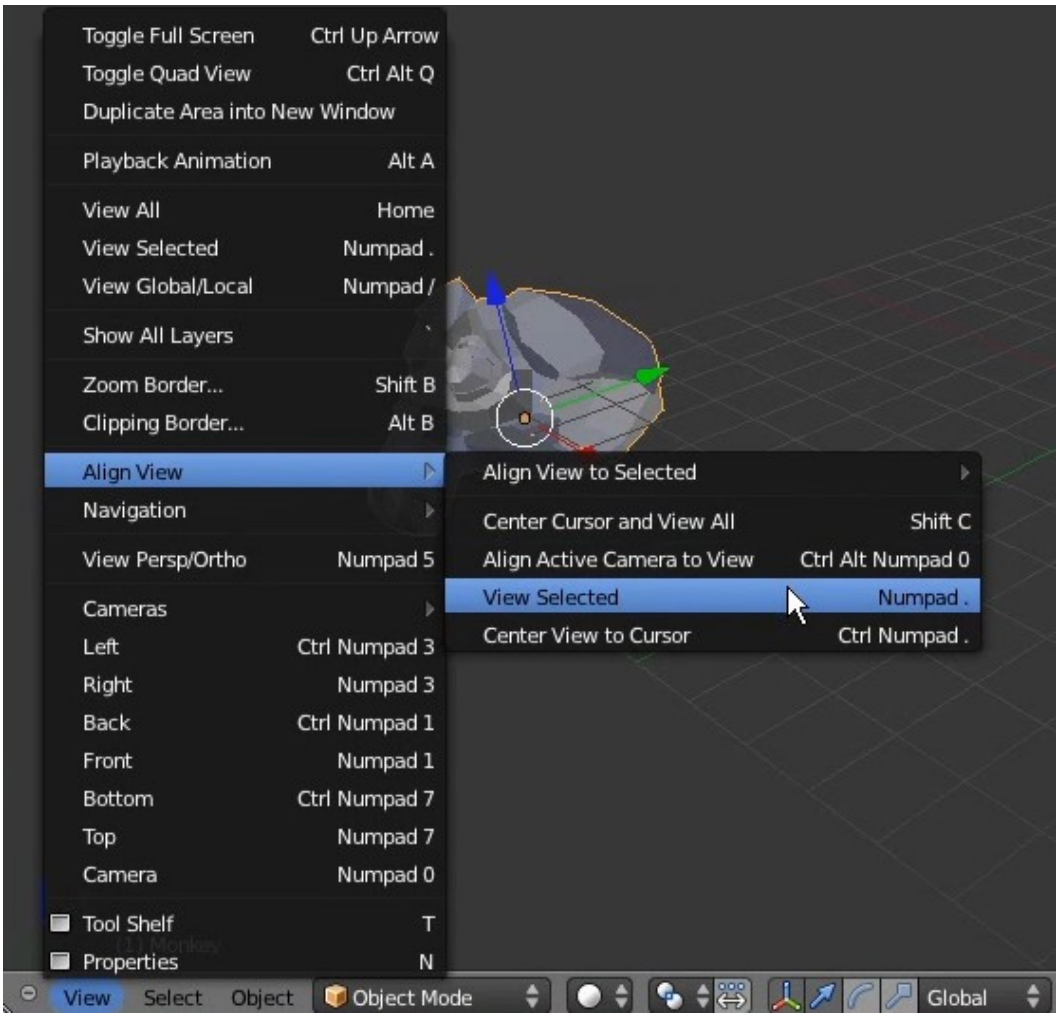
Queste voci si riferiscono al sistema di riferimento locale dell'oggetto (d'altronde, per impostare una vista del genere in base al sistema di riferimento globale della scena, è sufficiente scegliere direttamente tra *Top*, *Bottom*, *Front*, *Back*, *Left* e *Right* di *View*).

Per avere una panoramica di tutta la scena, visualizzando tutti gli elementi in essa presenti, possiamo scegliere "*Center Cursor and View All*", che serve appunto a posizionare il *3D Cursor* nell'origine dell'universo virtuale, ossia alle coordinate XYZ 0, 0, 0, e allontanare il punto di vista della *3D View* in modo da far rientrare nella vista tutti gli elementi presenti nella scena.



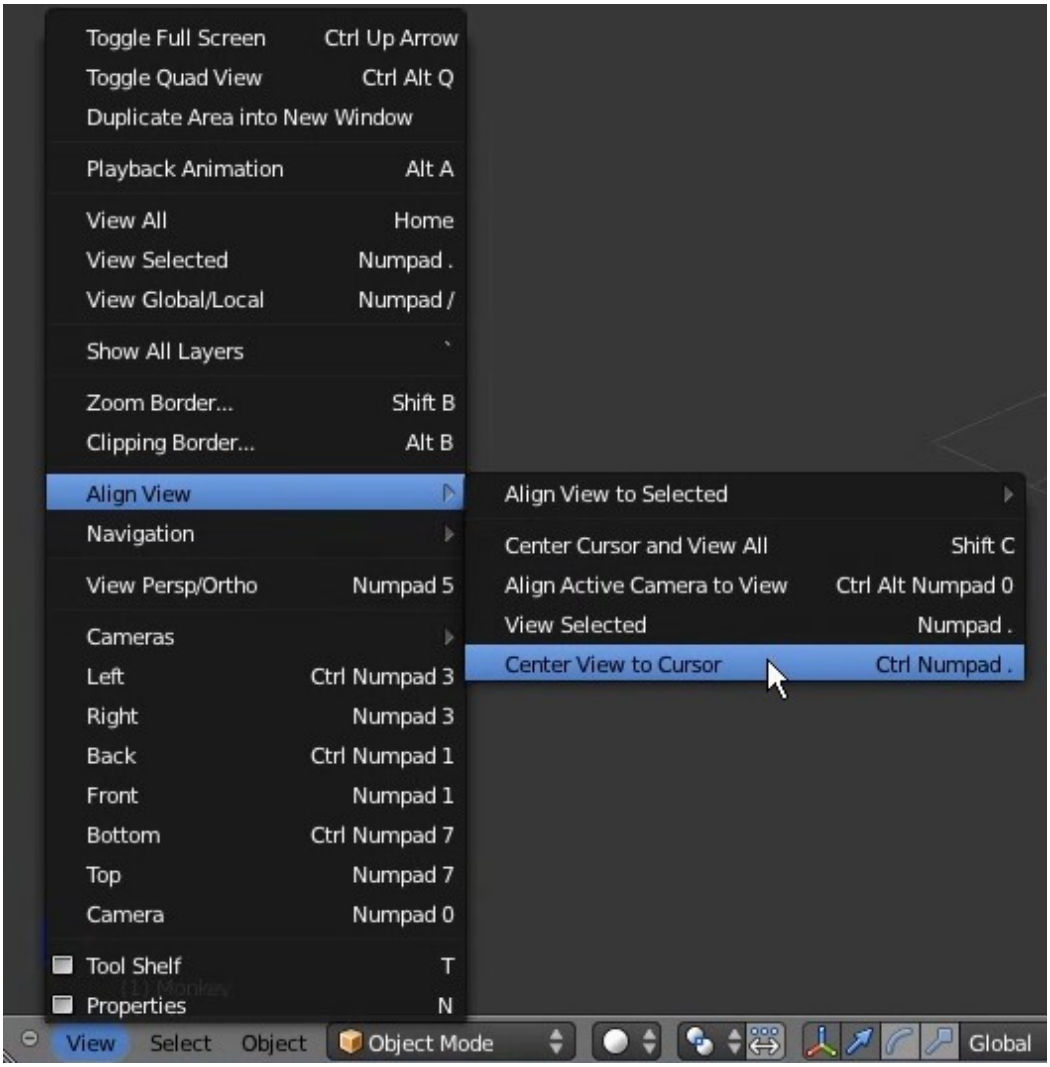
Center Cursor to View All

Se, invece, vogliamo zoomare su un elemento o un gruppo di elementi, selezioniamoli nella scena e scegliamo la voce "*View Selected*": il punto di vista verrà spostato per inquadrare la selezione.



View Selected

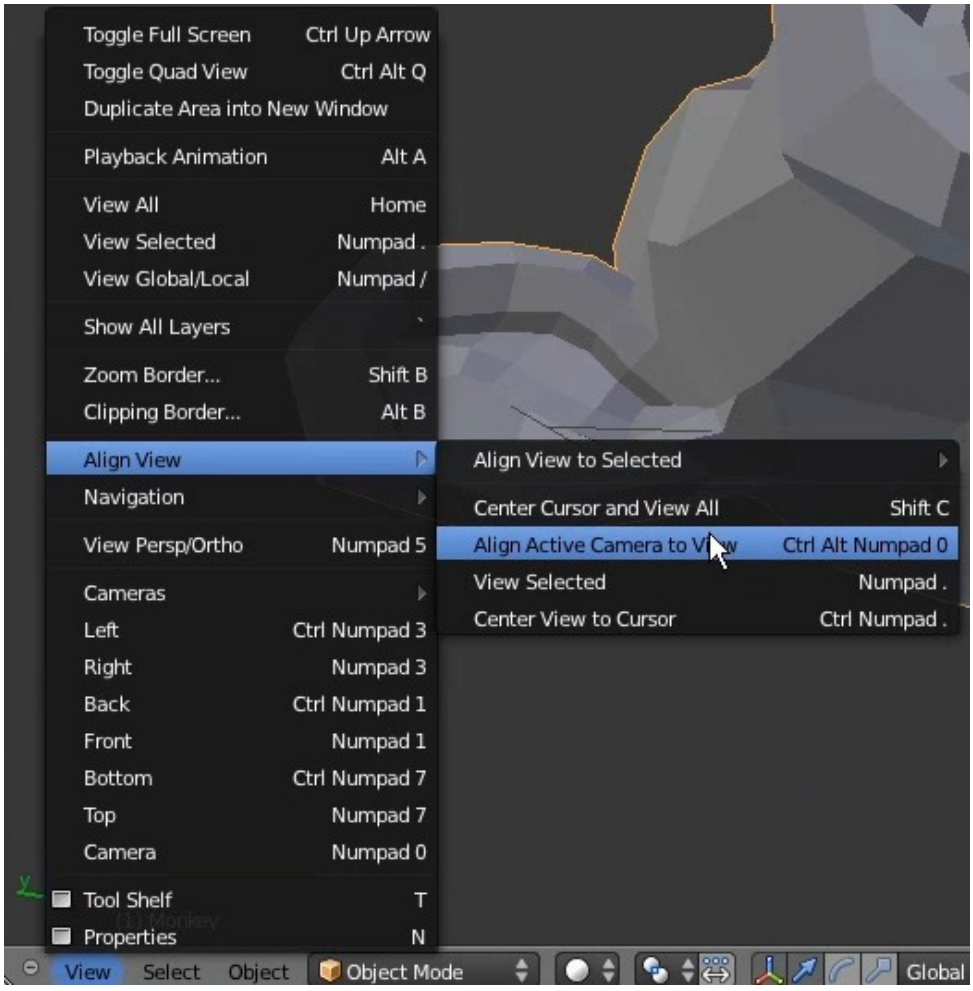
Possiamo centrare il punto di vista della scena anche sulla posizione del *3D Cursor*, che come sappiamo può essere posizionato con gli *snap* del menù *SHIFT S* o con le voci *3D Cursor Location* nella *Transform Window*; per posizionare la vista centrandola sul *3D Cursor*, scegliere la voce "*Center View to Cursor*".



Center View to Cursor

Passiamo ad *Align Camera to View*, che trasforma la telecamera, non il punto di vista della *3D View*, come facevano le altre voci viste finora.

Capita, a volte, di muoversi liberamente nella scena 3D fino a trovare una buon punto di vista per osservare la scena, con una composizione ben bilanciata, e voler posizionare in quel punto e con quell'orientamento la telecamera per avviare il rendering; in questo caso, la voce "*Align Active Camera to View*", ossia "Allinea la telecamera attiva al punto di vista" fa al caso nostro.



Align Active Camera to View

La telecamera verrà spostata in modo tale da assumere tale posizione; tenete conto, ovviamente, della differenza di lunghezza focale e delle modalità *Ortogonale/Prospettiva*, sia per la *3D View* nella quale avete effettuato il posizionamento che per la telecamera, che di default ha modalità di rendering *Prospettiva*.

I parametri lunghezza focale (*Focal Length*) e la modalità *Prospettiva/Ortogonale* in fase di rendering possono essere impostati nella scheda *Object Data* della telecamera, nella *Properties Window.*

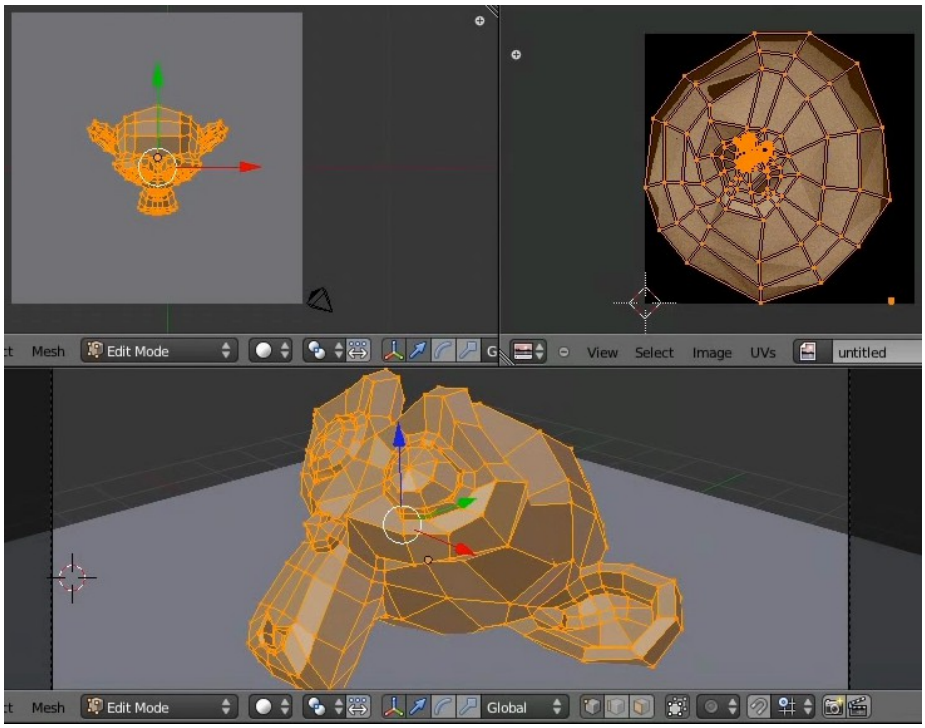
La scheda Object Data della telecamera

* * *

Sfocare il Bake di una mappa Ambient Occlusion col Nodes Editor

In questo brevissimo tutorial vedremo un trucchetto per pulire e rendere più luminosa una mappa *Ambient Occlusion* ottenuta mediante *baking* di tale canale in Blender 3D 2.5.

Le mappe *Ambient Occlusion* vengono in genere moltiplicate al canale *Color Diffuse* di un oggetto, quindi più scure sono e più renderanno scuro l'oggetto; inoltre, per renderizzarle in poco tempo, si mantiene un basso numero di *Samples*, tuttavia questo porta in genere a mappe rumorose, ossia con molti puntini scuri. Procediamo con ordine, generando la mappa *Ambient Occlusion* per una mesh, nel mio caso *Suzanne*: attiviamo *Ambient Occlusion* a livello globale con le impostazioni di base, dopodiché cambiamo una finestra in *UV – Editor delle immagini* ed inseriamo una nuova immagine di default, nera. Selezioniamo *Suzanne* in una *3D View*, passiamo in Edit Mode, selezioniamo tutte le facce dell'oggetto e facciamo un *unwrap* di base con *U* e scegliendo *Unwrap* dal menù che apparirà a video, in modo da creare un *Layer UV* per la mesh.



L'unwrap di Suzanne, su una nuova immagine nella finestra UV, per realizzare il Baking dell'Ambient Occlusion

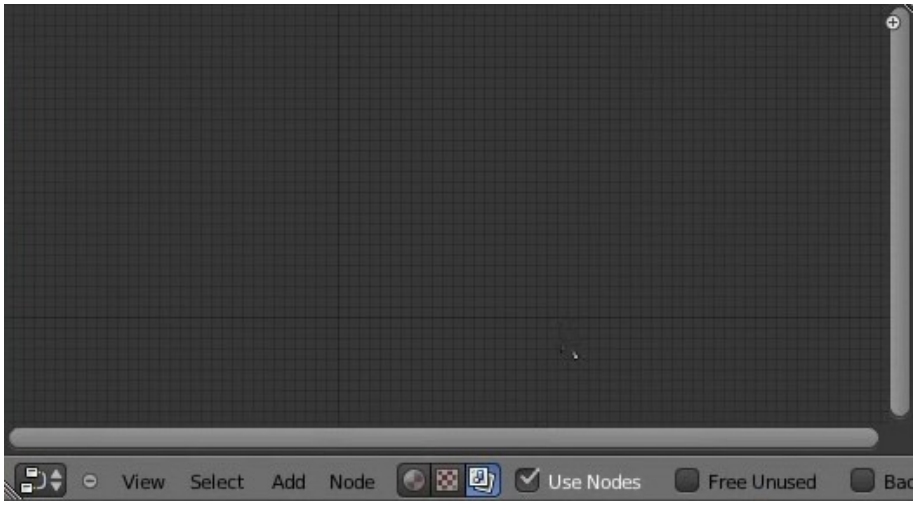
A questo punto possiamo aprire la sezione *Bake* della scheda Render, nella *Properties Window*, scegliere la modalità *Ambient Occlusion* ed avviare il *baking* di tale canale; nella finestra *UV – Editor delle immagini*, vedremo in tempo reale la creazione della mappa *Ambient Occlusion* sul *Layer UV* creato per *Suzanne*.

Baking dell'Ambient Occlusion

Terminata la creazione dell'immagine, salviamola su disco, chiamandola ad esempio *SuzanneOcclusion1*.

Cambiamo quindi una finestra in *Nodes Editor*, scegliamo la terza icona del gruppo di tre nell'header (ossia l'icona Compositing Nodes) ed attiviamo l'utilizzo dei Nodi selezionando la casella *Use Nodes*.

Cancelliamo tutti i nodi e i collegamenti presenti di default.

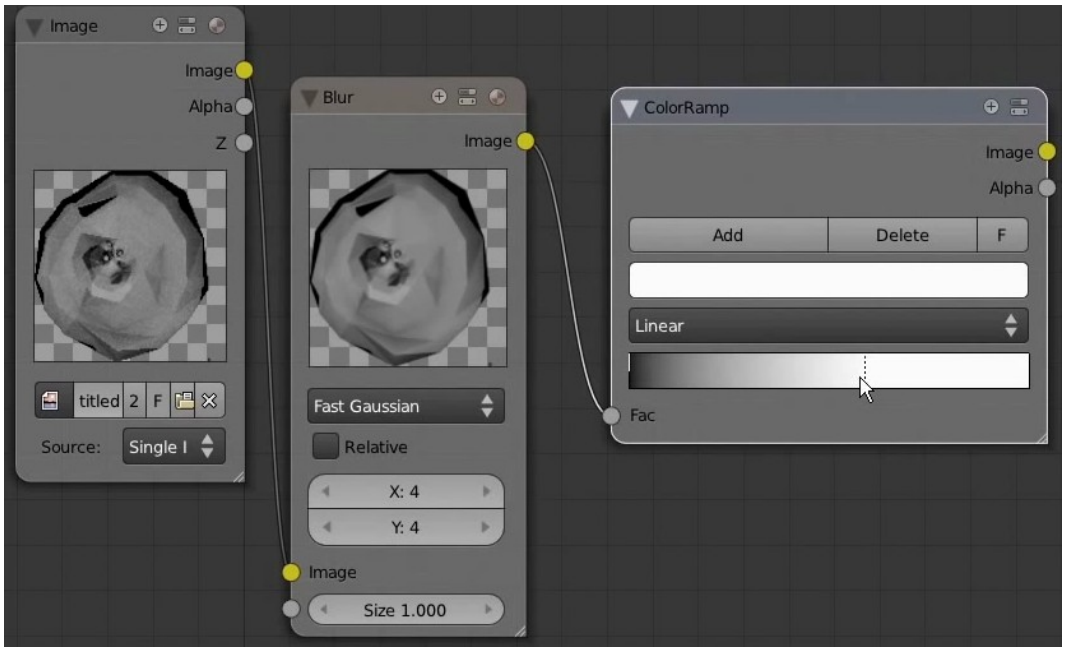


La finestra Nodes Editor, in modalità Compositing, vuota (dopo aver eliminato tutti gli elementi presenti all'inizio)

Quello che dobbiamo fare è prelevare l'immagine appena creata e salvata ed applicarle prima un filtro di sfocatura gaussiana, in modo da rimuovere i punti neri del "rumore", sfocandoli un po', quindi un Nodo *Ramp* per variare la luminosità della mappa (Per una descrizione approfondita del Compositing in post-produzione e dei nodi, rimando al mio *ebook sul Compositing dei Nodes in Blender 3D 2.5*).

Aggiungiamo un Nodo *Input Image* caricando in tale Nodo l'immagine creata e salvata poco fa, quindi passiamo l'output *Image* del Nodo ad un Nodo *Filter Blur* con modalità, ad esempio, *Fast Gaussian* e valori X e Y a piacere, ma bassi, ad esempio da 2x2 a 8x8.

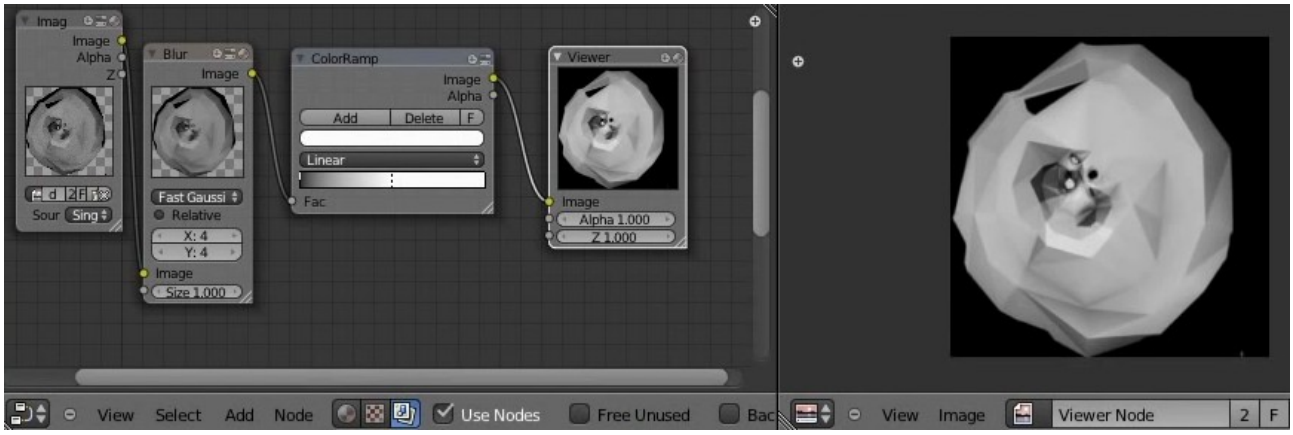
Colleghiamo l'output di *Blur* all'input *Factor* di un Nodo *Convertor ColorRamp*; tale nodo associa il colore e il livello di trasparenza della banda all'estremità sinistra alla parte più scura dell'immagine, il colore e il livello di trasparenza della banda all'estremità destra alla parte più luminosa dell'immagine e la gradazione intermedia (definita aggiungendo, eventualmente, altre bande) ai valori intermedi; intuitivamente, per rendere più chiara l'immagine dobbiamo assegnare il bianco alle zone che originariamente erano grigie, quindi trasciniamo la banda bianca presente all'estremità destra... verso sinistra, aumentando la porzione bianca della *Ramp*.



Lo schema di Nodi di Compositing realizzato fino a questo punto

Per visualizzare in tempo reale il risultato di tutte queste operazioni, colleghiamo l'output *Image* del Nodo *Color Ramp* ad un Nodo *Output Viewer*; in questo modo possiamo anche tornare indietro e variare ad esempio i parametri del Nodo *Blur* osservando i risultati in *Viewer*.

Terminato questo lavoro, torniamo in una finestra *UV – Editor delle immagini* per salvare il risultato appena ottenuto su disco, in modo da poter utilizzare in seguito l'immagine come *Texture Image* con mappatura *UV* (con l'*UV Layer* creato mediante *unwrap* all'inizio di questo tutorial).



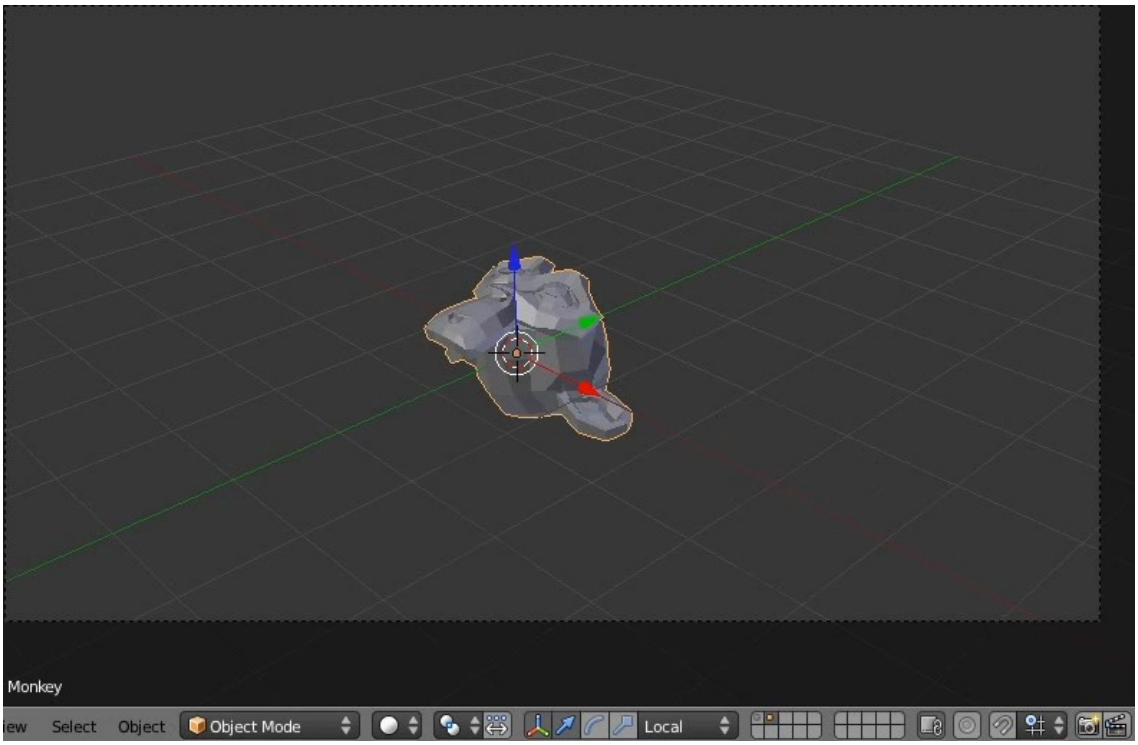
Il risultato ottenuto, da salvare su disco

Nella finestra Editor delle immagini clicchiamo sull'icona *Browse Image to be Linked* e scegliamo la voce "*Viewer Node*" dal menù che apparirà; a questo punto, è sufficiente salvare tale immagine con *Image - Save as Image*, nell'header della finestra.

* * *

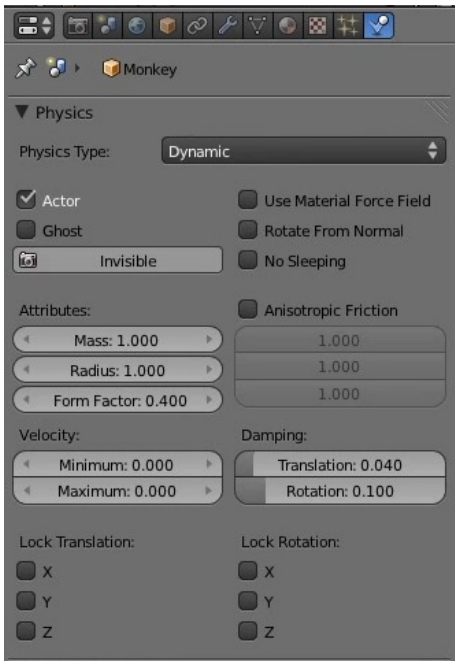
Inserire oggetti nella scena durante il gioco nel BGE 2.5

Questo brevissimo tutorial sul *Game Engine*, il motore di gioco di Blender 3D 2.5, nasce da una richiesta di un utente del Forum che chiedeva come poter inserire oggetti nella scena durante il gioco: un'operazione fondamentale per realizzare sparatutto, *platform game* e altre applicazioni.
Il tutorial è stato realizzato con la versione 2.58 stabile di Blender 3D.
In realtà, l'operazione non è particolarmente difficile: bisogna solo tenere a mente che un prototipo dell'oggetto da inserire al volo nella scena dev'essere posizionato in un altro *Layer*, non visibile durante l'esecuzione del gioco. Facciamo finta di voler generare alcune copie di *Suzanne* alla pressione della barra spazio, generando tali copie a partire da un cubo; in un videogioco, l'oggetto generatore potrebbe essere ad esempio il protagonista che lancia degli oggetti, un fucile che spara dei proiettili e così via. Lasciamo il cubo nel primo *Layer* (quello che verrà mandato in esecuzione), quindi spostiamoci nel secondo *Layer* della scena e inseriamo una copia di *Suzanne*.



Un'istanza di Suzanne nel secondo Layer; nel primo Layer è presente il cubo di default

Con tale elemento selezionato, apriamo la scheda *Physics* nella *Properties Window* (il motore dev'essere chiaramente *Blender Game*, nel selettore *Engine*, all'interno della finestra *Info*); cambiamo la natura di *Suzanne* da *Static*, voce selezionata di default, in *Dynamic* o *Soft* o *Rigid Body*, con casella *Actor* selezionata, visto che la copia dovrà essere "lanciata" nella scena e, quindi, dovrà muoversi all'interno della stessa.

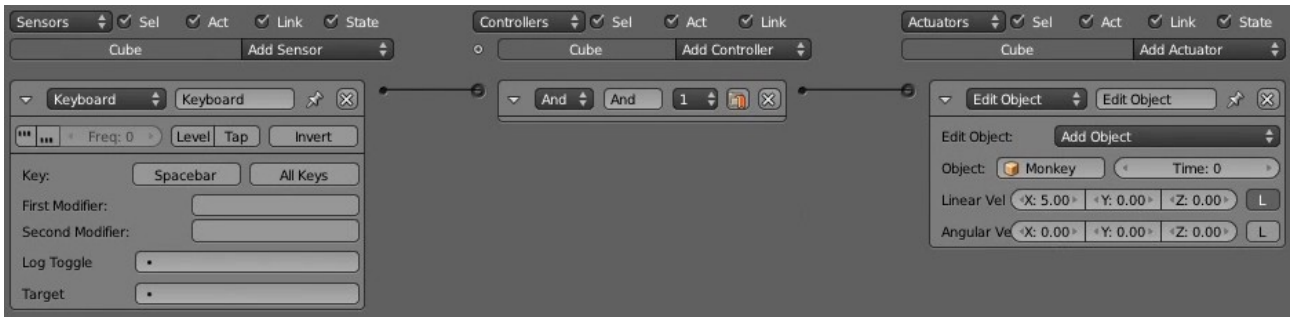


Le nuove impostazioni Physics per Suzanne

Torniamo al primo *Layer*. Volendo, possiamo impostare "*No Collision*" per il cubo generatore, nella scheda *Physics* dell'oggetto.

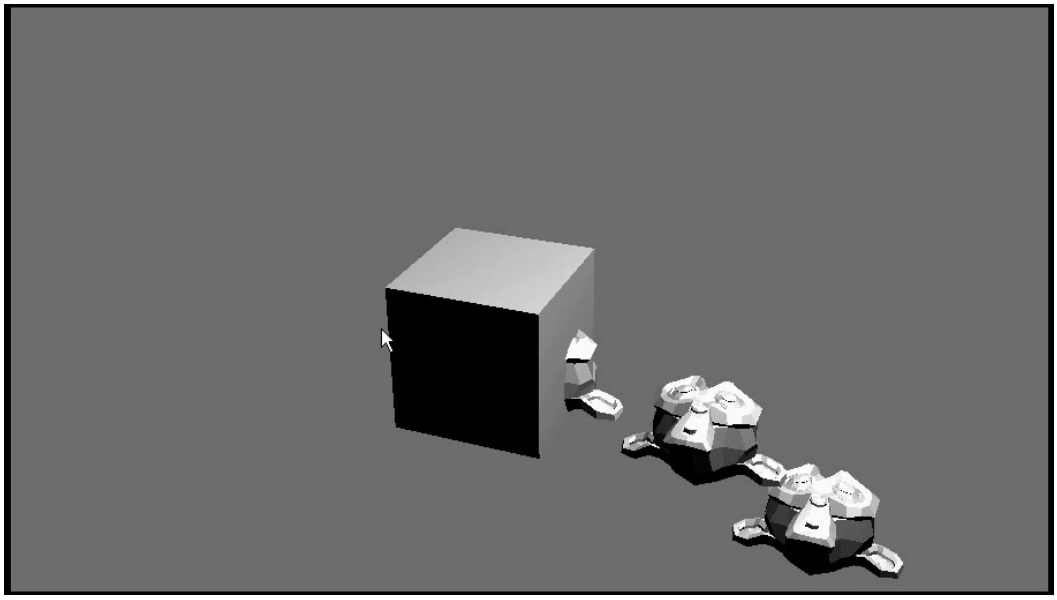
Il sensore, il controller e l'attuatore necessari per realizzare l'effetto vanno associati all'oggetto generatore, nel nostro caso il cubo, per cui con tale mesh selezionata cambiamo una finestra in *Logic Editor*, aggiungiamo un sensore per l'evento di attivazione (nel nostro esempio, *Keyboard* sul tasto *barra spazio*), un controller *AND* e un attuatore *Edit Object*, collegando i tre elementi.

Nell'attuatore, lasciamo la modalità *Add Object*, selezionata di default, dopodiché scriviamo il nome dell'oggetto da generare (nel nostro caso, *Monkey*) e valori di velocità e rotazione iniziali per il "lancio", selezionando la casella *L* se vogliamo che le coordinate si riferiscano al sistema di riferimento locale dell'oggetto generatore, come avviene di solito.



Sensore, controller e attuatore per il cubo (l'oggetto che "spara")

È sufficiente avviare il gioco e premere il tasto di attivazione dell'evento per osservare il risultato.



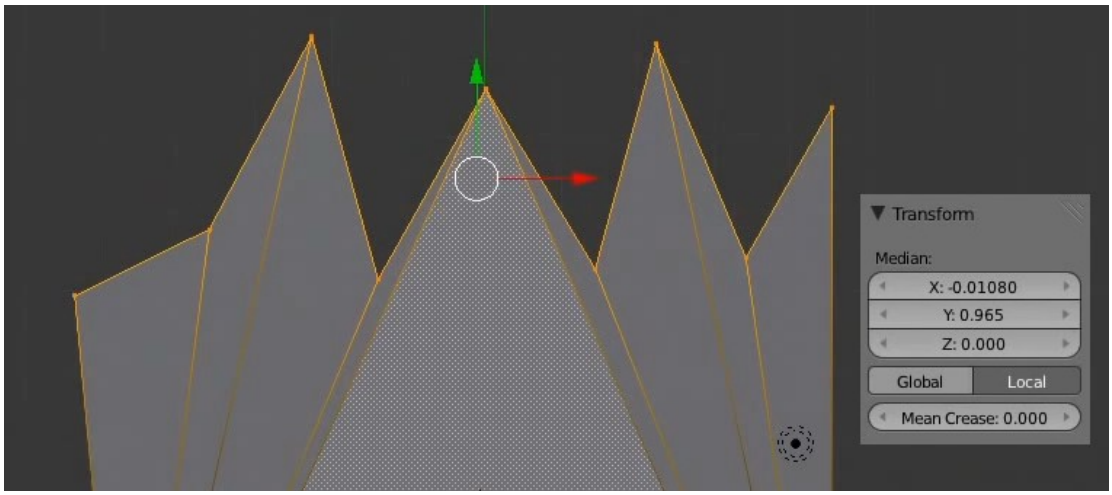
Screenshot dell'esecuzione del gioco

* * *

Modellazione: scaling a 0 per allineare vertici o appiattire mesh

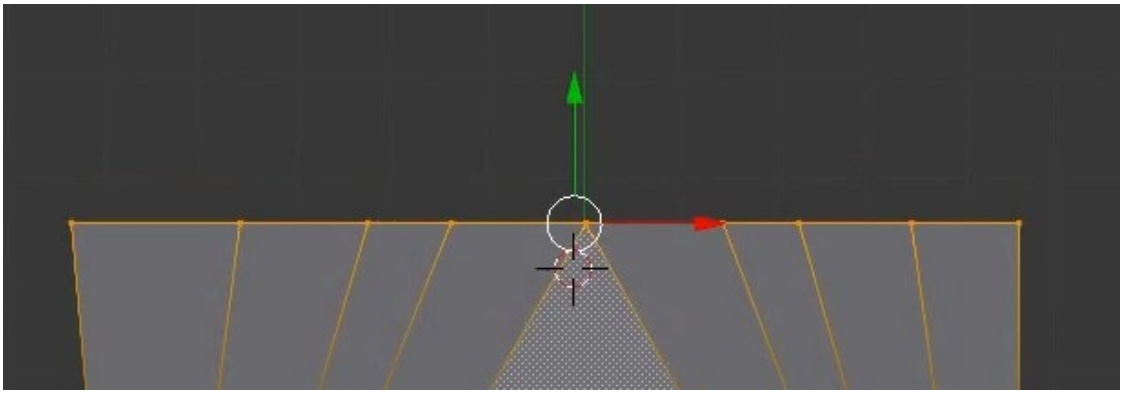
Questo è un piccolo "Tips & Tricks" di modellazione in Blender 3D 2.5, realizzato con la versione 2.58 stabile di Blender 3D: lo scaling a 0 su un asse in Edit Mode per allineare dei vertici o "appiattire" una mesh su un piano, una di quelle cose "banali" alle quali lì per lì non si pensa subito.

