## Premessa

***"BLENDER 3D tutorials – Volume 3 (versioni 2.6x)"*** è un pdf contenente le versioni testuali di **60 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 511 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.

Per il sommario completo: http://www.francescomilanese.com/blender3D-03-sommario.pdf

## SOMMARIO

1. È uscito B3D 2.6: niente panico! Novità e due tutorial 1
2. L'Add-On Screencast Keys 8
3. Altoparlanti nella scena 3D: gli oggetti Speaker e il Mixdown 11
4. Messa a fuoco selettiva con il nodo Defocus 16
5. Il bicchiere mezzo pieno: liquidi statici in contenitori 20
6. Risoluzione e dimensioni per la stampa dei rendering 27
7. Da Yafaray a Blender a Cycles... per caso 30
8. Immagine “illuminante” su World in Blender e Yafaray 0.1.2 34
9. Cycles per Blender 3D (tutorial con file blend in regalo) 38
10. Un trucchetto per velocizzare le operazioni con il Modifier Boolean 59
11. Compositing: animazioni con canale Alpha e Alpha Over 63
12. Cosa sono i file IES e le IES Lights; esempi con Yafaray 68
13. Animazione proiettili “stile Matrix” per Davidekyo 73
14. Le basi del Game Engine, il motore di gioco di Blender 2.6 – 1 79
15. Le basi del Game Engine, il motore di gioco di Blender 2.6 – 2 91
16. Le basi del Game Engine, il motore di gioco di Blender 2.6 – 3 110
17. Le basi del Game Engine, il motore di gioco di Blender 2.6 – 4 131
18. Le basi del Game Engine, il motore di gioco di Blender 2.6 – 5 149
19. Le basi del Game Engine, il motore di gioco di Blender 2.6 – 6 163
20. Le basi del Game Engine, il motore di gioco di Blender 2.6 – 7 179
21. Le basi del Game Engine, il motore di gioco di Blender 2.6 – 8 188
22. Le basi del Game Engine, il motore di gioco di Blender 2.6 – 9 200
23. Le basi del Game Engine, il motore di gioco di Blender 2.6 – 10 209
24. Il Video Sequence Editor, per montare clip audio, video, … 217
25. Riprodurre filmati, file video e animazioni nel Game Engine 227
26. Lock Camera to View e Align Camera to View 232
27. La spada laser: animazioni Shape Keys e bagliore animato 235
28. Modellare e renderizzare un cacciavite con Looptools Loft 245
29. Old Fashioned: tutorial Cycles su bicchieri e liquidi statici 253
30. Modellare l'elica di un aereo o di una nave con Spin e Twist 265

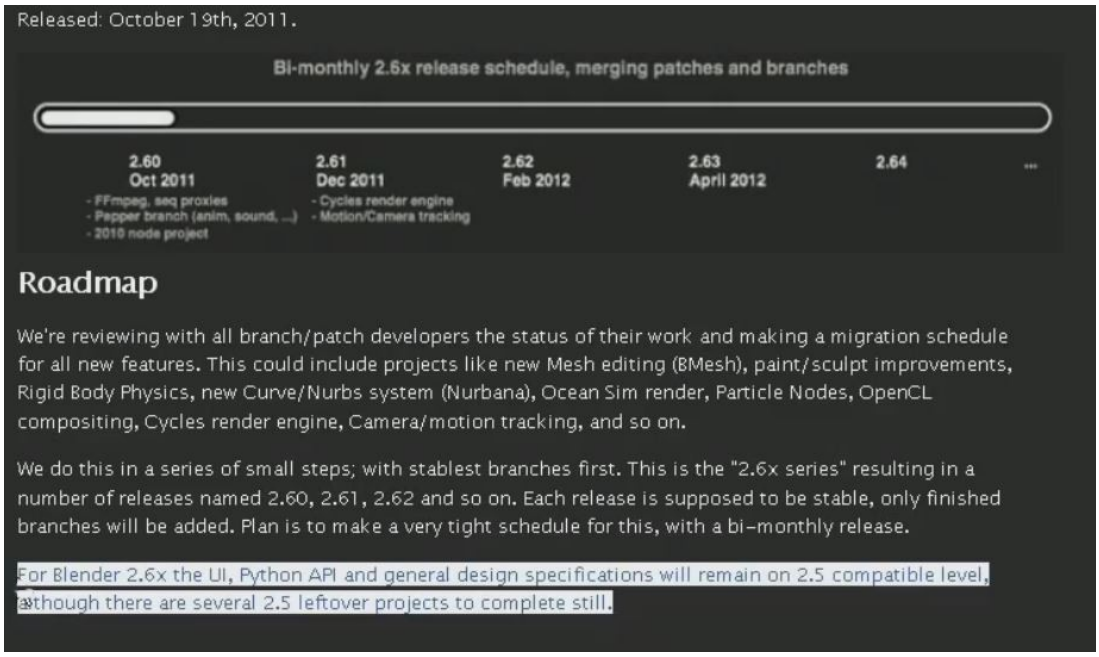
31. Unwrap e UV Mapping (e Texturing) di una scatola in Cycles 273
32. Un dado a sei facce in Blender 3D e Cycles 282
33. La scala di Giacobbe: arco elettrico con Hook, Wave e Compositing 290
34. Globuli rossi (red blood cells) in Blender 3D e Cycles 301
35. La scritta sul muro: Dynamic Paint e Nodes Compositing 313
36. Cambiare telecamera attiva al volo mediante i Marker 331
37. Blender compresse solubili: Array, Subsurf, Mean Crease 335
38. Rendering concept stile Sketch Up con AO ed Edge 345
39. Saturno in Blender 3D – I parte 349
40. Saturno in Blender 3D – II parte 363
41. Mappare textures Equirectangular e Angmap su World 374
42. Ricavare e utilizzare il canale Vertex Paint in Cycles 379
43. Realizzare un Material Shadeless in Cycles 385
44. Utilizzare i Limits nel modificatore Simple Deform 388
45. Renderizzare sezioni di oggetti o scene 395
46. Modellare un Gulfstream G650 398
47. Copie radiali con Spin, Pivot Point e Array intorno a Empty 421
48. Il nuovo strumento Wireframe per le mesh 428
49. Le lenti degli occhiali: tips & tricks di modellazione e rendering 432
50. Filtri di fotoritocco e post-produzione animabili nel VSE 436
51. Cell Fracture Tool e animazione fisica con Game Engine 439
52. Come modellare una pallina da golf con Icosphere e Bevel 448
53. Tips & Tricks: Edge Slide 453
54. Math Function e Array: modellare portici 457
55. Come fermare un sistema Emitter con Damp: una galassia a spirale 466
56. Script Python: un piccolo planetario – I parte 477
57. Script Python: un piccolo planetario – II parte 483
58. Hair e Strand Render. Realizzare un tappeto in B3D e Cycles 490
59. Mappe ambientali su Reflection per rendering veloci in Cycles 495
60. Modellare una punta da trapano con Bezier Circle, Screw, Bridge 502

*  *  *

# È uscito B3D 2.6: niente panico! Novità e due tutorial

Il 19 Ottobre 2011 la Blender Foundation ha rilasciato la versione 2.6 di Blender 3D; in questo articolo parleremo delle novità rispetto alla serie 2.5 e vedremo anche due minitutorial sul Nodes Compositing e sulla programmazione di script Python per Blender, visto che gli argomenti trattati nei due ebook realizzati con la versione 2.59 sono ancora validi con la 2.6.

In realtà, non ci sono grandi cambiamenti rispetto alla versione 2.59: abbiamo ancora le solite finestre nella Properties Window, i motori di rendering di default sono quello di Blender e il Game Engine, i Materials sono di quattro tipi (Halo, Volume, Surface e Wireframe) con le schede che conosciamo bene (Diffuse, Mirror, Transparency, ecc...); idem per le Texture; i motori particellari sono Hair ed Emitter, i simulatori fisici Force, Cloth, Fluid eccetera sono al loro posto... vediamo quindi quali sono le principali novità introdotte con la versione 2.6 così come indicato nel sito ufficiale del programma, blender.org.




*Release Notes da Blender.org*

In questa schermata, leggiamo che per Blender 2.6, la UI (interfaccia grafica), le API Python per la programmazione di script e Add-Ons e la struttura generale resteranno compatibili con la versione 2.5, mentre verranno completati dei progetti previsti già per la versione 2.5 ma messi da parte per rispettare la timeline.

Le novità della versione 2.6 riguardano i seguenti argomenti: l'audio, l'animazione, la traduzione dell'interfaccia grafica e il Weight Painting.

Sono presenti adesso degli oggetti "Audio Speaker", cioè altoparlanti, sorgenti sonore da posizionare all'interno della scena, il cui output (in arrivo da file audio ma posizionato nello spazio 3D) potrà essere registrato su file nelle animazioni prodotte.

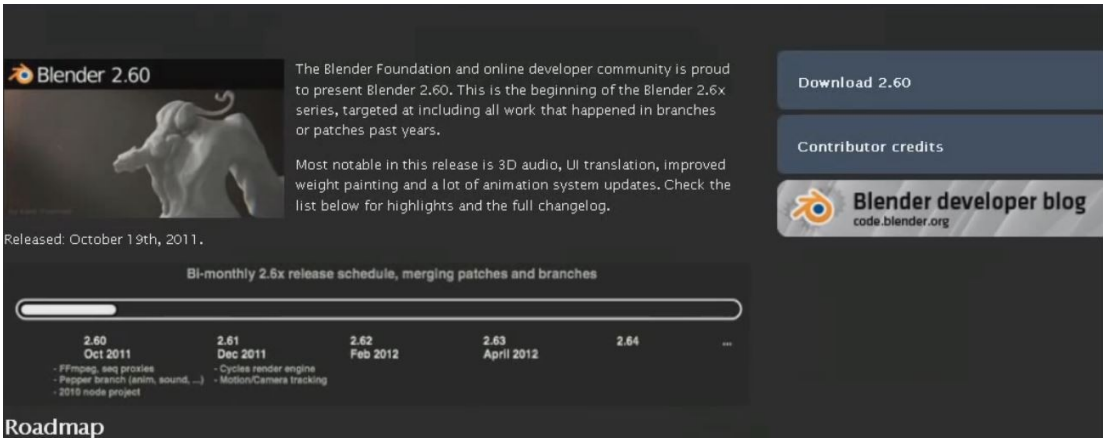
Sono stati introdotti dei cambiamenti nel sistema di animazione, soprattutto per quanto riguarda l'interfaccia utente e l'usabilità, modificando un po' il Graph Editor, il sistema dell'animazione non lineare, il Dopesheet, i Drivers e le curve di animazione IPO.

È stato introdotto un sistema di localizzazione dell'interfaccia, ossia la possibilità di avere Blender in altri linguaggi, inclusi quelli che prevedono ideogrammi o comunque caratteri non latini; la traduzione delle varie voci è comunque ancora in corso: finora è stata sviluppata l'architettura e la Blender Foundation cerca volontari per le traduzioni.

Le API Python non sono state sottoposte a riscrittura (mantenendo la compatibilità con la versione 2.59), ma ampliate, aggiungendo classi e metodi per catturare uno spettro più ampio di eventi.

Altre modifiche riguardano il Weight Painting e il Game Engine, il motore di gioco di Blender 3D; prossimamente pubblicherò tutorials ad hoc su questi argomenti (posso anticiparvi che il prossimo tutorial sarà sull'Add-On Screencast Keys, utile per chi, come me, realizza tutorials su Blender 3D).

Potete scaricare la nuova versione dalla pagina dei download del sito; come potete vedere, ci sono versioni per sistemi Win e Linux a 32 e 64 bit e per MacOSX su piattaforme PowerPC o Intel.



*Pagina di download del sito*

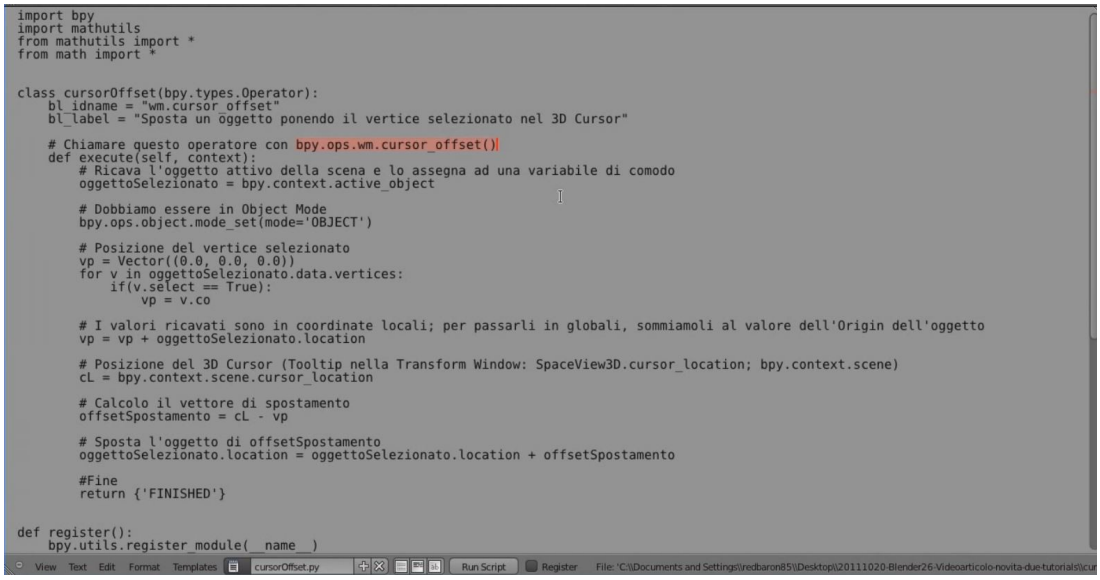
Adesso voglio tranquillizzare chi ha già acquistato i miei ebook (che trovate nella sezione ebooks del sito) realizzati sulla versione 2.59 di Blender 3D (ad esempio quello sul Nodes Compositing o quello sulla programmazione di script in linguaggio Python) e chi pensava di farlo a breve: i contenuti degli ebook sono tuttora validi, infatti in questo caso, come anticipato, non è stato effettuato un cambiamento radicale alla struttura del programma o alle API di programmazione, come è avvenuto ad esempio passando dalla versione 2.49 alla 2.5; più che altro, le funzioni sono state ampliate (cosa che vale soprattutto per le API Python che presentano classi e metodi per controlli più avanzati) mentre l'unica novità che noto nel Nodes Editor della post-produzione è il Nodo Frame: per il resto abbiamo i ben noti Glare, ID Mask, Alpha Over, Mix, eccetera, infatti (come indicato nel ChangeLog su Blender.org) le modifiche sono più che altro "interne", ossia implementative (per migliorare l'efficienza), per cui TUTTI gli argomenti trattati nell'ebook sul compositing e TUTTI i tutorials riguardanti la post-produzione pubblicati per Blender 2.5 sono ancora validi con Blender 2.6.

Lo stesso discorso può essere fatto per gli script presenti nell'ebook sulla programmazione degli Add-Ons in Blender 2.5: vediamo un esempio pratico, passando alla seconda parte di questo articolo.

*  *  *

Tempo fa ho pubblicato un tutorial nel quale descrivevo uno script che consentiva di spostare un oggetto posizionandone il vertice selezionato nel punto ove si trovava il 3D Cursor; lo script è stato

realizzato in riferimento alle API Python per Blender 2.59 e seguiva la pubblicazione del relativo ebook. Ricarichiamo in una finestra Text Editor di Blender 2.6 lo stesso script visto quella volta, quindi cambiamo una finestra in Python Console (infatti lo script andava eseguito in Text Editor per caricare il nuovo operatore nell'ambiente Python di Blender e richiamato dalla Python Console).

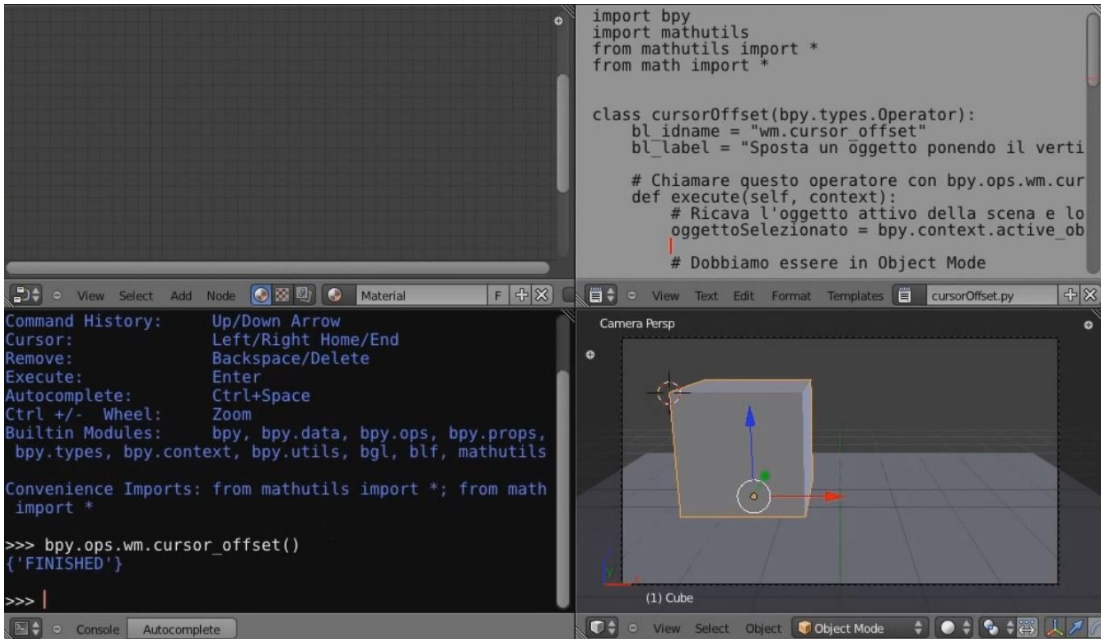


*Lo script nella finestra Python Console*

Per eseguire lo script, selezioniamo un vertice di un oggetto in Edit Mode nella scena 3D.

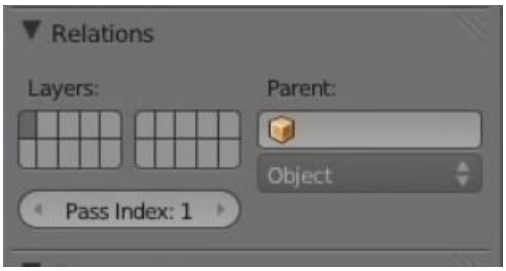
Posizioniamo da qualche parte il 3D Cursor, quindi premiamo su Run Script nella finestra Text Editor e digitiamo "`bpy.ops.wm.cursor_offset()`" nella finestra Python Console, premendo infine Invio per eseguire l'operazione: com'è possibile notare, lo script, realizzato in riferimento alla versione 2.59 delle API Python, continua a fare il suo lavoro anche con la versione 2.6 del programma.

Per mostrare la validità dei concetti esposti nell'ebook sul Compositing dei Nodes, facciamo un altro test: un effetto speciale di base, ossia l'aggiunta di un bagliore Glare ad un oggetto isolato mediante Pass Index.




*Lo script in esecuzione*

Selezioniamo il cubo nella scena e impostiamo il Pass Index di tale oggetto a 1 nella scheda Object, all'interno della Properties Window.

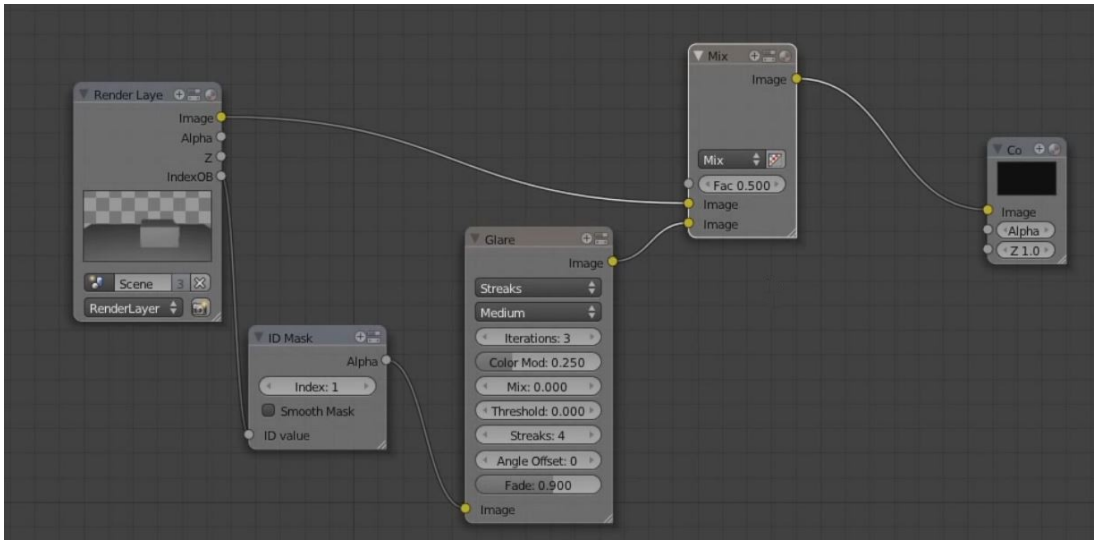



*Pass Index 1*

Nella scheda Render, sezione Layers, selezioniamo la casella Object Index, quindi cambiamo una finestra in Nodes Editor per configurare i nodi della post-produzione.

Nella finestra, scegliamo l'icona Compositing Nodes (la terza nell'header) e selezioniamo Use Nodes.

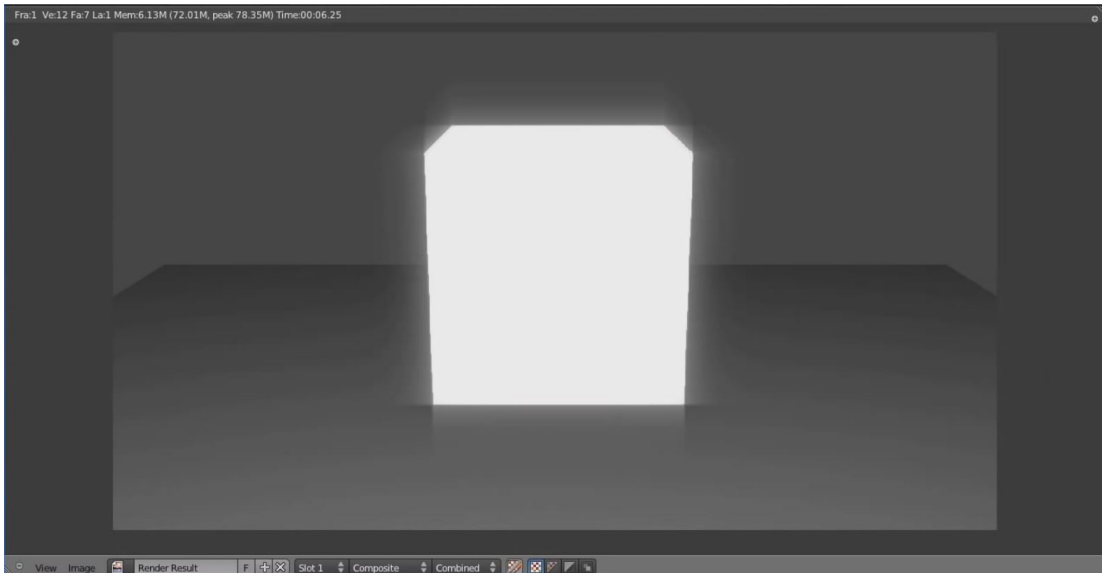
Tagliamo il collegamento tra Render Layers e Composite e aggiungiamo un nodo Convertor ID Mask, un nodo Color Mix e un nodo Filter Glare.




*ID Mask e Filter Glare*

Realizziamo questi collegamenti: da Index OB di Render Layers all'input di ID Mask, impostando qui Index 1 (lo stesso del cubo) e passando l'output al nodo Glare, abbassando qui la soglia Threshold a zero, quindi passiamo l'output Image di Render Layers al primo canale input Image del nodo Mix, passando sul secondo l'output in arrivo da Glare, e colleghiamo l'output di Mix all'input Image di Composite.

La modalità di default di Mix è... Mix, appunto, ma noi cambiamola in Screen lasciando Factor a 0.5 e avviamo un rendering veloce con F12, lasciando fare a Blender il primo passo di rendering e il successivo passo di Nodes Compositing, per osservare poi il risultato.




*Risultato finale*

Questa è una delle operazioni di base nel Nodes Editor di Blender 2.5 e, come potete vedere, è ancora validissima.

Neanche a dirlo, anche il videocorso di 80 puntate su Blender 2.5 (e, quindi, l'ebook pdf "*Blender 3D tutorials – Volume 1*", che ne raccoglie le trascrizioni) resta valido come primo approccio al programma: si trattava di un corso sulle basi di Blender 3D e, come avete notato, le basi sono rimaste quasi immutate: semmai sono stati aggiunti elementi, ma di queste novità parlerò a partire dal prossimo tutorial.

Da questo momento in poi, infatti, userò Blender 2.6 per i miei tutorials su tale programma.

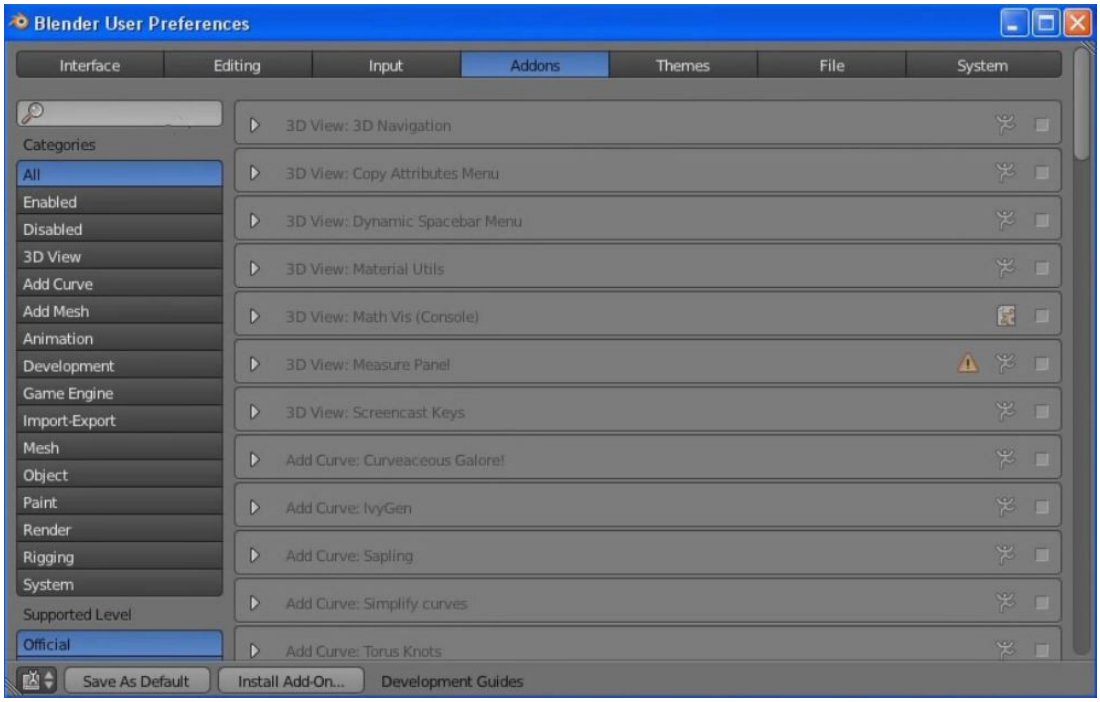
Secondo la timeline pubblicata su Blender.org e discussa all'inizio di questo articolo, l'integrazione definitiva del motore di rendering Cycles è prevista per Dicembre 2011, con la release 2.61 di Blender 3D, quindi fino ad allora NON pubblicherò tutorials su questo motore di rendering, iniziando proprio con la versione ufficiale.

*  *  *

# L'Add-On Screencast Keys

In questo tutorial parleremo dell'Add-On “Screencast Keys”, fornito di serie con la versione 2.6 del programma ma disabilitato di default; si tratta di un Add-On utile per chi, come me, realizza dei videotutorials, in quanto mostra a video, all'interno della finestra del programma, i tasti premuti.

Come anticipato, l'Add-On è fornito con la versione 2.6, quindi non dobbiamo scaricarlo e installarlo, ma semplicemente attivarlo; per far ciò, apriamo la scheda User Preferences dal menù File e spostiamoci nel pannello Add-Ons per cercare il plugin.



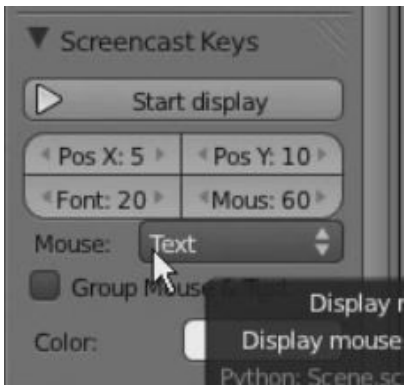
*User Preferences - Add-On*

Nella casella di ricerca veloce, scriviamo il nome dell'Add-On, “Screencast”, in modo da filtrare i risultati e trovare subito ciò che ci interessa.

Apriamo la scheda dell'Add-On: le informazioni di base ci dicono che i comandi relativi a questo script verranno visualizzati nella finestra Transform Properties di una 3D View. Attiviamo l'Add-On selezionando la checkbox sulla destra; a questo punto, se non pensate di dover usare l'Add-On al prossimo riavvio, chiudete semplicemente la scheda User Preferences; se, invece, volete trovare l'Add-On in funzione di default nelle prossime sessioni di Blender, cliccate su Save As Default, ma mi raccomando fatelo all'inizio di una sessione di lavoro perché questo comando salva la configurazione di tutto l'ambiente di lavoro.

*La scheda dell'Add-On*

A questo punto spostiamoci in una 3D View: effettuando una qualsiasi operazione non vedremo alcuna scritta a video, e questo perché l'Add-On è stato sì aggiunto all'ambiente, ma non attivato; in una 3D View, quindi, apriamo la finestra Transform Properties, con N, e scorriamo il pannello fino a trovare la sezione "Screencast Keys".

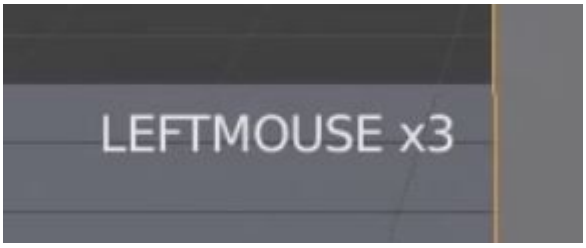


*Attivare Screencast Keys in una 3D View*

Intuitivamente, dobbiamo cliccare sul pulsante "Start Display" per attivare la visualizzazione del testo; gli altri strumenti presenti nella scheda ci consentono di personalizzare l'aspetto delle scritte informative cambiando posizione, colore e dimensione del testo o scegliendo di visualizzare un mouse quando viene premuto, appunto, un tasto del mouse: vediamo come.

L'origine delle coordinate XY è un punto in basso a sinistra della finestra, quindi aumentando X sposteremo il testo verso destra, nella 3D View, mentre aumentando Y lo sposteremo più in alto.

Premendo un tasto due o più volte, lo script non riscriverà il tasto ma aggiungerà un x2, x3, x4 eccetera alla scritta precedente.



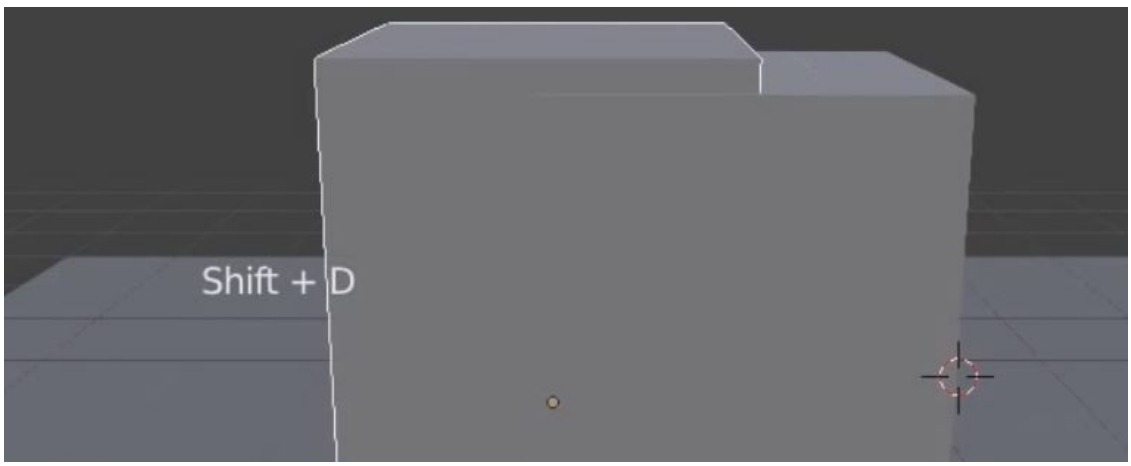
*x2, x3, x4, ...*

Di default, le informazioni riguardanti le azioni effettuate con il mouse vengono indicate mediante scritte ma è possibile disabilitare le informazioni sul mouse o visualizzare un'icona che lo rappresenta scegliendo, rispettivamente, None o Icon nel menù Mouse (l'opzione selezionata di default è, chiaramente, Text).

Con la ColorBox (che di default ha colore bianco) e il valore Font possiamo modificare colore e dimensioni delle scritte informative.

*Cambiare il Font*

Come avrete notato, se le azioni avvengono in poco tempo lo script mostra due o più scritte per volta, facendole scomparire a poco a poco; le combinazioni vengono mostrate con un +, per cui premendo SHIFT e D per una duplicazione vedremo appunto SHIFT + D.



*SHIFT + D*

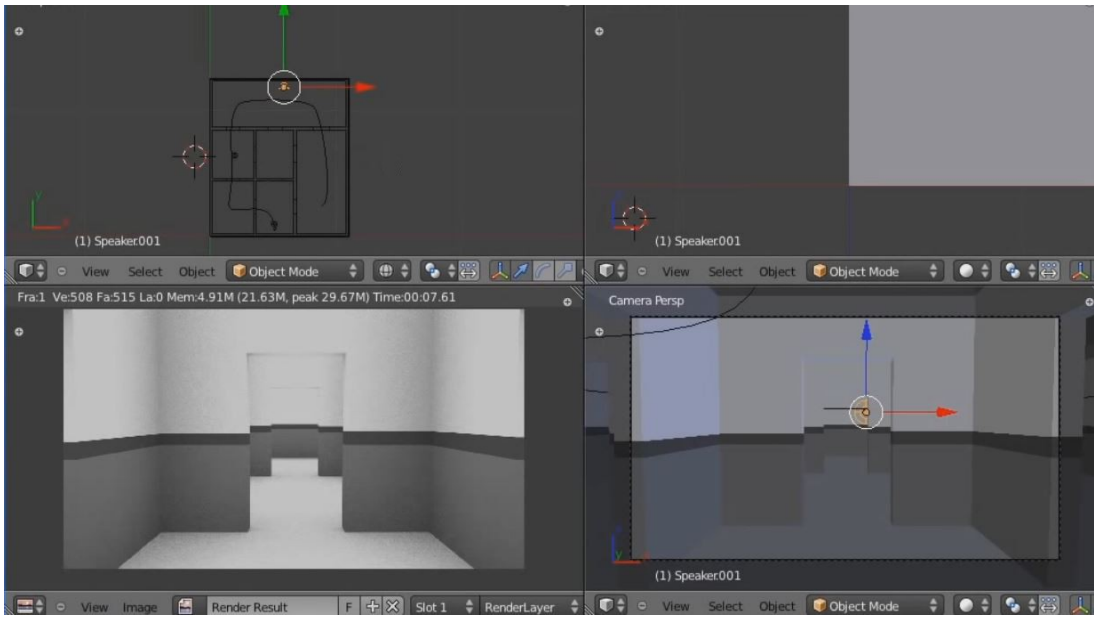
Per terminare la visualizzazione delle scritte informative, cliccare sul pulsante Stop Display.

*  *  *

# Altoparlanti nella scena 3D: gli oggetti Speaker e il Mixdown

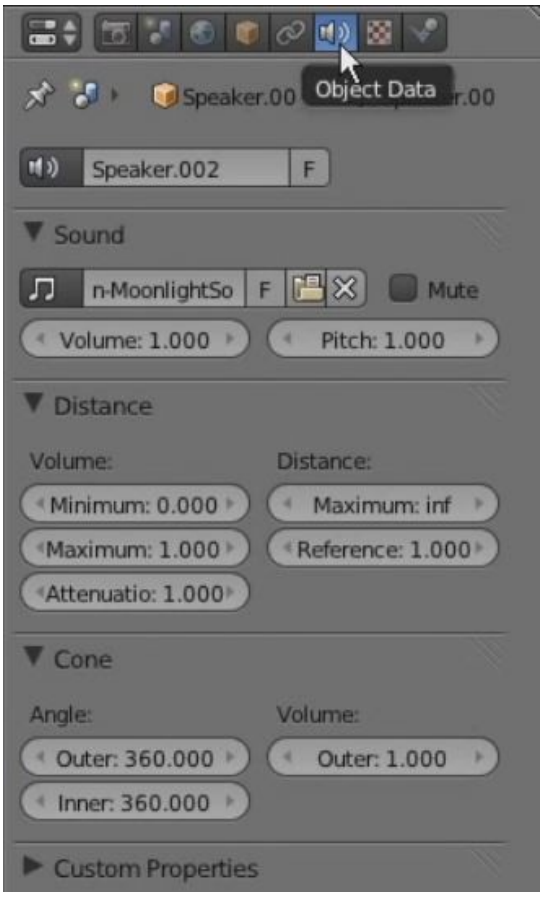
In questo tutorial parleremo di una delle novità più interessanti della versione 2.6 di Blender 3D: gli oggetti "Speakers" letteralmente "altoparlanti", fonti sonore da posizionare all'interno della scena 3D al pari delle fonti di illuminazione.
Il paragone non è casuale: in entrambi i casi si tratta di oggetti provvisti di icone ma tali simboli non vengono renderizzati, mentre i loro effetti sono evidenti, solo che nel caso delle luci si hanno effetti di ombreggiatura apprezzabili anche in rendering statici, mentre con gli Speaker si hanno delle tracce audio che possono essere "renderizzate", per così dire, anche a parte, con un'operazione nota come Mixdown, come vedremo.

Nella scena che sto utilizzando è presente un semplice scenario fatto di alcune stanze; la telecamera si muove all'interno di queste stanze seguendo un tracciato (una Bezier Curve) nell'arco di 480 frames. Ci sono tre oggetti Speakers in tre diverse stanze.



*Scena iniziale*

Si aggiunge un oggetto Speaker alla scena semplicemente con Add – Speaker.
L'icona rappresenta la posizione dell'emettitore ma, ovviamente, per impostare il file audio da riprodurre e le altre caratteristiche proprie di questo oggetto dobbiamo spostarci nel pannello Object Data, all'interno della Properties Window.



*Object Data per l'oggetto Speaker*

In generale, è possibile lasciare le impostazioni di default per i vari parametri qui presenti, ad esempio per il volume, i parametri sull'attenuazione e quelli sul cono di emissione; una piccola nota sul valore Minimum Volume: indica che una certa traccia audio dovrà avere un'intensità minima pari a quel valore, anche se la telecamera dovesse trovarsi a diversi km di distanza dall'emettitore.

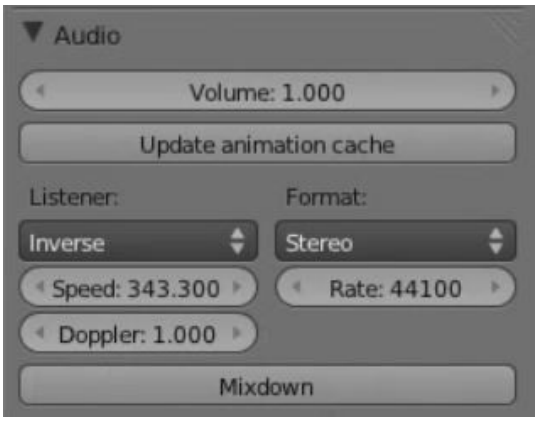
Con Minimum maggiore di 0, quindi, l'intensità di una traccia audio non scenderà mai sotto un certo valore, mentre con Maximum minore di 1 non potrà superare una certa soglia.

Un valore maggiore di 0 identico per Minimum e Maximum Volume può servire ad esempio a creare una musica o un rumore di sottofondo presente in tutta l'animazione con la stessa intensità.

Per associare il file audio da riprodurre all'oggetto Speaker corrente, cliccare sull'icona della cartella o su quella della nota, per caricare un file da disco o sceglierne uno tra quelli eventualmente presenti nella scena (un po' come avviene con le immagini per l'editor UV o per le Textures immagine).

*Scelta del file audio da riprodurre*

Nel mio caso, al primo altoparlante ho associato un brano di Beethoven, al secondo un brano di Mozart e al terzo uno di Bach; nel terzo altoparlante, inoltre, ho cambiato l'apertura del cono di emissione, che di default è 360 gradi (quindi omnidirezionale) in 30 gradi.



*Sezione audio nella scheda Scene*

Per quanto riguarda il posizionamento e la personalizzazione delle fonti sonore nella scena questo è tutto, ma non abbiamo ancora finito: spostiamoci infatti nella scheda Scene all'interno della Properties Window ed osserviamo la nuova sezione Audio.

In genere, qui non ci sarà bisogno di abbassare il volume o di cambiare il tipo di attenuazione, anzi riguardo al volume devo dirvi che Blender tende a tenerlo molto basso, quindi considerate la possibilità di alzare il volume dei file con un programma esterno oppure con il Video Sequence Editor di Blender, che consente anche di amplificare il volume delle tracce audio presenti nella scena.

Il parametro Speed indica la velocità del suono in metri al secondo nell'ambiente da riprodurre ed è impostato a 340, quella del suono nell'aria; serve per calcolare il ritardo del suono in scene molto ampie e per realizzare anche l'effetto Doppler, attivato di default, come vedremo (o meglio: sentiremo!) a breve.

Con il menù Format possiamo scegliere se generare suono Mono, Stereo o Surround (fino al 7.1, in questa versione di Blender) da salvare come traccia audio su disco.

Restano due pulsanti molto importanti: Update Animation Cache, che aggiorna le informazioni sull'audio prima di generare l'animazione e che andrebbe premuto dopo ogni modifica importante alla scena (ad esempio, la posizione di uno Speaker) o, se preferite, prima di avviare un rendering o un Mixdown, e il pulsante Mixdown, appunto, che consente di registrare l'audio "calcolato" senza dover renderizzare la scena, limitandosi alla sola traccia sonora.

Per “audio calcolato” intendo l'audio così come lo sentirebbe un osservatore in movimento al posto della telecamera. Osserviamo la scena: abbiamo la telecamera che parte da una stanza dove è presente il primo Speaker, quindi si passa in una seconda stanza priva di altoparlanti, mentre la terza e la quarta hanno un altoparlante con un diverso file audio ciascuna, infine la quarta sala senza altoparlanti... attenzione anche a destra e sinistra: nel movimento, il secondo Speaker è a destra della telecamera, mentre il terzo è sulla sinistra; infine, quando la telecamera è nell'ultima stanza tutte le fonti sonore si trovano sulla destra dell'osservatore.

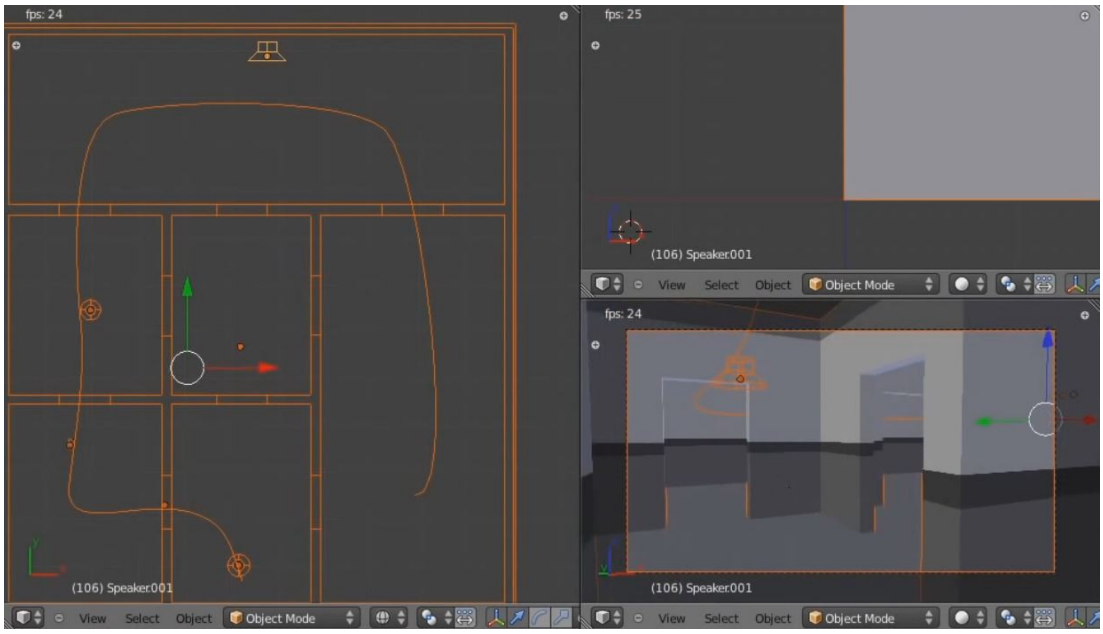
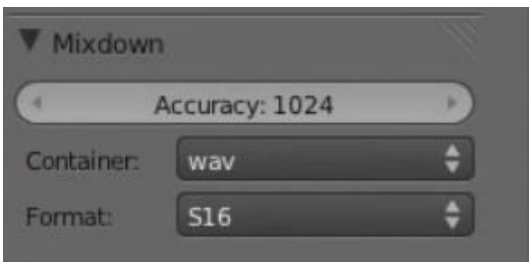



*Immagine del percorso della telecamera*

Clicchiamo quindi sul pulsante Mixdown: si aprirà una finestra stile “esplora risorse” dove potremo impostare il percorso del file audio da salvare su disco e il suo formato, da specificare a sinistra; nel mio caso, sto scegliendo Wave, con le impostazioni di default: ciò porterà alla creazione di un file di circa 4 MB, della durata di una ventina di secondi (la stessa dell'animazione della scena di Blender, ovviamente).

L'operazione di salvataggio richiederà un po' di tempo, a seconda dei calcoli da fare.




*Impostazioni del Mixdown (Wave, in questo caso) nella scheda del salvataggio file*

Al termine, apriamo il file Wave così generato con un qualsiasi programma ed ascoltiamolo (mettete le cuffie per notare la resa della "posizione" del suono, alla vostra destra o alla vostra sinistra, a seconda dei casi).

Nel mio caso si sente, verso il dodicesimo secondo, una distorsione del brano di Bach.
Tale "distorsione" è dovuta al fatto che in quel momento, al frame 300, la telecamera passa molto velocemente di fronte allo Speaker con quel file audio, che tra l'altro ha un cono di apertura di appena 30°, quindi l'effetto finale si ha per via del Doppler e della breve durata del passaggio nel cono.

A questo punto possiamo passare l'audio al responsabile del suono, che magari farà qualche effetto di equalizzazione o aggiungerà altri effetti, mentre noi possiamo continuare a personalizzare la scena 3D o passare al rendering, per poi inserire in un unico file con audio e video le due tracce, ad esempio mediante il Video Sequence Editor di Blender o un altro programma di Video Editing.

Caratteristica interessante degli Speakers è quella di poter essere imparentati agli oggetti, per cui potete imparentare il file audio di un respiro pesante o del rumore dei passi alla telecamera in movimento, il rumore di un motore ad una mesh che rappresenta un veicolo, e così via; come abbiamo visto, Blender "renderizza" l'audio tenendo conto di posizione e velocità sia dell'emettitore che della telecamera, consentendo poi di salvare su disco la traccia così calcolata.

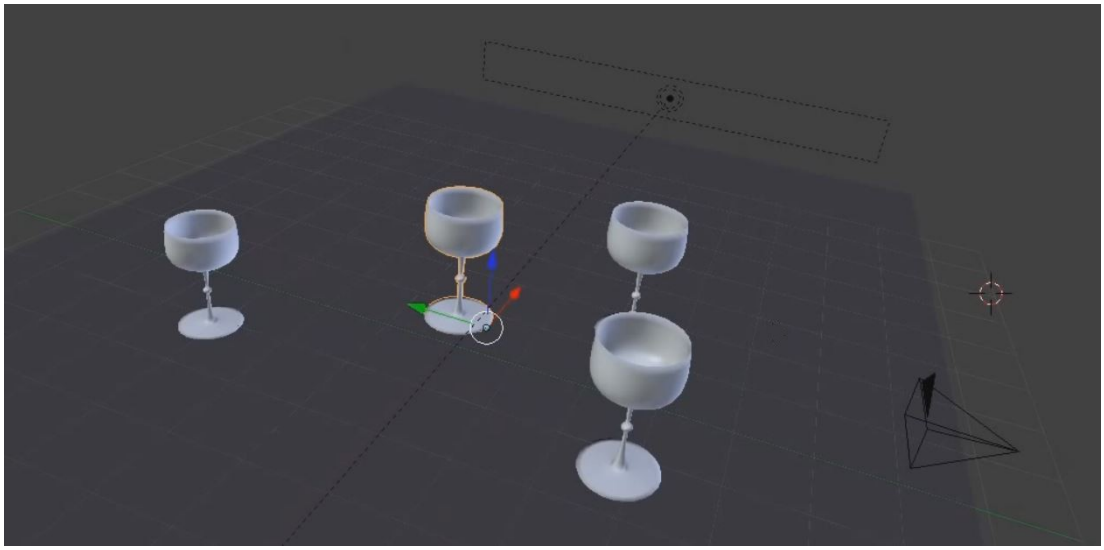
*  *  *

# Messa a fuoco selettiva con il nodo Defocus

In questo brevissimo tutorial vedremo come realizzare la messa a fuoco selettiva in Blender 3D 2.6, ossia come mettere a fuoco solo un oggetto della scena e sfocare tutto il resto utilizzando il Nodo Defocus in fase di Compositing, regolando anche l'intensità della sfocatura in base alla distanza degli altri oggetti da quello da tenere a fuoco, usato come punto di riferimento all'interno della scena.

Ho trattato l'argomento del defocus selettivo anche nel mio ebook dedicato al Compositing dei Nodes in Blender 3D 2.5, valido anche per la versione 2.6 del programma; in effetti, per la trattazione dei Nodi Glare e Blur utilizzati in questo tutorial per il bagliore sui riflessi speculari rimando proprio all'ebook.

Esaminiamo la scena: abbiamo diversi calici disposti su un Plane, con distanze variabili rispetto alla posizione della telecamera, e vogliamo mettere a fuoco solo uno di questi oggetti, sfocando tutto il resto.



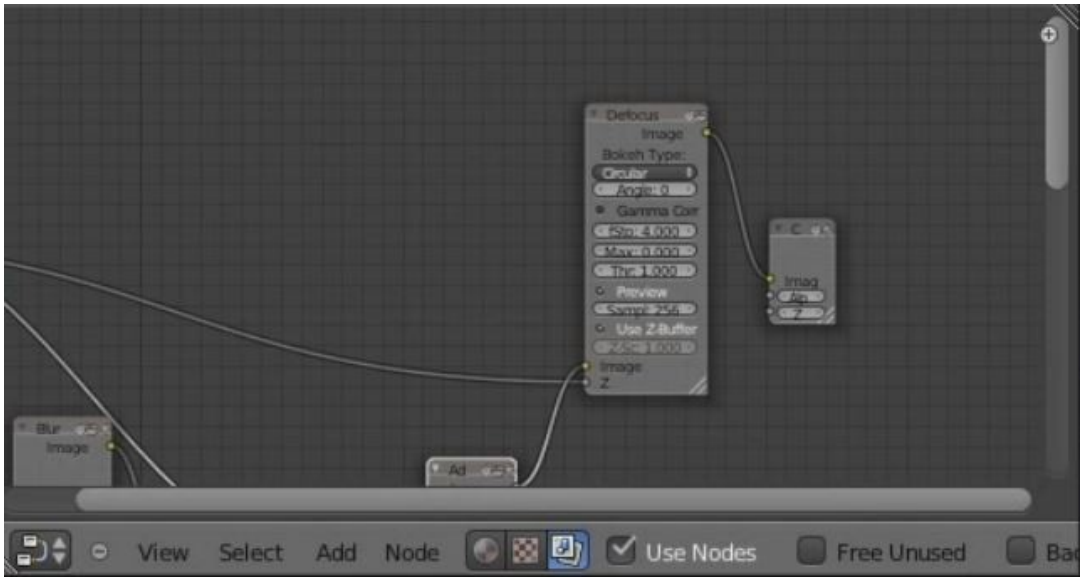
*La scena iniziale*

Le due operazioni da fare sono semplicissime e riguardano, in particolare, l'oggetto Telecamera e il Nodes Editor, per il Compositing dei Nodi in post-produzione.

*Impostare il nome dell'oggetto nel campo Depth of Field della telecamera*

Iniziamo dalla Telecamera: selezioniamo tale oggetto, apriamo la scheda Object Data nella Properties Window e digitiamo il nome dell'oggetto da mettere a fuoco (Mesh.002, nel mio caso) nel campo Depth of Field, che è un menù (quindi basta fare click col tasto sinistro del mouse sulla casella per scegliere comodamente l'oggetto da tenere a fuoco nel menù che apparirà a video).

Per quanto riguarda l'oggetto Telecamera questo è tutto: è sufficiente, quindi, specificare il nome dell'oggetto da mettere a fuoco in Depth of Field, o in alternativa una distanza dalla telecamera utilizzando il campo Distance, sempre sotto Depth of Field, la profondità di campo.
Vediamo quindi come implementare il defocusing mediante il Nodes Editor di Blender 3D 2.6; per prima cosa, quindi, apriamo una finestra Nodes Editor, selezioniamo la terza icona nell'header e selezioniamo la checkbox “Use Nodes”.



*Nodes Editor e Use Nodes (nell'header della finestra)*

Per realizzare il defocus ci serve solo un Nodo da porre tra l'immagine da trattare, generalmente in arrivo da Render Layers (ma non è detto) e Composite, comunque anche tra Defocus e Composite è possibile inserire altri nodi per realizzare altri effetti (Defocus non deve essere per forza il primo o l'ultimo nodo nella nostra pipeline di compositing).

*Il Nodo Defocus*

Nella configurazione da me utilizzata, in particolare, ci sono anche dei Nodi Mix Add, Blur e Glare, per aggiungere dei bagliori sia hard che soft glow ai soli riflessi speculari presenti nella scena; in questo tutorial, comunque, esamineremo solo il Nodo Defocus, i suoi collegamenti e il suo funzionamento.

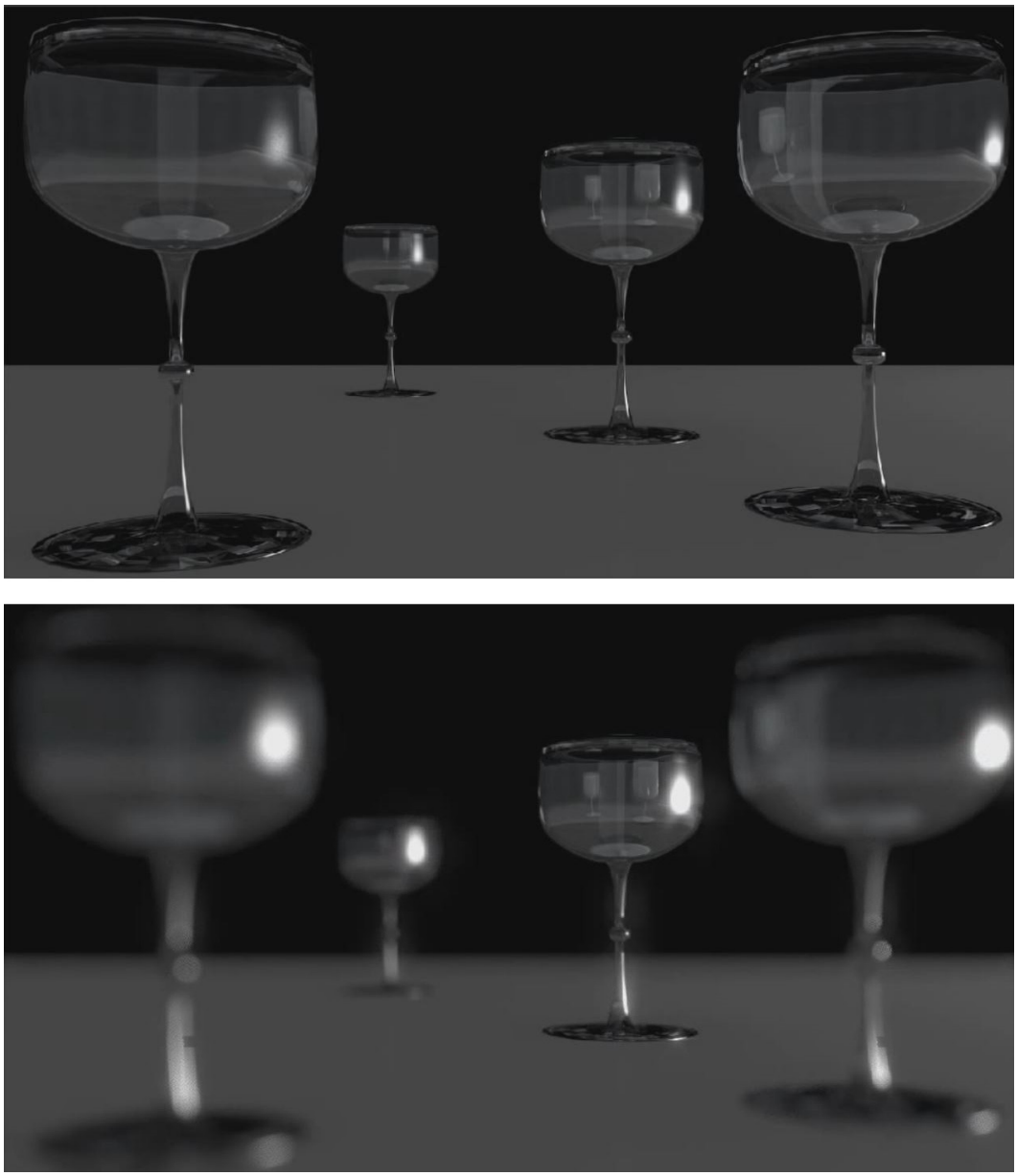
Notiamo che il nodo Defocus ha due porte in ingresso: Image (l'immagine da trattare) e Z (ossia informazioni sulla profondità dei pixel nella scena); in pratica, per ogni pixel dell'immagine in arrivo, il canale Z (se presente, ovviamente) ci dice qual è la distanza dalla telecamera del punto renderizzato in quel pixel. Questa informazione serve a Defocus per determinare la distanza degli altri punti da quello di messa a fuoco e, quindi, per calcolare l'entità della sfocatura da implementare.

L'intensità dell'effetto di sfocatura va regolato agendo sul parametro *fStop,* ed in particolare un valore basso (ad esempio 4) realizzerà una messa a fuoco MOLTO stretta sfocando notevolmente gli oggetti via via più distanti da quello utilizzato come riferimento, mentre valori più elevati metteranno a fuoco porzioni più ampie della scena.

Se avete dimestichezza con le macchine fotografiche, capirete subito come funzionano gli *fStop*, con un facile paragone con l'apertura del diaframma della macchina fotografica e la conseguente variazione della profondità di campo, solo che nel caso della macchina fotografica influenziamo anche la quantità di luce che entra nella scena e quindi variamo il tempo di esposizione richiesto, mentre con gli *fStop* del nodo Defocus influenziamo solo la profondità di campo.

La qualità dell'effetto va regolata impostando un valore più o meno elevato per Samples; a valori bassi, infatti, corrisponderanno tempi di Compositing minori ma risultati molto rumorosi, mentre a valori elevati corrisponderanno tempi elevati ma risultati di qualità.
Ricordarsi, infine, di selezionare la casella USE Z-BUFFER nel Nodo Defocus.

Nell'immagine seguente, la differenza tra un rendering effettuato senza Compositing e lo stesso rendering con i bagliori speculari accentuati e, ovviamente, l'effetto di messa a fuoco selettiva.



*Confronto tra rendering senza (sopra) e con (sotto) messa a fuoco selettiva*

*  *  *

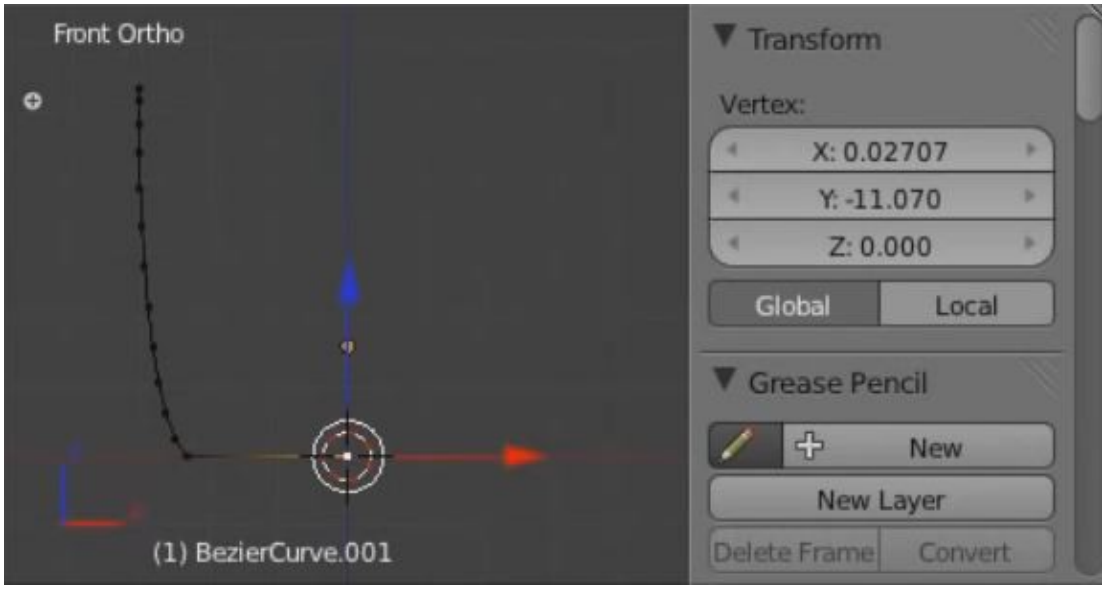
# Il bicchiere mezzo pieno: liquidi statici in contenitori

In questo brevissimo tutorial vedremo un modo (in effetti, un “trucchetto”) per inserire un liquido statico all'interno di un contenitore in Blender 3D 2.6, per realizzare ad esempio bicchieri, bottiglie e altri contenitori contenenti acqua, latte o altri elementi (nel rendering visibile nell'immagine seguente, un bicchiere di latte).

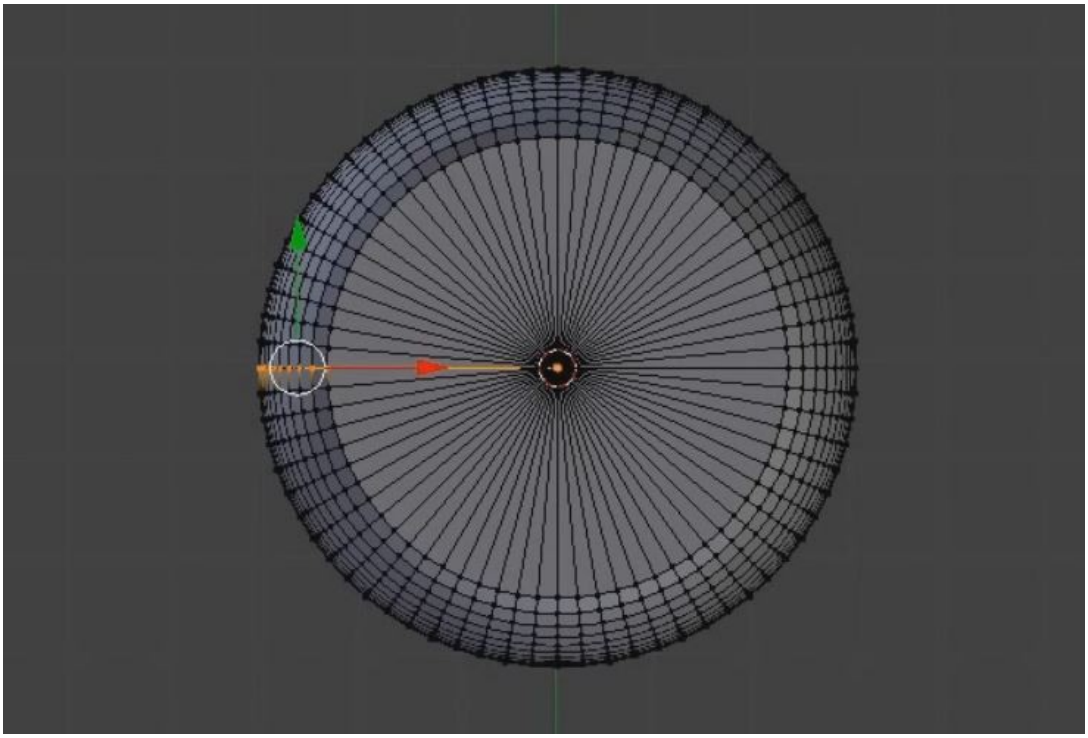
*Il rendering del bicchiere di latte*

Realizziamo un bicchiere partendo da una semplice curva Bezier deformata in modo tale da definire il profilo dell'oggetto. Non chiudete la curva e non datele uno spessore: effettueremo questa operazione in seguito, utilizzando Solidify.

Per creare il bicchiere dal profilo, convertiamo la Curve in Mesh con ALT-C e “Mesh from Curve”, quindi selezioniamo il vertice che starà al centro del bicchiere, spostiamo lì il 3D Cursor con SHIFT S e “Cursor to Selected”, andiamo nella finestra TOP, selezioniamo tutti i vertici in Edit Mode e applichiamo SPIN, con 360 gradi e, ad esempio, 72 passi o più.



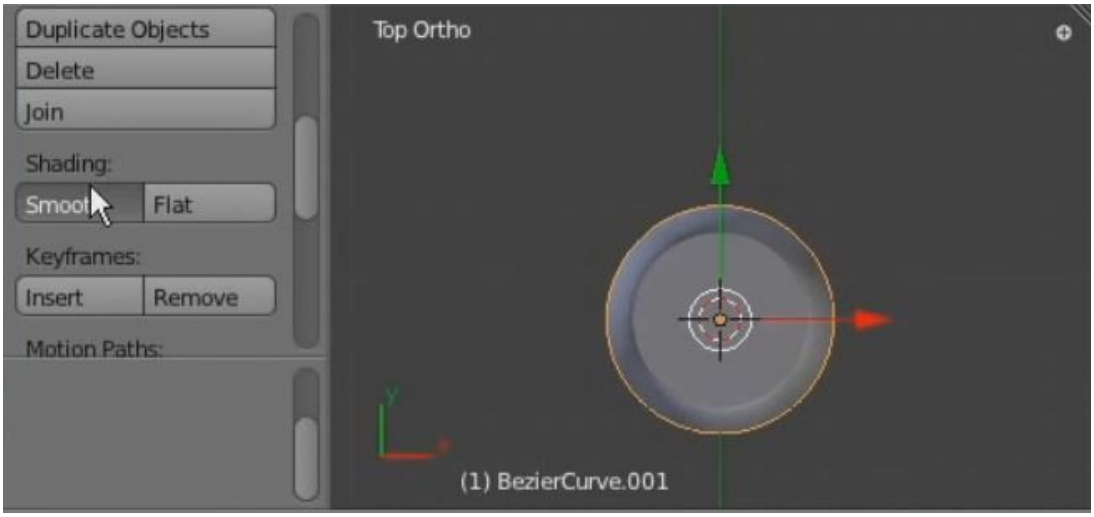

*La definizione del profilo del bicchiere con una curva Bezier (qui trasformata in mesh con ALT C)*



*SPIN applicato al profilo da una vista TOP ORTHO*

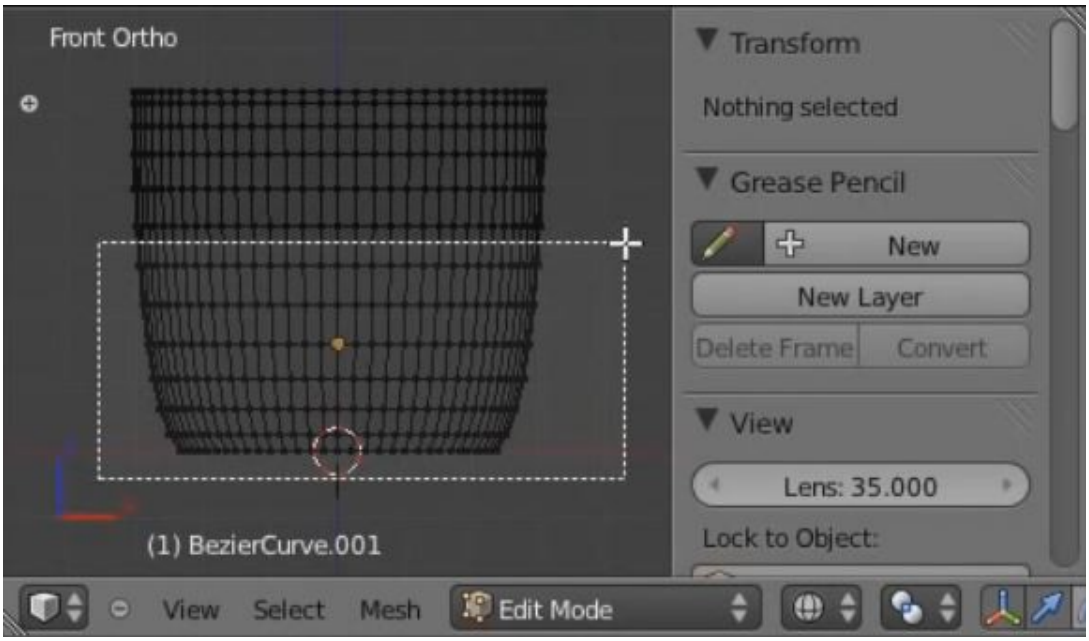
Selezioniamo tutti i vertici e rimuoviamo i doppioni con un click su Remove Doubles; il valore di default per la soglia (threshold) dovrebbe andare bene.

Smussiamo l'oggetto cliccando su Set Smooth, nella Tool Shelf.



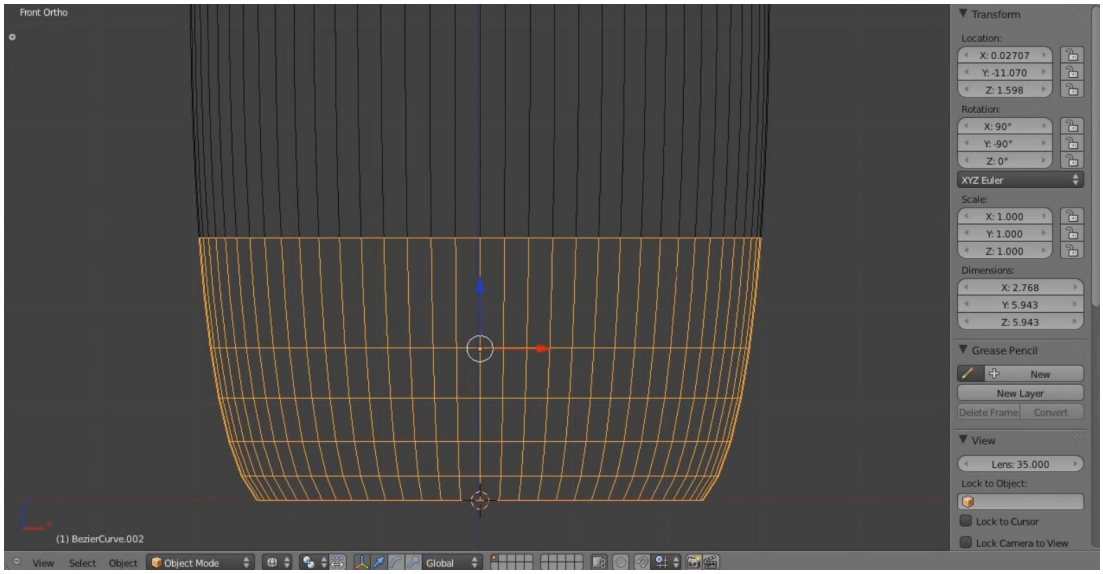

*Applicare Set Smooth*

Adesso, in Edit Mode, selezioniamo più o meno metà bicchiere, partendo dal basso, e duplichiamo la selezione con SHIFT D, senza spostarla, quindi confermiamo subito l'operazione con un click del tasto sinistro del mouse o premendo Invio.



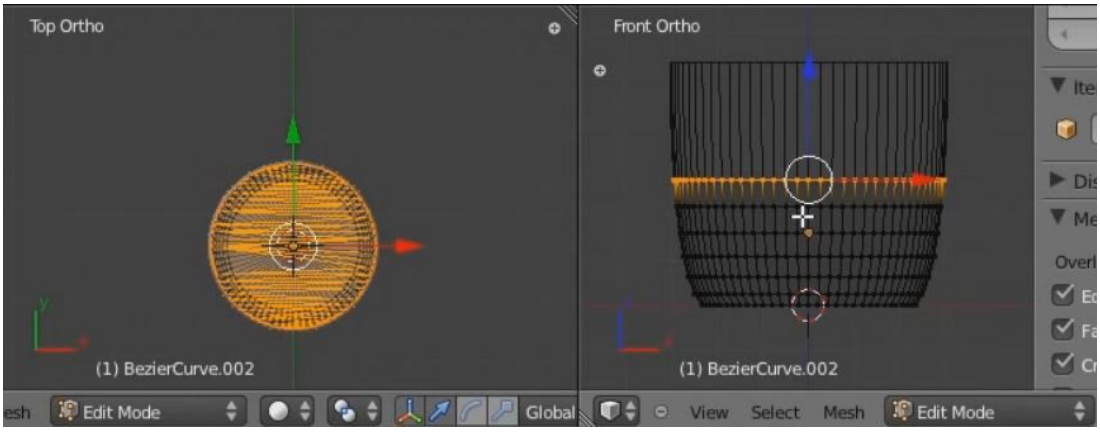

*Selezionare la porzione di bicchiere da duplicare*

Separiamo questa nuova selezione dall'oggetto originale premendo P e scegliendo Selection dal menù che apparirà a video; come potete vedere, abbiamo una nuova mesh, che rappresenta il volume occupato da un liquido nel bicchiere.



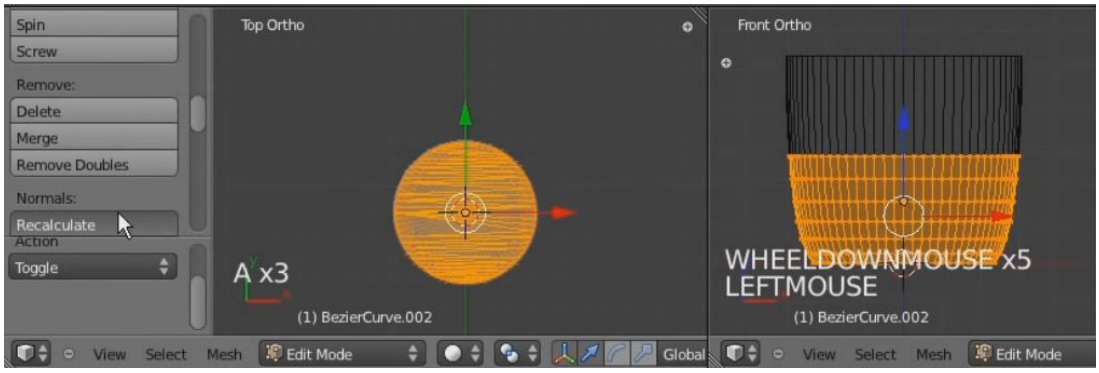

*Risultato della separazione con P - Separate*

Per chiudere la parte superiore della nuova mesh, in modo da realizzare la superficie del liquido, selezioniamo i vertici dell'anello superiore (possiamo effettuare questa operazione con estrema facilità nella vista FRONT in modalità di visualizzazione Wireframe) e premiamo ALT F.




*Chiusura della mesh*

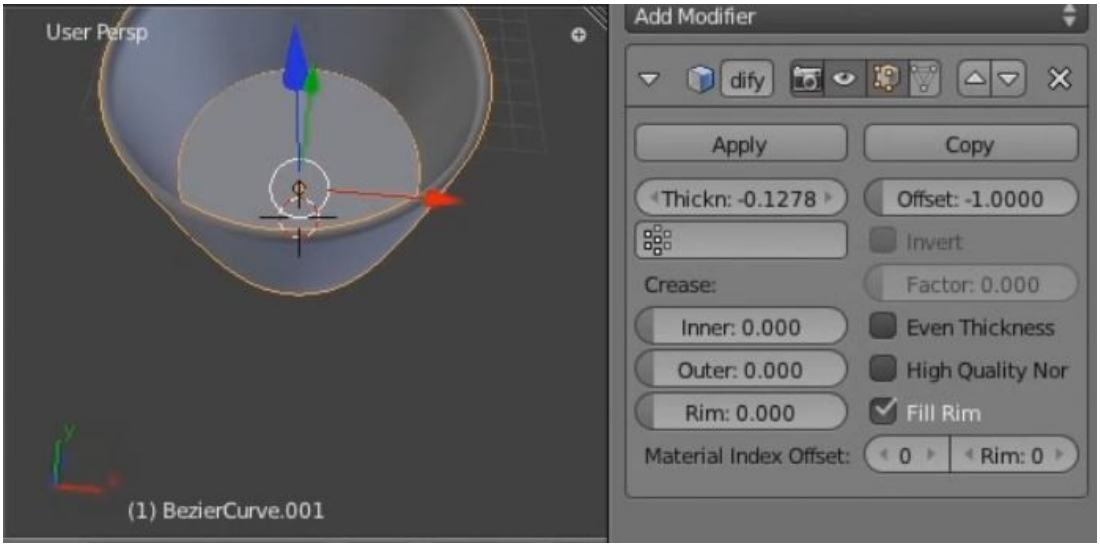
A questo punto ricalcoliamo le normali della nuova mesh selezionando tutti i vertici dell'oggetto e cliccando su Recalculate Normals, nella Tool Shelf.




*Recalculate Normals*

Scaliamo leggermente questa mesh, premendo ad esempio S 0.99, per evitare il problema delle facce sovrapposte in fase di rendering.

Per dare uno spessore al bicchiere, selezioniamolo e aggiungiamo un modificatore Solidify, impostando un valore negativo per Thickness, lo spessore, in modo da aggiungere spessore verso l'esterno dell'oggetto e non sovrapporre le facce al liquido. Applichiamo il modificatore.




*Utilizzo di Solidify*

Ricalcoliamo le normali anche per il bicchiere, ora dotato di spessore, passando in Edit Mode, selezionando tutti i vertici e cliccando su Recalculate, nella Tool Shelf. Torniamo in Object Mode.

Mentre ci siamo, parliamo dei Materials:

- per il vetro del bicchiere, Material Surface di default ma con Trasparenza Raytrace attiva, Alpha 0, IOR (indice di rifrazione) 1.5, Depth 4 o più; riflessi Mirror attivati con Reflectivity 0.1, Fresnel 5, Depth 4 o più;
- per il latte, Oren-Nayar bianco con intensità 1, Specular Blinn con intensità 1, IOR 6, Hardness 200, Sub Surface Scattering attivo col preset Whole Milk (il latte, appunto) e un valore appropriato per Scale ("appropriato" significa che il valore di Scale dipenderà dalle dimensioni dell'oggetto nella scena: impostando Scale 1, ad esempio, una 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).

Una nota importante: i Materials degli oggetti che dovranno ricevere le ombre proiettate da oggetti semitrasparenti dovranno avere la voce "Receive Transparent" selezionata nella sezione Shadows; in questo caso, il Material del Plane e quello del latte dovranno avere tale opzione selezionata.

*Receive Transparent*

Il rendering visibile nell'immagine seguente è stato ottenuto con un Defocus selettivo sul bicchiere in fase di Compositing.

*Rendering con Defocus Selettivo*

Ecco invece (qui di seguito) il rendering senza Defocus; ho parlato del Defocus selettivo in un tutorial precedente dal titolo “Messa a fuoco selettiva con il Nodo Defocus in Blender 3D 2.6”.

*Rendering senza Defocus selettivo*

*  *  *

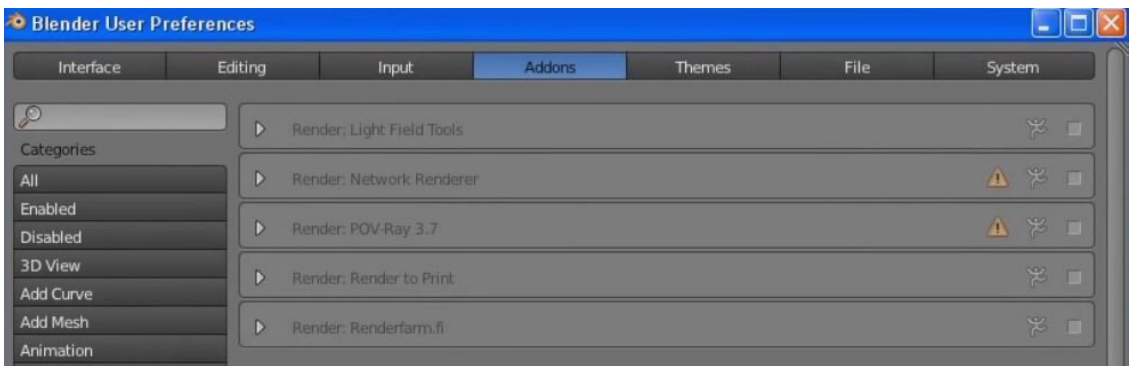
# Risoluzione e dimensioni per la stampa dei rendering

Questo brevissimo tutorial nasce in risposta ad una domanda di un utente del Forum, *BlueR3D,* che chiedeva come calcolare la risoluzione in pixel di un rendering per ottenere un'immagine in formato A4 con una risoluzione data.

Si tratta di un problema noto a chi, prima o poi, ha dovuto realizzare un'immagine per il video o la stampa, utilizzando tipicamente le risoluzioni 72 e 300 dpi (72 per il monitor, 300 per l'editoria), entro un formato di dimensioni prefissate, ad esempio A4 o A5. La buona notizia è che per Blender 2.58 e le versioni successive (quindi anche Blender 3D 2.6, utilizzato per realizzare questo tutorial) esiste uno script, realizzato da Marco Crippa, chiamato "*Render to Print*".

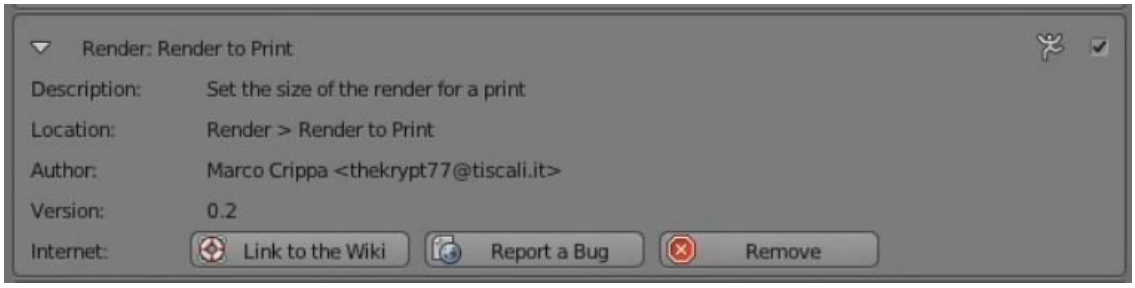
La prima cosa da fare è, ovviamente, scaricare il file Python dello script per installarlo in Blender 3D; potete scaricare il file, ad esempio, da *www.redbaron85.com/rp.zip* , dopodiché dovrete ovviamente scompattare il file zip e posizionare il file Python in una cartella a piacere.

Per installare l'Add-On, apriamo la finestra User Preferences (dal menù File) e selezioniamo il pannello Add-Ons, quindi clicchiamo sul pulsante Install Add-On e, dal menù stile "esplora risorse" che apparirà a video, scegliamo il file che avevamo scompattato e salvato precedentemente su disco.



*Installazione dell'Add-On in User Preferences*

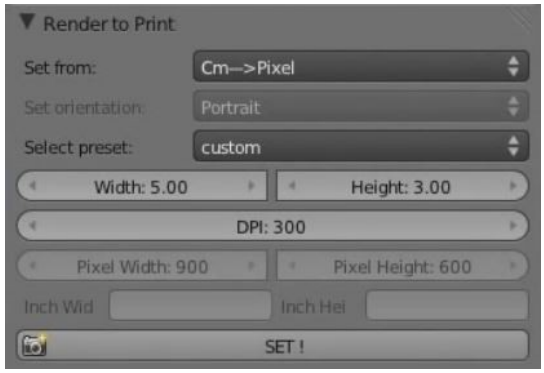
Nella scheda Add-Ons di User Preferences apparirà quindi la voce "Render to Print"; se allarghiamo la maschera, notiamo che l'Add-On verrà visualizzato nella sezione "Render", all'interno della Properties Window. Selezioniamo la Checkbox per attivare lo script.




*La scheda dell'Add-On in User Preferences*

Se vogliamo ritrovare l'Add-On anche ai prossimi riavvii di Blender, clicchiamo su Save As Default, in basso nella finestra User Preferences, altrimenti chiudiamo la finestra con la X in alto a destra; attenzione, perché scegliendo Save As Default salveremo come preimpostazioni TUTTE le impostazioni della scena corrente.

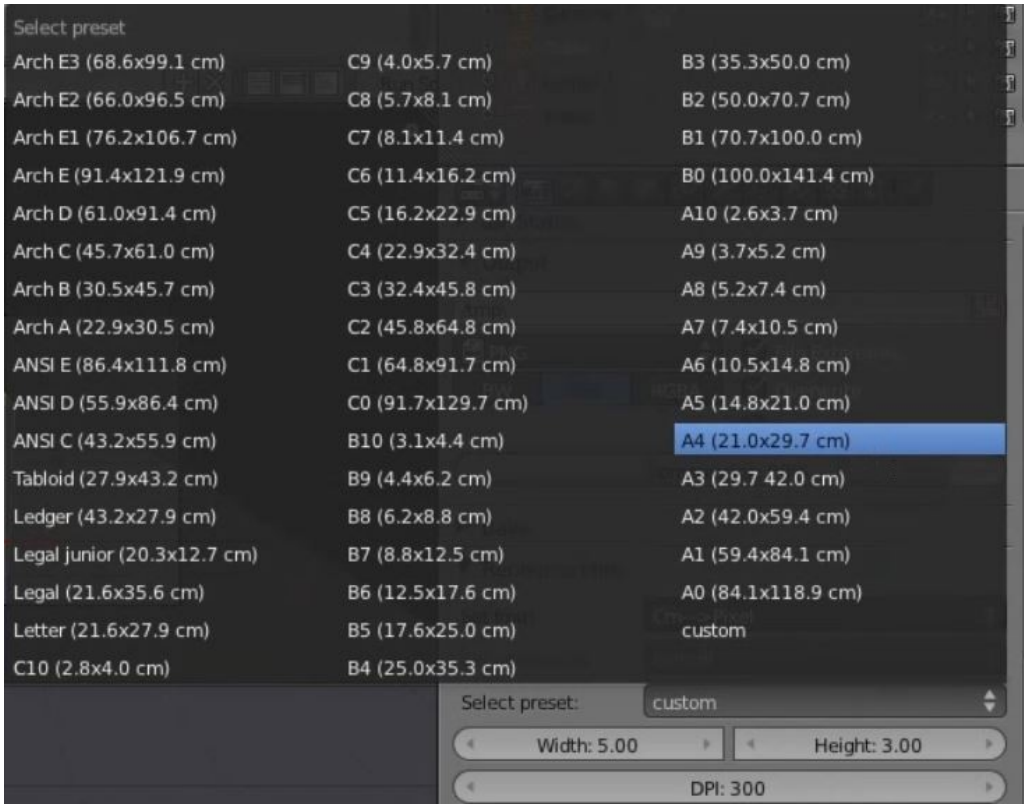
Esaminiamo quindi la nuova sezione apparsa in basso nella scheda Render, all'interno della Properties Window: intuitivamente, con il selettore Set From possiamo scegliere se impostare i centimetri (ad esempio se conosciamo le dimensioni e la risoluzione che l'immagine dovrà avere in stampa) e lasciar calcolare allo script i Pixel, oppure se impostare appunto i Pixel e vedere, con la risoluzione scelta, quali dimensioni avrà l'immagine finale, in centimetri.




*La sezione "Render to Print" nella scheda Render*

La risoluzione va impostata nel campo DPI, ovvero "*dots per inch*", punti per pollice; risoluzioni tipiche sono, ad esempio, 72 dpi per il video (cioè sul monitor del pc, ad esempio per immagini per siti web) e 300 dpi per la stampa e l'editoria (ad esempio per volantini e pubblicità in generale).

Sono disponibili inoltre molti *presets* per le dimensioni di stampa più comuni, accessibili mediante il menù Select Preset, per cui ad esempio se vogliamo renderizzare un'immagine destinata alla stampa in formato A4 con risoluzione 300 DPI possiamo scegliere appunto il *preset A4, 300 dpi* e specificare, con il menù Set Orientation, se utilizzare l'orientamento verticale (*Portrait*, ritratto) o orizzontale (*Landscape*, paesaggio).



*I Presets*

A questo punto le dimensioni della finestra di rendering non sono state ancora cambiate, infatti se osserviamo X e Y in alto nella scheda Render notiamo ancora i valori che avevamo all'avvio del programma; per sostituire questi valori con quelli calcolati dallo script, clicchiamo sul pulsante "*Set!*", cioè "Imposta", nella sezione dello script "*Render to Print*".

*"SET!": imposta*

*  *  *

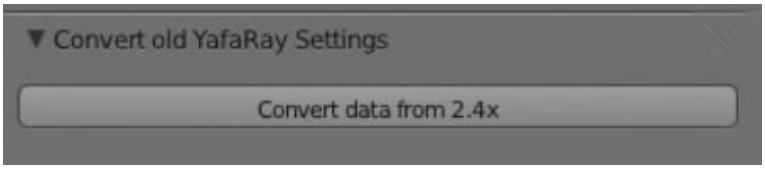
# Da Yafaray a Blender a Cycles... per caso

Questo brevissimo tutorial nasce per caso, nel senso che non era previsto: Nicolò Frontini ("*Nik8*" nel Forum del sito), con il quale collaboro occasionalmente per realizzare qualche tutorial, mi ha inviato una scena da utilizzare, appunto, per un tutorial; la scena è stata però realizzata in Blender 2.49 e renderizzata con Yafaray 0.1.1.

Ho caricato la scena con un doppio click sul file *blend,* non sapendo di doverla aprire con Blender 2.49, avviando così la versione di Blender attualmente installata sul mio sistema, ossia una build realizzata da Gabriele Ferreri e disponibile gratuitamente su *graphicall,* in attesa di un Add-On Yafaray 0.1.2 stabile per Blender 2.6 e solo allora, notando l'area vuota a sinistra, ho capito che la scena era stata realizzata con Blender 2.49, in quanto in genere quella era l'area destinata alla scheda dell'exporter Yafaray 0.1.1...

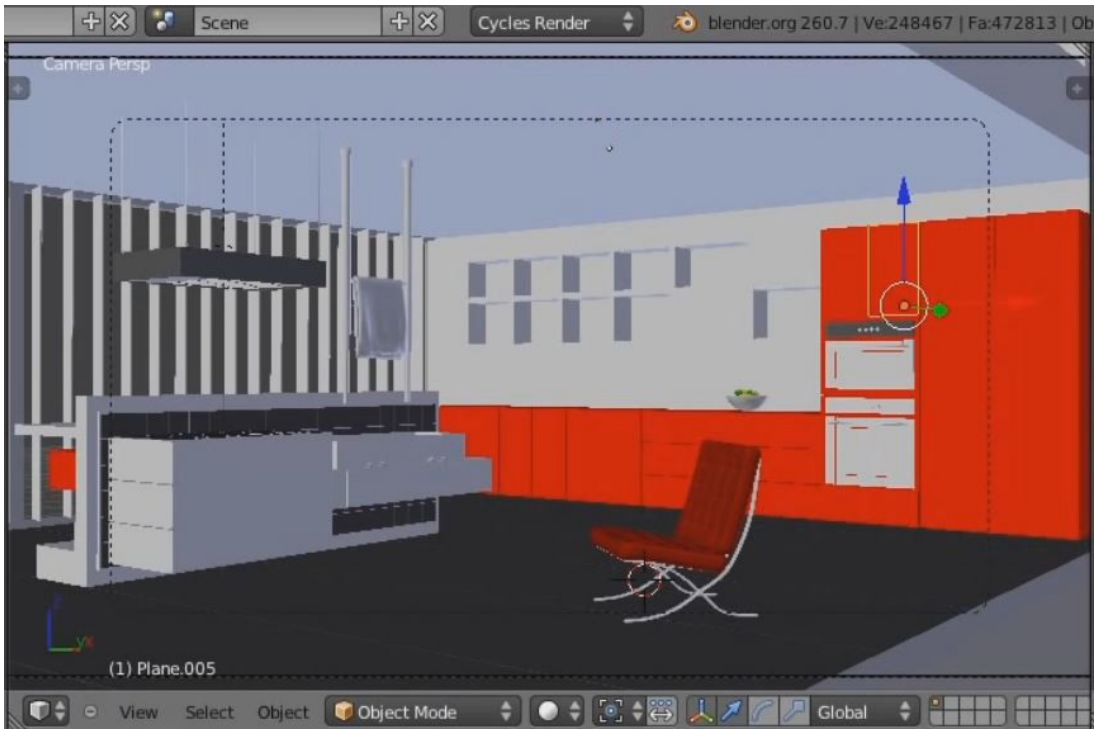
*La scena caricata in Blender 3D 2.6*

A questo punto però, anziché chiudere e riaprire la versione precedente di Blender, ho attivato lo script beta di Yafaray 0.1.2 (cosa che si può fare dalla scheda User Preferences, sezione Add Ons – Render, se si ha questa particolare build), quindi ho cambiato il motore di rendering in Yafaray e ho cliccato sull'icona "*Convert Data from 2.4*" nella scheda Render, all'interno della Properties Windows, in modo da aggiornare i Materials definiti nella versione 2.4 di Blender e 0.1.1 di Yafaray alle impostazioni e alle caratteristiche dei materiali in Blender 2.6 e Yafaray 0.1.2.




*Convert Data from 2.4*

Non ho renderizzato, però, la scena: notando, nel menù Render Engine, la voce "Cycles", ho cambiato ancora una volta motore di rendering, scegliendo appunto Cycles... ed ecco il risultato a video.




*Il risultato (nella 3D View, in Object Mode) dopo aver scelto Cycles*

Attenzione, voglio frenare gli entusiasmi: in questo modo abbiamo solo l'associazione dei vari Materials agli oggetti così come erano stati definiti nel file originale, mentre NON si ha la mappatura delle impostazioni dei singoli Materials che sono, ovviamente, diversi, infatti selezionando ad esempio uno sportello ed esaminando la scheda Materials, nella Properties Window, abbiamo che il Material è condiviso dagli altri sportelli (quindi non dobbiamo riassociare i Materials condivisi ai vari oggetti), il colore diffuso è quello corretto (infatti vediamo la scena "colorata", per così dire, nella 3D View) ma i Nodes dei Materials Cycles sono disattivati, quindi mancano tutte le impostazioni vere e proprie e, di conseguenza, gli effetti... comunque, è pur sempre un punto di partenza che ci consente di riciclare le vecchie scene realizzate con Yafaray 0.1.1, anche se poi dovremo "sistemare" i Materials in Cycles.

*Il rendering al volo senza aver sistemato i Materials in Cycles*

Direte voi: ma non bastava aprire il vecchio file, realizzato per Blender 2.49 e Yafaray 0.1.1, con Blender 2.6 e passare subito a Cycles? No, e il perché l'abbiamo detto all'inizio: per ricaricare i Materials Yafaray in Blender 2.6 è necessario cliccare sul pulsante "*Convert Data from 2.4*", POI da questo punto (una volta cioè riassegnati i Materials originali ai vari oggetti) possiamo passare a Cycles, saltando la parte iniziale, ossia il dover selezionare i vari elementi e creare i Materials da zero.

Ho avviato un rendering subito, senza attivare i Nodes dei Materials in Cycles e personalizzare i materiali; il risultato ottenuto è visibile nell'immagine visibile qui di seguito: non male, come "passo 0" della conversione delle vecchie scene.

*Il rendering ottenuto*

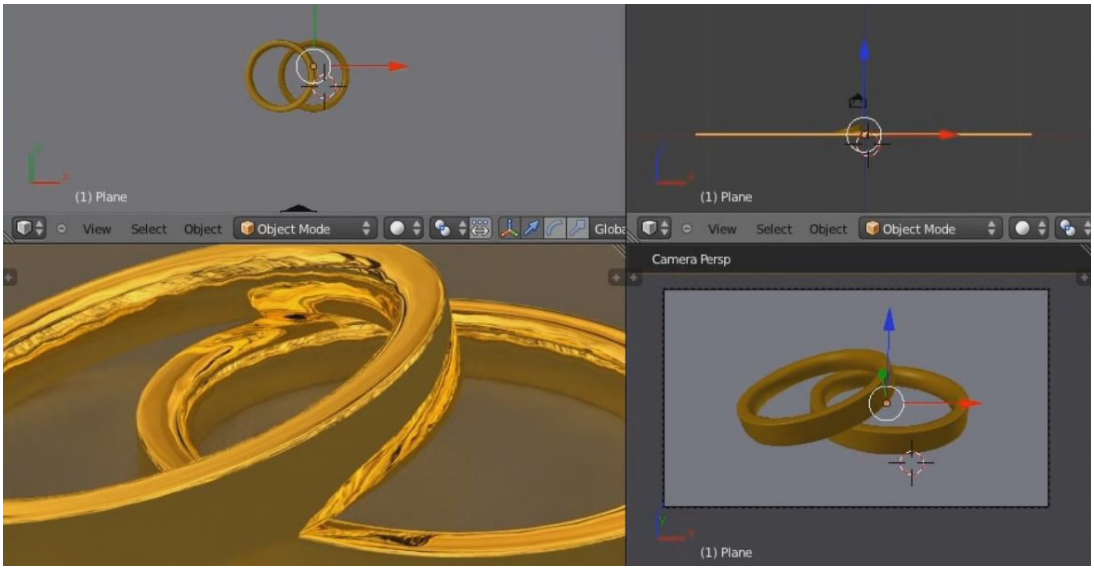
*  *  *

# Immagine “illuminante” su World in Blender e Yafaray 0.1.2

In questo brevissimo tutorial vedremo come applicare un'immagine come fonte di illuminazione mappata su World in una scena da renderizzare in Yafaray 0.1.2 su Blender 2.6; è una tecnica utile per fornire un aspetto non omogeneo ad oggetti con Materials riflettenti, come l'oro o altri metalli, quando tali oggetti non sono circondati da un set e quindi rifletterebbero solo il colore di World.

Ho trattato questo argomento anche per Blender Render nel tutorial “*Immagini HDR su Environment Lighting per oggetti riflettenti*” e, come vedremo, la tecnica in Yafaray 0.1.2 non è molto diversa; è una di quelle cose che, una volta note, risultano quasi ovvie, ma prima bisogna provarle almeno una volta!

Per questo tutorial sto utilizzando lo script Yafaray Exporter 0.1.2 in versione BETA per Blender 2.6, realizzato da Gabriele Ferreri e disponibile gratuitamente su *graphicall* come build *Blender + Yafaray*. Nella scena che sto utilizzando sono presenti due mesh modellate a forma di anelli e poggiate su un piano; faccio notare che NON sono presenti fonti di luce, di alcun tipo.




*La scena iniziale*

Per la cronaca, il Material degli anelli è un Material di base, con Shader Lambert e Mirror Strength 1.0, il massimo; i colori scelti sono discutibili, ok, ma questo è solo un esempio!
Il Plane sottostante ha invece un Material di default, con colore grigio chiaro.

Passiamo al motore di rendering Yafaray 0.1.2 mediante il selettore Engine nella finestra Info.

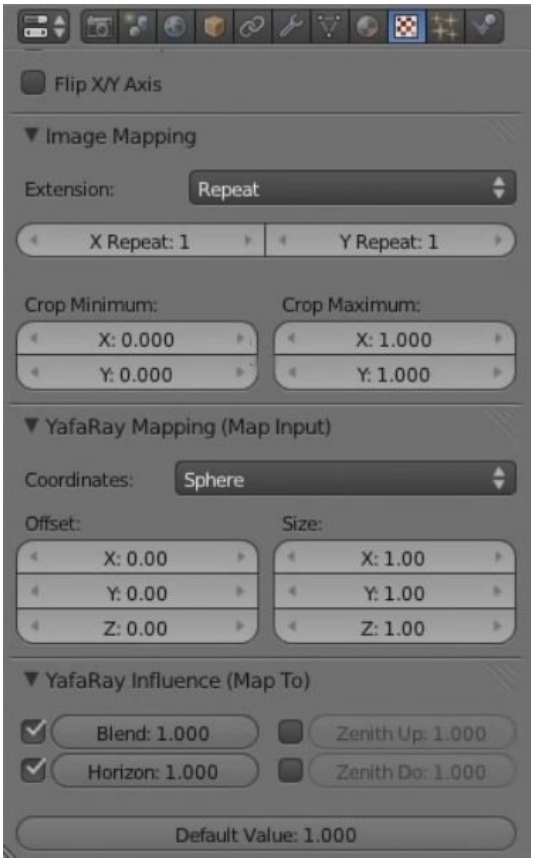
*Passaggio al motore di rendering Yafaray*

Per associare una Texture a World (l'universo virtuale della scena) dobbiamo selezionare prima World, nella Properties Window, quindi il pannello Texture e caricare, qui, una Texture di tipo immagine (possibilmente un'immagine HDR, tipicamente in formato EXR, ma va bene anche un file jpeg, come in questo caso).

*Texture immagine per World*

A questo punto, sempre in Texture, dobbiamo selezionare le caselle *Horizon* e *Blend* in Influence e scegliere la modalità di mappatura dell'immagine: in genere si fa uso di AngMap, ma potete fare una prova anche con Sphere, la mappatura sferica; a breve mostrerò due rendering della scena ottenuti con tali modalità.

Passiamo al pannello World per associare la Texture all'universo virtuale e, soprattutto, attivarla come "illuminante": dobbiamo selezionare, ovviamente, "Texture" in Background e impostare la Texture definita al passo precedente, quindi dobbiamo selezionare la casella IBL, il cui ToolTip ci dice appunto "*Use the background as the light source for your image*", ossia "Utilizza lo sfondo come fonte di luce per l'immagine", quindi aumentiamo il numero di Samples IBL per migliorare la qualità dell'illuminazione ed eventualmente ruotiamo l'immagine, per rendere i riflessi sugli oggetti più "confusionari" e non far riconoscere l'immagine originale.




*Influence e mappatura*

*IBL*

Impostiamo a piacere *Skylight Power*, la potenza dell'illuminazione derivante dall'immagine, ma qui in realtà vi conviene fare delle prove di rendering, magari a risoluzioni basse o con impostazioni di rendering modeste nella sezione Render, per trovare il valore che preferite; nel mio caso ho messo 1.5 e, lo ricordo, la Texture su World è l'unico illuminante presente nella scena: non ci sono Lamps, fonti di luce o altri effetti.

Vediamo i rendering, ottenuti in modalità Photon Mapping con 10 milioni di fotoni sia su Diffuse che su Caustic Photons e 4 rimbalzi e 24 Samples per il Final Gathering; il primo rendering mostra il risultato ottenuto con mappatura AngMap, il secondo con Sphere.

*Il primo rendering, con mappatura AngMap*

*Il secondo rendering, con mappatura Sphere*

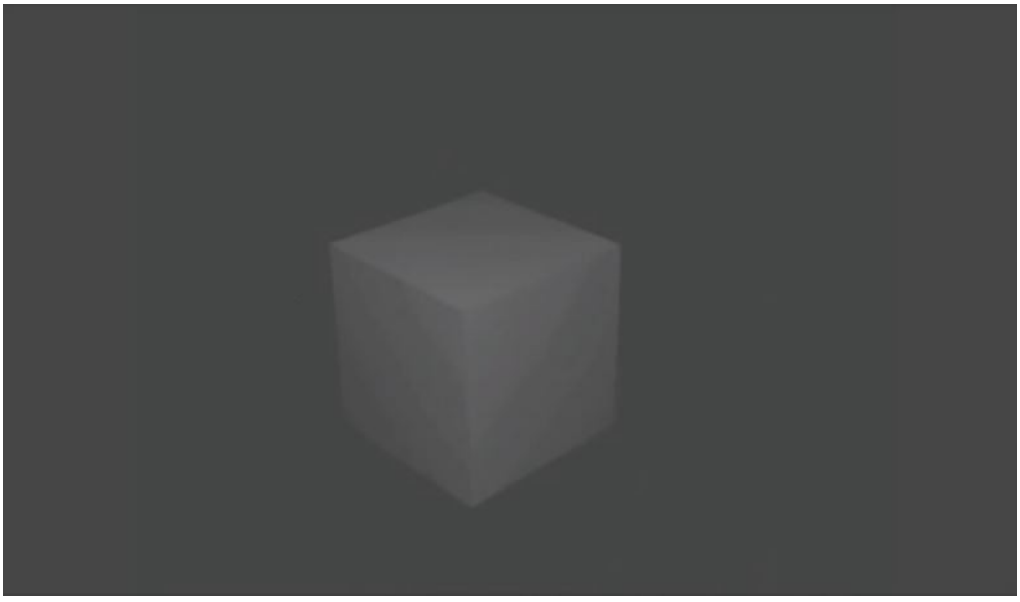
I risultati cambiano in base all'immagine utilizzata e alla modalità di mappatura, quindi vi consiglio di procurarvi un discreto set di immagini (ad esempio fotografie di interni, panorami, immagini fantasy, ecc...) e di fare delle prove, magari pre-elaborando le immagini (ad esempio cambiando contrasto e luminosità, oppure variando le curve tono RGB dell'immagine o ancora passandola in scala di grigi); questo tutorial serve a mostrare più che altro la tecnica.

*  *  *

# Cycles per Blender 3D (tutorial con file blend in regalo)

Questo è il primo tutorial riguardante Cycles, il nuovo motore di rendering node-based integrato in maniera nativa in Blender 3D a partire dalla versione 2.61 del programma; raccolgo materiale su Cycles da un po' ma ho aspettato la release ufficiale prima di realizzare dei tutorials: ora finalmente possiamo iniziare!
In questo tutorial vedremo gli elementi fondamentali del nuovo motore di rendering ed esamineremo una scena che potrete scaricare gratuitamente e senza registrazione dal mio sito web, in modo da poter fare le vostre prove con materiali, textures, illuminazione, eccetera, e magari pubblicando poi i vostri test nel Forum del sito.

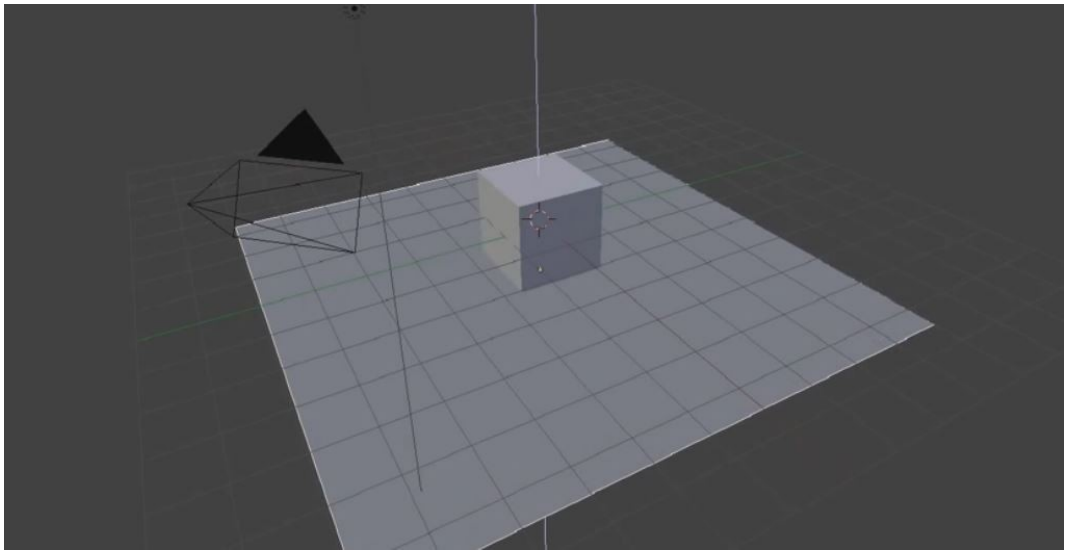
Cycles è un motore di rendering realizzato da Brecht van Lommel.
Cycles è perfettamente integrato in Blender 3D. In passato ho pubblicato qualche videotutorial su Yafaray, un motore di rendering fotorealistico utilizzabile anche in Blender al posto del motore di rendering standard del programma, Blender Render (il motore che utilizziamo di default quando premiamo F12 e renderizziamo la scena 3D, visibile nell'immagine seguente).



*Il rendering della scena di default con Blender Render*

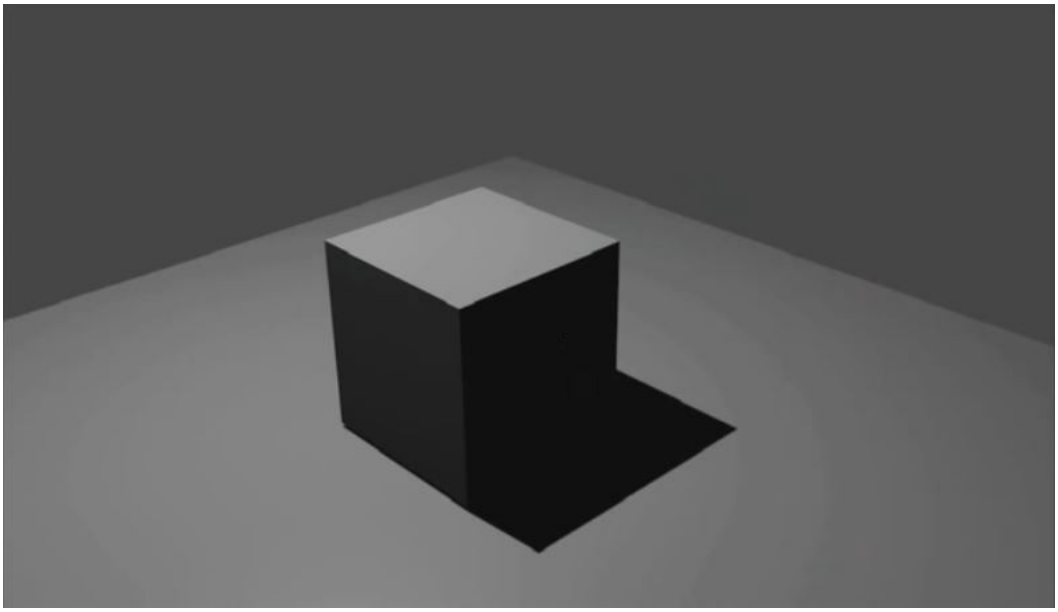
Se sposto un po' il cubo nella 3D View devo avviare nuovamente il rendering perché, con Blender Render, ogni volta che facciamo una modifica dobbiamo ri-renderizzare la scena per vedere gli effetti nel rendering finale: è una considerazione apparentemente banale, ma aspettate di vedere cosa possiamo fare con Cycles e capirete il perché di questa osservazione!

Aggiungo un Plane sotto al cubo presente di default e sposto la fonte di luce per parlare anche di un'altra caratteristica di Blender Render, ossia l'assenza (di default, almeno) del calcolo dell'illuminazione indiretta, ovvero quella di rimbalzo: un effetto di illuminazione globale che è fondamentale per ottenere il fotorealismo.



*Il Plane inserito nella scena*

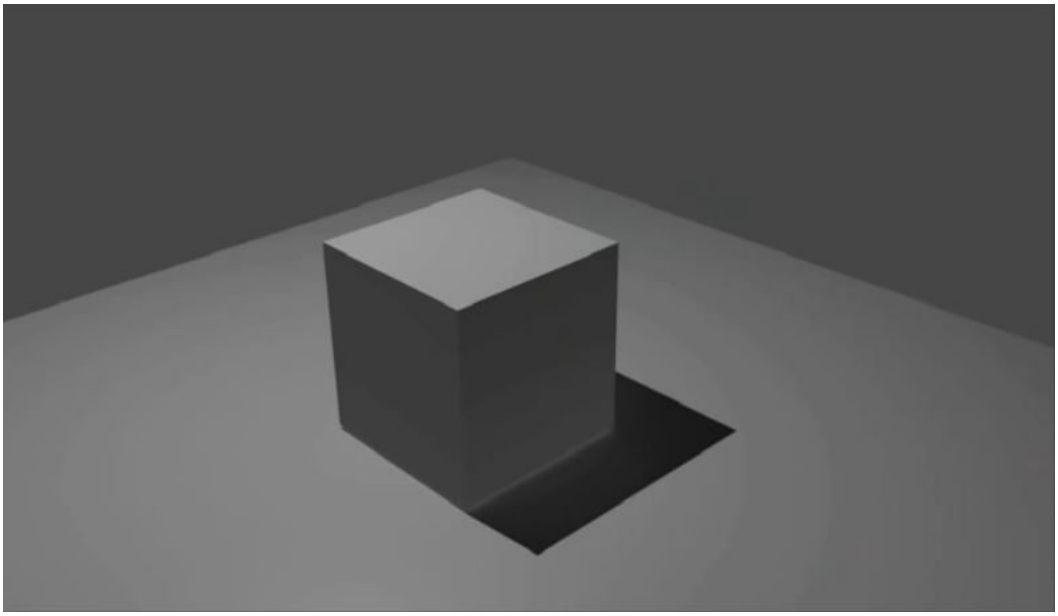
Se si renderizza ad esempio una cucina o comunque un interno in Blender senza Indirect Lighting o senza aggiungere fonti di luce “fittizie”, infatti, le facce degli oggetti non direttamente esposte alle fonti di luce risulteranno nere, come avviene nella scena trattata (visibile nell'immagine seguente) con le facce del cubo non direttamente illuminate.



*Il cubo con le facce nere*

Nel mondo reale, *non* si ha un risultato del genere.

Con Blender qualcosina cambia se si attiva Indirect Lighting, disponibile a partire dalla versione 2.5 di Blender 3D, che consente di calcolare la luce di rimbalzo per un certo numero di rimbalzi, infatti adesso anche la faccia non illuminata del cubo non è più nera (immagine seguente); il problema è che questo algoritmo è molto, MOLTO lento e, come tutto il resto del motore di rendering nativo di Blender, Blender Render, fa uso della CPU, uno collo di bottiglia notevole considerando che ora le GPU, soprattutto quelle con tecnologia CUDA, sono molto più potenti anche delle CPU di fascia medio-alta.

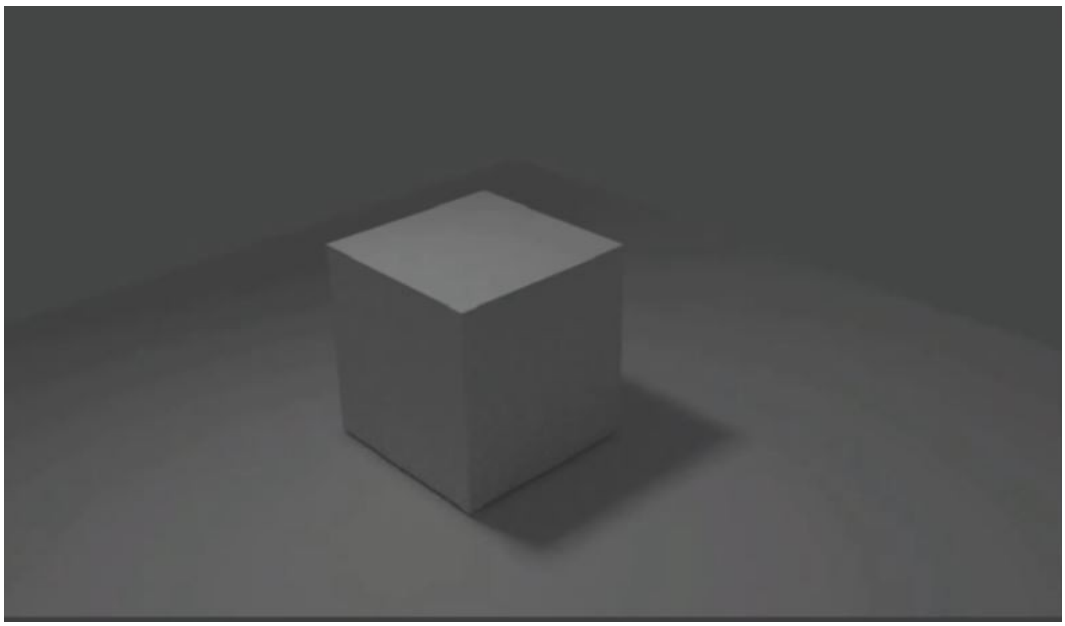


*Il rendering con Blender Render e Indirect Lighting*

Altra cosa da notare: quelle ombre sono troppo, troppo nette.

Stiamo utilizzando una fonte di luce Lamp omnidirezionale che, tecnicamente, dovrebbe avere la dimensione di un punto, quindi giustamente non proietta cono di penombra, ma nel mondo reale nessuna fonte di luce ha tale caratteristica: ecco perché abbiamo ombre sfumate o il famoso cono di penombra delle eclissi solari... con Blender, possiamo introdurre la sfumatura nelle ombre anche per le luci Point Light aumentando i Samples, ma ciò porta ancora una volta a tempi di rendering decisamente più elevati, soprattutto se si desidera un risultato di qualità, ottenibile in genere con le Area Lamp e molti Samples.

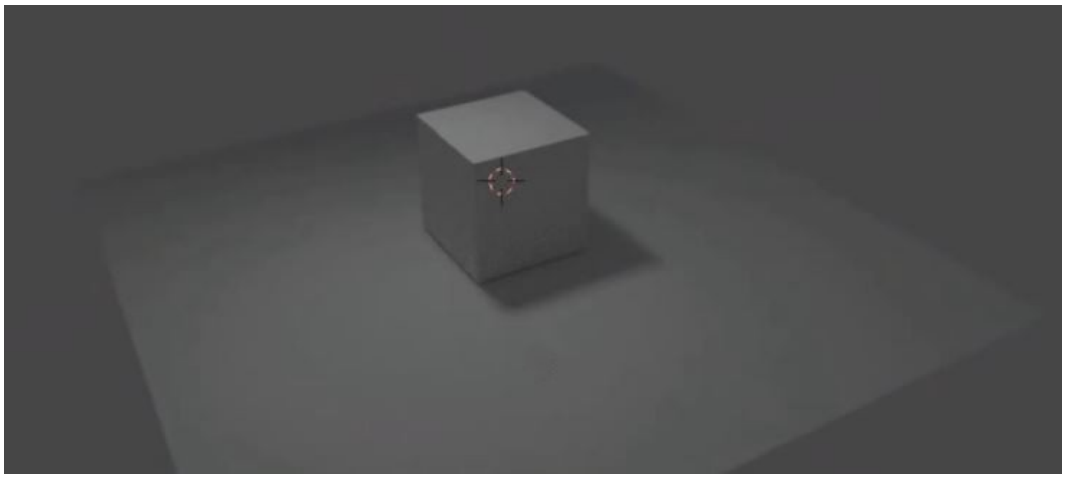
Con la versione 2.61 di Blender, stiamo uscendo dal medioevo: tra i motori di rendering nativi, selezionabili dal selettore Engine nella finestra Info, in alto al centro, abbiamo Cycles; selezioniamolo e premiamo subito F12, per avere il primo impatto a sangue freddo con una scena renderizzata col nuovo motore.



*Il primo rendering con Cycles*

La prima cosa che notiamo è che la scena è stata renderizzata velocemente, in circa 2 secondi (la CPU utilizzata è un i7 con 4 core, 8 thread): “velocemente “se si considera che c'è già l'illuminazione indiretta e le ombre sono sfumate, come visibile nell'immagine precedente.

Torniamo comunque alla 3D View per osservare una funzionalità davvero interessante: la possibilità, scegliendo la nuova voce "Rendering" (disponibile appunto con Cycles) nel selettore Viewport Shading della 3D View, di visualizzare la scena renderizzata, ma ancora modificabile in tempo reale, all'interno della 3D View stessa, senza dover premere F12 e osservare di volta in volta il risultato in una finestra "UV – Editor delle immagini"!



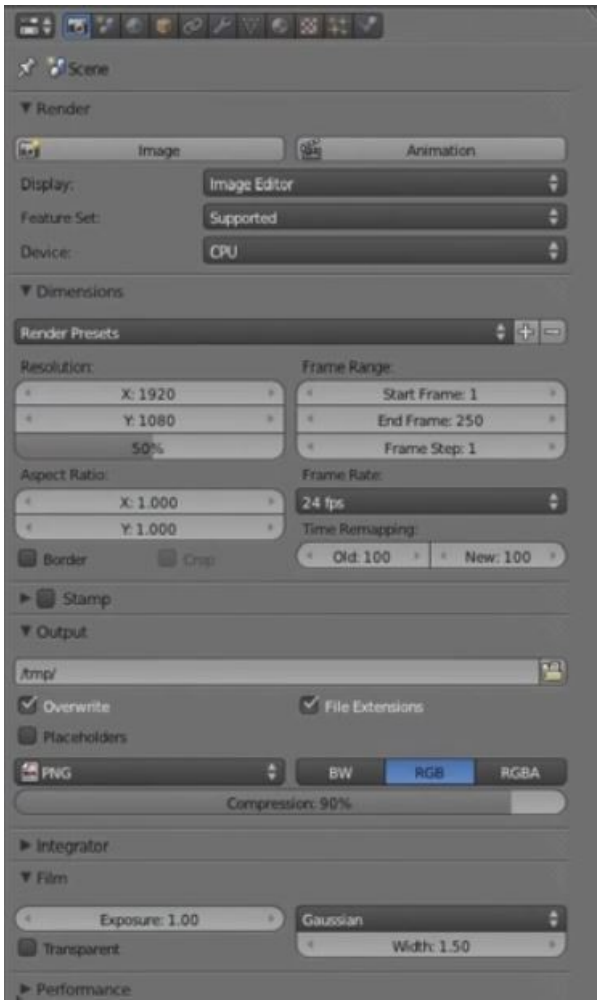
*La Preview di rendering interattiva in tempo reale*

I manipolatori e la griglia non sono visibili, ma è possibile selezionare un oggetto con un click del tasto destro oppure scegliendolo dall'Outliner e digitare ad esempio *G X* per traslarlo in tempo reale sull'asse X, osservando di volta in volta il nuovo rendering; all'inizio il rendering sarà molto granuloso (o, come si dice in questi casi, "rumoroso"), tuttavia non appena ci fermeremo (ossia non modificheremo nulla nella scena: né il punto di vista, né materiali o posizioni degli oggetti), a poco a poco la qualità dell'immagine migliorerà.

Ciò è dovuto al fatto che il rendering è realizzato, in un certo senso, in maniera incrementale, effettuando vari passi o iterazioni detti "Samples", indicati in alto a sinistra nella finestra del rendering, con l'immagine che sarà più pulita e accurata quanti più Samples verranno effettuati.

Ogni volta che cambiamo qualcosa in tempo reale, la 3D View in modo Rendering rifarà i calcoli dal primo Sample fino a un certo numero di iterazioni (di default 10, ma vedremo come cambiarlo).

Le voci riguardanti i calcoli da fare (i Samples, i rimbalzi da calcolare, le caustiche, eccetera) si trovano nella sezione Integrator della scheda Render, all'interno della Properties Window; notiamo in particolare due voci Samples: una per la View, ossia la vista 3D Rendering interattiva che vediamo a video nella 3D View, e una per il rendering, ossia il classico rendering che viene avviato, anche qui, con F12, per cui magari ora possiamo lavorare, ad esempio, con 10 Samples, per avere un'anteprima del rendering in tempo reale senza impiegare molto tempo e poi avviare il rendering finale, visto sempre dalla Telecamera, con 100 o più Samples, per avere un'immagine "pulita", poco rumorosa.



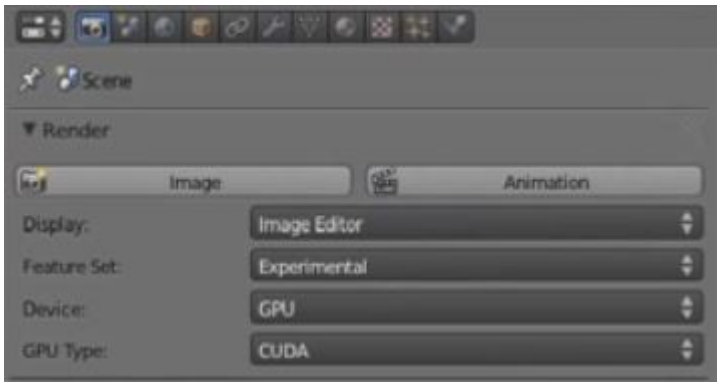
*Parametri e impostazioni nella scheda Render*

Nota importante: se mettiamo 0 in Preview Sample, l'algoritmo non terminerà più, aggiungendo continuamente Samples, quindi migliorando sempre più la scena... caratteristica utile magari per capire quanti Samples ci servono per il rendering finale, soprattutto se dobbiamo fare delle animazioni, perché possiamo mettere 0 in Preview, verificare a quale Sample siamo arrivati leggendo la relativa scritta in alto nella finestra 3D View e, se notiamo che ad esempio già con 100 o 200 Samples la scena è "pulita", possiamo fermare la Preview, scrivere 200 nel campo Render Samples e avviare il rendering finale, sia di un'immagine che di un'animazione.

Ok, abbiamo visto che con 10 Samples e un'immagine rumorosa ci voleva poco tempo, ma già per arrivare a 100 Samples stiamo impiegando un bel po'...

Nella scheda Render, quindi, scegliamo *Feature Set Experimental*, in alto, quindi cambiamo il dispositivo di rendering, *Device,* da CPU (ossia il processore) a GPU (la scheda video), operazione

importantissima se la nostra GPU è più veloce della CPU, anche perché Cycles sfrutta la tecnologia CUDA, presente su certe schede NVIDIA.



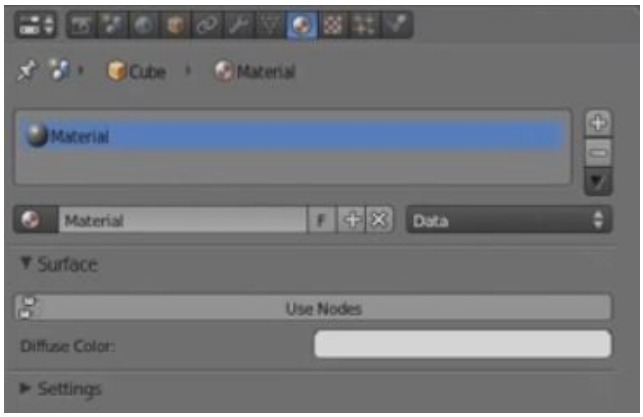

*Impostare la GPU come dispositivo di rendering*

Il mio pc ha proprio una scheda video NVIDIA GTX 470 (provvista di tecnologia CUDA) e la differenza si vede, infatti adesso arrivo a 100 Samples in circa 8 secondi contro i 36 di prima.

Da qui in poi, soprattutto per la scena della cucina, che vedremo a breve, utilizzerò la GPU CUDA per Preview e rendering.

È possibile ovviamente utilizzare due o più View 3D, magari con viste Solid, se questo ci aiuta a trasformare meglio la scena, visto che abbiamo bordi ben definiti per gli oggetti, icone per le fonti di luce o le Empty, manipolatori di trasformazione eccetera, guardando comunque l'anteprima in una 3D View Rendering.

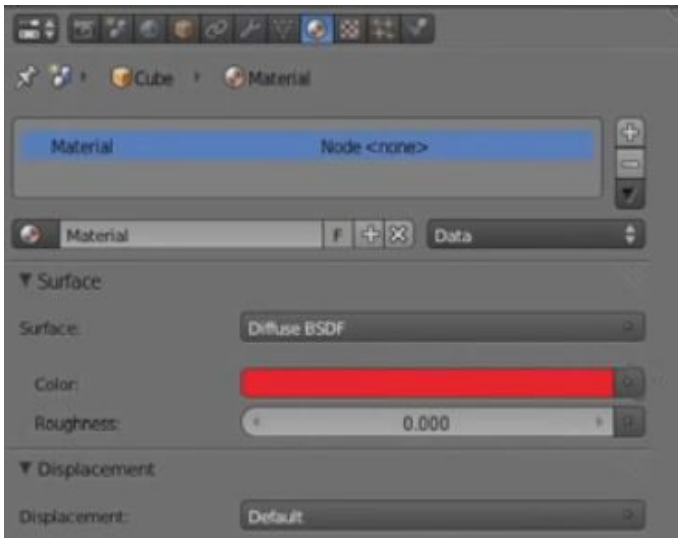
Selezioniamo il cubo e apriamo la scheda Material, nella Properties Window, per avere un primo impatto coi Materials in Cycles prima di esaminarli con la scena della cucina.




*La scheda Material del cubo*

Cos'è cambiato da Blender Render? Tutto!

Adesso i Materiali hanno, di base, un colore per l'ombreggiatore diffuso, ma funzionano con i Nodi, come vedremo; per prima cosa, quindi, cambiamo il colore diffuso dell'oggetto, per osservare l'effetto in tempo reale, quindi attiviamo i Nodes: notiamo la presenza di una sezione chiamata Surface che serve a definire, appunto, l'ombreggiatura sulle facce, sulla superficie dell'oggetto.

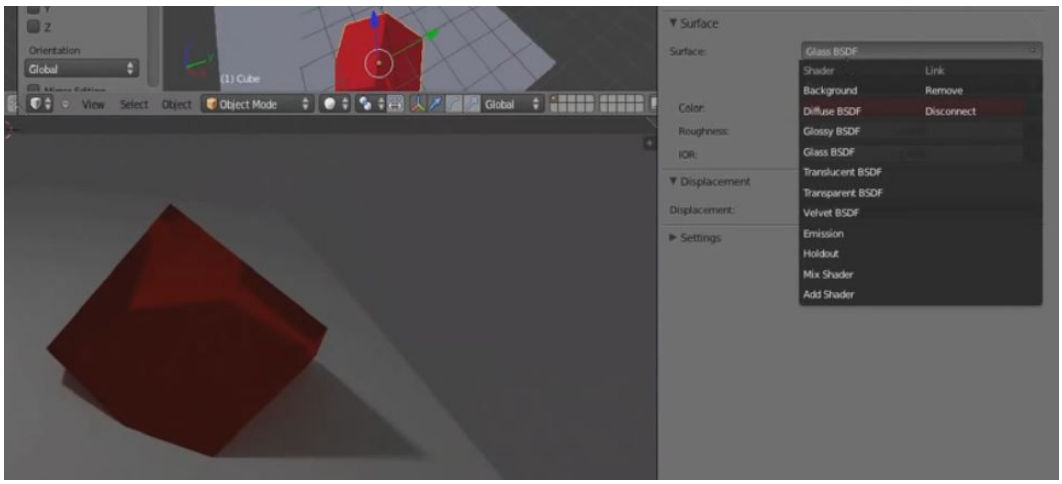



*Nodes e Surface*

Qui la prima cosa da fare è scegliere il modello di ombreggiatura, di default Diffuse: una specie di Lambert di Blender Render.

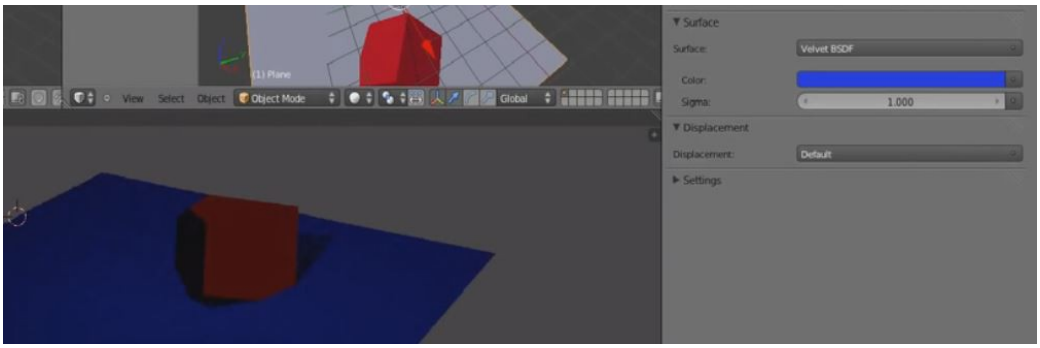
Accanto al colore e ad altri box notiamo un pulsante con un cerchietto: cliccando si aprono dei menù con varie opzioni relative alla voce scelta, per cui ad esempio possiamo scegliere se utilizzare una Texture immagine o una procedurale, come Noise o Wave, per definire l'aspetto diffuso dell'oggetto; su tutte queste cose torneremo comunque con calma nel corso del tutorial.

Gli Shader hanno nomi intuitivi, nella maggior parte dei casi: Diffuse è tipo Lambert, Glossy per i materiali riflettenti e c'è un materiale ad hoc per il vetro, per cui per diamanti, vetri, acqua ed altri elementi del genere è sufficiente scegliere Glass ed impostare l'indice di rifrazione, come sto facendo col cubo (immagine seguente).



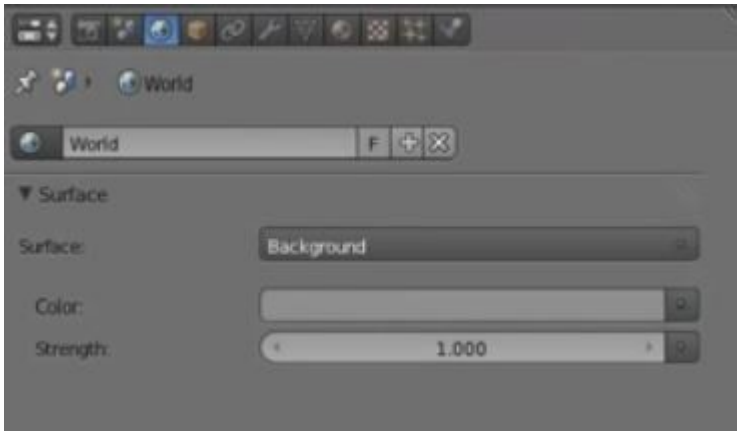

*Utilizzo dello Shader GLASS per il cubo*

Selezioniamo adesso il Plane con un click del tasto destro su tale oggetto nella 3D View Rendering ed impostiamo un Material anche per questo elemento: cambiamo ad esempio il colore in blu e scegliamo un Material Velvet (letteralmente, velluto) per osservare l'effetto che fa.



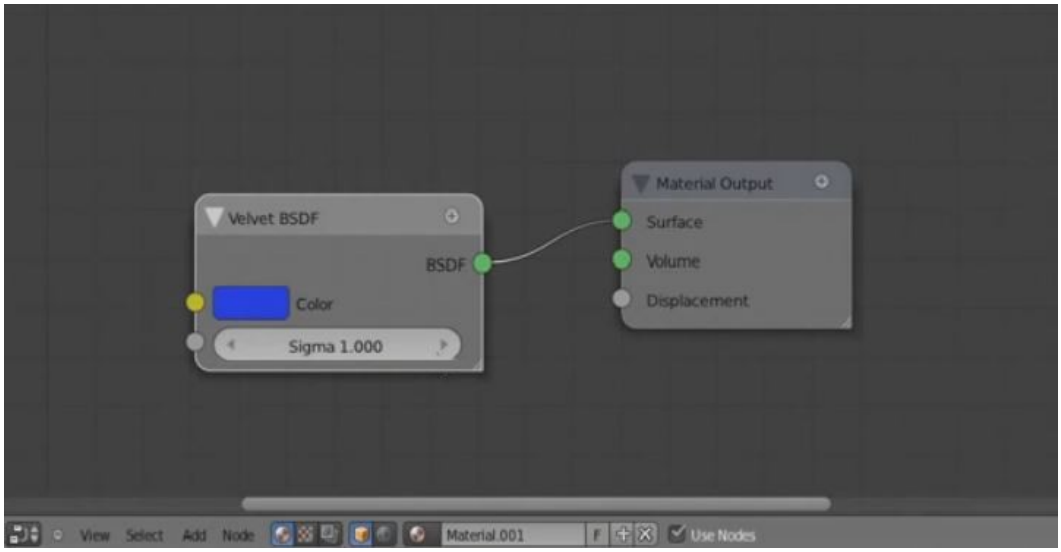

*Blue VELVET per il Plane*

Anche World, lo sfondo dell'universo virtuale, può avere un colore o una Texture, per cui nella scheda World, nella Properties Window, possiamo cambiare anche qui il colore e attivare i Nodi per definire ombreggiatori, Texture e l'intensità dell'illuminazione in arrivo da World, con Strength.




*Impostazioni World*

Abbiamo detto, all'inizio, che Cycles è un motore Node-based, quindi tutte le impostazioni dei Materials possono essere regolate dalla finestra Editor dei Nodi, scegliendo la prima opzione (quella di default, ossia Material Editor), cosa ben diversa dal Compositing Editor, per la post-produzione, terza icona nell'header... restiamo nella prima ed esaminiamo i Materials degli oggetti selezionati di volta in volta da questo punto di vista.



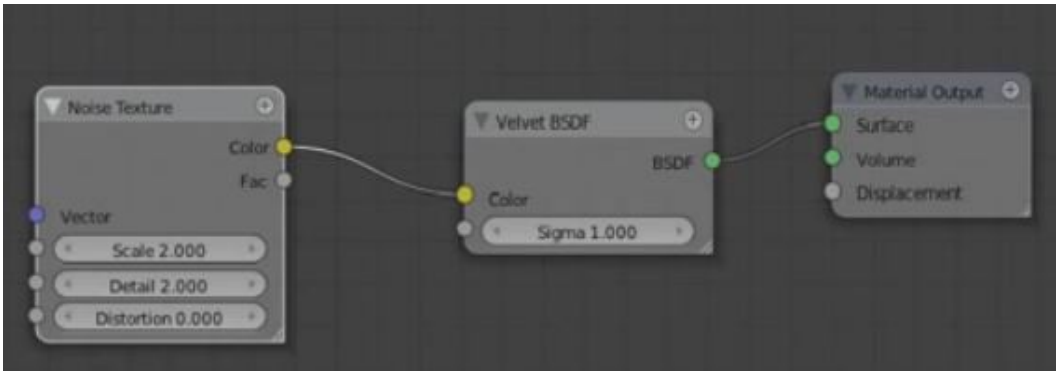
*I Nodi per i Materials*

L'aspetto finale del Material è quello in arrivo sul Nodo Material Output e, come possiamo notare, ci sono tre porte di ingresso per tale nodo (e, chiaramente, nessuna in uscita).

Le tre porte d'ingresso sono: Surface, per definire l'ombreggiatura delle facce; Volume, per i Materials volumetrici; Displace, per le deformazioni sulle mesh introdotte mediante Textures (come col Modifier Displace, quindi).

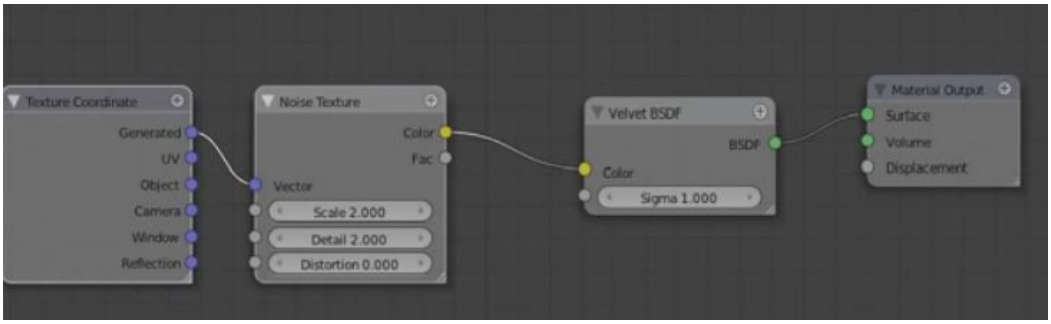
In questo caso è presente un Nodo Glossy che definisce l'ombreggiatore base e tale nodo è collegato a Material Output, quindi banalmente questo materiale è un semplice Glossy di colore rosso, mentre scegliendo il Plane avremo Velvet, e così via.

Se a questo punto inseriamo una Texture per l'aspetto del Material Velvet (ad esempio una Texture Noise) nella scheda Material all'interno della Properties Window, verrà creato automaticamente un Nodo Texture collegato in input al colore del Nodo Velvet, e questo perché adesso l'aspetto dell'ombreggiatore è regolato, appunto, da una Texture.




*Il Nodo della texture Noise*

Possiamo variare le impostazioni della Texture (come dimensione o distorsione) dalla scheda Material, oppure dal Nodo nell'Editor dei Nodes, mentre per impostare le coordinate di mappatura (ad esempio Generated o UV Mapping) dobbiamo agire sulla sezione Vector... ed ecco allora un Nodo Texture Coordinate, quindi possiamo cambiare la modalità di mappatura al volo dall'Editor, se preferiamo, cambiando semplicemente i collegamenti.



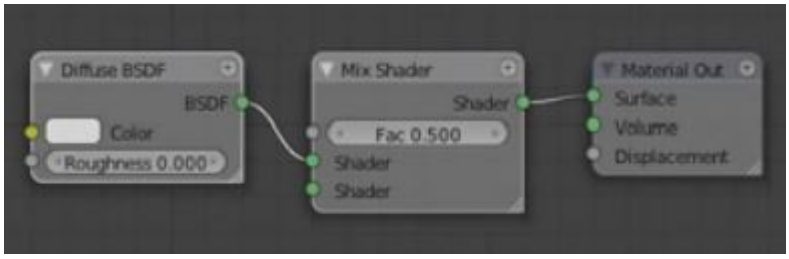

*La sezione Vector e il Nodo Texture Coordinate*

Possiamo aggiungere delle caratteristiche al Material direttamente dal Node Editor aggiungendo i Nodi che ci servono ed effettuando i vari collegamenti.

Per visualizzare il menù dei Nodes disponibili possiamo scegliere Add nell'header della finestra Nodes Editor oppure premere SHIFT A mentre il cursore del mouse si trova su tale finestra; ad esempio, possiamo aggiungere un Nodo Texture di tipo Wave e collegarlo al canale Displacement del Material, variandone poi i parametri, oppure ad altri Nodi del Material.

Le azioni svolte nel Node Editor avranno degli effetti anche sulla struttura della scheda Material, nella Properties Window, per cui diciamo che potete scegliere il metodo che preferite a seconda dei casi.

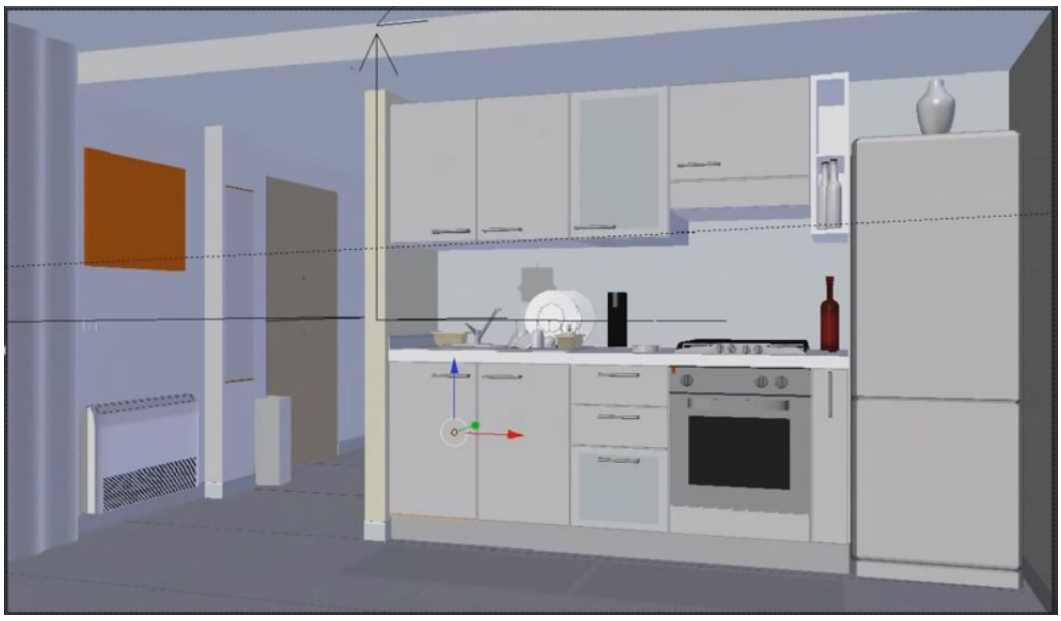
Glossy è quindi un Material riflettente, Transparency è ovviamente trasparente, Diffuse è un lambertiano e così via, ma chiaramente vogliamo poter creare materiali con due o più caratteristiche in un colpo solo, ad esempio un Diffuse con il 20% di effetto Mirror; ebbene, in questo caso dobbiamo utilizzare un ombreggiatore Mix Shader che mette a disposizione due slot per impostare due o più Materials da miscelare... dico "due o più" perché i sotto-materiali di un Mix Shader possono essere a loro volta dei Nodi Mix, quindi possiamo miscelare più effetti in cascata, ad esempio col primo Mix un Diffuse e un secondo Mix che miscelerà invece Trasparenza e Glossy: l'importante è impostare correttamente Factor per ogni Mix, con valori tra 0 e 1 e che serve a determinare l'influenza del primo e del secondo Nodo sull'effetto finale.



*Il Nodo MIX*

*  *  *

Ok, fatte queste premesse passiamo ad una scena decisamente più interessante: esamineremo una cucina che ho modellato qualche mese fa e renderizzato con Blender Internal; ho pubblicato quei rendering, effettuati con Indirect Lighting e che mostravano una discreta qualità, in alcuni tutorial e articoli pubblicati sul sito... i rendering erano sì di buona qualità ma richiedevano circa 40 minuti su questo computer con i7 e Blender Internal, per cui vediamo come vanno le cose ora con Cycles!

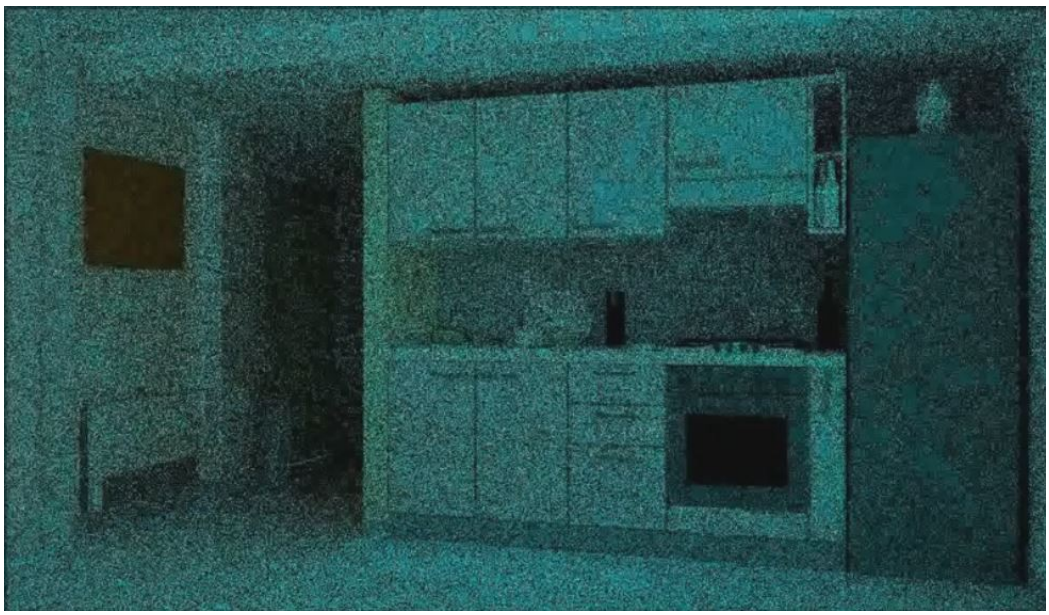


*La scena della cucina*

Potete scaricare liberamente questa scena all'indirizzo *www.redbaron85.com/cucina.zip* , in modo da provare voi stessi i vari Materials ed effetti: il file .blend è un piccolo regalo di Natale per consentirvi di provare Cycles con una scena già pronta e piena di elementi (e magari poi potrete pubblicare i vostri risultati nella Gallery degli Utenti, nel Forum). Il file è *packed*, quindi contiene le Textures da utilizzare, ma attenzione perché all'inizio avrete un problema con le Texture, lo vedremo tra poco e vedremo ovviamente come risolverlo.

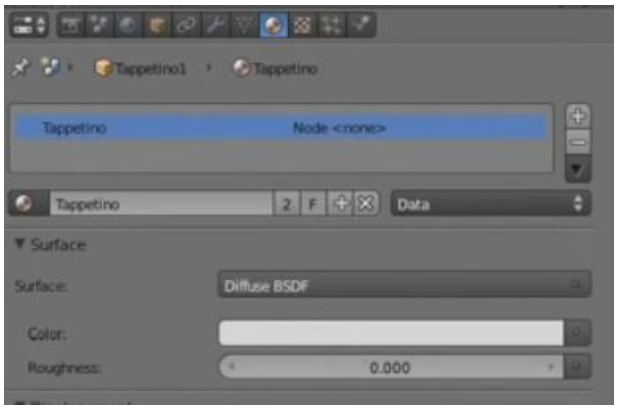
Un rendering con Blender Internal impiega tempi biblici, per cui proviamo ad utilizzare subito Cycles. Caricato il file, fondiamo le finestre in modo da avere solo la vista Camera e passiamo al motore di rendering Cycles, quindi scegliamo la modalità di visualizzazione Rendering nella Viewport Shading ed osserviamo il primo rendering, effettuato di default con 10 Samples... effettuato molto lentamente, in questo caso, visto che il Device di default è la CPU, per cui ora passo a GPU CUDA nella scheda Render, nella Properties Window... in ogni caso, 10 samples di Preview sono troppo pochi per una scena del genere: l'anteprima è ancora troppo rumorosa, per cui passiamo ad esempio a 100 o più (potete provare anche con 0, ovviamente).

Il risultato è visibile nell'immagine nella pagina seguente.



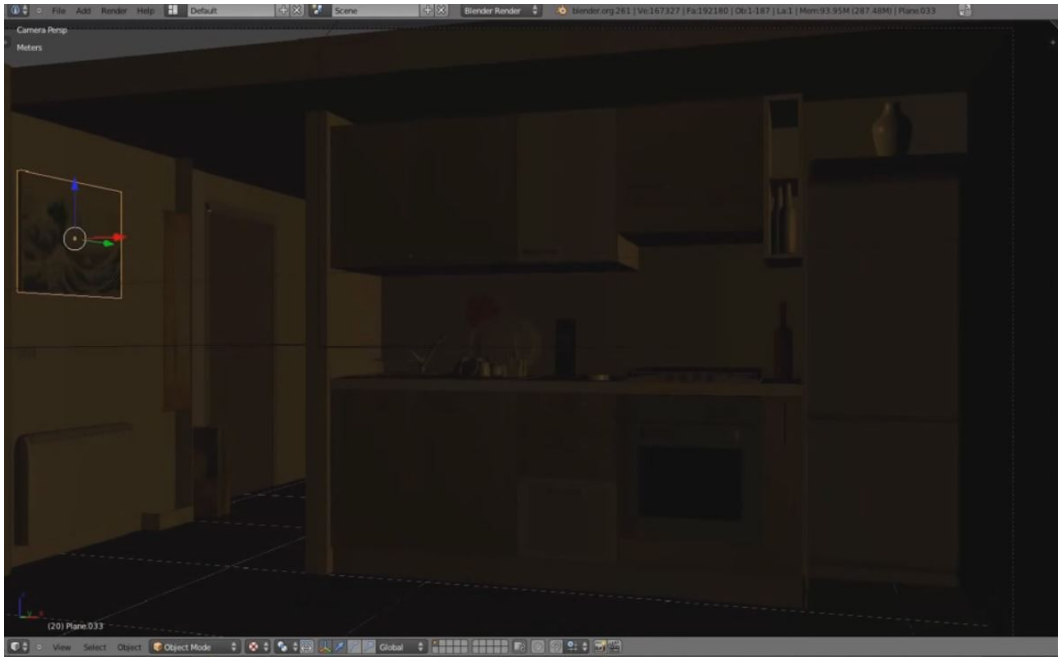
*La Preview in modalità Rendered interattiva (con pochissimi Samples e scarsa illuminazione)*

Torniamo per un attimo alla vista Solid per individuare e selezionare facilmente uno dei tappetini, quindi passiamo alla finestra Material per aggiungere un Materiale e fornire una Texture immagine, già mappata con UV Mapping (per la cronaca, tutti gli oggetti della scena sono già unwrappati, quindi potete applicare le immagini con coordinate UV).




*Il Material per i tappetini*

Attenzione però: sto passando al motore di default, Blender Render, per farvi notare che per certi Material in modalità di visualizzazione Textures le immagini sono già visibili nella finestra 3D View, in maniera interattiva (vi ricordo che in questo momento le immagini sono impacchettate, "packed", all'interno del file .blend, così come lo scaricherete da *www.redbaron85.com/cucina.zip* ).



*Visualizzazione delle Textures in modalità Textured*

Adesso torniamo al motore di rendering Cycles: ho selezionato il quadro, quindi procediamo con questo oggetto (poi passerò al tappetino). Scegliamo il Material del quadro ed aggiungiamo una Texture al canale del colore dell'ombreggiatore Diffuse del Material, quindi scegliamo l'immagine "*Hokusai – La grande onda*" dal menù *Browse Image to be Linke*d: l'immagine è visibile correttamente nella 3D View, in modalità di visualizzazione Textured, ma adesso passiamo alla modalità Rendering...

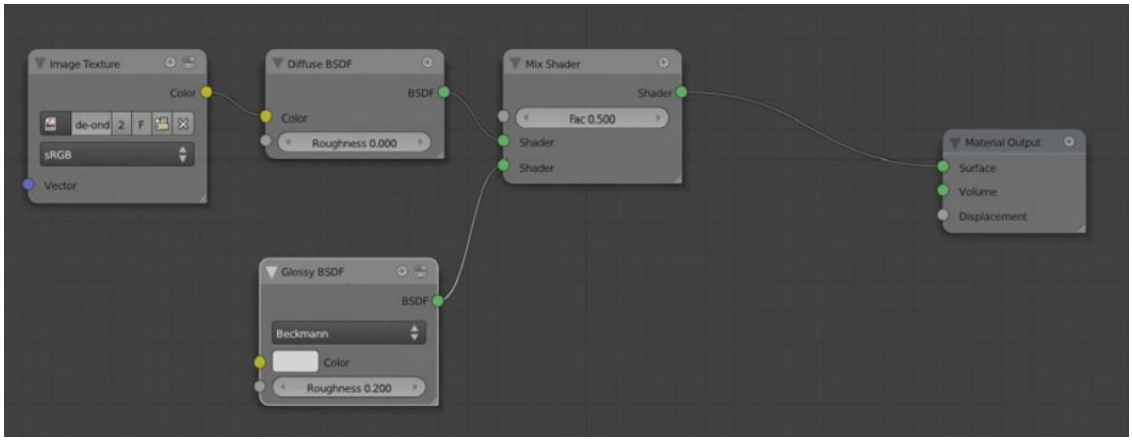
*Hokusai - "La grande onda"... non si vede!*

... quel Plane è nero! L'immagine non è visibile! Ciò è dovuto al fatto che Cycles non carica correttamente le immagini dal file *packed*: non so se si tratta di un bug di questa prima release o di un'impostazione del motore di rendering, comunque quello che dobbiamo fare in questi casi è "spacchettare" i file delle risorse dal file Blend originale e ricaricare i file immagini delle Textures, per cui scegliamo *File - External Data - Unpack into Files* e la prima voce di tale menù, in modo da creare, nella stessa cartella ove si trova il file *blend* della scena, una sottocartella "textures" contenente le immagini che ci servono.

Ricarichiamo il file immagine per la Texture del Material cliccando su Open e scegliendo il file da disco ed ecco nella Preview (e quindi anche nel rendering finale) l'immagine applicata correttamente alla mesh.

*Il Plane del quadro texturizzato correttamente*

A questo punto possiamo passare ad esempio al Node Editor e personalizzare il Material, aggiungendo nodi e collegamenti per implementare vari effetti, oppure possiamo svolgere tali operazioni dal pannello Materials, nella Properties Window; ad esempio, se vogliamo rendere questo Material leggermente riflettente (per simulare ad esempio un poster o un puzzle dietro una lastra di vetro o plexiglass) possiamo cambiare il Material in Mix ed impostare un Material Diffuse di base con la Texture per definire l'immagine e un Material Glossy con colore bianco per implementare l'effetto Mirroring (le riflessioni) dosando poi l'entità di tale effetto variando il valore di Factor del nodo Mix.



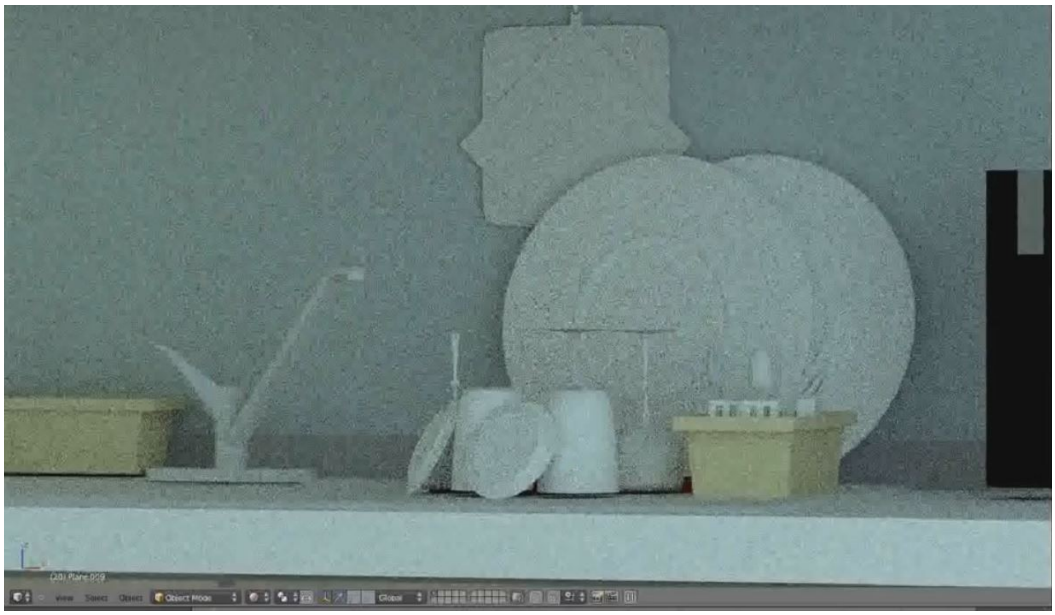

*Rendere il Material leggermente riflettente con Glossy*

Adesso sto applicando la Texture immagine anche al pavimento (già dotato di Layer UV per la mappatura del relativo file immagine) e sto cambiando un po' il colore e l'intensità di World per esaminare le differenze di illuminazione introdotte nella stanza cambiando appunto questi parametri per la luce in arrivo dall'esterno, come visibile nell'immagine seguente.

*Effetti dovuti alle nuove impostazioni di World*

Aggiungiamo altri Materials e altre Textures, ad esempio il legno per i mobili (con Texture immagine) o un Material Velvet per il tappetino con relativa immagine da fotografia... ecco, qui si capisce perché bisognerebbe scegliere nomi significativi per i file, infatti adesso spero di ricordare quello giusto (dovrebbe essere DSC3195), ma voi non fate così e date nomi chiari ai vostri file immagine, ai file blend, agli obj, eccetera!

Aggiungiamo un Material Glossy per le maniglie dei mobili, variando un po' il colore rispetto a quello di default, zoomando poi per osservare meglio gli oggetti disposti sul lavabo, ossia il rubinetto, i bicchieri e i piatti.



*Particolare della zona del lavabo*

Selezioniamo ad esempio il cucchiaino, che condivide lo stesso Material "*metallo cromato*" utilizzato anche dal lavabo e dal rubinetto, ed impostiamo un Material Glossy lasciando le impostazioni di default, che per i metalli così vanno più che bene; notate, a mano a mano che aumentano i Samples, come migliora la qualità del Material, ad esempio i riflessi luminosi sul cucchiaino (nell'immagine seguente).

*Il Material GLOSSY in azione*

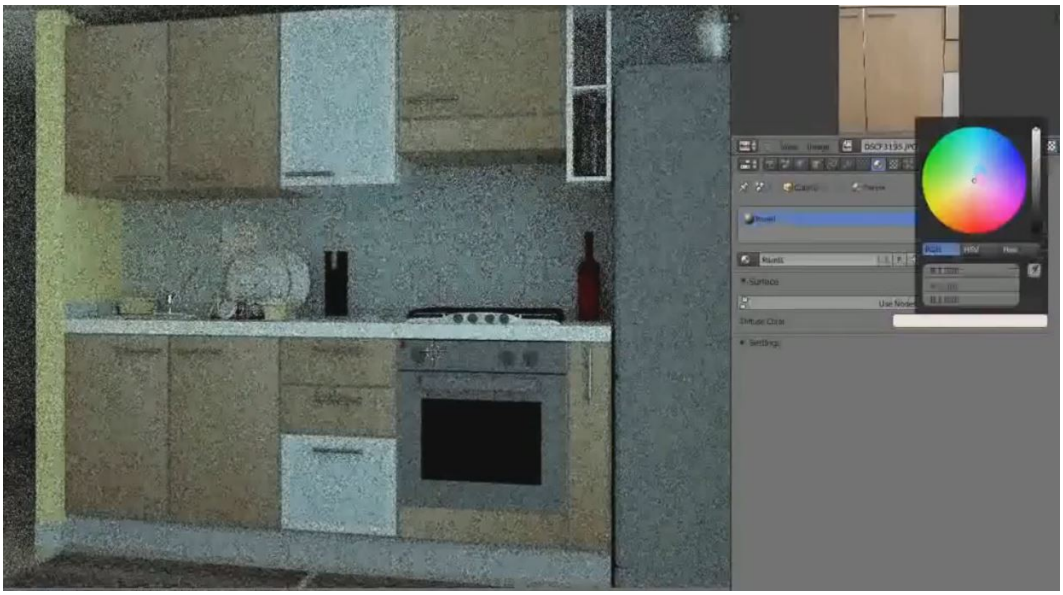
Come detto poco fa, c'è un Material per il vetro che ci consente di realizzare questo tipo di oggetti con un semplice click: proviamolo con i bicchieri, selezionandone uno e scegliendo un Material Glass: fatto! Tutto qui: abbiamo finito con vetro e bicchieri!

*Il Material GLASS per i bicchieri*

Adesso mettiamo un Material Mix per i piatti, miscelando un Diffuse e un Glossy in entrambi i casi di colore bianco, variando un po' Factor per provare i vari effetti.

*MIX per i piatti*

Per Materials semplici come i muri (ad esempio, adesso sto cambiando quello color crema, poi le altre pareti) non è necessario agire sui Nodi: è sufficiente cambiare il colore dell'ombreggiatore, ottenendo alla fin fine un Material tipo Diffuse (un lambertiano semplice).



*Le pareti*

Adesso invece sto azzerando l'intensità dell'illuminazione in arrivo da World perché voglio parlarvi della possibilità di rendere gli oggetti dei veri emettitori luminosi, fonti di luce a tutti gli effetti, grazie al Material Emission, che sto applicando in questo caso alle Tende, che infatti sono diventate luminose ed illuminano la scena.

*EMISSION applicato alle tende*

Ho utilizzato il primo oggetto a portata di mano ma, chiaramente, qualsiasi cosa può utilizzare Emission e tale ombreggiatore può essere inserito come sotto-materiale in un sistema con Mix Shader, quindi potete definire colore, trasparenza, riflessione, emissione luminosa ed altri effetti per i vostri materiali.

Mi avvicino ora al lavabo per mostrare l'effetto di Defocus, la sfocatura dovuta alla profondità di campo della telecamera, implementata da Cycles come proprietà dell'oggetto Camera.

Dopo aver selezionato la telecamera dobbiamo aprire la scheda Object Data nella Properties Window ed impostare il punto di messa a fuoco o specificando la distanza dalla telecamera in Distance, oppure scegliendo un oggetto dalla scena (in modo da tenerlo sempre a fuoco e sfocare il resto, ad esempio nelle animazioni).

*Impostare il punto di messa a fuoco per la telecamera*

La profondità di campo (e quindi la sfocatura) va regolata mediante il parametro Size; bisogna fare delle prove, per trovare il valore giusto.

*  *  *

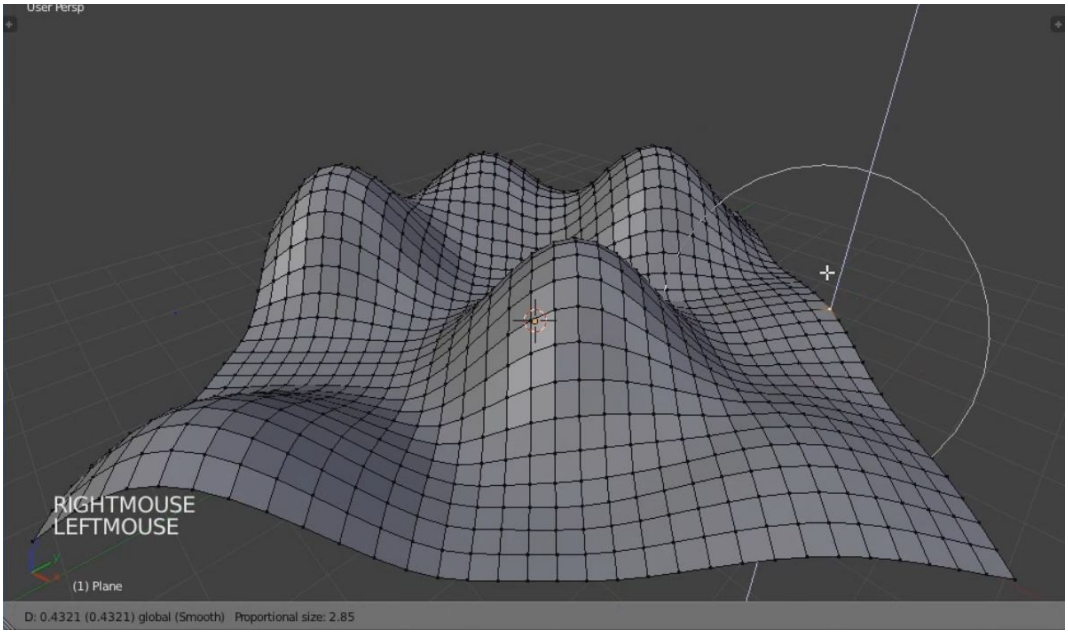
# Un trucchetto per velocizzare le operazioni con il Modifier Boolean

Questo breve tutorial sul modificatore Boolean in Blender 3D nasce da una domanda di un utente del Forum, *"argwin",* esasperato dalla lentezza del modificatore e dai risultati errati ottenuti tentanto di eseguire un'operazione booleana di sottrazione tra due mesh apparentemente banali.

Boolean non è esattamente la punta di diamante di Blender 3D ma certe condizioni, che descriveremo in questo breve tutorial, possono velocizzarlo e consentirgli di ottenere risultati corretti in poco tempo.

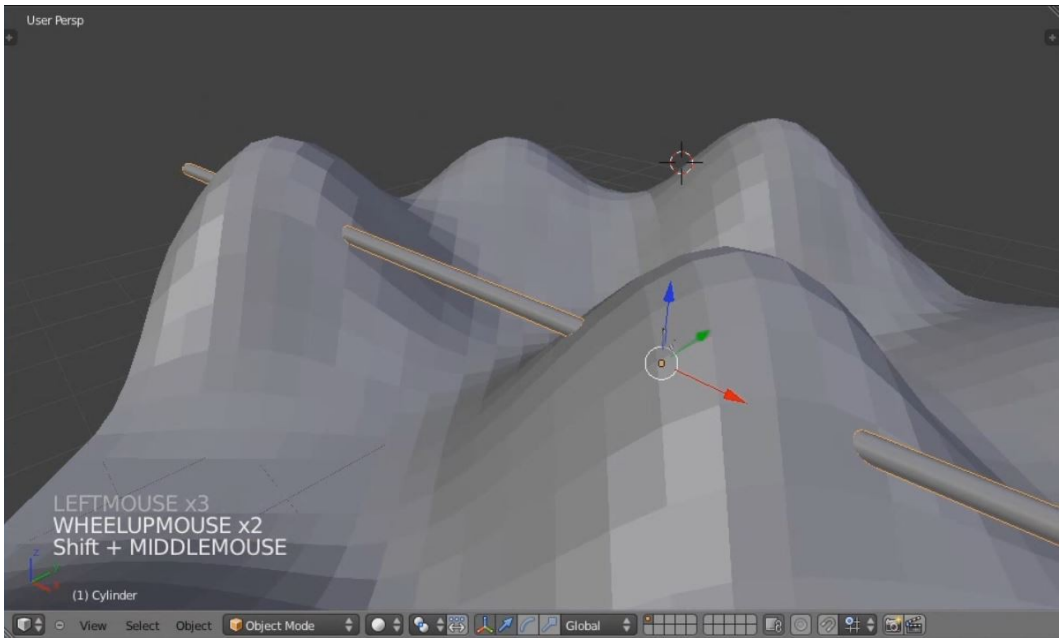
Il concetto chiave è che Boolean lavora molto bene con le mesh chiuse e non manifold, ossia le mesh dove è facile individuare "qual è il dentro e qual è il fuori", in un certo senso, e che abbiano inoltre le normali correttamente calcolate e non siano in modalità di ombreggiatura Shading Smooth, che richiede in generale più tempo e risorse di calcolo (ma questo vale sia in fase che editing che di rendering, non solo con Boolean).

Facciamo un primo esempio trattando proprio il caso di "*argwin*", che intendeva creare delle gallerie attraverso alcune montagne definite mediante un Plane modellato con Proportional Editing, quindi per prima cosa aggiungiamo un Plane alla scena, suddividiamolo tre-quattro volte con Subdivide e poi, mediante lo strumento Proportional Editing, attivabile nell'header della 3D View, deformiamo un po' questa mesh, creando qualche rilievo.