Iniziamo col primo caso, ovvero l'allineamento di vertici: capita a volte di avere vertici di una mesh o parti di essa da allineare, ad esempio se abbiamo cancellato alcune facce con una selezione dritta; in questi casi, passando in Edit Mode e selezionando gli elementi, si avranno a disposizione, nella *Transform Window*, le informazioni sul *Median Point*, il punto medio della selezione, sia in coordinate globali che locali. Cambiare le coordinate del *Median Point*, anche su un solo asse, non fornisce tale coordinata ai vertici della selezione: si limita a spostarli in modo tale da posizionare il baricentro nel punto specificato.



Median Point, nella Transform Window

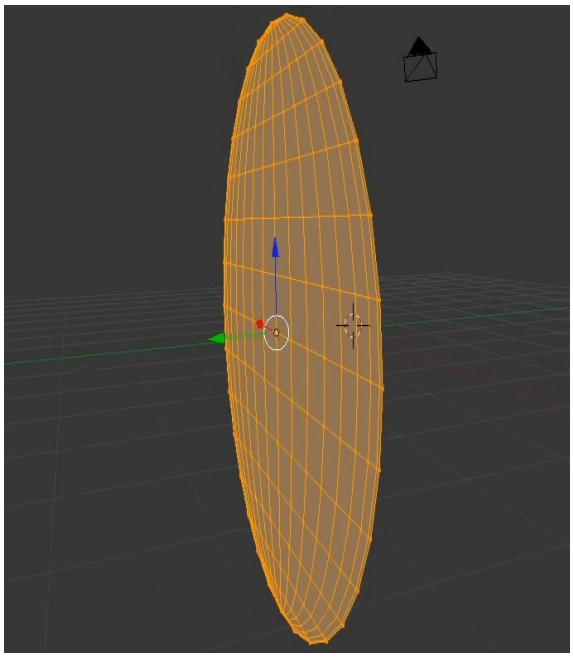
Poniamo il caso di aver messo il *Median Point* alla coordinata Y sulla quale allineare i vertici della selezione: per allineare tutti i vertici su tale coordinata, è sufficiente premere *S Y 0 Invio*, ossia: Scaling, vincolato sull'asse Y, con fattore di scaling 0, ossia nullo, in modo tale da schiacciare la selezione, facendo assumere a tutti i vertici quella coordinata Y.



Scaling a 0 sull'asse Y

A questo punto possiamo orientare tranquillamente la retta di vertici con *R*, se necessario.

Passiamo al secondo caso, in realtà molto simile al primo: appiattire una mesh.
Vediamo subito un esempio pratico: inseriamo una sfera nella scena, passiamo in Edit Mode, selezioniamone metà (occhio allo *Z-Clipping*, per selezionare correttamente anche i vertici nascosti) e cancelliamo tale metà. Se ad esempio abbiamo visto la sfera lateralmente, con l'asse Y che andava da sinistra a destra, e abbiamo cancellato la metà destra, per appiattire la sfera dobbiamo allinearne i vertici secondo la coordinata Y media, per cui selezioniamo tutti i vertici e digitiamo *S Y 0.0 Invio*.



Scaling a 0 di una mesh (in questo caso, una semisfera scalata a 0 lungo l'asse Y)

Ho eliminato metà sfera per evitare di creare vertici duplicati; ovviamente, mi è andata bene perché la sfera è simmetrica, ma per risolvere il problema dei vertici duplicati con mesh più complesse, che non possono essere ridotte così, si può utilizzare *Remove Doubles* con un basso valore di soglia.

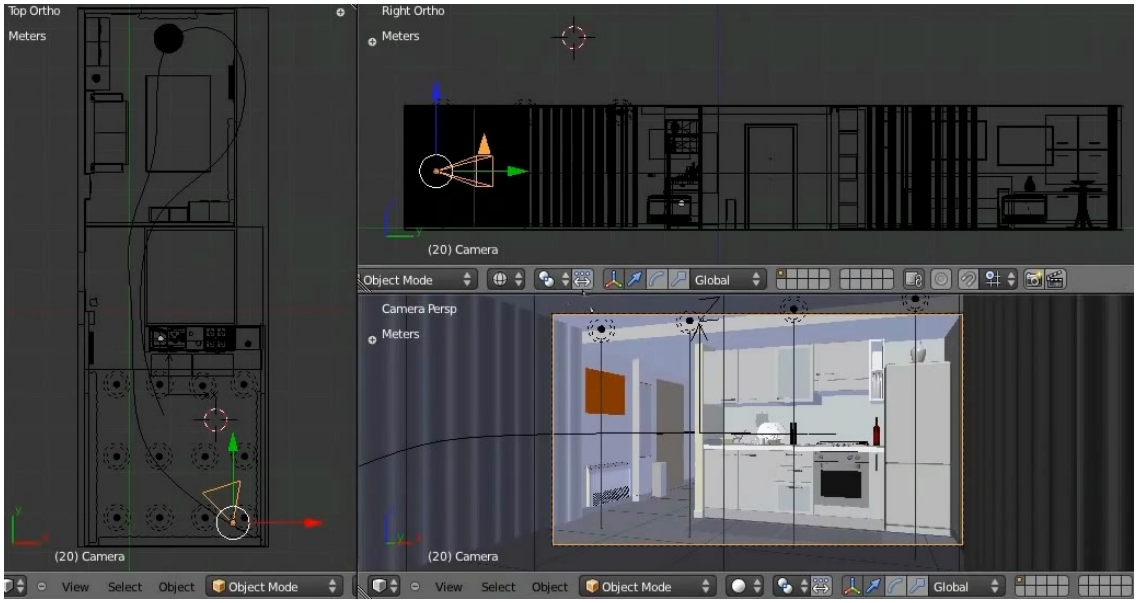
* * *

Rendering della 3D View con OpenGL Active Viewport

In questo brevissimo tutorial parleremo della funzionalità "*OpenGL Render Active Viewport*", che consente di realizzare screenshot o filmati della scena visibile in una finestra *3D View*.
Il tutorial è stato realizzato con la versione 2.59 di Blender 3D.

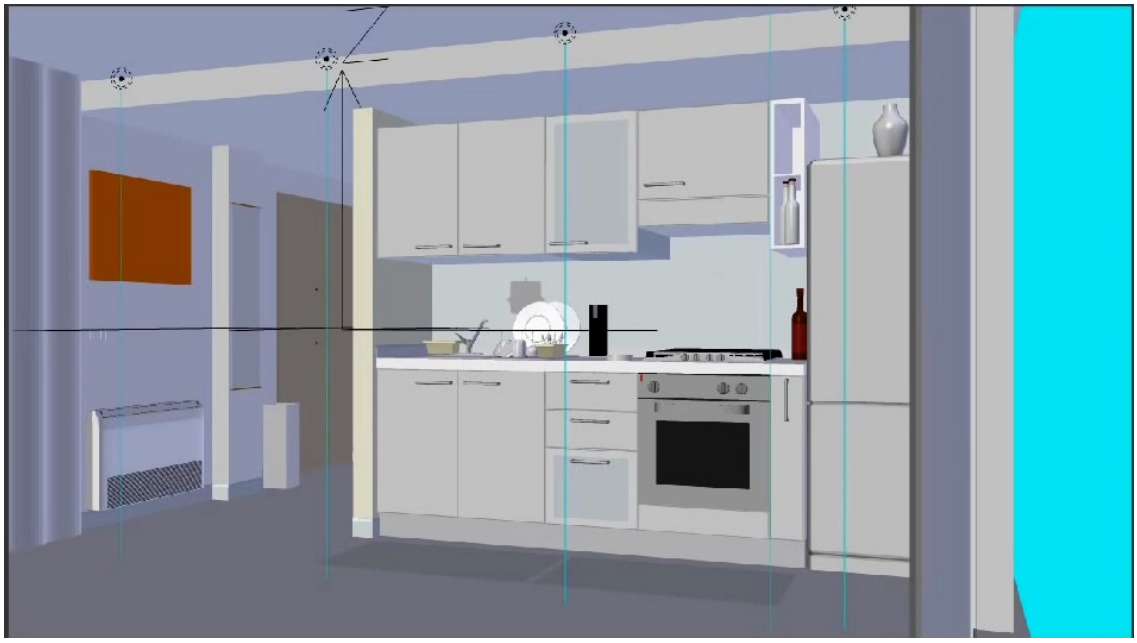
I filmati vengono realizzati sfruttando le *API OpenGL* di visualizzazione della scena, quindi questo particolare "rendering" viene realizzato dalle schede video, non dal processore, con tutte le conseguenze del caso.

Si tratta di rendering interessanti per le previsualizzazioni, ad esempio in campo architettonico o ingegneristico, per avere un'anteprima degli spazi e/o delle animazioni che verranno realizzate, esaminando accuratamente i tempi, oppure per verificare le *Normali* delle facce, l'ombreggiatura e la mappatura delle Textures *UV Mapped*, visto che la modalità di *shading Textured* delle *3D View* consente di esaminare questi aspetti.



La scena utilizzata per questo tutorial

Renderizzare la schermata corrente è semplicissimo: è sufficiente fare click sull'icona "*OpenGL Render Active Viewport*" col simbolo della macchina fotografica, penultima icona nell'header di una *3D View*. La scena verrà renderizzata proprio come la vediamo, nel senso che saranno visibili, ad esempio, i bordi evidenziati delle mesh selezionate, la linea nera della *Lamp*, il cerchietto rosso che identifica il baricentro degli oggetti selezionati, eccetera...; fortunatamente, non vengono renderizzati i manipolatori.



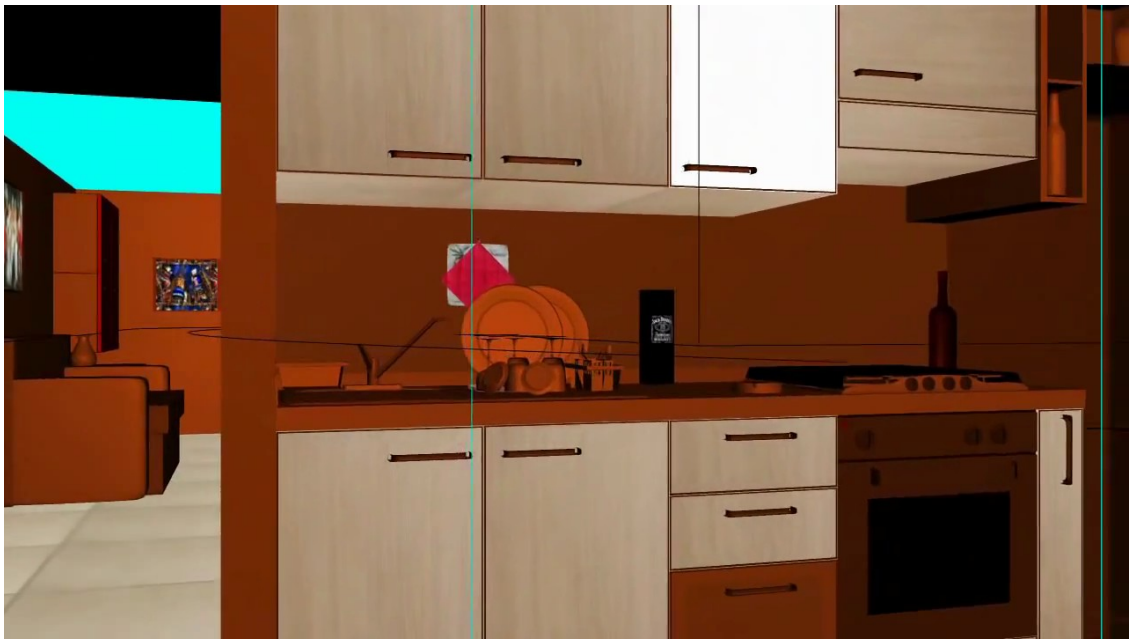
Il risultato ottenuto con un click sull'icona "OpenGL Render Active Viewport", nell'header di una 3D View

Il rendering verrà "depositato" in una finestra *UV - Editor delle immagini* come un vero e proprio rendering effettuato col *Blender Render* e con le impostazioni di risoluzione specificate nella finestra *Render*, all'interno della *Properties Window*.

È possibile renderizzare con questa modalità anche le animazioni, secondo la *Timeline* della scena (di default, 250 frames), ancora una volta con la risoluzione data da *Render Dimension* e con il formato e il percorso di output specificati in *Render Output*; per avviare questo tipo di rendering, cliccare sull'icona "*OpenGL Render Active Viewport*" col simbolo del *ciak cinematografico*, ultima icona a destra nell'header di una *3D View*.

Altro fotogramma di un'animazione ottenuta velocemente con OpenGL Render Active Viewport

Dal rendering realizzato in modalità *Textured* risulta evidente la presenza di eventuali *Normali* non impostate correttamente; dopo averle individuate, si può procedere alla correzione per passare infine al rendering vero e proprio della scena o dell'animazione.



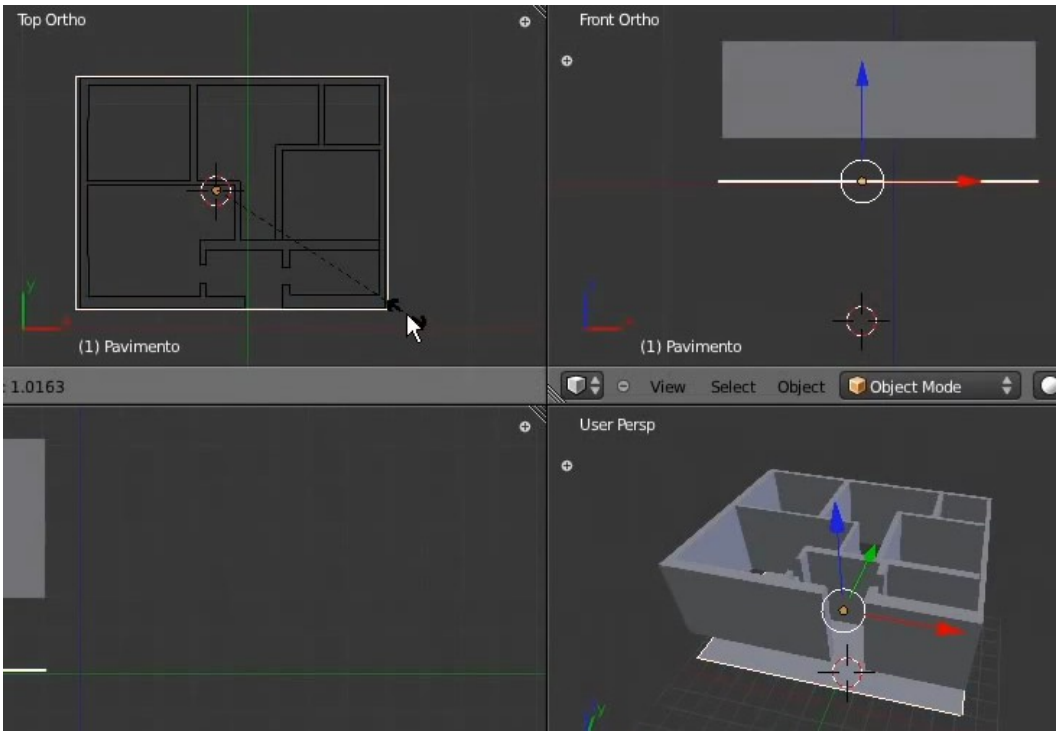
Fotogramma del rendering realizzato in modalità Textured

* * *

Floor per posizionare oggetti su pavimenti o pareti

Questo è un piccolo "Tips & Tricks" di modellazione in Blender 3D 2.5, riguardante l'utilizzo del *Constraint Floor* (lett.: pavimento) per poggiare correttamente un oggetto su un altro; si tratta di un vincolo utile, ad esempio, per posizionare degli oggetti su un pavimento o una parete.
Il tutorial è stato realizzato con la versione 2.59 di Blender 3D.

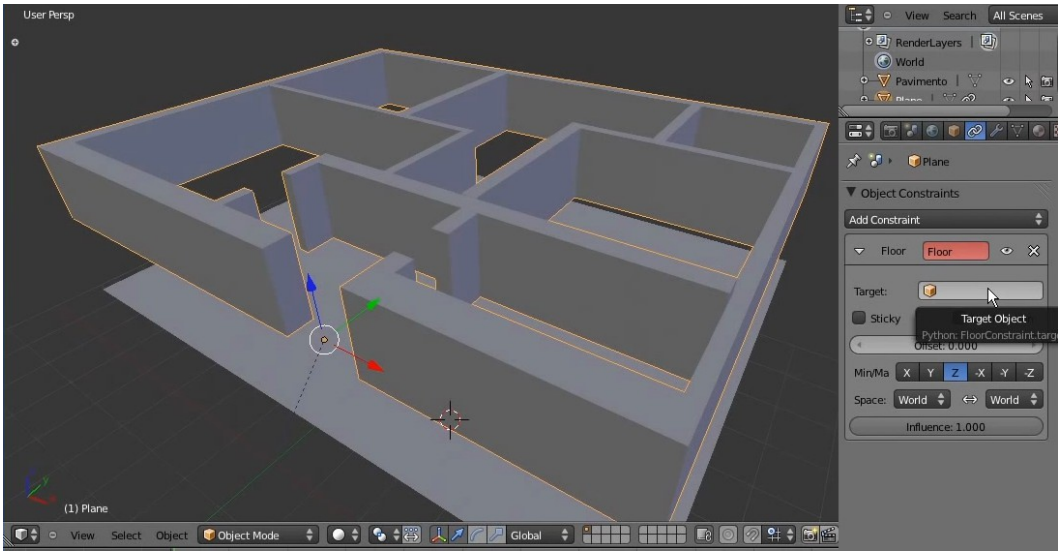
Supponiamo di avere una mesh rappresentante le pareti di una casa, come nell'immagine seguente.



La scena utilizzata per questo tutorial

Vogliamo posizionare questa struttura sul Plane rappresentante il pavimento della casa; anziché cercare di poggiare la mesh "a occhio" o basandoci su *snap* o altri strumenti, dotiamola di un *Constraint* di tipo *Floor* aggiungendo tale elemento nella scheda *Constraint* dell'oggetto, all'interno della *Properties Window.*

Qui dobbiamo specificare l'oggetto sul quale poggiare la mesh, indicandone il nome nel campo *Target*. La mesh (in questo caso, le pareti) non verrà spostata automaticamente: dobbiamo muoverla avvicinandola al *Target*, fino a far entrare in gioco il vincolo.

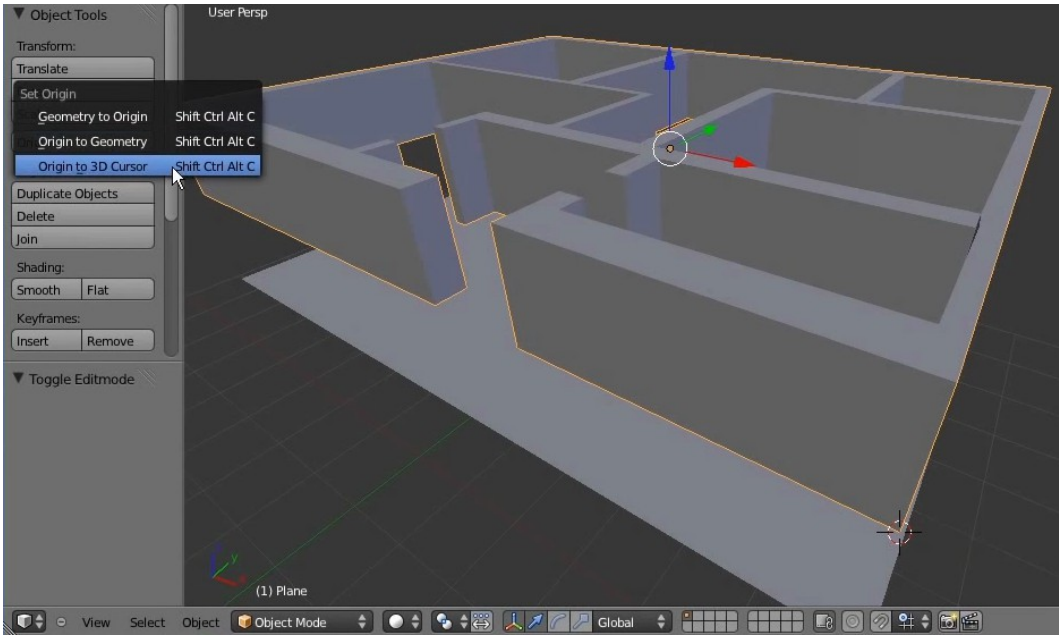


Utilizzo del Constraint Floor

Piccola sorpresa: *Floor* blocca il movimento della mesh attraverso il *Target* in base alla posizione dell'*Origin* della mesh, quindi quest'ultima potrebbe penetrare per un certo tratto nel *Target*.

Possiamo risolvere questo problema in due modi: o mediante il campo *Offset* del *Constraint* oppure cambiando l'*Origin* della mesh; nel primo caso, sposteremo la mesh con una distanza, tra la sua *Origin* e l'oggetto *Floor*, specificata appunto nel campo *Offset*, mentre per cambiare direttamente l'*Origin* dell'oggetto, lasciando a 0 il campo *Offset*, possiamo procedere come segue:

- passare in Edit Mode,
- selezionare un vertice della mesh che dovrà ritrovarsi poggiato sul Plane,
- spostare lì il *3D Cursor* con *SHIFT S – Cursor to Selected*,
- tornare in Object Mode,
- spostare l'origine in quel punto con *Set Origin to 3D Cursor*, richiamabile cliccando su *Origin nella Tool Shelf*, in una *3D View*.



Set Origin to 3D Cursor (in Object Mode, dopo aver posizionato il 3D Cursor nel punto desiderato)

È possibile specificare un *Vertex Group* dell'oggetto "pavimento" per specificare su quale faccia o gruppo di facce dell'oggetto dovrà basarsi il *Constraint* per applicare l'effetto; ad esempio, possiamo inserire una faccia della mesh in un gruppo "*Parete A*" per vincolare una mesh, come un quadro, solo ad una porzione delle pareti.

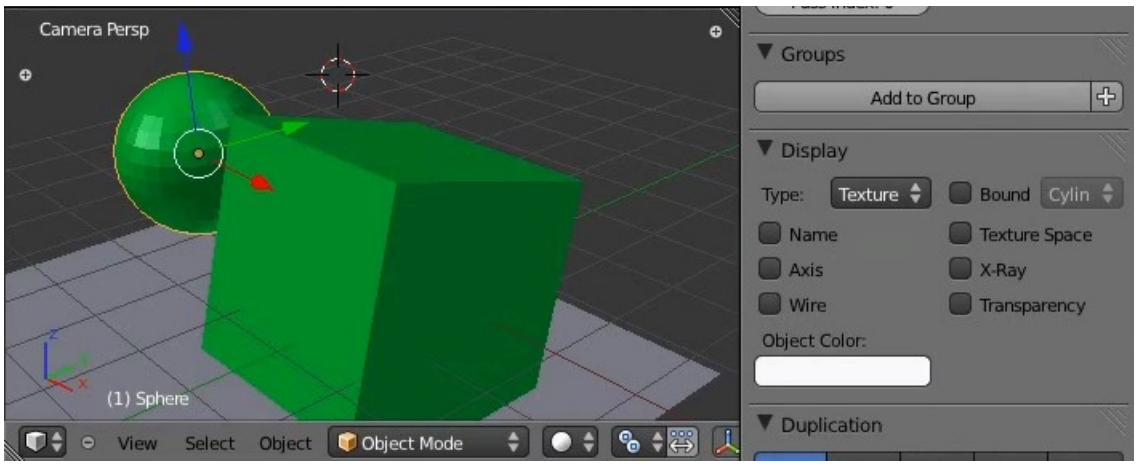
È possibile inserire più *Constraint Floor*, cosa che può tornare utile ad esempio per posizionare una vasca da bagno in un angolo: in questo caso, infatti, dovremo cambiare l'*Origin* posizionandolo nel vertice d'angolo in basso ed utilizzare due *Floor* per le pareti (che dovranno avere due *Vertex Group* distinti per le due facce in questione) e uno per il Plane rappresentante il pavimento della casa.

* * *

Tips and Tricks: le voci di Object Display

In questo brevissimo tutorial parleremo di alcune funzioni poco note dell'interfaccia utente di Blender 3D, ossia quelle raggruppate nella sezione *Display* della scheda *Object*, all'interno della *Properties Window*. Il tutorial è stato realizzato con la versione 2.59 stabile di Blender 3D.

Selezioniamo un oggetto della scena, dotandolo di un Material di base (che ci tornerà utile per un paio di casi) ed esaminiamo la sezione *Display* di *Object*. Le voci qui presenti si riferiscono a caratteristiche di visualizzazione dell'oggetto nelle *3D View*, non in fase di rendering.



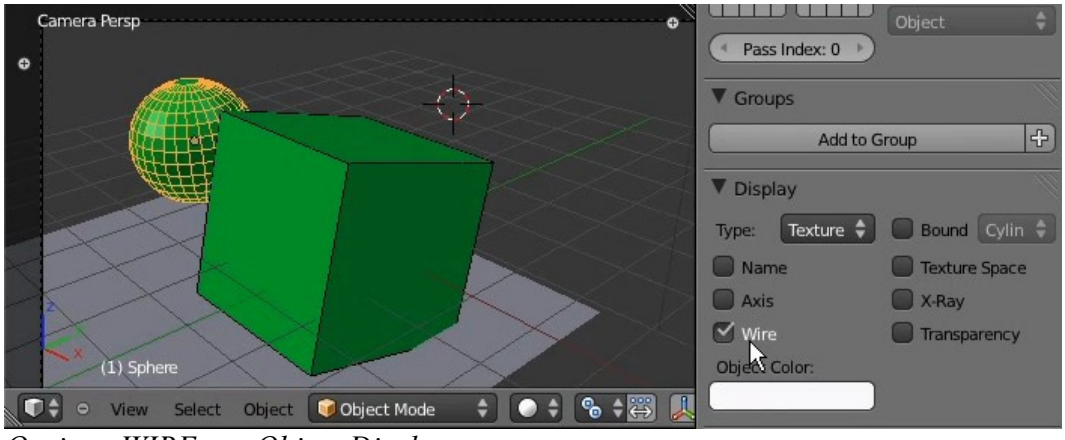
Le voci della sezione Object - Display, per un oggetto selezionato

Con il selettore *Maximum Draw Type* possiamo impostare il grado massimo di visualizzazione dell'oggetto nelle finestre *3D View*; come sappiamo, nelle *3D View* possiamo scegliere se visualizzare gli oggetti come *Bounding Box*, struttura (*Wireframe*), *Solid* e *Textured*, mediante il selettore *Viewport Shading*.

Questa scelta viene applicata a tutti gli oggetti della *3D View* corrente, ma se per esempio vogliamo visualizzare gli oggetti con modalità Textured tranne uno, del quale vogliamo visualizzare ad esempio il *Bounding Box* o la struttura, allora possiamo scegliere, per tale oggetto, le voci *Bounding Box* o *Wireframe* nel selettore *Maximum Draw Type, in Display*: ecco il significato del nome "massimo grado di resa" per l'oggetto. Ripeto che queste opzioni valgono solo per la visualizzazione nella *3D View*, non in fase di rendering.

Le checkboxes *Name*, *Axis*, *Texture Space* e *Transparency* hanno significati intuitivi: *Name* consente di visualizzare il nome dell'oggetto accanto alla sua *Origin*, nella *3D View*, e in questo stesso punto è possibile mostrare una rappresentazione del sistema di riferimento *LOCALE* dell'oggetto scegliendo la casella *Axis*, mentre con *Texture Space* possiamo visualizzare lo spazio di mappatura delle Textures associato all'oggetto e con *Transparency* possiamo visualizzare l'oggetto come semitrasparente se il suo Material è provvisto, appunto, di *Trasparenza* con valori *Alpha* minori di 1. La visualizzazione della *Trasparenza* in tempo reale funziona con tutte e tre le modalità *Mask*, *Z-Transp* e *Raytrace*, ma attenzione: essa viene resa secondo la modalità *Z-Transp*, ignorando quindi eventuali effetti di *Raytracing* di una trasparenza *Raytrace* come l'indice di rifrazione o il grado di *Glossiness* dell'effetto.

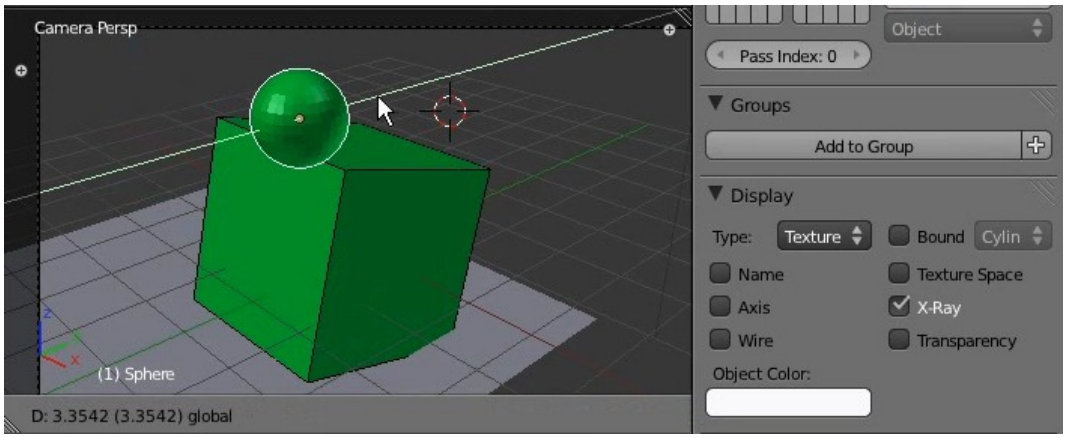
Torniamo alle voci di *Display*: attivando la casella *Wire* possiamo visualizzare la struttura *Wireframe* della mesh anche se nelle *3D View* la stiamo visualizzando in maniera *Solid* o *Textured*; si tratta quindi di un'opzione utile per tenere d'occhio la struttura della mesh mentre la si modella.



Opzione WIRE per Object Display

Attivando *X-Ray* possiamo porre l'oggetto in primo piano, visualizzandolo cioè interamente anche quando è coperto, anche solo parzialmente, da altri oggetti della scena; è un'opzione utile per tenere

d'occhio un oggetto se lo si "perde" d'occhio, ad esempio, durante una simulazione fisica, non tanto in fase di posizionamento nella scena, proprio perché non consente di capire la posizione relativa dell'elemento.



Opzione X-RAY

Attivando *Bounds*, possiamo scegliere che tipo di *Bounding Box* visualizzare per una mesh, ad esempio se di forma sferica, conica, cilindrica, ecc...

Per quanto riguarda il campo *Object Color*, dobbiamo aprire una piccola parentesi: a ciascun oggetto è possibile associare un colore di base, detto appunto *Colore Oggetto*, che non è il colore diffuso del Material; in un certo senso, è un colore ancora più "primitivo", nel senso che è associato alle primitive geometriche dell'oggetto, come avviene ad esempio in *Java 3D* o *OpenGL*.

Come detto nel mio *ebook sui Materials in Blender*, di default la modulazione di questo colore con il colore del Material ed eventualmente della Texture associata ad una mesh è, di default, disattivata, ma può essere attivata selezionando la casella *Object Color* nella sezione *Options* del Material; la scelta del *Per-Object Color* avviene, invece, proprio mediante la casella *Object Color* in *Object Display*. A differenza delle altre voci di *Display*, questa ha un effetto anche in fase di Rendering, se la voce *Object Color* di Material è selezionata, in quanto questo colore influenzerà quello diffuso dell'oggetto che lo possiede sia nella *3D View* che in fase di rendering.

Può sembrare strano voler modulare il colore di un Material con un altro colore, ma un motivo è, ad esempio, il fatto che un Material può essere associato a più mesh; volendo modulare leggermente il colore di un oggetto rispetto ad un altro, mantenendo lo stesso Material senza dover cambiare il colore diffuso, si può ricorrere appunto all'Object Color.

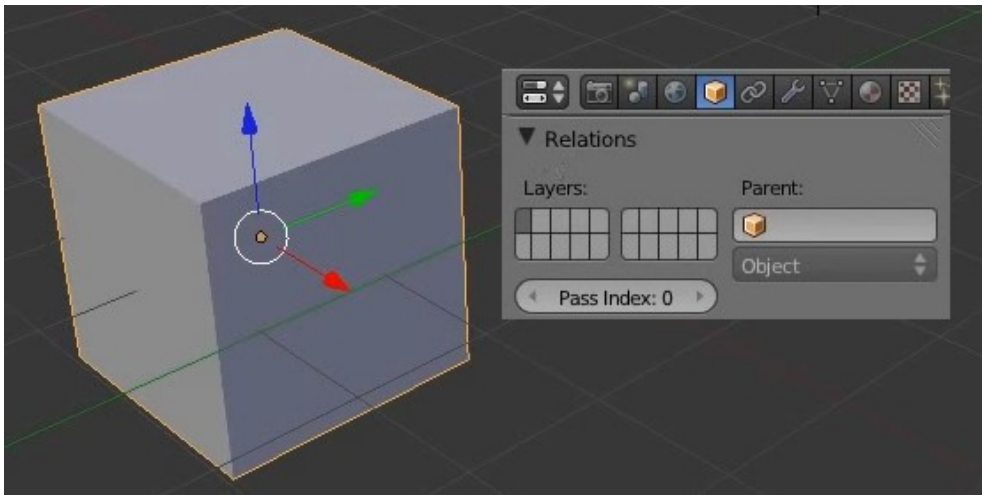
* * *

Animare le Layers Relations per far apparire e scomparire le mesh

In questo brevissimo tutorial parleremo di una funzionalità poco nota di Blender: *Layers Relations*, che consente di definire in quali *Layers* (e, soprattutto, quando) dovrà essere visualizzato un oggetto. Il tutorial è stato realizzato con la versione 2.59 stabile di Blender 3D.

Ho detto "quando deve essere visualizzato un oggetto" perché *Layers Relations* è, come la maggior parte degli elementi di Blender 3D, animabile, ed è possibile far comparire o scomparire un oggetto in certi istanti di un'animazione semplicemente inviandolo in un *Layer* da non renderizzare; procediamo comunque con ordine.

Selezioniamo un oggetto nella scena 3D.
Le caselle *Layers Relation* si trovano nella scheda *Object*, all'interno della *Properties Window*, per ciascun oggetto selezionato, e richiamano lo schema delle 20 caselle, associate ad altrettanti *Layers* della scena, presente nell'header della *3D View*.

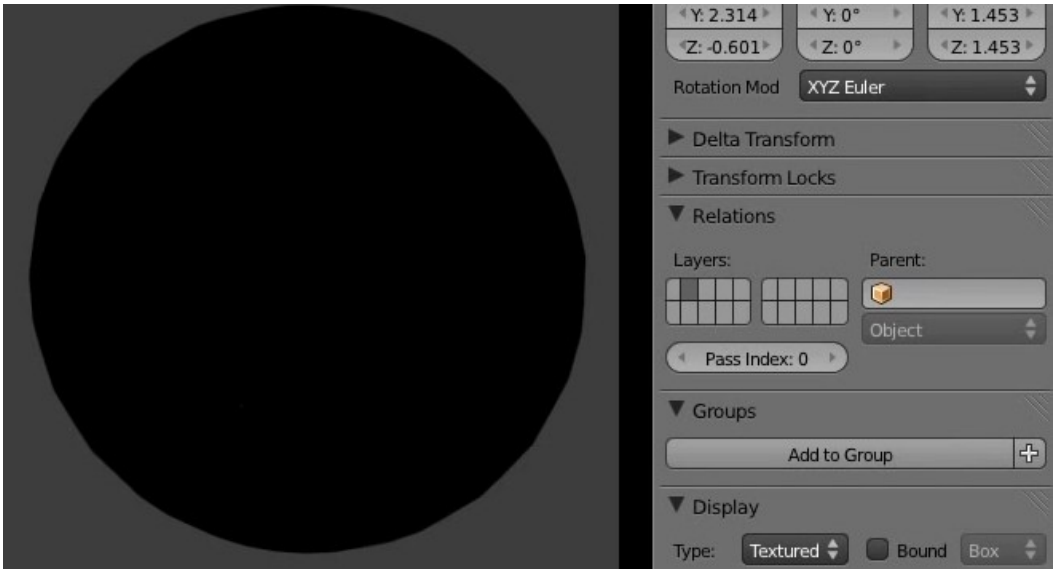


La casella Layer Relations, nella scheda Object, per l'oggetto selezionato

Il tooltip ci dice appunto che, se selezionata, ciascuna casella indica se l'oggetto si trova o meno nel relativo *Layer*. Questo gruppo è, come detto, animabile.

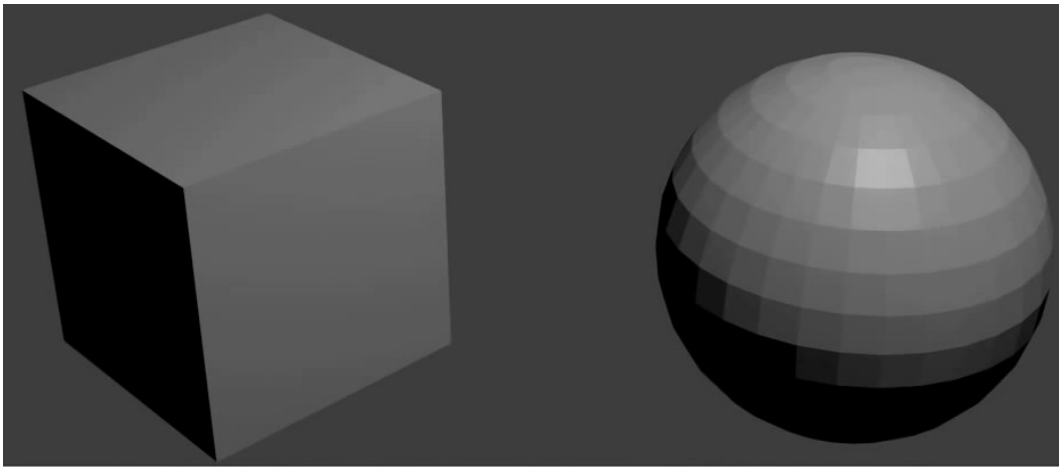
La distinzione con le caselle *Layer* presenti nella *3D View* è quindi la seguente: in quest'ultimo caso, le caselle selezionate indicano quali sono i *Layer* da visualizzare in fase di rendering (di default, solo il primo) e la selezione è multipla.

Spostiamoci ad esempio nel *Layer* 2 ed inseriamo un oggetto, ad esempio una sfera, dopodiché avviamo un rendering: vedremo solo l'oggetto presente nel *Layer* 2.



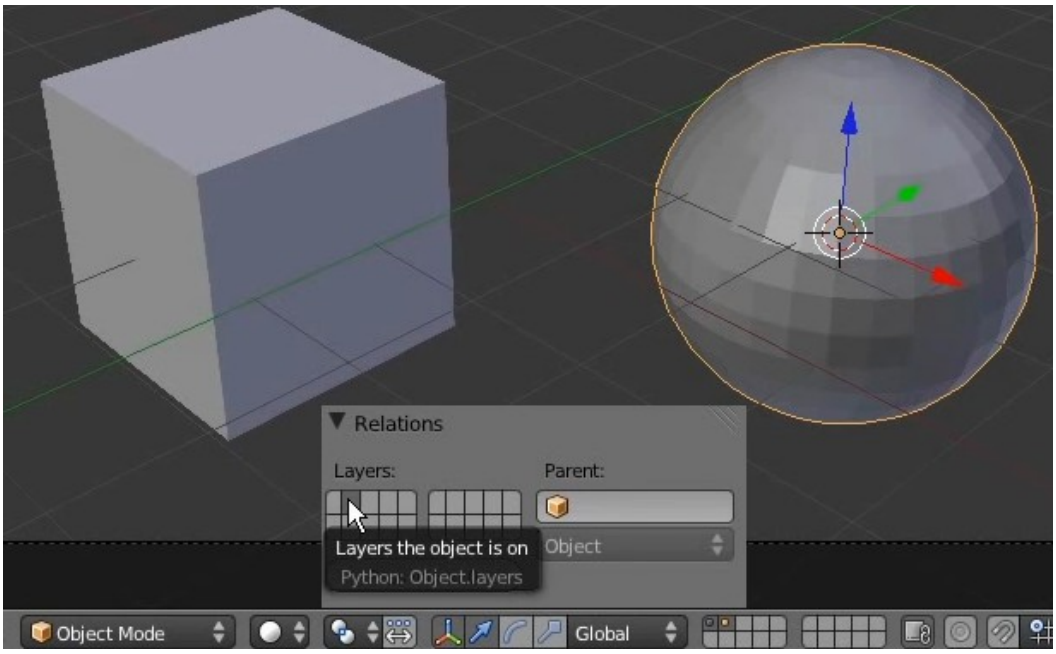
Il rendering del solo Layer 2, con la sfera appena inserita

Selezionando con una selezione multipla anche il primo *Layer*, con *SHIFT* e tasto sinistro del mouse, ed avviando un rendering, avremo entrambi gli oggetti renderizzati.



Il rendering dei due Layers

Con la sfera del secondo *Layer* selezionata, passiamo alla scheda *Object* ed esaminiamo la sezione *Layers Relations*: come possiamo notare, solo la casella del secondo *Layer* è selezionata, ad indicare appunto che l'oggetto corrente si trova solo nel secondo *Layer*.



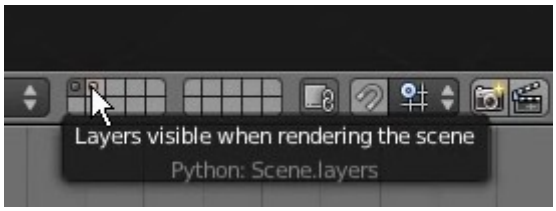
La sezione Layers Relations della sfera

Possiamo creare una copia linkata di un oggetto in un altro *Layer* con *M*, in Object Mode in una *3D View*, ed effettuando una selezione multipla nella casella che apparirà a video; Blender selezionerà automaticamente le relative caselle in *Layers Relations*, ad indicare appunto che un oggetto è presente in più *Layer*.

Possiamo inserire o togliere un oggetto da o in un determinato *Layer* direttamente mediante *Layer Relations*, ad esempio selezionando una casella o, se questa era già selezionata (e quindi l'oggetto era già presente nel relativo *Layer*), deselezionandola.

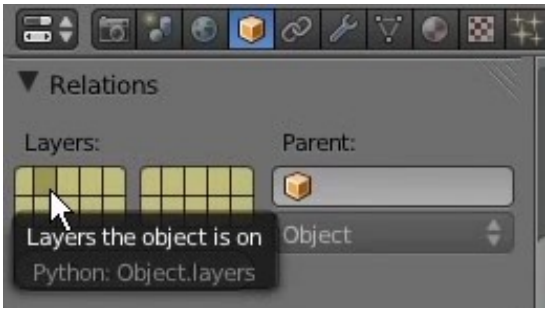
Dal momento che *Layer Relations* è animabile, se abbiamo la necessità di far scomparire un oggetto dalla scena possiamo registrarne la presenza nei *Layer* da renderizzare solo nei frames nei quali ci serve, inviandolo ad un *Layer* non renderizzabile negli altri; vediamo come fare.

Supponiamo di voler renderizzare solo il primo *Layer*, selezionando quindi solo la prima casella nel gruppo di 20 presenti nell'header della *3D View*.



Selezioniamo solo la prima casella (il primo Layer) nel gruppo di 20 nell'header di una 3D View

Supponiamo di voler renderizzare la sfera, che si trova nel secondo *Layer*, solo dal frame 50 al frame 150. Selezioniamo la sfera, andiamo al frame 49, apriamo la scheda *Object* nella *Properties Window* ed inseriamo un keyframe per *Layer Relations* semplicemente ponendo il cursore del mouse sopra le caselle e premendo *I*.



Inserimento di un Keyframe per Layer Relations (premendo "I" mentre il cursore del mouse si trova sulle caselle)

I pulsanti cambieranno colore, ad indicare che è presente un keyframe. Passiamo al frame 50 e selezioniamo, in *Layer Relations* per la sfera, il primo *Layer*, spostando di fatto l'oggetto nel *Layer* da renderizzare, ed inseriamo un nuovo keyframe, ancora una volta con *I*, per registrare la nuova configurazione.

Andiamo adesso al frame 150, dove vogliamo far scomparire l'oggetto.

Dal momento che *Layer Relations* crea *chiavi IPO* di tipo on/off, come quelle associate alle checkboxes (e non curve di interpolazione come quelle associate a valori numerici), non c'è bisogno di creare un keyframe al frame 149 per dire a Blender che in quell'istante la sfera è ancora al *Layer* 1: possiamo andare direttamente al *Layer* di cambiamento di stato, nel nostro caso il 150esimo.

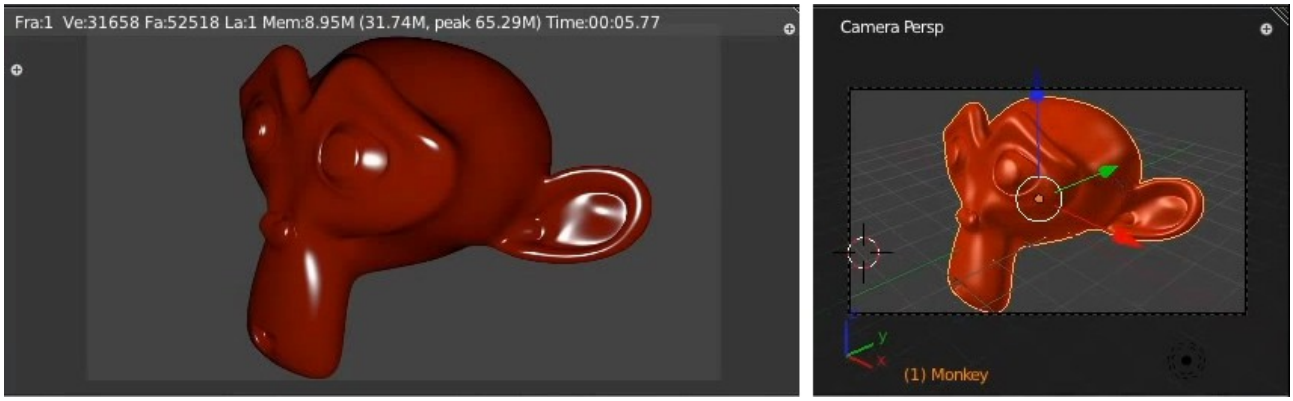
Il keyframe al frame 49 era invece necessario per memorizzare la configurazione di partenza, ma avremmo potuto inserirlo tranquillamente al frame 1, 15, 32... qualsiasi frame tra 1 e 50: alla fine, Blender avrebbe mutato il *Layer* della sfera solo al frame 50. Selezioniamo quindi la seconda casella in *Layer Relations*, con una selezione singola, in modo da togliere la sfera dal primo *Layer*, ed inseriamo un nuovo keyframe. Abbiamo appena realizzato la nostra animazione: la sfera sarà visibile nel primo *Layer*, quello da renderizzare, dal frame 50 al frame 150, mentre negli altri frames resterà per così dire "parcheggiata" in un *Layer* non visibile.

* * *

Effetti Glow sui riflessi speculari col Compositing

In questo brevissimo tutorial parleremo dell'applicazione di effetti in Post-Produzione, mediante il Compositing dei Nodes di Blender 3D 2.5, ad un solo *Render Pass* di rendering, ed in particolare al canale *Specular*, per mettere in evidenza mediante un bagliore *Glare* i riflessi speculari di un oggetto. Il tutorial è stato realizzato con la versione 2.59 stabile di Blender 3D.

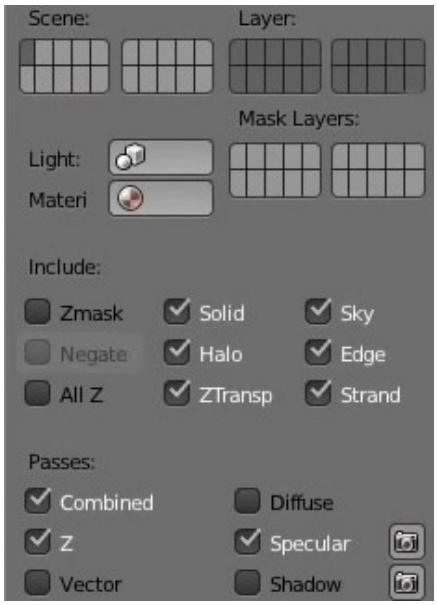
Al *Compositing dei Nodes in Blender 2.5* ho dedicato un *ebook*, dove vengono trattati anche altri temi come l'utilizzo di *ID Mask* per applicare gli effetti solo a determinati oggetti, per cui in questo breve tutorial non descriverò in dettaglio i Nodi, limitandomi a mostrare la configurazione nell'Editor per ottenere l'effetto voluto. Nella scena abbiamo *Suzanne* provvista di un Material di base, con colore diffuso rosso e un ombreggiatore speculare *WardIso* con intensità 1 e colore bianco; la fonte di luce *Lamp* è posta dietro la telecamera, cosa che (insieme all'utilizzo di *WardIso*) creerà riflessi speculari evidenti su *Suzanne*, anche senza *Compositing*, come è possibile osservare lanciando un rendering con i *Nodes* disattivati.



Il rendering della scena utilizzata in questo tutorial

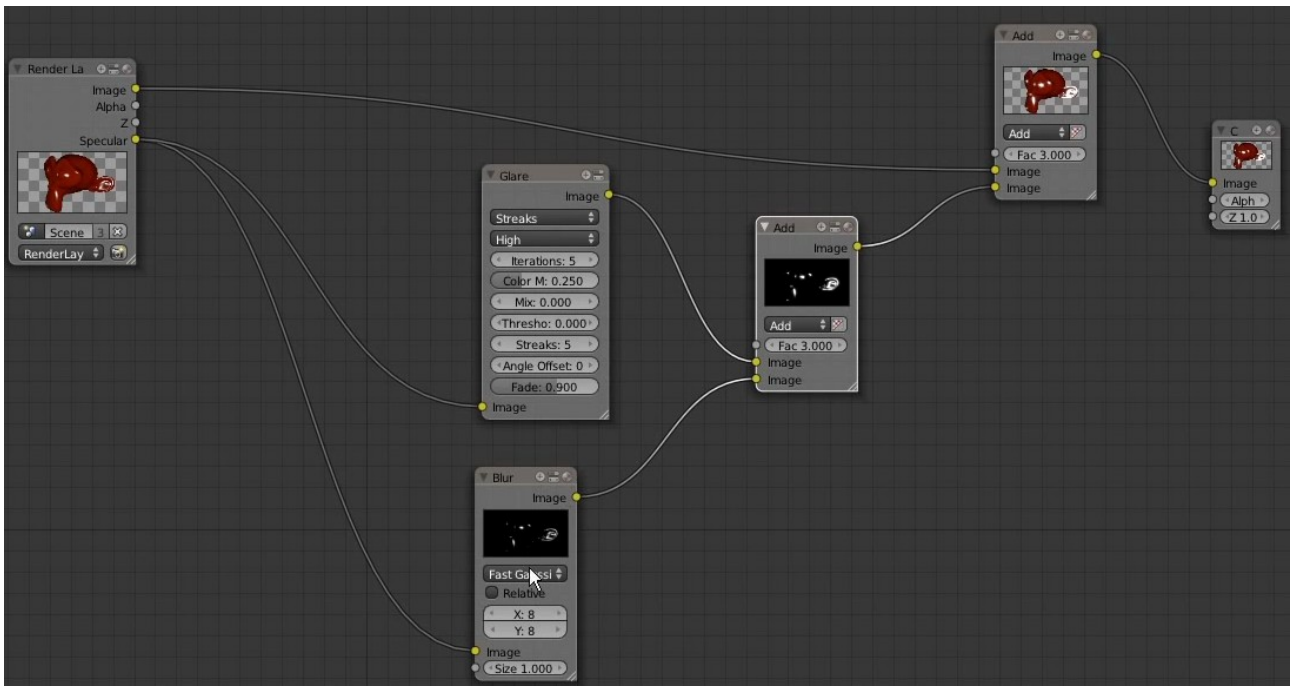
Ho creato una semplice animazione, una rotazione di 360 gradi intorno all'asse Z, per *Suzanne*, per generare il rendering che vedremo al termine del tutorial.

Per poter ricavare il *Render Pass Specular*, in modo da elaborare i soli riflessi speculari presenti all'interno di un rendering (per qualsiasi oggetto, a meno di non utilizzare *ID Mask* come filtro), dobbiamo selezionare la casella *Specular* all'interno del pannello *Layers* nella scheda *Render*, all'interno della *Properties Window*.



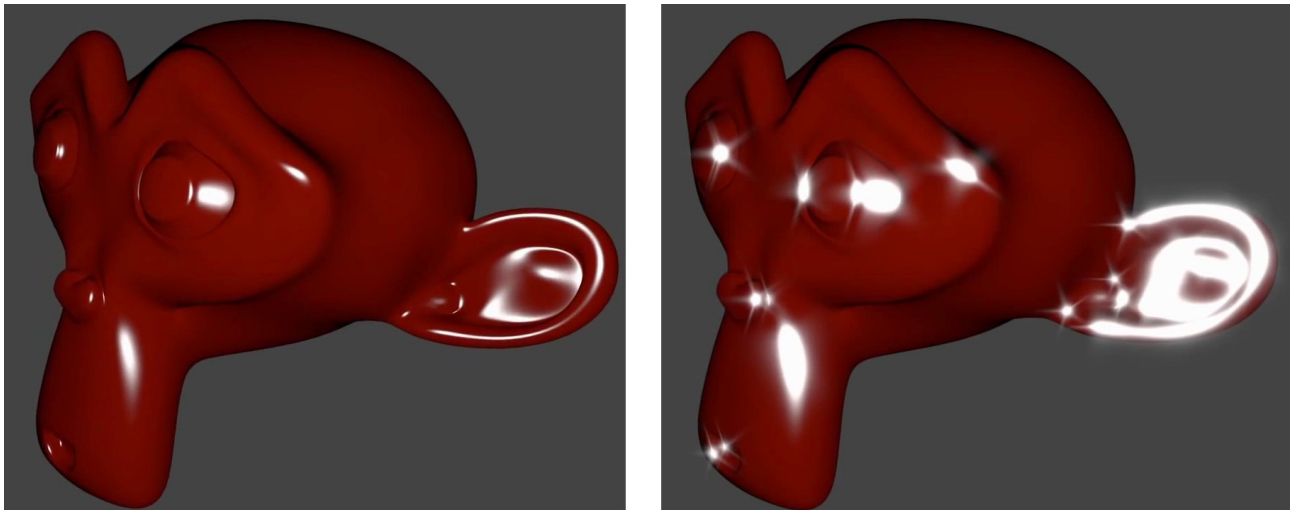
Attivare il Render Pass Specular, nella scheda Layers

Spostiamoci quindi in una finestra *Nodes Editor* per esaminare la configurazione dei Nodi: dal Nodo *Specular* di *Render Layers* escono due collegamenti diretti verso un Nodo *Blur* e un Nodo *Glare*, in modo da ottenere (dal nodo *Blur*) un *Soft Glow* dei soli riflessi speculari della scena e, dal Nodo *Glare*, un *Hard Glow*; gli effetti vengono quindi miscelati mediante un Nodo *Mix* in modalità *Add* e *Factor* 3. L'output di tale Nodo viene passato come secondo canale di input ad un secondo Nodo *Mix Add* con *Factor* 3, sommando quindi questo risultato parziale al rendering originale, evidenziando così i riflessi speculari; il risultato viene passato a *Composite* per il rendering finale.



Lo schema di Nodi di Compositing

I due screenshot seguenti mostrano *Suzanne* in rotazione, nel primo caso con i *Nodes* disattivati, nel secondo con il *Compositing* applicato. L'esempio mostrato qui è davvero banale e il rendering non è per niente entusiasmante, ma l'effetto, semplice da implementare, in scene più ricche e stimolanti consente di ottenere risultati davvero interessanti.



Due screenshot dell'animazione, nel primo caso con i Nodes disattivati e, nel secondo, con gli effetti di Compositing

* * *

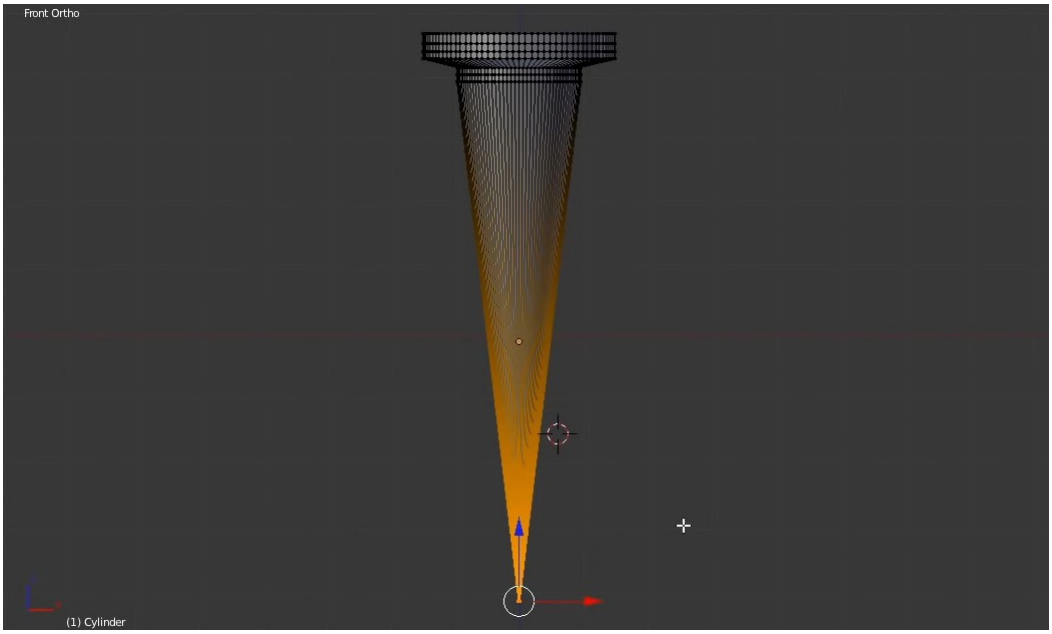
Modellare filettature e scalanature per viti, bulloni ed altro

In questo tutorial vedremo come realizzare la filettatura di viti, bulloni ed altri elementi in Blender.
Si tratta di un'operazione che apparentemente può sembrare difficile, ma in realtà è sufficiente utilizzare una Texture, che nei casi migliori può essere addirittura una Texture procedurale fornita di serie da Blender, ed anzi per il primo esempio vedremo proprio questo caso.
Il tutorial è stato realizzato con la versione 2.59 di Blender 3D.

Quello che dobbiamo fare è individuare le parti della mesh che dovranno presentare la filettatura, isolarle con un *Vertex Group*, dare loro una buona risoluzione in termini di vertici e facce ed applicare un modificatore *Displace*.

Il modifier *Displace* deformerà la mesh lungo le normali delle facce seguendo i valori di grigio di una *Texture*, ed in particolare "alzerà", per così dire, le parti più chiare, mantenendo basse o inalterate quelle scure e nere. Attraverso l'uso di una *Empty*, come vedremo, potremo traslare, ruotare o scalare la filettatura lungo la mesh.

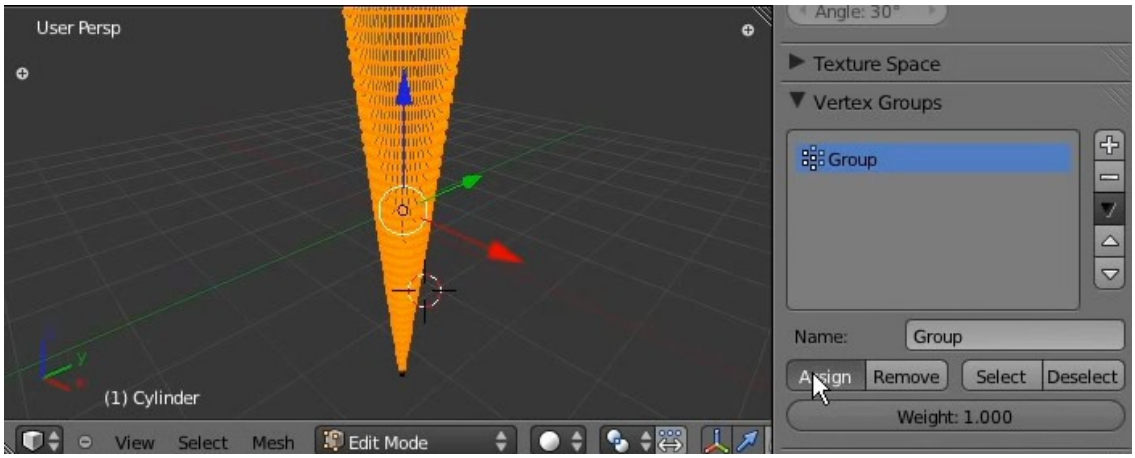
Partiamo dall'esempio della vite: inseriamo nella scena un cilindro con un elevato numero di vertici, ad esempio 96 o 128; avendo cura di disattivare lo *Z-Clipping*, in una finestra frontale o laterale selezioniamo una delle due basi del cilindro e scaliamola a un centesimo della dimensione iniziale, con *S 0.01*; non stiamo inserendo direttamente un cono perché, per ottenere una buona risoluzione della mesh, dobbiamo applicare Ring Cut (in modalità multi-taglio, con tagli equidistanti) e non possiamo utilizzarlo con un cono. Selezionando tutti i vertici, quindi, premiamo CTRL R, posizioniamo il mouse in modo da far apparire la linea di taglio orizzontale, ma non applichiamo subito il modificatore, ruotando invece la rotellina del mouse per far apparire nuove linee di taglio equidistanti; quando il loro numero (e la conseguente risoluzione spaziale) sarà sufficiente, applichiamo il modificatore con click del tasto sinistro del mouse.



Realizzazione del tronco di cono (cilindro scalato in una base), prima di effettuare i tagli equidistanti con CTRL R (Ring Cut)

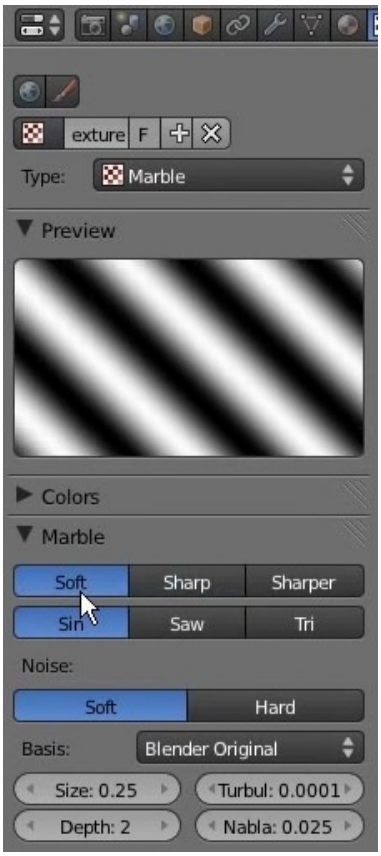
Estrudiamo un po' la base maggiore di quello che, in realtà, è un tronco di cono; se vogliamo, possiamo selezionare i vertici della base minore e premere *S 0* per rendere appuntita la vite.

A questo punto impostiamo un *Vertex Group* per le facce della mesh che dovranno presentare la filettatura, selezionandole magari in modalità *Face Select* in una 3D View; ricordo che, dopo aver creato il *Vertex Group*, è necessario cliccare su *Assign* per assegnare effettivamente le facce selezionate al *Vertex Group*.



Assegnare gli spigoli da influenzare con il Modifier Displace ad un Vertex Group creato per l'occasione

Adesso ci serve una Texture per definire la filettatura; per questo primo esempio, inseriamo nella scena una Texture procedurale di tipo *Marble*.



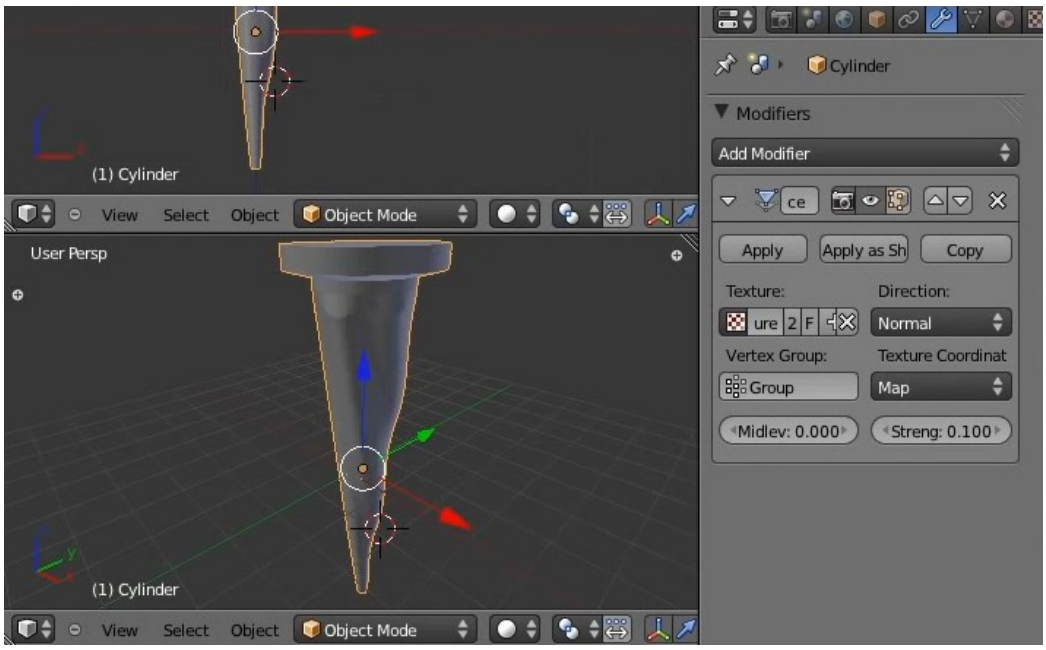
Le impostazioni della Texture MARBLE

Come spiegato nel mio *ebook sulle Textures in Blender 3D*, le bande delle Textures procedurali di tipo *Wood* e *Marble* rappresentano delle funzioni matematiche "viste", per così dire, dall'alto, dove il bianco è un valore di intensità associato al picco positivo dell'onda, mentre il nero è associato al picco negativo; se la *Turbolenza* è a 0, le bande di *Marble* e *Wood* saranno dritte: l'algoritmo serve infatti a definire la perturbazione di tali bande se la *Turbolenza* è maggiore di 0, così come *Sin/Tri/Saw* servono a specificare la forma dell'onda, e quindi se il passaggio nero-bianco-nero dovrà essere sinusoidale, triangolare o a dente di sega.

Tutte queste informazioni ci servono per definire l'aspetto della filettatura o, al contrario, delle scanalature. Impostiamo, per iniziare, *Turbolenza* 0, *Soft*, *Sin*.

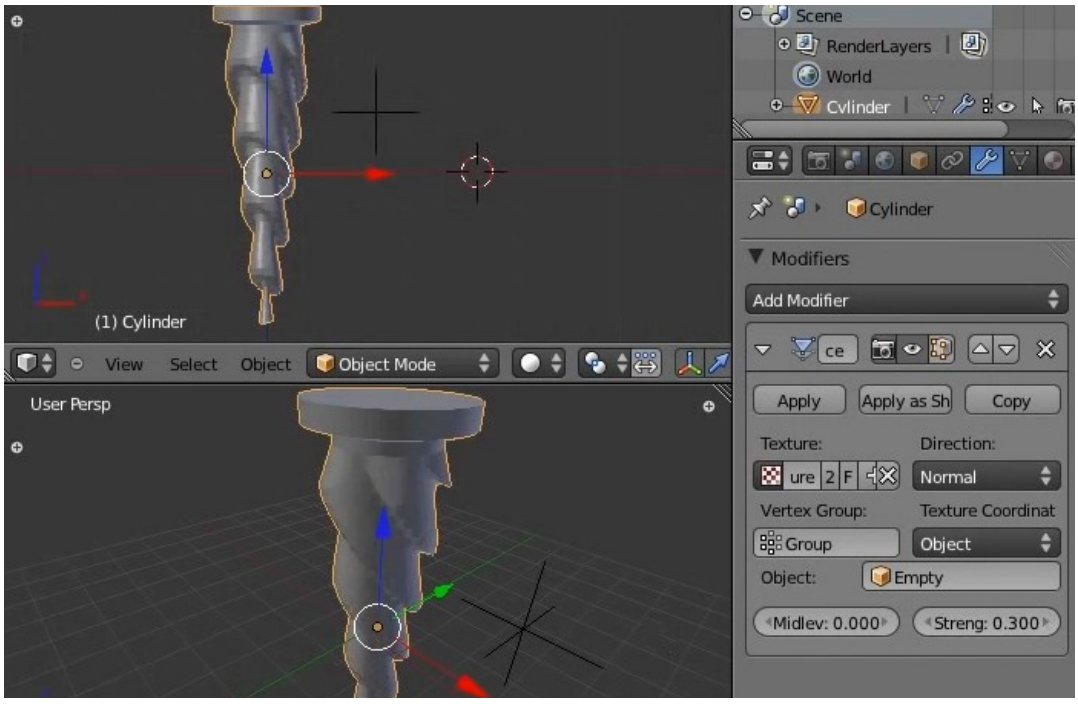
Selezioniamo quindi la mesh ed aggiungiamo un modificatore *Displace*, impostando il *Vertex Group* precedentemente creato nel relativo campo, la Texture appena definita e abbassando il valore di *Midlevel* e *Strength*, portando il primo a 0.0 e il secondo al valore che preferiamo; in genere, il valore di default di *Strength*, 1.0, si rivelerà eccessivo, ma non è detto. Il risultato sarà molto deludente; ciò è dovuto alla mappatura della Texture, applicata di default dal modificatore in modalità *Map*, come si può notare osservando il menù *Texture Coordinates* di *Displace*.

Per risolvere questo problema ottenendo, nel mentre, il massimo controllo possibile sulla mappatura, inseriamo nella scena una *Empty*, impostiamo *Texture Coordinates Object* per *Displace* e, nel campo che apparirà nella scheda del *Modifier*, digitiamo il nome della *Empty* appena creata.