*Modellazione del Plane di partenza*

Torniamo in Object Mode ed inseriamo nella scena una mesh cilindro, anche con un numero abbastanza elevato di segmenti (ad esempio, 120), quindi ruotiamola e posizioniamola in modo da farle oltrepassare una o più colline della mesh Plane che rappresenta il paesaggio.

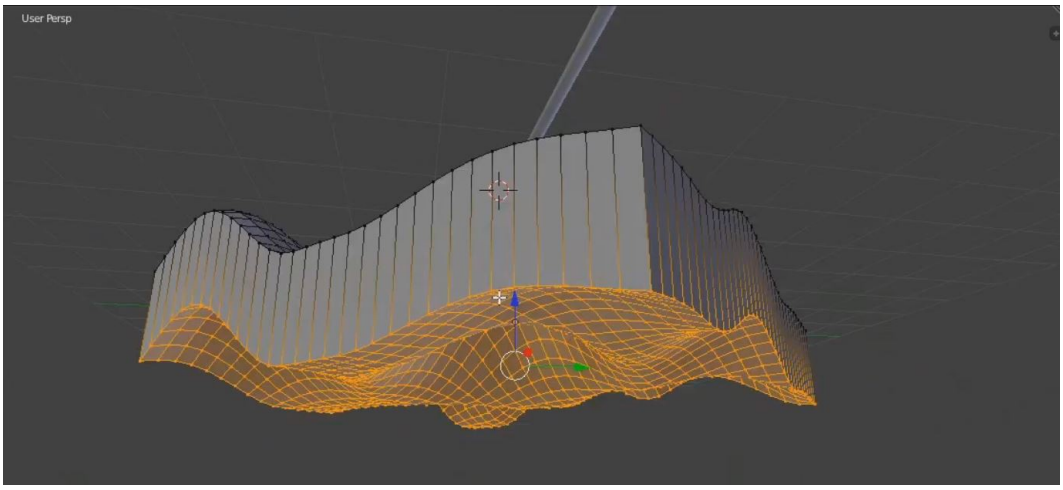



*Il cilindro che passa attraverso il Plane modellato*

*argwin* applicava il modificatore in questa fase, ottenendo però risultati errati (ad esempio, il Plane forato ma con le facce del cilindro attaccate all'esterno, quindi quasi un'unione e una sottrazione, contemporaneamente). Ricordiamo quello che abbiamo detto all'inizio: mesh chiuse, dove sia possibile individuare facilmente il dentro e il fuori.

Il cilindro è una mesh chiusa, quindi non ci resta che chiudere il paesaggio, dotando la sua mesh di una specie di “pavimento”; in questo particolare caso, ho suggerito ad *argwin* di chiudere la mesh seguendo queste fasi:

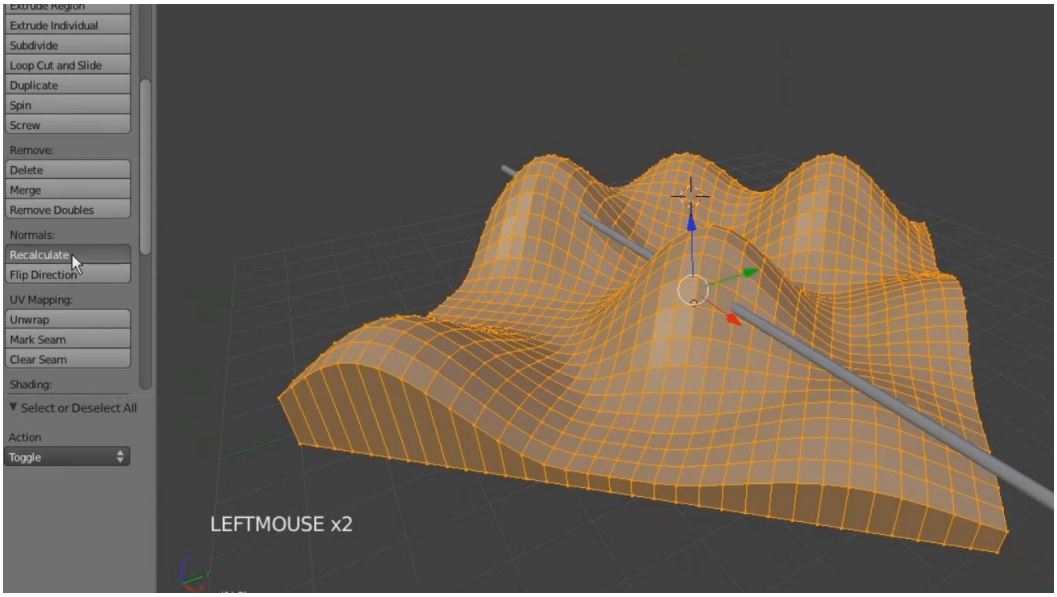
- con il Plane selezionato, andiamo in Edit Mode;
- selezioniamo tutti i vertici con A;
- estrudiamo tutto verso il basso, MOLTO in basso per non creare aree di sovrapposizione;
- scaliamo a zero sull'asse Z i nuovi elementi, ottenuti mediante estrusione, digitando S Z 0 e Invio.



*"Chiudere" il Plane*

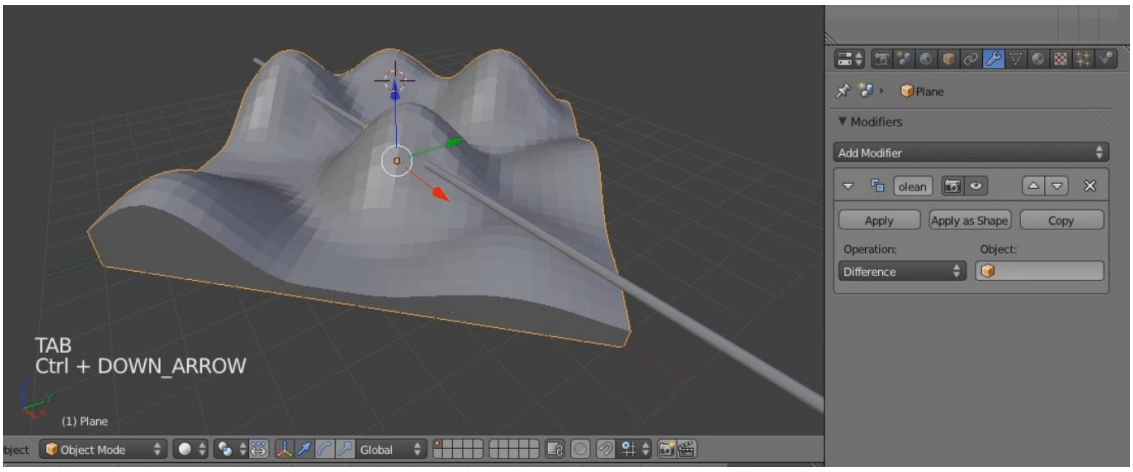
Quest'ultima operazione ha l'effetto di appiattire la nuova selezione che, ai lati, è connessa alla mesh di partenza, costituendo così un volume chiuso, senza aree di sovrapposizione.

Dopo una modifica del genere, è buona norma ricalcolare le Normali, in modo da far capire a Blender cos'è il dentro e cosa il fuori, quindi in Edit Mode selezioniamo tutto e clicchiamo su Recalculate Normals, nella Tool Shelf.




*"Recalculate Normals", nella Tool Shelf*

Adesso possiamo tornare in Object Mode e applicare il modificatore Boolean in modalità "sottrazione" alla mesh del paesaggio, indicando come secondo attore dell'operazione il cilindro.




*Applicazione di Boolean - Subtract*

L'operazione può richiedere più o meno tempo a seconda del numero di facce degli elementi e dell'hardware utilizzato, comunque in questo caso richiede in genere pochi secondi e restituisce un risultato corretto: un esito ben diverso da quello ottenuto senza questi piccoli accorgimenti!

*  *  *

# Compositing: animazioni con canale Alpha e Alpha Over

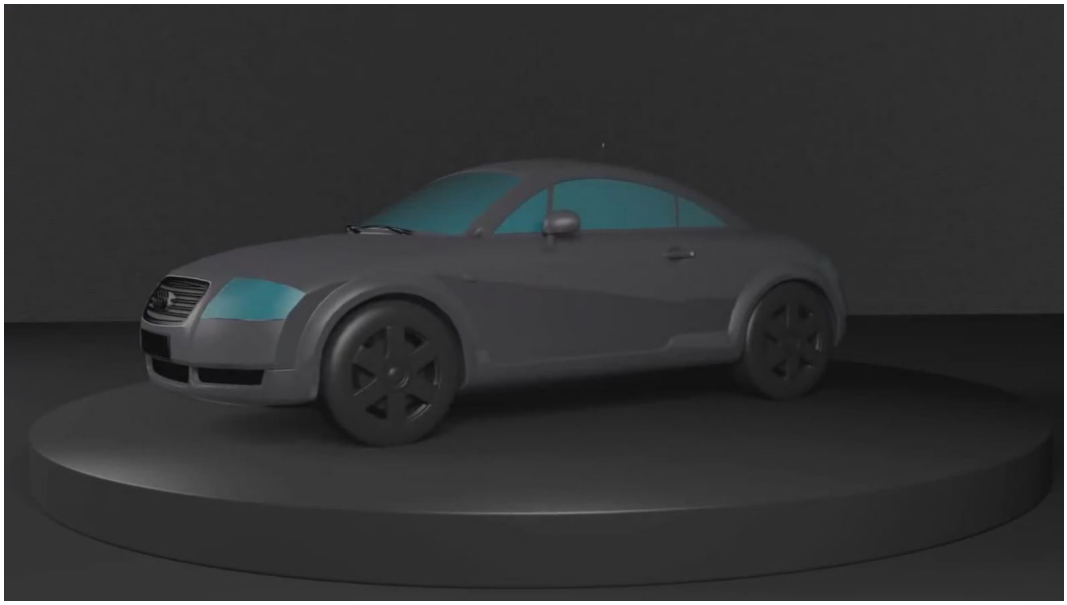
In questo breve tutorial vedremo come salvare, su un file AVI, il canale Alpha di un rendering, in modo da utilizzarlo poi per fornire le informazioni sulla trasparenza di un filmato da comporre successivamente con altri elementi in fase di Compositing in Blender 3D 2.6.

Il tutorial nasce da una richiesta di *Maxxximus* che aveva l'esigenza di renderizzare un'animazione ma memorizzare anche il canale Alpha della stessa nel file creato; il problema è che certi codec, come AVI JPEG, non salvano il canale Alpha, quindi dovevamo pensare ad un modo per memorizzare anche questa informazione ed utilizzarla poi per il compositing successivo.

Agli elementi di base del Compositing in Blender 3D 2.6 ho già dedicato un ebook, per cui in questo tutorial non esamineremo in dettaglio il funzionamento dei vari nodi, mostrando invece i collegamenti necessari per ottenere l'effetto desiderato.

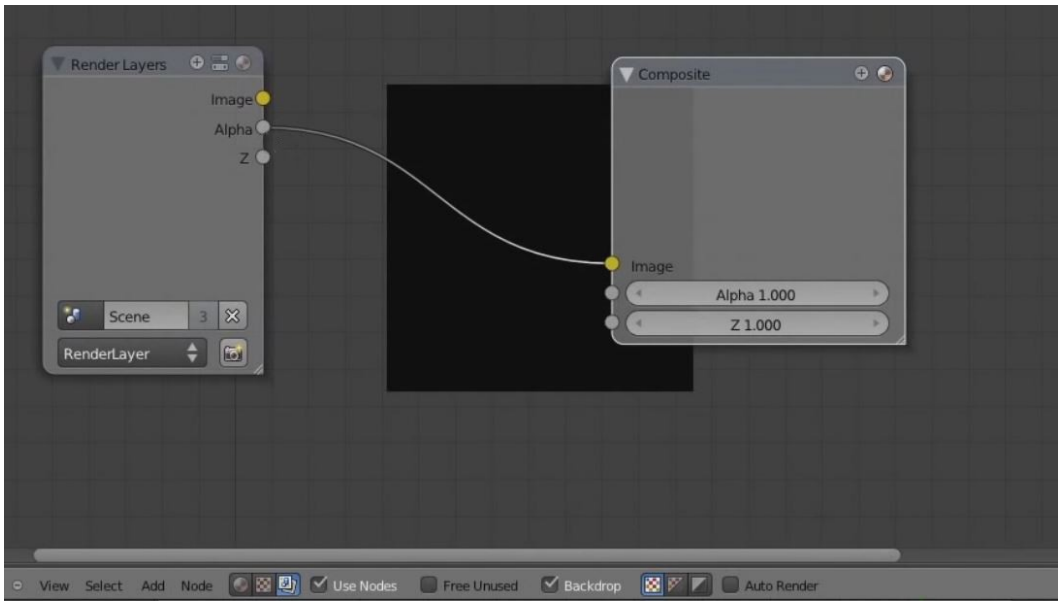
Iniziamo con la prima fase del lavoro: il salvataggio del canale Alpha dell'animazione su un file AVI dedicato. La soluzione che ho proposto a *Maxxximus* prevede la realizzazione di due rendering: il primo per l'animazione "classica" (ossia le informazioni sui colori RGB), il secondo per il solo canale Alpha, da salvare come animazione in scala di grigi (il bianco è l'opacità massima e il nero la trasparenza massima). Vi anticipo comunque che ciò NON comporterà tempi di rendering doppi in quanto, come vedremo, per salvare il solo Alpha possiamo velocizzare notevolmente il rendering con un paio di accorgimenti.

Il primo rendering avviene in maniera "standard", generando un normalissimo file video... non c'è molto altro da dire. Nell'animazione, lo sfondo (che ha valore Alpha 0) è una Texture immagine che dovremo sostituire con un'altra immagine, nella seconda parte del lavoro.



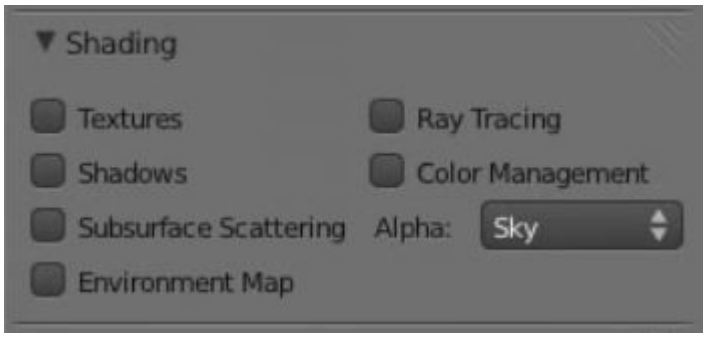
*Screenshot del primo rendering*

Il secondo rendering, relativo al canale Alpha, avviene effettuando di nuovo il rendering dell'animazione ma collegando, nel Node Editor del Compositing, il canale Alpha di Render Layers al canale input Image del nodo Composite, in modo da salvare le informazioni Alpha nel file AVI JPEG.




*Modifiche alle connessioni dei Nodi nel Nodes Editor*

Attenzione: avviando così l'animazione dovremo aspettare tutto il rendering della scena per salvare le sole informazioni Alpha: un notevole spreco di tempo! Per velocizzare questo processo, disattiviamo tutte le voci degli effetti nella scheda Render, ossia calcolo delle Texture, delle ombre, di SSS, eccetera, poi disattiviamo gli effetti di illuminazione globale come Ambient Occlusion, Environment Lighting e Indirect Lighting, se presenti, ed infine eliminiamo tutte le fonti di luce presenti nella scena.




*Disattivare le opzioni "superflue" per il rendering*

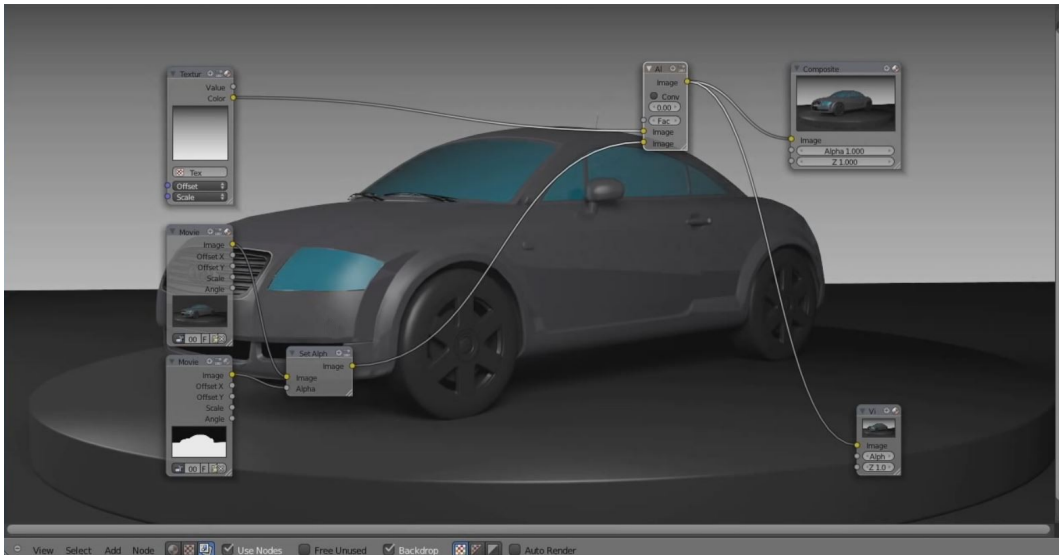
Il rendering di un fotogramma di una scena priva di illuminazione ed effetti richiede meno di un secondo, per cui il rendering di questa animazione richiederà pochissimo tempo: ecco perché vi ho detto che non avremo tempi di elaborazione raddoppiati.
L'immagine seguente mostra uno screenshot del risultato del rendering del solo canale Alpha della scena: auto e pedana sono opache, il resto è trasparente.

*Screenshot del secondo rendering*

Ottenuti questi due filmati AVI, passiamo alla seconda parte del lavoro: il Compositing.

Per questa fase sto utilizzando il Nodes Editor di Blender ma chiaramente potete utilizzare il programma di Compositing che preferite: alla fin fine, si tratta di un'operazione di Set Alpha e di un Alpha Over.

Apriamo una nuova scena di Blender, cancelliamo tutti gli elementi presenti di default (infatti non ci servono e anzi rischierebbero di farci perdere tempo inutilmente con il rendering) e passiamo subito all'Editor dei Nodi della scena.

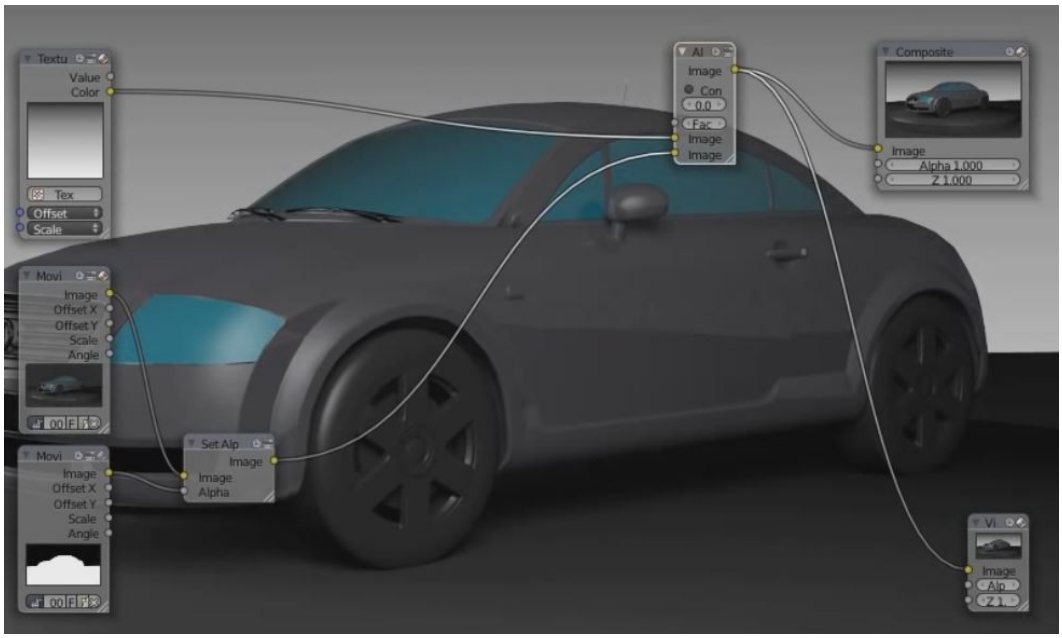



*L'editor dei Nodi in una scena vuota*

Dobbiamo utilizzare un'immagine o una Texture come sfondo per l'animazione della pedana e dell'auto; ovviamente, nessuno ci vieta di effettuare modifiche sulle animazioni generate, ad esempio una sfocatura sulla maschera Alpha o altri effetti, prima di procedere al Compositing finale... io sto mostrando direttamente la fase di Alpha Over per concentrarmi solo su questo aspetto dell'elaborazione.

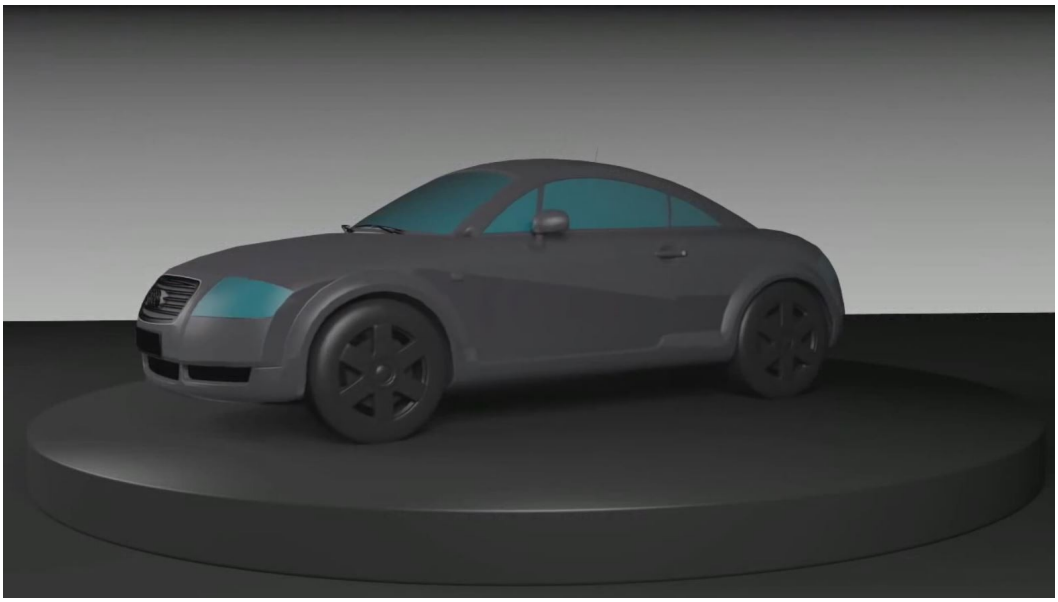
È sufficiente inserire nell'Editor un nodo Set Alpha per ottenere, dai due nodi Input Movie contenenti i videoclip RGB e Alpha, un filmato rappresentante l'animazione dotata, questa volta, di informazioni sul canale della trasparenza, in modo da passare il tutto ad un nodo Alpha Over per

applicare tale filmato sul nuovo sfondo (in questo caso, una Texture procedurale definita all'interno della scena, giusto per fare un esempio), passando poi il tutto a Composite e avviando l'animazione (dopo aver impostato formato e percorso dell'Output in Render, ovviamente).




*Lo schema dei Nodi*

Anche in questo caso, l'attesa sarà breve: la scena è vuota, Blender deve effettuare solo la fase di Compositing e inserire il risultato nel file di output stabilito.
Qui di seguito, uno screenshot del risultato finale.



*Screenshot del risultato finale*

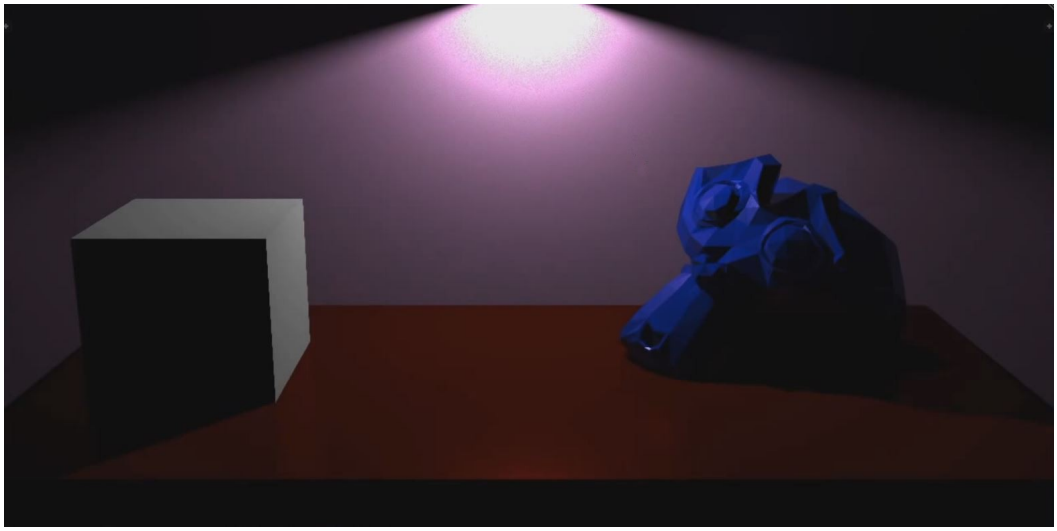
*  *  *

# Cosa sono i file IES e le IES Lights; esempi con Yafaray

In questo brevissimo tutorial parleremo di un argomento di notevole importanza nel campo dell'illuminazione, soprattutto per i rendering fotorealistici: le luci IES.

Prima di vedere un'applicazione pratica (in questo caso con Blender 2.6 e Yafaray 0.1.2 beta, in quanto la funzione IES Light non è ancora presente in Blender Internal e in Cycles), parliamo delle luci IES: IES è il nome di una società *no-profit* che mette a disposizione informazioni e conoscenze sulle tecniche di illuminazione e sulle fonti di illuminazione.

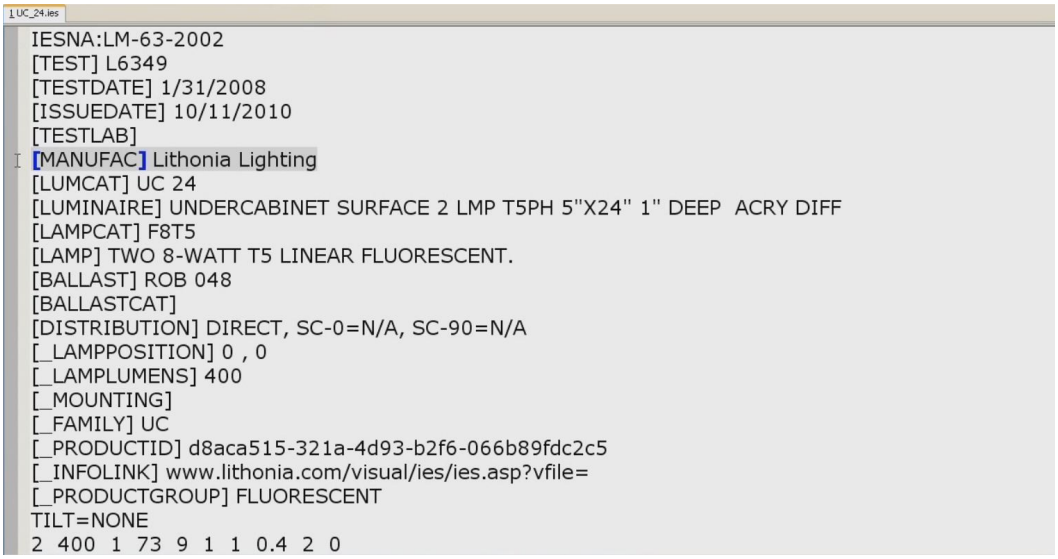
Questo è il caso che ci interessa, ossia l'utilizzo di file IES che contengono informazioni su come si diffondono i raggi luminosi di una fonte di luce specifica, ad esempio un particolare modello di lampadina. Avrete infatti notato che una fonte di luce Omni o Spot (come quella utilizzata in Yafaray per realizzare l'immagine seguente, proiettando i raggi luminosi su un piano utilizzato come parete di sfondo) proiettano un alone omogeneo, poco realistico, mentre le vere fonti di illuminazione hanno uno schema di distribuzione dei raggi molto diverso.



*Esempio di rendering con Spot Light in Yafaray*

Un file IES contiene proprio le informazioni sullo schema di illuminazione prodotto da una particolare fonte di luce; per "particolare" intendo proprio associato alla marca modello o al modello della lampadina, all'intensità in Watt, eccetera!

Su Internet trovate molti siti con archivi di file IES, probabilmente il più ricco è *Lithonia*, *www.lithonia.com* , alla voce *Risorse – Photometrics*; alla fin fine, un file IES è un file ASCII in chiaro, infatti ne sto visualizzando uno con un Text Editor in questo momento (immagine seguente).



*Screenshot di un file IES*

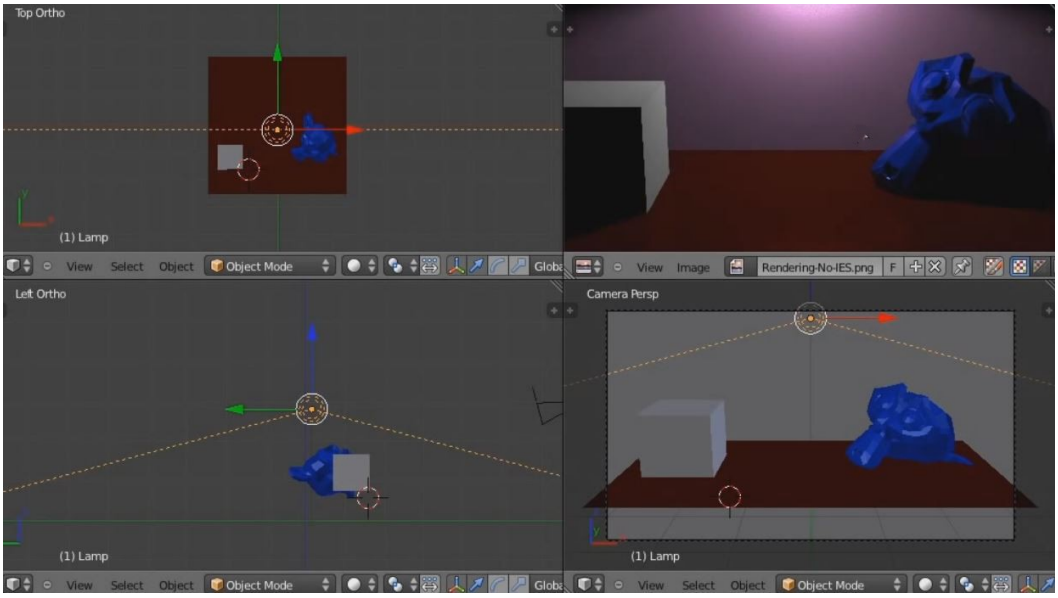
Esistono, comunque, anche dei Viewer che consentono di visualizzare tali informazioni su una mappa 2D, che rappresenta la diffusione dei raggi a 360° e mostra altre informazioni come il modello della lampadina, la potenza, ecc...

Certe applicazioni consentono di creare un proprio database di file IES in modo da cercare quello che ci interessa per la nostra scena in base a filtri e maschere di ricerca.

La cosa più interessante, quindi, è che questi file possono essere utilizzati con tutti i motori di rendering che supportano la modalità IES: potete creare la vostra libreria di file IES e utilizzarli, senza modifiche, con Yafaray, Indigo, 3D Studio MAX con V-Ray, eccetera.

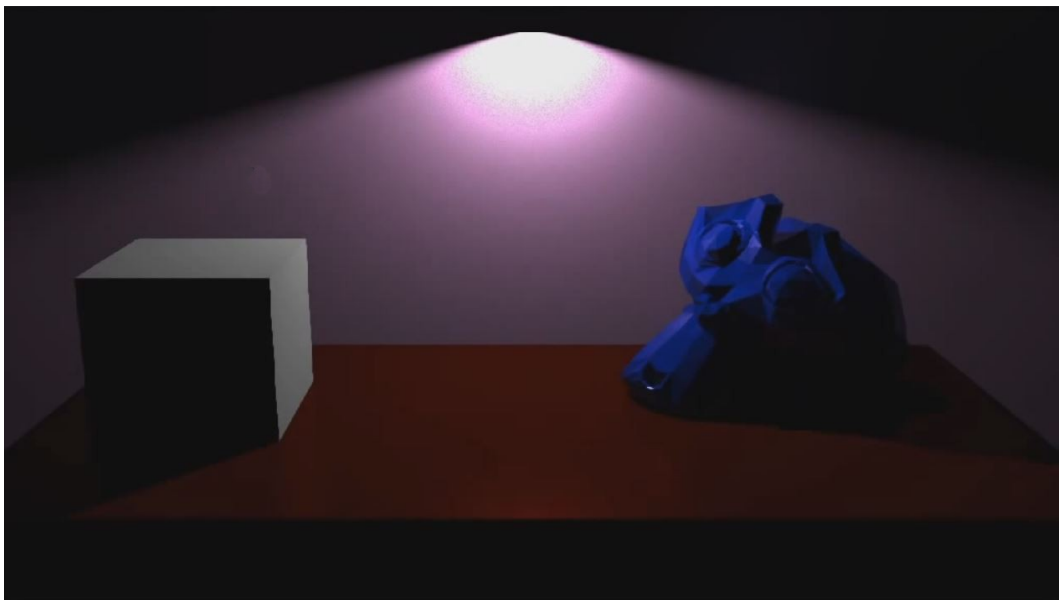
È chiaro che, se dobbiamo realizzare un rendering fotorealistico, le informazioni IES delle lampadine da inserire nella scena possono avere un ruolo importante nella resa finale della stessa; vediamo quindi come impostarle, nel nostro particolare caso, con Blender 2.6 e Yafaray 0.1.2, ancora in versione beta.

Nella scena visibile nell'immagine seguente abbiamo un piano, che rappresenta una parete (dotata di texture Clouds con dimensione 0 e colore magenta), altri oggetti che rappresentano degli ostacoli e una fonte di luce Spot Light classica. Il motore di rendering è Yafaray, modalità di illuminazione Direct Lighiting con caustiche attivate.



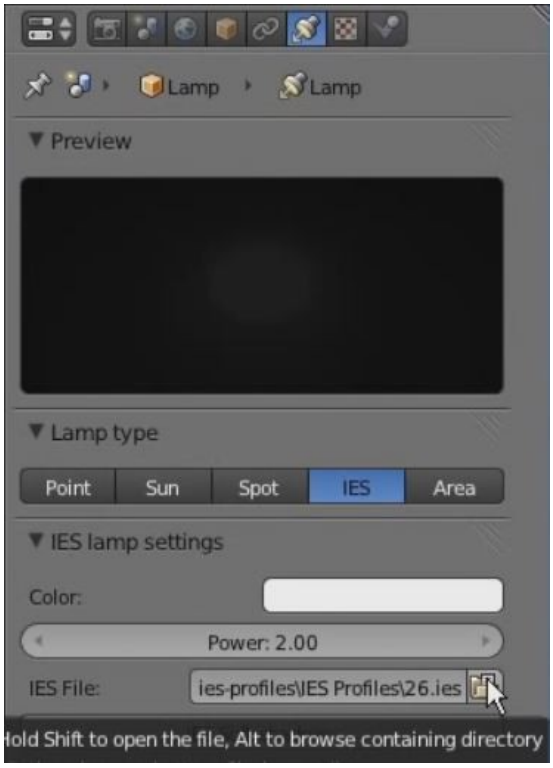

*La scena utilizzata per il nostro esempio*

Vediamo quindi il rendering di questa scena (nella pagina seguente): l'alone prodotto dall'illuminazione è omogeneo, non c'è illuminazione delle zone che si trovano dietro la sorgente luminosa.



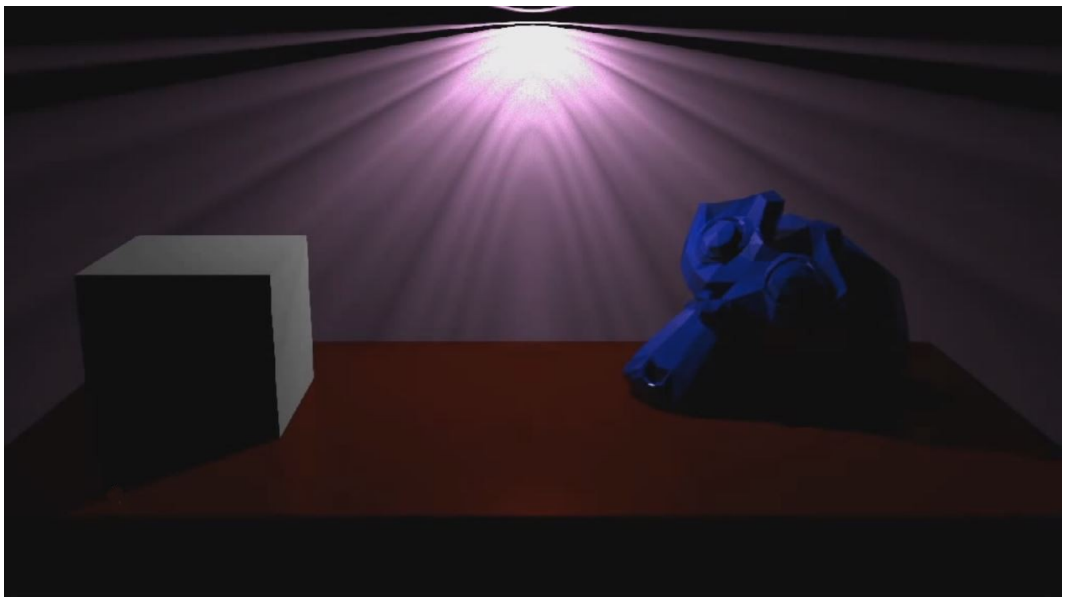
*Il rendering ottenuto senza informazioni IES*

Adesso cambiamo il tipo di illuminazione nella sezione Object Data della Lamp in IES e carichiamo il file IES da utilizzare nella sezione IES File.




*Impostare il file IES per la Lamp*

Renderizziamo nuovamente la scena: la differenza salta subito agli occhi.



*Il rendering ottenuto con la fonte di luce IES*

Il profilo utilizzato per questo primo esempio è piuttosto particolare, l'ho usato proprio per evidenziare le differenze nella resa rispetto alle semplici Spot Lights ma è alquanto eccessivo; vediamo quindi un altroue esempi, ottenuti con la stessa scena ma profili IES differenti.

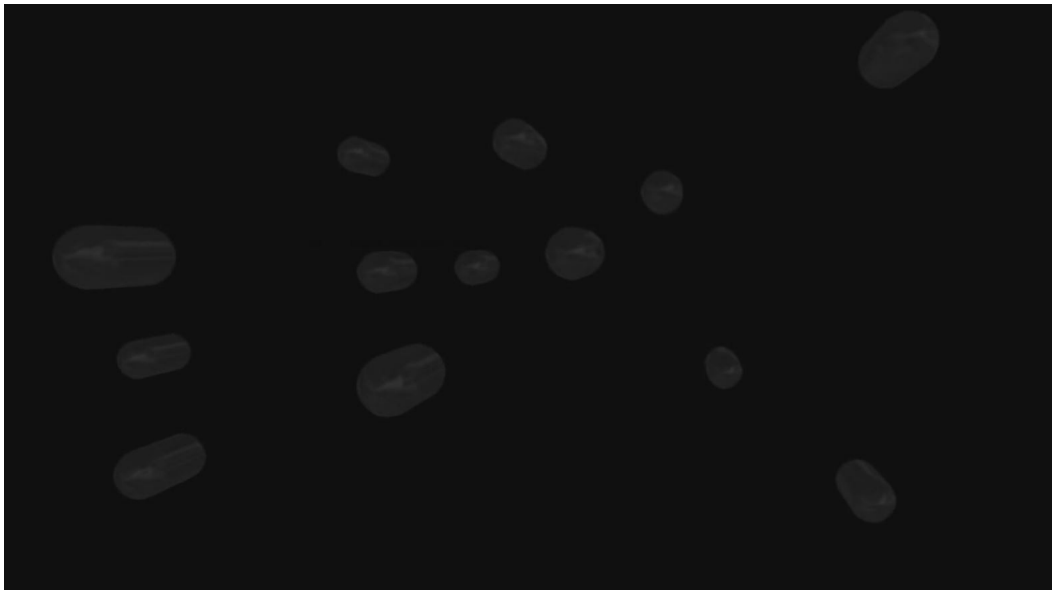
*Altro esempio di rendering, con un file IES diverso*

Questa è solo una breve introduzione alle informazioni dei file IES ma mi auguro che quanto appreso qui possa comunque servirvi per realizzare rendering sempre più interessanti e realistici.

*  *  *

# Animazione proiettili “stile Matrix” per Davidekyo

Quelle visibili qui di seguito sono immagine tratte da animazioni che ho realizzato per *Davidekyo* (che trovate su Youtube o Facebook), che mi ha proposto una piccola collaborazione per aggiungere qualche effetto ai suoi video, pubblicati sui suoi account Youtube.

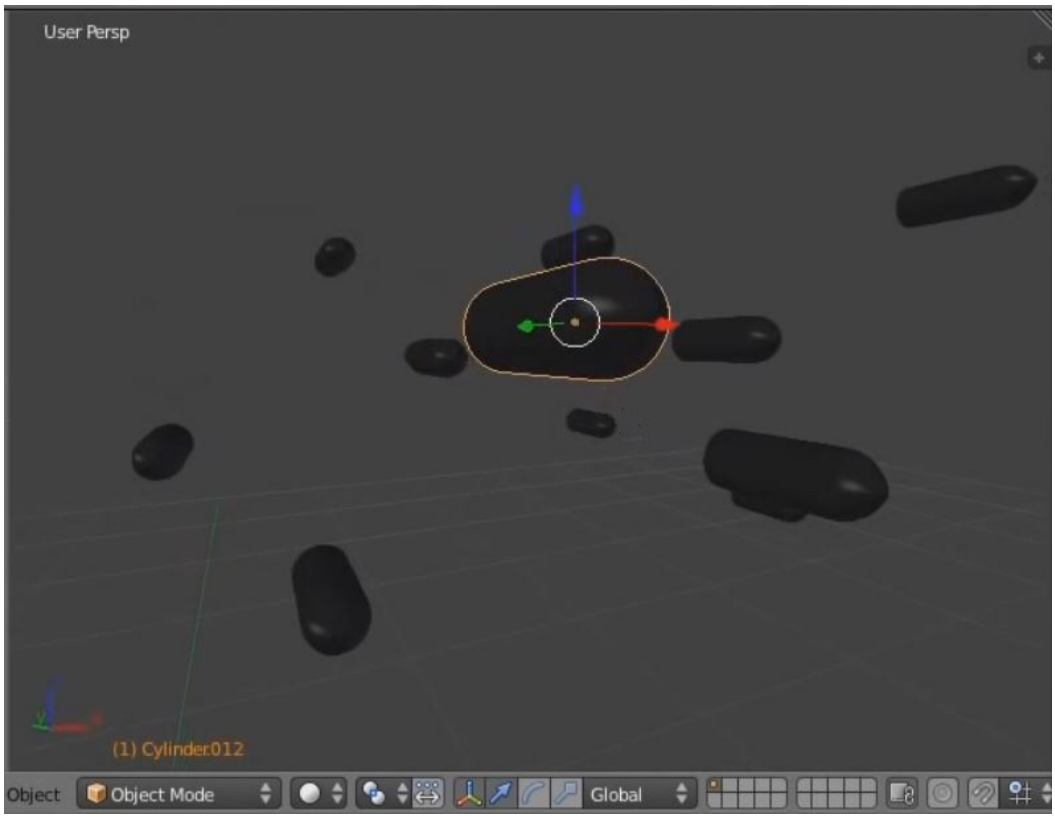


*Screenshot dell'animazione dei proiettili stile Matrix*

Si tratta di proiettili in volo, in un'animazione stile *Matrix,* ripresi da più telecamere virtuali e da aggiungere al filmato reale in post-produzione; Davide mi ha chiesto di realizzare solo l'animazione dei proiettili, senza effetti di Motion Blurring o altri effetti speciali: al Color Mapping, agli effetti aggiuntivi e al Compositing finale ci penserà lui.

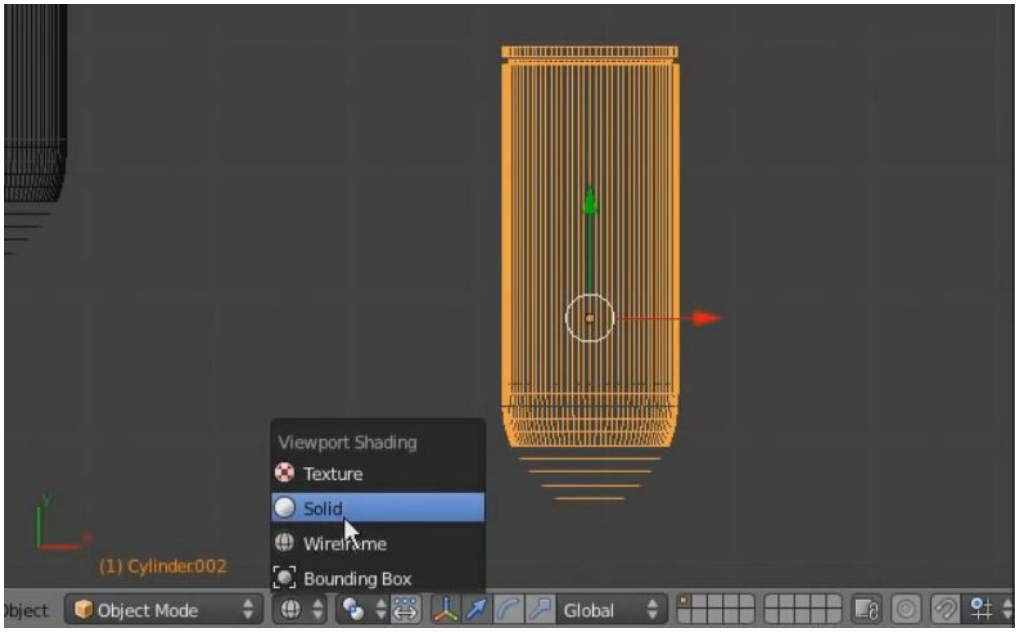
Vediamo quindi un piccolo “dietro le quinte” delle animazioni appena mostrate, realizzate con Blender 3D 2.61.

Iniziamo ovviamente con la modellazione dei proiettili: piuttosto che utilizzare una Bezier Curve per definire il profilo e realizzare in seguito il solido di rotazione con Spin, ho inserito una mesh cilindro nella scena, con un buon numero di lati, modellandola mediante estrusioni e scaling (a volte su assi vincolati), ricorrendo spesso e volentieri a Ring Cut per aggiungere "fette" da scalare o spostare senza complicare inutilmente la mesh con Subdivide.



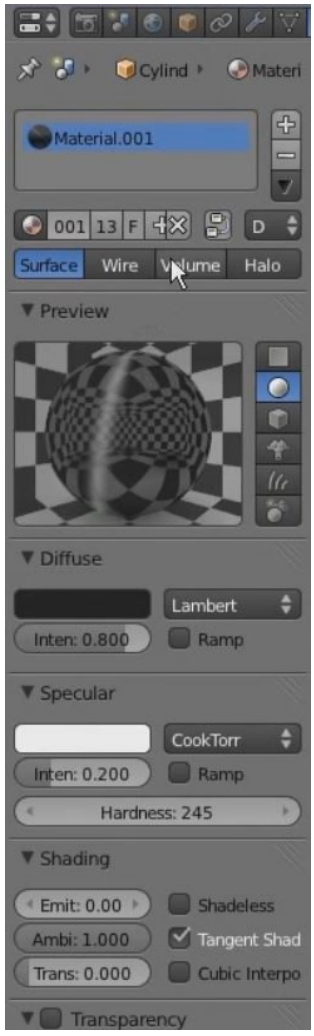
*Le mesh dei proiettili*

Alla fine ho applicato lo Shading Smooth, nella Tool Shelf, e qui c'è un piccola nota a margine: come specificato nel mio ebook sui Materials in Blender 3D 2.5, valido chiaramente anche per Blender 2.6 (tranne che per Cycles), lo Shading Smooth applicato alle strutture cilindriche senza alcuni accorgimenti può restituire risultati errati; conviene, in particolare, selezionare le facce che compongono la base del proiettile (operazione facile con una vista ortogonale dall'alto) e applicare lì Shading Flat; conviene, infine, aggiungere varie "fette" sul corpo del cilindro, aumentandone la risoluzione in termini di facce, mediante Ring Cut.




*Shading Smooth sui proiettili: accorgimenti*

Passiamo al Material: si tratta di un materiale semplice, con colore grigio molto scuro (ad esempio 0.01 per tutti e tre i canali RGB), con Tangent Shading per i riflessi speculari e un effetto Mirror con intensità 0.25.



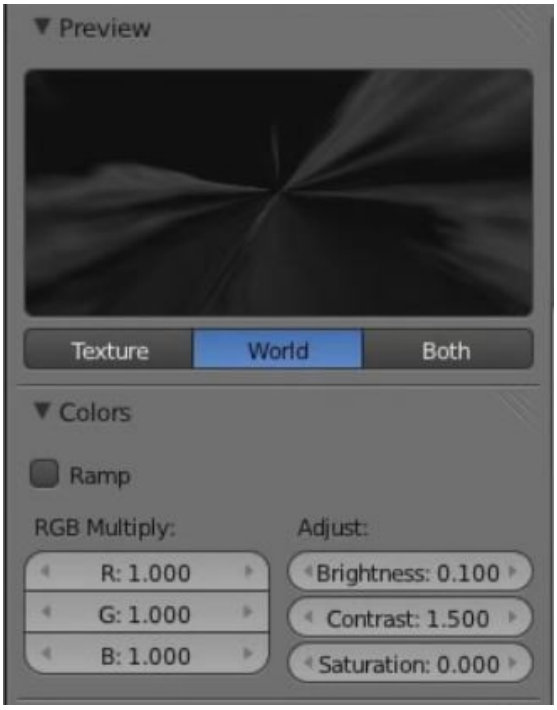

*I Materials dei proiettili*

Non sono presenti Textures per variare l'aspetto del Material e non ci sono fonti di luce nella scena: per la resa, utilizzeremo un'immagine mappata in modalità Ang Map su World; spostiamoci, quindi, nella scheda World, all'interno della Properties Window.

In questa scheda attiviamo Ambient Occlusion con le impostazioni di default (modalità additiva con fattore 1) e soprattutto Environment Lighting in modalità Sky Texture.

Andiamo nella scheda Texture per World ed aggiungiamo una Texture immagine.

Mentre lavoravo avevo a portata di mano solo la fotografia di un bel cielo azzurro con un paio di nuvole: una scelta poco adatta per illuminare dei proiettili dalla superficie riflettente in un interno scuro... o no? Beh, con qualche piccolo accorgimento, possiamo riciclare anche una foto del genere per il nostro progetto: apriamo il file immagine come Texture, quindi andiamo nella sezione Colors e portiamo la saturazione a 0.0, in modo da rendere l'immagine in scala di grigi.

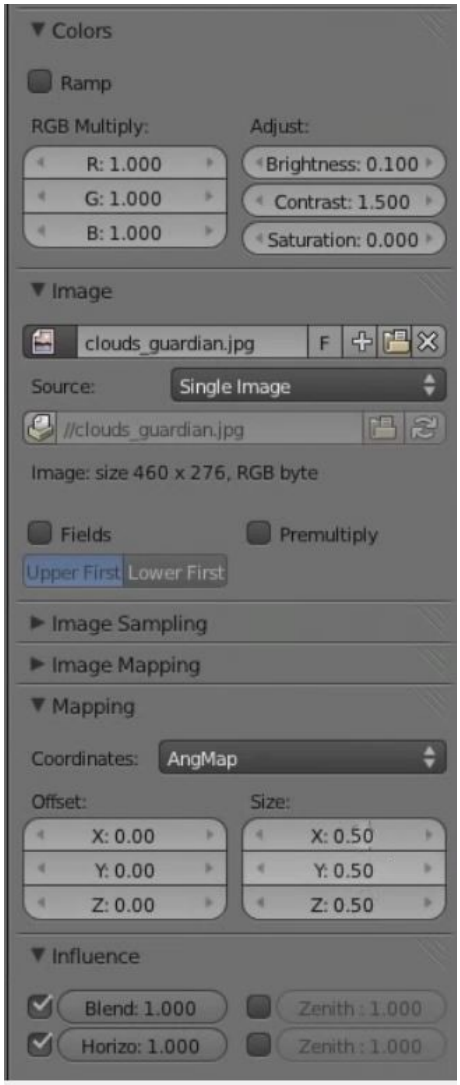



*La Texture, resa in scala di grigi, per World*

*La scheda della texture World*

Va già meglio ma si riconoscono ancora le nuvole, per cui portiamo la luminosità a 0.1, il contrasto a 1.5, cambiamo la modalità di mappatura in AngMap ed allarghiamo la Texture, in modo da rendere più diffuse le striature e le macchie che verranno proiettate sugli oggetti (in questo caso, ho impostato Size 0.5).

In Influence selezioniamo Horizon, lasciando selezionato Blend.



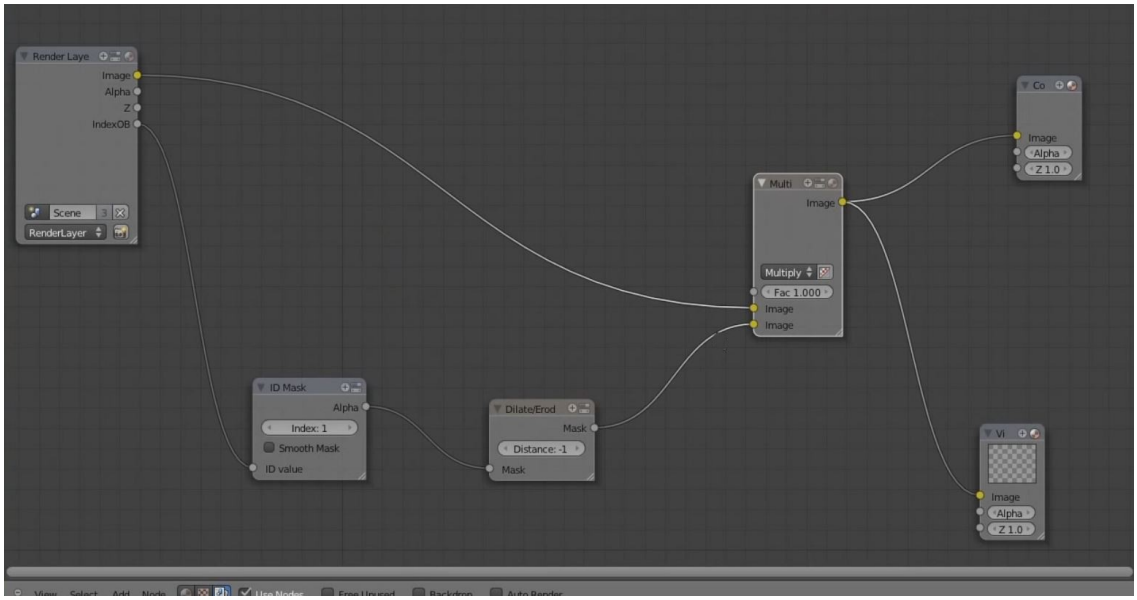

*Ulteriori modifiche alla Texture*

Un rendering lanciato in questo momento mostrerebbe i proiettili sullo sfondo dell'immagine stirata, proiettata in modalità AngMap; per passare a Davide solo le sagome dei proiettili su sfondo nero ho sistemato un paio di Nodi in Compositing.

Qui non parlerò in dettaglio dei singoli nodi e del loro utilizzo, argomento trattato invece nel mio ebook sul *Compositing dei Nodes in Blender 3D 2.5*, ma mi limiterò a descrivere il flusso dell'esecuzione.

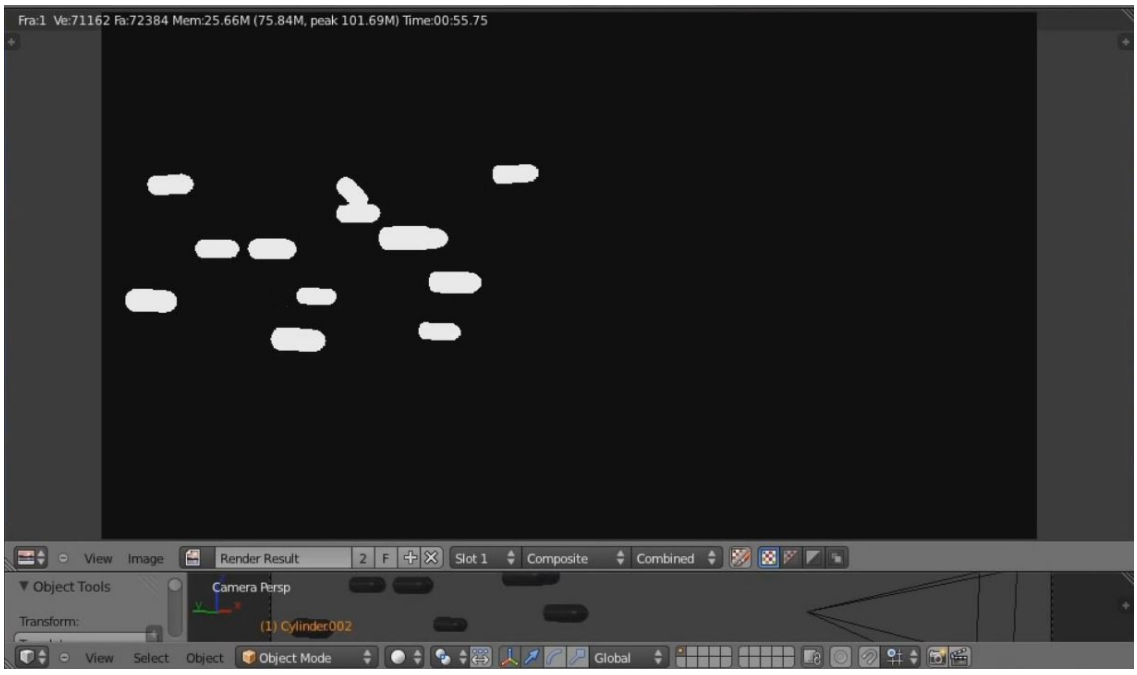
I proiettili hanno un ID Mask 1, per cui possiamo prelevare le sagome Alpha da Render Layers mediante ID Mask, restringere un po' la maschera con Dilate/Erode per togliere riflessi o parti di sfondo e moltiplicare tale maschera per l'immagine originale, in modo da ottenere i proiettili su sfondo nero.

Se l'output è un file o una sequenza di file con canale Alpha, allora possiamo passare l'output di Mix Multiply e di Dilate/Erode rispettivamente ai canali Image e Alpha di Composite; altrimenti (se ad esempio l'output è un file AVI) dobbiamo renderizzare prima l'immagine in arrivo da Mix Multiply, poi, su un file a parte, come maschera Alpha, quella in arrivo da Dilate/Erode (tecnica descritta in un tutorial precedente, dal titolo "Animazioni con Alpha + Alpha Over", realizzato su richiesta per Maxxximus e presente in questo volume).




*Nodes Editor*

In questo secondo caso, per effettuare il rendering del solo canale Alpha consiglio di disattivare tutte le voci di Shading nella scheda Render, cancellare le fonti di luce eventualmente presenti e disattivare gli strumenti di illuminazione globale in World: in questo modo, il rendering di un singolo frame impiegherà pochi decimi di secondo (in fin dei conti, a noi interessa la maschera Alpha, non i canali RGB).




*Rendering della maschera Alpha*

*  *  *

# Le basi del Game Engine, il motore di gioco di Blender 2.6 – 1

Questa è la prima puntata di un minicorso introduttivo sulle basi del Game Engine, il motore di gioco di Blender 3D.

Questo corso è rivolto a chi conosce già le basi di Blender Internal ed anzi ha dimestichezza con la modellazione, i materiali e l'illuminazione, per cui non mi soffermerò a descrivere la modellazione di un oggetto o di un ambiente o a spiegare la differenza tra una lampada Spot con Buffer Shadow e una Point Light, dando per scontati argomenti del genere; se non conoscete bene questi elementi, date prima un'occhiata al videocorso di 80 puntate sulle basi di Blender 3D (trascritto nel pdf "Blender 3D tutorials – Volume 1").

In questa prima puntata non parleremo dell'interazione e della logica del motore di gioco; vedremo, invece, come attivare il Game Engine e come risolvere i primi dubbi che vengono in mente a tutti quando premono il fatidico tasto P e non vedono il colore dello sfondo, le ombre o le Textures così come le hanno impostate nella 3D View o come le vedono in fase di rendering.

La prima cosa da fare è, ovviamente, avviare Blender 3D, in questo caso versione 2.61.
Siamo abituati a modellare, illuminare e texturizzare gli oggetti con il motore di rendering di default, Blender Internal, ma per sviluppare un'applicazione 3D, che prevede quindi la resa grafica in tempo reale (generalmente, a 25 frames al secondo o più) e l'interazione con l'utente, dobbiamo passare ad un altro ambiente di lavoro, detto appunto Game Engine, motore di gioco (anche se quel "gioco" è fuorviante perché col GE potete sviluppare anche applicazioni per lavoro, visite virtuali in musei, console di comando che interagiscono con dispositivi hardware mediante rete o altri dispositivi, ed altro ancora).

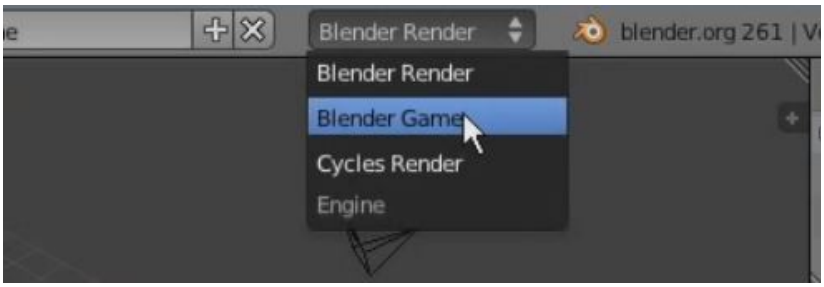
La resa grafica in tempo reale giustifica, in particolare, il passaggio ad un altro ambiente di lavoro: quando si imposta una scena per un rendering o un'animazione, infatti, si ha la possibilità di

premere F12, alzarsi e fare altre cose (a volte anche dormire tutta la notte) mentre il pc renderizza un frame, che a seconda dei casi può richiedere diversi minuti o ore (dipende dal grado di fotorealismo, dagli effetti implementati, dalla risoluzione, dal Compositing, eccetera); alla fine, terminato il rendering, salviamo l'immagine o l'animazione e abbiamo finito.

Chiaramente, non possiamo permetterci questi tempi con un videogioco, dove il frame rate, ossia il numero di fotogrammi da visualizzare (leggi: rendering della scena) dev'essere di almeno (almeno!) 24-25 fotogrammi al secondo.

Si arriva a queste performance con un mix di trucchetti per gestire meglio la resa; ad esempio, ombre non in tempo reale, assenza di effetti di rifrazione e mirroring, simulazioni fisiche precalcolate e hardware potente.

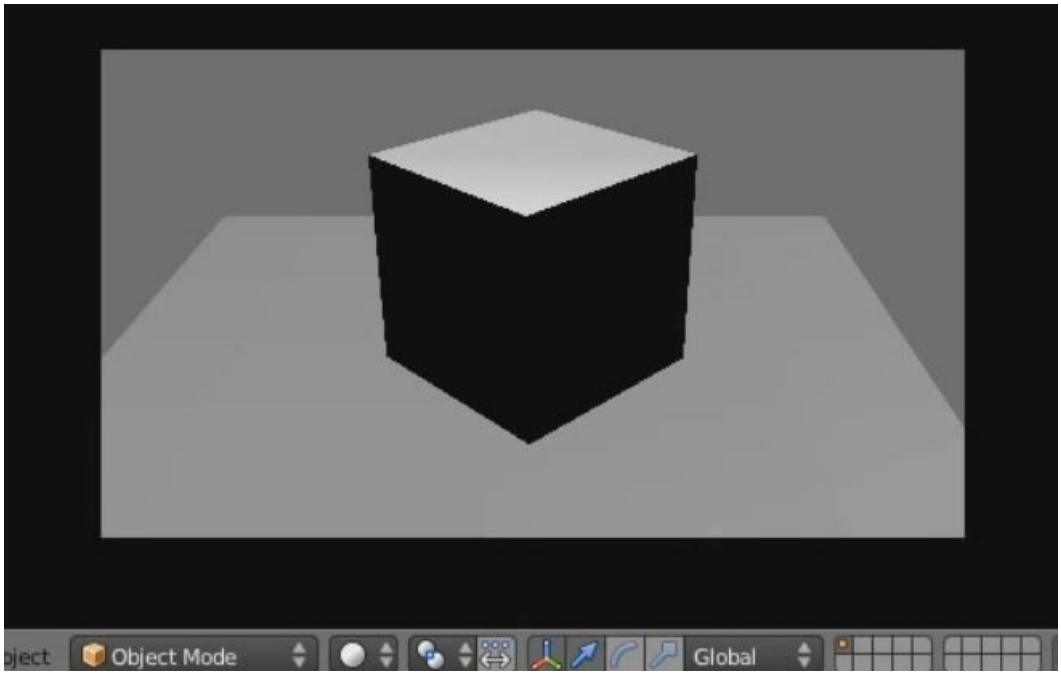
Per passare all'ambiente del Game Engine, il nostro ambiente di lavoro per tutto il minicorso, selezioniamo “Blender Game” dal selettore “Engine” (la cui voce di default è “Blender Render”, il motore di rendering interno di default di Blender 3D), nella finestra Info, in alto.




*Selezione del motore Blender Game*

Ad una prima occhiata, sono cambiate le voci nella scheda Render, nella Properties Window visibile a destra nella schermata e tra pochissimo ne parleremo, comunque notiamo che nella 3D View non è successo niente: possiamo ancora aggiungere, cancellare, spostare e, in generale, trasformare, sia in Object che in Edit Mode, gli oggetti nella scena; siamo, cioè, in un ambiente di editing (modellazione).

Per vedere l'esecuzione dell'applicazione 3D posizioniamo il cursore del mouse su una finestra 3D View (magari quella con la vista telecamera) e premiamo P: ecco, adesso siamo in modalità esecuzione, in modalità gioco; P, infatti, sta per Play.




*Modalità esecuzione (dopo aver premuto P in una 3D View)*

Non succede nulla perché di base gli oggetti non sono oggetti dinamici, soggetti cioè a forze (come quella di gravità), per cui non cadono e non si spostano, né abbiamo stabilito come interagire con l'ambiente, quindi è inutile premere tasti della tastiera o fare click col mouse (tra l'altro, non vediamo il cursore del mouse), né ci sono animazioni... niente di niente! Un po' deludente, come prima esecuzione dell'applicazione!

Terminiamo quindi l'esecuzione premendo ESC, per tornare alla modalità modellazione.

Diamo un'occhiata alla scheda Render, all'interno della Properties Window: qui notiamo la presenza del pulsante "Start" al posto di "Image" e "Animation", presenti con Blender Render, proprio perché ora dobbiamo eseguire l'applicazione, non

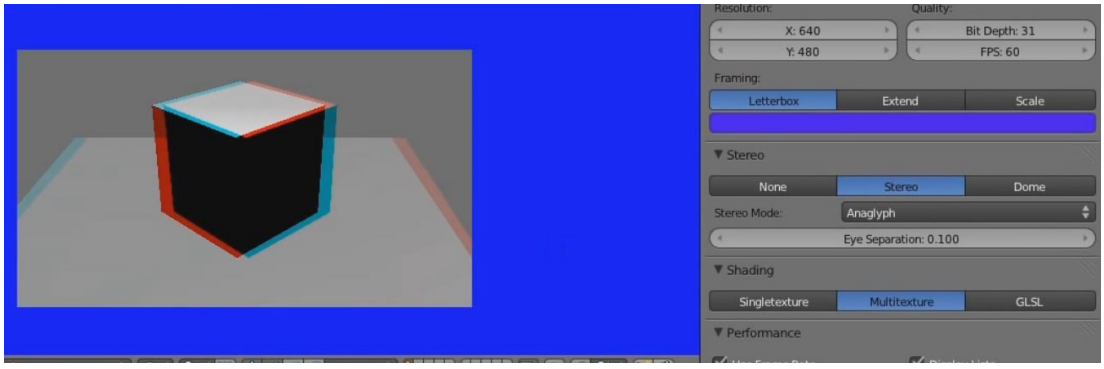
*La scheda Render, nella Properties Window*

avviare un rendering.
Il tasto P è proprio lo shortcut per questo "Start", quindi è possibile lanciare l'applicazione con P, Start oppure "Start Game Engine" nel menù "Game" (finestra Info, in alto).
La risoluzione della finestra, in Resolution X e Y, si riferisce alla grandezza della finestra del gioco quando lo eseguiamo in Blender oppure quando lo rilasceremo come applicazione a parte, o "stand-alone", infatti le applicazioni 3D realizzate col Game Engine possono essere rilasciate come eseguibili indipendenti, senza doverle lanciare ogni volta da Blender. Parleremo di questo tra qualche puntata.

Se volete realizzare un gioco con una risoluzione prefissata, ad esempio 640x480 (quella di default), impostate i valori nei campi X, Y e lasciate "Letterbox", in modo da colorare con il colore indicato nella Color Box sotto le altre parti dello schermo (default: nero), altrimenti scegliete Extend per estendere la visualizzazione a schermo intero.

Vi consiglio di visualizzare gli effetti delle varie opzioni cambiandole e premendo P, anche perché non sempre quello che si vede nella 3D View corrisponde all'esecuzione dell'applicazione in tempo reale (esempio immediato: il bordo colorato di Letterbox non è visibile nella 3D View).

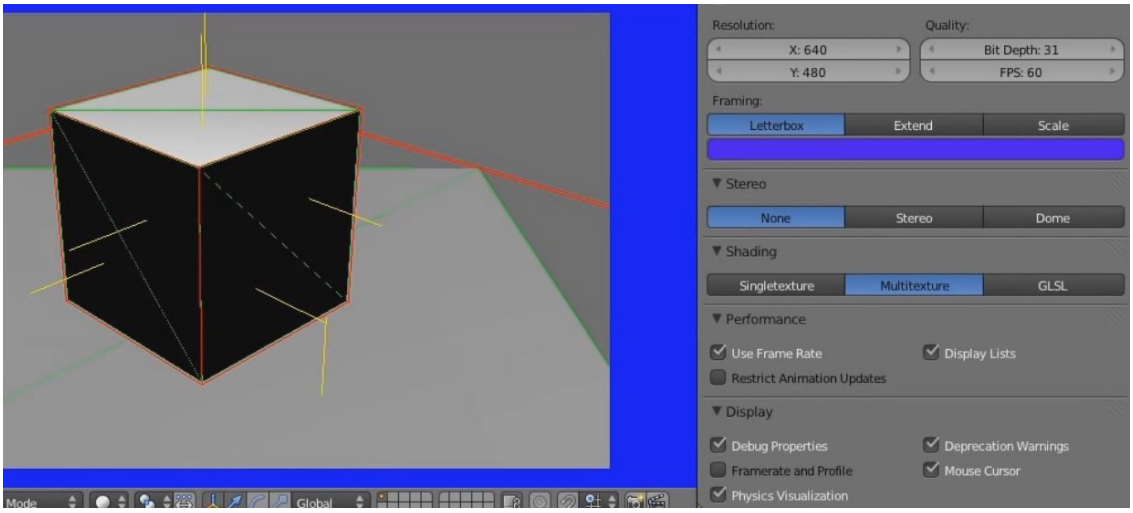
La sezione "stereo" fa riferimento alla possibilità di realizzare applicazioni da visualizzare in 3D, ossia alla visione stereoscopica, ad esempio mediante i famosi occhialini con le lenti colorate (tipicamente, rosso e ciano), con la modalità nota come "Anaglifo", infatti selezionando "Stereo", lasciando "Anaglyph" e premendo P vedremo gli oggetti nella scena con il doppio bordo, rosso e ciano, e potremo vederli davvero in 3D indossando quegli occhialini.



*Modalità Stereo Anaglyph*

Il Game Engine mette a disposizione anche altre modalità per la visualizzazione 3D ma in questa sede non ne parleremo, per cui torniamo a None, nessuna opzione stereo.

Saltiamo per un secondo la sezione Shading e diamo un'occhiata a Display, in basso nella scheda: qui possiamo attivare la visualizzazione del cursore del mouse a runtime, selezionando appunto "Mouse Cursor", e quella di alcune informazioni come le proprietà di sistema (Debug Properties) o fisiche degli oggetti (Physics Visualization, per visualizzare ad esempio informazioni su collisioni, deformazioni o rimbalzi): basta selezionare le voci e premere P per farsi un'idea del loro utilizzo.



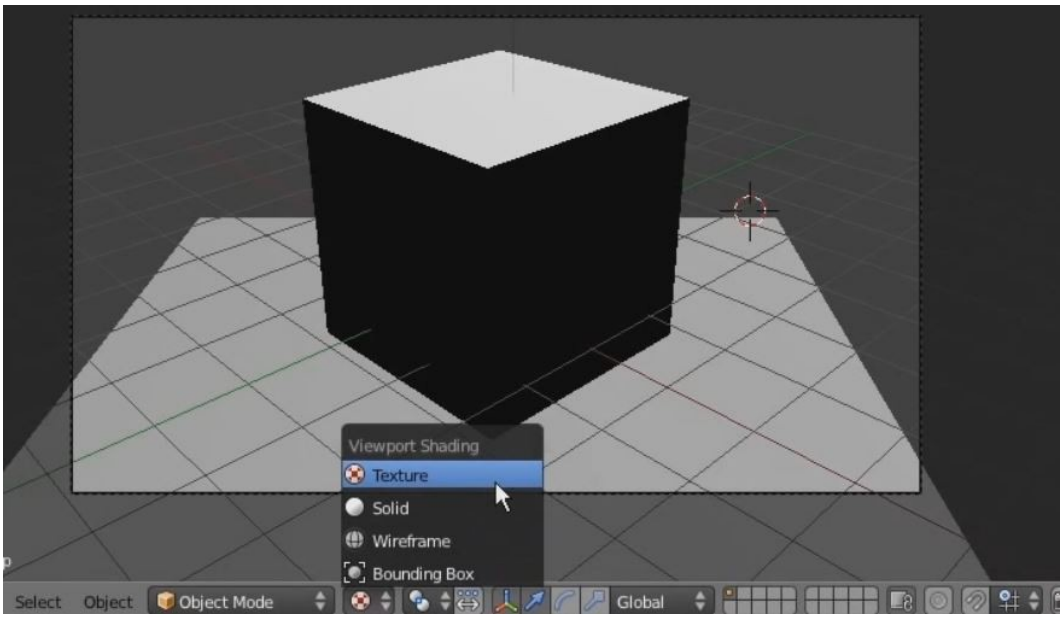
*Sezione Display*

Consiglio di non utilizzare Physics e di attivare, possibilmente, Mouse e Debug, per analizzare le proprietà del sistema o individuare il cursore del mouse quando si fanno dei test, ad esempio lo spostamento della visuale muovendo il mouse (opzione che di default non è disponibile) o altri esperimenti, prima di rilasciare l'applicazione come file eseguibile a parte.

*  *  *

Eccoci quindi che terrà banco per questa puntata e che avrà ripercussioni in tutte le altre: la scelta dell'ombreggiatore (shader) da utilizzare!

Prima di procedere con gli shader, una piccola nota: per visualizzare correttamente i colori degli oggetti, le ombre in tempo reale o le Texture, soprattutto con lo shader GLSL (che introdurremo a breve) dobbiamo sempre passare alla modalità di visualizzazione Textured nella 3D View, come visibile nell'immagine seguente.




*Modalità Textured nella 3D View*

Lo shader è l'algoritmo utilizzato da Blender per definire la colorazione e l'ombreggiatura degli oggetti in fase di visualizzazione degli stessi, ossia in fase di rendering o, come nel nostro caso, di esecuzione del gioco, fotogramma per fotogramma.

Il Game Engine di Blender 3D 2.61 mette a disposizione tre shaders, selezionabili (in maniera esclusiva, ovviamente) mediante tre pulsanti nella scheda Render, all'interno della Properties Window: SingleTexture, MultiTexture e GLSL.

È importante conoscere le loro caratteristiche principali prima di procedere con la definizione di ambienti, luci, materiali e textures nella nostra applicazione.

SingleTexture è il tipo base: la colorazione avviene per vertice, il che significa che mesh con pochi vertici potrebbero presentare degli errori; inoltre, è supportata una sola Texture per Material, che

deve essere mappata con mappatura UV e non sono disponibili le ombre in tempo reale.
Per effettuare la mappatura UV dobbiamo quindi aprire l'immagine utilizzata come Texture in una finestra UV Editor delle immagini e fare l'unwrapping come in Blender Render, impostando poi la mappatura UV Coordinates e il relativo Layer nella scheda Mapping della Texture.

> *NOTA importante: le Textures procedurali NON vengono visualizzate in tempo reale, per cui bisogna fare il Bake di tali Textures, in modo da ottenerle come immagini "scucite" da salvare a parte, con il Blender Internal, per poi caricarle come Textures immagine UV-mappate nel Game Engine; mostrerò come fare quanto detto nella prossima puntata.*

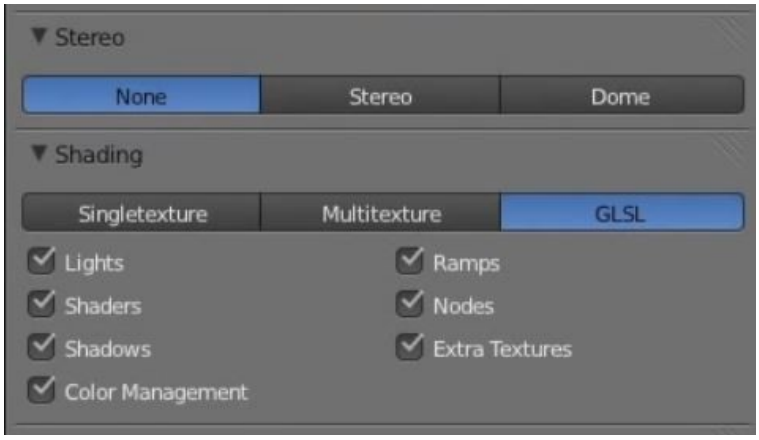
Come suggerisce il nome, Single Texture supporta una sola Texture, per cui se vogliamo aggiungere più Textures da miscelare in vari modi (ad esempio un'immagine ottenuta mediante Bake dell'Ambient Occlusion del modello da moltiplicare all'immagine utilizzata per il colore diffuso dello stesso, in modo da scurire le zone in prossimità delle occlusioni) dobbiamo passare per forza all'ombreggiatore MultiTexture, che ci consente di associare appunto più Textures ad un Material.

MultiTexture supporta due o tre Textures che devono essere comunque UV-Mapped sul modello e possono essere miscelate in modalità Add, Multiply, Subtract o Mix; in quest'ultimo caso, bisogna agire sullo slider Color Diffuse per definire in che misura la Texture corrente dovrà sostituirsi alle altre, ossia quanto dovrà "pesare" in fase di miscelazione.

Il terzo ombreggiatore, GLSL (OpenGL Shading Language), è il più interessante: si basa sulle API OpenGL, supporta l'accelerazione hardware (consentita da praticamente tutte le schede video attualmente in circolazione), implementando le ombre in tempo reale (a patto di utilizzare le Spot Lights con ombre Buffer Shadow, come vedremo) e consentendo l'utilizzo della Color Ramp, l'utilizzo di più Textures UV-mapped (come MultiTexture) e dei Nodes per il Compositing.

Tutte queste opzioni possono essere disabilitate, nella pipeline di rendering, deselezionando le relative checkbox (caselle di spunta) che appaiono nella scheda Shading, in Render nella Properties Window, quando si sceglie GLSL; questa possibilità può tornare utile quando si fanno delle prove e non è necessario implementare TUTTI gli effetti, appesantendo l'esecuzione: ne bastano alcuni.

Queste opzioni non sono disponibili con SingleTexture e MultiTexture, in quanto non supportate da tali shaders.



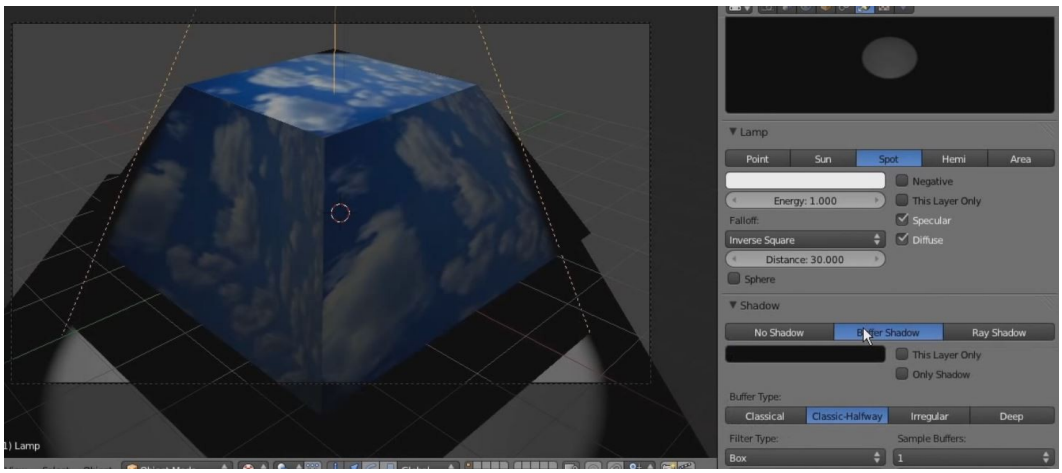

*Sezione Shading per la GLSL, nella scheda Render*

Purtroppo, neanche GLSL implementa gli effetti di RayTracing, quindi niente trasparenza Raytrace con effetti di rifrazione e Glossiness e niente riflessi Mirror, a meno di non utilizzare degli script ad hoc che consentono, sostanzialmente, di visualizzare su un oggetto "quello che l'oggetto sta vedendo"... ma di questo parleremo un'altra volta.

Per i riflessi statici o con animazioni "programmate" (dove cioè non ci sono mutamenti dovuti a interazioni dell'utente o di altri elementi, ma animazioni stabilite a priori da chi crea il gioco) è possibile ricorrere al trucchetto delle EnvMaps, descritte nella 53esima puntata del mio videocorso sulle basi di Blender 3D 2.5 e 2.6 (trascritto nell'ebook pdf "*Blender 3D tutorials – Volume 1*").

Neanche a dirlo, consiglio vivamente di lavorare con GLSL piuttosto che con SingleTexture o MultiTexture.

Come anticipato, GLSL consente la visualizzazione delle ombre in tempo reale, ma solo per le ombre proiettate da Spot Light con Buffer Shadow; cambiamo quindi la fonte di luce presente nella scena di default (o aggiungiamone una nuova) in Spot Light e, nella scheda Object Data di tale oggetto, scegliamo la modalità Buffer Shadow per le ombre.



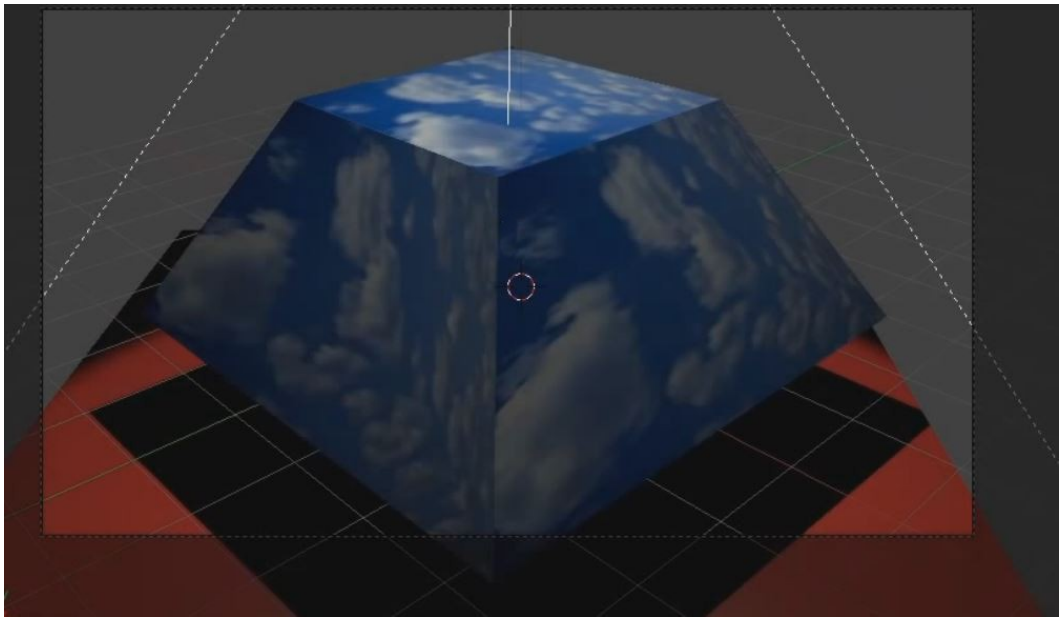
*Spot Light in modalità Buffer Shadow*

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); impostare 180° avrà l'effetto di SIMULARE una fonte di luce Area, buona ad esempio per pannelli luminosi o per gli esterni, in grado di proiettare ombre.

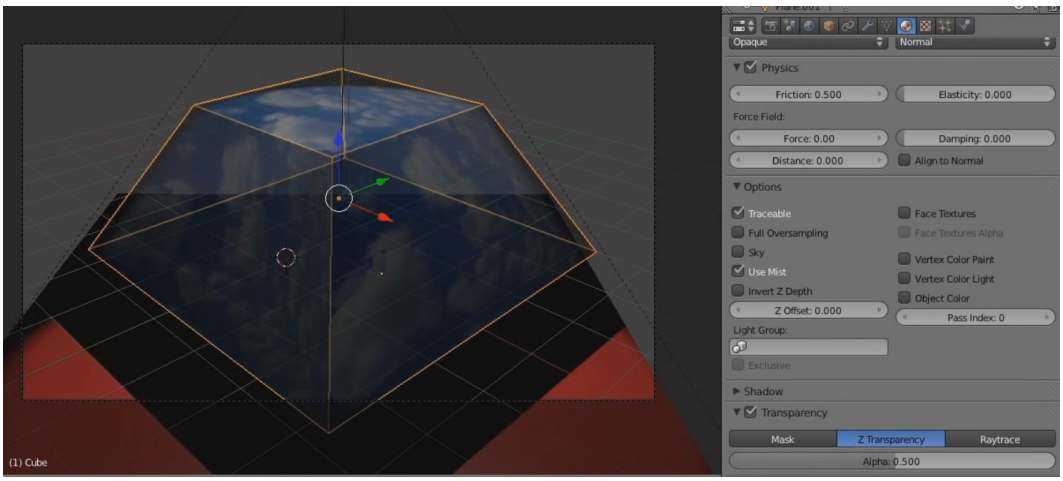
Conviene comunque non abusare delle ombre Buffer Shadow all'interno di un gioco, o perlomeno utilizzare valori modesti per le impostazioni, per evitare di avere rallentamenti durante l'esecuzione.

Possiamo vedere le ombre in tempo reale già nella 3D View, senza bisogno di lanciare il Rendering, se abbiamo GLSL selezionato e la modalità di visualizzazione Textured.



*Ombre in tempo reale nella 3D View Textured con GLSL*

Con queste condizioni (cioè GLSL e Textured) possiamo vedere anche la trasparenza degli oggetti provvisti di Material con Transparency attivato e un valore minore di 1 per Alpha, l'opacità, come visibile nell'immagine seguente.




*Transparency ed effetto in tempo reale con GLSL*

Ricordo che la trasparenza RayTrace non è supportata: se si sceglie RayTrace e si abbassa il valore di Alpha, si finisce col miscelare il colore dello sfondo, World, con il colore e le Texture della mesh; portando Alpha a 0, il colore della mesh sarà sostanzialmente il colore dello sfondo applicato in maniera shadeless, ovvero senza ombreggiature; conviene, quindi, utilizzare la trasparenza Z-Transp.

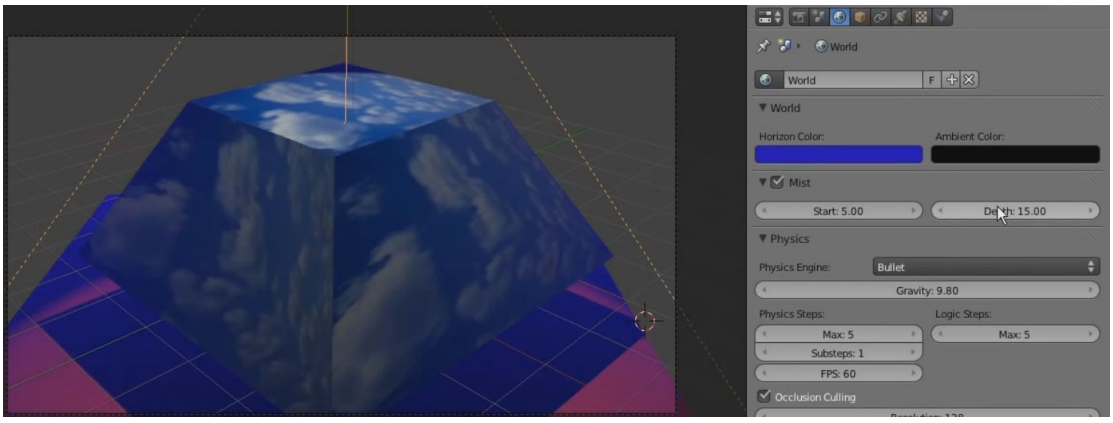
Tra i problemi che si hanno con le ombre proiettate da Spot con Buffer Shadow:

- le ombre degli oggetti semitrasparenti non vengono proiettate (nemmeno se attiviamo Receive Transparent nei Materials degli oggetti che le riceveranno);
- in generale (cioè anche con oggetti opachi) i bordi non sono sfumati e comunque non vogliamo dover inserire Spot Lights per le ombre, anche perché a volte le ombre delle nostre scene sono statiche e non muteranno in seguito all'interazione dei personaggi con l'ambiente o altri eventi, per cui il calcolo in tempo reale è uno spreco...

… beh, la soluzione c'è e consiste nel precalcolare le ombre, ottenendole come Texture mediante Baking, per poi applicare tale Texture come immagine UV-Mappata, in questo caso in modalità Multiply, per scurire gli oggetti: è lo stesso discorso fatto per Ambient Occlusion poco fa e tutto si basa su questo famoso Bake, che tratteremo nella prossima puntata.

> *Nota sul colore dello sfondo: potete impostarlo cambiando Horizon Color nella scheda World; lo sfondo non viene colorato nella 3D View in fase di Editing (neanche in modalità Textured e GLSL), ma solo a runtime.*

In World è possibile attivare anche MIST (la nebbia o foschia) per nascondere gli oggetti molto lontani con una specie di dissolvenza, utile soprattutto con scenari ampi perché la telecamera non renderizza a infinito, quindi gli oggetti posti oltre una certa distanza (da impostare con Clip End, nella scheda Object Data della telecamera) potrebbero sparire all'improvviso superata la distanza massima Clip End; ecco, per evitare questo brutto effetto, potete fissare l'inizio della nebbia con Mist Depth ad un valore leggermente inferiore di quello impostato per Clip End.




*MIST (nebbia, foschia)*

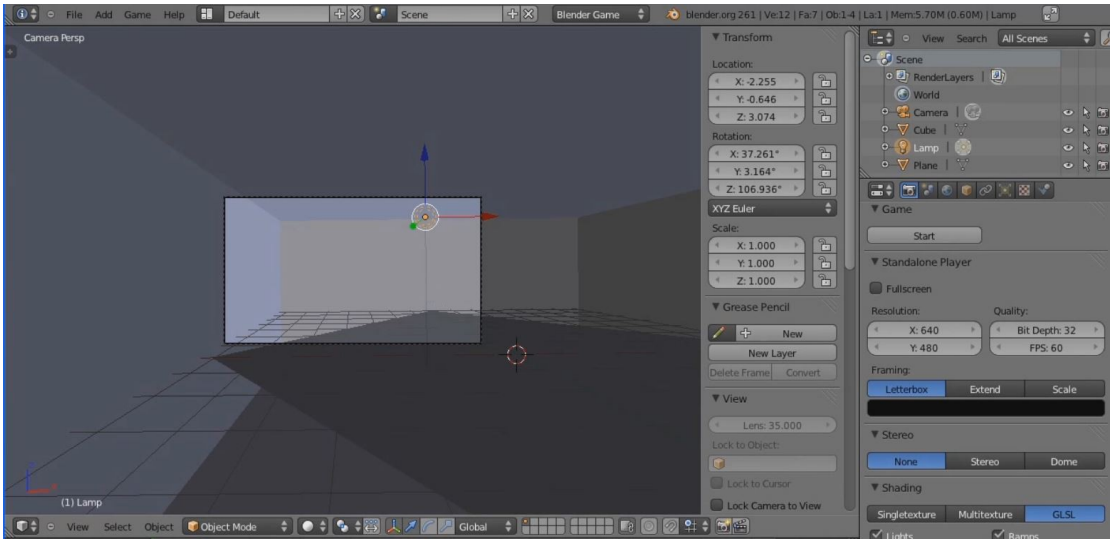
Per questa prima puntata sul Game Engine è tutto; abbiamo visto alcuni elementi di base dell'ambiente del motore di gioco e capito perché dobbiamo scegliere bene lo Shader, passare alla modalità Textured nella 3D View, utilizzare le Spot Light per le ombre in tempo reale, preparare dei modelli semplici per poter mappare poi le Textures UV con facilità ed altro ancora; nella prossima puntata parleremo del BAKE delle Textures e di come fare per inserire Textures animate a runtime, mentre nella terza parleremo degli aspetti legati all'interazione, quindi la logica del motore di gioco, le Properties di scena e oggetti e una piccola introduzione sulla fisica nel GE.

*  *  *

# Le basi del Game Engine, il motore di gioco di Blender 2.6 – 2

In questo secondo tutorial del minicorso sulle basi del Game Engine, il motore di gioco di Blender 3D, parleremo ancora dell'aspetto degli oggetti e degli ambienti, trattando in particolare due argomenti: il Bake delle Textures e degli effetti (mostrando il Bake delle ombre e la loro mappatura sugli oggetti mediante UV Mapping) e l'utilizzo di Sprites o Textures animate a runtime.
Il tutorial è stato realizzato con la versione 2.61 di Blender 3D.

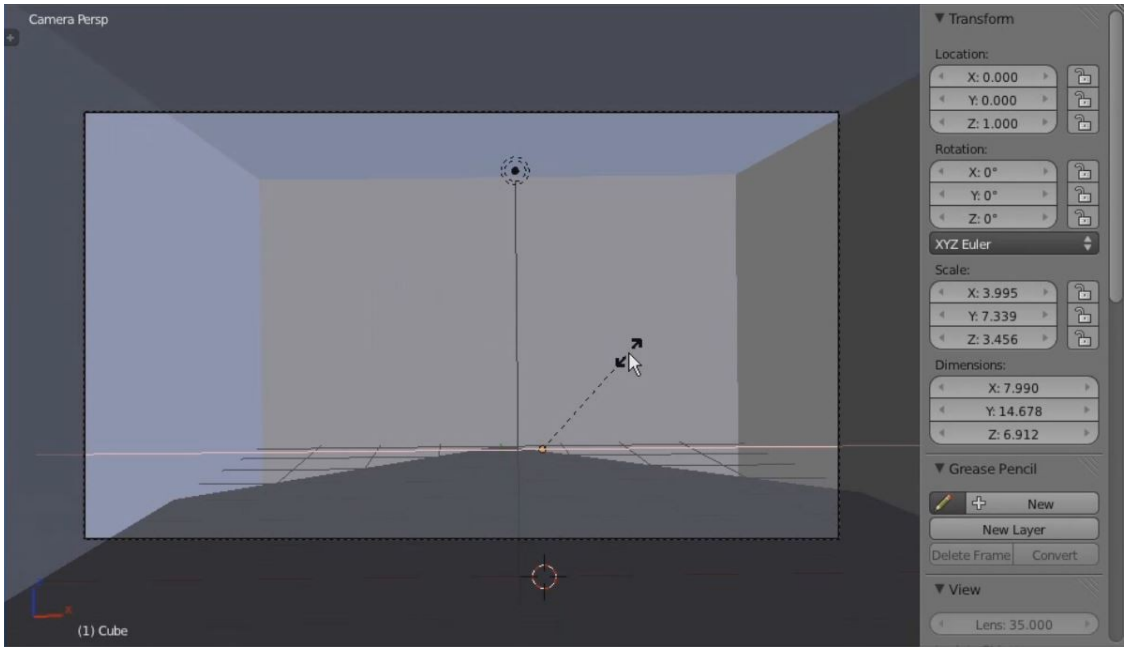
Passiamo al motore Blender Game, il motore di gioco, con modalità di ombreggiatura GLSL, lasciando però Solid nelle 3D View.



*Blender Game, GLSL, anteprima SOLID*

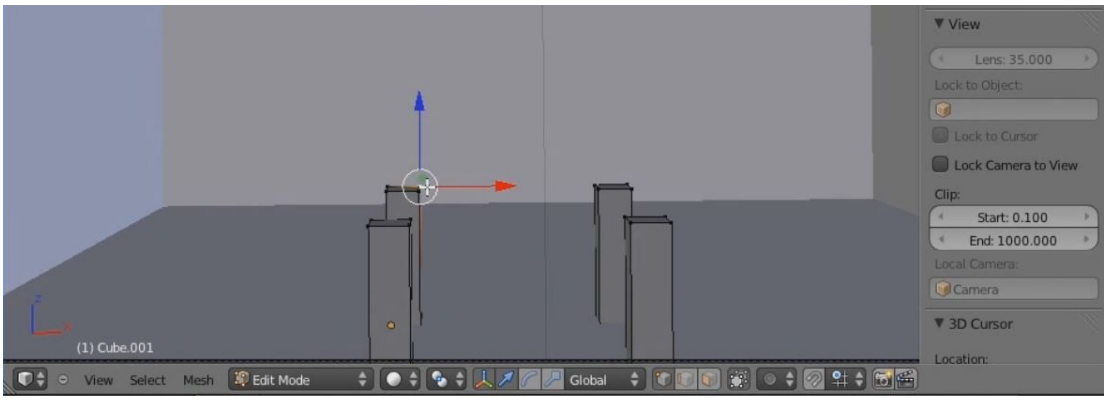
Iniziamo la nostra puntata da lontano, in modo da vedere, strada facendo, quali imprevisti possono verificarsi nella realizzazione della scena... ho detto infatti che in questa puntata parleremo del Bake delle Textures (per mostrare ad esempio come ottimizzare le ombre) e delle Textures animate, ma un tutorial riguardante solo questi argomenti sarebbe troppo isolato dal resto, mentre un buon corso mostra vari elementi collegati e qualche trucchetto.

Iniziamo posizionando la telecamera all'interno di un cubo che scaleremo a dovere in modo da rappresentare una stanza vista dall'interno. Al centro di questa stanza metteremo un semplice tavolino, da utilizzare per proiettare un'ombra da fissare sulla Texture del pavimento mediante Bake.



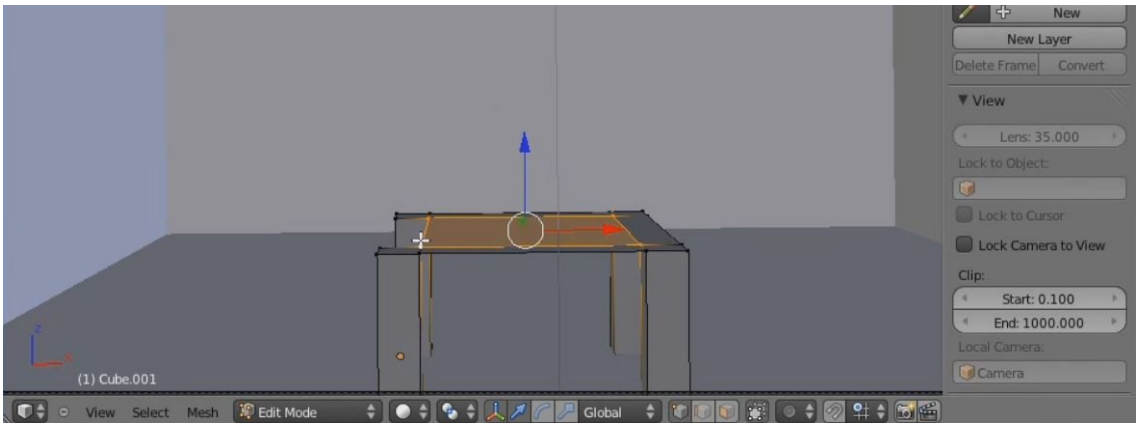

*La telecamera all'interno del cubo*

Per aggiungere il tavolino iniziamo aggiungendo un cubo, quindi restringiamolo e scaliamolo solo lungo l'asse Z, quello verticale, in modo da ottenere una gamba del tavolo; passiamo in Edit Mode, selezioniamo tutto e facciamo un paio di duplicazioni e spostamenti per posizionare le altre gambe del tavolino nello spazio.




*Prima parte della modellazione del tavolino*

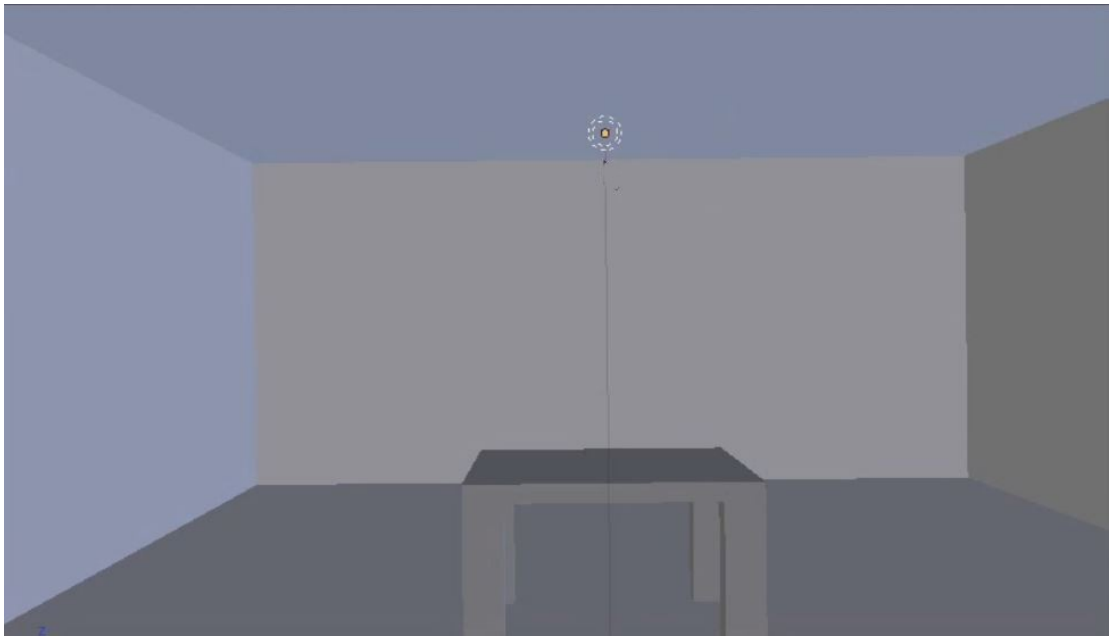
Per chiudere il tavolino, aggiungendo sostanzialmente il piano vero e proprio, selezioniamo vertici o spigoli delle gambe del tavolo e realizziamo le facce di collegamento con F; realizzata una prima superficie chiusa, estrudiamola verso l'alto in modo da dare spessore al piano.




*"Chiusura" della mesh del tavolino*

Torniamo in Object Mode. In questo momento non ce ne rendiamo conto, ma la mesh appena realizzata ha un grosso difetto... ci torneremo tra poco, in questa stessa puntata.

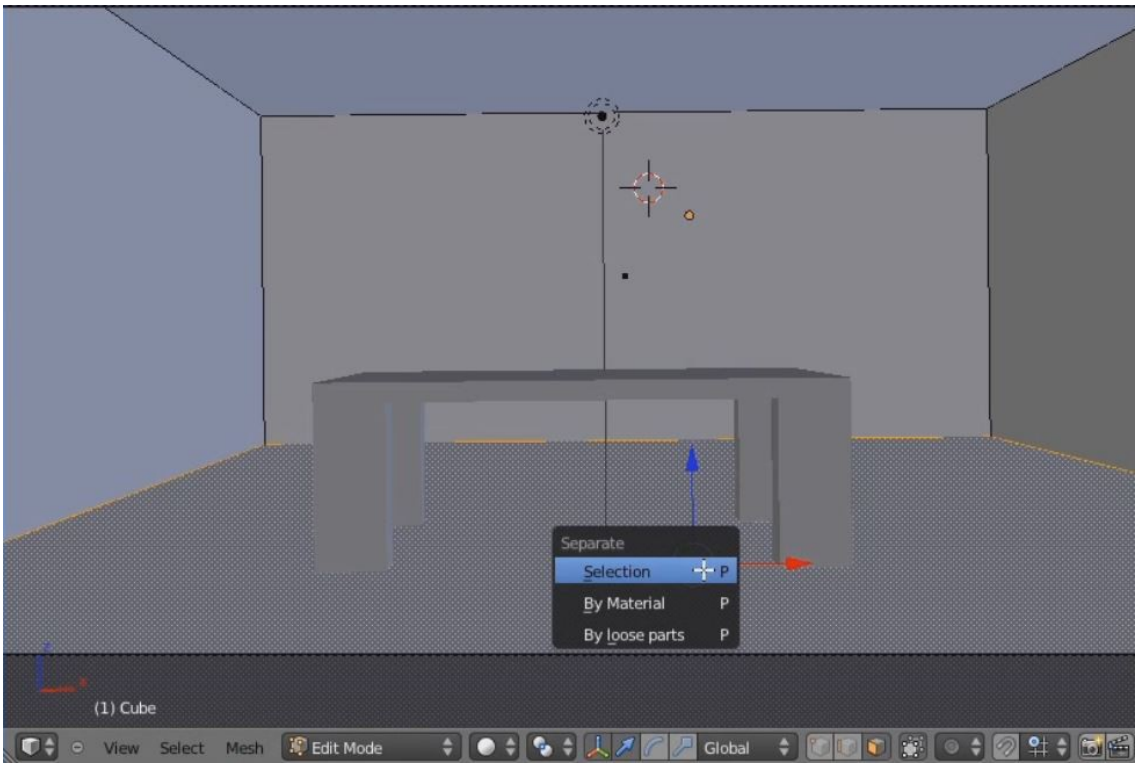
Sopra il tavolino posizioniamo una fonte di luce Omni Light, la classica Lamp; dalla puntata precedente, sappiamo che questa luce non può proiettare ombre in tempo reale nella scena.



*Fonte di luce Omni Light sul tavolino*

Dal momento che il nostro personaggio non proietterà ombre sugli altri elementi presenti nella scena, useremo la Lamp a runtime, senza ombre in tempo reale, così da non appesantire l'esecuzione del gioco.

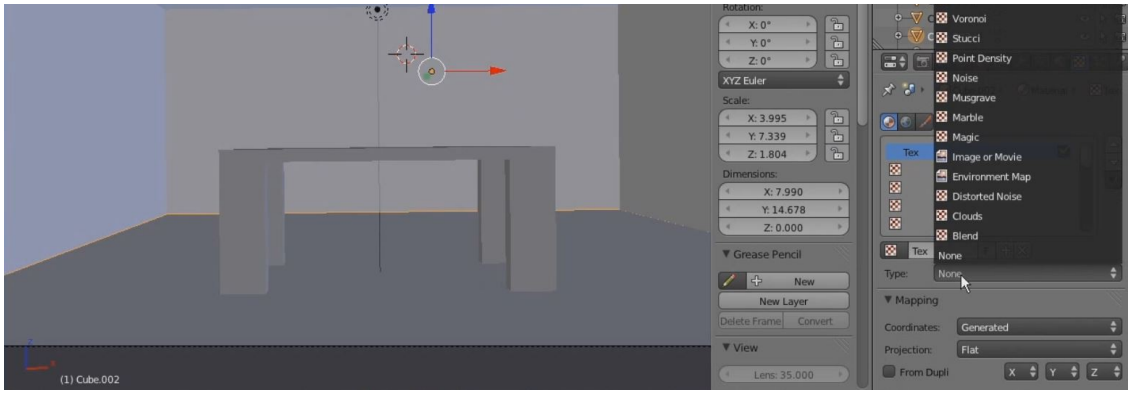
Per isolare la mesh del pavimento rispetto al resto del cubo, selezioniamo la faccia inferiore di quest'ultimo (in Edit Mode, modalità selezione facce) e premiamo P, confermando poi l'operazione di separazione dal menù che apparirà a video.




*Separazione del pavimento dal resto del cubo con P*

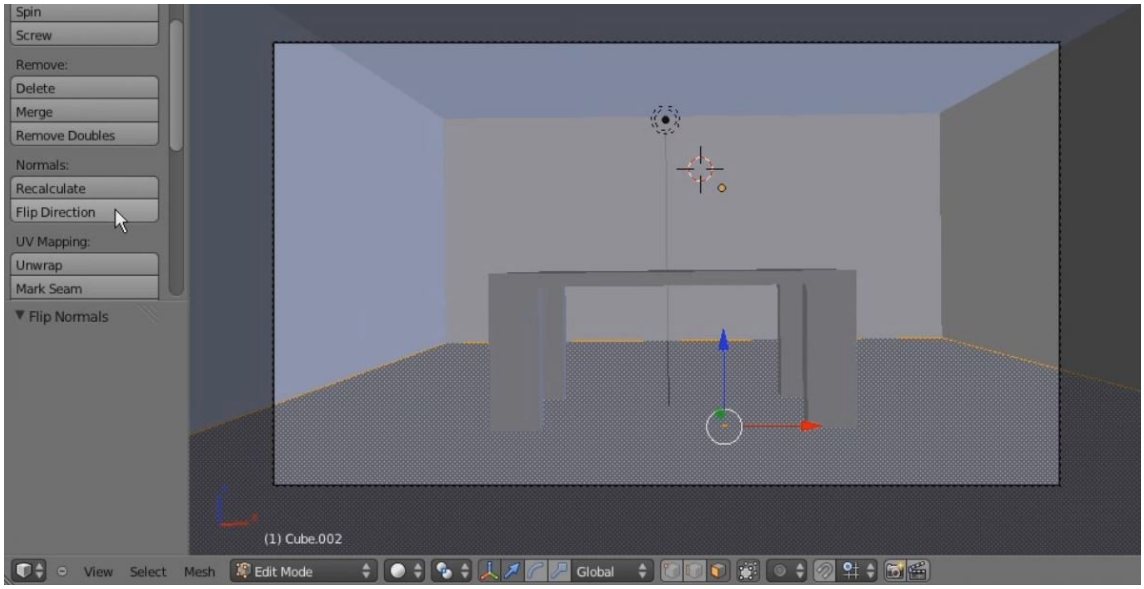
Torniamo quindi in Object Mode e selezioniamo il pavimento.

Forniamo una Texture al pavimento selezionandolo, aggiungendo un Material di base e una Texture di tipo immagine.



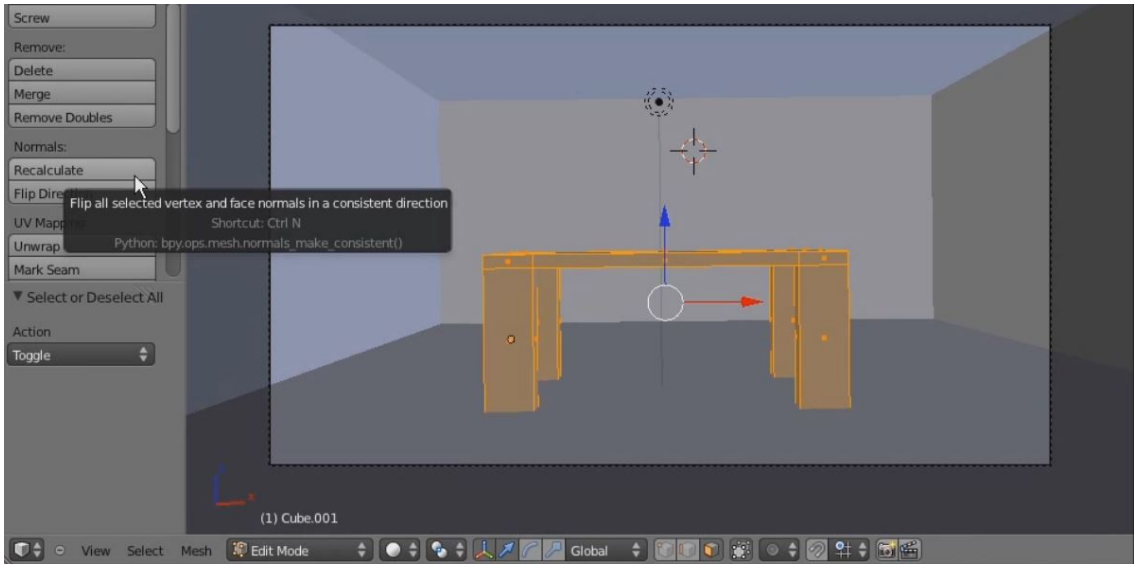

*Material e Texture di base per il pavimento*

Nella 3D View, passiamo alla modalità di visualizzazione Textured per vedere un'anteprima.
Qui sorge il problema presente, come anticipato, anche per il tavolino: le normali delle facce non sono impostate correttamente. Nel caso del pavimento, il motivo è che esso era in realtà parte di un cubo, dove ogni faccia aveva le normali verso l'esterno, mentre qui ci troviamo all'interno del cubo, per cui dobbiamo andare in Edit Mode e flippare le Normali, sia per il pavimento che per la mesh che, ora, rappresenta il resto della stanza, ossia pareti e soffitto.




*Flippare le Normali per il pavimento*

Il problema del tavolino è simile ma la soluzione è diversa: in questo caso, selezioniamo l'oggetto, passiamo in Edit Mode, selezioniamo tutti i vertici, gli spigoli e le facce e facciamo ricalcolare a Blender le normali in maniera corretta con Recalculate.



*Ricalcolare le Normali per il tavolino*

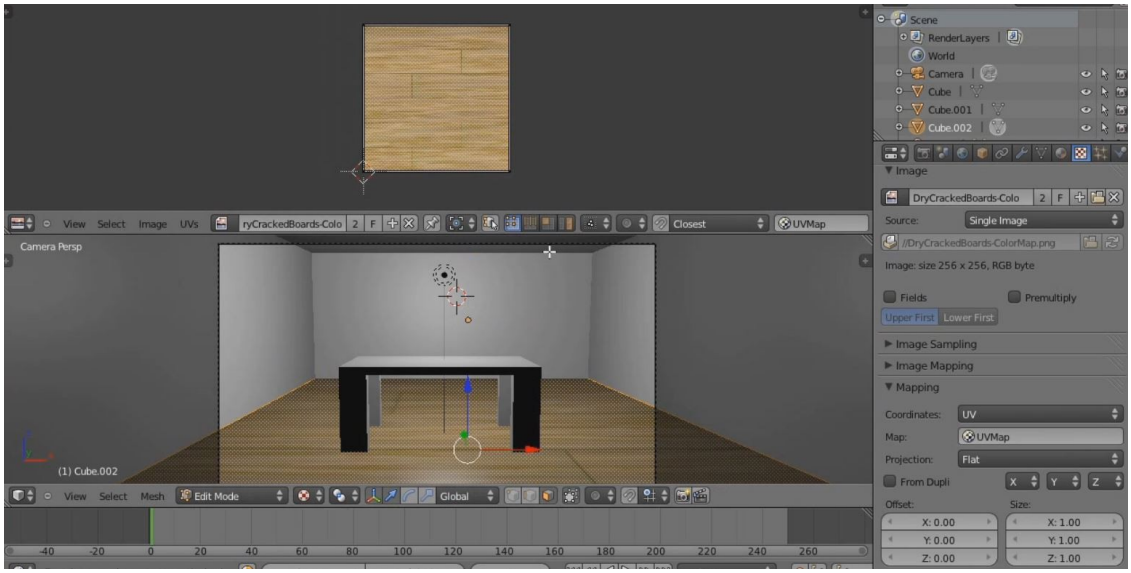
L'operazione viene svolta correttamente perché siamo partiti da un'unica mesh, duplicata e modellata in Edit Mode (non creando cioè oggetti distinti, da fondere in seguito) e realizzando una mesh chiusa, per cui Blender può individuare facilmente il dentro e il fuori; se partite da più oggetti distinti da fondere in uno solo, ricorrete a Boolean Union, non al semplice Join.

A questo punto vi anticipo una cosa: l'utilizzo di mappe Normal Maps, che useremo tra poco, non funziona correttamente se non si fornisce un Layer UV agli oggetti; per questo motivo, anche se il pavimento è un semplice Plane, realizziamo l'UV Mapping. È comunque un buon esercizio per cercare di abituarsi a ricorrere sempre e comunque all'UV Mapping, quando si realizzano giochi col Game Engine.

Implementiamo un UV Mapping elementare: va bene anche il semplice Unwrap, visto che c'è solo una faccia nell'oggetto; ricordiamoci, ovviamente, di cambiare la mappatura UV Layer nella sezione Mapping della Texture.

*UV-Unwrapping del pavimento*

La Texture immagine è un po' piccola per il nostro pavimento, quindi eventualmente scaliamo la scucitura nella finestra "UV – Editor delle immagini"; conviene munirsi di Textures predisposte per il tiling, dove cioè la parte destra e la parte superiore dell'immagine si "agganciano" perfettamente alla parte sinistra e inferiore, rispettivamente, per cui affiancando più immagini (tiling, appunto) non notiamo punti di discontinuità nell'aspetto finale.




*Texture immagine per il pavimento*

Ok, abbiamo impostato correttamente la Texture per definire l'aspetto del pavimento. Di questa Texture ho a disposizione anche la versione Normal Map, per simulare dei piccoli rilievi o delle impurità nella superficie di un oggetto, in questo caso il pavimento, senza dover modificare la mesh.

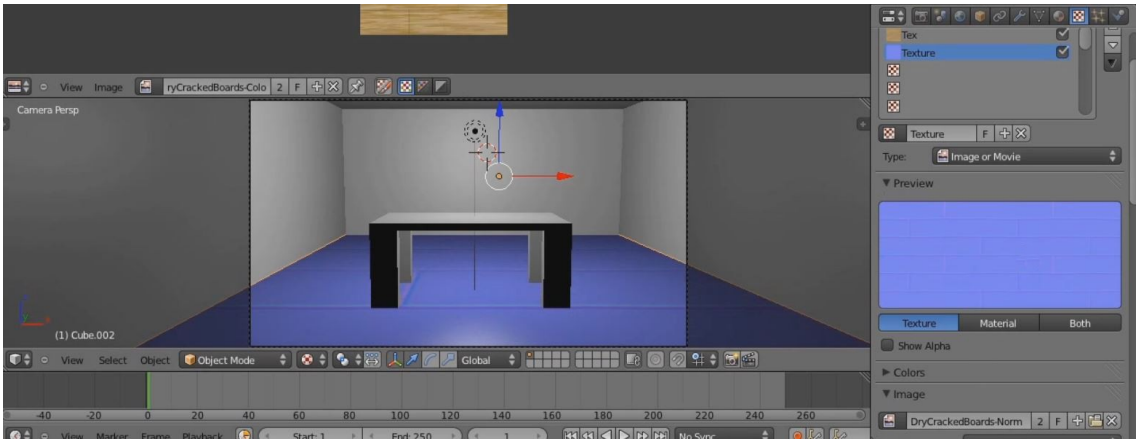
Questo è un punto molto importante nella progettazione di un ambiente o di un personaggio per un videogioco: evitare mesh complesse, in quanto i calcoli per gestire le coordinate dei vertici degli oggetti vengono effettuati dalla CPU, che deve occuparsi anche dello stato generale del gioco, dell'interazione con l'utente, delle comunicazioni di rete, eccetera... Ci sono calcoli che vengono effettuati invece dalla GPU, la scheda video, per cui (dove possibile) è buona norma "spostare il carico di lavoro" dalla CPU alla GPU.

Nel caso del pavimento del nostro scenario, lo sviluppatore poco esperto potrebbe pensare di realizzare i vari listelli del pavimento come mesh a parte, magari suddividendo e deformando le mesh con qualche modificatore, in modo da realizzare un pavimento davvero realistico... troppo realistico, forse anche inutilmente realistico! Uno scenario complesso, che richiederà molti calcoli da parte della CPU, per un semplice pavimento!

L'alternativa è aggiungere dettagli mediante le Textures sul canale Normal, anche perché questo effetto viene gestito senza complicare la mesh (infatti, fateci caso, stiamo utilizzando un Plane con una sola faccia rettangolare, per il pavimento, niente suddivisioni o altro) e i calcoli vengono effettuati dalla GPU, non dal processore, rendendo l'esecuzione del gioco più fluida.

Aggiungiamo quindi una seconda Texture immagine al Material, caricando il file immagine della mappa di rilievo nel secondo slot delle Texture e cambiando Influence da Color Diffuse (che va disattivato) a Geometry Normal; va bene anche 1 come valore, ma attenzione: dobbiamo selezionare anche la casella Normal Map nella sezione Image Mapping della Texture, per dire a Blender di utilizzare la Texture come mappa per i rilievi sulle Normali, appunto.

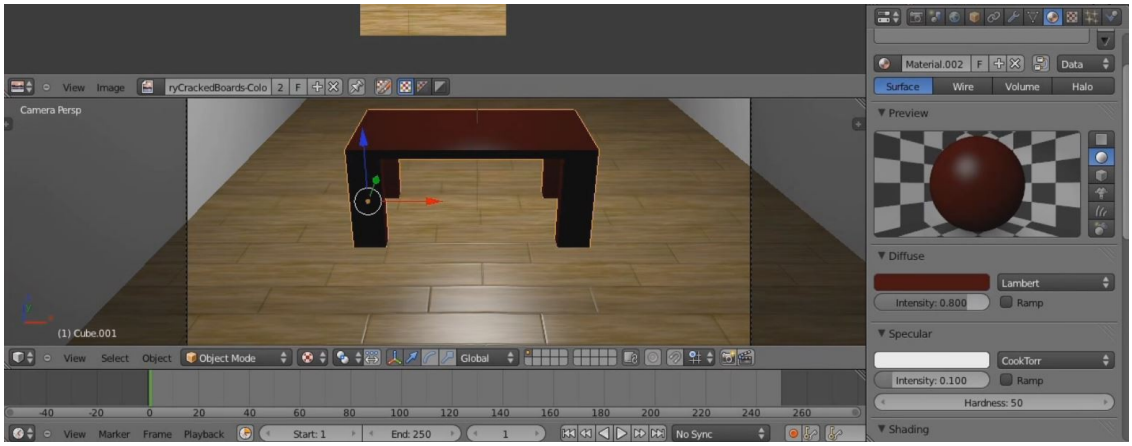
Attenzione: dobbiamo impostare la mappatura UV, con il layer UV creato automaticamente da Blender, anche per questa seconda Texture.




*La Texture NORMAL*

Le immagini delle Textures (quella per il colore e questa per le Normali) hanno la stessa risoluzione, quindi non c'è bisogno di effettuare scaling o altre operazioni per farle combaciare (abbiamo scalato la scucitura nel Layer UV per replicare la Texture in fase di UV Mapping).

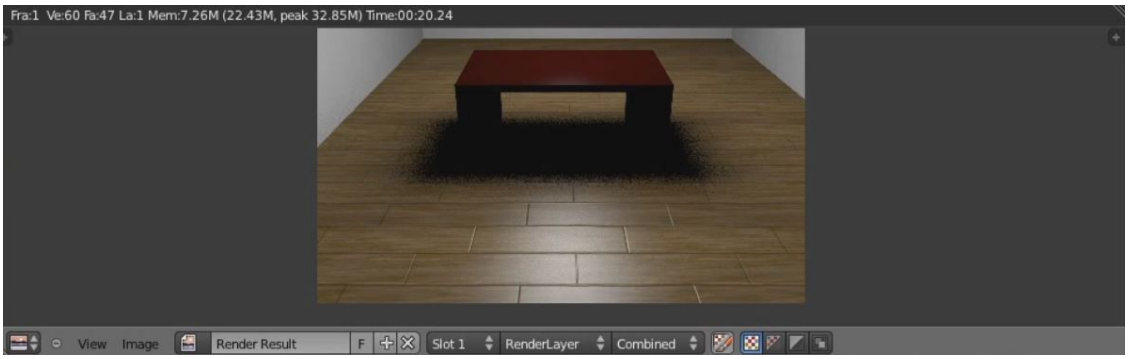
Forniamo un Material di base al tavolino in modo da completare le operazioni preliminari prima di passare al primo dei due argomenti di questo tutorial: il Bake delle Textures.




*Il Material del tavolino*

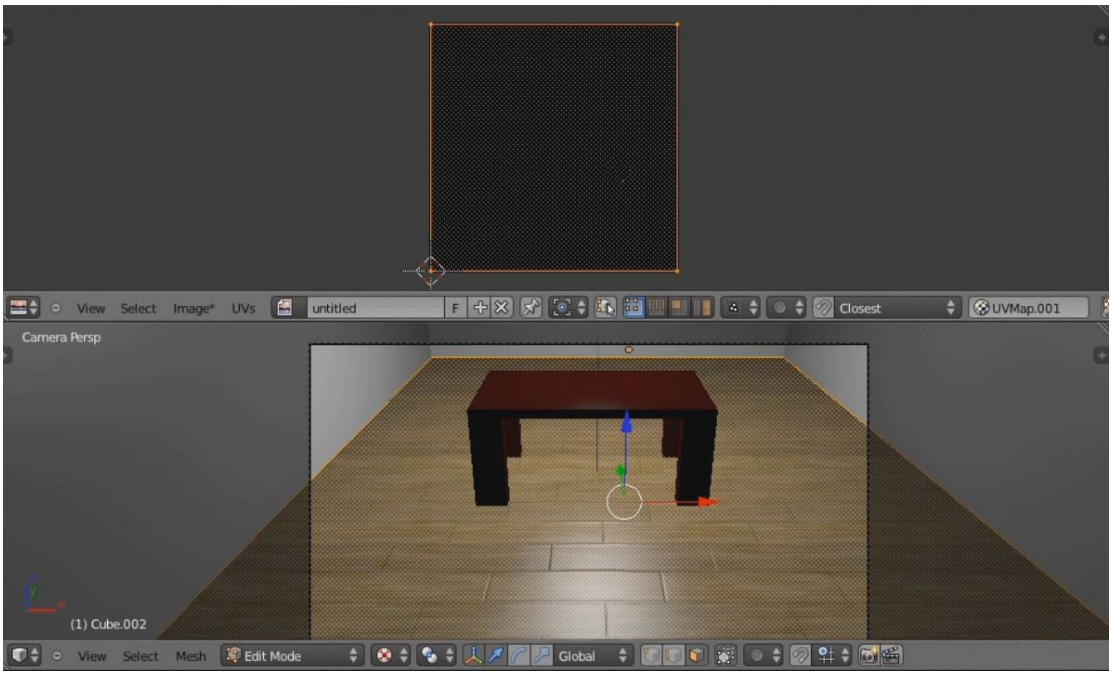
In realtà, il Bake può essere utilizzato per più scopi (ombre, Ambient Occlusion, Displace su Normals per mappare mesh High Poly su Low Poly, eccetera); in questa puntata vedremo quello delle ombre.

Torniamo a Blender Render e impostiamo Soft Size 2 e magari 8 o più Samples per la fonte di luce. Chiaramente, in realtime non abbiamo ombre, specialmente sfumate. Lanciamo invece un rendering della scena con Blender Render: qui le ombre ci sono... e ci vanno pure bene.




*Il rendering delle ombre*

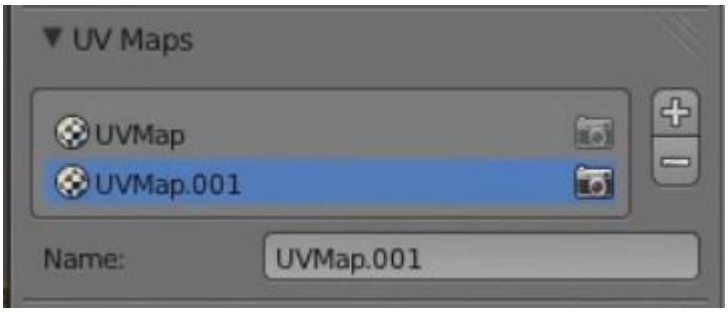
Come fare per ottenere la Texture delle sole ombre da applicare al Plane?
Selezioniamo la mesh del pavimento, già unwrappata, e passiamo in Edit Mode, selezionando tutti i vertici, gli spigoli e le facce e visualizzando l'unwrap in una finestra "UV – Editor delle immagini".
… ALT: c'è un problema! Non possiamo effettuare il Bake ora perché l'unwrap visualizzato è quello scalato su una delle due Texture aperte prima (Color o Normal).
Clicchiamo quindi su Image New per creare una nuova immagine con le impostazioni di default (1024 per 1024, nera).




*Nuova immagine (nera, inizialmente) per il Baking*

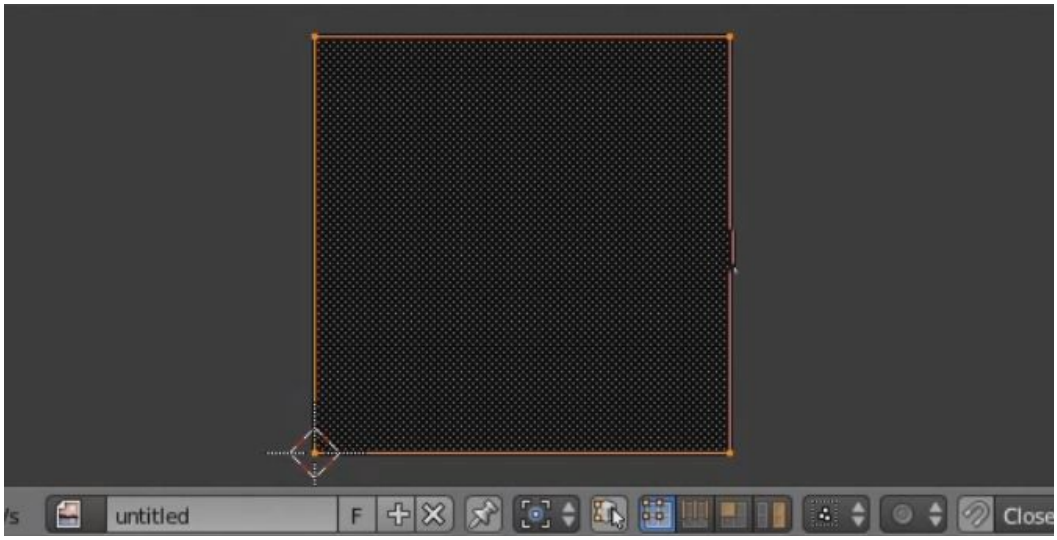
Problema: l'unwrap di prima è ancora scalato ed è più grande dell'immagine.

Realizziamo quindi un secondo layer UV aggiungendone uno nella pila UV Maps nella scheda Object Data, nella Properties Window e selezioniamolo, cliccando in particolare sulla macchina fotografica a destra, per dire a Blender che ora stiamo usando questo Layer.



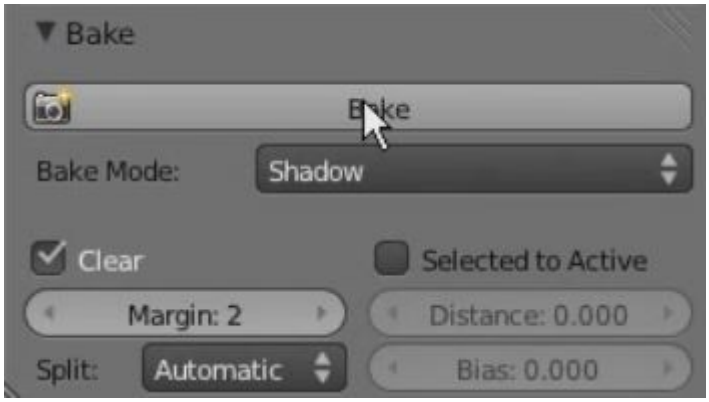

*Secondo Layer UV*

Rifacciamo l'unwrap della mesh sull'immagine nera: adesso l'unwrap, associato a questo secondo Layer UV, resterà dentro l'immagine nera.



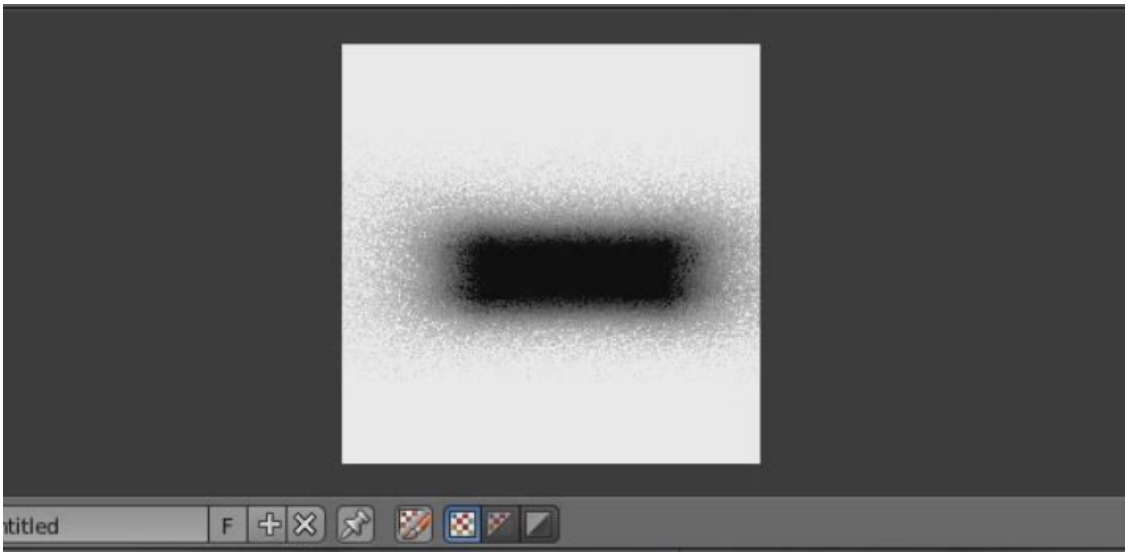

*Nuovo Unwrap, sul nuovo Layer UV, con l'immagine nera*

È giunto il momento di fare il Bake delle ombre: nella scheda Render, all'interno della Properties Window, apriamo la sezione Bake, scegliamo Shadows (ombre) e avviamo il Bake cliccando sul relativo pulsante.




*Baking*

A seconda dei casi, questa operazione richiederà poco o molto tempo; in questo caso poco, perché la scena e la mesh sono molto semplici: nella finestra UV vedremo il disegno delle ombre presenti sull'oggetto pavimento in questo preciso momento.



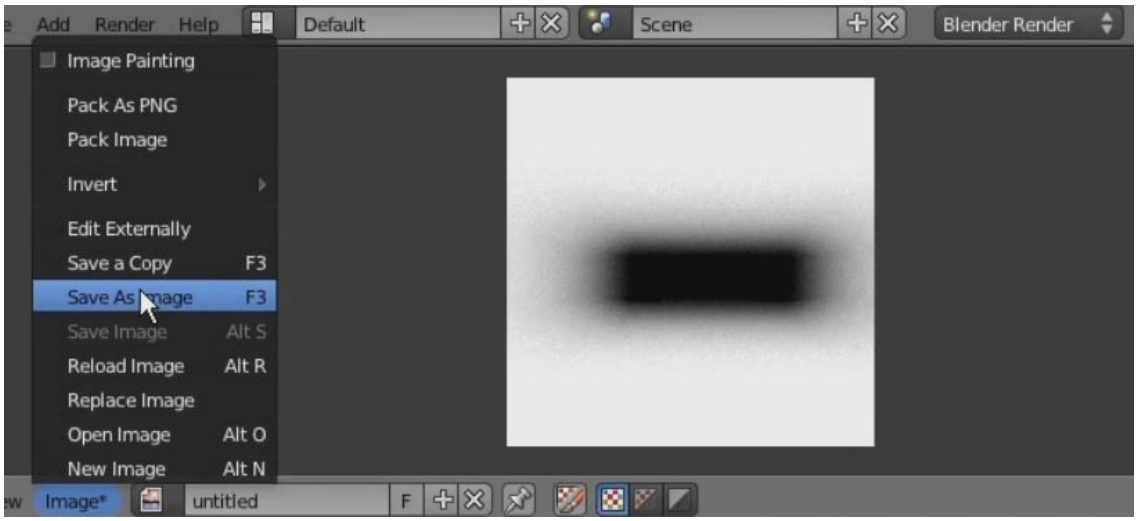

*Risultato del Baking*

Ecco cosa fa quindi il Bake: “fissa”, per così dire, l'aspetto di un oggetto in un dato momento su un'immagine 2D (ecco perché dobbiamo fare l'unwrap e, dopo, la mappatura UV); tale immagine potrà essere applicata all'oggetto per SIMULARE, quindi, certi effetti, in questo caso le ombre, senza implementarli davvero. È un trucco utilizzato per ottenere un certo effetto con pochi calcoli ed utilizzando quindi poche risorse.

Il rovescio della medaglia è che spostando gli oggetti o le fonti di luce, le ombre non varieranno, in quanto in realtà si tratta di una Texture immagine precalcolata, non di vere e proprie ombre.

Vediamo comunque come applicare la Texture appena ottenuta per realizzare le ombre.

Per prima cosa, salviamo l'immagine ottenuta mediante Bake nella finestra UV Editor delle immagini come file immagine a parte, su disco.

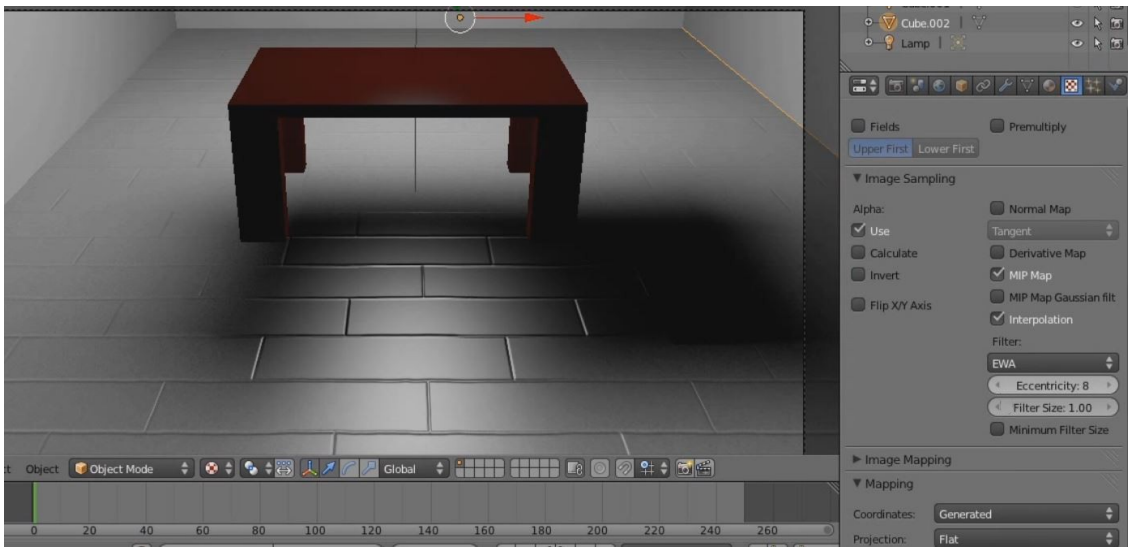



*Salvataggio dell'immagine ottenuta mediante Baking*

Faccio presente che possiamo elaborare come vogliamo questo file immagine con un qualsiasi programma di fotoritocco, ad esempio sfumando o colorando le ombre, aggiungendo altri elementi, eccetera.

Passiamo alla scheda Texture del Material del pavimento, selezioniamo il terzo slot e aggiungiamo una nuova Texture immagine, caricando ovviamente l'immagine del Bake delle ombre.

Nella sezione Mapping, cambiamo le coordinate in UV e scegliamo il secondo Layer UV (quello creato appositamente per la Texture Bake).




*Utilizzare l'immagine Baked come Texture per il pavimento*

In Influence, lasciamo selezionato Diffuse Color (in quanto vogliamo influenzare il colore dell'oggetto, in effetti) ma cambiamo la modalità di miscelazione in Multiply.

Come spiegato nei miei ebook su Texture e Compositing, l'effetto pratico di Multiply è quello di scurire gli oggetti nei punti corrispondenti a pixel grigi o neri della Texture e di lasciare il colore originale nei punti corrispondenti ai pixel bianchi.

*La Texture del legno, quella delle Normali e quella delle Ombre, miscelate insieme*

Multiply è infatti la moltiplicazione del colore sottostante per il valore numerico del pixel della Texture, che vale 1 se è bianco (quindi ad esempio il rosso puro, moltiplicato per 1, restituirà il rosso puro), 0 se è nero (quindi il rosso puro, moltiplicato per 0, restituirà nero) e valori intermedi per i grigi (con l'effetto di scurire il rosso, nel nostro esempio).

Torniamo al Game Engine e avviamo il gioco con P: adesso, come potete vedere, c'è l'ombra del tavolino sul pavimento, anche se in effetti la nostra fonte di luce NON proietta ombre in tempo reale, cosa che possiamo vedere in due modi: osservando le gambe del tavolino, che non sono ombreggiate (e da qui uno spunto: fare il Bake delle ombre anche per il tavolino), oppure spostando il tavolino o la fonte di luce nella scena 3D in fase di modellazione ed avviando nuovamente il gioco (l'ombra resterà al suo posto).

*Ombra fasulla*

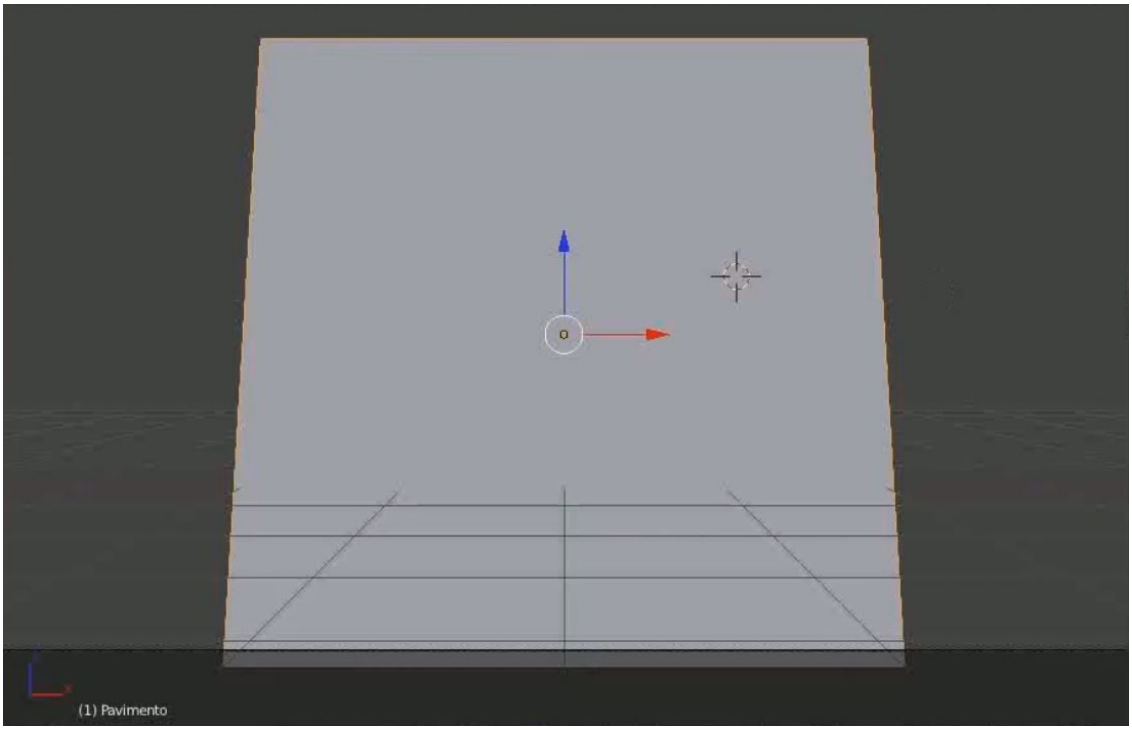
Abbiamo parlato del Bake di un effetto (le ombre) ma, come avrete notato scegliendo “Shadow” dal menù del baking, è possibile effettuare il Bake anche per altri canali, come ad esempio per le Textures. Quest'opzione, in particolare, ci consente di fare il Bake delle Textures procedurali, mappandole su un'immagine mediante unwrap per applicarle in seguito come immagini UV Mapped agli oggetti. Si tratta di un'operazione utile perché il Blender Game Engine NON visualizza a runtime le Textures procedurali, per cui, se vogliamo utilizzare una Texture del genere (ad

esempio Cloud, Voronoi, Marble, Wood, eccetera), dobbiamo fare l'unwrap, il Bake del canale Texture, salvare l'immagine della Texture procedurale mappata sulla scucitura della mesh ed applicare poi tale immagine come Texture UV Mapped alla mesh, così come fatto poco fa per l'immagine che rappresentava le ombre.

Quanto detto conclude la prima parte di questo tutorial; passiamo quindi alla seconda, riguardante le Textures animate nel Game Engine!

*  *  *

Aggiungiamo una mesh nella scena (in questo caso un semplice Plane); 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.

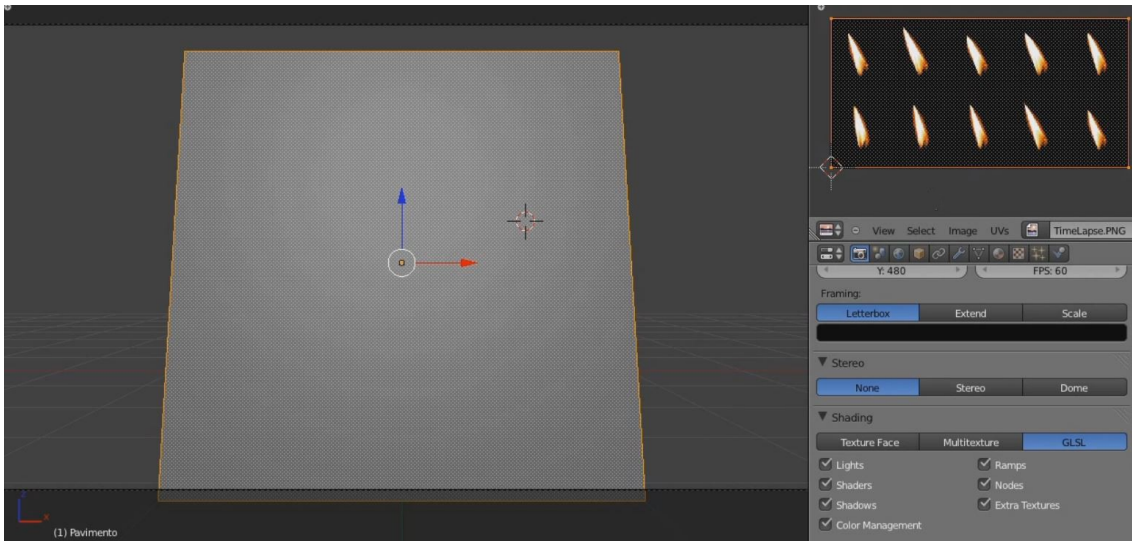



*Aggiungere la mesh (un Plane, in questo caso) nella scena*

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.

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.

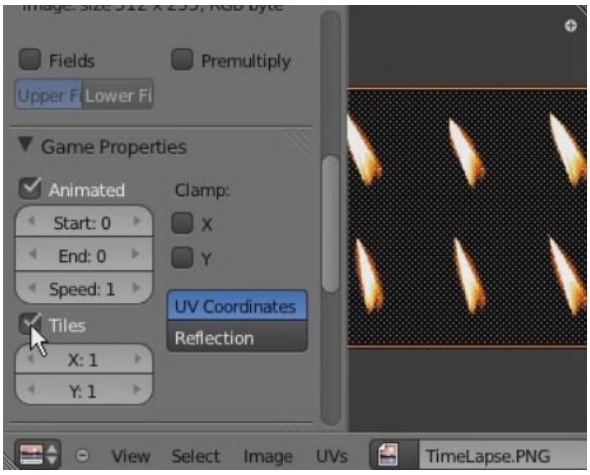
Facciamo l'unwrap del Plane assegnandogli, in una finestra "UV – Editor delle immagini", l'immagine con le varie Sprites; sto utilizzando un'immagine composta da 10 sprites, disposte su 5 colonne e 2 righe.




*Unwrap e assegnazione della Texture con le Sprite al Plane*

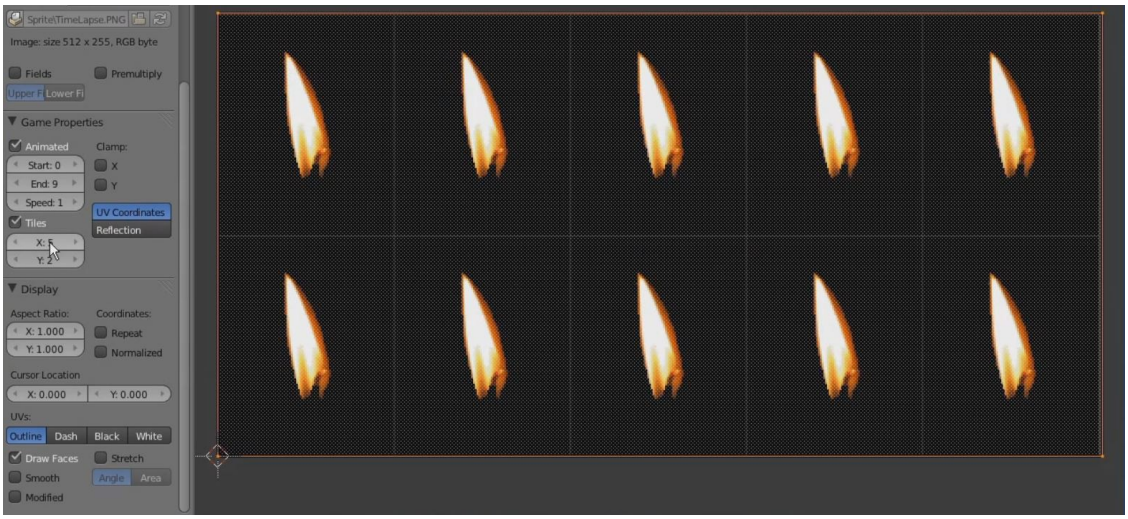
Nel fare l'unwrap, copriamo tutta l'immagine con la sagoma del Plane (non solo la prima sprite), dopodiché apriamo il pannello Properties della finestra UV con N (o con View - Properties) e, in basso nel pannello, scegliamo Animated e Tiles.

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.




*Animated Tiles*

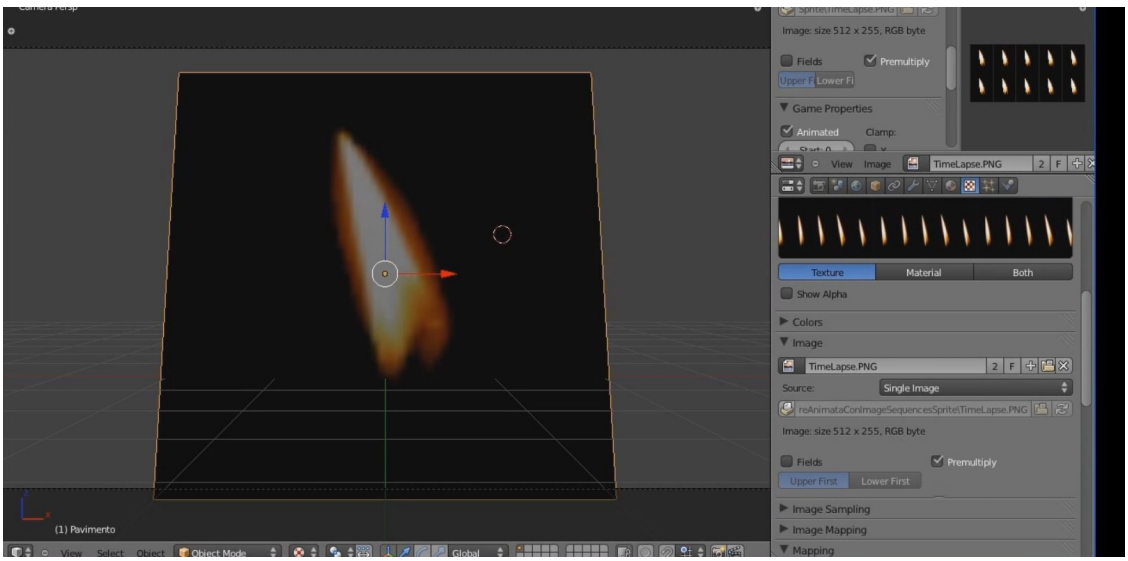
In Tiles specifichiamo 5 per X e 2 per Y: in questo modo, Blender suddividerà in 10 parti l'immagine e prenderà una Sprite per volta, secondo l'ordine dalla prima riga in alto all'ultima in basso, dalla colonna più a sinistra a quella più a destra.



*Impostazioni Tiles*

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.

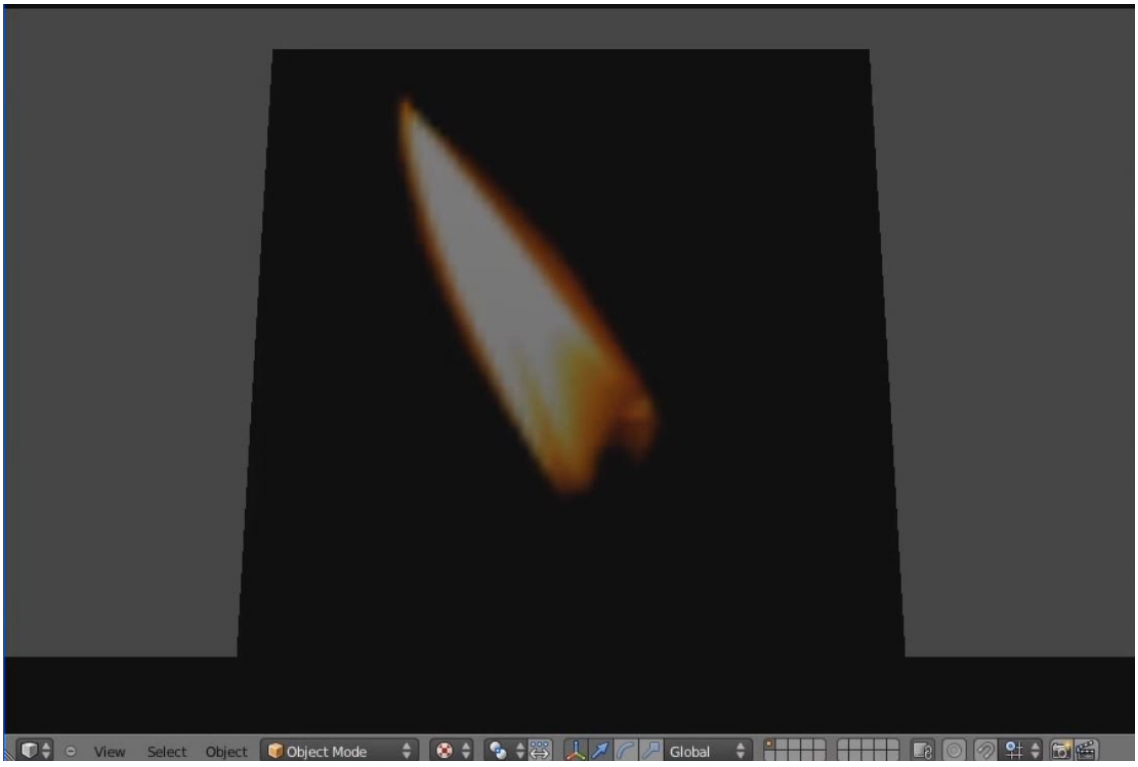
Aggiungiamo un Material all'oggetto, personalizzando a piacere gli ombreggiatori e le altre caratteristiche proprie del Material, dopodiché aggiungiamo una Texture di tipo immagine all'oggetto.



*Aggiungere la 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 UV 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 real time 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.



*Fotogramma del risultato, dopo aver premuto P*

*  *  *

# Le basi del Game Engine, il motore di gioco di Blender 2.6 – 3

Eccoci alla terza puntata del corso sulle basi del Blender GE, il Game Engine, motore di gioco di Blender 3D 2.6; in questa puntata, come anticipato nelle puntate precedenti, inizieremo a parlare dell'interazione (in particolare, mediante tastiera), dello schema logico sensori-controller-attuatori, delle proprietà e della fisica nel GE.

Dico "inizieremo" perché questi argomenti sono piuttosto vasti e verranno approfonditi nelle prossime puntate: non possiamo parlare di tutto in questo tutorial, per cui qui vedremo un'introduzione e un esempio pratico. Per prima cosa, quindi, due parole sulla logica nel GE.

Il Blender Game Engine è stato progettato per consentire un approccio il più possibile visuale alla definizione dell'ambiente di gioco, definendo cioè i comportamenti e le interazioni degli oggetti con poche proprietà o condizioni collegate tra loro e rappresentate in maniera visiva mediante "blocchi", detti Logic Bricks (letteralmente, i "mattoncini della logica" del gioco).

È quindi possibile realizzare applicazioni senza conoscere nemmeno le basi della programmazione o del linguaggio Python, ma la cosa positiva è che è possibile estendere le funzionalità di base del Game Engine mediante la programmazione, se si conoscono le API di programmazione, ossia le librerie di classi e metodi per interfacciarsi col GE... e, ovviamente, il linguaggio Python.

È più facile comprendere il sistema "logico" di sensori, controller e attuatori mostrando direttamente un esempio pratico piuttosto che descrivendolo a parole, per cui con la scena di base passiamo subito al motore di gioco nel selettore Engine, nella finestra Info, al sistema di ombreggiatura GLSL e alla modalità Textured (questo non è fondamentale, ma così ci abituiamo a farlo), dopodiché selezioniamo ad esempio il cubo di default e soprattutto cambiamo una finestra del programma in Logic Editor, ossia l'ambiente di sviluppo visuale della logica del gioco.

Blender mette a disposizione proprio un Layout per lavorare meglio alla realizzazione di un'applicazione, per cui è possibile cambiare lo Screen in Game Logic per visualizzare immediatamente 3D View, editor di testo (per realizzare o visualizzare script in Python o altre risorse), Outliner, Properties Window e Logic Editor.

*Finestra Logic Editor*

Come possiamo notare, la finestra Logic è divisa in due parti: a sinistra una sezione chiamata "Properties" inizialmente vuota (a parte il pulsante "Add Game Property"), a destra una sezione con tre colonne: sensori, controller e attuatori.

Le due sezioni possono essere ridimensionate ed è possibile spostarsi nelle aree e zoomare; in particolare, il panning si fa con click della rotellina o testo centrale del mouse, lo zoom con + e – del tastierino numerico oppure con la rotellina del mouse.

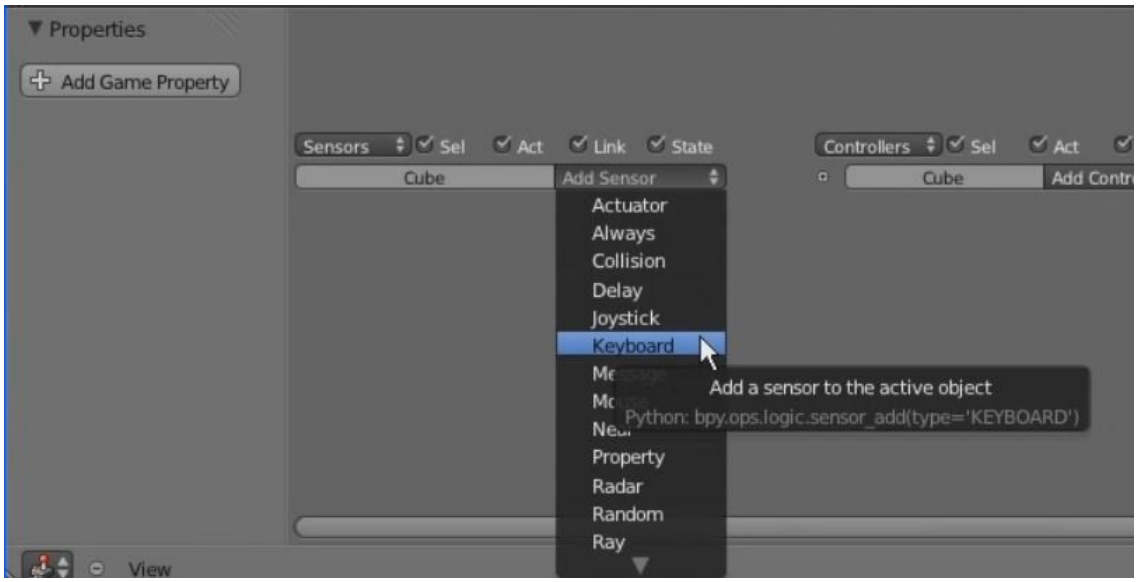
> *Nota importante: questi elementi (ossia proprietà, sensori, controller e attuatori) fanno riferimento all'oggetto attivo della scena, che nel nostro caso coincide con l'unico oggetto selezionato, il cubo, come possiamo notare comunque osservando i pulsanti "Object Name" sotto le etichette di sensori, controller e attuatori.*

Ed eccoci al discorso della logica del GE: ad ogni elemento possiamo associare delle proprietà (che possono essere dei valori, ad esempio il livello di energia, il numero di vite, il nome, eccetera), dei sensori (che rilevano degli eventi o condizioni, ad esempio la pressione di un tasto della tastiera o una collisione con un oggetto) e degli attuatori, che effettuano delle azioni.

Se ad esempio vogliamo che alla pressione della freccia in su il cubo si sposti in avanti, ossia (ad esempio) lungo il suo asse Y (e per "suo" intendiamo l'asse LOCALE), allora dovremo inserire un sensore in grado di rilevare la pressione del tasto freccia in su per attivare un attuatore in grado di spostare l'oggetto.

I controller sono ponti di collegamento tra sensori e attuatori (per far passare il "segnale", in un certo senso); inizialmente useremo quello di default, *AND*, poi vedremo di approfondire anche questo aspetto.

Clicchiamo quindi su Add Sensor per il cubo: abbiamo a disposizione vari sensori e nelle prossime puntate li vedremo un po' tutti, trattando le loro caratteristiche. Per sviluppare il nostro esempio, ossia lo spostamento in avanti del cubo, aggiungiamo un sensore Keyboard, la tastiera.



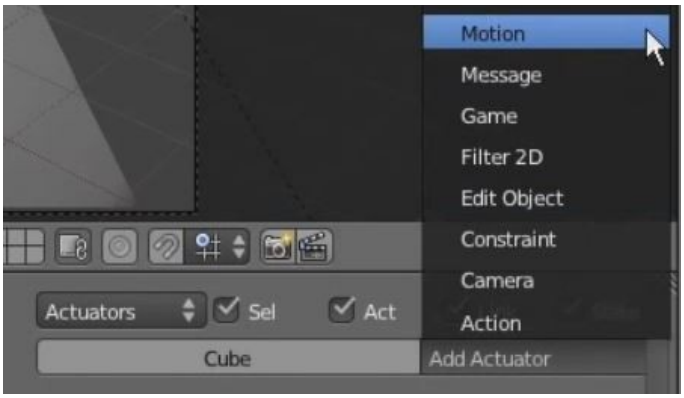
*Add Sensor - Keyboard, per il cubo*

Nella scheda del sensore possiamo impostare il tasto da rilevare (che attiverà il sensore) in Key, oppure scegliere "All Key" per attivare il sensore alla pressione di un tasto qualsiasi (ad esempio per realizzare un "premere un tasto per continuare"); nel nostro caso, quindi, facciamo click sul pulsante accanto a Key e premiamo la freccia in su nella tastiera: nel campo apparirà la scritta "Up Arrow" ("freccia in su").

*Specificare il tasto da rilevare con il sensore Keyboard*

Lanciando il gioco ora con P e premendo la freccia in su non noteremo comunque nulla: il sensore infatti c'è, ma non abbiamo associato a tale elemento l'azione da intraprendere.

Torniamo quindi alla finestra Logic Editor e facciamo click, questa volta, su Add Actuator, per aggiungere un attuatore; anche qui abbiamo varie possibilità, che tratteremo in seguito, ma qui scegliamo subito "Motion", letteralmente "Movimento".




*Add Actuator - Motion*

La scheda che ci verrà mostrata è nella versione ridotta, visto che di default gli oggetti non sono sottoposti a forze fisiche e non reagiscono a urti e collisioni; tra poco vedremo la differenza tra oggetti Statici, Dinamici e Rigid Body e noteremo anche le nuove voci nella scheda Motion, che attualmente presenta solo Loc e Rot, ossia posizione e rotazione.

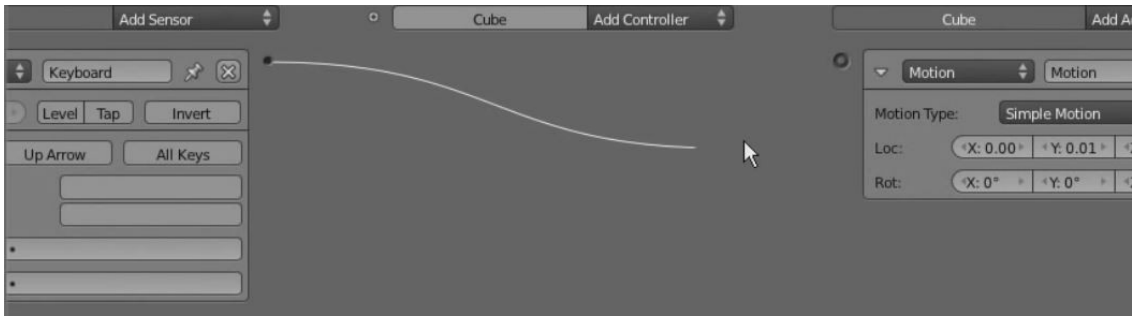
Notate le caselle L, di default attive, nelle righe Loc e Rot: indicano che gli spostamenti o le rotazioni fanno riferimento agli assi LOCALI dell'oggetto, non alle coordinate globali.
Dal momento che vogliamo spostare l'oggetto cubo lungo l'asse Y locale, lasciamo selezionate tali caselle.

Intuitivamente, possiamo scrivere in Loc Y locale l'entità dello spostamento da effettuare quando l'utente preme il tasto freccia in su, per cui scriviamo ad esempio 0.01: alla pressione di quel tasto, il cubo avanzerà di 0,01 unità di Blender lungo l'asse Y locale e continuerà ad avanzare finché terremo premuto il tasto.

*Specificare il movimento nell'attuatore*

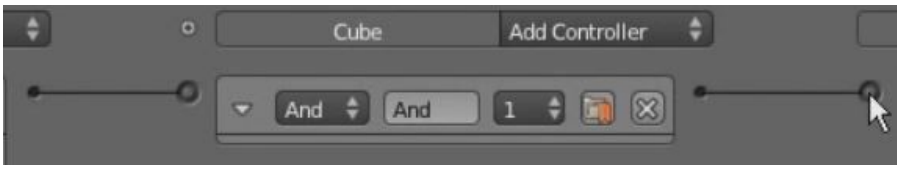
Lanciamo di nuovo l'applicazione con P, premiamo il tasto freccia in su e... ancora una volta, non succede niente. Il motivo è chiaro: sensore e attuatore ci sono, ma non sono collegati!

Nel Logic Editor, quindi, facciamo click col tasto sinistro del mouse sul cerchietto a destra della scheda di Keyboard (cerchietto detto "socket di output": porta di uscita) e, tenendo premuto il tasto sinistro del mouse, trasciniamo fino a rilasciare sulla porta di ingresso (socket di input) dell'attuatore Motion, realizzando appunto un collegamento.



*Realizzare il collegamento tra sensore e attuatore*

Anche se non l'abbiamo inserito, è apparso un controller di default nel collegamento tra il sensore e l'attuatore, in particolare un controller *AND* (dal nome della porta logica rappresentata), che nel nostro caso lascia passare tranquillamente il segnale.



*Il Controller AND*

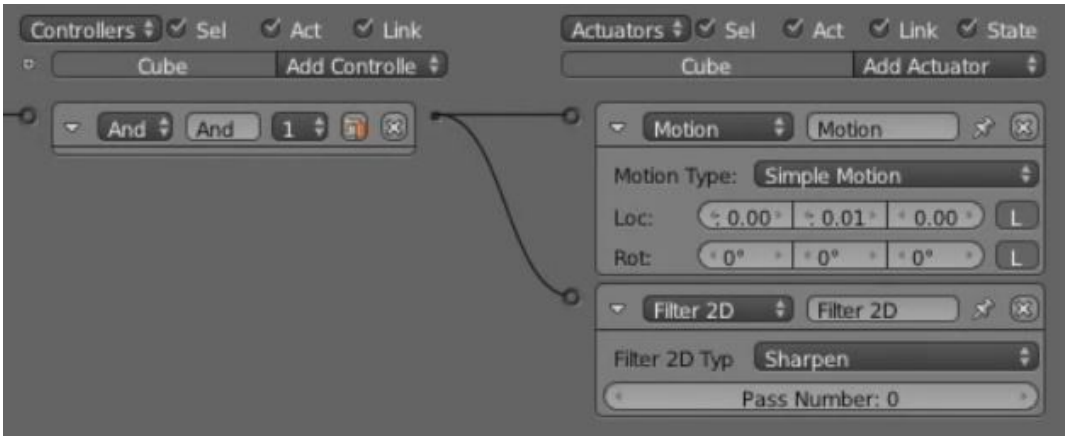
Ricapitolando: alla pressione del tasto freccia in su, il Game Engine attiverà il sensore Keyboard sensibile alla freccia in su per il cubo, che trasmetterà un segnale che passerà nel controller AND ed arriverà all'attuatore Motion, che in questo caso sposterà il cubo in avanti lungo l'asse Y locale.

Premiamo P per avviare il gioco e, dopo, la freccia in su: questa volta, il cubo si sposterà come previsto.

Da notare che, facendo uscire il cubo dal piano, l'oggetto non cadrà, in quanto di default statico (cioè non soggetto a forze o urti, nemmeno alla forza di gravità). Usciamo dal gioco e torniamo al Logic Editor.