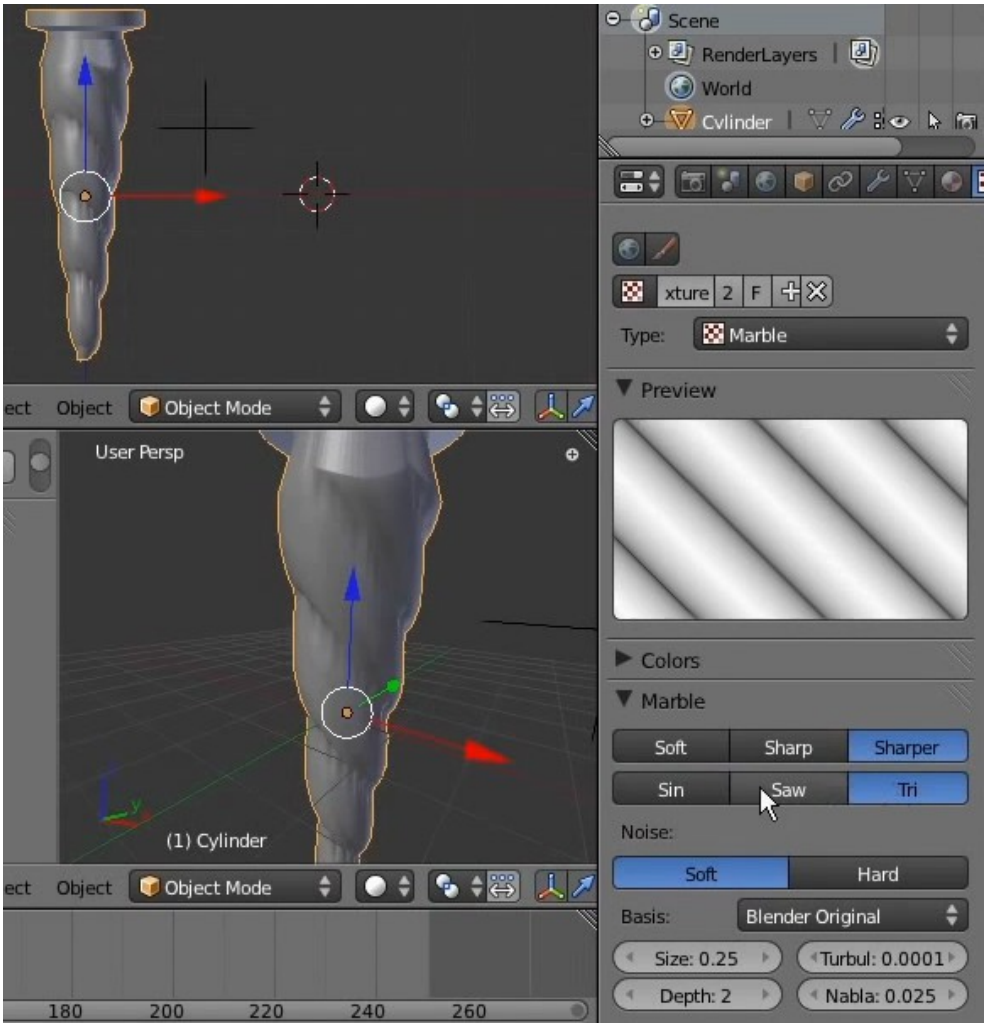
Primo risultato (deludente) dell'utilizzo di Displace

Adesso dobbiamo lavorare sulla *Empty*, trasformandola ed osservando in tempo reale gli effetti sulla mesh; in particolare, la traslazione avrà l'effetto di spostare la "fase" della filettatura, mentre la rotazione (da effettuare singolarmente intorno a ciascun asse) varierà l'inclinazione, mentre lo scaling aumenterà la densità. Per rendere più o meno evidente l'effetto, invece, è necessario variare *Strength* in *Displace*. Può essere utile, per ottenere un risultato più gradevole, attivare *Set Smooth*, se non altro per le sole facce appartenenti al *Vertex Group* selezionato.



Il nuovo effetto, utilizzando una Empty per la mappatura della Texture su Displace

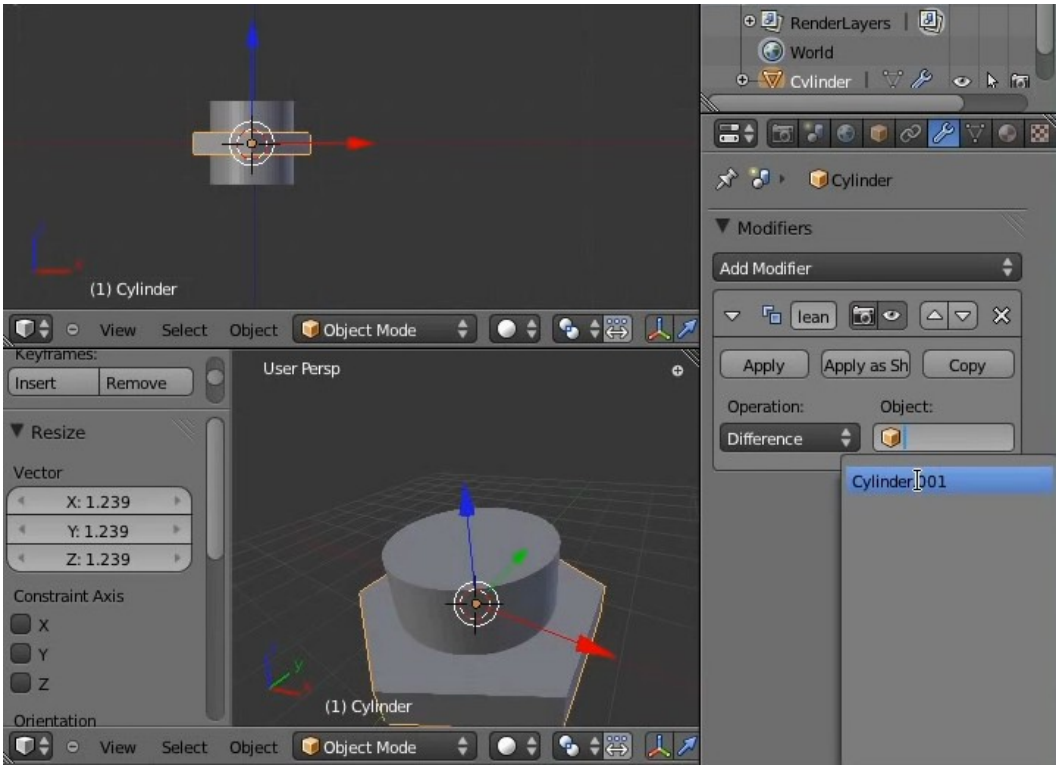
Il nostro grado di controllo sull'aspetto della filettatura non si limita però a questo: come anticipato, possiamo cambiare la forma d'onda utilizzando, ad esempio, *Tri* o *Saw* nella Texture, o rendere più o meno ripidi i passaggi con *Soft*, *Sharp* e *Sharper*... e non è finita: attivando la *Color Ramp* possiamo personalizzare al massimo la forma d'onda, inserendo nuove bande per variare le transizioni nero-bianco (e quindi la filettatura), cambiare il contrasto ed altro ancora, il tutto osservando in tempo reale le modifiche generate, cosa che semplifica enormemente il lavoro.



Modificare la forma d'onda della Texture procedurale usata come Displacement Texture

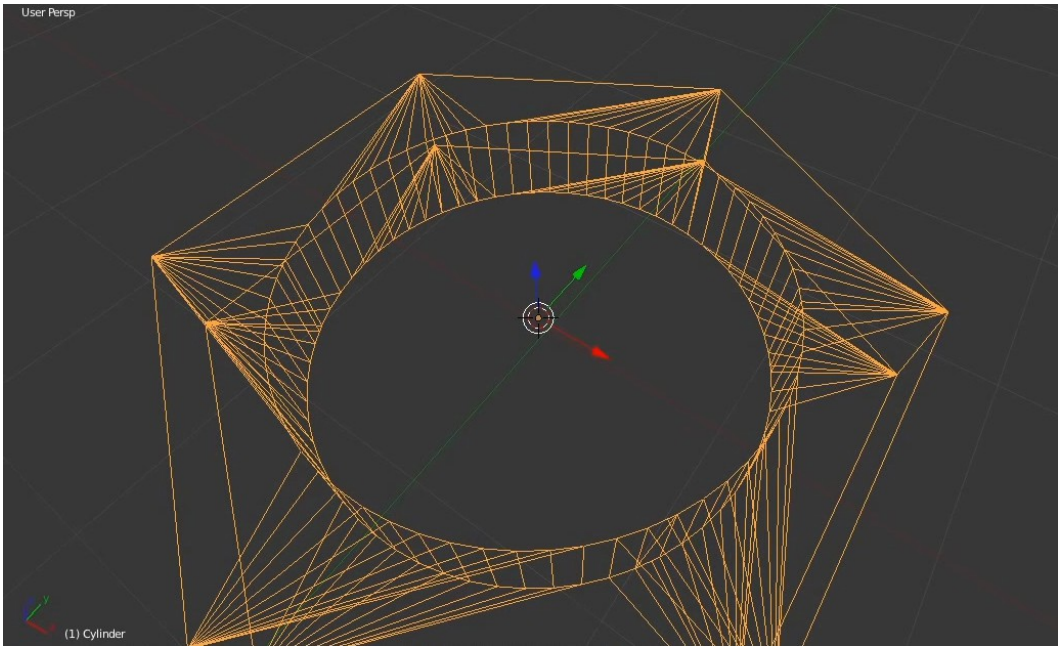
Va da sé che, se applicato con *Strength* negativa, *Displace* può essere utilizzato per generare delle scanalature, ed eccoci al secondo esempio di questo tutorial: il bullone.

Inseriamo un cilindro nella scena impostando, però, solo 6 vertici: ciò avrà l'effetto di creare un prisma esagonale. Per poter ottenere un buon foro circolare in questa mesh, inseriamo un nuovo cilindro, questa volta con un numero elevato di vertici (ad esempio 64 o più) e ricorriamo al modificatore *Boolean*, ed in particolare: posizioniamo il cilindro in modo da passare il prisma da parte a parte, selezioniamo il prisma esagonale, apriamo la scheda *Modifiers* dello stesso, aggiungiamo un modificatore *Boolean*, impostiamo la modalità *Difference*, selezioniamo il nuovo cilindro nel campo *Object*, e... aspettiamo, ed in particolare aspettiamo che Blender applichi per la prima volta il modificatore, operazione che, a seconda dei casi (ad esempio, dal numero di vertici degli oggetti) può richiedere poco o molto tempo.



La realizzazione del bullone: utilizzo del modificatore Boolean Difference

Possiamo osservare un'anteprima del risultato passando in modalità *Wireframe* in una *3D View*; in questa fase, ossia quando ancora non abbiamo cliccato su "*APPLY*" nella scheda del modificatore *Boolean*, possiamo ancora effettuare delle trasformazioni, ad esempio scalando lungo uno o più assi uno dei due oggetti, per variare le dimensioni del foro.



Osservare gli effetti in modalità Wireframe

Quando il risultato ci soddisfa, clicchiamo su "*Applica*" nella scheda del modificatore.
Possiamo anche cancellare il secondo cilindro, a questo punto, ottenendo il bullone.

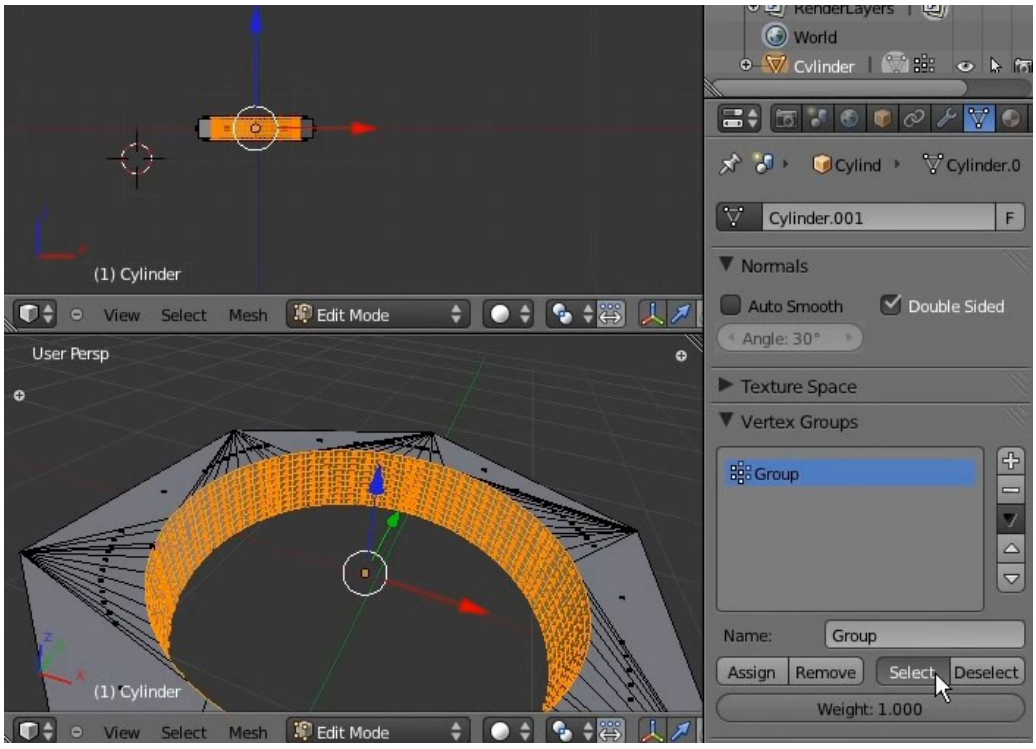
Come detto in un mio precedente tutorial, per realizzare mesh da passare ad esempio in *STL* per la prototipazione rapida e la stampa 3D, è preferibile utilizzare *Boolean* per unioni o sottrazioni di volumi. Le mesh realizzate con le modalità descritte in questo tutorial, dopo aver applicato (per rendere definitivi gli effetti) i *Modifiers Boolean* e *Displace*, si presteranno bene alla stampa 3D. Torniamo comunque al nostro esempio.

La cosa interessante è che *Boolean* crea le facce interne del bullone seguendo le facce della mesh utilizzata per creare il foro, creando quindi delle facce rettangolari (infatti siamo partiti da un cilindro) per cui per selezionarle tutte sarà sufficiente selezionarne una (in modalità selezione facce) e scegliere "*Linked Flat Faces*" dal menù *Select*, nell'header di una finestra *3D View.*

Qualora *Boolean* dovesse generare erroneamente qualche faccia triangolare, possiamo procedere col lavoro, noioso ma necessario, di selezionare gli spigoli per così dire "diagonali" di confine tra facce triangolari, eliminarli, selezionare a 4 a 4 i vertici dello spazio vuoto che verrà creato e premere *F*, per *FILL*, per riempire gli spazi con facce quadrangolari.

Prima di continuare, una piccola parentesi: qui ho utilizzato delle mesh per parlare, nel mentre, del modificatore *Boolean*, tuttavia una mesh con dei fori può essere realizzata in maniera più agevole ricorrendo alle *Curves*, visto che una *Curve* dentro un'altra crea appunto dei fori, ed estrudendo l'oggetto *Curve* così definito prima di procedere alla conversione in mesh; maggiori informazioni su queste operazioni con le *Curves* nelle relative puntate del mio *corso di base su Blender 3D*.

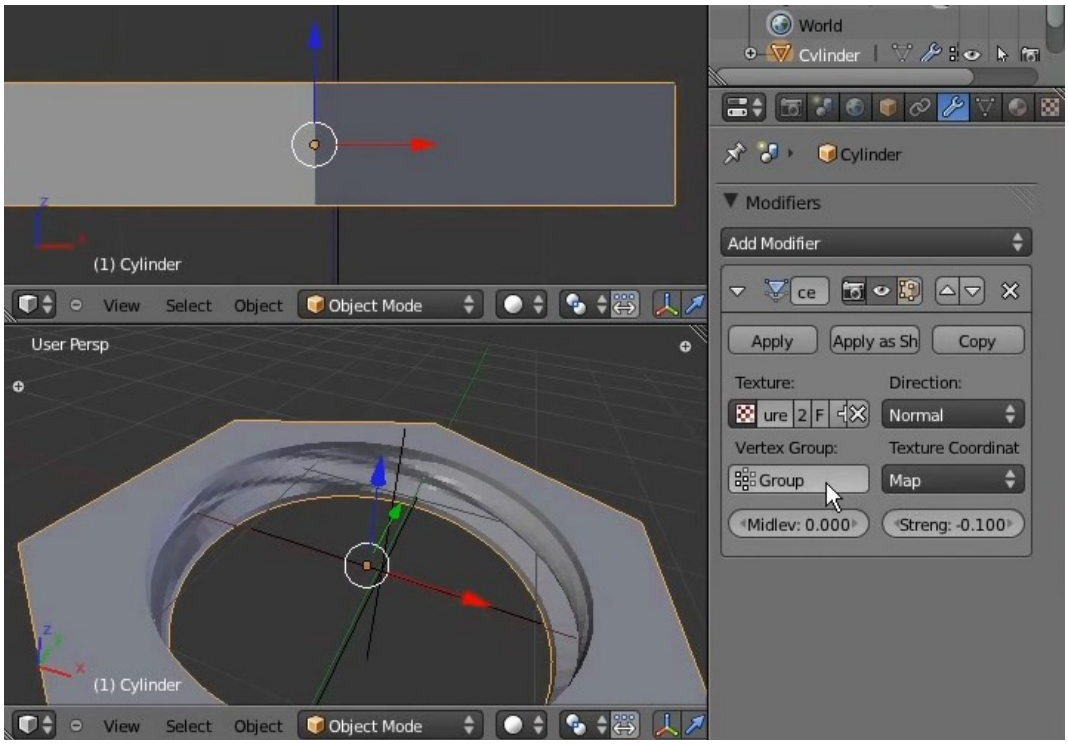
La selezione delle facce interne ci serve per aumentare la densità delle stesse e applicare *Displace* solo a tali facce, non a tutta la mesh. Per aumentare la densità, ricorriamo ancora una volta a *Ring Cut* con *CTRL R* e muovendo la rotellina del mouse per aumentare il numero di suddivisioni equidistanti. Questa suddivisione creerà molte facce ma fortunatamente l'operazione di selezione è ancora facile, grazie a *Linked Flat Faces*.
Impostiamo quindi un *Vertex Group* per tali facce, ma ATTENZIONE: vi consiglio fortemente di TOGLIERE dalla selezione di facce la prima e l'ultima "striscia", in quanto queste facce sono collegate alle facce superiore e inferiore del bullone e una distorsione dei loro vertici (in seguito all'applicazione di *Displace*) avrà effetti indesiderati su tali parti della mesh; in alternativa, è possibile inserire anche queste facce ma procedere, in seguito, all'allineamento dei vertici che compongono il cerchio superiore e quello inferiore del foro, allineamento che avviene selezionando (un cerchio per volta) i vertici e premendo *S Z Z 0.0*, se Z è l'asse locale verticale del foro.



Realizzazione del Vertex Group per le facce interne del bullone

È possibile rimuovere certi vertici, spigoli e facce da un *Vertex Group* selezionando tali elementi nella mesh in Edit Mode, selezionando il *Vertex Group* con *Select* e cliccando su *Remove*, per rimuovere appunto la selezione dal gruppo di vertici selezionato. Blender non aggiorna la *3D View* dopo aver fatto quest'operazione, per cui vedremo i vertici ancora selezionati anche dopo aver cliccato su *Deselect* nel *Vertex Group*.

Inseriamo quindi un modificatore *Displace* per la mesh, impostando il *Vertex Group*, *Midlevel*, *Strength*, la *Texture* (possiamo riutilizzare *Marble* creata precedentemente per la vite) e una nuova *Empty* per definire la mappatura della Texture sulle facce del *Vertex Group*, tutto come visto poco fa per la vite, con la sola differenza che in questo caso *Strength* di *Displace* dovrà avere valori negativi, in modo da lasciare al giusto livello le facce corrispondenti alle parti nere della Texture e incidere quelle corrispondenti alle parti bianche.

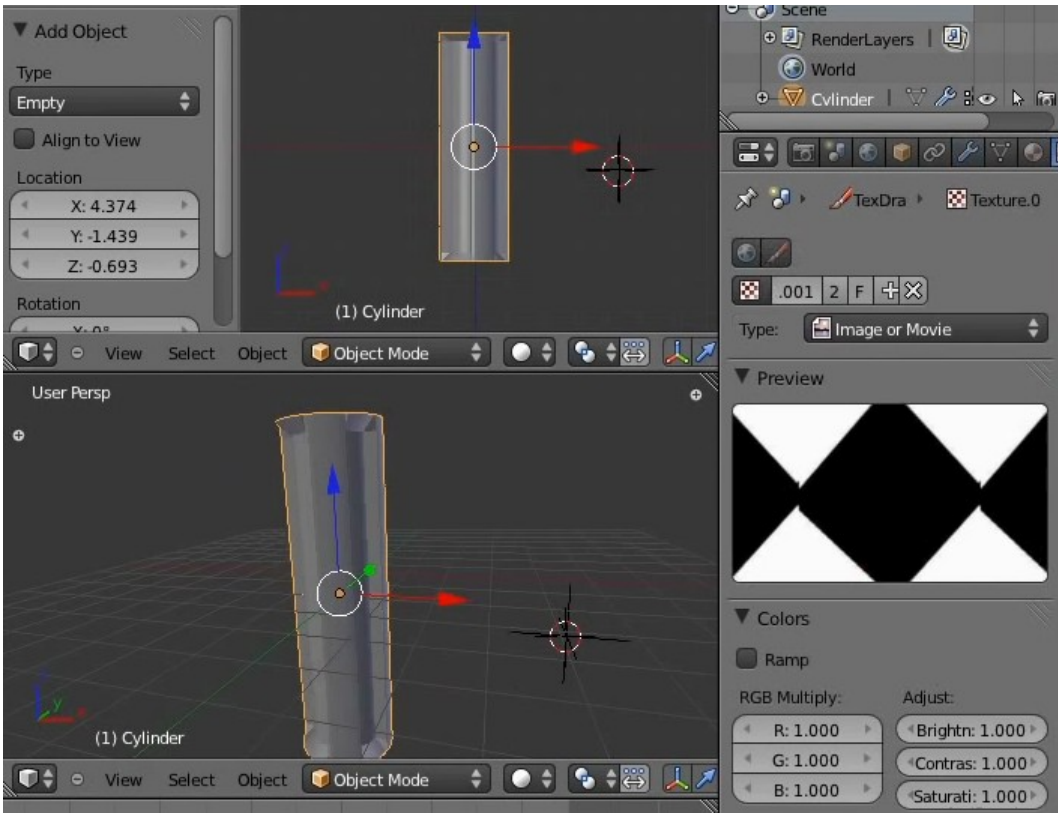


L'utilizzo del modificatore Displace per realizzare la filettatura interna del bullone

Abbiamo fatto uso di Texture procedurali ma ovviamente è possibile ricorrere a immagini in scala di grigi, opzione utile se il disegno della filettatura è estremamente preciso o se si desidera effettuare rilievi o incisioni con "motivi ornamentali".

Facciamo quindi un terzo esempio, per mostrare nel mentre come dovrebbe essere impostata una Texture per replicare correttamente un motivo su una mesh.

Supponendo di voler creare dei rilievi a forma di rombo, non dovremo disegnare un rombo bianco centrato in un'immagine a sfondo nero (eventualmente sfumando l'immagine per creare una transizione dolce nel rilievo): dovremo, invece, posizionare 4 copie del rombo agli angoli del rettangolo nero, mostrando in ogni angolo solo un quarto del rombo.



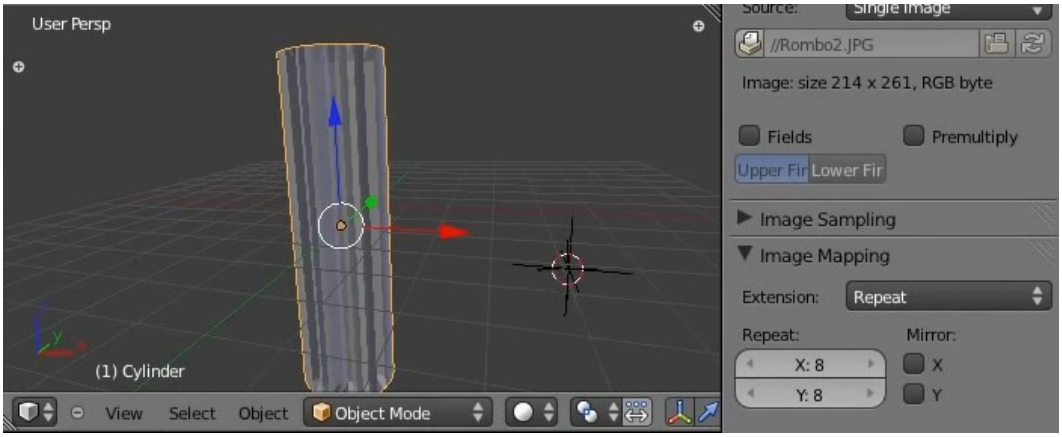
La Texture immagine da utilizzare per il terzo esempio

Questa operazione ci serve per generare una Texture adatta al *TILING*, ossia la ripetizione: affiancando infatti varie copie della Texture in verticale e in orizzontale, comporremo correttamente i rombi bianchi su sfondo nero (le parti d'angolo si completeranno con quelle delle copie adiacenti).

Questo concetto va applicato alla generazione di una qualsiasi Texture dedicata allo scopo qui descritto; ovviamente, si procederà con la definizione della giusta risoluzione delle facce alle quali

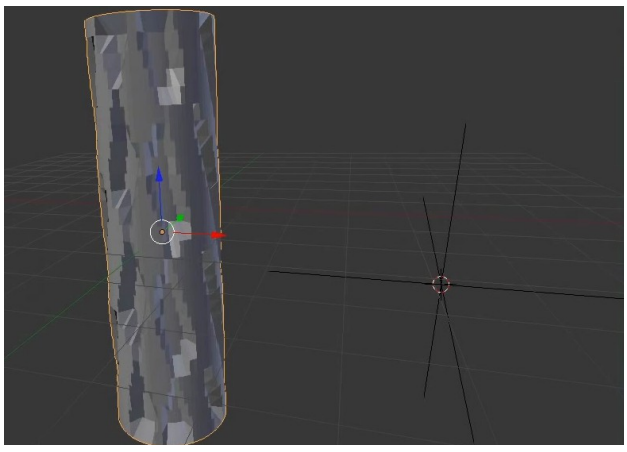
applicare l'effetto, l'assegnazione di tali facce ad un *Vertex Group* e la creazione di un *Modifier Displace* "pilotato" da una *Empty* per orientare e scalare la Texture sull'oggetto, per cui la novità, in questo caso, è rappresentata dai campi *Repeat X* e *Repeat Y* della sezione *Image Mapping* della Texture immagine utilizzata.

A differenza delle Texture procedurali, infatti, una Texture immagine verrà applicata, di default, solo UNA VOLTA a tutte le facce appartenenti al *Vertex Group*; l'operazione di scaling della *Empty* di controllo avrà l'effetto di ridimensionare tale immagine senza però replicarla, come avveniva invece con le Texture Procedurali.



Replicare la Texture immagine sulla mesh con Extension - Repeat

Per poter replicare l'immagine e, quindi, il "motivo" sulla mesh, è necessario: aprire il pannello *Image Mapping* della Texture immagine nella scheda Texture, all'interno della *Properties Window*; impostare *Extension*: *Repeat* (di default dovrebbe avere proprio questo valore); impostare valori maggiori di 1 per i campi *Repeat X* e *Repeat Y* (ripetizione in orizzontale e in verticale).



Screenshot del risultato finale, scalando la Empty

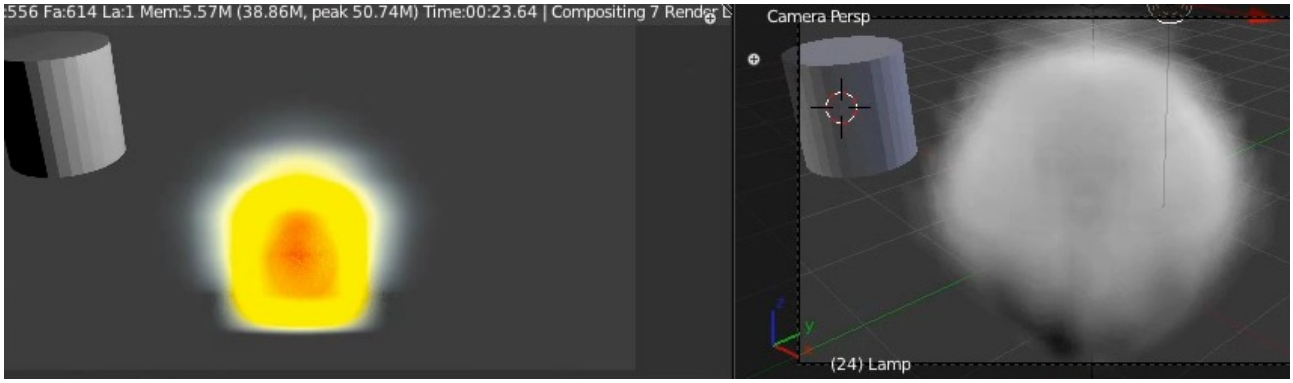
A questo punto potremo ruotare la *Empty* per impostare il "motivo" sull'oggetto.

* * *

Render Layers per effetti di Nodes Compositing su Smoke

In questo tutorial vedremo come utilizzare i *Render Layers* per isolare il rendering dell'effetto *Smoke*, in Blender 3D, in modo da aggiungere degli effetti come *Blurring* e *Glare* in fase di Compositing; dobbiamo ricorrere alle maschere sui *Render Layers* perché *ID Mask*, come vedremo, è di poco aiuto con *Smoke*. Il tutorial è stato realizzato con la versione 2.59 di Blender 3D.

Vediamo subito qual è il problema: vogliamo isolare le fiamme, prodotte in questo rendering mediante il simulatore *Smoke* di Blender, per aggiungere un *soft* e un *hard glow* con *Blur* e *Glare*, tecnica vista in altri tutorial pubblicati precedentemente.



Screenshot del rendering dal quale vogliamo isolare le fiamme, per aggiungere degli effetti in Compositing

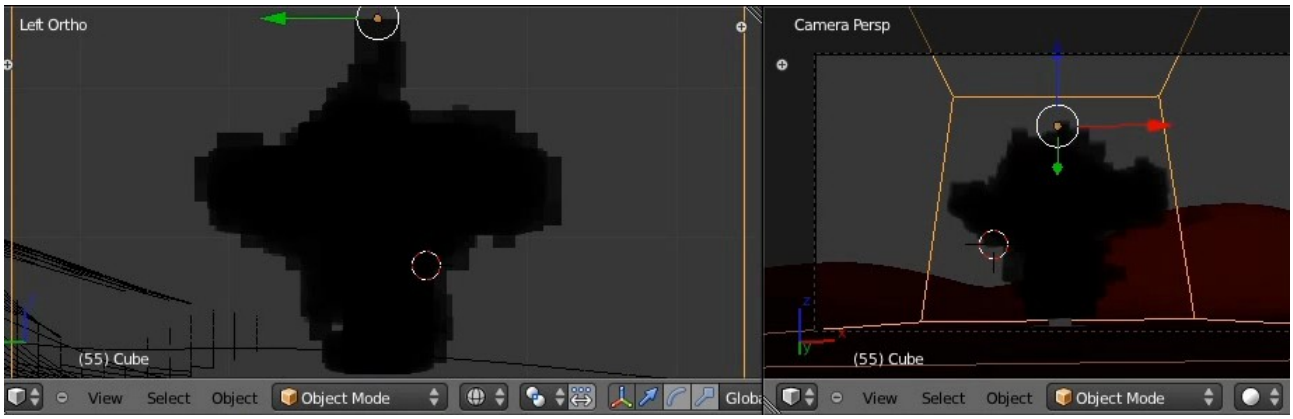
In genere per far ciò ricorriamo a *ID Mask*, in modo da isolare un oggetto in base al suo *Pass Index*; dei Nodi *ID Mask*, *Mix*, *Alpha Over*, dei *Render Layers* e del compositing in generale si parla in maniera più approfondita nel mio *ebook "Il Compositing in Blender 3D 2.5"* (sezione ebook del mio sito web).

Con *Smoke* non possiamo utilizzare *ID Mask*: il *Pass Index* sull'oggetto *Flow* restituisce infatti solo la sagoma dell'oggetto emettitore, nella scena visibile a video la sfera, mentre il *Pass Index*

sull'oggetto *Domain* restituisce tutto quello che c'è all'interno del *Domain*, incluso lo sfondo, tranne gli spazi occupati, ad esempio dagli oggetti *Flow*, renderizzati come neri.

Per evitare di creare un groviglio di nodi, filtri, maschere e collegamenti, utilizziamo due *Render Layers* all'interno della stessa scena per isolare il risultato di *Smoke*, tecnica che può essere estesa, in generale, anche ad altri elementi, non per forza risultati di simulazioni fisiche ma anche mesh per così dire "standard".

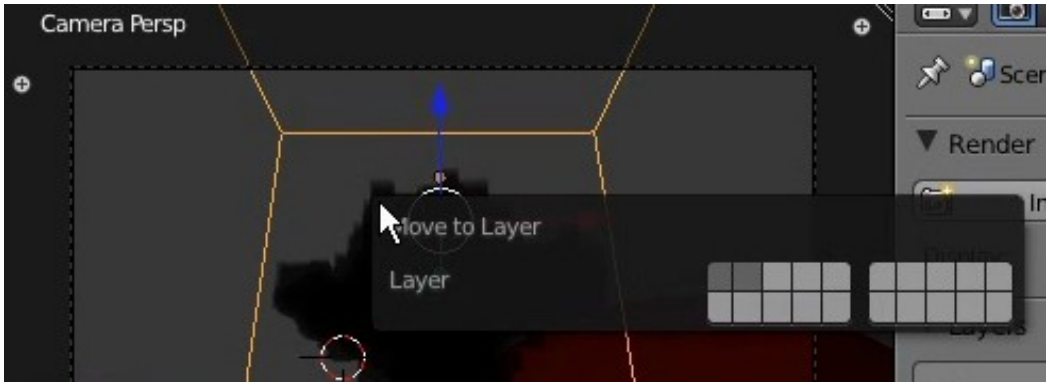
Nella scena visibile qui di seguito abbiamo una mesh di sfondo, un cubo *Domain* della simulazione *Smoke* e due oggetti *Smoke Flow*: un altro cubo e un Plane.



La scena utilizzata in questo tutorial

La prima cosa da fare è duplicare gli elementi che parteciperanno alla simulazione *Smoke*, e che vogliamo isolare per il Compositing, copiandoli in un altro *Layer* della *Scene*; tale *Layer* non dev'essere renderizzato nel *Render Layer* di default di Blender.

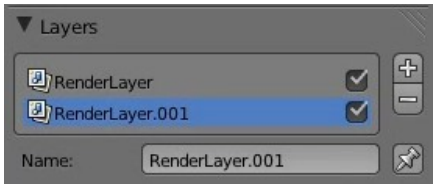
Per copiare gli elementi in un altro *Layer*, ad esempio il secondo, selezioniamoli (con una selezione multipla) e premiamo *M*, dopodiché selezioniamo sia la prima casella, che rappresenta il *Layer* di rendering "standard", che un'altra, con una selezione multipla effettuata con *SHIFT* e click del tasto sinistro del mouse.



Copiare gli elementi selezionati in un altro Layer, utilizzando SHIFT e click sinistro nella maschera dei Layer aperta con M

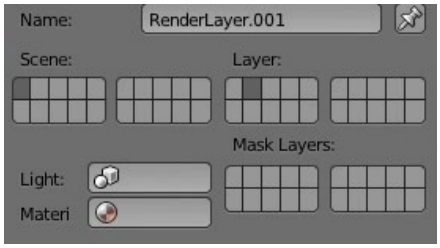
A questo punto spostiamoci nella sezione *Layers* della scheda *Render*.

Dobbiamo aggiungere, per prima cosa, un altro *Render Layer*, ossia un'altra immagine di rendering prodotta mediante la stessa scena (ma *Layers* differenti) e dalla quale sia possibile estrapolare le informazioni di Compositing. Clicchiamo quindi sul + posto accanto allo slot *Layers*, che conterrà di default il *Render Layer* di base di Blender, aggiungendone uno nuovo che avrà per nome *RenderLayer.001*, che possiamo ovviamente rinominare, ad esempio in *SmokeLayer*.



Creazione di un nuovo Render Layer

Dobbiamo dire a Blender che il secondo *Render Layer* deve contenere solo il rendering del secondo *Layer* della scena, per cui selezioniamo *SmokeLayer* nella pila di *Layers* e, appena sotto, selezioniamo solo la seconda casella del gruppo di 20 sotto l'etichetta *Layer*: questa operazione di filtraggio ci consentirà di non includere il primo *Layer* della scena nel rendering del secondo *Render Layer*.

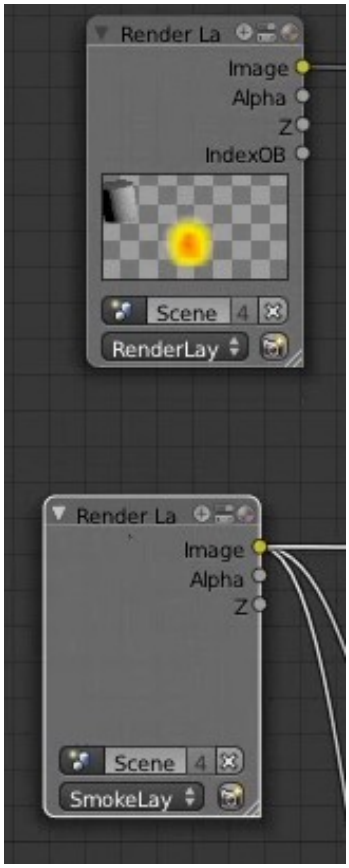


"Filtraggio" dei Layer per un Render Layer, utilizzando la relativa maschera

Qui abbiamo finito: spostiamoci ora in una finestra *Nodes Editor* e attiviamo i Compositing Nodes.