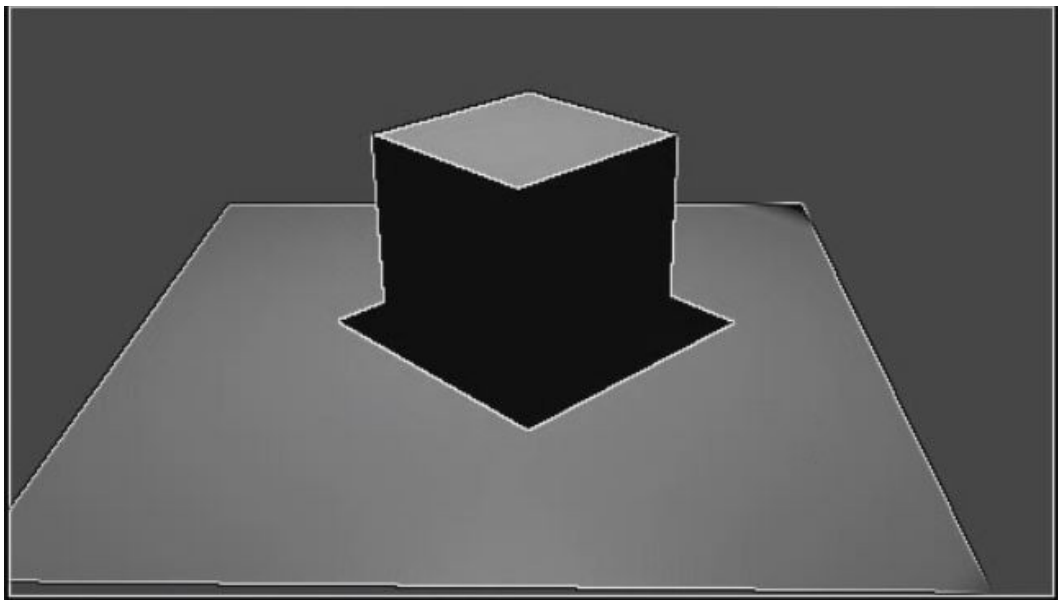
Va detto subito che da un sensore possono uscire più collegamenti, diretti a più controller, e da un controller possono uscire più collegamenti, diretti a più attuatori; inoltre, più collegamenti possono finire in un attuatore e più collegamenti possono finire in un controller, ma questo caso è un po' particolare, per cui lo esamineremo per ultimo. Vediamo quindi degli esempi.

Aggiungiamo un nuovo attuatore, questa volta di tipo “Filter”. Questo attuatore implementa un filtro di post-produzione all'immagine visibile in tempo reale; in particolare, possiamo scegliere tra più filtri, come ad esempio Sharpen (per mettere in rilievo i bordi), Blur (la sfocatura, molto utile per gli oggetti in movimento) ed altri; per notare l'effetto in questo test, scegliamo “Sharpen” e colleghiamo il controller AND anche a tale attuatore, oltre che a Motion.




*Aggiungere l'attuatore FILTER*

Esaminiamo il flusso logico: alla pressione della freccia in su, verrà generato un segnale che passerà attraverso il controller AND e arriverà sia all'attuatore Motion (spostando in avanti l'oggetto) che a Filter, attivando il filtro. Premiamo P e proviamo il tutto a runtime, premendo il tasto freccia in su.

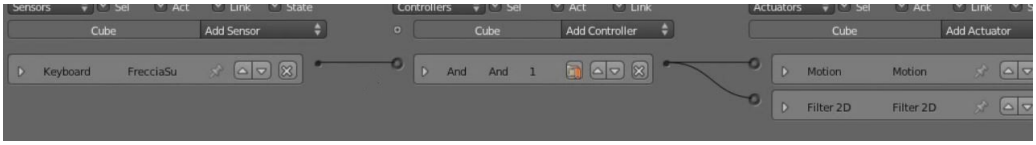


*Effetto di Filter - Sharpen a runtime*

Torniamo quindi nella finestra Logic Editor, in fase di modifica.

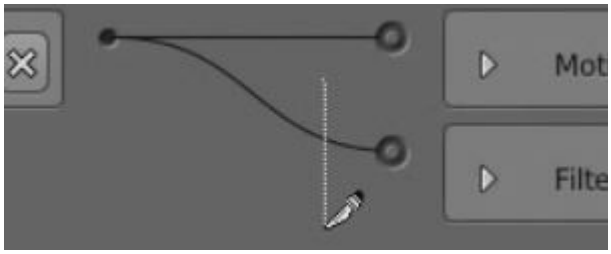
È possibile associare un nome ai vari sensori, controller e attuatori in modo da riconoscerli velocemente quando diventano parecchi o vengono minimizzati, per risparmiare spazio e rendere lo schema logico di più facile lettura; le schede vengono minimizzate cliccando col tasto sinistro del

mouse sulla freccia in alto a sinistra nelle schede, come visibile nell'immagine seguente, e vengono massimizzate cliccando nuovamente su tale freccia.



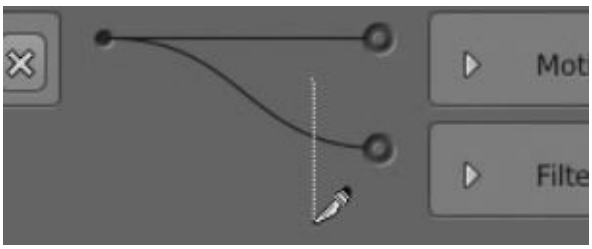

*Minimizzare le schede*

Il nome della scheda va digitato nel campo testuale in alto al centro, per cui ad esempio possiamo scrivere FrecciaSu per il sensore Keyboard che si attiva con tale tasto, "Avanti" nell'attuatore Motion su Y locale, e così via.




*Dare un nome agli elementi del Logic Editor*

Tagliamo il collegamento tra *AND* e Filter: questa operazione va fatta premendo CTRL e il tasto sinistro del mouse e, tenendoli entrambi premuti, trascinando in modo da "tagliare", appunto, il collegamento, come visibile nell'immagine seguente.




*"Tagliare" un collegamento*

*Dopo aver effettuato un "taglio"*

Togliamo l'attuatore Filter facendo click sul pulsante con la X in alto a destra nella scheda dell'attuatore; come è possibile notare, questo pulsante è presente in tutte le schede di sensori, controller e attuatori e consente la rimozione dell'elemento dal Logic Editor.

Come anticipato, è possibile collegare più sensori ad un controller, ma qui bisogna aprire una piccola parentesi.

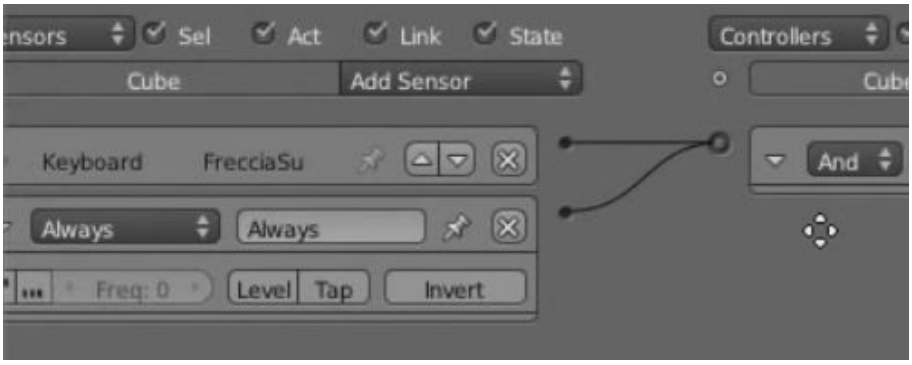
Il controller di default, *AND*, implementa appunto la porta logica *AND* che, detto brevemente, fa passare un segnale se e solo se sono presenti dei segnali positivi (o "True", "veri") in tutti i collegamenti in ingresso.
Quando c'è un solo collegamento in ingresso, come nel nostro caso, il segnale viene trasmesso nel momento in cui arriva, infatti non c'è bisogno di controllare con altri segnali: non ce ne sono!

Facciamo alcuni esempi utilizzando due sensori; il primo è Keyboard, già presente, mentre per il secondo scegliamo il tipo "Always", letteralmente "sempre", che difatti invia costantemente un segnale in uscita (ottimo, ad esempio, per realizzare contatori, timer, eccetera).

Se colleghiamo Always all'attuatore di movimento mediante un nuovo controller *AND,* creato per l'occasione, allora a runtime il cubo si sposterà da solo, perché Always manderà costantemente un segnale che, passando da un *AND* privo di altri sensori in ingresso, raggiungerà indisturbato l'attuatore Motion e sposterà il cubo.

Togliamo il secondo controller *AND* e colleghiamo Always alla socket di input del primo controller *AND,* quello che riceve anche il segnale di ingresso di "FrecciaSu": adesso il cubo non si sposterà fino a quando non premeremo il tasto freccia in su, infatti adesso *AND* deve ricevere un segnale contemporaneamente da due ingressi per trasmettere il comando agli attuatori collegati (in questo caso, solo Motion).

*AND* è il controller di default ma non è l'unico: è possibile scegliere tra altre porte logiche, come *OR* (che trasmette un segnale se almeno uno dei sensori in ingresso lo sta trasmettendo), oppure *Expression* che verifica un'espressione (formulata in linguaggio Python) o, ancora, *Python,* che esegue uno script o un modulo Python.




*I sensori Always e Keyboard collegati in AND*

> *Nota a margine: il segnale trasmesso è, di default, positivo, o "True" per gli informatici, ma un sensore può trasmettere anche un comando False, negativo, quando viene attivato, purché si selezioni la casella "Invert" (inverti segnale, appunto) presente nella scheda di ogni sensore.*

Le Properties o proprietà degli oggetti sono invece caratteristiche degli stessi, memorizzate come dati numerici, booleani (cioè vero o falso) o stringhe di caratteri; l'esempio tipico è il numero di vite del personaggio, memorizzato appunto mediante un numero da associare ad un elemento.

La scheda delle Proprietà si trova a sinistra nel Logic Editor e il funzionamento di tali elementi è abbastanza semplice: per associare una proprietà numerica al cubo, ad esempio, clicchiamo su Add Game Property e personalizziamo la scheda proprietà appena creata.

*Add Game Property*

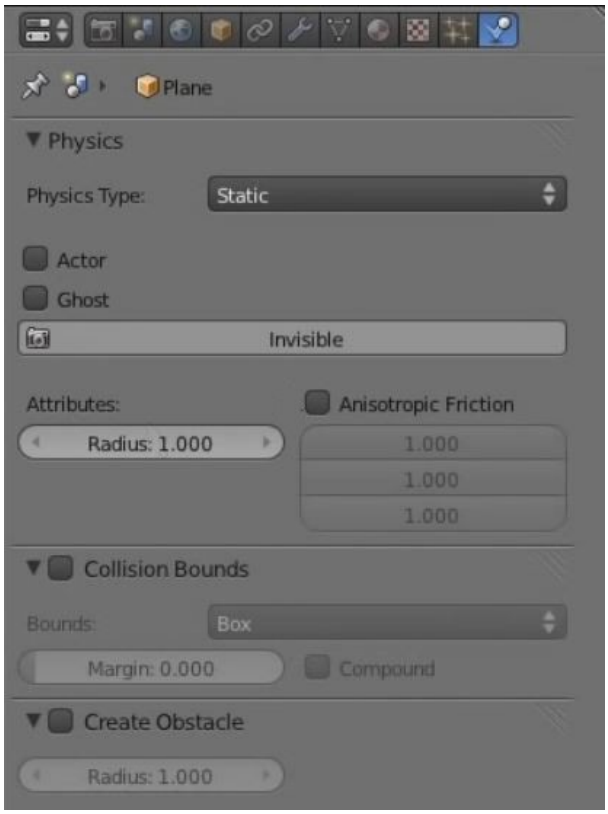
In questa puntata non lo vedremo ma è possibile, ovviamente, ricavare o modificare i valori delle Properties mediante gli attuatori, oltre che generare degli eventi se le proprietà assumono determinati valori (ad esempio, terminare il gioco se finisce il tempo o il giocatore perde tutte le vite).

Questa è la punta dell'iceberg riguardo a proprietà, sensori, controller e attuatori ma in questa puntata la panoramica introduttiva su questi argomenti si ferma qui per passare ad un'altra introduzione: quella relativa alla fisica nel GE; sempre in questa puntata, per finire, vedremo un esempio pratico che riguarderà interazione e fisica.

*  *  *

Eccoci alla seconda parte di questo tutorial: gli elementi di base della fisica degli oggetti nel GE.

Come avrete notato, di base gli elementi sono statici, ossia non sono soggetti a forze, nemmeno alla forza di gravità (che pure è presente). Le informazioni che riguardano questi aspetti si trovano principalmente (ma non solo) nella scheda Physics, all'interno della Properties Window, che con l'Engine Blender Game mostra strumenti diversi da quelli presenti con Blender Render.



*La scheda Physics nel Game Engine*

La prima voce da esaminare è quella che determina la natura fisica dell'oggetto: il menù Physics Type, il "tipo fisico" appunto, che ha voce di default Static.

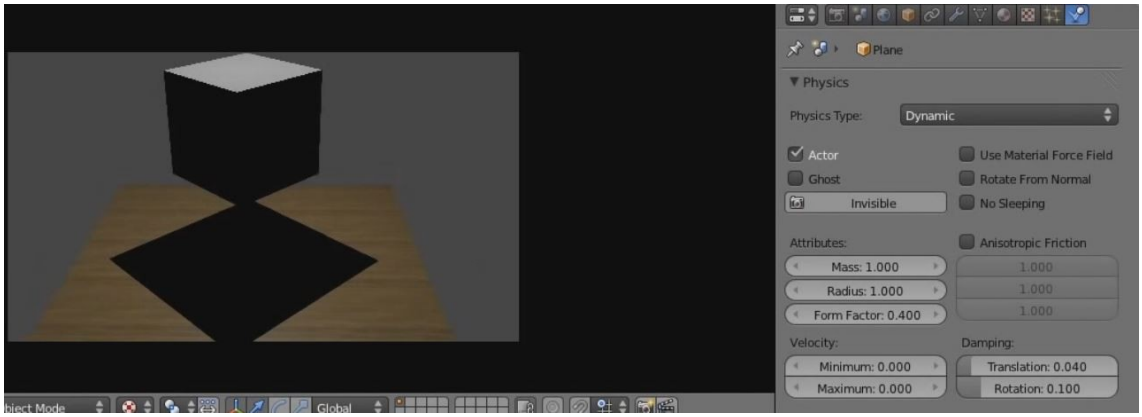
Un oggetto Static può essere mosso mediante attuatori, infatti abbiamo spostato il cubo, ma questo non reagisce a forze e impatti. Static è il tipo di default e, per inciso, anche il Plane che fa da pavimento per il cubo è di tipo Static, infatti non cade nel vuoto. La funzione di Static non è solo quella di fare da pavimento o comunque da contorno immutabile, ma anche di risparmiare risorse a tempo di esecuzione: se un oggetto non può essere spostato, raggiunto o distrutto (ad esempio, un vaso su un mobile), è inutile farlo partecipare alla simulazione dinamica, per cui tanto vale lasciarlo Static.

Il tipo base per la simulazione dinamica è, appunto, Dynamic: un oggetto Dynamic è soggetto a forze, quindi precipita se lasciato nel vuoto, e reagisce agli urti, in particolare rimbalzando, traslando o scivolando, non rotolando; implementa, cioè, la "Linear Physics".

Se ad esempio rendiamo il cubo Dynamic e lo posizioniamo inizialmente ad una certa quota rispetto al piano, lo vedremo cadere all'inizio dell'esecuzione del gioco (e, ovviamente, impostando come Dynamic anche il Plane, li vedremo precipitare entrambi nel vuoto, mentre impostando il Plane come "No Collision", cioè esente da qualsiasi tipo di collisione, vedremo il cubo passare attraverso il Plane e precipitare nel vuoto; No Collision va bene per oggetti che non possono essere raggiunti, come oggetti di sfondo, per risparmiare risorse a runtime).

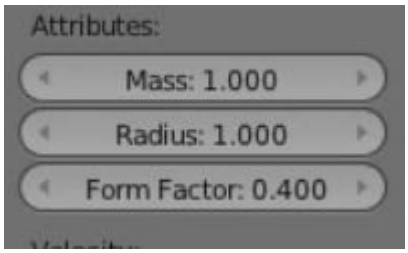
Il passaggio a Dynamic rende disponibili anche altre parametri sia nell'attuatore Motion che all'interno della stessa scheda Physics.

Per quanto riguarda Motion, abbiamo dei campi (Force e Torque) che ci permettono di impostare le trasformazioni come forze, non come semplici traslazioni o rotazioni immediate, per cui ad esempio cambiando l'attuatore di movimento su Y da Location a Force su Y, cambiando i parametri nei relativi campi, durante l'esecuzione dovremo tenere premuta la freccia in su per muovere il cubo gradualmente e l'entità dello spostamento aumenterà sempre più finché terremo premuto il tasto (in pratica, come premere sull'acceleratore di un'automobile).



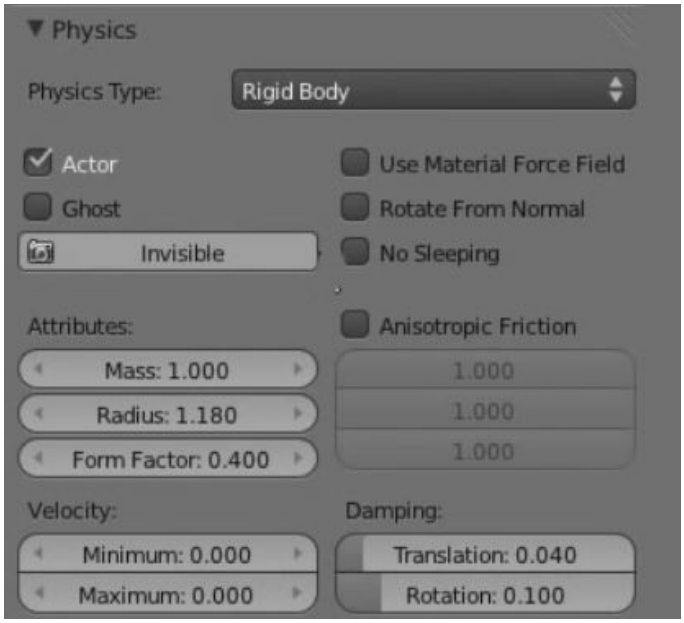
*Dynamic Motion*

Nella scheda Physics per il momento notiamo solo, tra i nuovi parametri presenti, Mass, che indica appunto la massa dell'oggetto (importante per valutare le forze da applicare) e Radius, che si riferisce al raggio della sfera (invisibile in tempo reale e visualizzata come un cerchio tratteggiato nella 3D View) che serve a rilevare le collisioni per l'oggetto, almeno fino a quando non si impostano forme ad hoc per rilevare le collisioni in Collision Bounds, in basso nella scheda.



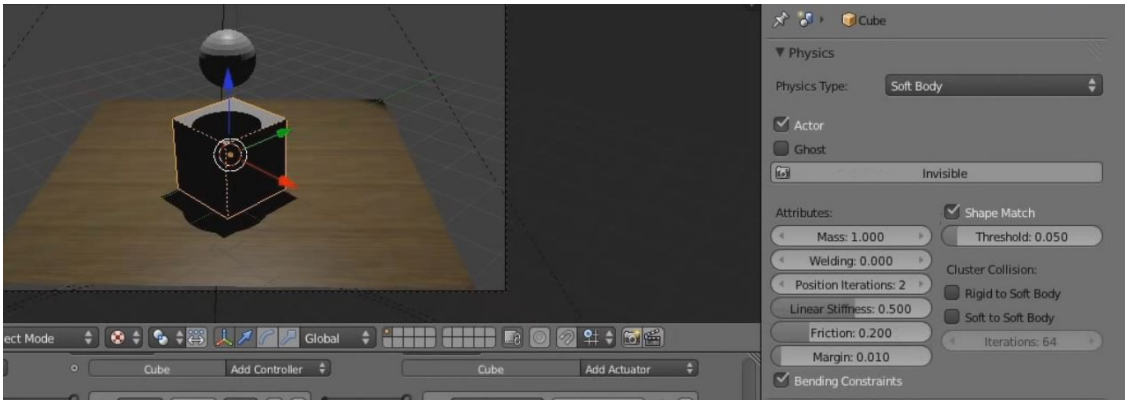
*Mass e Radius*

Per implementare, oltre alla Linear Physics, anche la Angular Physics (ossia le rotazioni a seguito di forze, collisioni o altro), dobbiamo scegliere il tipo Rigid Body. È un tipo distinto da Dynamic perché non tutti gli oggetti hanno bisogno di questo tipo di simulazione fisica, che richiede più risorse rispetto a Dynamic, quindi se possibile si sceglie Dynamic, altrimenti (ad esempio, per far rotolare una sfera) si fa ricorso a Rigid Body.



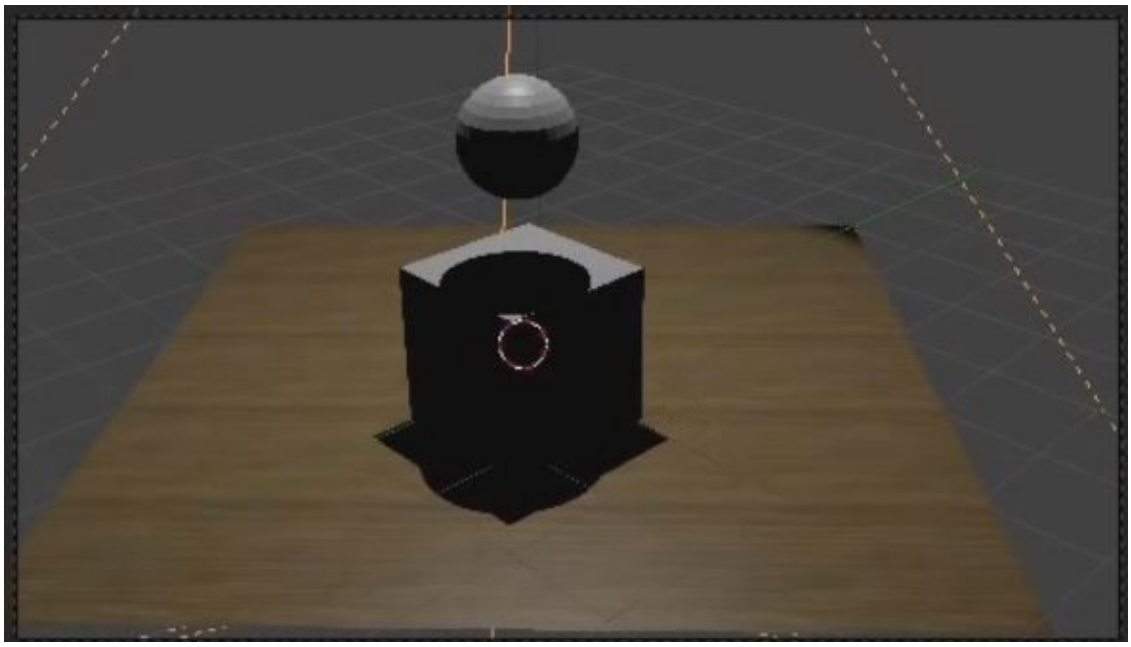
*Rigid Body*

Soft Body è il tipo più complesso e che richiede più risorse: consente la deformazione a runtime degli elementi, ad esempio la deformazione di un oggetto in seguito ad un urto o altre simulazioni fisiche. Richiede molte risorse ed è in grado di rallentare sensibilmente l'esecuzione del programma; è ottimo per realizzare animazioni da renderizzare poi col rendering classico partendo da simulazioni fisiche interattive.




*Soft Body*

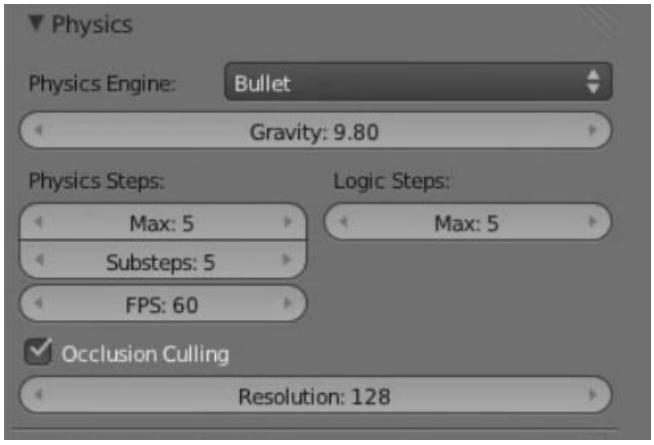
Facciamo un esempio pratico: rendiamo il Plane Static, suddividiamo il Cubo un bel po' di volte e rendiamolo un Soft Body, quindi posizioniamo sopra al cubo, sospesa nel vuoto, una sfera, da rendere di tipo Dynamic.



*La scena d'esempio per Soft Body*

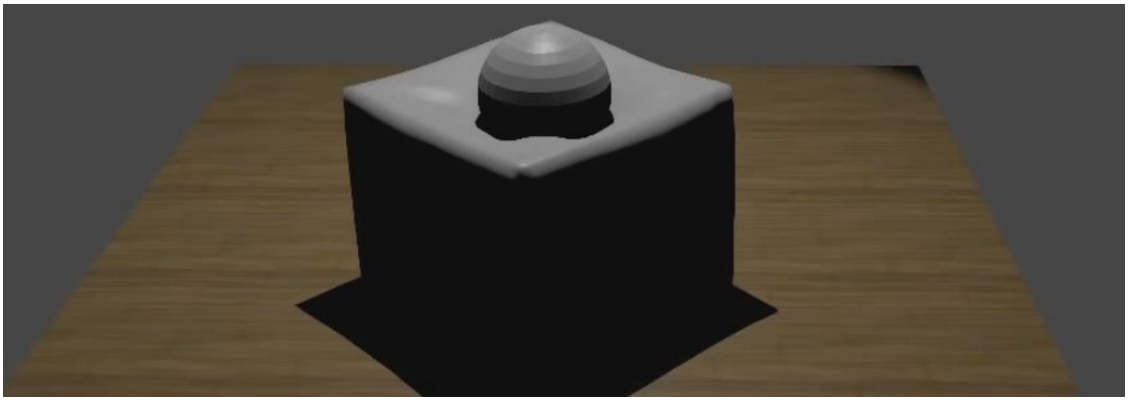
Avviando l'esecuzione, molto probabilmente vedremo la sfera rimbalzare sul cubo ma far precipitare questo al di sotto del Plane utilizzato come pavimento... un comportamento anomalo dovuto ai pochi calcoli di simulazione fisica realizzati di default dal Game Engine a runtime, ancora una volta (tanto per cambiare) per rendere l'esecuzione del gioco più fluida a scapito dell'accuratezza. Per risolvere questo problema possiamo aumentare il valore del campo Physics

Substeps nella scheda World, all'interno della Properties Window, ed osservare il comportamento a runtime con valori differenti.

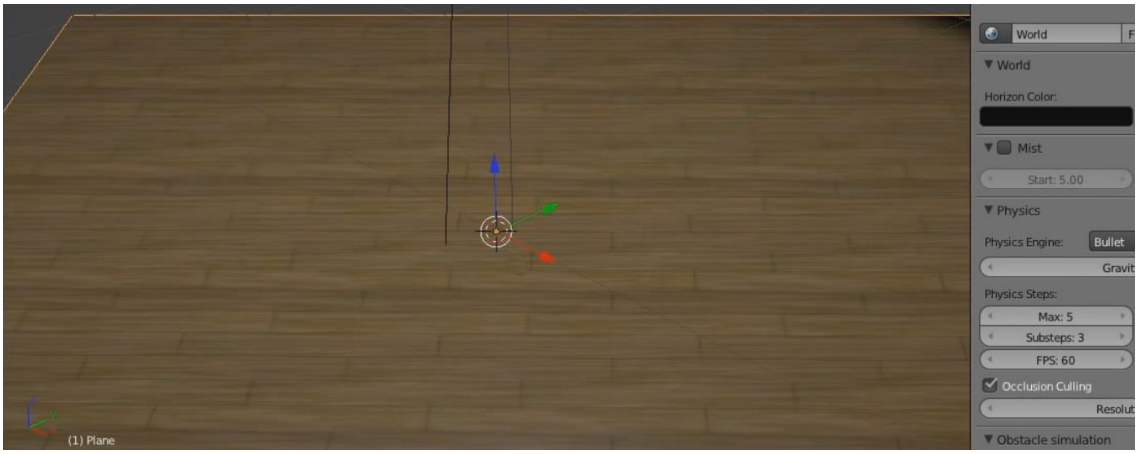


*Physics Substeps*

Valori elevati produrranno una simulazione fisica più accurata ma anche rallentamenti sensibili, a volte non tollerabili, durante l'esecuzione dell'applicazione, per cui si tratta di trovare il giusto compromesso tra qualità e velocità.


*Screenshot del risultato di Soft Body con più Substeps*

Cancelliamo sia il cubo che la sfera e lasciamo, per il momento, solo il Plane, Static, da utilizzare come pavimento per l'esempio pratico finale. Associamogli un Material e una Texture, giusto per non lasciare il solito Material grigio, e cambiamo anche il colore di World, ad esempio in nero.



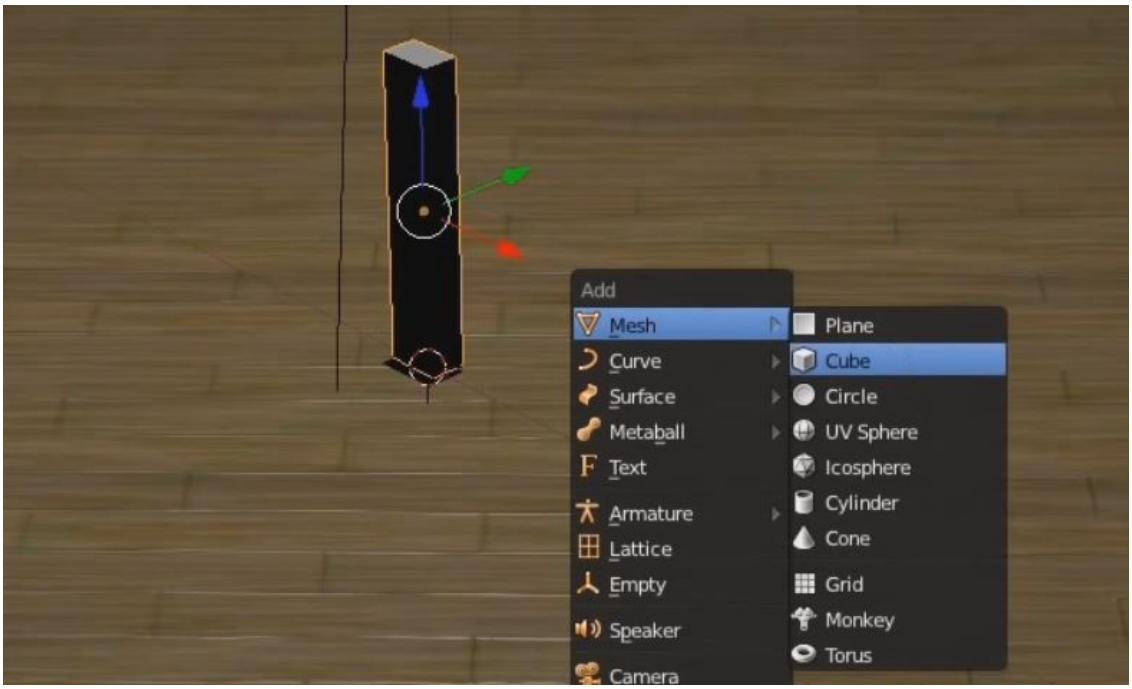
*Il Plane, con World impostato a nero*

Fisseremo meglio le differenze tra Static, Dynamic e Rigid Body tra pochissimo, realizzando un esempio pratico passo per passo, dove inizialmente sbaglieremo di proposito in modo da ricordare meglio in futuro quale tipo fisico scegliere a seconda dell'effetto che vogliamo ottenere.

> *NOTA importante: alcune proprietà fisiche degli oggetti vanno impostate nei loro Materials, ma in questa puntata non tratteremo questo argomento, che vedremo quindi un'altra volta.*

È giunto il momento di passare ad un esempio pratico per mettere insieme gli elementi discussi in questo tutorial; in particolare, realizzeremo un effetto domino mediante una sfera e alcuni parallelepipedi, rappresentanti appunto le tessere del domino.

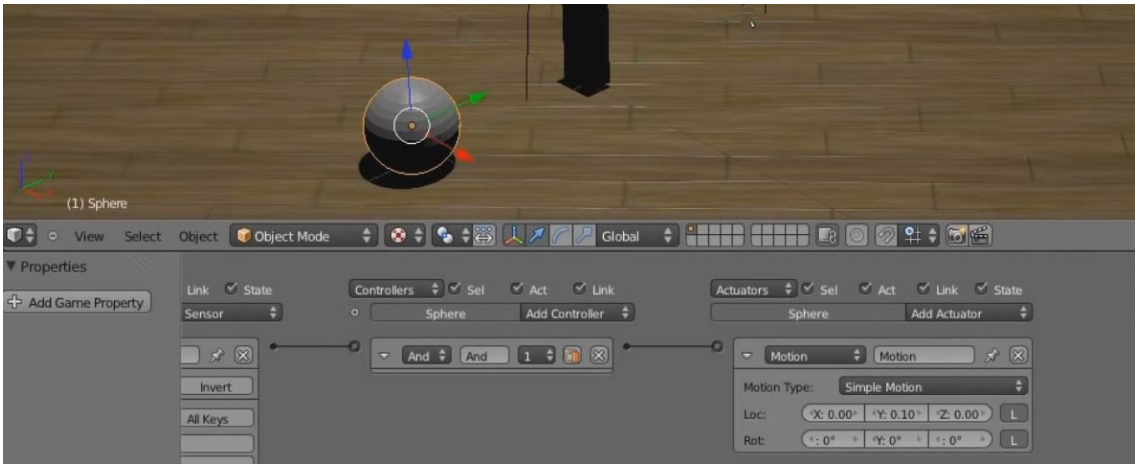
Inseriamo nella scena un nuovo cubo, che di default sarà di tipo Static, posizionandolo sul Plane-pavimento e scalandolo lungo i vari assi in modo da fargli assumere la forma di una tessera del domino in verticale; per risparmiare tempo non fornirò Textures UV Mapped ai pezzi, ma solo un Material di default (d'altronde non è una cosa importante per questo esempio).




*Realizzare la prima tessera del domino*

Di fronte al pezzo appena creato, inseriamo (poggiandola sul pavimento) una sfera. Anche questo oggetto sarà, almeno all'inizio, di tipo Static.

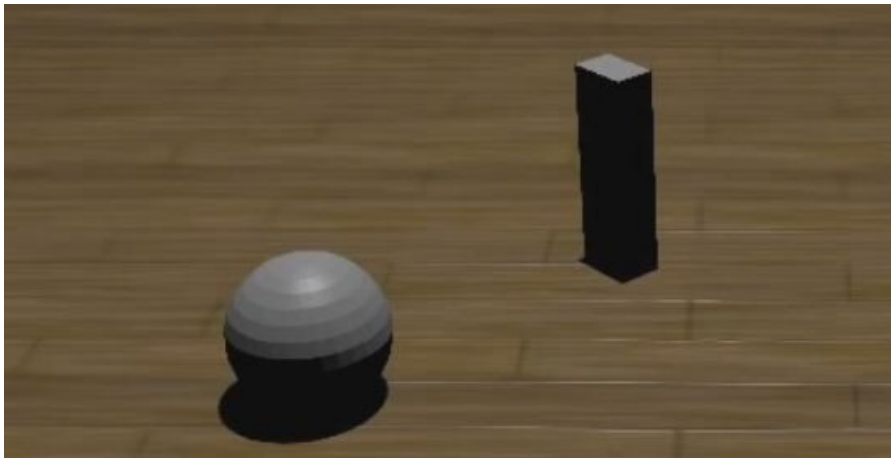
Dotiamo la sfera di sensore Keyboard con tasto da rilevare la barra spazio, attuatore Motion con spostamento lungo l'asse locale diretto verso il pezzo del domino e colleghiamo i due elementi con un semplice controller *AND*.




*La sfera con sensore Keyboard e attuatore Motion*

Avviamo l'esecuzione: premendo la barra spazio, e tenendola premuta, sposteremo la sfera (che traslerà, non rotolerà) fino a farla sbattere con il pezzo del domino, ma a quel punto la sfera passerà attraverso il pezzo e continuerà a spostarsi fintantoché terremo premuta la barra spazio.

Ormai sappiamo che per spostare gli oggetti in seguito a urti o forze dobbiamo renderli almeno di tipo Dynamic, per cui cambiamo la natura del pezzo del domino in Dynamic, avviamo il gioco e... vediamo che succede.

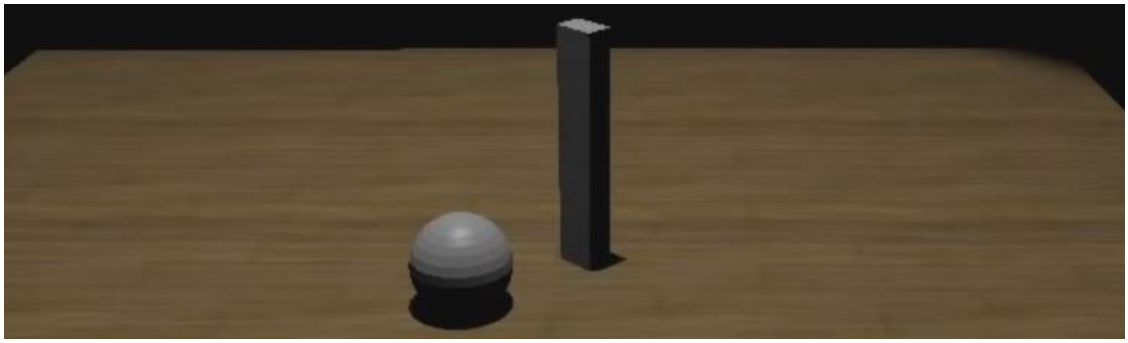


*Screenshot del risultato a video*

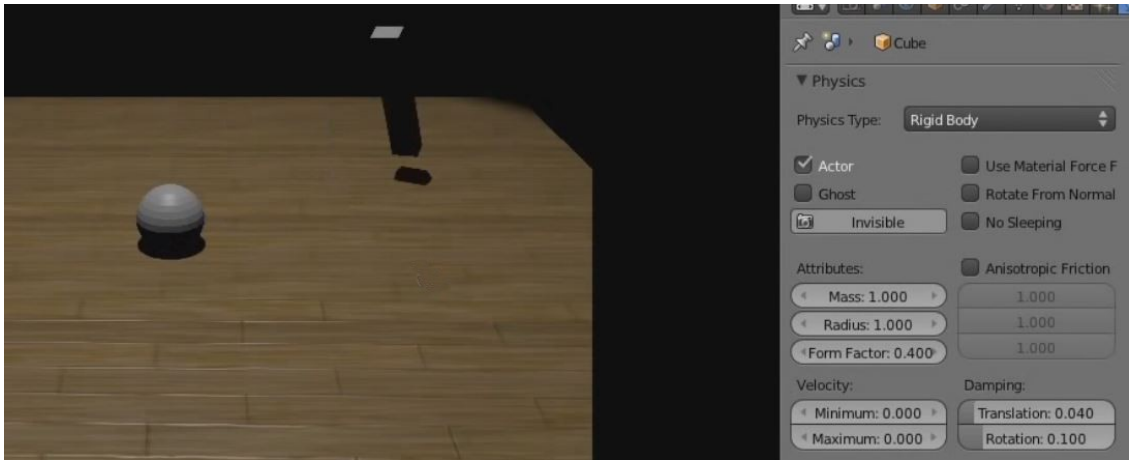
Il pezzo, all'inizio dell'esecuzione, è caduto un po', rimanendo a metà all'interno del pavimento: perché? Perché il bound iniziale utilizzato per rilevare le collisioni è la sfera, indicata col cerchietto tratteggiato nella 3D View, che ha diametro inferiore alla dimensione massima del parallelepipedo e, comunque, forma diversa, per cui quello che dobbiamo fare è attivare la casella Collision Bounds nella scheda Physics del parallelepipedo, all'interno della Properties Window, impostando Box come "forma" per rilevare le collisioni.

*Collision Bounds - Box*

Eseguiamo nuovamente il gioco: adesso ci siamo.

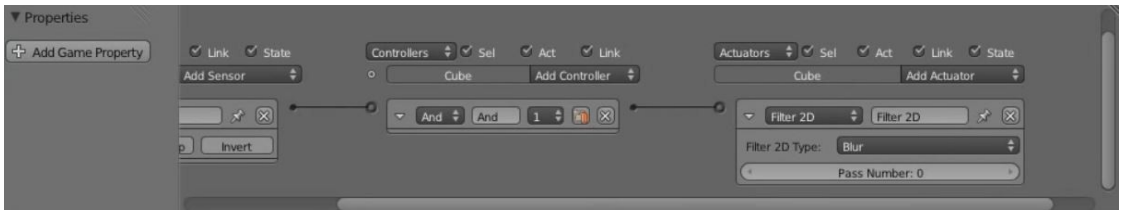

*Risultato corretto (facendo scivolare la tessera)*

C'è da dire però che, facendo sbattere la biglia con il pezzo del domino, l'urto farà scivolare il pezzo sul piano, senza farlo cadere; per implementare quest'effetto, il pezzo del domino deve diventare un corpo rigido, non un semplice Dynamic, in modo da attuare la Linear Physics, per cui cambiamo la sua natura in Rigid Body e riproviamo (NOTA: se prima abbiamo lasciato un valore alto per Physics Substeps, possiamo mettere nuovamente 1, perché in questo caso va più che bene e non rallenta inutilmente l'esecuzione del gioco).



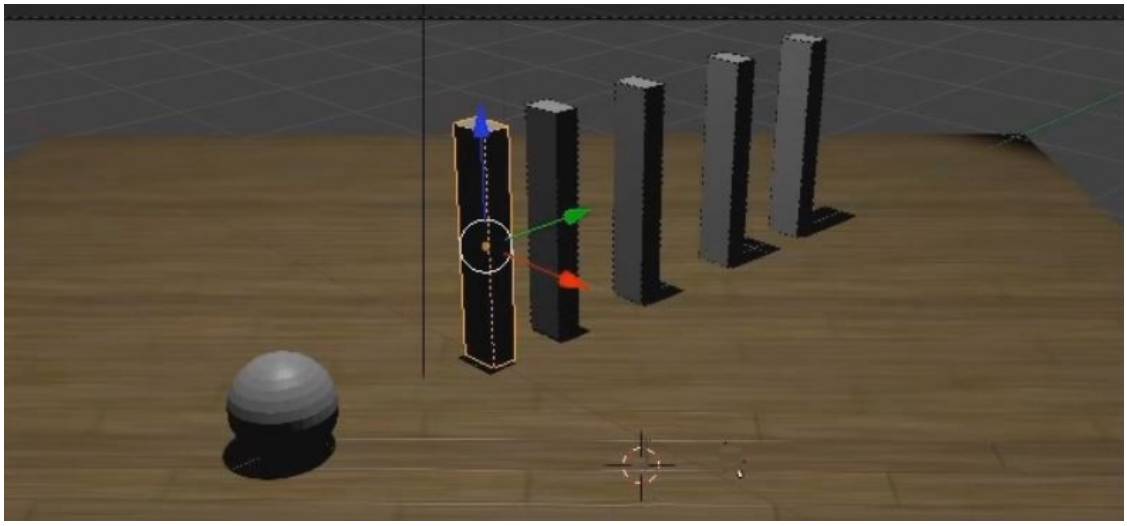

*Rigid Body per la tessera del domino*

Prima di procedere, aggiungiamo un sensore Always e un attuatore Filter 2D, con filtro di Blurring, al parallelepipedo che rappresenta il pezzo del domino, in modo da implementare una sfocatura permanente sugli oggetti, che tornerà utile quando saranno in movimento.



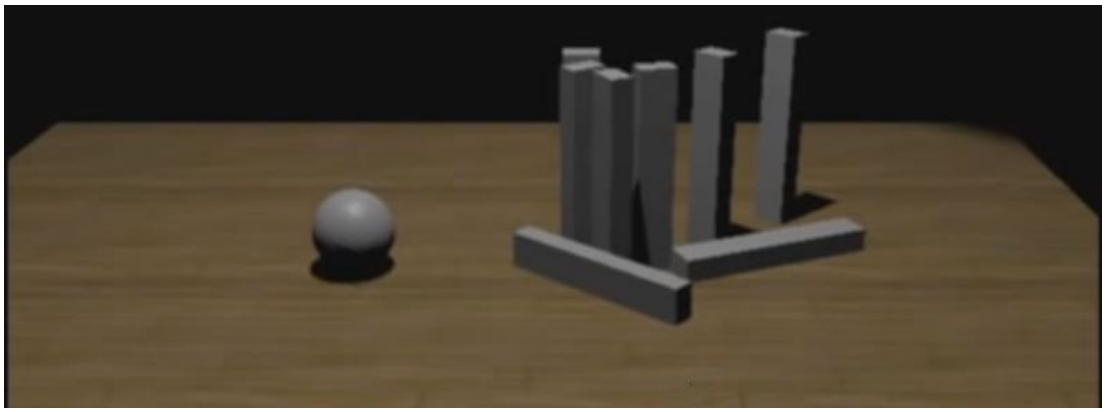

*Sfocatura da movimento (filtro Blurring)*

Adesso con il movimento del pezzo ci siamo e possiamo iniziare a posizionare altri pezzi, realizzati con semplici copie in Object Mode mediante SHIFT D e posizionati con semplici spostamenti sul piano X e Y e rotazioni intorno all'asse Z.



*Creare e posizionare le altre tessere del domino*

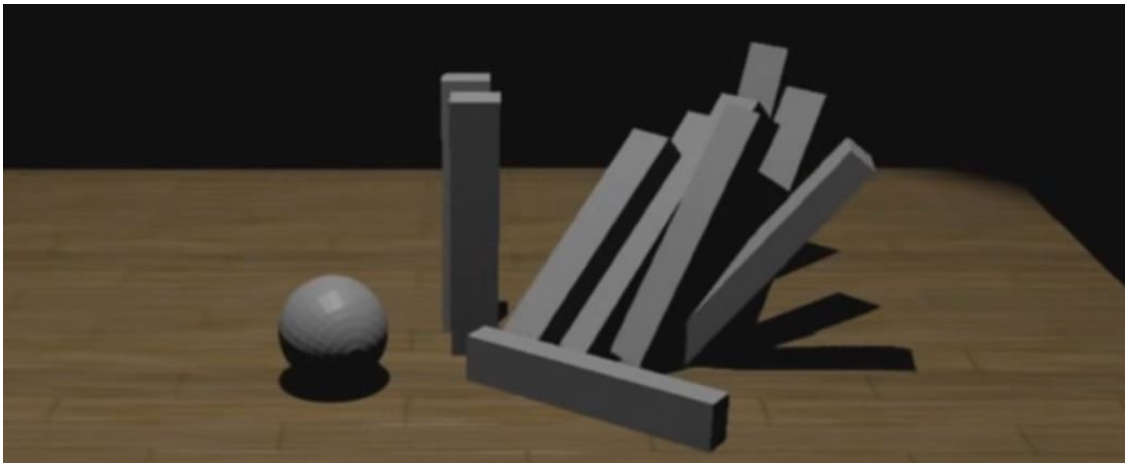
L'ultima cosa da sistemare è il movimento della biglia, che si sposta con intensità fissa e senza rotolare. Per quanto riguarda la biglia, visto che ci serve la sua rotazione cambiamo subito la natura in Rigid Body, quindi portiamo a 0 il valore di Location per l'attuatore Motion ed impostiamo un buon valore (da determinare facendo delle prove) per Force.



*Screenshot dell'effetto finale*

Ora che la biglia è un Rigid Body, può reagire all'urto con il pezzo del domino tornando indietro, senza far cadere il pezzo, per cui magari aumentiamo il valore della forza da applicare in seguito alla pressione della barra spazio.

Faccio notare che probabilmente dopo la prima collisione la biglia non si muoverà più nella direzione precedente, se premeremo ancora la barra spazio; ciò è dovuto al fatto che la forza viene applicata sull'asse Y locale della biglia, oggetto che ora può orientarsi diversamente nello spazio in seguito alle collisioni; vi invito quindi a fare delle prove con gli assi globali oppure con Torque anziché Force.



*Altro screenshot dell'effetto finale*

Concludo questa puntata con uno spunto: con le conoscenze acquisite finora, provate a realizzare un giochino dove l'utente può lanciare una sfera con una traiettoria parabolica, con forze diverse, con lo scopo di abbattere dei pilastri o delle strutture poste ad una certa distanza. Nella prossima puntata riprenderò il corso proprio da questo esempio, mostrando tra l'altro come fare per attivare certi attuatori solo in condizioni particolari (nel nostro caso, per evitare di spostare ulteriormente la sfera mentre è in volo, applicando cioè la forza di propulsione solo quando questa è a terra).

*  *  *

# Le basi del Game Engine, il motore di gioco di Blender 2.6 – 4

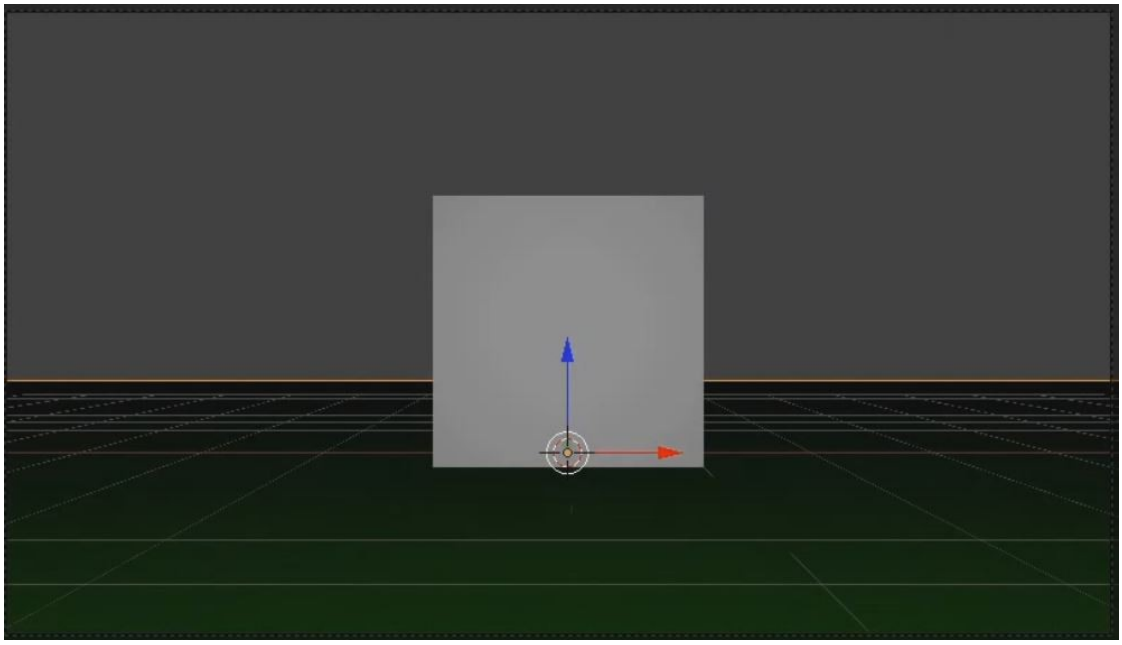
Eccoci alla quarta puntata del corso sulle basi del Blender GE (il Game Engine, motore di gioco di Blender 3D 2.6); questa sarà una puntata alquanto ricca, infatti tratteremo pivoting, parenting, properties, inserimento di oggetti nella scena e constraints, il tutto partendo dallo spunto dato nella puntata precedente: lanciare un proiettile per demolire una struttura.

Se avete realizzato un gioco partendo da quanto visto nelle puntate precedenti, probabilmente avete inserito uno o più oggetti nella scena e li avete lanciati utilizzando uno o più tasti; tuttavia, è molto più interessante creare un "emettitore", come un cannone o una pistola, e sparare di volta in volta dei proiettili creati per l'occasione.

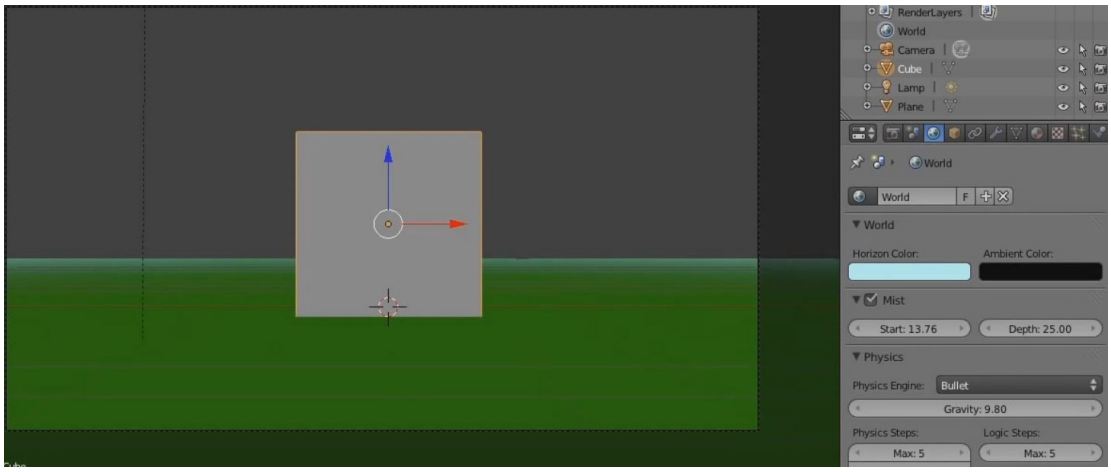
Per far ciò, utilizzeremo l'attuatore Edit Object, ma prima impostiamo la scena e trattiamo anche altri argomenti utili. Neanche a dirlo, passiamo a Blender Game, GLSL e ombreggiatura Textured nella 3D View della telecamera.

Iniziamo scalando un po' il Plane utilizzato come pavimento, in modo da arrivare fino all'orizzonte; magari dotiamolo di un Material di base con colore verde o verde scuro ed eventualmente una Texture, se volete, quindi cambiamo la fonte di luce presente di default in Sun, per dare un'illuminazione uniforme e senza ombre in tempo reale (in questo caso, non sono fondamentali e rallentano il sistema).

Aggiungiamo, volendo, anche un po' di nebbia, in modo da sfocare il Plane sulla distanza, e cambiamo il colore dello sfondo, World Horizon, ad esempio in bianco o celeste... beh, qualsiasi colore è da preferire al grigio di default, ecco!



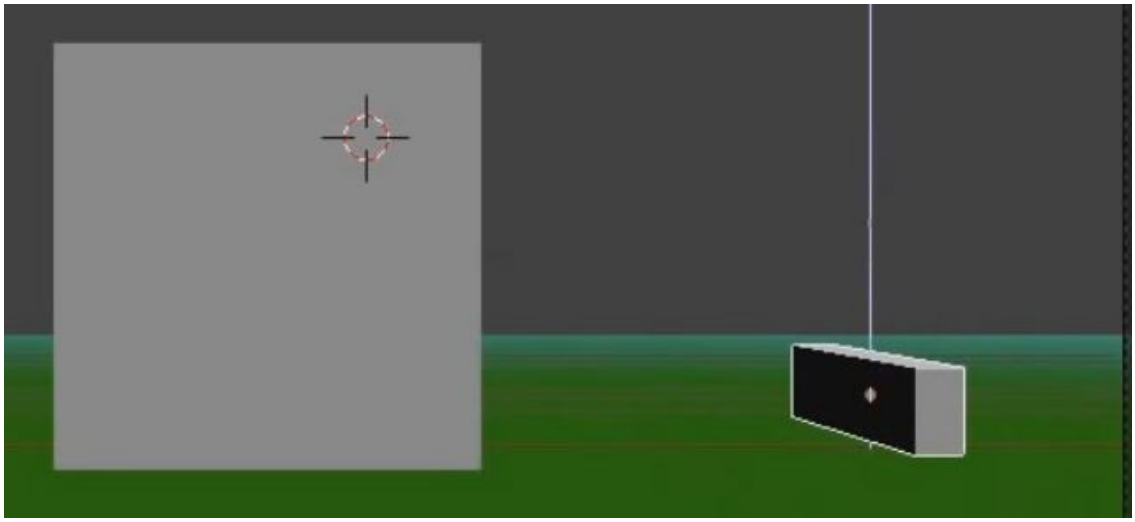
*La scena iniziale*




*MIST e World Horizon*

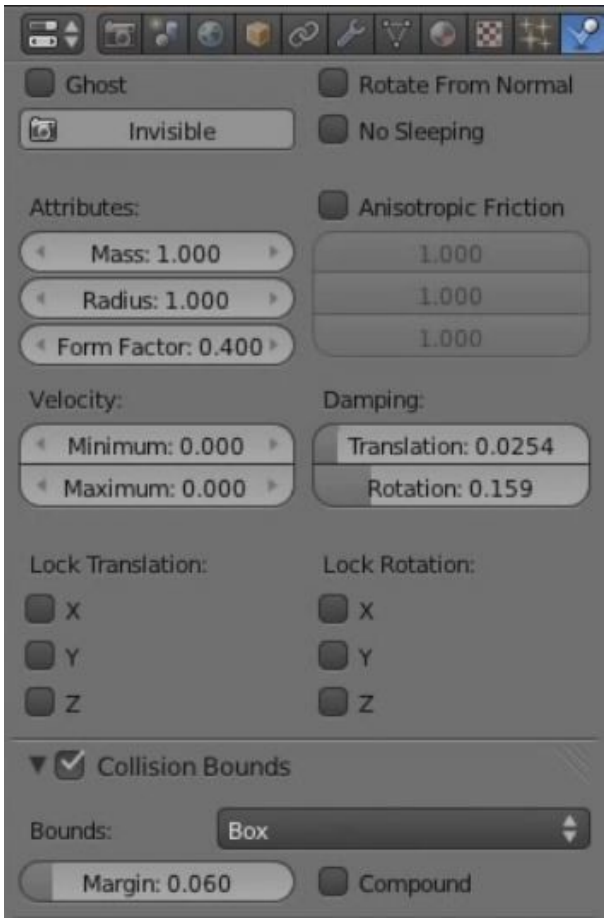
Duplico il cubo presente di default e allontano le due copie; l'idea è quella di creare un cannoncino (molto stilizzato, a dire il vero) sulla sinistra e una struttura da abbattere sulla destra. Se tali oggetti hanno dei Materials di default, eliminiamoli.

Selezioniamo quindi il cubo sulla destra e scaliamolo in maniera opportuna lungo uno o più assi, quindi poggiamolo sul pavimento.



*Cubo scalato e poggiato sul pavimento*

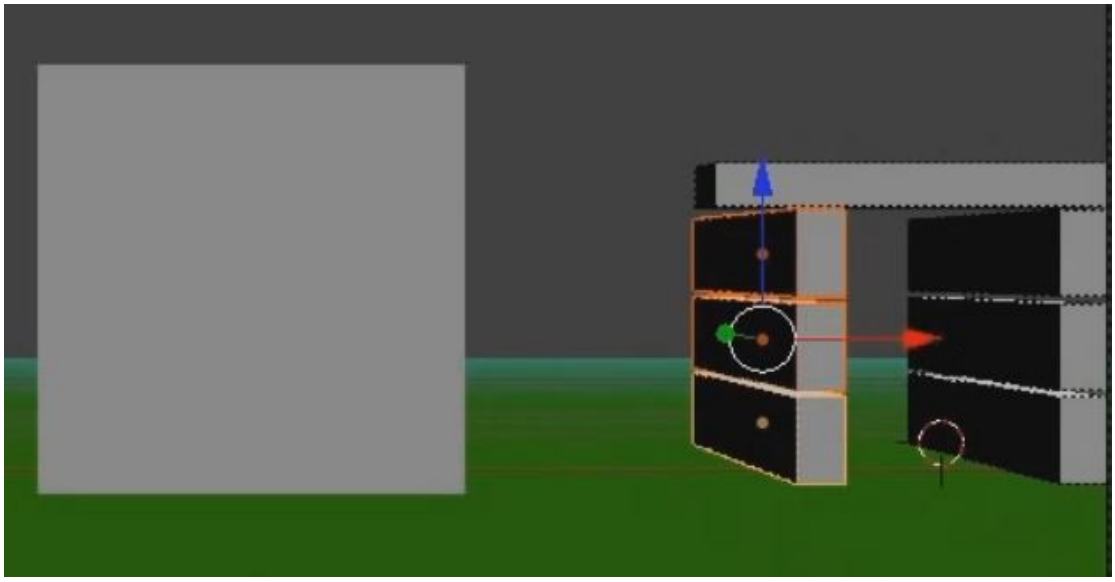
Prima di fare le copie per posizionare gli altri elementi, rendiamo questo oggetto un Rigid Body e, ricordando il caso del pezzo del domino nella puntata precedente, attiviamo i Collision Bounds in modalità Box.




*Rigid Body, Collision Bounds BOX*

Adesso facciamo volontariamente un errore, o comunque un'omissione: NON forniamo un Material a questo pezzo e passiamo subito a duplicarlo e a posizionare le copie in modo da formare una struttura.

Con l'inquadratura che sto utilizzando, è sufficiente spostare le copie sull'asse X e, poi, su Z, per realizzare la parete posteriore, impilando i vari pezzi. Attenzione a non metterli troppo distanti perché all'avvio del gioco cadranno per via della gravità, per cui cercate di poggiarli bene uno sopra l'altro in modo da limitare la caduta e non far crollare la struttura prima del tempo.

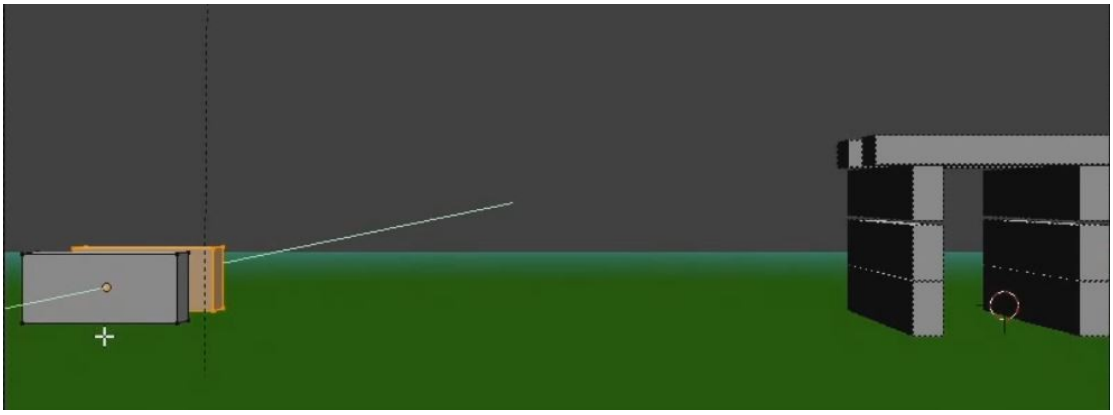


*Posizionare le copie*

Per mettere un paio di pezzi in cima, duplichiamo, ruotiamo e trasliamo.

Diamo un'occhiatina a runtime, con P, per assicurarci che i pezzi non crollino; se tutto va bene, andiamo avanti.

Riprendiamo il cubo a sinistra, scaliamolo, magari deformiamolo un po', quindi passiamo in Edit Mode, selezioniamo tutto e duplichiamo con SHIFT D, spostando un po' la copia per realizzare l'altra parete del supporto del cannoncino. Torniamo in Object Mode e aggiorniamo l'Origin dell'oggetto, quindi posizioniamo lì il 3D Cursor, in mezzo alle due pareti, con il menù Snap del cursore 3D.



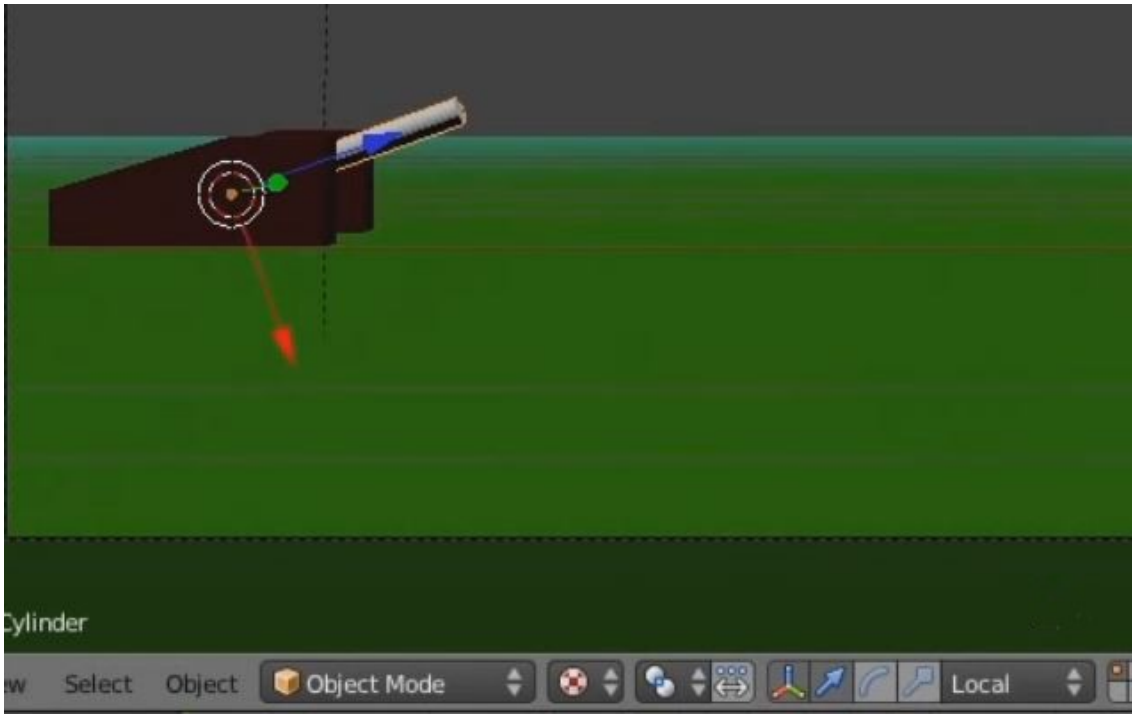
*Basi del cannoncino posizionate*

Ho realizzato la copia in Edit Mode perché in seguito sposteremo questo oggetto e, siccome non ci sono parti con movimenti indipendenti, è inutile utilizzare il parenting qui: tanto vale fare un unico oggetto.

Il parenting ci serve, invece, per il cilindro che rappresenta il cannoncino. Con il 3D Cursor nel punto giusto, al centro dei due supporti, aggiungiamo una mesh cilindro nella scena, quindi scaliamola e ruotiamola per posizionarla all'interno del supporto.

*Posizionamento del cilindro*

Passiamo alla visualizzazione del sistema di riferimento locale degli oggetti per vedere qual è l'asse uscente dalla parte finale del cannoncino: Z positivo, ricordiamocelo.



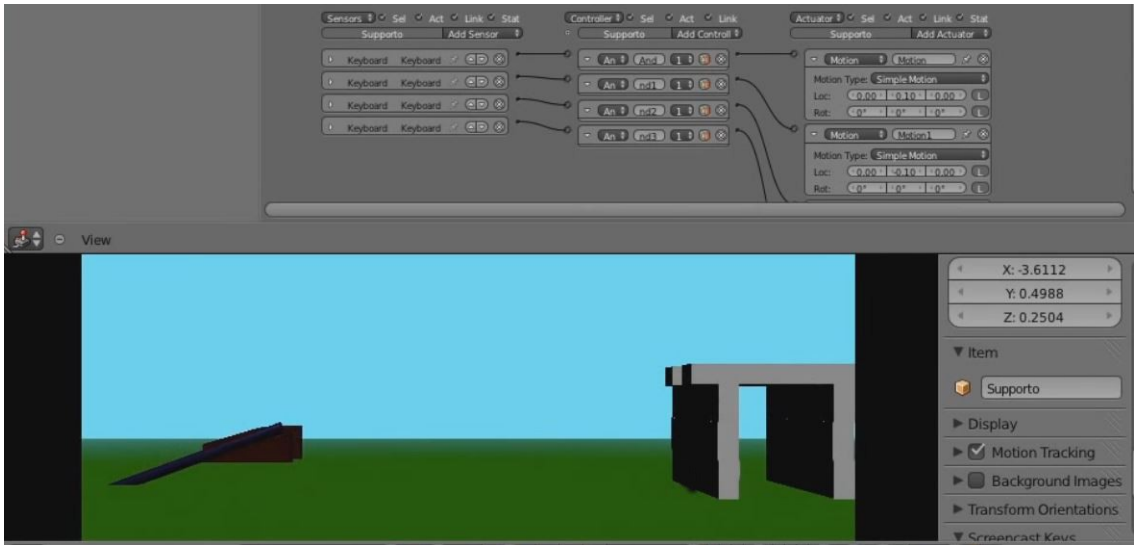

*Visualizzazione assi locali (LOCAL)*

Dotiamo cannoncino e supporto di due Materials, giusto per colorarli un po'; ovviamente, voi poi potrete utilizzare Textures per questi oggetti, per il Plane di riferimento, eccetera... diamo, comunque, nomi significativi ai Materials, in modo da poterli individuare facilmente in seguito (dovremmo dare nomi significativi anche alle mesh, in effetti, ma in questo particolare tutorial i nomi dei Materials sono importanti, vedrete presto perché).

Bene, è giunto il momento di iniziare a impostare un po' di Logic Bricks, quindi apriamo una finestra Logic Editor e, nella 3D View, selezioniamo il supporto del cannoncino, iniziando proprio da questo oggetto.

Vogliamo spostare il supporto sul piano X e Y utilizzando le frecce, quindi quello che dobbiamo fare è aggiungere 4 sensori keyboard (che rileveranno ovviamente la pressione dei tasti freccia), 4 controller *AND* (anche se possiamo collegare al volo sensori e attuatori, come sappiamo) e 4 attuatori Motion..

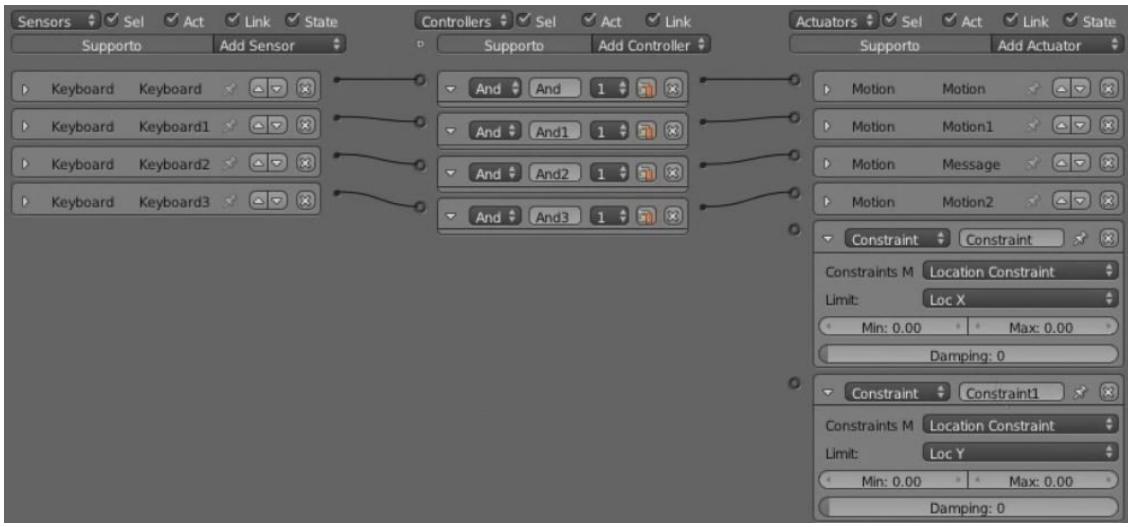
Per definire l'entità dello spostamento, in modo da non realizzare movimenti troppo piccoli o troppo accentuati, conviene fare delle prove con alcuni valori e osservando il tutto a runtime; io ho visto che 0.1 sui vari assi (X e Y locali, quindi con la L selezionata) va bene, quindi impostiamo (in questo caso) 0.1 su Y per la freccia in su, -0.1 su Y per la freccia in giù, 0.1 per X per la freccia a destra e -0.1 per X per la freccia a sinistra.
Colleghiamo sensori, controller e attuatori.



*Sensori, Controller e Attuatori per gli spostamenti*

Passando a runtime possiamo notare che gli spostamenti vengono effettuati, ma il cannoncino non è imparentato e comunque l'utente ha troppa libertà nel muovere il supporto.

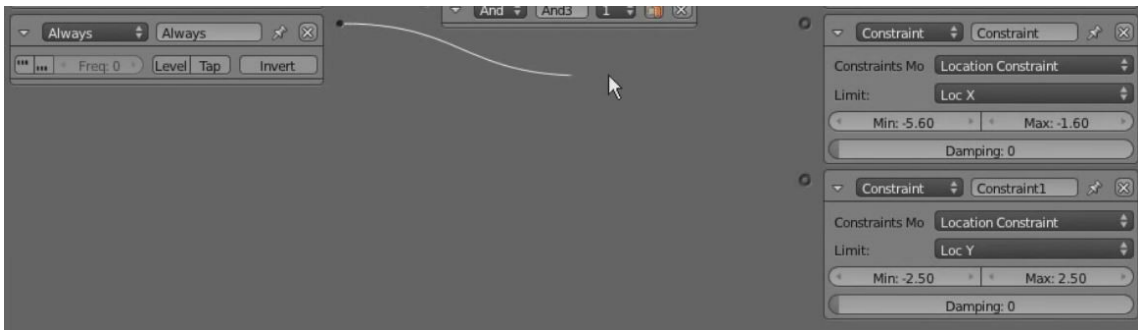
Partiamo dal secondo problema: per vincolare gli spostamenti ad un'area ristretta, abbiamo a disposizione (così come con Blender Render) i Constraint: i vincoli, appunto, per cui (sempre con il supporto selezionato) aggiungiamo due attuatori di tipo Constraint, lasciando la modalità di default Location Constraint. Abbiamo bisogno di due attuatori perché il primo lavorerà sull'asse X, il secondo sull'asse Y; non abbiamo bisogno di un constraint su Z in quanto non c'è il rischio che il supporto si sposti in verticale.




*Aggiungere i due Constraints*

Per impostare i vincoli di spostamento dobbiamo dare un'occhiata ai valori Location X e Y della mesh supporto nella 3D View; tali valori fanno riferimento all'Origin dell'oggetto e questo è un altro buon motivo per impostare correttamente l'Origin degli elementi. Decidete voi quale "Bounding Box", in un certo senso, dare al supporto, ad esempio lasciando 2 unità per direzione rispetto alla posizione iniziale. Se, ad esempio, Location X del supporto è -3, potete scrivere -5 per min e -1 per max nel Constraint per l'asse X, e così via.

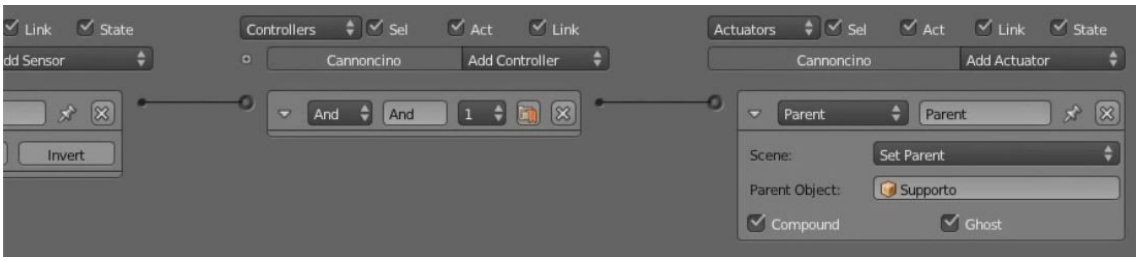
I due Constraint ci sono ma non sono attivi, in quanto non collegati ad alcun sensore; visto che ci servono sempre, tanto vale aggiungere un sensore Always. Ne basta uno solo, da collegare ad un controller *AND,* che avrà in uscita due collegamenti, diretti ai due attuatori: in questo modo, dal sensore Always partirà un segnale che verrà sdoppiato e trasmesso da *AND* ai due attuatori, consentendoci di risparmiare un controller e un attuatore.




*Aggiungere e collegare il sensore Always*

Ok, col supporto abbiamo finito, adesso passiamo al cilindro che rappresenta il cannoncino.

La prima cosa da fare è legare il cannoncino, nei suoi spostamenti, al supporto, per cui aggiungiamo un sensore Always (sempre) e un attuatore di tipo Parent, che ci consente (come suggeriscono le voci di menù) di realizzare o annullare vincoli di parentela tra oggetti in base a determinate situazioni date dai sensori.

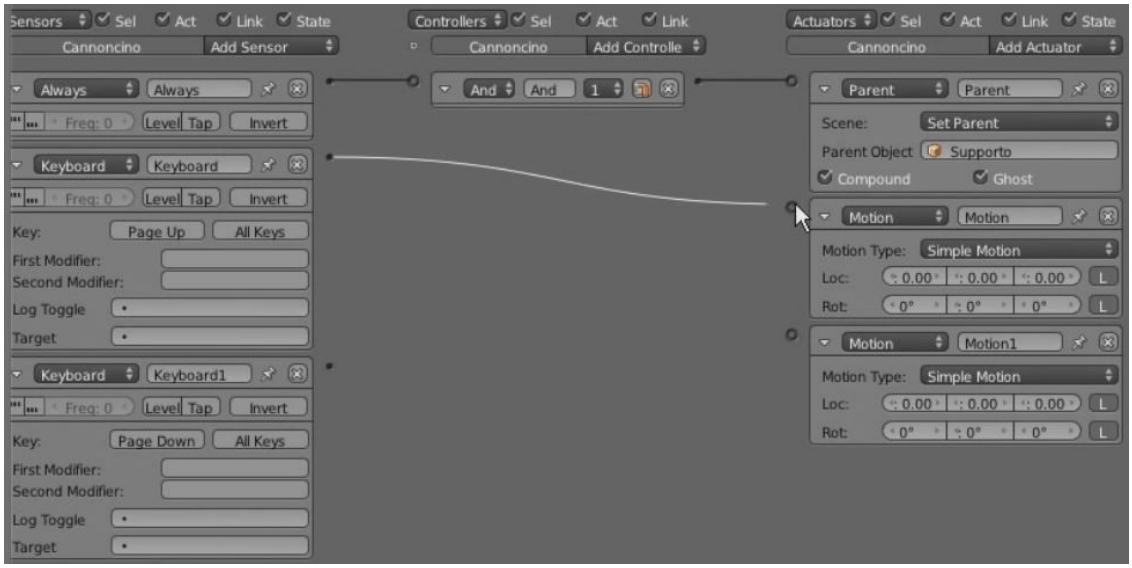



*Attuatore PARENT per il cilindro (cannoncino)*

Ci va benissimo la modalità Set Parent ("imposta genitore"), per cui colleghiamo Always a Parent e impostiamo, nel campo Parent Object, il nome della mesh supporto (non dobbiamo scriverla: facciamo click sul campo e scegliamo l'oggetto dal menù... ed ecco perché conviene dare nomi significativi alle mesh).

Diamo un'occhiata a runtime: adesso, spostando il supporto con i tasti freccia, sposteremo anche il cannoncino. Torniamo in modalità Edit per aggiungere un'altra funzionalità: la variazione dell'inclinazione del cannoncino.

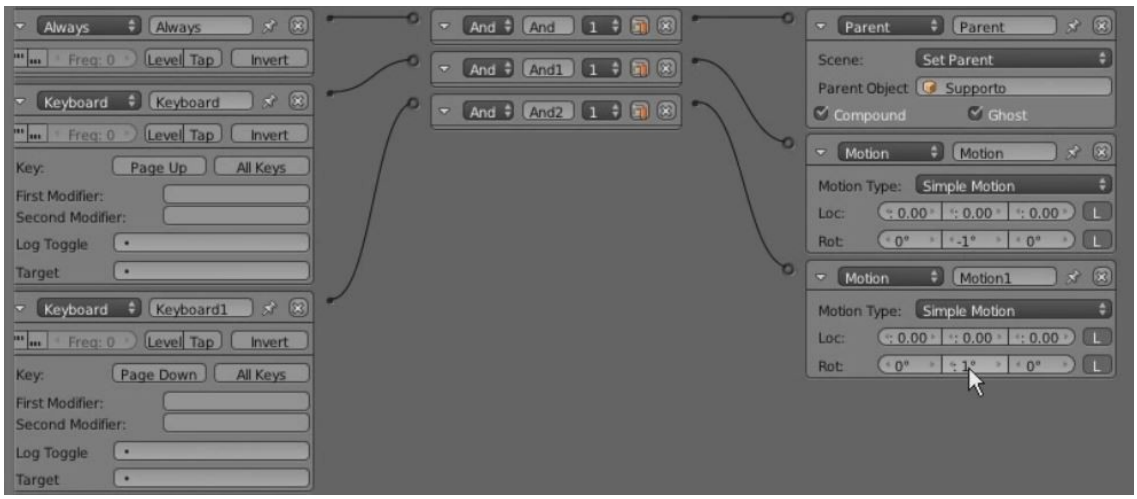
Con tale mesh selezionata, dando un'occhiata a qual è l'asse di rotazione da impostare (nel mio caso, Y locale), aggiungiamo due sensori Keyboard e due attuatori Motion.




*Asse di rotazione, sensori Keyboard e attuatori Motion*

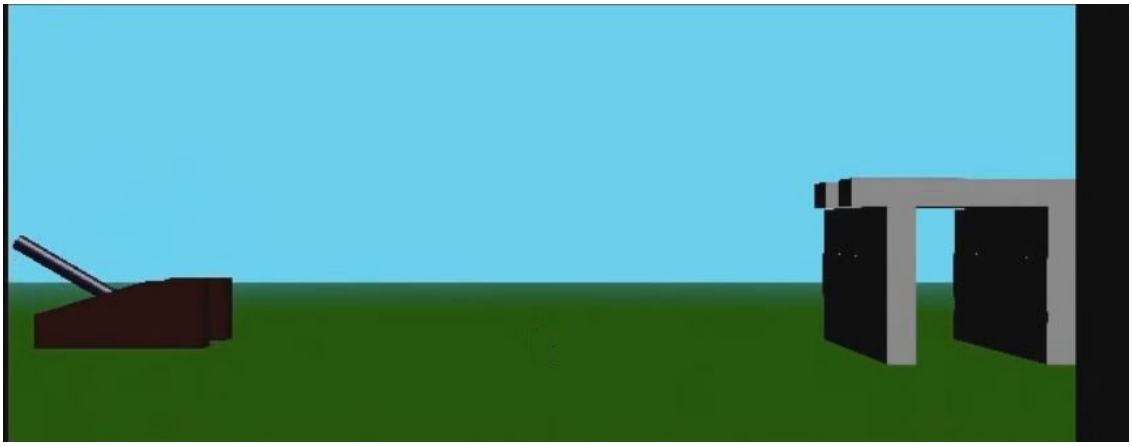
Nei sensori Keyboard impostiamo due tasti delegati alla variazione dell'inclinazione del cannone, ad esempio PagSu e PagGiù.

Nei due attuatori Motion, impostiamo ad esempio -1° per Rot Y Locale per l'attuatore associato a PagSu e +1° per Rot Y Locale per quello associato a PagGiù.



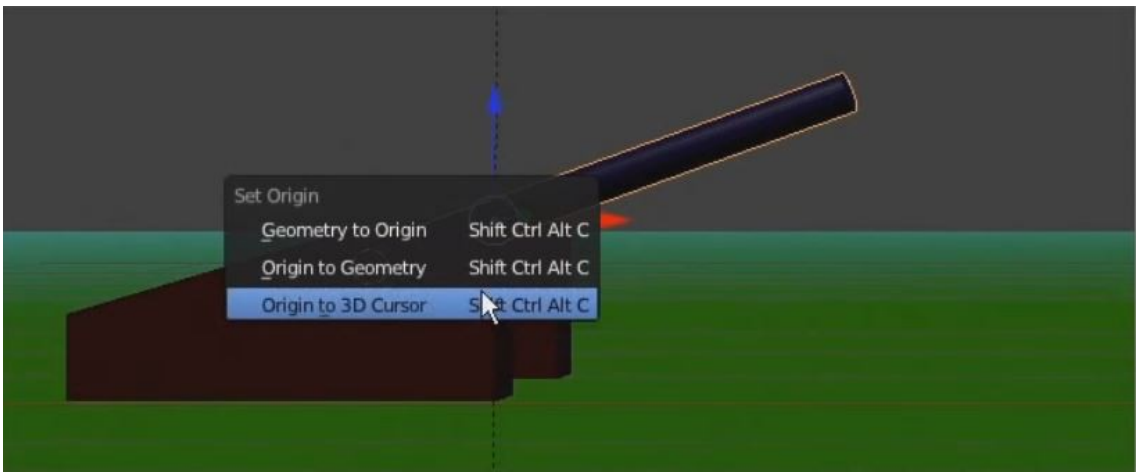

*Impostare i sensori*

Provando il tutto a runtime, ci accorgiamo che c'è un problema: il cilindro ruota intorno alla sua Origin che, in questo caso, corrisponde al centro dell'oggetto, che però si trova fuori dal supporto.



*Problemi a runtime...*

Per spostare indietro l'Origin, quindi, passiamo in Edit Mode, aggiungiamo una suddivisione con Ring Cut, spostando tale taglio indietro (in un punto coperto dal supporto), quindi spostiamo lì il 3D Cursor con il menù di Snap, torniamo in Object Mode e aggiorniamo l'Origin scegliendo Set Origin to 3D Cursor.



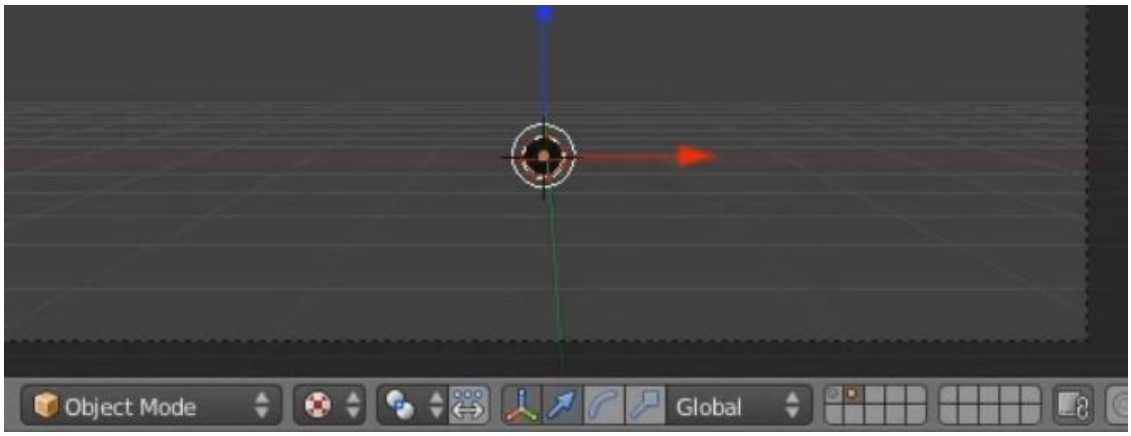

*Riposizionare l'Origin*

> *Piccolo spunto: aggiungete un Constraint Orientation Constraint per limitare l'inclinazione del cannoncino in un range credibile. Suggerimento: potete riciclare l'Always e l'AND creati per l'attuatore Parent.*

*  *  *

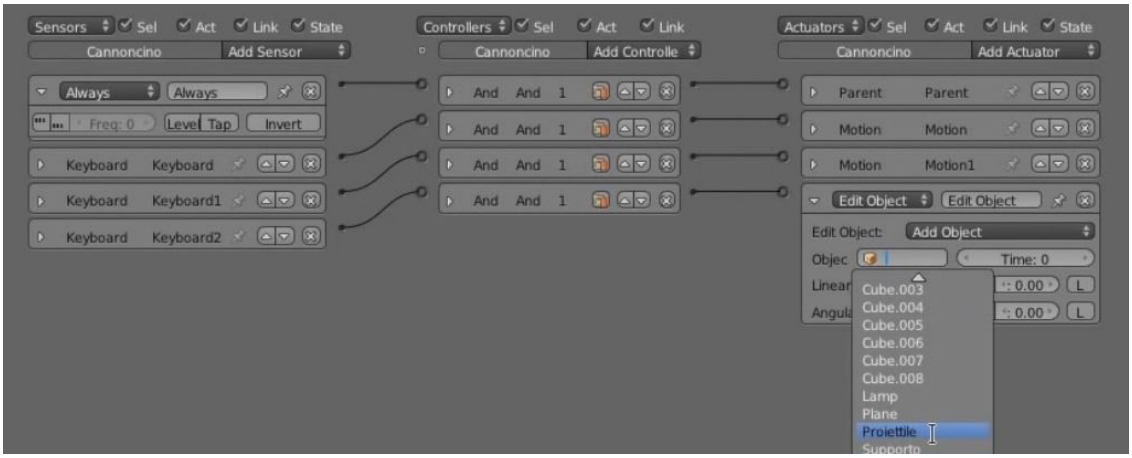
Aggiungere un oggetto a runtime nella scena non è difficile: sostanzialmente, si tratta di mettere un'istanza di tale elemento in un Layer nascosto e impostare un attuatore Edit Object per l'oggetto che dovrà immettere l'elemento nella scena.

Andiamo con ordine.
Spostiamoci nel secondo Layer della scena e aggiungiamo un oggetto, ad esempio una sfera che chiameremo “Proiettile”. Se vogliamo, dotiamolo anche di un Material e scaliamolo a piacere.




*Definire il prototipo del proiettile in un Layer nascosto*

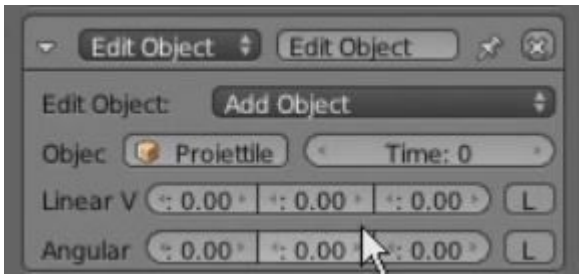
Torniamo al primo Layer (quello della scena) e selezioniamo la mesh del cannoncino.
Aggiungiamo, nel Logic Editor, un sensore Keyboard, impostando il tasto per sparare (ad esempio, la barra spazio) e un attuatore Edit Object, collegando i due elementi.




*Sensore Keyboard e attuatore Edit Object*

La modalità presente di default per tale attuatore è Add Object ("aggiungi oggetto") e ci va benissimo; come oggetto, impostiamo "Proiettile" nel campo Object.

Prima abbiamo detto che l'asse uscente dalla bocca del cannoncino è Z positivo, quindi impostiamo un valore di velocità iniziale, ad esempio 20, per tale campo nella scheda dell'attuatore, selezionando la casella L (per Local) che, di default, è deselezionata.



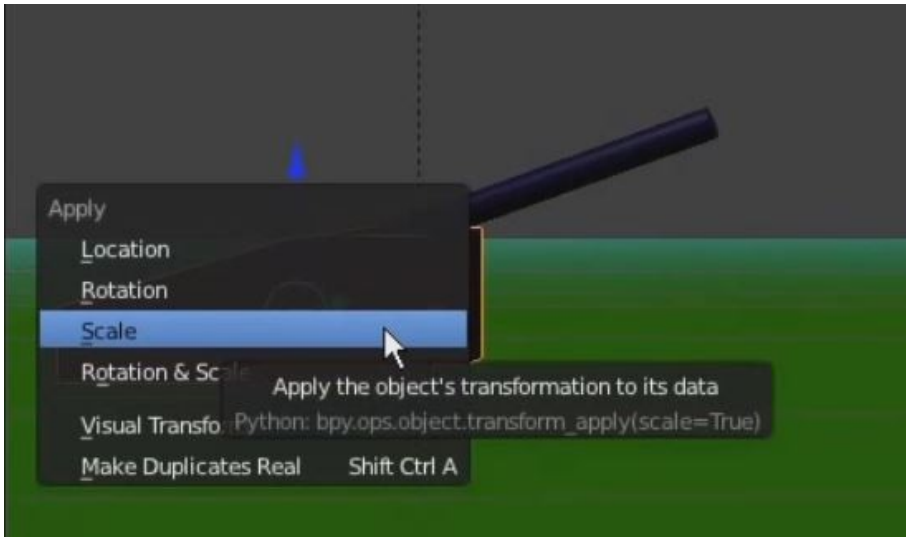
*Impostare Edit Object - Add Object*

Ansiosi di vedere il risultato, premiamo P, premiamo la barra spazio, e... sorpresa: il proiettile si materializza (deformato, tra l'altro), ma resta quasi incastrato nel cannoncino.

*Primo risultato a runtime*

Prima di continuare, risolviamo il problema della deformazione del proiettile: è dovuta allo scaling del cilindro, che è stato ridimensionato per fargli assumere la forma attuale e che trasmette tale deformazione anche agli oggetti figli o creati con Add Object.

Per risolvere il problema dobbiamo applicare la trasformazione di scaling come definitiva, per cui selezioniamo il cilindro, premiamo CTRL A e applichiamo lo scaling.

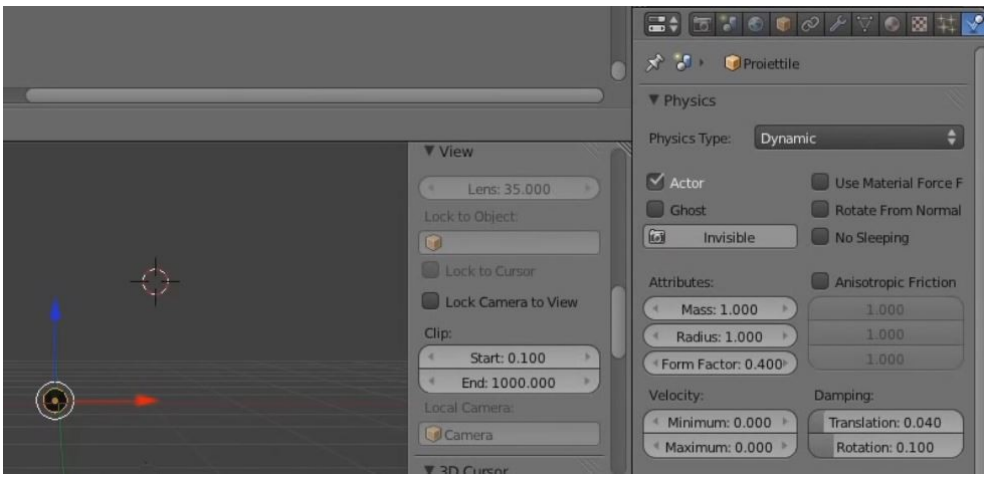



*Applicare lo Scaling*

Finito? No: dobbiamo applicare lo scaling anche per oggetti genitori del cilindro (in questo caso, il supporto del cannoncino), quindi selezioniamo tale oggetto e, di nuovo, premiamo CTRL A e scegliamo Scale dal menù che apparirà a video.

Vediamo ora come lanciare correttamente il proiettile.

La mesh proiettile è, di default (come tutti gli oggetti appena creati) Static, per cui il campo di velocità iniziale impostato in Edit Object non ha effetto su tale elemento; per risolvere il problema, passiamo al secondo Layer (dove si trova il "prototipo" del proiettile), selezioniamo questo oggetto e cambiamo la sua natura in Dynamic nella scheda Physics, all'interno della Properties Window, cambiando i Collision Bounds in Sphere oppure ridimensionando il Radius della sfera delle collisioni.




*Rendere Dynamic il prototipo del proiettile*

Altra cosa da fare prima di provare il gioco, visto che ora il proiettile è Dynamic, è rendere "No Collision" sia il supporto che il cannoncino.

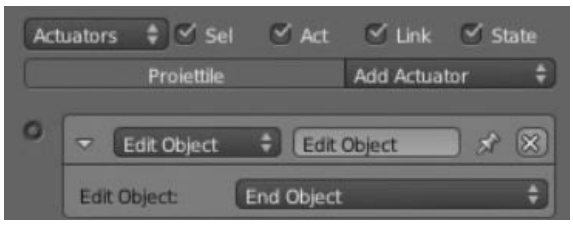
Proviamo nuovamente il gioco: ok, adesso ci siamo! Se la velocità o l'effetto della collisione sono eccessivi, personalizzate i valori di velocità iniziale in Edit Object o la massa del proiettile.

*Screenshot dell'esecuzione*

Notiamo, comunque, che il proiettile resta nella scena dopo le collisioni: come fare per eliminarlo se (e solo se) colpisce un pilastro? Quel "solo se" è dovuto al fatto che, impostando l'eliminazione con qualsiasi collisione, toglieremmo il proiettile già al momento dello sparo, perché in un certo senso impatta col cannoncino.

L'eliminazione va impostata nel prototipo del proiettile, per cui tutti gli oggetti creati da questo prototipo avranno la stessa caratteristica; selezioniamo quindi la sfera nel secondo Layer e passiamo alla scheda Logic Editor.

Per rimuovere l'oggetto dalla scena possiamo sfruttare la modalità End Object di Edit Object, per cui aggiungiamo un attuatore Edit Object per la sfera proiettile e scegliamo End Object.




*Edit Object - End Object, per il prototipo del proiettile*

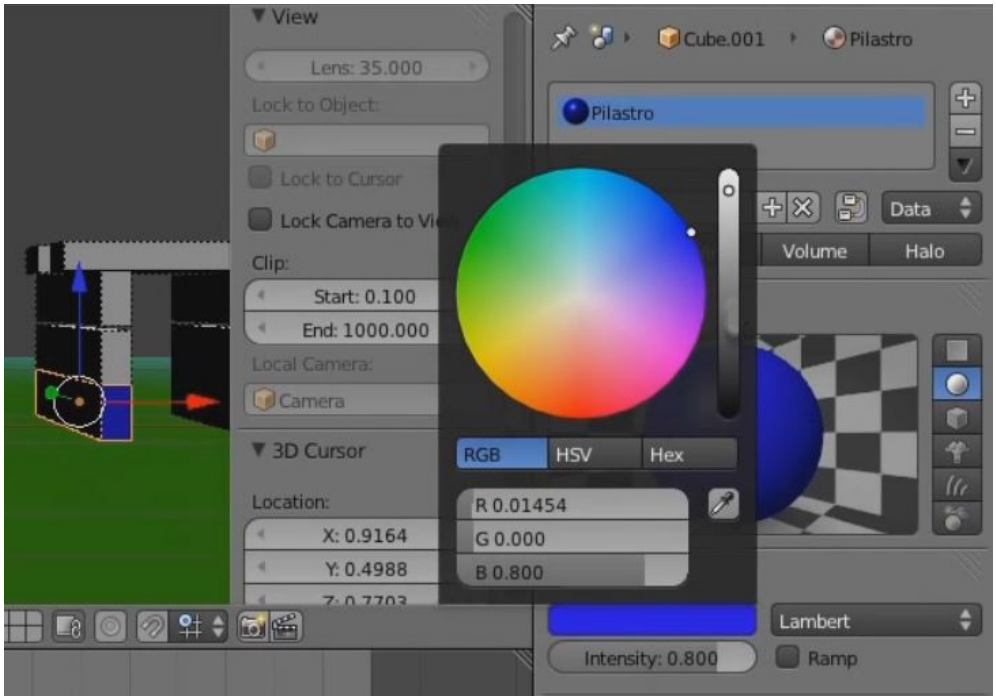
Il proiettile deve sparire al momento della collisione: per rilevare un contatto, possiamo utilizzare il sensore Touch, per cui aggiungiamo tale elemento e colleghiamolo all'attuatore Edit Object.

Nella scheda di Touch è presente il campo Material, di default vuoto.

*Sensore TOUCH*

Intuitivamente, possiamo limitare la distruzione dell'oggetto al contatto con altri oggetti dotati di questo Material (o di più Material: basta creare più sensori Touch e collegarli ad un unico controller OR e, da qui, ad un unico End Object, con un singolo collegamento); lasciando il campo vuoto, il sensore verrà attivato al contatto con qualsiasi elemento.

Torniamo quindi al primo Layer, selezioniamo uno dei mattoncini della struttura e dotiamolo di un Material, chiamandolo ad esempio "pilastro".



*Il Material per i pilastri*

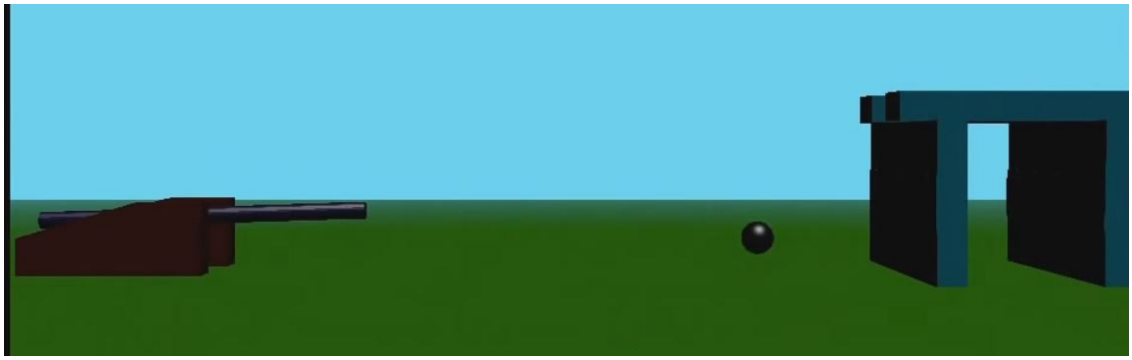
Gli altri pilastri, ottenuti per duplicazione dal primo che, inizialmente, non aveva Material, non avranno il nuovo Material; per risolvere questo problema, selezioniamoli con una selezione multipla, quindi ri-selezioniamo per ultimo (in modo da renderlo oggetto attivo) il pilastro col Material, premiamo CTRL L (per Link) e scegliamo Materials dal menù che apparirà, in modo da far condividere agli oggetti selezionati il materiale dell'oggetto attivo.

Torniamo quindi al secondo Layer, selezioniamo il proiettile e impostiamo il nome del Material "pilastro" nel campo Material del sensore Touch.

*Il nome del Material nella scheda del sensore TOUCH*

Neanche a dirlo, potete eliminare dalla scena anche i pilastri quando vengono colpiti dal proiettile: basta mettere un End Object nei pilastri specificando come Material del sensore Touch di attivazione quello del proiettile.

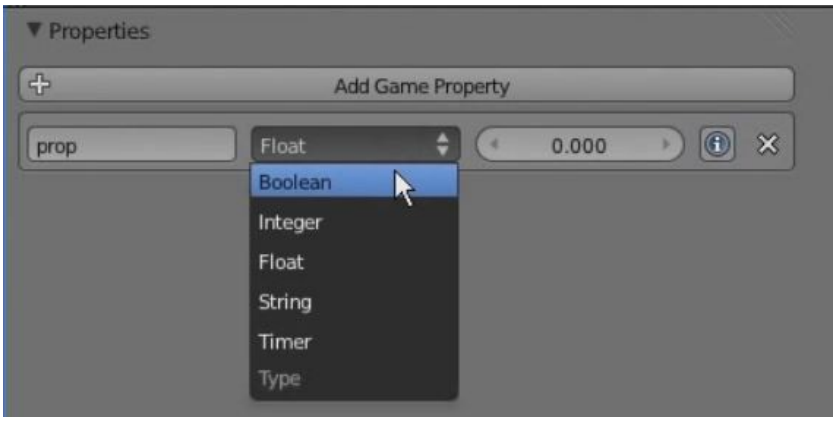
Bene, non ci resta che tornare al primo Layer, premere P per eseguire il gioco e spostare il supporto, variare l'inclinazione del cannone e fare fuoco.


*Altro screenshot dell'esecuzione*

Giusto per complicarci un po' la vita, approfittandone in realtà per parlare delle Properties, aggiungiamo una nuova caratteristica: il giocatore ha a disposizione solo tre colpi.

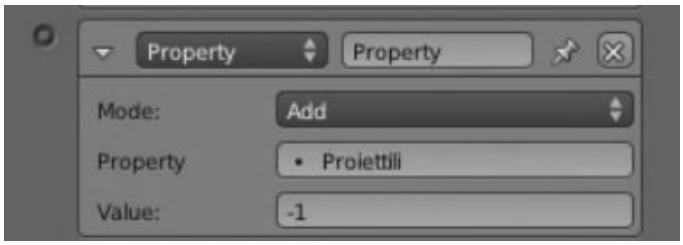
Si tratta quindi di aggiungere una proprietà "*Proiettili*", che avrà valore iniziale 3, da decrementare ad ogni sparo (quindi possiamo riciclare il sensore Keyboard della barra spazio) e di fare in modo che sia possibile sparare se il numero di Proiettili rimasti è maggiore di 0, quindi una condizione da mettere in AND con lo sparo.

Andiamo con ordine: selezioniamo il cannoncino e, nella scheda Logic Editor, aggiungiamo una Property di tipo numero intero (Integer) che chiameremo “Proiettili” e che avrà valore iniziale 3 (da impostare nel relativo campo numerico).




*Aggiungere la Property "Proiettili"*

Nella sezione di sensori, controller e attuatori, aggiungiamo un nuovo attuatore di tipo “Property” e, nella scheda di tale elemento, impostiamo la modalità “Add” (somma algebrica), la Property “Proiettili” e il valore -1.

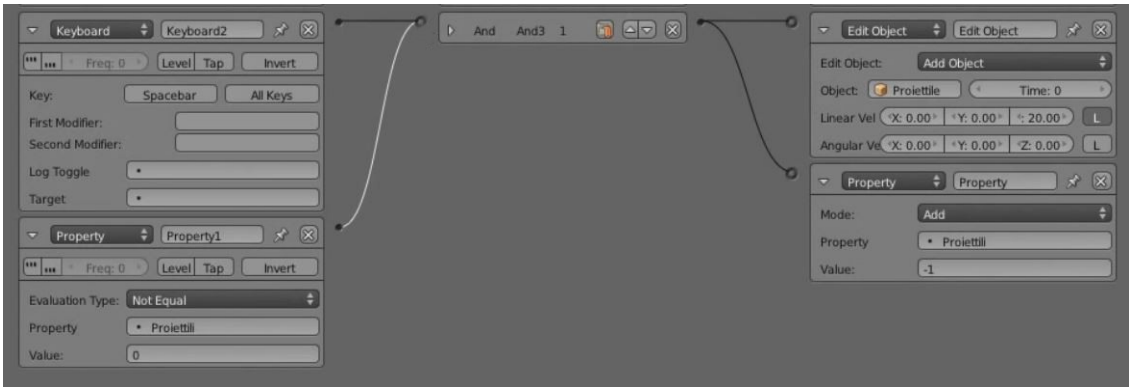



*Attuatore Property -1*

Colleghiamo questo attuatore (che quindi decrementerà il valore della proprietà “Proiettili” quando attivato) all'*AND* che trasmetteva il segnale dello sparo: ad ogni pressione della barra spazio, verrà sparato un proiettile e verrà decrementato il valore della relativa Property.

*Collegare l'attuatore Property al sensore dello sparo*

Il problema è che in questo momento gli spari non vengono bloccati se il valore diventa minore di 1; dobbiamo quindi aggiungere anche un sensore Property, che di default ha tipo “Equal”, cambiando il tipo in “Not Equal” (diverso), impostando la Property “Proiettile” e il valore 0: in questo modo, fintantoché il valore di “Proiettili” sarà diverso da 0, il segnale trasmesso dal sensore sarà positivo e l'utente potrà sparare, mentre a 0 colpi il segnale non passerà più oltre il controller *AND* e non verranno più creati proiettili né verrà decrementata ulteriormente la Property, che resterà a 0 (caratteristica utile se è prevista la possibilità di rifornirsi di munizioni, in modo da ripartire da 0 nel conteggio, non da un valore negativo).



*Sensore Property in AND con Keyboard*

Proviamo il tutto: adesso il giocatore ha a disposizione solo tre tiri.

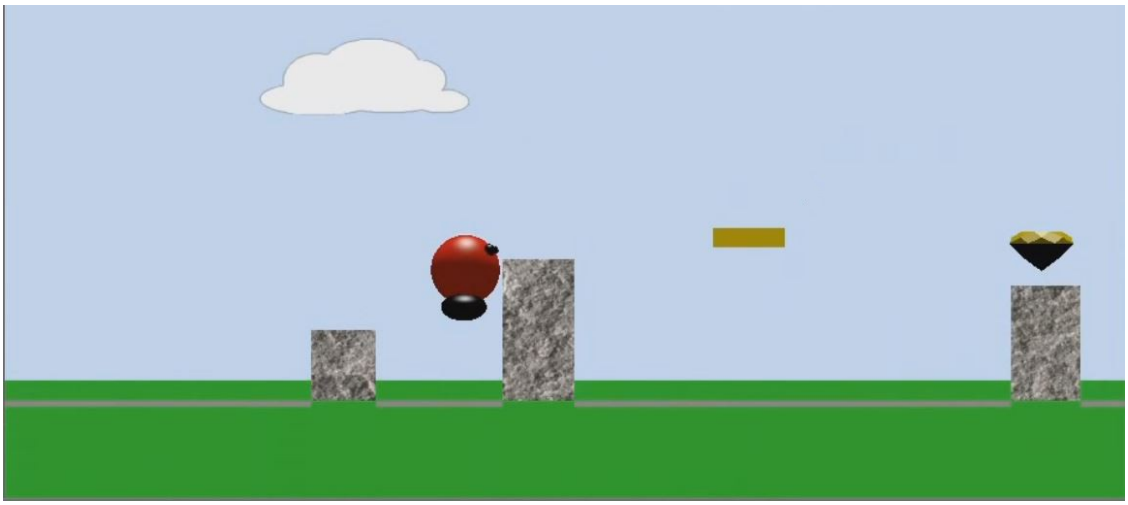
Magari non si tratta del gioco più avvincente della storia, ma se ci pensate ci ha permesso di trattare temi come il parenting, il pivoting, i constraints, il numero di munizioni e l'immissione (con conseguente eliminazione selettiva) di oggetti dinamici a runtime nella scena.

*  *  *

# Le basi del Game Engine, il motore di gioco di Blender 2.6 – 5

Eccoci alla quinta puntata del corso sulle basi del Blender GE, il Game Engine, motore di gioco di Blender 3D 2.6; in questa puntata e nella prossima vedremo come realizzare animazioni Object Mode da eseguire nel gioco e come far apparire delle sprites in seguito a certi eventi; inoltre, daremo un'occhiata alle proprietà fisiche dei Materials ed altro ancora.

Nell'immagine seguente è visibile un'anteprima del risultato che otterremo al termine del tutorial: si tratta di uno scenario stile platform game 2D anni '80-'90; tra le altre cose, la telecamera seguirà il personaggio principale (parleremo anche di questo argomento, nel tutorial).



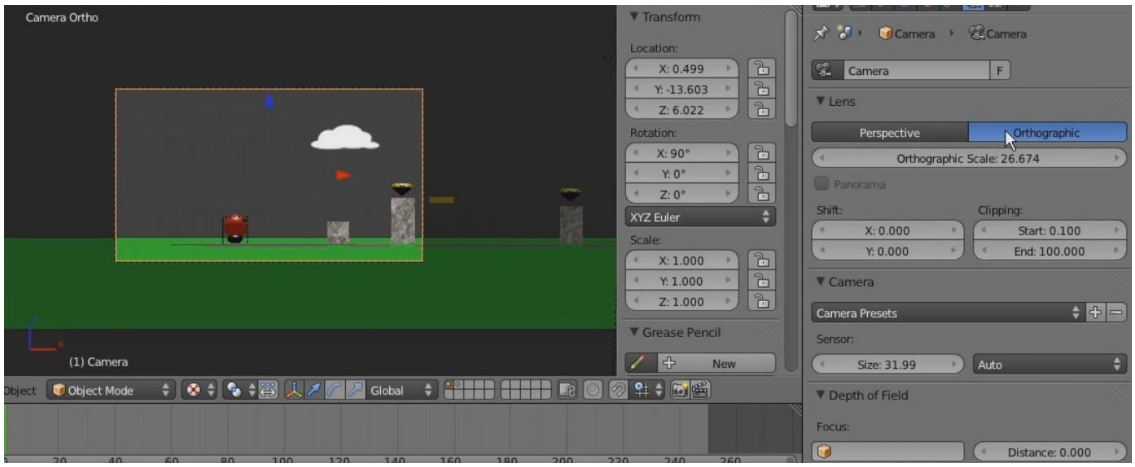
*Screenshot del risultato finale*

Come anticipato, mostrerò il dietro le quinte di questo scenario in due puntate, mostrando nella prossima come animare il personaggio con le armature e le collisioni, per cui per le vostre prove adesso utilizzate ad esempio un oggetto senza armature, da muovere con Simple Motion, come mostrato nelle puntate precedenti.

Iniziamo!

Lo scenario è stile platform 2D, con la telecamera che segue il personaggio principale e le prime cose da dire riguardano proprio la telecamera.

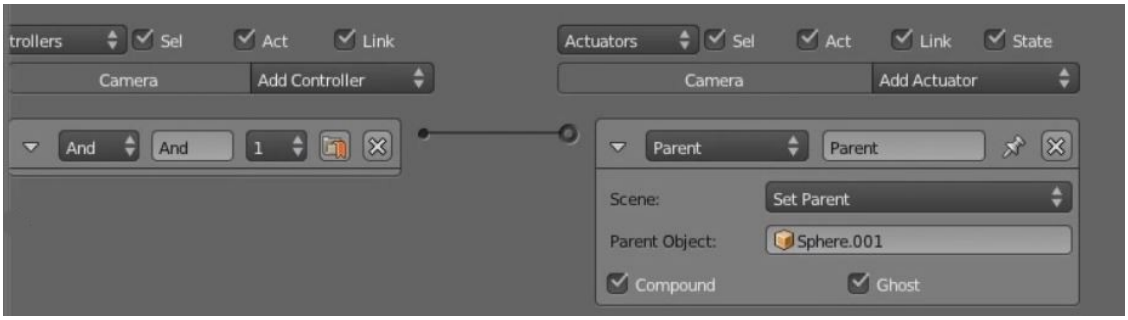
Per non implementare la prospettiva, dobbiamo cambiare la natura della telecamera da Perspective in Ortographic mediante l'apposito pulsante nella scheda Object Data, all'interno della Properties Window.




*Telecamera in modalità ORTHO*

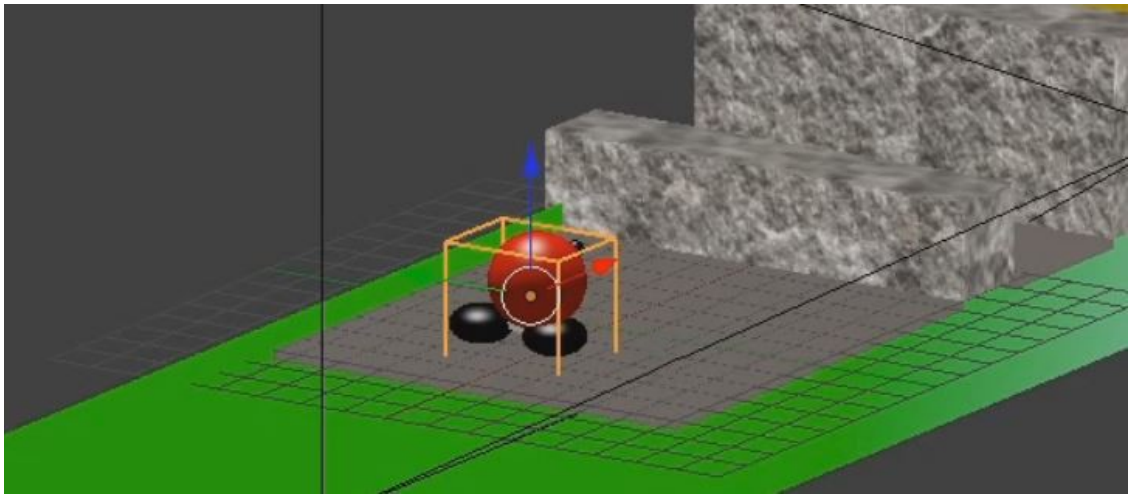
Quando si passa a Ortographic, per allontanare o avvicinare la telecamera alla scena non dobbiamo spostare tale oggetto avanti o indietro nella 3D View, perché adesso non ha più nessun effetto; quello che dobbiamo fare, invece, è variare il valore di Ortographic Scale, sotto il pulsante Ortographic in Object Data, per cui teniamo da una parte la vista Camera in una 3D View e, nel mentre, variamo Ortographic Scale nella Properties Window, fino a trovare l'inquadratura che ci soddisfa.

Tra gli attuatori del Logic Editor vi è Camera, che consente ad un oggetto di seguire un altro oggetto impostando una quota di riferimento e un range di distanze minima e massima; tuttavia, per questo particolare caso (ossia il Platform 2D), meglio utilizzare un attuatore Parent, per evitare che la telecamera ruoti un po' intorno al suo asse mentre segue il protagonista. Con la telecamera selezionata, quindi, aggiungiamo un sensore Always, un controller *AND* e colleghiamo il tutto ad un attuatore Parent che avrà, come oggetto bersaglio, il nostro protagonista.




*Attuatore PARENT per la telecamera*

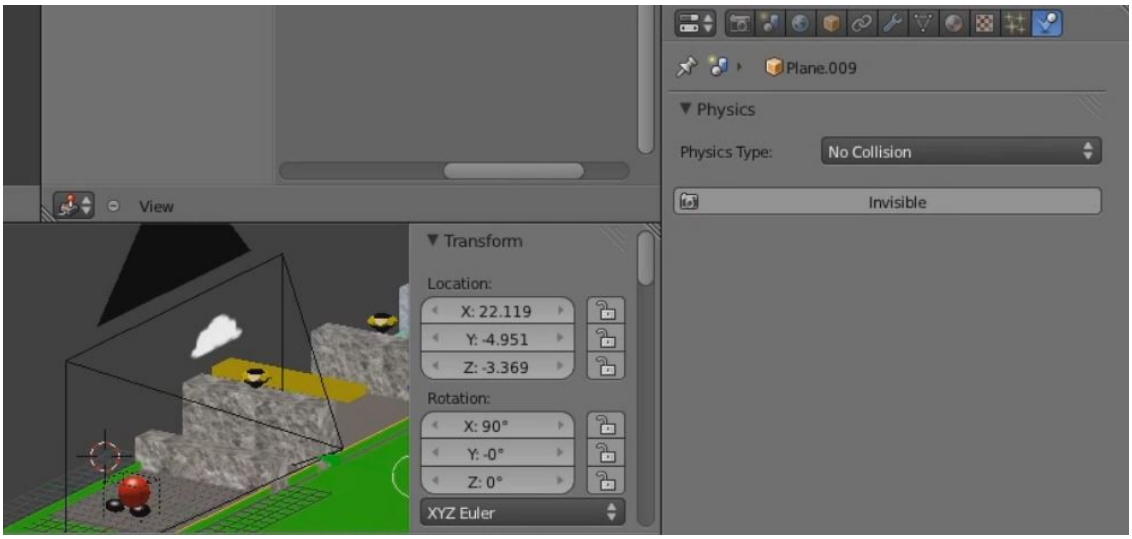
Prima di procedere, un consiglio: impostate un attuatore Constraint sull'asse della profondità per l'attore, in modo da impedirgli di avvicinarsi o allontanarsi dalla telecamera (e cadere nel vuoto), evento che può verificarsi ad esempio in seguito a impatti con oggetti (l'attore è di tipo Dynamic); anche se vediamo il tutto in 2D, infatti, l'ambiente è 3D e Blender calcola anche gli spostamenti in profondità.



*L'attore nelle tre dimensioni*

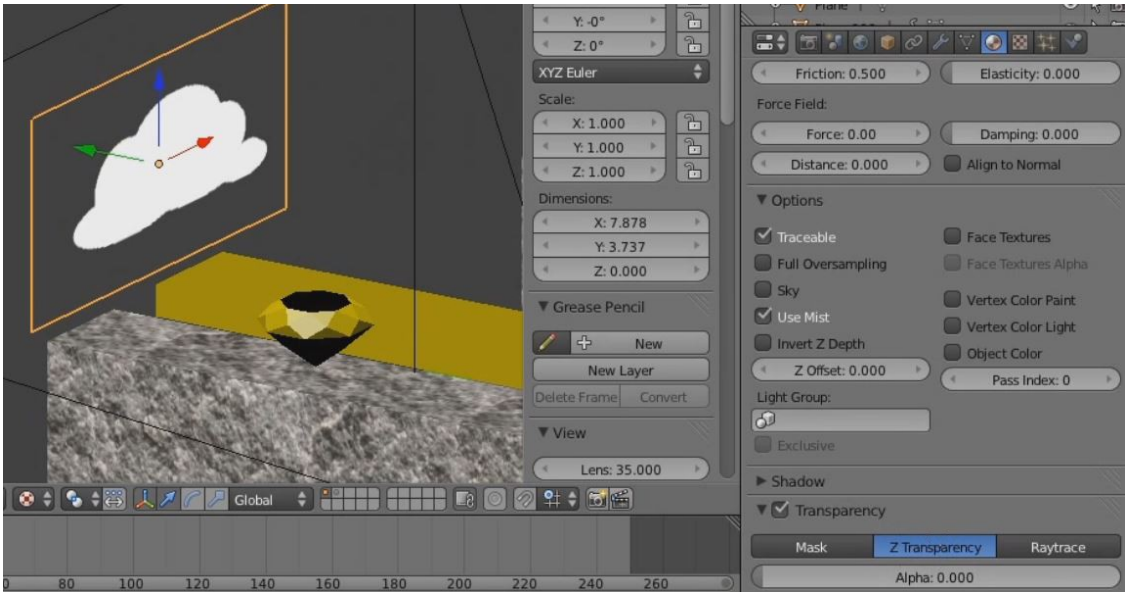
Diamo un'occhiata ad altri elementi della scena: World è dotato di un colore chiaro per lo sfondo ed è presente la foschia, MIST, in questo caso con Start e Depth rispettivamente a 15 e 20 ma ovviamente dipende dalla scala della vostra scena; è presente poi un piano di sfondo per simulare l'orizzonte in lontananza (in questo caso con un semplice colore diffuso verde, ma ovviamente potete utilizzare delle Textures ed anzi potete mettere degli elementi in primo piano, come fili d'erba, sassi o altro, mettendo un altro Plane tra la telecamera e il personaggio; niente geometrie particolari, quindi: solo una Texture con canale Alpha applicata a un Plane, infatti il gioco è 2D e la telecamera fa solo un panning, quindi questa soluzione può andare bene).

In ogni caso, per sicurezza, rendiamo questi Planes No Collision nella scheda Physics, nella Properties Window.




*NO COLLISIONS per il Plane*

Altri Plane "No Collision" che servono come elementi di sfondo, in questo scenario, sono quelli delle nuvole (ottenute semplicemente applicando una Texture immagine in bianco e nero ad un Plane, mediante mappatura UV). In questo particolare caso, bianco su nero, non c'è nemmeno bisogno di fornire un canale Alpha all'immagine: applichiamo al Plane un Material Shadeless con trasparenza Z-Buffer con Alpha a 0, quindi la Texture immagine con mappatura UV (per cui impostiamo UV Layer in Mapping), disattiviamo Use Alpha e selezioniamo Calculate nella sezione Image Sampling della Texture; in questo modo, diremo a Blender di ricavare i valori di opacità dai colori dell'immagine, e poiché bianco puro è completamente opaco e nero puro completamente trasparente, avremo solo il disegno della nuvoletta. Per finire, selezioniamo anche Diffuse Alpha oltre a Diffuse Color, nella sezione Influence della Texture.

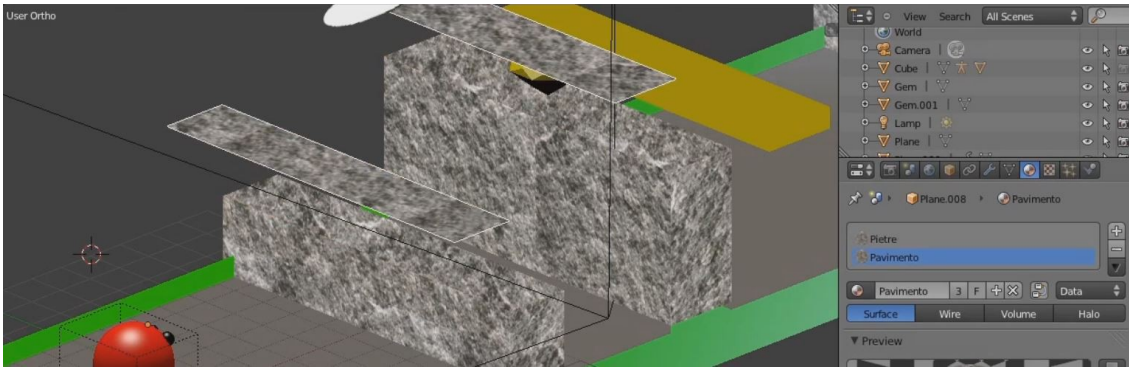


*Material e Textures delle nuvole*

L'utilizzo di immagini Alpha Mapped è quindi simile a quello che si ha in Blender Render, nel senso che dobbiamo mettere Alpha Z-Buffer a 0 nel Material (cosa che tecnicamente lo rende completamente trasparente) e attivare Diffuse Alpha nella Texture, per indicare che sarà tale elemento a determinare le zone e i valori di trasparenza per l'oggetto (in questo caso, con mappatura UV). Per evitare effetti di ombreggiatura dovuti alle posizioni delle fonti luminose (in questo caso, una luce Sun, senza ombre), ho reso il Material Shadeless.

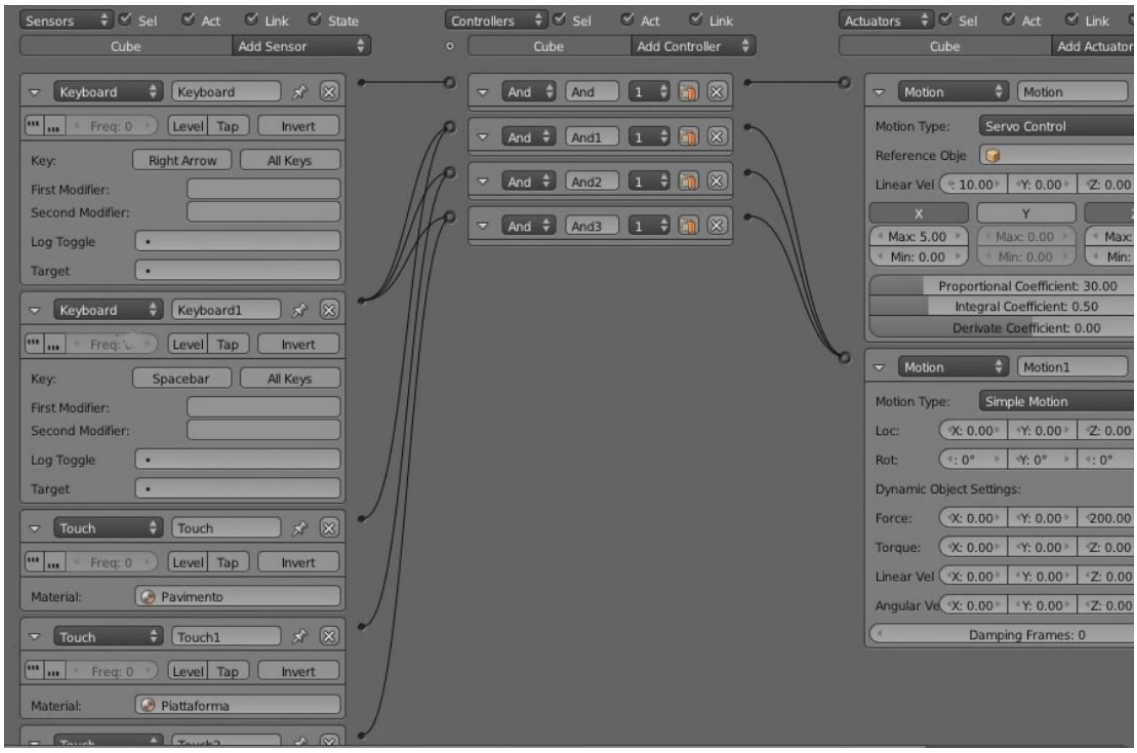
Passiamo agli altri elementi: la piattaforma è un semplice Plane, ma attenzione al nome del suo Material perché limiteremo al contatto con tale Material la possibilità del protagonista di saltare (non vogliamo farlo saltare mentre è già in volo).

Per evitare che il nostro personaggio possa saltare con balzi incredibili quando tocca una parete, i parallelepipedi utilizzati come blocchi o rocce dovranno avere un Material diverso, cosa che però impedirebbe al protagonista di saltare quando si trova correttamente poggiato sulla sommità di un blocco. Per risolvere questo problema, possiamo selezionare le facce superiori dei blocchi, separarle da tali elementi in modo da creare un nuovo oggetto a parte (con P, *Separate*) e fornire loro un nuovo Material, diverso da quello dei blocchi originali (ad esempio, quello del pavimento o un Material ad hoc).




*Separare le facce superiori dei blocchi*

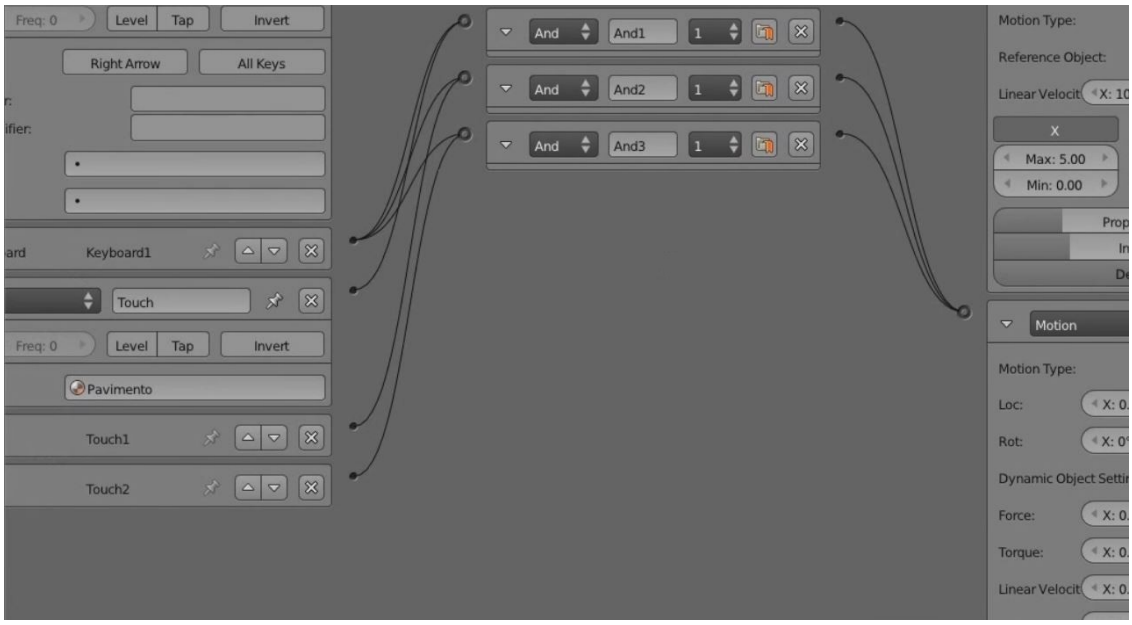
Vediamo, quindi, come impostare le condizioni di salto: si tratta di selezionare il protagonista e mettere un sensore Touch in *AND* con il sensore Keyboard che registra la pressione del tasto per il salto, nel mio caso la barra spazio, per cui il personaggio potrà saltare solo quando entrambe le condizioni si verificheranno.




*Le condizioni per il salto*

Questa configurazione ci permette di realizzare altri effetti, ad esempio salti diversi a seconda del materiale utilizzato per la piattaforma (un salto da una pedana speciale potrebbe far alzare il protagonista con forza 1000, dalla sabbia con forza 100, e così via).

L'entità del salto, ossia la forza su Z, va specificata in base alle dimensioni degli oggetti e della scena, facendo delle prove; qui ho impostato 200 su Z e, come potete notare dallo schema dei Nodi, ci sono altri controller *AND* che portano il segnale a Simple Motion per il salto, in quanto ci sono più Materials che consentono il salto e sono quelli della palla rossa rimbalzante (più avanti nello scenario) e quello della piattaforma in movimento.

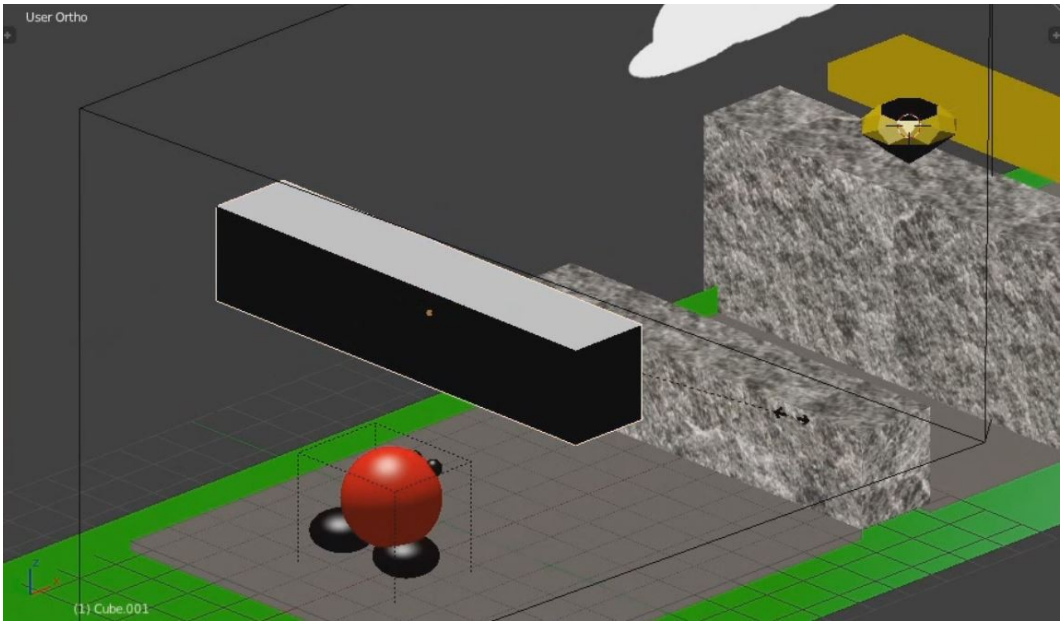



*Schema Logic Bricks per il salto (particolare)*

Passiamo proprio alla piattaforma in movimento: un oggetto (di tipo Static) che effettua in continuazione un movimento, quindi una trasformazione in Object Mode, all'interno della scena.

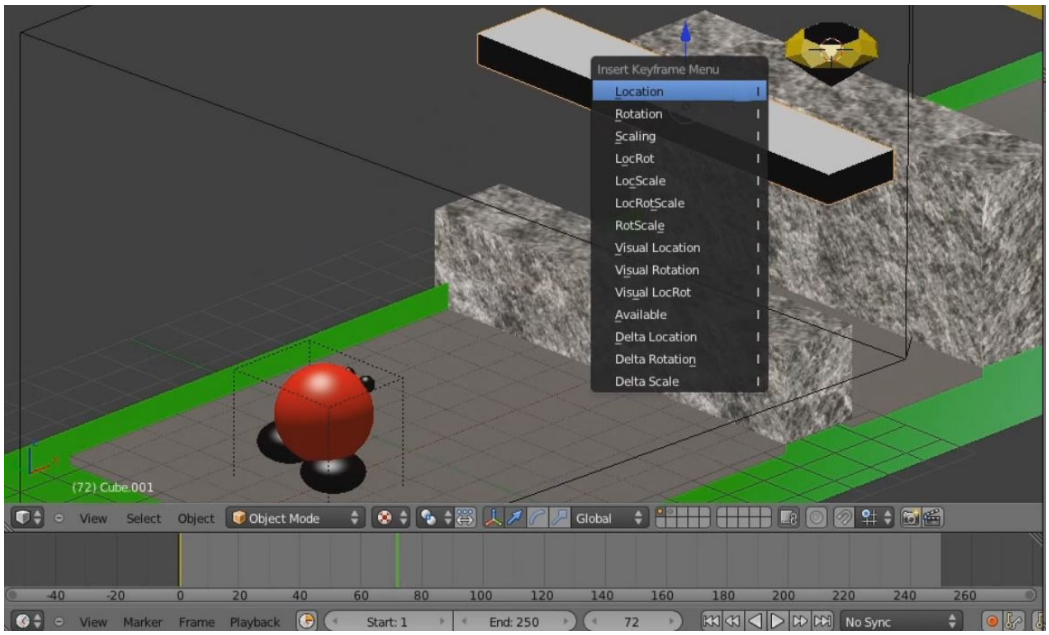
Per realizzare questo tipo di animazioni (che non sono quindi Sprites animate ma oggetti in movimento secondo percorsi prestabiliti) dobbiamo realizzare l'animazione in Object Mode in fase di editing della scena e implementare poi una catena sensore-controller-attuatore nel Logic Editor dell'elemento utilizzando l'attuatore Action. Procediamo con ordine.

Inseriamo un oggetto nella scena, magari un altro parallelepipedo come la piattaforma presente nello scenario. Decidiamo quanto deve durare l'animazione: vogliamo far spostare ad esempio l'oggetto avanti e indietro sull'asse X, come per collegare due blocchi di pietre, in 3 secondi per l'andata e 3 per il ritorno?




*Inserire il parallelepipedo che rappresenta la piattaforma*

Mettiamoci allora al Frame 1 della Timeline, posizioniamo l'oggetto nel punto iniziale e inseriamo un Keyframe Loc; andiamo al 72esimo frame (se Blender Render è a 24fps), posizioniamo l'oggetto nel punto finale e inseriamo un Keyframe Loc; andiamo quindi al frame 144 (6 secondi per 24 frames al secondo), riposizioniamo l'oggetto nel punto di partenza e aggiungiamo un Keyframe LOC.

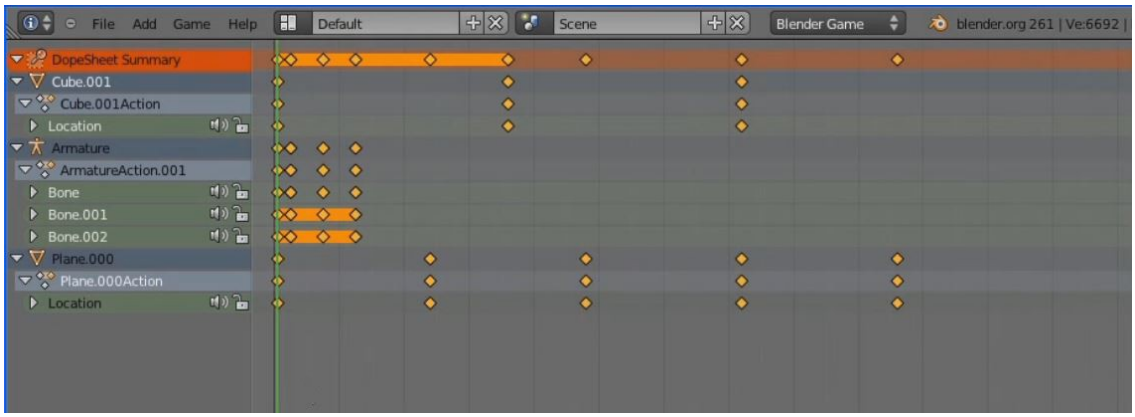



*Registrazione dei keyframes*

L'animazione appena realizzata viene eseguita, in modo Edit, premendo ALT A, solo una volta, dal frame 1 al frame 144 dell'animazione, ma non viene eseguita affatto in modo gioco se premiamo P, infatti per essere eseguita lì deve essere attivata mediante un attuatore Action.

Una Action è un blocco di informazione generato quando un oggetto viene animato; è un'astrazione delle IPO, che consentono personalizzazioni sull'animazione (ad esempio, possiamo cambiare l'interpolazione ai vari Keyframes da Bezier a Lineare) e, come le IPO, viene creata automaticamente da Blender quando si inserisce almeno un Keyframe.

Per vedere l'Action creata per il parallelepipedo, con tale oggetto selezionato cambiamo una finestra in DopeSheet: abbiamo varie strisce, che si riferiscono agli oggetti presenti nella scena (raggruppandone i canali, quindi non abbiamo ad esempio Loc o Rot) e dei rombi in corrispondenza dei Keyframes.

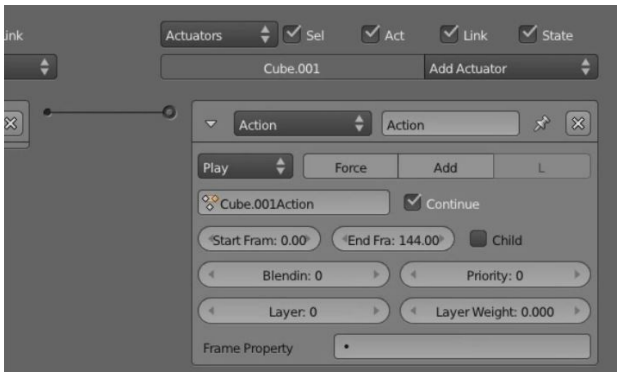



*Il Dopesheet con l'Action dell'animazione della piattaforma*

Passiamo quindi al Logic Editor e aggiungiamo un sensore Always, un Controller *AND* e un attuatore Action.

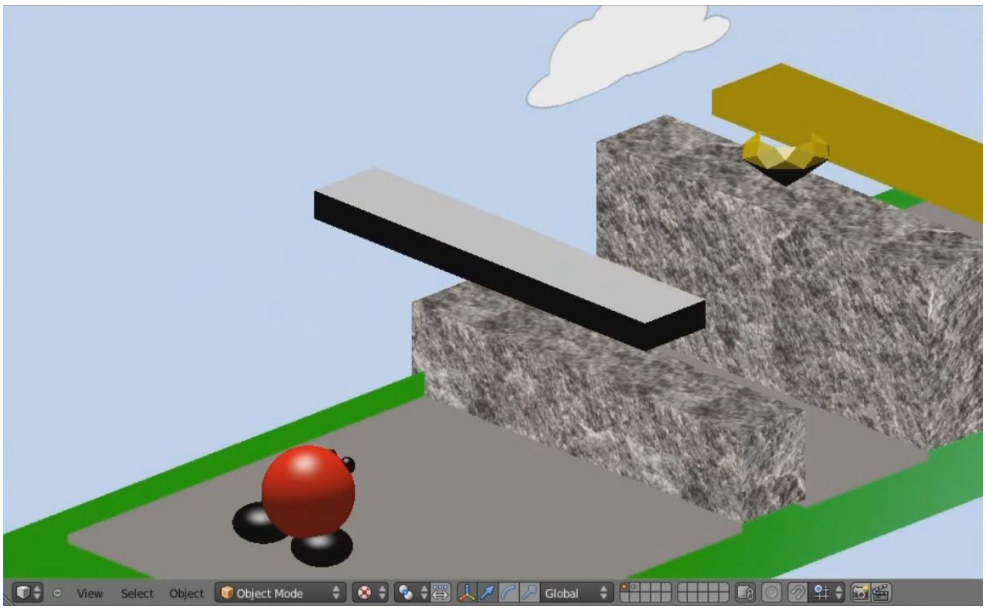
Nell'attuatore, impostiamo il nome della Action creata automaticamente da Blender per il parallelepipedo, quindi impostiamo i frames iniziale e finale, nel nostro caso 1 e 144; questo passaggio è importante perché determina la durata dell'animazione e soprattutto ci consente di definire più animazioni per un oggetto, su un'unica Action, così potremo scegliere quale animazione eseguire impostando i frame Start ed End nell'attuatore Action.

Altra cosa da fare è scegliere l'Action Playback Type nel relativo menù dell'attuatore: la voce di default è Play, che però esegue l'animazione solo una volta, quando viene lanciato per la prima volta il sensore di attivazione (nel nostro caso, con Always, all'avvio del gioco), mentre noi vogliamo un'animazione in loop, quindi scegliamo Loop End.




*Impostare i Logic Bricks per la piattaforma*

Avviamo il gioco: il parallelepipedo esegue correttamente l'animazione impostata, iniziando di nuovo quando finisce il ciclo di 6 secondi.




*Screenshot dell'esecuzione del gioco*

Quanto detto illustra come inserire semplici animazioni in Object Mode nella scena e vi dà qualche spunto per provarne di più interessanti; ad esempio, definire un'animazione in un altro range di Frame, ad esempio lo spostamento verticale andata e ritorno del cubo in 80 frames, quindi (utilizzando magari una Property del gioco da cambiare in base ad un evento, tipo la raccolta di un oggetto o la pressione di un tasto) fate in modo da passare dal primo attuatore Action al secondo, per elaborare la seconda animazione.

Restano ancora due elementi da discutere in questa puntata: l'animazione generata quando il protagonista prende un oggetto (in questo caso la gemma) e la palla che rimbalza da sola.

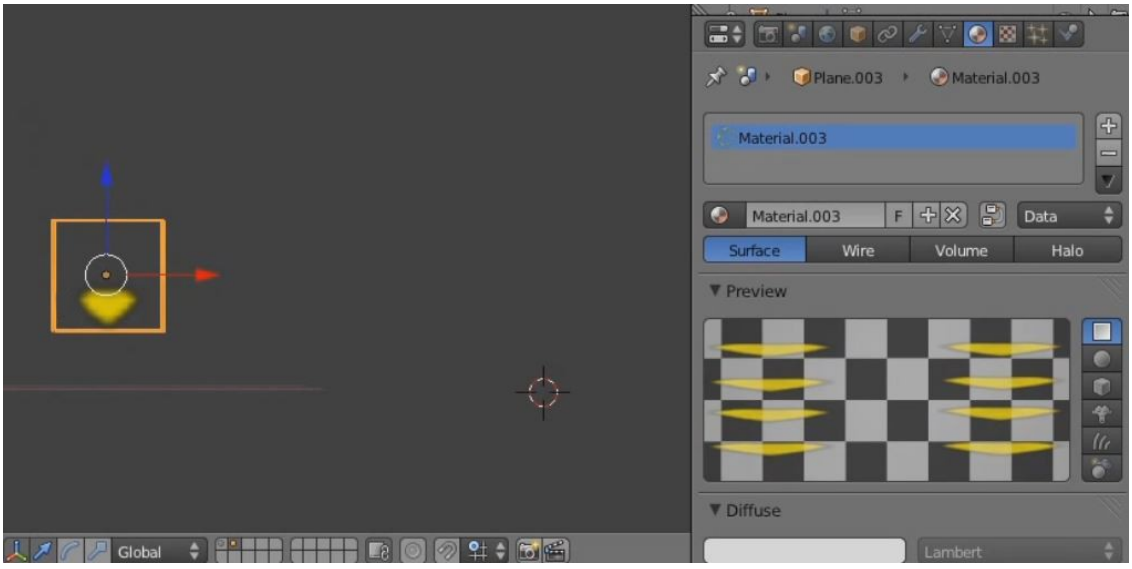
*  *  *

Eccoci alla seconda parte di questo tutorial.

Abbiamo visto come inserire animazioni Object Mode all'interno della scena, attivandole una sola volta (con Play) o in maniera ciclica (con Loop) in base a particolari eventi; a volte, comunque, si ha l'esigenza di mostrare brevi animazioni puramente 2D, ad esempio dopo aver raccolto un oggetto, per cui si può ricorrere all'esecuzione di Sprites animate da mostrare a comando.

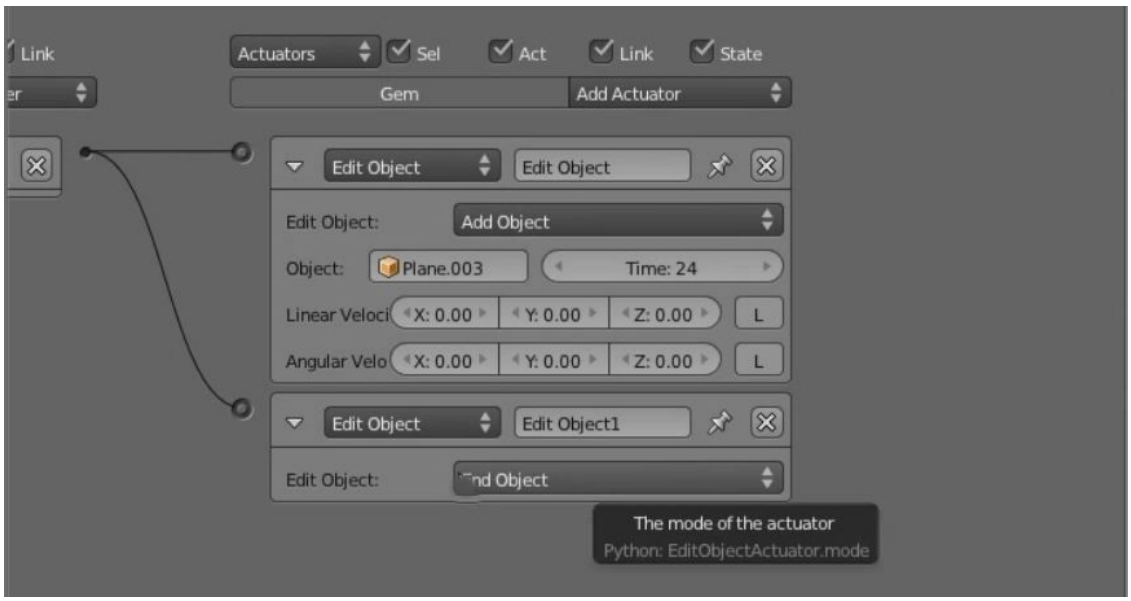
Nelle puntate precedenti abbiamo visto come eseguire delle animazioni partendo da una immagine tiled applicata mediante UV Mapping ad un oggetto, quindi qui non rivedrò quella parte, trattando solo l'esecuzione a comando.

Nel secondo Layer della scena ho messo un Plane con sopra, mappata con Tiling, l'animazione di una gemma stilizzata, che sale un po'.




*Il Plane con la gemma UV-mappata*

Nella scena principale, per eseguire questa animazione quando il protagonista prende una gemma, dobbiamo selezionare tale oggetto, aggiungere un controller Touch sensibile al Material del protagonista (perché solo lui può prelevarlo) collegato, mediante un controller *AND,* a due attuatori, entrambi Edit Object ma con azioni differenti: il primo è un Add Object che, neanche a dirlo, aggiunge il Plane dell'animazione visto nel Layer 2 alla scena, posizionandolo dove si trova l'oggetto che ha attivato l'attuatore; il secondo è un End Object che rimuove la gemma dalla scena.




*I Logic Bricks per l'animazione della gemma*

> *NOTA: non possiamo mettere, qui, un attuatore Property per incrementare il numero delle gemme raccolte dal protagonista, perché quella proprietà è del protagonista, non della gemma. Ecco uno spunto per personalizzare il gioco: oggetti come chiavi colorate o gemme che, se raccolte, mettono a True proprietà relative del protagonista, consentendogli ad esempio di aprire porte, saltare più in alto, eccetera.*

Quando il protagonista toccherà una gemma, questa sparirà e mostrerà il Plane con l'animazione... attenzione, però! In questo modo, quel Plane resterà lì per sempre, eseguendo l'animazione continuamente! Per dargli un tempo di vita pari alla durata della Sprite, nel mio caso 24 secondi, specifichiamo Time 24 nell'attuatore Add Object che avvia l'animazione.

Questa caratteristica è utile, in generale, per far sparire gli oggetti inseriti nella scena dopo un po' di tempo: se il protagonista spara un proiettile, un missile o altro, possiamo impostare la distruzione di questo elemento in base alla collisione con un altro elemento o ad un tempo massimo, per evitare di lasciare oggetti disseminati nella scena per tutta la durata del gioco, sprecando memoria inutilmente.

Ricordatevi di applicare le trasformazioni di scaling sia al piano con la Sprite animata che all'oggetto che lo inserisce nella scena, ossia (in questo caso) la gemma, per i motivi visti nel caso della palla di cannone, in un'altra puntata di questo corso.

*  *  *

Passiamo quindi all'ultimo argomento di questa puntata: la palla che rimbalza!
La palla è un oggetto di tipo Dynamic, mentre il pavimento è di tipo Static.

Una cosa poco nota e, volendo, poco intuitiva della fisica nel GE, è che le proprietà di elasticità degli oggetti non vanno definite nella scheda Physics degli stessi, bensì nei loro Materials.

In tale scheda nella Properties Window, infatti, troviamo la sezione Physics, attiva di default, ed in particolare i campi Friction ed Elasticity.

*I campi Friction ed Elasticity per il Material "fisico"*

Friction indica l'attrito, quindi una forza che si oppone al moto relativo degli oggetti e che consente di frenare gli elementi anziché lasciarli scivolare senza freni come oggetti sul ghiaccio.

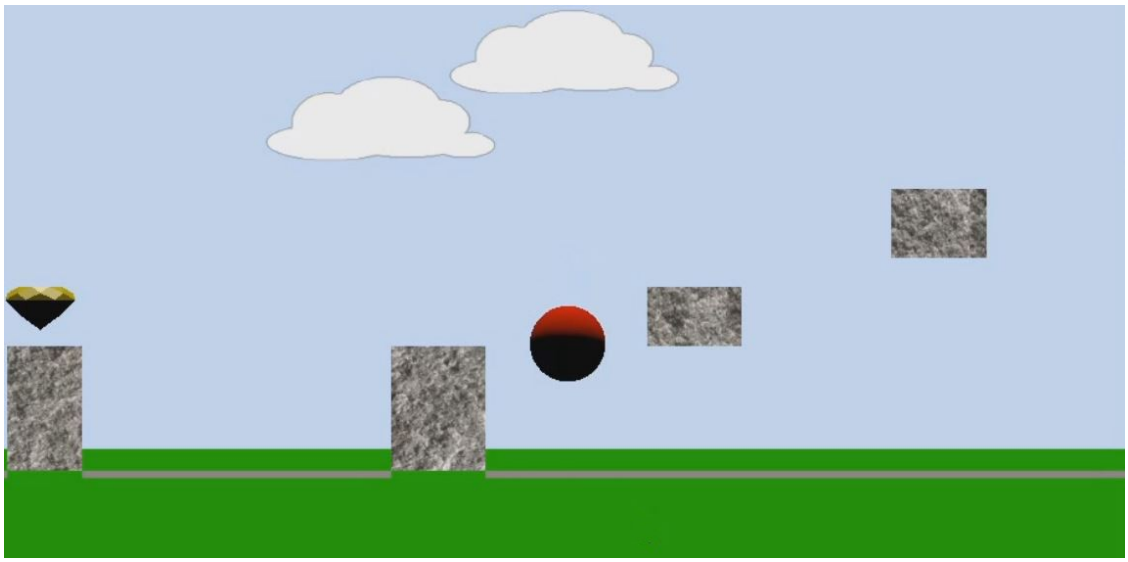
Neanche a dirlo, conviene impostare un valore maggiore di 0 per Friction del pavimento, in modo da non far scivolare il nostro personaggio all'infinito, se è di tipo Dynamic e lo spostiamo mediante forze, a meno che ovviamente non vogliamo proprio questo effetto (ad esempio se il personaggio si muove su lastre di ghiaccio o altri elementi con attrito nullo o quasi). Nel mio caso, per bloccare il protagonista in poco tempo, ho messo Friction 0.5.

Elasticity rappresenta, invece, il coefficiente di elasticità di un Material, quindi quanta energia cinetica dovrà restituire agli oggetti in caso di urti e collisioni, implementando così i rimbalzi.

Attenzione però, perché Elasticity va impostata maggiore di 0.0 per entrambi gli oggetti coinvolti nell'impatto. Il valore massimo è 1.0.

Conviene fare delle prove per trovare i giusti coefficienti per i due oggetti: il più delle volte, con valori bassi, il rimbalzo risulterà smorzato, annullandosi alla fine, e questo è un comportamento fisico corretto, mentre con entrambi i valori a 1 (sia per il pavimento che per la palla) la palla riceverà più energia di quella trasmessa e finirà per rimbalzare sempre più in alto nella scena.

Per realizzare un rimbalzo costante, in questo caso, ho impostato l'elasticità a 1 per la palla e 0.9 per il pavimento.



*Rimbalzo della palla durante l'esecuzione del gioco (screenshot)*

In questo caso non c'è grande interazione tra la palla e altri elementi ma è possibile, ad esempio, spingere tale oggetto con un impatto e far calcolare poi a Blender i rimbalzi; la novità, quindi, sta nella gestione delle proprietà elastiche del materiale per far rimbalzare l'oggetto anziché farlo bloccare quando cade; proprietà che, come abbiamo visto, si trovano nella scheda Material e riguardano i due attori coinvolti nell'impatto, non uno solo.

*  *  *

# Le basi del Game Engine, il motore di gioco di Blender 2.6 – 6

Eccoci alla sesta puntata del corso sulle basi del Blender GE, il Game Engine, motore di gioco di Blender 3D 2.6. In questa puntata, nella prima parte riprenderemo la scena vista nella puntata precedente e parleremo delle animazioni delle Armatures; nella seconda parte, parleremo dei Messages per inviare informazioni ad altri oggetti del gioco, di come mostrare scritte e altri elementi in sovrimpressione e dei sensori Near, Ray e Radar.

Iniziamo con l'animazione delle Armatures.
Nella puntata precedente abbiamo visto come realizzare delle animazioni "automatiche" nella scena, impostando delle Actions nella fase di Editing di Blender e gli attuatori Actions che le eseguivano (da uno start frame ad un end frame) nel Logic Editor.

Con le Armatures la situazione non cambia molto, ma bisogna impostare correttamente le parentele e, soprattutto, il *bound* dell'oggetto dotato di Armatura, per rilevare correttamente ostacoli e collisioni.

In particolare, l'Armatura dev'essere imparentata con l'attore da deformare e dev'essere dotata di una Action propria che determina come animarla quando, ad esempio, il protagonista deve camminare; la mesh da deformare, a sua volta, può doversi spostare quando deformiamo l'Armatura, e questo avviene ad esempio con la camminata: se premiamo la freccia in avanti per far spostare l'oggetto, lo stesso evento deve attivare l'animazione dell'Armatura e lo spostamento in avanti della mesh, perché la sola Armatura lo farebbe camminare sul posto, mentre la sola mesh farebbe scivolare l'oggetto senza animazione. Non animeremo, però, la mesh.

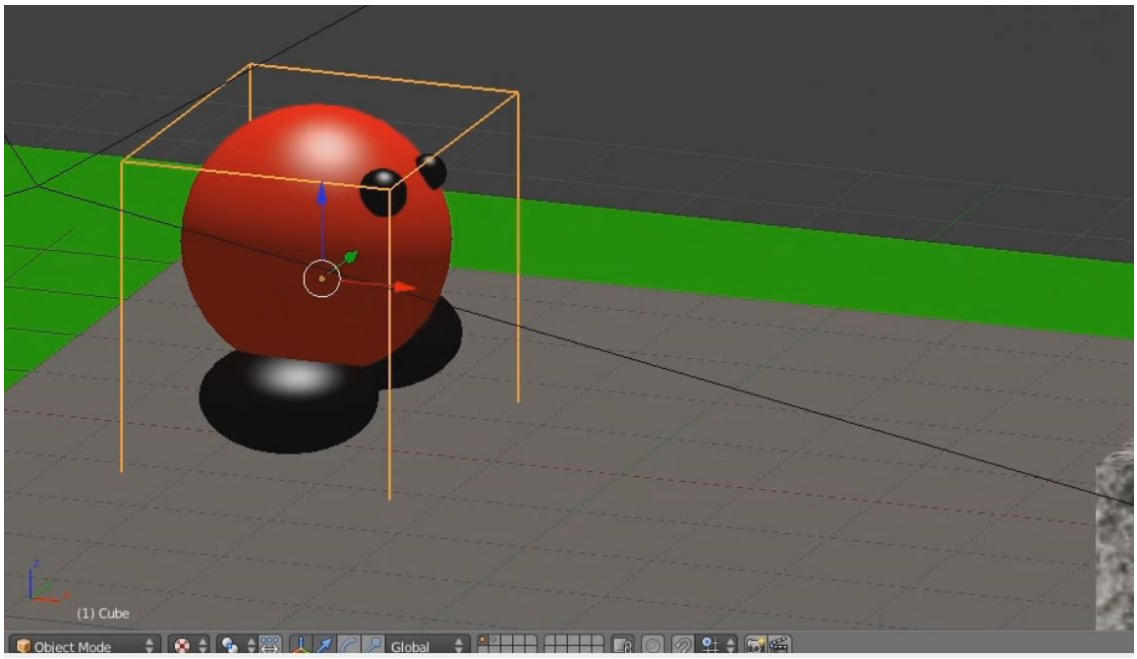
C'è infatti un problema: le collisioni, adesso, non possono essere rilevate né dall'Armatura né dalla mesh, per cui dovremo inserire un altro elemento invisibile (una specie di "gabbia-contenitore") che reagirà alle collisioni e sarà padre sia dell'Armatura che della Mesh.

L'animazione modo oggetto dello spostamento non riguarderà quindi la mesh, ma la sua "gabbia".

Fatta questa premessa, vediamo un esempio pratico.

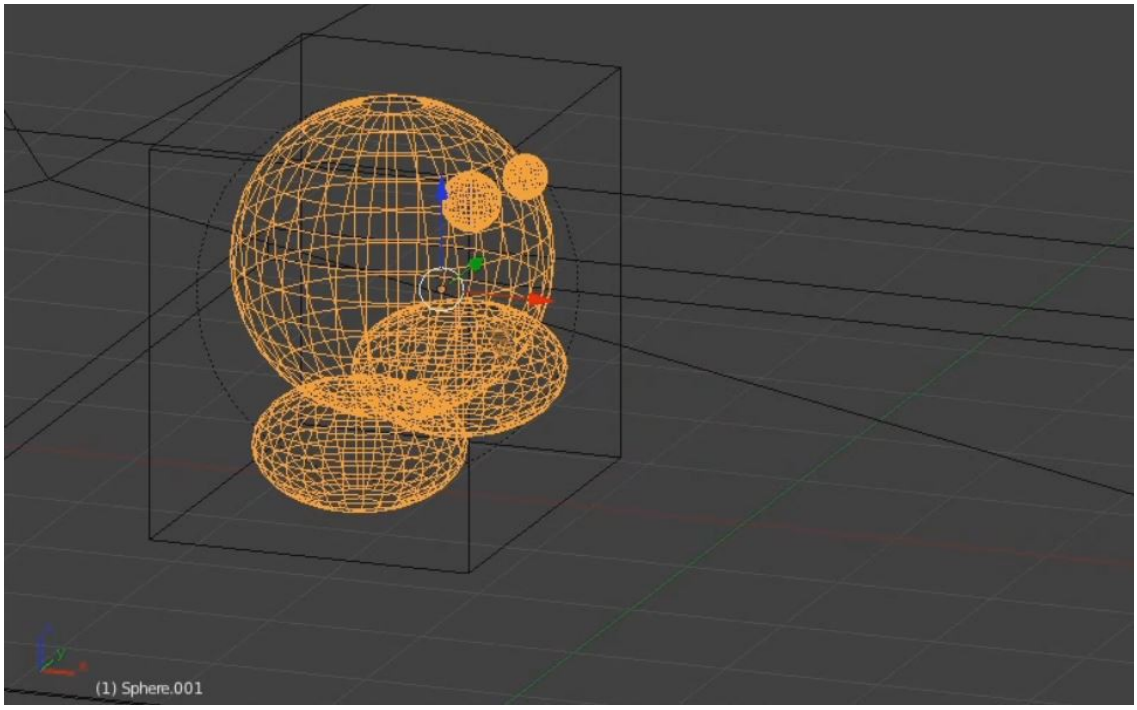
Il pupazzo che sto utilizzando è una semplice sfera con, al posto dei piedi, altre due sfere, un po' schiacciate; è una figura molto stilizzata, è vero, ma il principio è applicabile a figure molto più complesse come animali o esseri umani.

Le sfere che rappresentano i piedi e quelle degli occhi fanno parte del corpo principale: si tratta quindi di un'unica mesh, non di vari elementi connessi con parenting o altro.




*L'oggetto utilizzato per l'esempio*

Selezioniamo l'oggetto, spostiamo il 3D Cursor al suo baricentro con SHIFT S e scegliendo *Cursor to Selected*; passiamo, magari, in modalità di visualizzazione Wireframe, per vedere meglio l'interno dell'oggetto.

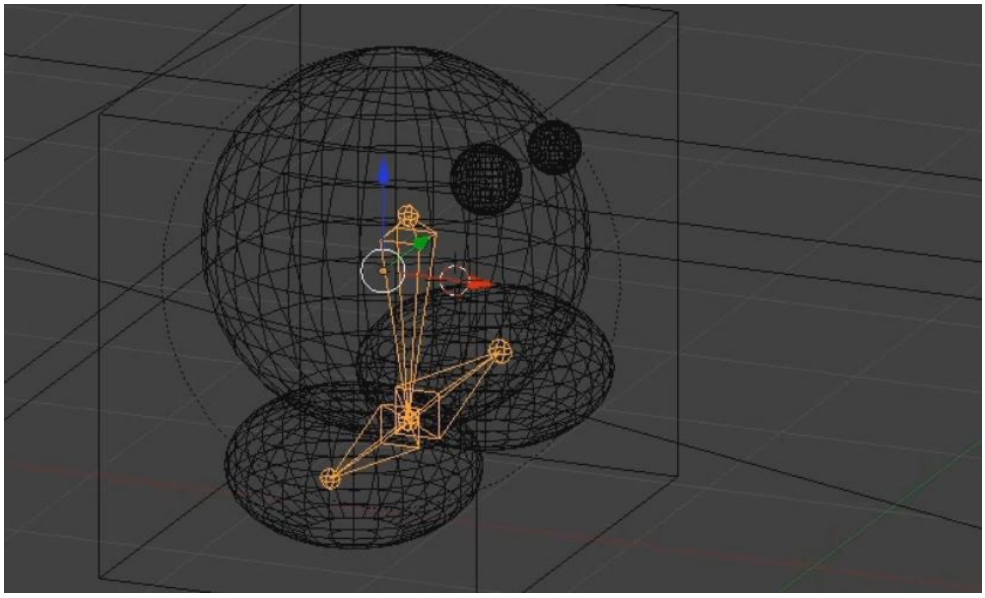



*L'interno dell'oggetto (visualizzazione Wireframe)*

Aggiungiamo quindi un oggetto Armature, composto di default da un osso.

In questo tutorial non parlerò delle basi delle Armatures e di come associare le ossa a varie parti dell'oggetto con modalità come Automatic, Envelope e Groups, perché a questo argomento ho dedicato 5 puntate del mio videocorso sulle basi di Blender 3D 2.5, valido comunque anche per Blender 2.6, ed in particolare dalla 75esima alla 79esima puntata; qui sto aggiungendo l'armatura, con due ossa più piccole partendo dall'osso principale (per regolare i piedi) e sto applicando l'Armatura alla mesh con la modalità Automatic Groups, perché in questo caso Blender effettua in maniera corretta le associazioni automatiche.

Selezioniamo prima la mesh, poi l'Armatura, con una selezione multipla e facendo attenzione all'ordine, quindi premiamo CTRL P e scegliamo, in questo caso, "*With Automatic Weights*".

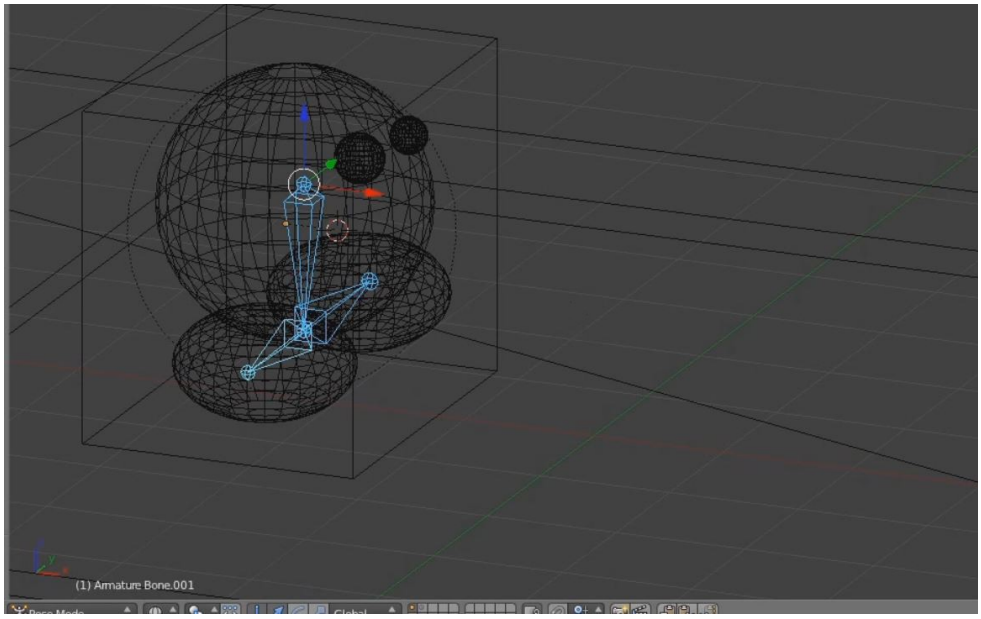


*L'Armatura associata in maniera "automatica" all'oggetto*

Passiamo in Pose Mode (tra parentesi, il gioco non può essere lanciato in Pose Mode, quindi se premete P e vedete che non succede niente verificate se siete correttamente in Object Mode).

Selezioniamo tutte le ossa con A.

L'animazione che realizzerò qui è una versione davvero primitiva di camminata (anzi in realtà è proprio uno "scivolamento", muovendo prima un piede e poi l'altro), tuttavia qui dobbiamo concentrarci sui principi, poi ciascuno di voi potrà realizzare animazioni più interessanti, con più ossa e movimenti fluidi.