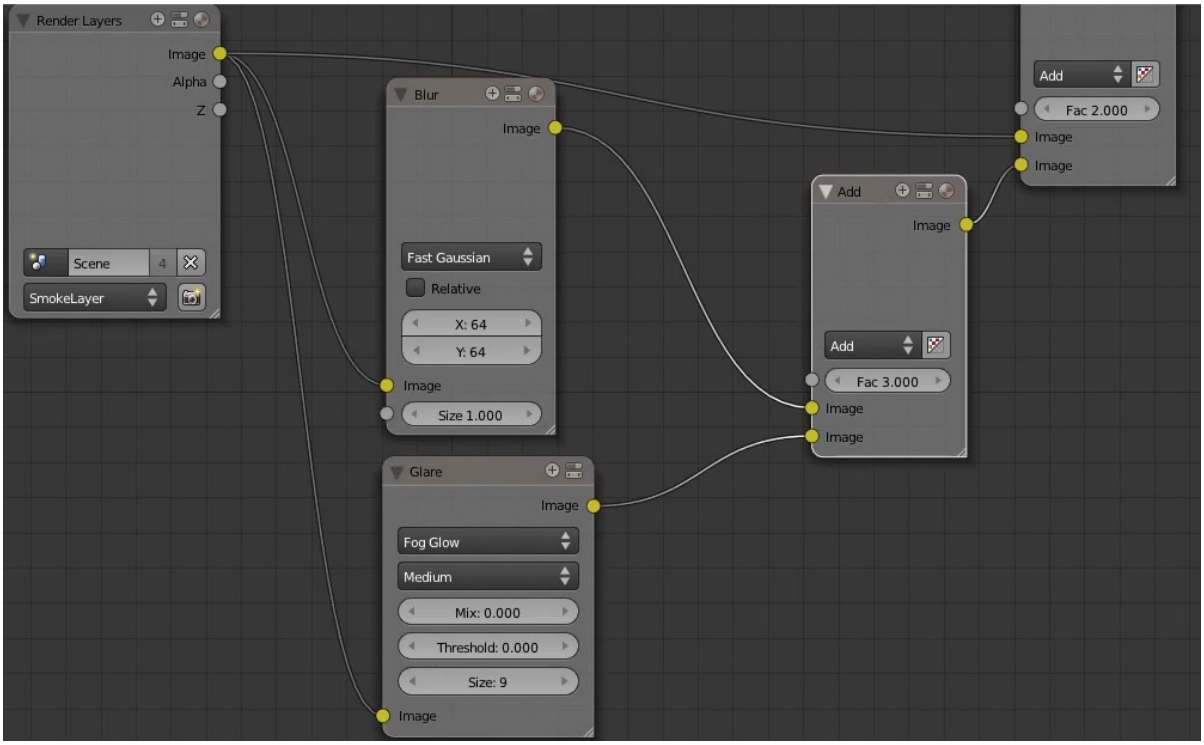
Dobbiamo inserire un secondo Nodo *Render Layers*, oltre a quello presente di default e contenente il primo *Render Layer* della scena; in questo secondo Nodo specificheremo che la sorgente dei dati dev'essere *SmokeLayer*.

A questo punto possiamo aggiungere gli effetti *soft* e *hard glow*, implementati mediante nodi *Glare* e *Blur*, all'immagine in arrivo da *SmokeLayer*; dovremo quindi fondere gli effetti all'immagine originale con un paio di nodi Mix *Add* e, infine, con l'immagine complessiva in arrivo dal primo *Render Layer*, per ottenere l'effetto finale; attenzione, perché in realtà stiamo sommando due volte l'immagine *Smoke* di base, in quanto il rendering del fuoco è presente sia nel *Render Layer* di base che in *SmokeLayer* ed anzi è sommato anche agli effetti *Glare* e *Blur*.



Creazione del nuovo Nodo Render Layers

È possibile, ad esempio, saltare quest'ultimo punto e sommare i due effetti direttamente al primo *Render Layer*, portando da 3 a 2 il numero di Nodi *Mix Add* da utilizzare.



Schema di Nodi di Compositing per implementare gli effetti Soft e Hard Glow

Screenshot dell'effetto finale

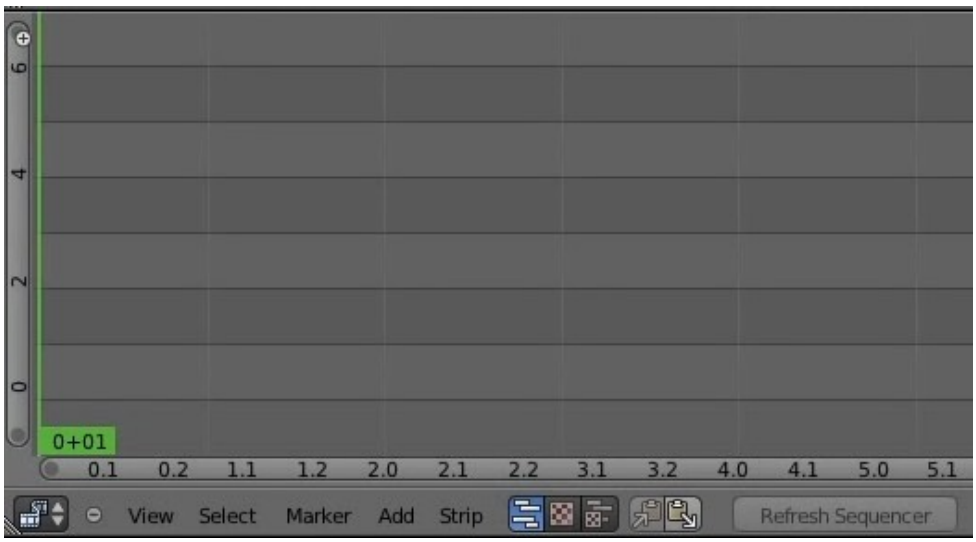
* * *

Videoclip da sequenza di immagini con il Sequence Editor

Questo brevissimo tutorial è dedicato all'utilizzo del *Video Sequence Editor* per ottenere filmati da sequenze di videoclip, immagini e frames, in Blender 3D.
Il tutorial è stato realizzato con la versione 2.59 di Blender 3D.

A volte si ha la necessità di realizzare un filmato partendo da più videoclip o, come nel caso che tratteremo in questo tutorial, da varie immagini, ciascuna rappresentante ad esempio un frame di un'animazione, ad esempio perché si è scelto di renderizzare diversi range di frames su computer diversi (per lavorare in parallelo, riducendo i tempi di rendering), oppure per evitare di perdere il lavoro in caso di black out o altro ancora.

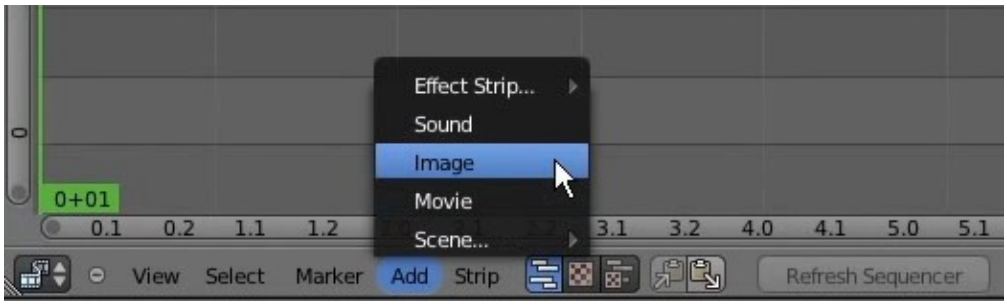
Il *Sequence Editor* di Blender è, di default, sempre attivo, com'è possibile constatare osservando l'omonima checkbox nella sezione *Post Processing* della scheda *Render*, tuttavia se non si aggiunge niente alla scheda del *Sequence Editor* non si noterà la sua presenza in fase di rendering, fase nella quale verrà mostrato a video solo il frame corrente (elaborato eventualmente col Compositing).
Cambiamo quindi una finestra in *Video Sequence Editor*.



Una finestra VSE

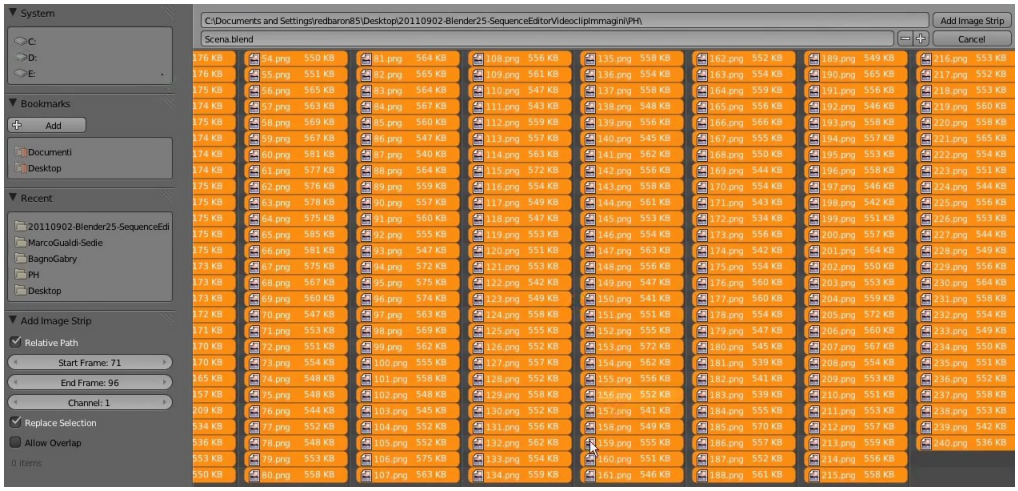
Non parleremo delle varie funzionalità di questo strumento, che consente anche di creare transizioni tra *strip* e altri effetti, limitandoci invece a mostrare cosa fare per assemblare un videoclip da un set di file, siano essi altri videoclip o singole immagini da "concatenare", per così dire.

Posizioniamoci al primo frame della sequenza.
Clicchiamo su *Add* e scegliamo la voce relativa al tipo di elemento che vogliamo aggiungere nel *Sequence Editor*, nel nostro caso *Image*; si aprirà una finestra stile esplora risorse, che ci consentirà di scegliere quali file inserire nella finestra.



Add Image, nel VSE

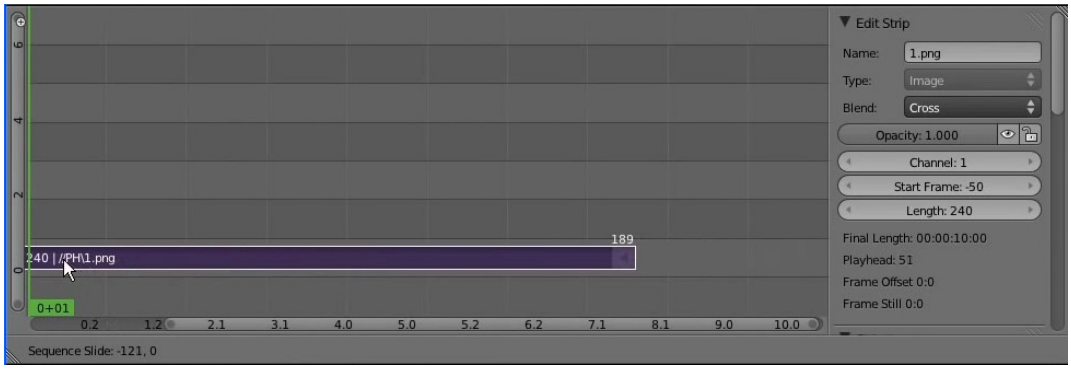
La cosa positiva è che è possibile effettuare una selezione multipla: nel mio caso, sto selezionando con il mouse (con click e tasto sinistro per definire l'area di selezione) tutti i file presenti nella cartella (operazione che possiamo effettuare premendo *A*, shortcut che in Blender serve a selezionare tutti gli elementi di un contesto).



Selezionare tutti i file immagine presenti nella cartella

Cliccando su *Add Image Strip*, la sequenza (importata in ordine, seguendo i nomi dei file) verrà aggiunta a partire dal frame nel quale ci trovavamo, ed ecco perché prima ci siamo posizionati al primo frame dell'animazione; in ogni caso, niente paura: possiamo sempre trascinare la strip in avanti o indietro nel tempo o in un altro canale con *G* (per *Grab*) e muovendo il mouse, o

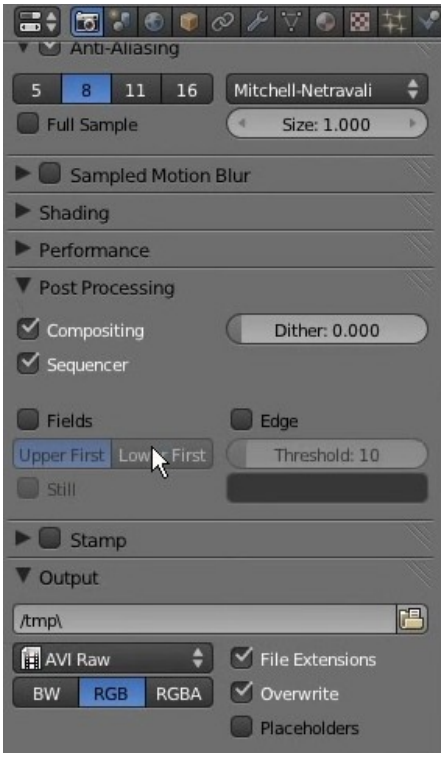
selezionarla e impostare in maniera parametrica canale, frame iniziale e durata dell'animazione con gli strumenti della *Transform Window* della finestra *Video Sequence Editor*, richiamabile con *N*.



La "Strip" (sequenza di immagini) nel VSE

In questo modo, nei frame coperti dalla striscia stiamo sostituendo le immagini importate al rendering del relativo frame della scena di Blender 3D.

Nel momento in cui abbiamo aggiunto almeno un elemento al *Video Sequence Editor*, infatti, Blender ha "cambiato natura", in un certo senso, diventando un *editor video*; il rendering della scena non verrà effettuato, a meno di non aggiungere una o più strip "*Scene*" nell'Editor.



Alcune impostazioni della scheda Render, nella Properties Window

A questo punto, spostiamoci nella scheda *Render* all'interno della *Properties Window* ed impostiamo come preferiamo i parametri di *Dimensions* e *Output*; nel mio caso, sto riducendo a 240 il numero di frames del videoclip da produrre, in quanto ho importato 240 immagini (e l'associazione fatta di default da Blender è un'immagine per frame), mentre la durata dell'animazione di default di Blender è 250 frames e negli ultimi 10 frames verrebbe raffigurato lo schermo nero. In output, scegliamo un formato della sezione *Movie* per generare un'animazione, quindi clicchiamo su *Animation* (o premiamo *CTRL F12*) per avviare il rendering. Questo particolare rendering richiederà poco tempo, visto che non stiamo renderizzando le scene importate: stiamo semplicemente "concatenando" delle immagini già renderizzate, ottenendo in output un filmato (salvato come file su disco).

* * *

Edges sui rendering delle mesh via Nodes Compositing

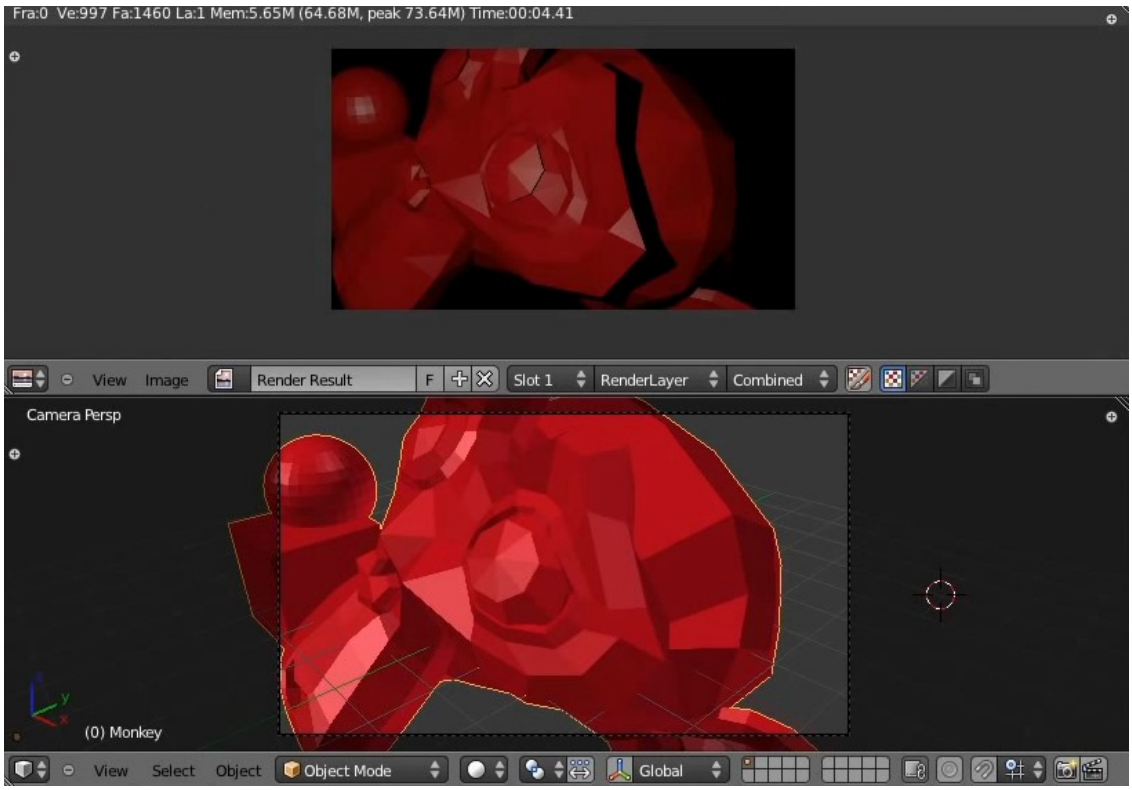
In questo tutorial vedremo un modo per renderizzare solo gli *edges* (spigoli) visibili di una mesh e sovrapporli al rendering di una scena mediante il rendering *OpenGL Render Active View* e il Compositing dei Nodes. Il tutorial è stato realizzato con la versione 2.59 stabile di Blender 3D.

Il problema del rendering di un Material *Wireframe*, per evidenziare gli spigoli di una mesh, è che questo visualizza anche gli spigoli che non dovrebbero essere visibili, e lo fa per il semplice motivo che, non essendoci facce nella mesh, non ci sono occlusioni; a volte però è necessario sovrapporre ad una mesh renderizzata come piena, con Material *Surfaces*, gli spigoli visibili, per evidenziarli, e la funzione *Edges* nativa di Blender viene applicata a tutti gli oggetti e lascia poco spazio alle personalizzazioni.

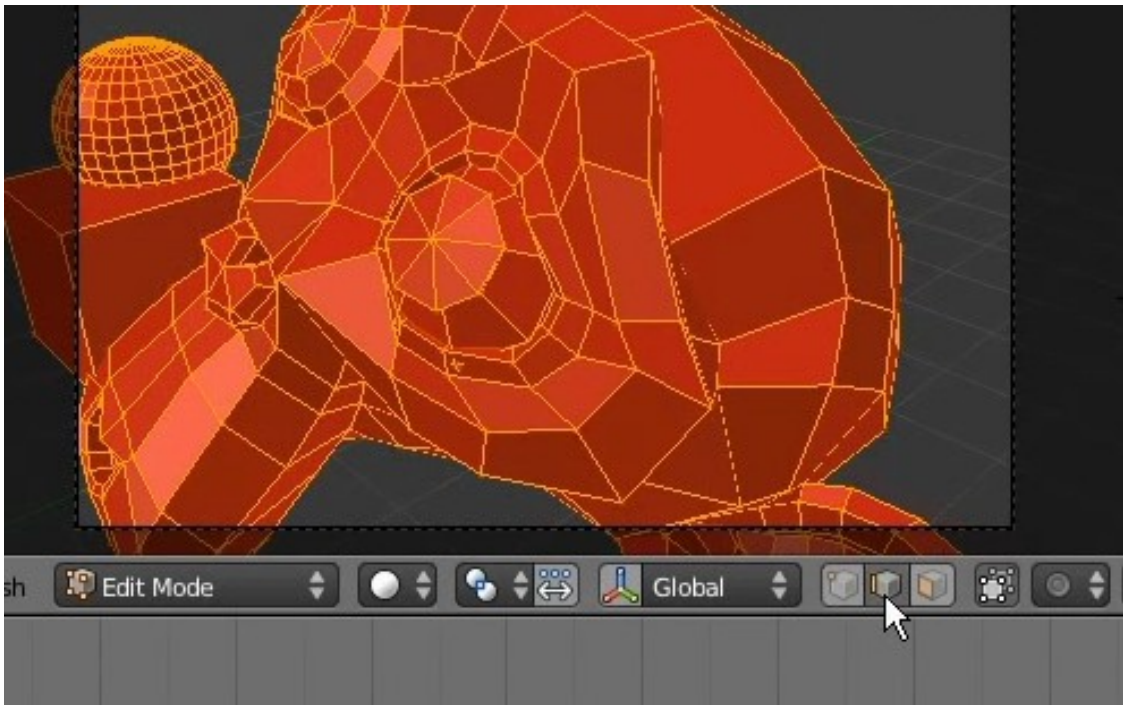
Esistono diverse soluzioni per questo problema, come l'utilizzo di due mesh identiche e imparentate ma renderizzate con due Materials diversi, ma in questo tutorial sfrutteremo il rendering della *Active View* (opzione descritta in un tutorial precedente, presente in questa raccolta) sovrapponendo questo risultato al rendering "classico" della scena mediante *Nodes Compositing*.

La prima cosa da fare è procurarsi il rendering "classico" della scena e salvarlo come immagine su disco, aprendolo poi in una finestra *UV - Editor delle immagini* all'interno di Blender.

Nella *3D View* con vista telecamera lasciamo la modalità di visualizzazione di default, ossia "*Solid*", quindi selezioniamo l'oggetto del quale ci interessano gli spigoli e passiamo in Edit Mode, in modalità *Edge Select*, proprio per evidenziare gli spigoli nella visualizzazione.



Realizzare il rendering "classico" della scena, visualizzato in una finestra UV - Editor delle immagini (in alto)



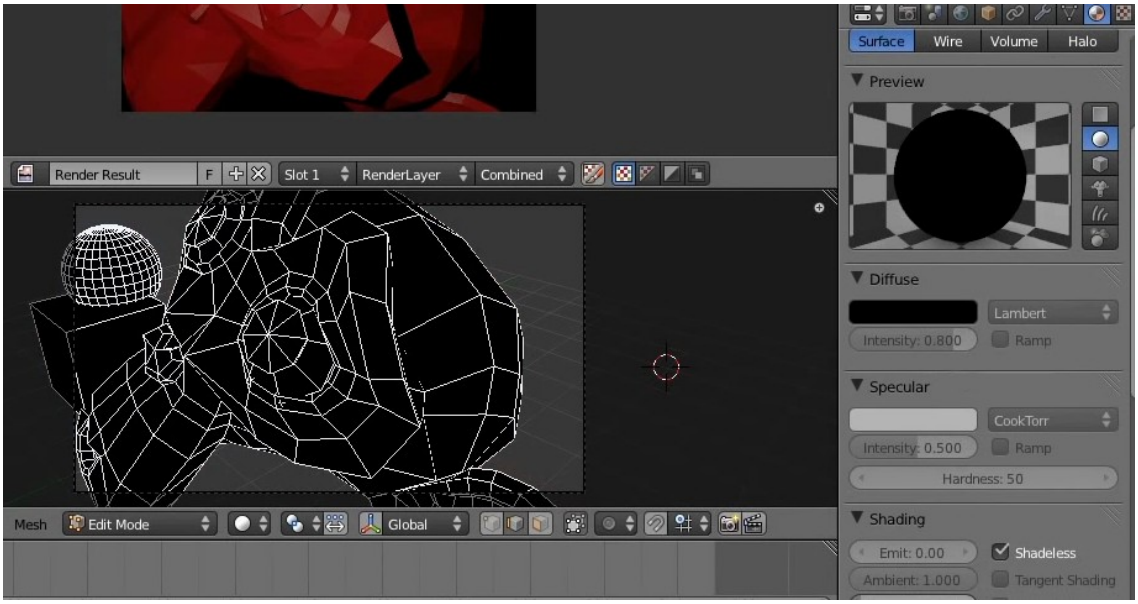
Modalità Edge Select in una 3D View

Prima di effettuare il rendering della *Active Viewport* è necessario personalizzare il tema di colori della stessa, aprendo la finestra *User Preferences* di Blender e, al suo interno, la scheda *Themes*, sezione *3D View*, impostando il colore di "*Edge Select*", ossia il colore da dare agli spigoli della mesh, come bianco (non impostiamo qui il colore che vogliamo dare agli spigoli nel risultato finale: lo faremo in fase di compositing, dopo aver definito la maschera *Set Alpha*), e trasparenza *Alpha 0* per il campo "*Face Selected*".

Cambiare i "temi" dei colori di visualizzazione della scena, nella finestra User Preferences - Themes

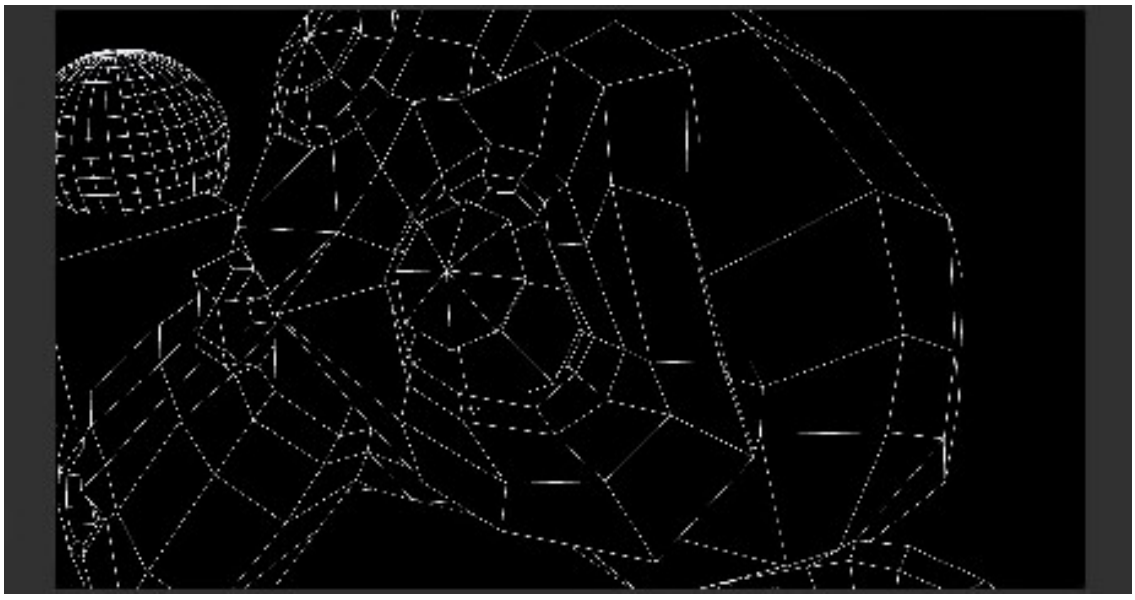
Dobbiamo poi cambiare il colore dell'ombreggiatore diffuso della mesh in nero puro e selezionare "*Shadeless*" in *Shading*, per togliere gli effetti di ombreggiatura.

Così facendo, avremo gli oggetti completamente neri tranne che per il colore degli spigoli, visualizzati in Edit Mode. Utilizzeremo questo nero come colore per la maschera *Alpha* del rendering in fase di compositing.



Material Shadeless

Poniamo quindi il colore di *World Horizon* a nero, in caso di rendering in esterni, e clicchiamo sull'icona *OpenGL Render Active Viewport*, nell'header della finestra *3D View* con vista telecamera, per effettuare il secondo rendering. Salviamo anche questo secondo rendering su disco.



Il rendering effettuato con OpenGL Render Active Viewport

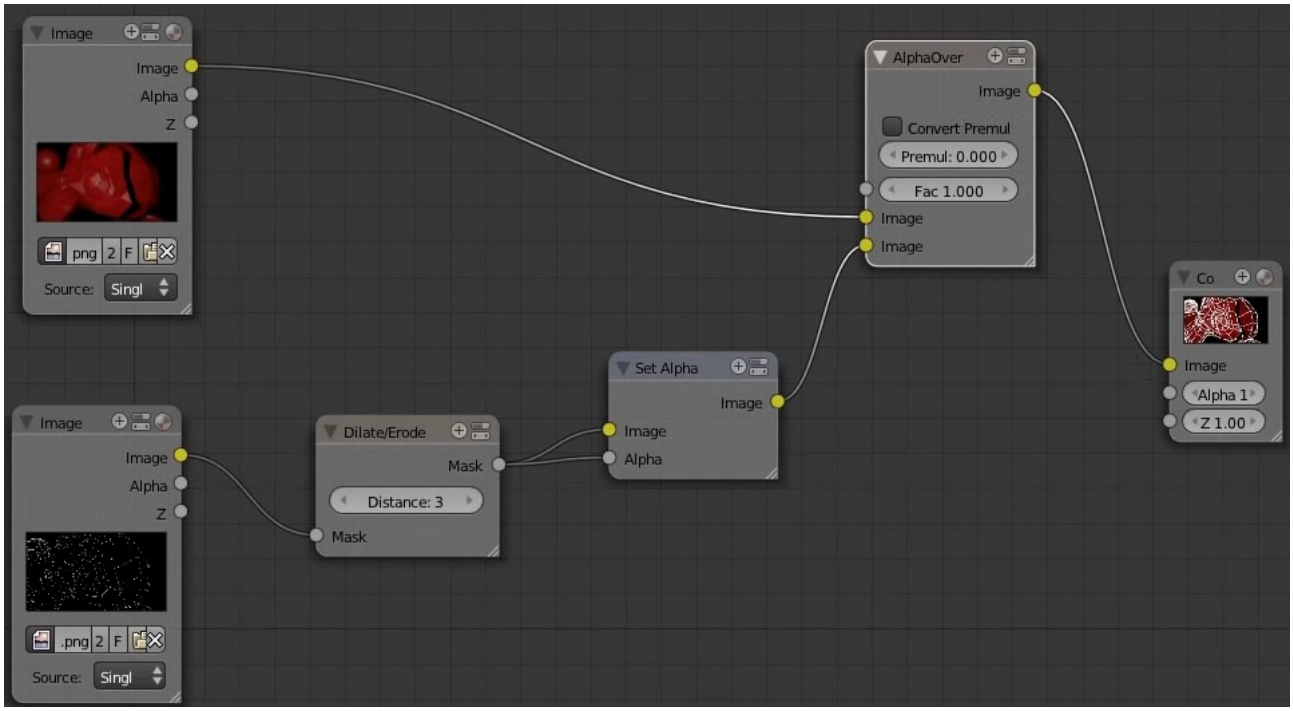
Spostiamoci quindi in una finestra Nodes Editor per effettuare la fase di Post-Processing.

In questo tutorial non descriveremo in dettaglio i vari Nodi, limitandoci ad esaminare la configurazione nell'Editor; per una trattazione approfondita, rimando al mio *ebook sul Compositing dei Nodes in Blender 3D 2.5* (sezione ebooks del mio sito web).

Abbiamo bisogno di due nodi input di tipo *Image*: uno per caricare il rendering "standard" fatto per primo, l'altro per il rendering ottenuto con *Render Active Viewport*.

Al Nodo con l'immagine degli spigoli associamo un Nodo *Dilate* per rendere più spessi gli spigoli, dopodiché passiamo l'output ai due canali input di un Nodo *Set Alpha*.

Se vogliamo cambiare il colore degli spigoli, passiamo l'output di *Set Alpha* ad un Nodo *RGB Curves* da personalizzare a piacere, altrimenti passiamolo al secondo canale *Image Input* di un Nodo *Alpha Over*, che riceverà sul primo canale *Image Input* l'immagine del rendering standard.



Lo schema di Nodi di Compositing tra le due immagini (Nodi IMAGE, non i "soliti" Render Layers)

Portiamo *Factor* del Nodo *Alpha Over* a 1 e passiamo il suo output al Nodo Composite, per realizzare il rendering finale. La procedura è poco pratica e probabilmente si ricorrerà ad altre tecniche per risolvere questo problema, ma non è da sottovalutare e in ogni caso ne abbiamo approfittato per mostrare vari strumenti.

Il risultato finale

* * *

Animazione particelle e scrittura su file con script Python

Questo tutorial è tratto da un esempio pratico presente nel mio *ebook "Programmare Script e Add-Ons per Blender 2.5"*, realizzato facendo riferimento alle API della versione 2.59 stabile di Blender. Nell'ebook si parla anche di come rilasciare gli script sotto forma di *Add-Ons*, per integrarli in maniera stabile nell'applicazione facendoli apparire ad esempio nel menù *Info* o nella *Tool Shelf*, ma in questo particolare tutorial non tratteremo queste operazioni; per maggiori informazioni (come sommario e pagine d'esempio), rimando alla sezione ebooks del mio sito web.

In questo tutorial vedremo come impostare dei sistemi particellari, dei *Force Fields*, e come scrivere alcuni dati su file; ovviamente, nell'ebook sono presenti capitoli introduttivi sul linguaggio *Python* e le API di Blender, dove vengono descritte alcune istruzioni riutilizzate in questo tutorial, ma qui ci limiteremo ad esaminare lo script, che ho copiato nella finestra *Text Editor* del programma per eseguirlo in un colpo solo, mentre nell'ebook è presentato come serie di istruzioni da inserire nella *Python Console*.

```
1  # Sistema Particellare su file; Francesco Milanese, 2011
2  # www.redbaron85.com; redbaron85@redbaron85.com
3  import bpy
4  import mathutils
5  from mathutils import *
6  from math import *
7
8  bpy.ops.object.add(type="EMPTY", location=(-10.0,0.0,0.0))
9  Empty1 = bpy.context.selected_objects[-1]
10 Empty1.name = "Empty1"
11 Empty1.select = False
12
13 bpy.ops.object.add(type="EMPTY", location=(10.0,0.0,0.0))
14 Empty2 = bpy.context.selected_objects[-1]
15 Empty2.name = "Empty2"
16 Empty2.select = False
17
18 dir(Empty1)
19 type(Empty1.field)
20
21 ForceField1 = Empty1.field
22 ForceField1.type = "FORCE"
```

View Text Edit Format Templates script.py

Screenshot di una parte dello script

Iniziamo da una scena completamente vuota, cancellando gli elementi eventualmente presenti di default. Supponiamo di voler ricavare la posizione di una decina di particelle immesse nella scena da una mesh e soggette all'influenza di due *Force Field* in grado di modificarne la traiettoria; dobbiamo inoltre scrivere i dati ottenuti frame dopo frame su un file di testo (in chiaro) perché, ad esempio, il nostro professore userà quel file con un altro programma per interpretare i dati ricavati dalla simulazione.

Posizioniamo per prima cosa due *Empty*, che forniremo di *Force Fields*, in (-10, 0, 0) e (10, 0, 0), con istruzioni descritte nei capitoli precedenti dell'ebook, rinominandole nel mentre in "*Empty1*" ed "*Empty2*". Le informazioni che ci consentono di utilizzare un oggetto, nel nostro caso la *Empty*, come *Force Field*, e di impostarne contestualmente i parametri, si trovano in *[oggetto].field*, elemento di tipo "*FieldSettings*" (lett.: "impostazioni campo", appunto).

```
20
21 ForceField1 = Empty1.field
22 ForceField1.type = "FORCE"
23 ForceField1.strength = -10.0
24
25 ForceField2 = Empty2.field
26 ForceField2.type = "TURBULENCE"
27 ForceField2.strength = 20.0
28
29 bpy.ops.mesh.primitive_uv_sphere_add()
30 Emettitore = bpy.context.selected_objects[-1]
31 Emettitore.name = "Emettitore"
32
33 # In questo momento, l'unico oggetto selezionato e'
   proprio Emettitore
34 bpy.ops.object.particle_system_add()
35
36 SistemaParticellare = Emettitore.particle_systems[0]
37 SistemaParticellare.name = "Emissione"
38 SistemaParticellare.settings.frame_start = 1
39 SistemaParticellare.settings.frame_end = 10
40 SistemaParticellare.settings.lifetime = 250
```

View Text Edit Format Templates script.py

Altra porzione di script

Le *API Blender Python* di *FieldSettings* indicano che, tra le altre cose, possiamo:

- definire il tipo di forza, con *type*;
- impostare l'intensità del campo di forza, con *strength*;
- impostare l'intensità della turbolenza, specificando un valore float tra 0 e 10 in *size*;
- impostare la forma del campo di forza (default: "*POINT*", omnidirezionale) con *shape*.