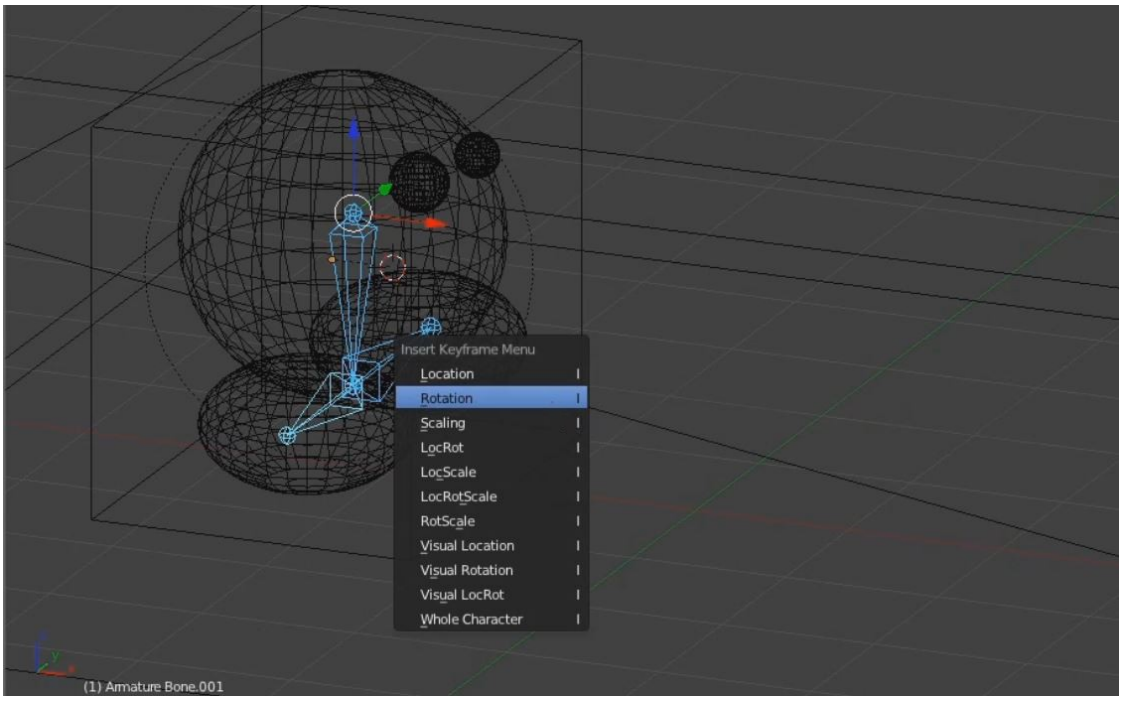



*Pose Mode, con tutte le ossa selezionate*

Decidiamo la durata del passo: diciamo, ad esempio, che dalla posizione iniziale si passa al primo piede in avanti in 5 frames; quindi, per far passare avanti l'altro piede dovremo impiegare 10 frames (5 per raggiungere il primo piede, 5 per superarlo), poi altri 10 per chiudere il ciclo; tuttavia, queste operazioni vanno fatte sul posto.

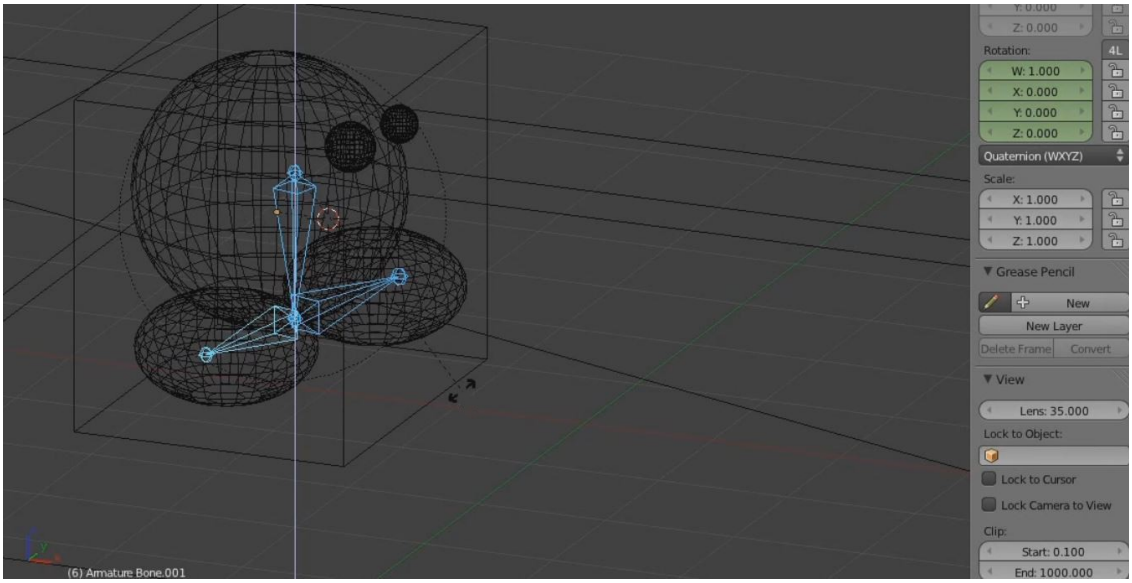
Per questo movimento di base, quindi, è sufficiente selezionare tutta l'Armatura ed effettuare delle rotazioni intorno all'asse verticale, quello Z.

Portiamoci al frame 1 dell'animazione e, con tutte le ossa selezionate, inseriamo un Keyframe Rotation.



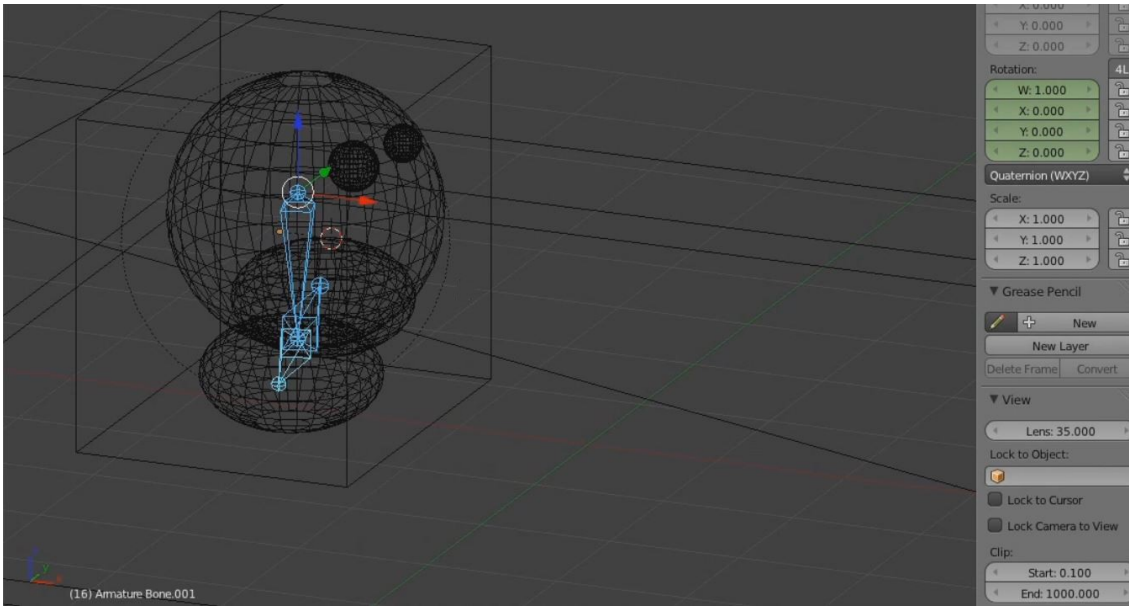
*Inserimento Keyframe al Frame 1*

Andiamo al sesto frame dell'animazione, quindi ruotiamo un po' l'armatura intorno all'asse Z, ad esempio di 15 gradi, in modo da far ruotare il modello, ponendolo ad esempio col piede sinistro in avanti, e inseriamo un altro keyframe Rot. È chiaro che, con modelli come gli esseri umani, dovremo invece ruotare cosce, gambe e piedi (e magari ancheggiare leggermente il bacino), utilizzando come perni testa del femore, ginocchio e caviglia... ci vuole un bel po' di tempo, ecco perché qui sto utilizzando questo pupazzo.



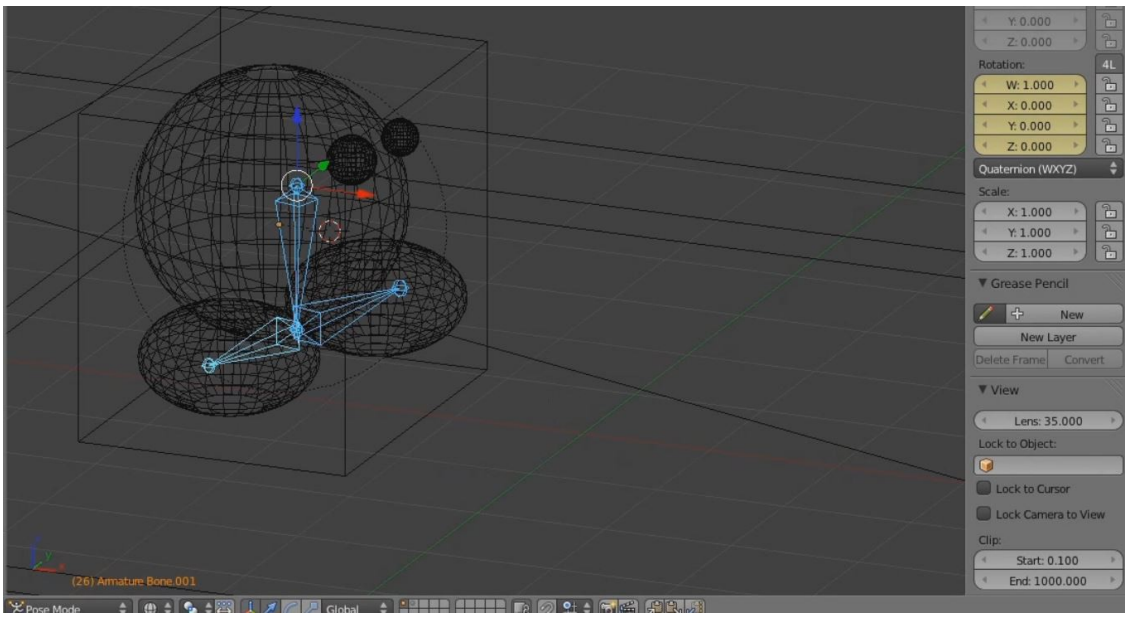
*La situazione al sesto frame*

Andiamo al sedicesimo frame dell'animazione, quindi ruotiamo intorno a Z, nel senso opposto a quello di prima, con valore doppio (ad esempio 30 gradi) ed inseriamo un altro Keyframe Rot.



*La situazione al sedicesimo frame*

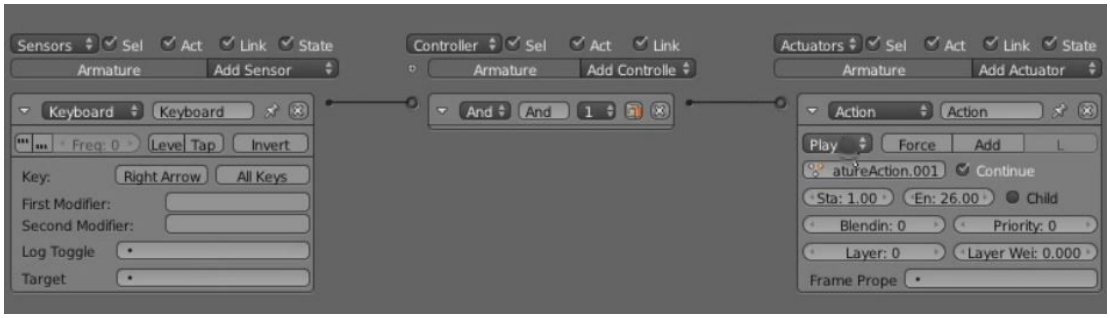
Andiamo al 26esimo frame, ruotiamo di 30 gradi nel verso opposto ed inseriamo un Keyframe Rot.




*La situazione al ventiseiesimo frame*

Abbiamo così definito una Action che parte dal frame 1 e finisce al frame 26.

Apriamo quindi il Logic Editor ed impostiamo, per l'Armatura, l'esecuzione della Action con Loop Stop o Loop End dal frame 1 al frame 26 quando l'utente preme il tasto di avanzamento, che sposta la mesh e, adesso, esegue l'animazione dell'armatura.



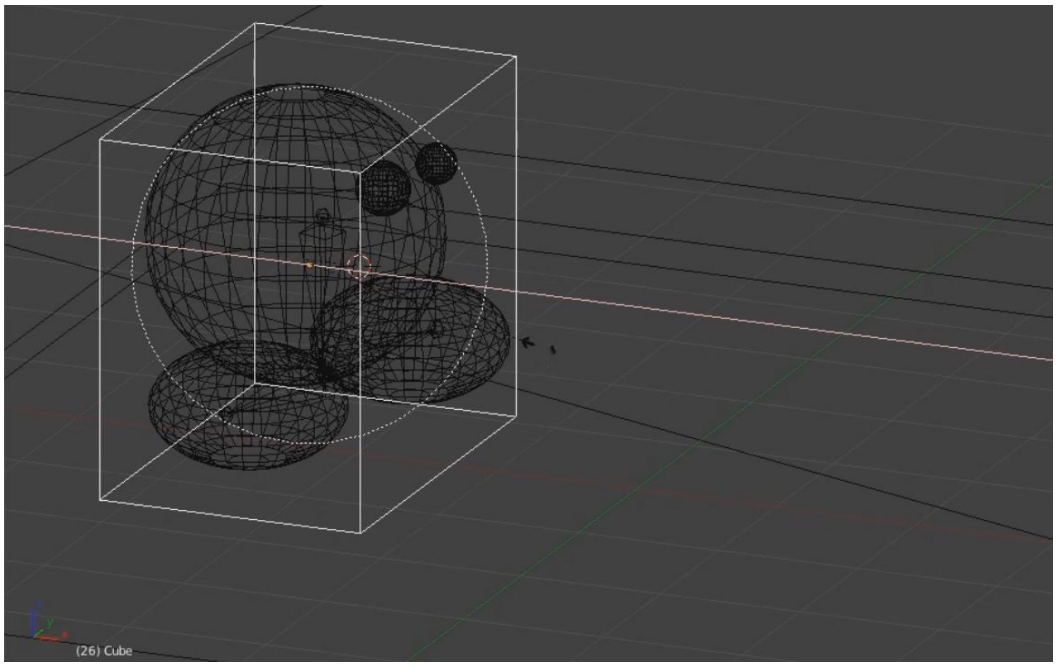

*Impostazione dell'animazione nel Logic Editor*

Facciamo un paio di considerazioni.

L'animazione così implementata non è proprio corretta: in realtà il ciclo dovrebbe prevedere una prima Action dove, alla prima pressione del tasto di movimento, si passa dalla quiete al moto, poi il moto vero e proprio ed infine, quando si rilascia il tasto, il ritorno alla posa stazionaria; in questo momento, invece, si inizia da fermi, si avanza col sinistro, poi col destro, poi si col sinistro ma poi, prima di passare al destro, si torna per un attimo allo stazionario.

Il problema non si nota molto perché l'animazione è molto veloce, ma con situazioni più complesse bisogna pianificare con cura la sequenza, eventualmente utilizzando più Actuators, verificando condizioni (ad es: se premo ed ero fermo, fai la Action 1, con l'avvio del movimento, ma se premo e già la proprietà "in moto" era vera, effettua la Action 2, che magari è solo la parte in moto, senza frame iniziali, della Action 1).

Lasciamo comunque la situazione così e passiamo al discorso dei bounds: i bound di collisione di armature o oggetti dotati di armature possono infatti dare qualche problemino col Game Engine, per cui posizioniamo nuovamente il 3D Cursor nel baricentro del personaggio ed aggiungiamo un Cube, scalandolo per farlo diventare un bounding box per l'oggetto.




*La nuova mesh "bounding box" per il personaggio*

Nel Logic Editor togliamo dal personaggio i comandi relativi allo spostamento in avanti, al salto (quindi anche i sensori touch) e lasciamo solo un sensore Always e un attuatore Parent che implementa costantemente la parentela col Cube, che ora è padre della mesh.

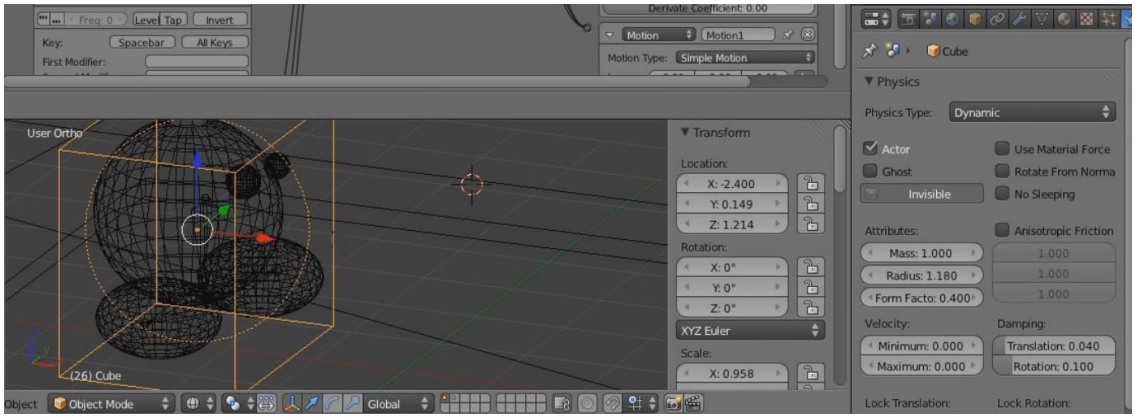
*Physics GHOST*

Impostiamo inoltre il personaggio come oggetto Static e selezioniamo anche Ghost, nella scheda "Physics" all'interno della Properties Window: adesso l'oggetto non è più sensibile alle collisioni con altri elementi.

Aggiungiamo un sensore Always e un attuatore di parentela, col cubo “bounding box” come padre, anche per l'oggetto Armature che deforma il personaggio (Armature che è di tipo Static).

Diamo quindi un'occhiata alle impostazioni del cubo “bounding box” utilizzato per le collisioni e che, a runtime, sarà padre sia dell'Armatura (a sua volta genitore del protagonista) che del protagonista, che ha quindi due genitori (anche se in realtà la relazione Armatura-protagonista è in modo Edit, per definire la deformazione, mentre le relazioni Cubo-Armatura e Cubo-Protagonista sono a runtime, nel Logic Editor).

Il cubo, che possiamo dotare magari di un Material trasparente per vedere meglio protagonista e Armatura nella 3D View in fase di editing, è di tipo Dynamic, quindi reagisce alle collisioni, con Collision Bounds di tipo Box, opzione “Compound” attivata (infatti va attivata quando l'oggetto selezionato ha dei figli che devono essere aggiunti al box di collisione principale) e soprattutto l'opzione “Invisibile” selezionata, in alto nella scheda, per cui a runtime non lo vedremo, ma gli effetti della sua presenza verranno calcolati, un po' come un buco nero che non si vede, ma i suoi effetti sì (piccolo spunto per altri giochini, volendo).



*Impostazioni del cubo - Bounding Box*

> *NOTA: tra gli attuatori c'è Visibility che consente, appunto, di far apparire o far scomparire un oggetto a comando. Modifica proprio la proprietà impostata in Edit Mode nella scheda Physics, quindi magari all'inizio un oggetto può essere invisibile ma un evento intercettato da un sensore può renderlo visibile con l'attuatore Visibility; oppure, al contrario, da visibile a invisibile deselezionando la casella Visibile nell'attuatore.*

Ora, tutte le operazioni fatte nella puntata precedente per la mesh del protagonista, come il salto con la barra spazio se si toccano solo certi Materials e lo scivolamento in avanti della mesh se si preme la freccia in avanti, vanno re-impostate qui, per questo Cube, che trascinerà con sé nel movimento (durante il quale potrà rilevare le collisioni) sia il protagonista che la sua Armatura, che nel frattempo effettuerà le deformazioni: pensateci quindi per tempo, quando realizzate sensori, controller e attuatori per un oggetto, se poi dovrete aggiungere Armature e Collision Bounds!

Ultima considerazione: le gemme e gli altri oggetti da prendere all'interno della scena dovranno reagire, ora, al Material del Cube utilizzato come Collision Bound, non più al Material del protagonista, perché ora il protagonista è "ingabbiato" e, a seconda della forma e delle dimensioni della mesh (non per forza un Cube scalato) utilizzata come gabbia, potrebbe non toccare fisicamente gli oggetti.

*  *  *

Passiamo quindi alla seconda parte, più teorica, su Messages, immagini in sovrimpressione, sensori Ray, Radar e Near.

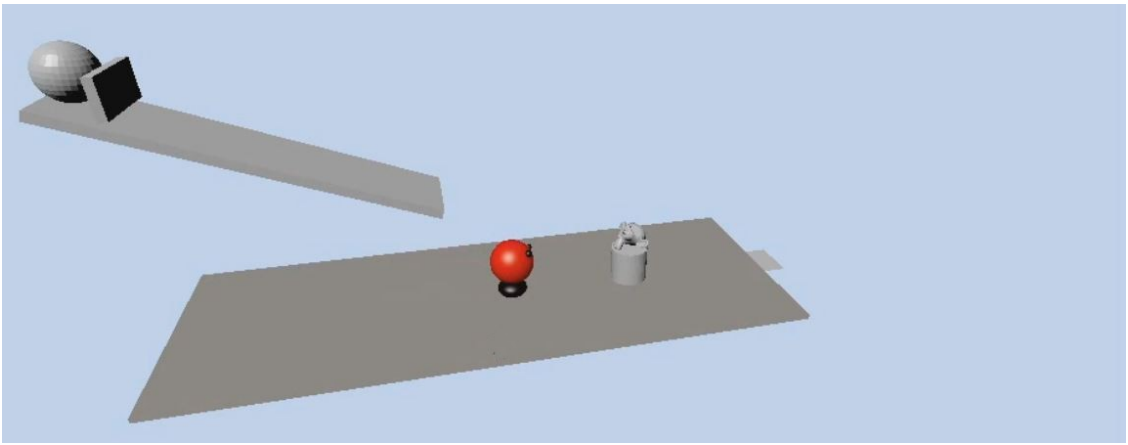
Nella puntata precedente abbiamo detto, parlando delle gemme e del punteggio del protagonista, che i sensori e gli attuatori di tipo Property si riferiscono alle proprietà dell'oggetto corrente e non possono influenzare le proprietà di altri oggetti.

C'è però un modo per risolvere questo problema e, nel mentre, aprire la strada a molte altre possibilità: lo scambio di "Messages" (messaggi) tra gli elementi.

Un Message è un'informazione inviata mediante un Actuator (quindi in seguito a qualche azione o condizione scatenata da uno o più sensori) ad un altro elemento della scena.

Per ricevere un messaggio, il secondo attore dovrà avere un sensore Message "sensibile", per così dire, ai messaggi che gli vengono inviati e in grado di filtrarli, in modo da scegliere l'azione (Actuator) o le azioni da intraprendere (un movimento, la variazione di una Property, eccetera).

Nella scena visibile nell'immagine seguente abbiamo il protagonista di prima, dotato di armatura e animazioni varie, in un ambiente con un pavimento; abbiamo poi Suzanne (posta ad una certa distanza dal protagonista, che deve raggiungerlo: in pratica, l'idolo che il nostro intrepido eroe deve raccogliere) e, sopra il protagonista, una rampa inclinata con in cima una sfera, bloccata da un peso.



*La scena utilizzata per l'esempio sui Messages*

Vogliamo far in modo che, quando il protagonista si avvicina troppo all'oggetto in fondo al corridoio, il peso si sposti per far rotolare la sfera, per cui poi il protagonista deve correre o scansarsi per non rimanere “schiacciato”.

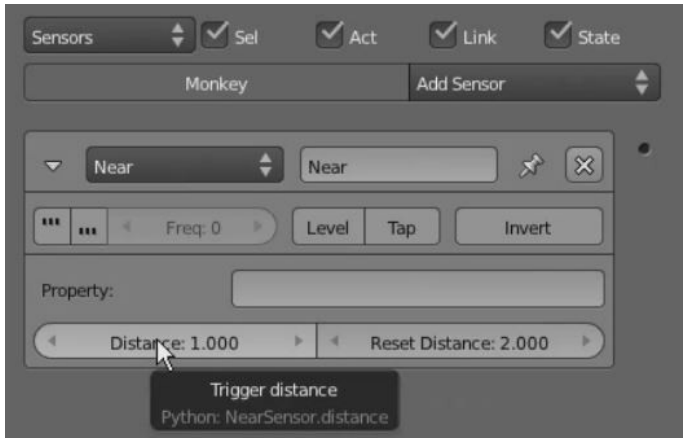
Tra le altre cose da fare, quindi, dobbiamo rilevare una distanza minima di attivazione di un attuatore, che verrà calcolata a partire dall'idolo da prendere, ed ecco che entrano in gioco i sensori Ray, Near e Radar, simili tra loro ma con alcune differenze.

Significati e utilizzi di questi sensori sono davvero molto intuitivi:

- NEAR emette un segnale positivo se un oggetto si avvicina ad una certa distanza, specificata nel campo Dist; quando un oggetto entra nel campo d'azione, il sensore inizia a emettere segnali positivi e continua a emetterli fino a quando l'oggetto non si trova ad una distanza maggiore di quella specificata nel campo Reset, resettando appunto il sensore; Near è omnidirezionale;
- RAY proietta un raggio (invisibile) fino ad una certa distanza, specificata nel campo Range, ed emette segnali positivi se intercetta un oggetto; a differenza di Near non è omnidirezionale ma vincolato ad un asse del sistema di riferimento locale, da specificare nel

menù, per cui per più direzioni dovremo mettere più sensori Ray in OR (ad esempio, una sentinella può reagire se l'oggetto da puntare si trova di fronte a lei, ossia se è "visibile", entro una certa distanza; l'esempio di una sentinella nemica è buono perché, se non si seleziona la casella della modalità X-RAY, Ray viene bloccato dagli ostacoli, ossia "non vede attraverso i muri");

- RADAR è come RAY, ossia direzionale fino ad un certo range, con la differenza che può "vedere attraverso i muri" e il raggio non inquadra tutta l'area che si trova in una certa direzione, ma solo un cono visivo lungo la direzione specificata; l'apertura del cono è data dal parametro Angle.



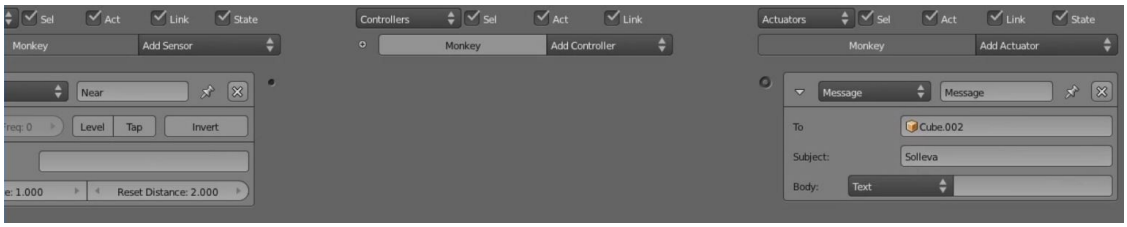
*Screenshot del sensore NEAR*

Tutti questi sensori hanno a disposizione un campo Property, vuoto di default, che limita l'attivazione del sensore agli oggetti dotati della Property specificata nel campo; se tale campo viene lasciato vuoto, tutti gli oggetti nel raggio d'azione del sensore (ed eventualmente nella direzione data) attiveranno il sensore. Tornando all'esempio della sentinella con Ray, magari questa può attivarsi solo con la Property del nemico, solo in una certa direzione (di fronte) e non può vedere attraverso i muri (se X-RAY è disabilitato, come avviene di default).

*NOTA: è possibile escludere un oggetto dal campo d'azione di sensori Near e Radar presenti nella scena deselezionando la casella Actor nella scheda Physics, all'interno della Properties Window, per l'oggetto da escludere; ciò può essere utile, ad esempio, se nella scena mettiamo delle guardie che si muovono e la loro presenza non deve sbloccare la sfera sulla rampa.*
*Un altro modo consiste, al contrario, nell'impostare le Property che possono attivare uno o più sensori Near, Ray o Radar utilizzando i campi Property, appunto, di uno o più sensori messi in OR.*

Procediamo: selezioniamo Suzanne e dotiamola di un sensore Near collegato, mediante AND, ad un attuatore Message. Nell'attuatore, impostiamo come destinatario del messaggio il peso che blocca la sfera, in questa scena Cube.002 (nome per niente significativo... non fate di questi errori).

Lasciamo, come tipo di messaggio, Text, impostando come soggetto la scritta “Solleva”.




*Il sensore NEAR e l'attuatore MESSAGE per Suzanne*

Near in questo momento rileva la presenza di qualsiasi elemento, quindi per limitare l'azione al protagonista selezioniamo non la mesh del protagonista, ma quella che rappresenta il suo bounding box, come descritto nella prima parte di questa puntata, e dotiamola di una Property booleana, selezionata (True), chiamata ad esempio “Attivazione”.

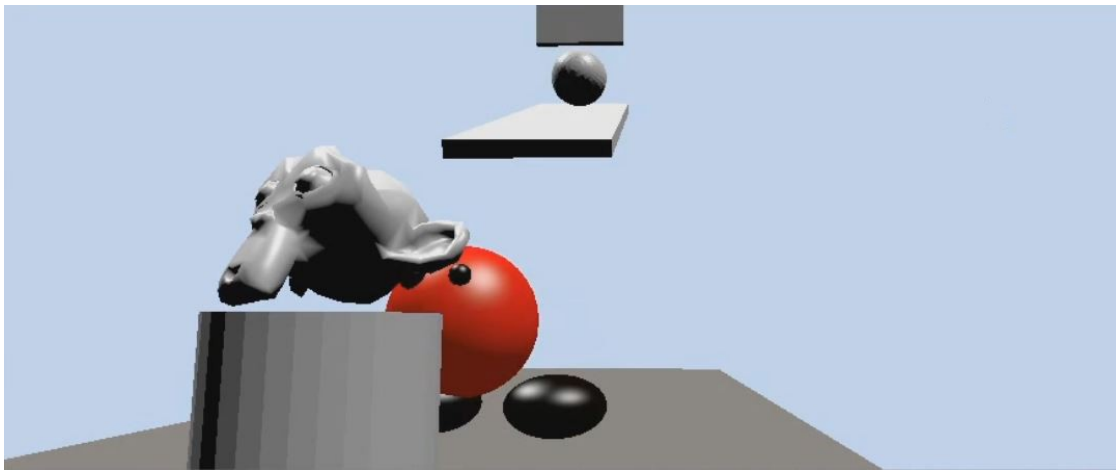
*Creazione della property booleana "Attivazione" per il Bounding Box*

Torniamo quindi in Near di Suzanne, mettendo magari distanza 15 per l'attivazione (dipende dalla vostra scena) e scriviamo, nel campo Property (che qui non mostra suggerimenti) la proprietà data al bounding box padre del protagonista, in questo caso “Attivazione”.

Selezioniamo ora Cube.002, il blocco che tiene ferma la sfera (che, tra parentesi, è un Rigid Body, per implementare gravità e rotolamento), quindi aggiungiamo un sensore Message e qui scriviamo, nel campo Subject (che non mostra suggerimenti) il nome del messaggio da recepire (nel nostro caso, “Solleva”). Attenzione a maiuscole e minuscole, in Blender.

*Il sensore MESSAGE per il cubo che tiene ferma la sfera*

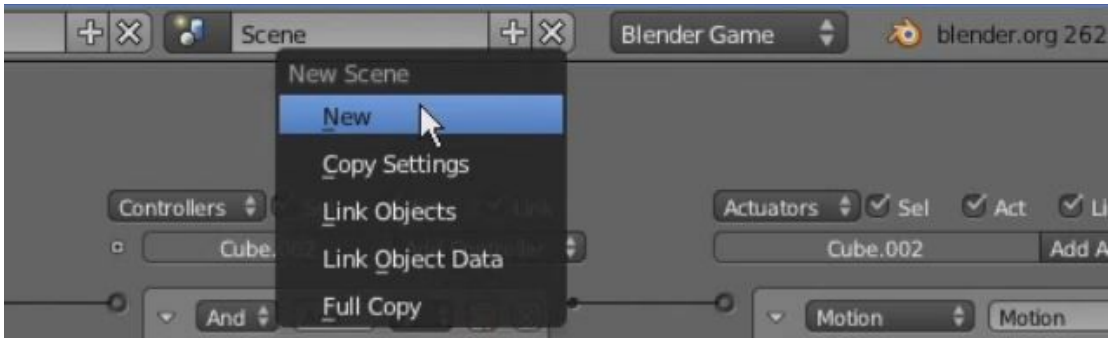
Colleghiamo questo sensore ad un attuatore, ad esempio un End Object o un Simple Motion che sposta in alto, diciamo di 0.01 unità, l'oggetto, liberando la sfera.



*Screenshot dell'esecuzione del gioco*

Diamo un piccolo suggerimento al giocatore: quando il sensore Near viene attivato, mostriamo la scritta “Attento” o qualche altra immagine in sovrimpressione.

Per mostrare elementi in sovrimpressione dobbiamo aggiungere una seconda Scene al file .blend ed attivare anche lì il Game Engine, quindi aggiungiamo una nuova Scene dalla finestra Info, chiamandola ad esempio HUD, e selezioniamo il Game Engine per tale Scene.



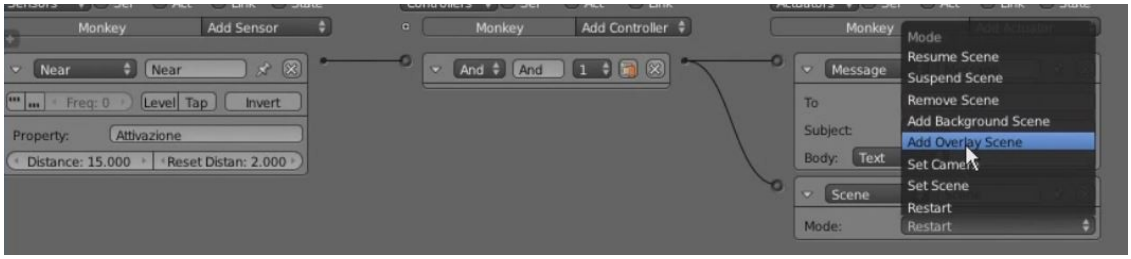

*Creazione della Scene "HUD"*

È necessario inserire un oggetto telecamera in questa nuova Scene.
La posizione e l'orientamento non sono importanti (anche se in realtà dovremo posizionare gli elementi nell'inquadratura), ma deve esserci.

In questa scena mettiamo, come esempio, un testo 3D; ovviamente possiamo mettere altri elementi, come oggetti 3D in movimento con sensori, controller e attuatori, oppure Planes con Textures immagine Alpha-mapped, eccetera... questo è solo un esempio, anche abbastanza banale.

*La creazione del Text 3D nella scena "HUD"*

Torniamo adesso alla scena principale.
L'attuatore che ci interessa è Scene con modalità Add Overlay Scene; poiché deve essere attivato quando sblocchiamo la sfera, selezioniamo Suzanne, aggiungiamo un attuatore Scene e colleghiamolo all'output dell'*AND* che attivava Message, per cui adesso all'attivazione di Near verrà inviato un messaggio a Cube.002 e verrà mostrata la Scene *HUD* in sovrimpressione; ovviamente, dobbiamo specificare il nome della scena da visualizzare in sovrimpressione (nel nostro caso, HUD) nell'attuatore *Add Overlay Scene*.




*Attuatore SCENE in modalità "Add Overlay"*

Questa tecnica può essere utilizzata (magari con immagini con zone trasparenti) per mostrare ad esempio la barra dell'energia, un mirino, un HUD, scritte informative, eccetera... le applicazioni sono davvero tantissime.

Non ci resta che provare il tutto.

*Screenshot del risultato finale*

*  *  *

# Le basi del Game Engine, il motore di gioco di Blender 2.6 – 7

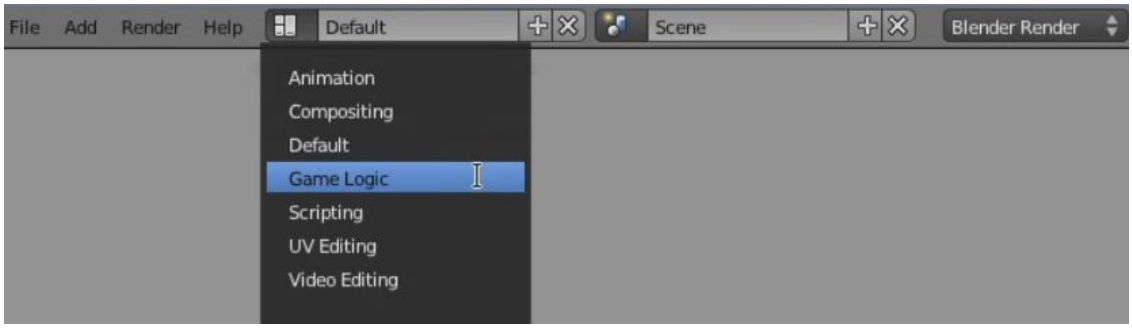
Eccoci alla settima puntata del corso sulle basi del Blender GE, il Game Engine, motore di gioco di Blender 3D 2.6. In questa puntata parleremo di come muovere la visuale dell'osservatore mediante il mouse, come avviene ad esempio negli sparatutto o nelle applicazioni di tour virtuale; vedremo inoltre come impostare un Timer per effettuare delle azioni con un certo ritardo rispetto a un evento o stimolo e come registrare su IPO in Blender le animazioni realizzate con il GE.

Iniziamo con un tema che interessa molti ma che non è di facile realizzazione: la visuale stile First Person Shooter, ossia mediante movimento del mouse. Sveliamo l'arcano: si tratta di utilizzare uno script Python e, purtroppo, a volte le API Blender Python subiscono delle riscritture tra le varie versioni, specialmente tra una major release e l'altra (tipo dalla 2.4 alla 2.5), per cui bisogna anche trovare lo script giusto.

Per questi tutorial sto utilizzando la versione 2.62 di Blender; potete scaricare lo script adatto per implementare la vista mobile col mouse per questa versione del Game Engine all'indirizzo *www.redbaron85.com/mouse26.zip* .

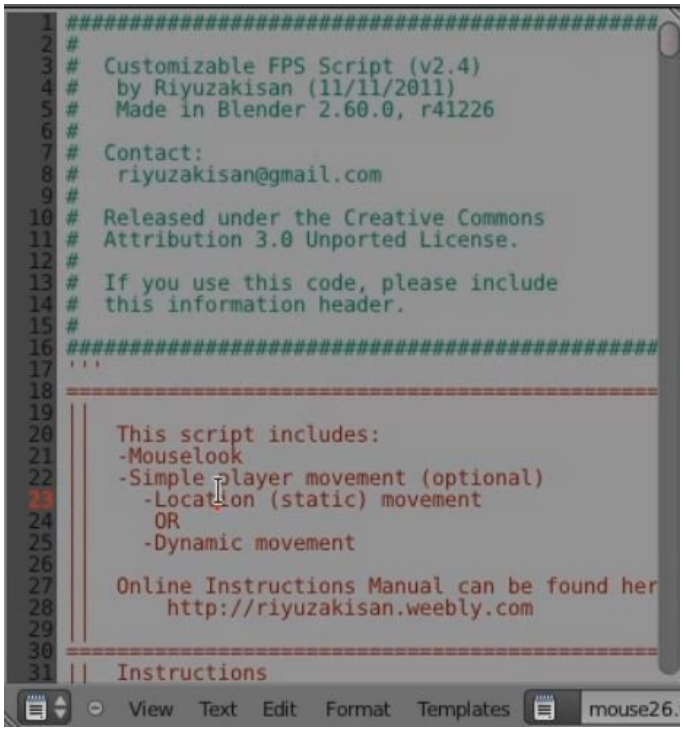
Questo script mette a disposizione diverse opzioni e parametri per configurarlo a piacere; le informazioni si trovano nella documentazione dello script, per cui qui parlerò delle operazioni e delle configurazioni di base.

Caricheremo lo script non come modulo ma, appunto, come script, quindi ci serve per prima cosa un Editor di testo. La Screen Layout *Game Logic* fa al caso nostro, per questo esempio, perché mette a disposizione in un colpo solo tutte le finestre che ci servono, quindi cambiamo il Layout della finestra in *Game Logic*.




*Selezionare la Screen Layout "Game Logic"*

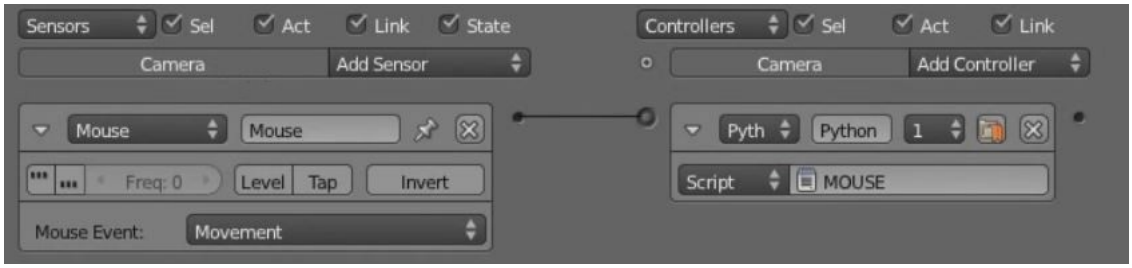
Dopo aver scompattato il file zip contenente lo script *mouse26.txt nella* cartella del progetto, apriamolo con l'editor di testo della scena di Blender con *Text – Open Text Block* e scegliendo il file da disco. Di default, questo blocco testo avrà il nome del file di testo, ma possiamo cambiarlo (specialmente se tale nome non è significativo), mediante il campo testuale nell'header della finestra.




*Lo script caricato in un Editor TEXT*

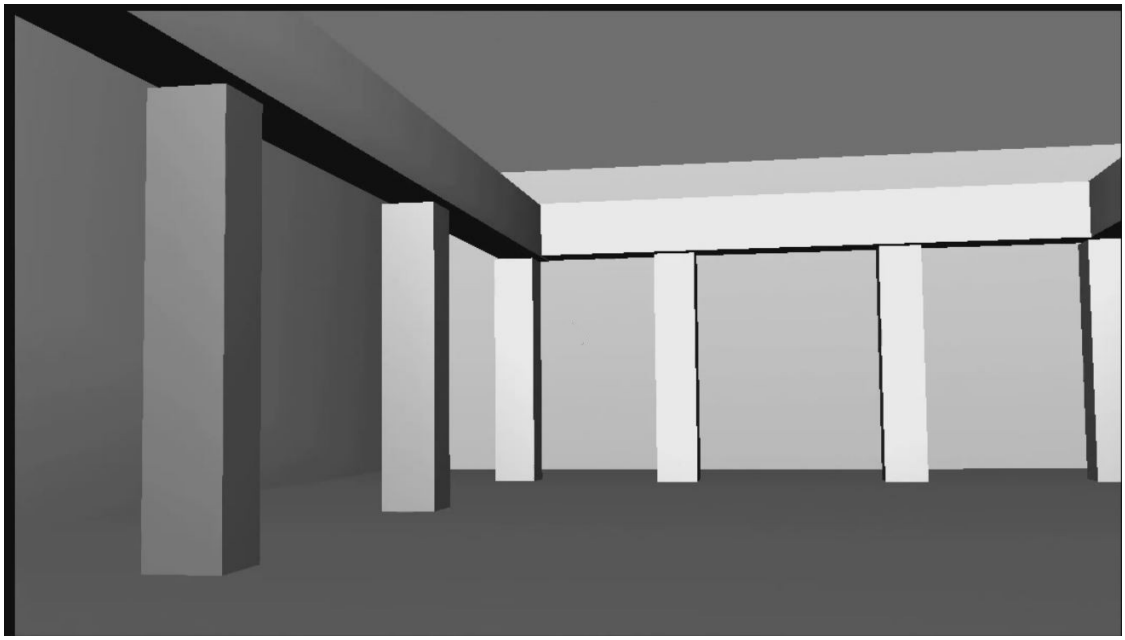
Lo script va attivato per l'oggetto che deve essere orientato mediante il mouse, quindi selezioniamo nel nostro caso la telecamera e, in una finestra Logic, aggiungiamo un sensore Mouse, cambiandone il tipo in Movement (movimento) e collegandolo ad un controller Python.

In questo controller specifichiamo il nome del blocco testo con lo script nel campo testuale (se facciamo click su tale campo, apparirà un menù a tendina con i nomi degli script caricati, nel nostro caso solo *mouse26.txt*, per aiutarci a selezionarlo velocemente senza dover digitare il nome).




*Sensore MOUSE e Controller PYTHON*

Tecnicamente è tutto pronto: possiamo avviare il gioco e muovere il mouse, osservando gli spostamenti della visuale dell'osservatore. Ricapitolando, ci serve solo un sensore che rileva il movimento del mouse e passa lo stimolo allo script, invocato mediante un controller Python (non servono attuatori) e possiamo ruotare l'oggetto intorno alla sua Origin (nel nostro caso, la telecamera, quindi il punto di vista dell'osservatore).



*Screenshot dell'esecuzione*

Dal momento che possiamo applicare questo script a tutti gli oggetti della scena, possiamo ad esempio variare l'inclinazione del cannoncino visto in una puntata precedente o ruotare oggetti e modelli in un museo 3D virtuale, e così via; ovviamente, possiamo utilizzare altri sensori in *AND*, ad esempio per controllare una proprietà, e decidere per quale elemento della scena attivare il sensore, in base a determinate condizioni; ad esempio, di default il mouse può controllare l'osservatore ma, se questo si avvicina a degli oggetti e preme un pulsante, il controller può

riguardare gli oggetti, così l'osservatore potrà ruotarli ed esaminarli (nella prossima puntata vedremo come gestire gli "stati", *states*, dell'applicazione, che ci consentono di realizzare tali operazioni senza dover creare Properties e controlli ad hoc).

Come anticipato, ci sono varie opzioni disponibili in tale script e possiamo scoprirle leggendo le note presenti nel file, che adesso è aperto nel blocco testo.

Scopriamo, ad esempio, che se per l'oggetto con il controller inseriamo una Property di tipo intero con nome "Adjust" possiamo specificare la sensibilità del mouse, ad esempio 4 o 5 (e quindi l'utente può specificarla premendo il relativo tasto o una combinazione di tasti intercettati da un sensore keyboard), oppure possiamo abilitare o disabilitare lo script con una proprietà booleana chiamata "Enable", e molto altro ancora...

Come avrete intuito, il sensore Mouse ci consente di intercettare vari eventi legati al mouse, come ad esempio lo scorrimento in alto o in basso della rotellina, il passaggio del mouse su un elemento o la pressione di uno dei tasti.

Mettiamo ad esempio Mouse Movement con collegato lo script *Mouse Look* per la telecamera, quindi inseriamo un oggetto (giusto per fare un esempio, una mesh di base, l'importante è che sia di tipo Dynamic) nel secondo Layer, poi torniamo al primo Layer e, con la telecamera selezionata, aggiungiamo un sensore Mouse Left Button (cioè click del tasto sinistro), collegandolo ad un attuatore Add Object che inserirà l'oggetto del secondo Layer nella scena (con velocità iniziale negativa su Z locale, ad esempio -10 con L selezionato in Linear Z, in modo da "sparare" l'oggetto in avanti). Colleghiamo sensore e attuatore, ovviamente, mediante un controller *AND* o *OR*, non con Python o altri.

*"Mouse Left Button" e "Add Object"*

Questo strumento, unito al piccolo spunto che vi ho appena dato, è la base per la costruzione di un FPS, anche se stand-alone, in quanto sapete già come muovervi nella scena, aggiungere oggetti, rilevare collisioni (quindi anche impatti di proiettili su altri oggetti), gestendo i punteggi o altri eventi con le Properties e vari sensori o attuatori.

Chiudiamo quindi questa prima parte di questo tutorial e apriamo una scena nuova per parlare del secondo argomento: l'utilizzo delle Properties di tipo Timer per realizzare degli eventi dopo un determinato periodo di tempo.

*  *  *

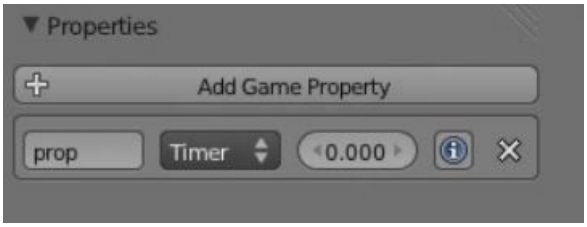
*Timer* non è il nome di una proprietà o di un attuatore, ma un tipo di Property.

Una proprietà Timer viene avviata correttamente quando l'oggetto che la possiede è presente nella scena dall'inizio (ovvero, non viene creato e aggiunto con Add Object).

La proprietà ha un valore iniziale specificato nel campo numerico associato alla Property (di default, 0.0); inoltre, viene incrementata automaticamente da Blender, ma può essere facilmente resettata, come vedremo.

Per prima cosa, proviamo una Property Timer di base ed approfittiamone per visualizzare il suo valore con l'opzione *Show Debug Properties*, che mostra a video, durante l'esecuzione, i valori delle Properties con la casella "i" selezionata.

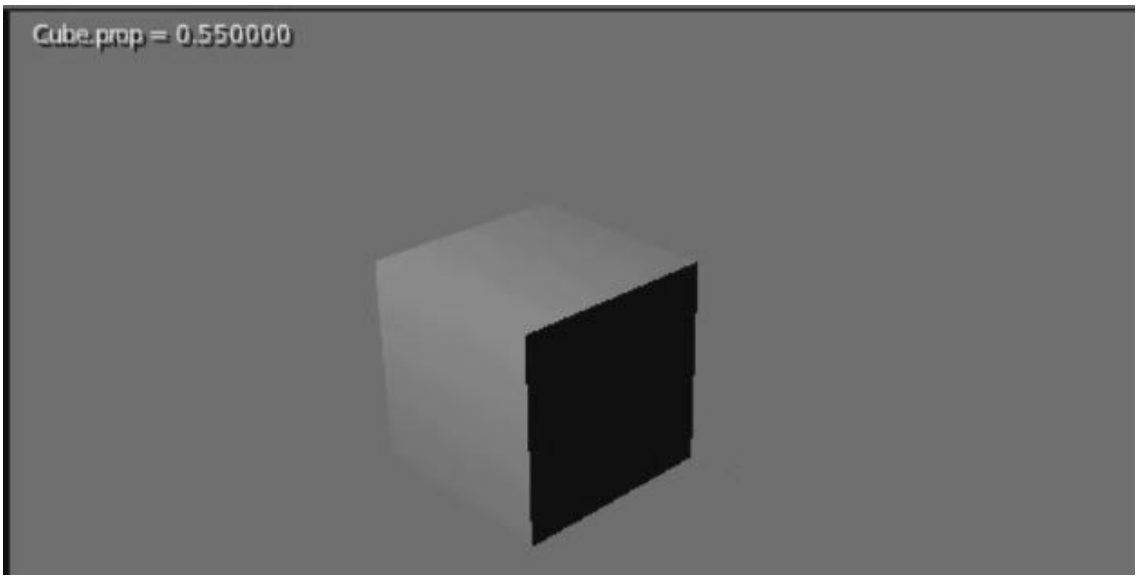
Selezioniamo il cubo presente nella scena e, nel Logic Editor, aggiungiamo una Property di tipo Timer, lasciando tutte le impostazioni come di default.



*La Property di tipo TIMER per l'oggetto*

Nella finestra Info, dal menù Game scegliamo "*Show Debug Properties*" e avviamo il gioco: non succede nulla e non vediamo il valore della proprietà Timer a video. Ciò avviene perché non abbiamo detto a Blender che ci interessa vedere tale valore: dobbiamo indicarlo esplicitamente perché di default la visualizzazione è disattivata, anche perché con giochi con centinaia di proprietà sarebbe un problema.

Nella scheda Logic Editor del cubo, quindi, selezioniamo la casella "i" nella Property Timer e avviamo nuovamente il gioco: adesso, in altro a sinistra vedremo il nome della proprietà (indicata come *nomeOggetto.nomeProprietà*, utile nel caso ci siano più oggetti con Properties dai nomi identici) e, come possiamo vedere, il valore aumenta costantemente.




*"Show Debug Properties" in azione in maniera corretta*

Il Timer è, comunque, una Property, quindi possiamo utilizzare un sensore Property per attivare degli attuatori se, ad esempio, il tempo è in un certo intervallo, oppure possiamo utilizzare un attuatore Property per incrementare, decrementare o reimpostare il valore del Timer (che comunque riprenderà subito dopo ad aumentare).

Questo ci suggerisce, ad esempio, come distruggere un proiettile dopo pochi secondi di volo (se nel frattempo non ha colpito qualcosa) o piazzare una bomba a tempo: quando si effettua l'operazione, si aggiunge un attuatore che riporta a 0.0 un Timer, facendo partire nuovamente il conteggio, quindi si utilizza un sensore che controlla se il valore è in un intervallo (Blender ha qualche problema a

intercettare il valore preciso, quindi magari mettete *Interval* con un intervallo di un secondo o poco meno); attenzione, però: date alla proprietà Timer un valore iniziale maggiore di quello dell'intervallo, altrimenti il sensore verrà lanciato almeno una volta all'inizio del gioco!

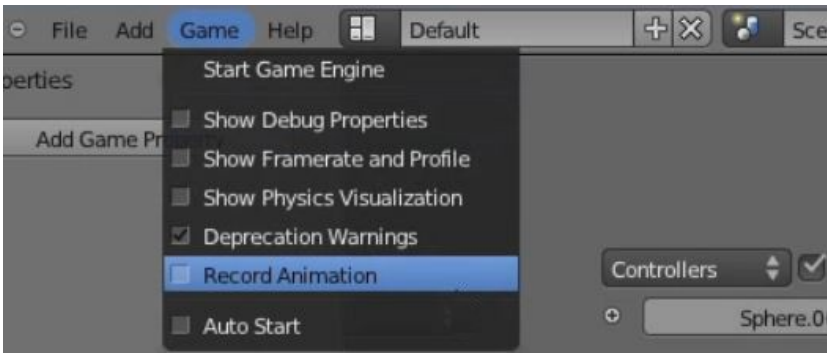
Mi spiego meglio: se la bomba deve esplodere 5 secondi dopo l'attivazione, create una Property di tipo Timer con valore iniziale, ad esempio, 10, quindi un attuatore che resetta il Timer quando piazzate la bomba e un sensore che controlla se il valore del Timer è nell'intervallo 5-6 ed in caso affermativo genera qualche evento.

*  *  *

Passiamo al terzo e ultimo argomento di questo tutorial: la registrazione degli eventi a runtime come animazioni in Blender Render.

Con il Blender GE è facile implementare simulazioni fisiche anche complesse, tuttavia quando si torna in modalità editing le animazioni non ci sono più; per fortuna, è possibile registrare le curve IPO delle animazioni generate ad ogni test in maniera interattiva sugli oggetti che le implementano e salvarle in maniera permanente nella scena in fase di editing, in modo da modificarle in seguito, ad esempio agendo sulle curve IPO o aggiungendo materiali e nodi di compositing per realizzare rendering o animazioni.

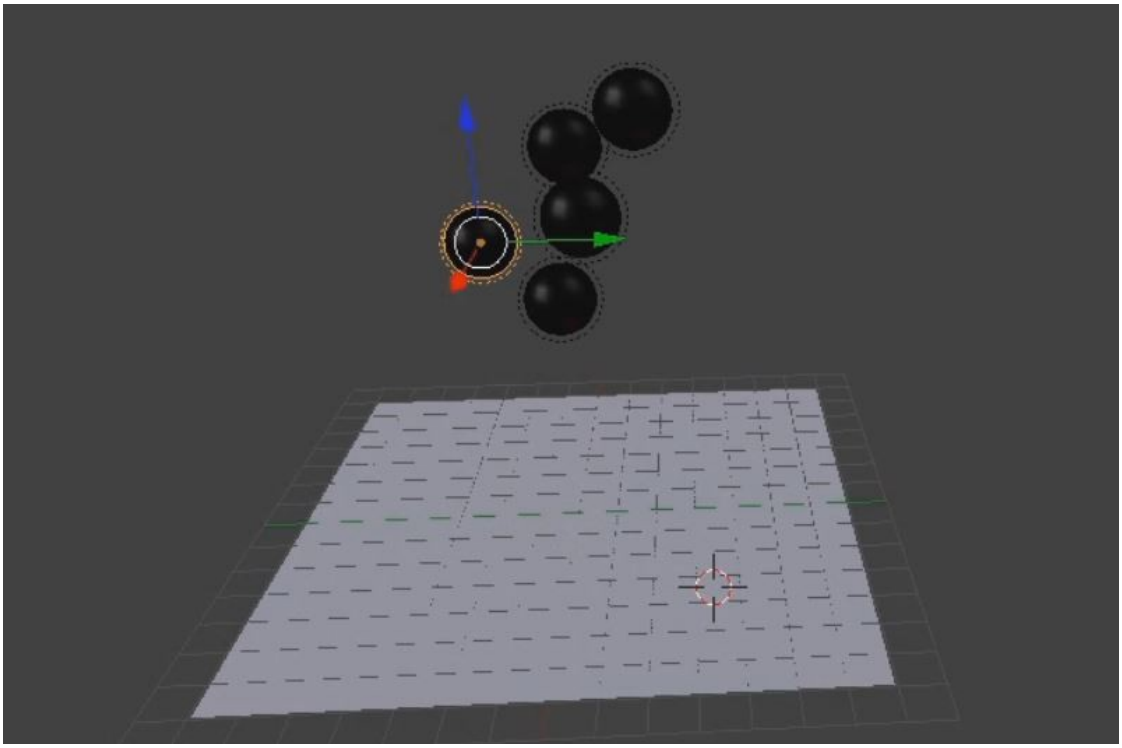
Registrare una curva IPO per gli oggetti (per TUTTI gli oggetti) di un'esecuzione a runtime è semplice: è sufficiente scegliere, dal menù Game nella finestra Info (con motore Game Engine, ovviamente) la voce "*Record Animation*".




*"Record Animation" nel menù GAME*

A questo punto, ogni volta che avvieremo il gioco Blender registrerà le IPO Curves dei vari elementi presenti (attenzione, perché ad ogni esecuzione ripartirà dal primo frame della Timeline e si fermerà quando premeremo ESC, quindi sovrascriverà le animazioni realizzate in precedenza).

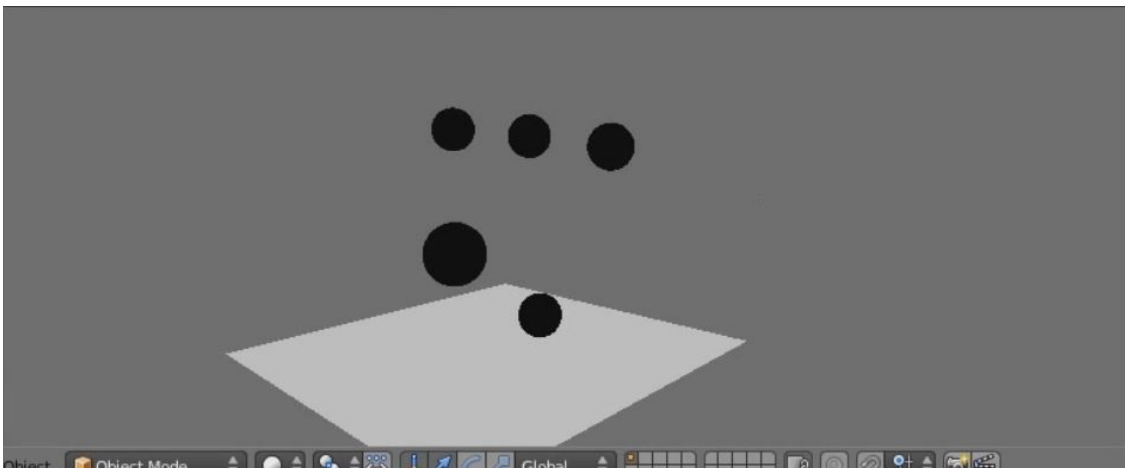
Giusto per fare un test, aggiungiamo degli oggetti Rigid Body con Material con Elasticity non nulla, sospesi nel vuoto ad una certa altezza dal Plane Static utilizzato come pavimento (e dotato a sua volta di un Material con Elasticity non nulla, per implementare dei rimbalzi, come visto in una puntata precedente).



*La scena utilizzata per gli esempi di Record Animation*

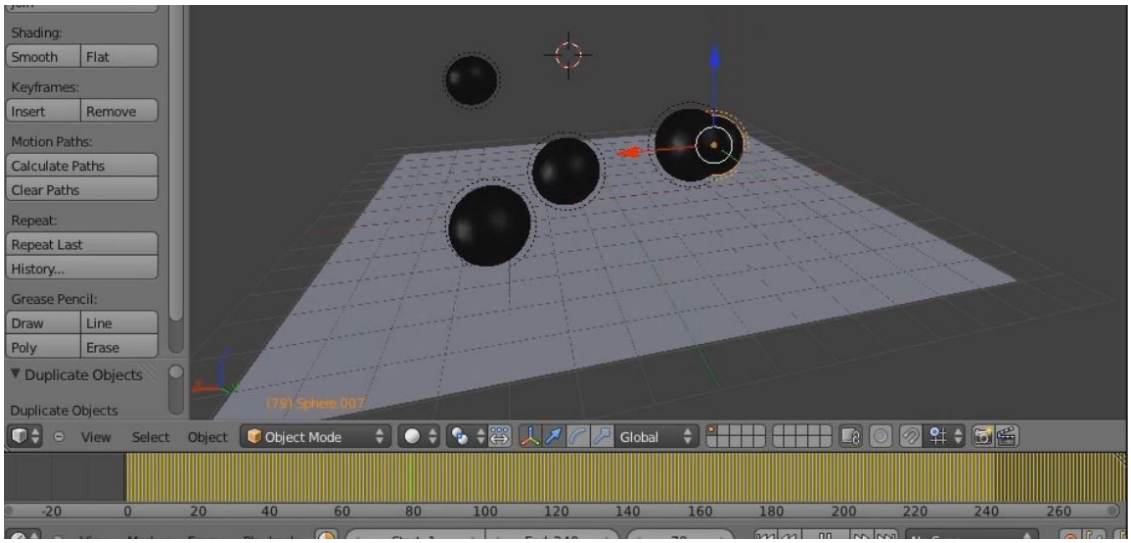
Selezioniamo Record Animation nel menù Game.

Avviamo il gioco e lasciamolo scorrere per qualche secondo, premendo poi ESC per terminare la fase interattiva. Noterete che l'esecuzione andrà un po' a scatti: ciò è dovuto alla registrazione delle curve IPO a runtime, operazione che richiede un bel po' di risorse al sistema, specialmente con scene complesse e con valori alti per Substeps, nella sezione Physics Substeps della scheda World, all'interno della Properties Window.



*Screenshot dell'esecuzione del gioco, con Record Animation attivo*

Adesso possiamo anche tornare a Blender Render (o restare nel Game Engine, è lo stesso) e premere ALT A per visualizzare un'anteprima dell'animazione presente nella scena nella 3D View.




*Anteprima dell'animazione registrata in Blender Render*

Come detto, possiamo inserire altri elementi o modificare l'animazione così generata agendo su curve IPO o altri elementi. Si tratta quindi di un buon metodo per generare animazioni fisiche (pensate, ad esempio, ad una palla da bowling, al domino o altre situazioni simili) o, perché no, animare con facilità un personaggio, infatti adesso possiamo definire le Actions per le Armatures di un personaggio e muoverlo in maniera interattiva nel Game Engine, definendo posizioni, orientamenti e tempi dell'animazione, registrando il tutto... ed ecco che abbiamo un personaggio animato da renderizzare in una scena di Blender.

*  *  *

# Le basi del Game Engine, il motore di gioco di Blender 2.6 – 8

Eccoci all'ottava puntata del corso sulle basi del Blender GE, il Game Engine, motore di gioco di Blender 3D 2.6. In questa puntata parleremo dell'attuatore Camera, degli States (con relativo attuatore) e di come salvare il gioco come eseguibile stand-alone con *Save as Runtime*.
Abbiamo citato l'attuatore Camera, anche se di sfuggita, in una puntata precedente.
Questo attuatore implementa una forma particolare di Parenting, ossia lega un oggetto (non per forza la Telecamera) ad un altro, con qualche grado di personalizzazione; ad esempio, un range di distanze minima-massima tra l'oggetto da seguire e l'oggetto Camera o una quota di riferimento.

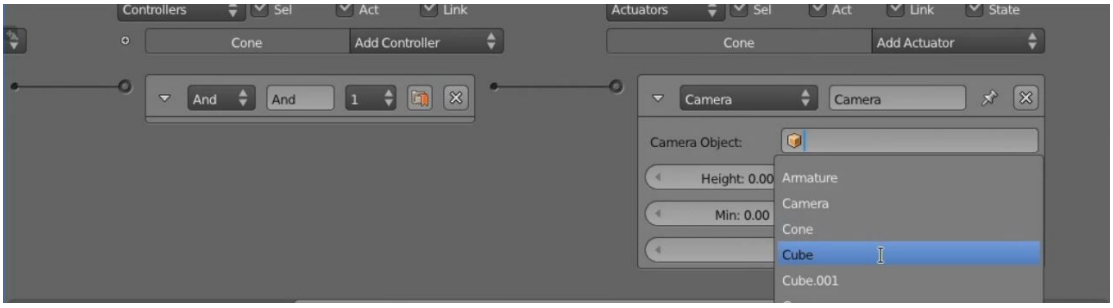
Facciamo dei test, non con la Telecamera ma con un oggetto fittizio che seguirà l'attore principale, giusto per vedere dall'esterno come si comporta un oggetto con attuatore Camera, in modo da comprendere alcuni suoi comportamenti che, all'inizio, possono sembrare strani.
Sto utilizzando una scena vista tempo fa: imparenterò l'oggetto al bounding box che serviva a rilevare le collisioni per il protagonista, quindi per prima cosa posizioniamo il 3D Cursor al centro di tale Bounding Box, in modo da posizionare lì il nuovo elemento.

*La scena utilizzata per questo esempio*

Aggiungiamo un oggetto Cone; sto utilizzando proprio un Cone perché così sarà facile capire qual è l'orientamento dell'oggetto, visto che l'asse Z verticale locale è positivo nella direzione base-punta.

Cambiamo una finestra in Logic Editor e aggiungiamo un sensore Always (cioè sempre attivo) collegandolo ad un attuatore di tipo Camera.

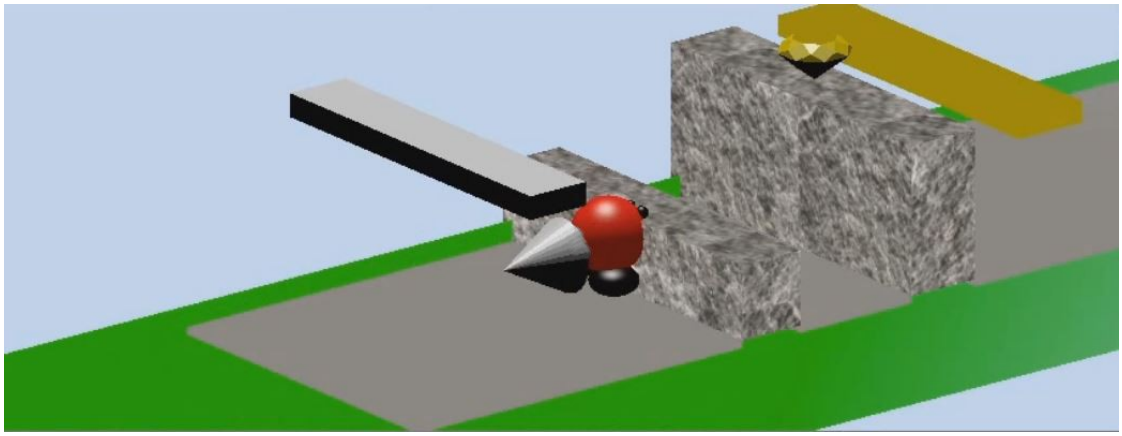
Nel campo Camera Object specifichiamo (mediante click e scelta dal menù che apparirà) l'oggetto da seguire, che nel mio caso si chiama Cube.




*Il Logic Editor per l'oggetto Cone (particolare)*

Impostiamo asse +X, il che significa che il Cone si piazzerà sull'asse X dell'oggetto da seguire... ma in questo momento ci sono un paio di cose che non vanno (e tra un attimo le esamineremo).

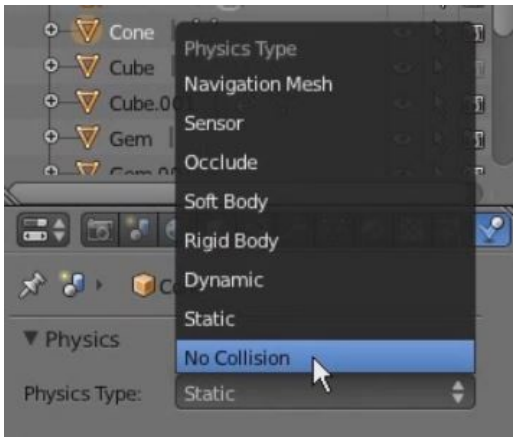
Lasciamo Height, Min e Max a 0.0, quindi torniamo nella 3D View e premiamo P per vedere cosa succede. L'esito dipende da dove abbiamo posizionato il Cono ma in genere questo, all'inizio della simulazione, si piazza con la base alle spalle del protagonista e inizia a spingerlo per tutta la scena!



*Primo esempio di esecuzione... e primo problema*

Torniamo quindi all'editing.

Il problema della spinta è dovuto al fatto che il cono è, di base, un oggetto Static, che con il movimento iniziale spinge il bounding box, di tipo Dynamic, che poi se lo tira dietro (per via dell'attuatore Camera), quindi il Cono gli sbatte addosso e lo spinge ancora, e così via.

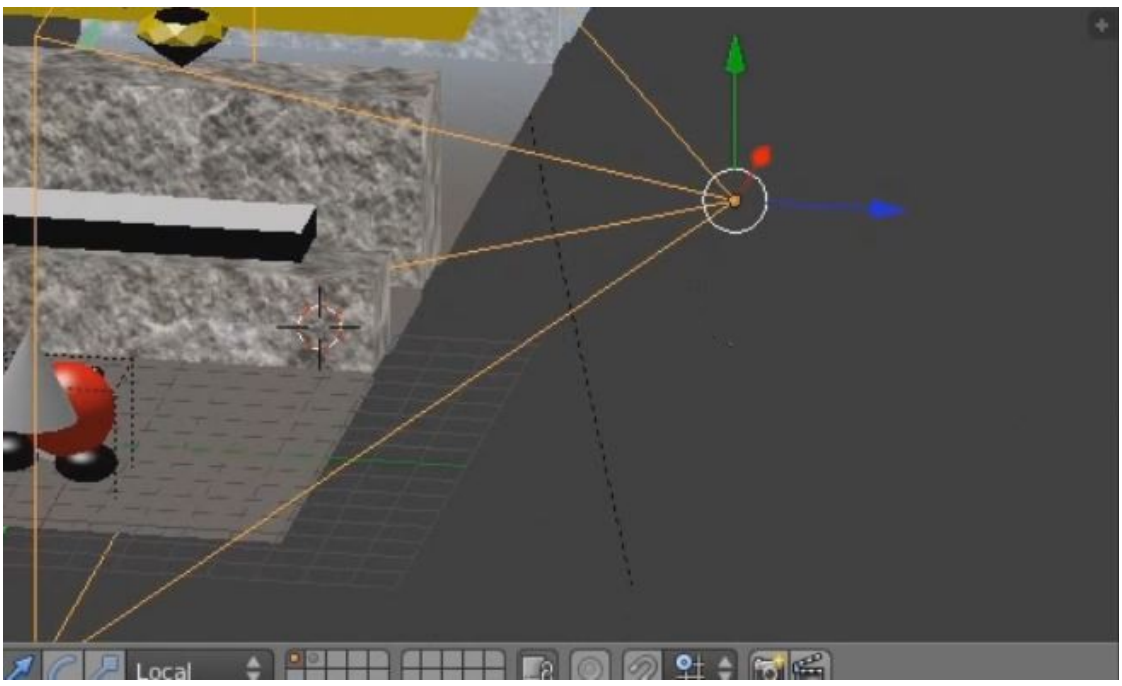


*NO COLLISIONS per il cono*

Risolviamo il problema aprendo la scheda Physics del Cono, nella Properties Window, e cambiando il tipo in No Collision.

Premiamo di nuovo P nella 3D View: il Cono ruota, come faceva prima, e si piazza dietro il protagonista... ma perché ruota? Beh il motivo è dato dagli assi del sistema di riferimento locale dell'oggetto e di quelli di un oggetto Camera.

Mi spiego meglio: selezioniamo un oggetto Camera nella 3D View.
Cambiamo la visualizzazione degli assi nella 3D View in Local e osserviamo i manipolatori di traslazione dell'oggetto Camera: l'asse che identifica la vista dell'osservatore è quello Z negativo.



*Un oggetto Camera, nella 3D View, in modalità LOCAL*

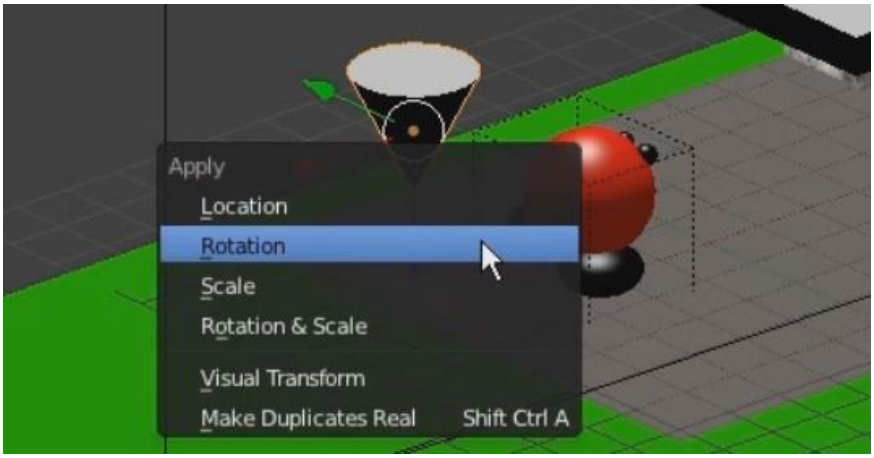
Per questo motivo (ossia perché la direzione di osservazione è Z negativa) un attuatore Camera è implementato in modo da allineare l'asse Z negativo dell'oggetto che lo possiede (nel nostro caso, il cono), con l'asse specificato nel campo Axis dell'attuatore; in questo campo, poi, possiamo scegliere tra +X e -X, e la differenza sta nel fatto che se scegliamo +X faremo puntare nella stessa direzione (correttamente, nel nostro caso) l'asse Z negativo del Cono e l'asse X positivo del protagonista.

Siccome nel nostro cono l'asse Z negativo è la base, ecco che quando premiamo P il cono viene ruotato piazzando la base alle spalle (asse X) del protagonista, con la punta indietro (asse X negativo del protagonista).

Se a questo punto vogliamo allineare la punta del Cono in modo da seguire l'oggetto, dobbiamo invertire l'asse Z di tale elemento... ma come?

Nella scena 3D, selezioniamo il cono, quindi ruotiamolo intorno all'asse Y locale di 180 gradi con R Y Y 180, cosa che avrà l'effetto di capovolgerlo.

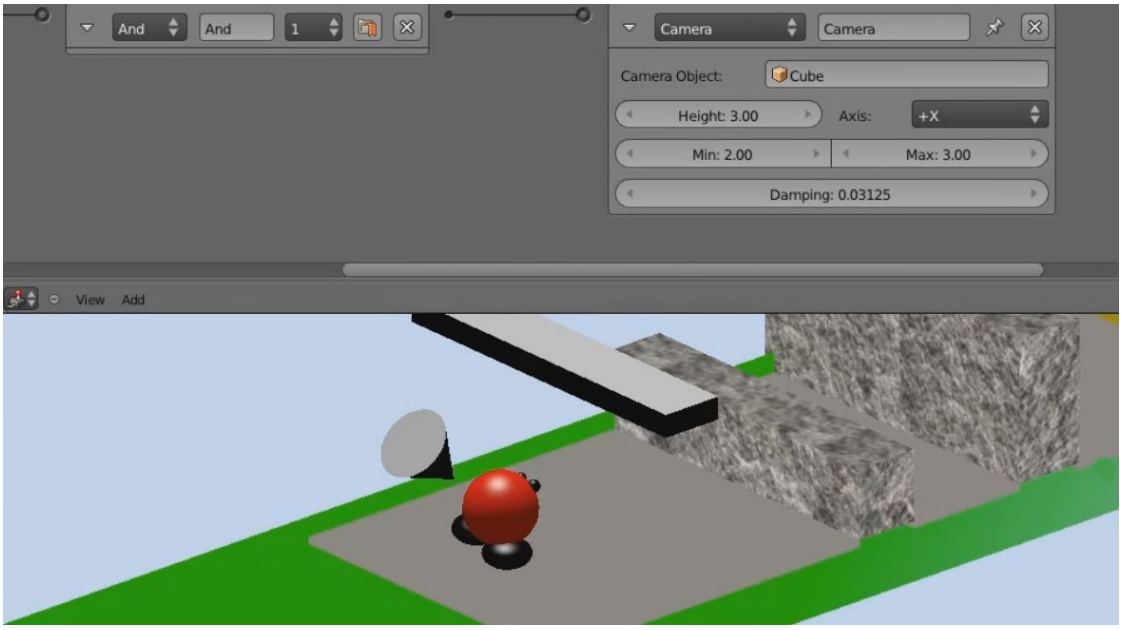
Adesso premiamo CTRL A e, dal menù che apparirà, scegliamo Rotation, per applicare il sistema di riferimento globale attuale a questa rotazione del Cono; se siamo in modalità visualizzazione Local, vedremo ora l'asse Z positivo uscire dalla base del cono, quindi l'asse Z negativo (quello della vista, per intenderci) esce dalla punta.




*Applicare la trasformazione di rotazione al cono*

Se premiamo P, il Cono viene ruotato con l'orientamento "corretto", ma c'è sempre il problema della distanza minima e della quota rispetto al protagonista. Questi problemi si risolvono nella scheda dell'attuatore, infatti se impostiamo 2 e 3 rispettivamente per Min e Max, metteremo la telecamera ad una distanza media tra 2 e 3 (i valori estremi servono a implementare un range di "tolleranza" negli spostamenti, quando l'oggetto da seguire accelera o si ferma, in modo da non fare movimenti bruschi, a scatti).

Se premiamo P, vediamo ora che il Cono sta ad una certa distanza dal protagonista, alla stessa quota della sua origine; se, ora, nella scheda dell'attuatore Camera impostiamo un valore, ad esempio 3, nel campo Height (letteralmente "altezza", "quota"), forniremo un offset (uno spostamento, in questo caso in verticale) al Cono, rispetto all'oggetto da seguire, ma niente paura: Blender orienterà correttamente l'asse Z negativo del Cono, facendolo puntare verso il target.




*Esecuzione dopo aver modificato Min, Max e Height*

Bilanciando bene assi, Height, Min e Max, quindi, possiamo posizionare la telecamera in modo tale da seguire l'oggetto da dietro, dall'alto, dalla spalla, e così via; ovviamente, possiamo anche cambiare "al volo" il punto di vista, durante il gioco... come?
Con una Property, anche numerica (un intero che identifica la configurazione della telecamera) in *AND* con il sensore Always collegato ad un particolare attuatore Camera, oppure con gli States, che affrontiamo subito.

*  *  *

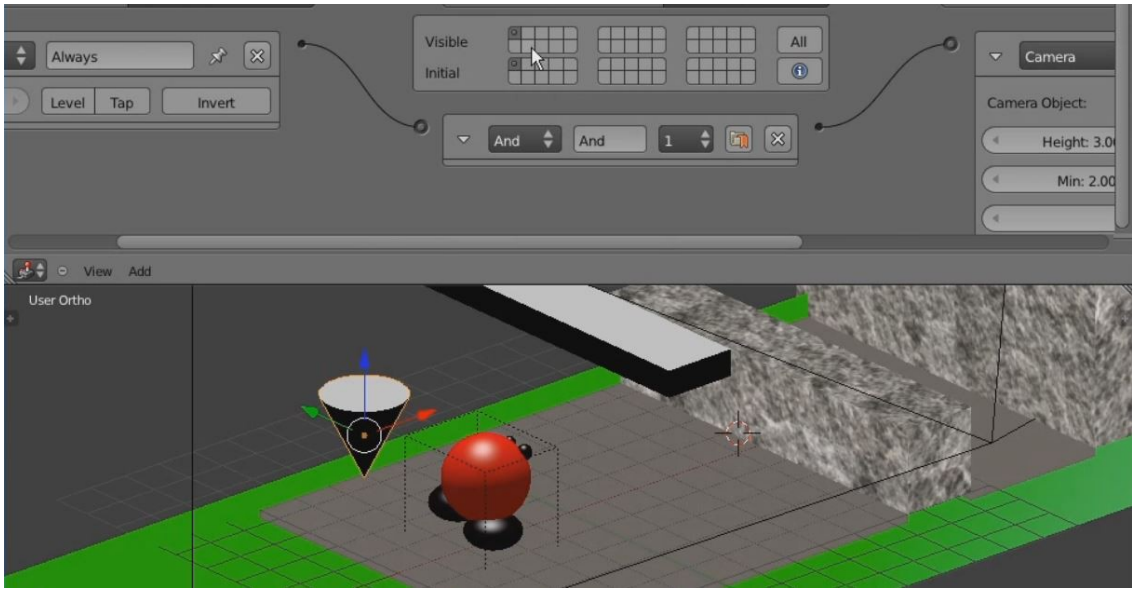
Dal punto di vista pratico, gli “States” o “stati” del gioco sono variabili che indicano quali configurazioni di sensori, controller e attuatori sono attivi in un determinato momento.

Piccolo esempio a parole (ma sono sicuro che quando vedrete l'esempio pratico capirete immediatamente): in condizioni normali, che chiameremo “stato 1”, il giocatore subisce la forza di gravità, ma ad un certo punto prende un oggetto che gli consente di vincere la gravità, cosa che nel Game Engine significa sottoporlo ad una Forza di intensità 9,8 sull'asse Z globale verticale.

Questo schema sensore-controller-attuatore, insieme ad altri, può essere attivo se si passa allo “stato 2”, determinando questo “stato 2” con l'attivazione del congegno anti-gravità; quando tale congegno viene disattivato, si torna allo “stato 1”.

Perché servono gli States e non bastano le Properties? Perché se a cambiare sono MOLTE proprietà e MOLTI schemi sensori-controller-attuatori, l'utilizzo delle Properties diventa confusionario e di difficile interpretazione, sia in fase di progettazione che di debug, mentre gli States consentono una gestione su larga scala di vari aspetti del gioco.

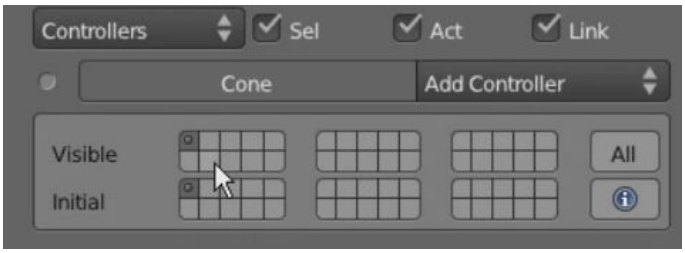
Vediamo quindi un esempio pratico: la scena che sto utilizzando è quella di prima, con il cono provvisto di un attuatore Camera che punta il protagonista con Min, Max ed Height rispettivamente a 2.0, 3.0 e 3.0.




*La scena da utilizzare per l'esempio degli States*

Di default, lo stato di un gioco appena creato è il primo... ma dove si vede qual è lo stato attuale e quali sono i suoi sensori, controller e attuatori?

È presto detto: nella finestra Logic Editor facciamo click sul cerchietto posto accanto al nome dell'oggetto attivo (nel nostro caso, il Cono) nella colonna dei Controller... ed ecco gli States.




*Dove si trovano gli States*

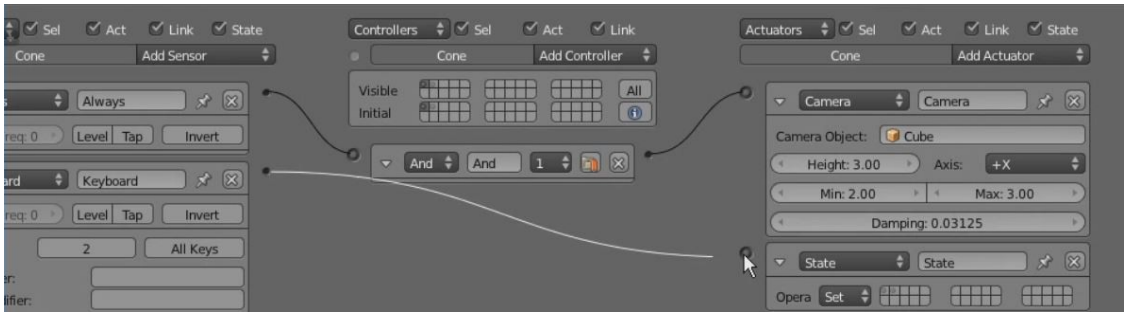
La maschera (e il funzionamento, d'altronde) è molto simile a quella dei Layers: ogni casella indica un possibile Stato, ma solo quelle con un pallino dentro hanno delle configurazioni attive (sono, cioè, utilizzate) e solo quelle premute sono attualmente visibili nell'Editor (questo almeno per quanto riguarda la linea "Visible". La linea Initial indica gli stessi States di sopra ma mostra quali sono quelli attivi all'inizio del gioco, infatti è possibile selezionare più States con SHIFT e click del tasto sinistro sulle caselle, sia in Visible che in Initial).

Di default, vediamo lo "State 1" e questo è anche lo State disponibile all'avvio del gioco.

Nella scena, selezioniamo il Cono, quindi torniamo alla finestra Logic Editor.

Facciamo click col tasto sinistro sulla casella a destra del primo stato, in Visible: l'Editor è vuoto.

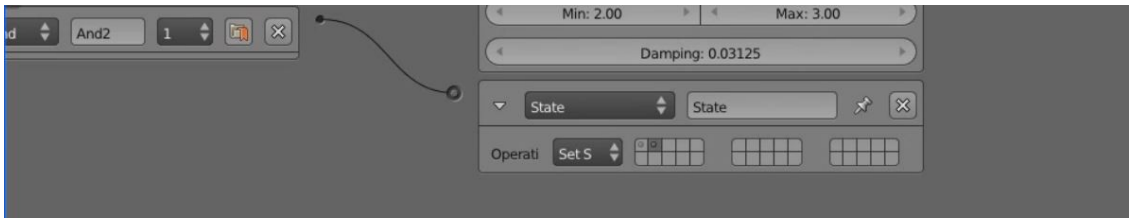
Aggiungiamo qui un sensore Always, un controller *AND* e un attuatore Camera, impostando come Target Cube e 5 per i campi Height, Min e Max.




*Sensore, Controller e Attuatore per il cono*

Torniamo al primo State, quello iniziale, e avviamo il gioco: il Cono si piazza con la configurazione indicata nel primo State e non c'è un modo per passare al secondo... non ancora. Torniamo in fase di Editing.

Nel primo State, aggiungiamo un sensore Keyboard (sensibile, ad esempio, alla pressione del tasto 2), un controller *AND* e colleghiamo tutto ad un attuatore STATE.




*Utilizzo dell'attuatore State*

In questo attuatore, possiamo specificare quali States attivare quando il sensore di attivazione viene invocato; ovviamente, possiamo selezionare più States, mediante selezioni multiple, e disattivarne altri cliccando su una casella già selezionata.

Neanche a dirlo, selezioniamo lo “State 2” e avviamo il gioco.

Se adesso premiamo il tasto 2, il Cono si sposterà, perché passerà allo “State 2” che gli dice di seguire l'attuatore con i valori impostati a 5.

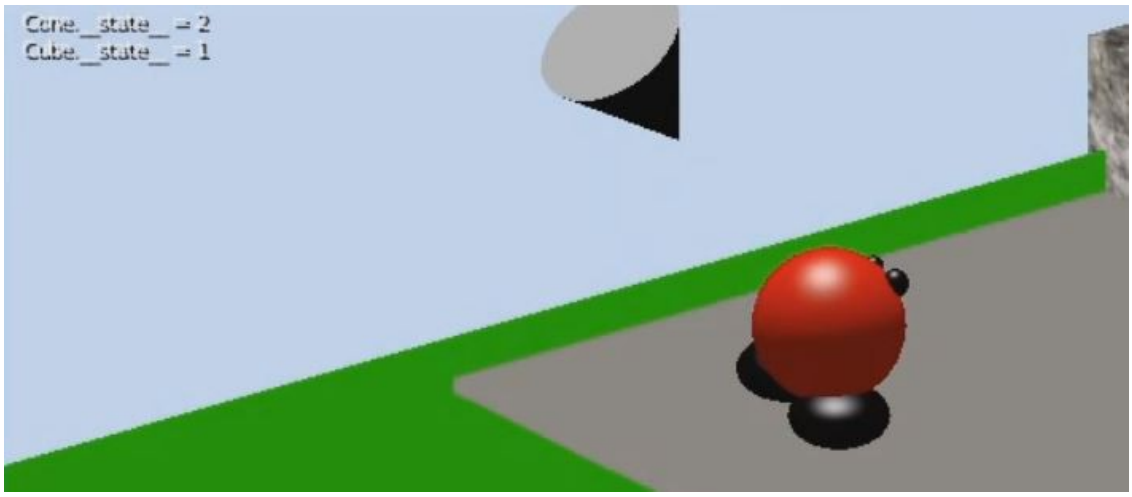
Se ora premiamo le frecce, abbiamo una sorpresa: il protagonista si sposta! Eppure i controlli di movimento erano impostati per il primo State!

Il motivo è questo: ogni elemento della scena ha la sua maschera di States. Con l'attuatore State per il Cono abbiamo cambiato solo i suoi Stati.

Per capire meglio questo concetto selezioniamo il Cono e, nel Logic Editor, selezioniamo il pulsante "i" nella maschera degli States, il pulsante cioè che mostra la proprietà se Debug Properties è attivo (quindi selezioniamo la voce *Show Debug Properties* nel menù Game, altrimenti non vedremo comunque nulla).

Adesso selezioniamo il bounding box del protagonista e anche per questo oggetto, nella maschera degli States, selezioniamo la casella "i".

Avviamo il gioco: le scritte informative in alto a sinistra ci dicono che gli States di Cubo e Cono sono entrambi a 1.




*Debug Properties per gli States*

Se premo il tasto 2, rilevato dal Cono (che attiva il cambio di stato con l'attuatore State), per il Cubo abbiamo ancora 1 (ed ecco perché può muoversi), mentre per il Cono siamo a 2.

Quanto detto offre una panoramica sul sistema degli States in Blender: possiamo evitare di utilizzare le Properties e utilizzare gli States per un oggetto, cosa che renderà il suo schema Logic di più facile interpretazione; inoltre, ciascun elemento della scena ha la sua maschera di States, quindi non c'è il rischio di alterare il comportamento di altri elementi, a meno che, ovviamente, non siano

provvisti di attuatori States attivati dallo stesso sensore di attivazione (infatti inserendo un sensore Keyboard sensibile al tasto 2 e un attuatore State anche per il cubo, premendo il tasto 2 cambieremo sia lo State del Cono che quello del Cubo).

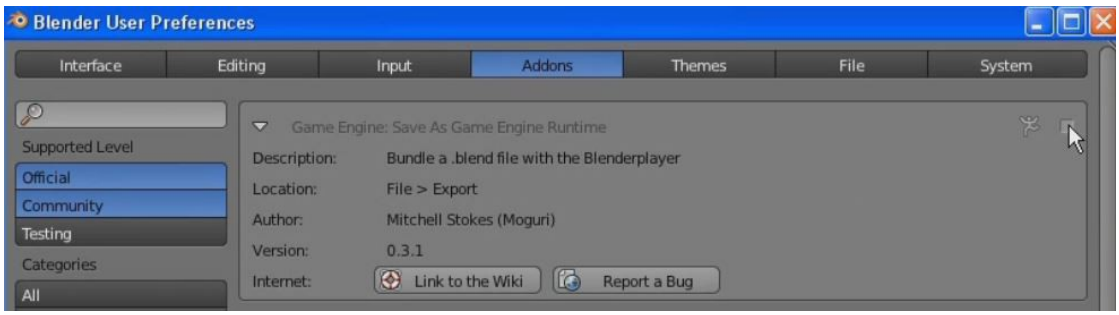
*  *  *

Restiamo in questa scena per trattare il terzo e ultimo argomento di questa puntata: il deployment del gioco, ovvero la realizzazione dell'eseguibile stand-alone dell'applicazione per la piattaforma utilizzata (nel senso che un deployment fatto su Win produrrà un eseguibile .exe, mentre per Mac, Linux o altri sistemi dovrete rifare il deployment, chiamato in Blender "*Save as runtime*", su quei sistemi).

ATTENZIONE: prima di procedere, impacchettare tutti i file delle risorse utilizzate nel file .blend attuale, scegliendo *File - External Data - Pack Into Blend File* e salvando nuovamente il file per rendere effettive le modifiche (il file .blend della scena occuperà ora molto più spazio perché porterà con sé, come un archivio zip, i file utilizzati, come le immagini per le Textures).

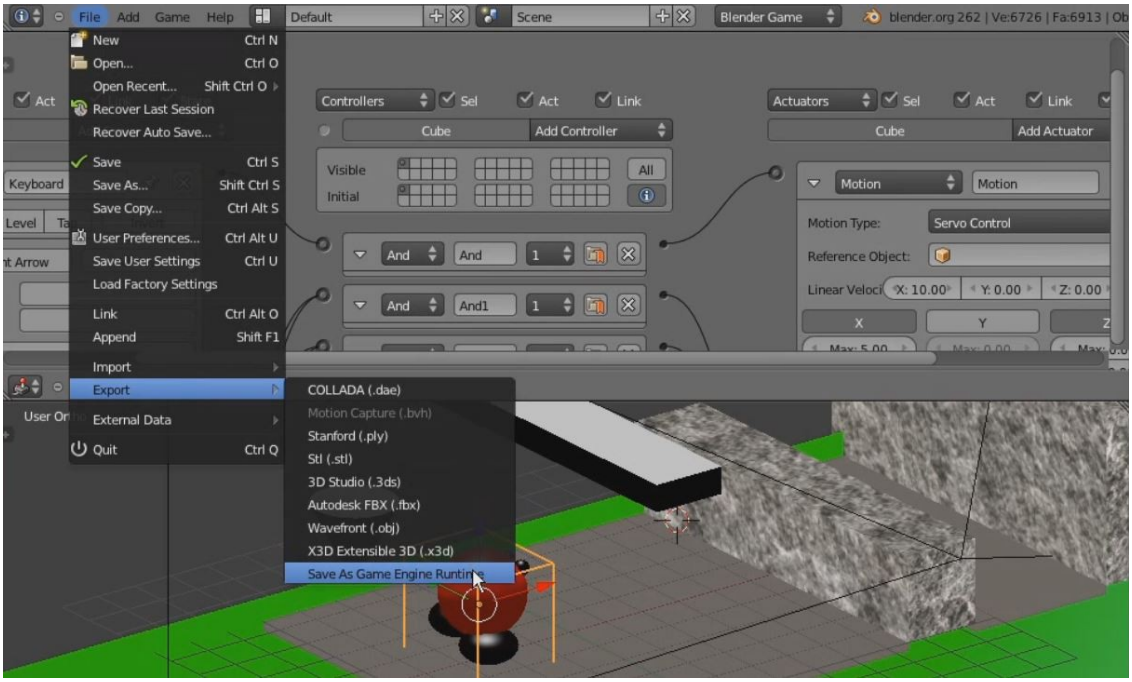
Il deployment del gioco avviene con un click sulla voce "*Save as Runtime*", solo che di default (almeno nella versione 2.62 del programma, quella che sto utilizzando) tale voce non è presente in alcun menù; in realtà, lo script di esportazione c'è, ma va attivato.

Dal menù File, apriamo la finestra User Preferences, quindi andiamo nella sezione Add-Ons, scegliamo la sotto-sezione Game Engine per filtrare un po' le voci disponibili e selezioniamo la casella relativa alla voce *Save As Game Engine Runtime*, quindi chiudiamo la finestra User Preferences.




*Attivare lo script "Save As Game Engine Runtime" in User Preferences*

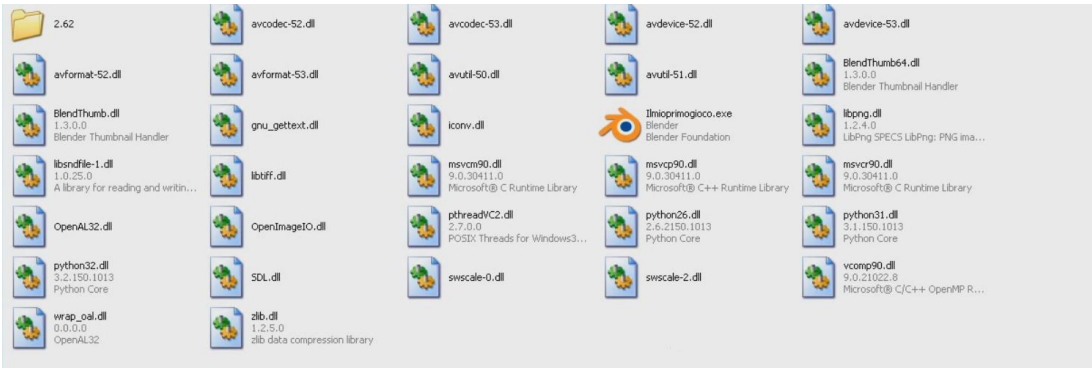
Adesso, quando siamo pronti per distribuire il gioco, scegliamo *File - Export - Save As Game Engine Runtime*; nella finestra che apparirà, scegliamo un percorso e il nome per l'eseguibile da creare. Vi consiglio di mettere l'eseguibile in una cartella a parte perché (e questo è un aspetto positivo dello script) il sistema di esportazione metterà nella stessa cartella dell'eseguibile tutti i file di libreria e risorse necessari per far partire il gioco.




*Save As Game Engine Runtime*

A seconda della complessità del gioco e degli elementi presenti, Blender prenderà un po' di tempo per effettuare il deployment, dopo aver fatto click sul pulsante "*Save as Game Engine Runtime*", nella finestra stile File Browser.

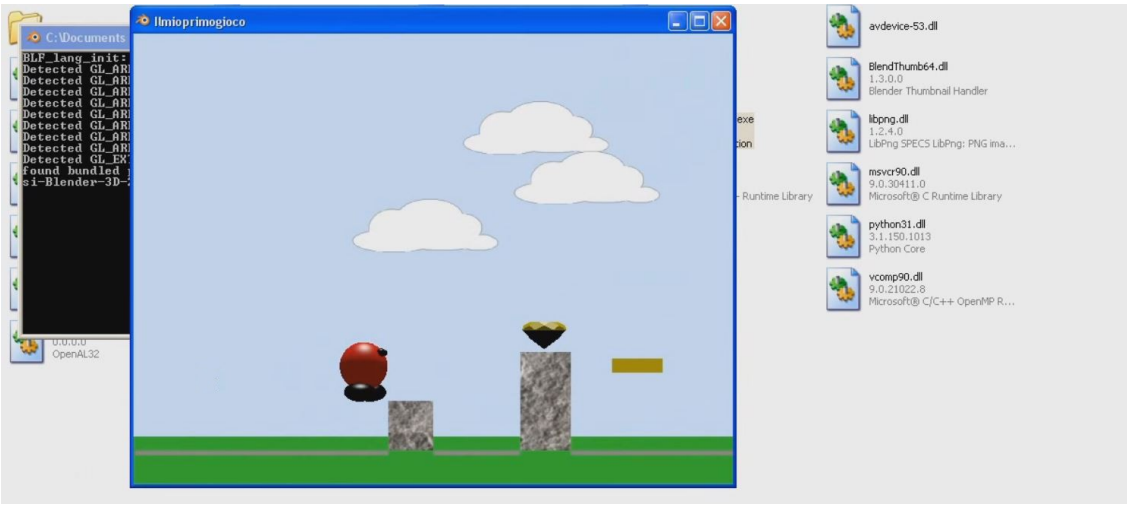
Al termine dell'operazione, andiamo a guardare i file creati su disco.




*I file creati su disco*

Nella cartella indicata come percorso troveremo il file *.exe,* con il nome che abbiamo specificato in Blender, più qualche file di libreria e altri file accessori creati automaticamente dallo script di deployment.

Per eseguire il gioco come stand-alone, facciamo doppio click sul file *.exe*... ed ecco il nostro gioco in esecuzione, questa volta "da solo", non all'interno di Blender!




*Il gioco in esecuzione come stand-alone*

A questo punto, se volete passare la vostra applicazione a qualche amico, o se volete metterla in vendita, dovrete passare tutta la cartella, cioè file eseguibile e file accessori: ecco perché vi consigliavo di mettere il tutto in una cartella a parte.

Per implementare un'applicazione o un gioco come si deve, comunque, restano un bel po' di cose da vedere, come i menù, i livelli o i modi per terminare un gioco, quindi con le puntate di questo corso non abbiamo ancora finito!

Alla fine della puntata precedente ho detto che avrei parlato, in questa, anche dell'attuatore Scene, tuttavia rimando alla prossima, visto che in quel caso parlerò anche dei menù e dei cambi di livello.

*  *  *

# Le basi del Game Engine, il motore di gioco di Blender 2.6 – 9

Eccoci alla nona puntata del corso sulle basi del Blender GE, il Game Engine, motore di gioco di Blender 3D 2.6. In questa puntata parleremo di argomenti vari, come la creazione di una "barra dell'energia", l'animazione delle proprietà dei Materials, la creazione di menù interattivi e i livelli.

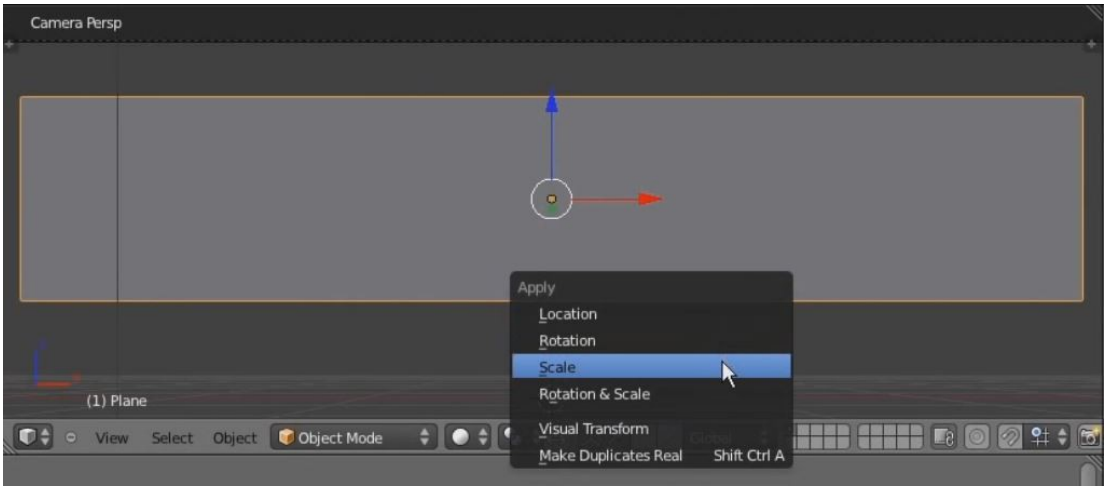
Ho detto che parleremo della barra dell'energia ma in realtà questa è una delle possibili applicazioni di una tecnica che, come vedremo, consiste nell'impostare un particolare frame di una Action (e, quindi, una particolare "posa" o configurazione di un oggetto) in base al valore di una Property.

Sappiamo infatti che possiamo eseguire delle animazioni pre-registrate in fase di modellazione mediante l'attuatore Action, impostando qui un frame di inizio e fine (per stabilire quale animazione riprodurre), e ovviamente ad un frame intermedio si avrà una configurazione intermedia; se, quindi, animiamo lo scaling di un oggetto, come un Plane che rappresenta la barra dell'energia del personaggio, dal frame 10 (energia massima) al frame 1 (energia nulla) con interpolazione lineare (cioè uniforme), allora al frame 5 avremo la barra con una lunghezza che rappresenterà circa metà energia, e per visualizzare quella configurazione non dovremo far altro che eseguire SOLO il quinto frame della Action. Vediamo come.

Per questo esempio pratico utilizzerò i tasti + e – del tastierino numerico per variare la Property "Energia", che rappresenta appunto il valore di energia del personaggio, ma va da sé che il discorso può essere esteso ad eventi vari e si possono trasmettere informazioni dal personaggio alla barra (per aggiornare il valore della Property "Energia") mediante i Messages, visti in una puntata precedente.

Aggiungiamo quindi un semplice Plane alla scena e scaliamolo in orizzontale: tale Plane, in questo momento, rappresenta l'energia massima.

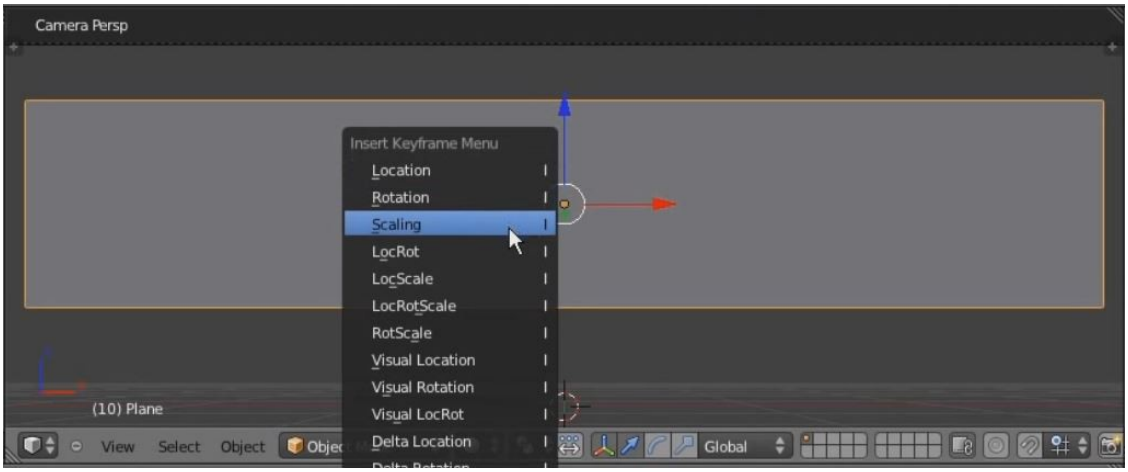
Premiamo CTRL A e scegliamo Scale per applicare il ridimensionamento all'oggetto: adesso, questa è la sua dimensione di partenza.




*Il Plane scalato, durante l'applicazione della trasformazione Scale*

Poiché vogliamo associare il valore 10 dell'Energia al frame 10, registriamo le dimensioni del Plane che rappresentano l'energia massima al frame 10 e quella dell'energia minima al frame 1.

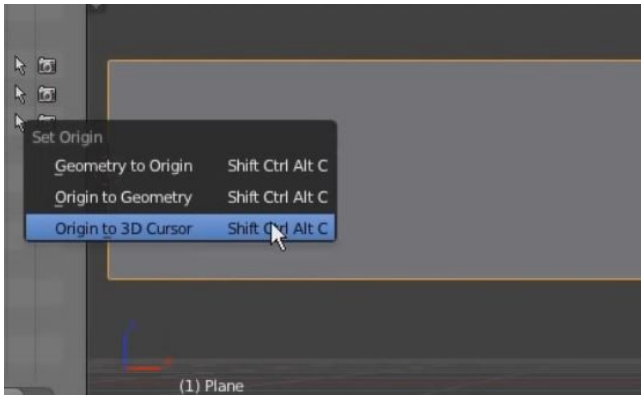
Andiamo quindi al frame 10 dell'animazione, in Object Mode, quindi con il cursore del mouse in una 3D View premiamo I e scegliamo Scaling dal menù che apparirà a video, in modo da registrare queste dimensioni.




*Registrazione delle dimensioni al frame 10*

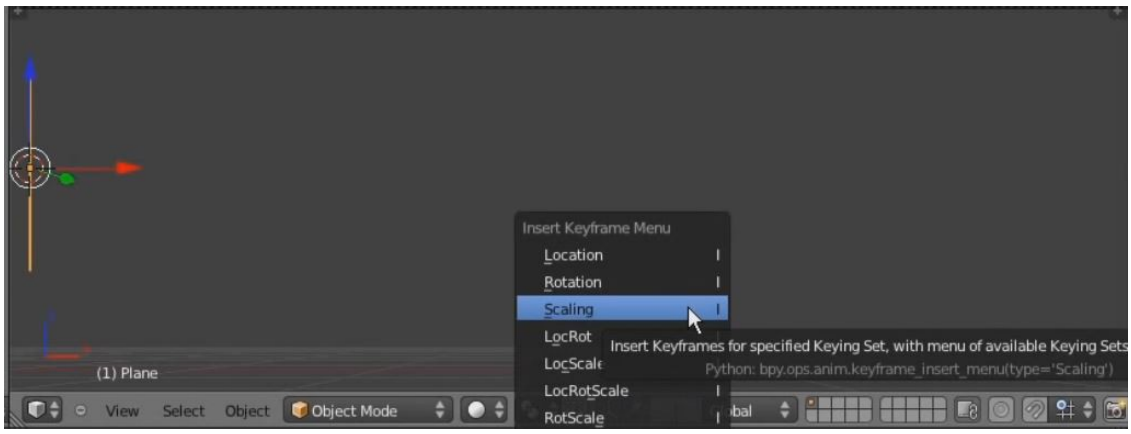
Per registrare il minimo dovremo mettere un keyframe con la barra nulla al frame 1; per prima cosa, quindi, andiamo al frame 1 dell'animazione.
In genere, in un gioco una barra dell'energia si accorcia non da entrambi i lati ma da un solo lato, mantenendo l'altro fisso, ma se scaliamo ora la mesh (sul solo asse orizzontale) la accorceremo intorno all'Origin.

Andiamo in Edit Mode, selezioniamo i due vertici dello spigolo da mantenere fisso, mettiamo il 3D Cursor nel baricentro della selezione (con SHIFT S e *Cursor to Selected*), torniamo in Object Mode e impostiamo l'Origin dell'elemento nel punto ove si trova il 3D Cursor con *Origin to 3D Cursor*.



*Impostare correttamente l'Origin*

Mentre siamo al frame 1 dell'animazione, premiamo S X X 0 (perché nel mio caso l'asse orizzontale è quello X locale, per questo due volte X), quindi inseriamo un keyframe per la barra annullata (scaling a 0, annulla la dimensione) con I – Scaling.



*Registrazione del Keyframe al frame 1*

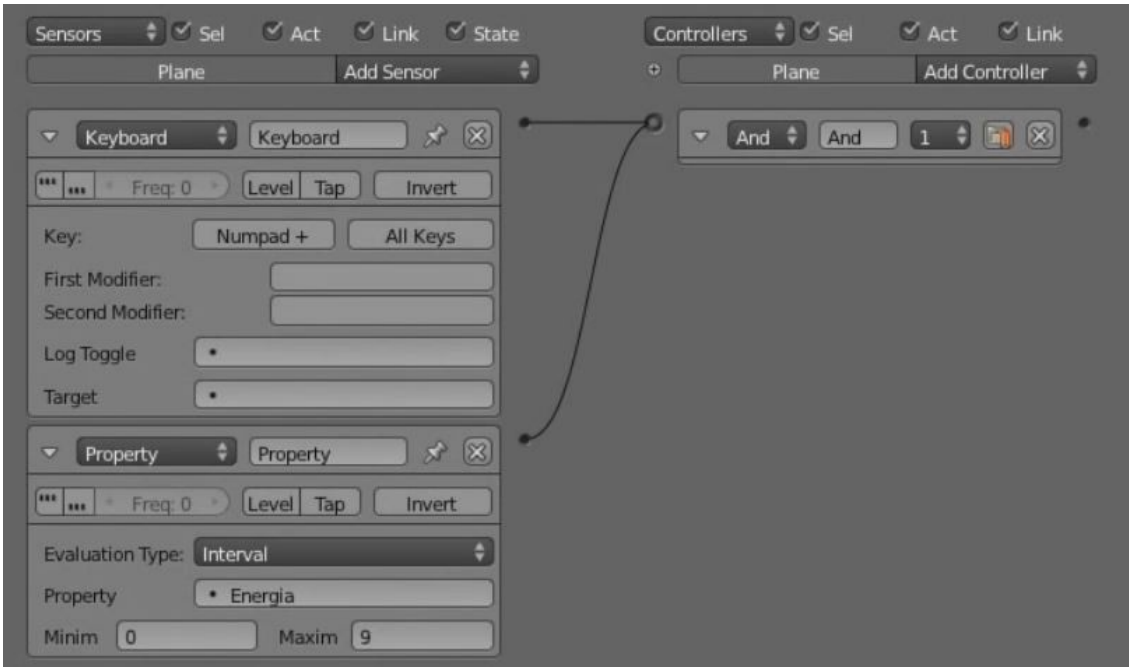
Attenzione: chi ha seguito il corso base su Blender 2.5 e 2.6 sa che, di default, Blender realizza un'interpolazione "dolce" ("smussata") in prossimità dei keyframe e sa anche che per risolvere questo problema è sufficiente aprire il Graph Editor, selezionare i punti di controllo da sistemare (nel nostro caso, gli unici due presenti) e premere V per impostare la modalità Vector.

L'animazione in Object Mode è pronta: ora vediamo come gestirla secondo le nostre esigenze in fase di gioco! Cambiamo la Screen Layout in Game Logic, per comodità.

Come anticipato, non creerò uno scenario particolare per gestire la Property "Energia", ma userò solo i tasti + e – del tastierino numerico; in ogni caso, selezioniamo la barra e inseriamo una Property di tipo Intero, con nome Energia e valore iniziale 10.

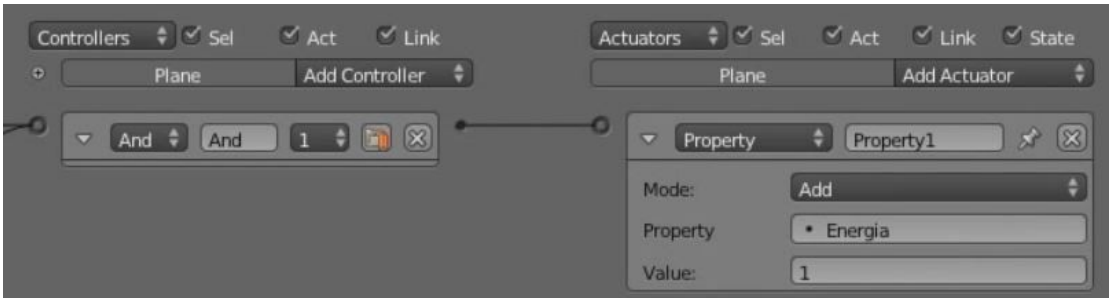
Inseriamo un Sensore Keyboard, impostando come evento la pressione del tasto + del tastierino numerico e colleghiamolo in *AND* con un altro sensore, di tipo Property, impostando "Energia" come proprietà da controllare, modalità *Interval* e intervallo 0-9: in questo modo, il tasto + avrà effetto solo se il valore dell'energia sarà tra 0 e 9, consentendo di raggiungere al massimo il valore 10.

*La Property "Energia"*



*Sensori KEYBOARD e PROPERTY in AND*

Colleghiamo il controller *AND* ad un attuatore Property, con modalità Add (somma) e valore 1 da sommare alla Property "Energia".




*Attuatore Property Add*

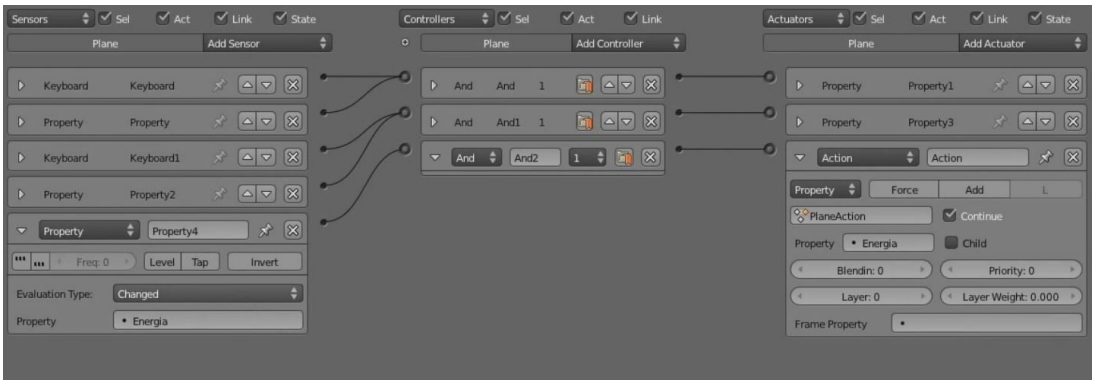
Adesso inseriamo un sensore Keyboard, impostando l'attivazione alla pressione del tasto – del tastierino numerico.

Aggiungiamo anche un sensore Property, per la proprietà "Energia" (da specificare nel relativo campo) con modalità *Interval* e intervallo 1-10.

Colleghiamo questi due nuovi sensori in *AND* e passiamo l'output ad un attuatore Property che questa volta sommerà il valore -1 (quindi Add -1) a "Energia".

Aggiungiamo adesso un sensore Property, che terrà d'occhio la proprietà "Energia", con modalità Changed, per cui questo sensore verrà attivato ad ogni cambiamento del valore della Property "Energia".

Colleghiamo tale sensore ad un attuatore Action, impostando ovviamente l'unica Action creata per il Plane e cambiando l'Action Type da Play (default) a Property; nel campo Property che apparirà, impostiamo la proprietà "Energia".




*Lo schema di Nodi con l'attuatore Action Property*

Ricapitolando: alla pressione del tasto + (che simula per noi l'acquisizione di un medikit, di un power up o un altro elemento che aumenta l'energia del giocatore), se l'energia è nei valori consentiti, la Property "Energia" verrà aggiornata; idem per la pressione del tasto –.

Ad ogni aggiornamento, il valore della Property verrà utilizzato dall'attuatore Action per mostrare il relativo frame dell'animazione registrata in fase di modellazione.

Per vedere in pratica quanto detto, avviamo il gioco e premiamo i tasti + e – del tastierino numerico (possiamo tenere d'occhio il valore della Property selezionando la sua casella "i" nel Logic Editor e *Show Debug Properties* nel menù Game).

*Screenshot dell'esecuzione del gioco*

Il secondo argomento di questo tutorial, ossia l'animazione dei Materials, è in realtà uno "spin off", per così dire, dell'argomento appena trattato, infatti è possibile inserire dei Keyframes per i vari campi di un Material (ad esempio, un valore di trasparenza Alpha, un colore, l'utilizzo di una Texture, eccetera) ed è possibile richiamare questa animazione nel Game Engine con l'attuatore Action selezionando, questa volta, una Material Action, creata automaticamente da Blender dopo aver realizzato l'animazione in fase di modellazione.

Non mostrerò quindi la procedura appena descritta applicandola al Material Action, ma vi invito a realizzarla come esercizio; in particolare, provate a regolare la trasparenza Z-Transp di un Material da Alpha 0.0 a 1.0 in 10 scatti (come nel caso della barra dell'energia) con i tasti + e – del tastierino numerico.

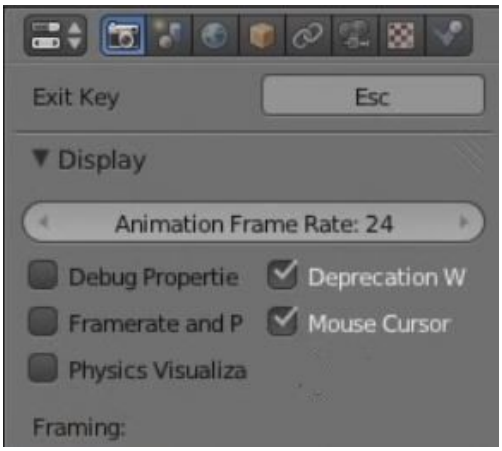
*  *  *

Passiamo quindi alla seconda parte di questo tutorial, riguardante la creazione di menù interattivi.

Il modo più semplice per creare un menù interattivo in Blender è utilizzare una Scene a parte e inserire Planes con immagini, mesh rappresentanti pulsanti ed altri elementi atti a realizzare un'interfaccia grafica: un "finto pannello 2D", quindi, realizzato in realtà con oggetti 3D messi di fronte alla telecamera (per annullare la prospettiva, si potrebbe passare ad uno State nel quale si visualizza la Scene del menù e si cambia la modalità della telecamera da Prospettiva, default, a Ortogonale).

In una puntata precedente abbiamo visto come aggiungere una Scene in sovrimpressione con Add Overlay Scene; per passare invece proprio da una Scene all'altra (nel nostro caso, da un livello del gioco al menù) si deve utilizzare l'attuatore Scene, questa volta però in modalità *Set Scene* e specificando, nel campo che apparirà, la scena da utilizzare, che sostituirà quella in uso.

La visualizzazione del cursore del mouse a runtime viene abilitata selezionando la checkbox "*Mouse Cursor*" nella sezione Display della scheda Render, all'interno della Properties Window.



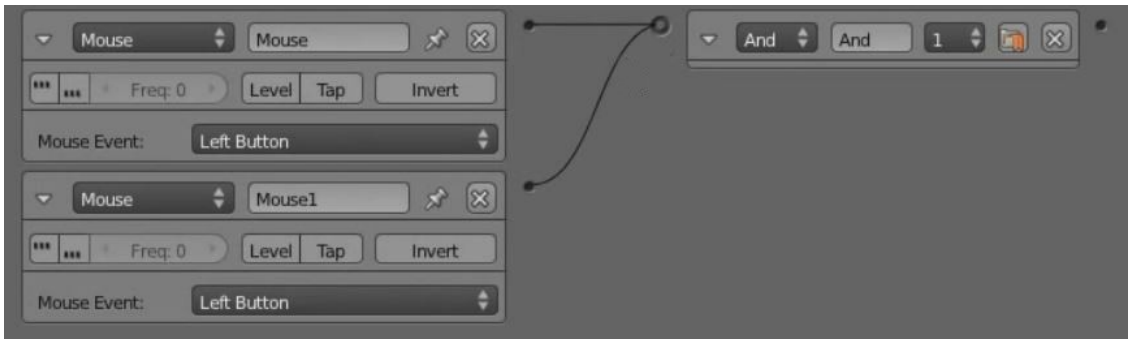
*Abilitare la visualizzazione del mouse con Mouse Cursor*

Per rilevare il click su una mesh che rappresenta una voce di menù ed effettuare delle operazioni, quindi, sarà sufficiente mettere un sensore *Mouse – Left Button* sulla mesh e collegarle l'attuatore o gli attuatori per effettuare un'azione... o no?

In realtà, c'è un errore subdolo: l'evento click del mouse non tiene conto della posizione del cursore nella scena, per cui se ci limiteremo a mettere un sensore *Mouse – Left Button* sugli oggetti che rappresentano le voci di menù, ad un click del tasto sinistro del mouse IN UN QUALSIASI PUNTO DELLA FINESTRA DELL'APPLICAZIONE attiveremo TUTTI i sensori con tale modalità e, di conseguenza, tutti gli attuatori!

Per risolvere questo problema, dobbiamo mettere in AND con *Mouse – Left Button*, per ogni oggetto che rappresenta una voce di menù, un sensore *Mouse – Mouse Over*, in modo da trasmettere il segnale all'attuatore solo quando il tasto sinistro del mouse viene premuto mentre il mouse si trova su (Mouse *Over*) QUEL determinato sensore in quel momento.

Questo è un piccolo accorgimento che, una volta noto, appare ovvio, ma lì per lì non viene in mente mentre si realizza il gioco, in quanto si pensa semplicemente ad associare l'evento del click alla mesh, osservando poi anomalie a runtime.

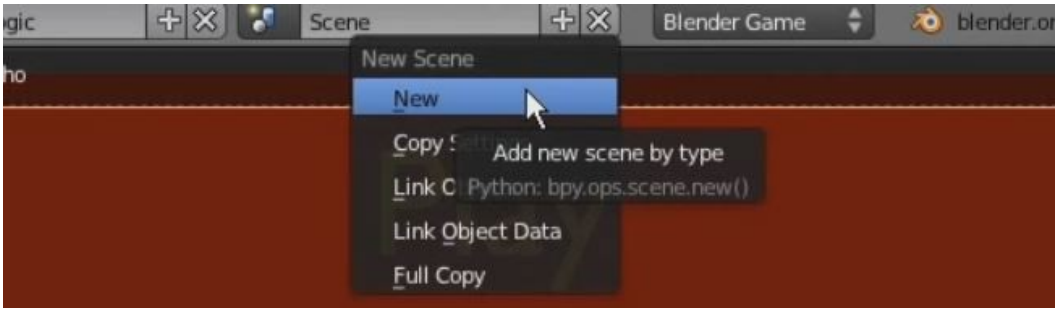


*I sensori MOUSE (Left Button e Over) in AND*

L'utilizzo dell'attuatore Scene in modalità Set Scene ci serve per parlare dell'ultimo argomento di questo tutorial: la gestione dei livelli in un gioco realizzato col Game Engine.

Non mi dilungherò molto su questo argomento, perché è abbastanza chiaro: l'attuatore Scene in modalità *Set Scene* consente, quando attivato da uno o più sensori, di eseguire l'azione del gioco in un altro ambiente, definito appunto in un'altra Scene (ad esempio, perché il giocatore ha raggiunto la fine del livello, superato un certo punteggio o oltrepassato un varco dimensionale).

Quando si crea una Scene con un click sul pulsante + nel menù Scene, nella finestra Info, si può scegliere se creare una scena nuova e completamente vuota, con New, oppure una copia di quella attuale (con Full Copy) o una copia-istanza di quella attuale, copiando (in modalità istanza) gli oggetti presenti in quella corrente, con Link Objects.




*Modalità di creazione delle SCENES*

Quest'ultima modalità è molto importante perché consente di creare altre scene importando, in questi nuovi ambienti, gli oggetti della scena precedente, con le loro caratteristiche, inclusi sensori, controller, attuatori e Materials; non solo: le modifiche fatte ad una caratteristica dell'oggetto si avranno anche nelle istanze di tale oggetto nelle altre Scene ottenute con tale metodo, in quanto si tratta di copie linkate.

Consiglio quindi di definire ad esempio un personaggio, con le sue Actions e altre proprietà, nella prima scena-livello, quindi fare una nuova scena con Link Objects in modo da partire già con il personaggio o i personaggi (e gli elementi sui Layers nascosti, per l'inserimento al volo nella scena) e procedere alla personalizzazione dell'ambiente del nuovo livello-Scene.

Il numero di Scene (e quindi di livelli) che è possibile inserire è virtualmente infinito (in realtà, dipende dalla memoria disponibile).

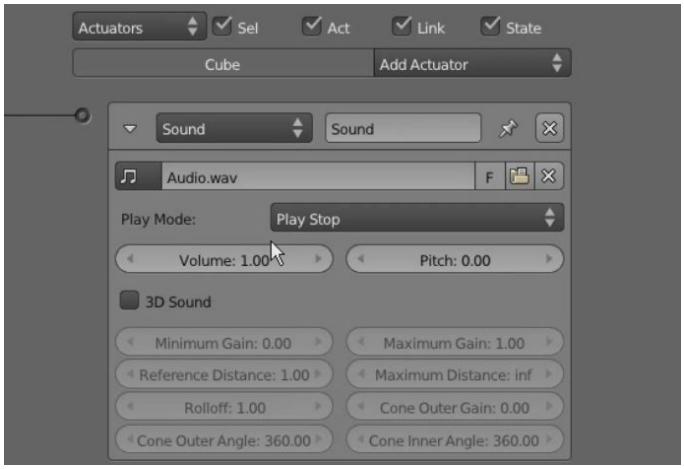
> *NOTA: è possibile collegare, in altre Scene del file .blend, elementi creati successivamente (ad esempio, un oggetto o un personaggio aggiunti in seguito nel primo livello) selezionandoli e scegliendo, nella casella di ricerca veloce di Blender, "Link Object to Scene"; scelta questa operazione, bisognerà scegliere la Scene nella quale inserire una copia-istanza dell'oggetto selezionato dal menù che apparirà a video, che è appunto un elenco delle Scenes disponibili.*

*  *  *

# Le basi del Game Engine, il motore di gioco di Blender 2.6 – 10

Questa è l'ultima puntata del corso sulle BASI del Blender Game Engine 2.6. In questa puntata tratteremo i suoni, l'end game, il conto alla rovescia con un timer e il testo in real time.

In realtà, aggiungere un suono al proprio progetto è semplicissimo: il Game Engine mette infatti a disposizione l'attuatore Sound che, una volta attivato, riproduce un file audio (ad esempio in formato WAV) preso da disco.



*L'attuatore SOUND*

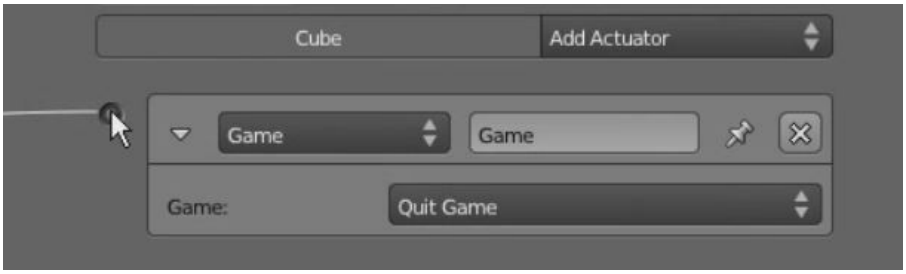
Il sensore o i sensori di attivazione possono essere di vario tipo: può trattarsi della pressione del tasto di sparo, del contatto di un oggetto con un altro, dell'apertura di una porta, eccetera; l'importante è creare un attuatore Sound, caricare il file da disco (con click su Open nella scheda del sensore, per poi scegliere un file da disco con la finestra file browser che apparirà) e regolare, se necessario, il Volume, ma in generale le impostazioni di default dovrebbero andare bene.

Una considerazione riguardo l'opzione "Suono 3D": Blender consente di gestire il "suono 3D" implementando attenuazione, doppler e stereo in relazione alla posizione del protagonista rispetto alla sorgente sonora; tale opzione non sempre è necessaria, infatti per aggiungere una colonna sonora, che tipicamente avrà intensità costante e modalità Loop (anziché Play Stop, quella di default dell'attuatore) non ci dovrà essere suono 3D, ed ecco perché abbiamo la possibilità di scegliere.

*  *  *

La riproduzione del suono, quindi, è molto semplice da gestire... al pari dell'end game: finora infatti abbiamo trattato esempi "senza fine" (al massimo abbiamo visto come cambiare o resettare un livello), ma quando si distribuisce un gioco completo bisogna mettere anche un modo per terminarlo!

L'attuatore che ci interessa è Game, in particolare in modalità "Quit Game".




*L'attuatore GAME in modalità Quit Game*

Questa modalità non mette a disposizione opzioni o parametri perché, quando l'attuatore viene attivato, viene effettuata la chiusura dell'applicazione, nient'altro.

*  *  *

*NOTA: molti mi hanno detto, sul Forum e privatamente, che pur esportando il gioco come eseguibile stand-alone con la vista della telecamera in modalità Textured, non vedono le Texture nel gioco eseguibile, solo superfici bianche. In questo caso, prima di esportare il gioco come eseguibile con "save as runtime", fate File - "External Data" - "Pack into Blend file", quindi salvate nuovamente il progetto (File - Save, sovrascrivendolo) e, a questo punto, esportate come eseguibile standalone.*

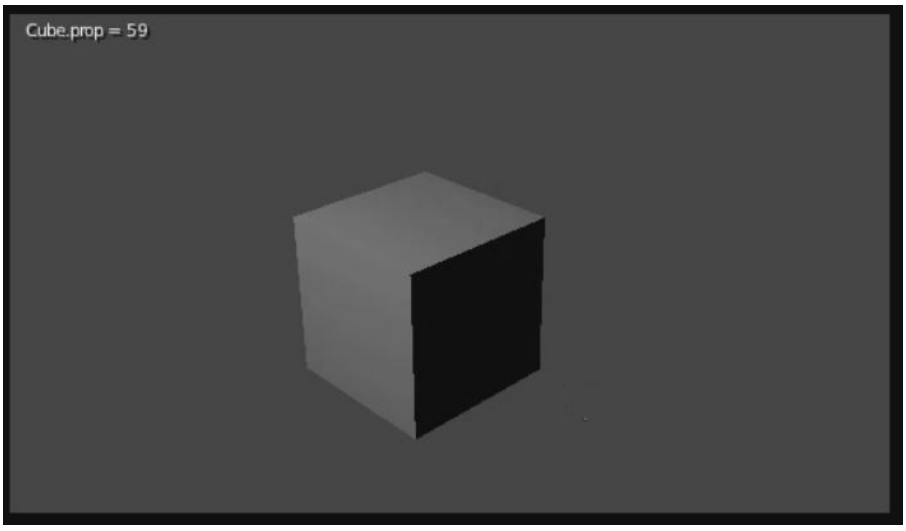
*  *  *

Anche il terzo argomento di questa puntata, ossia il Timer (ad esempio, per un conto alla rovescia), è molto semplice.

Abbiamo visto che una Property di tipo Timer parte da un valore (tipicamente 0.0, se non specificato altrimenti) all'inizio del gioco e va avanti fino a quando non la si termina o la si resetta con un attuatore Property. Non c'è però modo di farla andare al contrario, per il conto alla rovescia.

Quello che dobbiamo fare è aggiungere una Property di tipo intero, ad esempio con valore iniziale 60, quindi aggiungere un sensore Always e collegarlo ad un attuatore Property in modalità Add e valore -1, per togliere cioè 1 secondo ad ogni tick...

... ma c'è un primo problema: avviando il gioco, vedremo passare il valore da 60 a 59, all'inizio, poi stop.




*Prima esecuzione - con errore - del conto alla rovescia*

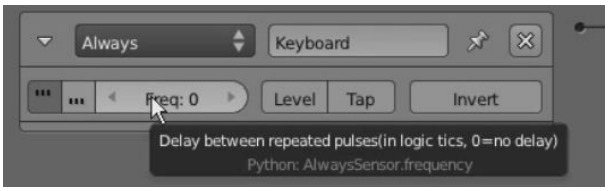
Per risolvere questo problema, dobbiamo selezionare la casella "*Activate TRUE level triggering (pulse mode)*", nella scheda del sensore Always, in modo da inviare SEMPRE, ad ogni "logic tick" (ossia ad ogni aggiornamento dello stato del gioco, da non confondere col *refresh rate* a video), un impulso all'attuatore, decrementando la Property.

*PULSE mode*

Questa considerazione fatta per Always, riguardo il *pulse mode*, vale per tutti i sensori, infatti praticamente tutti hanno i pulsanti TRUE e FALSE per *pulse mode*: di base, quando vengono attivati inviano UN impulso (il primo), poi stop, mentre per far inviare impulsi in continuazione (almeno, finché la condizione di attivazione del sensore è presente) bisogna selezionare la casella TRUE...

... attenzione però, c'è un altro problema: di default, un'applicazione fatta col Game Engine va a 60 logic ticks al secondo, come indicato dalla voce *Standalone Player – Refresh Rate*, all'interno della Properties Window, quindi il nostro contatore andrà a 0 già al primo secondo di esecuzione del gioco e poi andrà in negativo!

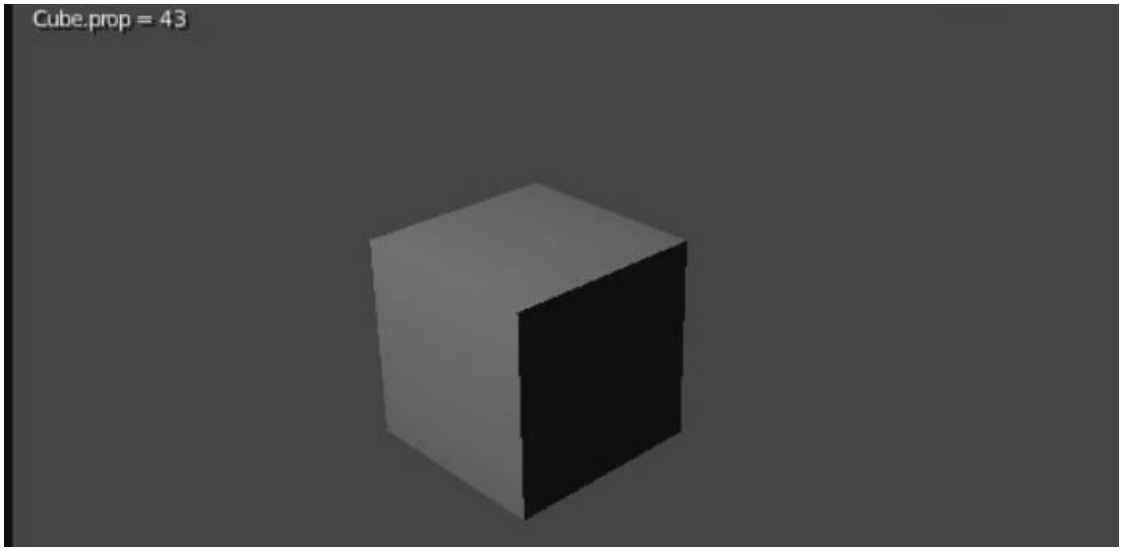
Notiamo che, dopo aver attivato TRUE *pulse mode*, sempre nella scheda del sensore è ora disponibile la casella numerica *Freq,* per *Frequency* (frequenza); il valore di default, 0, indica che verrà inviato un impulso ad ogni logic tick dell'applicazione.



*Frequency*

Considerando che, nel nostro caso, il refresh rate è di 60 logic ticks al secondo, impostiamo appunto 60 nel campo *Freq* del sensore Always, in modo da inviare un impulso ogni secondo. È un concetto che può tornarvi utile anche per altre situazioni, non solo per un conto alla rovescia.

A questo punto possiamo mettere uno o più sensori di controllo della Property intera che fa da Timer, per controllare ad esempio lo scadere del tempo o implementare altri effetti; ad esempio, il giocatore potrebbe effettuare un'azione con associato un attuatore Property con Add sul Timer, in modo da dargli del tempo extra, oppure si potrebbe congelare/resettare il conto alla rovescia, e così via.




*Screenshot dell'esecuzione del countdown*

Da notare che al posto di un Intero possiamo usare un float, con i decimali, per visualizzare anche i decimi e i centesimi di secondo, solo che in quel caso dovremo anche far meglio i conti, in base al tick rate dell'applicazione, per calcolare il valore da sottrarre alla Property!

*  *  *

In questa seconda parte parleremo del cosiddetto "testo in tempo reale" nel Blender Game Engine, ossia della possibilità di mostrare del testo (con un font che potremo personalizzare a piacere definendolo in un file immagine), cambiando al volo il messaggio da mostrare, leggendolo da una Property.

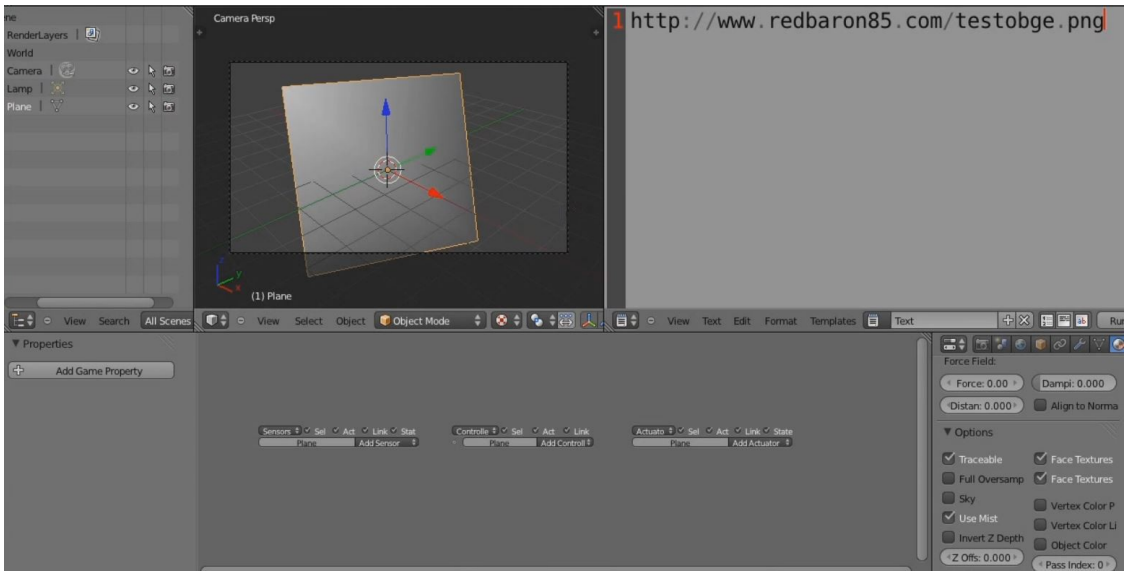
Questa caratteristica è molto utile per realizzare, ad esempio, il contatore dei proiettili rimasti per un'arma in un fps, oppure i valori delle Properties a runtime, mostrandole con una forma grafica più interessante rispetto alla scritta di base di Show Debug Properties che conosciamo.

*Ombreggiatura Textured, ma non in modo GLSL*

Impostiamo il motore Blender Game e la visualizzazione Textured nella 3D View, ma attenzione: se vogliamo utilizzare il testo in real time nella nostra applicazione, la modalità di ombreggiatura deve essere SingleTexture o MultiTexture, non GLSL.

Aggiungiamo un Plane alla scena, posizionandolo più o meno di fronte a noi per vedere meglio il risultato.

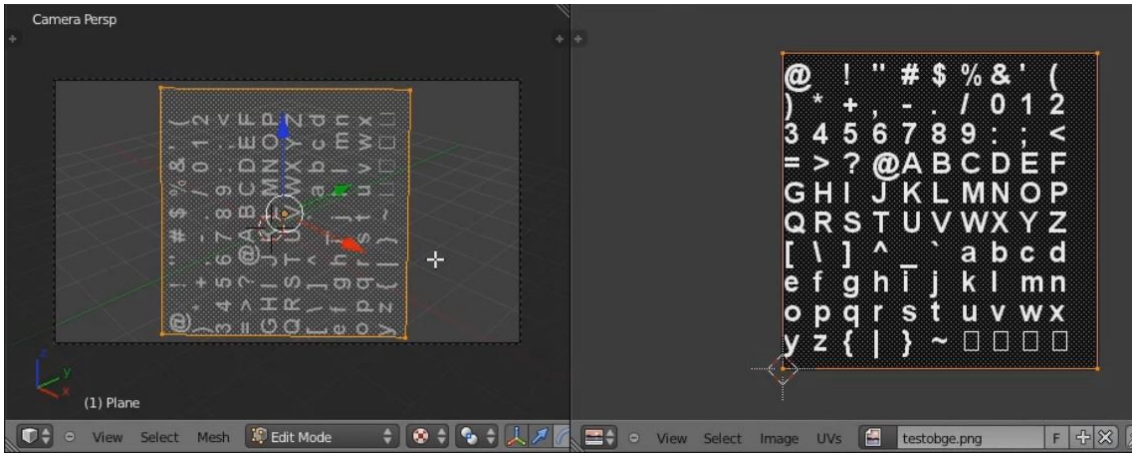
Forniamo un Material al Plane e impostiamo le proprietà in questo modo: trasparenza di tipo Z-Transp attiva, con Alpha a 0, quindi tecnicamente oggetto invisibile; sempre nella scheda Material, nella sezione Game Settings impostiamo la modalità *Alpha Clip* per la trasparenza e selezioniamo la checkbox *Text*; per finire, nella sezione Options, selezioniamo *Face Texture* e *Face Texture Alpha*.




*Il Plane per il test e il suo Material*

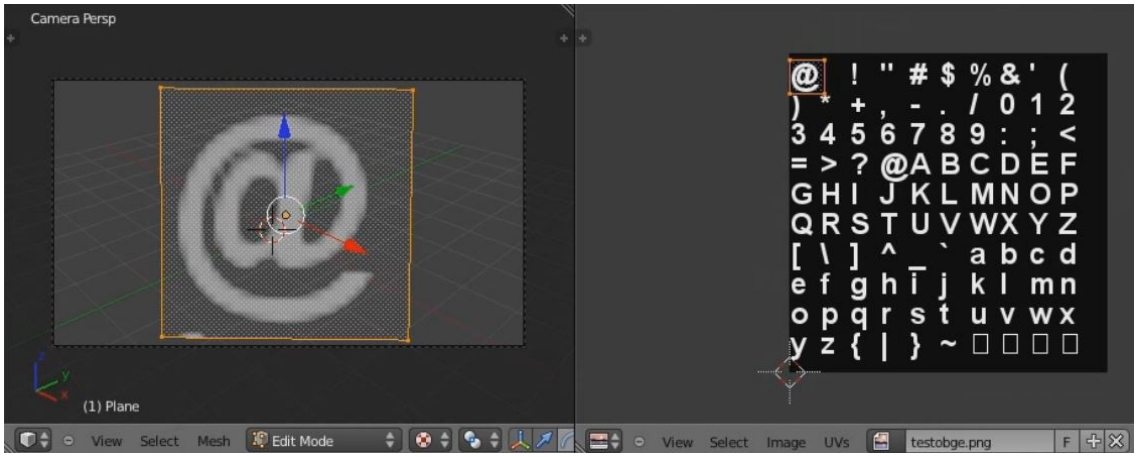
Adesso dobbiamo fare l'unwrap di un'immagine sul Plane; non un'immagine qualsiasi, ma un'immagine che rappresenta dei caratteri (associati a tasti della tastiera) disposti secondo uno schema ben preciso, infatti Blender leggerà i valori delle Properties e andrà a prendere i relativi simboli in tale immagine.

Carichiamo quindi l'immagine dei font in una finestra “UV – Editor delle immagini” (trovate questa immagine all'indirizzo *http://www.redbaron85.com/testobge.png* ).




*L'immagine con i caratteri per il testo*

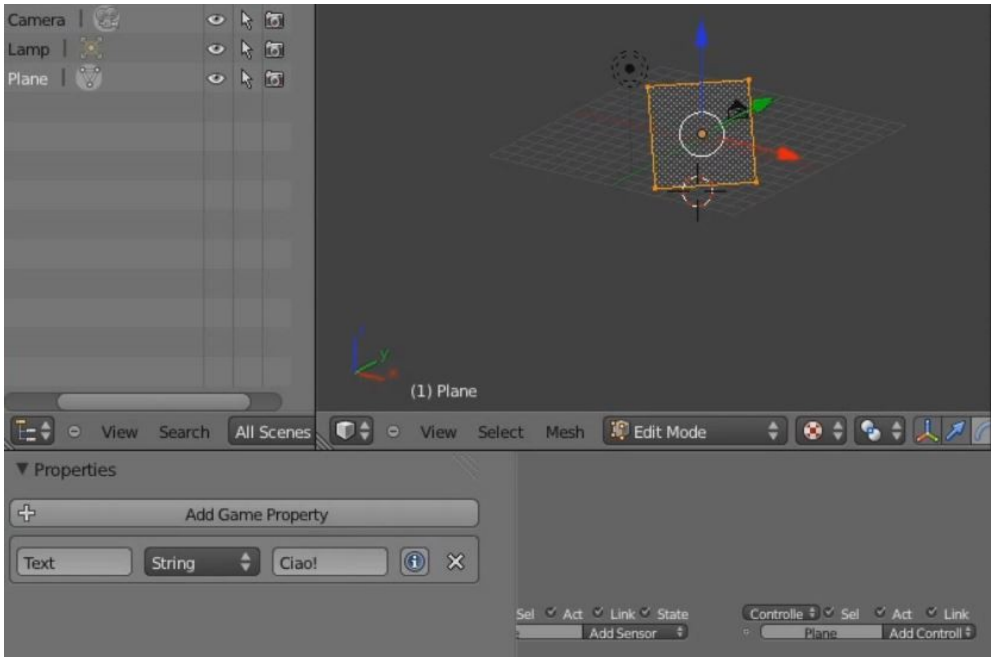
Si tratta di un'immagine png, senza canale Alpha; non dobbiamo fornire una Texture al nostro Material, dobbiamo fare solo fare l'unwrap, ma attenzione: dobbiamo mappare solo la prima lettera, la chiocciola, sul Plane. Blender farà il resto.




*Mappatura del primo carattere (la chiocciola)*

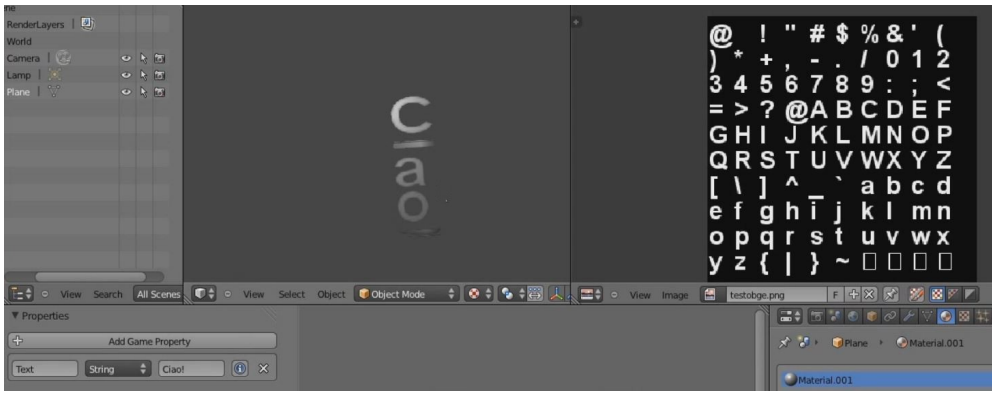
Ultima operazione da fare: aggiungere una Property al Plane, ad esempio un intero (utile per un timer o un contatore) o una stringa (il nome di un personaggio o altri messaggi).

Chiamiamo questa Property “Text” e diamole un valore.




*La Property TEXT*

Avviamo il gioco per vedere il risultato.




*Il risultato in tempo reale, durante l'esecuzione*

Ovviamente possiamo cambiare al volo, con sensori e attuatori, in real time, il valore della Property, e Blender prenderà automaticamente le lettere e i simboli dall'immagine dei fonts e comporrà le scritte. A questo punto possiamo trasformare il Plane, ad esempio inclinandolo, imparentandolo ad un'arma per realizzare il display del numero di proiettili, il tempo per la ricarica, e altro ancora; ovviamente, il file del font (che, come avete visto, è un file immagine) può essere modificato con colori o anche simboli personalizzati (che comunque andranno associati a lettere o numeri secondo lo schema dell'immagine); ci sono davvero tantissime possibilità di personalizzazione e utilizzo.

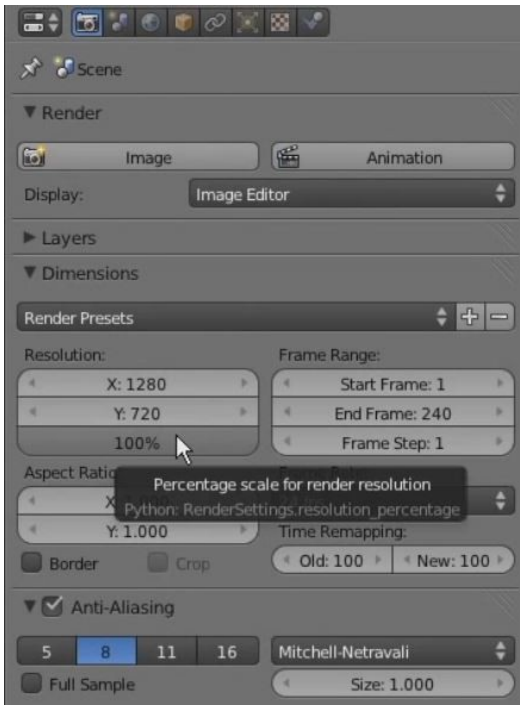
*  *  *

# Il Video Sequence Editor, per montare clip audio, video, ...

Questo tutorial tratta uno strumento poco conosciuto, ma abbastanza potente, di Blender 3D: il Video Sequence Editor. Il tutorial è stato realizzato con la versione 2.62 di Blender 3D, ma il Video Sequence Editor esiste da diverse versioni e i concetti chiave sono sostanzialmente gli stessi.

Il VSE consente, come suggerisce il nome, di realizzare dei filmati a partire da una o più immagini o videoclip, come rendering o altre animazioni, aggiungendo inoltre tracce audio ed effetti di transizione tra le clip. Procediamo con ordine.

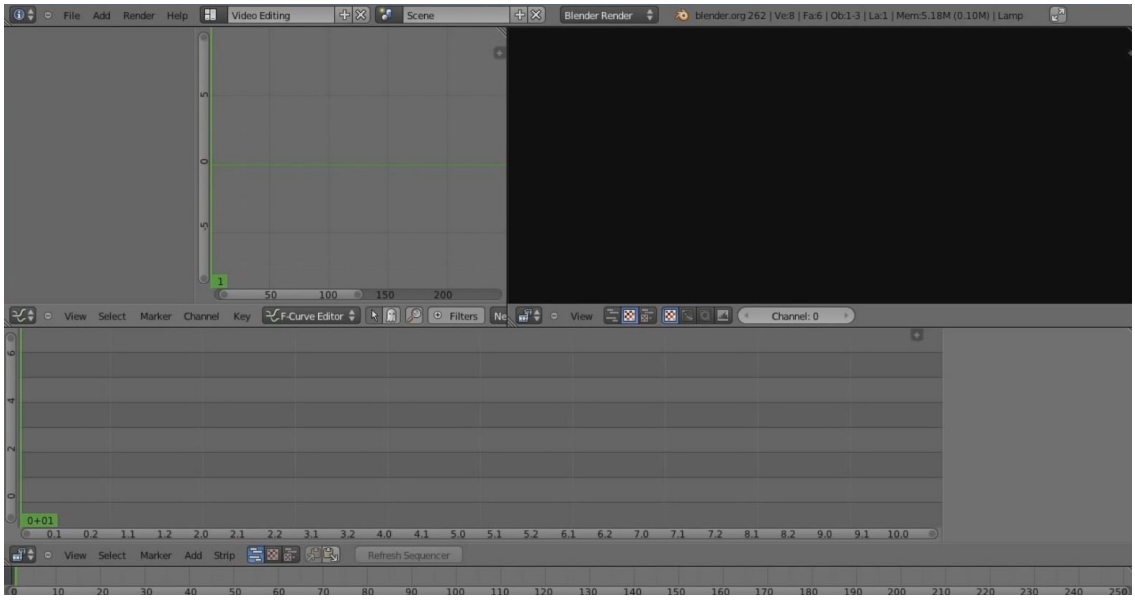
La prima cosa da fare, quando si inizia un progetto, è definire il frame rate del filmato finale (ad esempio, 24 frames al secondo), la risoluzione (nel mio caso, 1280x720, HD) e la durata; queste impostazioni vanno definite nella scheda Render, all'interno della Properties Window, dove dovremo definire anche il formato di output (ad esempio, AVI RAW) e il percorso ove salvare il filmato generato, nella sezione Output. Assicuriamoci inoltre che la casella "Sequencer" sia correttamente selezionata, nella sezione Post-Processing della scheda.



*Impostazioni nella scheda Render*

Per quanto riguarda la durata, in frames, dobbiamo moltiplicare il numero di secondi del nostro filmato per il frame rate, per cui se ad esempio vogliamo un filmato con durata totale 20 secondi a 24 frames al secondo, dovremo impostare end frame 480.

Effettuate queste operazioni preliminari, possiamo iniziare ad utilizzare il VSE; consiglio di utilizzare un Layout creato ad hoc da Blender, scegliendo *Video Editing* dal menù *Choose Screen Layout*, nella finestra Info.

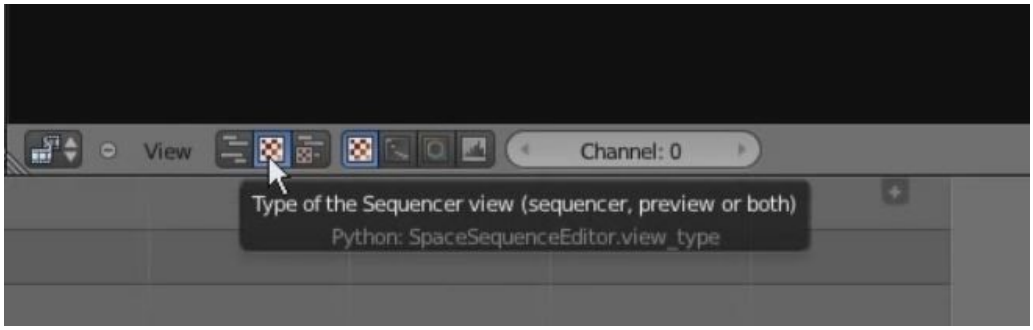



*Utilizzo del Layout "Video Editing"*

Adesso, la finestra di Blender presenterà 4 sottofinestre: in alto a sinistra, un Graph Editor; in alto a destra, un VSE in modalità Preview; al centro, un VSE in modalità Sequencer; per finire, sul fondo, una Timeline.

Come si può intuire già in questa fase, una finestra VSE ha due modalità di visualizzazione: un'anteprima, che mostra quello che verrà visualizzato nel filmato finale, e un Sequencer, che mostra invece in maniera compatta le “strip” (cioè i clip audio, i videoclip e gli effetti) come blocchi da mettere lungo una Timeline per comporre il filmato finale; c'è poi una terza modalità, “Both” (letteralmente, “entrambi”), che suddivide la schermata in due parti mostrando sia la Preview che il Sequencer.

Si passa da una modalità all'altra mediante una delle tre icone presenti nell'header di una finestra VSE, come visibile nell'immagine seguente.




*Scelta della modalità di visualizzazione del VSE*

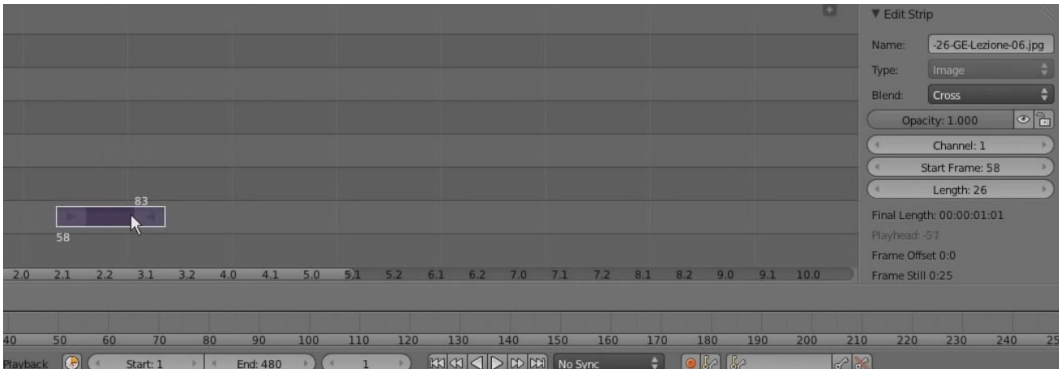
Diamo un'occhiata alla finestra in modalità Sequencer.

Se facciamo click col tasto sinistro su un punto a caso della finestra, notiamo che ci spostiamo in vari "momenti" del filmato, con il fotogramma corrente indicato nel Graph Editor e nella Timeline; tuttavia, nel VSE l'istante viene indicato nella forma *n*+*x*, con *n* che è il tempo in secondi, calcolato dal frame 0, e *x* che è il frame extra a partire dall'*n*-esimo secondo, per cui se il filmato è a 24 frames al secondo, il 100esimo frame verrà indicato come 4+04, cioè 4 secondi più 4 frames, e ovviamente i sottoframes saranno al massimo 23, prima di passare al secondo successivo e riprendere la numerazione dei frame extra da 0.

*Visualizzazione del tempo nel formato n+x*

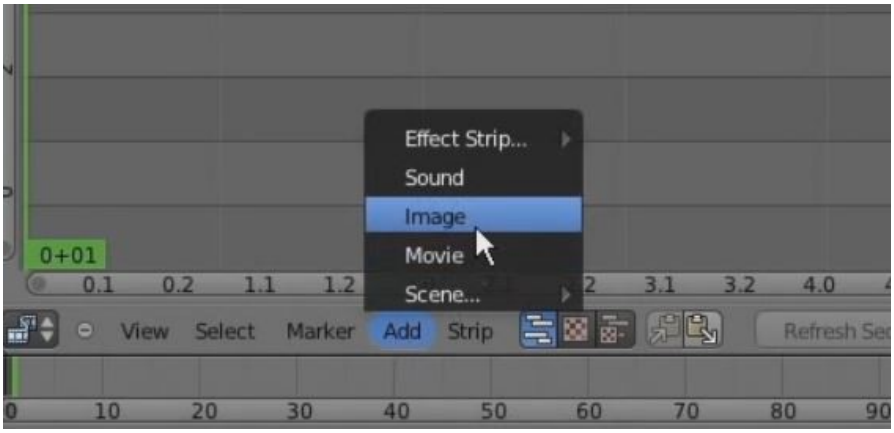
Saltando da un frame all'altro, l'anteprima della situazione corrente verrà visualizzata in una finestra VSE Preview. È possibile spostarsi nella Timeline anche con le frecce o avviare l'anteprima con ALT A, come avviene in una 3D View.

Altra cosa da dire su tale finestra è che dispone di una Transform Window, visualizzabile (come quella della 3D View) con lo shortcut N.



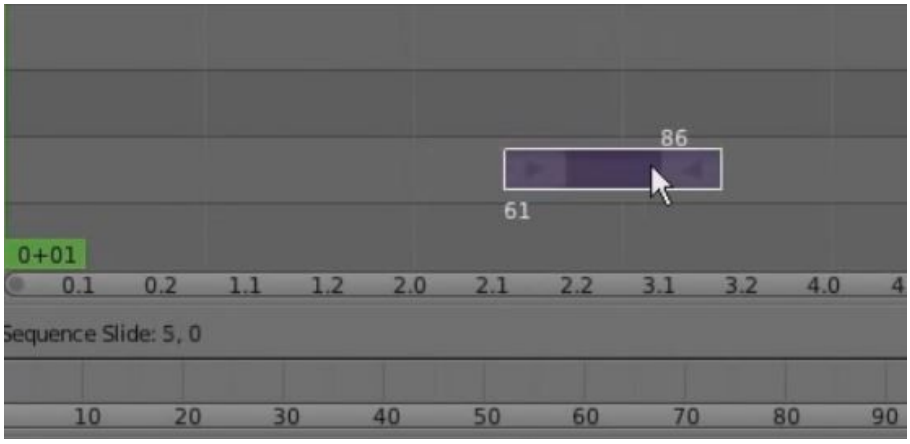

*La Transform Window del VSE*

Come avrete intuito, immagini, clip video e audio vanno inseriti nella varie strisce orizzontali della finestra, come dei "canali-informazione". Per aggiungere una clip, scegliamo *Add Image* dall'header e aggiungiamo un'immagine presa da un file su disco.



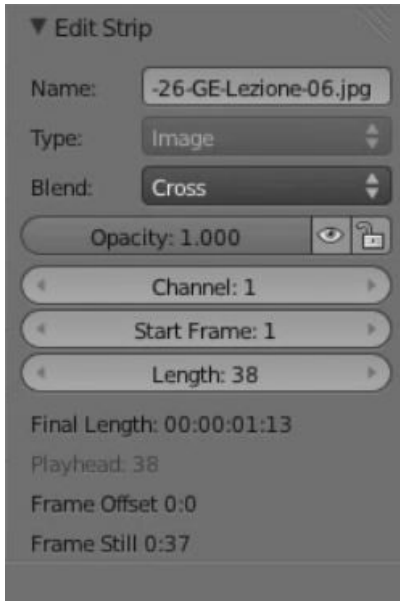

*Aggiungere una clip con Add Image*

Per spostare la clip in un dato frame di un canale-informazione, possiamo procedere in vari modi.
Per prima cosa, selezioniamola con un click del tasto destro (da notare che è possibile effettuare selezioni multiple di clip con SHIFT e tasto destro del mouse), quindi spostiamola con G (per Grab, trascina) e muovendo il mouse, portandola ad esempio sul primo canale-informazione e spostandola in orizzontale per farla apparire in momenti diversi. A sinistra e a destra verranno visualizzati dei numeri, che indicano il frame di inizio e fine visualizzazione.




*Spostare una Clip*

Possiamo quindi selezionare una delle due estremità, con click del tasto destro ad esempio sulla freccia destra, che ora verrà mostrata in bianco e con un numero che indica il frame finale, quindi possiamo trascinare solo questa estremità, con l'effetto di accorciare o prolungare la visualizzazione dell'immagine; idem con la freccia all'estremità sinistra.




*Impostare i parametri della Clip nella Transform Window del VSE*

Possiamo stabilire il frame iniziale e la durata della clip selezionata anche nella Transform Window del VSE, con le voci Start Frame e Length, rispettivamente frame di inizio visualizzazione e durata (in frames).

Le clip, di tutti i tipi, possono essere anche spezzate in più parti ricorrendo al taglierino (knife): si seleziona la clip con un click del tasto destro, quindi si preme K (per knife) e, tenendolo premuto, si preme e si tiene premuto il tasto sinistro del mouse, "tagliando" appunto con il cursore la clip al frame desiderato, rilasciando poi sia K che il mouse a fine operazione.

*Utilizzo del taglierino (Knife) sulla Clip*

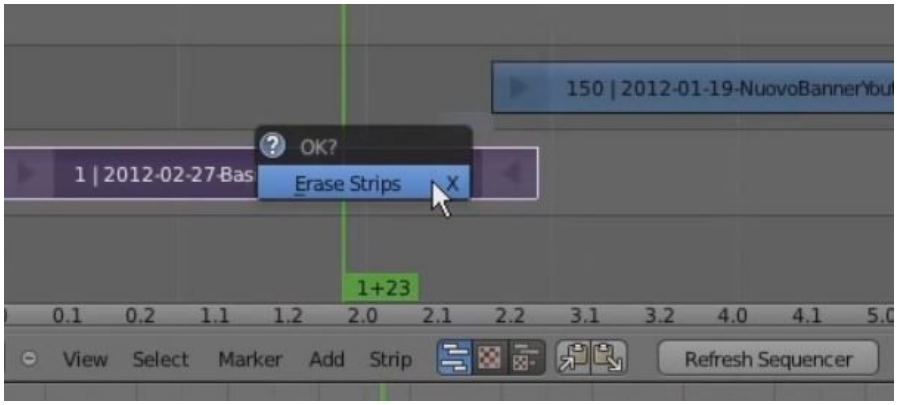
*NOTA: per zoomare in questa finestra, in modo da vedere più o meno frames del filmato complessivo, premere insieme CTRL e il tasto centrale o rotellina del mouse e, tenendoli entrambi premuti, muovere il mouse in orizzontale per allargare o restringere la Timeline, oppure in verticale per allargare o restringere la visualizzazione dei canali-informazione presenti.*

Così come abbiamo aggiunto una clip immagine, possiamo aggiungere e spostare, o in generale manipolare, una clip video o un clip audio.

Aggiungiamo ad esempio una clip video nella scena. Blender supporta un'ampia varietà di formati, sia per quanto riguarda le immagini che per i filmati o i file audio; solo una raccomandazione: è consigliabile utilizzare immagini o filmati con la stessa risoluzione del filmato da produrre in output (o almeno rapporto larghezza/altezza uguale) e, nel caso dei filmati, lo stesso frame rate.

*Aggiungere una Clip video (Movie)*

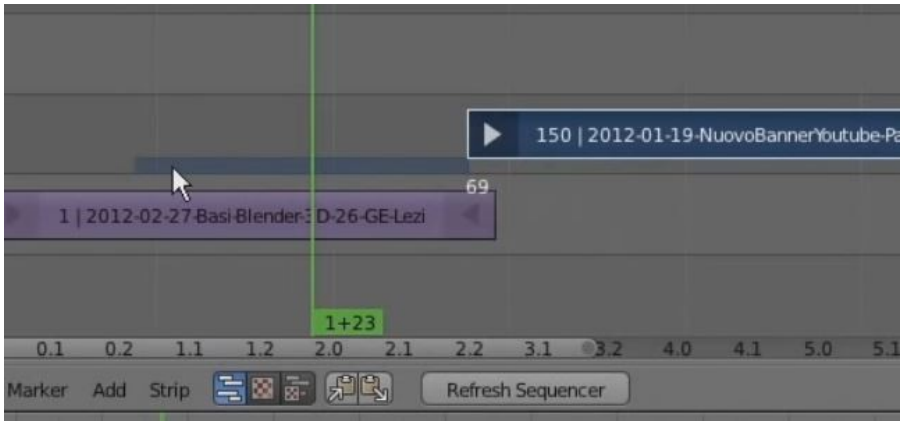
Quando aggiungiamo una clip video, Blender aggiunge anche un canale audio, anche se in realtà la clip era sprovvista di sonoro; in ogni caso, possiamo selezionare tale traccia e cancellarla con X o CANC (operazione valida per qualsiasi tipo di strip).



*Cancellare la traccia audio della Clip video importata*

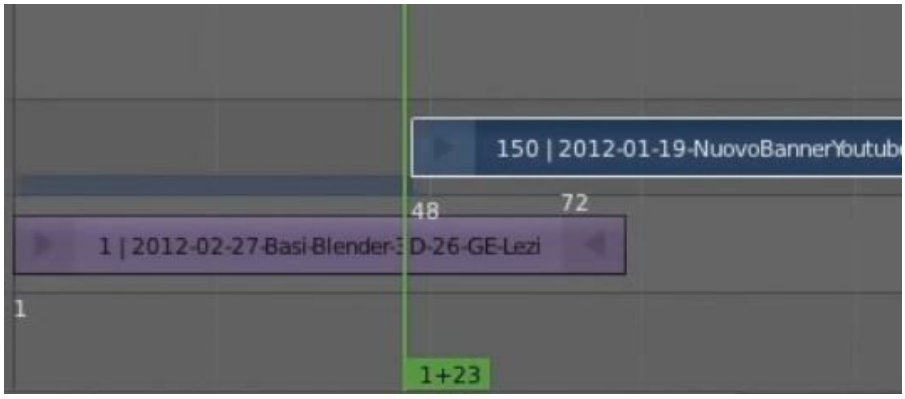
Da notare che accorciare o restringere una strip di immagine significa mostrarla per un tempo più breve o più lungo, ma accorciare o restringere una clip video ha effetto diverso: restringendola, si taglierà un pezzo del filmato, mentre allungandola alla fine si prolungherà solo l'ultimo frame: in sostanza, non si cambia il *frame rate* della clip video.

Mettiamo la clip immagine sul canale 1, dal frame 1 al 72esimo, cosa che corrisponde a 3 secondi di visualizzazione, quindi mettiamo la clip filmato sul canale 2, a partire dal 48esimo frame.




*La Clip immagine e quella filmato "posizionate"*

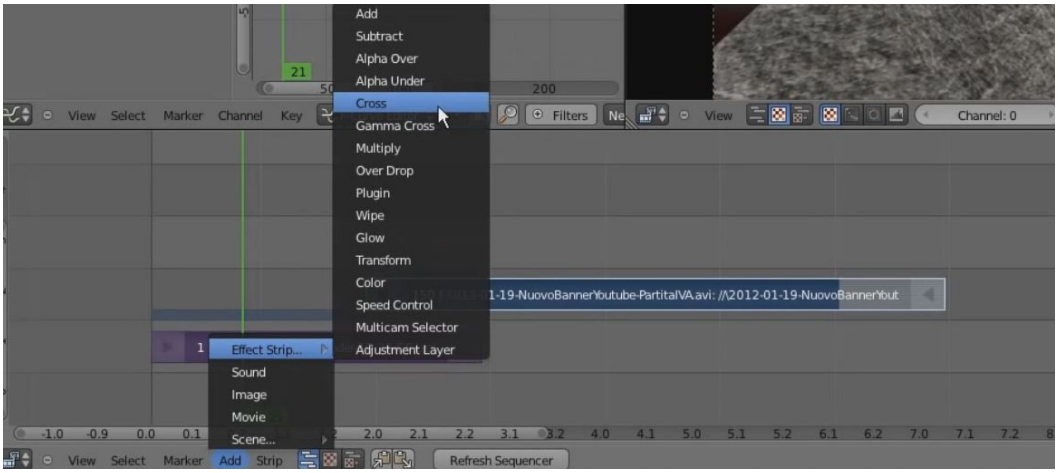
Adesso accorcio di qualche frame il filmato, selezionando l'estremità sinistra e spostandola: Blender ci fa notare che stiamo saltando un pezzo di filmato, mostrando una mini-striscia che si allunga quando trasciniamo l'estremità. Il modo corretto per spostare un filmato è selezionarne il corpo e, quindi, premere G per trascinare tutto il blocco.
Per spostare il blocco con il pezzo tagliato, dobbiamo selezionare ancora una volta il blocco e trascinare, spostando sia la strip visualizzata che la mini-striscia iniziale che indica il pezzo escluso.
Posizioniamo l'inizio della strip da visualizzare al frame 48.




*Posizionare l'inizio della strip al frame 48*

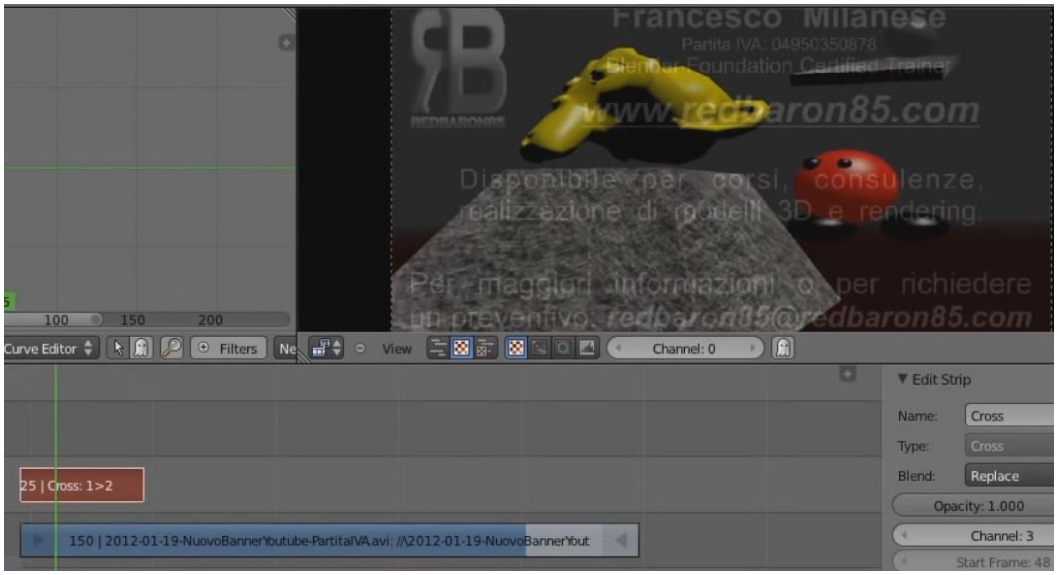
Stiamo creando una zona di sovrapposizione di un secondo con l'immagine caricata precedentemente, ma se lanciamo ora l'anteprima nella finestra VSE Preview, noteremo che il passaggio è alquanto brusco.

Per aggiungere la classica dissolvenza tra i filmati, selezioniamo le due strip con una selezione multipla (prima click col tasto destro sulla strip immagine, al canale 1, poi sulla strip del filmato al canale superiore), quindi scegliamo *Add – Effect Strip – Cross*.



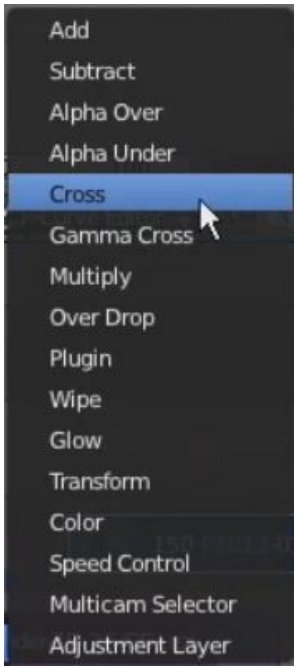

*Creazione dell'effetto dissolvenza (CROSS)*

Abbiamo aggiunto una dissolvenza tra le strip, come possiamo vedere riportando a 1 la Timeline e premendo ALT A per visualizzare l'anteprima nella finestra VSE Preview.




*Screenshot della dissolvenza tra le clip*

Da notare che, spostando adesso una delle estremità di una delle clip, la dimensione della strip della dissolvenza varierà automaticamente.



*Voci del menù Effect Strip*

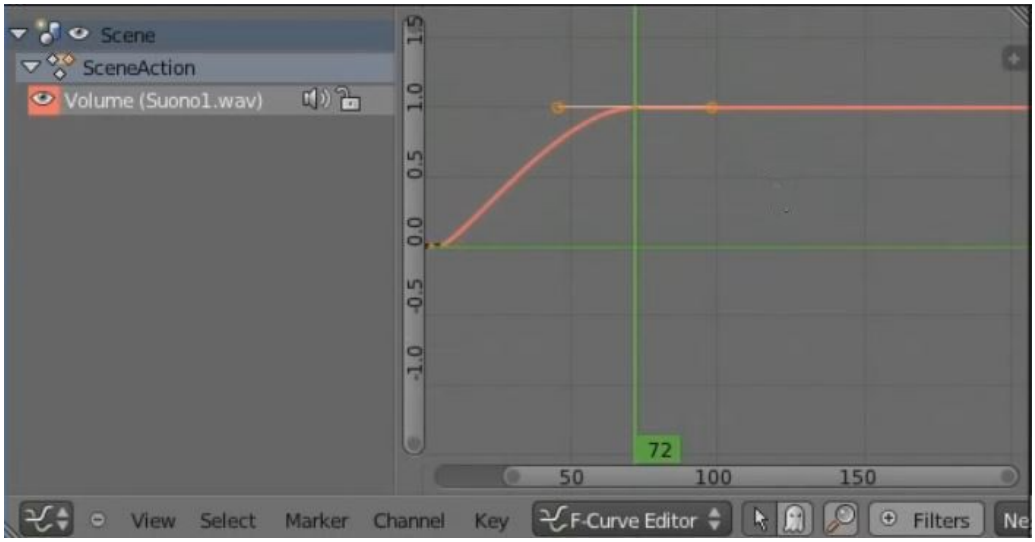
In questo tutorial non tratterò i vari effetti presenti nel menù Effect Strip, comunque i significati sono abbastanza intuitivi; ad esempio, con Wipe possiamo scegliere poi, nella Transform Window della finestra, tra le dissolvenze radiali (Clock), a tendina (Single), eccetera; se invece abbiamo un'immagine provvista di canale della trasparenza Alpha (ad esempio una scritta o un logo) possiamo metterla in un canale posto sopra quello del filmato e scegliere l'effetto Alpha Over.

Ricordiamoci che tutti i parametri, in Blender, sono animabili mediante i Keyframes, per cui ad esempio possiamo realizzare una dissolvenza in entrata o in uscita per una clip audio animando il parametro Volume, nella Transform Window.

Per inserire un Keyframe, posizioniamo il cursore del mouse sul parametro e premiamo I (per Insert Keyframe), oppure click col tasto destro del mouse e Insert Keyframe (da questo menù possiamo poi togliere un Keyframe, se necessario, con Remove); ad esempio possiamo registrare il valore 0 al frame 1 per Volume e 1.0 (valore di default; il valore massimo è 100.000) al frame 72, implementando così una dissolvenza in entrata per la traccia audio in 72 frames.

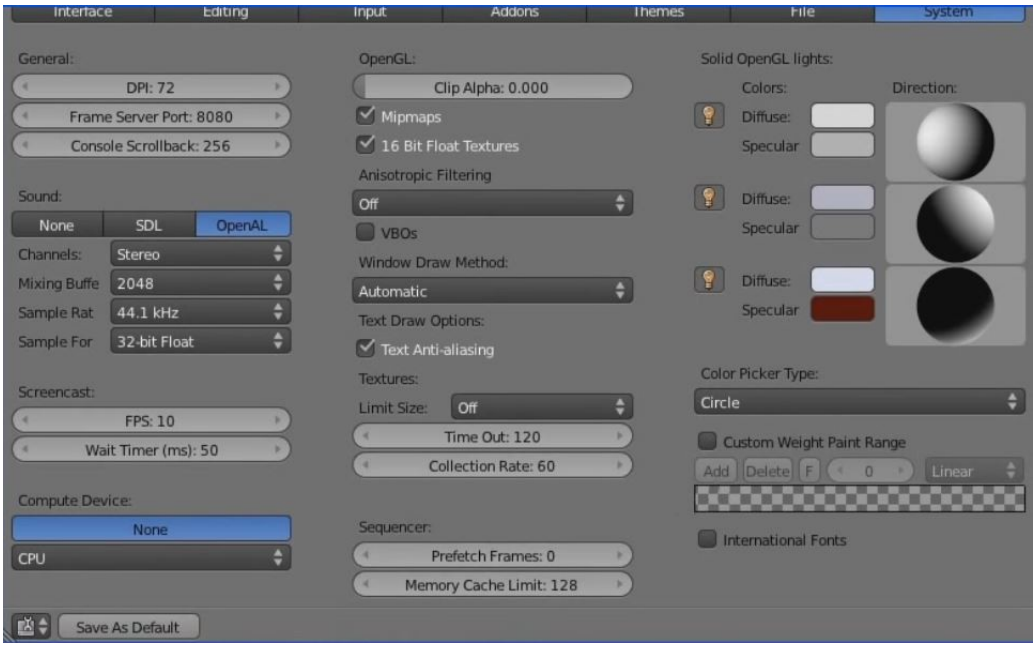
*Inserimento di Keyframes nella Transform Window del VSE*

La curva IPO creata per ogni canale nel VSE apparirà nel Graph Editor, per cui lì potremo personalizzare ulteriormente gli effetti variando i valori o la forma dei punti di controllo o delle maniglie, inserendo e spostando keyframes, utilizzando i modificatori delle IPO Curves, eccetera.




*Curva IPO realizzata*

In genere, è possibile sentire l'audio delle strip inserite già in fase di anteprima, con ALT A, ma in caso di problemi possiamo provare ad aumentare la quantità di memoria allocata per il VSE variando, nel pannello System all'interno della finestra User Preferences, il parametro *Sequencer Memory Cache Limit*.




*Sequencer Memory Cache Limit nella scheda User Preferences*

Le strip sostituiscono interamente la visualizzazione della scena di Blender, ma ovviamente è possibile inserire il rendering di determinati frames della scena nel clip finale, magari aggiungendo appunto una colonna sonora con un file mp3 o wav in una traccia, un logo in sovrimpressione o un'introduzione.

*  *  *

# Riprodurre filmati, file video e animazioni nel Game Engine

Questo è un brevissimo tutorial nato da una richiesta di un utente del Forum, *"French93",* che chiedeva come poter eseguire un filmato AVI, MP4 o di altro tipo su una mesh (ad esempio un Plane rappresentante uno schermo) a runtime nel Blender Game Engine, il motore di gioco di Blender 3D 2.6.

Per ottenere questo risultato bisogna utilizzare uno script Python, *Video.py*, distribuito con licenza Creative Commons, come specificato nell'header dello script.

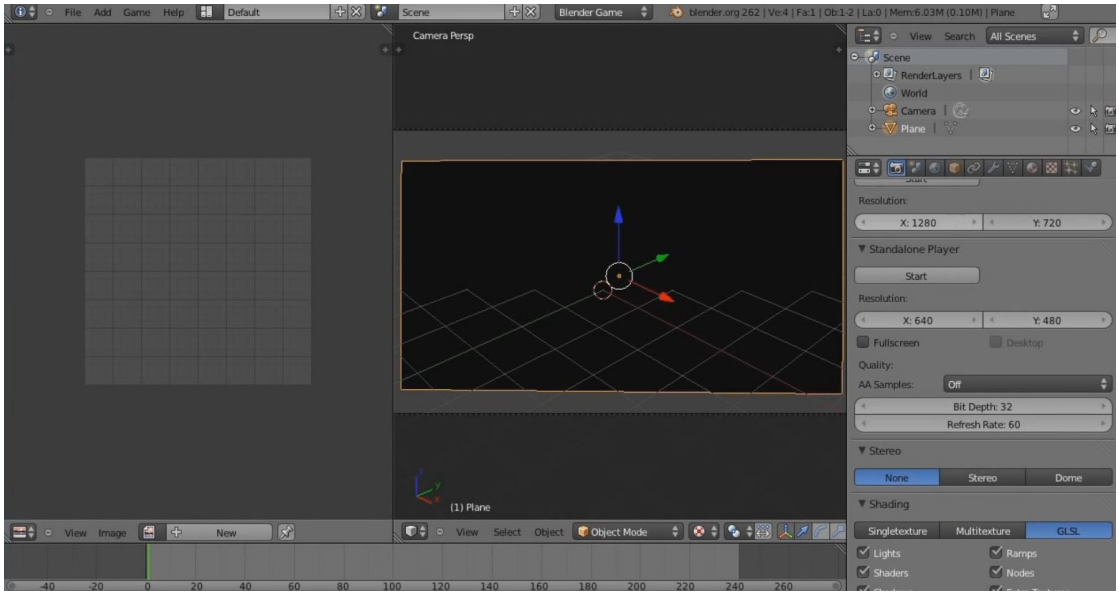
Questo script fa uso del modulo Video delle API Blender Python, è composto da pochissime righe ma il risultato è notevole; purtroppo, nell'header non è presente il nome del programmatore, quindi non so chi ringraziare per il risultato! Trovate il file .zip con lo script da utilizzare all'indirizzo: *www.redbaron85.com/bge26movie.zip* .

Vediamo quindi quali sono i passi da seguire per implementare l'effetto.

Passiamo all'engine Blender Game e alla modalità di ombreggiatura GLSL; nella 3D View, cambiamo la Viewport Shading in Textured (immagine nella pagina seguente).

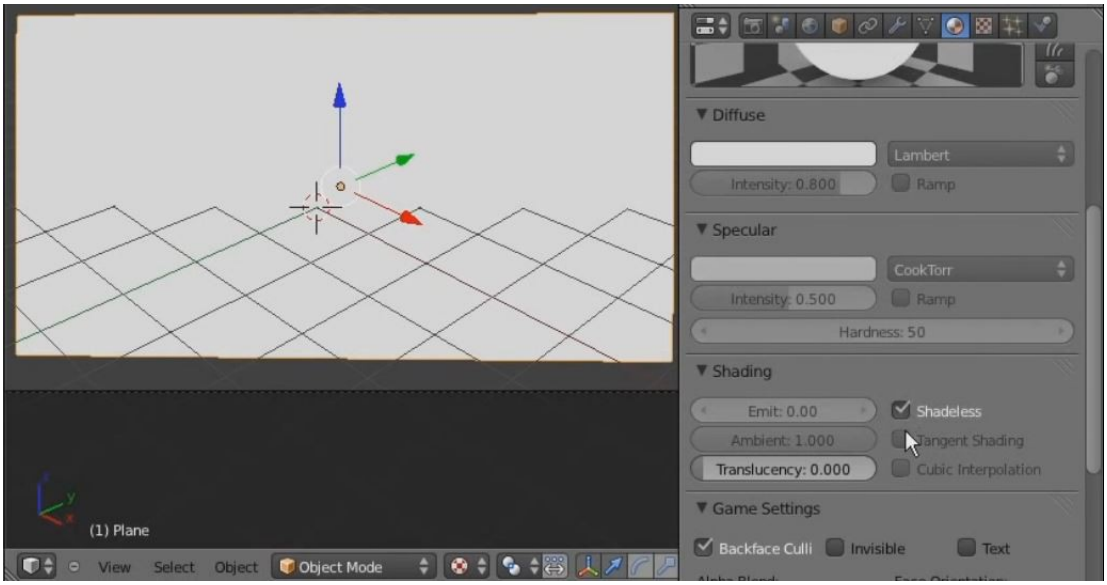
Per prima cosa ci serve una superficie sulla quale proiettare il nostro filmato; per questo esempio, utilizziamo un semplice Plane.

È importante fornire al Plane un Material con un nome significativo (perché in seguito dovremo usarlo per lo script), facendo attenzione a minuscole e maiuscole (Blender fa distinzione) ed unwrappare la superficie sulla quale proiettare il filmato.




*Game Engine, Textured, GLSL*

Aggiungiamo quindi un Material al Plane e chiamiamolo ad esempio MA, tutto maiuscolo, e rendiamo il Material Shadeless.

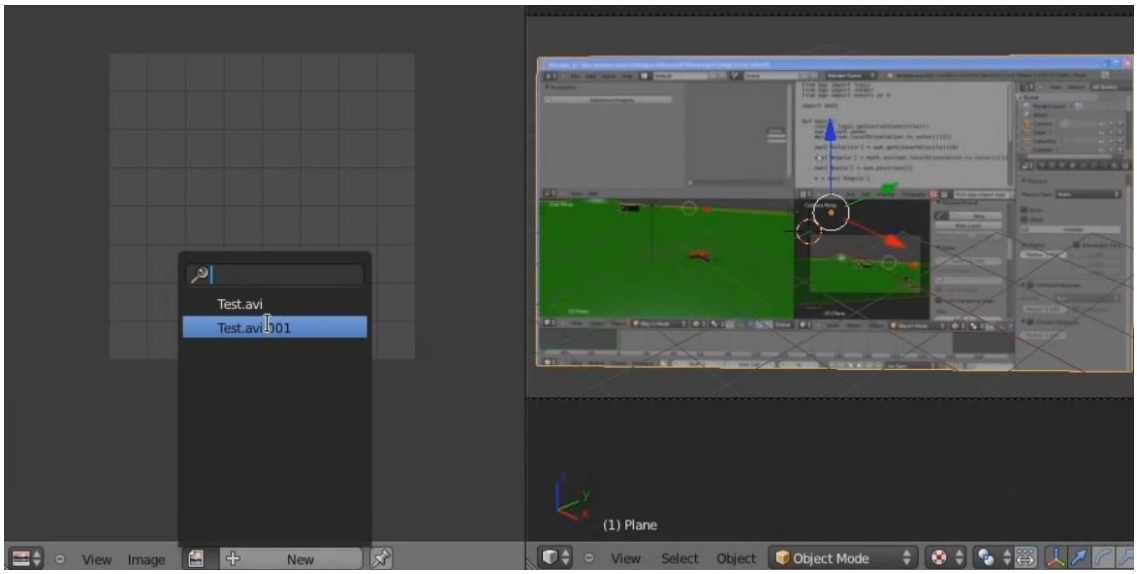



*Il Material Shadeless per il Plane*

Aggiungiamo una Texture di tipo Image or Movie, caricando qui con Open il filmato da proiettare sul Plane.

A questo punto apriamo anche una finestra UV – Editor delle immagini ed effettuiamo l'unwrap del Plane sul filmato, caricato anche nell'Editor delle immagini.

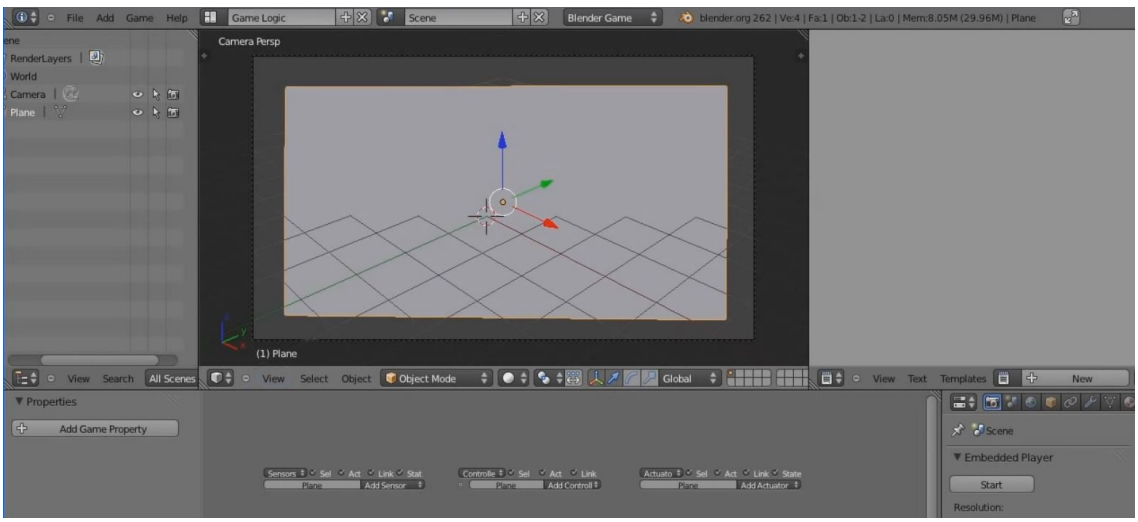
Possiamo vedere, nella 3D View, il risultato della mappatura; eventualmente possiamo ruotare o scalare la scucitura per mappare correttamente il filmato sulla mesh.




*Mappatura UV della texture filmato sul Plane*

Nella scheda della Texture, impostiamo modalità di mappatura UV, utilizzando il Layer creato automaticamente da Blender in fase di unwrap.

Associati Material e Texture filmato al Plane unwrappato, passiamo allo Screen Layout Game Logic, visto che ci servono una finestra Text Editor per caricare lo script e il Logic Editor, oltre ovviamente alla 3D View con Viewport Shading Textured.

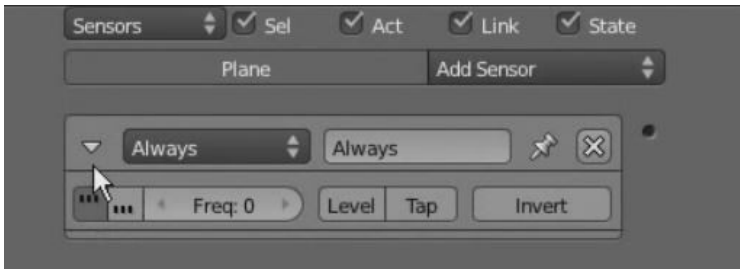



*Screen Layout Game Logic*

Nel Text Editor, carichiamo il file dello script rinominandolo in *Movie.py* (con la M iniziale maiuscola) nella casella testuale col nome dello script nell'header della finestra.

Selezioniamo il Plane, in modo da configurarne i Logic Bricks.

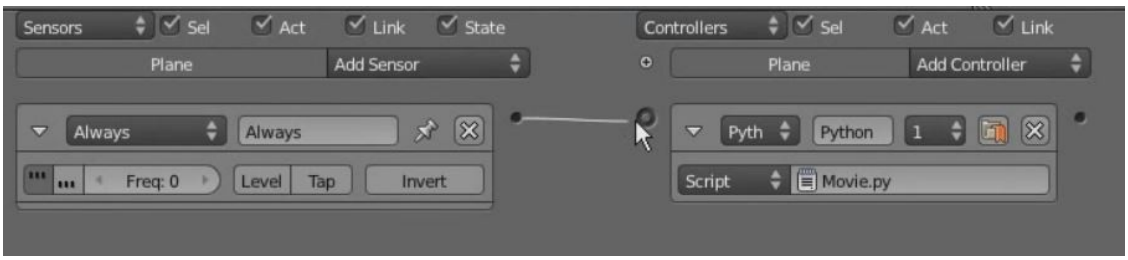
Nel Logic Editor, aggiungiamo un sensore Always e selezioniamo il pulsante *Activate TRUE level triggering (pulse mode)* del sensore.




*ALWAYS con True Level Triggering (PULSE mode)*

Aggiungiamo un controller Python e carichiamo qui lo script *Movie.py* facendo click sul campo testuale del controller e scegliendo, dal menù che apparirà, lo script, che abbiamo caricato e rinominato poco fa nel Text Editor.

Colleghiamo Sensore e Controller.



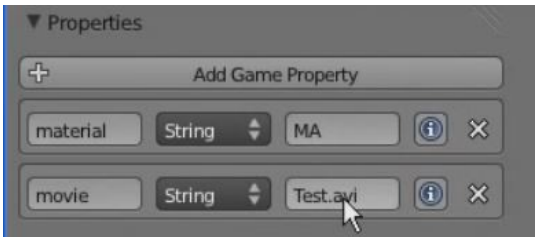

*Collegamento del Controller PYTHON*

Lo script può essere utilizzato anche da più oggetti (per visualizzare filmati diversi) all'interno dello stesso scenario; l'importante è impostare correttamente almeno due Property di tipo String per gli oggetti che dovranno mostrare i vari filmati: lo script non va modificato, per cui il nome del filmato da utilizzare va indicato con delle Property dell'oggetto utilizzato per la visualizzazione.

Vediamo quindi quali sono queste Properties.

Aggiungiamo una Property di tipo String al Plane, chiamando tale proprietà "material" (tutto minuscolo) e dandole come valore il nome del Material dell'oggetto, nel nostro caso MA.

Aggiungiamo una seconda Property di tipo String al Plane, chiamando tale proprietà "movie" (tutto minuscolo) e dandole come valore il nome del filmato da riprodurre, nel mio caso *Test.avi*.




*Le due Property STRING per l'oggetto*

Attenzione, una nota importante: il file da riprodurre deve trovarsi nello stesso percorso del file *blend* (se si trova in una sottocartella dovete specificare il percorso relativo come nome della Property "movie", altrimenti lo script non caricherà il filmato).

Se vogliamo, ma non è obbligatorio, possiamo aggiungere anche una terza Property, questa volta di tipo boolean, chiamata "loop" (tutto minuscolo) che, se a True, eseguirà continuamente il filmato, in maniera ciclica.

Questa è la tecnica base, ma va da sé che è molto flessibile e si presta a vari utilizzi; ad esempio, per far partire il filmato in un certo momento (ad esempio se il protagonista preme un pulsante per far partire la riproduzione di un video su un televisore, un ologramma (in questo caso rendendo semitrasparente il plane con Transparency in Material) e così via), potete inserire il Plane al volo nella scena con un attuatore Add Object e farlo sparire a fine riproduzione con *Delay*, la durata del filmato ed End Object.

Se, invece, volete eseguire un filmato a schermo intero in seguito ad un evento, ad esempio se il personaggio ha finito un livello o raccolto un oggetto importante, potete mettere questo Plane su una seconda Scene in modo da fargli occupare tutta l'inquadratura e passare alla visualizzazione di questa seconda Scene come descritto, ad esempio, nel sesto tutorial sulle basi del BGE 2.6.

Come indicato nel corso sul BGE, inoltre, è possibile controllare i valori delle Properties a runtime con i sensori Property o variarne i valori con gli attuatori, anche in questo caso di tipo Property.

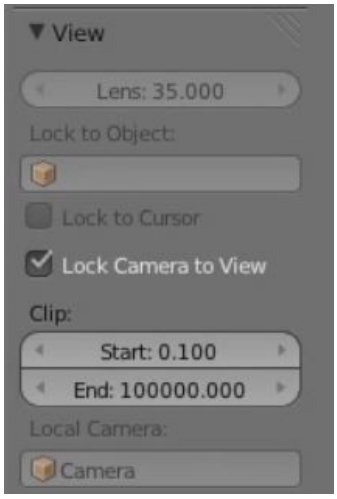
*  *  *

# Lock Camera to View e Align Camera to View

Questo è un brevissimo tutorial sull'utilizzo della telecamera in Blender 3D, in particolare sulla funzione “Lock Camera to View”; ne approfitterò per ricordare anche la funzione “Align Camera to View”.

A tutti è capitato, prima o poi, di osservare la scena dalla 3D View con vista telecamera e premere il tasto centrale del mouse con l'intenzione di spostare il punto di vista della telecamera nella scena, per sistemare l'inquadratura, col risultato di tornare in vista User libera, lasciando ferma la telecamera, oppure di ruotare la rotellina per avanzare o indietreggiare con la telecamera, effettuando invece uno zoom all'interno dell'inquadratura.

Per poter spostare la telecamera con le modalità appena indicate, è necessario andare in visualizzazione Camera in una 3D View, aprire la Transform Window (con N) e selezionare, nella sezione View del pannello, la voce “Lock Camera to View”.



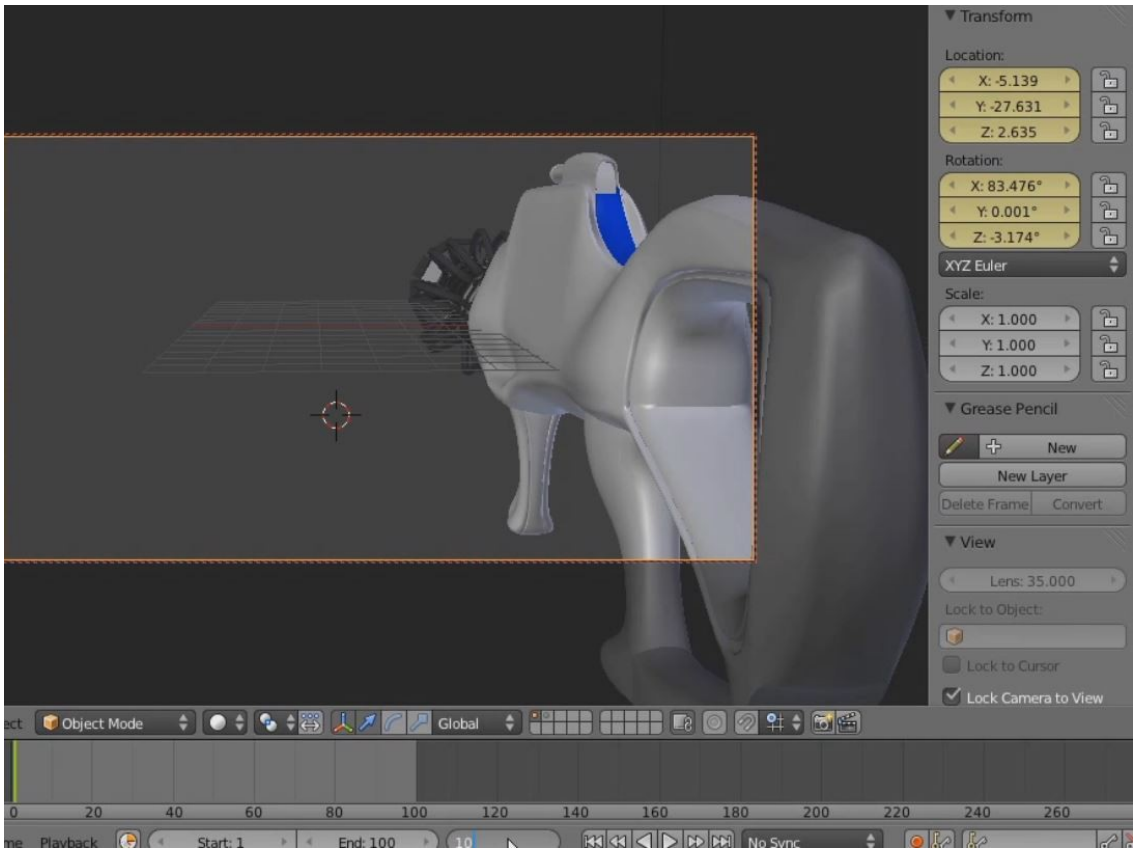
*Lock Camera to View*

Apparirà un bordino tratteggiato colorato, nella 3D View, intorno alla maschera della telecamera, ad indicare appunto che ora la telecamera e la 3D View corrente sono agganciate (locked).

Adesso, con un click del tasto centrale o rotellina del mouse e muovendo il mouse, ruoteremo la vista spostando e ruotando, nel mentre, la telecamera all'interno della scena 3D; lo zoom equivarrà a spostare avanti o indietro, ossia sul suo asse Z locale, la telecamera, mentre con SHIFT e click del tasto centrale potremo effettuare il panning (che prima era il panning della vista ma mantenendo ferma l'inquadratura, mentre ora è proprio una traslazione della telecamera nella scena).

Neanche a dirlo, questa modalità può tornare molto utile non solo per trovare la giusta inquadratura per i rendering "statici", ma anche per le animazioni.

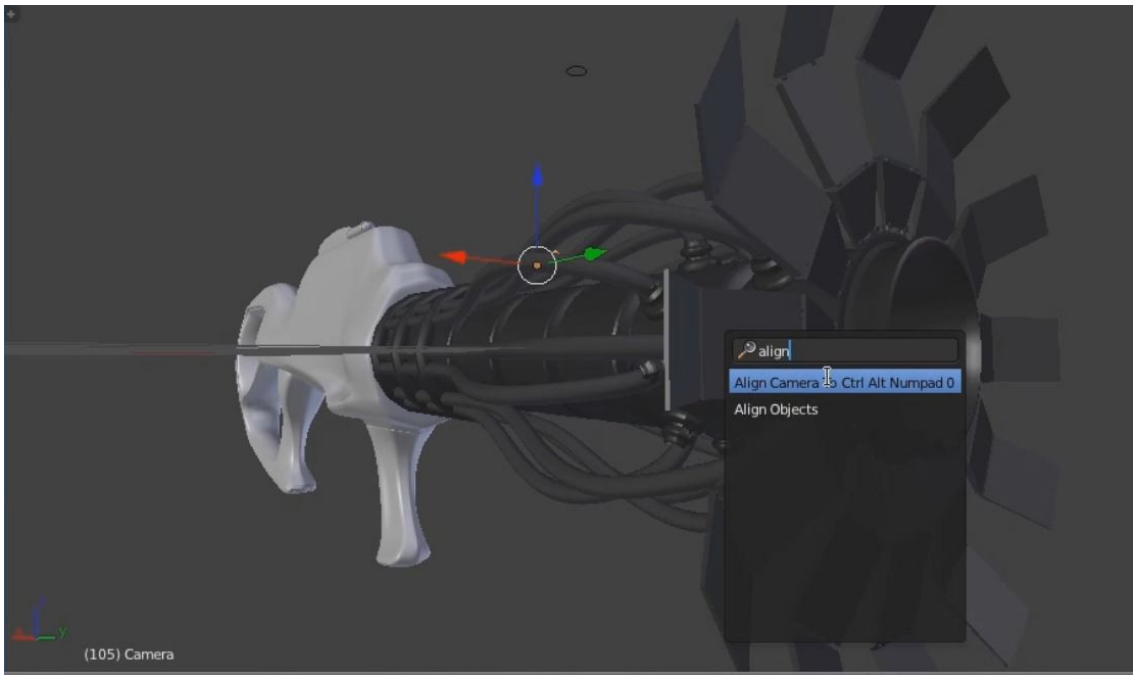
Inseriamo ad esempio un keyframe per LocRot della telecamera al frame 1 con la configurazione corrente, quindi spostiamoci ad esempio al frame 100, spostiamo e ruotiamo a piacere la telecamera nella scena 3D dalla 3D View (approfittando di "Lock Camera to View") ed inseriamo un nuovo keyframe con *I - LocRot*, quindi torniamo al primo frame e osserviamo l'anteprima dell'animazione così realizzata con ALT A.



*Inserimento dei Keyframe*

Mentre ci sono, anche se ne ho già parlato in un altro tutorial pubblicato precedentemente, vi ricordo un'altra opzione, "Align Camera to View", che consente di posizionare la telecamera in modo da riprodurre una visuale Persp libera.

Questa funzione torna utile quando, girando liberamente nella scena 3D, si trova un'inquadratura perfetta e si desidera posizionare lì la telecamera, in modo da effettuare il rendering di quella vista; ebbene, in questo caso è necessario aprire la casella di ricerca di Blender (con la barra spazio), scrivere “Align Camera” e scegliere “Align Camera to View”, oppure utilizzare lo shortcut *CTRL ALT 0* (del tastierino numerico) mentre il cursore del mouse si trova nella vista 3D da utilizzare.



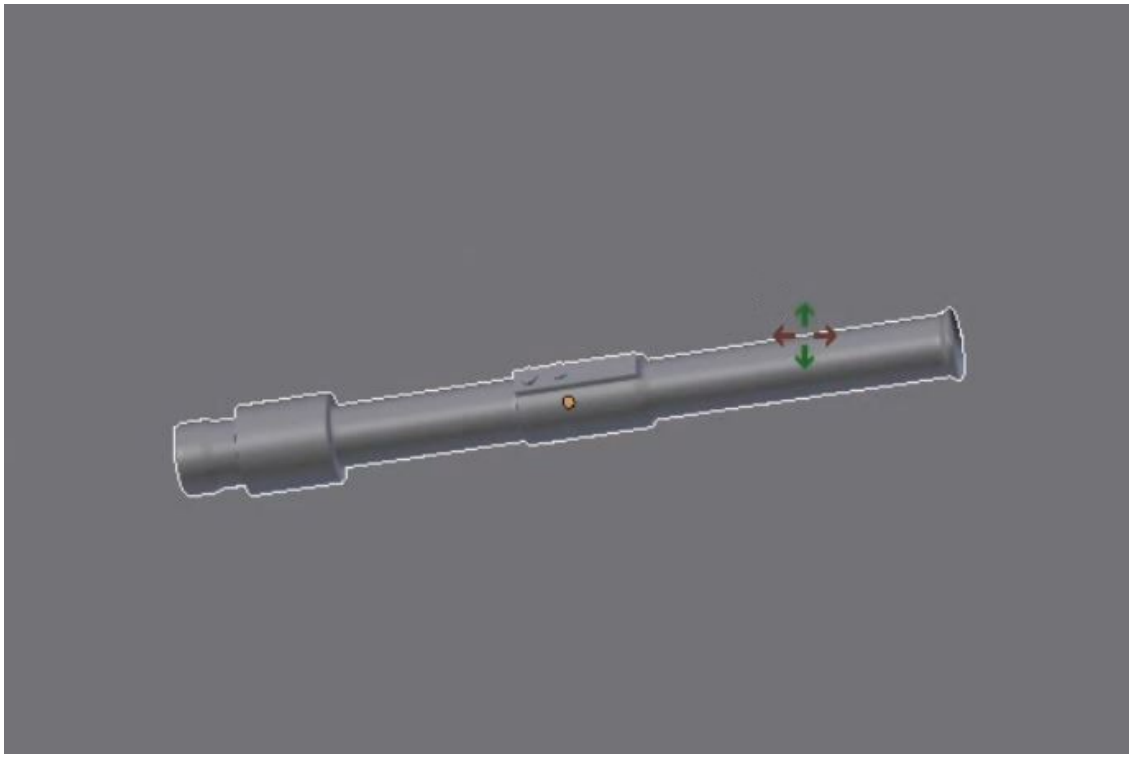

*Align Camera to View*

*  *  *

# La spada laser: animazioni Shape Keys e bagliore animato

In questo tutorial tratteremo due argomenti: l'animazione dell'accensione di una spada laser mediante le animazioni Vertex Key e il tremolio del bagliore del laser con il Nodes Compositing e le Textures Noise in Blender 3D 2.6.

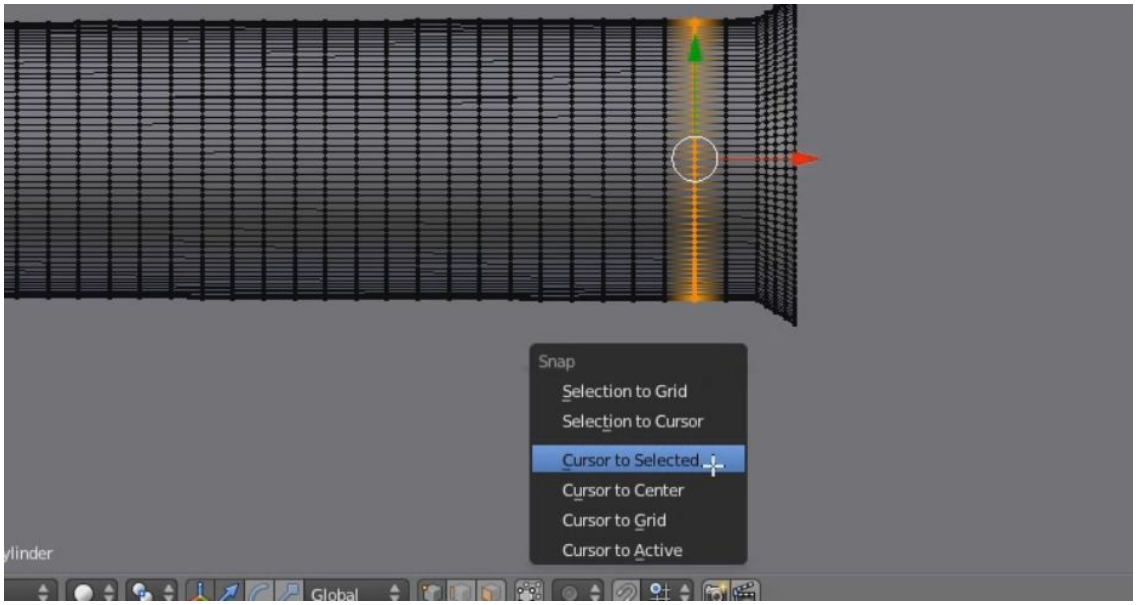
Iniziamo dall'accensione del laser, partendo da un modello 3D (che ho realizzato precedentemente) del solo manico della spada laser.



*La scena di partenza*

L'oggetto da utilizzare per realizzare il laser non è per niente difficile, anzi: utilizzeremo infatti un semplice cilindro, inizialmente con altezza praticamente nulla e posizionato dentro al manico, poi lo allungheremo mediante Vertex Shape e gestiremo l'effetto bagliore con il Nodes Compositing.

Per prima cosa, visto che l'oggetto è poggiato sul piano, andiamo in una vista TOP Ortho e posizioniamo il 3D Cursor all'interno del manico, non proprio in un punta ma leggermente dietro, in modo da nascondere il cilindro-laser dentro il manico quando la spada laser è "spenta", per cui: in Edit Mode, selezioniamo uno degli anelli del corpo della spada laser (attenzione a disattivare lo Z-Clipping, nell'header della 3D View), premiamo SHIFT S, scegliamo "Cursor to Selected" e torniamo in Object Mode.

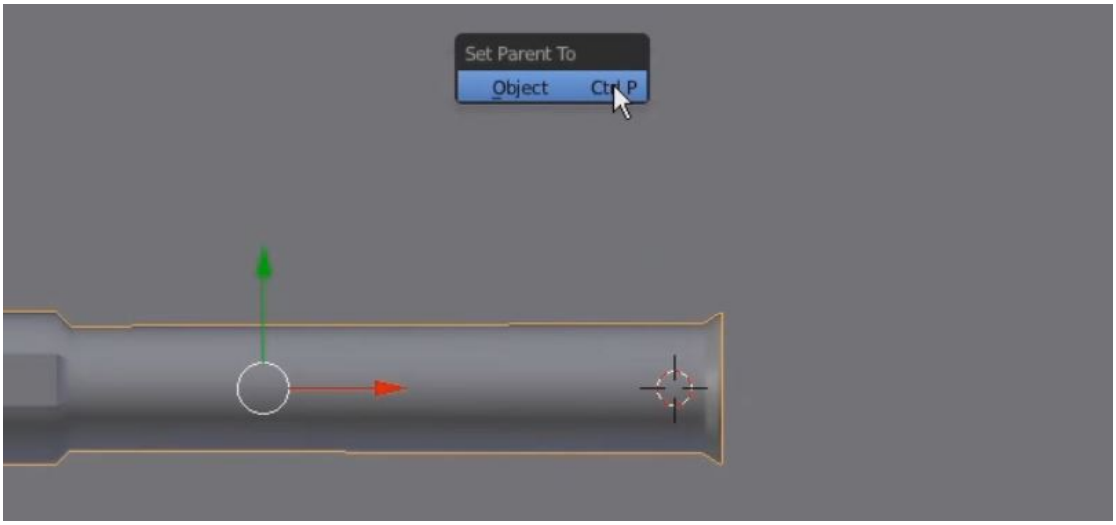



*Posizionamento del 3D Cursor per creare il cilindro "laser"*

Senza fare click col tasto sinistro in altri punti della scena, aggiungiamo un cilindro, che verrà posizionato proprio nel punto ove si trova il 3D Cursor.

Ruotiamo il cilindro (in questo caso, intorno all'asse Y), in modo da allinearlo al manico, e scaliamolo per farlo entrare dentro il manico, eventualmente spostandolo un po' indietro.

Il cilindro è un oggetto distinto dal manico, quindi per evitare errori in fase di animazione imparentiamolo al manico, così spostando, scalando o ruotando il manico trasformeremo anche il cilindro laser in maniera coerente; selezioniamo quindi prima il cilindro laser, poi, con una selezione multipla (SHIFT e click destro) anche il manico, premiamo CTRL P e confermiamo l'operazione di parenting dal menù che apparirà a video.

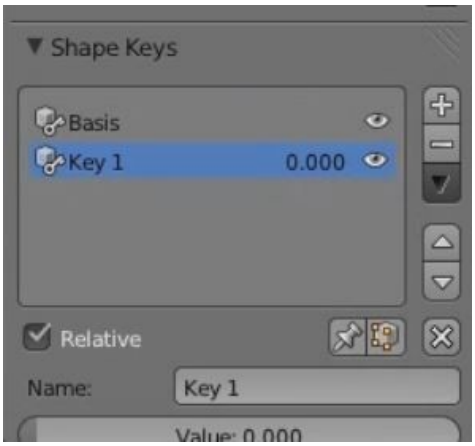



*Parenting del cilindro "laser" (che si trova all'interno del manico) al manico*

A questo punto, meglio passare in modalità di visualizzazione Wireframe per lavorare sul cilindro-laser; adesso, tra l'altro, abbiamo una selezione multipla, quindi selezioniamo solo il cilindro laser con un click del tasto destro su tale oggetto.

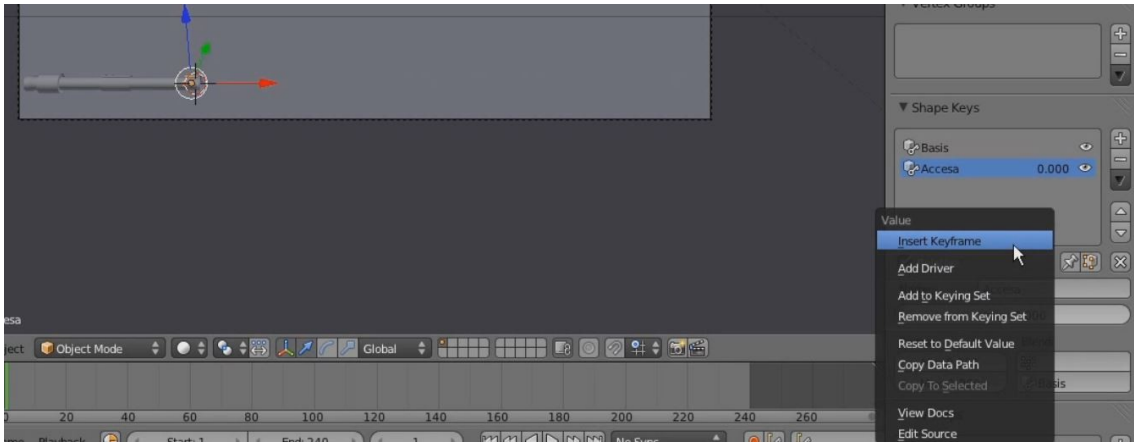
In Object Mode, apriamo la scheda Object Data, nella Properties Window, e facciamo click sul pulsante + nella sezione "Shape Keys", aggiungendo la Shape Key base.

Facciamo nuovamente click sul pulsante + in modo da aggiungere una seconda Shape Key, che avrà nome di default "Key 1" (ma possiamo rinominarla, ad esempio, in "accesa") e, con tale Shape Key selezionata, andiamo in Edit Mode, selezioniamo la base superiore del cilindro (la punta) e spostiamola, in modo da realizzare appunto la spada laser accesa.




*Creazione delle due Shape Keys*

Tornando in Object Mode, il cilindro tornerà alle dimensioni originali, perché il valore della Shape Key "accesa" è 0, ed è giusto che sia così al frame 1 dell'animazione, quando la spada laser è ancora spenta... anzi, registriamo questo valore come keyframe: posizioniamo il cursore del mouse sul campo Value della Shape Key "Accesa", facciamo click col tasto destro e scegliamo "Insert Keyframe".

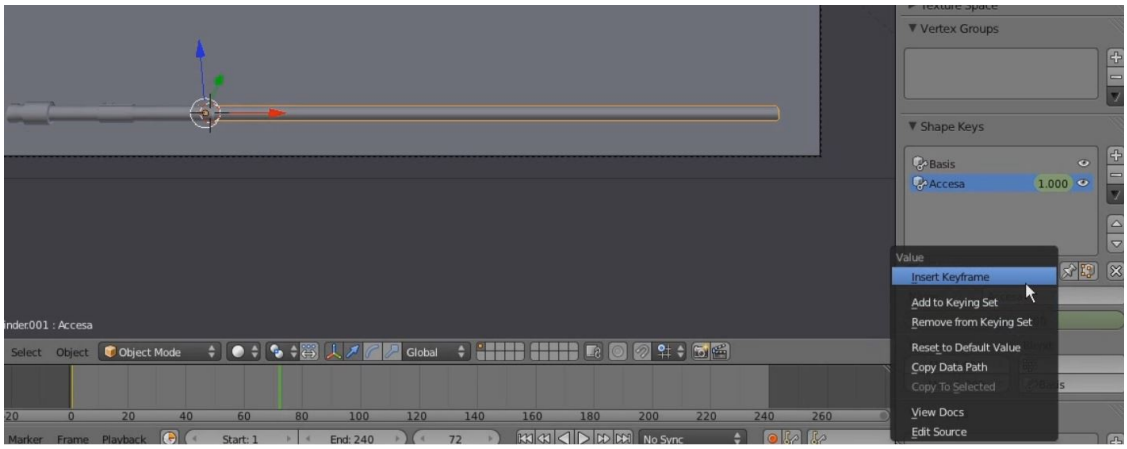



*Inserimento Keyframe per la Shape Key "accesa"*

Dobbiamo determinare in quanto tempo si accenderà la spada laser.

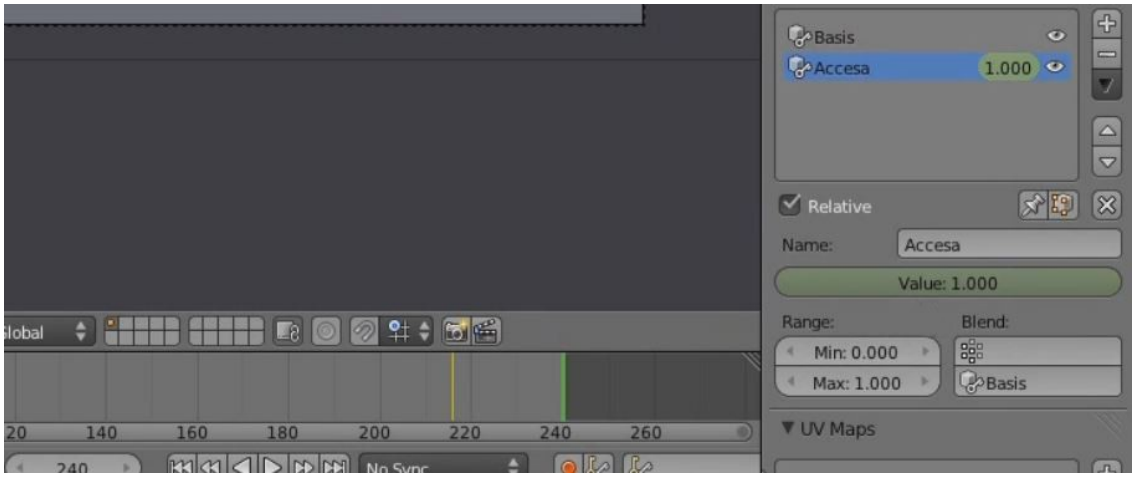
Per mostrare meglio l'effetto, sto impostando un tempo esageratamente lungo, 3 secondi (ossia 72 frames, in un videoclip a 24 frames al secondo), realizzando quella che forse è la spada laser più lenta della galassia, ma per questo rendering ci va più che bene.

Andiamo quindi al 72esimo frame della nostra animazione, sempre con il cilindro laser selezionato, quindi portiamo a 1 il valore di Value per la Shape Key e inseriamo un keyframe per Value; come possiamo vedere con l'anteprima dell'animazione, con ALT A nella 3D View: abbiamo realizzato l'animazione dell'accensione della spada laser.




*Keyframe al 72esimo frame (spada laser accesa)*

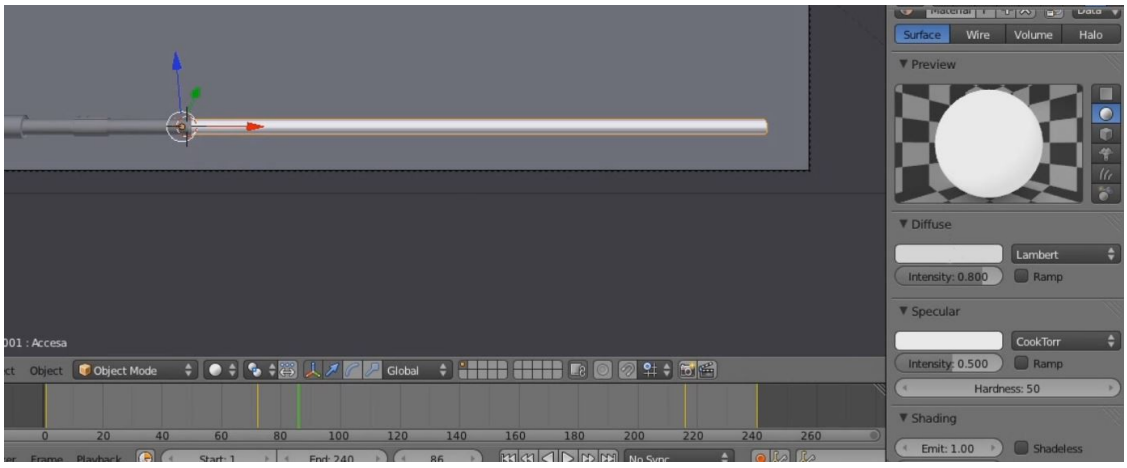
Mentre ci siamo, realizziamo anche lo spegnimento, questa volta magari in un secondo, quindi andiamo al frame 216 dell'animazione (240 – 24) e inseriamo un keyframe per Value a 1.0 (in questo modo, diremo a Blender di mantenere il valore a 1, quindi la spada laser accesa, dal frame 72 al numero 216), quindi andiamo all'ultimo frame dell'animazione (numero 240), portiamo il Value della Shape Key "Accesa" a 0 e inseriamo un keyframe.




*Spegnimento: due keyframes (216 e 240)*

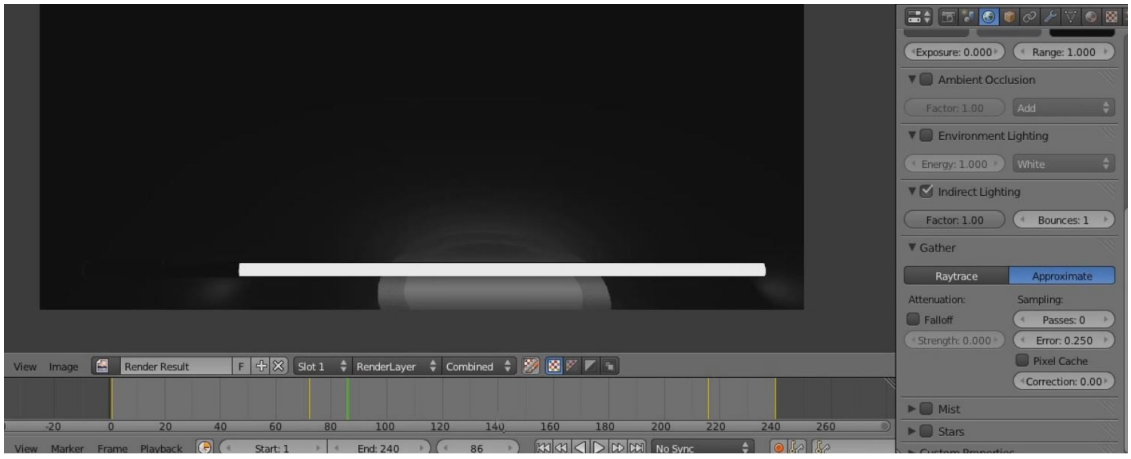
Da notare che, dal momento che il cilindro laser è figlio del manico, possiamo trasformare (animandolo) anche questo oggetto senza preoccuparci dell'animazione del laser, che verrà effettuata coerentemente da Blender.

Per quanto riguarda il Material del cilindro-laser, impostiamolo a bianco e con Emit a 1: ciò lo renderà completamente bianco; non produrrà il famoso bagliore (che realizzeremo in fase di Compositing, tra un attimo) ma lo renderà un emettitore luminoso, nel senso che con Indirect Lighting attivo il laser illuminerà l'ambiente circostante (ed ecco perché non stiamo utilizzando un semplice Material Shadeless).




*Il Material del laser*

Ho messo il manico su un Plane, a mo' di pavimento, proprio per mostrare questo effetto, quindi prima di passare al Compositing esaminiamolo: attiviamo l'opzione Indirect Lighting nella scheda World all'interno della Properties Window (con Gather Approximate, obbligatoriamente), andiamo ad un frame della Timeline dove il laser è completamente fuori e avviamo il rendering, osservando il risultato (ovviamente l'effetto si nota di più se non ci sono fonti di luce o altri effetti di illuminazione, tipo Ambient Occlusion).

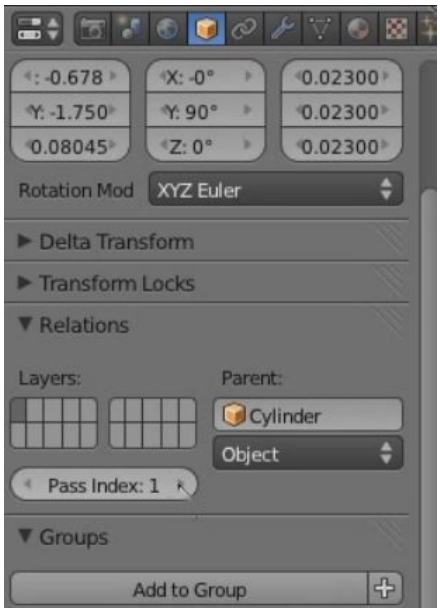



*Il rendering con Indirect Lighting*

* * *

Passiamo quindi alla seconda parte di questo tutorial: la post-produzione, mediante Nodes Compositing, per aggiungere un bagliore con disturbo animato alla spada laser; non parlerò in dettaglio dei singoli Nodes o delle loro funzioni, limitandomi a descrivere lo schema dei Nodi: per una trattazione approfondita, rimando al mio ebook "*Il Compositing in Blender 2.5*", sezione ebooks del mio sito web.

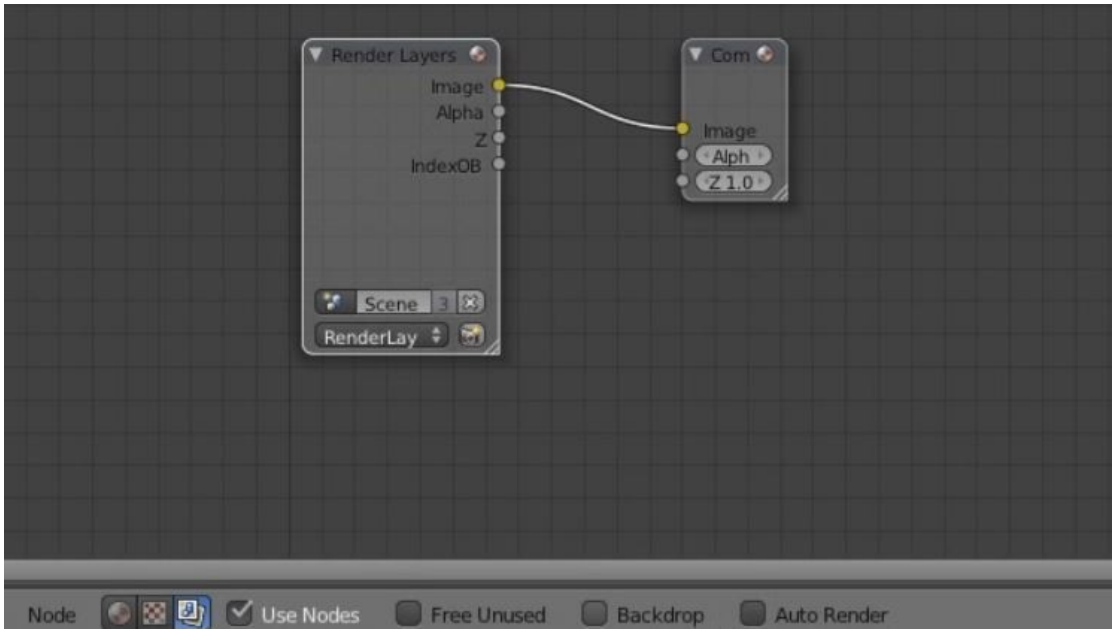
Per fornire un bagliore (ottenuto mediante Blur) in Compositing al laser, dobbiamo isolare tale oggetto dal resto, quindi selezioniamo il cilindro laser e, nella scheda Object all'interno della Properties Window, impostiamo il *Pass Index* a 1 per tale elemento.




*Pass Index 1*

Nella sezione Layers della scheda Render attiviamo la casella Object Index per poter ricavare l'informazione *Pass Index* (detta *IndexOB*) dal nodo Render Layers e passiamo al Layout Compositing, concentrandoci sulla finestra Node Editor.

Selezioniamo la casella *Compositing Node* e attiviamo *Use Nodes*, quindi cancelliamo il collegamento di default tra Render Layers e Composite.

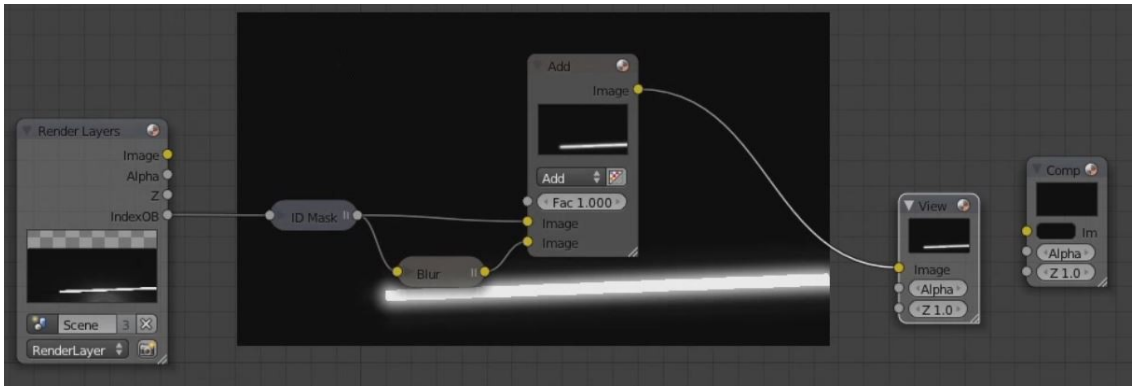


*Preparare lo schema di nodi nell'Editor NODES*

Dal momento che il cilindro laser è completamente bianco, possiamo lavorare direttamente sulla maschera in arrivo da *IndexOB*, a patto ovviamente di isolare l'elemento di Pass Index 1, ossia il laser, quindi colleghiamo un *Nodo Convertor ID Mask* con Index 1 all'uscita *IndexOB* del nodo.

Aggiungiamo un Nodo *Filter Blur*, con modalità *Fast Gaussian* e valore, ad esempio, 24, sia per X che per Y, collegando l'output di *ID Mask* a tale Nodo, in modo da ottenere una sfocatura del laser, quindi aggiungiamo un Nodo *Color MIX*, con modalità ADD (la somma) e Factor 1, collegandogli in input:

- sul primo canale Image, l'output Alpha del Nodo ID Mask;
- sul secondo canale Image, l'output del Nodo Blur.



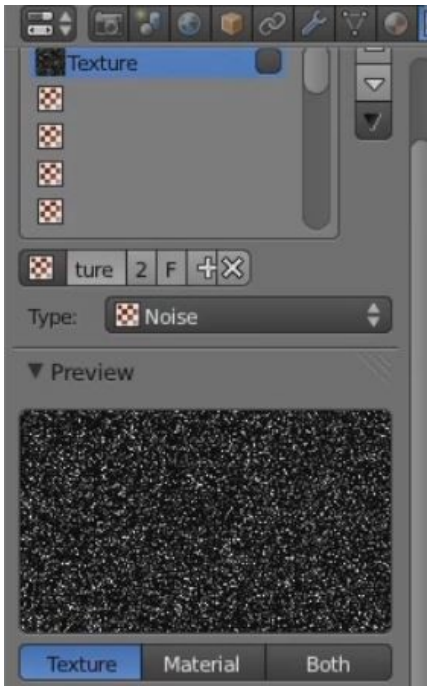

*Lo schema dei nodi "parziale" (senza rumore casuale)*

Abbiamo quindi un bagliore, che coloreremo tra poco con un Nodo ad hoc in modo da sommarlo, poi, al rendering iniziale, ma manca ancora qualcosa: un po' di rumore casuale, di granulosità, in questo bagliore, che in questo modo è troppo pulito (d'altronde, si tratta di una sfocatura gaussiana).

È una cosa poco nota (ma descritta, ad esempio, nel mio ebook sulle Textures in Blender 3D 2.5), ma l'output di una Texture Noise cambia da un frame all'altro; per questo motivo, è indicata per simulare il rumore di una ripresa reale in un rendering digitale, cosa utile se bisogna fondere appunto un rendering con un'immagine o videoclip dal vivo.

Nel nostro caso, utilizzeremo questa caratteristica per aggiungere del rumore casuale, animato automaticamente, al bagliore della spada laser.

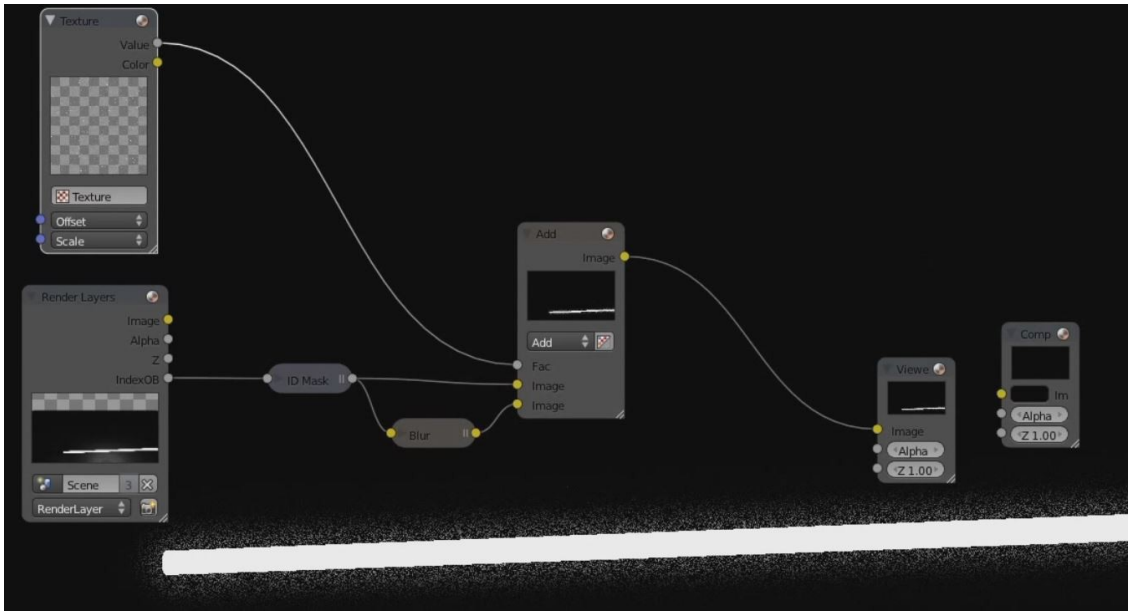
Andiamo quindi nella Properties Window, scheda Textures, e aggiungiamo una Texture di tipo Noise al cilindro laser, ma disattiviamola nel pannello delle Textures: non vogliamo, infatti, utilizzarla per il Material, ma solo nel Nodes Editor come fattore di disturbo per la sfocatura gaussiana.




*Creazione della Texture NOISE, disattivata*

Torniamo al Node Editor e inseriamo un Nodo Input Texture, scegliendo la Texture Noise appena creata.

Colleghiamo l'output Value del Nodo Input Texture all'input Factor del Nodo Mix Add che somma la maschera Alpha con la sfocatura Gaussiana; utilizziamo Value perché non vogliamo il colore della Texture, ma vogliamo utilizzare i valori in scala di grigio per determinare in quali punti applicare (e con quale intensità) la sfocatura gaussiana all'immagine finale.




*Collegare la Texture NOISE in Compositing*

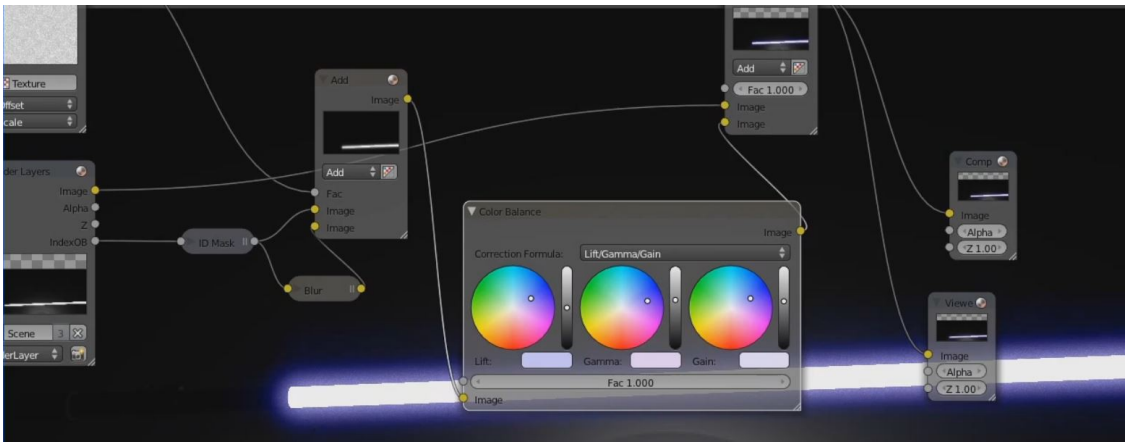
Facciamo un primo rendering: con ogni probabilità, il rumore prodotto con le impostazioni di default sarà eccessivo.

Per ridurlo, dobbiamo agire sulle impostazioni della Textures utilizzata, nella Properties Window, in particolare rendendo più chiara l'immagine o comunque riducendo i punti neri presenti.

Ci sono vari modi per effettuare questa operazione, agendo sui campi della sezione Colors della Texture, ma probabilmente il più semplice consiste nell'aumentare la luminosità Brightness (portandola, ad esempio, a 1.5) e diminuendo il contrasto (portandolo ad esempio a 0.5); come detto, i colori qui indicano valori, intensità dell'effetto Factor di miscelazione delle Textures, quindi il bianco o grigio chiaro indicano la presenza completa o quasi dell'effetto.

Torniamo all'Editor dei Nodi: duplichiamo il Nodo Mix Add in modo da sommare l'output del primo Nodo Mix Add all'immagine in arrivo da Render Layers e passare il tutto a Composite, ma

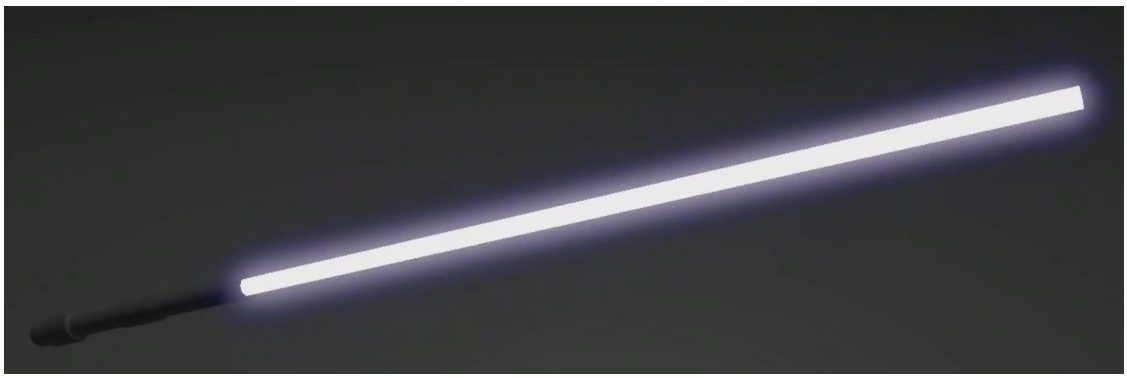
tra i due Nodi Mix Add poniamo un Nodo Color – *Color Balance* e giochiamo un po' con gli strumenti messi a disposizione da tale Nodo per fornire una tinta al bagliore della nostra spada laser.




*Utilizzo del Nodo COLOR - COLOR BALANCE*

Tecnicamente la nostra spada laser è pronta, sia a livello di modellazione che di rendering, per essere inserita in un set virtuale o applicata a riprese reali, per un effetto speciale.

Abbiamo visto, quindi, come realizzare l'animazione con le Shape Keys e come gestire il bagliore con rumore casuale animato e la colorazione in Compositing.



*Screenshot del risultato finale*

*  *  *

# Modellare e renderizzare un cacciavite con Looptools Loft

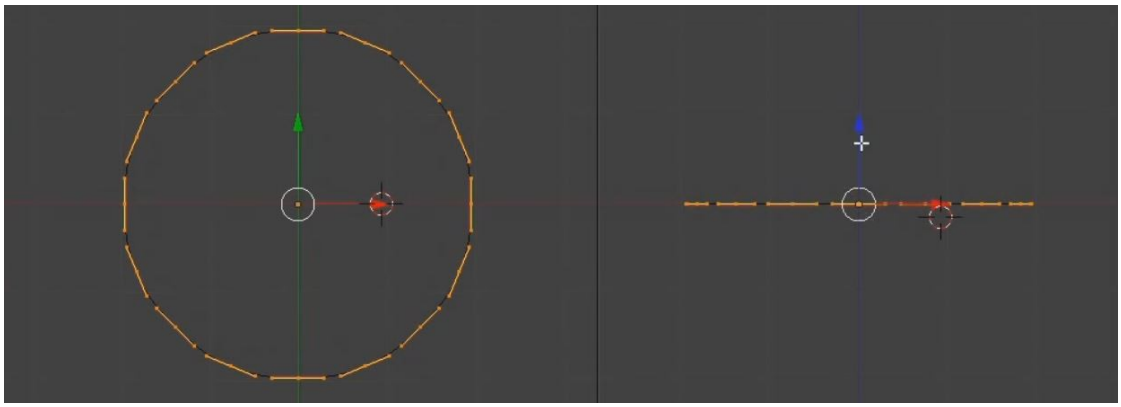
In questo tutorial vedremo un modo per modellare un cacciavite in Blender 3D utilizzando lo strumento Loft dell'Add-On LoopTools sulle Bezier Curves.

Esistono vari modi per modellare un oggetto come questo ma, in ogni caso, il procedimento qui descritto ci consente di fare pratica con lo strumento Loft.

Loft crea una superficie, un solido e in generale un rivestimento partendo da sagome: in pratica, modelleremo dei profili e da lì otterremo una superficie; è una tecnica del tutto simile a quella dello skinning con le NURBS Surface ma in quel caso il rivestimento NURBS presenta delle distorsioni rispetto al risultato che ci si aspetta (ed ecco perché useremo Loft).

Loft funziona quando i profili appartengono ad un unico oggetto (ossia: forme definite in Edit Mode, all'interno di un'unica mesh) e, appunto, con le mesh, ma per certi elementi (come il cacciavite) si lavora meglio con le Curves; fortunatamente, questo non è un problema.

Passiamo alla pratica! Aggiungiamo un oggetto Bezier Circle alla scena, andiamo in Edit Mode e suddividiamolo un paio di volte, ottenendo così un cerchio con 16 punti di controllo.

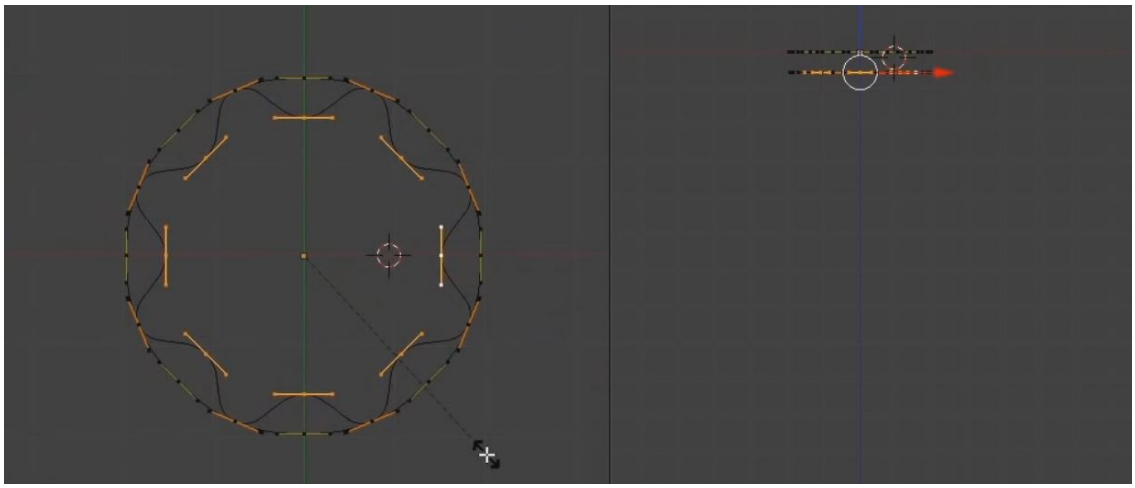


*Il Bezier Circle suddiviso*

Disattiviamo la visualizzazione delle Normali, deselezionando l'opzione “Normals” in Curve Displays, nella Transform Window, in modo da visualizzare meglio la curva.

In Edit Mode, selezioniamo tutti i punti di controllo e duplichiamo il profilo, quindi spostiamo la copia un po' in basso lungo l'asse verticale dell'oggetto (che nel mio caso, lavorando in vista TOP Ortho, corrisponde all'asse Z globale della scena).

Selezioniamo, in questo nuovo profilo, i punti di controllo in maniera alternata, uno sì e uno no, quindi scaliamo tutta la selezione verso il centro, ottenendo il profilo desiderato per il manico del cacciavite... o meglio: questo è il profilo che voglio io, ma va da sé che potete personalizzare come volete questa forma e in generale questo è solo un esempio di utilizzo di Loft, potete applicarlo a qualsiasi oggetto e qualsiasi forma!

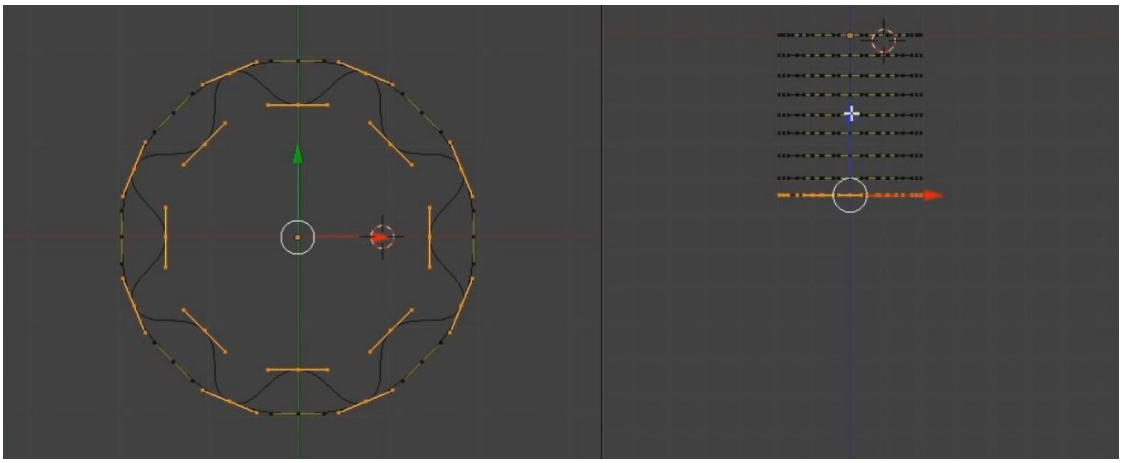


*Selezione e Scaling dei punti di controllo*

Cercate, comunque, di mantenere vicini i vari profili, soprattutto quando ci sono cambiamenti notevoli nella forma, e mantenere costante il numero di punti di controllo o vertici delle sagome (ecco perché ho suddiviso due volte anche il profilo perfettamente circolare, anche se in teoria in quel caso non c'era bisogno di introdurre nuovi punti di controllo).

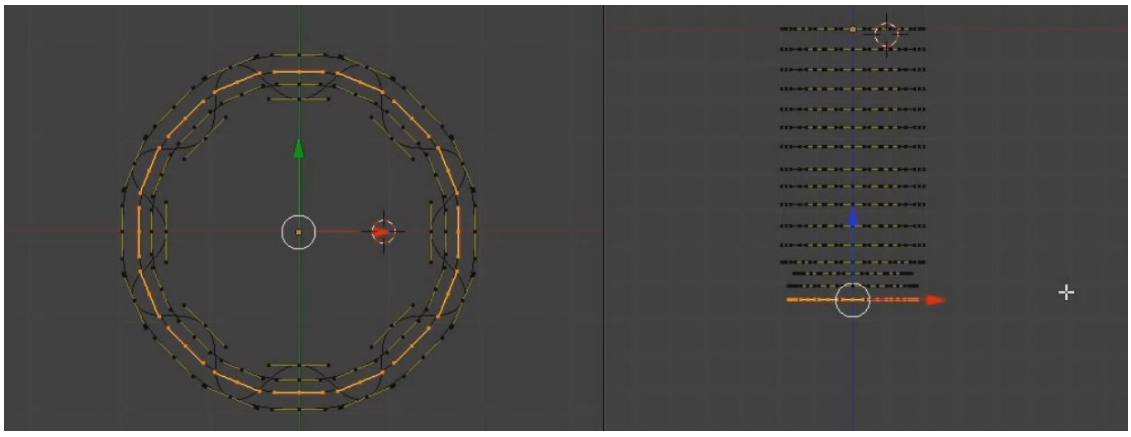
Per selezionare velocemente un solo profilo, al fine di duplicarlo e spostarlo in basso, selezioniamo un punto di controllo e premiamo *L* (shortcut per *Linked*, “collegato”), quindi duplichiamo il profilo del manico e spostiamolo più in basso. Ripetiamo questa operazione un paio di volte (qui sto

definendo a occhio le dimensioni dell'oggetto, ma per riproduzioni realistiche è meglio utilizzare una Background Image; in ogni caso, potremo scalare il modello su uno o due assi in seguito, quindi per il momento non preoccupiamoci più di tanto).



*Duplicazione dei profili*

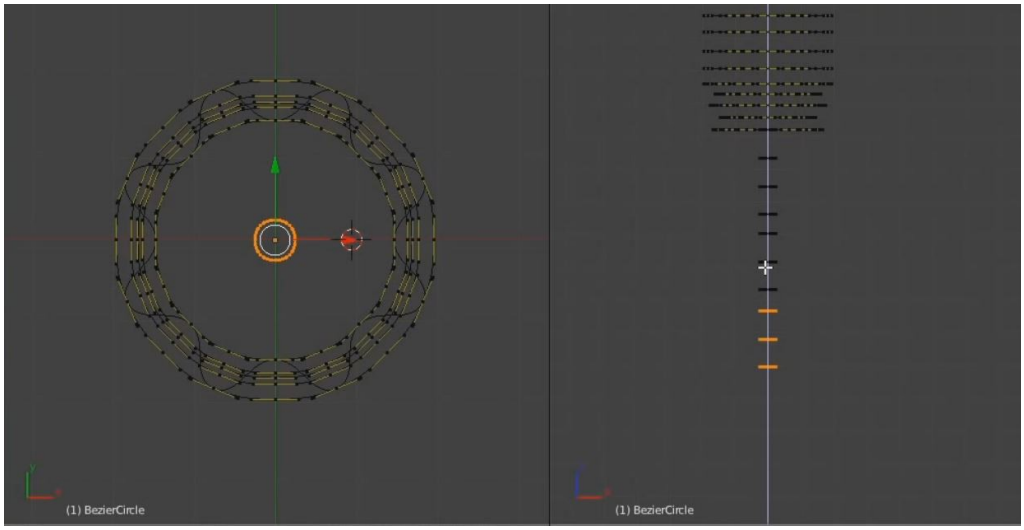
Definite le sagome del manico, riprendiamo il primo profilo (quello perfettamente circolare), duplichiamolo e spostiamolo in basso, facendo un paio di copie (continuando a spostarle in basso) e scalandole, per variare ulteriormente il profilo.



*Completare il profilo del manico*

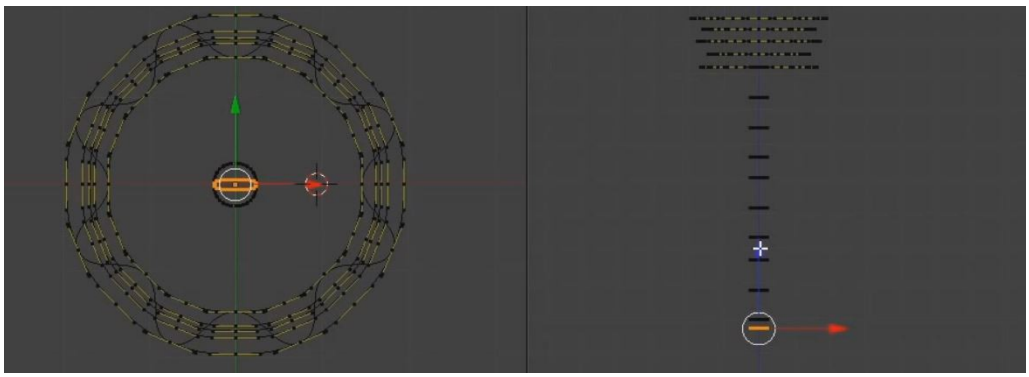
Facciamo infine una copia dell'ultimo profilo ma scaliamola senza spostarla lungo l'asse verticale: questa sagoma rappresenterà il primo profilo della punta metallica.

Partendo da questa sagoma, quindi, facciamo altre copie e spostiamole lungo l'asse verticale, senza scalarle: costituiranno la parte metallica.




*Definizione della parte metallica*

Arrivati alla punta, deformiamo come vogliamo il profilo, in modo da ottenere ad esempio una punta a stella, quindi facciamo un paio di copie anche di questo profilo e spostiamole in basso, scalandole in punta.

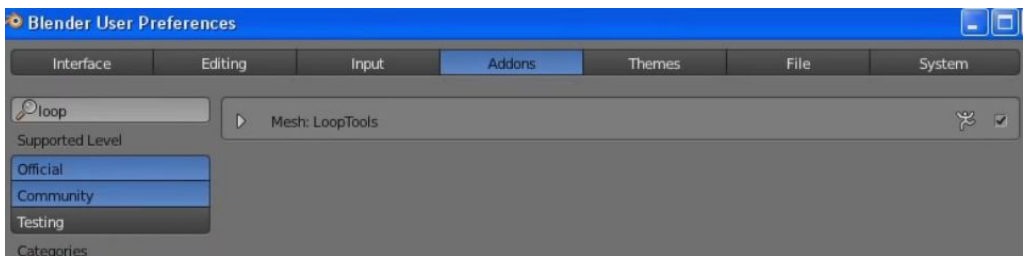


*Definizione della parte terminale (la punta)*

Definiti tutti i profili, torniamo in Object Mode e, con l'oggetto selezionato, premiamo ALT C e scegliamo “Convert to Mesh”.

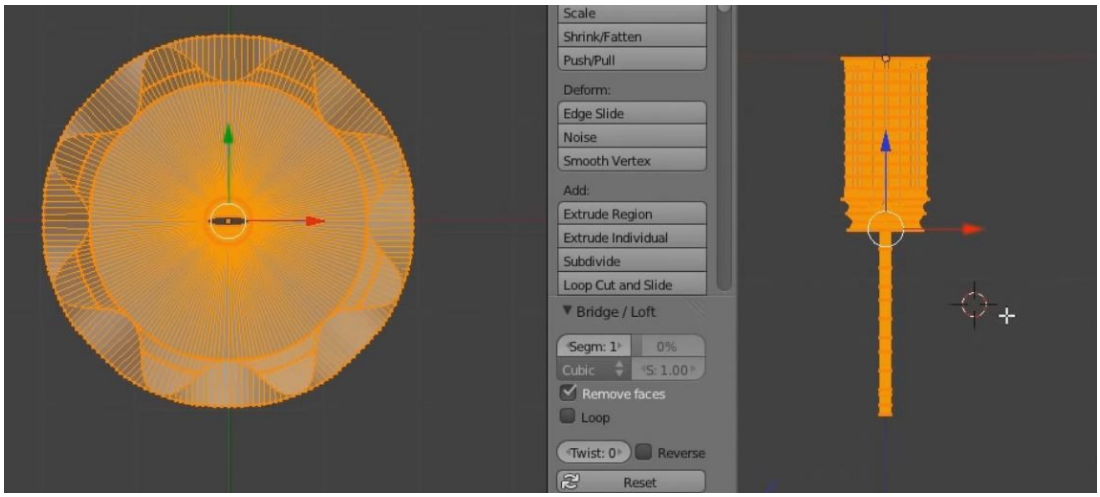
A questo punto, andiamo in Edit Mode e selezioniamo tutti i vertici dell'oggetto.

Di default, l'Add-On Loop Tool non è attivo, quindi apriamo la finestra User Preferences, pannello Add-On, scriviamo Loop nella casella filtro di ricerca e attiviamo l'Add-On LoopTools.



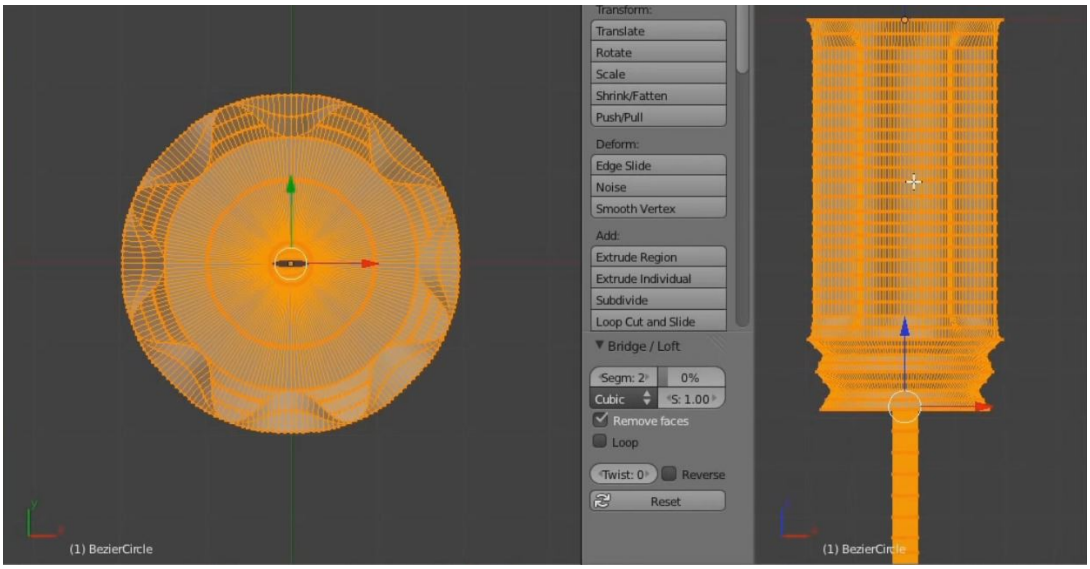

*Attivazione dell'Add-On Loop Tools*

Torniamo alla 3D View: con tutti i vertici selezionati, premiamo W e scegliamo *Loop Tools - Loft* dal menù *Specials*. Le impostazioni di default dell'Add-On, nel menù Bridge-Loft nella Tool Shelf, dovrebbero andare bene; magari possiamo aumentare il numero di segmenti e impostare l'interpolazione cubica.




*Utilizzo di Loft*

In Object Mode, scaliamo l'oggetto intorno a uno o più assi o comunque modifichiamolo per migliorarne la forma.




*Il risultato parziale*

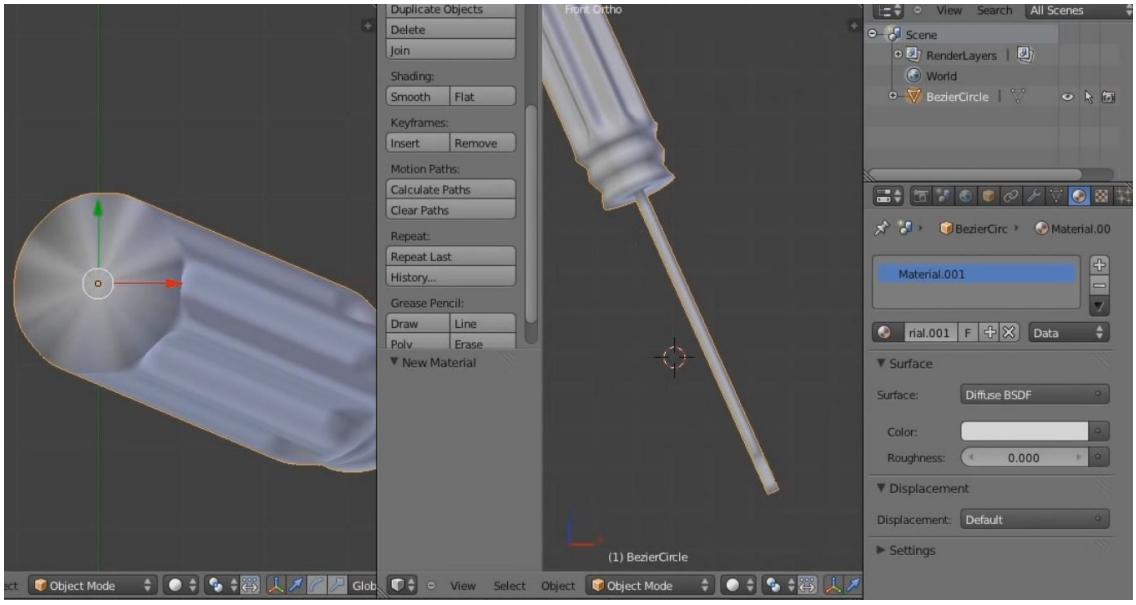
Torniamo in Edit Mode per alcuni ritocchi.

Chiudiamo le due estremità (selezioniamo i vertici, duplichiamo sul posto e facciamo un *Merge At Center*, prima per la punta e poi per la base superiore dell'oggetto; attenzione allo Z-Clipping, quando selezioniamo i profili).

Selezioniamo tutti i vertici del modello, che ora è chiuso, e scegliamo *Recalculate Normals* nella Tool Shelf. Clicchiamo su *Set Smooth*.

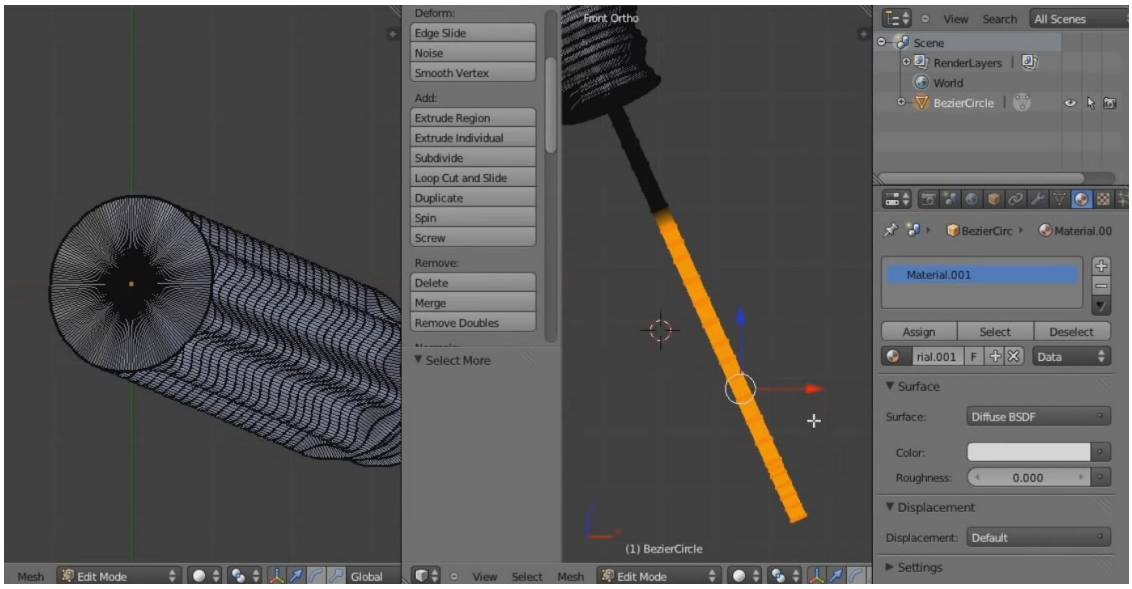
Torniamo in Object Mode: il modello è pronto, ora vediamo come fornirgli due Materiali per il rendering in Cycles.

Forniamo un primo Material a tutto l'oggetto; questo sarà il Material che utilizzeremo per il manico.



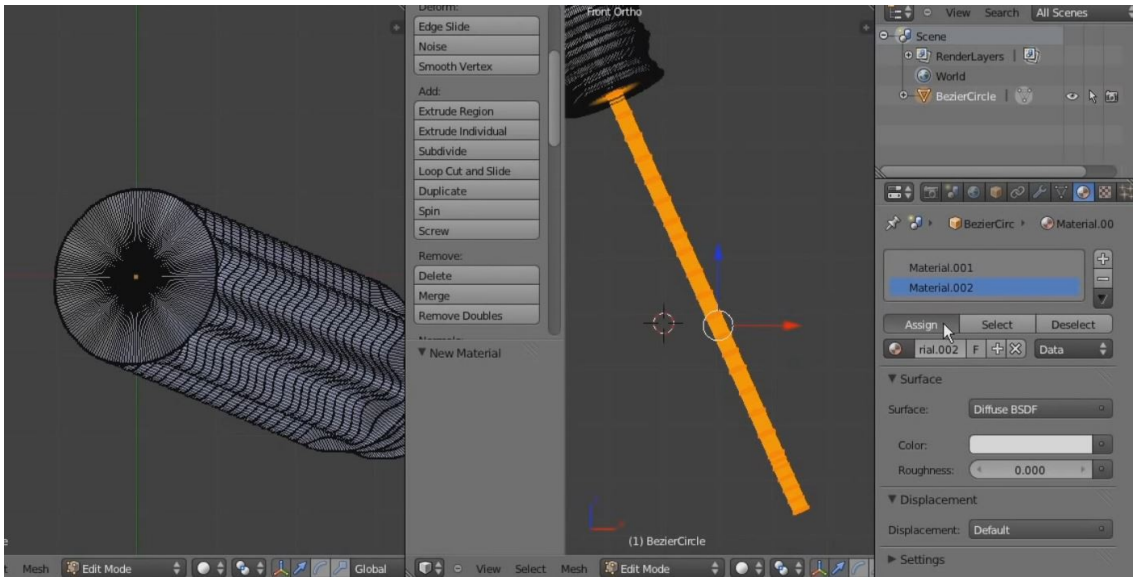

*Il primo Material*

In Edit Mode, selezioniamo i vertici della punta, quindi premiamo CTRL + per allargare la selezione ai vertici adiacenti, selezionando velocemente tutta la porzione metallica.




*Allargare la selezione con CTRL +*

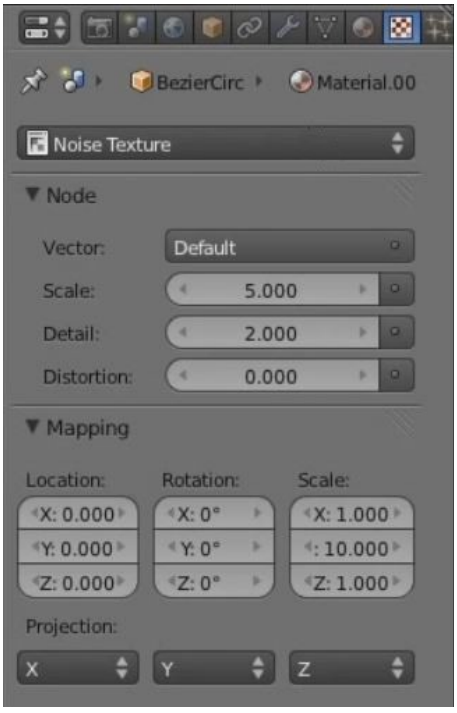
Nella scheda Materials, nella Properties Window, aggiungiamo un secondo sotto-materiale e clicchiamo su *Assign* per assegnarlo alla selezione.




*Assegnare il secondo sottomateriale alla parte metallica*

Torniamo in Object Mode.

La scelta dei Materiali, soprattutto per il manico, sta a voi: potete realizzare un Material di tipo plastica, vetro, ceramica, eccetera; può sembrare una scelta strana, ma per il rendering promo di questo tutorial ho impostato un Material Velvet.

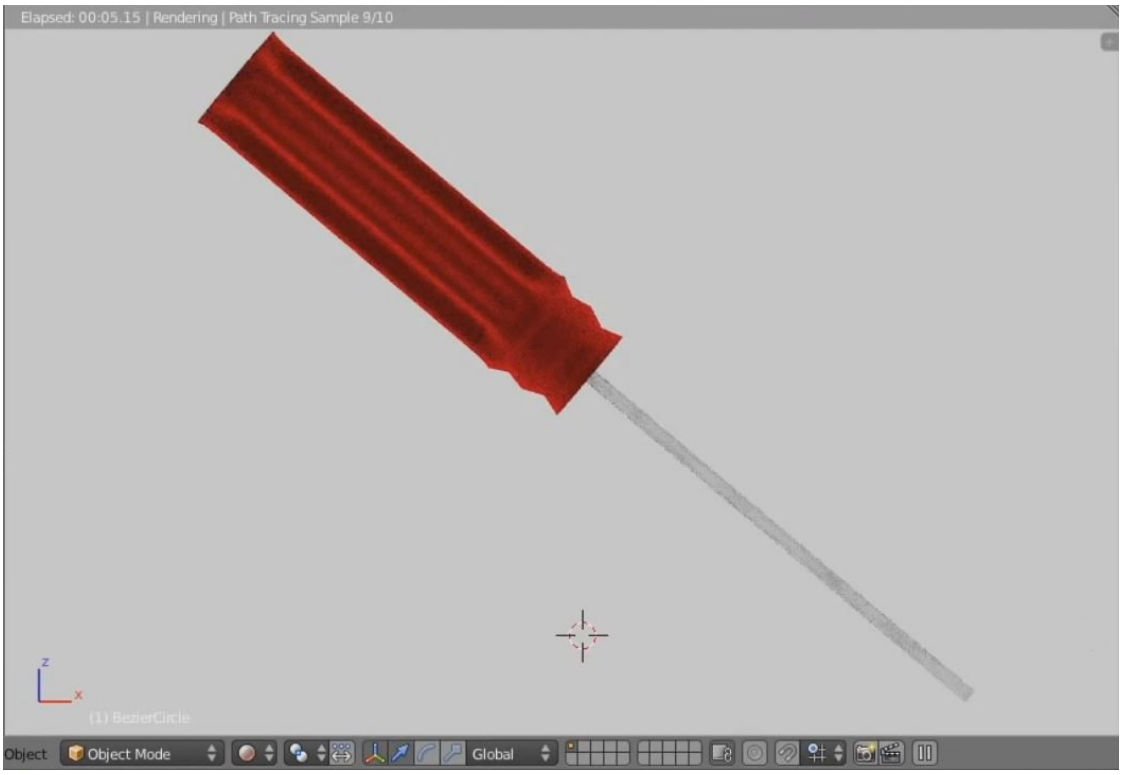



*La Texture NOISE "stirata" per ottenere l'effetto "brushed"*

Per quanto riguarda la punta metallica, la scelta è praticamente obbligata: Glossy, ma possiamo agire sul fattore Roughness per non avere un metallo troppo lucido e soprattutto possiamo aggiungere una Texture per graffiare un po' il metallo, con un effetto tipo “alluminio spazzolato”... vediamo come.

Con il Material Glossy selezionato, clicchiamo sul pulsante accanto al valore *Roughness* e aggiungiamo una Texture “Noise” per tale parametro. Nella scheda Texture, cambiamo Scale Y della Texture, ad esempio in 10 o più: questo avrà l'effetto di stirare il rumore in una direzione, realizzando appunto dei graffi.

Completano il tutto luci, ambiente o l'utilizzo di una Texture su World, magari un'immagine HDR, per rendere più interessante il rendering... ma questo è un altro discorso! Per questo tutorial, invece, è tutto!

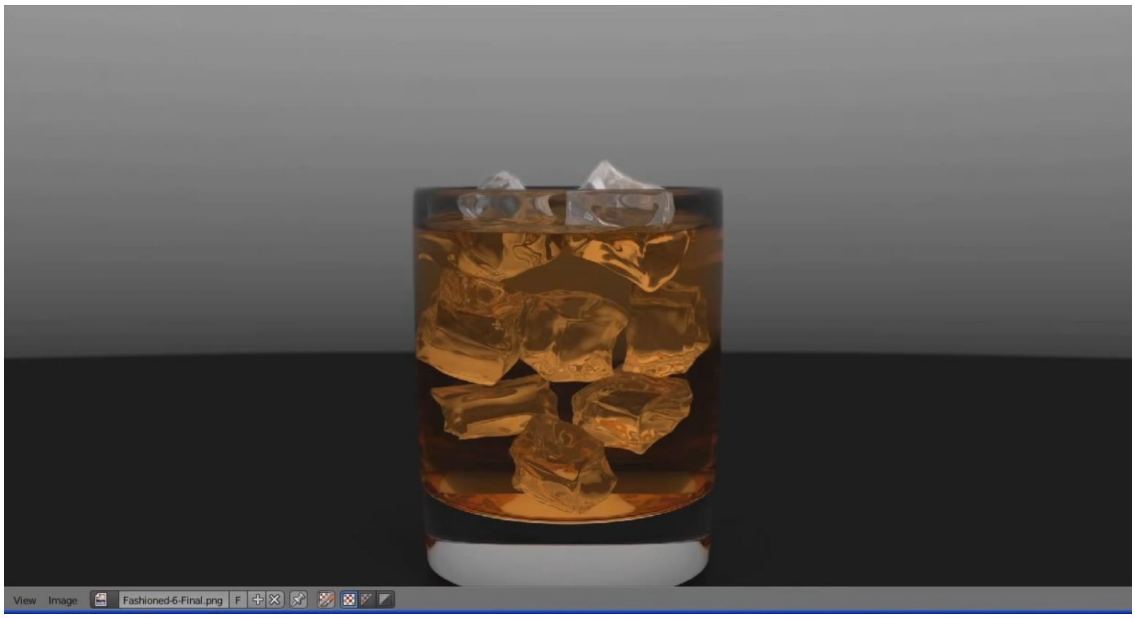



*Il risultato finale (nella preview interattiva, in una 3D View)*

*  *  *

# Old Fashioned: tutorial Cycles su bicchieri e liquidi statici

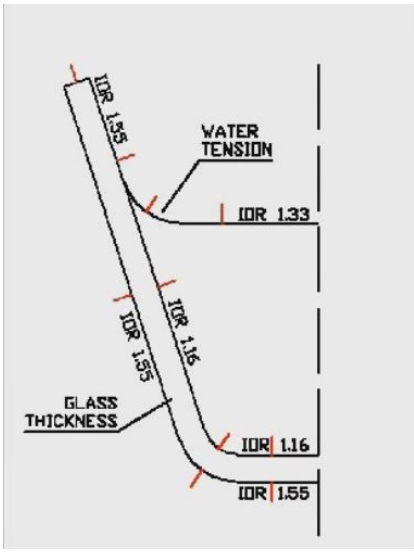
Questo rendering è un piccolo omaggio alla serie televisiva Mad Men, di cui sono un grande fan: si tratta di un bicchiere di Old Fashioned, il drink preferito di Don Draper, realizzato in Blender 3D 2.63 e Cycles.




*Il rendering che otterremo*

In questo tutorial vedremo come modellare il bicchiere, con il liquido statico al suo interno, nella maniera corretta per renderizzarlo poi in Cycles; c'è qualche differenza rispetto al tutorial sul bicchiere di latte, che ho pubblicato precedentemente, visto che in Cycles (e altri motori di rendering unbiased, come Yafaray) dobbiamo creare mesh "corrette", cioè chiuse, con spessore e diversi indici di rifrazione per le varie parti e normali che puntano nelle giuste direzioni.

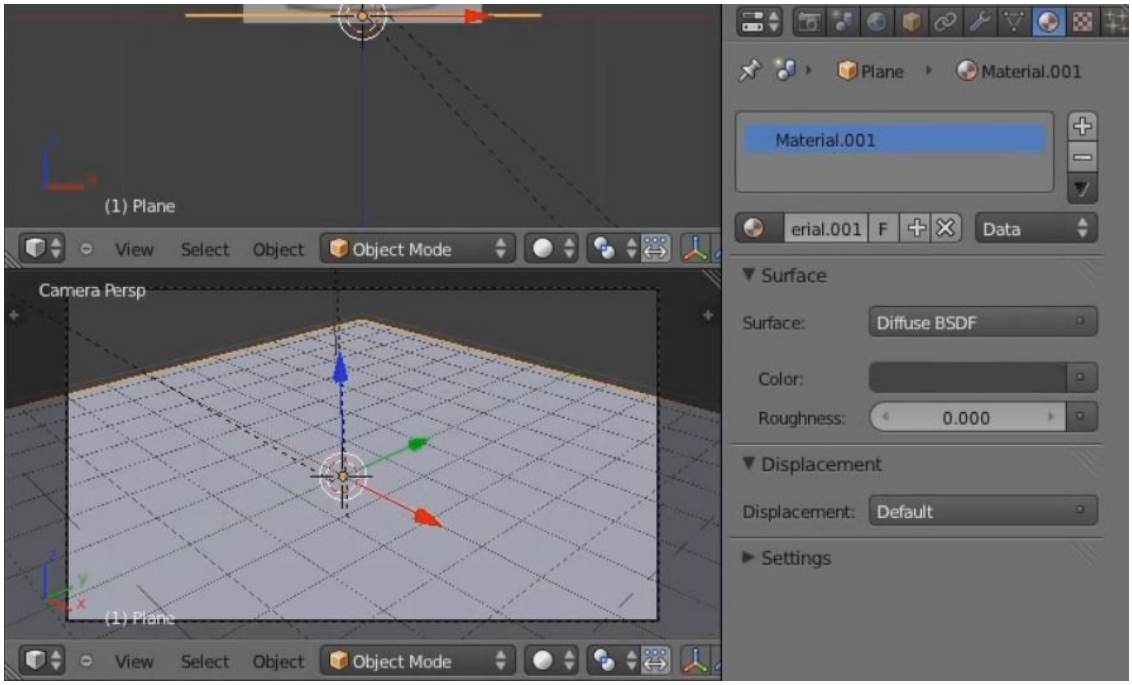
L'immagine a lato è tratta dal manuale Yafaray originale (relativo alla versione 0.1 di Yafaray e 2.4 di Blender), ma va bene anche per Cycles; come potete vedere, le facce esterne del bicchiere hanno indice di rifrazione (IOR) 1.5, quelle del liquido in alto 1.3 (l'indice proprio del liquore) e quelle di confine tra liquore e vetro, che sono singole facce e non due mesh separate, hanno le normali verso l'interno e indice di rifrazione pari a IOR del vetro fratto IOR del liquore, ossia 1.15.



*L'impostazione degli IOR per bicchieri con liquidi*

Passiamo al motore di rendering Cycles, cancelliamo il cubo di default, spostiamo il 3D Cursor al centro della scena con SHIFT-S e "Cursor to Center" e aggiungiamo un Plane.

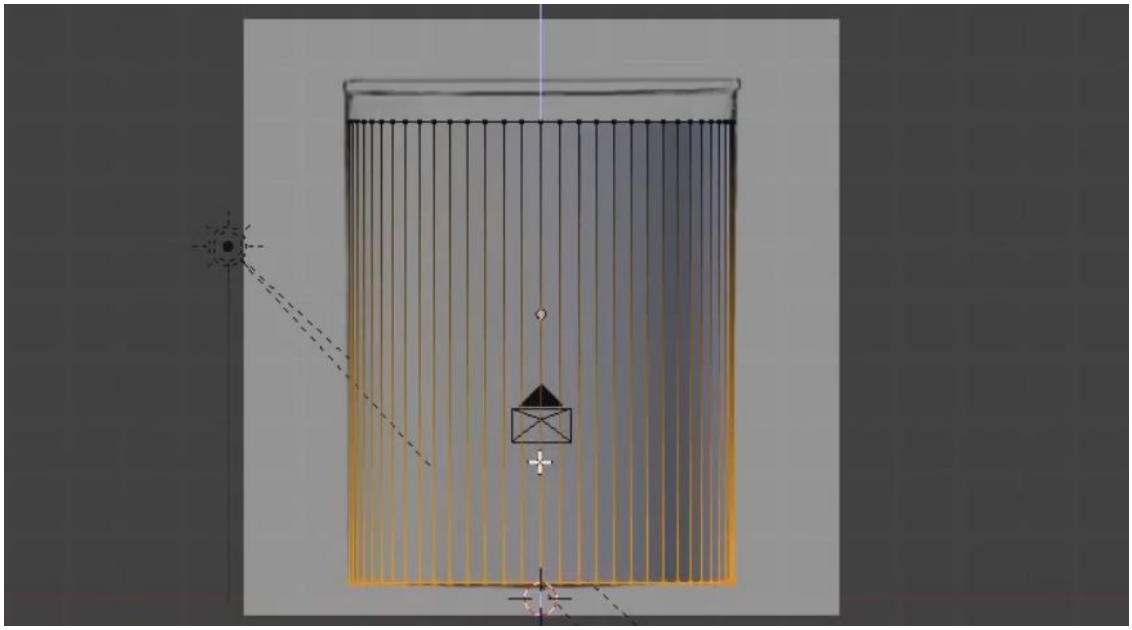
Creiamo un Material con Shader Diffuse di base per il Plane, con colore grigio scuro.



*La creazione del Plane*

Posizioniamo e orientiamo la telecamera in modo da sistemare l'inquadratura; ad esempio, io sto usando -10 e 3 per Location Y e Z e 85 gradi per Rotation X, gli altri valori a 0.

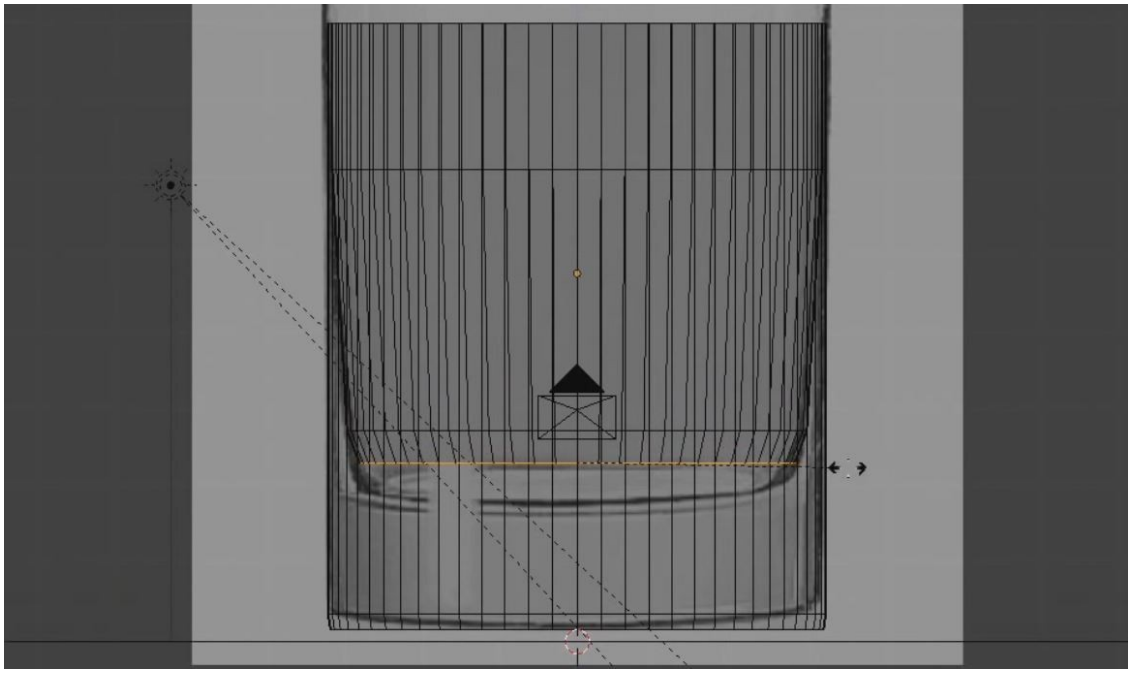
Ora iniziamo a modellare il bicchiere di vetro; potremmo farlo con una curva Bezier, da convertire in mesh e rendere solida con Spin, ma per andare velocemente aggiungeremo un cilindro con, ad esempio, 64 suddivisioni; adesso, aggiungiamo un taglio con Ring Cut (CTRL R) in basso, per restringere un po' la base.



*Modellazione del bicchiere*

Dal momento che sto usando la versione 2.63 di Blender, con la modellazione Bmesh, la base superiore è composta da un'unica faccia, non più un vertice centrale e gli spigoli di collegamento, quindi passiamo in modalità selezione facce, selezioniamo la base superiore e cancelliamola con X o CANC e scegliendo "Faces" dal menù che apparirà.

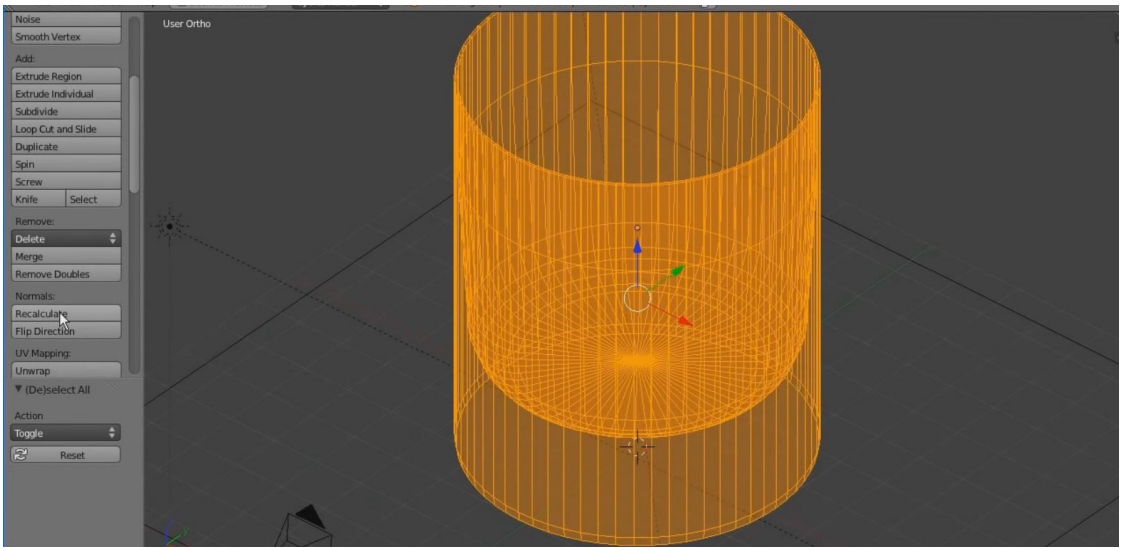
Torniamo alla modalità selezione vertici e selezioniamo l'anello superiore del cilindro con ALT e click del tasto destro del mouse su uno spigolo, quindi estrudiamo, scaliamo e spostiamo i nuovi vertici per modellare la parte interna del bicchiere: dobbiamo quindi estrudere, scalare, spostare, per un certo numero di volte (creando, nel mentre, un anello, lì dove dovremo posizionare la superficie del liquido).



*La modellazione della parte interna del bicchiere*

Alla base, premiamo W e scegliamo *Merge at Center* per chiudere la mesh.

In Edit Mode, selezioniamo tutto, con A, e clicchiamo su *Recalculate Normals*, nella Tool Shelf; adesso, le Normali puntano nella giusta direzione per tutte le parti dell'oggetto.



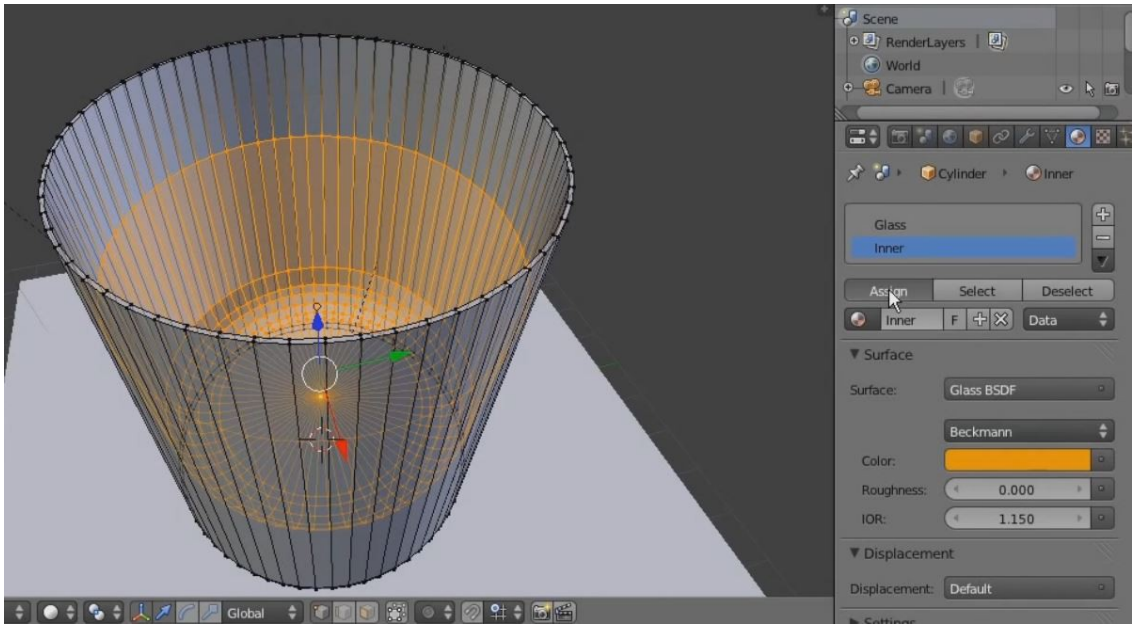

*Recalculate Normals, con la mesh chiusa*

Prima di aggiungere la superficie del liquido, creiamo un Material con due sottomateriali per il bicchiere.

Il primo Material, applicato a tutta la mesh, sarà di tipo Glass (vetro), con colore bianco puro, Roughness 0.0 e indice di rifrazione 1.5.

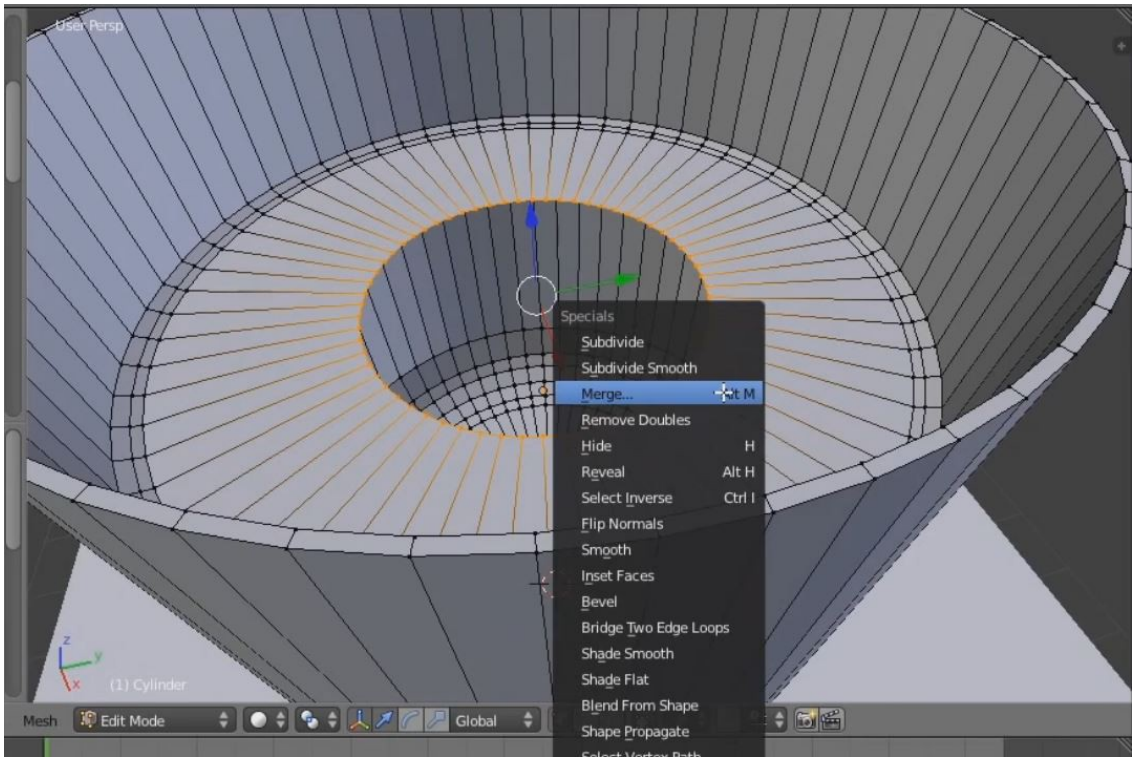
Il secondo Material sarà quello interno, a contatto col liquido: si tratta di un Glass con indice di rifrazione 1.15, Roughness 0.0 e il colore che preferite.

Per assegnare questo Material alla parte di bicchiere con il liquido, selezioniamo tali facce in Edit Mode, selezionando per primo il vertice in fondo, all'interno, e premendo CTRL + due o più volte per allargare la selezione.




*Assegnare il sottomateriale alla geometria del liquido*

Clicchiamo quindi su *Assign* nel pannello dei materiali per assegnare il secondo sottomateriale alla selezione. Selezioniamo ora l'anello che coincide con la superficie del liquido, premendo ALT e click del tasto destro su uno spigolo dell'anello per selezionarlo velocemente, quindi estrudiamo e scaliamo, trascinando un po' giù, un paio di volte, quindi estrudiamo e fondiamo al centro (con *Merge at Center*) per creare la superficie del liquido.



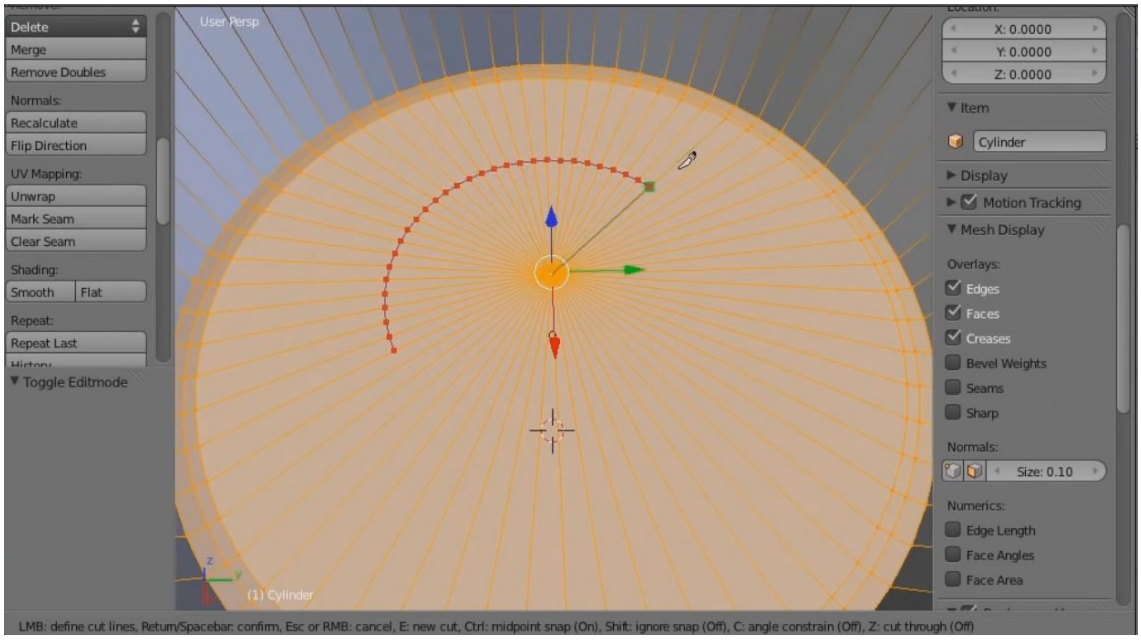
*La superficie del liquido*

Diamo un'occhiata alle normali: qualora dovessero esserci problemi, selezioniamo solo le facce appena create e clicchiamo su *Recalculate Normals* nella Tool Shelf, per farle puntare verso l'alto. Possiamo usare anche *Flip Normals*.

Nel pannello dei Materiali, creiamo un terzo sottomateriale, partendo da quello utilizzato per il liquore interno ma creando, da qui, un nuovo sottomateriale con "Add New Material"; dobbiamo cambiare, infatti, il valore di IOR, l'indice di rifrazione, portandolo a 1.3.

*Creazione del terzo sottomateriale*

Per agitare un po' la superficie del liquido devo creare un anello, cosa che adesso posso fare con la nuova versione del taglierino *Knife*, premendo *K*, poi CTRL per attivare lo *snap* mediano e (tenendo CTRL premuto) facendo click col tasto sinistro del mouse in modo da effettuare i tagli intorno all'oggetto, premendo infine Invio per confermare: come potete vedere (nella prossima pagina), ho creato un anello!




*La creazione degli "anelli" con Knife*

Creiamo quindi altri due o più anelli (cosa che possiamo fare, ora, o con Ring Cut, con CTRL R, o ancora col taglierino *knife*).

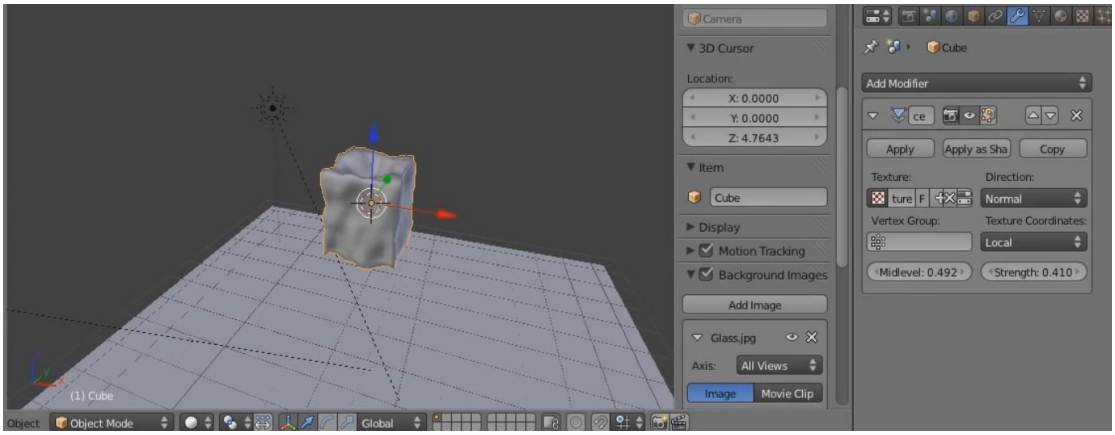
Selezioniamo quindi, in maniera casuale, due o più vertici, attiviamo il *Proportional Editing* (shortcut: O) e trasciniamo la selezione su o giù.

*Utilizzo del Proportional Editing*

Impostiamo lo Shading *Smooth* per il bicchiere e aggiungiamo, sempre per il bicchiere, un Modifier *Subsurf*, con 2 suddivisioni.

Ora dobbiamo creare i cubetti di ghiaccio.

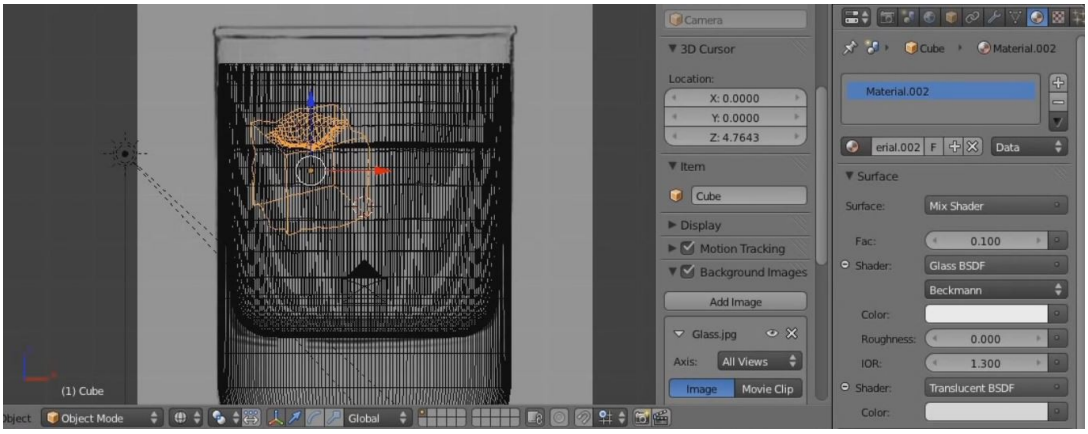
In realtà non è niente di complicato: dobbiamo creare un cubo, suddividerlo un buon numero di volte, passare all'ombreggiatura *Smooth*, aggiungere uno o due modificatori *Displace*, utilizzando Textures Clouds o Voronoi, variando i parametri fino a ottenere un risultato che ci soddisfa, per poi scalare ad esempio intorno a uno o due assi (per restringere un po' l'oggetto), creare fossette con il *Proportional Editing* e lo scaling, e così via... davvero, qui potete provare vari strumenti e vari modi fino a ottenere il risultato che più vi soddisfa!



*Modellazione del primo cubetto di ghiaccio*

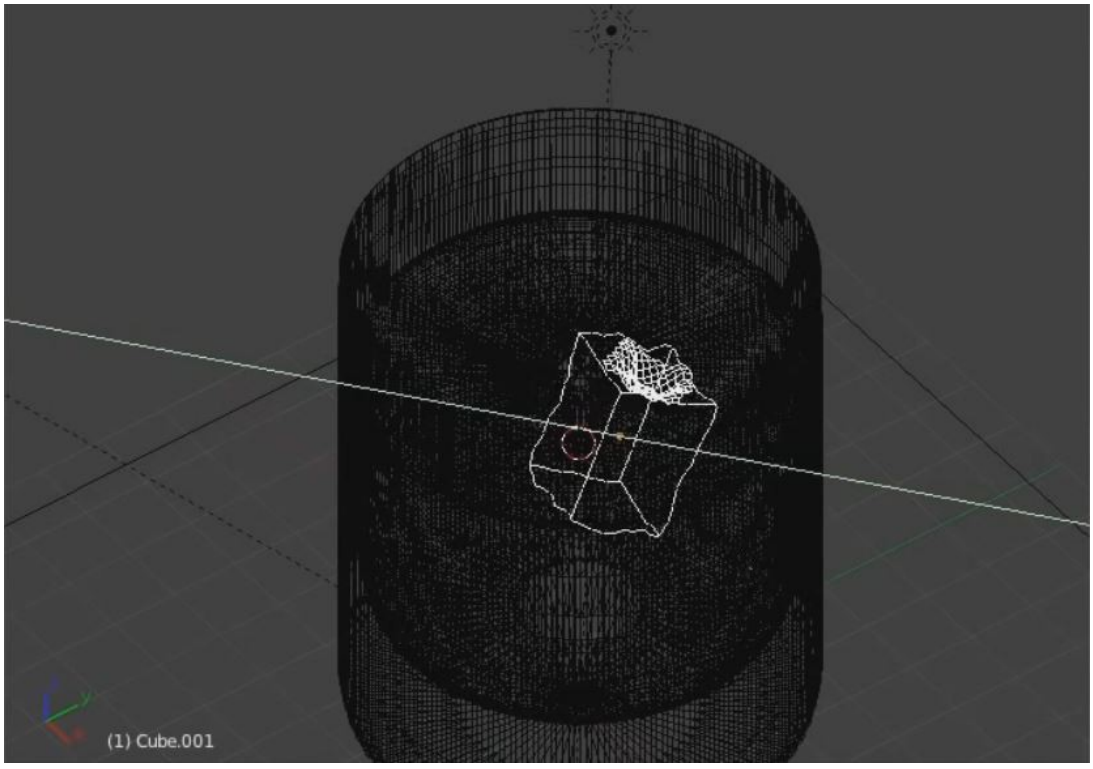
Aggiungiamo un Material per questo primo cubetto di ghiaccio; io sto usando un Nodo *Mix Shader* con Factor 0.1, con il primo Shader di tipo Glass con Roughness zero, colore bianco e indice di rifrazione 1.3, mentre il secondo Shader è di tipo Translucent con colore bianco puro.

Mentre ci siamo, impostiamo a 1 il Material Index per questo materiale: ci aiuterà a recuperare i cubetti di ghiaccio nell'Editor dei Nodi di Compositing, in Post Produzione; dobbiamo selezionare, poi, anche le voci "Material Index" e "Transmission Indirect" nella scheda Render, sezione Layers.



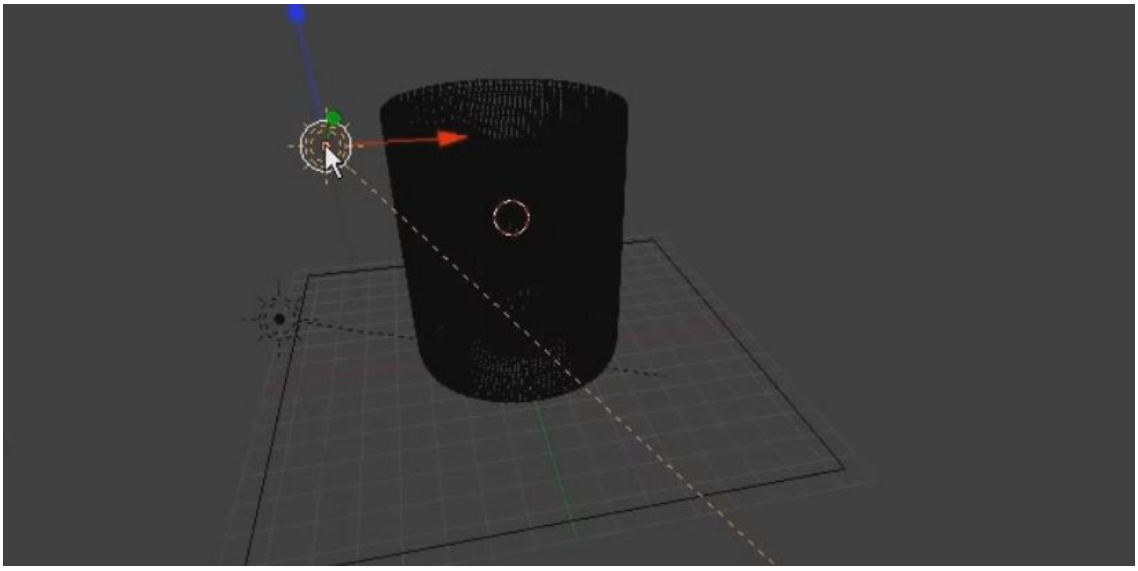

*Definizione del Material per il ghiaccio*

Creiamo ora quattro o più cubetti di ghiaccio duplicando il primo e modificando le nuove mesh, che avranno già il giusto Material e il giusto Material Index.



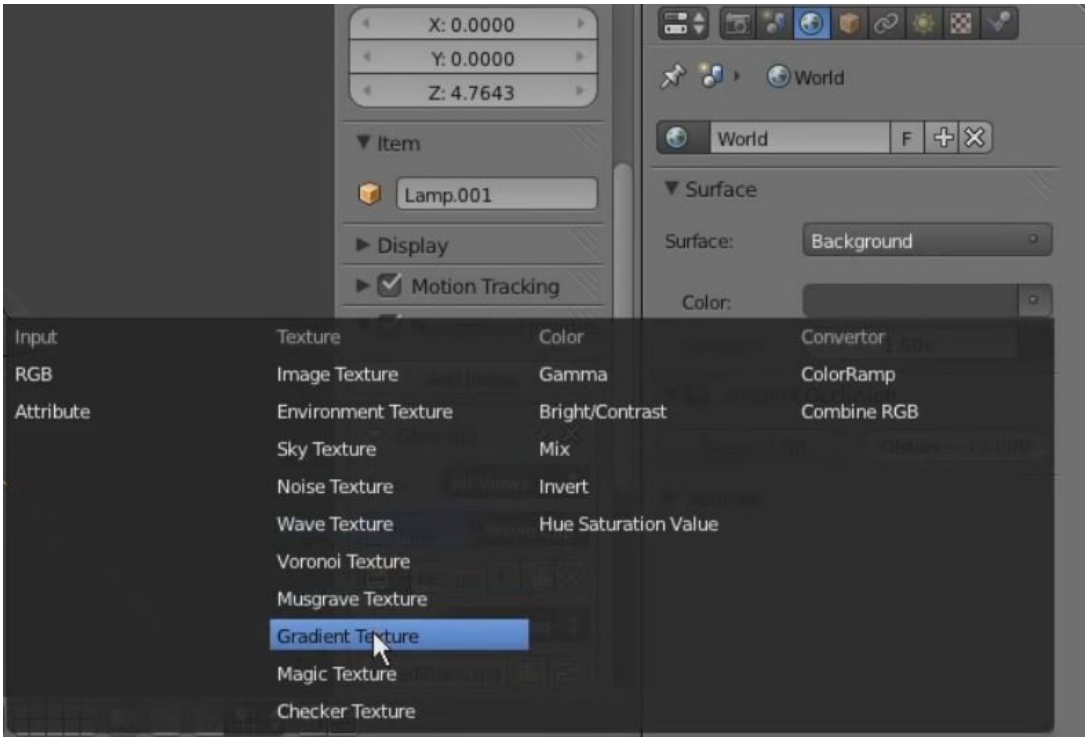

*Creazione e posizionamento degli altri cubetti*

Dobbiamo aggiungere anche delle fonti di luce, ma questa è un'operazione abbastanza semplice; possiamo aggiungere, ad esempio, due o più fonti di luce Sun e personalizzare i parametri come vogliamo. Per ottenere il rendering mostrato all'inizio, ho utilizzato proprio due Sun Lamps più Ambient Occlusion e World Background.



*Creazione e posizionamento delle fonti di luce*

Occupiamoci proprio del Background: per World, sto utilizzando una Texture Gradient con Strength 1.5 e Ambient Occlusion attivo; apriamo quindi una finestra Node Editor in modalità *Materials – World Shaders*.

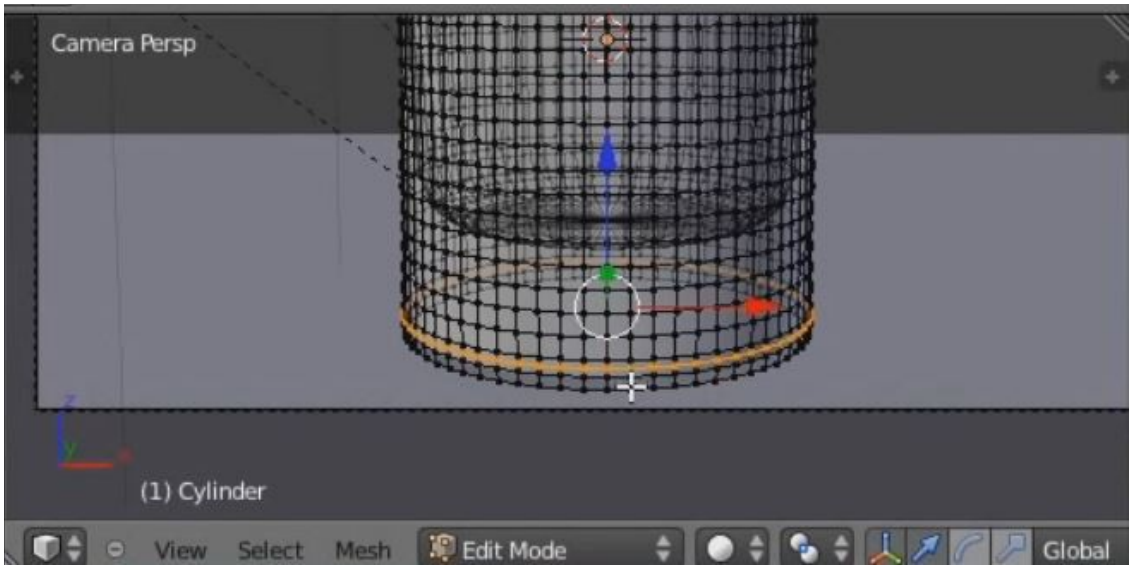



*Creazione della Texture "gradiente" per World*

Collego l'output Factor della Texture Gradiente all'input Color del Nodo Background, in modo da ottenere una texture in scala di grigi. Aggiungo un Nodo *Vector – Mapping*, collegandolo alla Texture Gradiente per ruotarla di 270 gradi intorno all'asse Z e per scalarla, ad esempio con valore 0.75, rispetto all'asse Y. Collegando infine l'output Camera View di un nodo di tipo *Input – Texture Coordinates* all'input Vector del nodo Mapping.

Passiamo alla modalità Compositing Nodes per sistemare la post-produzione in modo da evidenziare i riflessi luminosi sui cubetti di ghiaccio.

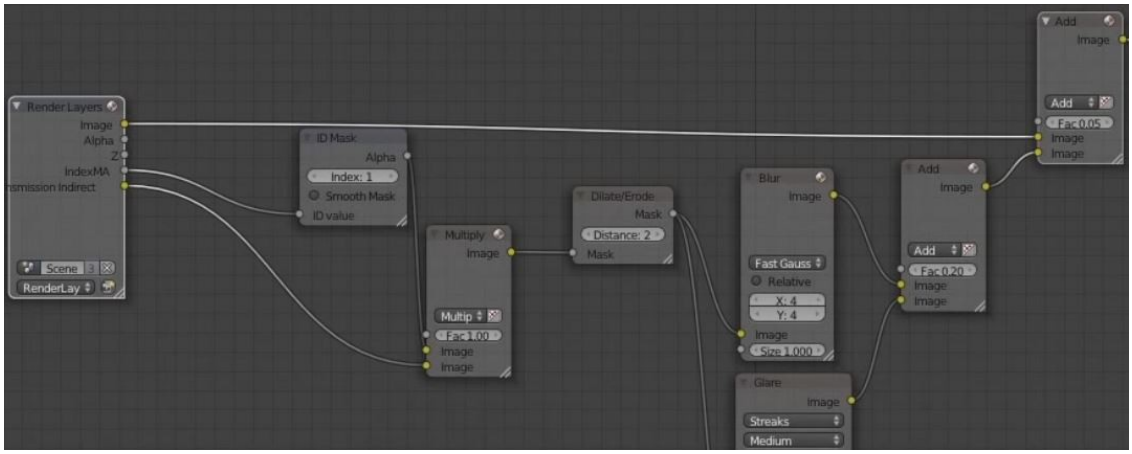
Dobbiamo aggiungere anche un paio di anelli alla mesh; se potete, prima di avviare il rendering finale, date un'occhiata alla mesh in modalità preview rendering, per gli ultimi ritocchi, come qualche Normale, altri anelli con Ring Cut, lo spessore del bicchiere, eccetera.




*Aggiungere degli anelli di spigoli alla mesh, dove necessario*

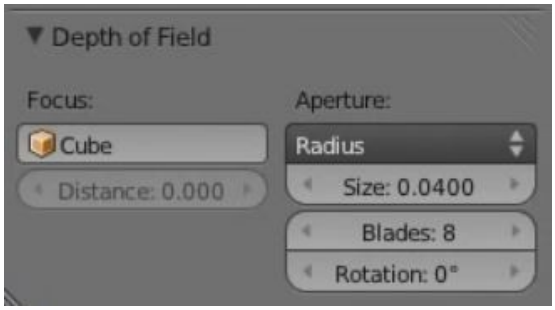
Aggiungiamo un Nodo *Convertor – ID Mask* con "pass" 1, collegandogli in input Material ID di Render Layers, e moltiplichiamolo mediante un Nodo *Color – Filter Node*, con Factor 1, con l'uscita Transmission Indirect di Render Layers. Allarghiamo un po' la maschera ottenuta in output dal nodo Multiply con un Nodo *Filter - Dilate/Erode* con valore 2 o più, collegando poi quest'output a due Nodi: un Filter *Glare* e un Filter *Blur.*

Le impostazioni per questi Nodi sono a piacere e, ovviamente, potrete cambiarle quando volete; in ogni caso, dovremo mixare gli output di questi due nodi con un Nodo *Color – Mix*, ad esempio in modalità Add, sommando poi questo risultato all'output Image di Render Layers utilizzando un altro Nodo *Mix*, in modalità Add (con Image sulla prima porta Color in Input e Factor molto basso, ad esempio 0.1 o meno).




*Schema di Nodi di Compositing*

Possiamo passare l'output a Composite per il rendering, ma prima di renderizzare, possiamo aggiungere anche un po' di Defocus selezionando l'oggetto Telecamera, aprendo il pannello Object Data e impostando un cubetto di ghiaccio nel campo "Focus Object" di "Depth of Field" (profondità di campo), con 8 o più Blades e un valore basso per Radius Size, ad esempio 0.03 (ma, anche qui, potete personalizzare questi valori come preferite).




*Defocus*

Possiamo anche impostare "Clamp", nel pannello Render, a 0.98 o 0.95, in modo da limitare le fireflies, i puntini bianchi di rumore che appaiono in alcuni rendering fatti con Cycles o altri motori unbiased. Non ci resta che avviare il rendering per osservare il risultato finale (visto nella prima immagine di questo capitolo) ed eventualmente ritoccare qualche parametro del Nodes Compositing.

*  *  *

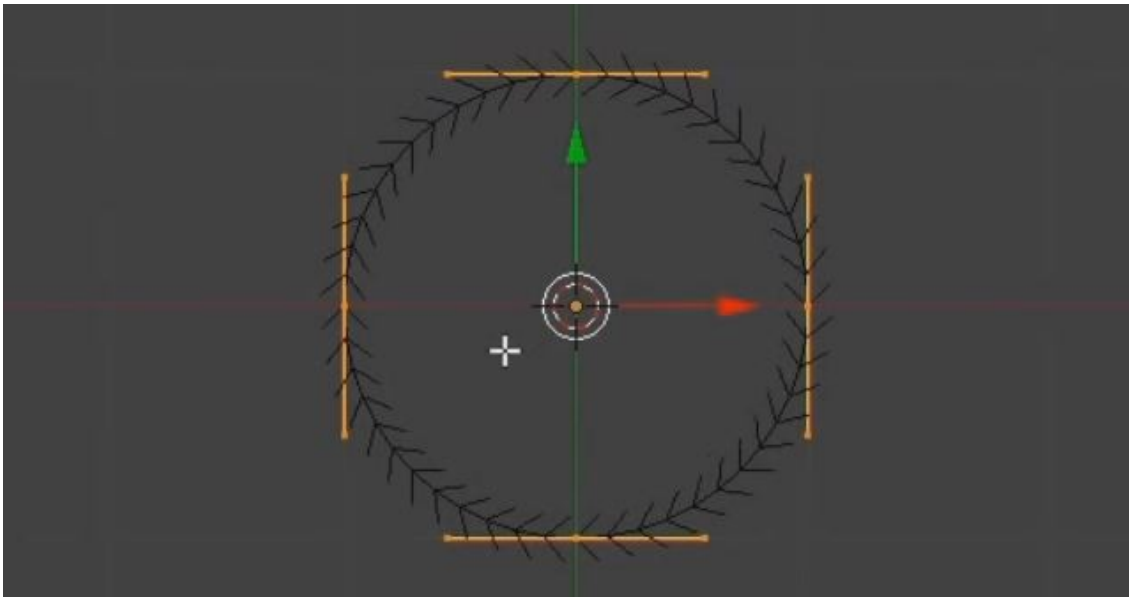
# Modellare l'elica di un aereo o di una nave con Spin e Twist

In questo breve tutorial vedremo un modo per realizzare le eliche di un aereo o di una nave e, in generale, pale per turbine, ventilatori e altri elementi simili in Blender 3D 2.63; esistono vari modi per ottenere questo risultato ma qui ne approfitteremo per parlare di curve, Bmesh, modificatore Twist in Simple Deform, di Spin Dup e soprattutto di come impostare il sistema di coordinate Locale degli oggetti per applicare correttamente i modificatori.

Iniziamo con l'elica di un aereo.

Partiamo da una scena vuota, massimizzando la vista Top (superiore) ortogonale.
Aggiungiamo un cerchio Bezier; di default, possiamo vedere questo cerchio frontalmente nella vista TOP (in questo caso, ORTHO).

Passiamo in Edit Mode: ci sono quattro punti di controllo.



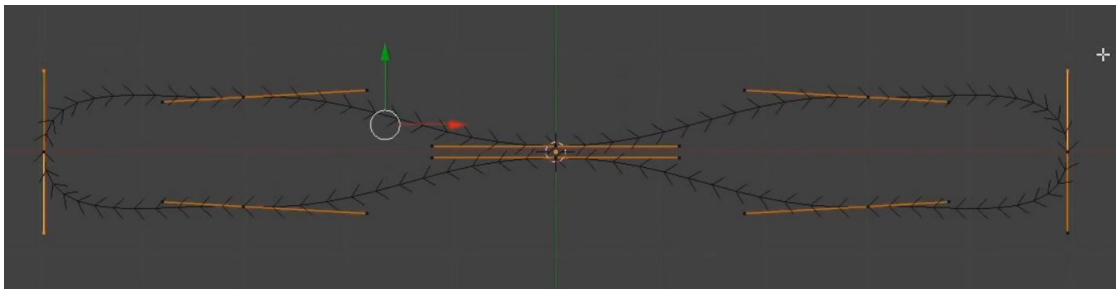
*Il cerchio Bezier, nella vista TOP ORTHO in Edit Mode*

Selezioniamoli tutti con A e premiamo W – *Subdivide* per aggiungerne altri quattro.

A questo punto dobbiamo definire la forma dell'elica "in piano", senza torsione;se avete un'immagine di riferimento, potete aggiungerla come Background Image per riprodurre una determinata elica.

Quando effettuate degli scaling, utilizzate X, Y o Z per vincolare l'operazione su un solo asse, ad esempio per allontanare tra loro due o più punti di controllo sull'asse X.

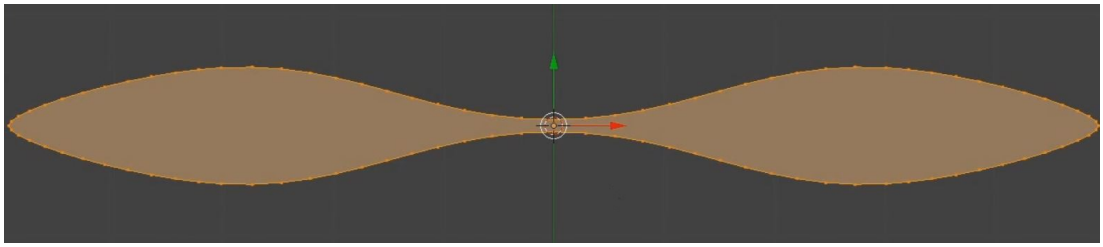
Per restringere le punte, non selezionate i punti di controllo ma le maniglie, scalando poi sull'asse Y.



*Definizione della sagoma della pala*

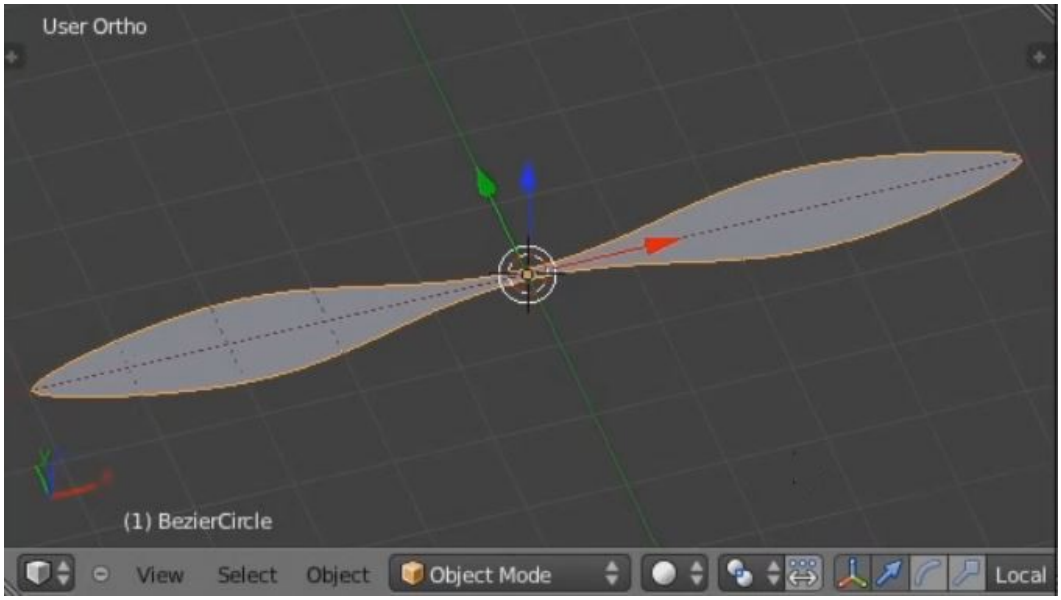
Non sto creando una curva 2D, piena, perché la conversione da curva 2D a mesh produce un brutto riempimento; con la curva 3D, quindi, in Object Mode, premiamo ALT C e scegliamo "*Mesh from Curve*".

Adesso passiamo in Edit Mode, selezioniamo tutto con A e premiamo F, per Fill (riempimento); in Blender 2.63, con la modellazione Bmesh, abbiamo ora solo una faccia di riempimento, che produce un buon risultato.



*La mesh riempita con F (fill)*

Adesso, per realizzare la torsione delle pale dobbiamo utilizzare un modificatore Simple Deform in modalità Twist, ma c'è un problema: tale modificatore funziona intorno all'asse Z locale, per cui in questo caso non funzionerà correttamente perché l'asse locale Z punta verso l'alto (coincide infatti con l'asse Z globale). Possiamo dare un'occhiata all'asse Z locale dell'oggetto selezionato passando a Local nel menù Orientation, nell'header della 3D View.

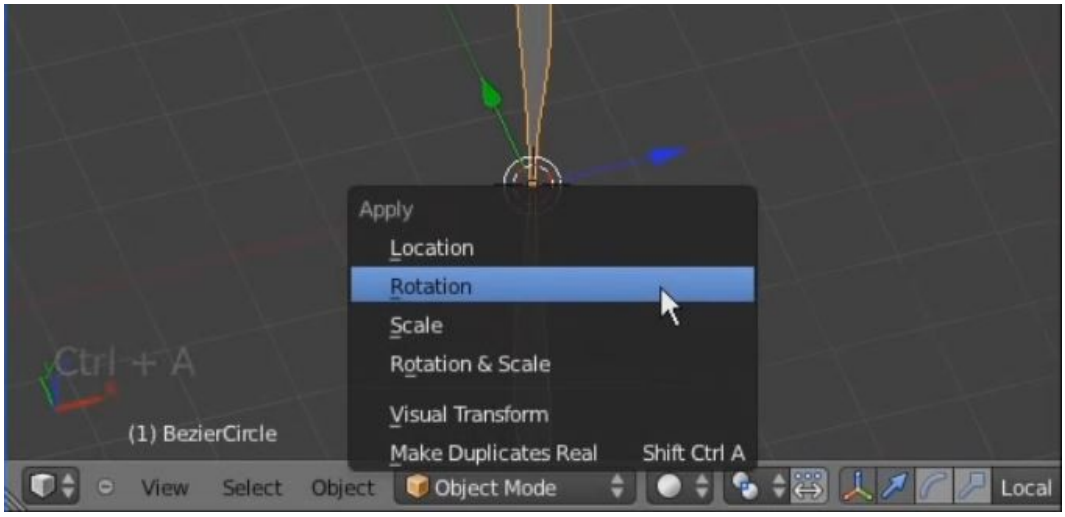



*L'asse Z locale*

Nel nostro caso, vogliamo effettuare la torsione intorno all'asse X, quindi dobbiamo impostare l'asse X come asse Z locale. Ruotiamo l'elica di 90 gradi intorno all'asse Y (digitando R Y 90 e Invio). Diamo un'occhiata all'asse in modalità Local Orientation.

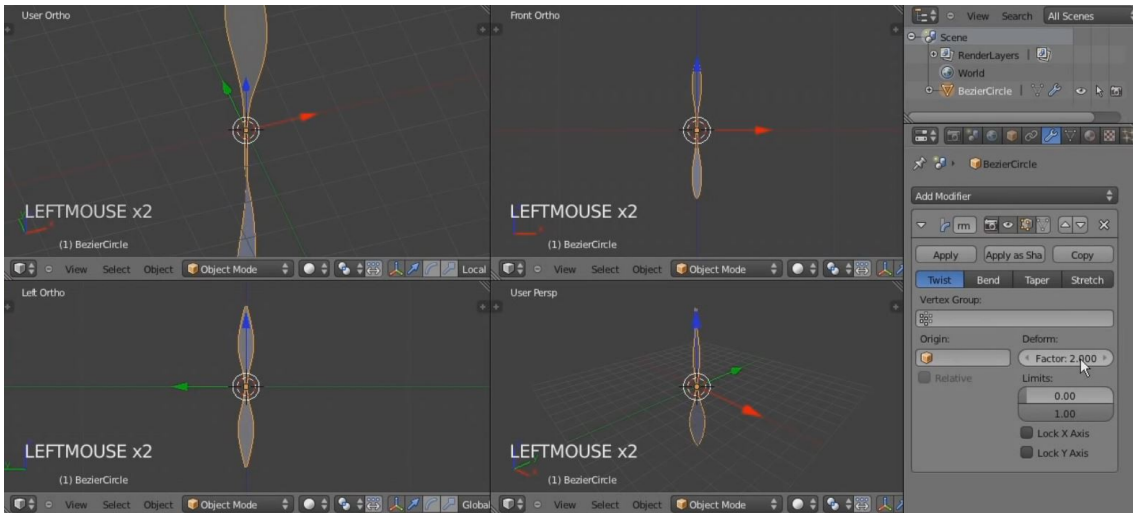
Adesso applichiamo la rotazione con CTRL A e scegliendo “Rotation” dal menù che apparirà.

L'asse Z locale punta ora verso l'alto.




*Applicazione della rotazione*

Possiamo quindi aggiungere il modificatore Simple Deform in modalità Twist alla mesh e impostare il Factor che preferiamo, per esempio 2 o più... ed ecco l'elica per il nostro aereo!



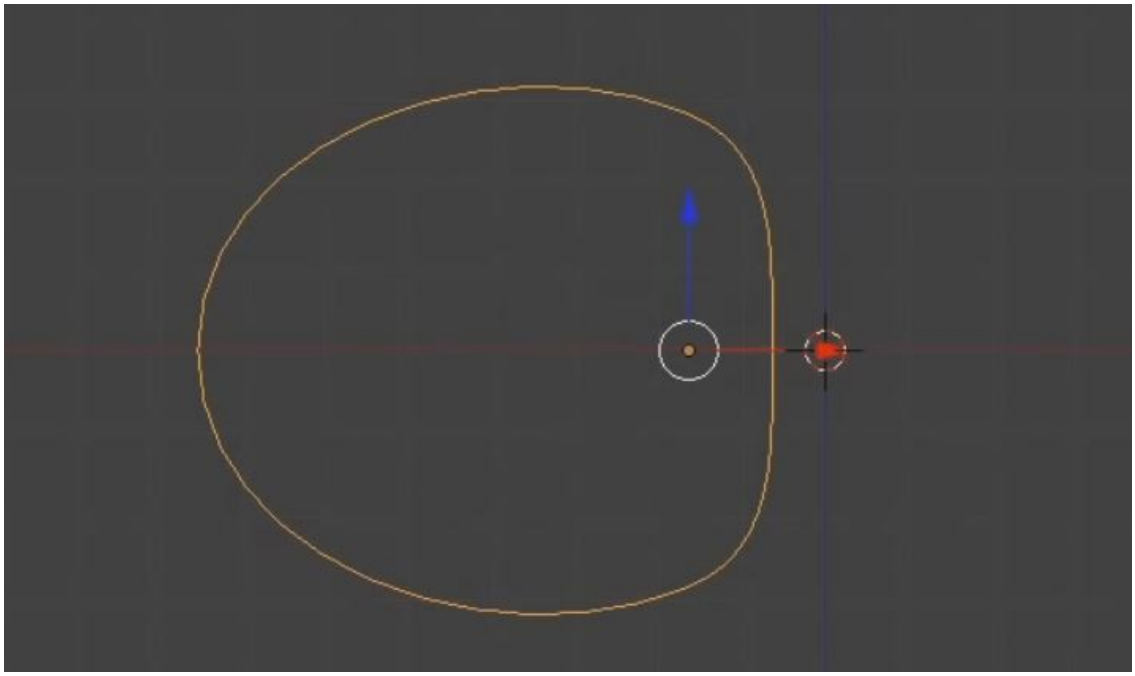
*Simple Deform - Twist*

*  *  *

L'elica navale è più interessante perché in questo caso dobbiamo definire tre o più pale, quindi dobbiamo utilizzare Spin Dup per creare le pale attorno al giusto Pivot Point e applicare Twist di Simple Deform intorno al giusto asse.

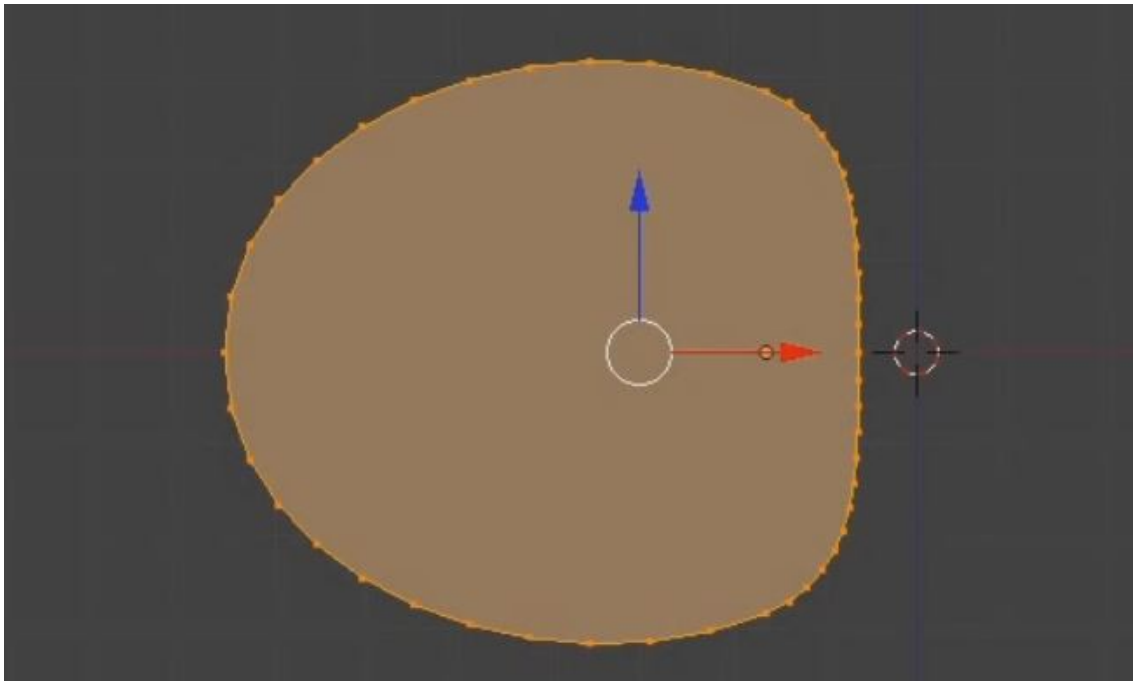
Cancelliamo tutto e partiamo da una scena vuota, con il 3D Cursor al centro (con SHIFT S e *Cursor to Center* dal menù Snap).

Andiamo nella vista Front Ortho, aggiungiamo un cerchio Bezier, ruotiamolo di 90 gradi intorno all'asse X, quindi muoviamolo verso sinistra (non vogliamo infatti mantenerlo al centro della scena... vedrete presto perché); passiamo quindi in Edit Mode e muoviamo i punti di controllo per definire la forma della prima pala.



*Definizione della prima pala*

Torniamo in Object Mode, convertiamo questo oggetto Curve in Mesh e riempiamolo con *F* (Fill) in Edit Mode, come abbiamo fatto per la prima elica.