Per il primo campo di forza, impostiamo il tipo a "*FORCE*", con intensità -10.0 e gli altri valori come di default, mentre per il secondo impostiamo "*TURBULENCE*" con intensità 20.0 e gli altri valori come di default.

I campi di forza sono stati impostati, adesso passiamo alla mesh emettitrice: va bene anche una *UV Sphere*, l'importante in questo caso sono i metodi riguardanti il sistema particellare.
Poniamo la sfera al centro dell'universo virtuale, con tutte le impostazioni di default (*size*, *rings*, *location*, *segments*, …), rinominandola in "*Emettitore*" ed associandole una variabile omonima.

L'accesso alle informazioni dei sistemi particellari associabili ad una mesh è dato da "*particle_systems*", una collezione di "*ParticleSystem*", infatti ad una mesh è possibile associare più sistemi particellari, inseriti in questa collezione nelle *API Blender Python*.

Di default, una mesh appena creata non ha alcun sistema particellare associato, per cui la prima cosa da fare è creare un "*particle_system*" e aggiungerlo alla lista dei sistemi della mesh, nel nostro caso la sfera "*Emettitore*"; la creazione di un sistema particellare via codice avviene selezionando l'oggetto al quale bisogna assegnare il sistema particellare ed invocando il metodo *particle_system_add()* di *bpy.ops.object*, che crea un sistema particellare con valori di default per l'oggetto selezionato.

```
proprio Emettitore
34 bpy.ops.object.particle_system_add()
35
36 SistemaParticellare = Emettitore.particle_systems[0]
37 SistemaParticellare.name = "Emissione"
38 SistemaParticellare.settings.frame_start = 1
39 SistemaParticellare.settings.frame_end = 10
40 SistemaParticellare.settings.lifetime = 250
41 SistemaParticellare.settings.count = 10
42 SistemaParticellare.settings.normal_factor = 0.0
43
44 bpy.context.scene.gravity = Vector((0.0, 0.0, 0.0))
45
46 SistemaParticellare.particles[0].location
47
48 import math # Necessaria per round(n,2), che arrotonda n a
   due cifre decimali
49 dati = []
50 for i in range(100):
51     frameCorrente = i+1;
52     riga = "(";
53     bpy.ops.anim.change_frame(frame=frameCorrente)
 View  Text  Edit  Format  Templates  script.py
```

Creazione di un sistema particellare via codice

Assegniamo il sistema particellare alla variabile "*SistemaParticellare*" e personalizziamolo sfruttando i metodi di "*Particle_System.settings*" (un oggetto di tipo *ParticleSettings*), elencati nelle *API Blender Python*. In questo modo l'emissione di particelle inizierà al frame 1, terminerà al frame 10 e genererà 10 particelle con *Lifetime* 250 (la durata dell'animazione di default di Blender), con velocità iniziale (*Normal*) 0.0. La forza di gravità "disturba" la nostra animazione, ma anziché toglierla solo per le particelle del sistema particellare in uso togliamola per tutta la scena, approfittandone per menzionare le impostazioni di *Scene*.

Il nostro professore, nel mentre, ci informa che gli interessano i dati relativi ai primi 100 frames, in un file *.txt*, con questo formato:

- ogni riga corrisponde ad un frame dell'animazione, quindi serviranno 100 righe;
- in ogni riga, la posizione di una particella deve essere indicata mediante tre valori (coordinate XYZ) con due cifre decimali, separati da "," e posti tra "()";
- il carattere di separazione tra le coordinate delle varie particelle è "#".

Quello che faremo sarà quindi:

1. creare una lista di stringhe, ciascuna delle quali conterrà i dati di un frame;
2. in ogni stringa, inserire i valori recuperati dalle particelle ai vari frame, formattandoli secondo le modalità espresse precedentemente;
3. scrivere questa lista su file, un elemento per riga.

Nel sistema particellare, una singola particella è identificata mediante "*particle*"; intuitivamente, l'oggetto "*particles*" di un "*particle_system*" è una *collection*, ovvero una collezione ordinata di oggetti "*particle*". Per recuperare le informazioni di posizione di una particella ad un dato frame, quindi, è sufficiente accedere al campo *location*, per cui bisogna fare un ciclo sulle particelle presenti nel sistema, per tutti i frames dell'animazione. Segue la fase di scrittura dei dati su file: è sufficiente aprire un descrittore di file (un oggetto, associato ad un file) e scrivere, mediante ciclo *for*, le stringhe contenute in dati (stringhe che terminano in *\n* per implementare il ritorno a capo automatico), per poi chiudere il descrittore del file con "*close()*".

Se non si specifica il path assoluto, il file verrà depositato nella cartella principale di Blender (ad esempio, "*C:\Programmi\Blender Foundation\Blender*"); nello specificare un path assoluto,

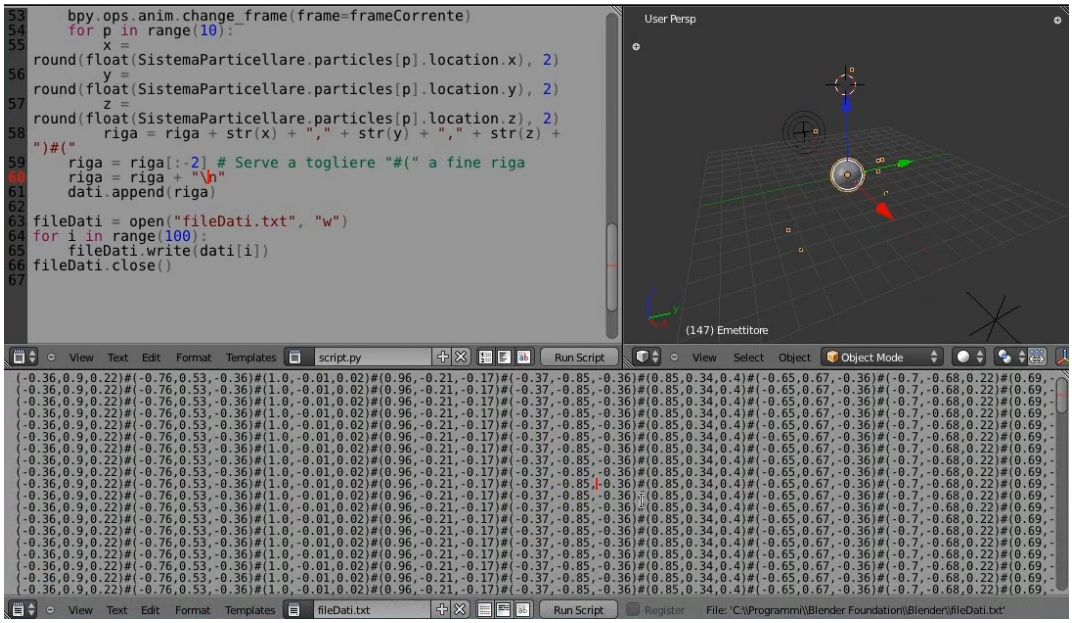
ricordarsi del carattere di escape \ prima di ogni \ nel path (ad es.: "*C:\\dati.txt*" se si desidera salvare il file "*dati.txt*" in "*C:*").

```
    for p in range(10):
        x = round(float(SistemaParticellare.particles[p].location.x), 2)
        y = round(float(SistemaParticellare.particles[p].location.y), 2)
        z = round(float(SistemaParticellare.particles[p].location.z), 2)
        riga = riga + str(x) + "," + str(y) + "," + str(z) + ")#("
    riga = riga[:-2] # Serve a togliere "#(" a fine riga
    riga = riga + "\n"
    dati.append(riga)

fileDati = open("fileDati.txt", "w")
for i in range(100):
    fileDati.write(dati[i])
fileDati.close()
```

La parte finale dello script

Eseguendo lo script con *Run Script* ed avviando ora l'animazione in una *3D View*, vedremo le particelle partire dalla sfera e muoversi, nel corso dei 250 frames "standard", all'interno della scena 3D; nel mentre, Blender ha scritto i dati su file.



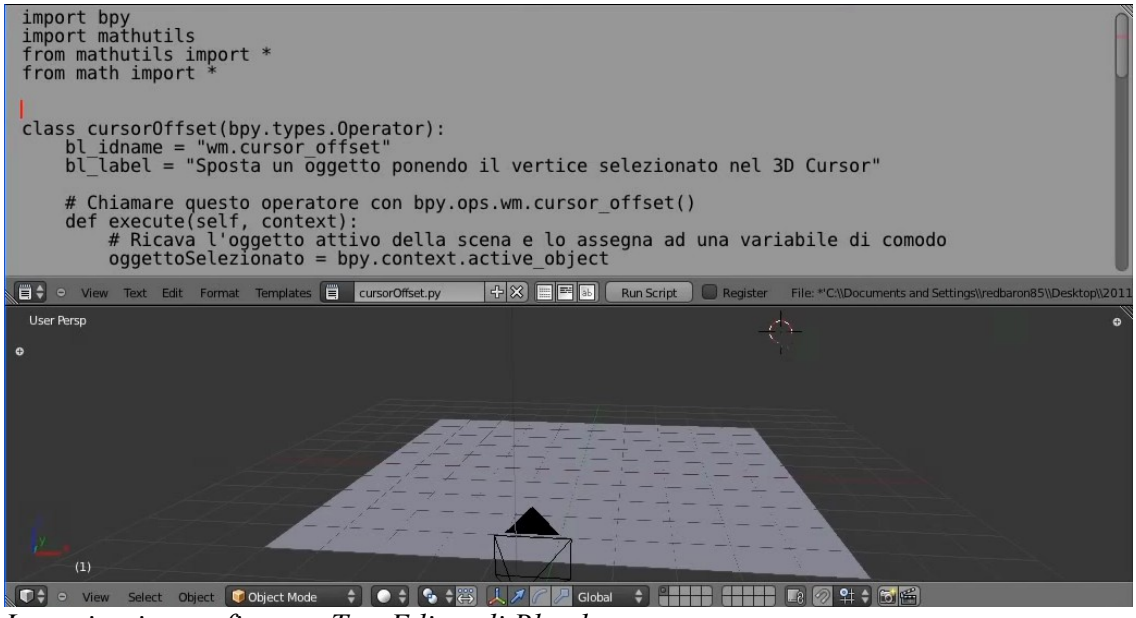
L'output dello script, così come si presenta su file

* * *

Semplice script Python per ancorare le mesh a punti della scena

In questo tutorial vedremo come realizzare un semplice script per spostare un oggetto ponendo un suo vertice nelle coordinate date dalla posizione del *3D Cursor* in Blender 3D 2.59: uno script utile quando si vuole far combaciare un punto di una mesh con un determinato punto dell'universo virtuale; volendo, si tratta di una funzione di ancoraggio degli oggetti.

Questo script non è presente nel mio ebook sulla *programmazione di script e add-ons in Python per Blender 3D 2.5* ma, dopo aver studiato l'ebook, non sarà difficile scrivere o interpretare script del genere. Nell'ebook si parla anche di come rilasciare gli script sotto forma di *Add-Ons*, per integrarli in maniera stabile nell'applicazione facendoli apparire ad esempio nel menù *Info* o nella *Tool Shelf*, ma in questo particolare tutorial non tratteremo queste operazioni; per maggiori informazioni sul contenuto dell'ebook (sommario e pagine d'esempio) rimando alla sezione ebooks del mio sito web.



Lo script, in una finestra Text Editor di Blender

Ho aperto lo script in una finestra *Text Editor* di Blender; lo script non è un *Add-On*, per cui verrà eseguito con *Run Script* e il suo operatore richiamato o con la *Python Console* oppure con la casella dell'help veloce, nella *3D View*.

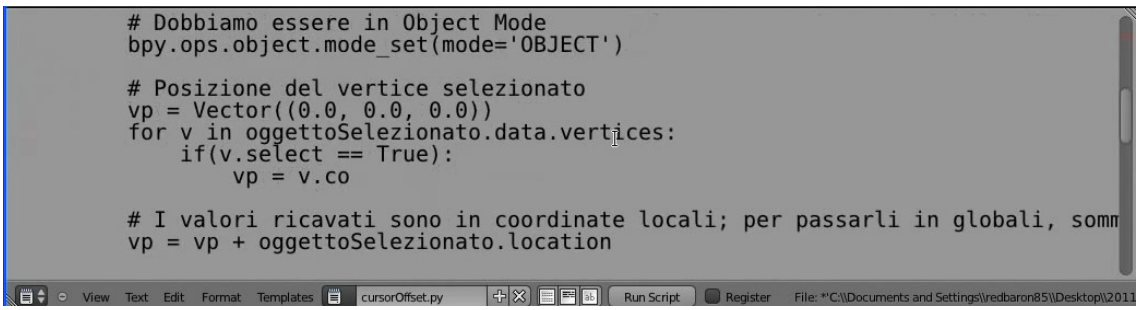
Esaminiamo lo script: all'inizio troviamo, ovviamente, gli *import* di base, mentre a seguire abbiamo la definizione della classe dell'operatore che vogliamo implementare (unica classe presente nello script e registrata nell'ambiente di esecuzione di Blender mediante i metodi *register* e *unregister* e il controllo che li richiama, a fine script).

Il corpo della classe è composto, in realtà, da poche istruzioni, ma prima di esaminarle dobbiamo parlare di come funziona lo script in generale.

A volte si ha la necessità di spostare un oggetto facendo coincidere un suo particolare punto (identificato, in pratica, mediante un vertice) con un altro punto della scena, sia esso il vertice di un altro oggetto che un punto qualsiasi.

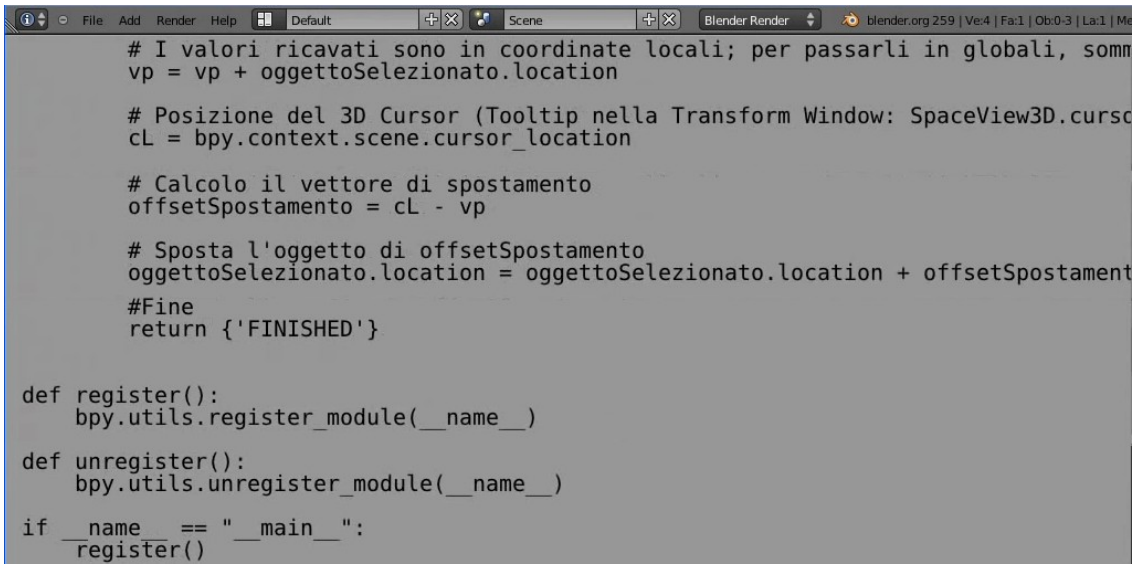
Lo script qui presentato consente di fare proprio questo e funziona in questo modo: dopo aver posizionato il *3D Cursor* nella coordinata di destinazione, si seleziona l'oggetto da spostare, si sceglie in Edit Mode il vertice che dovrà finire nel punto individuato dal *3D Cursor* e, una volta tornati in Object Mode, si esegue lo script, che calcola l'offset di spostamento tra il *3D Cursor* e il vertice, e sposta tutta la mesh, in Object Mode.

Torniamo allo script: la prima istruzione serve a recuperare l'oggetto attivo della scena, quello che vogliamo spostare, assegnandogli una variabile di comodo, dopodiché si forza il passaggio a Object Mode, visto che lo script funziona correttamente solo in questo contesto.



Le prime istruzioni dello script

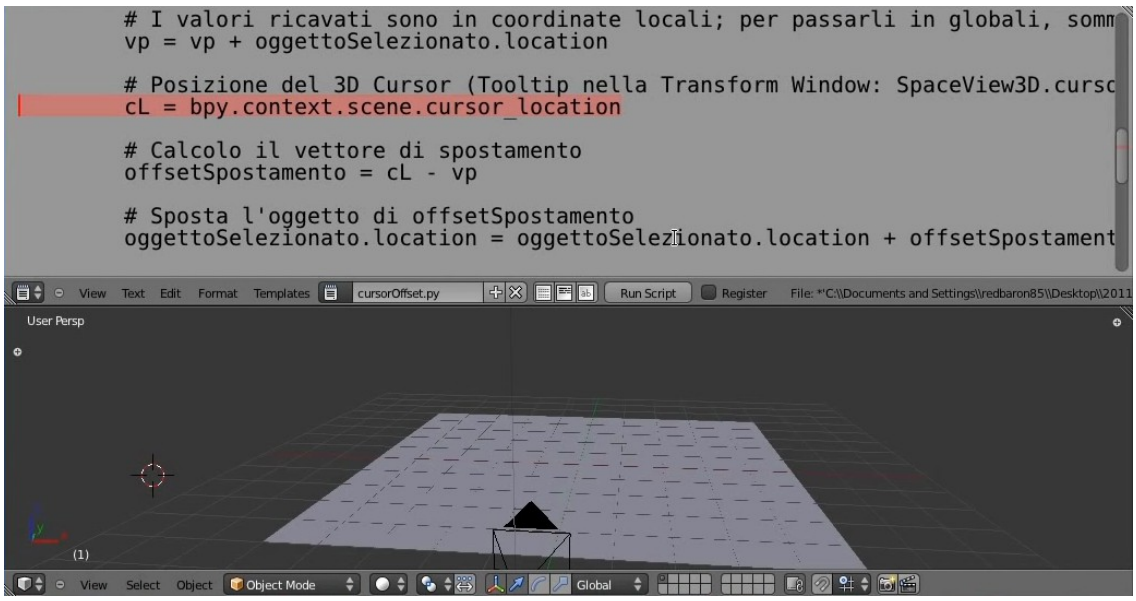
Si recuperano il vertice selezionato e le sue coordinate mediante ciclo *for* con controllo sul valore del campo booleano *select*, depositando le coordinate in *vp* (per *Vertex Position*); dal momento che queste sono coordinate locali del vertice rispetto all'*Origin* della mesh, per recuperare le coordinate globali dobbiamo sommare i valori di *vp* a quelli di *location* della mesh: un vettore con le coordinate XYZ dell'*Origin* della mesh (per cui la somma implementata col + è proprio la somma tra due variabili vettore, membro a membro).



Il resto delle istruzioni dello script

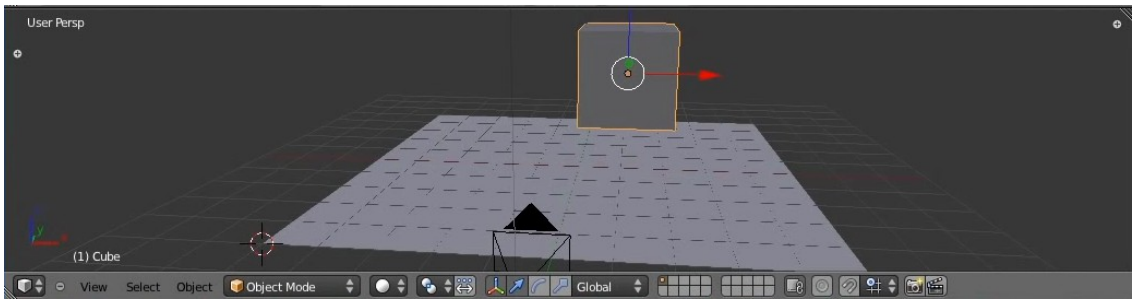
Le informazioni sulla posizione del *3D Cursor* nella scena si trovano, come suggerito dal tooltip su *3D Cursor Location* in una *Transform Window*, in *scene.cursor_location* del modulo *context*, che si riferisce allo stato attuale della scena; depositiamo tali coordinate nel vettore *cL*, per *Cursor Location*, una variabile di comodo.

Segue l'istruzione di spostamento della mesh, implementata ancora una volta con la somma vettoriale: all'*Origin* della mesh, indicata da *Location*, si somma il vettore di *Offset*, calcolato come differenza tra *vertex position* e *cursor location*.



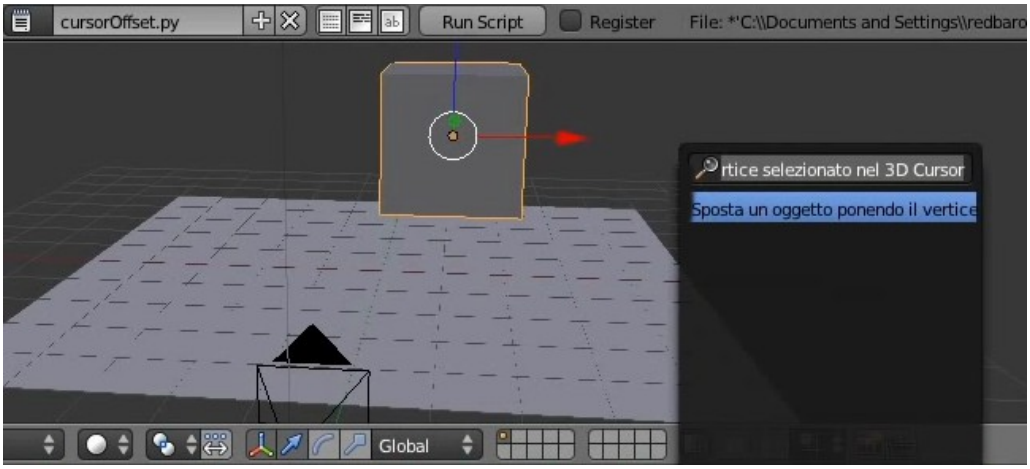
Recuperare le coordinate del 3D Cursor e spostare la mesh

Passiamo alla pratica: nella scena è presente un Plane, adesso passando in Edit Mode e in modalità selezione vertici poniamo il *3D Cursor* in uno dei vertici del piano, selezionando un vertice e sfruttando il menù *SNAP*, richiamabile con *SHIFT S* per scegliere *Cursor to Selected*.
Torniamo in Object Mode e aggiungiamo un altro elemento alla scena, ad esempio il classico cubo, la cui *Origin* sarà, all'inizio, nel punto individuato dal *3D Cursor*.
Spostiamo un po' il cubo per apprezzare meglio l'effetto, poi passiamo in Edit Mode e scegliamo uno dei vertici da posizionare nel punto ove si trova il *3D Cursor*, quindi torniamo in Object Mode.



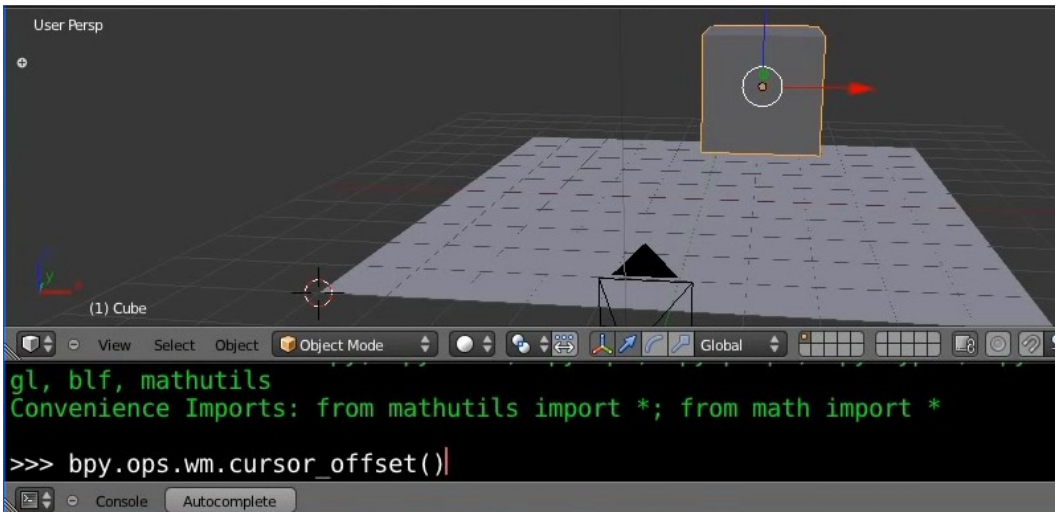
La scena utilizzata per questo esempio

Nella finestra *Text Editor* clicchiamo su *Run Script*: apparentemente non è successo nulla, ma in realtà l'operatore definito nello script è stato registrato all'interno di Blender; richiamiamolo, ad esempio, aprendo la casella dell'help veloce nella *3D View* e iniziando a digitare il nome dato all'operatore in *bl_idlabel*, scrivendo "*Sposta un*" e selezionando la voce che apparirà nella casella dell'help per eseguire lo script.



Richiamare lo script (aggiunto a Blender dopo aver cliccato su Run Script nella finestra Text Editor)

Nella *3D View* premiamo *CTRL Z* per annullare l'operazione ed eseguire lo script richiamandolo, questa volta, dalla Python Console: utilizziamo, in questo caso, l'istruzione specificata nel campo *bl_idname*, più il prefisso *bpy.ops* (in quanto si tratta, ora, di un operatore registrato nelle *bpy*), ossia *bpy.ops.wm.cursor_offset* e le parentesi tonde, perché si tratta di una funzione.



Richiamare lo script (dopo averlo "registrato" in Blender) dalla Console Python

Se si desidera eseguire lo script non appena si clicca su *Run Script*, è sufficiente inserire l'istruzione appena citata nel file dello stesso, subito dopo la definizione della classe dell'operatore, fuori da tale blocco (cioè con indentazione nulla); per la definizione di *Add-Ons* da integrare stabilmente nell'applicazione, invece, rimando alle istruzioni contenute nell'ebook.

* * *

Textures animate nel BGE 2.59 con le Sprites (tiled images)

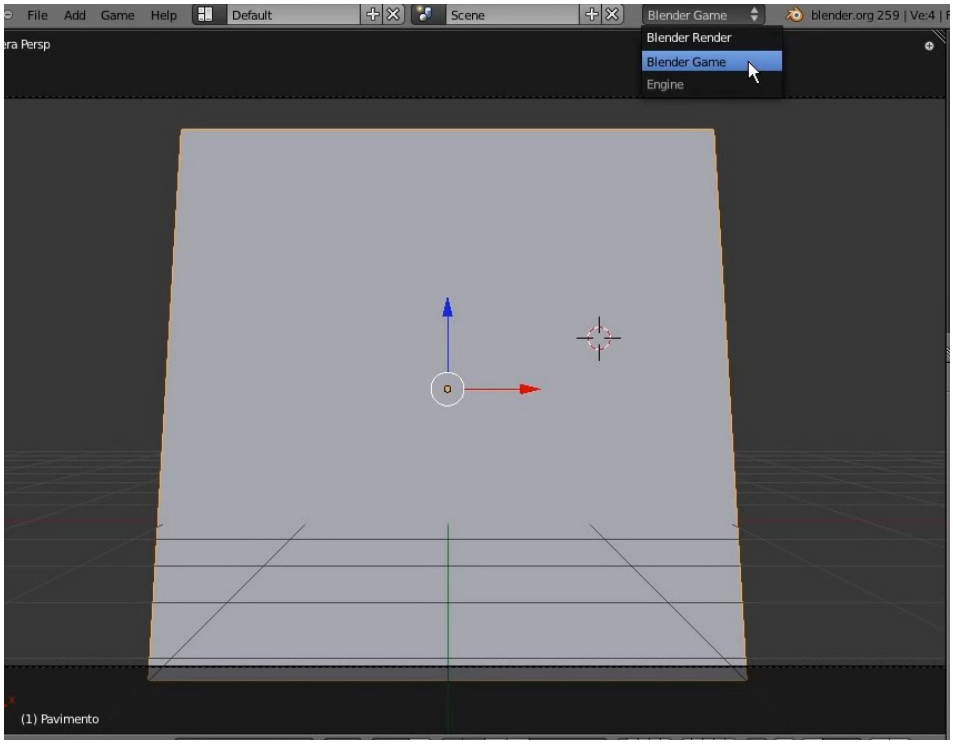
Questo brevissimo tutorial nasce in risposta ad una richiesta di un utente del Forum che chiedeva come poter utilizzare un tutorial, disponibile in Inglese per la versione 2.49 del Blender *Game Engine* (il motore di gioco di Blender 3D) sul sito tutorialsforblender3d.com, con la versione 2.59.

In questo tutorial vedremo come utilizzare un'immagine composta da più *tiles*, o *sprites*, ossia immagini di un elemento scattate in vari istanti, per realizzare una Texture animata nel *Game Engine*, indicando a Blender come suddividere l'immagine per recuperare le varie *sprites* e con che *frame rate* visualizzarle su di un oggetto, in questo caso un piano.

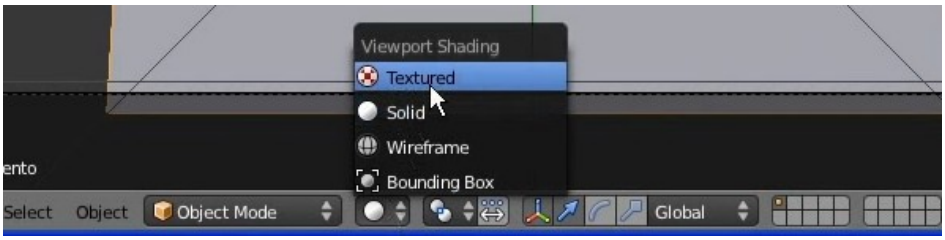
Aggiungiamo una mesh nella scena, in questo caso un semplice Plane, ma potete provare l'effetto anche su superfici più complesse, ad esempio il volto di un personaggio, utilizzando un'immagine delle *sprites* del personaggio mentre parla anziché dover modellare la faccia con le armature o altre tecniche. Cambiamo il motore di rendering, che di default è *Blender Render*, in *Blender Game*, scegliendo tale voce nel menù *Engine*, nella finestra *Info* del programma (prima immagine nella pagina seguente).

Cambiamo inoltre la modalità di ombreggiatura in *GLSL Materials*, scegliendo tale voce nella scheda *Render* all'interno della *Properties Window*, e la modalità di ombreggiatura nella finestra 3D in *Textured*, scegliendo tale voce nel menù *Viewport Shading*, nell'header della *3D View* (seconda immagine nella pagina seguente).

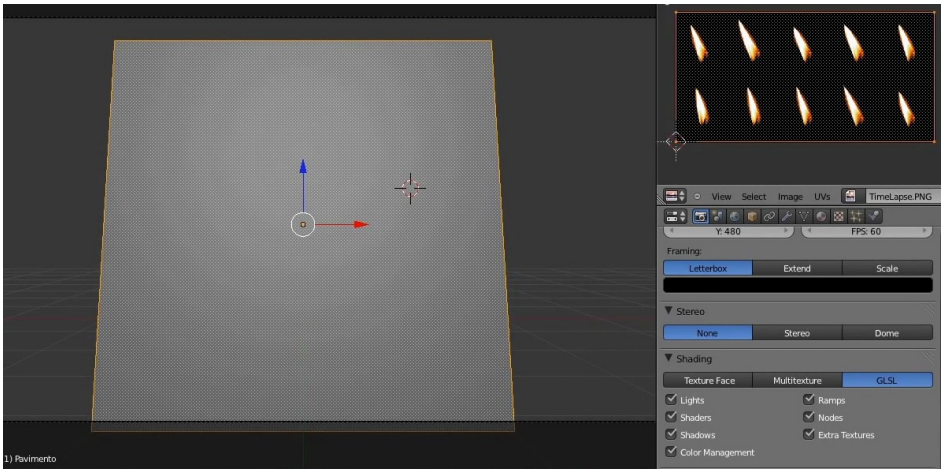
Facciamo l'*unwrap* del Plane assegnandogli, in una finestra *UV – Editor delle immagini*, l'immagine con le varie *Sprites*; sto utilizzando proprio l'immagine presente nel tutorial originale che, come potete vedere, è composta da 10 sprites, disposte su 5 colonne e 2 righe (terza immagine nella pagina seguente).



Passaggio a Blender Game, dopo aver inserito il Plane (inquadrato frontalmente) nella scena



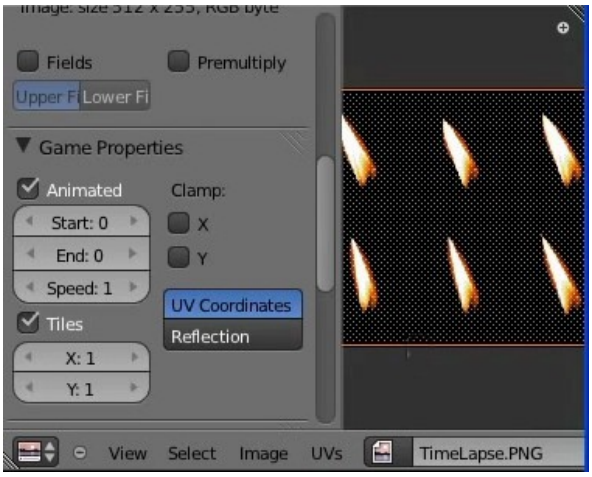
Viewport Shading Textured e ombreggiatura GLSL



Unwrapping e mappatura della Texture delle Sprites

Nel fare l'*unwrap* copriamo tutta l'immagine con la sagoma del Plane, non solo la prima *sprite*, dopodiché apriamo il pannello *Properties* della finestra *UV* con *N* (o con *View* - *Properties*) e, in basso nel pannello, scegliamo *Animated* e *Tiles*.

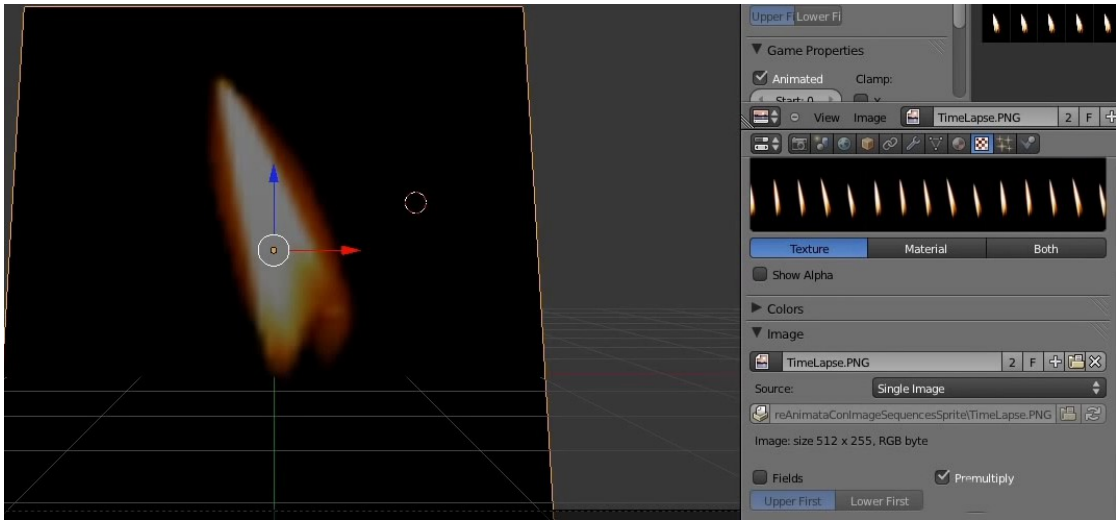
In *Animated* dobbiamo specificare il frame di inizio animazione, il frame di fine animazione e il frame rate, ossia il numero di frame al secondo da visualizzare, il tutto nei campi *Start*, *End* e *Speed*; per questo esempio, per vedere l'animazione lentamente, poniamo ad esempio 0, 9 e 1, cioè un fotogramma al secondo in tempo reale.