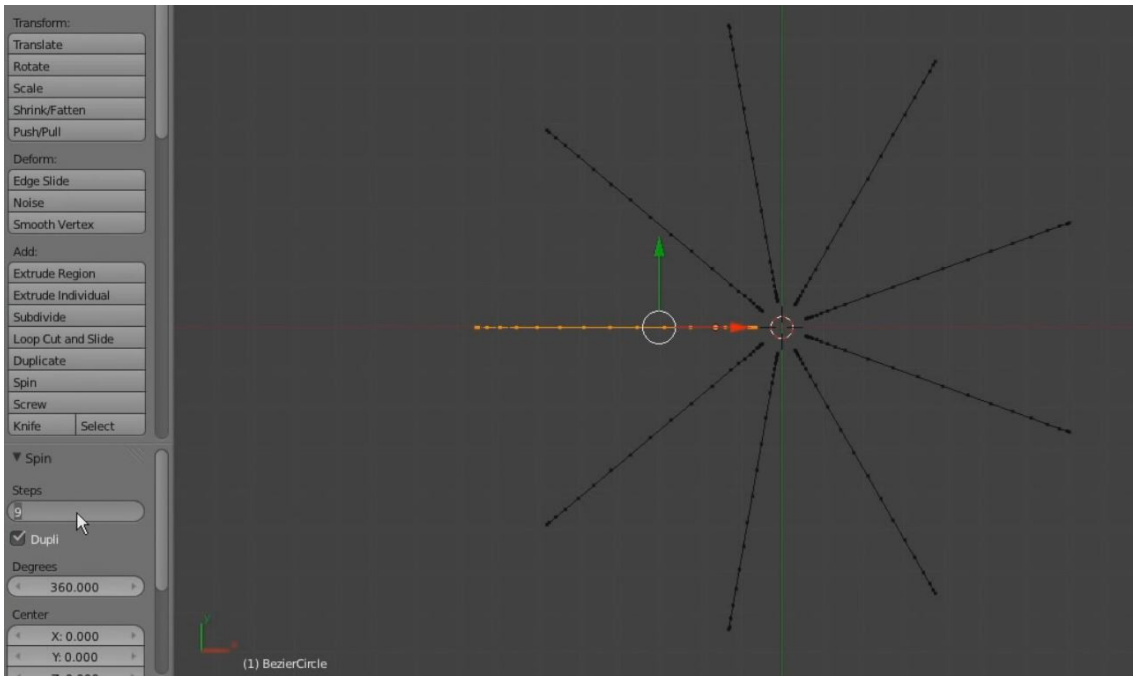


*La mesh riempita*

Per creare le pale, dobbiamo impostare il pivot point per lo strumento Spin; utilizzeremo il centro della scena, quindi spostiamo lì il 3D Cursor.

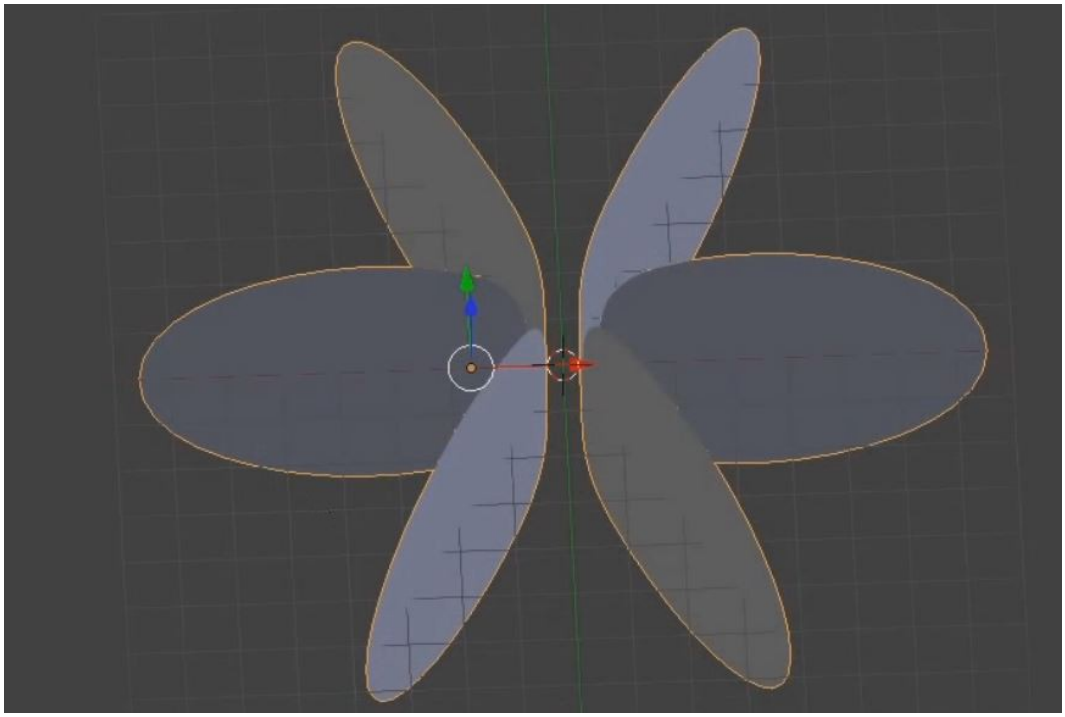
Dobbiamo poi passare in Edit Mode, selezionare tutto (con A) e spostarci nella vista Top Ortho, e questo perché l'asse di rotazione deve essere perpendicolare al piano di osservazione.

Adesso clicchiamo su Spin, nella Tool Shelf: Spin effettua una rotoduplicazione con estrusione, quindi dobbiamo selezionare la casella Dupli per ottenere copie separate, impostare 360 gradi e tre o più *steps* (dipende da quante pale vogliamo), quindi premiamo X o CANC per cancellare i vertici della copia superflua.



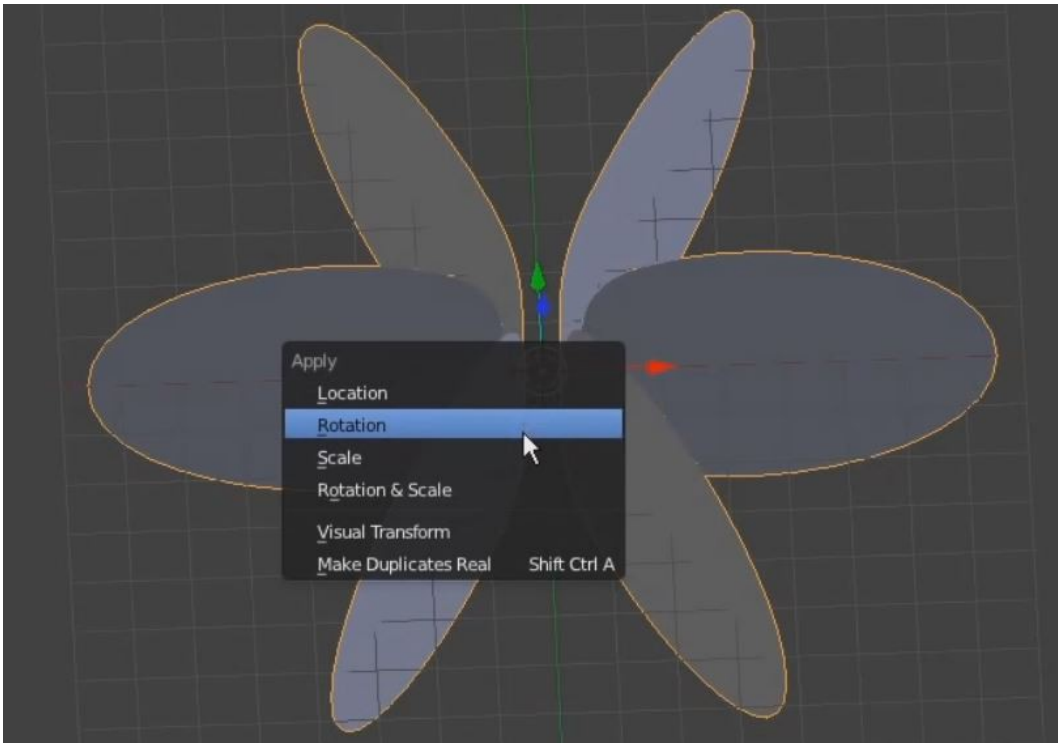
*Spin Dupli*

Possiamo tornare in Object Mode ma non possiamo ancora aggiungere il modificatore Simple Deform Twist perché dobbiamo ruotare il sistema di coordinate locali e spostare l'Origine dell'oggetto (e, di conseguenza, l'Origine del modificatore) perché adesso, come potete vedere nell'immagine seguente, l'Origin si trova all'interno della prima pala.



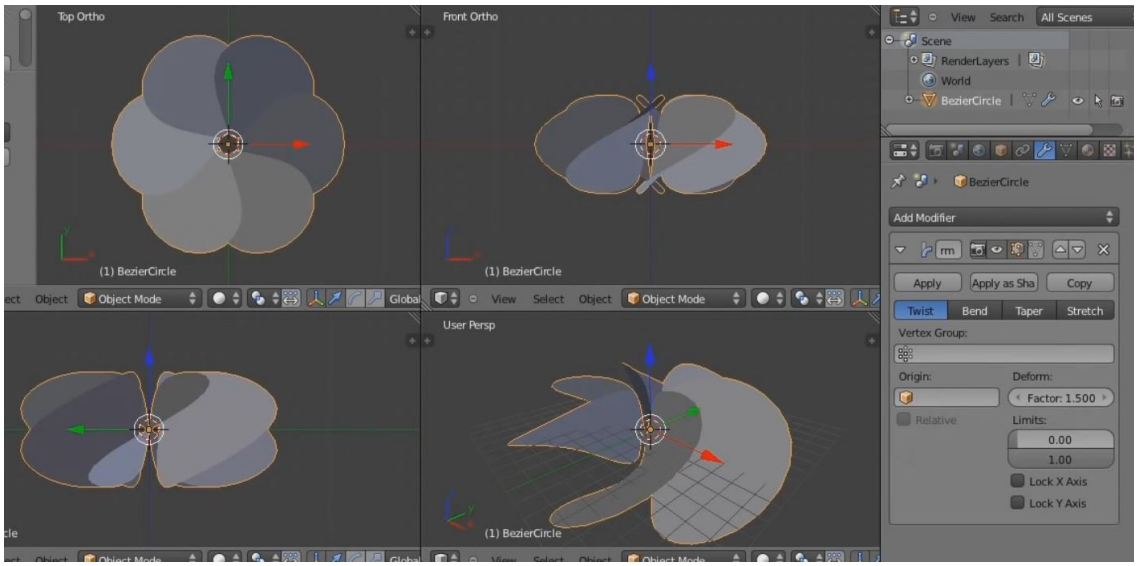
*L'Origin si trova all'interno della prima pala*

In Object Mode, clicchiamo quindi su *Origin – Origin to Geometry*, nella Tool Shelf. Premiamo ora CTRL A e clicchiamo su *Rotation*.




*Applicazione della rotazione, con l'origine re-impostata*

A questo punto possiamo aggiungere il modificatore *Simple Deform – Twist* e impostare il Factor che preferiamo... ed ecco la nostra elica navale!




*Simple Deform - Twist*

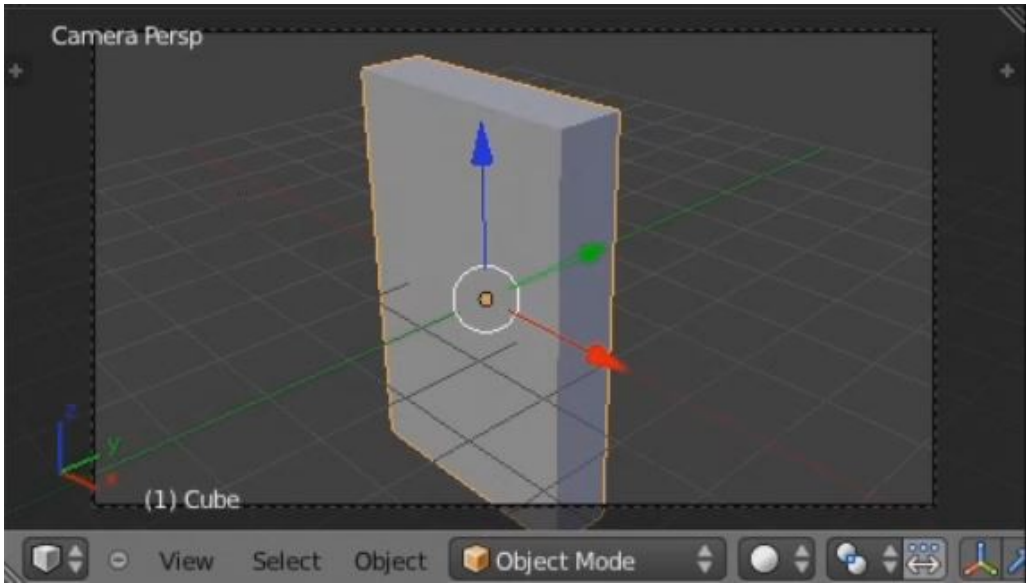
*  *  *

# Unwrap e UV Mapping (e Texturing) di una scatola in Cycles

Questo è un tutorial di base sul Texturing con UV Mapping e rendering in Blender 3D e Cycles.
Il tutorial, realizzato con la versione 2.63 di Blender 3D, nasce da una richiesta di un utente che chiedeva come mappare un'immagine su una scatola in Blender 3D per realizzare l'immagine di una confezione (in particolare, la scatola di un antivirus), ma va bene per molti prodotti da rappresentare in opuscoli pubblicitari, cataloghi, volantini, eccetera.

In questo tutorial vedremo anche come sfruttare il World illuminante di Cycles per impostare l'illuminazione per il rendering.

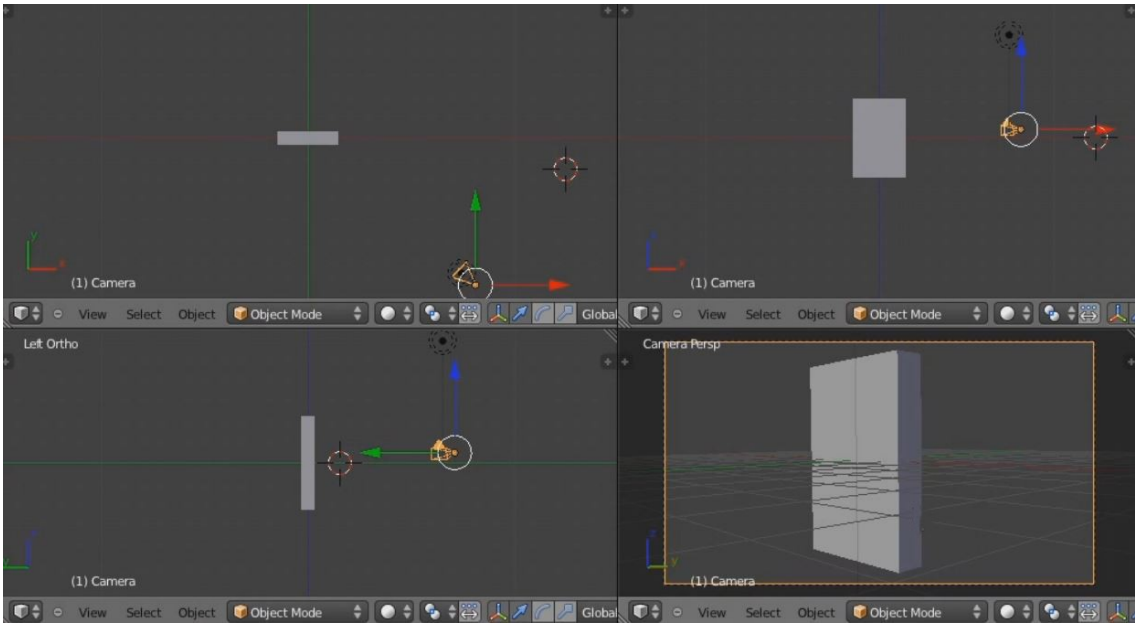
Per la scatola utilizzeremo, ovviamente, il cubo di default: dobbiamo semplicemente scalarlo su uno o più assi premendo S e, poi, due volte la lettera dell'asse, ad esempio X X, in modo da specificare che vogliamo effettuare l'operazione sull'asse X locale, proprio dell'oggetto.
Premiamo quindi S X X, muoviamo il mouse per ridimensionare l'oggetto e premiamo Invio per confermare, poi facciamo le stesse operazioni sulla profondità (S Y Y) e sull'altezza (S Z Z).



*Il cubo ridimensionato (dopo alcuni Scaling sui vari assi locali)*

Se abbiamo le misure reali dell'oggetto e vogliamo seguire quelle proporzioni, possiamo aprire la Transform Window premendo *N* mentre il mouse è in una finestra di navigazione 3D, e, con il box selezionato, digitare i valori in *Dimensions X Y Z*.

Abbassiamo e ruotiamo la telecamera con G per la traslazione e R, sui vari assi, per la rotazione, quindi ad esempio R X X o R Z, oppure utilizzando *Location* e *Rotation* nella Transform Window.

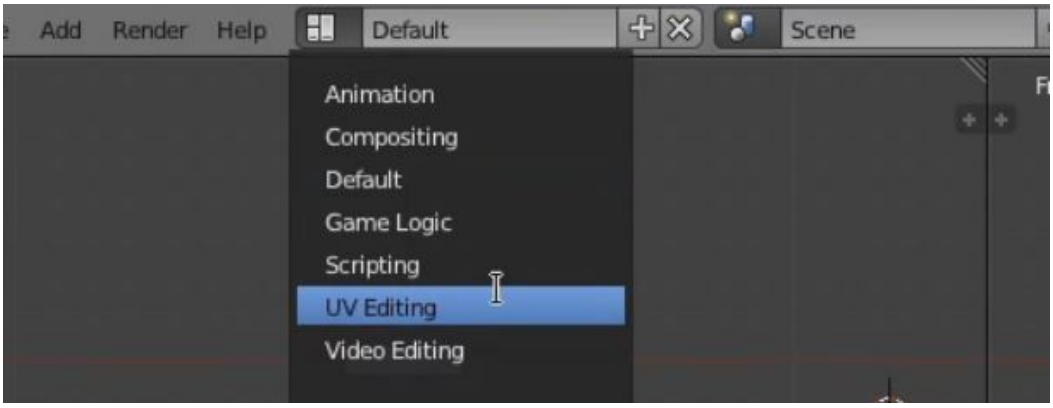



*Posizionamento della telecamera*

Parleremo dell'illuminazione in seguito: per il momento, selezioniamo il box e occupiamoci del texturing.

Prima di iniziare, una precisazione: per mappare un'immagine su un oggetto dobbiamo procedere, per così dire, su due fronti, ossia sulla scucitura della mesh (detta unwrapping) con mappatura delle facce della mesh sull'immagine (operazione di UV Mapping) e, in seguito, l'applicazione vera e propria dell'immagine con le informazioni di scucitura come texture del materiale dell'oggetto (fase di Texturing).

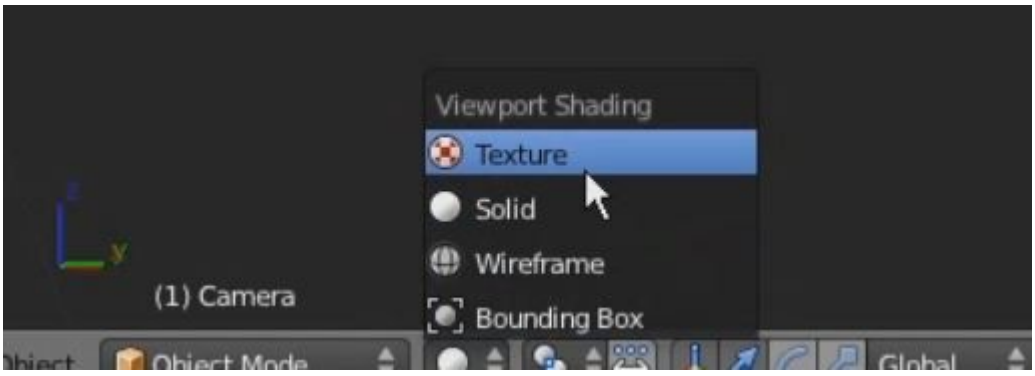
Cambiamo il layout della schermata di Blender in *UV – Editing* dal selettore Screen Layout e iniziamo con la prima fase: l'unwrapping.




*Screen Layout UV Editing*

Nella finestra a sinistra (la finestra "UV – Editor delle immagini") clicchiamo su *Image – Open Image* e carichiamo da disco l'immagine che rappresenta, in piano, i vari lati della scatola.

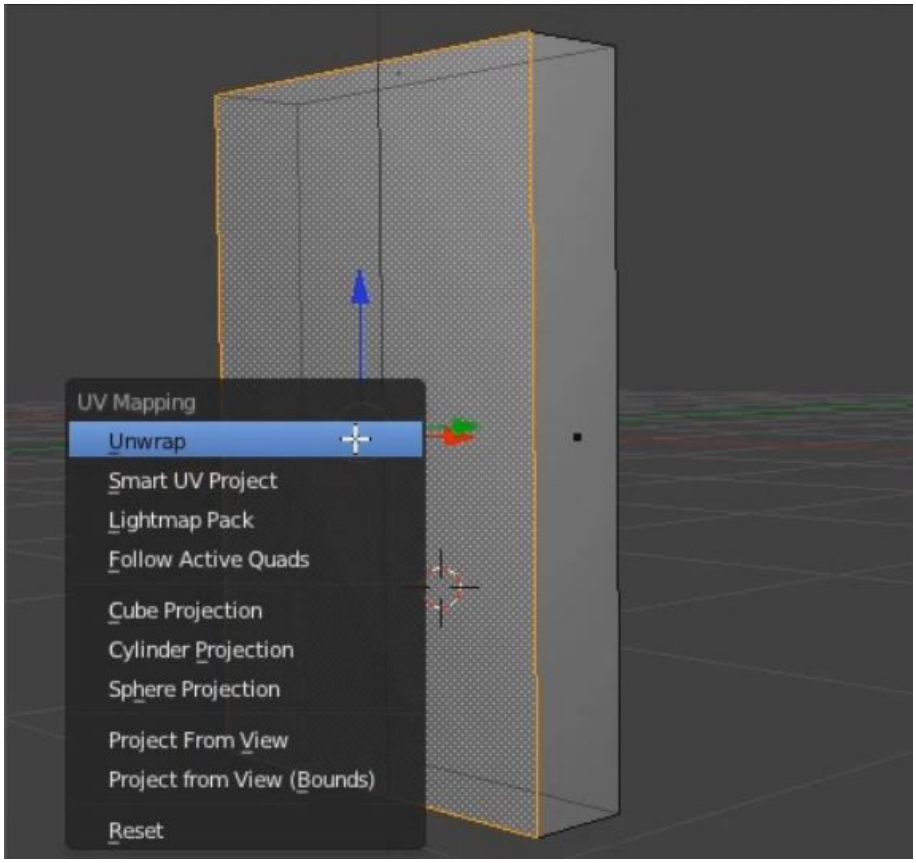
Spostiamoci ora nella finestra a destra (la 3D View), cambiamo il punto di vista in telecamera con *View – Camera*, in basso, e, per avere un'anteprima della scucitura, cambiamo la modalità di ombreggiatura da SOLID a TEXTURE con il selettore *Viewport Shading*, nella 3D View.




*Viewport Shading - Textured, nell'header della 3D View*

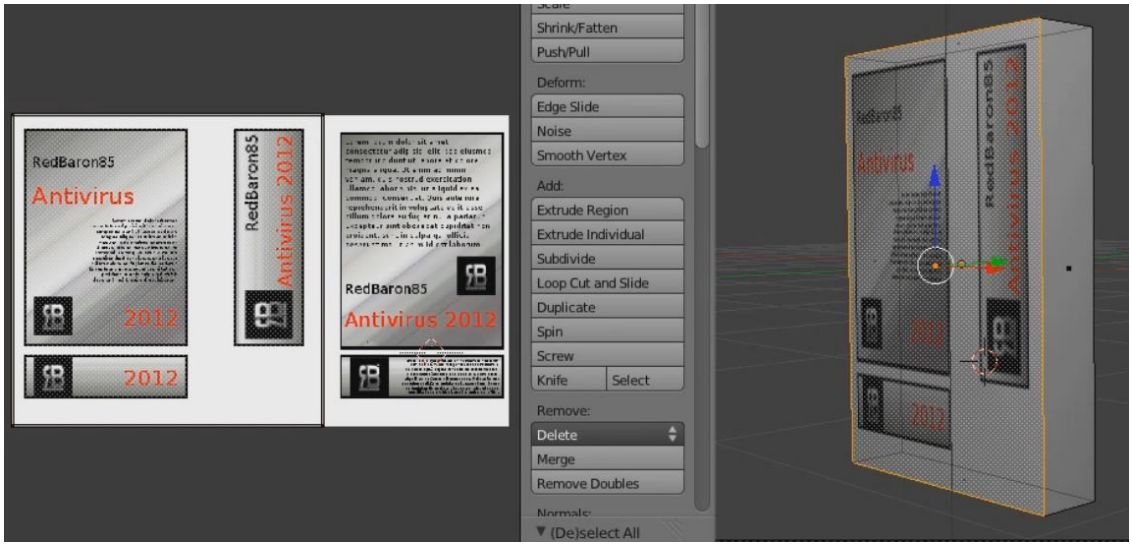
Con il box selezionato, passiamo in Edit Mode e in modalità selezione facce.

Dal momento che questo oggetto è composto da appena 6 facce (e in realtà non ci interessano tutte, per questo rendering), procederemo alla mappatura dei singoli elementi senza scucitura iniziale generale. Facciamo click col tasto destro sulla faccia frontale, per selezionarla, e premiamo U, scegliendo *Unwrap* nel menù che apparirà.



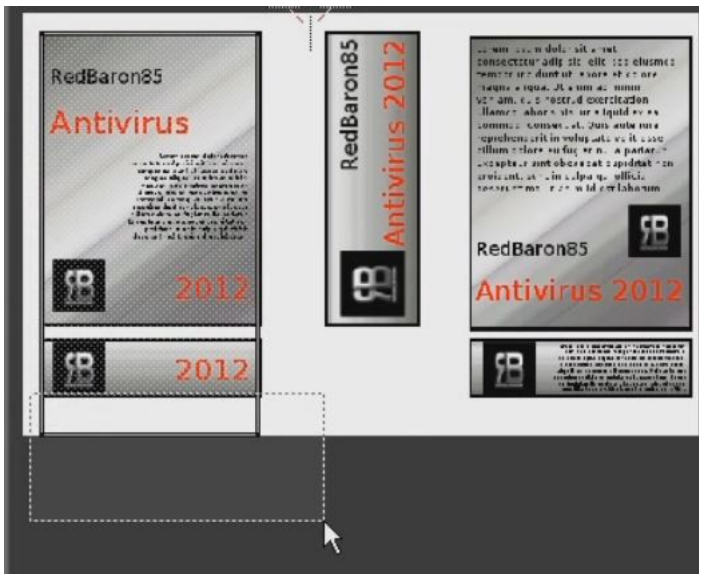

*U - Unwrap su una faccia selezionata*

A sinistra, vedremo un rettangolo sull'immagine: si tratta della faccia selezionata a destra, scucita sull'immagine a sinistra (e, visto che a destra siamo in modalità Textured, abbiamo un'anteprima in tempo reale della mappatura).




*L'effetto della scucitura*

I vertici che, a sinistra, rappresentano la faccia selezionata dell'oggetto, possono essere selezionati, traslati, ruotati e scalati come vertici di una mesh, quindi ad esempio possiamo mettere il cursore del mouse nella finestra a sinistra, premere A per deselezionare tutto, quindi premere B e fare una selezione dei vertici di destra con click del tasto destro del mouse, per poi premere G X (cioè Grab, "trascina", su X, l'asse orizzontale), muovere il mouse per spostarli e fare click col tasto sinistro per confermare.



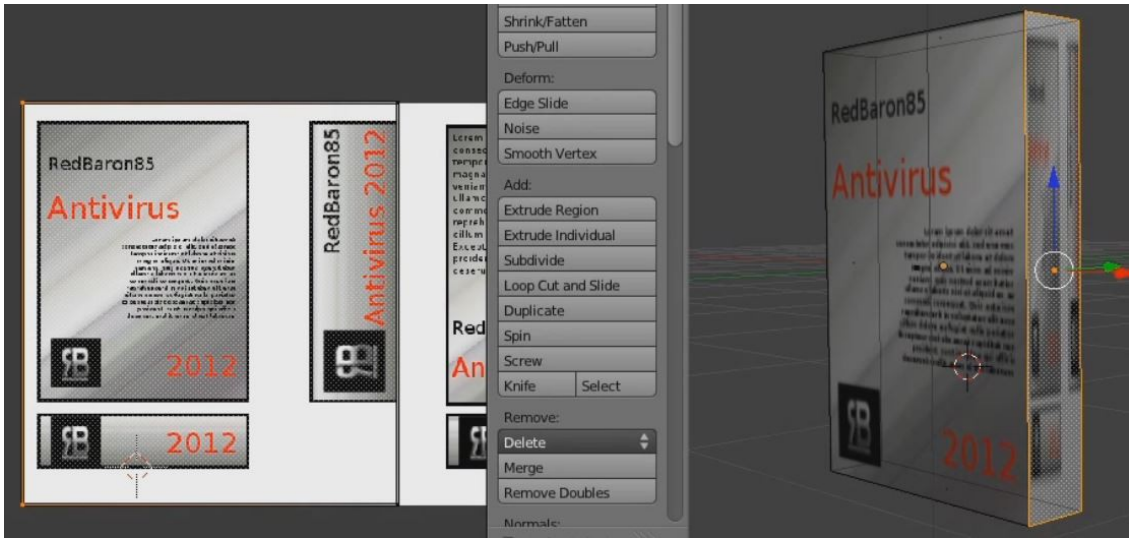
*Posizionamento dei vertici della scucitura*

Deselezioniamo tutto con A e spostiamo le altre coppie di vertici (con B e poi G X o G Y, a seconda dell'asse) in modo da "inquadrare" l'immagine.

Ruotando la rotellina del mouse possiamo zoomare sulla finestra UV, mentre cliccando il tasto centrale e trascinando potremo fare il panning, per vedere meglio certi punti.

Può capitare di dover ruotare la scucitura: in questo caso, con i vertici da ruotare selezionati (premere A per selezionare o deselezionare tutto) premiamo ad esempio R 90 Invio per ruotare la selezione di 90 gradi.

A questo punto dobbiamo fare la scucitura delle altre facce, così come fatto per la prima: facciamo quindi click destro sulla faccia laterale, premiamo *U*, scegliamo *Unwrap* e, nella finestra a sinistra, facciamo l'UV Mapping come fatto con la prima faccia.

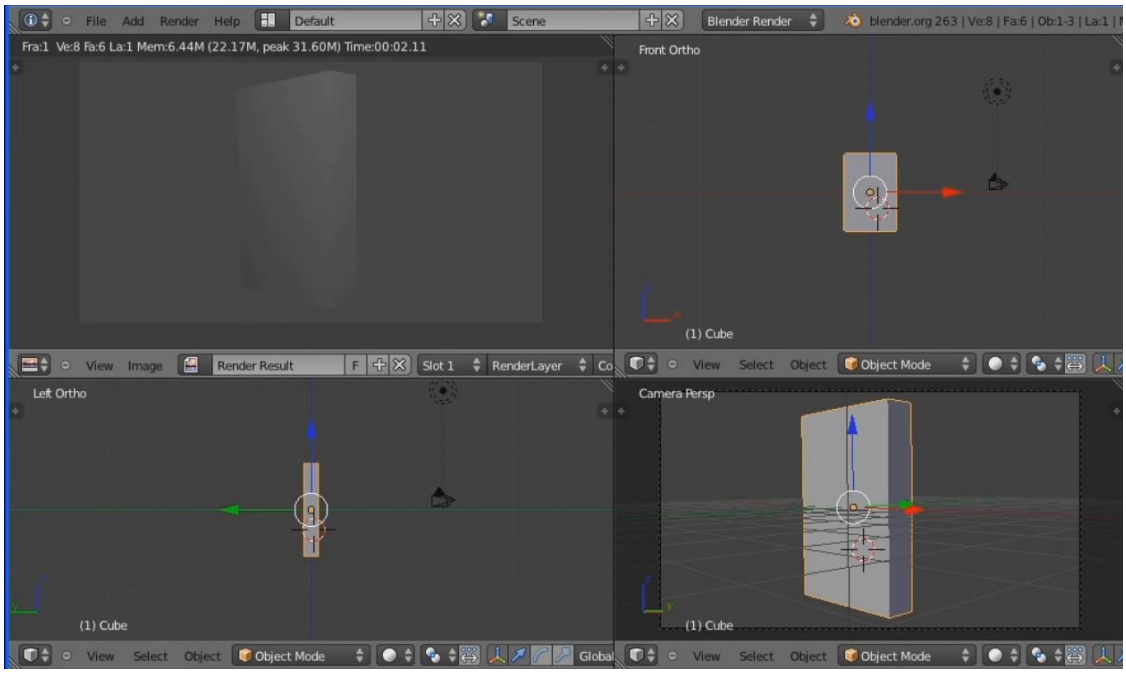



*Scucitura e sistemazione delle altre facce*

Queste operazioni vanno fatte anche per le altre facce ma visto che nell'inquadratura si vedono solo queste due mi sto fermando qui.

Con la prima fase (Unwrapping e UV Mapping) abbiamo quindi finito.

Torniamo alla Screen *Default* e in modalità Object Mode.

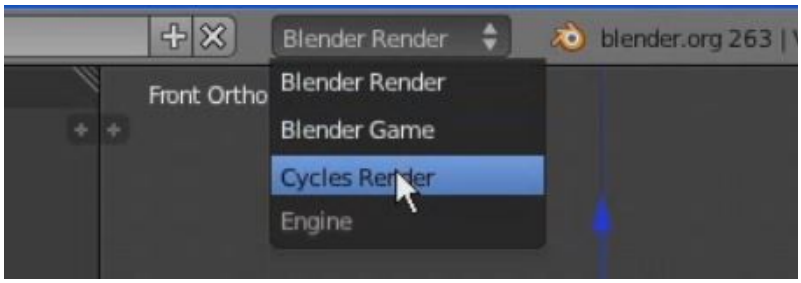



*Screen Default in Object Mode*

Premendo *F12* per un rendering veloce, comunque, avremo una brutta sorpresa: il box è grigio.

Abbiamo realizzato, infatti, la scucitura della mesh, ma non abbiamo associato all'oggetto una Texture immagine con quelle informazioni di UV Mapping.

Passiamo quindi alla seconda fase: il Texturing.
Dal selettore Engine, in alto, scegliamo Cycles.




*Selezione del motore Cycles*

Nella 3D View, con vista telecamera, passiamo alla modalità Rendered per avere un'anteprima in tempo reale in bassa qualità. Con il box selezionato, apriamo la scheda Material, nella Properties Window, quindi clicchiamo su *Use Nodes*.
Lasciamo lo Shader Diffuse di default con Roughness 0.0 e facciamo click sul quadratino col cerchio accanto alla Color Box, scegliendo *Image Texture*.

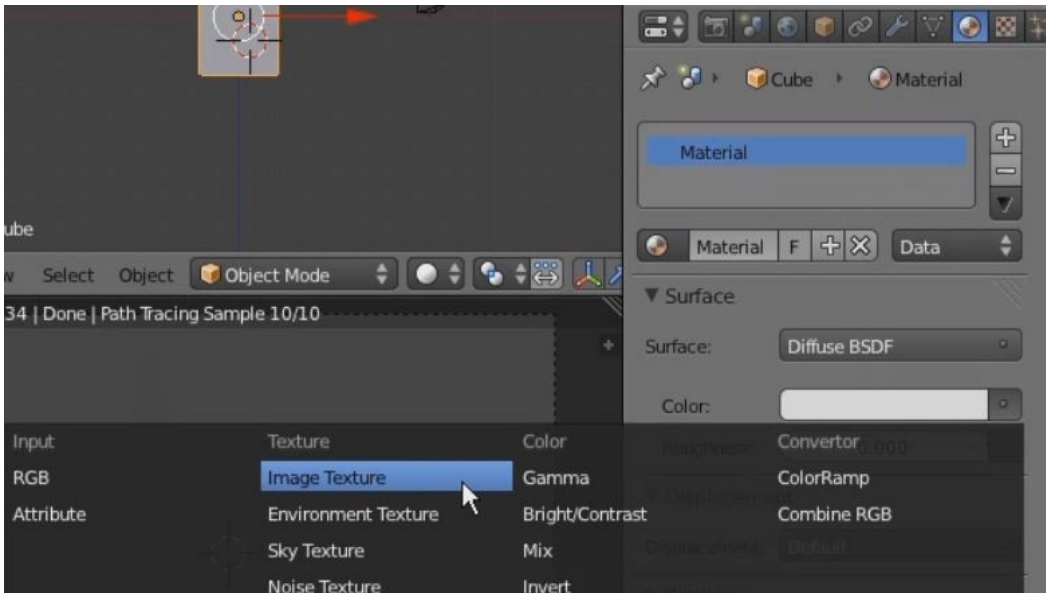
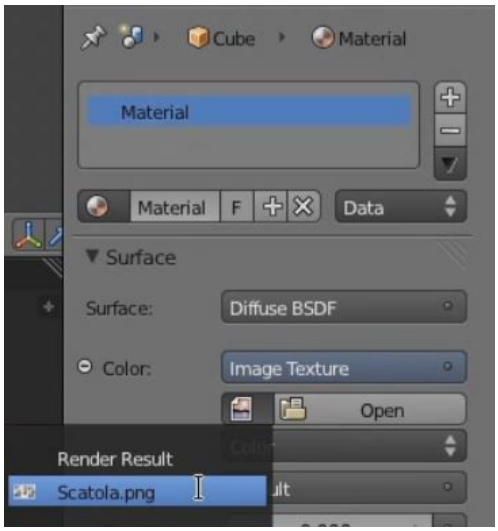


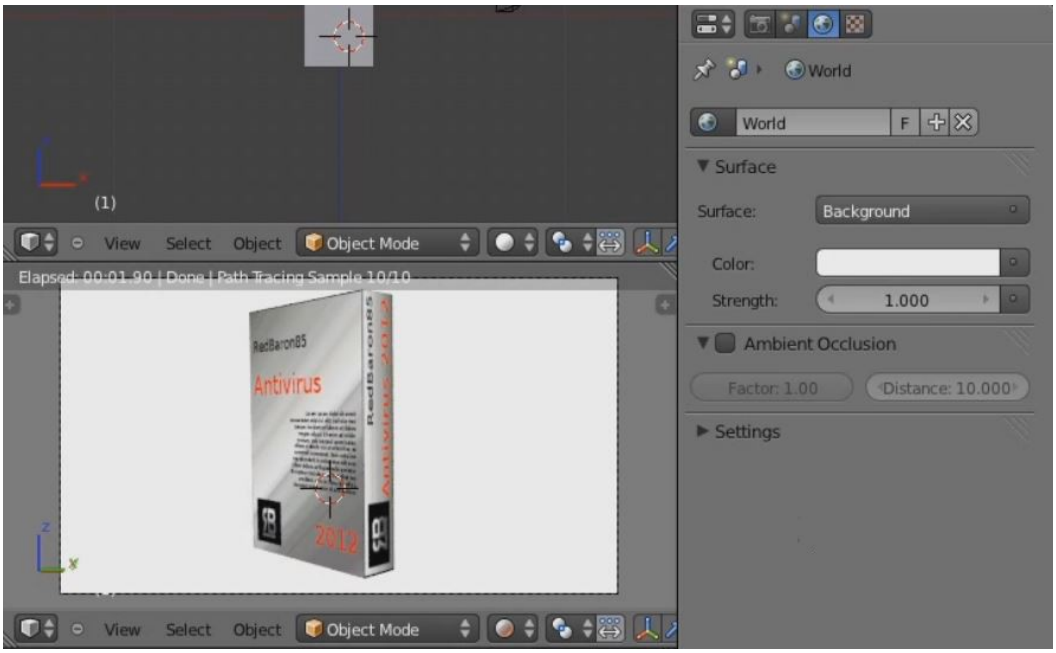

*Image Texture per lo Shader DIFFUSE*

L'immagine che ci serve è, nel mio caso, già in memoria, quindi clicchiamo sull'icona *Browse Image* e scegliamo l'immagine usata per l'Unwrap.
In Cycles, la mappatura dei file immagine come Textures avviene di default con modalità UV, quindi non dobbiamo fare altro con la scatola e possiamo passare a illuminazione e rendering.



*Selezione dell'immagine da utilizzare*

Selezioniamo la fonte di luce Lamp presente di default e... cancelliamola! Apriamo quindi la scheda World, nella Properties Window, attiviamo i nodi e cambiamo il colore in bianco puro, quindi variamo la Strength per regolare l'intensità dell'illuminazione, osservando i risultati in tempo reale nella 3D View *Rendered*.

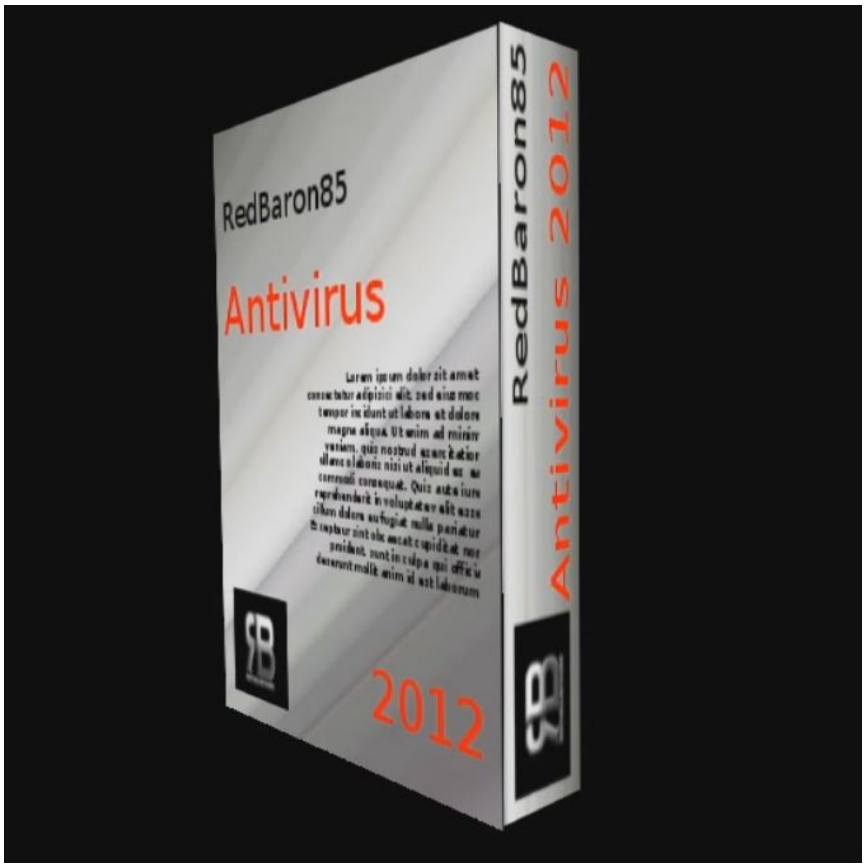


*Impostazione dell'illuminazione WORLD*

Prima di avviare il rendering, andiamo nella scheda Render e selezioniamo *Film – Transparent* e *PNG – RGBA*, cioè immagine con trasparenza (*A* per Alpha) di sfondo: in questo modo, l'immagine del prodotto sarà già scontornata e potremo utilizzarla facilmente in un software di fotoritocco (ad esempio per impostare lo sfondo, aggiungere bordi o ombre) o di desktop publishing, per l'impaginazione.

Impostiamo *Resolution* X e Y a seconda delle nostre esigenze. Dal momento che c'è solo un box con Shader Diffuse, saranno sufficienti pochi Render Samples in Integrator.

Non ci resta che premere *F12* per renderizzare e, terminata l'operazione, salvare tutto con F3.



*Il risultato finale*

Con questa procedura è possibile renderizzare molti oggetti per un catalogo o un volantino, anche se per oggetti riflettenti o cromati è meglio utilizzare immagini HDR in background... ma questo è un altro discorso! Per questo tutorial, invece, è tutto!

*  *  *

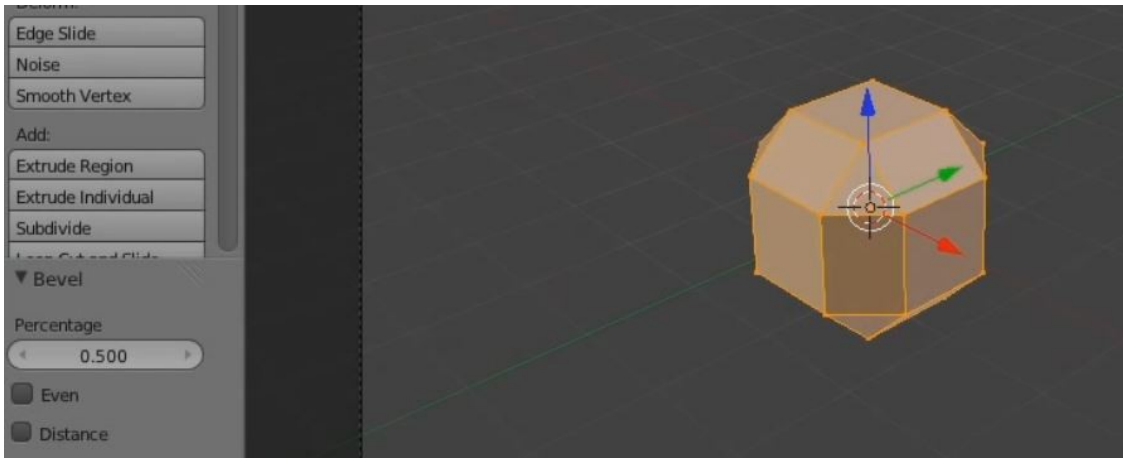
# Un dado a sei facce in Blender 3D e Cycles

Questo breve tutorial riguarda la modellazione di un dado a sei facce in Blender 3D 2.63. Utilizzeremo Bevel in Edit Mode, Ring Cut e i materiali multipli per il rendering con Cycles.

Come mesh, il cubo di default andrà benissimo.

Per prima cosa, passiamo in Edit Mode e, con tutti i vertici selezionati, apriamo la Tool Shelf con T, premiamo W e scegliamo Bevel dal menù che apparirà.

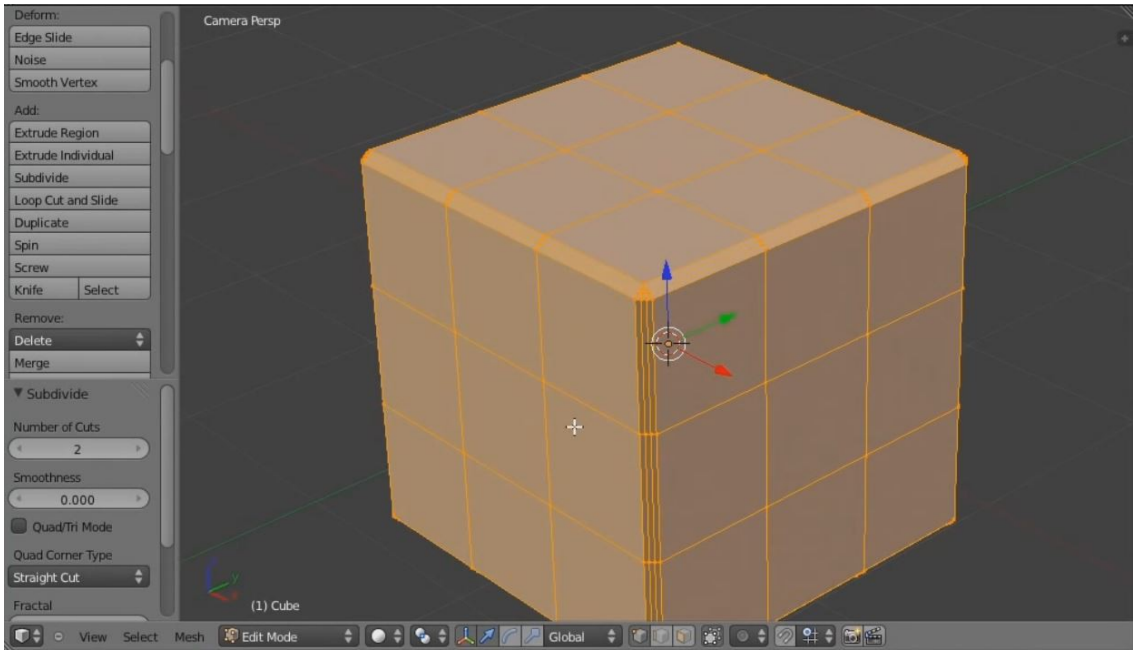
Nella Tool Shelf, in basso, possiamo variare il parametro *Percentage*, nella sezione Bevel; portiamolo, ad esempio, a 0.05.




*Utilizzo di Bevel in Edit Mode*

A questo punto, nella Tool Shelf, facciamo 2 tagli per ogni asse (X, Y e Z) con Subdivide, in modo da creare una "griglia" sulle facce del cubo: ci servirà per creare i cerchi per le fossette, che realizzeremo mediante estrusione e arrotonderemo con Subdivision Surface.

Con tutti i vertici selezionati in Edit Mode, quindi, facciamo click su *Subdivide* nella Tool Shelf e, in basso, digitiamo 2 nel campo *Number of Cuts*.

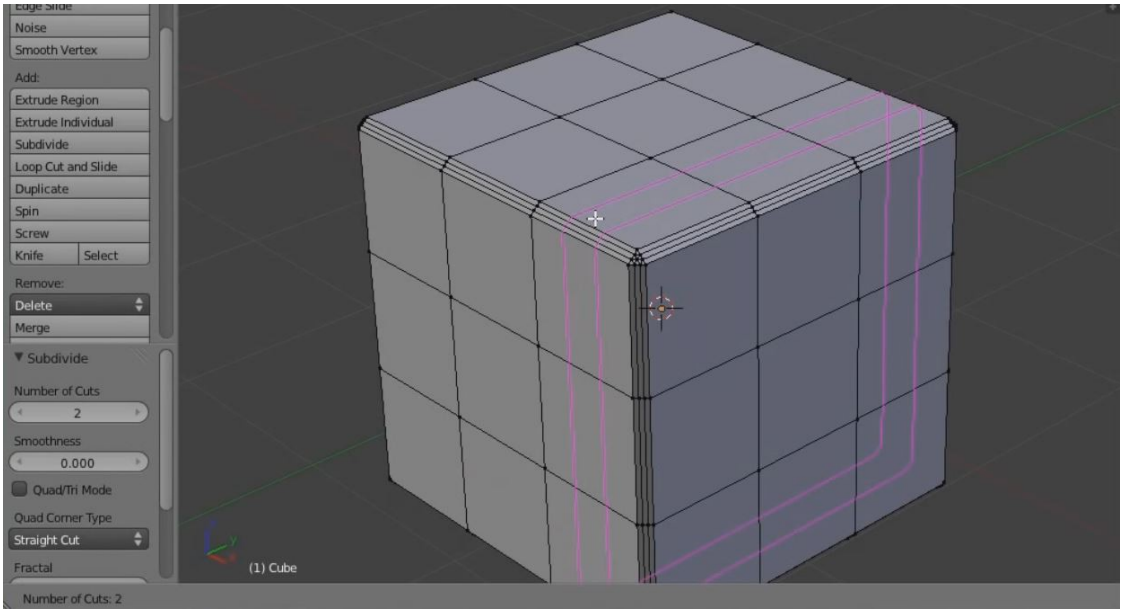



*Realizzare i tagli con Subdivide - 2 cuts*

A questo punto, dobbiamo fare le fossette circolari per i valori; mostrerò come fare per una fossetta, poi bisognerà ripetere la stessa operazione per le altre.

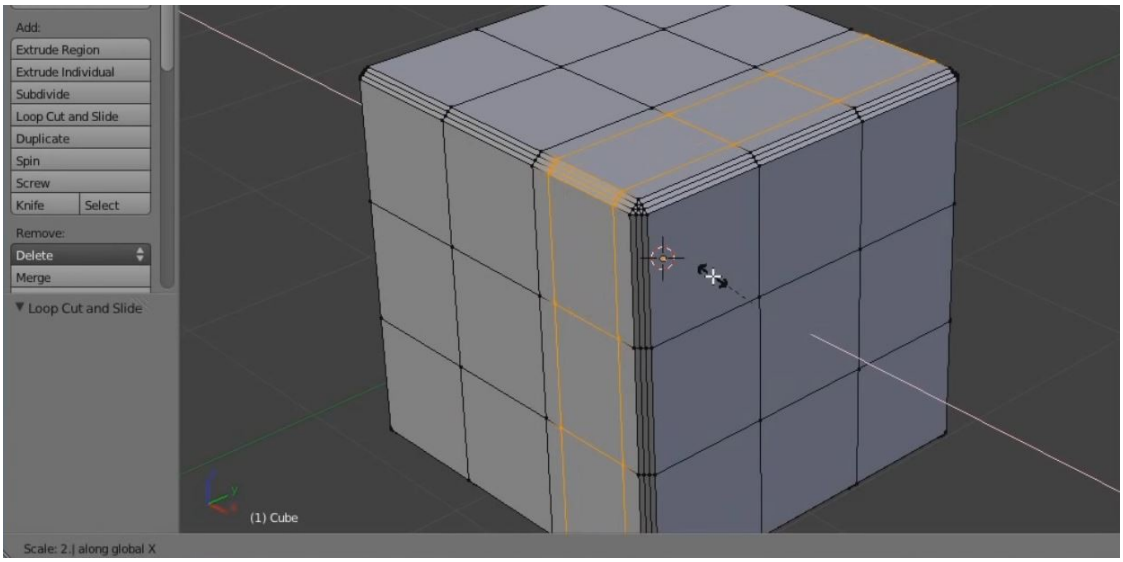
Prima di passare alle fossette, comunque, dobbiamo fare dei "corridoi" con Ring Cut per distanziare le facce presenti, altrimenti i cerchi delle fossette si toccheranno l'un l'altro; per fare questa operazione, utilizzeremo 9 volte Ring Cut, ogni volta con due tagli e allontanando tra loro i tagli in maniera parametrica con lo Scaling.

Premiamo quindi *CTRL R* e posizioniamo Ring Cut in una direzione, tagliando una fila di facce; ruotiamo la rotellina del mouse in modo da creare due tagli all'interno di questa fila, inizialmente equidistanziati, e facciamo click col tasto sinistro per confermare.




*Creazione dei tagli con Ring Cut*

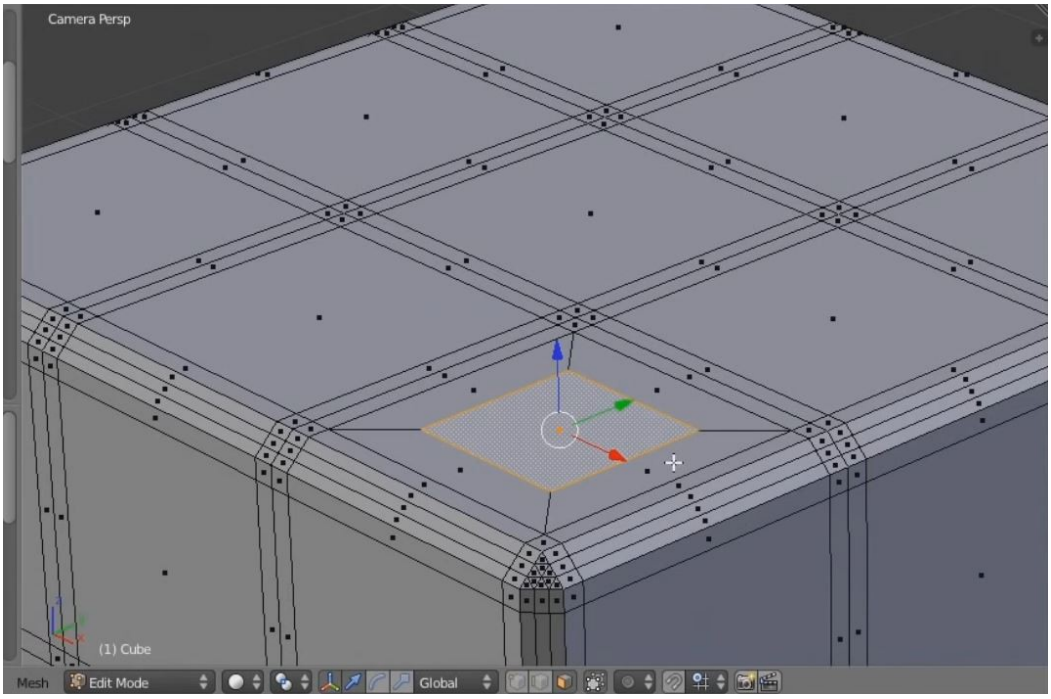
Con i nuovi segmenti ancora selezionati, premiamo S seguito dall'asse perpendicolare ai segmenti e dal valore 2.6, premendo poi Invio per concludere: lo scaling di 2.6 lungo l'asse perpendicolare serve infatti ad allontanare tra loro gli elementi.




*Creazione dei "corridoi", allontanando i segmenti*

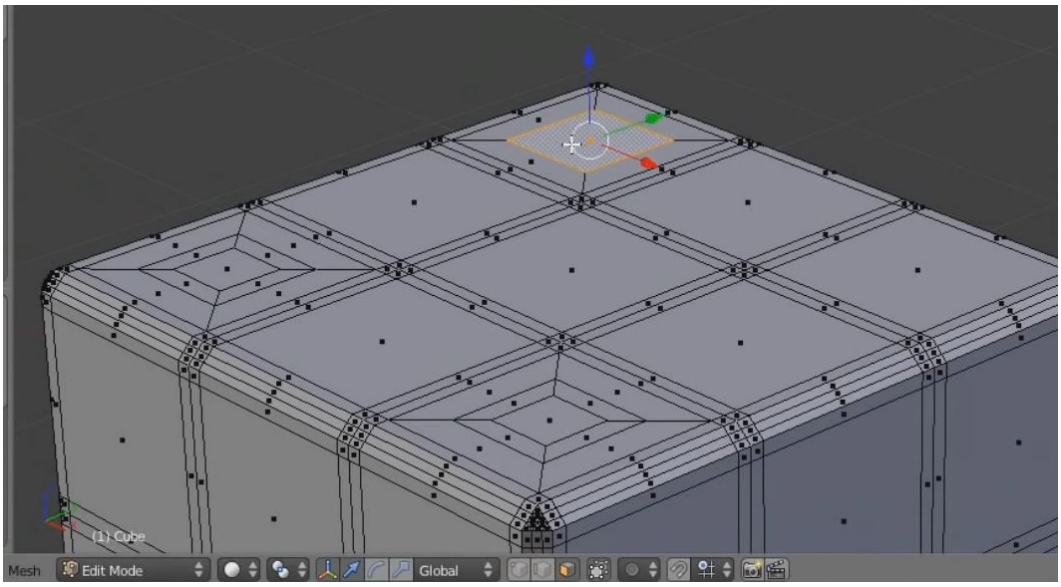
Dobbiamo ripetere questa operazione anche nelle altre file, 3 volte per direzione, in modo da creare questi “corridoi” di separazione per tutti i quadrati della griglia (9 quadrati per lato).

In Edit Mode, passiamo in modalità selezione facce, quindi selezioniamo la faccia che dovrà contenere la fossetta, premiamo *E* (estrusione) e subito dopo Invio, quindi digitiamo S 0.6 Invio: in pratica, abbiamo creato, mediante estrusione e scaling con fattore di scaling 0.6, un quadrato all'interno del quadrato originale.



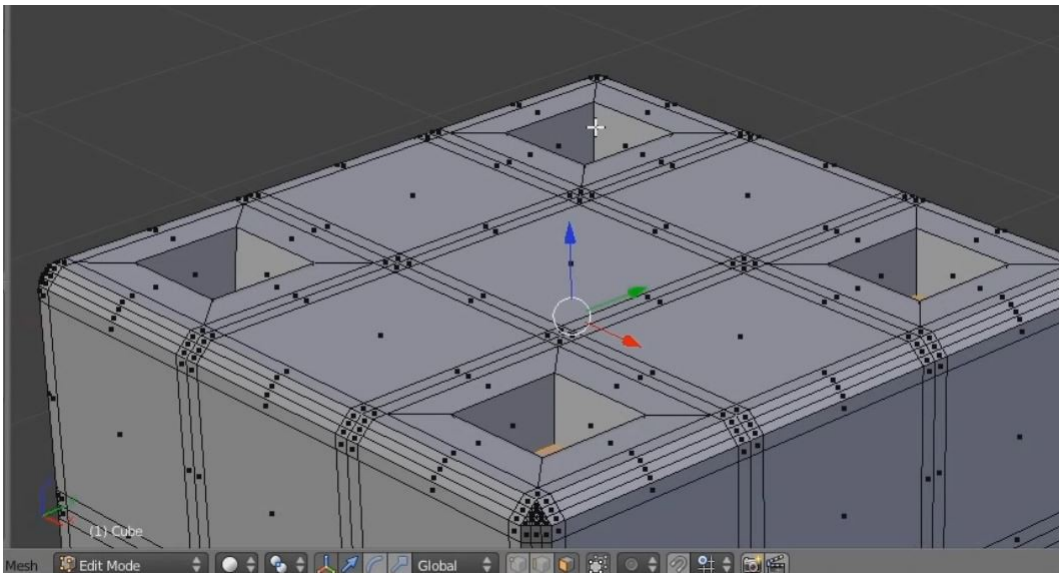

*Creazione di un quadrato (per una fossetta)*

Con questa nuova faccia selezionata, facciamo di nuovo Estrusione e Invio seguito da S 0.6 Invio, per creare un altro quadrato dentro l'altro. Dobbiamo ripetere queste operazioni anche per le altre fossette su questo lato, quindi di nuovo *Face Select Mode*, estrusione, scaling, estrusione, scaling.



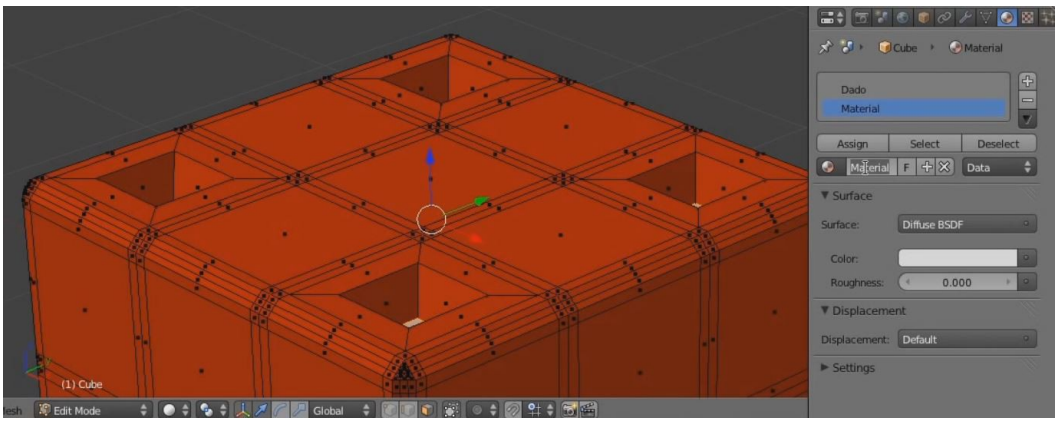

*Creazione dei quadrati per le altre fossette*

Una volta disegnate le fossette su un lato del cubo, facciamole entrare un po' nel dado, per creare davvero la fossetta: dobbiamo selezionare solo la faccia centrale di ogni fossetta e spostarla, operazione che possiamo fare in maniera parametrica; il dado, infatti, è stato creato col cubo di default, che ha lato 2 e ha le facce orientate con gli assi, quindi ad esempio per il lato superiore possiamo selezionare le facce centrali delle relative fossette e digitare G (grab) Z (asse verticale) – (spostamento verso il basso) 0.15 o poco più.



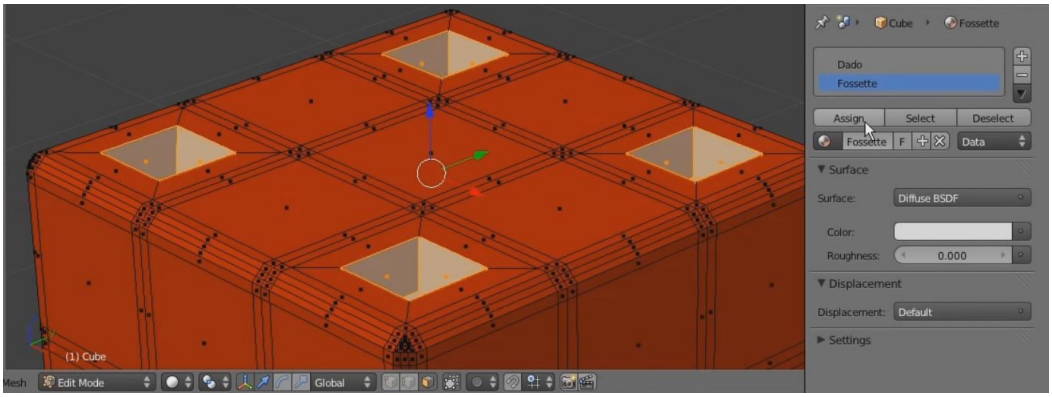

*"Rientrare" le fossette*

Alt, fermi! Prima di ripetere l'operazione con le altre facce, cambiamo il motore di rendering in Cycles, quindi andiamo nella finestra Material, nella Properties Window, e aggiungiamo (se non presente di default) un Material per tutto il cubo, ad esempio con colore rosso, chiamandolo ad esempio “Dado”; aggiungiamo poi un secondo sotto-materiale, con colore ad esempio bianco, chiamandolo “Fossette”.




*Creazione dei sottomateriali in Cycles*

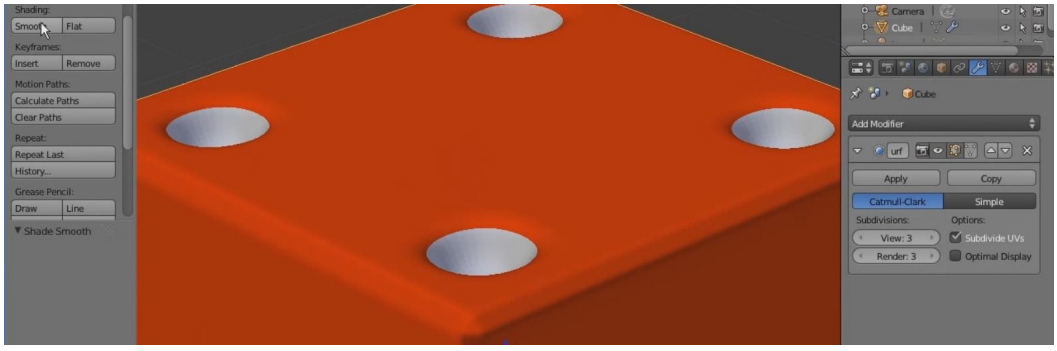
Nella 3D View, in Edit Mode, selezioniamo le facce centrali, quelle in basso, delle fossette, quindi premiamo CTRL + per allargare la selezione ai vertici vicini; con questa selezione nella 3D View, andiamo nella scheda Material, selezioniamo il secondo sottomateriale e clicchiamo su *Assign*, in modo da assegnarlo ai vertici selezionati.




*Assegnare il sottomateriale bianco alle fossette*

Conviene fare questo lavoro subito dopo aver definito una fossetta o in generale le fossette per un lato del cubo (come visto poco fa, dopo aver fatto lo spostamento per creare la fossetta basta premere CTRL + per selezionare anche i vertici di contorno e assegnare il secondo sottomateriale alla selezione).

A questo punto non dobbiamo far altro che ripetere queste operazioni per le altre fossette degli altri lati, assegnando nel mentre il secondo sottomateriale ai quadrati.
Terminato questo lavoro, torniamo in Object Mode e aggiungiamo un Modificatore Subdivision Surface con 3 suddivisioni sia per View che per Render, e applichiamolo; come potete vedere, le fossette ora sono circolari, ma l'oggetto è ancora troppo sfaccettato, quindi premiamo su Shading *Smooth* nella Tool Shelf per dargli un aspetto liscio.




*Dopo aver applicato Subsurf e lo Shading SMOOTH*

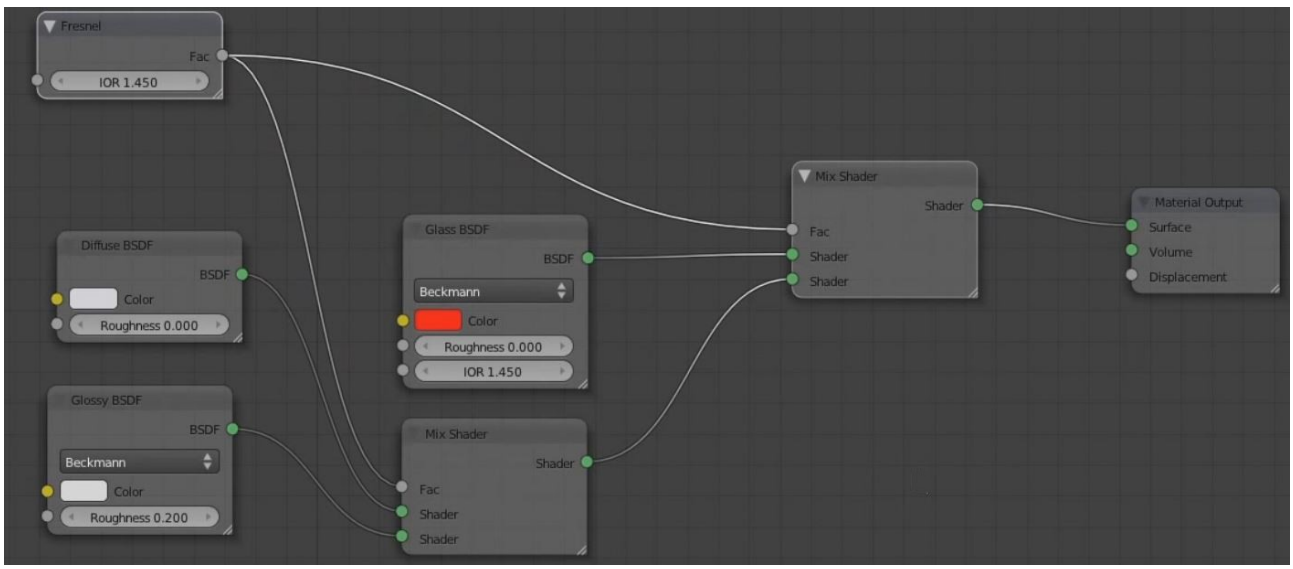
Adesso, due parole sugli Shaders utilizzati per ottenere il rendering visto in apertura.
Il Material delle fossette bianche è un semplicissimo Diffuse Bianco, non c'è molto altro da dire.

Per quanto riguarda il Material dei dadi, passiamo allo Screen Layout *Compositing* con i Material Nodes selezionati e clicchiamo su *Use Nodes* per il materiale nella Properties Window.

Aggiungiamo due Nodi *Shader – Mix Shader* all'editor.
Aggiungiamo all'editor anche un Nodo *Input – Fresnel*, un Nodo Shader *Glass* e un *Glossy*, lasciando inoltre il Diffuse presente di default. Cambiamo il colore del Glass in rosso.

Se preferite, potete portare la Roughness del nodo Glass a 0.1 o più, ottenendo un effetto acidato/satinato. Nel primo Nodo Mix, colleghiamo sul primo input Shader il Nodo Diffuse e sul secondo il Glossy.

Passiamo l'output di questo Mix al SECONDO input Shader del secondo Nodo Mix; sul primo Nodo Shader, colleghiamo il Nodo Glass rosso. Colleghiamo quest'output a Surface di Material Output. Colleghiamo il Nodo Fresnel, con IOR 1.15 (Glass avrà 1.45) sui Factor sia del primo che del secondo nodo Mix.



*Lo schema di Nodi per la parte trasparente (rossa)*

*Il risultato finale*

*  *  *

# La scala di Giacobbe: arco elettrico con Hook, Wave e Compositing

In questo tutorial vedremo come realizzare l'animazione di un effetto noto come “scala di Giacobbe”: un arco elettrico ad alto voltaggio che si muove lungo due terminali a V, dal basso verso l'alto, ripetendo questo movimento più volte (un effetto visibile ad esempio in molti film dove ci sono degli scienziati pazzi nei loro laboratori).

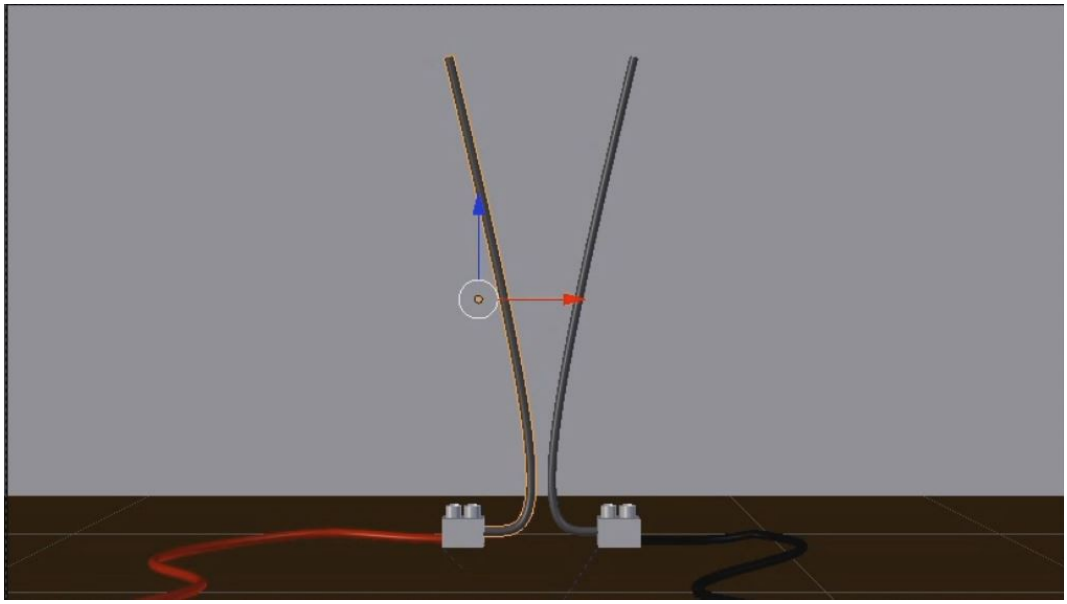
Tratteremo i seguenti argomenti: Hooking Vertices per animare una curva; Wave e il modificatore Cycles sulle IPO; Emit e Indirect Lighting per l'illuminazione; Post-Produzione con Glare e Blur per aggiungere il bagliore.

Esaminiamo la scena di partenza (riportata nell'immagine nella pagina seguente): abbiamo 4 Bezier Curves, due che rappresentano gli elettrodi (e infatti si chiamano Steel Electrode 1 e 2) e due che rappresentano i cavi di alimentazione, chiamati RedWire e BlackWire, dai colori utilizzati per identificarli; per lo spessore di queste 4 curve Bezier non ho utilizzato Extrude delle curve ma un BezierCircle utilizzato, per ogni curva, come Bevel Object, nella scheda Object Data delle curve.

Ci sono inoltre 4 mesh: due Planes (uno per lo sfondo, l'altro per il piano di appoggio) e due mesh realizzate partendo da un Cube e due Cilindri e che rappresentano due morsetti Mammut.

Non sono presenti fonti di luce nella scena, né Ambient Occlusion o Environment Lighting, infatti avviando un rendering ora l'immagine sarà completamente nera; ho scelto queste impostazioni per mostrarvi gli effetti di Indirect Lighting generata dalla curva con Emit maggiore di 0 nel Material.

Prima di procedere, parliamo un po' di cosa vogliamo ottenere: vogliamo una specie di curva che, partendo dalla base degli elettrodi, sale seguendo il loro percorso, quindi riparte dalla base, per un certo numero di volte, realizzando appunto l'effetto noto come “scala di Giacobbe”.



*La scena di partenza*

Primo punto da risolvere: come facciamo a far salire la Curva spostandone gli estremi per seguire gli elettrodi? Utilizzeremo due oggetti Empty e una tecnica nota come *Hooking Vertices*, che aggancia (*To hook*, appunto) uno o più punti di controllo di un oggetto a una Empty, per cui poi spostando la Empty sposteremo il punto di controllo della curva.

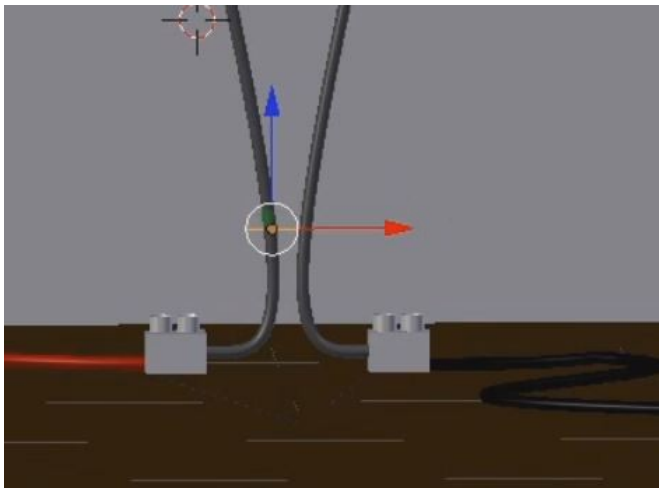
Secondo punto da risolvere: come fare per ripetere la stessa animazione per, ad esempio, 5 volte, animando però solo una volta il movimento? Per risolvere questo problema, utilizzeremo il modificatore Cycles per la curva IPO delle Empty, in modo da ripetere continuamente l'animazione stabilita la prima volta.

Terzo punto da risolvere, riguardo la curva: come generare del rumore, per simulare appunto una scarica? Ci sono vari modi per risolvere questo problema; quello che utilizzeremo consiste nell'introdurre un modificatore Wave su una direzione; poiché Wave agisce su X o su Y, potrebbe essere necessario ruotare la curva intorno ad un asse per orientare il modificatore sull'asse Z, quello verticale.

Quarto e ultimo punto: come renderizzare la curva con illuminazione e un po' di bagliore Glare?
Per risolvere questo problema, utilizzeremo il Material con Emit e Indirect Lighting e un po' di Nodes Compositing, in post produzione.

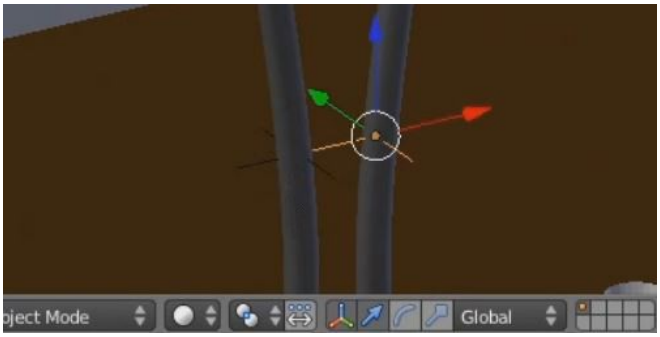
Fatte queste premesse, iniziamo sul serio!
Assicuriamoci di essere al frame 1 della timeline, con 240 frames totali e tutte le altre impostazioni nella scheda Render come vogliamo (ad esempio, 24 fps, 1280x720, nel mio caso).

Selezioniamo uno dei due elettrodi e spostiamo lì il 3D Cursor con SHIFT S e scegliendo *Cursor to Selected*; aggiungiamo, quindi, il primo oggetto Empty alla scena, spostandolo nel punto ove vogliamo far partire l'arco voltaico. Chiamiamo questa Empty "Hook 1". Scaliamo un po' questo oggetto Empty per non fare confusione con gli altri oggetti, quindi premiamo I e inseriamo un Keyframe *Location* per l'oggetto: stiamo registrando la sua posizione al frame 1.



*Creazione della prima Empty, per la quale registreremo il primo Keyframe al frame 1*

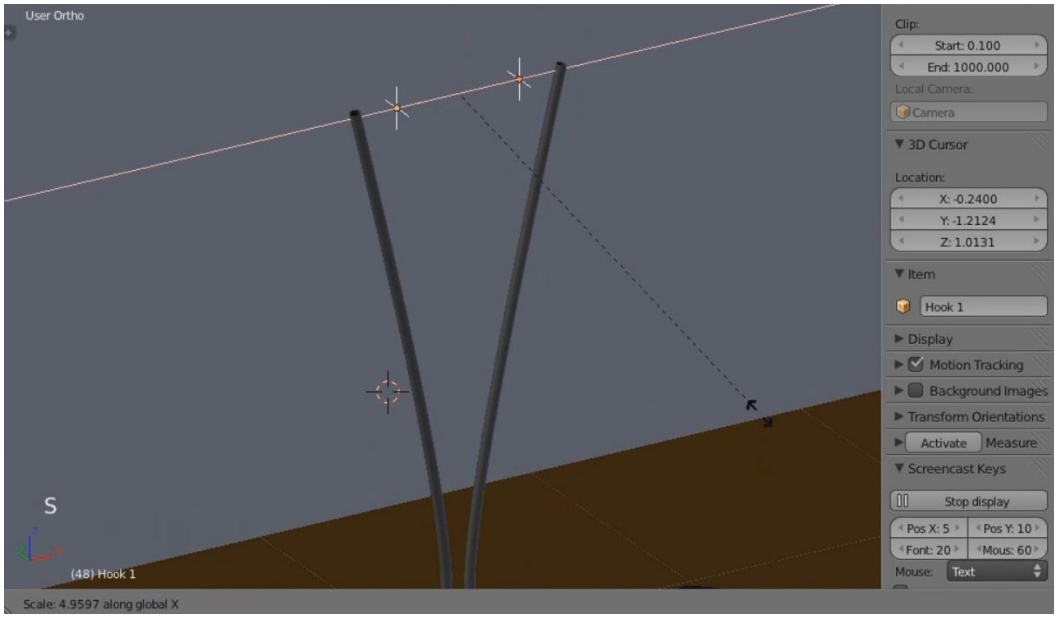
Duplichiamo questa Empty e spostiamo la copia in orizzontale, posizionandola sul secondo elettrodo; chiamiamo questa Empty "Hook 2" e inseriamo un keyframe *Location* al frame 1.




*Creazione e posizionamento della seconda Empty*

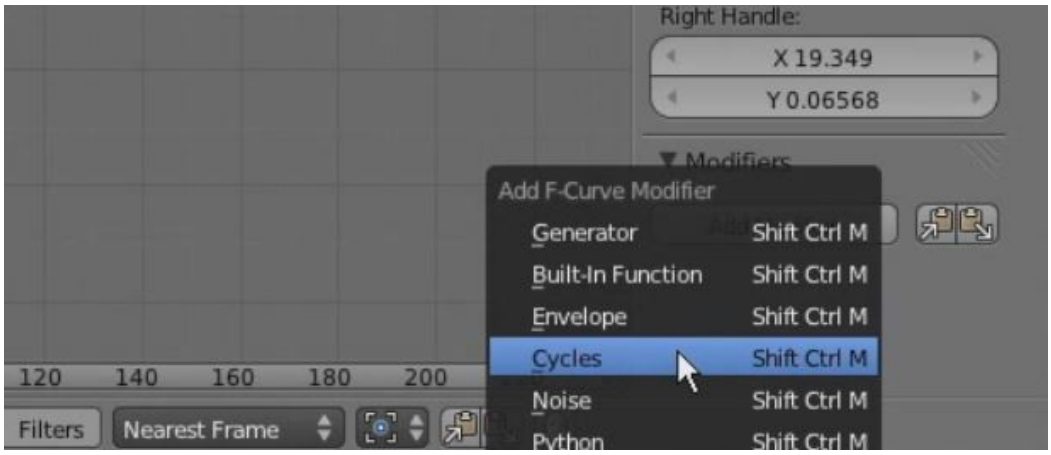
Adesso andiamo avanti nella Timeline; ad esempio, se vogliamo far durare la salita 2 secondi, per vederla bene, andiamo al frame 48, quindi selezioniamo sia la prima che la seconda Empty (con una selezione multipla, con SHIFT e click del tasto destro), spostiamo in alto le Empty lì dove ci sono le punte degli elettrodi, quindi allontaniamole (scalando sull'asse orizzontale, quindi S X) e inseriamo due Keyframe *Location*, per registrare la posizione al 48esimo frame.

Con S X allungheremo anche le Empty su X, ma qui non è un problema.



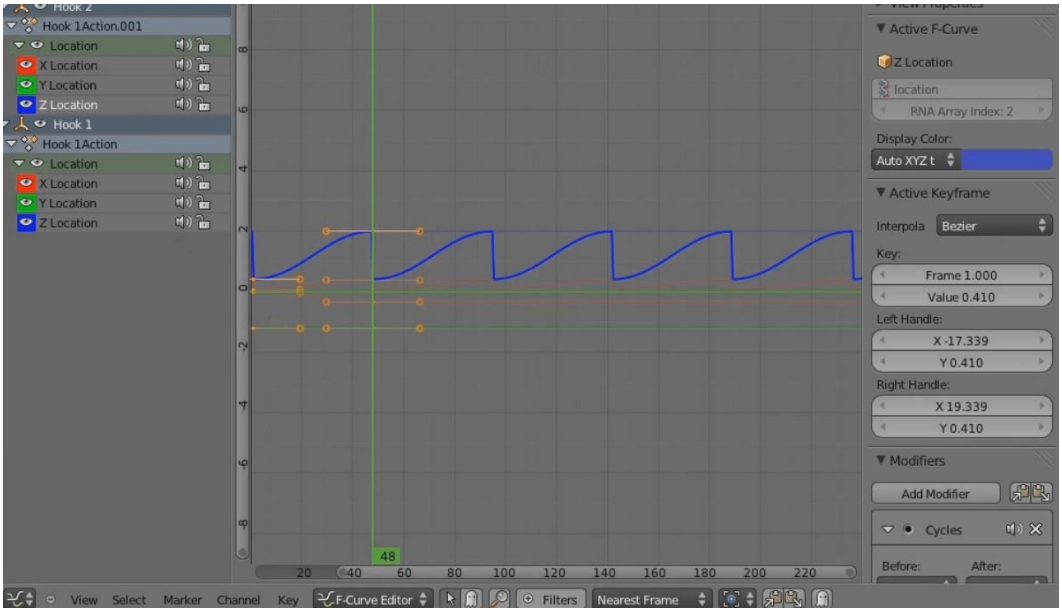

*Posizionamento e registrazione delle Empty al frame 48*

Per rendere ciclica l'animazione, con le due Empty selezionate apriamo una finestra Graph Editor, selezioniamo ad esempio il canale X della prima Empty, quindi nella Transform Window a destra (da aprire con N, se inizialmente non visibile) clicchiamo su *Add Modifier* e scegliamo *Cycles*.



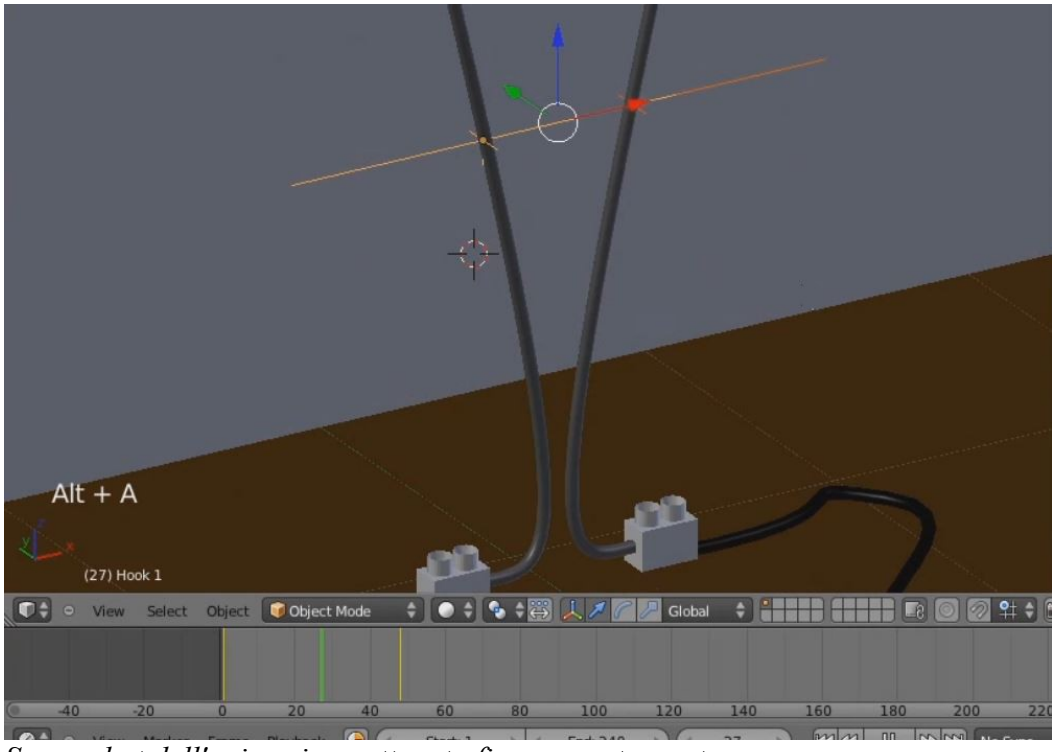

*Modificatore Cycles per le IPO della prima Empty*

L'animazione sul canale X è ora ciclica; dobbiamo ripetere la stessa operazione per il canale Z della prima Empty e per i canali X e Z della seconda... perché X e Z? Perché stiamo spostando le Empty orizzontalmente, quindi nel nostro caso sull'asse X, e in verticale, quindi, sempre nel nostro caso, sull'asse Z; non ci sono spostamenti in profondità, quindi l'asse Y, qui, non ci interessa.




*Modificatore Cycles per la seconda Empty*

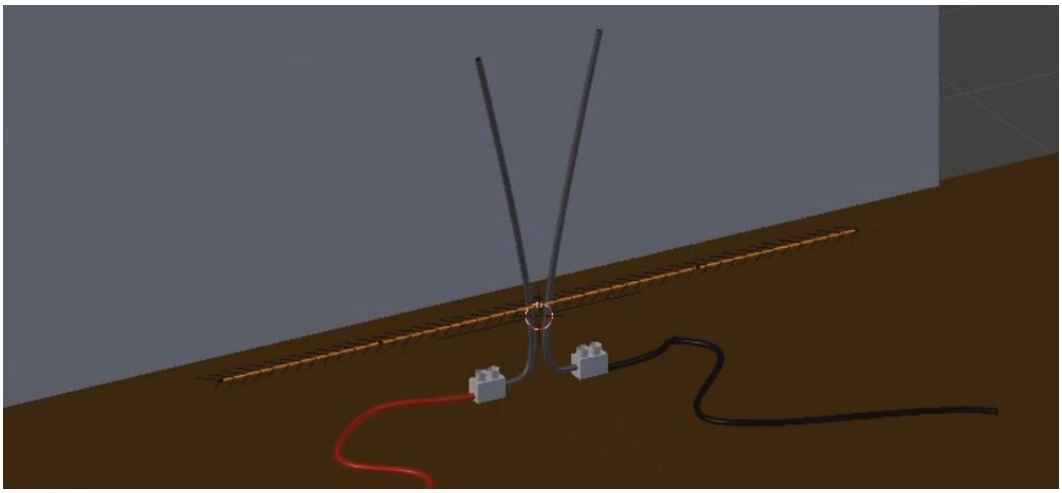
Fatto questo lavoro, torniamo alla 3D View e premiamo *ALT A* per avere un'anteprima dell'animazione: le due Empty "salgono" lungo gli elettrodi in 48 frames, quindi ripetono l'animazione per tutto il filmato.




*Screenshot dell'animazione ottenuta fino a questo punto*

Torniamo al frame 1 e, sempre con le due Empty selezionate, premiamo SHIFT S e scegliamo *Cursor to Selected*, per spostare il 3D Cursor in mezzo ai due oggetti: ciò ci consentirà di posizionare lì la Bezier Curve che stiamo per creare e che rappresenta la scarica elettrica.

Aggiungiamo una Curva di tipo PATH, andiamo in Edit Mode e cancelliamo tre punti di controllo (di default, una curva PATH ha inizialmente 5 punti di controllo), quindi spostiamo i due punti di controllo in modo da far coincidere le estremità con le posizioni delle due Empty.

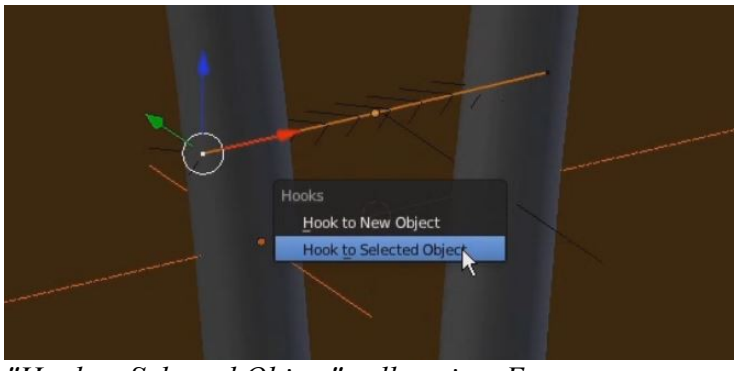


*Creazione e modifica della curva PATH*

Selezioniamo il punto di controllo a sinistra.

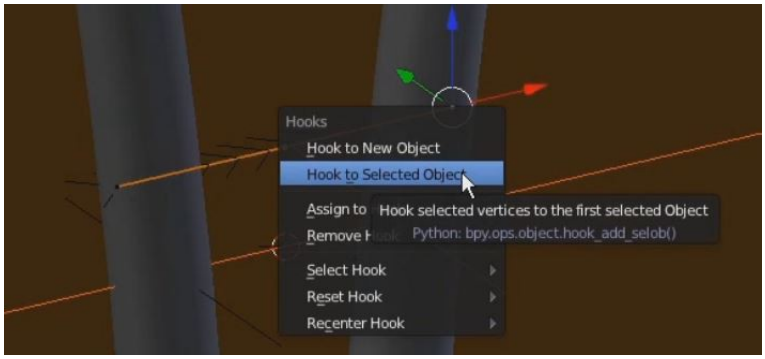
Torniamo in Object Mode.

Per legare un punto di controllo a un oggetto Empty esistente, come nel nostro caso, dobbiamo selezionare prima la Empty da utilizzare come punto di ancoraggio, quindi ANCHE la curva (selezione multipla, quindi), andare in Edit Mode e selezionare il punto di controllo o i punti di controllo da ancorare, nel nostro caso solo quello a sinistra, e premere CTRL H, scegliendo "*Hook to Selected Object*" dal menù che apparirà.




*"Hook to Selected Object" sulla prima Empty*

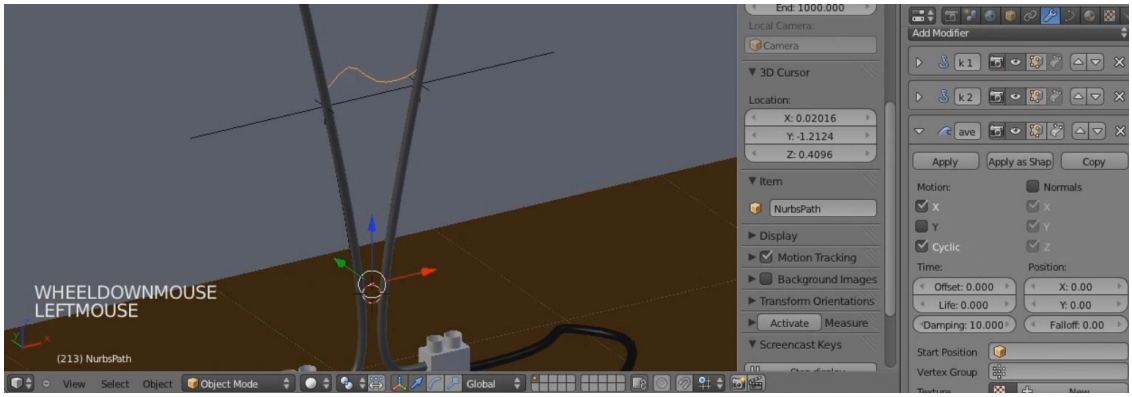
Torniamo in Object Mode, selezioniamo la seconda Empty, quindi selezioniamo la curva PATH, andiamo in Edit Mode, selezioniamo il secondo punto di controllo, premiamo CTRL H e scegliamo "*Hook to Selected Object*" per ancorare il secondo punto di controllo della curva alla seconda Empty.




*Hooking della parte finale della curva PATH alla seconda Empty*

Torniamo in Object Mode, selezioniamo solo la curva PATH e premiamo ALT A per osservare l'animazione della curva.

Con questo oggetto selezionato, andiamo nella scheda Modifiers, all'interno della Properties Window, e aggiungiamo un Modificatore Wave, deselezionando Y e modificando il valore di Height come preferiamo, osservando gli effetti in tempo reale nella 3D View.



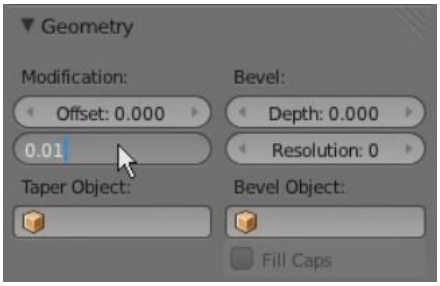

*Effetto dell'utilizzo del primo modificatore WAVE*

Se l'oscillazione vi sembra troppo regolare, potete aggiungere un secondo modificatore Wave cambiando, oltre a Height, anche altri parametri come Offset, Speed o Narrowness; ad esempio, io sto aggiungendo un secondo Wave, solo su X (come il primo), con Offset 1.5, Speed 1, Height 0.2, Width 1.5 e Narrowness 3.0 (comunque questo è solo un esempio, potete fare tutte le modifiche che volete).

*Utilizzo del secondo modificatore WAVE*

Con l'animazione della curva ci siamo, ora tocca all'aspetto finale!

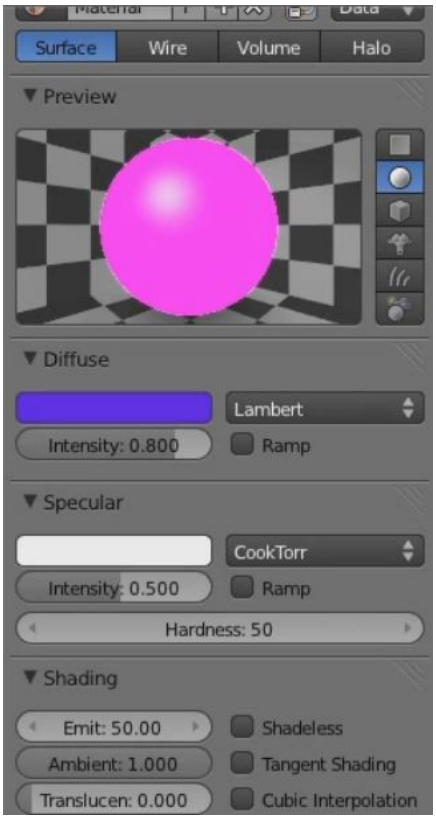
Per prima cosa, sempre con la PATH selezionata, andiamo nella scheda Object Data, nella Properties Window, e portiamo l'estrusione ad esempio a 0.01.




*Estrusione della curva PATH*

Adesso andiamo nella scheda Object e impostiamo il *Pass Index* a 1; questo ci servirà per isolare l'oggetto dal resto del rendering in fase di Compositing.

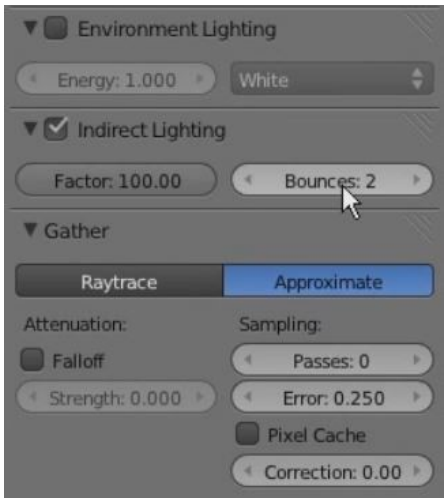
Apriamo la scheda Material, forniamo alla PATH un Material di base, ad esempio con colore Blu, colore che comunque verrà modificato da Emit, che attiveremo con un valore elevato, ad esempio 50 o più.




*Material per la curva PATH (che ha Pass Index 1)*

Avviando ora un rendering, comunque, vedremo solo la curva, non l'ambiente circostante: per rendere l'oggetto davvero illuminante, dobbiamo andare nella scheda World, nella Properties Window, e attivare Indirect Lighting, impostando *Gathering Method Approximate*, 2 Bounces e un valore molto alto per Factor, ad esempio 100.

L'illuminazione è data, quindi, da Emit (che regola l'intensità dell'emissione dell'oggetto) e da Factor, che regola in generale l'intensità dell'illuminazione dovuta, in tutta la scena, all'illuminazione indiretta (il primo è un parametro *Locale*, il secondo è *Globale*; in questa scena l'unico illuminante è la Path, quindi un valore così alto per Indirect Lighting non è un problema).




*Attivazione di Indirect Lighting in WORLD*

È giunto il momento di aggiungere un po' di bagliore alla scarica! Abbiamo già impostato il Pass Index di questo oggetto a 1, quindi dobbiamo aprire la sezione Layers nella scheda Render, all'interno della Properties Window, e selezionare la casella Object Index.

*Selezionare la casella "Object Index" per i Passes*

Cambiamo lo Screen Layout in *Compositing* per dedicarci ai Nodes.

Nell'header della finestra, scegliamo la terza icona (Compositing Nodes) e selezioniamo *Use Nodes*.

Tagliamo il collegamento tra *Image* di Render Layers e *Image* di Composite.

Aggiungiamo due Nodi *Color Mix* in modalità Add con Factor 1 all'editor.

Aggiungiamo un Nodo *Convertor ID Mask*, impostando il *Pass Index* a 1 e collegandolo, sulla socket di input del Nodo, all'uscita *IndexOB* di *Render Layer.*

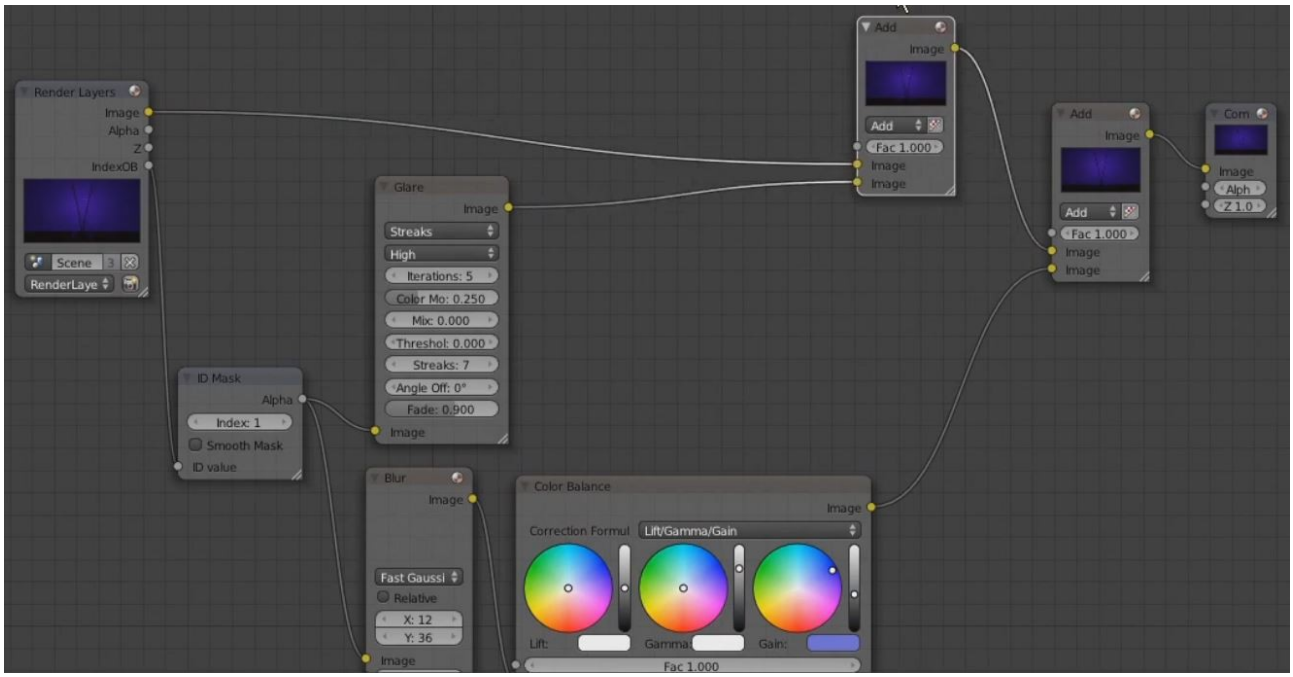
Aggiungiamo un Nodo *Filter Blur*, in modalità Fast Gaussian, con ad esempio 12 per X e 24 o 36 per Y, collegandogli in input l'output Alpha di *ID Mask*.

Colleghiamo questo Nodo *Blur* a un Nodo *Color Balance*, per modificare la tinta della sfocatura agendo su Gamma e su Gain.

Aggiungiamo un Nodo *Filter Glare* all'editor, con Streaks, qualità High, magari 6 o 7 Streaks, e soprattutto Threshold a 0.0, altrimenti non vedremo l'effetto. Colleghiamo l'input di *Glare* all'output Alpha di *ID Mask*, che ha quindi due collegamenti in uscita.

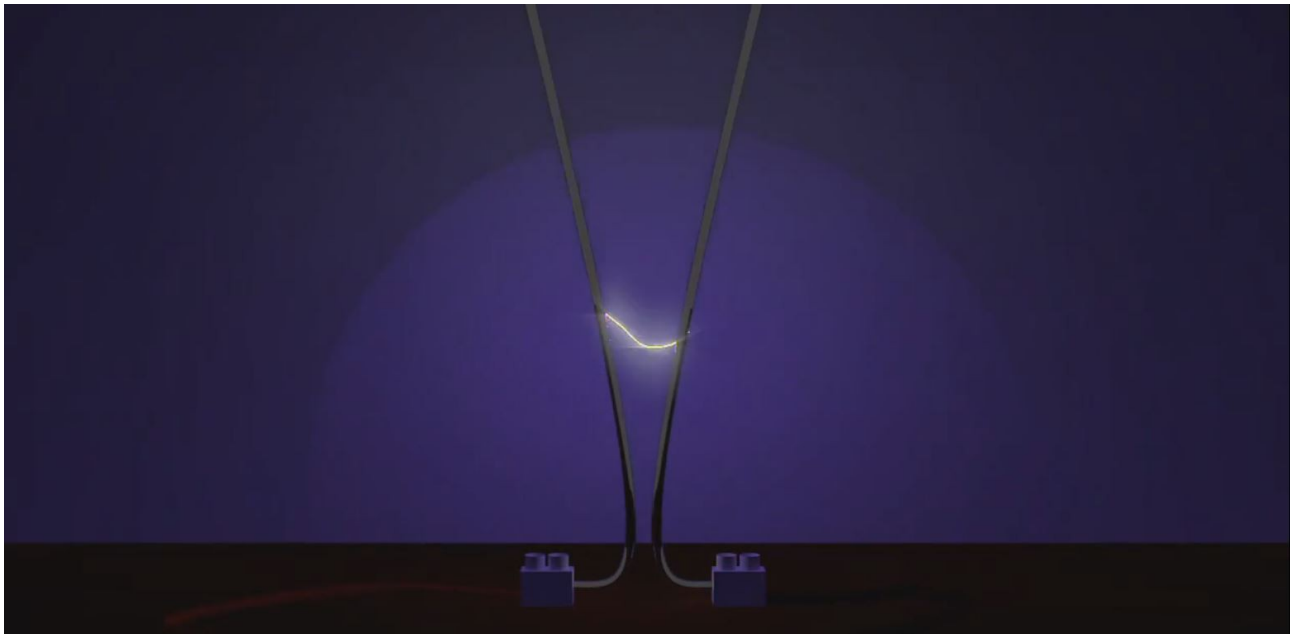
Colleghiamo Image di *Render Layers* e Image di *Glare* sui due canali input Image del primo Nodo *Mix Add*.

Colleghiamo l'output del primo Nodo *Mix Add* e Image di *Color Balance* agli input Image del secondo Nodo *Mix Add*, collegando infine questo output a Image di *Composite*.



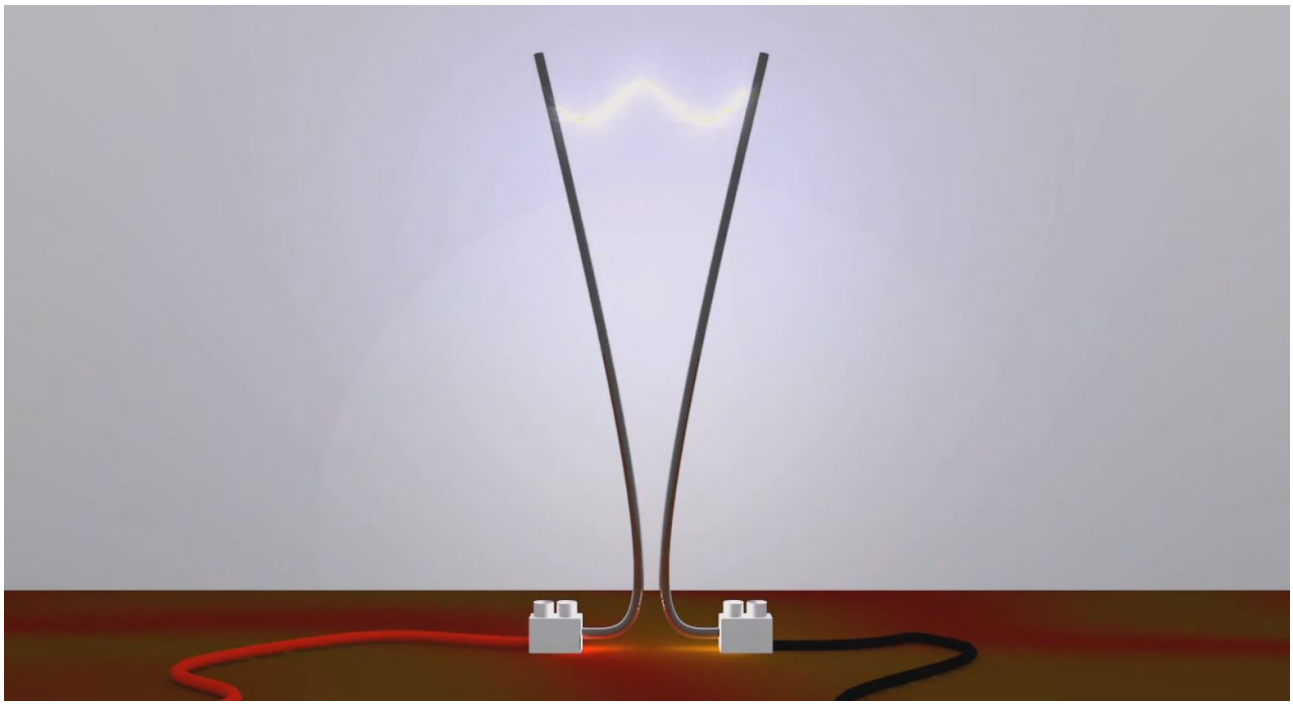
*Lo schema di Nodi di Compositing*

È tutto pronto: possiamo cliccare su Animation per renderizzare il nostro videoclip e osservare il risultato.



*Il risultato ottenuto (screenshot dell'animazione)*

Vi mostro anche la stessa scena renderizzata aggiungendo Ambient Occlusion, per mostrarvi l'effetto in un ambiente illuminato.

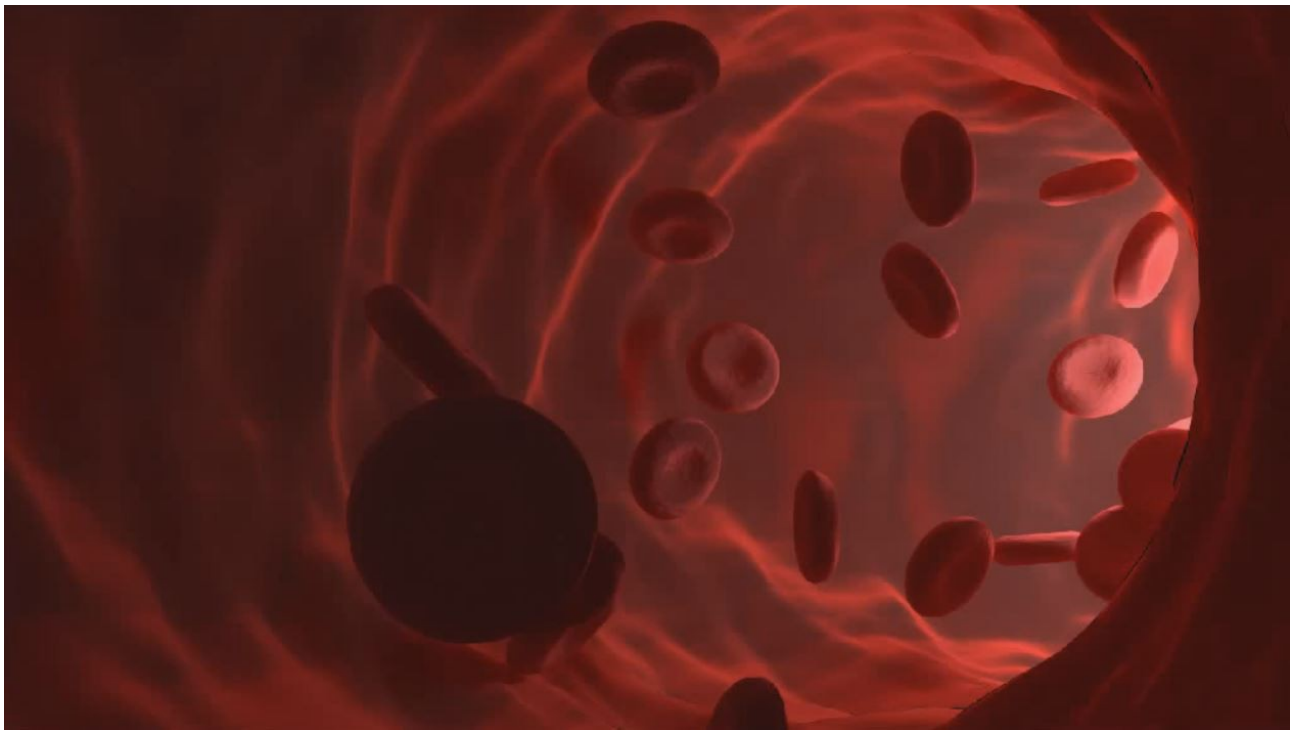


*Screenshot dell'animazione con Ambient Occlusion*

*  *  *

# Globuli rossi (red blood cells) in Blender 3D e Cycles

In questo tutorial vedremo il dietro le quinte del rendering “Red Blood Cells”, visibile nell'immagine seguente, realizzato mediante Blender 2.63 e Cycles.



*Il rendering "Red Blood Cells"*

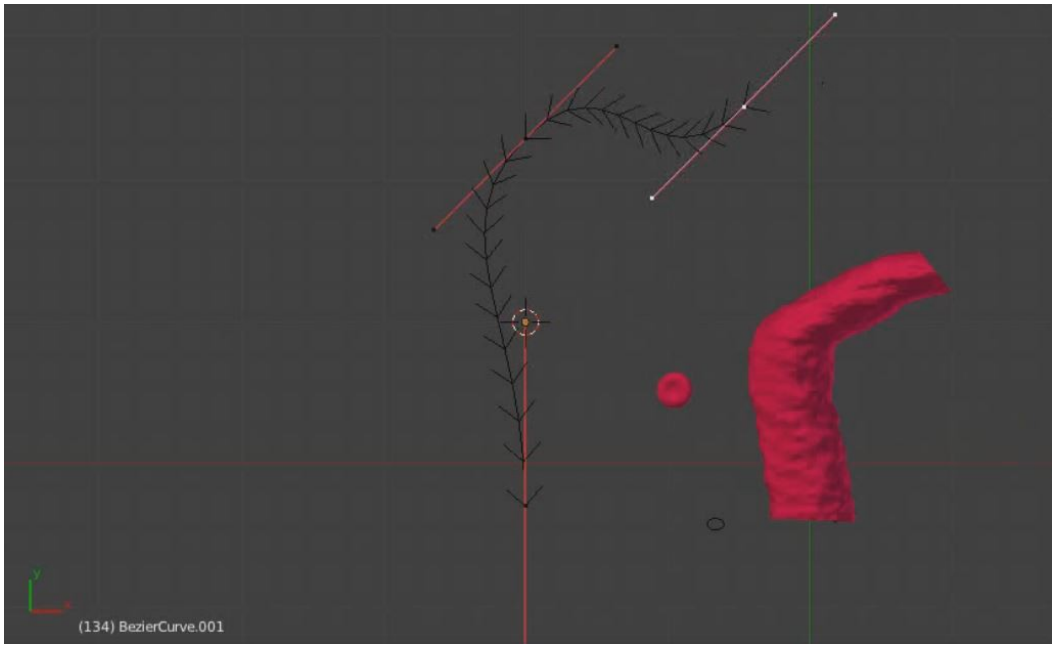
In realtà, la scena è molto molto semplice: sono presenti, in effetti, appena tre mesh e una fonte di luce, ma per arrivare a questo risultato da zero dovremo prima creare degli oggetti accessori e utilizzare certi modificatori.

Prima di iniziare, una considerazione: l'immagine non è, ovviamente, fotorealistica, e in generale le immagini che troverete online cercando, ad esempio, proprio “Red Blood Cells” sono più che altro rappresentazioni o interpretazioni artistiche; in questi risultati, l'aspetto dei globuli rossi varia

parecchio, inoltre dubito che le pareti interne dei vasi siano così e, ovviamente, nessuno di noi ha una fonte di luce nelle vene, quindi tutto il rendering è di fantasia... il che significa che bisogna giocare su effetti come chiaroscuri, contrasti, *flou* e altro per renderlo "accattivante".

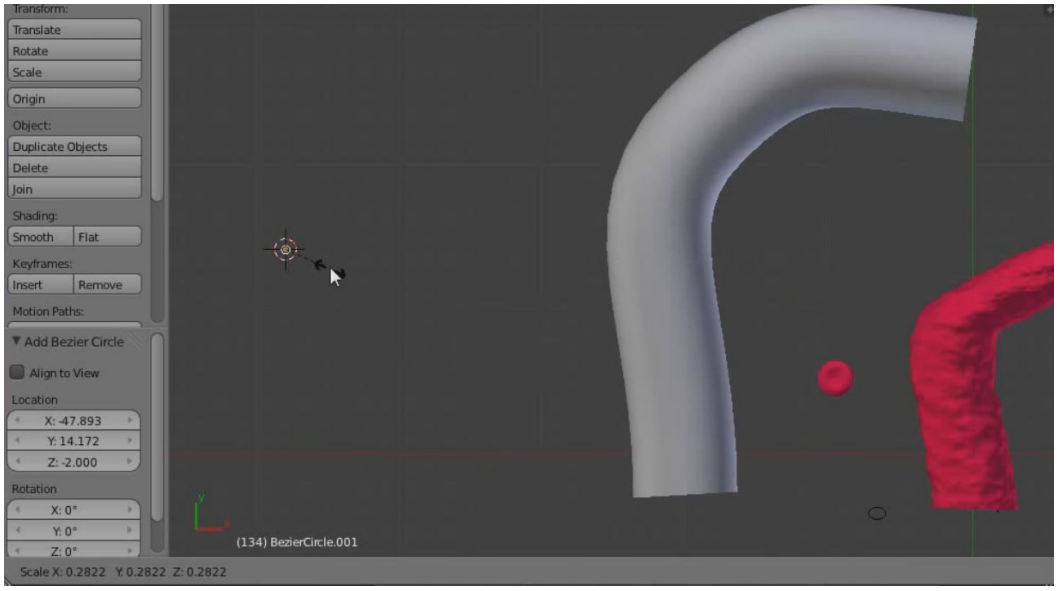
Io ci ho provato con questo schema di luci e inquadratura, vi invito ovviamente a provare altre configurazioni e altri effetti.

Iniziamo la descrizione del lavoro dal vaso sanguigno: per realizzarlo, ho creato prima una Bezier Curve che rappresenta il percorso del vaso, disponendola sul piano XY e orientandola in modo da farle fare una curva sul fondo. Per aggiungere un punto di controllo e prolungare la curva basta andare in Edit Mode, selezionare un'estremità e premere *E*, per "estrudi", trascinando, in questo caso sul solo piano XY.



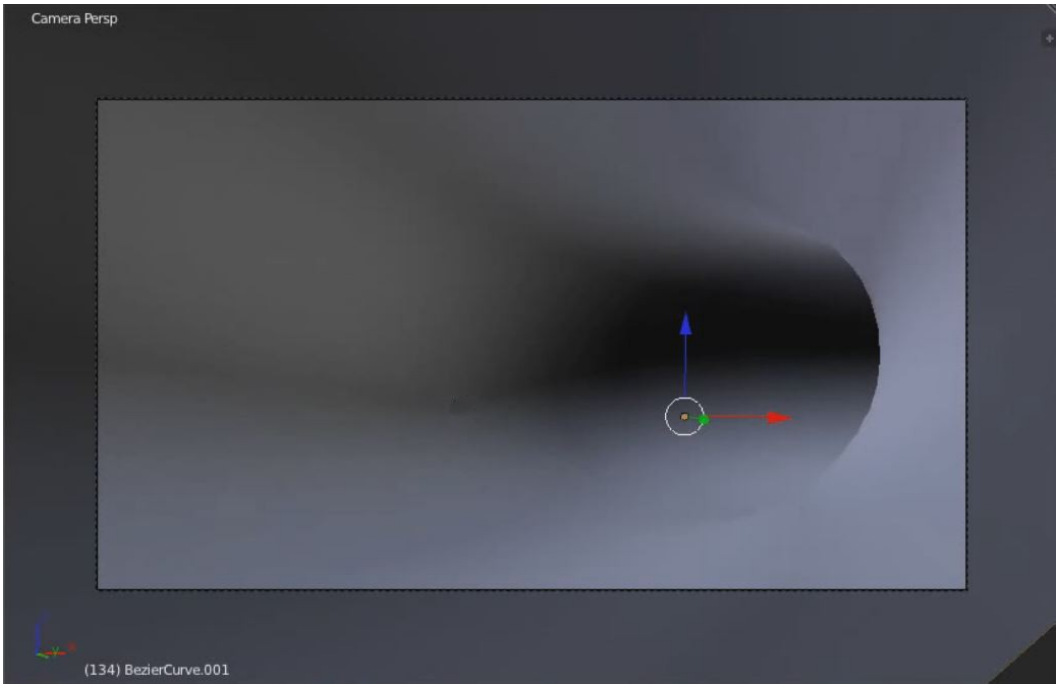
*Definizione del "percorso" del vaso sanguigno*

Per assegnare un profilo a questa curva, torniamo in Object Mode e aggiungiamo un oggetto Bezier Circle alla scena. Sempre in Object Mode, selezioniamo nuovamente la curva Bezier Curve di prima, apriamo la scheda Object Data, nella Properties Window, clicchiamo sul campo Bevel Object e, dal menù che apparirà, scegliamo il Bezier Circle appena creato. Il cerchio verrà utilizzato come profilo da estrudere lungo il percorso definito dalla Bezier Curve.




*Il cerchio come "Bevel Object" per la curva-percorso*

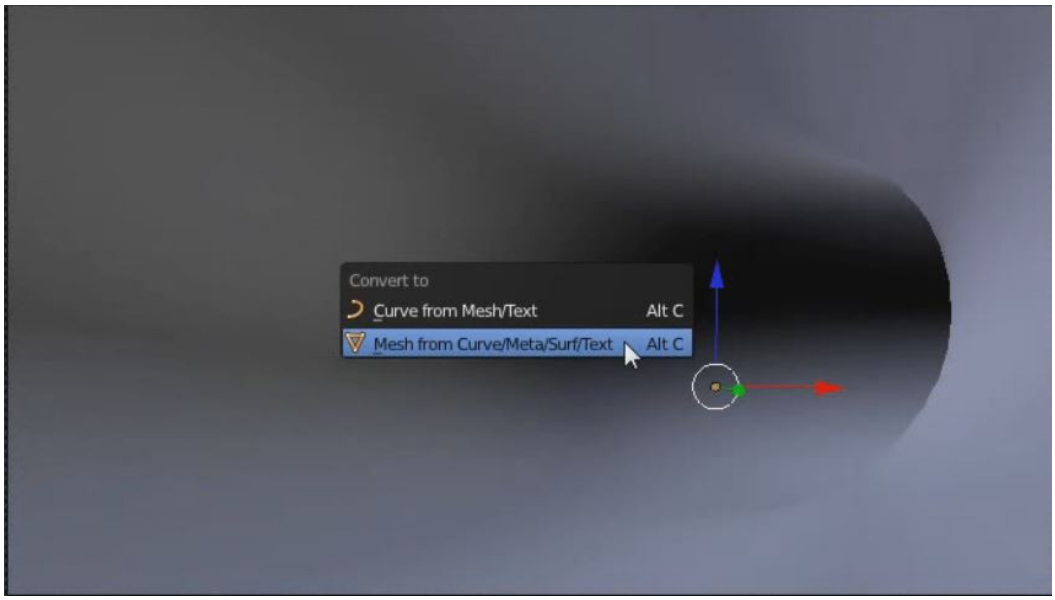
In questa fase possiamo ancora modificare interattivamente sia il profilo che il percorso, quindi posizioniamo la telecamera all'ingresso del vaso e facciamo le modifiche che riteniamo più opportune, prima di convertire tutto in Object Mode.




*Posizionamento della telecamera*

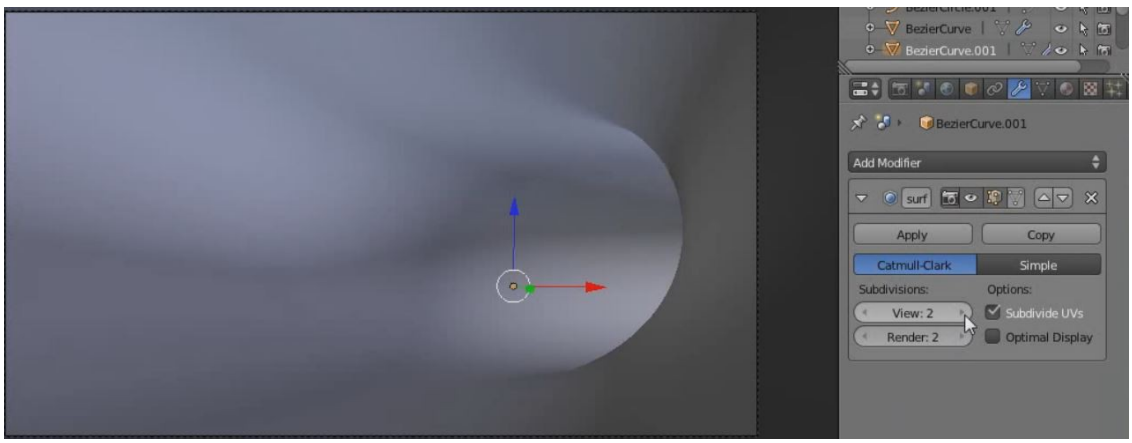
Per poter applicare il modificatore *Displace* e deformare un po' le pareti, infatti, dobbiamo trasformare l'oggetto da Curve in Mesh, solo che a quel punto non potremo modificarlo interattivamente con pochi punti di controllo o con il Circle profilo come stiamo facendo ora.

Quando siamo pronti per passare a mesh, selezioniamo il vaso sanguigno, in Object Mode, quindi premiamo ALT C, scegliendo poi "*Mesh from Curve*" dal menù che apparirà.



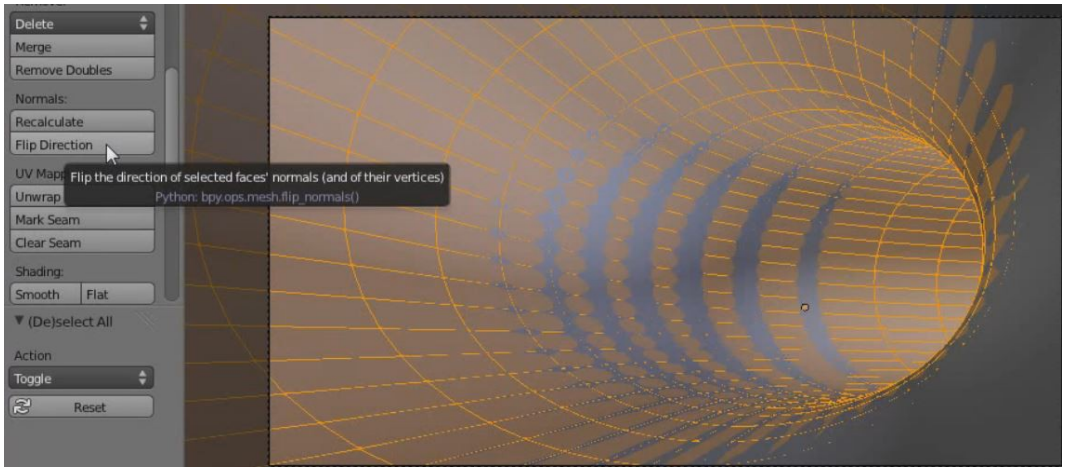

*Passaggio a mesh*

Per ottenere un buon *Displacement* dobbiamo aumentare il numero di vertici; in più, vogliamo un risultato levigato, quindi aggiungiamo un modificatore Subdivision Surface, con due suddivisioni, all'oggetto.




*Utilizzo del modificatore Subsurf*

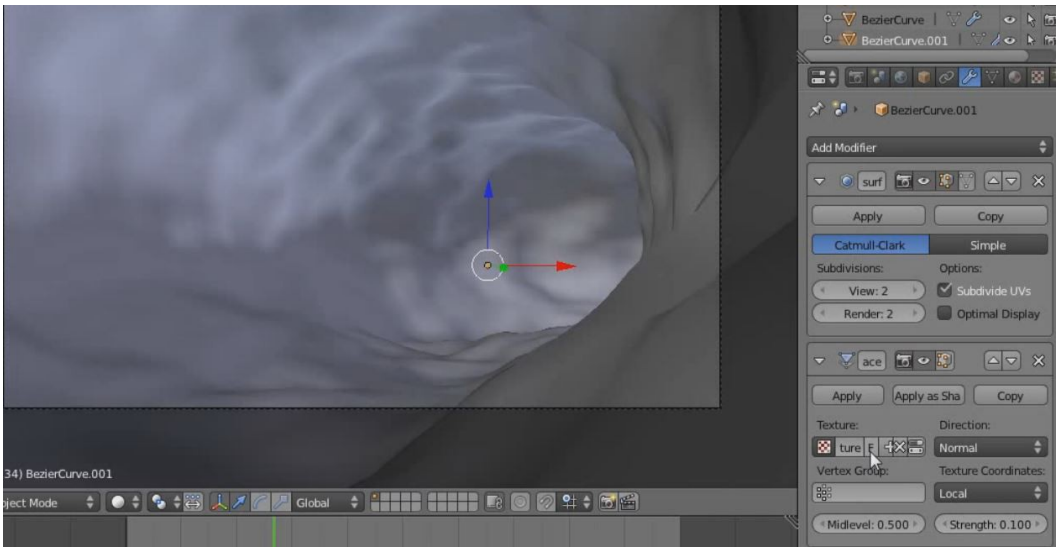
Prima di proseguire, passiamo in Edit Mode, selezioniamo tutto e, nella Tool Shelf, clicchiamo su *Flip Direction* per invertire le Normali alle superfici che, di default, per un oggetto del genere vengono create verso fuori, mentre noi qui stiamo inquadrando l'interno.




*Flip Direction per le Normali (Edit Mode, tutto selezionato)*

Torniamo in Object Mode e aggiungiamo un modificatore *Displace*, impostando magari un valore basso per la Strength, ad esempio 0.1, comunque potremo modificarlo interattivamente in seguito.

Nel pannello del modificatore, clicchiamo su *New* per aggiungere una Texture di *Displacement* e, dopo, sull'icona *Show Texture* in Texture Tab, per impostare i parametri della Texture (ovviamente, potete personalizzare questi valori a piacere).




*Creazione (e utilizzo) del modificatore Displace, provvisto di Texture*

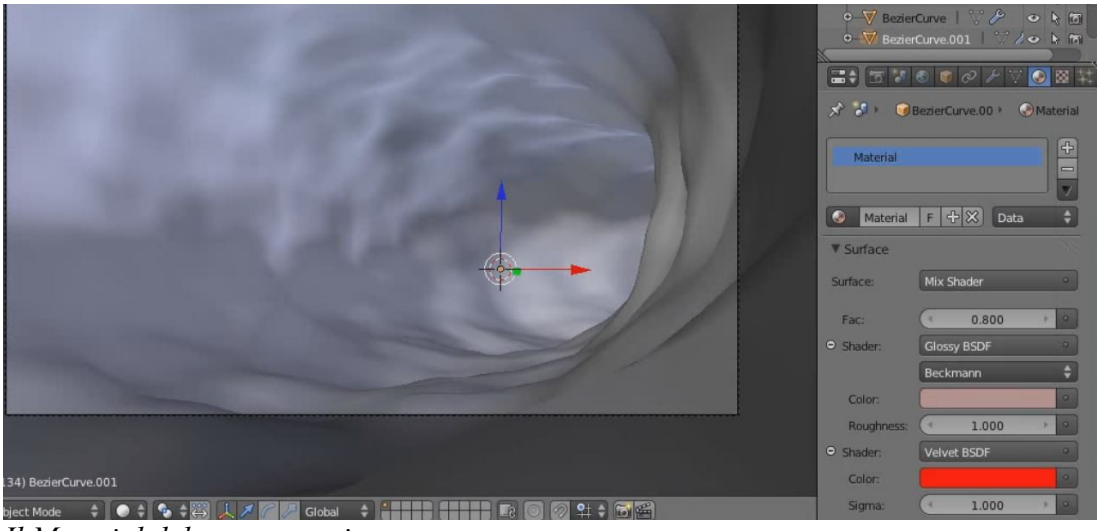
Come tipo, *Clouds* (la Texture creata di default) andrà benissimo, quindi l'importante è variare il parametro Size che, nel mio caso, non è molto diverso da quello di default (0.25).

*Impostare la Texture di Displacement*

Parliamo ora dell'aspetto del vaso sanguigno: apriamo la scheda Materials e aggiungiamo un nuovo Material. Cambiamo lo Shader di default in *Mix*, con Factor 0.8, quindi più sbilanciato verso il secondo Shader.

Impostiamo, come primo Shader, un *Glossy* con colore rosa (ad esempio 0.5, 0.3, 0.3) e Roughness 1; questo darà l'effetto dei riflessi lievi, diffusi e sfocati, contribuendo a creare un effetto flou.

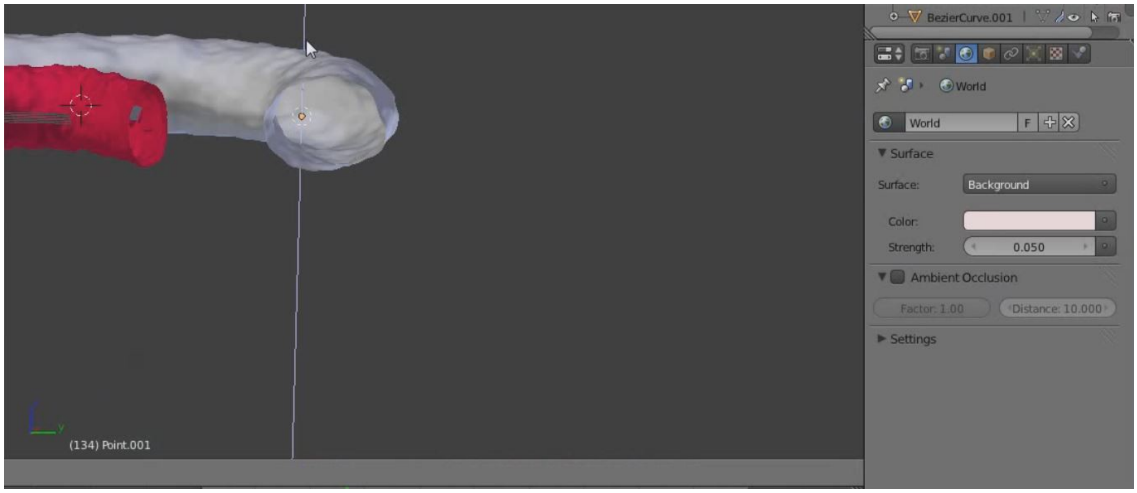
Come secondo Shader, impostiamo *Velvet* con colore rosso puro.



*Il Material del vaso sanguigno*

Con il vaso sanguigno abbiamo finito.

Sul fondo, dietro la "curva", posizioniamo una fonte di luce *Point Light* con intensità elevata, ad esempio 500. Come "backlight", invece, utilizzeremo World, quindi in questa scheda, nella Properties Window, attiviamo i Nodi e impostiamo, per Background, un colore leggermente rosato (ad esempio 0.97, 0.82, 0.82) e un valore molto piccolo per Strength, ad esempio 0.05.



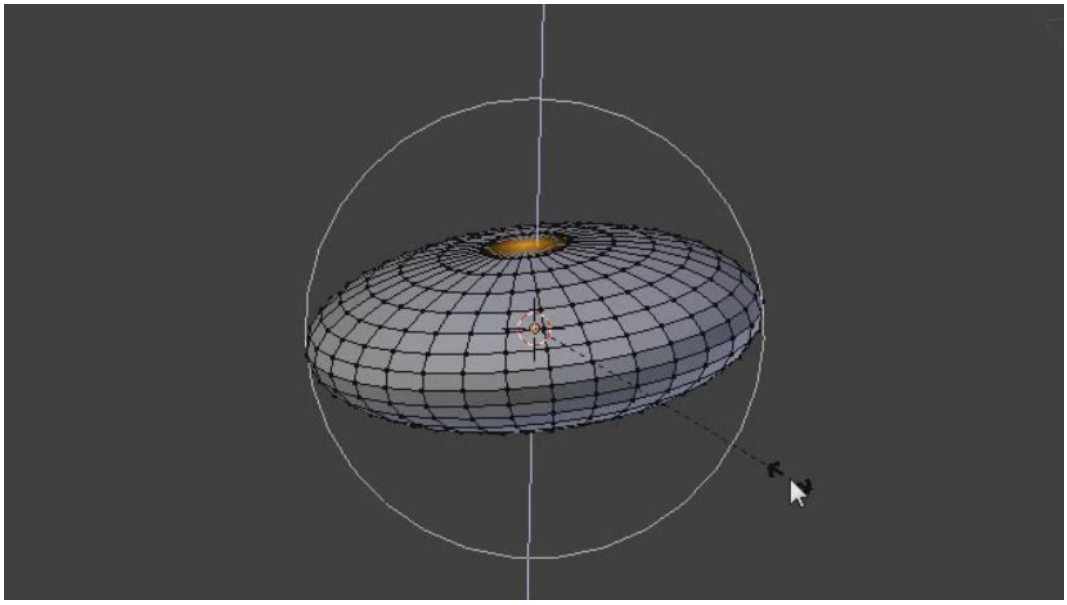

*Impostazioni WORLD, dopo aver posizionato una Point Light*

È giunto il momento di realizzare il prototipo di globulo rosso.

In seguito potrete scegliere se fare delle copie e posizionarle manualmente nel vaso sanguigno, in modo da avere la massima libertà nel bilanciare la composizione, gli orientamenti, eccetera, oppure affidarvi all'emissione di un sistema particellare. Il primo caso è banale, per cui qui vi mostrerò il secondo; prima, però, facciamo il modello di base!

Ci sono vari modi per modellare questo oggetto, forse il più semplice consiste nel seguire questi passi:

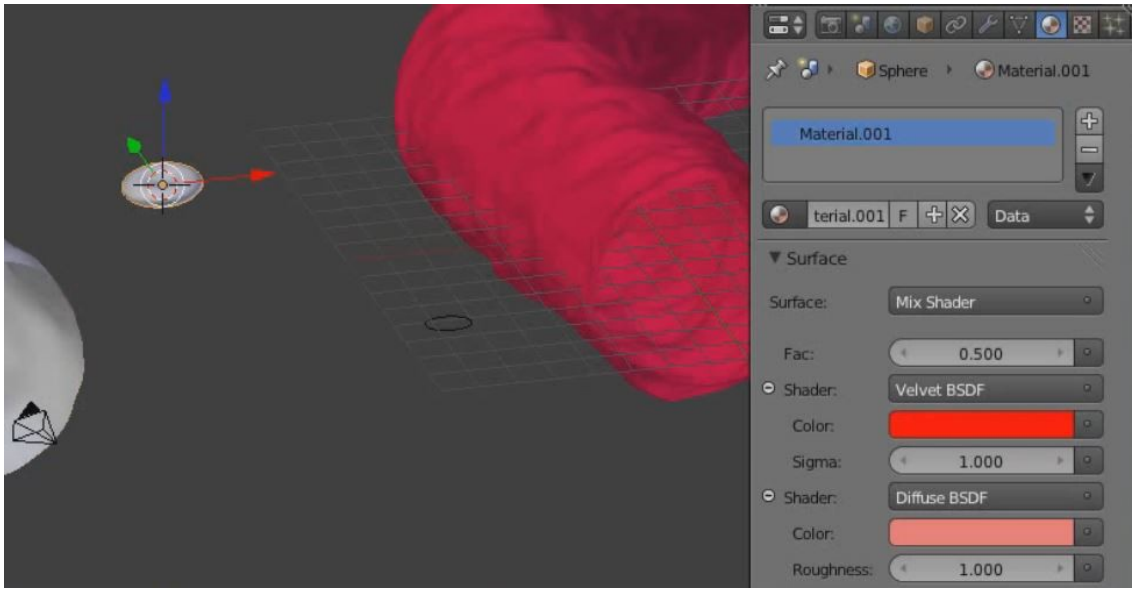
1. aggiungere una UV Sphere alla scena;
2. schiacciare un po' la sfera sull'asse verticale, con S Z;
3. passare in Edit Mode e selezionare i vertici che rappresentano i poli;
4. attivare, nell'header della 3D View, l'Editing Proporzionale (shortcut: O), regolando la grandezza dell'area di influenza con la rotellina del mouse;
5. scalare su Z, in modo da avvicinare i vertici dei poli e, conseguentemente, le zone influenzate dal Proportional Editing.



*Modellazione della sfera utilizzando il Proportional Editing per lo scaling*

Se volete, potete aggiungere un modificatore Displacement, come visto per il vaso sanguigno, in modo da aggiungere un po' di rumore sulla superficie dell'oggetto; in questo caso, vi consiglio di aumentare il numero di vertici con Subdivide o con Subdivision Surface.

Passiamo al Material del globulo rosso: anche qui, un *Mix* Shader, con Factor 0.5.
In questo caso, come primo Shader ho scelto un *Velvet* con colore rosso puro, mentre per il secondo un *Diffuse* con colore 0.9, 0.2, 0.2 e Roughness 1.

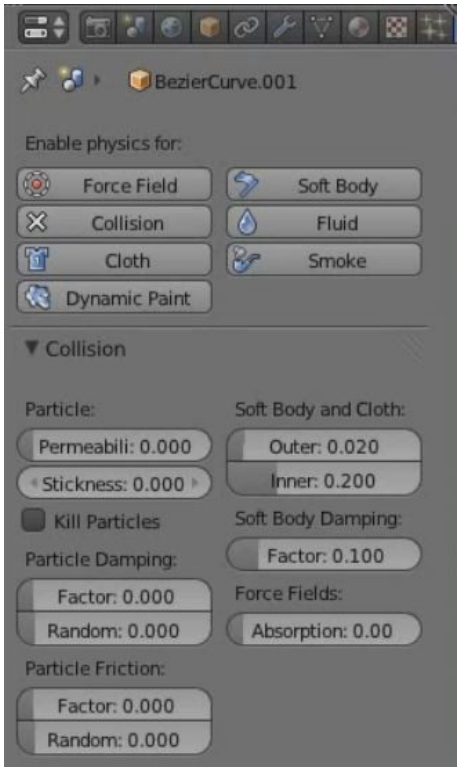



*Il Material per il globulo rosso*

Prima di continuare, vi consiglio di impostare lo Shading *Smooth* (cliccando su *Smooth* nella Tool Shelf con una mesh selezionata) sia per il vaso sanguigno che per il prototipo di globulo rosso.

*Shading SMOOTH per tutti gli oggetti*

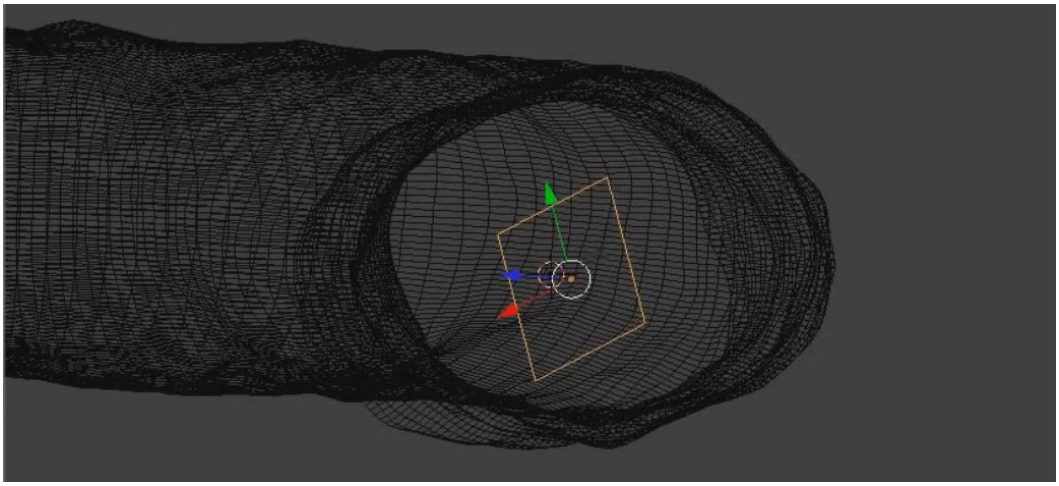
A questo punto, come detto prima, potete scegliere se fare manualmente delle copie dei globuli rossi e posizionarle e ruotarle a piacere, magari dando un'occhiata all'anteprima del risultato con la Viewport *Rendered* messa a disposizione da Cycles, oppure affidarvi a un sistema particellare.




*Physics Collision*

Parliamo brevemente del secondo caso.
Prima di procedere, selezioniamo il vaso sanguigno e, nella scheda *Physics*, all'interno della Properties Window, clicchiamo su *Collision* per abilitare la collisione con le particelle.

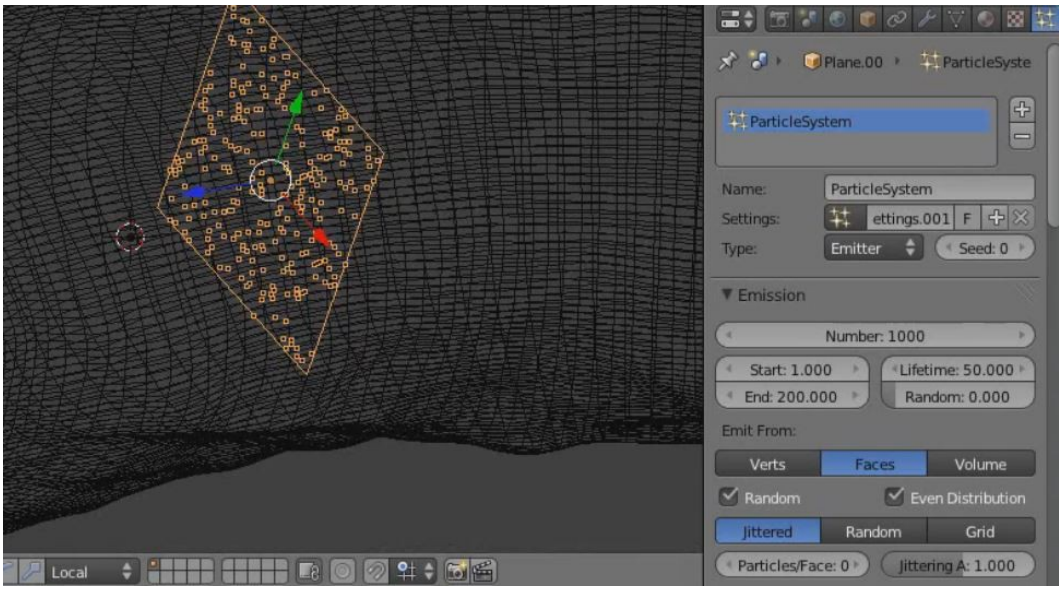
Adesso aggiungiamo un Plane dietro la fonte di luce, facendo attenzione a posizionare la normale positiva (identificata di default con l'asse Z locale positivo) nella direzione della telecamera.



*Creazione e posizionamento del Plane emettitore*

Con questo oggetto selezionato, apriamo la scheda *Particle Systems* nella Properties Window e aggiungiamo un sistema particellare *Emitter*.

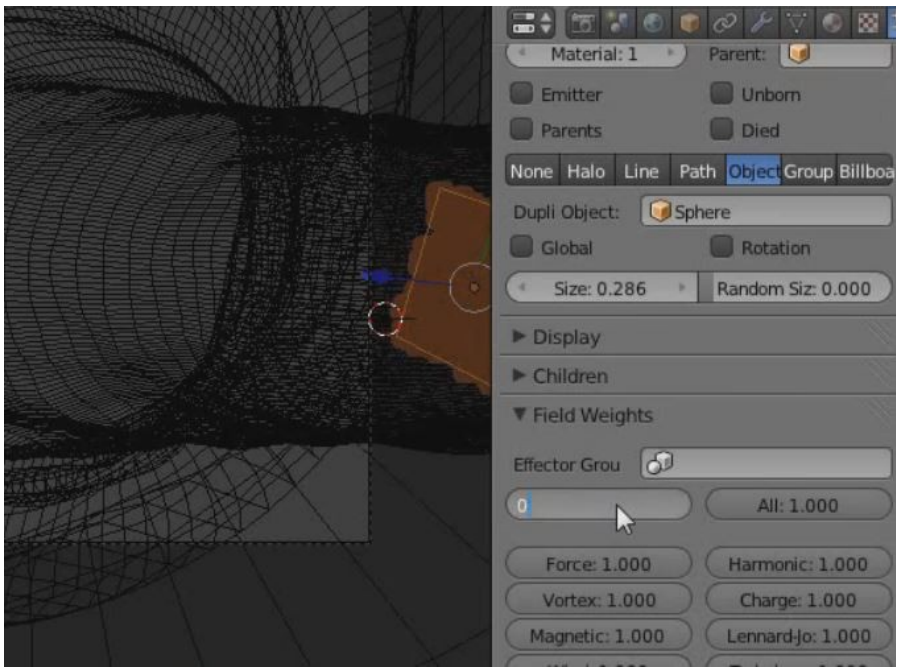
Nella sezione Render del sistema particellare deselezioniamo *Emitter*, selezionando poi Object e impostando, nel campo che apparirà, il nome del prototipo di globulo rosso.




*Impostazioni del Particle System Emitter del Plane*

Se i nuovi oggetti sono troppo grossi, variamo la *Render Size* in questa scheda; lo scaling sul prototipo avrà effetto sulle particelle solo se, dopo, applicheremo le trasformazioni con CTRL A e scegliendo Scale dal menù che apparirà.

Nella sezione *Force Field*, portiamo *Gravity* a 0, per far fluttuare liberamente le particelle.

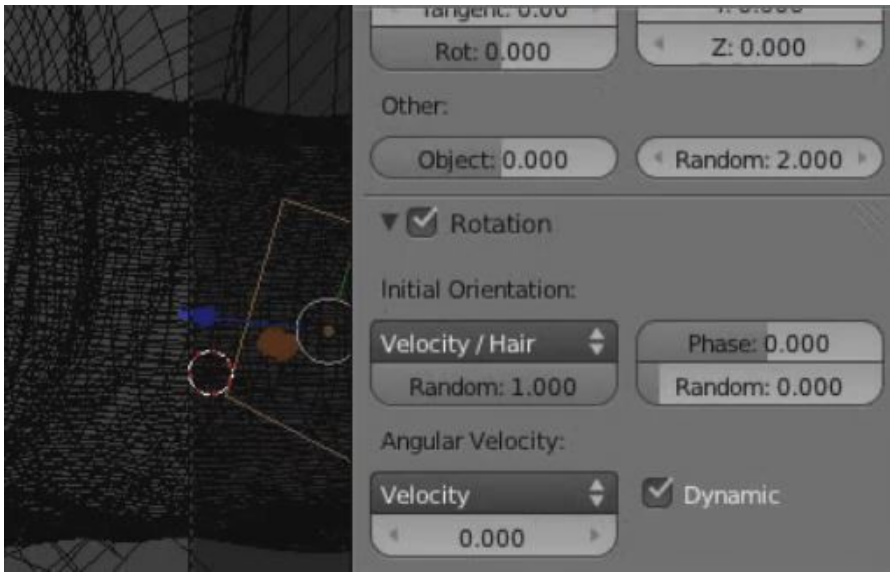



*Field Weight GRAVITY a 0*

Nella sezione *Emission*, impostiamo un valore basso per *Amount* e per *End* (ad esempio io ho messo 30 particelle da emettere in 60 frames); aumentate invece quello di *Lifetime* (e, se necessario, l'intera durata del videoclip, dipende dalla velocità delle particelle).

In *Velocity,* impostiamo Normal (la velocità di emissione) e Random (la casualità) a 2.0.

Selezioniamo la sezione *Rotation* e, all'interno, selezioniamo anche *Dynamic* e portiamo *Initial Velocity Random* a 1.0.

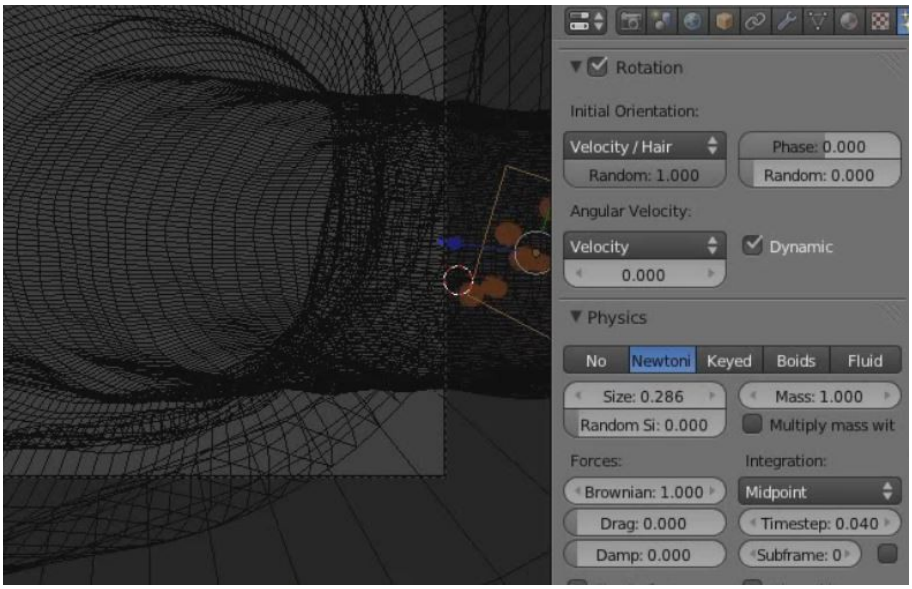



*Velocity, Rotation, Dynamic*

In *Physics*, portiamo *Brownian* a 1.0, per aumentare ancora la casualità nell'emissione.

Non ci resta che posizionarci in vista telecamera, premere ALT A, e... incrociare le dita!

La generazione delle particelle è infatti casuale (ma potremo scegliere tra diversi *Seed* nella scheda dell'Emitter, in Particle Systems), inoltre potrebbe esserci qualche problema con le collisioni... se non dovete fare proprio un'animazione ma un rendering statico, quindi, molto probabilmente andrà meglio la prima soluzione, ossia il posizionamento manuale delle copie.




*Il Plane sta emettendo le particelle*

*  *  *

# La scritta sul muro: Dynamic Paint e Nodes Compositing

In questo tutorial vedremo come generare l'animazione della "scritta sul muro" ("The writing on the wall"), in riferimento al celebre quadro di Rembrandt "la festa di Baldassarre", con il motore di rendering interno di Blender 3D 2.63; in particolare, vedremo come generare la sequenza di immagini della composizione della scritta sul muro mediante Dynamic Paint, come fare il mapping di tale animazione su un altro oggetto e come aggiungere, mediante i Nodes del Compositing in post-produzione, un po' di bagliore con Glare e Blur.

Iniziamo con una scena completamente vuota per trattare la prima fase: la scritta mediante Dynamic Paint. Una delle caratteristiche più interessanti del Dynamic Painting è la possibilità di salvare i disegni effettuati sugli oggetti o sugli stessi oggetti (modalità *Vertex*) oppure, con la modalità che utilizzeremo noi in questo tutorial, in *Images*, ossia sequenze di file immagine, in formato PNG o EXR, da memorizzare su disco.

I files così ottenuti sono immagini con canale Alpha (opacità), completamente trasparenti tranne nelle zone "dipinte", appunto, con i "pennelli" del Dynamic Painting, che poi sono oggetti o particelle che toccano gli oggetti-tela, detti Canvas.

Come brushes (ossia "pennelli") utilizzeremo tre sfere, che lasceranno una traccia nei punti di contatto con il Plane utilizzato come Canvas (ossia "tela"); questo effetto non si vedrà né nella 3D View, in tempo reale, né in un rendering avviato al volo, ma verrà salvato, come detto, in una sequenza di immagini su disco.

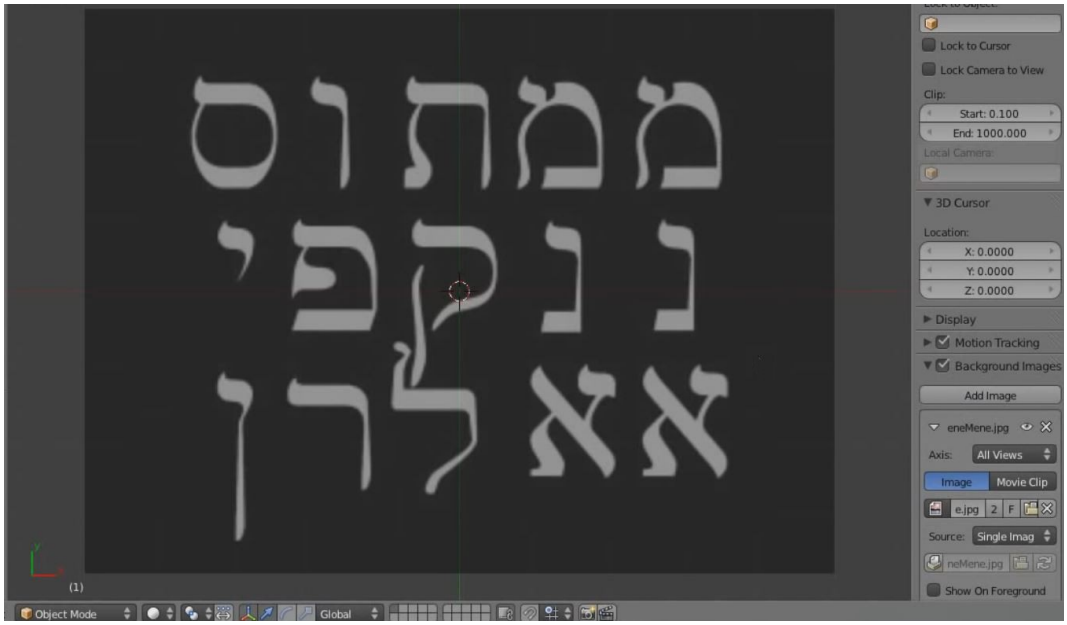
Questo ci consente di realizzare l'animazione della scritta a parte, su un piano con dimensioni simili a quelle della Background Image (che utilizzeremo per ricalcare il tracciato) e utilizzare poi la sequenza su qualsiasi altro oggetto, anche complesso, mediate mappatura UV-Mapping.

Trattandosi di una Texture, potremo utilizzare la sequenza su più canali, infatti in questo tutorial vedremo come utilizzarla per la colorazione dell'oggetto e per l'emissione luminosa sul canale Emit da isolare in fase di Nodes Compositing.

In una scena completamente vuota, quindi, posizioniamo il cursore 3D al centro della scena, con SHIFT S e "*Cursor to Center*", in modo da posizionare gli oggetti che creeremo al centro della scena.

Passiamo in vista TOP Ortho, cioè dall'alto ortogonale, in modo da utilizzare un'immagine di sfondo (Background Image) con il tracciato da ricalcare.

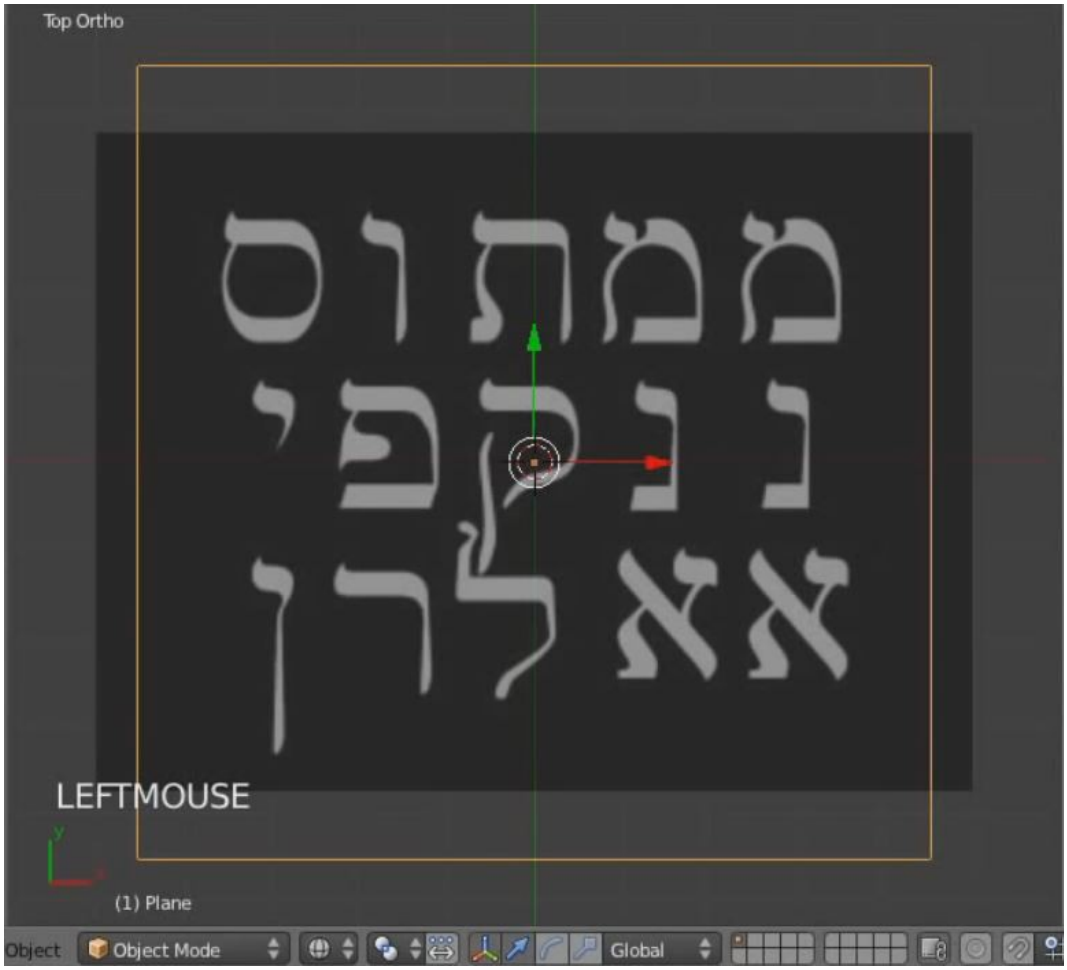
Apriamo la Transform Window della 3D View, con *N*, e aggiungiamo una Background Image, caricando un'immagine che rappresenta la scritta da riprodurre; per trovare la scritta del quadro vi basta cercare, come ho fatto io, "*Mene mene*" su Google immagini, mentre se volete fare altre scritte, ad esempio la vostra firma, dovrete magari disegnarle (ad esempio con GIMP o con un software di disegno vettoriale come Inkscape) o passarle allo scanner, prenderle da una fotografia, eccetera.



*Importare la Background Image di riferimento*

Passiamo in modalità di visualizzazione Wireframe.

Aggiungiamo un Plane, scalandolo in modo di farci entrare la scritta dell'immagine di Background lasciando anche un po' di margine (oppure ridimensioniamo l'immagine di Background con gli strumenti nel relativo pannello, nella Transform Window).



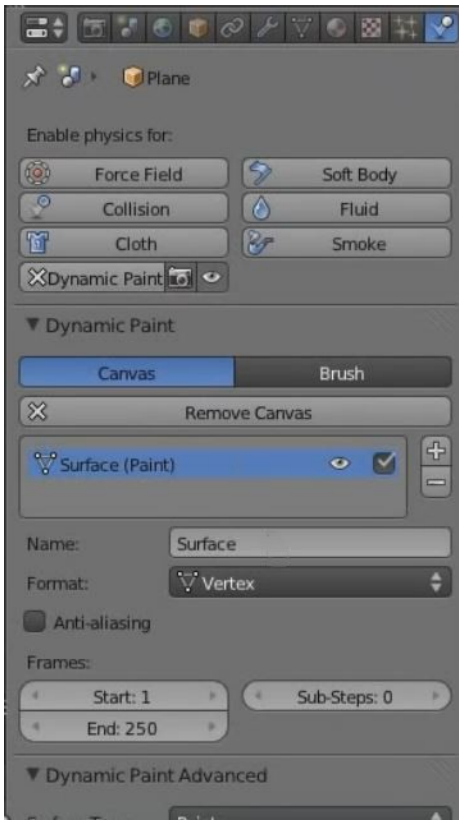
*Il Plane, scalato, in modalità Wireframe*

Con il Plane selezionato andiamo nella scheda *Physics*, nella Properties Window, e scegliamo *Dynamic Paint*.

Per prima cosa dobbiamo scegliere se utilizzare il Plane come Canvas, ossia come tela, o come Brush, pennello; scegliamo, ovviamente, Canvas.

Nella sezione *Format*, cambiamo la modalità da Vertex (default) a *Image Sequence*.

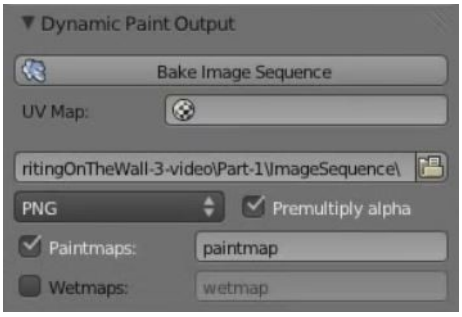
In Start ed End bisogna scrivere la durata dell'animazione in frames; nel mio caso sto lasciando i valori di default, 1 e 250, perché penso che 10 secondi siano più che sufficienti per generare le scritte (tenendo conto che faremo le righe in parallelo, come vi mostrerò a breve), comunque ricordate che questa non è la durata del videoclip finale ma solo della composizione della scritta (e, in seguito, vedremo come farla partire a videoclip iniziato, non al primo frame del rendering finale).




*Prime impostazioni di Dynamic Paint in Physics*

Apriamo la sezione *Dynamic Paint Output*, specificando poi nel campo *Directory to Save Texture* il percorso ove salvare la sequenza di file immagine da generare; consiglio di creare una sottocartella, chiamandola ad esempio "*Image Sequence*", all'interno della cartella principale del progetto.

Il formato PNG, con *Premultiply Alpha*, andrà più che bene.

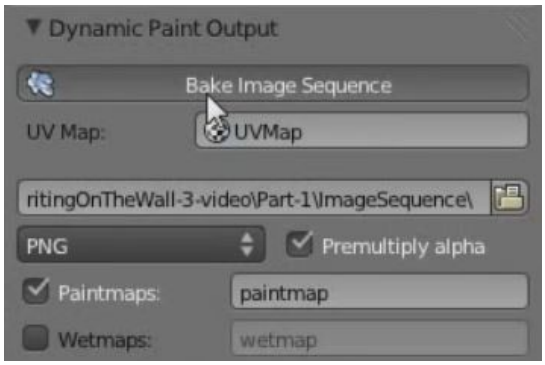



*Altre impostazioni in Dynamic Paint Output*

Prima di poter fare il *Bake* del Dynamic Paint, dobbiamo ovviamente impostare i brushes ma, per quanto riguarda la Canvas, dobbiamo anche generare un Layer UV, anche di base, quindi andiamo nella 3D View, passiamo in Edit Mode, selezioniamo tutto e premiamo *U*, scegliendo ad esempio *Project from View.*

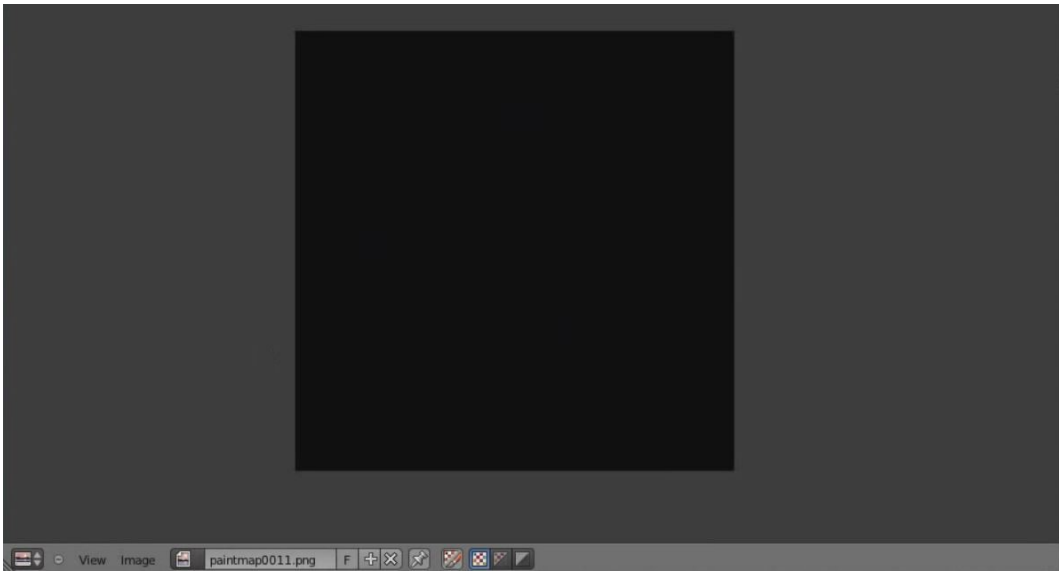
*Creazione della scucitura UV*

A questo punto torniamo in Object Mode, torniamo nella scheda di Dynamic Paint nella Properties Window, impostiamo il Layer UV-Map creato automaticamente con l'unwrap nel campo *UV-Map* e, come test, facciamo click su *Bake*.



*Click su BAKE, dopo aver specificato il Layer UV*

In questo momento non ci sono pennelli, quindi il *Bake* durerà pochissimo e, aprendo una delle immagini generate e salvate nella cartella specificata prima, vedremo un'immagine completamente nera, che in realtà è un'immagine completamente trasparente, come si può notare aprendola, anziché con l'Image Editor di Blender, con un programma di visualizzazione immagini o di fotoritocco.



*L'immagine "trasparente" generata*

Spostiamoci nella 3D View, con il 3D Cursor al centro; l'importante, in questa fase, è mantenere la coordinata Z (l'altezza globale) del 3D Cursor e dei punti di controllo che andremo a creare, alla stessa quota del Plane-Canvas, ossia (in questo caso) 0.0.

Aggiungiamo una Bezier Curve o una curva Path alla scena, andiamo in Edit Mode e spostiamo i punti di controllo in modo tale da ricalcare il tracciato; ovviamente dovremo aggiungerne di nuovi per farlo, cosa che possiamo fare selezionando un'estremità della curva e premendo E (estrusione).

*Inserimento della Bezier Curve o della PATH nella scena*

Questa è la parte noiosa del lavoro, comunque ci sono un paio di considerazioni da fare...

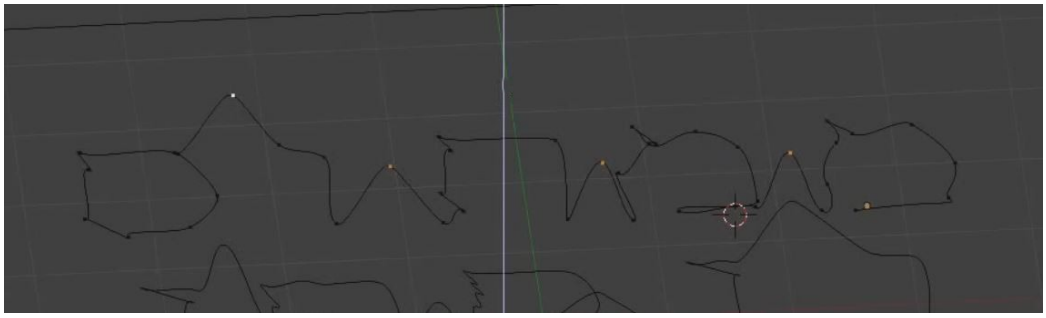
*"Disegnare" la scritta con la Curva in Edit Mode*

Il primo punto è: ricordatevi di mantenere, per ora, la coordinata Z a 0.0 per tutti i punti di controllo; se, in seguito, noterete di aver spostato per errore un control point su Z, potrete riposizionarlo manualmente o, se sono più di uno, selezionarli tutti, premere S Z 0 per appiattire la curva con uno scaling sull'asse Z, poi riportate il tutto alla coordinata Z 0 globale.

Fanno eccezione alcuni punti particolari: ecco perché non possiamo impostare il tipo di curva in 2D nella scheda Object Data, ma dobbiamo lasciare la modalità 3D.

Questi punti che fanno eccezione sono i punti di spazio, di “stacco” tra un carattere e l'altro... ed ecco quindi un altro punto da seguire: dove ci sono degli stacchi, inserite lì in mezzo un punto di controllo, più due punti di controllo alle estremità.

Il motivo è il seguente: dovremo alzare, sull'asse Z, i punti di controllo sugli spazi vuoti, in modo da alzare le sfere-pennelli, come se stessimo staccando la penna dal foglio per fare uno spazio.



*"Alzare" i punti di controllo in corrispondenza degli "spazi"*

Vi potrà capitare, nel seguire le lettere, di dover tornare due volte sullo stesso tratto: non preoccupatevi, non è un problema.

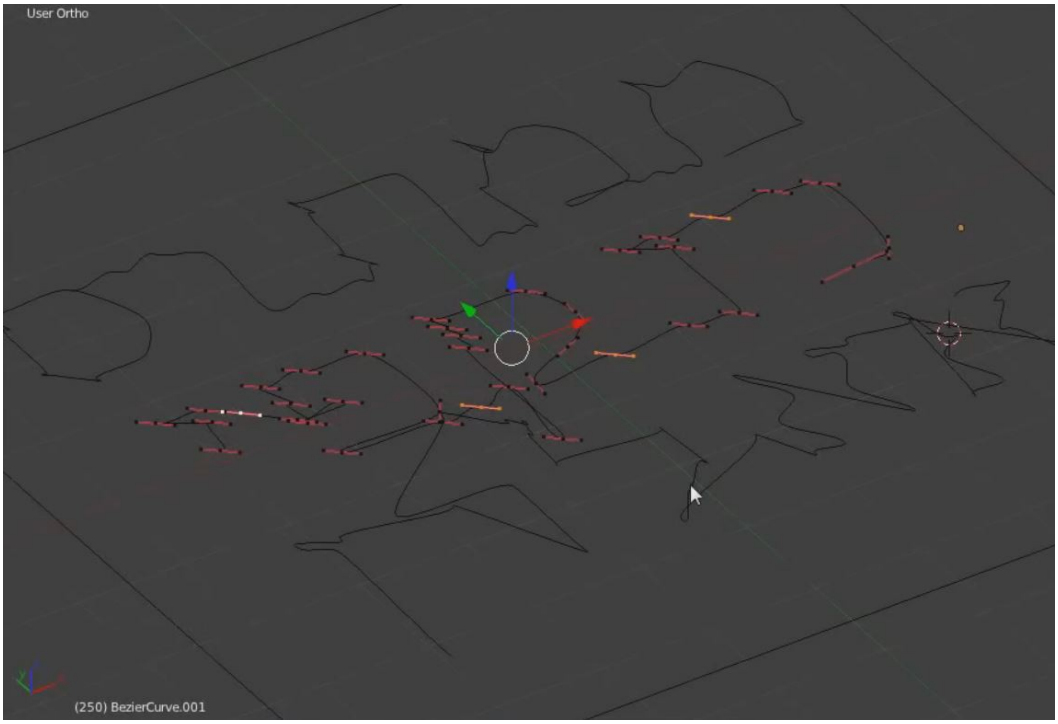
In questo tutorial, per sbrigarmi, non varierò la dimensione delle sfere che utilizzeremo come Brushes, ma vi faccio notare che è possibile riprodurre lo spessore del tratto, che come potete vedere varia anche all'interno di uno stesso carattere, animando lo scaling degli oggetti utilizzati come Brushes, perché l'area dipinta è quella data dal contatto tra il pennello e la tela, quindi sarà più o meno grande a seconda delle dimensioni o della posizione dell'oggetto pennello (e dimensioni e posizioni possono essere animate).

Per questioni di tempo e di effetto visivo, anziché far apparire le lettere una per una ho deciso di far apparire le tre righe in parallelo, da destra a sinistra, per cui bisognerà utilizzare tre DIVERSE Curve per i tracciati e tre DIVERSE sfere da utilizzare come Brush per il disegno, comunque le operazioni da fare in seguito (parenting di una sfera alla sua curva e modifica del valore *Path Animation* per la curva) sono identiche per tutte e tre le coppie curva-sfera.

*La definizione delle tre curve in parallelo*

> *Una piccola curiosità: quello visibile nel quadro originale (e nella scritta che stiamo riproducendo) non è il modo corretto di scrivere in ebraico, in quanto qui le parole sono su colonne, dall'alto il basso, da destra a sinistra (come potete vedere, le due colonne a destra sono identiche, Mene Mene), mentre dovrebbero essere su righe, da destra a sinistra; probabilmente Rembrandt dipinse così le lettere perché nel passo del libro di Daniele si dice che solo Daniele, appunto, era in grado di leggere la scritta... noi resteremo fedeli all'interpretazione di Rembrandt e disegneremo le lettere come ha fatto lui.*

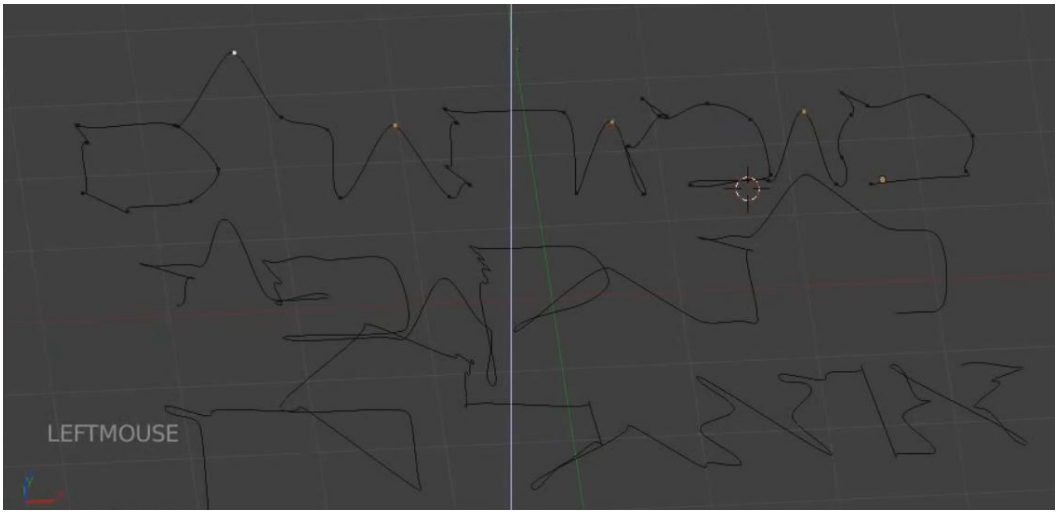
Una volta completati i tre tracciati con tre diverse curve, se tutto è ok vedremo le scritte sul piano XY; inoltre, dovremmo avere un control point in ogni spazio vuoto tra una lettera e l'altra (in ogni caso, è possibile aggiungere punti di controllo anche in un secondo momento, selezionando due punti di controllo contemporaneamente e premendo W – *Subdivide*).




*I tre tracciati completati*

In questa fase può essere utile disattivare la visualizzazione delle normali (le frecce sul percorso della curva) deselezionando "*Normals*" in *Curve Display*, nella Transform Window, con la curva selezionata in Edit Mode.

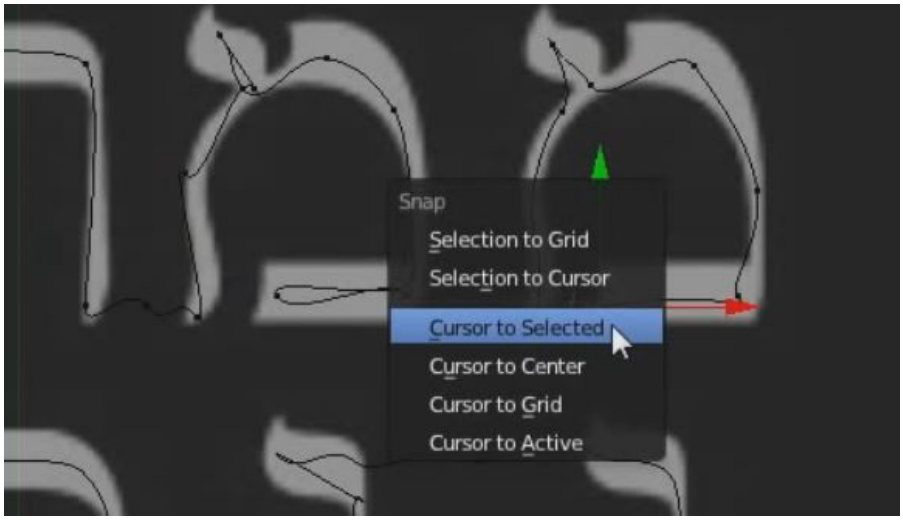
Dobbiamo quindi selezionare i punti di controllo presenti in questi spazi e alzarli un po' sull'asse Z: facciamolo.




*Alzare i punti di controllo negli spazi*

A questo punto dobbiamo inserire i pennelli veri e propri; dirò ora come fare per il primo pennello, poi bisognerà ripetere queste operazioni anche per gli altri due.

Selezioniamo la prima curva, andiamo in Edit Mode e selezioniamo il primo punto di controllo, ossia il punto di partenza nella scrittura.
Spostiamo lì il 3D Cursor con SHIFT S e *Cursor to Select*, quindi torniamo in Object Mode per aggiungere un nuovo oggetto in quel punto.



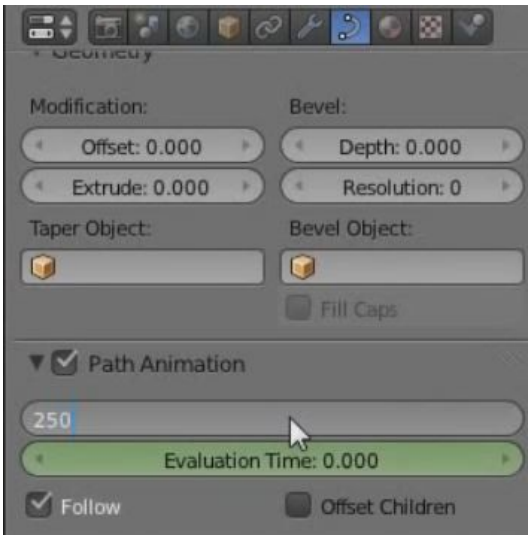

*"Cursor to Selected" nel primo punto di controllo della prima curva*

Aggiungiamo ad esempio una sfera e scaliamola a piacere per regolare la grandezza del tratto (data dalla superficie derivante dall'intersezione della sfera e del piano).

Per generare l'animazione della sfera lungo il percorso dobbiamo selezionare prima la sfera, poi (con una selezione multipla) la curva percorso, premere CTRL P e scegliere, dal menù che apparirà a video, "*Follow Path*".

*"Follow Path" per la sfera lungo la curva*

Avviando ora un'anteprima dell'animazione con ALT A mentre il cursore del mouse è nella 3D View, vedremo la sfera seguire il percorso... ma c'è un problema!



*Impostare la durata del "disegno" con Path Animation Frames*

Il tutto viene fatto con i tempi di default del Follow Path, ossia 100 frames. Per risolvere questo problema, selezioniamo solo la curva percorso, andiamo nella scheda Object Data, nella Properties Window, e cambiamo *Path Animation Frames* da 100 (valore di default) a 250.

Ora l'animazione è corretta.

Come detto precedentemente, dobbiamo ripetere le stesse operazioni con altre due sfere per gli altri due percorsi, in modo da scrivere in parallelo.

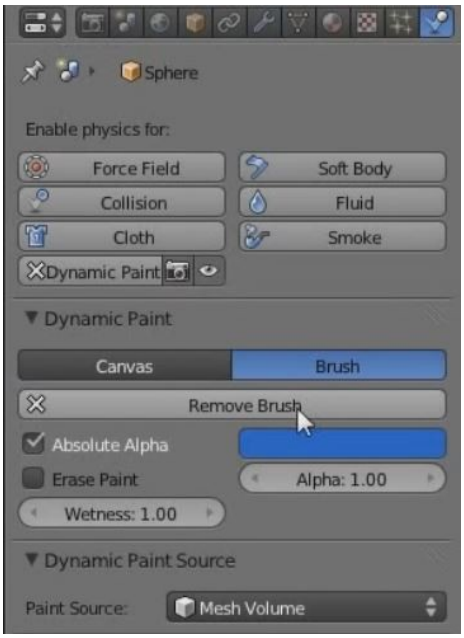
Riepilogando: si mette il 3D Cursor nel primo control point di una curva con SHIFT S e *Cursor to Selected*; si aggiunge lì una sfera, da scalare a piacere; si imparenta la sfera alla curva con la modalità *Follow Path*; si cambia *Path Animation Frames* per la curva da 100 a 250.

*L'animazione delle altre due sfere*

Adesso le sfere seguono il percorso, ma ancora non "sporcano" la nostra tela.

Selezioniamo la prima sfera, andiamo nella scheda *Physics*, nella Properties Window, e clicchiamo su Dynamic Paint, scegliendo come tipo "Brush" e, poi, "Add Brush" (aggiungi pennello).

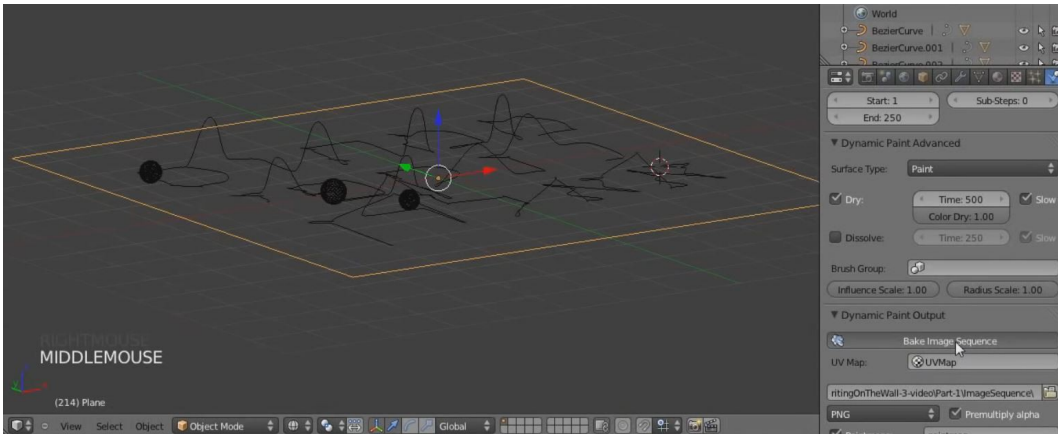
Il colore della scritta che verrà generata dal singolo Brush a contatto con la Canvas va impostato nella Color Box "*Color of the Paint*", qui nella scheda del Brush.



*Impostare le sfere come "Brush"*

Vi consiglio di portarlo a bianco, quindi avremo scritte bianche su sfondo trasparente, perché così potremo utilizzarlo come *Value*, come fattore di intensità di un effetto, non come *Color*, ad esempio per effetti come *Normal* o *Displace;* comunque, poi potremo cambiare facilmente anche il colore con *Influence* nelle Textures o con *Color Balance* in Compositing.

Fatto questo lavoro per tutte e tre le sfere, abbiamo quasi finito con questa prima fase; adesso, dobbiamo selezionare il Plane-Canvas e fare il *Bake* della simulazione Dynamic Paint, generando la sequenza di immagini come visto prima.



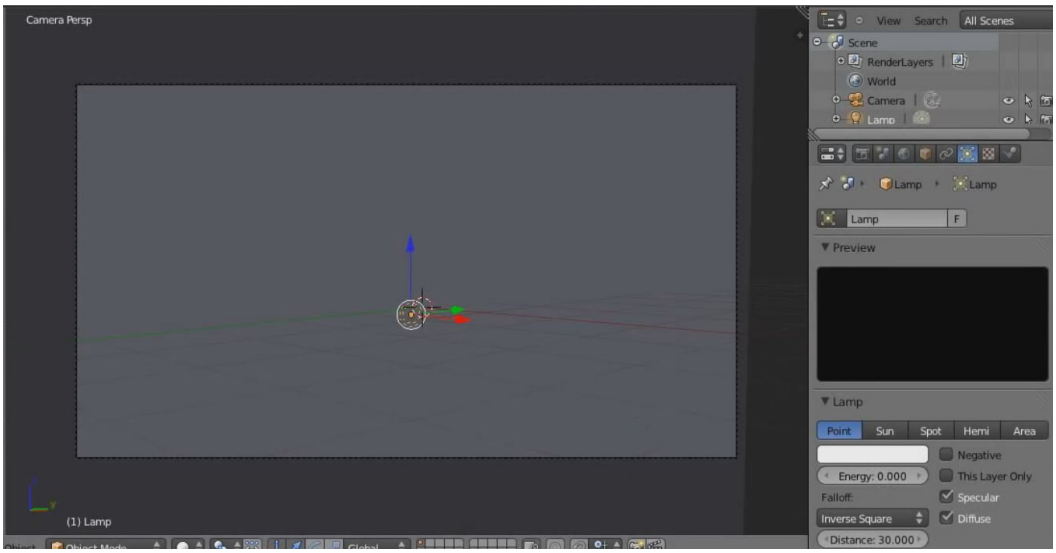
*Effettuare il Bake del Plane-Canvas*

Terminata anche questa operazione, possiamo chiudere questo progetto (magari dopo averlo salvato, ad esempio per modificare il tratto o i tempi e rifare il *Bake* in seguito); apriamo quindi un nuovo progetto per realizzare il rendering finale.

*  *  *

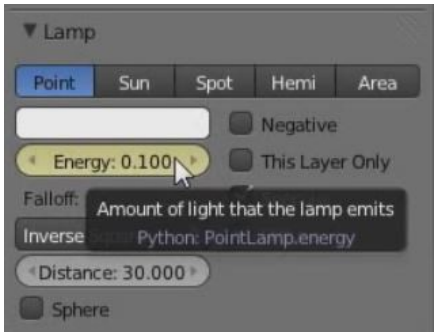
Prima di procedere, stabiliamo la durata del videoclip: voglio lasciare 5 secondi prima e 5 secondi dopo l'apparizione della scritta, che come sappiamo richiede 250 frames, quindi portiamo l'end frame di questo progetto a 500.

Partiamo quindi con una nuova scena, con all'inizio solo la telecamera e una fonte di luce.
Posizioniamo la telecamera in modo da definire l'inquadratura, quindi aggiungiamo un Plane, non proprio frontalmente ma con un certo angolo rispetto a noi.
Spostiamo la fonte di luce vicino al Plane, dove vogliamo far apparire la scritta.



*Plane e fonte di luce nella nuova scena*

Inizialmente la scena è completamente nera (a proposito: se si vedono porzioni di World Background, cambiamo *Horizon Color* a nero nella scheda World, nella Properties Window), quindi selezioniamo la fonte di luce e impostiamo *Energy* a 0.0 al primo frame, nella scheda Object Data della Lamp.



*Registrazione dei Keyframes per la fonte di luce*

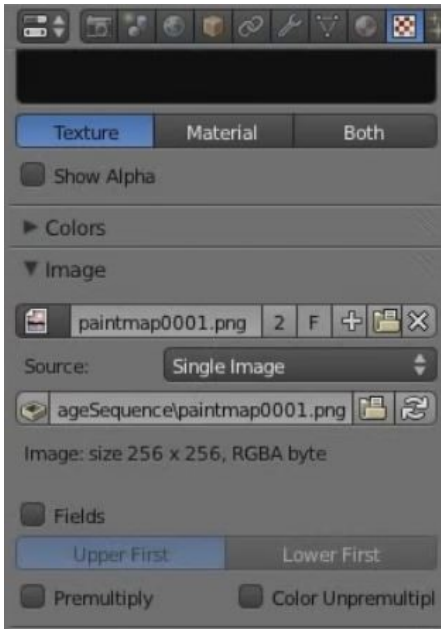
Posizioniamo il cursore del mouse sul campo *Energy* e premiamo *I* per registrare questo valore al frame 1, quindi andiamo al frame 125 (cioè al quinto secondo, quando inizierà a formarsi la scritta), impostiamo *Energy* a 0.1 e, con il cursore del mouse su questo campo, premiamo *I*, in modo da registrare questo valore, per cui l'energia della fonte luminosa andrà da 0 a 1 in 5 secondi, illuminando una porzione di parete.

Sempre nella scheda Object Data della Lamp, disattiviamo *Specular* per evitare la formazione di riflessi speculari sul Plane.

Selezioniamo ora il Plane e dotiamolo di un Material di base, con colore a piacere.

> *NOTA: se volete renderizzare la sola scritta su sfondo trasparente, ad esempio perché volete applicare un background particolare o volete applicarla ad una ripresa reale, magari con Camera Tracking, impostate Z-Transp con Alpha a 0.0 e attivate il canale Alpha nella Texture che stiamo per creare; in questo tutorial, comunque, NON mostrerò questa tecnica.*

Spostiamoci nella scheda Texture e aggiungiamo una Texture di tipo *Image or Movie*.
Clicchiamo su Open, quindi selezioniamo il primo file della sequenza di immagini e, subito dopo, selezioniamo tutte le immagini presenti con A.

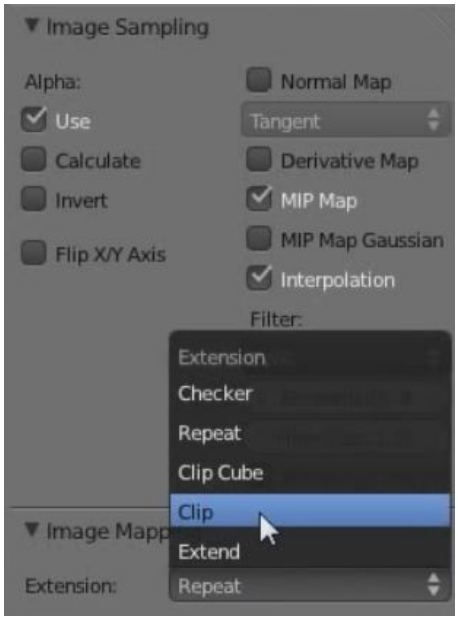


*Impostare la Texture "Image or Movie" per il Plane, caricando la sequenza di immagini ottenute mediante Baking*

Nella scheda *Image*, passiamo da *Single Image* a *Image Sequence*, specificando 250 frames, start 125 (infatti vogliamo far apparire la scritta da questo frame) e selezionando *Premultiply*.

Lasciamo *Use Alpha* in Image Sampling, mentre in Image Mapping cambiamo il tipo da *Repeat* a *Clip*, opzione che imposterà a 0 l'Alpha della Texture nelle zone non coperte dall'immagine in fase di mappatura UV; a proposito di mappatura UV, nella sezione Mapping cambiamo la mappatura da *Generated* a *UV-Map*.

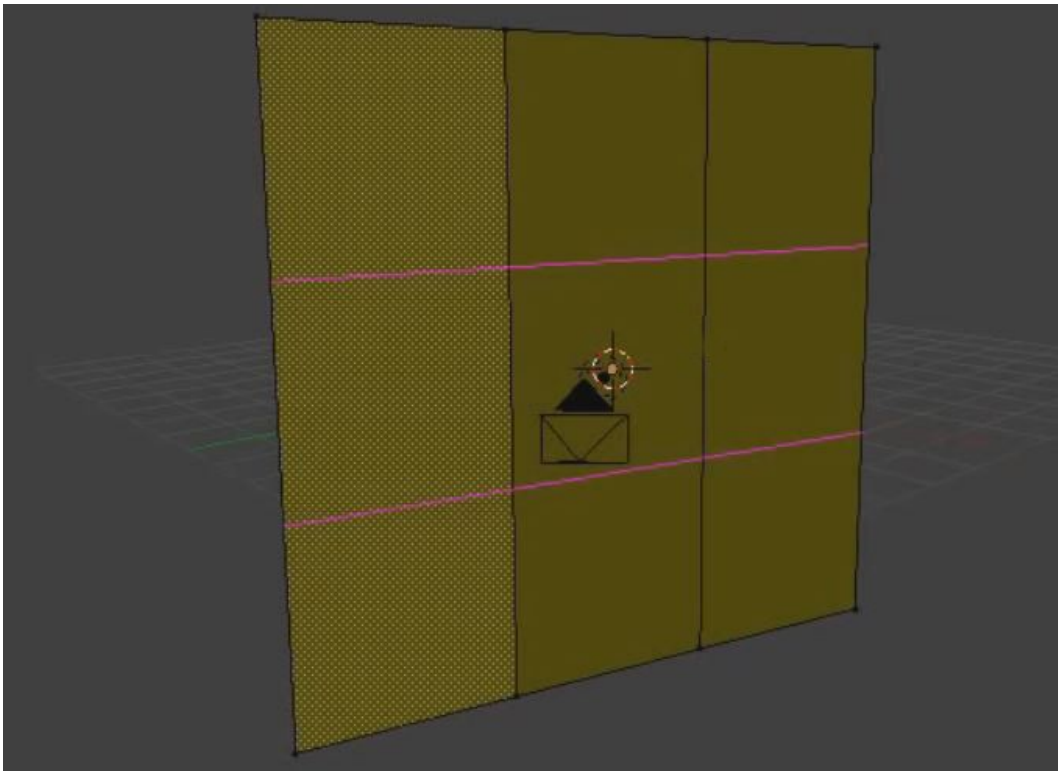
Non c'è ancora un Layer UV perché non abbiamo fatto l'unwrap, ma prima dobbiamo fare un'altra operazione.




*Impostazioni Alpha e Mapping per la sequenza di immagini*

Spostiamoci nella 3D View, selezioniamo il Plane, andiamo in Edit Mode e tramite CTRL R, per Ring Cut, aggiungiamo dei tagli in modo da ricavare un quadrato all'interno del Plane originale, da utilizzare come riferimento per piazzare lì la scritta.

Quando premiamo CTRL R, ruotiamo la rotellina del mouse per fare due tagli; una volta applicato il taglio potremo allontanare i due segmenti a piacere facendo uno scaling di questi due elementi lungo l'asse perpendicolare, quindi ad esempio per allontanare due tagli orizzontali faremo S Z.

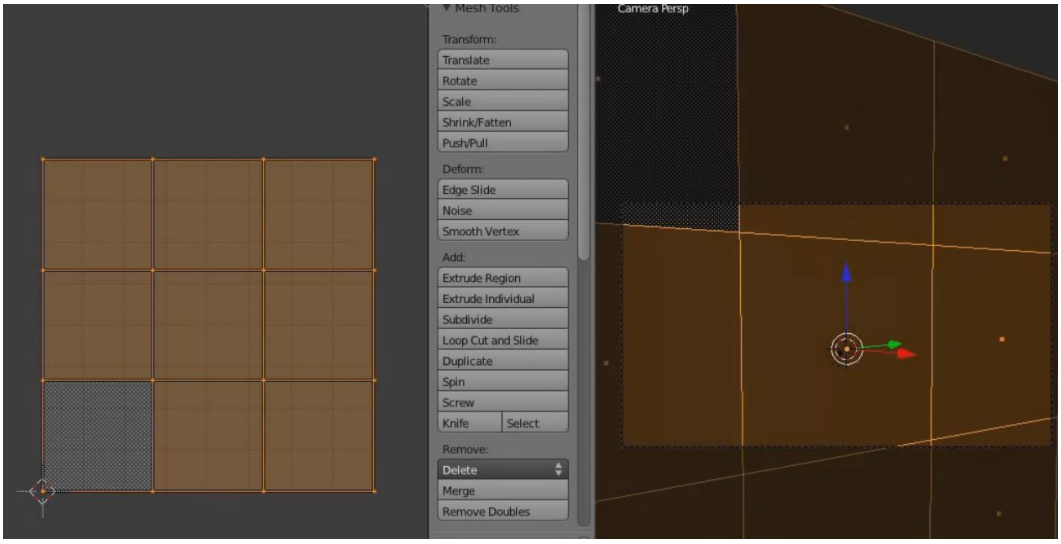


*Realizzare i tagli nel Plane con Ring Cut*

Fatta questa operazione, passiamo allo Screen Layout *UV Editing*.

Nella 3D View, passiamo alla visualizzazione Textured e, in Edit Mode, selezioniamo tutto con *A*, premiamo *U* per Unwrap e scegliamo la prima voce, *Unwrap*.

Nella finestra Image Editor carichiamo, sotto il Layer UV, una delle immagini della sequenza, in modo da ruotare e scalare a piacere la scucitura UV sull'immagine, per effettuare la mappatura.

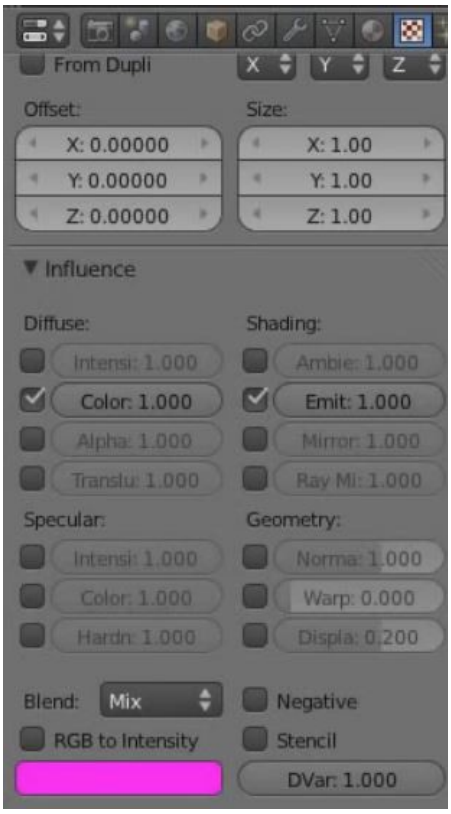



*Realizzazione di Unwrap e mappatura*

Quando abbiamo finito, nella scheda Textures del Plane impostiamo il Layer UV nel campo *Map* per la mappatura UV, quindi torniamo in Object Mode.

Nella sezione *Influence* della Texture lasciamo selezionato *Diffuse Color*, quindi selezioniamo anche *Emit* con intensità 1 (*Emit* del Material è 0, quindi l'intensità dell'emissione sarà regolata interamente dalla Texture).

Se volete cambiare il colore della scritta, che nell'immagine è bianca, attivate la *Color Ramp* della Texture e cambiate il colore della banda all'estremità destra, che di default è bianca, senza modificare la banda all'estremità sinistra, che regola le zone trasparenti.




*Impostazioni Influence nella Texture del Plane*

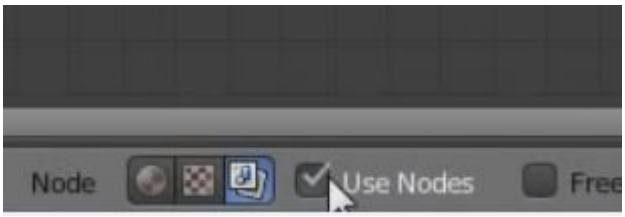
In questo momento, l'animazione della scritta è già presente, come possiamo constatare avviando un rendering veloce (ovviamente in un frame successivo al numero 125), comunque come vi ho anticipato vedremo ora come applicare del bagliore colorato mediante Compositing.

*Casella EMIT nei Passes di rendering*

Per poter ricavare il canale Emit in fase di Compositing e modificarlo, dobbiamo attivare tale opzione nella sezione *Layers* della scheda Render, quindi apriamo tale sezione e selezioniamo la casella "*Emit*".

Adesso possiamo passare ad una finestra Nodes Editor e dedicarci al Compositing.

Selezioniamo la terza icona nell'header, cioè "*Compositing Nodes*", e la casella *Use Nodes*.



*Use Compositing Nodes*

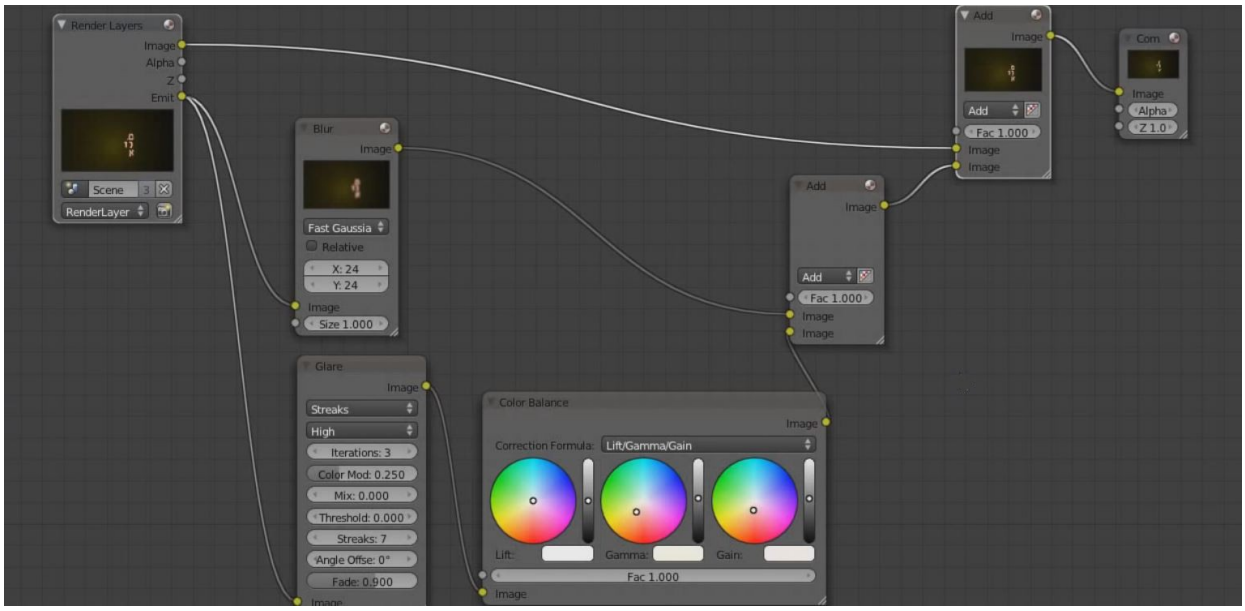
Aggiungiamo un Nodo *Filter Blur*, impostando ad esempio Fast Gaussian con 24 pixel sia per X che per Y, e un Nodo *Filter Glare*, con Threshold a 0, High quality, 5 o più Streaks.

Colleghiamo Emit di *Render Layers* agli input di *Blur* e di *Glare*.

Aggiungiamo due Nodi *Color Mix*, con modalità Add, la somma, e Factor 1.

Mixiamo sul primo Nodo *Add* i due output di *Glare* e *Blur;* se volete, tra Glare e Mix Add (oppure tra Blur e Mix Add) potete aggiungere anche un Nodo *Color Balance*, in modo da cambiare la tonalità degli effetti, per utilizzare un colore diverso dalla scritta per il soft glow o per il Glare.

Mixiamo l'Image di *Render Layers* e l'output del primo Nodo *Add* sul secondo nodo *Mix Add*, regolando a piacere il valore di Factor, portandolo ad esempio a 2 per accentuare i bagliori, quindi passiamo il tutto a *Composite*.




*Lo schema finale dei Nodi di Compositing*

*Screenshot del rendering finale (con effetti di Compositing)*

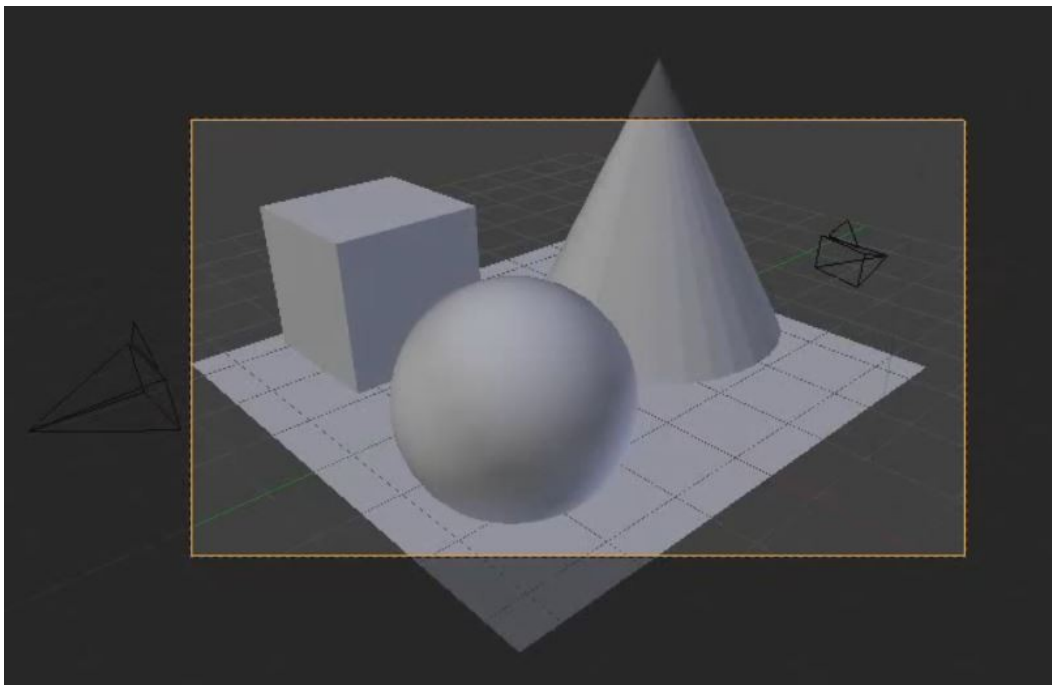
*  *  *

# Cambiare telecamera attiva al volo mediante i Marker

Questo è un brevissimo tutorial su una funzionalità poco nota ma molto utile: l'utilizzo dei *Marker* nella Timeline per cambiare automaticamente la telecamera attiva quando si realizza un'animazione in Blender 3D.

A volte infatti si ha la necessità di renderizzare una scena cambiando l'inquadratura nel corso del videoclip, magari con telecamere in movimento; la telecamera che viene utilizzata per il rendering è detta attiva, è specificata nel campo Camera nella scheda Scene all'interno della Properties Window e può essere cambiata al volo con dei *Marker*, come vedremo.

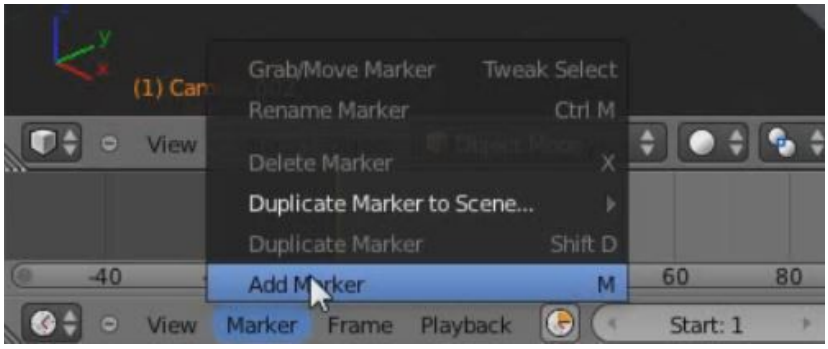
La scena che sto utilizzando contiene tre telecamere, ciascuna delle quali dotata di una propria animazione, ma ovviamente l'esempio può essere esteso ad un numero qualsiasi di oggetti Camera.



*La scena utilizzata per l'esempio*

La prima cosa da fare è scegliere i fotogrammi ai quali si desidera cambiare l'inquadratura (ossia la Camera attiva) inserendo dei *Marker*; questa operazione va svolta nella Timeline e non riguarda gli oggetti Camera.

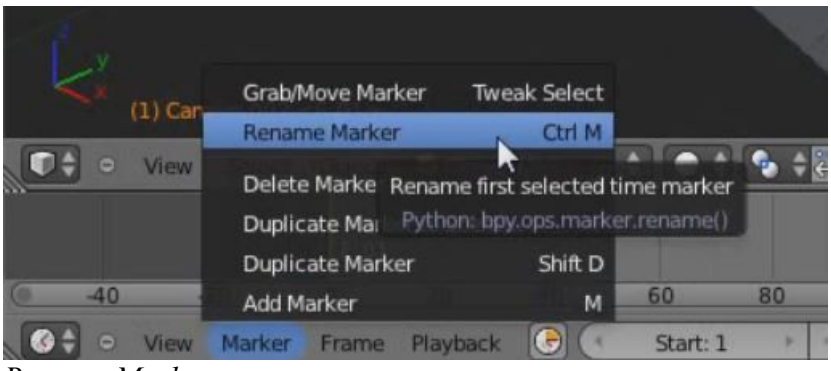
Mettiamo il primo *Marker* proprio al frame 1 del videoclip; l'operazione è semplicissima: basta aprire il menù *Marker*, nella Timeline, e scegliere *Add Marker*.




*Add Marker, nella Timeline*

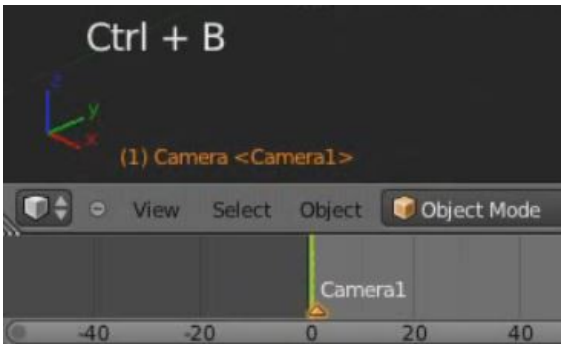
Lo shortcut da tastiera è il tasto *M*, ma attenzione: va premuto mentre il cursore del mouse è nella finestra Timeline, altrimenti svolgerà operazioni diverse a seconda della finestra attiva (ad esempio, nella 3D View, il tasto *M* serve per spostare la selezione da un Layer ad un altro).

Il nome di default per i *Marker* è *F_* e il numero del frame, ma possiamo cambiarlo per utilizzare un nome più significativo; con la Timeline al frame 1, quindi, apriamo di nuovo il menù *Marker*, scegliamo *Rename Marker* e digitiamo, ad esempio, "*Camera1*" nel campo che apparirà a video (anche se a volte può essere utile lasciare il numero del frame nel nome per capire meglio qual è quello marcato).




*Rename Marker*

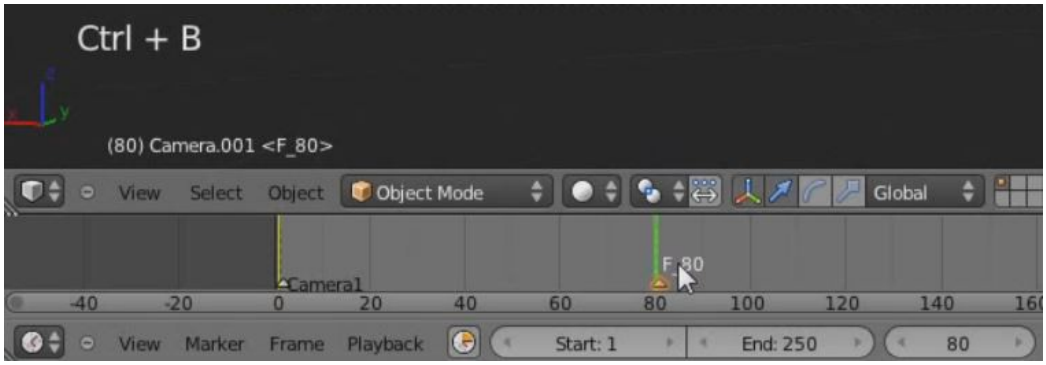
Adesso dobbiamo legare (*To bind*, da cui lo shortcut CTRL B) la telecamera desiderata al *Marker*: selezioniamo l'oggetto *Camera* da utilizzare e, con il cursore ancora una volta nella Timeline, premiamo CTRL B (questa voce si trova anche nel menù View del menù della Timeline).



*Binding della telecamera al Marker*

Le due operazioni da fare sono quindi queste: creare un *Marker* e premere CTRL B nella Timeline con la telecamera da associare al *Marker* selezionato.

Andiamo ad esempio al frame 80, selezioniamo al seconda telecamera, spostiamoci nella Timeline, inseriamo un *Marker* con *M* e associamogli la telecamera selezionata con CTRL B.



*Il Marker per la seconda telecamera al frame 80*

Andiamo ora, ad esempio, al frame 160 e ripetiamo le operazioni con la terza telecamera, quindi: selezioniamo tale oggetto nella scena, spostiamo il cursore del mouse nella Timeline, inseriamo un *Marker* con *M* e associamogli la telecamera selezionata con CTRL B.

*Il Marker per la terza telecamera al frame 160*

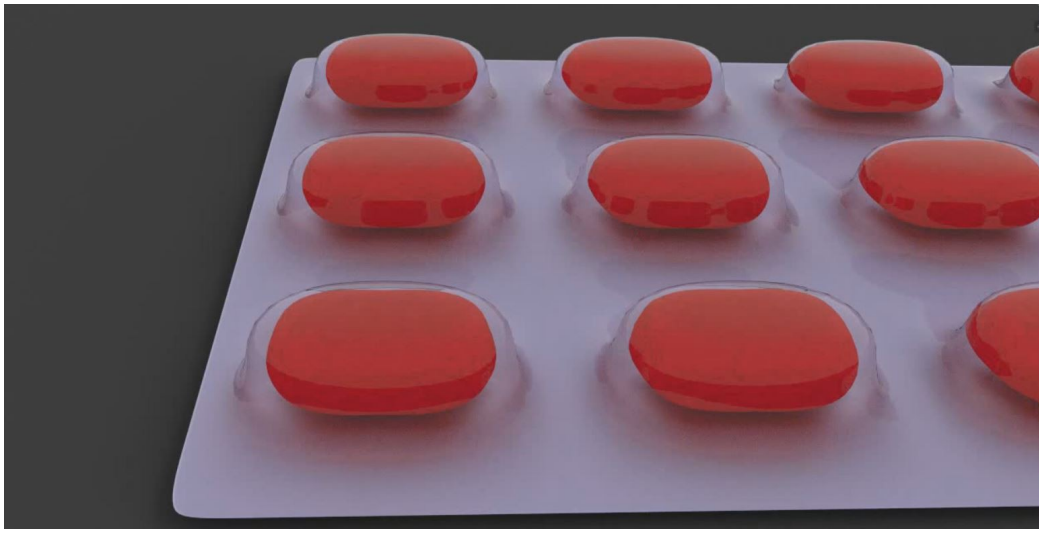
Neanche a dirlo, in seguito potrete cambiare comunque l'associazione *Camera – Marker*, per usare un'altra telecamera ad un dato fotogramma, oppure eliminare proprio il *Marker* e non realizzare quindi il cambio.

Da notare che nella 3D View non vedremo il passaggio di selezione da una telecamera all'altra perché la selezione fa riferimento all'oggetto o agli oggetti della scena attualmente selezionati, siano essi curve, mesh o telecamere; per verificare il passaggio possiamo: saltare da un frame all'altro tenendo d'occhio il campo Camera nella scheda Scene (non fatelo con l'anteprima automatica, con ALT A, perché i campi nella Properties Window non si aggiornano in tempo reale durante l'anteprima dell'animazione, solo andando a salti nella Timeline col tasto sinistro del mouse); l'altro metodo consiste nel posizionarsi nella telecamera attiva con *View – Cameras – Active Camera*, nell'header della 3D View, e poi avviare l'anteprima dell'animazione con ALT A.

*  *  *

# Blender compresse solubili: Array, Subsurf, Mean Crease

In questo breve tutorial vedremo come utilizzare i modificatori Array, Subdivision Surface e la proprietà Mean Crease per realizzare la classica striscia contenente dei medicinali (capsule o compresse) come visibile nel rendering finale di questo tutorial, realizzato mediante Cycles in Blender 3D 2.63.

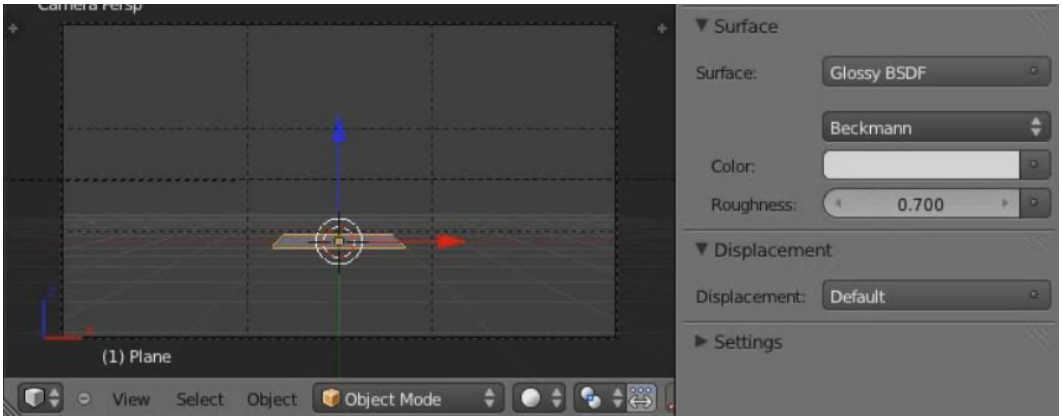


*Il rendering finale di questo tutorial*

Iniziamo da una scena completamente vuota (o, al limite, contenente una telecamera e un paio di fonti di luce Area Lamp, come nel mio caso).

Con il 3D Cursor al centro della scena, aggiungiamo un Plane.

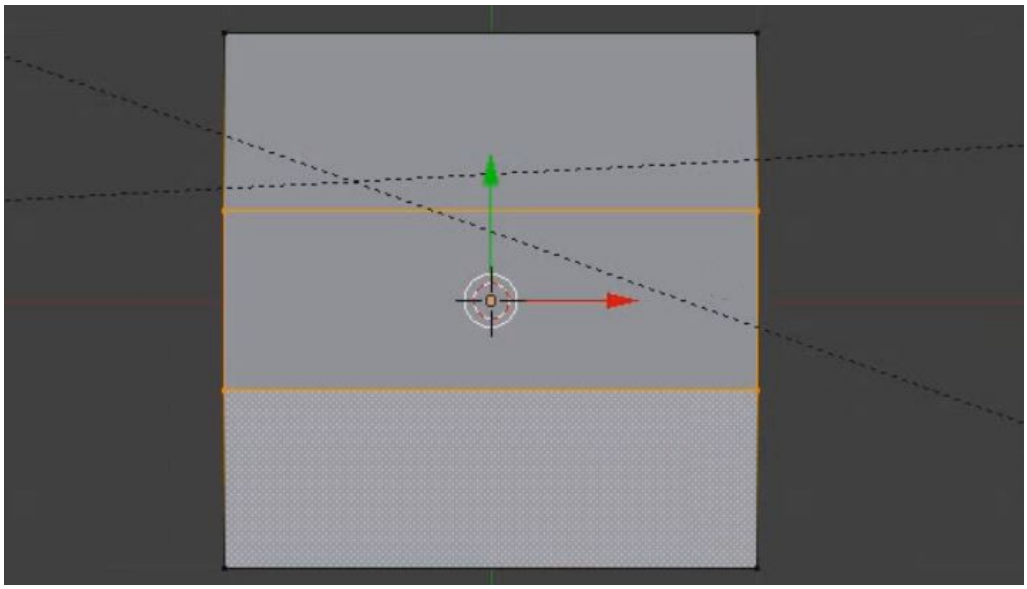
Per portarci un po' avanti col lavoro, dotiamolo subito di un Material con ombreggiatore di base Shader Diffuse o Glossy (ma magari con Roughness alta, per non farlo riflettente "a specchio"), impostando il colore piacere (se volete, potete aggiungere anche una Texture con una scritta rappresentante la marca del farmaco o altri simboli).




*Il Plane con il suo Material*

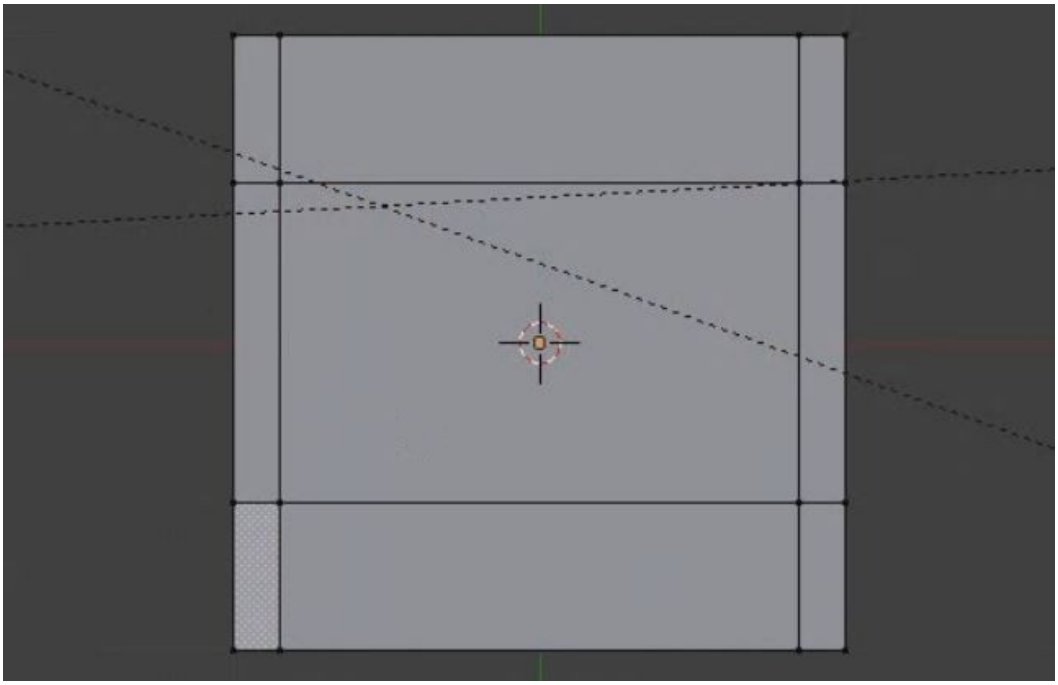
In Edit Mode, premiamo CTRL R e ruotiamo la rotellina del mouse, realizzando due tagli nel verso della lunghezza del Plane.

Quando facciamo click col tasto sinistro per confermare i due tagli, con i nuovi spigoli ancora selezionati facciamo uno scaling solo sull'asse loro perpendicolare, in modo da allontanarli senza però deformare il resto della mesh.



*Inserimento e scaling dei tagli con CTRL R*

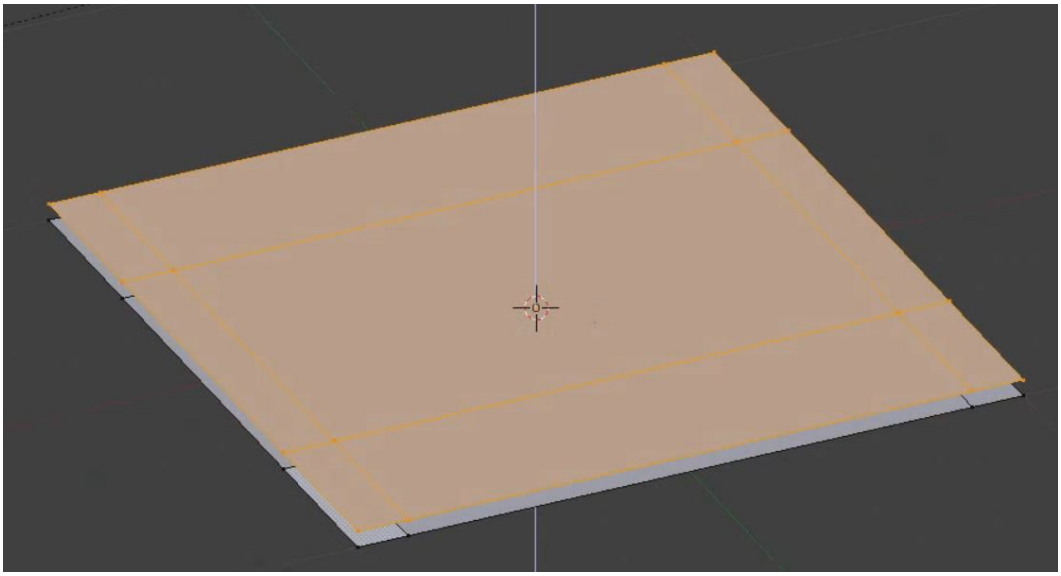
Ripetiamo la stessa operazione dei due tagli con Ring Cut (CTRL R) e lo scaling ma nel verso della larghezza del Plane; in pratica, alla fine otterremo un rettangolo dentro il Plane quadrato originale.



*Il risultato dopo aver fatto tutti i tagli*

Selezioniamo ora tutti i vertici e duplichiamo la selezione con SHIFT D, ottenendo un nuovo Plane, quindi spostiamo leggermente in alto questo Plane, sopra quello di partenza (ad esempio con G Z 0.05); il nuovo Plane è, in pratica, lo strato plastico trasparente posto sopra la base opaca.

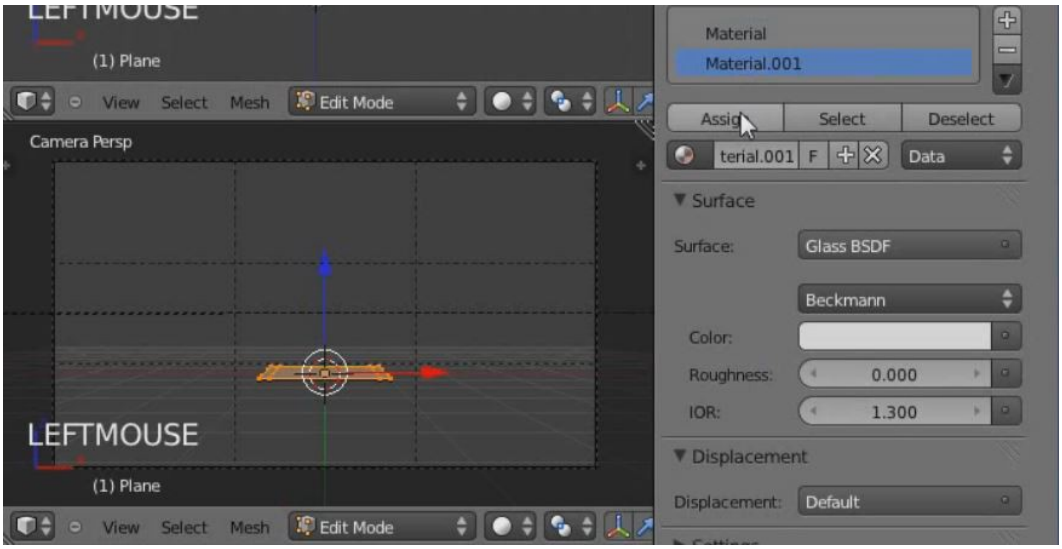
L'operazione va svolta in Edit Mode, così da lasciare il nuovo Plane (ottenuto per duplicazione) sempre all'interno dell'oggetto originale (non creando cioè un nuovo oggetto distinto, a parte).



*Creazione, mediante duplicazione, del nuovo Plane*

Mentre ci siamo, aggiungiamo un nuovo sottomateriale nello slot dei Materials dell'oggetto, scegliendo questa volta un ombreggiatore Glass con IOR 1.3.

Con il nuovo Plane e il nuovo sottomateriale selezionati, clicchiamo su *Assign* nella scheda dei Materials, per assegnare il nuovo sottomateriale trasparente al plane superiore.



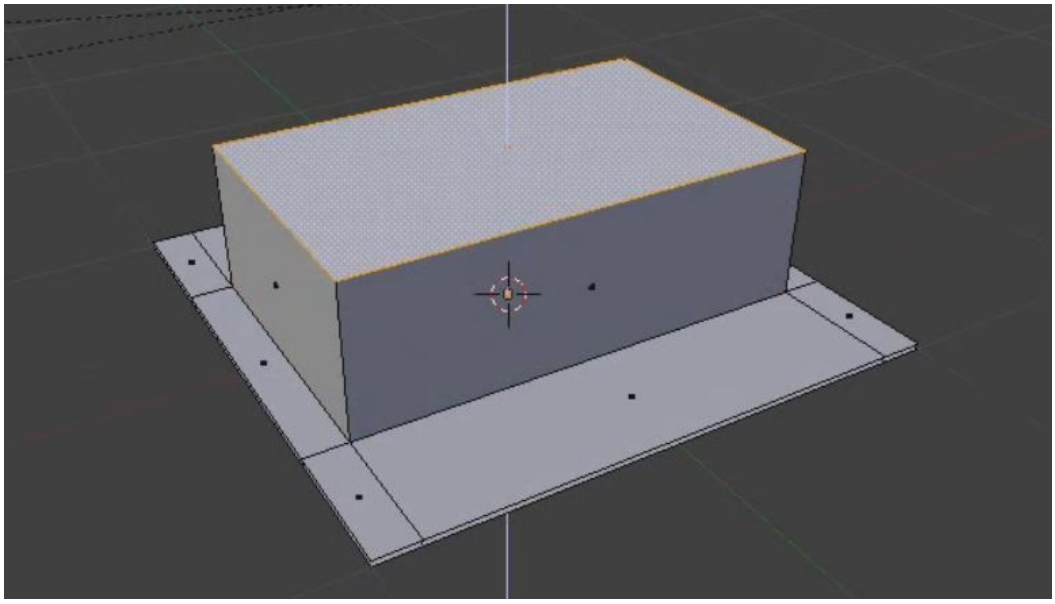

*Assegnare il sottomateriale GLASS al nuovo Plane*

Passiamo in modalità selezione facce e selezioniamo la faccia rettangolare realizzata al centro del Plane superiore, e solo di quello superiore: il Plane inferiore deve rimanere piatto.

> *NOTA: abbiamo dovuto tagliare il Plane con Ring Cut perché Subdivision Surface accorcerà entrambi i Plane e l'effetto varia se ci sono tagli e spigoli all'interno della figura, per cui ci servono i nuovi spigoli anche per il Plane inferiore.*

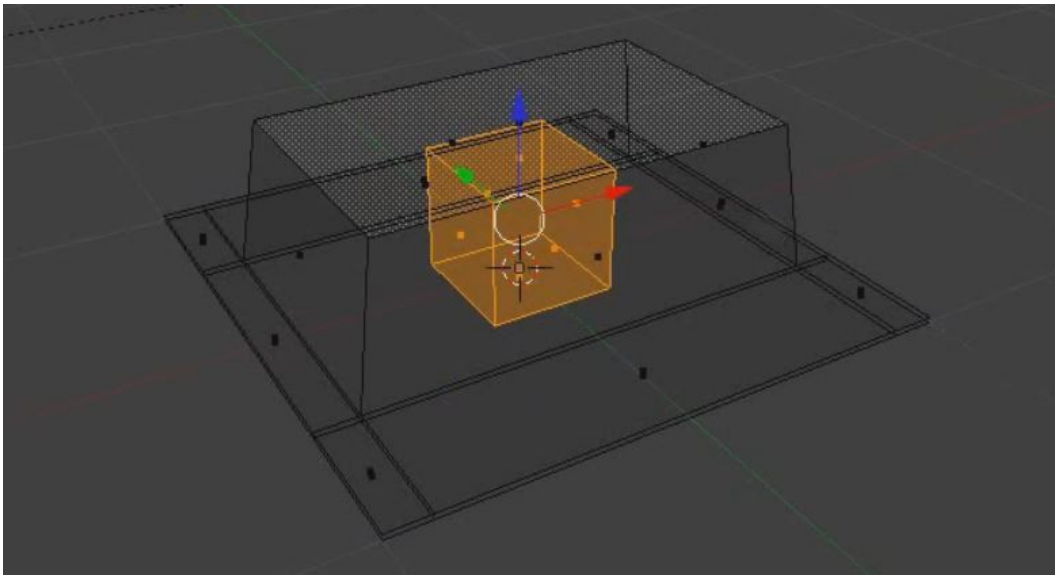
Occupiamoci, comunque, del Plane superiore.

Estrudiamo sull'asse verticale con *E Z* e trascinando un po' la faccia centrale del Plane superiore, in modo da ottenere lo spazio per la compressa.



*Estrusione per creare lo spazio per la compressa*

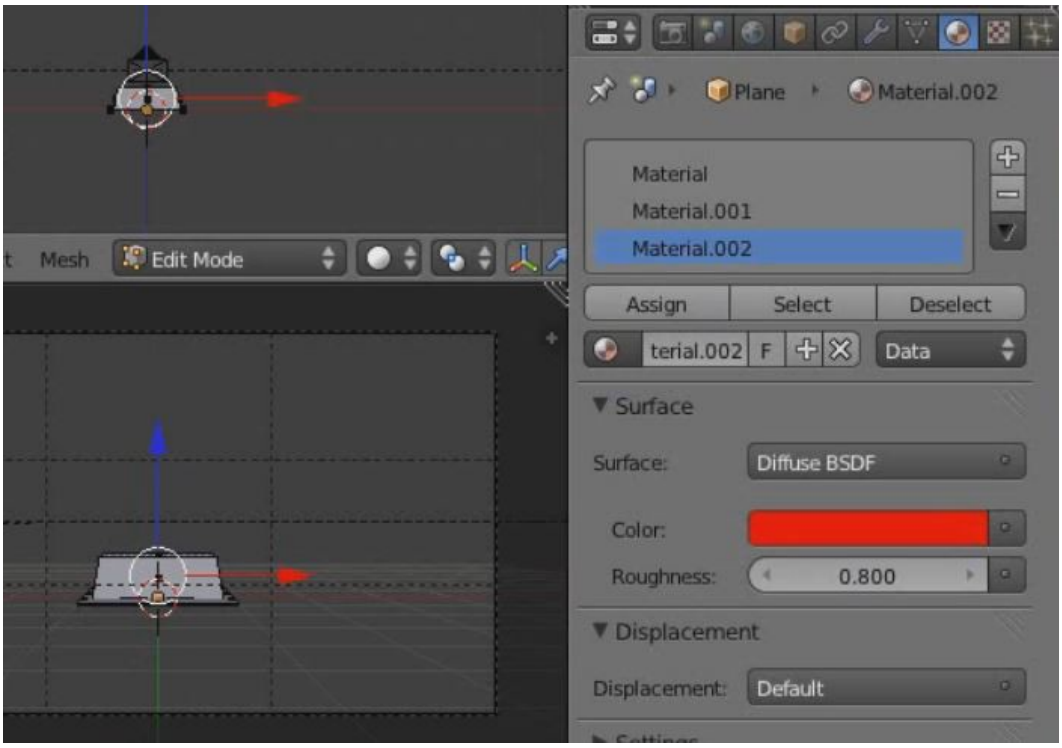
Aggiungiamo anche la compressa all'interno dell'oggetto di partenza, cioè sempre in Edit Mode, come fatto per il Plane superiore; inseriamo, ad esempio un Cube, scalandolo un po' lungo uno o più assi. Non preoccupatevi se si tratta di un parallelepipedo, per niente arrotondato: presto risolveremo il problema con Subdivision Surface.



*Creazione della compressa*

Per svolgere queste operazioni può essere utile passare alla modalità di visualizzazione Wireframe nel menù Viewport Shading delle varie 3D View, in modo da verificare posizione e dimensioni della "compressa" all'interno del secondo Plane.

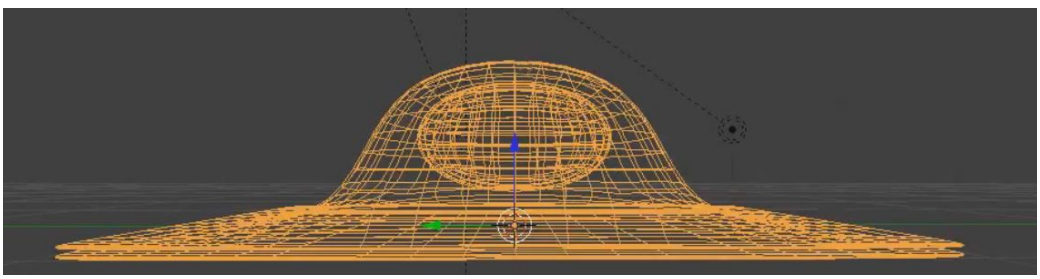
Con questo parallelepipedo selezionato, aggiungiamo un terzo sottomateriale nello slot dei Materials, ad esempio di tipo Diffuse o anche Velvet, magari colorato in modo da distinguerlo un po' dagli altri elementi, quindi assegniamo tale sottomateriale ai soli vertici del parallelepipedo, come fatto poco fa con il secondo sottomateriale e i vertici del Plane superiore.




*Il sottomateriale per la compressa*

Vi ho fatto creare gli oggetti in Edit Mode, realizzando un unico Object, perché ora i modificatori che applicheremo a tale Object verranno applicati automaticamente a tutte le varie parti, già dotate dei relativi sottomateriali, permettendoci di risparmiare un bel po' di tempo.

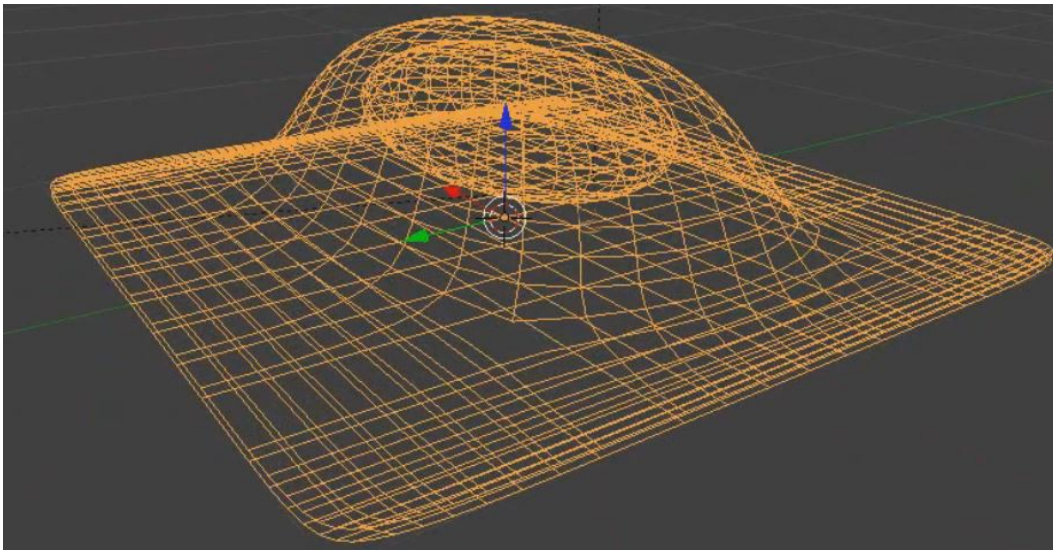
Andiamo quindi in Object Mode, clicchiamo magari su Shading *Smooth* nella Tool Shelf e, nella scheda Modifiers, all'interno della Properties Window, aggiungiamo un modificatore Subdivision Surface.



*Shading SMOOTH e il modificatore Subsurf*

Impostiamo a 3 il numero delle suddivisioni sia per View che per Render.

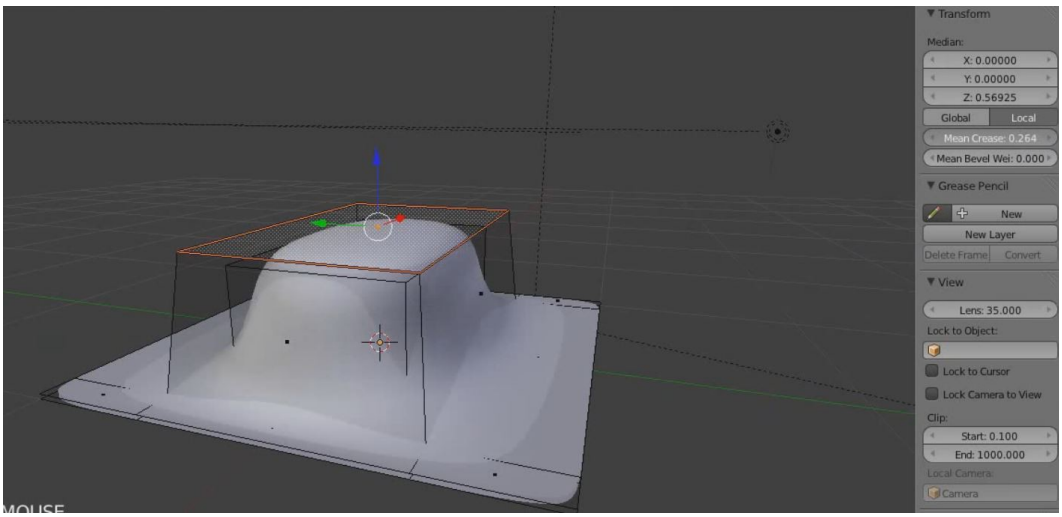
Notiamo subito, soprattutto se siamo in visualizzazione Wireframe, che ci sono delle cose che non vanno: la curvatura dello spazio per contenere la compressa è eccessiva e la compressa stessa è troppo schiacciata.



*Problemi con Subsurf*

Per risolvere il primo problema, torniamo in Edit Mode, selezioniamo i quattro spigoli che costituiscono la base superiore dello spazio per contenere la compressa, quindi apriamo la Transform Window in una finestra 3D View e aumentiamo il valore del parametro *Mean Crease*, osservando in tempo reale le deformazioni apportate alla mesh.

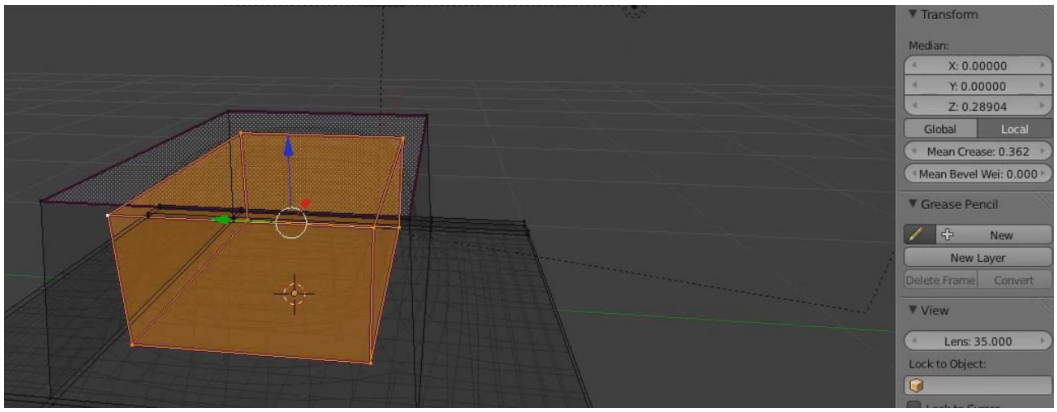
*Mean Crease* regola il "peso" matematico associato allo spigolo selezionato (nel nostro caso, i 4 spigoli che costituiscono la base superiore dell'oggetto); dal punto di vista pratico, con Subdivision Surface, ciò corrisponde a limitare l'effetto del modificatore (sono inversamente proporzionali).




*Utilizzo di Mean Crease sul "contenitore"*

A questo punto possiamo regolare anche la forma della compressa: deselezioniamo tutto, selezioniamo un vertice della compressa e premiamo *L* (per *Linked*, collegati), in modo da selezionare velocemente tutti i vertici che costituiscono la compressa.

Modifichiamo quindi il *Mean Crease* della selezione, osservando gli effetti nella 3D View; ovviamente, possiamo effettuare anche altre operazioni, come uno scaling sul solo asse verticale della selezione, oppure potete aggiungere e scalare dei "tagli" con Ring Cut, eccetera.

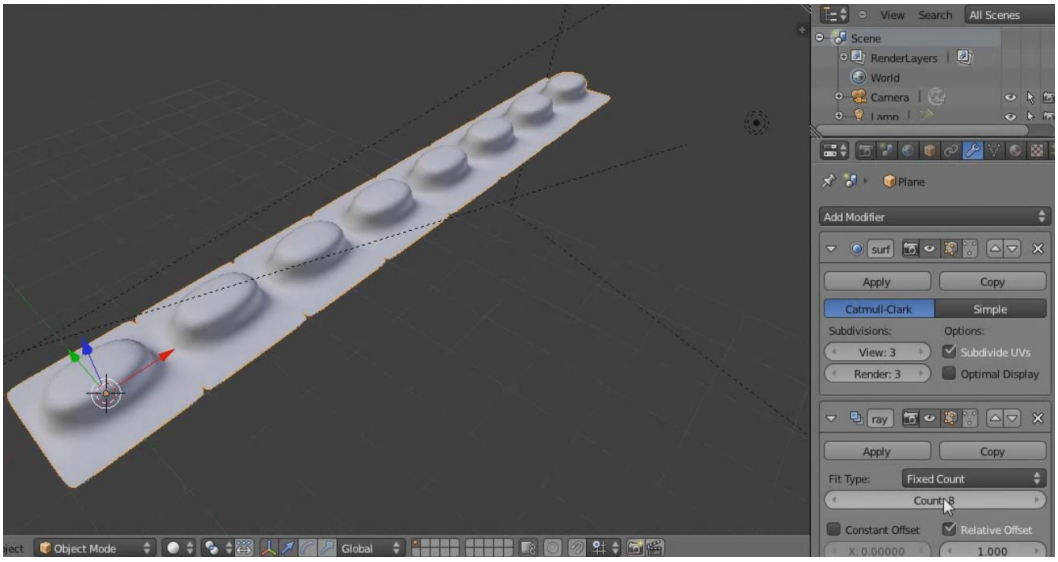


*Mean Crease sulla compressa*

Tra l'altro, qui ho applicato lo stesso *Mean Crease* a tutti gli spigoli della compressa, ma ovviamente potete dare valori diversi a spigoli diversi, osservando i vari effetti, come fatto ad esempio con la faccia superiore dello spazio che conterrà la compressa, dove ho modificato appunto solo il *Mean Crease* degli spigoli superiori, non di tutti gli spigoli del Plane.

Torniamo in Object Mode e inseriamo un nuovo modificatore per l'oggetto; questa volta, si tratterà di un Modifier Array.

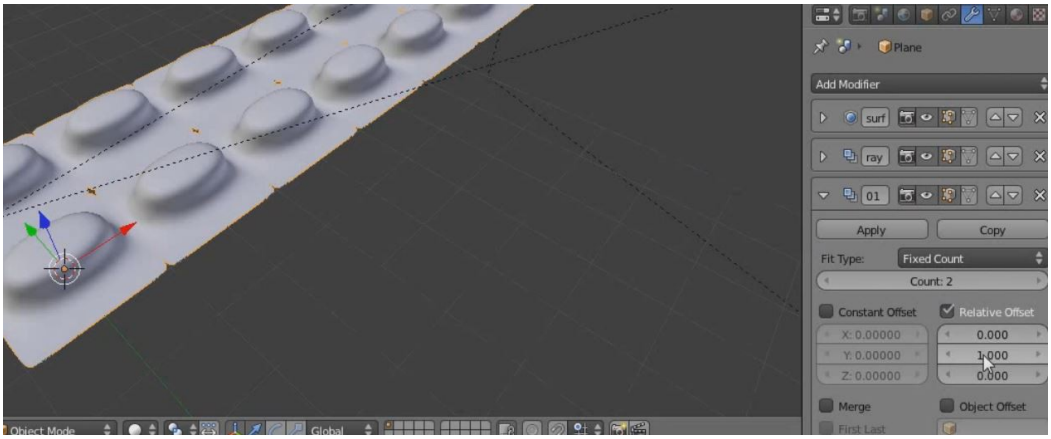
Di default, un Array realizza due copie distanziate in maniera relativa (cioè una copia segue subito l'altra) sull'asse X locale; portiamo il valore di Count ad esempio a 8, per aumentare il numero delle copie, lasciando per il momento le altre impostazioni come di default.




*Inserimento del primo modificatore Array*

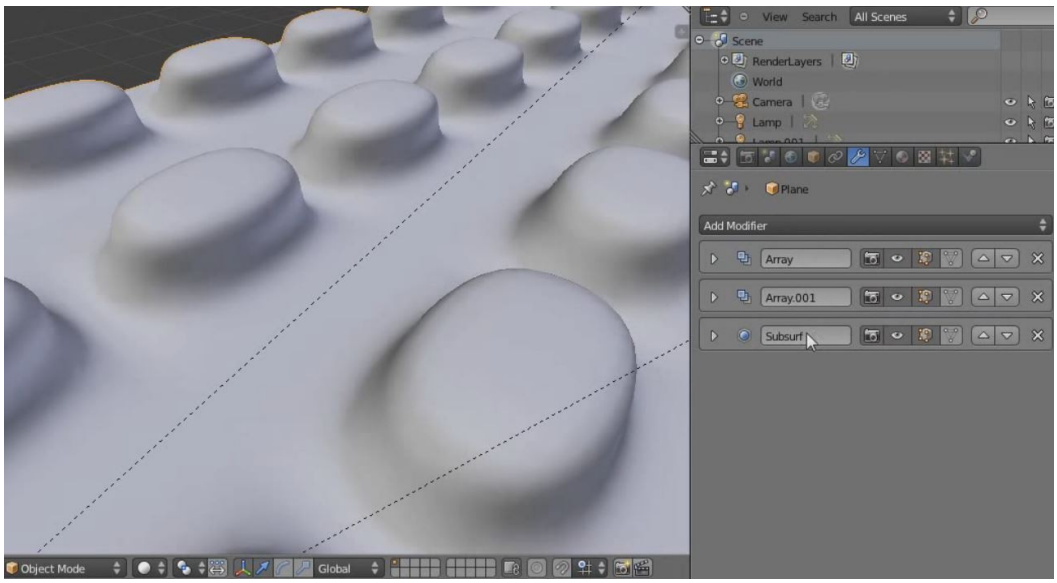
Per realizzare le copie anche in larghezza, cioè lungo l'asse Y, perpendicolare a X, aggiungiamo un altro modificatore Array, impostando questa volta 0 per *Relative Offset X* (che di default è 1) e 1 per *Relative Offset Y*, subito sotto. Portiamo Count, ad esempio, a 3.

Come potete vedere, c'è un inconveniente: Subsurf smussa l'oggetto originale e l'effetto si propaga alle copie realizzate mediante gli Array, quindi abbiamo dei buchi tra un oggetto e l'altro.




*Il secondo Array; c'è un problema...*

Per risolvere questo problema, dobbiamo selezionare la casella "*Merge*" (letteralmente: unisci, per unire un elemento al successivo) nelle schede sia del primo che del secondo modificatore Array; non solo: dobbiamo anche spostare il Modifier Subsurf in basso, nella pila dei modificatori utilizzando le frecce a destra nella scheda del Modificatore.




*Il risultato con Merge per Array e dopo aver "spostato in basso" Subsurf*

L'ordine con il quale vengono applicati i modificatori è infatti importante: applicando prima Subsurf, arrotonderemo l'oggetto iniziale e le copie fatte con gli Array saranno già arrotondate, mentre applicandolo dopo prima uniremo le copie con i *Merge* degli Array, poi arrotonderemo l'oggetto finale, per cui solo i bordi dell'intera figura verranno smussati.

Da notare che gli Array mantengono i valori di *Mean Crease* per tutte le copie, quindi tutti gli spazi e le compresse hanno la forma dell'originale.

Se dovete esportare l'oggetto, ad esempio come file OBJ a parte, allora dovete applicare i Modificatori, altrimenti vi consiglio di non farlo, in modo da far occupare poco spazio al file *.blend*.

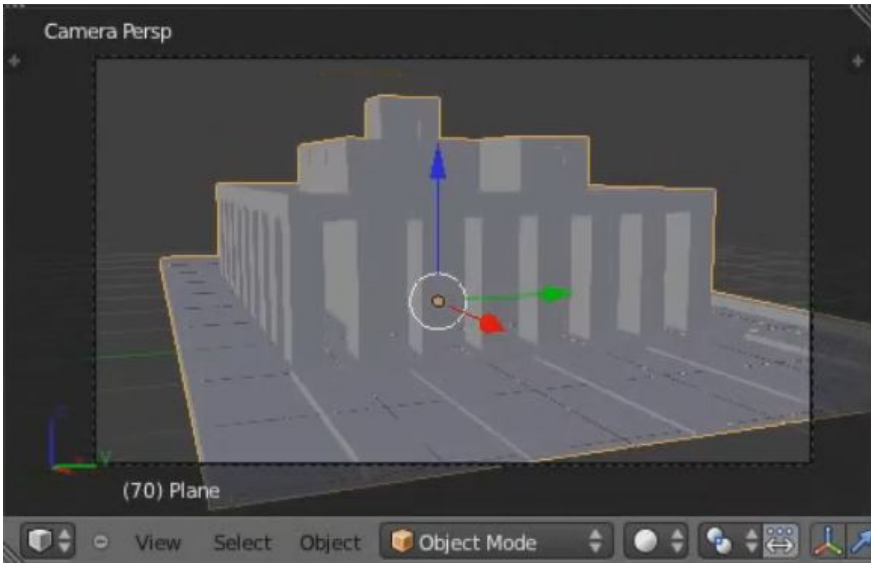
Tutto è pronto per personalizzare gli Shaders, posizionare le luci e, in generale, realizzare il set per il rendering finale; per quanto riguarda la modellazione (argomento di questo tutorial), è tutto.

*  *  *

# Rendering concept stile Sketch Up con AO ed Edge

In questo brevissimo tutorial vedremo come impostare una scena per ottenere un rendering stile "Concept", con una resa simile a quella data ad esempio da SketchUp, in Blender 3D a partire dalla versione 2.5. Il rendering verrà effettuato con il motore interno di Blender.

La scena che sto utilizzando contiene una mesh molto semplice, da "concept", appunto, non certo per un lavoro realistico.



*La scena utilizzata per questo tutorial*

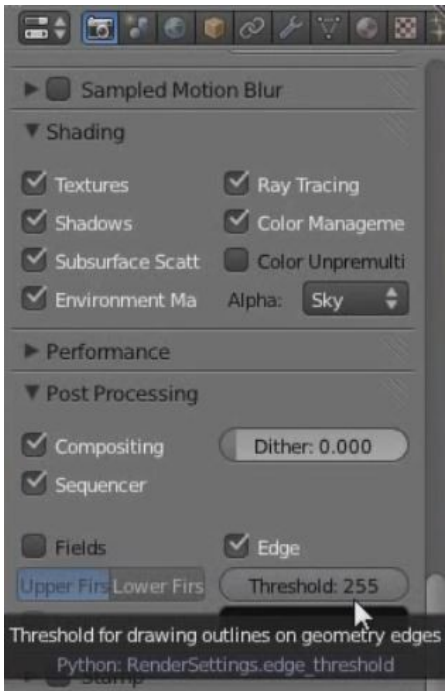
Non sono presenti fonti di luce, né World viene trattato come illuminante con Environment Lighting o altre tecniche simili; l'unica illuminazione è data, in effetti, da Ambient Occlusion, in modalità additiva, da attivare nella scheda World, all'interno della Properties Window.

Il valore di default per Factor, 1.0, il più delle volte dovrebbe andare bene, ma ovviamente potete fare dei rendering veloci provando valori diversi per regolare l'intensità dell'illuminazione.

I Materials degli oggetti dovrebbero essere di base, senza effetti come Mirror, Trasparenza o utilizzo di Textures, in quanto (anche se si tratta comunque di un rendering) vogliamo ottenere un effetto Concept, non fotorealistico o comunque definitivo. È possibile ovviamente cambiare i colori dei Materials a piacere ma per il resto le altre impostazioni dovrebbero essere lasciate come di default.

*Impostazioni di Ambient Occlusion*

Il terzo e ultimo aspetto da considerare, dopo Ambient Occlusion e i Materials di default, è la funzione *Edge* in Render, per evidenziare gli spigoli. Questa funzione va attivata selezionando la casella *Edge* nella sezione Post-Processing della scheda Render, nella Properties Window, portando poi al massimo (cioè 255) il valore del campo Threshold. La Color Box posta sotto Threshold fa riferimento al colore da dare agli spigoli e ovviamente potete scegliere qualsiasi colore.



*La casella Edge*

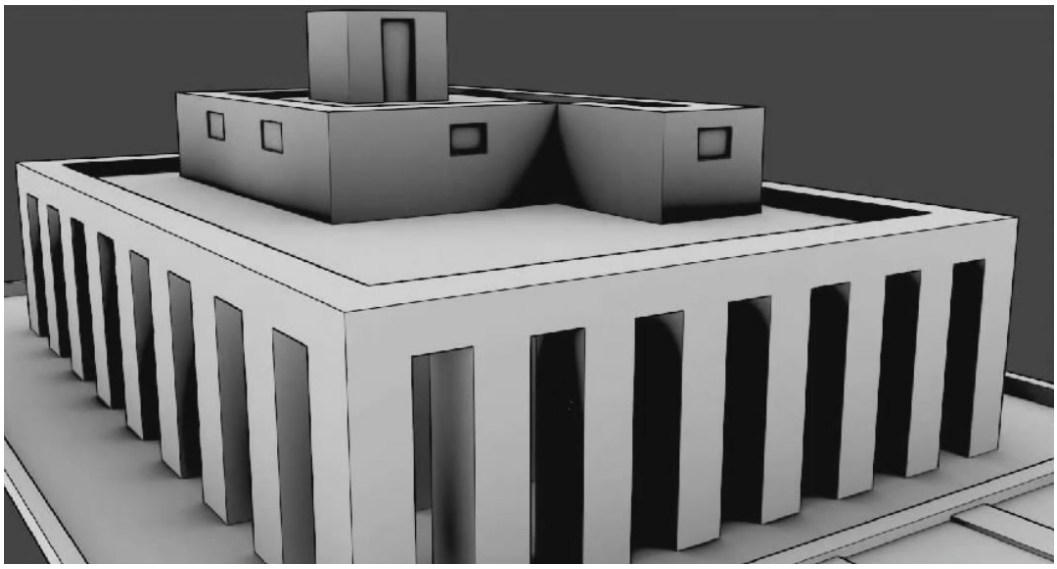
L'effetto verrà applicato a tutti gli oggetti presenti nella scena ma in misura diversa da oggetto a oggetto e, per ciascun oggetto, non tutti gli spigoli verranno evidenziati, ma in genere l'effetto è soddisfacente.

Ricapitolando, i tre punti principali per questa tecnica sono: Ambient Occlusion, i Materials di base e Edge in Render.

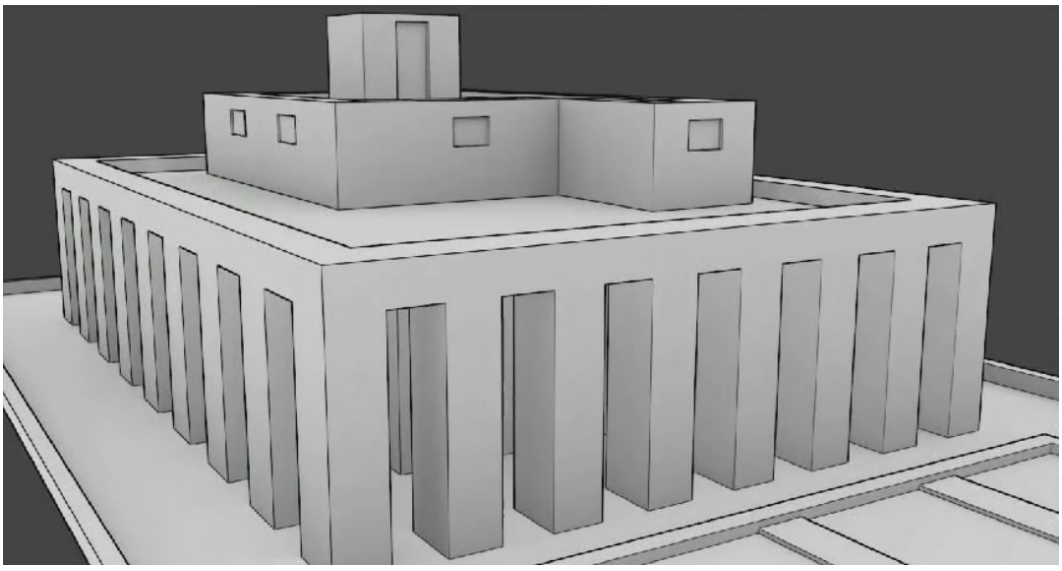
Qualche considerazione sul Gathering Method di Ambient Occlusion.
Il metodo Raytrace, come Gathering Method nella scheda World, con un basso numero di Samples produce, si sa, molto rumore. È il metodo indicato per un maggiore realismo e per evitare problemi che si verificano, in genere, con il metodo Approximate quando ci sono delle facce troppo vicine tra loro, tuttavia in questo caso ci interessa un concept, non un rendering realistico, inoltre nella maggior parte dei casi impostare un Pass per Approximate evita, come vedremo, molti effetti

indesiderati. Un altro motivo che spinge a scegliere Approximate è che è quasi esente da rumore e, generalmente, è più veloce di Raytrace (ma ovviamente aumentando il numero di Passes il vantaggio si riduce). In Approximate, il parametro *Error* può essere lasciato al valore di default, 0.25, mentre è importante impostare almeno 1 Pass (anziché 0, valore di default).

Nelle immagini seguenti sono visibili sia il rendering fatto con Approximate e 0 Passes, che restituisce molti errori in un gran numero di facce, sia il rendering fatto con 1 Pass, che invece restituisce un risultato più che soddisfacente.



*Rendering con 0 Pass*



*Rendering con 1 Pass*

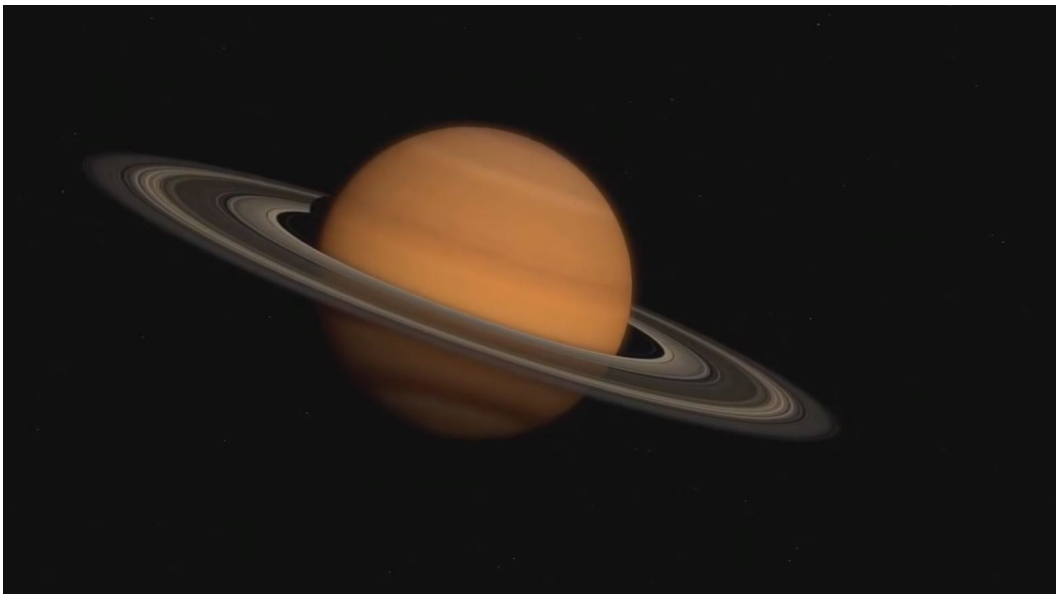
Il metodo non dovrebbe dare problemi con oggetti singoli, in inquadrature “da studio”, mentre nel caso di animazioni in ambienti chiusi o ricchi di elementi vi consiglio di fare qualche rendering di frame presi in vari momenti dell'animazione, per controllare la situazione ed eventualmente agire sui parametri di Ambient Occlusion (Factor, metodo, Samples o Passes) prima di procedere al rendering definitivo.

Non è necessario ricorrere ai Nodes del Compositing ma, ovviamente, nulla vi vieta di utilizzarli come con un qualsiasi rendering, ad esempio per regolare il contrasto, la gamma, fare uno sharpening o realizzare altri effetti.

*  *  *

# Saturno in Blender 3D – I parte

Questo è il primo di due tutorial nei quali vedremo come realizzare l'animazione del “fly-by” (un passaggio ravvicinato) del pianeta Saturno, che ci consentirà di trattare temi come Curve Guide sui sistemi particellari, la mappatura degli anelli, l'utilizzo di Warp, le stelle, il clipping della telecamera e soprattutto un ampio schema di Nodes nella fase di Compositing, in Post-Produzione.



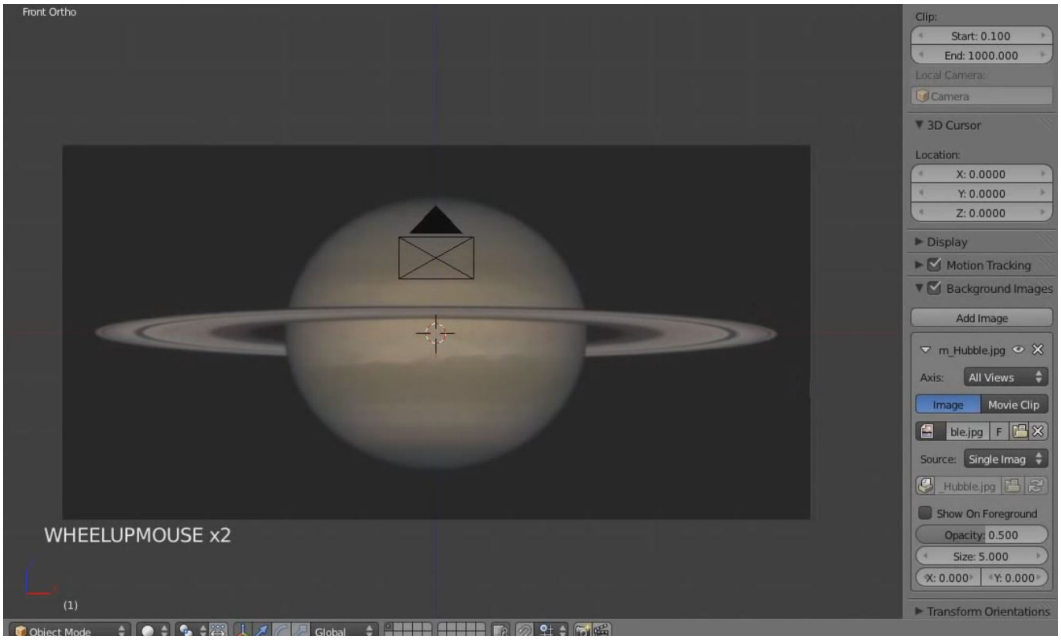
*Screenshot del risultato finale*

I tutorial sono stati realizzati con la versione 2.63 di Blender 3D e riguardano, come anticipato, vari argomenti; iniziamo con calma da una scena completamente vuota per trattare, in questa prima parte, il pianeta, lo sfondo stellato, l'illuminazione e la versione “solida” degli anelli, mentre nella prossima puntata vedremo come realizzare la versione particellare degli anelli e soprattutto l'ampio schema dei Nodes in Compositing.

La scena di per sé non è particolarmente complessa ma i temi abbracciati sono molti e di complessità media, per cui questi due tutorial sono rivolti a chi ha già seguito il videocorso sulle basi di Blender 2.5 e possibilmente l'ebook sul *Compositing* in Blender 3D 2.5, in modo da sapere a cosa si riferiscono ad esempio le voci di Stars o a cosa servono i Nodi per il Compositing.

La prima cosa da fare è procurarsi un paio di fotografie di Saturno, possibilmente in vista frontale; queste foto ci servono per darci un'idea delle dimensioni degli anelli rispetto al pianeta, perché (anche non conoscendo la vera dimensione degli anelli) uno spettatore può dirvi quasi in maniera intuitiva se gli anelli sembrano "troppo grandi" o "troppo piccoli", rispetto al pianeta... c'è una proporzione naturale, in effetti, che si riconosce intuitivamente... ma qui stiamo sforando nella filosofia, torniamo alla Computer Grafica!

Per aggiungere un'immagine di riferimento come Background Image, dobbiamo trovarci in una vista Ortho, ad esempio quella frontale (quindi Front Ortho), aprire la Transform Window della 3D View con N e aggiungere, nell'apposita sezione, la Background Image, scalandola e posizionandola a piacere nella vista 3D con gli appositi strumenti presenti nella scheda.

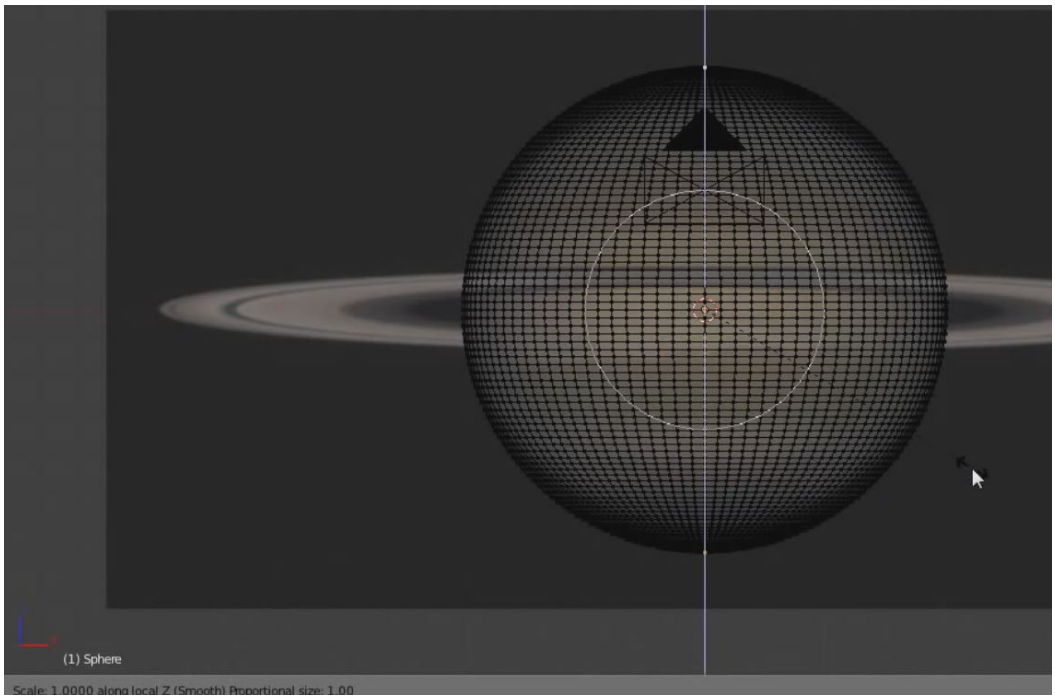


*La Background Image*

Per il pianeta, possiamo creare una UV Sphere con molte suddivisioni (ad esempio, 96 o più anelli e segmenti) e applicare, ovviamente, l'ombreggiatura *Smoot*h, dal menù Shading nella Tool Shelf.

Saturno, come praticamente tutti i corpi celesti sferici, è leggermente schiacciato ai poli; per realizzare questo effetto, andiamo in Edit Mode, selezioniamo i due poli (con una selezione multipla sui vertici), attiviamo la modalità Proportional Editing nell'header della 3D View, premiamo S Z Z per uno scaling sull'asse verticale locale (che avrà l'effetto di avvicinare i due poli), regolando

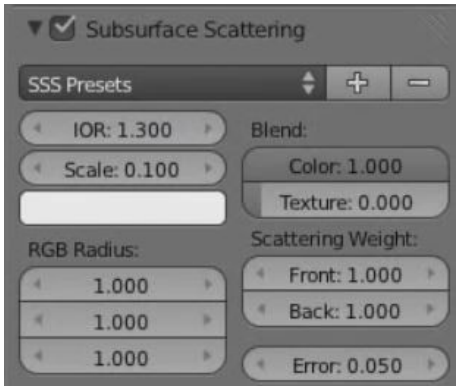
l'ampiezza dell'area influenzata dal Proportional Editing con la rotellina del mouse... potete utilizzare la modalità Wireframe per vedere l'immagine di riferimento Background Image per definire la trasformazione. Alla fine, torniamo in Object Mode.




*Scaling dei poli con il Proportional Editing*

Nella scheda Material del pianeta (con motore di rendering Blender Internal), aggiungiamo un Material, portando l'intensità della specularità a 0.0, in quanto non vogliamo certo degli *specular highlights* come se si trattasse di una biglia.

Altro parametro da impostare qui è l'effetto SSS, Sub Surface Scattering, un effetto utilizzato generalmente con liquidi o superfici organiche, come latte, pelle, crema, eccetera; qui lo stiamo utilizzando perché Saturno è un pianeta gassoso, quindi non realizzeremo l'atmosfera come mostrato, ad esempio, da Andrew Price per il pianeta Terra, o in generale per i pianeti rocciosi dotati di atmosfera, ma useremo SSS per rendere più "dolce", in un certo senso, la transizione tra la parte illuminata e quella scura, e in generale dando l'impressione, sempre con SSS, di un oggetto liquido, con uno strato sotto la superficie, non di una sfera solida.




*La scheda dell'effetto SSS*

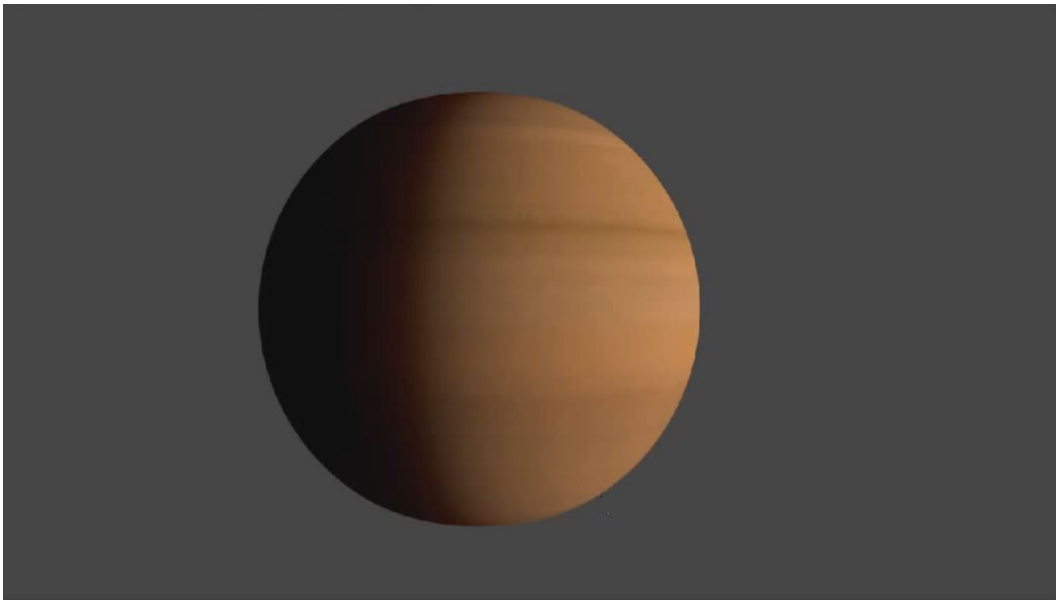
Come filtro, nella sezione SSS, utilizziamo *Skin 1*, la pelle, ma in realtà qui il parametro importante è Scale, il fattore di scala, che dipende dalle dimensioni della mesh nella scena... ci torneremo tra un attimo, prima impostiamo le luci e la Texture, perché per trovare un buon valore di scala dovremo fare dei rendering di prova, per cui ci servono questi elementi.

*La Texture per il pianeta*

Posizioniamo una Area Lamp in qualche punto lontano nella scena, inquadrando chiaramente il pianeta; anche posizionandola ad una notevole distanza, il valore di default per *Energy* (1.0) risulterà eccessivo, quindi abbassiamolo un po', ad esempio a 0.1 o anche 0.05, comunque anche per questo parametro, come per Scale di SSS, potremo fare delle prove con dei rendering al volo.

Applichiamo quindi la Texture a questo oggetto: si tratta di una Texture immagine, che dovrete cercare su Internet, digitando magari "*Saturn Map*", perché non vogliamo una foto del pianeta ma una sua rappresentazione sul piano, quindi verosimilmente l'immagine sarà rettangolare.

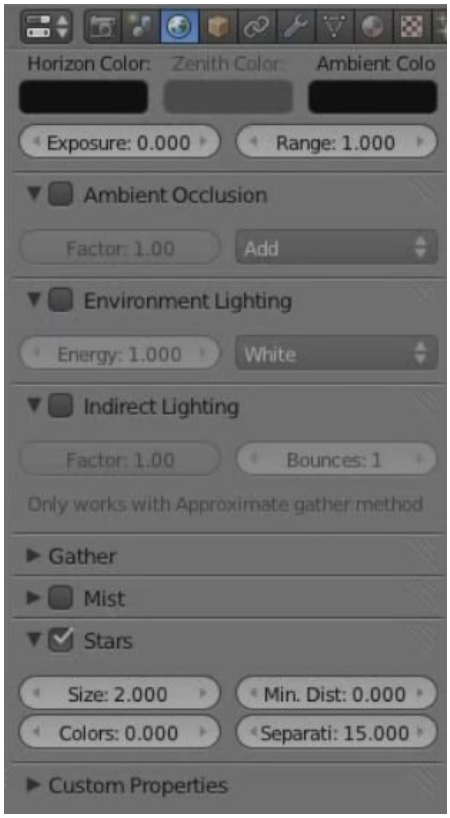
Nella scheda della Texture, l'unico parametro da cambiare rispetto a quello di default è la mappatura: in *Mapping*, cambiamo Projection da *Flat* a *Sphere*, così Blender avvolgerà la sfera con la Texture immagine, che è una rappresentazione in piano del pianeta... e con le impostazioni della Texture abbiamo finito: aggiungiamo una telecamera (ci serve per fare il rendering!) e posizioniamola in modo da inquadrare il pianeta, possibilmente non illuminato frontalmente ma di lato, e avviamo un primo rendering di prova per dare un'occhiata alla situazione.



*Il primo rendering*

In base al risultato, variamo l'intensità di *Energy* per la fonte di luce, magari spostiamo proprio la fonte di luce nella scena per ombreggiare meglio il pianeta, quindi proviamo alcuni valori di *SSS Scale* per il Material del pianeta, fino a che non troviamo quello che più ci soddisfa.

Prima di aggiungere il sistema degli anelli, apriamo la scheda World, nella Properties Window, e cambiamo lo sfondo dell'universo virtuale (*Horizon Color*) dal grigio di default a nero; poi, nella sezione *Stars*, attiviamo le stelle di sfondo.




*World e Stars*

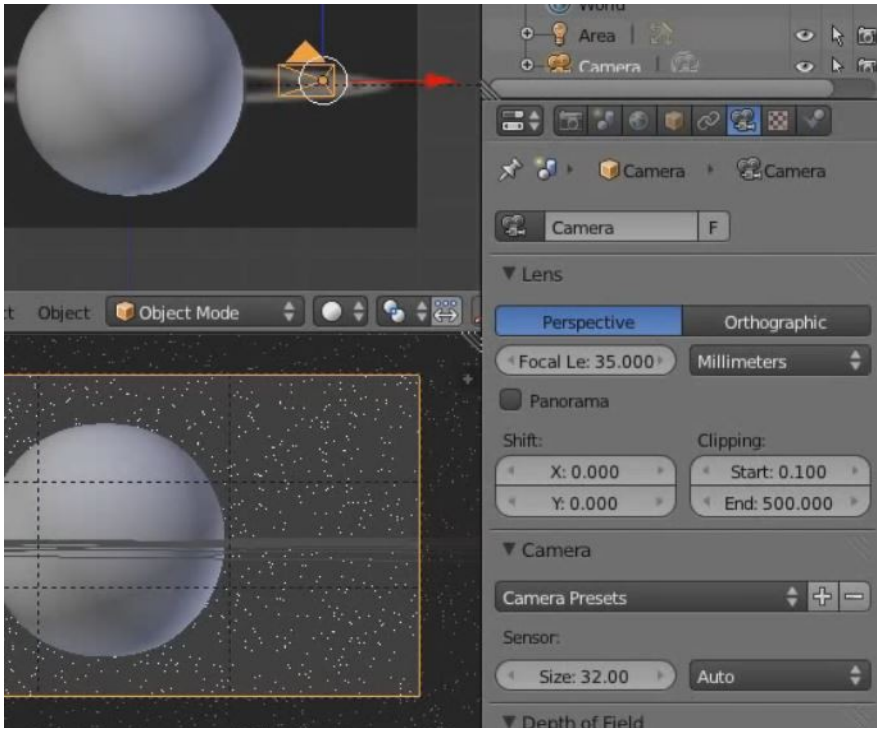
Anche qui, dimensioni, separazione e distanza dipendono fortemente dalle dimensioni degli oggetti nella scena e, in un certo senso, della scena stessa: se, ad esempio, tra la telecamera e il pianeta, nel punto di maggiore distanza, ci sono 50 unità di Blender, allora conviene impostare *Stars Minimum Distance* a 100 o anche più, perché di certo non vogliamo stelle tra la telecamera e Saturno in un incontro ravvicinato, sarebbe impossibile!

In base a questo valore, dovrete poi impostare *Separation*, per regolare la densità del campo stellare, e *Size*, per la dimensione media delle stelle; il consiglio è sempre lo stesso: fate dei rendering veloci con *F12* fino ad arrivare al risultato desiderato.

Nel mio caso, la distanza massima tra telecamera e pianeta era di 50 unità, quella tra telecamera e "Sole" 100 unità, per cui ho impostato Stars Minimum Distance 200, Separation 20, Size 0.2 e Color 0.6.

Una piccola nota a margine: se metterete, con le impostazioni di default, Distance a 200, nel rendering (e forse già nella 3D View) non vedrete alcuna stella. Ciò è dovuto al fatto che la Telecamera non renderizza gli oggetti che si trovano oltre una certa distanza, fissata di default a 100 unità di Blender, quindi non vedremo le stelle, qui poste a distanza minima 200.

Per risolvere questo problema selezioniamo l'oggetto telecamera, apriamo la scheda Object Data, nella Properties Window, e cambiamo il valore di *Clipping End*, portandolo ad esempio a 500, in modo da lasciare un bel po' di spazio al campo stellare che inizierà da 200 unità di distanza dalla telecamera.



*Clipping End, per la telecamera*

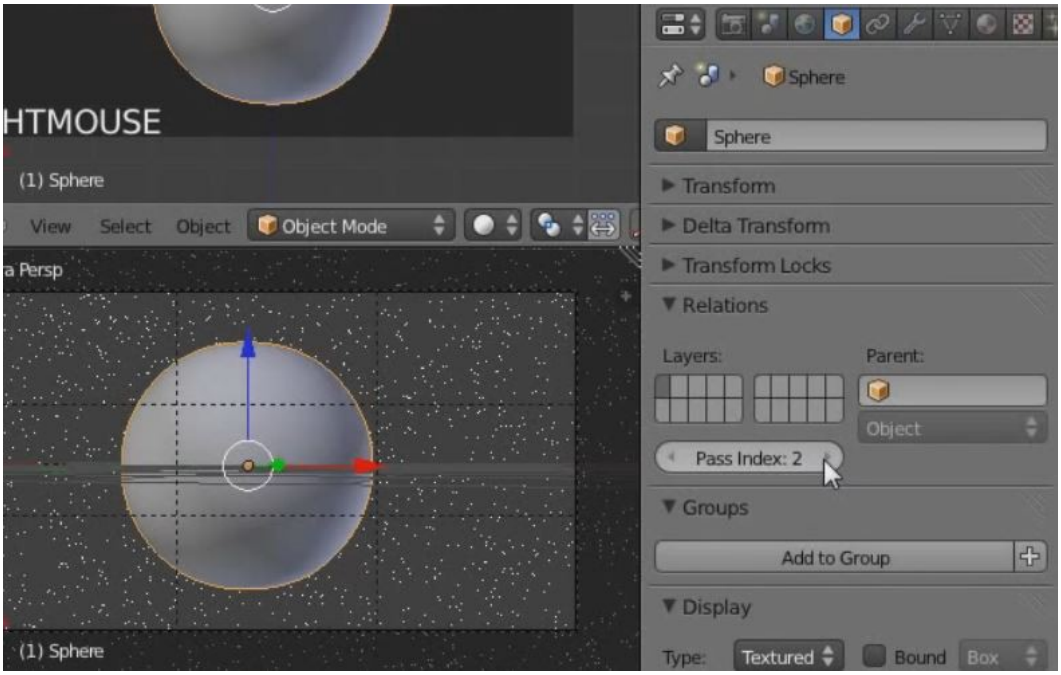
Prima di procedere con gli anelli, aggiungiamo, nel punto ove si trova la Area Lamp che illumina la scena, una Mesh Circle chiusa (scegliamo l'opzione *Fill Ngon* nella Tool Shelf in fase di creazione dell'oggetto), orientandola magari verso Saturno, e scaliamola per renderla molto piccola, quindi assegniamole il *Pass Index* 1 nella scheda Object, all'interno della Properties Window, per isolarla facilmente in fase di Compositing, infatti questa mesh produrrà il bagliore anamorfico e gli effetti lenticolari visibili nella parte finale del videoclip.

*Creazione della mesh che rappresenta il Sole*

Come Material per questo oggetto, impostiamo semplicemente *Shadeless* nella sezione Shading, con colore a piacere (ma, trattandosi del Sole e non di una gigante rossa o azzurra, bianco-giallognolo).

Visto che in seguito dovremo utilizzare il *Pass Index*, abilitiamo questo canale per il Compositing già da ora, selezionando la casella Object Index nella sezione *Layers* della scheda Render, nella Properties Window... non solo: posso anticiparvi che in fase di Compositing dovremo isolare anche il pianeta, senza gli anelli, per dargli un po' di blurring, per cui selezioniamo la sfera e impostiamo *Pass Index* 2 nella scheda Object dell'oggetto.

Il blurring o sfocatura non potrà essere impostato per tutto il rendering, anche se gli oggetti sembrano essere solo questi, perché in realtà ci sono le stelle, e applicando una sfocatura a tutto il rendering in arrivo a *Render Layers* lo applicheremmo anche a quelle... ecco perché stiamo predisponendo anche il pianeta per il Compositing.



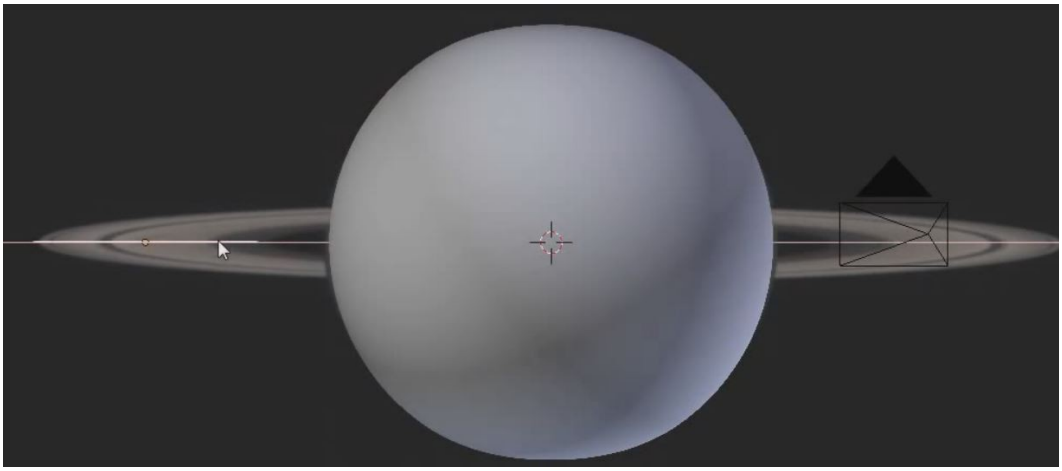

*Pass Index 2 per la sfera*

Il sistema degli anelli consta in realtà di due sistemi: una mesh per raffigurare gli anelli visti da lontano e una Mesh con sistema particellare e Curve per l'animazione per un passaggio all'interno degli anelli, dove vogliamo vedere proprio i detriti e le polveri.

Iniziamo col primo caso, quello più semplice... beh, apparentemente più semplice!

La mesh da realizzare è una corona circolare, la Texture è un'immagine (cercate su Internet *Saturn Rings*), ma probabilmente online troverete una Texture rettangolare, una sezione degli anelli, non l'inquadratura “dall'alto”, per così dire, degli stessi, da mappare da una vista superiore alla corona circolare... e la mappatura della Texture orizzontale è comunque difficile anche per via delle scuciture realizzate da Blender... che fare, quindi?

Aggiungiamo un Plane alla scena e posizioniamolo di taglio rispetto a Saturno, scalandolo e spostandolo seguendo l'immagine di Background come riferimento.

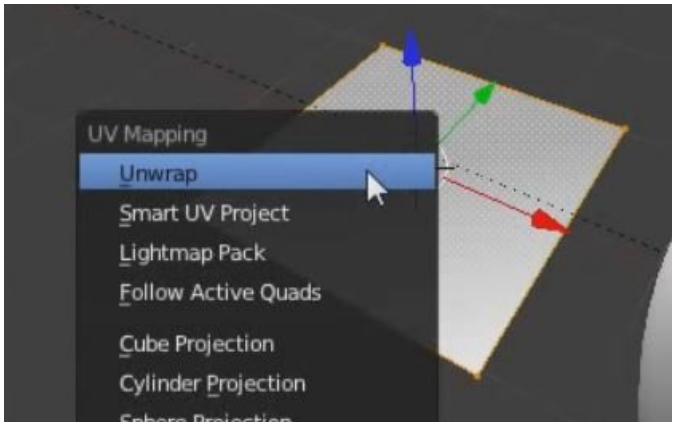


*Aggiungere il Plane per gli anelli*

La prima cosa da fare, ora, è l'UV-Mapping della Texture immagine degli anelli sul Plane; in seguito, prolungheremo e deformeremo questo Plane per realizzare gli anelli e la Texture verrà applicata automaticamente, ma intanto applichiamola a questa porzione.

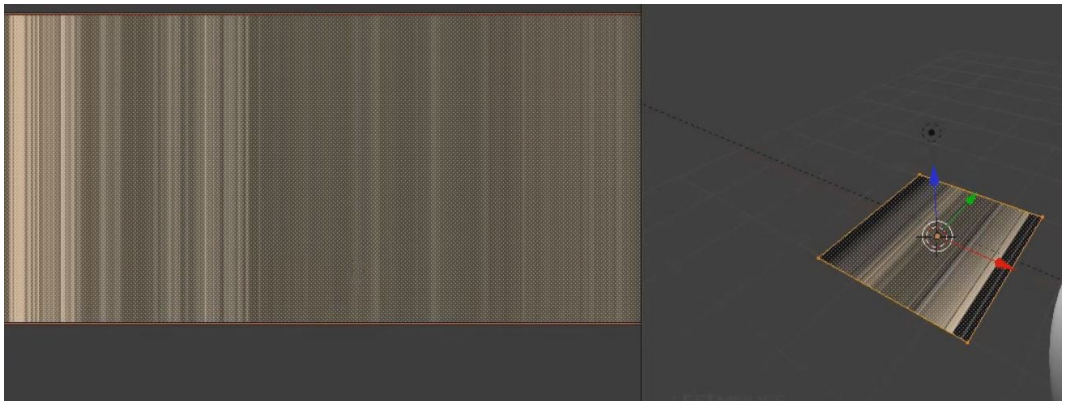
Passiamo per comodità ad uno Screen Layout *UV Editing* e impostiamo, nella vista 3D, la visualizzazione *Textured*, per vedere in tempo reale la mappatura e ruotarla, se sarà necessario.

Con il Plane selezionato, passiamo in Edit Mode, selezioniamo tutti i vertici e premiamo *U*, scegliendo poi *Unwrap* (dal momento che ci sono appena 4 vertici, l'unwrap produrrà un quadrato, non figure strane).



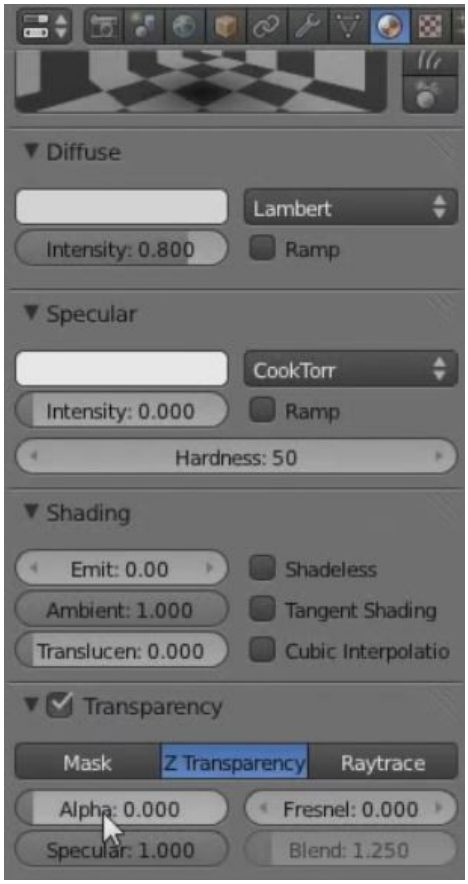

*Unwrapping del Plane*

Nella finestra UV, carichiamo l'immagine da utilizzare come pattern per gli anelli e mappiamoci sopra la scucitura. Fate attenzione, nella 3D View, all'orientamento del pattern: le linee che rappresentano gli anelli devono essere tangenti al pianeta o, se volete, perpendicolari al raggio.



*Mappatura della Texture degli anelli*

Se nell'immagine che avete preso trovate un paio di divisioni nere (attenzione, non gli spazi neri eventualmente presenti ai bordi, ma proprio stacchi neri all'interno degli anelli: la divisione Cassini e la Encke), posizionatele verso l'esterno, lontane dal pianeta (nel senso che la più ampia starà più o meno al centro ma leggermente a sinistra e la meno ampia a sinistra).




*Il Material per gli anelli*

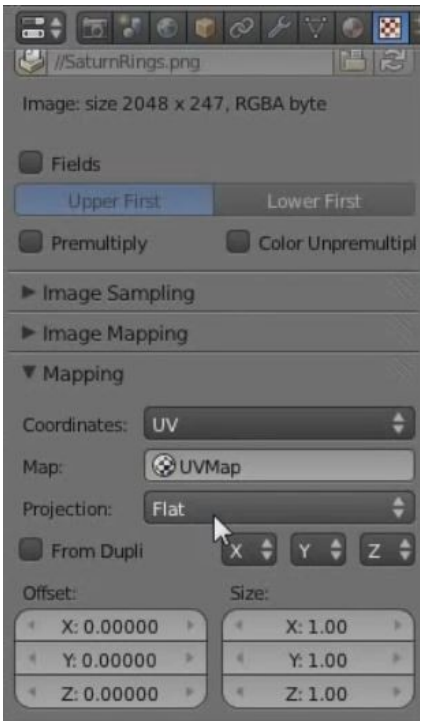
Fatta la mappatura UV, dobbiamo ancora applicare questa immagine come Texture all'oggetto.

Apriamo il pannello Material dell'oggetto e aggiungiamo un Material di default, portando però l'intensità della specularità a 0.0 e attivando la trasparenza di tipo Z-Transp, impostando 0.0 sia per Alpha che per Specular in questa zona (la trasparenza verrà gestita dalla Texture o dalle Texture, se avremo una Texture Alpha separata).

Passiamo quindi al pannello Texture, aggiungiamo una Texture di tipo *Image or Movie* e come immagine carichiamo, ovviamente, quella usata poco fa per l'unwrapping.

Nella sezione *Mapping*, cambiamo Coordinates da Generated a UV e impostiamo il *Layer UV* creato automaticamente da Blender in fase di unwrap nel campo "*Map*".

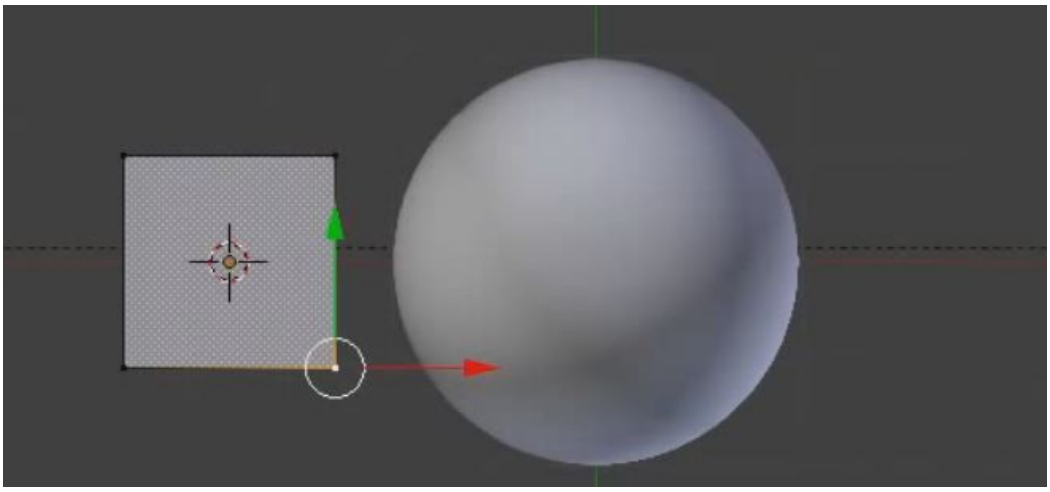
Dobbiamo lasciare, ovviamente, Diffuse Color selezionato, in *Influence*; per quanto riguarda la trasparenza, invece, dipende: online si trovano immagini in scala di grigi che rappresentano il solo canale di trasparenza degli anelli, quindi potete utilizzare la prima Texture, a colori, solo per Diffuse Color, e una seconda Texture, in scala di grigi, per il solo canale Diffuse Alpha, a patto ovviamente che le due immagini siano correttamente sovrapponibili e mappate con la stessa mappatura UV... altra opzione è aggiungere il canale Alpha all'immagine a colori, da salvare quindi in PNG con Alpha per utilizzare sia Diffuse Color che Diffuse Alpha in un'unica Texture... nel mio caso, per risparmiare tempo, non ho fatto niente di tutto questo, limitandomi a portare Diffuse Alpha a 2 in *Influence* per la Texture a colori... il risultato mi è sembrato comunque più che soddisfacente.



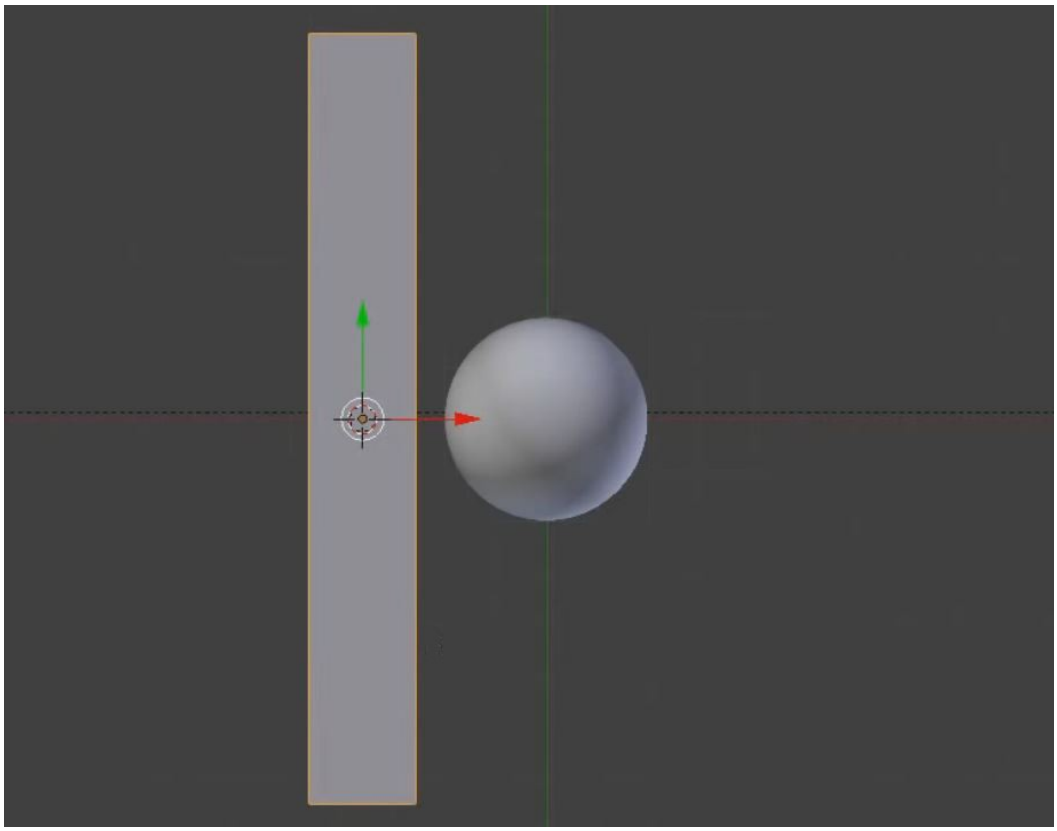
*La Texture per gli anelli*

Deselezionate, nella sezione *Image Sampling* della Texture, la voce "*Use Alpha*".

Terminato il lavoro su Materiali e Texture, torniamo alla modellazione: dobbiamo allungare questo Plane e avvolgerlo (ad anello, appunto) intorno a Saturno... ok, ma di quanto dobbiamo allungarlo? Qui ci viene in aiuto la matematica: la lunghezza del Plane (visto dall'alto) è la circonferenza dell'anello, quindi se ad esempio un vertice interno (cioè vicino al pianeta) del Plane si trova a distanza (esempio a caso) 5 unità dal centro del pianeta, allora la striscia dovrà essere lunga $2*5*\pi$.


*Il Plane visto dall'alto*

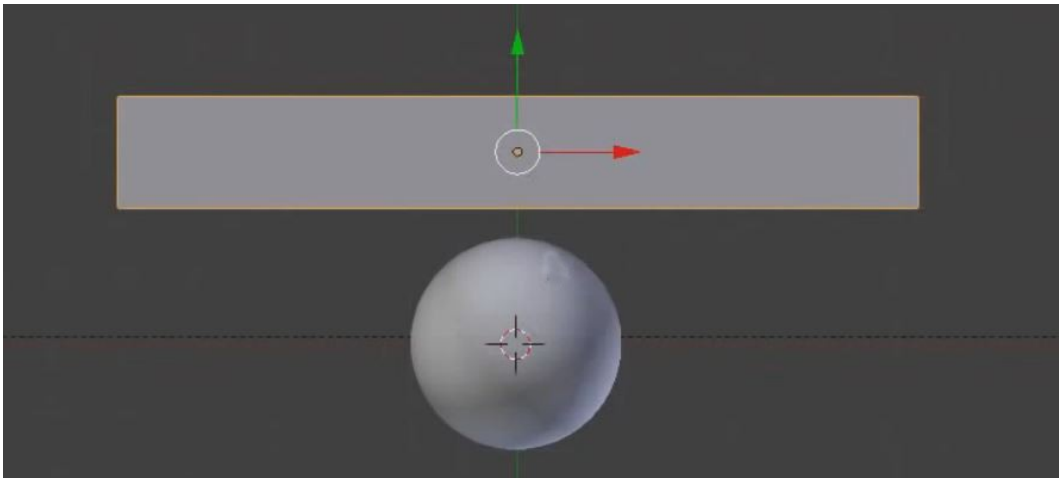
Calcoliamo quindi di quanto dovremo allungare il Plane nel verso perpendicolare al raggio ed effettuiamo questa operazione in Edit Mode (per semplicità, fate *2*raggio*3*, anziché 3,14). La Texture si allungherà di conseguenza coprendo tutto il Plane.



*Il Plane allungato*

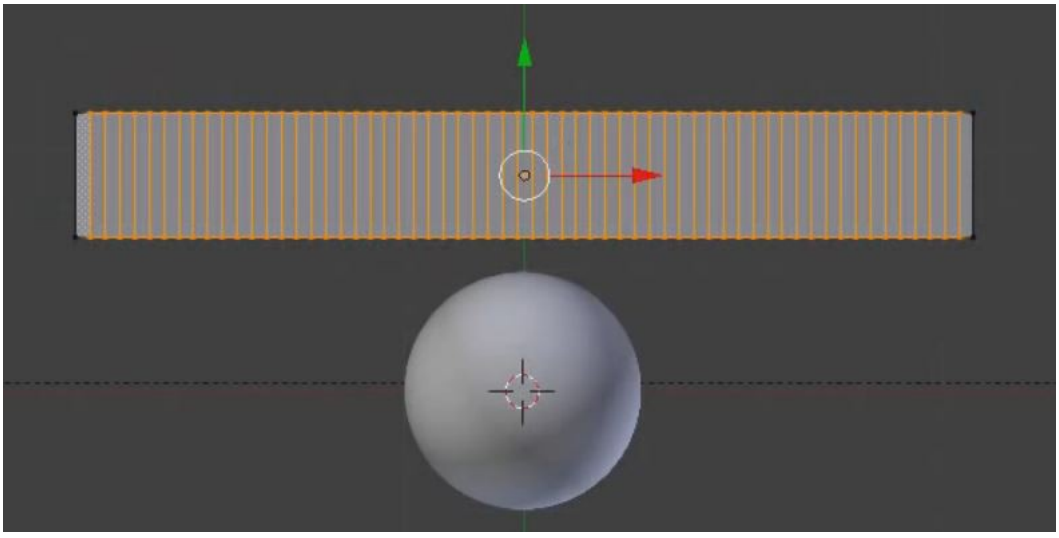
Ora, io qui ho posizionato il Plane alla sinistra della sfera, per definire distanza e dimensioni del Plane-anello... il modificatore *Warp*, però, non si comporta bene se il 3D Cursor, che rappresenta il centro della rotazione (e che quindi va messo al centro di Saturno, che qui coincide col centro della scena, con SHIFT S e "*Cursor to Center*") è di lato rispetto all'oggetto da deformare, ma solo se si trova sotto di esso (inteso nel senso dell'asse Y).

Dobbiamo quindi, in vista TOP, ruotare il Plane di 90 gradi, rendendolo quindi orizzontale, e porlo sopra Saturno, facendo attenzione a mantenere la distanza-raggio tra la parte interna della striscia e il centro del pianeta.



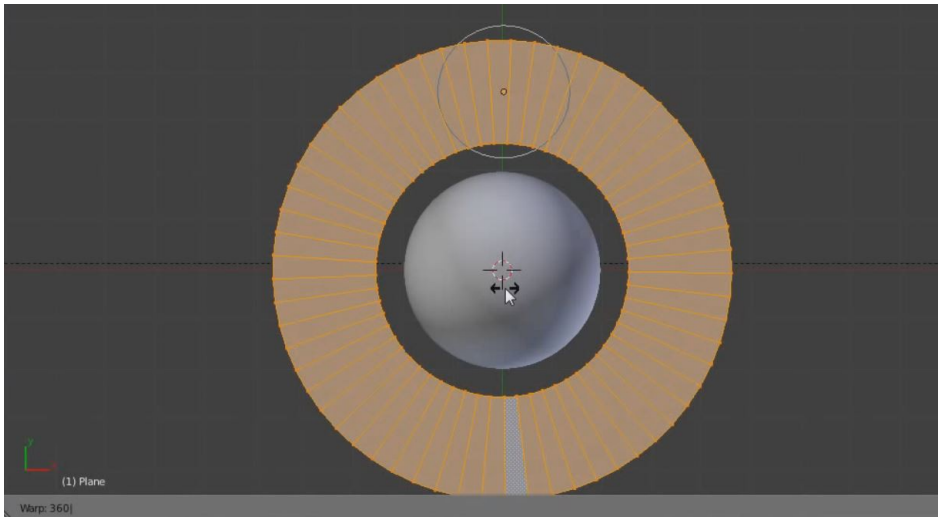
*Il Plane ruotato, nella vista TOP*

Prima di applicare *Warp*, dobbiamo suddividere un bel po' di volte il Plane, altrimenti la deformazione non verrà realizzata correttamente perché ci saranno troppi pochi vertici, quindi passiamo in Edit Mode, premiamo CTRL R per attivare Ring Cut e realizziamo tante "fette" verticali sul Plane.



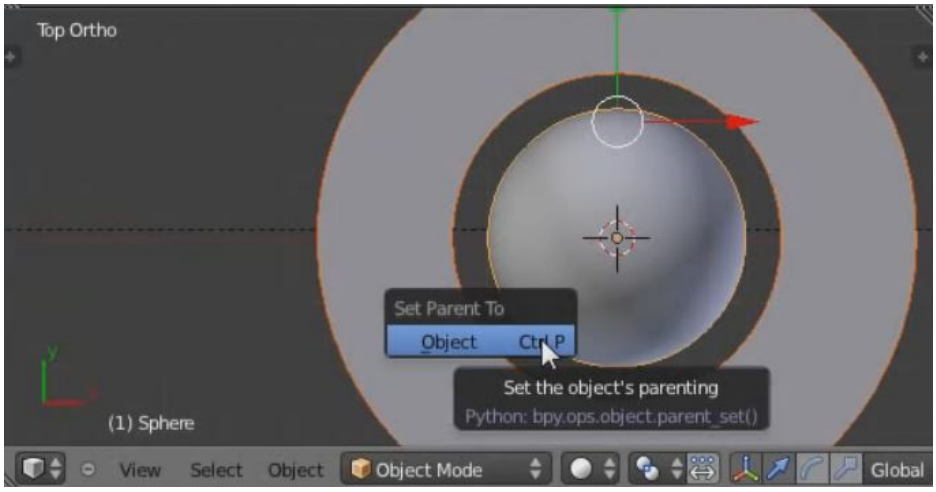
*Ring Cut sul Plane*

Restiamo in Edit Mode, con il 3D Cursor al centro di Saturno, quindi selezioniamo tutti i vertici del Plane, premiamo SHIFT W (per *Warp*), digitiamo 360 (per un giro completo) e premiamo Invio: abbiamo appena realizzato gli anelli, già texturizzati!




*Warping*

A questo punto voglio darvi un consiglio: imparentate gli anelli come figli della sfera pianeta (selezionando prima gli anelli, poi la sfera – con una selezione multipla, quindi SHIFT e click col tasto destro – per poi premere CTRL P e confermare): in questo modo, se deciderete di spostare, ruotare o trasformare in generale, sia in maniera statica che animata, il pianeta, gli anelli si trasformeranno coerentemente con l'oggetto padre, non dovrete preoccuparvi di questo aspetto.




*Parenting degli anelli alla sfera*

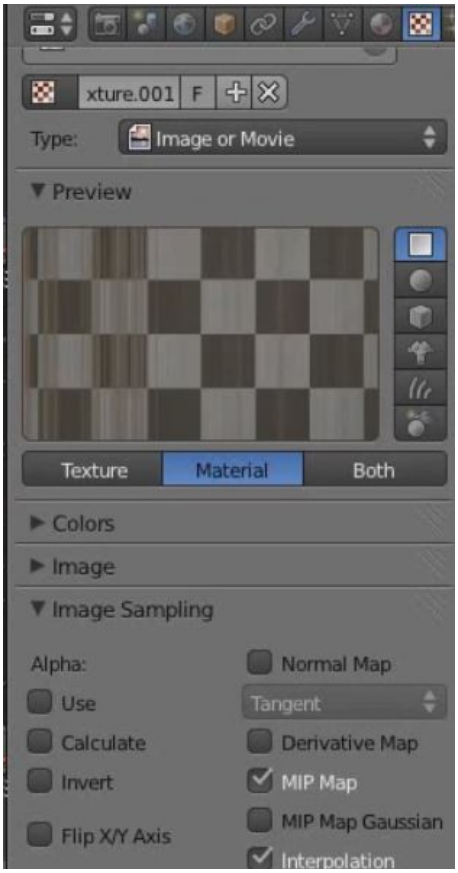
Come detto all'inizio del tutorial, se volete fare dei rendering da lontano questo sistema di anelli va più che bene... nel prossimo tutorial vedremo invece come generare un sistema particellare per i detriti (ma che va bene, ad esempio, anche per la fascia di asteroidi o altre scene simili) vedendo soprattutto come fare per "confinare", per così dire, gli elementi sul piano degli anelli e animare la loro rotazione sia sul posto che intorno al pianeta principale... e, ovviamente, ci occuperemo del Compositing per aggiungere un po' di blurring e gli effetti lenticolari.

*  *  *

# Saturno in Blender 3D – II parte

In questa seconda e ultima parte del tutorial dedicato all'animazione di Saturno in Blender 3D vedremo come realizzare degli anelli con il sistema particellare di Blender e come aggiungere qualche effetto di Compositing per rendere più interessante il rendering.

Prima di procedere, una piccola correzione riguardante la prima parte: se, come nel mio caso, state utilizzando solo una Texture immagine senza canale Alpha per gli anelli, nella scheda *Image Sampling* di tale Texture deselezionate la casella "*Use Alpha*", selezionata di default, che – se attiva – preleva le informazioni sull'Alpha dal relativo canale dell'immagine, se questo canale è presente... ma nel mio caso tale canale non c'è, quindi gli anelli vengono renderizzati come solidi. In casi analoghi, quindi, se volete associare il nero del colore alla trasparenza piena, e così via per gli altri colori, in un'immagine che non ha canale Alpha, deselezionate *Use Alpha* in *Image Sampling* per la Texture immagine.

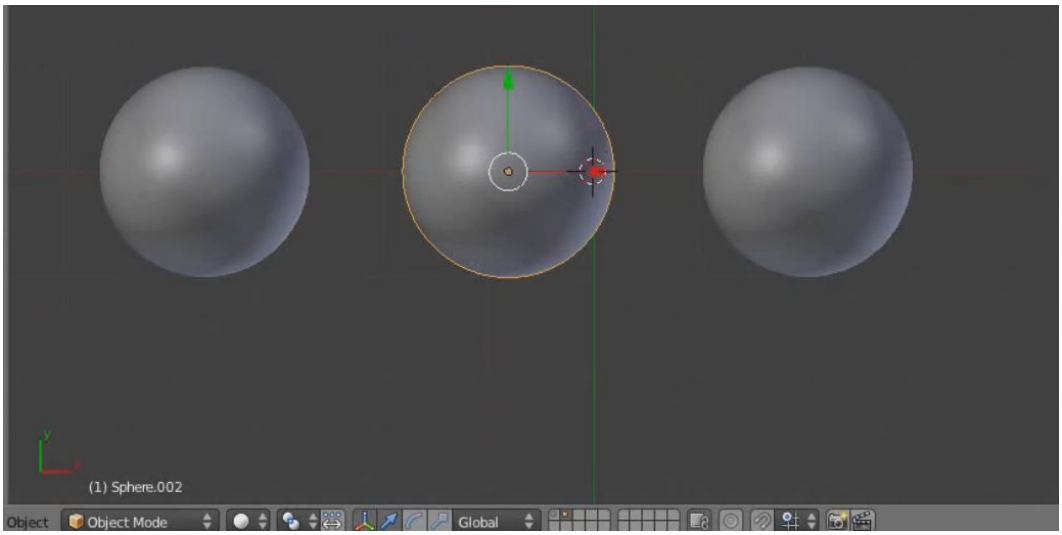


*Deselezionare USE ALPHA in Image Sampling*

Ok, fatta questa correzione, procediamo con i nuovi argomenti.

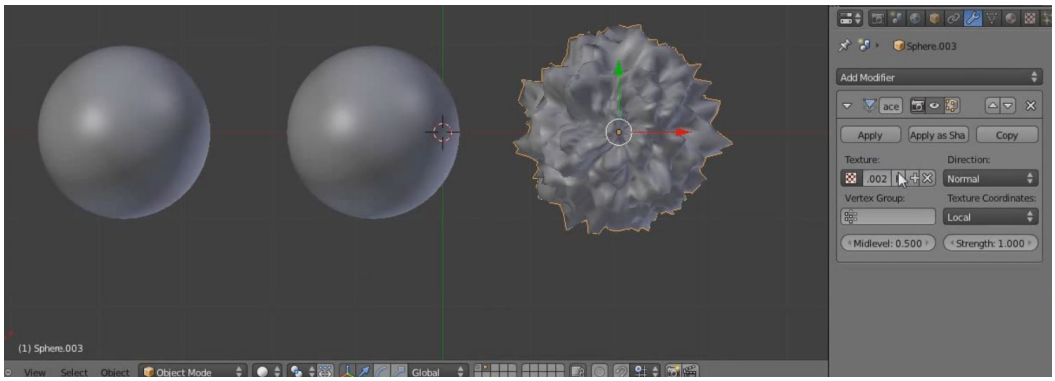
Per aggiungere le particelle degli anelli ci servono, per prima cosa, i modelli delle particelle da realizzare; questi non sono particolarmente complicati: a meno di non dover fare dei primissimi piani di una particella, ci basterà utilizzare una sfera e un modificatore.

Spostiamoci nel secondo Layer per creare lì i prototipi delle particelle e aggiungiamo, in Object Mode, 3 o 4 sfere UV Sphere, magari con 64 divisioni sia per i segmenti che per gli anelli (valori da impostare nella Tool Shelf, in basso a sinistra, appena si crea una sfera) e con Shading *Smooth*.




*Creare le sfere (particelle) in un Layer diverso*

Aggiungiamo, per ciascuna sfera, un modificatore di tipo Displace e, per ciascun Displace, impostiamo una nuova texture di *Displacement* (cliccando su "New" e dando alla Texture un nome significativo, ad esempio *Disp1*, *Disp2*, *Disp3*, e così via).

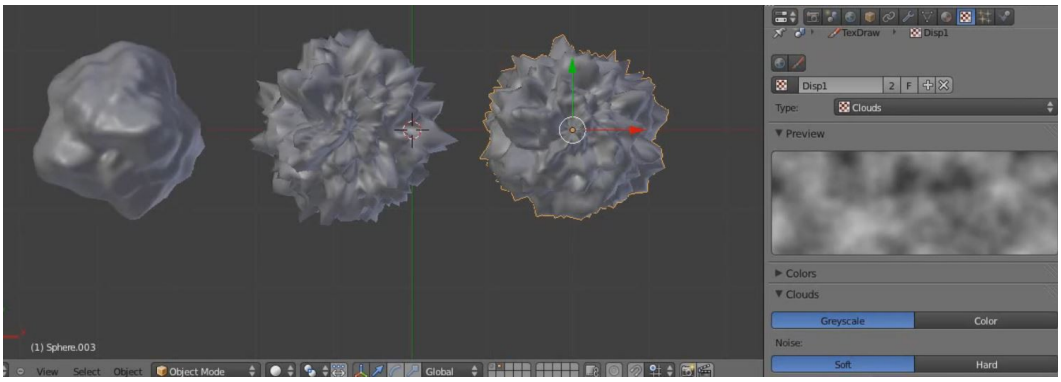



*Displacement*

Iniziamo a regolare i valori di Strength per i vari modificatori (selezionando quindi di volta in volta le sfere e variando Strength nella scheda del Modifier *Displace*), comunque in seguito, dopo aver definito la grandezza della Texture di *Displacement*, potrete tornare su questa scheda e variare la Strength, per provare i vari effetti; ad esempio, è possibile impostare un valore negativo per la Strength di un *Displace* e ottenere risultati interessanti.

Apriamo ora la scheda Texture (sia di una sfera che di World, non ha importanza) e aggiungiamo, nei vari slot, le Texture *Displacement* create, scegliendole dal selettore "*Browse Data*"; attenzione, queste Texture non devono essere attive per l'aspetto della scena o dell'oggetto, quindi se si tratta di Texture World poco importa, ma se sono Texture di oggetti (ad esempio, della prima sfera) dobbiamo disattivarle con la relativa checkbox a destra del nome nella pila delle Textures.

A questo punto, per ciascuna Texture, che di default sarà di tipo Clouds (e ci va benissimo così), variamo il valore del parametro *Cloud Size*, a piacere, osservando i risultati nella 3D View. Come vi dicevo, potete anche variare i valori di Strength del modificatore, oltre che *Cloud Size* per la Texture.

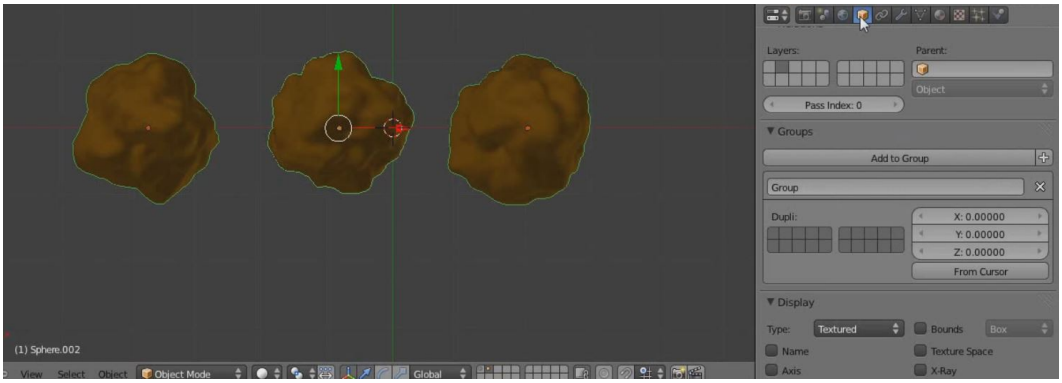


*Applicazione delle Textures ai Displace*

Terminata la modellazione, selezioniamo la prima sfera, apriamo la scheda Materials, impostiamo il colore che vogliamo dare agli oggetti (a piacere, anche se verosimilmente la scelta andrà su un marrone o un arancione scuro) e portiamo a 0.0 l'intensità di Specular.

Applichiamo il Material anche alle altre sfere... ora, qui gli elementi sono pochi e possiamo fare questa operazione un elemento alla volta, ma una scorciatoia consiste nel selezionare gli oggetti da influenzare (con una selezione multipla), selezionando poi l'oggetto che possiede la proprietà da copiare (sempre con una selezione multipla, ma per ultimo, in modo da renderlo oggetto selezionato *e* attivo), premere *CTRL L* (per *Link*, collega) e scegliere, in questo caso, Materials dal menù che apparirà a video.

A questo punto selezioniamo le sfere e premiamo *CTRL G* per creare un *Group*, un gruppo: questo ci serve perché per definire le particelle del *Particle Emitter* diremo a Blender di prendere proprio il *Group*, ossia di creare copie degli elementi appena definiti, per tutte le particelle.

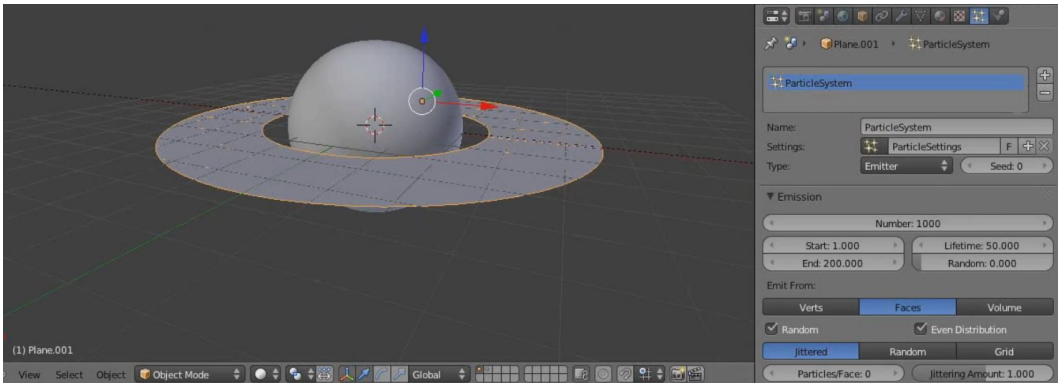


*Creazione del Group*

Creato il gruppo, possiamo anche tornare al primo Layer, quello della scena.

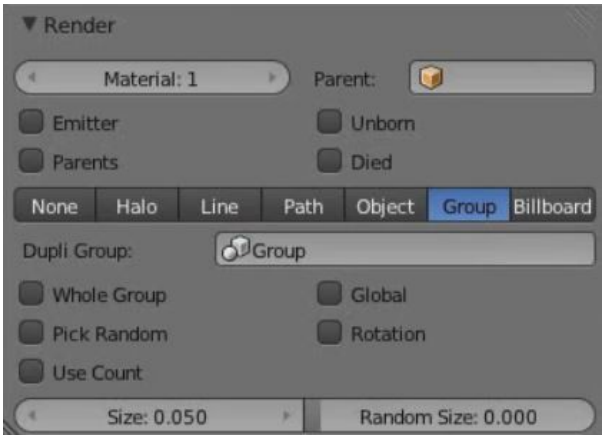
Per creare velocemente il solido che conterrà le particelle, selezioniamo l'anello creato nella puntata precedente e duplichiamolo, in Object Mode, con SHIFT D, lasciandolo sul posto.

Con il nuovo oggetto selezionato, andiamo nella scheda Particles e aggiungiamo un sistema particellare *Emitter*.



*Emitter per la "copia" dell'anello*

Per il momento, impostiamo Amount a 100, poi impostiamo 1 sia per Start che per End frame (per cui tutte le particelle saranno visibili dal primo momento) e, come Lifetime, sempre per il momento, la durata del videoclip (nel mio caso, si trattava di 500 frames, comunque è il valore usato come End Frame nella Timeline).

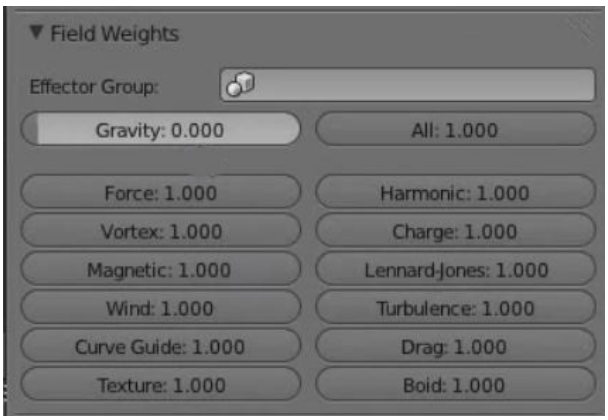


*Dupli Group per il sistema Emitter*

Nella sezione Render, deselezioniamo la casella *Emitter* (non vogliamo ri-renderizzare l'anello emettitore), scegliamo poi *Group* e, nella casella *Dupli Group* che apparirà, il nome del gruppo creato nel secondo Layer (che di default sarà appunto "*Group*").

Qualora le copie dovessero essere troppo grandi o troppo piccole (anche se in teoria DEVONO essere molto piccole), possiamo variarne le dimensioni con *Size* nella sezione *Physics* del sistema particellare.

Ora come ora, le particelle cadono nel vuoto; è vero che presto aggiungeremo un Curve Guide e quindi le confineremo in un percorso, ma in generale per risolvere il problema della caduta è sufficiente aprire la sezione *Field Weights* del sistema particellare e portare a 0.0 il valore di *Gravity*.



*Field Weights - Gravity*

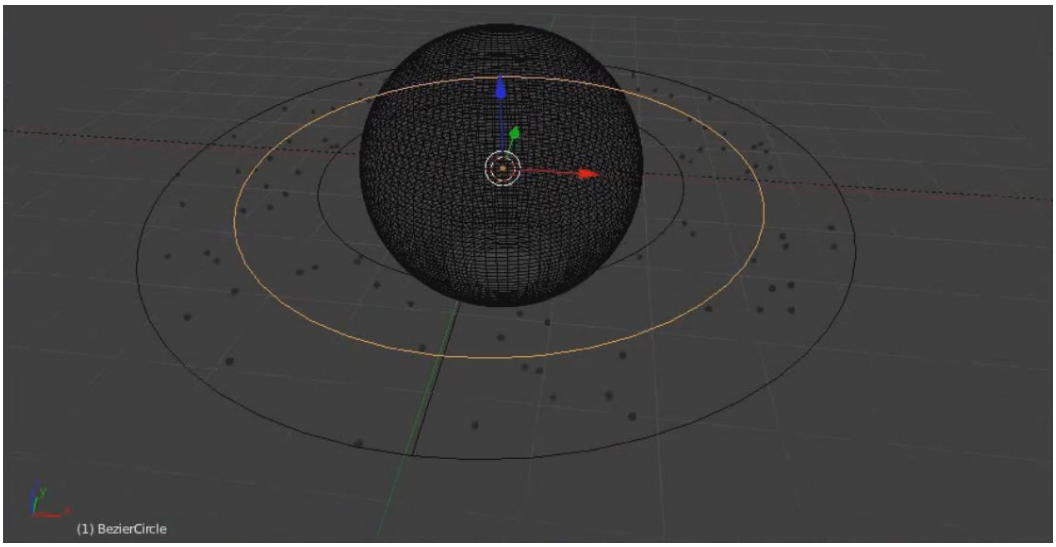
Torniamo nella parte alta della scheda: nella sezione *Emit From*, lasciamo *Faces* ma selezioniamo, sotto, *Random*, cioè "casuale".

Appena sotto, nella sezione *Velocity*, impostiamo 0.0 per *Normal*: non vogliamo proiettare i frammenti nello spazio!

Attiviamo la sezione *Rotation*, quindi al suo interno attiviamo la sezione *Dynamic*, portiamo a 1.0 tutti i campi numerici presenti in questa sezione e cambiamo *Initial Orientation* in *Normal*.

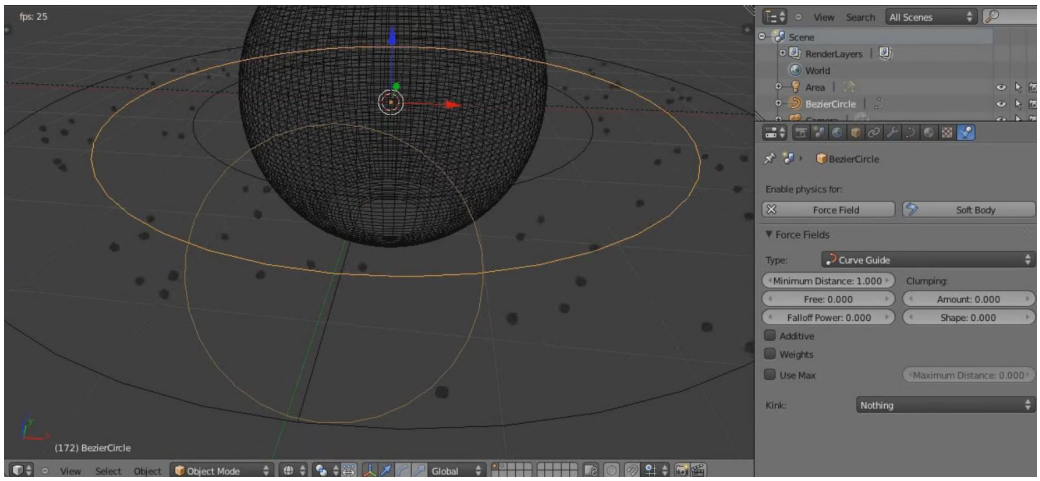
Queste impostazioni fanno ruotare le particelle sul posto, non intorno al pianeta; per definire quest'ultimo tipo di trasformazione, ricorreremo ad un *Curve Guide*, che ci servirà anche per confinare le particelle sul piano (gli anelli, e qui cito wiki, sono composti da particelle molto piccole, dal micrometro al metro, e sono molto, molto sottili, appena una decina di metri).

A questo punto, centriamo il 3D Cursor al centro del pianeta (e quindi al centro del sistema di anelli) e inseriamo lì un Bezier Circle, una curva circolare, scalandola in modo da farla passare all'interno dell'anello.




*Inserimento della "Curve Guide" (un Bezier Circle)*

Con tale cerchio (che non verrà renderizzato) selezionato, apriamo la scheda *Physics* e aggiungiamo un *Force Field* di tipo *Curve Guide*.




*Il Force Field "Curve Guide"*

In teoria, l'animazione delle particelle intorno all'anello è pronta: basta premere ALT A per rendersene conto... in pratica, ci sono ancora un paio di considerazioni da fare.

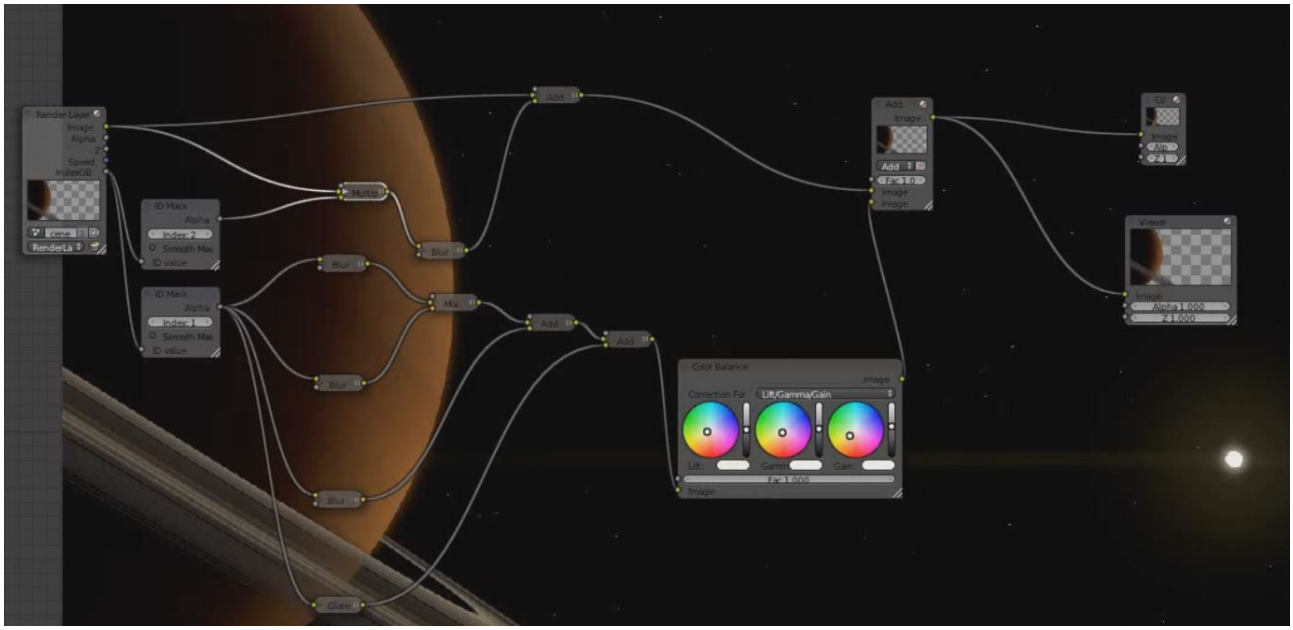
La prima è che per variare la velocità di rotazione delle particelle intorno al pianeta non dobbiamo toccare né la Curva né il Force Field, ma la durata di vita delle particelle: queste effettueranno un giro completo (o meglio: in generale, percorreranno la curva utilizzata come Curve Guide, sia essa una Bezier aperta che un Circle) nel numero di frames definiti in Lifetime, per cui se volete farle muovere molto lentamente dovete aumentare questo valore... se, ad esempio, il vostro videoclip dura 500 frames e volete far compiere alle particelle un quarto di giro, facendole ruotare lentamente intorno al pianeta, dovete impostare Lifetime 2.000 nella scheda dell'Emitter.

L'altra considerazione riguarda numero e dimensioni: le particelle devono essere molto molto piccole, non si devono vedere da lontano, ma solo se la telecamera passa dentro l'anello, quindi abbassate il valore di Size, mentre per quanto riguarda il numero questo va impostato con Amount, NON è possibile utilizzare i Children, i figli, perché questi si comportano in maniera strana con Curve Guide, non rispettando il percorso... purtroppo, un valore alto per Amount rallenta sensibilmente il programma, per cui il consiglio che vi do è di lavorare con un valore basso (ecco perché vi ho detto di porlo inizialmente a 100) e di alzarlo proprio un attimo prima del rendering finale (per la cronaca, ho realizzato il mio rendering con 25.000 particelle, che però non avevano così tanti vertici come i modelli usati per il gruppo, anzi avevano poche decine di vertici, e questo è un aspetto da tenere a mente... vi ricordo che non è possibile utilizzare i Children).

Ok, con la definizione della scena “in sé”, cioè per luci, materiali, modelli, Textures e animazioni, è tutto; passiamo quindi ai Nodes in Compositing.

*  *  *

Lo schema dei Nodes è molto ampio, per cui anziché ricrearlo lo analizzerò con voi qui, in modo da mostrarvi il flusso dei dati.