Le sezioni Animated e Tiles della Sprite

In *Tiles* specifichiamo 5 per *X* e 2 per *Y*: in questo modo, Blender suddividerà in 10 parti l'immagine e prenderà una *Sprite* per volta, secondo l'ordine dalla prima riga in alto all'ultima in basso, dalla colonna più a sinistra a quella più a destra.

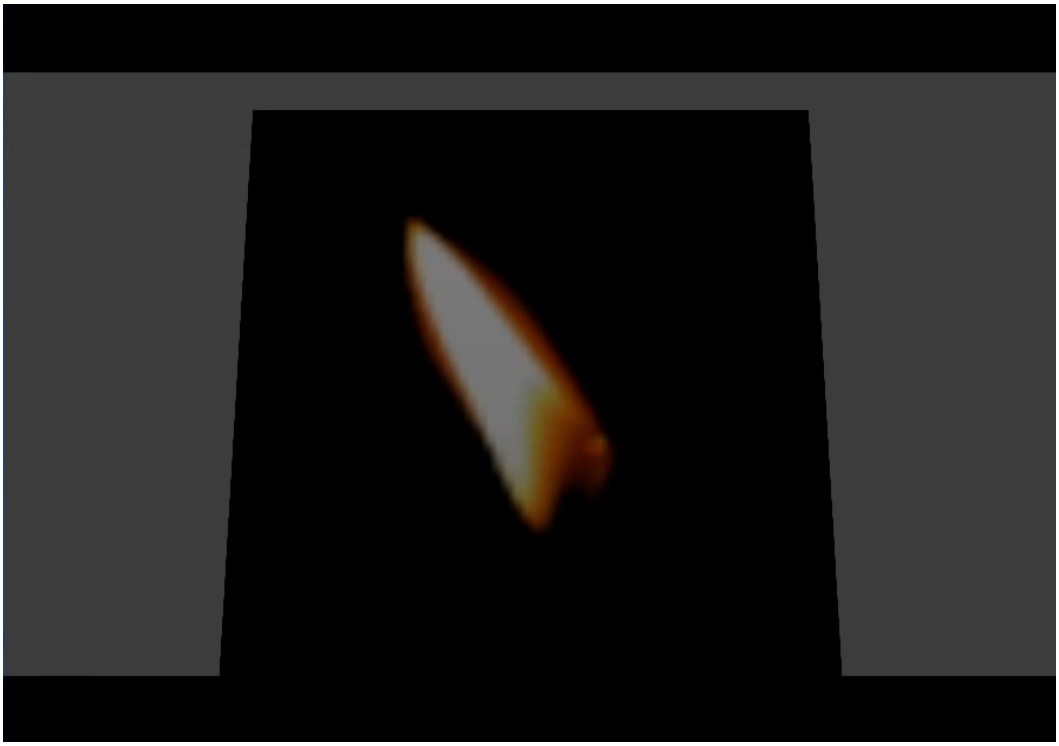
Con l'*unwrap* abbiamo fornito anche un *Layer UV* al Plane, per cui ora non ci resta che assegnare effettivamente questa immagine, con le informazioni sul *tiling*, ad una Texture. Aggiungiamo un Material all'oggetto, personalizzando a piacere gli ombreggiatori e le altre caratteristiche proprie del Material, dopodiché aggiungiamo una Texture di tipo immagine all'oggetto.



Aggiungere una Texture immagine all'oggetto

Qui dobbiamo caricare, ovviamente, la stessa immagine utilizzata per l'*unwrap* nella finestra *UV - Editor delle immagini*, selezionando la casella *Premultiply* nella sezione *Image* e specificando le coordinate di *mappatura UV* con il *Layer* creato automaticamente nella sezione *Mapping*; in *Influence* lasciamo la voce di default, *Diffuse Color* con intensità 1, in quanto questa immagine serve a definire l'aspetto dell'oggetto.

Non ci resta che premere *P* mentre il cursore del mouse si trova in una finestra *3D View* per osservare il risultato in tempo reale prodotto dal *Game Engine*; con le impostazioni date, l'animazione verrà eseguita a 1 frame per secondo, per cui per ottenere un risultato più interessante portiamo *Speed* (nel pannello *Properties* nella finestra *UV – Editor delle immagini*) ad esempio a 10 o 24; da notare che l'animazione verrà eseguita in "loop", in maniera ciclica.



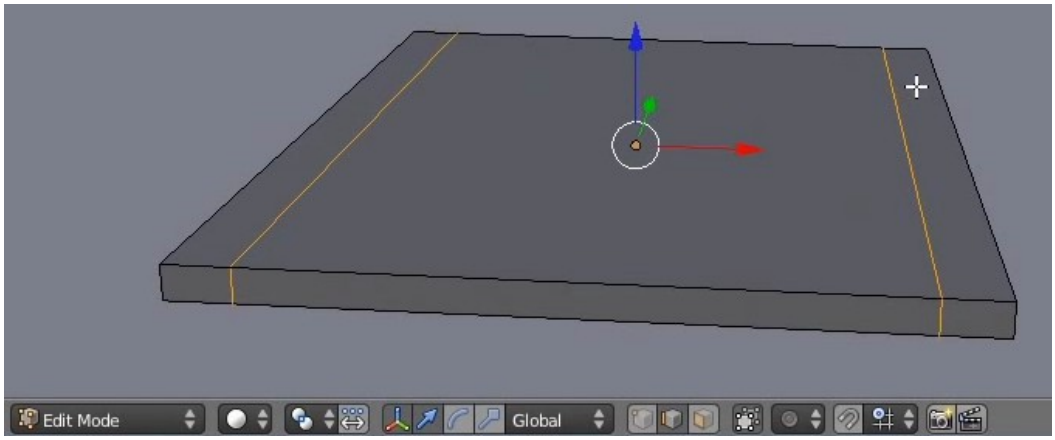
Screenshot dell'esecuzione del "gioco"

* * *

Modellare un portasapone con Loop Cut e Subsurface

In questo breve tutorial vedremo come modellare un oggetto per il bagno: un portasapone dalle forme arrotondate; niente di particolarmente complicato se si fa uso correttamente di SubSurf (*Subdivision Surface*) e di *Ring Cut* per fornire una buona "risoluzione", in termini di facce, alla mesh, come vedremo. Il tutorial è stato realizzato con la versione 2.59 stabile di Blender 3D.

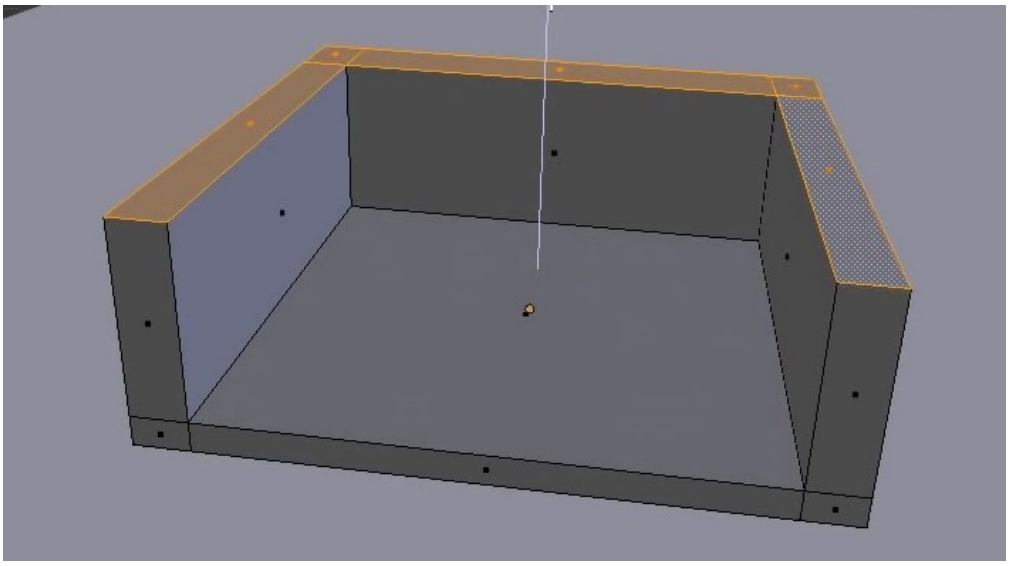
Iniziamo dalla base, inserendo il solito cubo e schiacciandolo in verticale, per definire lo spessore della base e modificandone lunghezza e larghezza con scaling selettivo su uno o due assi per volta. Passiamo alla prima applicazione di *Loop Cut*: premiamo *CTRL R* e muoviamo il mouse in modo da far apparire la linea di taglio in lunghezza, quindi ruotiamo la rotellina del mouse di uno scatto per creare due linee di taglio; facciamo click col tasto sinistro per confermare l'operazione e allontaniamo gli spigoli appena creati, che sono selezionati, con uno scaling limitato all'asse della larghezza: in questo modo stiamo definendo la posizione delle facce che verranno estruse per fare le pareti laterali del portasapone.



Creare e allontanare i "tagli" sulla base del portasapone, realizzata schiacciando e allargando un cubo

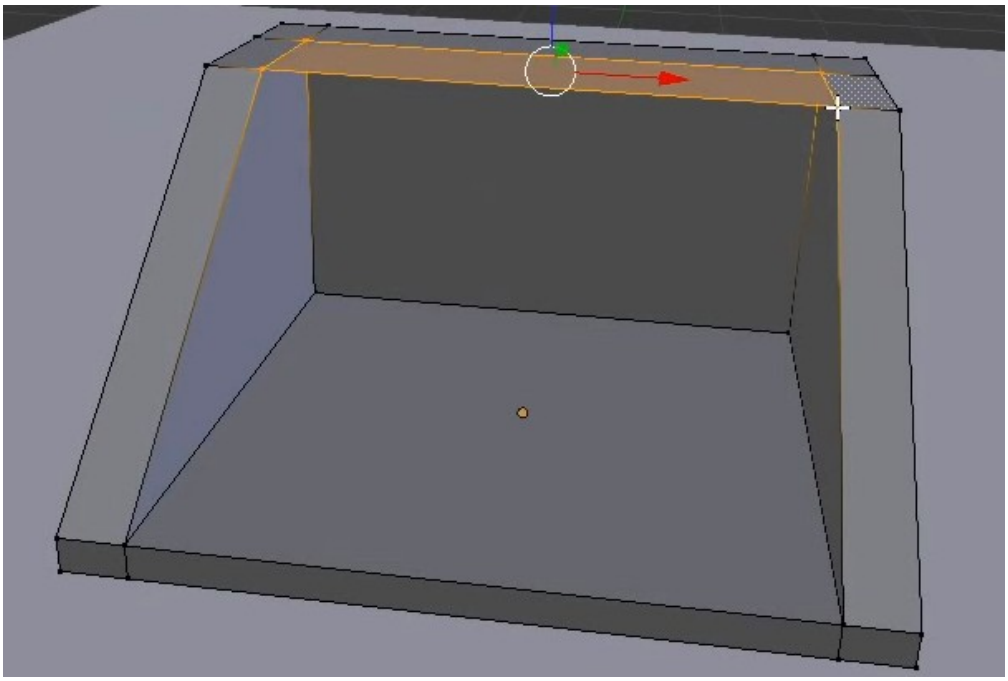
Con *Ring Cut* (*CTRL R*) facciamo un taglio nel verso della larghezza della base e posizioniamo tale taglio sul fondo: ci servirà per la parete posteriore.

A questo punto selezioniamo le facce laterali e posteriori appena create ed estrudiamole verso l'alto, realizzando le “pareti” del portasapone; spostiamo un po' indietro gli spigoli frontali, in modo da non creare pareti “dritte” ma leggermente oblique.



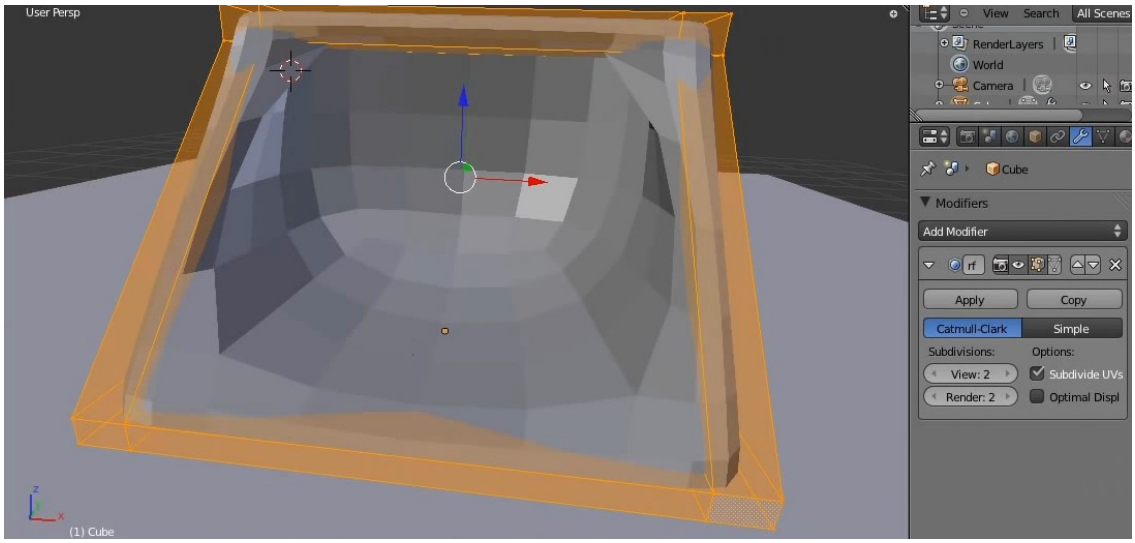
Estrusione verso l'alto delle facce ai bordi della superficie superiore del cubo

Se lo desideriamo, possiamo creare una faccia superiore per l'oggetto selezionando i 4 vertici aperti in alto e premendo *F*, per poi estrudere il tutto verso l'alto in modo da creare una base superiore; conviene, dopo aver fatto questa operazione, ricalcolare le normali dell'intera mesh, per sicurezza, selezionando tutti i vertici in Edit Mode e cliccando su *Recalculate Normal*, nella *Tool Shelf*.



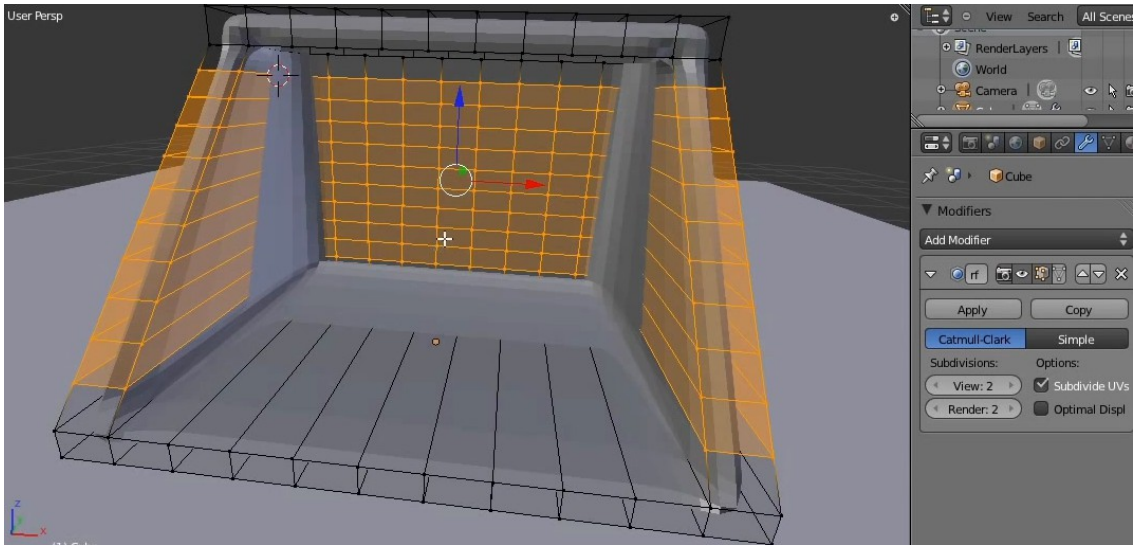
Chiudere la base superiore, con F, per poi estruderla, realizzando un volume

Aggiungiamo un modificatore *Subdivision Surface* per l'intero oggetto con 2 passi sia di *Preview* che di *Rendering*, in modalità *Catmull-Clark* (quella di default). Il risultato sarà orribile, e questo dipende soprattutto dalla presenza di poche facce nella mesh, tuttavia non conviene usare *Subdivide*, infatti sono presenti volumi piccoli, nelle basi, e volumi grandi, nelle pareti, e *Subdivide* finirebbe per creare poche facce dove ne servono molte (ossia ai lati e nella lunghezza delle basi) e molte dove ne servono poche (ossia negli angoli e nella larghezza delle pareti).



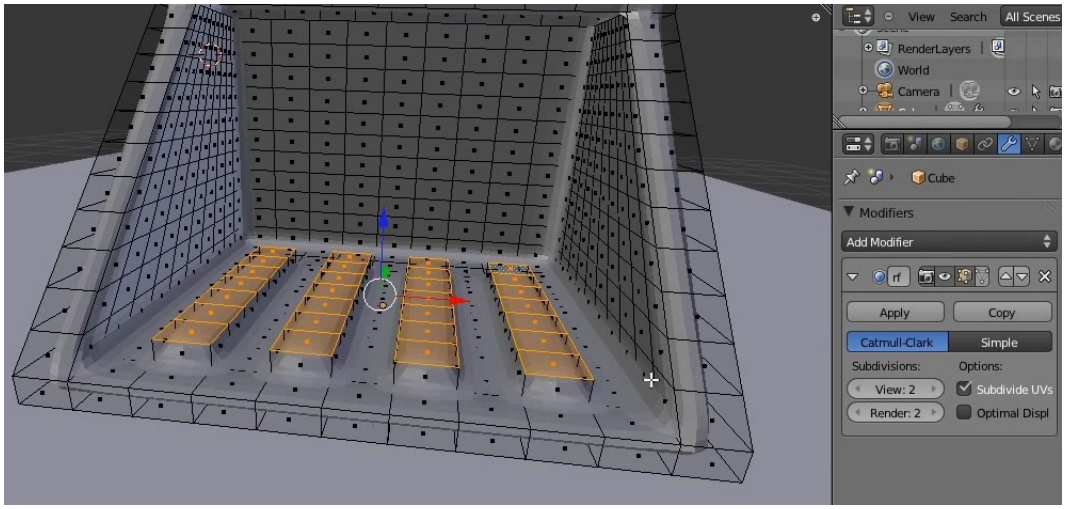
Il risultato dell'applicazione di Subdivision Surface

Torniamo quindi nella *3D View* e facciamo tre operazioni di *Ring Cut* multiplo (*CTRL R* e rotellina del mouse) per suddividere il modello nelle pareti, nelle basi e sul fondo, seguendo le linee di taglio suggerite da *Ring Cut* muovendo il mouse nella *3D View.*



Il risultato dopo aver effettuato alcuni tagli con Ring Cut

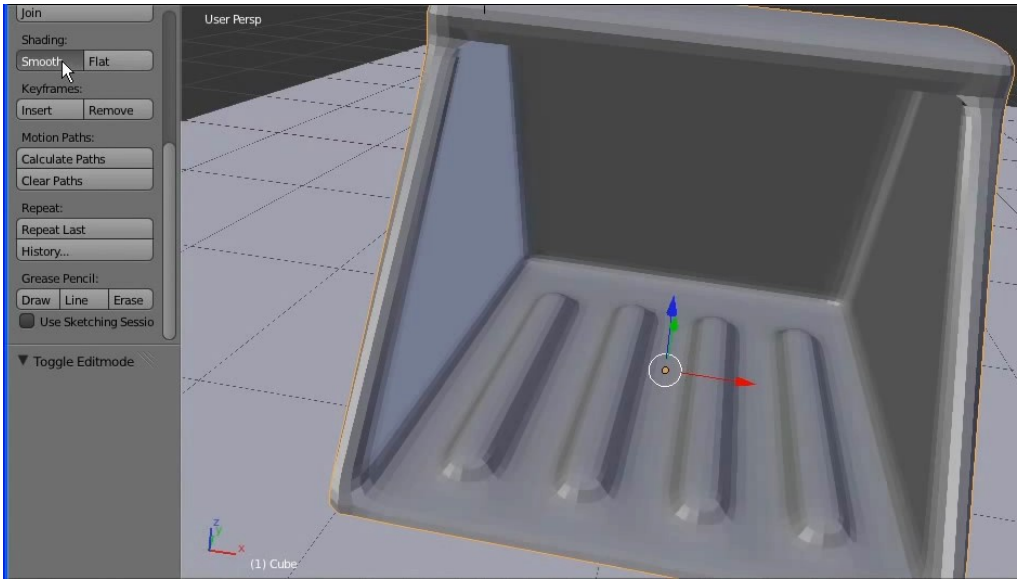
Quando abbiamo finito, selezioniamo alcune facce sulla base inferiore del portasapone, a strisce, quindi estrudiamole un po' verso l'alto, passiamo alla modalità *Pivot Point "Individual Origins"* e scaliamole un po', ottenendo come dei lingotti: si tratta di rilievi sul fondo del portasapone, utili per non far scivolare gli oggetti.



Estrusione verso l'alto, con Pivot Point "Individual Origin", di alcune facce della base inferiore interna del portasapone

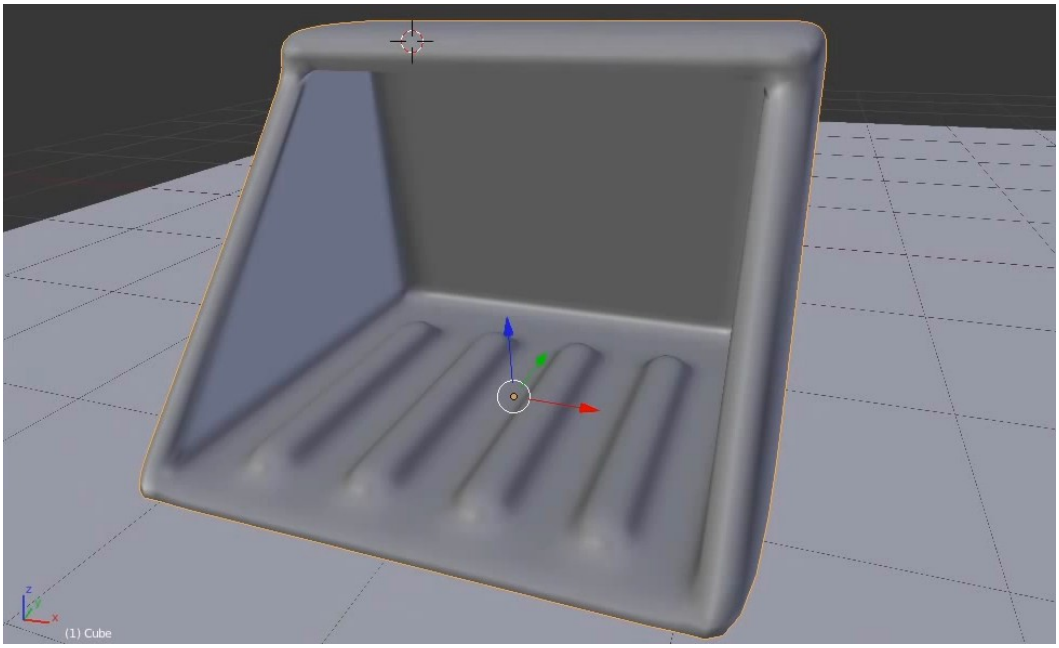
Sto usando *Individual Origins* perché l'opzione di default, *Median Point*, scalerebbe la selezione spostando tutti gli elementi verso il centro della stessa, mentre noi vogliamo scalare le facce connesse intorno ai singoli centri delle strisce, senza spostarle.

In fase di modellazione, questi "lingotti" sono squadrati: sono composti da poche facce, ma ci penserà *SubSurf* a smussarli... ...e, a proposito di smussare, torniamo in Object Mode e clicchiamo su *Smooth* nella *Tool Shelf* per levigare tutto il modello.



"Levigare" tutto il modello con Smooth, nella Tool Shelf

Le operazioni svolte e l'oggetto ottenuto sono davvero banali ma il tutorial serve più che altro a suggerire l'utilizzo di *Ring Cut* anziché dello stra-abusato *Subdivide* per creare delle "buone" suddivisioni di una mesh in vista dell'utilizzo di *Subdivision Surface* per arrotondare il tutto: un concetto che torna utile per modellare, ad esempio, divani, poltrone, elettrodomestici ed altri elementi.



Screenshot del risultato finale

* * *

Immagini HDR su Environment Lighting per oggetti riflettenti

Questo breve tutorial è dedicato all'utilizzo di *immagini HDR* su *Environment Lighting*, una tecnica particolarmente utile con oggetti dotati di materials riflettenti come acciaio, metallo cromato, telai di automobili, eccetera.

Come detto in diversi tutorials pubblicati precedentemente, infatti, questi Materials riflettono il colore dell'ambiente circostante e, se si renderizza un oggetto riflettente in una scena vuota con il solo *Sky Color*, si avranno oggetti dall'aspetto banale, privi di riflessi interessanti, mentre in questo modo possiamo aggiungere sfumature di colore anche su superfici prive di particolari interessanti e senza dover aggiungere un set, una scena allestita (tra l'altro, pesante da renderizzare) intorno all'oggetto. Il tutorial è stato realizzato con la versione 2.59 stabile di Blender 3D.

Le impostazioni della scheda World, prima di aggiungere una Texture immagine proprio per l'universo virtuale

Quello che ci serve è un'immagine panoramica, a 360 gradi, in *HDR*. Al formato *HDR* e al salvataggio dei dati in formati come l'*EXR* ho già dedicato altri tutorials, sia per Blender 3D che per Photoshop, che trovate negli elenchi dei tutorials del mio sito web e del mio canale Youtube, quindi qui non parlerò delle immagini *HDR*.

Nella *Properties Window* selezioniamo la scheda *World* e, qui, selezioniamo *Real Sky*, dopodiché passiamo alla scheda Texture, aggiungendo una Texture di tipo immagine per *World*, l'ambiente.

Come immagine, utilizziamo la foto panoramica a 360° gradi, su Internet trovate molte immagini HDR e a seconda delle circostanze potete scegliere se utilizzare immagini di locali, boschi, spiagge, alta montagna, eccetera.

Le impostazioni della Texture immagine HDR impostata per World

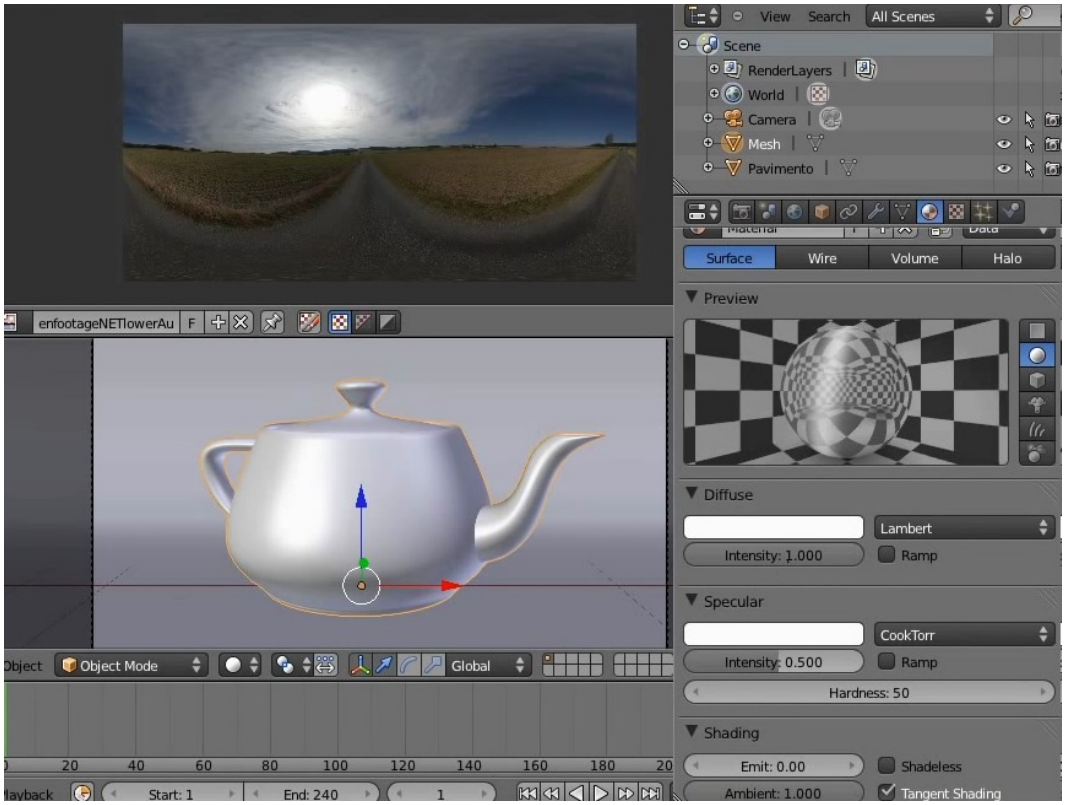
Nella sezione *Colors* portiamo la luminosità *Brightness* ad esempio a 2 e il *contrasto* a 3, ma questi sono solo valori indicativi, potete personalizzarli come preferite; in *Mapping* scegliamo *AngMap*, che proietta correttamente la Texture panoramica, quindi selezioniamo *Blend* e *Horizon* in *Influence*.

Torniamo alla scheda *World* e selezioniamo la voce *Environment Lighting*, illuminazione ambientale, cambiando però l'*Environment Color* da *White*, voce di default, a *Sky Texture*.

Environment Color - Sky Texture, per World

In questo modo lo sfondo dell'universo virtuale verrà renderizzato, ma potete eliminarlo in Compositing con i Nodes; per maggiori informazioni su questo punto, rimando ai vari tutorials sul Nodes Compositing e all'*ebook "Il compositing dei Nodes in Blender 3D 2.5"*.

Allestiamo la scena fornendo un Material riflettente di base all'oggetto o agli oggetti che ci interessano; ovviamente, potete utilizzare un qualsiasi Material: qui mi sto limitando alle impostazioni di base.



La scena di prova utilizzata in questo tutorial

Il rendering richiederà MOLTO tempo, soprattutto se fatto con *Gathering Method Raytrace*, un po' meno se con *Approximate*. Il rendering visibile qui di seguito è stato realizzato senza fonti di luce classiche nella scena ma attivando *Ambient Occlusion* con *Gathering Method Approximate*.

Un rendering d'esempio

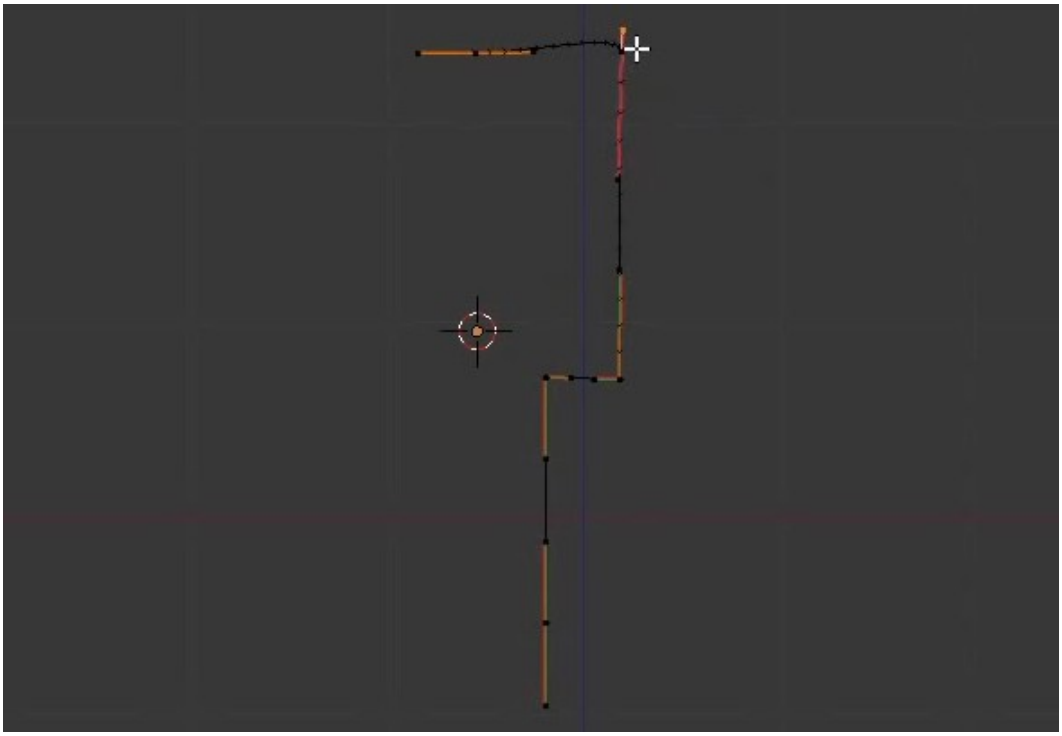
* * *

Modellare e texturizzare l'oculare di un telescopio

In questo tutorial vedremo come modellare e texturizzare un oculare per telescopio in Blender 3D 2.5, trattando argomenti come *Spin*, i materiali multipli e l'utilizzo di immagini come Textures per serigrafie. Vi anticipo qual è il risultato che vogliamo ottenere: è l'oggetto visibile nell'immagine seguente; come è possibile notare, l'oggetto è composto principalmente da tre parti: una base cilindrica in metallo riflettente, un altro cilindro in plastica scura e la lente, con un buon effetto di rifrazione dei raggi luminosi, inoltre abbiamo un'immagine / logo sulla parte cilindrica in metallo.

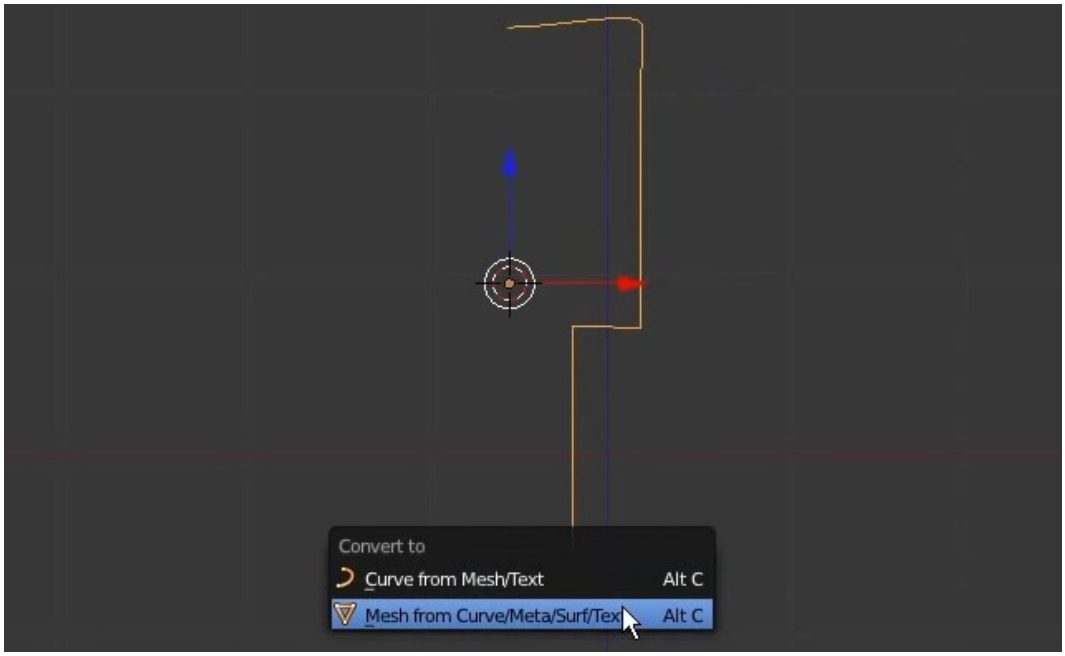
Il risultato finale

Realizzeremo un'unica mesh, variando poi l'aspetto delle varie aree mediante l'utilizzo dei materiali multipli. Iniziamo quindi definendo il profilo dell'oculare mediante una curva *Bezier* aperta nella vista frontale, realizzando solo metà profilo, in quanto useremo *Spin* a 360°.
Iniziamo da quello che sarà il punto centrale della lente e procediamo definendo il profilo dell'oggetto; non c'è bisogno, in questa fase, di dare uno spessore alle parti mediante una curva chiusa: utilizzeremo *Solidify* in seguito.



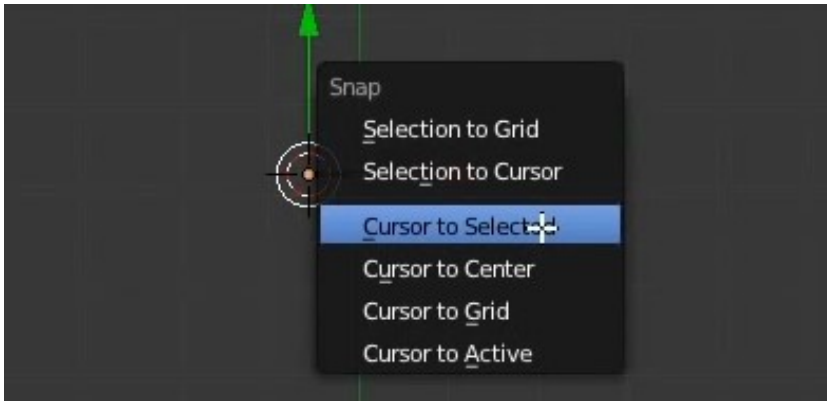
Fasi iniziali della modellazione, con la curva Bezier per modellare il profilo dell'oggetto

La base in metallo è rientrata rispetto a quella in plastica perché è la parte dell'oggetto che dovrà inserirsi nel telescopio. Quando abbiamo finito di definire il profilo, convertiamo la *Curva* in mesh con *ALT C* in *Object Mode* e scegliendo *Mesh from Curve* dal menù che apparirà a video.



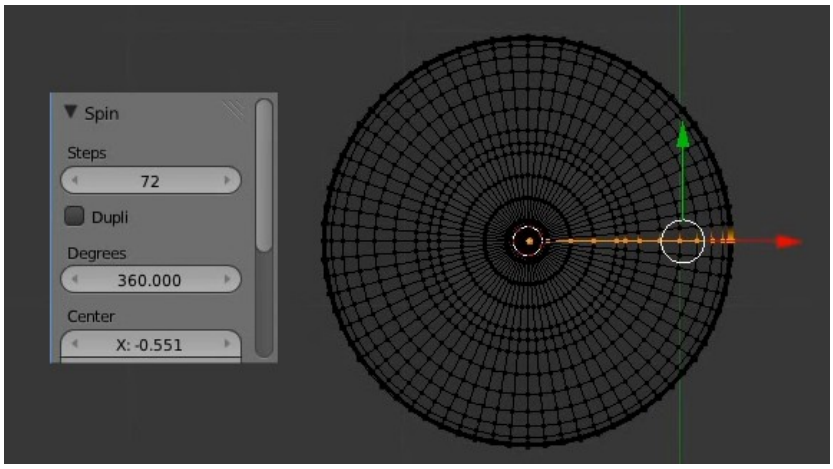
Conversione del profilo in mesh con ALT C (in Object Mode)

A questo punto in Edit Mode selezioniamo il vertice che corrisponde al centro della lente, spostiamoci nella vista *TOP*, premiamo SHIFT S e scegliamo *Cursor to Selected* dal menù che apparirà a video: il *3D Cursor*, infatti, identificherà la posizione dell'asse di rotazione, con l'asse che sarà perpendicolare al piano attualmente visibile, che è *TOP*, infatti vediamo l'oggetto dall'alto.



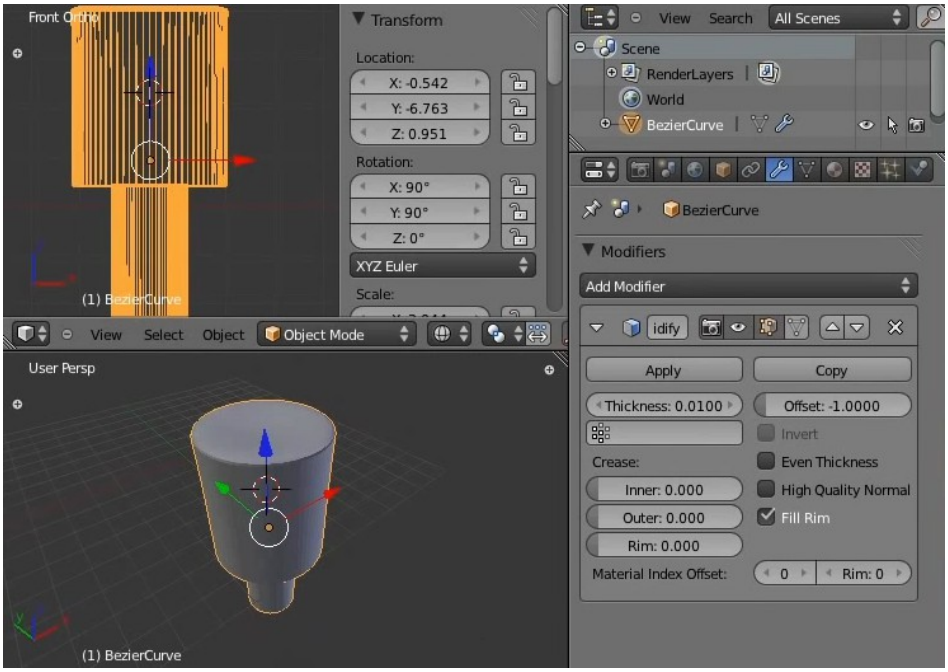
Posizionamento del 3D Cursor in un punto al centro di quello che sarà l'oculare, con un'inquadratura Top Ortho, pronti per utilizzare SPIN

Selezioniamo tutti i vertici con *A* e clicchiamo su *Spin* nella *Tool Shelf*, scegliendo 360° e un buon numero di passi, ad esempio 72. Dopo aver cliccato su *Spin*, selezioniamo tutti i vertici della mesh, clicchiamo su *Remove Doubles* ed impostiamo un buon valore di soglia (*threshold*, in basso nella *Tool Shelf* dopo aver cliccato su *Remove Doubles*) per rimuovere i vertici duplicati.



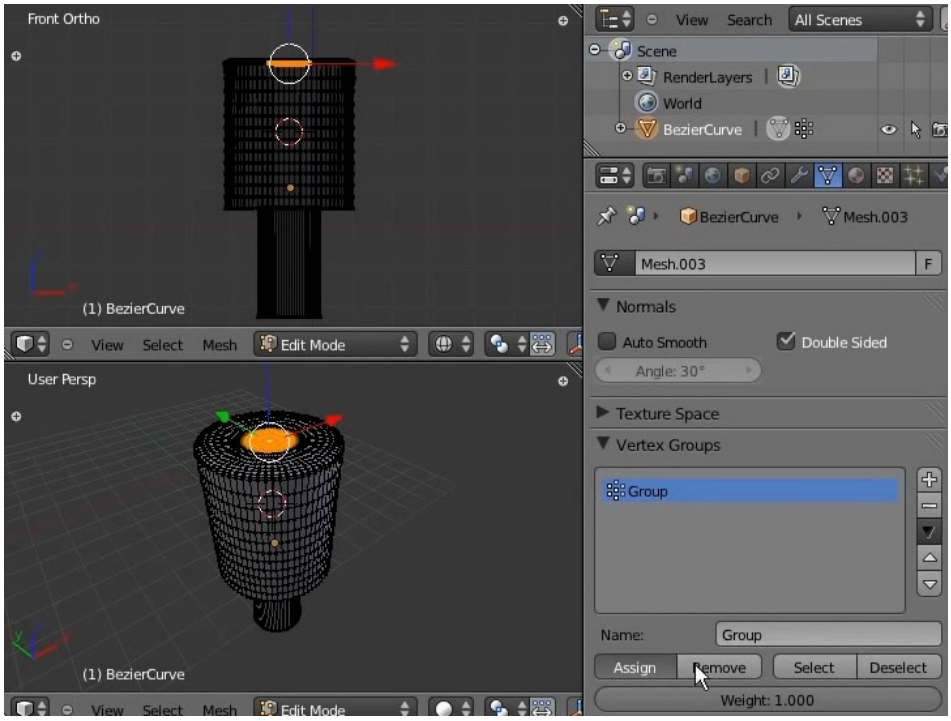
Il risultato dopo aver applicato SPIN (su tutti i vertici selezionati in una vista Top Ortho)

Passiamo in Object Mode e apriamo la scheda *Modifiers*, nella *Properties Window*, aggiungendo un modificatore *Solidify*; in generale, anche le impostazioni di default andranno più che bene, per cui applichiamo subito il modificatore, che conferirà un po' di spessore all'oggetto, cosa importante soprattutto per gli effetti di rifrazione della lente.



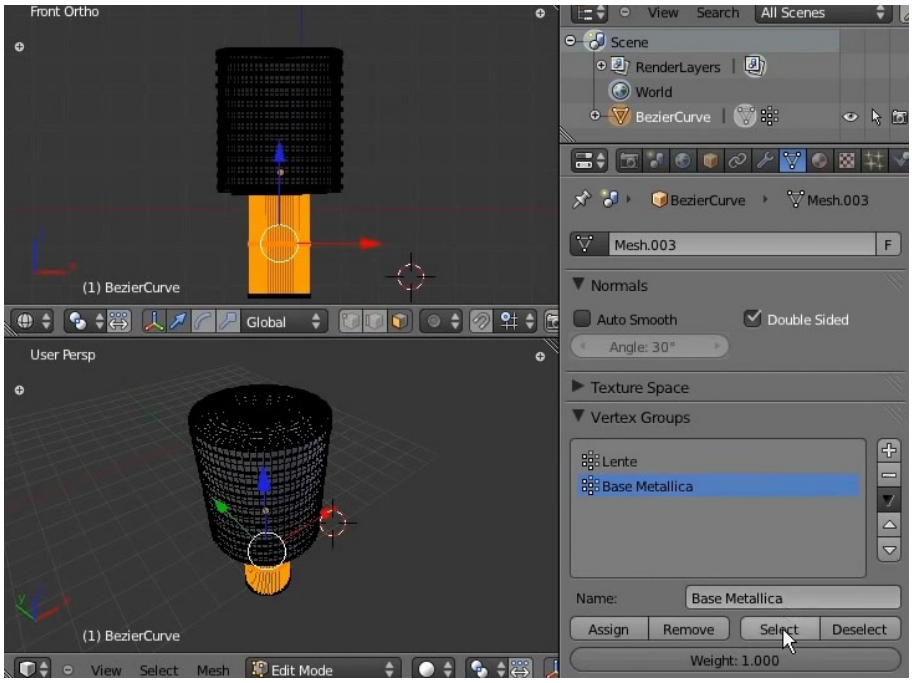
Utilizzo di Solidify

Torniamo alla vista *TOP*, entrando in modalità di visualizzazione *Wireframe*, per iniziare a separare le varie parti della mesh, iniziando dalla lente: in Edit Mode, attiviamo la selezione circolare con *C* e selezioniamo le facce che fanno parte della lente, quindi aggiungiamole ad un *Vertex Group* (per comodità) creandone uno nella scheda *Object Data*, nella *Properties Window*, con la "sequenza" *Select – Assign – Deselect* e rinominando il gruppo, ad esempio, in "*Lente*".

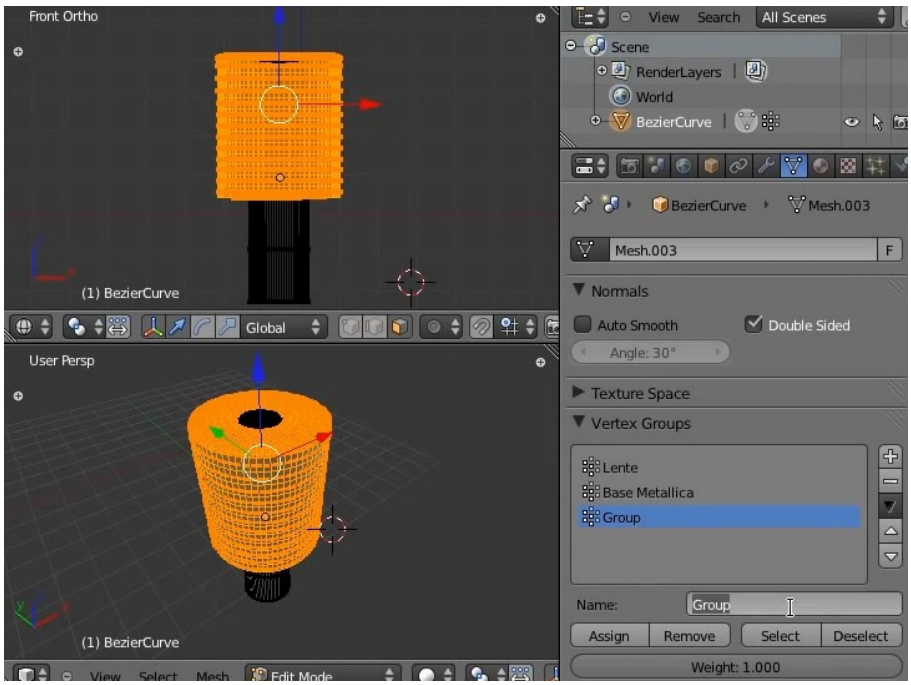


Assegnare la parte della lente ad un Vertex Group dedicato

Da una vista frontale o laterale, selezioniamo ora le facce della base metallica, sempre in *Wireframe* (in modo da selezionare anche le facce dall'altra parte) e aggiungiamole ad un *Vertex Group* chiamato, ad esempio, "*base metallica*".

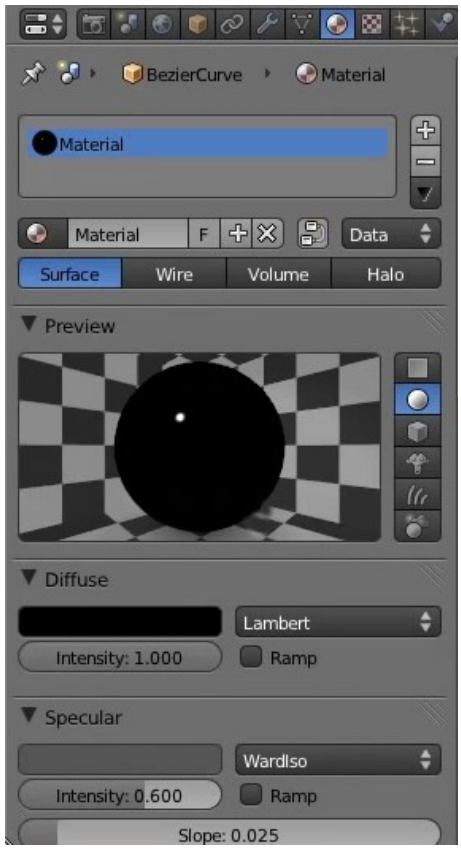


Assegnare la "base metallica" ad un altro Vertex Group



Per selezionare i vertici rimanenti, selezionare i vertici dei due Vertex Group creati e premere CTRL I (o scegliere "Select - Invert" dall'header di una 3D View)

Per selezionare facilmente le facce della parte in plastica, procediamo come segue: selezioniamo sia il *Vertex Group "Lente"* che il *Vertex Group "Base metallica"*, quindi premiamo *CTRL I* in una *3D View* per invertire la selezione e creiamo un nuovo *Vertex Group*, chiamato ad esempio "*plastica*", assegnandogli la selezione.



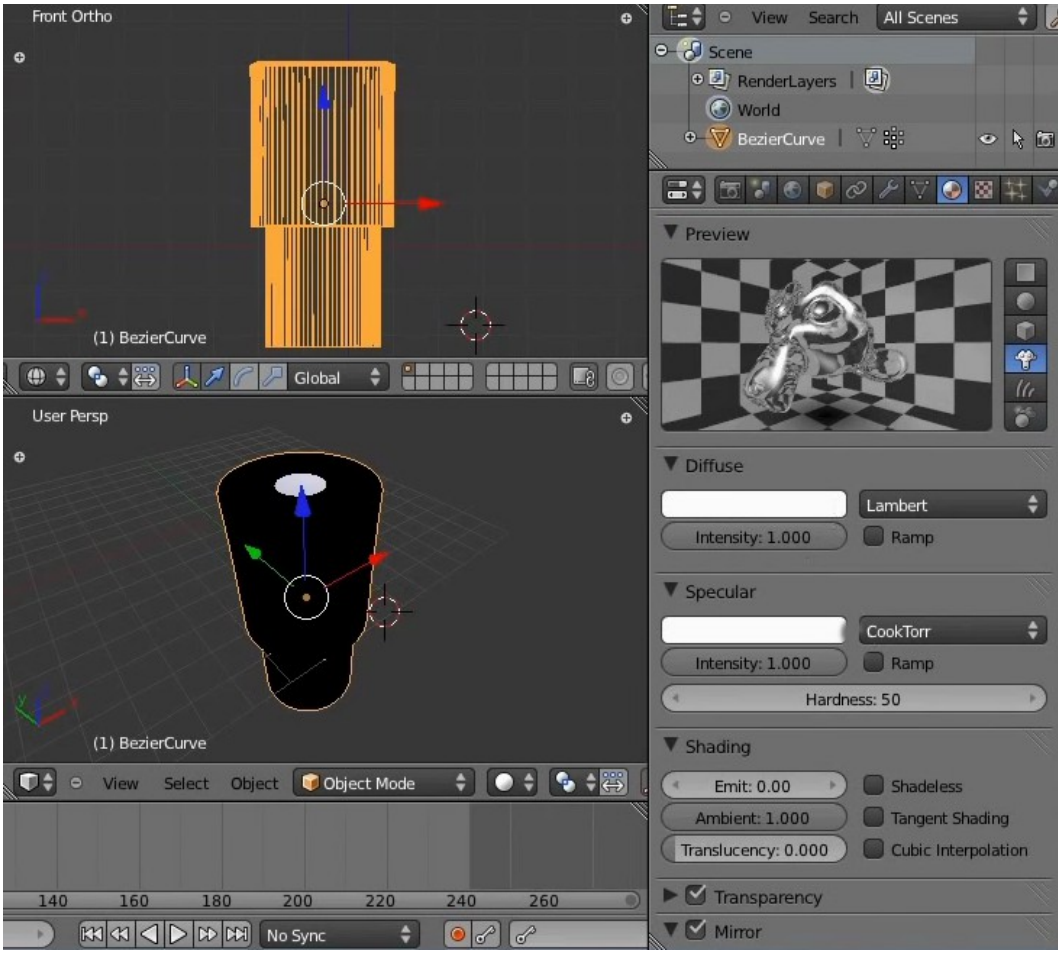
Il primo Material (plastica nera leggermente riflettente)

Con la suddivisione delle zone abbiamo finito, per cui passiamo alla definizione dei Materials.
In realtà, si tratta di un unico Material composto da tre sottomateriali.
Iniziamo assegnando un materiale a tutta la mesh, poi passeremo agli altri.

Il primo materiale è quello della plastica nera riflettente; ecco le impostazioni: ombreggiatore diffuso *Lambert* con intensità 1.0, colore nero; ombreggiatore speculare *WardIso* con colore grigio scuro (ad esempio 0.1 per tutti e tre i canali RGB), intensità 0.6, *slope* 0.025; riflessi *Mirror* attivi con *Reflectivity* 0.05 e colore dei riflessi 0.05 per tutti e tre i canali RGB.

A questo punto passiamo in Edit Mode, selezioniamo il *Vertex Group "Lente"*, passiamo alla scheda Materials e aggiungiamo un secondo Material nello slot di quello associato all'intero oggetto, creando così un secondo sotto-materiale, quindi clicchiamo su *Assign* per assegnare questo secondo sottomateriale ai vertici del *Vertex Group "Lente"* e deselezioniamo tale *Vertex Group*, per evitare di fare confusione in seguito.

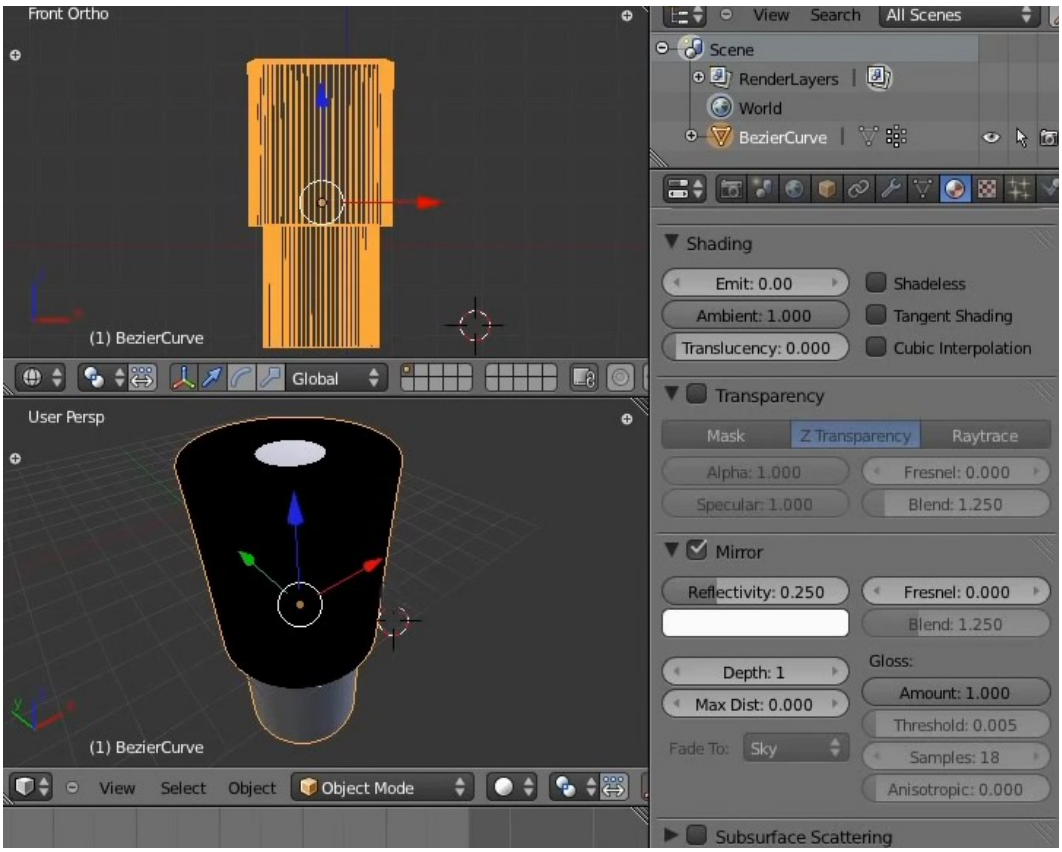
Impostiamo il materiale della lente come segue: ombreggiatori diffuso e speculare di default, cioè *Lambert* e *Cook-Torr*, con colore bianco puro e intensità 1 in entrambi i casi; trasparenza *RayTrace* attivata con *Alpha* 0.0 e *IOR* (indice di rifrazione del materiale) 1.5 (è un po' esagerato ma serve ad ottenere un effetto più interessante); riflessi *Mirror* attivati con *Reflectivity* 0.05 o meno con colore bianco puro; le altre impostazioni come di default.



Il sottomateriale per la lente

Passiamo quindi alla terza parte della mesh selezionando i vertici della base metallica mediante il *Vertex Group* creato precedentemente e aggiungiamo un terzo sottomaterial anche per questa parte.

Impostiamo il terzo Material in questo modo: ombreggiatore diffuso *Lambert* con intensità 1 e colore grigio (es.: 0.3 per tutti e tre i canali RGB); ombreggiatore speculare *WardIso* con intensità 1, *slope* 0.2, colore bianco puro; riflessi *Mirror* attivati con *Reflectivity* 0.25, le altre impostazioni come di default; eventualmente, potete provare ad attivare *Tangent Shading*.

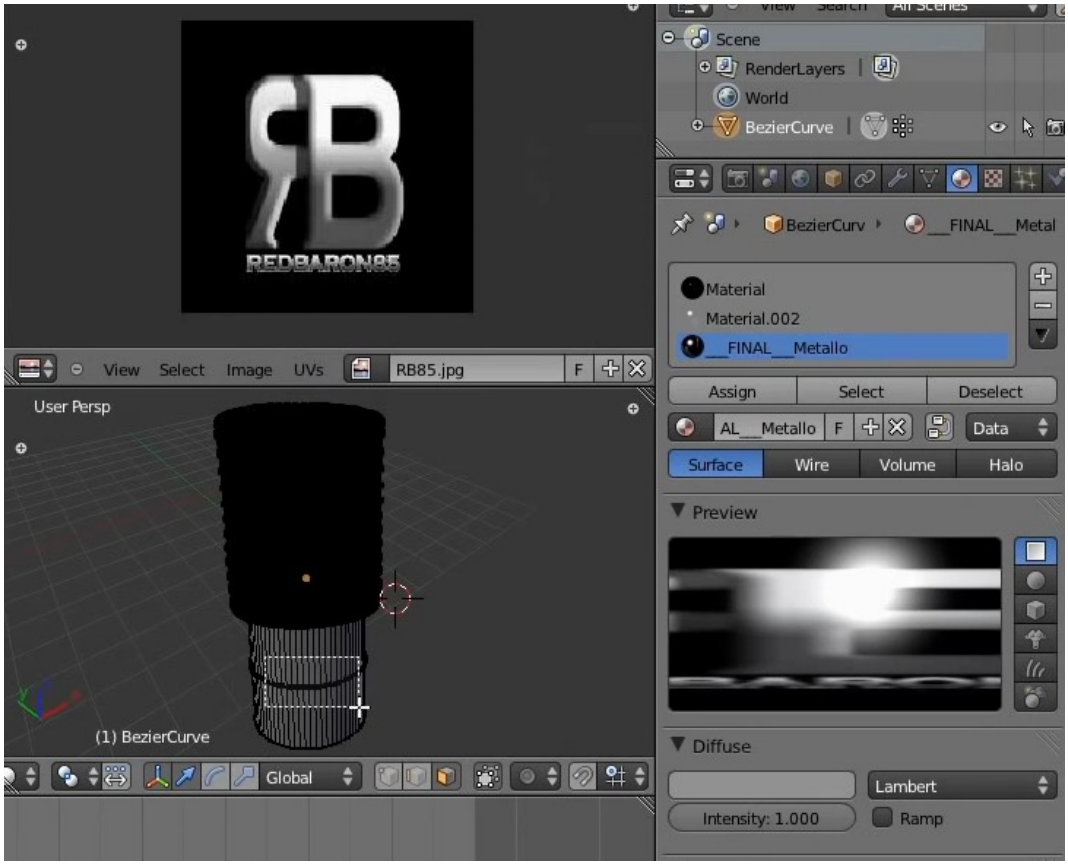


Il terzo sottomateriale, per la parte metallica

Non abbiamo ancora finito: manca il logo su una parte della base metallica.

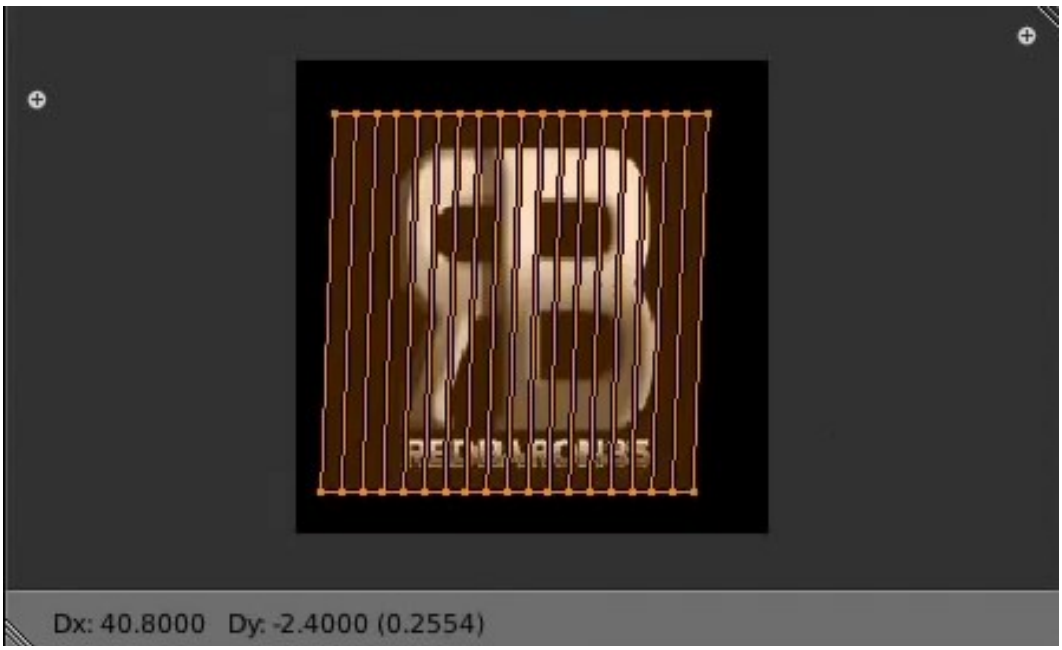
Il logo è un'immagine, che nel mio caso ho caricato precedentemente nella scena, ma ovviamente ora vedremo come fare per caricarne una e applicarla all'oggetto mediante *unwrapping* e Texture con *mappatura UV.*

Per prima cosa, in Edit Mode selezioniamo tutte le facce della mesh, premiamo *U* e scegliamo "*UNWRAP*", quindi cambiamo una finestra in *UV – Editor delle immagini* e carichiamo l'immagine da utilizzare con *Open* e scegliendo il file da disco nella finestra stile esplora risorse che apparirà oppure mediante il menù *Browse Image to be Linked*, se l'immagine, come nel mio caso, era già presente nella scena.



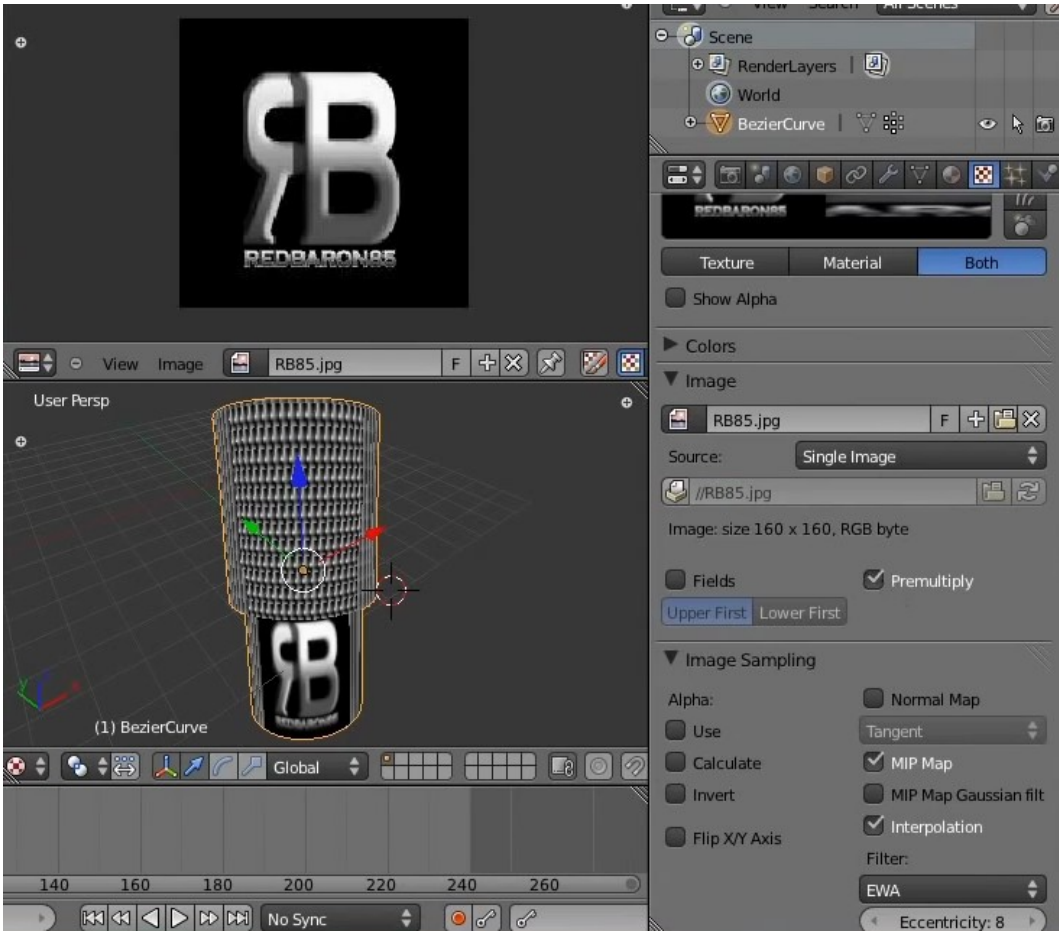
Caricare l'immagine da utilizzare come Texture nella scena

Posizioniamo tutte le facce fuori dall'area dell'immagine, quindi torniamo in una *3D View* e selezioniamo, questa volta, solo le facce da ricoprire con l'immagine, rifacciamo l'*unwrap* e torniamo nella finestra *UV – Editor delle immagini*.



Unwrapping delle facce da ricoprire con la Texture

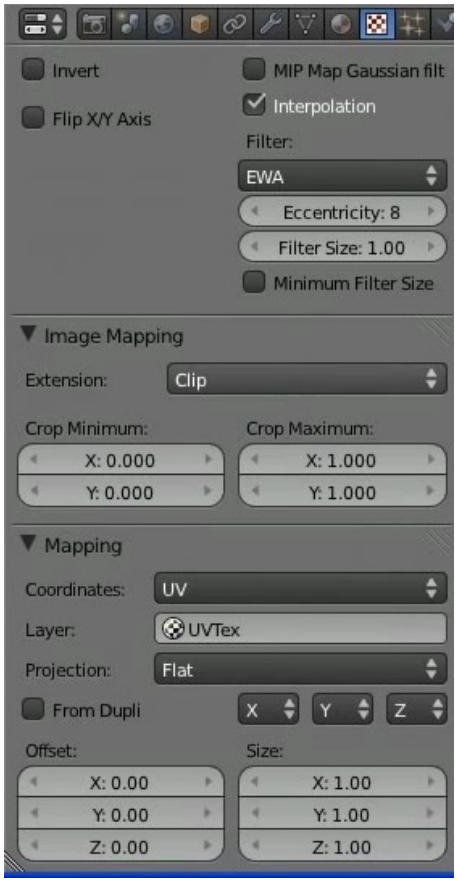
Posizioniamo le facce unwrappate in modo da mappare correttamente la mesh; per vedere un'anteprima del risultato in tempo reale, cambiamo la modalità di visualizzazione di una *3D View* in *Textured*: non preoccupatevi se vedete l'immagine anche in altre parti della mesh, perché in fase di rendering non succederà... anzi, lanciando un rendering ora non vedremo proprio nulla, perché prima dobbiamo creare una vera e propria Texture e assegnarla al Material (ed anzi, attenzione a partire dal giusto Material, quello della base metallica); finora, infatti, abbiamo definito solo la *mappatura UV* e creato automaticamente un *Layer UV*, non la Texture vera e propria.



La visualizzazione Textured, in una 3D View

Aggiungiamo quindi una Texture di tipo immagine caricando l'immagine utilizzata per l'unwrapping e presente ora nel menù *Browse Image to be Linked*, quindi selezioniamo *Premultiply*, deselezioniamo *Use Alpha* (in questo caso comunque non c'è un canale *Alpha*), cambiamo la mappatura in *Clip* con coordinate *UV* e il *Layer* creato automaticamente, quindi selezioniamo sia *Diffuse Color* che *Mirror* in *Influence* (quest'ultima voce, con valore di default 1.0, per modulare i riflessi nelle zone ove è presente la Texture).

Con questa tecnica è possibile mettere loghi, serigrafie o scritte informative sugli oggetti, ricorrendo all'*unwrap* per una mappatura precisa e sfruttando tra l'altro il canale *Alpha* per le immagini che lo prevedono. Per maggiori informazioni sull'utilizzo di Materials e Textures e ad altre modalità di mappatura in Blender 3D 2.5, rimando ai miei due *ebook* su tali argomenti.



Le impostazioni della Texture immagine

Per quanto riguarda il rendering, se renderizzate in una scena vuota vi consiglio di aggiungere qualche elemento per creare riflessi interessanti sulla parte metallica e di cambiare il colore dello sfondo, *World Horizon*, in quanto influisce parecchio sul colore dei Materials "*Lente*" e "*Base Metallica*"; potete ricorrere anche ad immagini *HDR* su *Environment Ligthing* come spiegato nel tutorial precedente.

L'immagine che ho realizzato (vista all'inizio del capitolo) è stata renderizzata con due fonti di luce: una *Area* posta nello stesso punto dove si trova la telecamera e diretta verso il centro dell'insieme di oggetti, e una luce *omnidirezionale* posta sopra gli oggetti; entrambe le lampade proiettano ombre *Ray Shadows* con *16 Samples*; il colore dello sfondo era grigio molto chiaro e ho smussato solo le facce laterali dei cilindri con *Smooth* per non creare distorsioni in prossimità delle basi superiori (argomento spiegato all'inizio dell'*ebook sui Materials*).

* * *

* * *
* * *