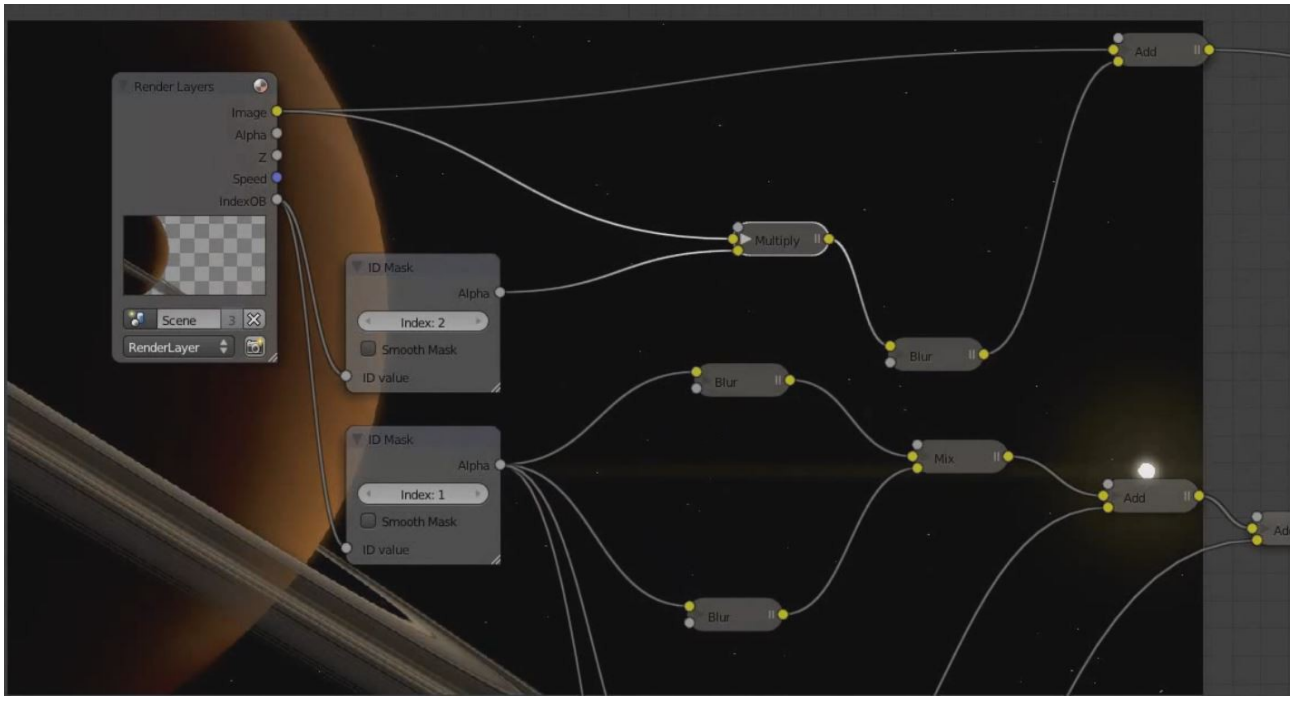



*Primo screenshot dello schema di Nodi di Compositing: si riconoscono gli effetti Blur e Glare, a sinistra, per le maschere ID Mask, sommati poi in cascata tra loro e all'immagine in arrivo dal rendering*

Prima di iniziare, un breve ripasso dalla puntata precedente: abbiamo attivato, nella sezione Layers della scheda Render, nella Properties Window, la voce *Object Index*, in modo da prelevare un indice numerico dal Nodo *Render Layers* in fase di Compositing per isolare il pianeta e il cerchio che rappresenta il Sole e trattarli a parte; proprio per il pianeta e per il Sole abbiamo quindi impostato dei *Pass Index* (i suddetti valori numerici) nella scheda Object, impostando in particolare 1 per il Sole e 2 per il pianeta.

Passiamo quindi alla schermata Nodes Editor, in modalità *Compositing Nodes*.

Dalla porta *Index OB* di Render Layers escono due collegamenti diretti a due Nodi *ID Mask*, per ottenere le maschere Alpha del Sole e del pianeta; seguendo il flusso del Nodo con ID 2, il pianeta, vediamo che viene moltiplicato con l'immagine in arrivo da Render Layers, in modo da ottenere, sostanzialmente, il pianeta colorato su sfondo nero (come detto nella prima parte, qui mi limiterò a descrivere lo schema dei nodi, non a spiegare le funzioni di base, per le quali rimando all'ebook sul Compositing e ai tutorial su tale argomento presenti nel sito e nel canale Youtube).




*Particolare della sezione ID Mask*

Ottenuta quindi la maschera a colori del solo pianeta, la sfochiamo un po' con un Nodo *Blur*, in modo da ottenere una sfumatura sull'atmosfera, un effetto "soft glow", da sommare poi all'immagine originale in arrivo da *Render Layers* con un Nodo *Mix Add*, con Factor 0.5.

L'uscita del Nodo *ID Mask* con indice 1, quella del Sole, non viene moltiplicata all'originale, in quanto il colore dell'effetto verrà regolato qui in Compositing, come vedremo a breve; dal Nodo *ID Mask* escono invece ben 4 collegamenti: 3 per altrettanti nodi *Blur*, per aggiungere hard e soft glow, 1 per un nodo *Glare*, per aggiungere gli effetti Ghost, responsabili dei bagliori lenticolari che si vedono nella parte finale del videoclip.

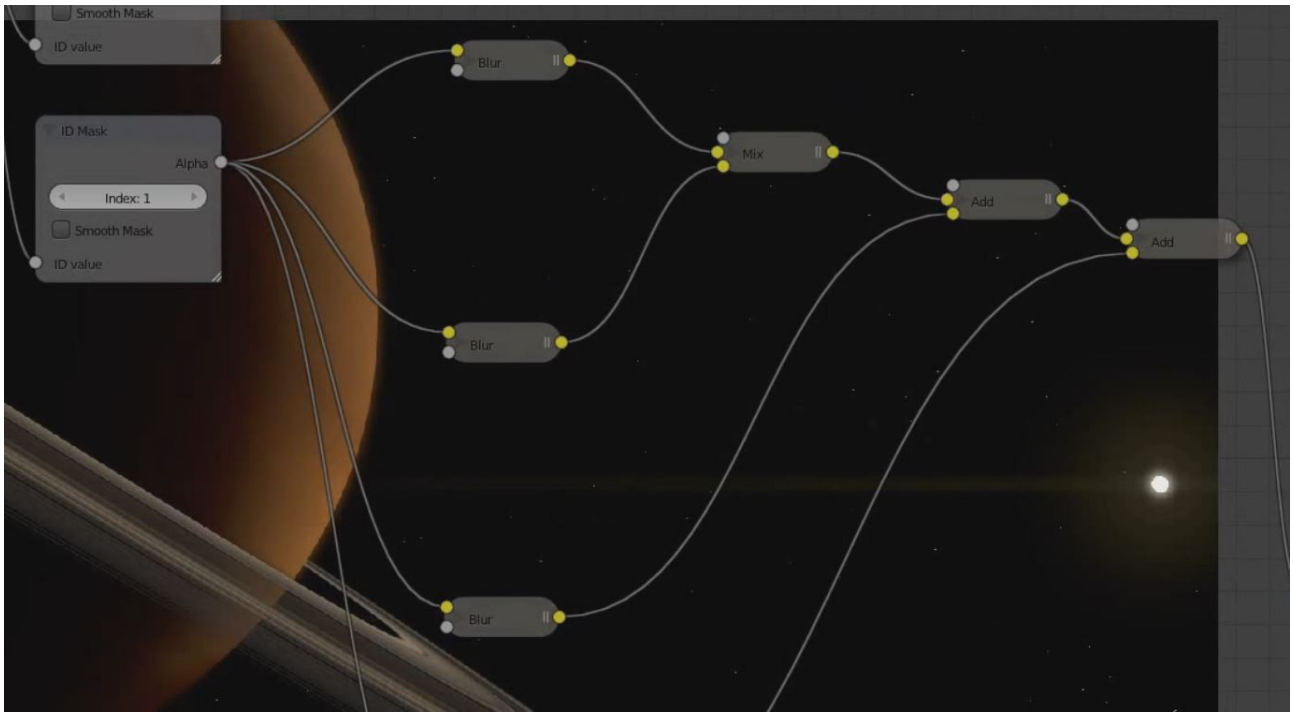
I bagliori prodotti da Glare non sono quindi "animati", nel senso che si spostano automaticamente cambiando il punto di osservazione, l'unica cosa da fare è impostare le iterazioni e soprattutto la Threshold, che ho portato a 0.0, dopodiché l'effetto va sommato al resto.

Tutti i Nodi *Blur* sono in modalità Fast Gaussian, a variare sono i valori X e Y, per produrre effetti diversi.

Il primo Nodo *Blur* è responsabile del bagliore anamorfico, ossia della "striscia" orizzontale: ecco perché per X c'è un valore esagerato, nel mio caso 1.000, mentre per Y il valore è appena 5.

Il secondo Nodo *Blur* produce un bagliore ristretto, per sfumare i bordi del Circle che rappresenta il Sole, infatti i suoi valori X e Y sono entrambi 10.

Il terzo Nodo *Blur* produce invece il soft glow, un bagliore più ampio e più tenue, per cui i valori X e Y sono entrambi a 100.



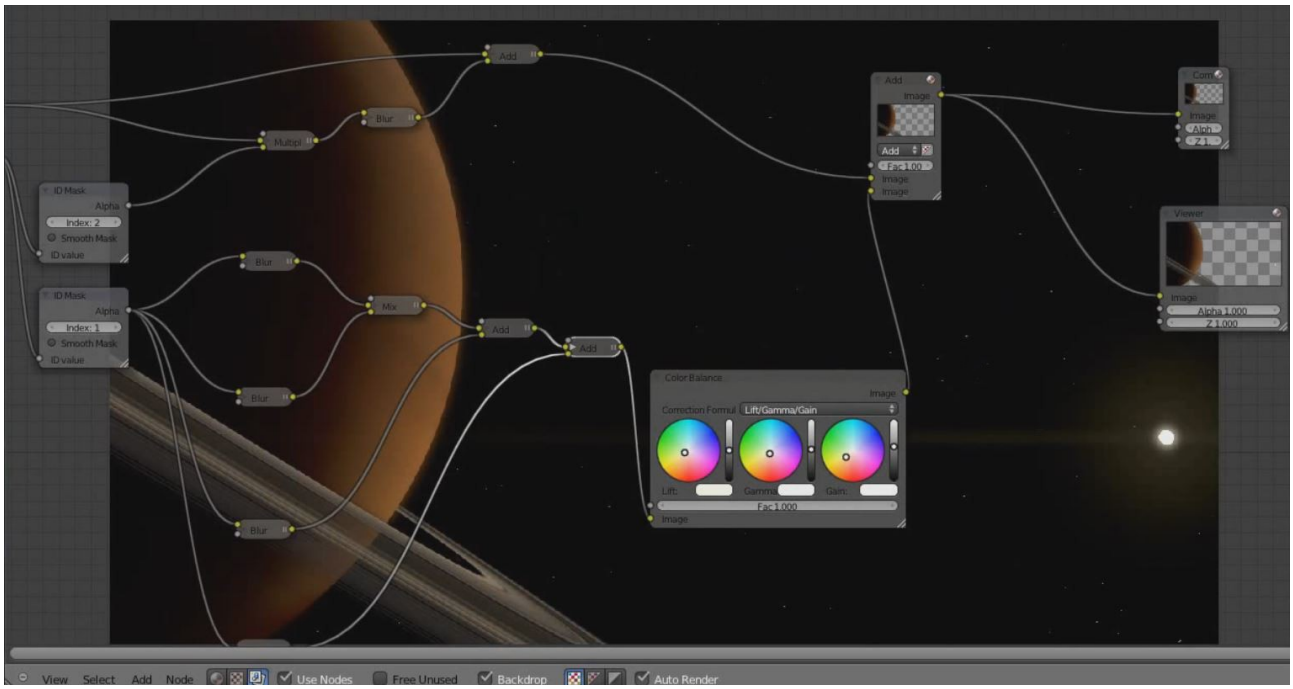

*Nodi BLUR, sommati in cascata*

Questi nodi vanno ovviamente sommati tra loro, ecco perché abbiamo in cascata tre nodi *Mix*, ma si può lavorare anche su questa fase, infatti nel mio caso il primo Nodo *Mix*, che mescola il bagliore anamorfico e la sfocatura ridotta, è di tipo *Mix* con Factor 0.7, mentre gli altri due sono degli Add, comunque queste non sono regole, anzi vi invito a provare con varie modalità e cambiando i Factor, osservando di volta in volta i risultati.
All'uscita del terzo Nodo di mixaggio, comunque, il bagliore prodotto è bianco, perché arriva da una maschera *ID Mask*, che è in scala di grigi; per colorarlo, utilizziamo un nodo *Color Balance*, e per un valido motivo: questo nodo consente di definire la tinta in maniera separata per le zone con tonalità media (Gamma), per le zone scure o "in ombra" (Lift) e per le zone illuminate direttamente,

dotate di una forte intensità (Gain), per cui con un unico strumento possiamo definire delle vere e proprie sfumature colorate sulle sfocature e gli effetti lenticolari in arrivo dai vari nodi Blur e Glare.

*Color Balance*

Fatto questo, non ci resta che sommare il risultato a quello in arrivo dalla parte superiore dello schema, che poi è la somma dell'immagine originale e della sfocatura del pianeta, ottenendo così il rendering finale.




*La somma finale*

*  *  *

# Mappare textures Equirectangular e Angmap su World

Questo è un tutorial sulla mappatura di due tipi di immagini, dette *AngMap* (mappe angolari) e *Equirectangular*, come sfondi dell'universo virtuale (World) in Blender 3D, sia col motore interno (Blender Render) che con Cycles.

Questa mappatura consente di realizzare uno sfondo a 360° mediante fotografie di ambienti reali, operazione molto utile in certi casi ma che non è propriamente intuitiva.

Vedremo quattro combinazioni, ossia *Equirectangular* o *AngMap* con Blender Internal e Cycles, iniziando con il motore di rendering interno, Blender Render.

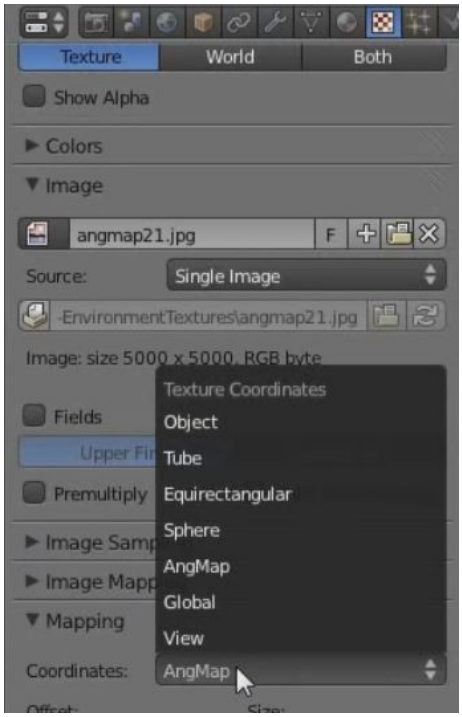
Per le prove, sto utilizzando immagini prese online; cercate, ad esempio, "*Mirror ball texture*", o "*equirectangular background*", "*Equirectangular panorama*", "*AngMap texture*" e così via.

Ok, iniziamo con Blender Render!
La prima cosa da fare è selezionare la scheda World nella Properties Window, quindi passare alla scheda Texture, in modo da creare una Texture che verrà associata, appunto, allo sfondo dell'universo virtuale.

Aggiungiamo quindi una Texture *Image or Movie* e carichiamo un'immagine da disco o, se già presente nel file del progetto, mediante il pulsante *Browse Image to be Linked*.

Le immagini *Equirectangular* sono immagini rettangolari che rappresentano panoramiche a 360° di un ambiente; nella metà superiore troviamo quindi dall'orizzonte al cielo, mentre nella metà inferiore abbiamo il terreno (o nero, se il pavimento, per così dire, non è importante, perché ad esempio inseriremo un terreno digitale nella scena).

Le immagini *AngMap*, che in Cycles chiameremo *MirrorBall*, sono immagini di formato quadrato e dall'aspetto peculiare: l'immagine in sé è circolare ed appare schiacciata ai bordi, come le foto prese con un fish-eye... in realtà, si tratta della foto di una *MirrorBall*, una sfera riflettente che riflette, appunto, l'ambiente circostante, e quando la si produce bisogna evitare di fotografare... il fotografo!



*Mapping Coordinates (Textures in Blender Render)*

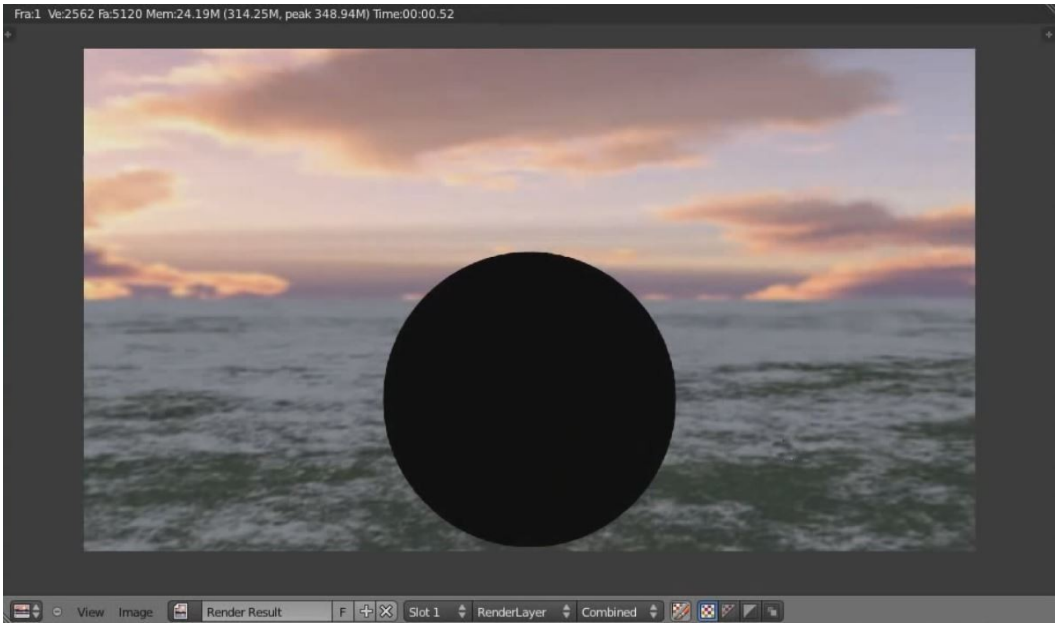
Torniamo comunque alla pratica.

Nella scheda *Mapping*, dobbiamo cambiare la modalità di proiezione mediante il menù *Coordinates: Equirectangular* per le panoramiche rettangolari, *AngMap* per le immagini di tipo *MirrorBall*.

*World Real Sky*

Non abbiamo ancora finito: nella sezione *Influence* della Texture dobbiamo selezionare *Horizon Color*; torniamo quindi nella scheda World e selezioniamo, lì, la voce *Real Sky*, letteralmente “cielo reale”.

Non ci resta che fare dei rendering, con *F12*, cambiando di volta in volta l'orientamento della telecamera per vedere il risultato guardando in direzioni diverse.

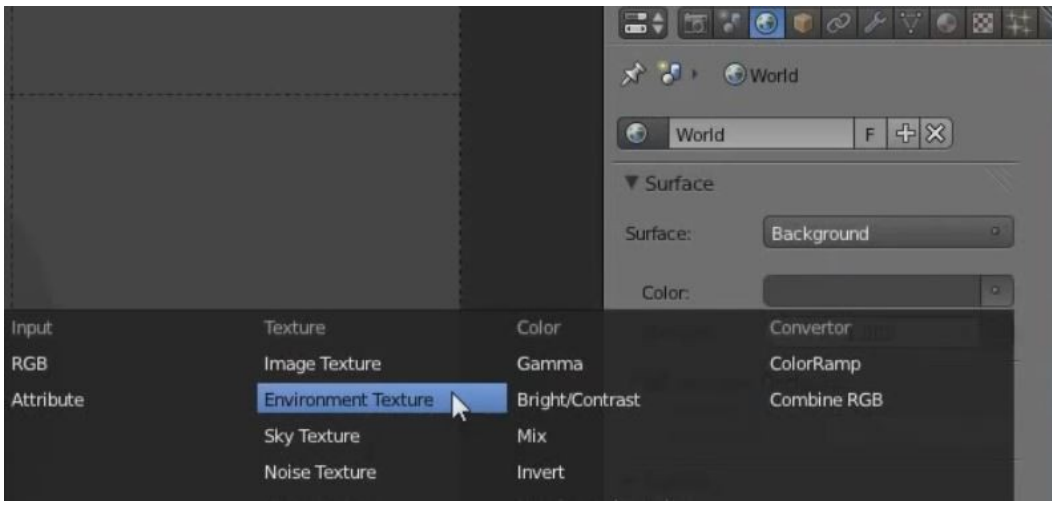


*Rendering d'esempio (Blender Internal)*

Passiamo ora ad una nuova scena con il motore di rendering Cycles, dove le impostazioni di mappatura per una Texture World sono decisamente più semplici, visto che tutto si svolge nella scheda World... non solo: abbiamo la possibilità di vedere l'anteprima in tempo reale nella 3D View, anziché dover avviare ogni volta un rendering come con Blender Render, per cui per prima cosa cambiamo la modalità Viewport Shading della 3D View in *Rendered*.

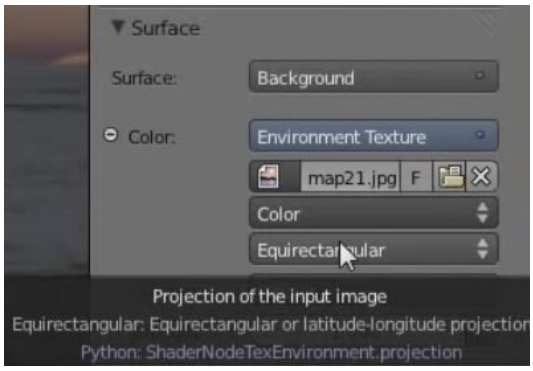
In questo caso, la Texture va associata a World nell'omonima scheda, ma si tratta, anziché di una Texture Image, di una *Environment Texture*, ossia una Texture Ambientale.

Per associare una Texture a World, in questa scheda attiviamo per prima cosa i Nodi, poi facciamo click sulla casella col cerchietto accanto a Color e scegliamo "*Environment Texture*".



*Environment Texture (Cycles)*

La prima cosa da fare è, ovviamente, caricare l'immagine, o con Open e scegliendo poi da disco oppure con *Browse Image to be Linked*, se l'immagine è già presente nel file del progetto, perché caricata precedentemente.

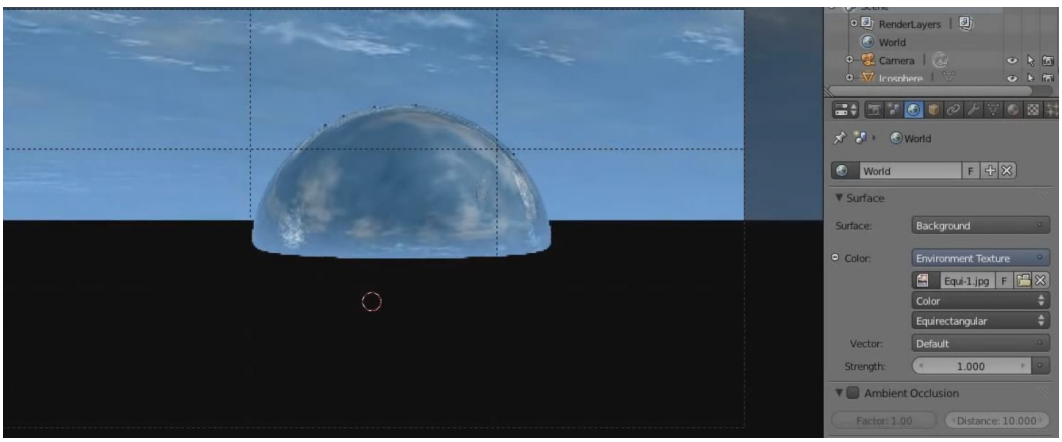


*Projection Mode, in Cycles*

A questo punto la seconda – e ultima – cosa da fare è scegliere la modalità di proiezione nel selettore *Projection: Equirectangular* (voce di default) per le mappe rettangolari panoramiche o *Mirror Ball* per le *AngMap*.

Va detto che una *MirrorBall* realizzerà uno sfondo buono per una certa apertura (dietro e ai lati dell'osservatore), ma non esattamente di fronte, dove verrà introdotta una distorsione evidente.

In realtà, queste distorsioni ci sono anche nelle altre combinazioni di modalità e motori di rendering, ma sono meno evidenti. Per risolvere questo problema si può ricorrere a più tecniche, ad esempio uno schema di Nodi di mappatura dell'immagine originale... ma magari di questo parleremo un'altra volta!



*Esempio di rendering in Cycles*

Come avrete notato, e come potete provare in prima persona cercando online immagini *Equirectangular* o *AngMap* o *MirrorBall* da applicare a World nelle vostre scene, la risoluzione di queste immagini è di fondamentale importanza: dal momento che le foto vengono stirate su tutto lo

sfondo, un'immagine in bassa risoluzione mostrerà effetti di compressione evidenti nel rendering finale. Buone immagini di sfondo possono avere anche più di 6.000 pixel nel lato minore (o per lato, nel caso delle *AngMap*) e occupare diversi MB su disco, quindi se cercate le foto online mettete un filtro di ricerca per cercare solo foto grandi.

Prima di chiudere, una considerazione finale: per rendere le Texture di sfondo degli illuminanti, operazione particolarmente interessante quando l'immagine utilizzata è una foto *HDR*, in formato *EXR* o, appunto, *HDR*, in Cycles non dobbiamo fare niente, perché questa modalità, per come è strutturato il motore di rendering, è implicita; per quanto riguarda il motore interno Blender Render, invece, l'unica cosa da fare è selezionare la casella *Environment Lighting* e impostare il colore da *White* (default) a *Sky Texture*.

*Sky Textures, per usare le HDR in Blender Internal*

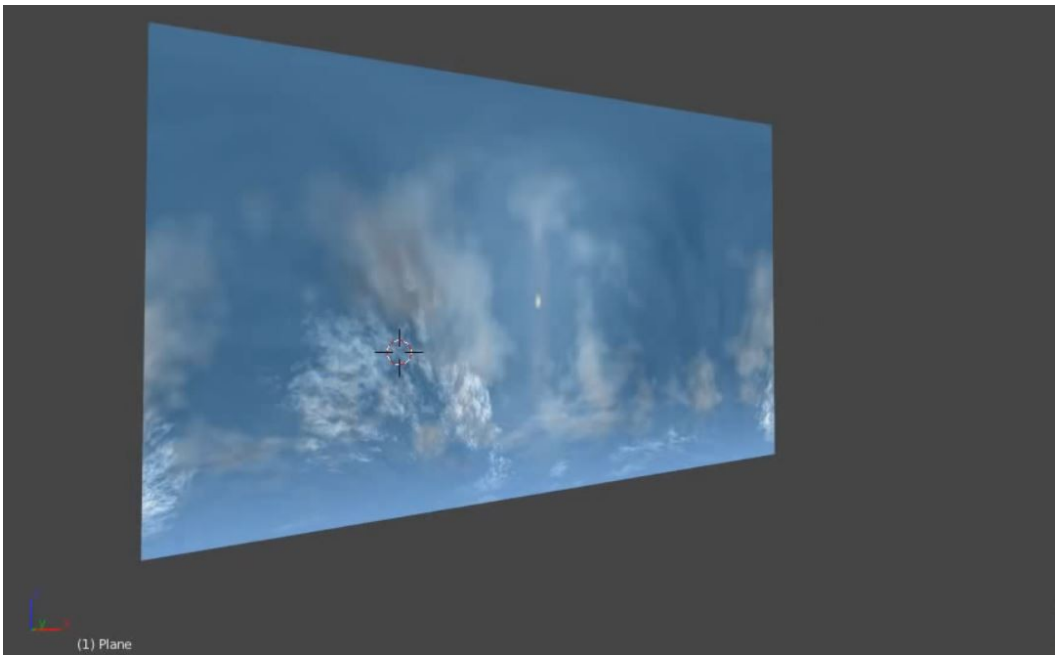
*  *  *

# Ricavare e utilizzare il canale Vertex Paint in Cycles

In questo tutorial vedremo come ricavare e utilizzare le informazioni *Vertex Paint* in Cycles per modificare i Materials degli oggetti; sarà anche un modo per utilizzare il poco noto Nodo *Attribute* in Cycles. La tecnica del *Vertex Paint* consiste nell'andare a dare ai vari vertici della geometria un colore, che però non è direttamente associato al Material ma proprio al vertice che lo possiede e che poi può essere miscelato al Material della superficie in qualche modo.

Da notare che i colori vanno definiti per vertice, poi sarà il software a fare l'interpolazione e creare le sfumature sugli spigoli e le facce tra i vari pixel, prima di procedere alla miscelazione finale.

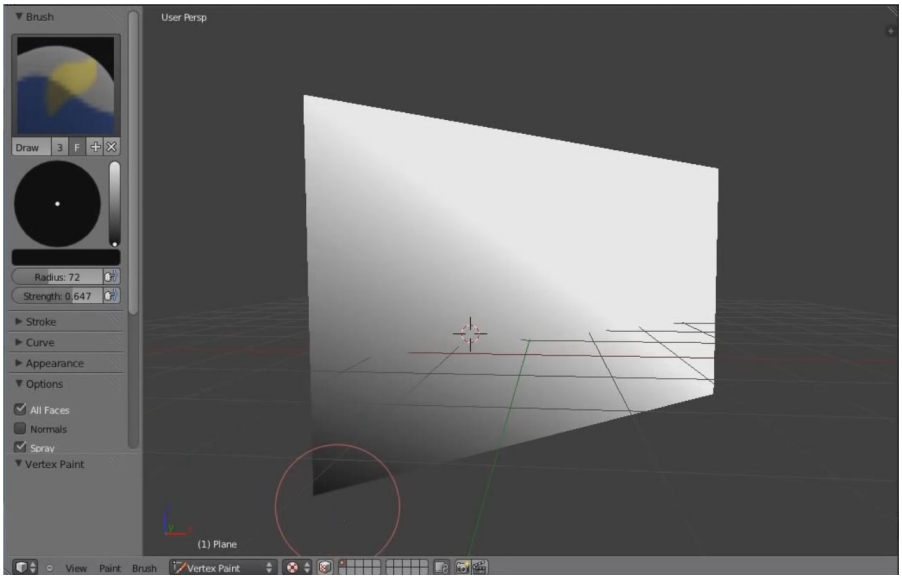
Prendiamo quindi un oggetto, che in questa fase può essere o meno provvisto di un Material; io, per risparmiare tempo, ho già applicato una Texture immagine, con mappatura UV, ad un oggetto della scena, d'altronde questa è un'operazione elementare, per cui passiamo subito al *Vertex Paint*.



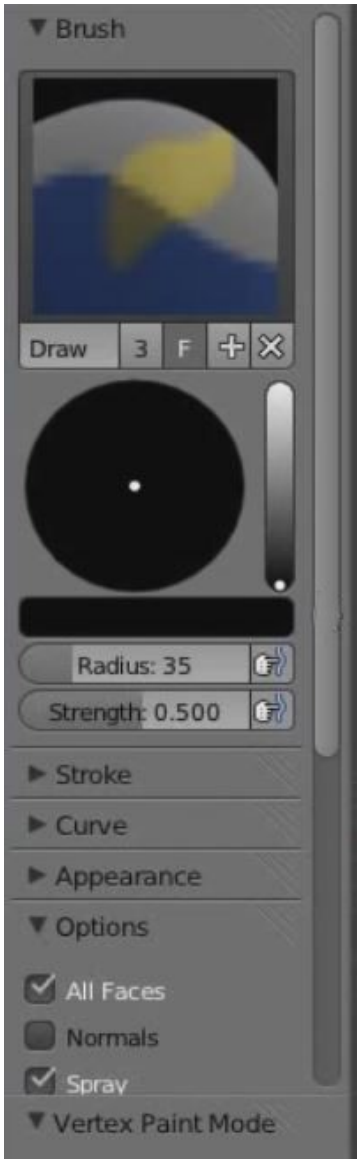
*La scena che utilizzeremo (con l'oggetto texturizzato)*

Dal menù Mode nell'header di una finestra 3D View, scegliamo la modalità *Vertex Paint*. Qualora dovesse essere chiusa, apriamo la Tool Shelf nella 3D View, con *T*; tale finestra mostra ora gli strumenti propri della pittura *Vertex Paint*.

Impostiamo ad esempio il colore a nero, Strength (intensità) massima e Size, dimensione del pennello, a piacere, quindi facciamo click col tasto sinistro del mouse su un angolo della mesh, nella 3D View, per "dipingere". Piccola sorpresa: verrà pitturato di nero solo l'angolo, mentre spostando il mouse verso il centro mentre si tiene premuto il tasto sinistro non succede niente... perché?




*Primo "problema"*



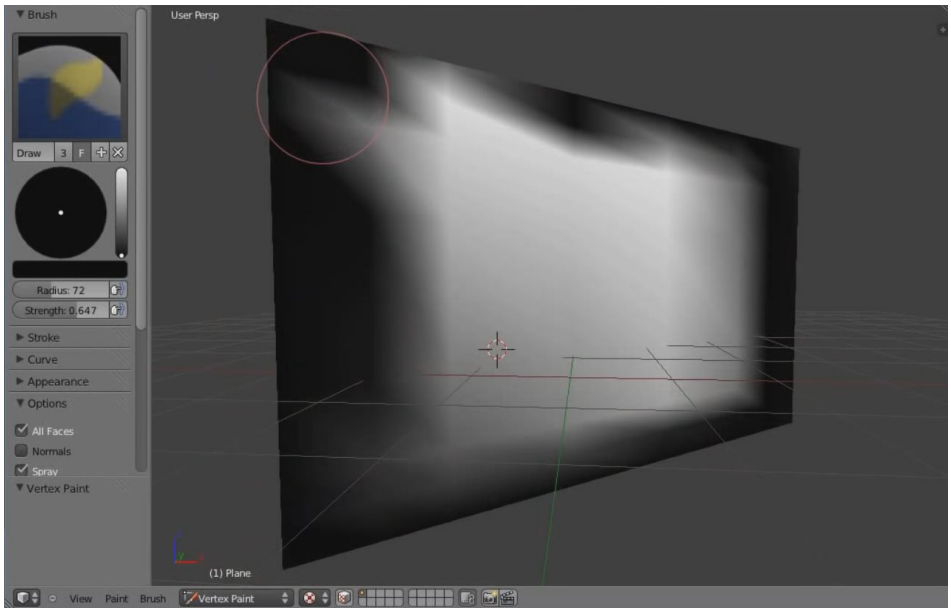

*Strumenti Vertex Paint*

Perché lì non ci sono vertici, per cui, un po' come avviene con lo *Sculpt Mode*, non è possibile inserire queste informazioni nella mesh, che ha una "bassa risoluzione", in un certo senso.

Vi consiglio di non ricorrere subito a Subdivide per incrementare il numero di vertici di una mesh, in quanto l'incremento potrebbe risultare eccessivo: a seconda dei casi, provate prima con Ring Cut (con CTRL R) o con lo strumento taglierino (Knife), quindi utilizzate Subdivide come ultima risorsa e comunque solo in una sottoselezione (la parte che vi serve) di una mesh.

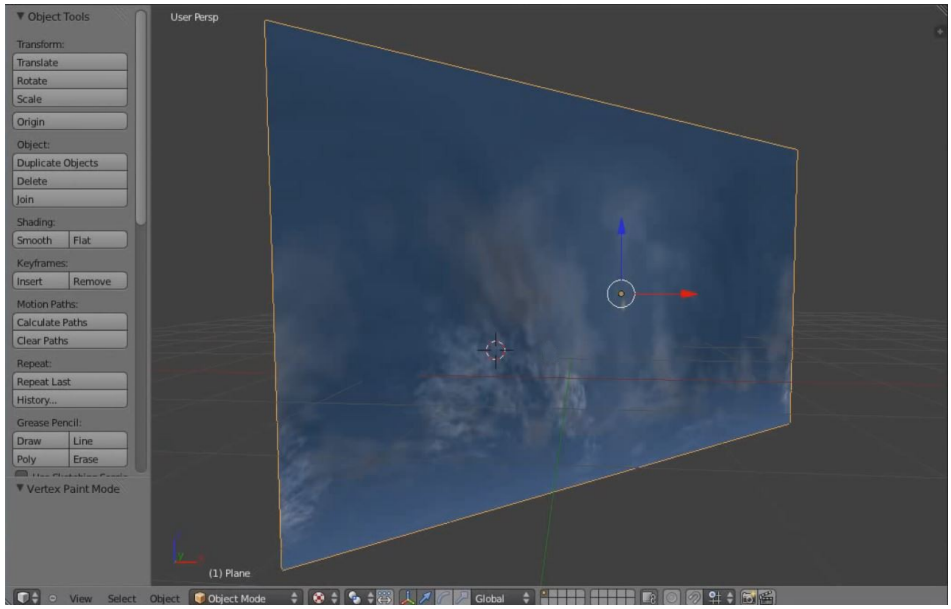
I nuovi vertici vanno aggiunti, ovviamente, in Edit Mode.

Fatta questa operazione, torniamo in modalità *Vertex Paint* e dipingiamo. Nel mio caso, sto utilizzando solo la scala di grigi (ed in particolare il nero) per scurire i bordi della mesh, quasi a voler creare un effetto vignettatura, ma ovviamente potete utilizzare, per i vostri lavori, varie intensità, dimensioni del pennello e colori, tutte voci disponibili nella Tool Shelf dello strumento.




*Dipingere in Vertex Paint*

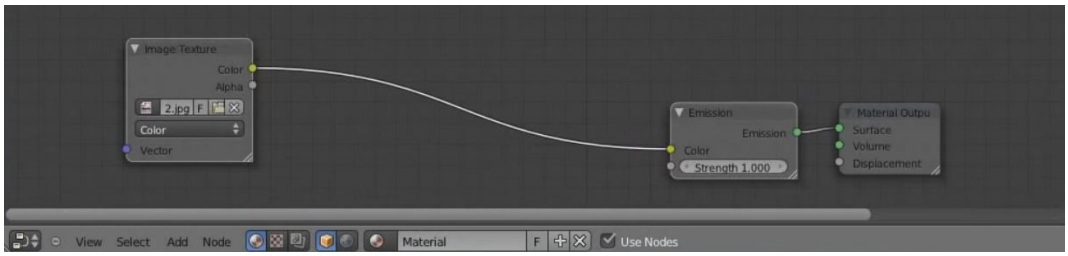
Torniamo in Object Mode e passiamo in modalità di visualizzazione Rendered, nel selettore Viewport Shading della 3D View, per avere un'anteprima in tempo reale del risultato... risultato che, però, non c'è: a quanto pare, non è cambiato niente!




*Preview Rendered: non è cambiato niente*

Diamo un'occhiata alla sezione *Vertex Colors* della scheda Object Data, nella Properties Window: c'è un nuovo canale, *Col*, che è stato aggiunto automaticamente da Blender non appena abbiamo iniziato a dipingere in modalità *Vertex Paint*... ricordiamoci il nome di tale canale (che, comunque, possiamo rinominare a piacere, se vogliamo).

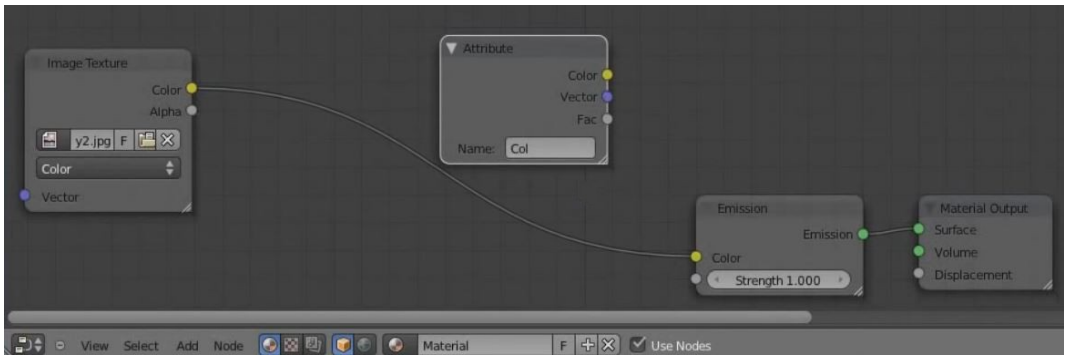
Splittiamo la 3D View, aggiungendo una finestra *Nodes Editor* in modalità *Materials* Node: in questo modo potremo vedere l'Editor dei Nodi della superficie e, contemporaneamente, l'anteprima in tempo reale del risultato, nella 3D View con modalità Rendered.




*Material Nodes*

Nell'Editor troviamo, collegati in successione, un Nodo *Image Texture*, un Nodo *Emission* (infatti ho scelto questo Shader per la mesh) e il Nodo *Material Out* che definisce l'aspetto finale del Material associato all'oggetto.

A questa configurazione aggiungiamo un Nodo di tipo *Input Attribute*.



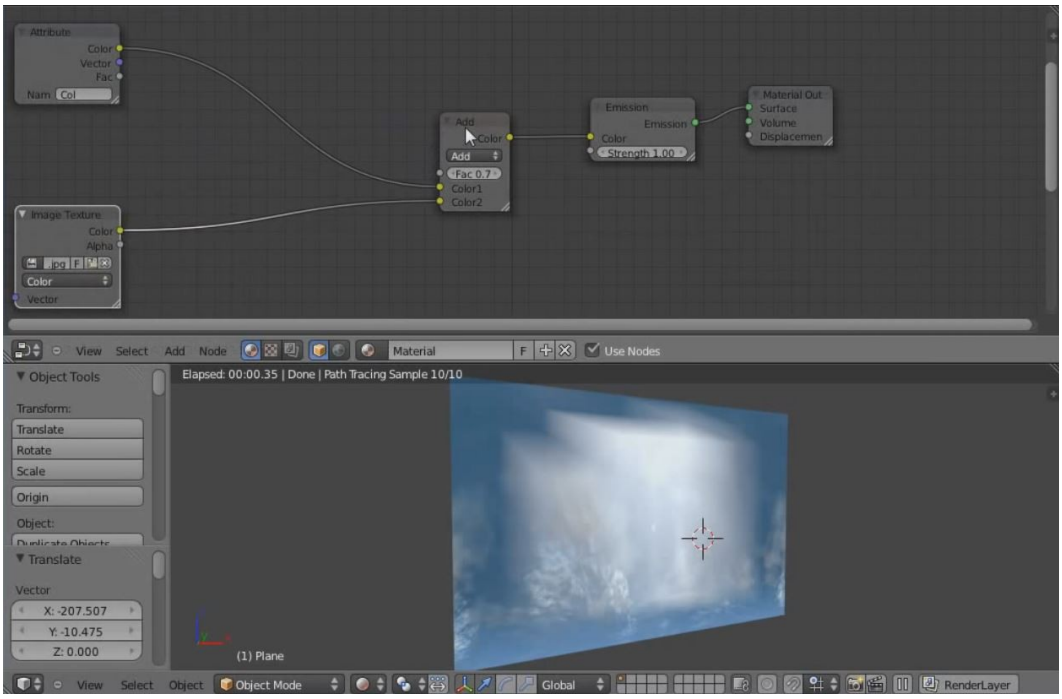

*Aggiungere il Nodo Input Attribute*

In questo Nodo, nel campo *Name*, scriviamo il nome dell'attributo, proprietà o canale informazione presente nel nostro oggetto che definisce le informazioni che vogliamo prelevare; nel nostro caso, tale nome è quello del canale *Vertex Color*, per cui scriviamo appunto "*Col*".

Dal Nodo escono tre tipi diversi di canali; ciò è dovuto al fatto che Blender non conosce a priori la natura dell'informazione che vogliamo prelevare, quindi in un certo senso mette le mani avanti e presenta un tipo colore, un vettore e un fattore di intensità.

Anche se abbiamo usato la scala di grigi sulla mesh, il canale Vertex Paint restituisce un'informazione di tipo Color, per cui è quella l'uscita che utilizzeremo.

Come primo test, togliamo il collegamento tra *Image Texture* e *Emission* e colleghiamo, sull'input Color dello Shader, l'output Color di *Attribute*: il colore della superficie è ora il colore in arrivo dal layer *Vertex Paint*, che possiamo ancora modificare nella 3D View, in modalità *Vertex Paint* e Viewport Rendered.

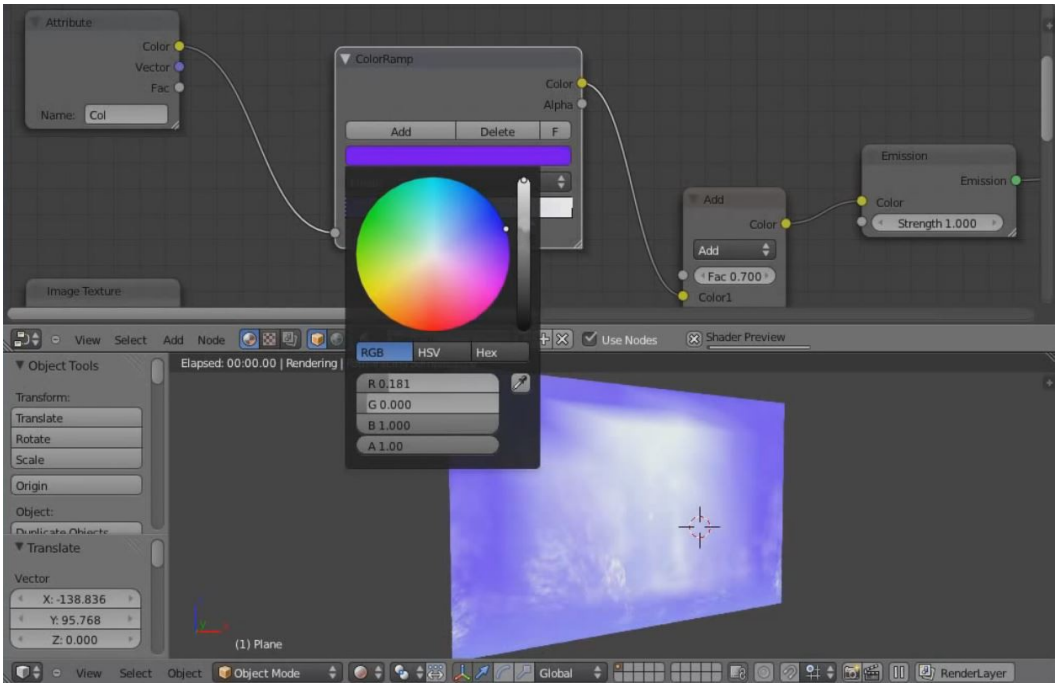



*Utilizzo del Nodo Attribute per ricavare il Vertex Paint*

A questo punto, abbiamo l'imbarazzo della scelta sulla modalità di miscelazione di tale canale con la Texture di partenza. La tecnica base, ma che offre già una buona flessibilità per via delle modalità che mette a disposizione, è l'utilizzo del Nodo *Color Mix* (da non confondere con Shader *Mix*), che miscelerà quindi i colori in arrivo dal Nodo *Attribute Col* e da *Image Texture*, ad esempio con modalità Add, Mix, Multiply, eccetera, con fattore di miscelazione a piacere (possiamo sempre dare un'occhiata al risultato nella 3D View in tempo reale).

Non c'è limite all'utilizzo di Nodi informazione (ad esempio, altre Textures) o di miscelazione intermedia, né di altre tecniche... ad esempio, supponiamo di voler cambiare la tinta del *Vertex Paint* in maniera uniforme, ad esempio dal nero al blu, senza però dover ridipingere in *Vertex Paint Mode* (anche perché la mesh potrebbe essere molto più complessa di questa): che fare?

La soluzione consiste nell'inserire un Nodo di tipo *Convertor – Color Ramp* all'uscita del Nodo *Attribute* e definire lì la rimappatura del colore.



*Utilizzo del Nodo Convertor - Color Ramp*

Nel nostro esempio, il colore va dal nero (utilizzato ai bordi) al bianco di default, per cui facciamo click sulla banda nera all'estremità sinistra della Ramp e cambiamo questo colore con la Color Box che apparirà cliccando sulla casella del colore sopra.

Ricapitolando, quello che bisogna fare per ricavare le informazioni di *Vertex Painting* di una mesh è inserire un Nodo *Attribute* nell'Editor dei Nodi del Material e specificare, come nome dell'attributo, il nome del layer *Vertex Paint*, che si trova nella relativa sezione all'interno della scheda Object Data, nella Properties Window; da lì, potete utilizzare tutti i Nodi che volete per variare tonalità, modalità di miscelazione e altri effetti di composizione.

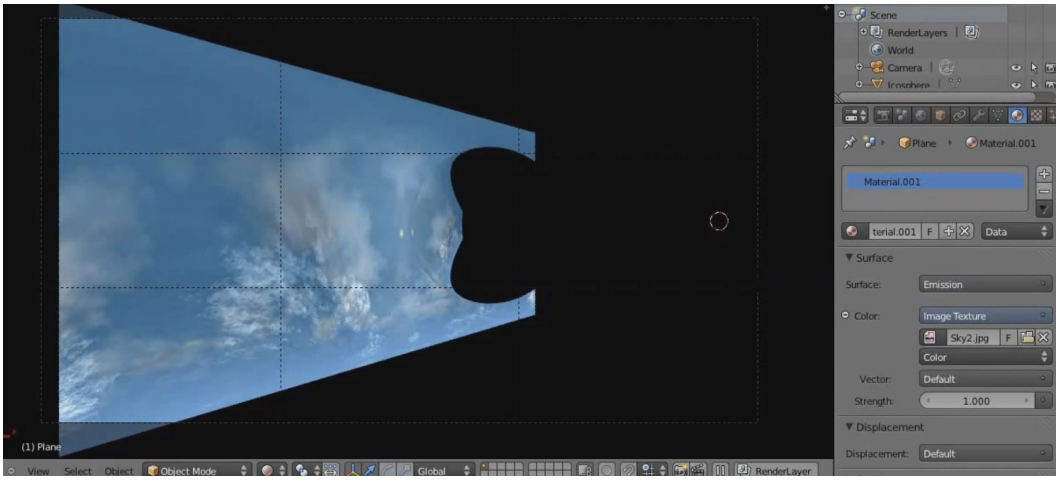
*  *  *

# Realizzare un Material Shadeless in Cycles

In questo brevissimo tutorial vedremo la configurazione dei Nodi da utilizzare per ottenere un Material “Shadeless” in Cycles per Blender 3D.

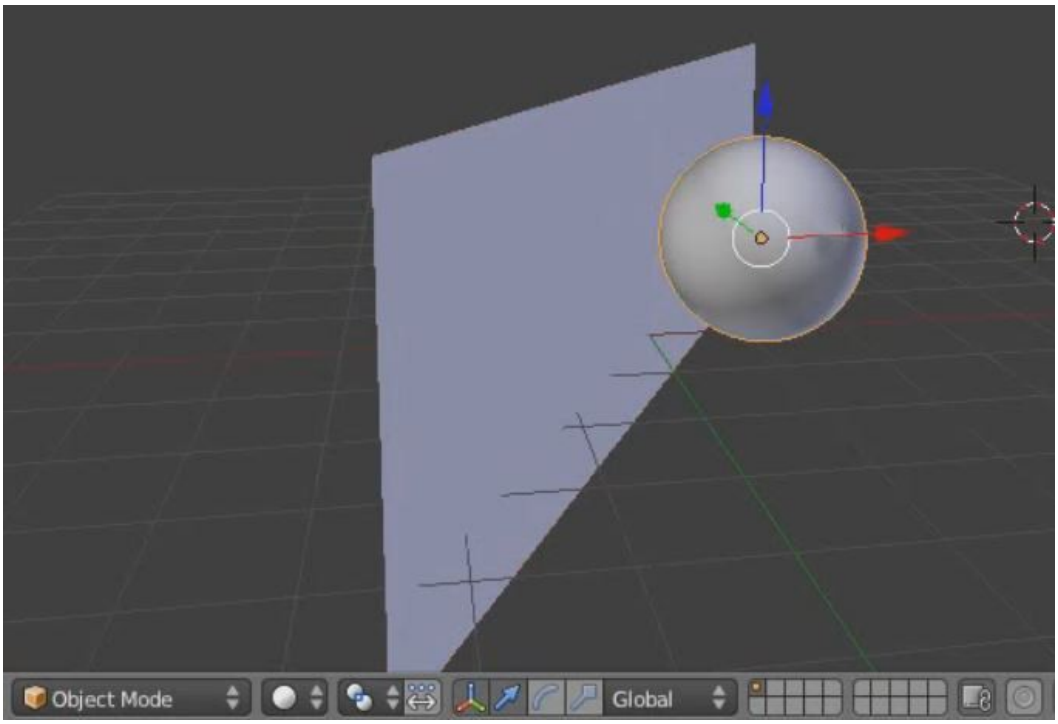
Non spiegherò il funzionamento del Nodo *Light Path* di Cycles, né quello dei Nodi in generale, mostrando invece direttamente lo schema dei Nodi da implementare per arrivare al risultato voluto; per una trattazione sui Nodi in Cycles, rimando ai tutorial di base su tale argomento e, ancora meglio, all'ebook.

Torniamo all'argomento di questo tutorial.
Vogliamo una superficie “Shadeless”, ossia una superficie sempre visibile con intensità uniforme su ogni punto e con qualsiasi orientamento rispetto all'osservatore, non ombreggiata (quindi insensibile alle fonti di luce o all'illuminazione indiretta della scena) e non emettitrice, ossia che non illumini gli oggetti vicini. Il punto di partenza per ottenere questo risultato è uno Shader *Emission*, tuttavia (come mostra l'immagine seguente) Emission, per definizione, emette luce, quindi illumina gli oggetti circostanti.



*Utilizzo di uno Shader Emission*

La scena è composta da un Plane (che vogliamo rendere Shadeless) con Shader Emission e, di fronte, una sfera con Shader Glossy, quindi altamente riflettente. Non ci sono altre fonti di luce e lo sfondo dell'universo virtuale è nero.

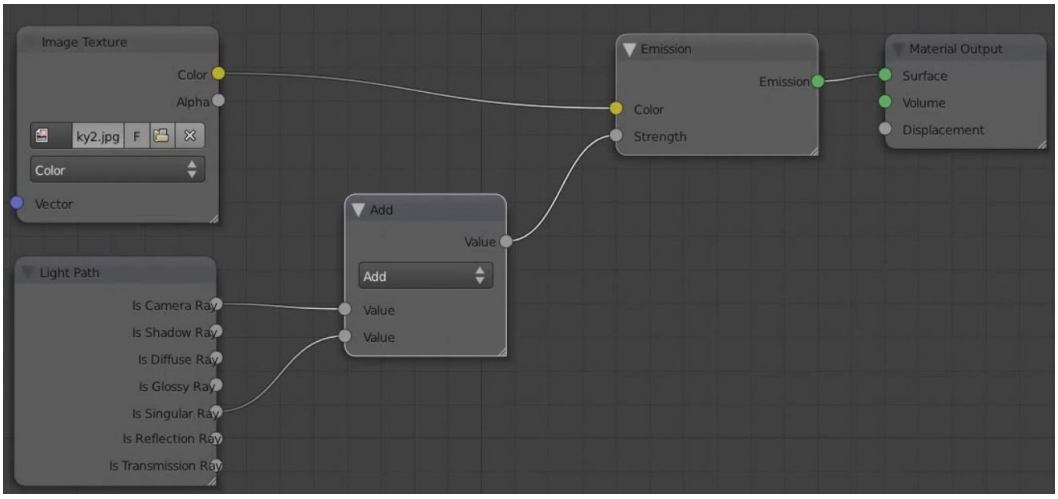


*La scena utilizzata in questo tutorial*

Apriamo una finestra Nodes Compositing per esaminare lo schema dei Nodi del Material del Plane.

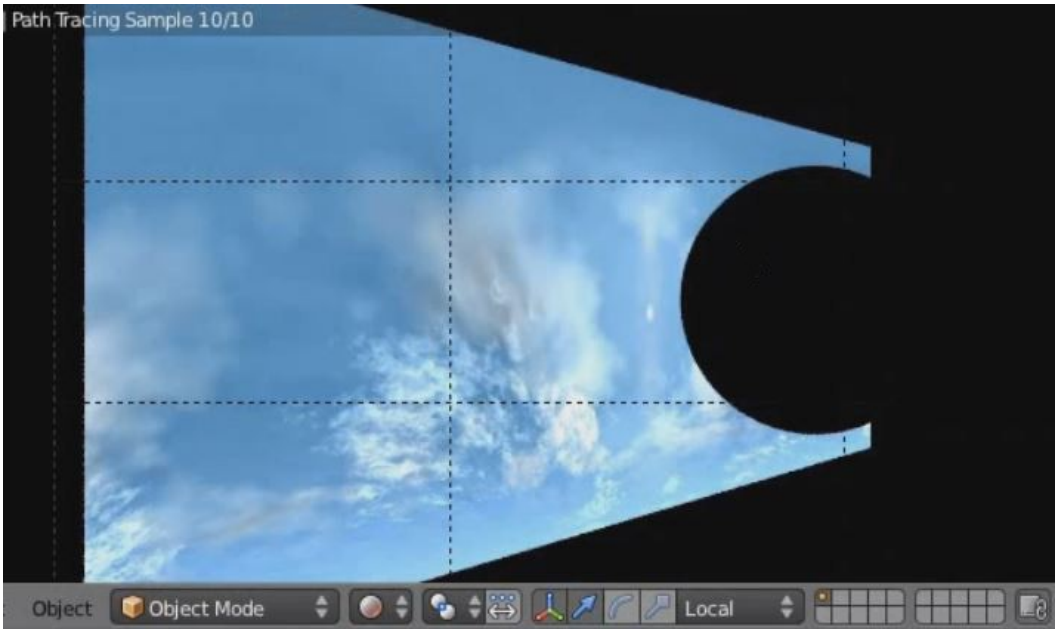
Quello che bisogna fare, qui, è aggiungere un Nodo *Input – Light Path* e un Nodo *Convertor - Math* in modalità Add (da non confondere con un Color Mix in modalità Add o con un Nodo *Shader – Add*). Colleghiamo le uscite "*Is Camera Ray*" e "*Is Singular Ray*" sugli ingressi Value del Nodo *Math – Add*.

A questo punto, colleghiamo l'uscita del Nodo *Add* sull'input Strength, ossia intensità, del Nodo *Emission* (che, tra parentesi, in questo caso ha una Texture immagine su Color, ma potrebbe trattarsi anche di una Texture procedurale o di un colore RGB, è indifferente).




*Lo schema dei Nodi del Material*

In questo modo, l'emissione luminosa si avrà solo per i pixel direttamente visibili dalla telecamera, in modo da non illuminare altri oggetti; Emission, infine, renderà la superficie insensibile alle fonti di illuminazione esterne, quindi avremo una superficie non ombreggiata (possiamo fare una prova, ad esempio portando a bianco il colore di World nella relativa scheda). Tutte queste caratteristiche ci restituiscono, sostanzialmente, una superficie Shadeless, come desiderato.




*Screenshot del risultato finale*

*  *  *

# Utilizzare i Limits nel modificatore Simple Deform

In questo breve tutorial parleremo di una funzionalità poco nota dei modificatori Simple Deform, quali Twist e Bend, in Blender 3D: i *Limits*.

Il tutorial è di difficoltà media in quanto dà per scontata la conoscenza del modificatore Simple Deform e anche una buona comprensione del concetto di sistema di riferimento locale degli oggetti; per un'introduzione a questi temi, rimando alle prime puntate del videocorso di base su Blender 3D 2.5, valido anche per la versione 2.63 (utilizzata per realizzare questo tutorial).

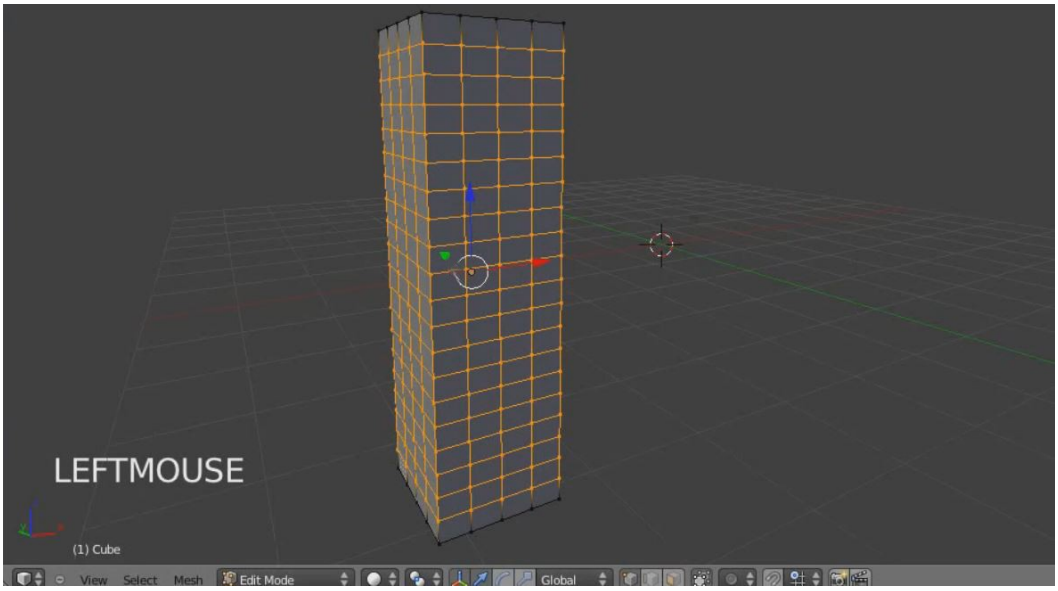
I *Limits* sono due campi numerici, nel range 0.0 e 1.0, che servono a delimitare il raggio d'azione del modificatore Simple Deform lungo l'oggetto; in pratica, l'altezza dell'oggetto viene messa in proporzione con il range 0.0 – 1.0, per cui se come Limite minimo si imposta ad esempio 0.2, il Modificatore avrà effetto a partire da un quinto (0.2, quindi 20%) della quota dell'oggetto, fino al punto stabilito dal Limite Massimo.

Di default, questi limiti hanno valore 0.0, cioè la base, e 1.0, ossia 100%, altezza massima, quindi il modificatore agisce su tutto l'oggetto, ed ecco perché non ci rendiamo conto della loro presenza.

Quanto detto risulta più chiaro con un esempio pratico, quindi passiamo, appunto, alla pratica.

Nella scena visibile nell'immagine nella pagina seguente è presente il cubo di default; scaliamolo lungo l'asse Z con S Z a piacere, in modo da ottenere un parallelepipedo.

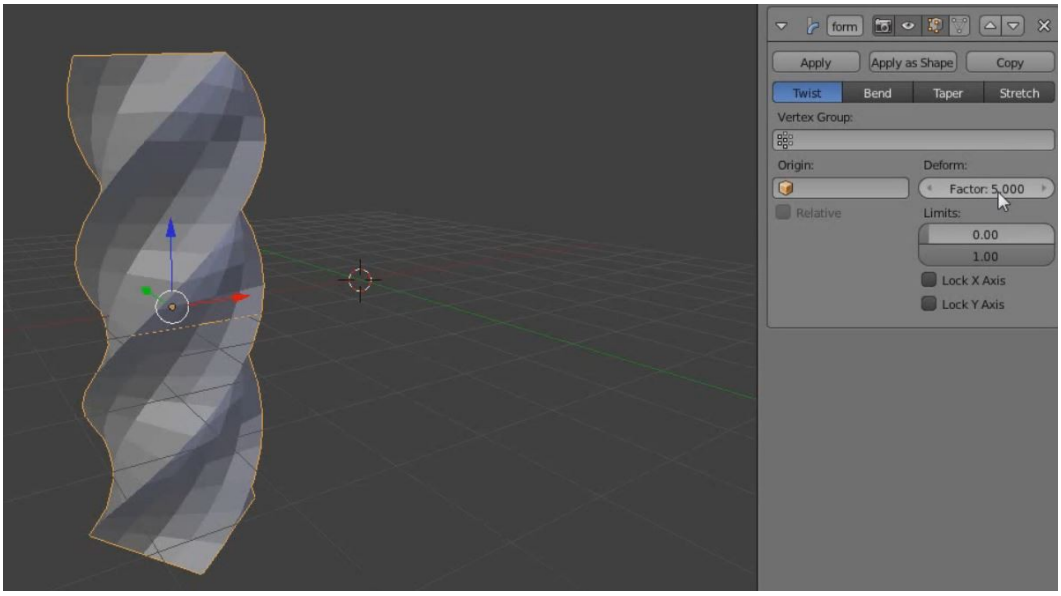
Passiamo in Edit Mode e con Ring Cut, con shortcut CTRL R, aggiungiamo (con tre diverse applicazioni, una per asse) delle suddivisioni, in modo da aumentare la risoluzione dell'oggetto: ciò è fondamentale per l'applicazione di un Modifier Simple Deform, come ad esempio Twist, che richiede un bel po' di vertici per produrre un buon risultato.




*Il cubo utilizzato per i nostri esempi, suddiviso con Ring Cut*

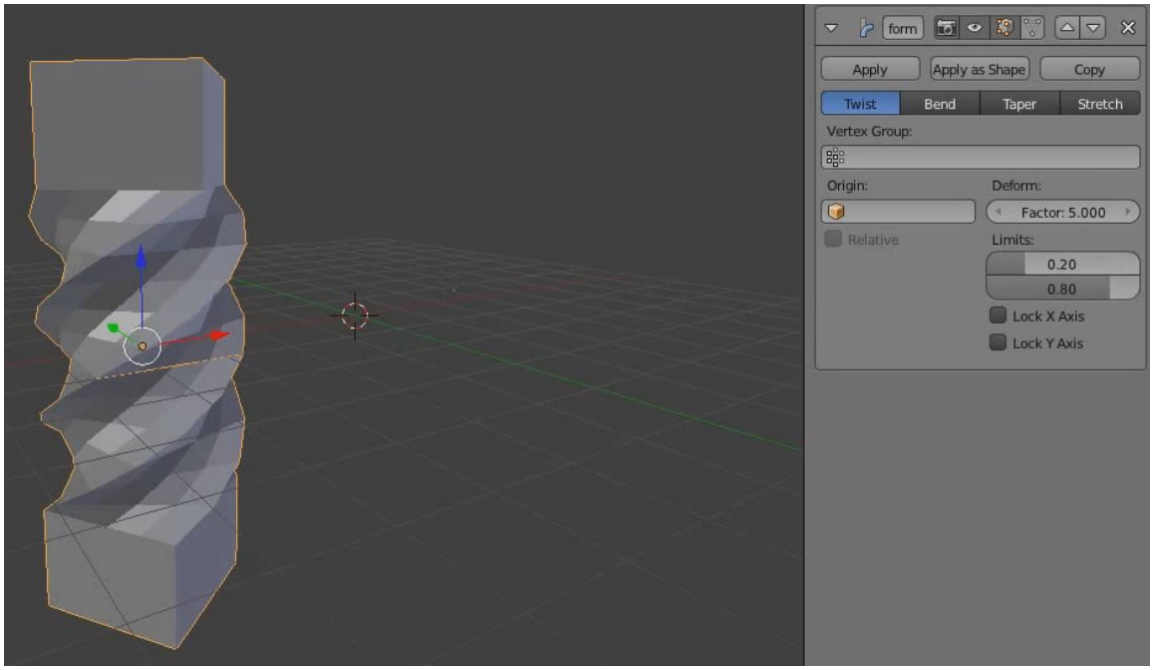
Torniamo in Object Mode.

Aggiungiamo ora un Modificatore Simple Deform, in modalità *Twist*, all'oggetto; il valore di default di Factor realizzerà già una piccola torsione, ma noi rendiamo quest'ultima più evidente portando Factor, ad esempio, a 5.




*Prima applicazione di Simple Deform - Twist*

La torsione riguarda tutto l'oggetto, ed è qui che entrano in gioco i *Limits*: per limitare la torsione alla parte centrale, diciamo a tre quinti della mesh, impostiamo il primo campo *Limits*, nella scheda del modificatore, a 0.2, e il secondo a 0.8: fatto!



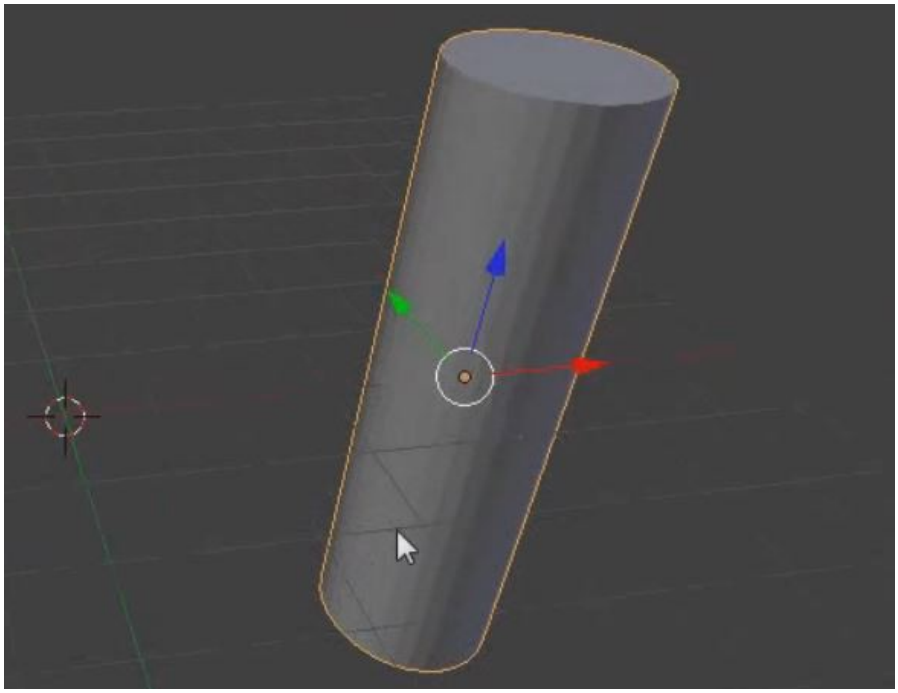

*Utilizzo dei Limits*

L'effetto è evidente e ci fa intuire come sia possibile realizzare oggetti che presentano pieghe o torsioni, ossia modifiche implementate dalle varie modalità di Simple Deform, solo in certe porzioni dell'oggetto originale, senza dover ricorrere a Vertex Groups o altre tecniche (anche se i Vertex Groups sono indispensabili se la modifica va applicata a più porzioni isolate di una mesh, non solo a una porzione delimitata dai due *Limits*).

Facciamo un altro esempio piegando un cilindro di 90° mediante Simple Deform – *Bend*, approfittandone per mostrare come ruotare il sistema di riferimento locale dell'oggetto per utilizzare bene il modificatore.

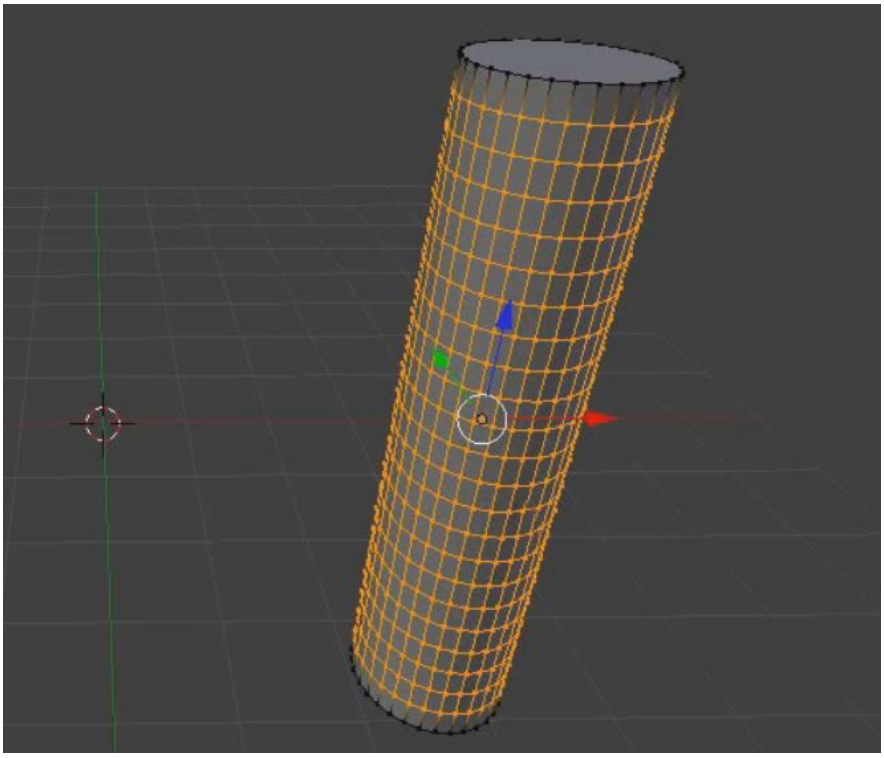
In Object Mode (per creare un oggetto distinto dal parallelepipedo di poco fa), aggiungiamo un cilindro nella scena.

Scaliamo questo cilindro verticalmente.



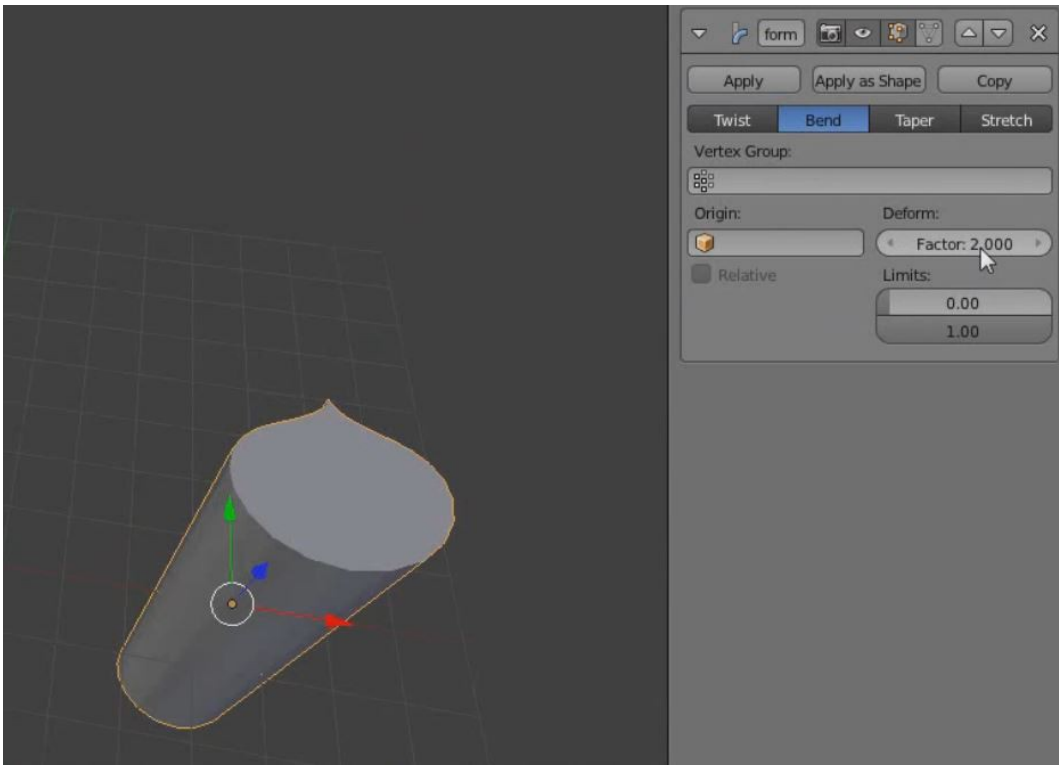
*La nuova scena, con il cilindro scalato verticalmente*

Il numero di default delle facce laterali andrà più che bene, mentre non ci siamo con il numero di tagli orizzontali, quindi passiamo in Edit Mode, aumentiamo la risoluzione della mesh con un buon numero di tagli con Ring Cut, quindi torniamo in Object Mode.



*Inserimento dei "tagli" nel cilindro*

Aggiungiamo un modificatore Simple Deform, questa volta in modalità *Bend* (cioè curva, piega) al Cilindro, impostando il Factor ad esempio a 2.0.

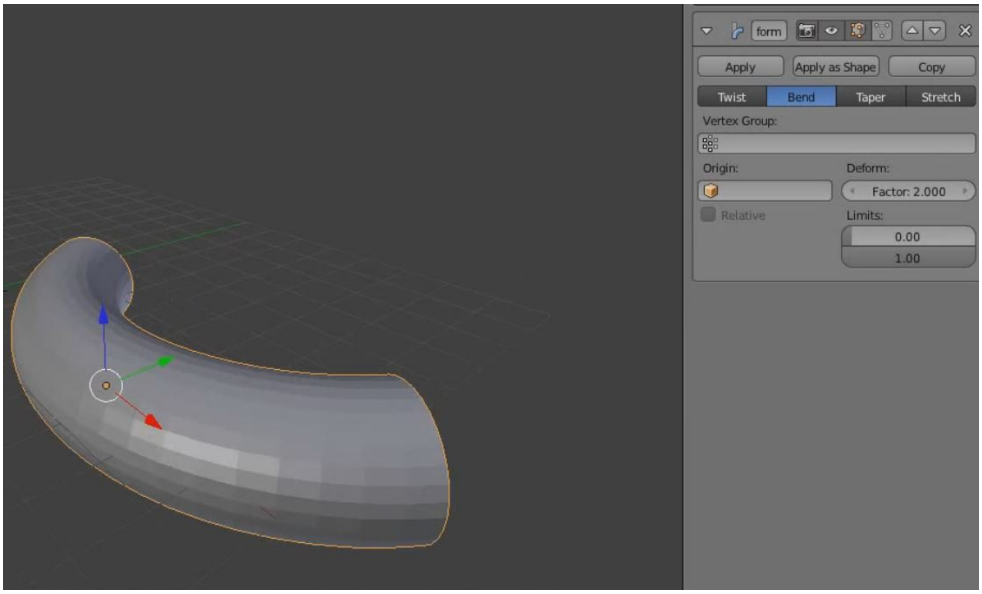



*Primo utilizzo (con problema) di BEND*

C'è un problema con il sistema di riferimento locale del cilindro, sistema al quale fa riferimento il modificatore; in particolare, la piega avviene intorno all'asse Z locale (che, almeno all'inizio, come in questo caso, coincide con l'asse Z globale della scena), ma noi vogliamo una curvatura intorno all'asse X globale della scena.

Per risolvere questo problema, ruotiamo l'oggetto di 90° intorno all'asse Y globale, con R Y 90 e Invio per confermare.

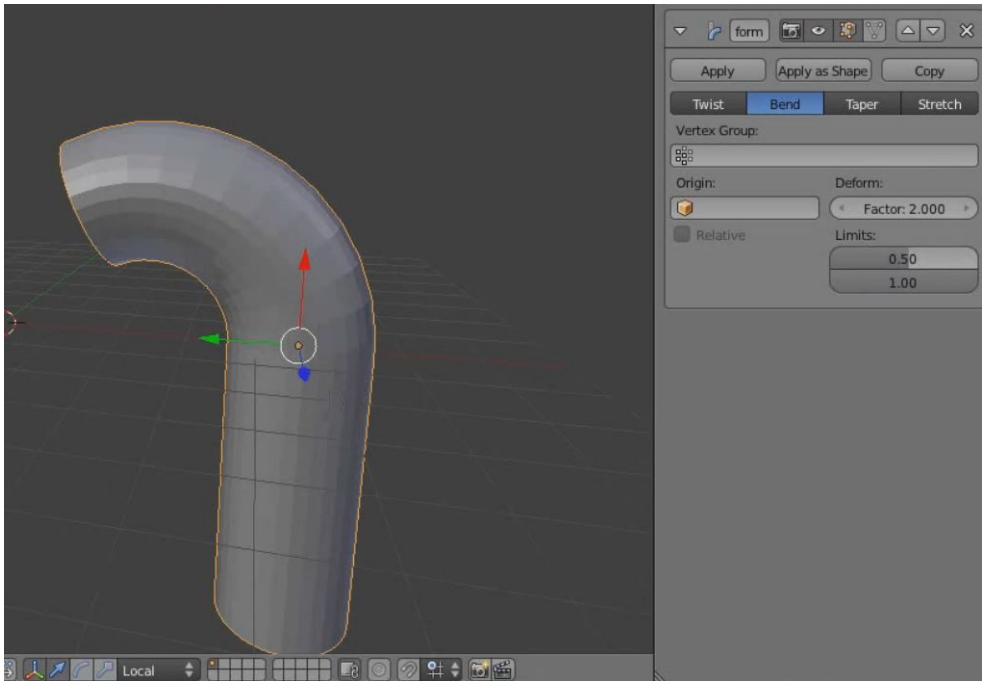
A questo punto, aggiorniamo il sistema di riferimento dell'oggetto con CTRL A e scegliendo Rotation dal menù che apparirà a video.




*Effetto dell'applicazione della rotazione, con il cilindro ruotato*

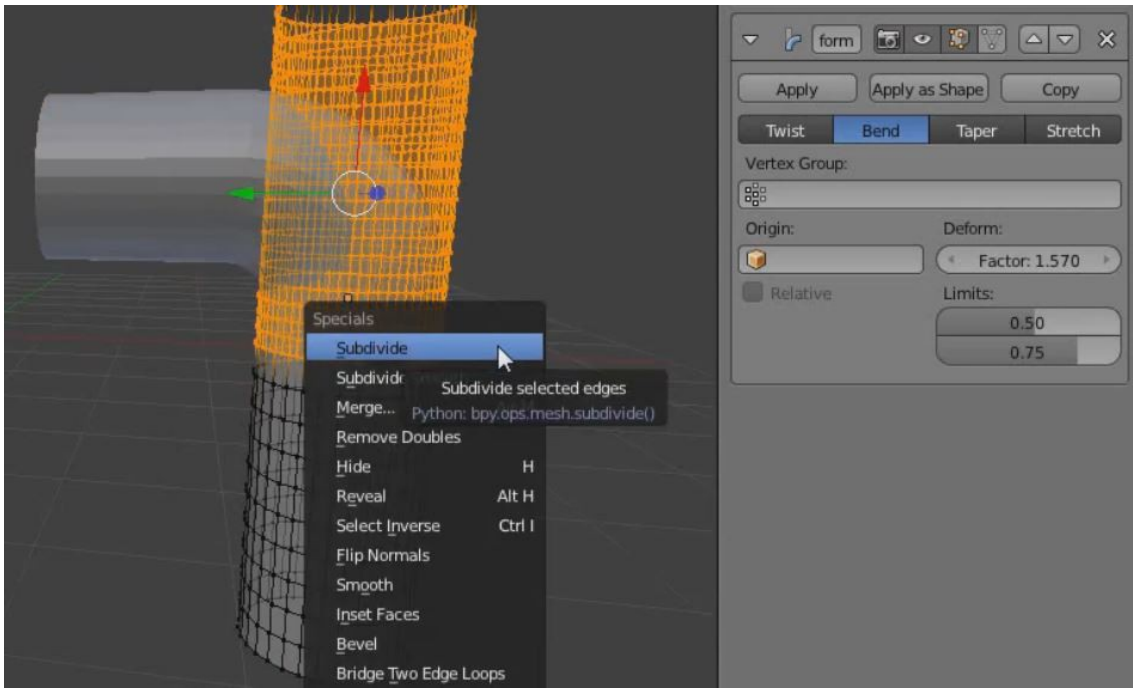
Il modificatore è ora orientato correttamente, quindi riportiamo l'oggetto all'orientamento iniziale (nel sistema di riferimento globale) con R Y -90 e Invio.

Adesso possiamo aumentare o diminuire a piacere il Factor e soprattutto, per limitare la curvatura ad una porzione dell'oggetto, impostare a piacere i *Limits*; ad esempio, 0.5 per quello inferiore (per dargli una base dritta), 0.75 per quello superiore e Factor 1.57.




*Nuovo utilizzo di BEND, con i Limits*

Qualora la mesh dovesse risultare troppo spigolosa nei punti di curvatura, per via di una scarsa risoluzione in termini di vertici, possiamo sempre passare in Edit Mode e aggiungere dei vertici in quei punti, questa volta anche con una selezione dell'area e uno o due colpi di Subdivide, come visibile nell'immagine seguente.




*Aumentare la "risoluzione" della mesh, inserendo dei tagli in Edit Mode prima di applicare il modificatore*

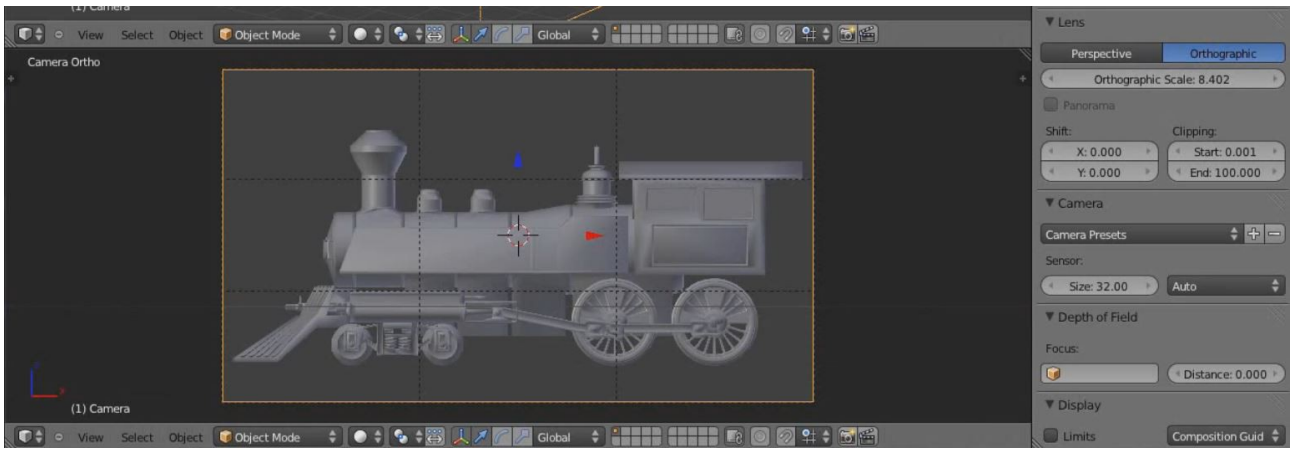
*  *  *

# Renderizzare sezioni di oggetti o scene

In questo brevissimo tutorial vedremo un trucchetto per ottenere anteprime o rendering di sezioni di oggetti o di intere scene in Blender 3D.

Capita infatti di dover renderizzare una sezione di un oggetto singolo o complesso (da un soprammobile a un intero edificio) e di scervellarsi su come fare per ottenere tale risultato; in realtà, l'operazione da fare è piuttosto semplice...

… prima, però, una considerazione: tipicamente la modalità di visualizzazione per questo tipo di immagini è ORTHOGRAPHIC; per cambiare la modalità di rendering della telecamera da PERSP (la modalità di default) a ORTHO, è necessario scegliere questa opzione nel pannello Object Data dell'oggetto telecamera, all'interno della Properties Window.

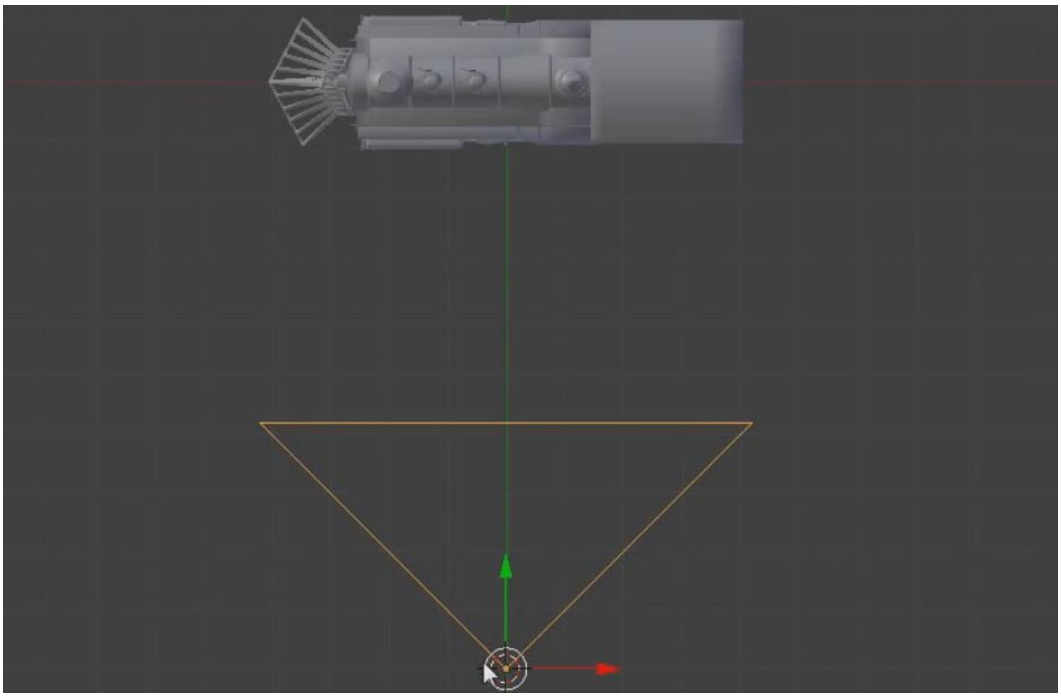


*Passaggio da PERSP a ORTHO per la telecamera*

In questa modalità, per zoomare bisogna agire sul valore Orthographic Scale, apparso ora sotto i pulsanti Persp/Ortho nella scheda.

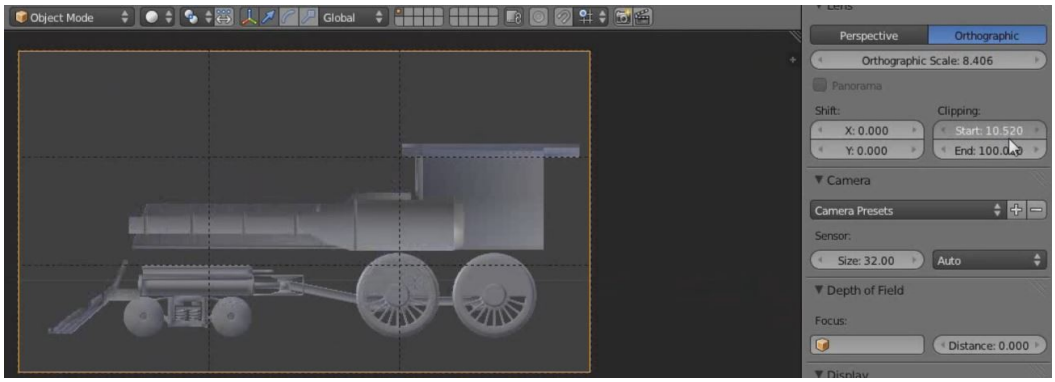
Posizionata la telecamera, possiamo realizzare la sezione: sempre nel pannello Object Data della telecamera, impostiamo un valore di *Start Clipping* pari alla distanza tra la telecamera e il piano di taglio della sezione.

Mi spiego meglio con un esempio (utilizzando la scena visibile nell'immagine seguente): nella vista Top Ortho vediamo la posizione della telecamera e quella dell'oggetto da inquadrare.



*La scena utilizzata per questo esempio*

Supponiamo di voler renderizzare una sezione dell'oggetto esattamente a metà dello stesso: se il centro si trova, come in questo caso, a 10 unità di distanza dalla telecamera, allora dovremo scrivere proprio 10 come campo in *Start Clipping*.




*Utilizzo di Start Clipping*

La sezione è visibile già nella 3D View e lo sarà anche una volta avviato il rendering con F12; il taglio viene applicato a tutti gli elementi presenti nella scena a quella distanza dalla telecamera perché in realtà non è una sezione degli oggetti, ma proprio un piano di taglio (*clipping plane*) che renderizza i vertici, gli spigoli e le facce degli elementi che si trovano a distanza maggiore di *Start Clipping* dalla telecamera (e, ovviamente, entro la distanza data dal campo *End Clipping*, ed ecco perché gli oggetti molto distanti a volte non vengono renderizzati).

La distanza dalla telecamera è calcolata dall'Origin della stessa, ossia dalla punta che la rappresenta, non dalla base dell'icona, per cui la distanza *Start Clipping* non cambia se si varia il valore di Orthographic Scale.

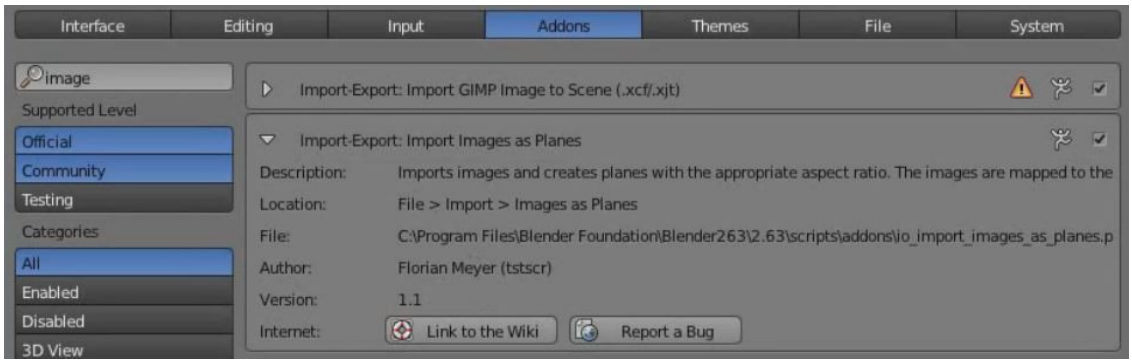
*  *  *

# Modellare un Gulfstream G650

In questo tutorial vedremo come modellare un jet, in particolare un Gulfstream G650, in Blender 3D 2.63; la versione è importante perché si farà uso della nuova modellazione Bmesh, presente appunto a partire da Blender 2.63.

Iniziando da una scena completamente vuota, la prima cosa da fare è caricare le Blueprints, le immagini di riferimento delle viste frontale, superiore e laterale.

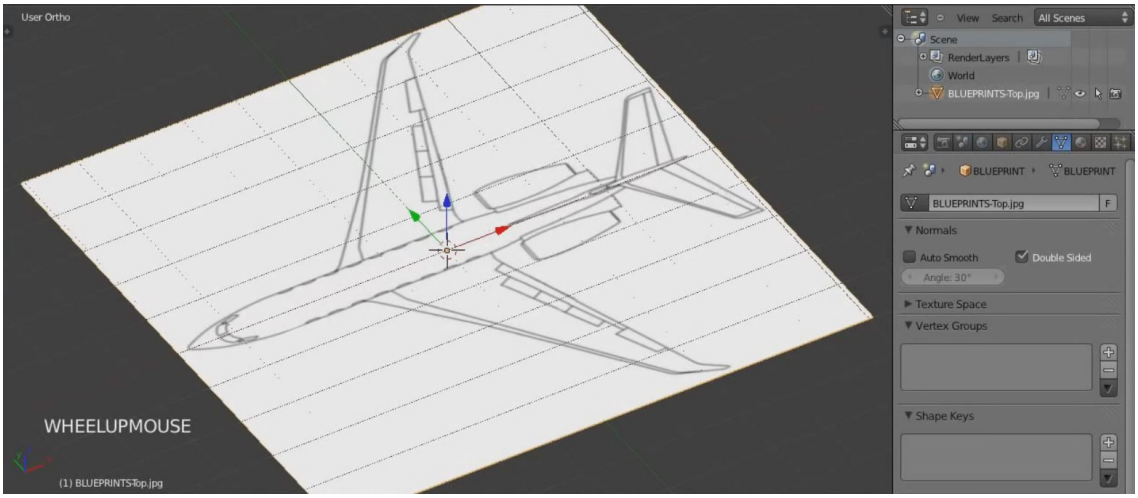
Anziché caricarle come Background Images nelle viste Ortho, le sto caricando come Planes mediante l'Add-On "*Import Images as Planes*", dal menù Import.




*Caricamento delle Blueprints utilizzando "Import Images as Planes": attivare l'Add-On*

Qualora lo script non dovesse essere attivo, cercatelo nel pannello Add-Ons della finestra User Preferences (menù File) e attivatelo lì; qualora non dovesse esserci proprio, cercatelo online, scaricatelo, installatelo e attivatelo.

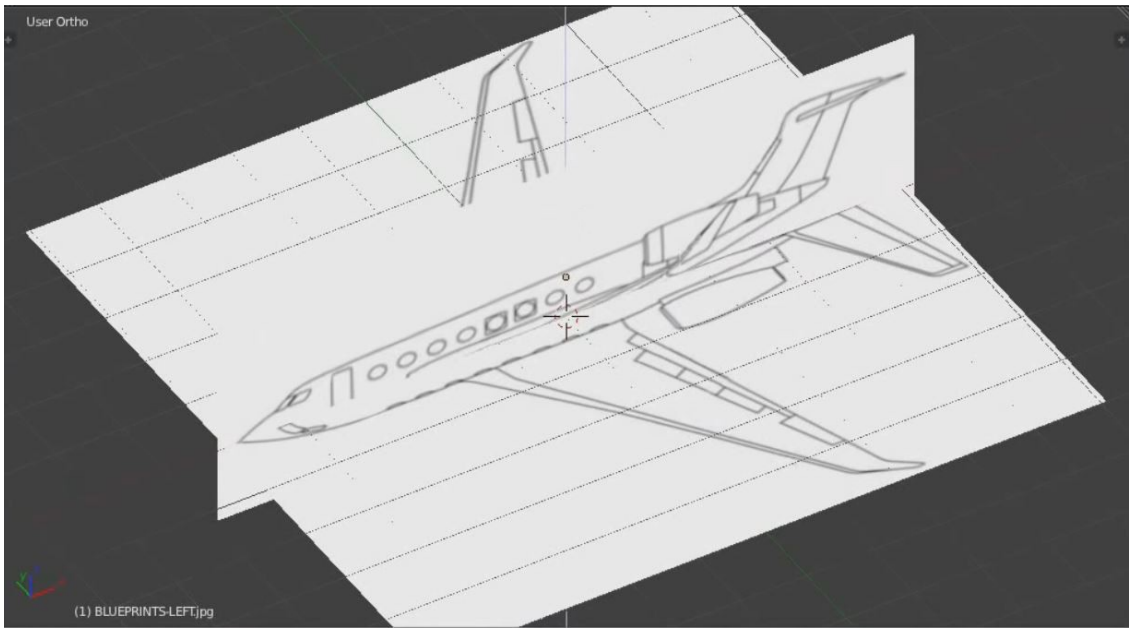
Per poter utilizzare questo script è necessario passare all'Engine Blender Internal, perché ancora Cycles non lo supporta;poco male: passiamo a Blender Internal e scegliamo, nella 3D View, la modalità di visualizzazione *Textured*, per vedere il Plane con l'immagine applicata.




*La prima immagine di riferimento inserita nella scena*

All'inizio ho caricato solo due immagini di riferimento, in altrettanti Planes, quindi ho iniziato a modellare; caricherò la terza immagine in seguito.

Qualora le immagini non dovessero avere, come in questo caso, la stessa scala, farle combaciare può richiedere un po' di tempo; bisogna scalare i Planes e spostarli, cercando di individuare dei punti di riferimento nelle immagini per effettuare questa operazione... comunque se non corrispondono al 100%, non vi preoccupate.

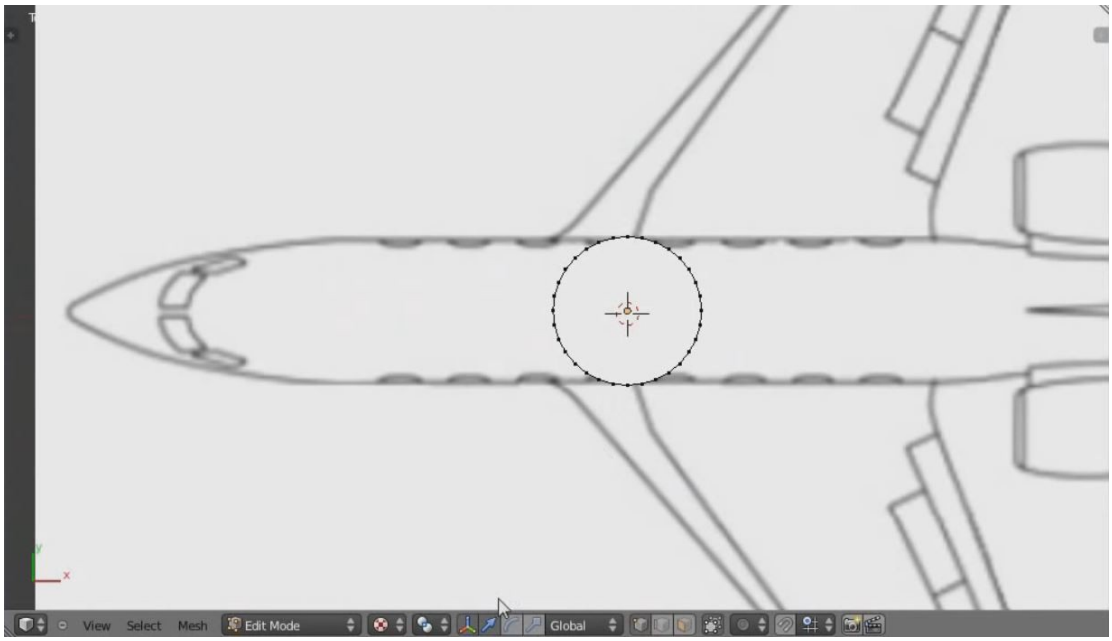



*Far corrispondere le Blueprints*

A questo punto, iniziamo a modellare.

Anziché disporre un cilindro sul lato, in modo da estrudere le basi per modellare la fusoliera dell'aereo, l'ho disposto verticalmente. Il motivo è che la disposizione sul lato portava a delle distorsioni nei segmenti in prossimità del muso, e siccome proprio lì ci vanno delle strisce colorate (che volevo fare ricorrendo ai materiali multipli, senza textures) ed altri elementi, quelle distorsioni erano inaccettabili.

Ho quindi deciso di usare un cilindro verticale e, come potete vedere nell'immagine seguente, spostare i segmenti alle estremità, lasciando dritta la parte centrale e spostando poi la punta e le coppie di segmenti per seguire il profilo dell'aereo.

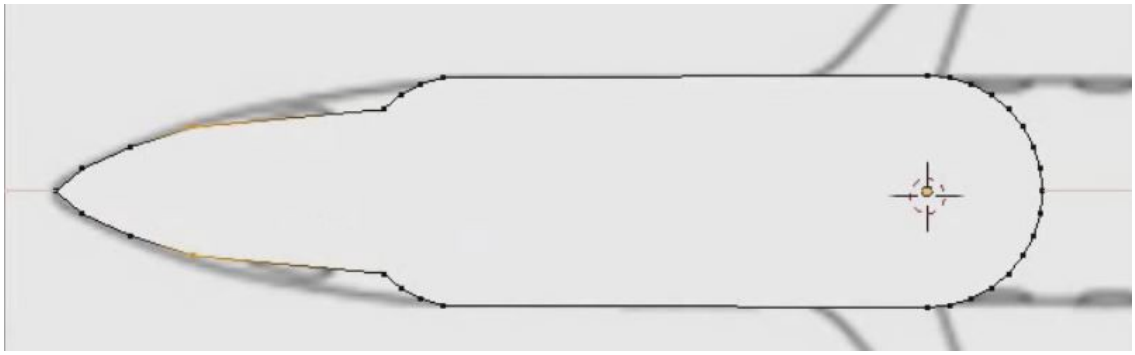



*Utilizzo di un cilindro verticale*

Assicuratevi di avere *Limit Selection to Visible* DISATTIVATO, nella 3D View, perché in caso contrario, con la vista Textured, la selezione riguarderebbe solo i vertici visibili, quindi in questo caso solo la faccia superiore, mentre noi vogliamo selezionare e spostare anche i vertici della faccia inferiore.

*"Limit Selection to Visible" deve essere disattivato*

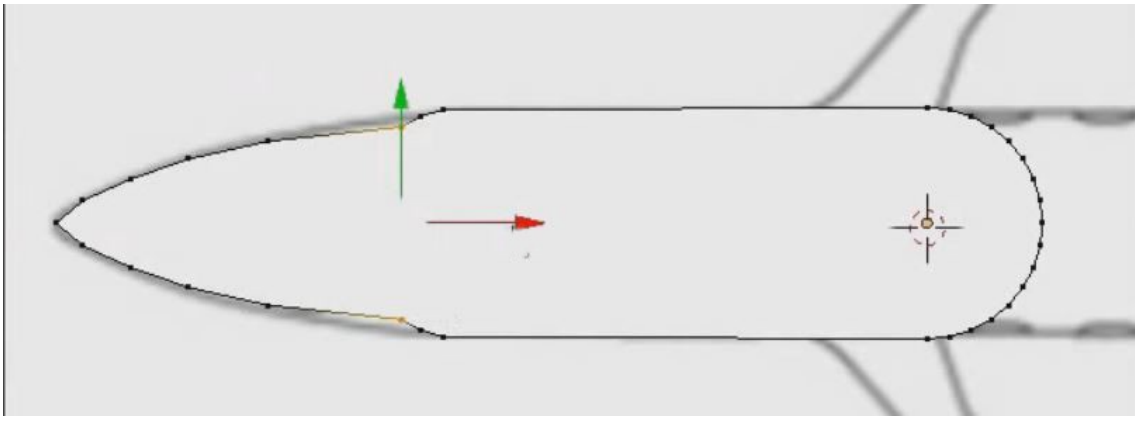
Ogni tanto scalerò coppie di vertici per allontanarle, senza però rendere più grande il profilo; si tratta, quindi di uno scaling non uniforme. Per farlo, osservate nella 3D View qual è l'asse lungo il quale volete scalare e digitate la lettera dopo aver premuto S; nel mio caso, tale asse è Y, quindi dopo aver selezionato le coppie di vertici (in realtà, 4 vertici, perché ci sono anche quelli di sotto) premo S Y e scalo, a volte anche tornando su coppie di vertici per cercare di rendere abbastanza levigato il profilo, sia per il muso che per la coda.



*Allontanare coppie di vertici mediante selezione e scaling*

Vi anticipo subito che non sono un grande fan dello strumento Mirror, almeno quando la modellazione è semplice e con pochi vertici, come in questo caso, per cui (anche se può sembrare scomodo, ma vi assicuro che non lo è) vedrete appunto la selezione di coppie di vertici e la loro trasformazione qui, durante la modellazione delle ali, della coda e di altre parti dell'aereo.

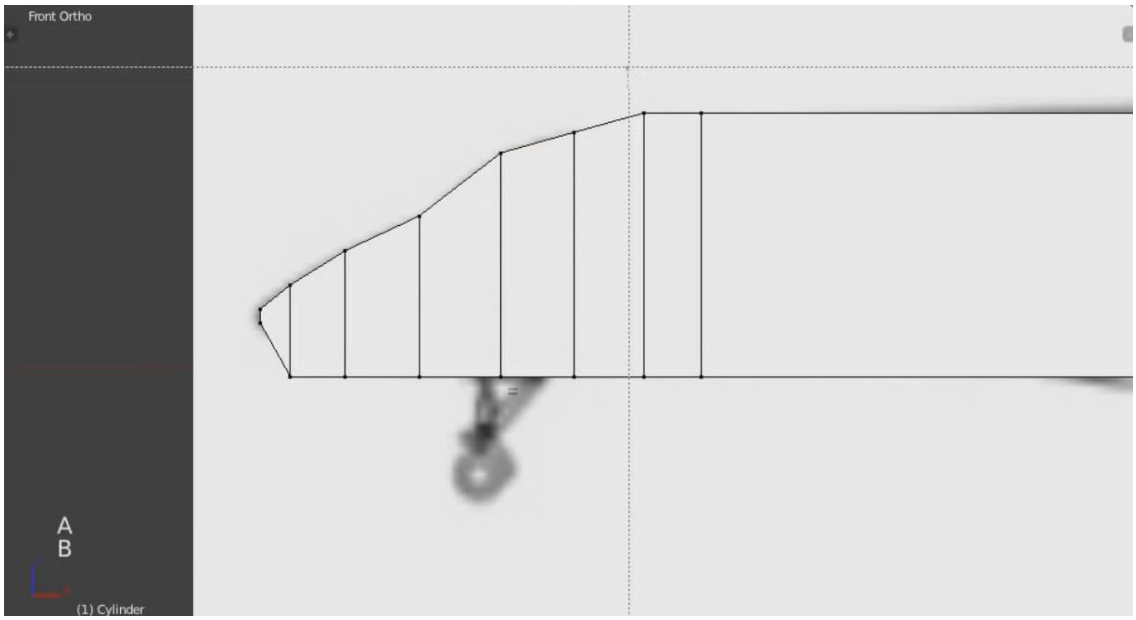
Questa parte, insieme al posizionamento dei piani delle Blueprints, è forse quella che richiede più tempo, ci vuole un po' di pazienza...



*Posizionamento iniziale dei vertici, mediante spostamenti e scaling*

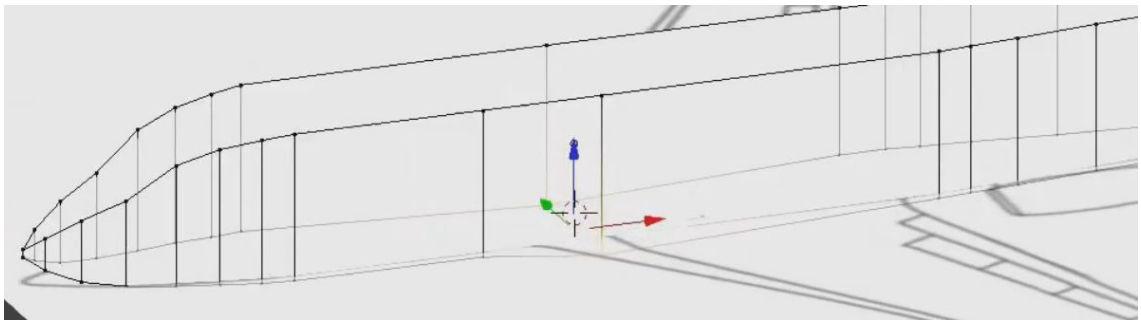
L'aver scelto di modellare il cilindro verticalmente si rivela utile quando si passa alla vista laterale, quando arriva il momento di modellare la parte del muso dell'aereo. Avremo qualche problemino in seguito con la vista frontale, per rendere più circolare il profilo, ma lo risolveremo facilmente.

Segue un'altra fase, alquanto noiosa, di posizionamento dei vertici; attenzione sempre a Limit Selection, che deve essere disattivato.




*Posizionamento dei vertici del muso dell'aereo*

A questo punto, al momento di introdurre dei tagli con Ring Cut (CTRL R) per spostare certe parti della fusoliera, mi sono accorto che la creazione delle facce, fatta in automatico da Blender con la nuova modalità Bmesh, dava qualche problema, infatti c'erano parti centrali nelle quali "spuntavano" delle facce. Si vede, ad esempio, che il taglio centrale con Ring Cut ha creato dei segmenti ai lati e sotto, ma non nella faccia superiore.

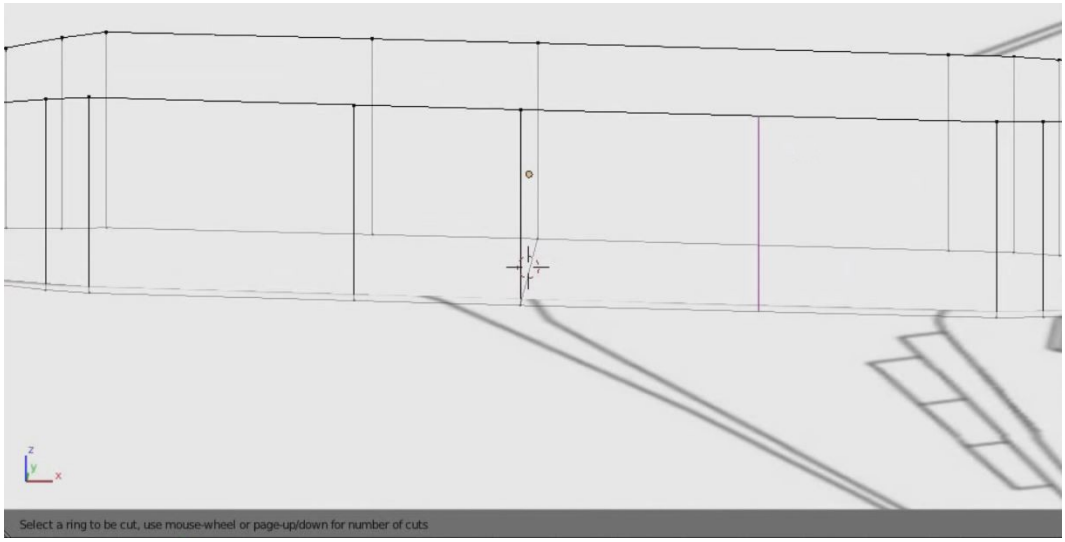


*Alcuni problemi dovuti a BMESH*

Lì per lì ho provato aggiungendo uno spigolo di collegamento selezionando due vertici e premendo *F,* ma questo non risolve il problema della presenza di un'unica faccia per la base superiore; il problema è proprio questa superficie (e, in maniera speculare, la base inferiore).

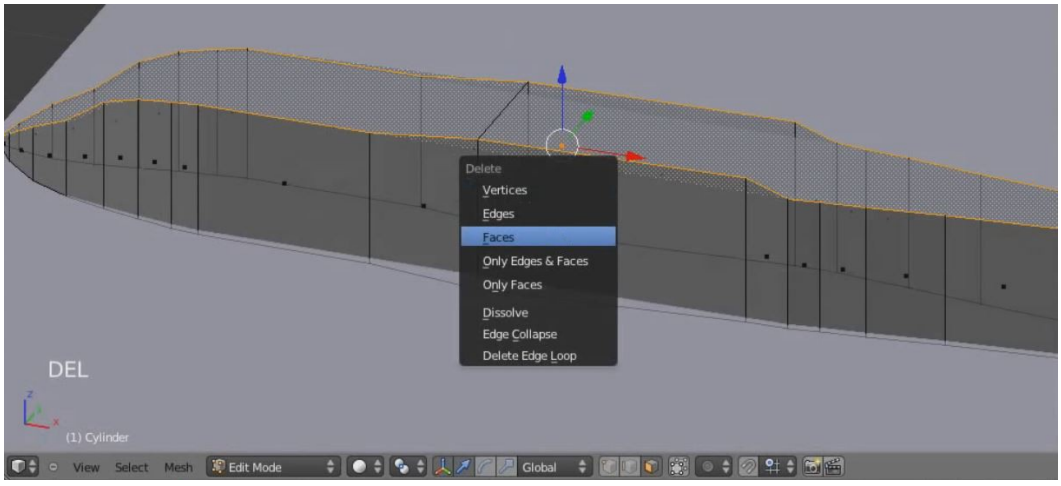
Ho risolto il problema con una soluzione forse poco ortodossa.

Per prima cosa, ho fatto i tagli che mi servivano, in due tempi (prima un lato della fusoliera, poi l'altro), spostando il taglio fatto con CTRL R per allinearlo a quello dall'altro lato (infatti dopo aver cliccato per confermare il taglio, è possibile spostarlo, per cui dopo il primo click andavo in vista laterale – ortogonale, per evitare scherzi della prospettiva – e spostavo il taglio).



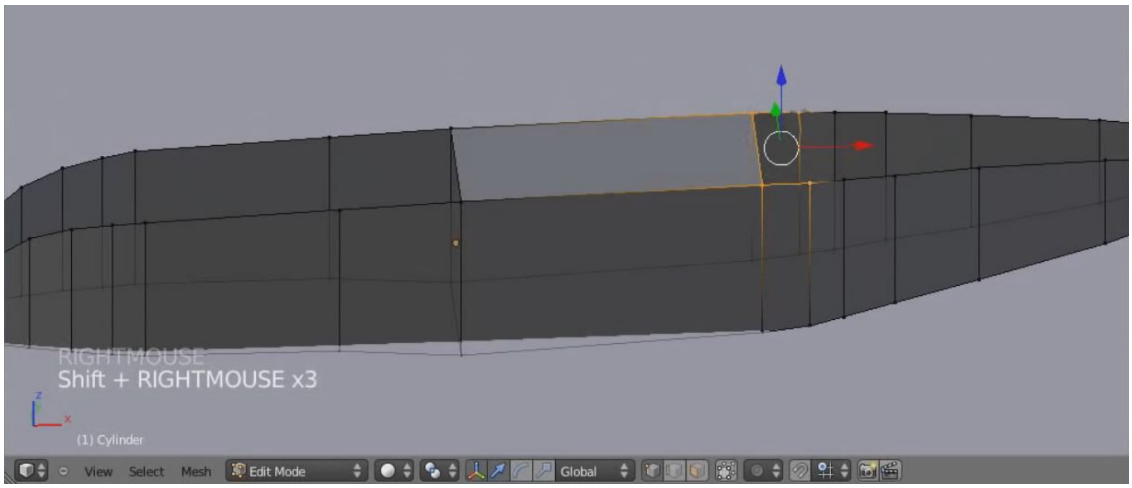

*Primo passo della soluzione: realizzare i tagli*

L'operazione successiva è stata la rimozione delle facce superiore e inferiore; per farla, si passa in modalità selezione facce, si selezionano le facce da rimuovere, si preme X o CANC e si sceglie FACES, in modo da lasciare vertici e spigoli di contorno.

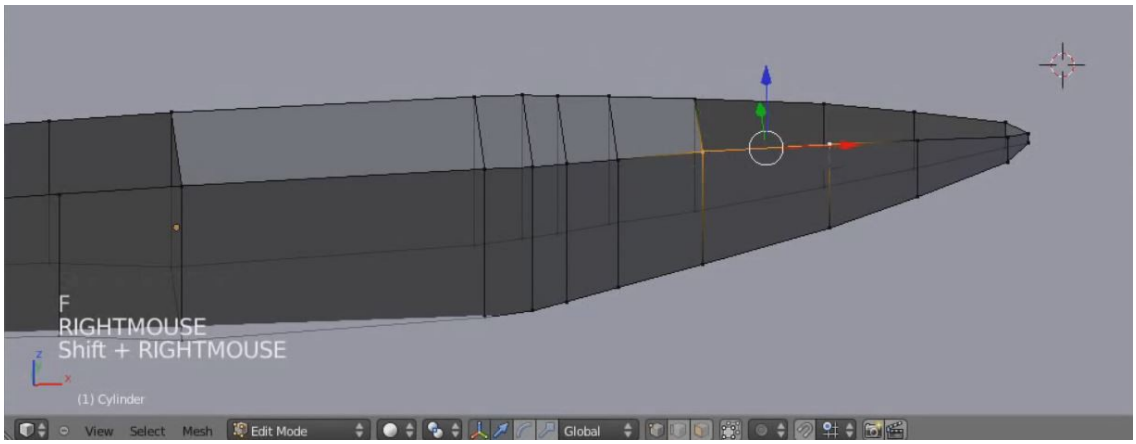



*Rimozione facce*

A questo punto, un lavoro noioso, ma da fare (per evitare il problema di prima): anziché un'unica faccia per ciascuna base, creare delle facce quadrangolari (e triangolari nelle punte), selezionando i vertici a 4 a 4 e premendo *F* (Fill); noioso, ma da fare, comunque i vertici sono pochi.



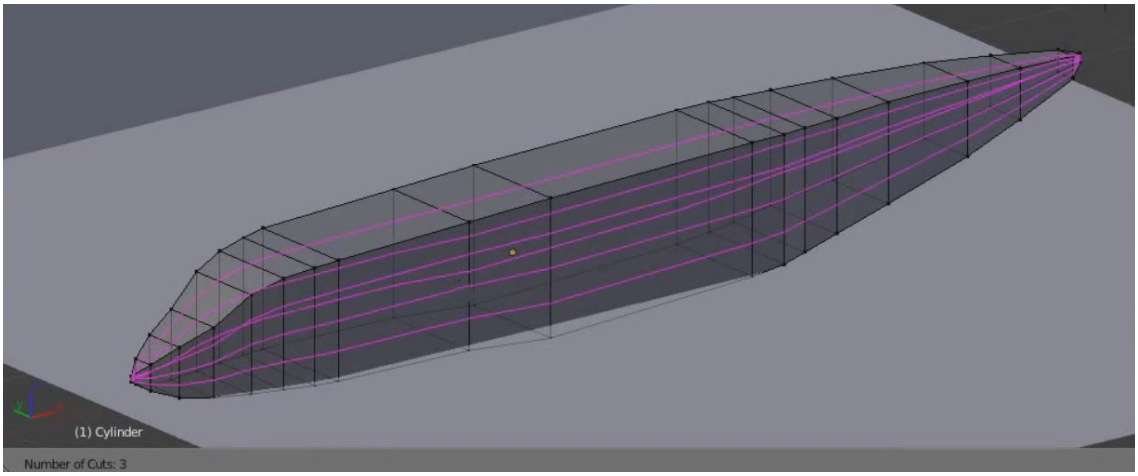

*Filling, I parte*




*Filling, II parte*

Ok, a questo punto bisogna lavorare sulla larghezza del modello.

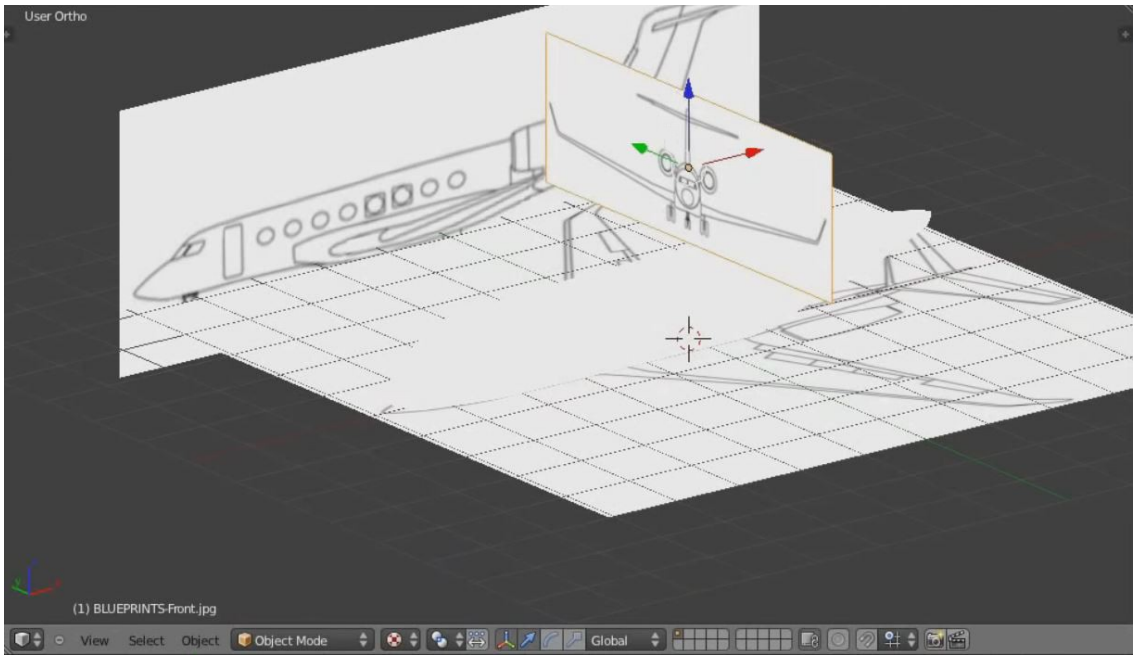
Con Ring Cut, sto aggiungendo tre tagli al modello, da scalare successivamente per dargli un profilo più circolare.




*Aggiungere tagli al modello*

È giunto il momento, tra l'altro, di caricare la terza immagine Blueprint, quella della vista frontale, sempre sotto forma di Plane mediante l'Add-On “Import Images as Planes”.

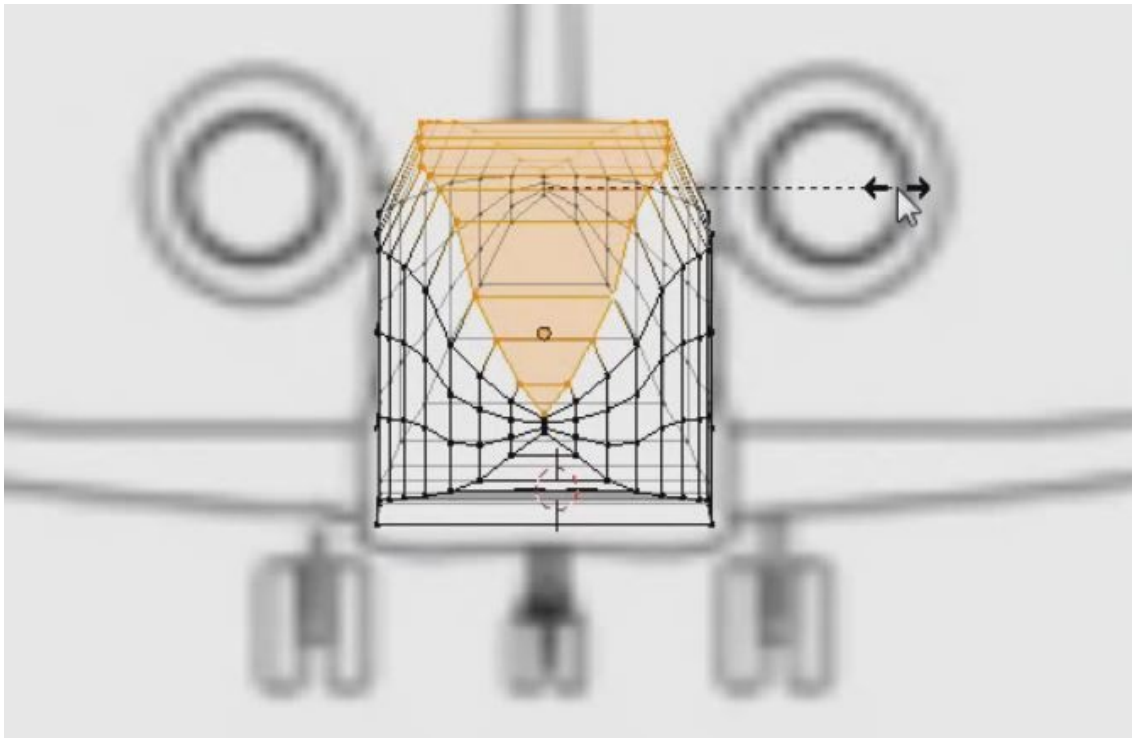
Ci vuole quindi un po' per posizionare questa Blueprint, da allineare sia all'immagine TOP sia, per quanto riguarda le quote, rispetto all'immagine Left; Tra l'altro, sia in questa fase che in seguito, per rendere il tutto più comprensibile nasconderò ogni tanto degli oggetti, cosa che va fatta selezionando gli elementi da nascondere e premendo H, per Hide; per farli riapparire, poi, bisognerà premere ALT H.




*Posizionamento della terza blueprint*

A questo punto ho iniziato a selezionare i vertici a file per scalarli orizzontalmente, quindi con uno scaling vincolato sull'asse, per avvicinarli o allontanarli.

Per selezionare una fila di vertici, fare ALT e click del tasto destro su uno spigolo della fila; per selezionarne due, fare SHIFT ALT e click destro sul secondo spigolo, dopo aver selezionato la prima fila.

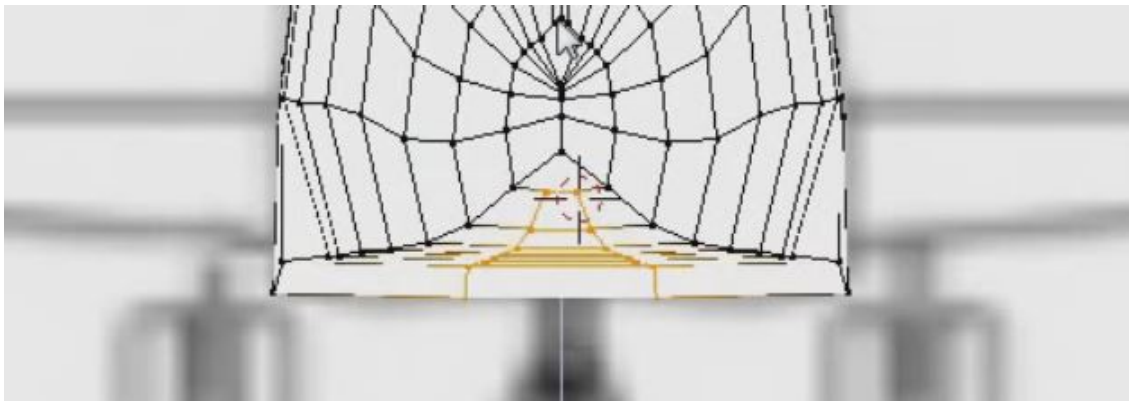


*Selezionare e scalare "file" di vertici*

Adesso, col senno di poi, sto pensando che forse avrei potuto inserire una suddivisione verticale e procedere con uno scaling mediante Editing proporzionale, un po' in orizzontale e un po' in verticale... potete provare eventualmente anche questa soluzione; quando ho registrato il video, ho inserito un paio di tagli verticalmente e ho scalato i tagli orizzontali lungo l'asse orizzontale, nel mio caso Y, e quelli verticali lungo l'asse Z. I tagli vanno inseriti sempre con CTRL R (Ring Cut).

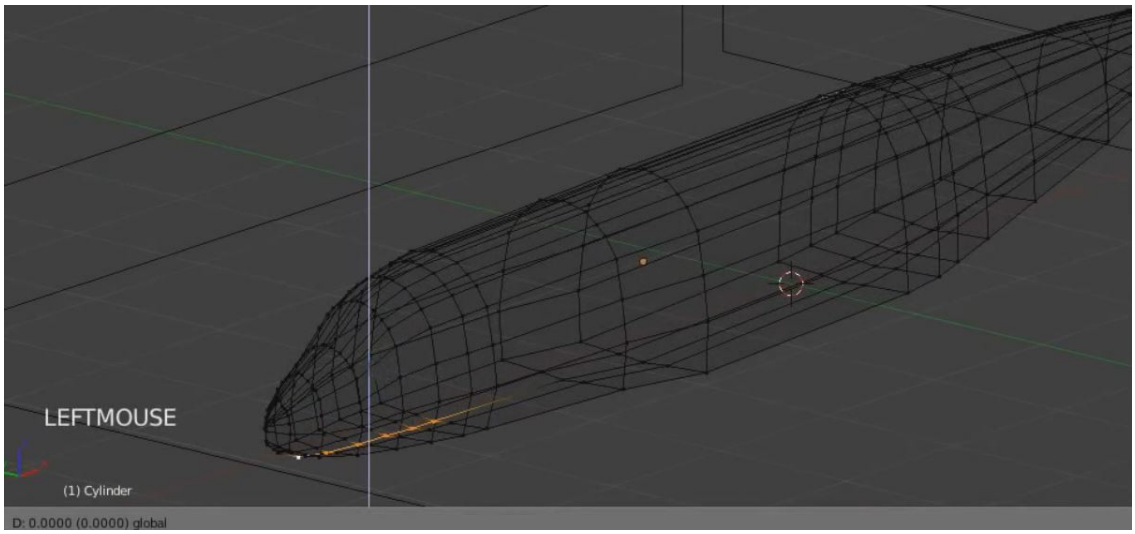
Per certi punti ho preferito un approccio manuale, selezionando gruppi di vertici da traslare o scalare manualmente, in questa fase più che altro “a occhio”, cercando di evitare parti troppo spigolose.

La parte centrale alla base non è un problema, può rimanere un po' squadrata, visto che lì andranno attaccate le ali.



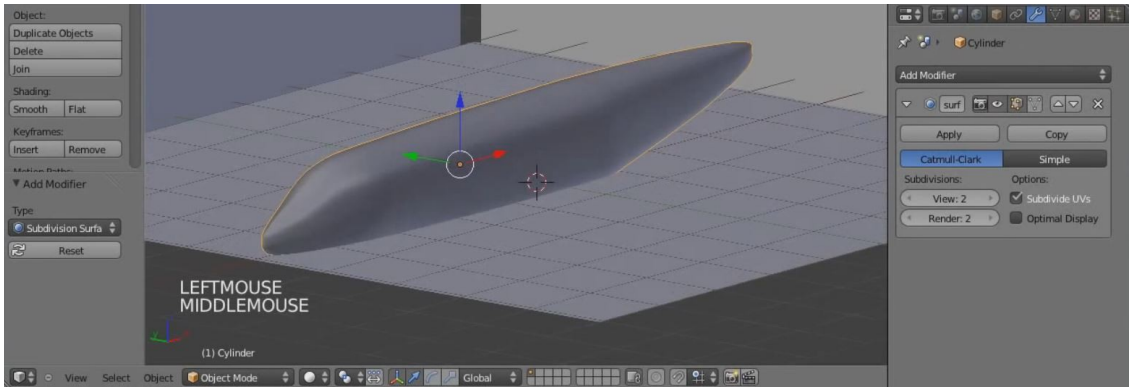
*La parte centrale alla base*

Qui non l'ho fatto, ma può essere utile anche selezionare i vertici della fila centrale di sotto, nella zona del muso, e ruotarli un po', piuttosto che traslarli semplicemente verso il basso, perché la sola traslazione provoca uno “scalino”, che ho cercato di correggere con il Proportional Editing.



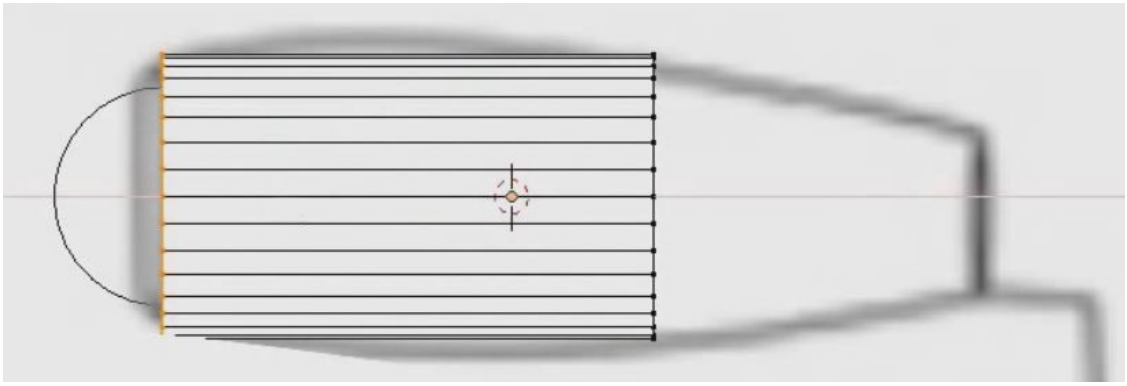

*Cercare di correggere lo "scalino"*

Forse tutti questi accorgimenti per la smussatura manuale, vertice per vertice, non erano nemmeno necessari, infatti poi applicando Shading *Smooth* nella Tool Shelf e, nel pannello Modifiers, *Subsurf* con 2 suddivisioni, il modello viene comunque levigato sia a livello di Shading che di geometria, risolvendo automaticamente certi punti sui quali ho perso un po' di tempo.




*Shading Smooth e Subdivision Surface*

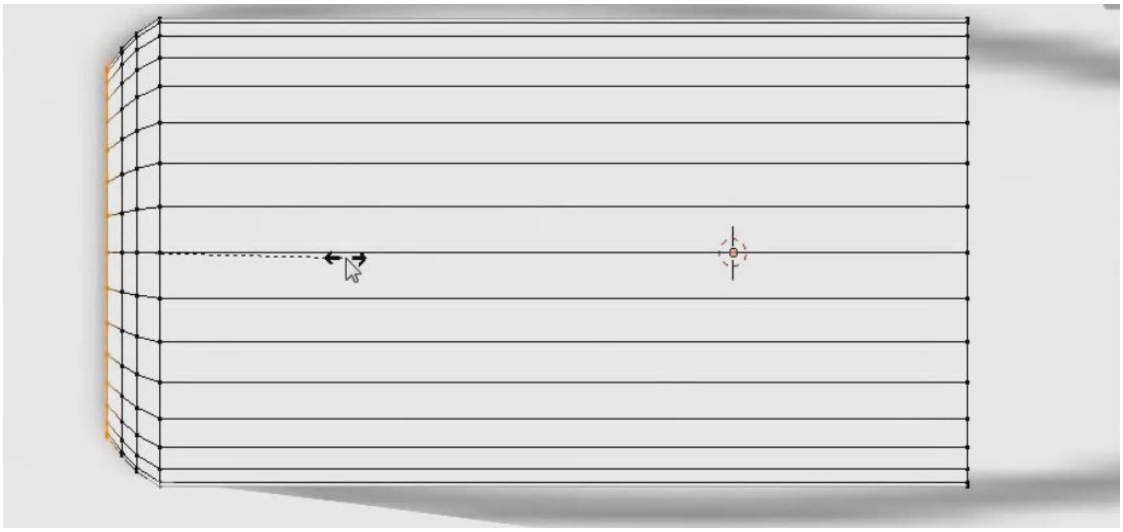
Ok, la fusoliera è pronta: ora possiamo nascondere questa mesh con H e passare alle turbine. In questo caso, ho aggiunto un cilindro e, questa volta, l'ho ruotato sul lato.



*Il cilindro per le turbine*

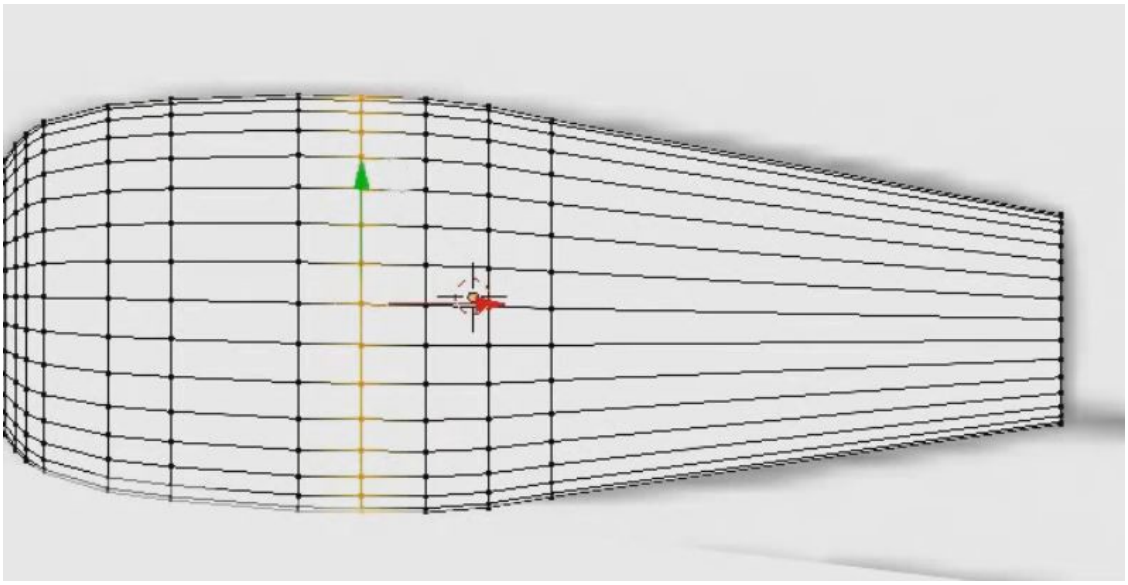
Nel video si vede che ho fatto, inizialmente, due errori: il primo è il non aver disattivato il Proportional Editing, per cui estrudevo e spostavo ma non vedevo la nuova selezione... non la vedevo perché, col Proportional Editing attivo, mi portavo dietro anche la selezione precedente, durante lo spostamento. Ricordate, quindi, di disattivare il Proportional Editing (shortcut: O), dopo averlo utilizzato, per evitare brutte sorprese!

Il secondo errore riguarda il famoso *Limit Selection to Visible*: in questa vista era attivato di default, non l'ho disattivato e me ne sono accorto quando, cambiando punto di vista, ho notato che i vertici inferiori erano rimasti alla posizione di partenza... ecco perché, nel video, ad un certo punto ho cancellato il cilindro e l'ho rifatto da capo. Ci vuole quindi un po' di attenzione anche nell'utilizzo di questo strumento.



*Modellazione della turbina - 1*

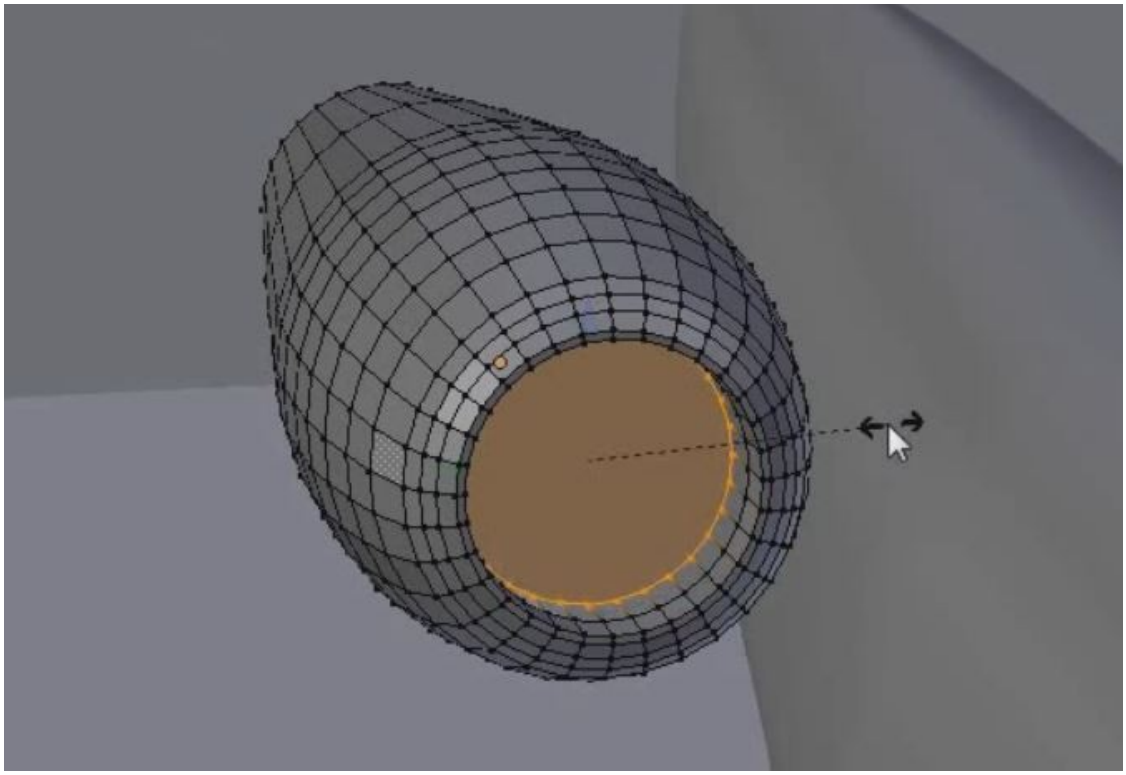
A questo punto si procede di estrusione, traslazione e scaling per modellare la turbina... turbina che non è dritta, perché nella parte posteriore si avvicina all'aereo.



*Modellazione della turbina - 2*

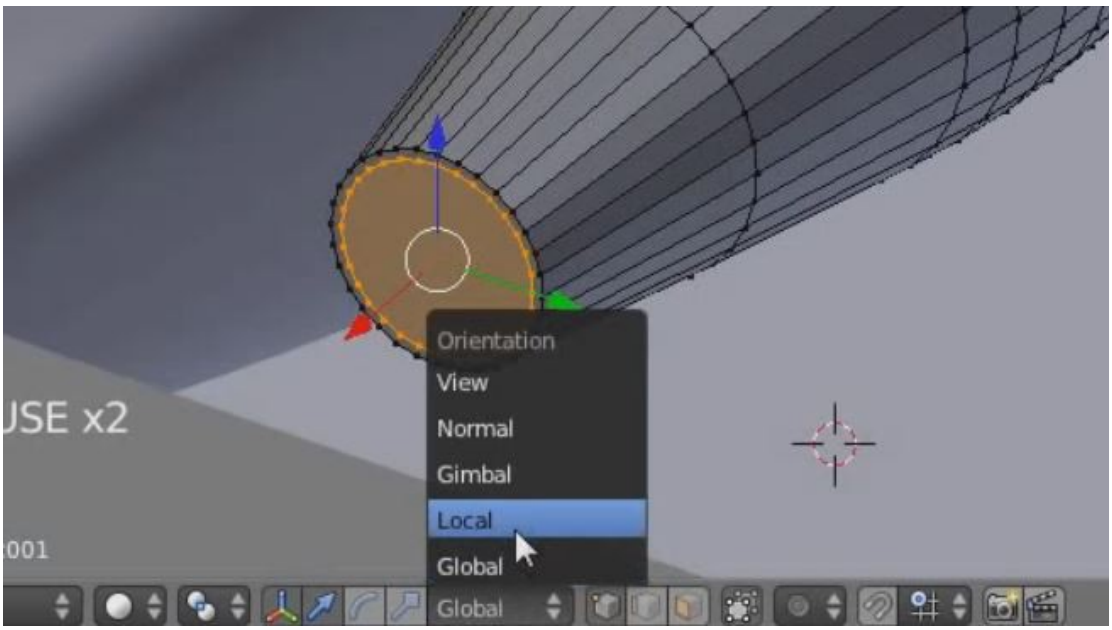
Per realizzare le aperture, selezionare la faccia (in pratica, la base del cilindro) e fare estrusione e scaling sul posto, eventualmente con traslazione.

Estrusione e scaling (a rimpicciolire) sul posto introducono un cerchio all'interno di quello precedente; lo spostamento verso l'interno della turbina di questo nuovo cerchio crea la profondità.



*Realizzare le "aperture" nella turbina (nelle sue basi)*

Il lavoro va fatto per entrambe le basi; nel caso del retro, si vede (immagine seguente) che ho cambiato la modalità delle maniglie in Local, il sistema di riferimento locale della selezione, in modo da spingere gli elementi nella giusta direzione all'interno della turbina.

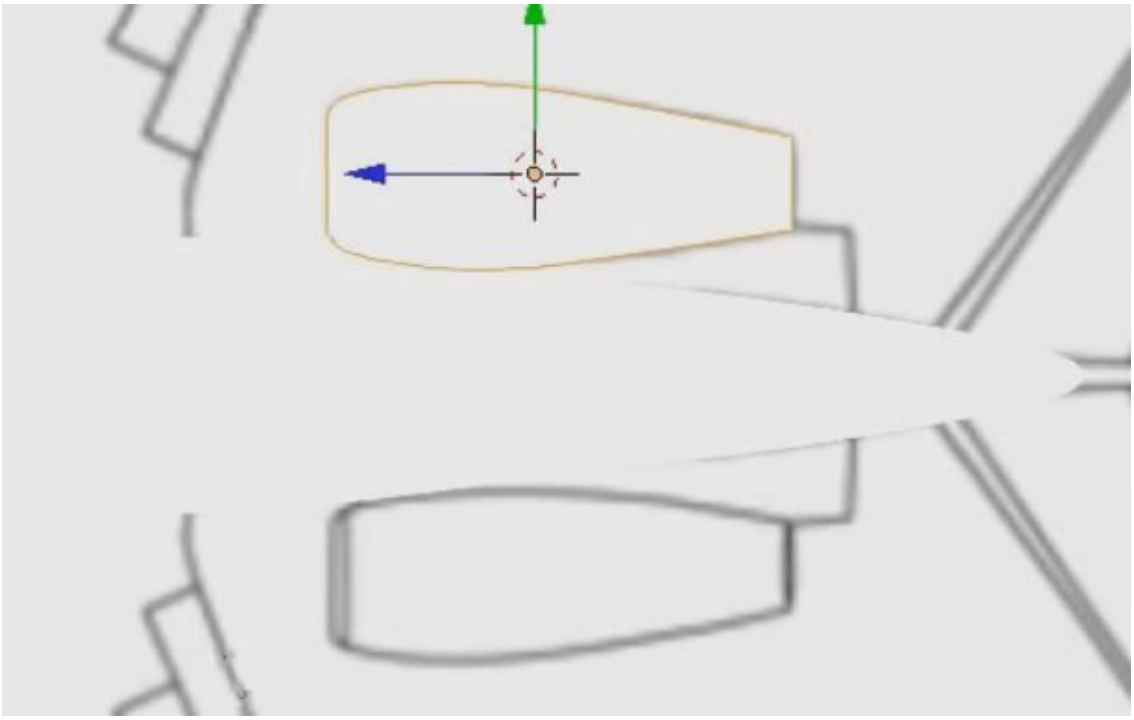



*Passaggio al sistema Local per le basi del cilindro / turbina*

Questo problema si ha in generale quando si ruota un oggetto o una sua parte: se si continuano a fare le trasformazioni rispetto agli assi globali, che non coincidono più con quelli locali dell'oggetto, i risultati appariranno "storti" rispetto al resto del modello; per risolvere il problema, passare alla modalità Local o Normal.

In questa fase, tra l'altro, ho riattivato *Limit Selection to Visible*, perché c'era troppa confusione per selezionare la base (con ALT e click destro su uno degli spigoli della circonferenza), per cui in questo caso *Limit Selection* torna utile, mostrando solo i vertici direttamente visibili; il problema, poi, sta nel non dimenticarsi che è attivo, come è successo prima proprio creando la turbina.

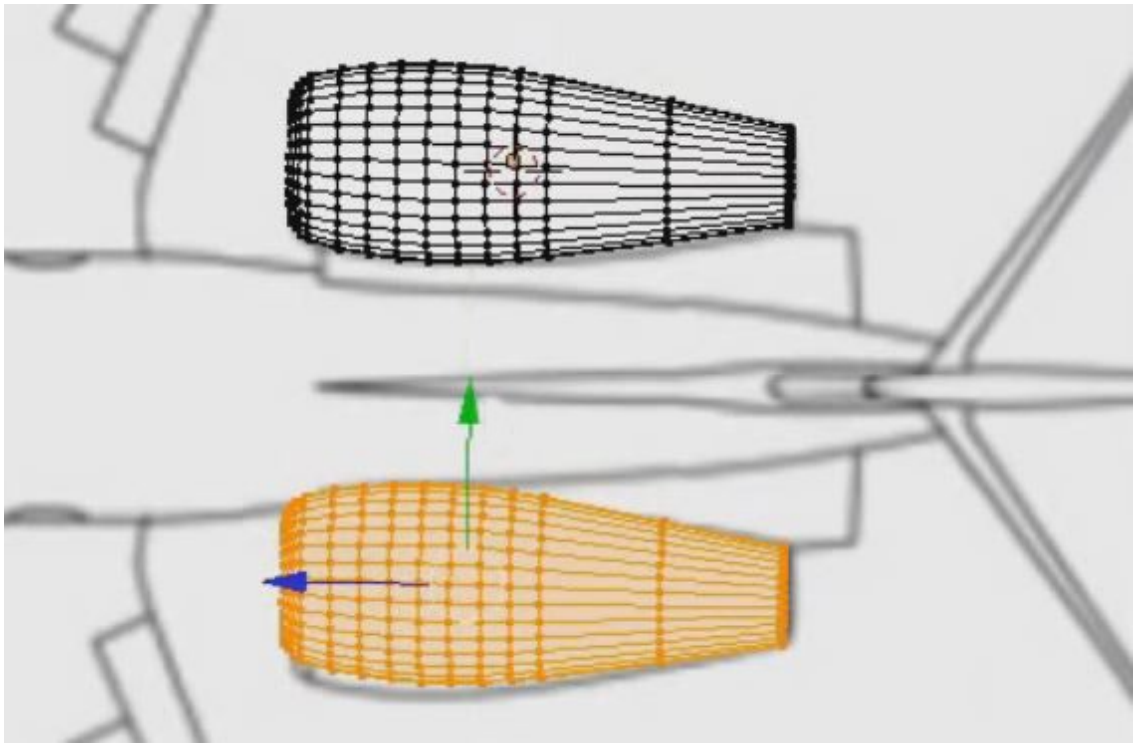
Seguono un paio di rotazioni del modello in Object Mode per orientarlo seguendo le Blueprints di riferimento.



*Ruotare il cilindro, per orientarlo seguendo le blueprints*

Per realizzare la seconda turbina, sono andato in Edit Mode, ho effettuato una duplicazione e un Mirroring, ottenendo quindi il secondo elemento all'interno dello stesso Object (perché la duplicazione è avvenuta in Edit Mode); a ben pensarci, forse avrei fatto meglio a realizzare una copia linkata in Object Mode, con ALT D, in vista di possibili modifiche alla mesh della turbina (in modo da applicarle automaticamente anche alla copia linkata.

Con la duplicazione in Edit Mode o con quella non linkata in Object Mode, ottenuta con SHIFT D, gli elementi sono invece distinti e indipendenti).



*La seconda turbina*

Per completare, Shading Smooth e Subdivision Surface anche su questi elementi.

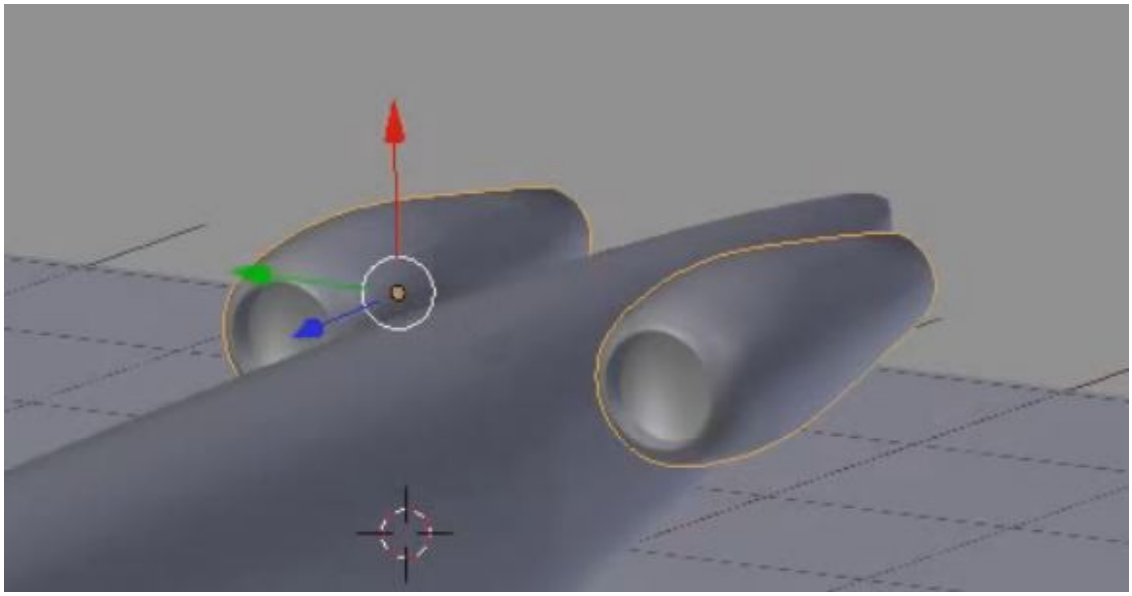
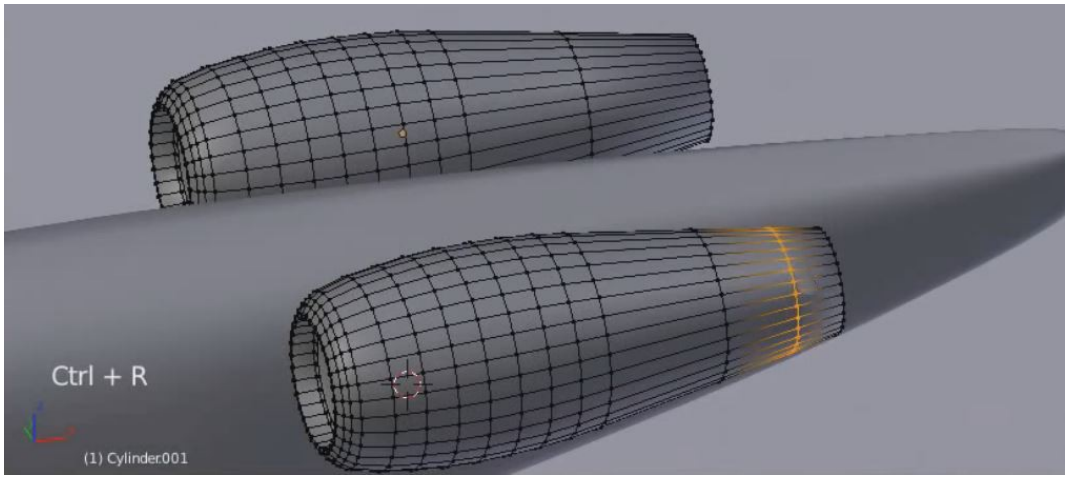


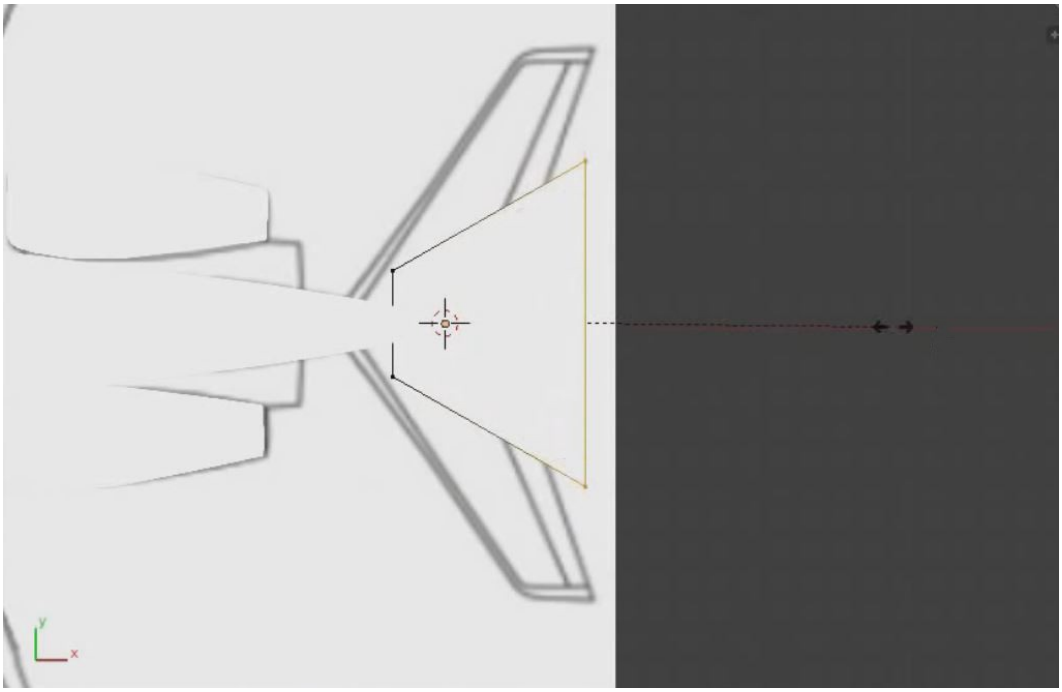
*Illustrazione 1: Le turbine "levigate" (con un problema di ombreggiatura)*

La parte finale dei cilindri ha una brutta ombreggiatura, cosa che può essere risolta con un altro anello di vertici lì (come ho fatto io con Ring Cut e traslazione dei nuovi elementi) o con Mean Crease, che userò in seguito in un altro punto.



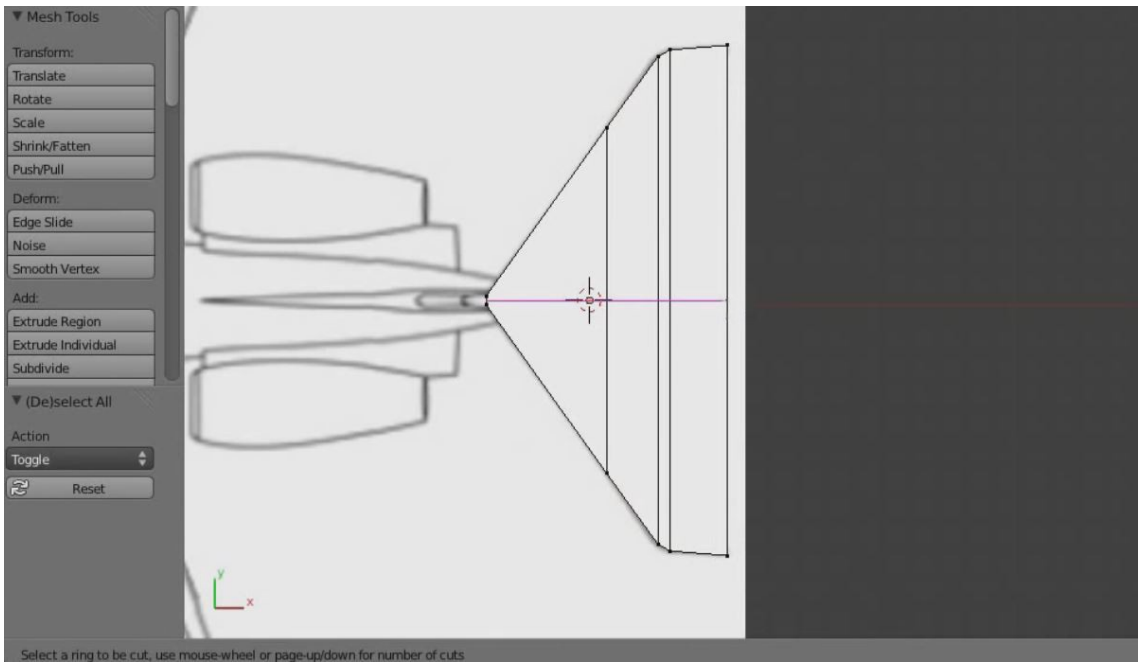

*Risolvere i problemi di ombreggiatura nei cilindri*

Mettiamo di nuovo Viewport Shading *Textured*, nascondiamo fusoliera e reattori e passiamo alla coda, da fare in due parti: elemento orizzontale e verticale. Iniziamo con quello orizzontale.
Qui ci bastano pochi vertici: inseriamo un Plane e iniziamo a spostare i vertici per dargli la forma della coda osservando dalla vista TOP; per inserire nuovi vertici da spostare, ricorriamo come sempre a Ring Cut in Edit Mode.



*Il Plane di partenza per la coda*

Ci vuole anche un taglio orizzontale per rientrare il centro della coda.




*Inserimento di un taglio orizzontale per rientrare il centro della coda*

Questo punto, dopo l'estrusione, darà qualche problema con Subdivision Surface.

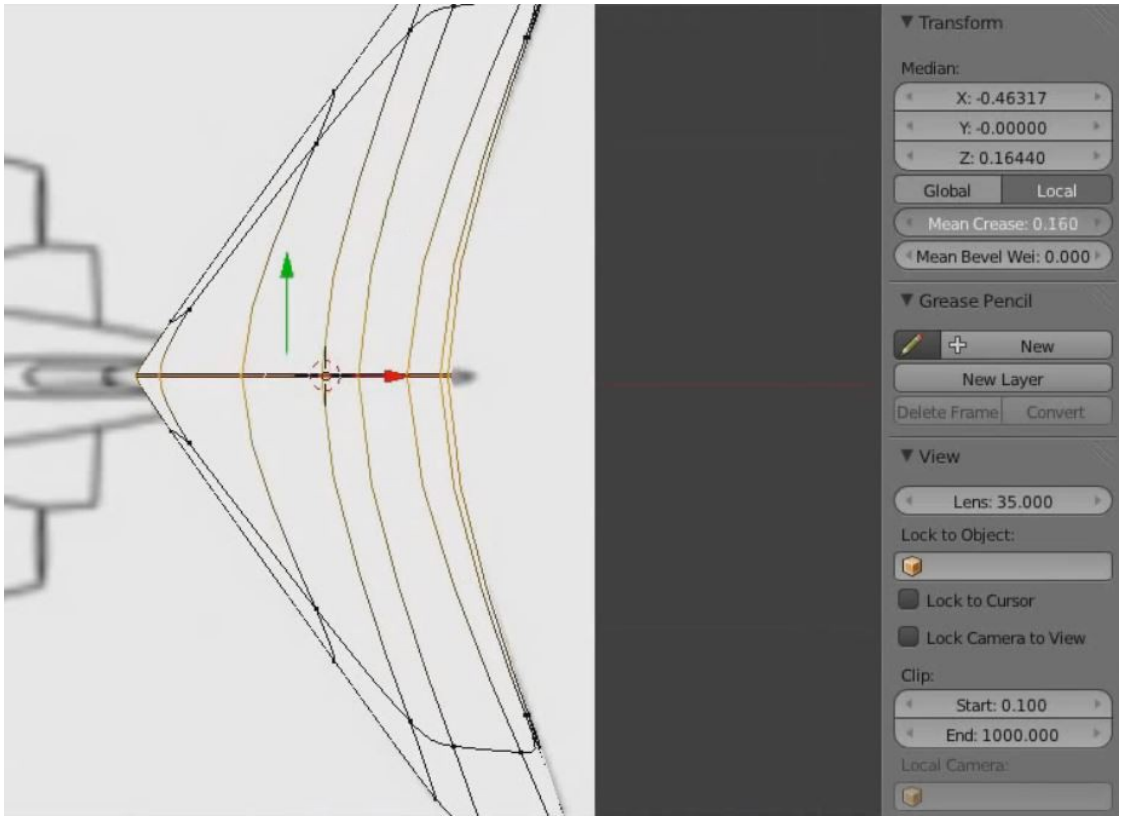
*Problemi dopo il taglio centrale, l'estrusione e Subsurf*

Subito dopo l'estrusione in verticale, per creare lo spessore, c'è un taglio orizzontale con Ring Cut, quindi uno scaling della nuova selezione; un metodo alternativo per ottenere un risultato simile consiste nell'utilizzare il modificatore Bevel (in generale, non c'è solo un modo per ottenere un risultato, ecco perché ogni tanto vi dico "si potrebbe fare anche così ecc...": decidete voi con quali metodi vi trovate meglio, ad esempio utilizzando Mirror, Bevel, eccetera).

Il problema con Subdivision Surface è che arrotonda troppo la coda, mentre al centro la vogliamo proprio ad angolo.

All'inizio ho provato a risolvere con Ring Cut e una nuova suddivisione, come fatto con la parte finale dei reattori, ma questa soluzione qui non va bene.

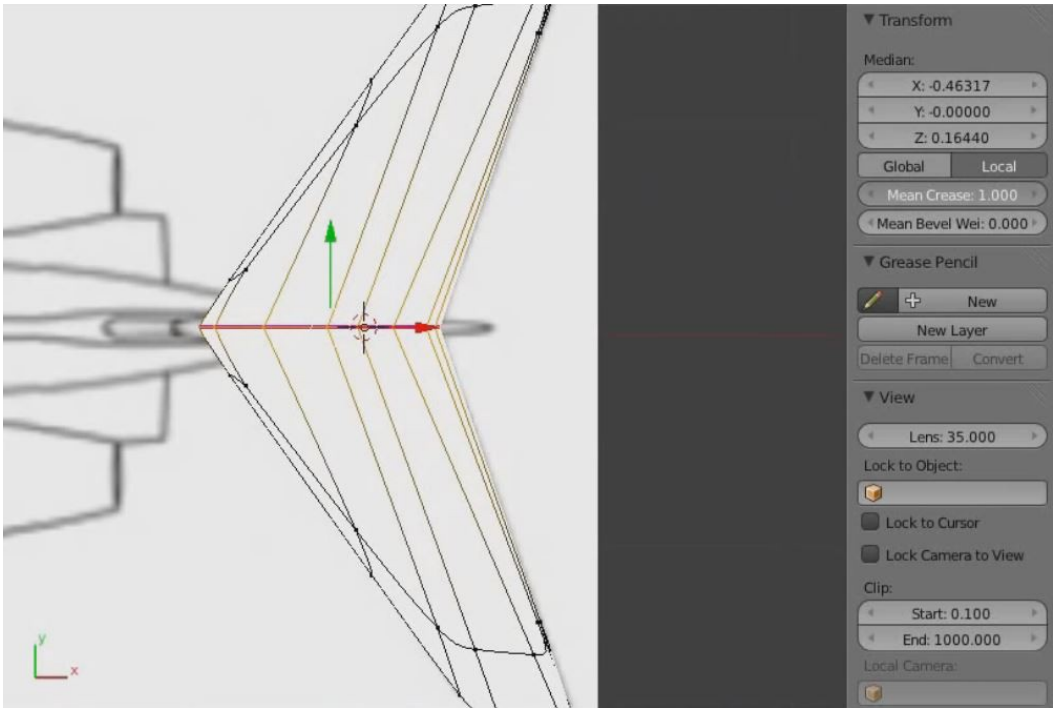
Il problema è che, con queste prove, ho danneggiato un po' la topologia della mesh, infatti si vede come una fessura nel retro della coda, che poi non ho sistemato; la soluzione, comunque, consiste nel selezionare gli spigoli che costituiscono la parte centrale della coda e portare il loro valore di Mean Crease, nella Transform Window, a 1.




*Utilizzo di Mean Crease sulla selezione*

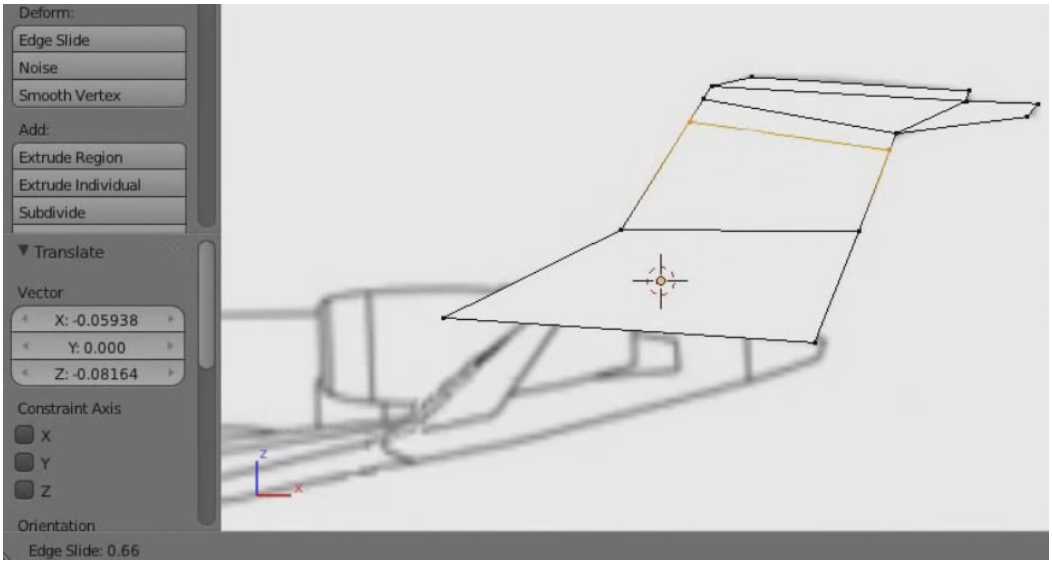
Mean Crease media tra la geometria originale, con le posizioni e le dimensioni originali degli spigoli selezionati, e l'effetto ottenuto con i modificatori, in questo caso Subdivision Surface.

Mean Crease a 0 restituisce il risultato ottenuto con i modificatori, a 1 ci dà la geometria originale; siccome qui vogliamo proprio la forma originale, con gli angoli, portiamo quel campo a 1.




*Ancora su Mean Crease*

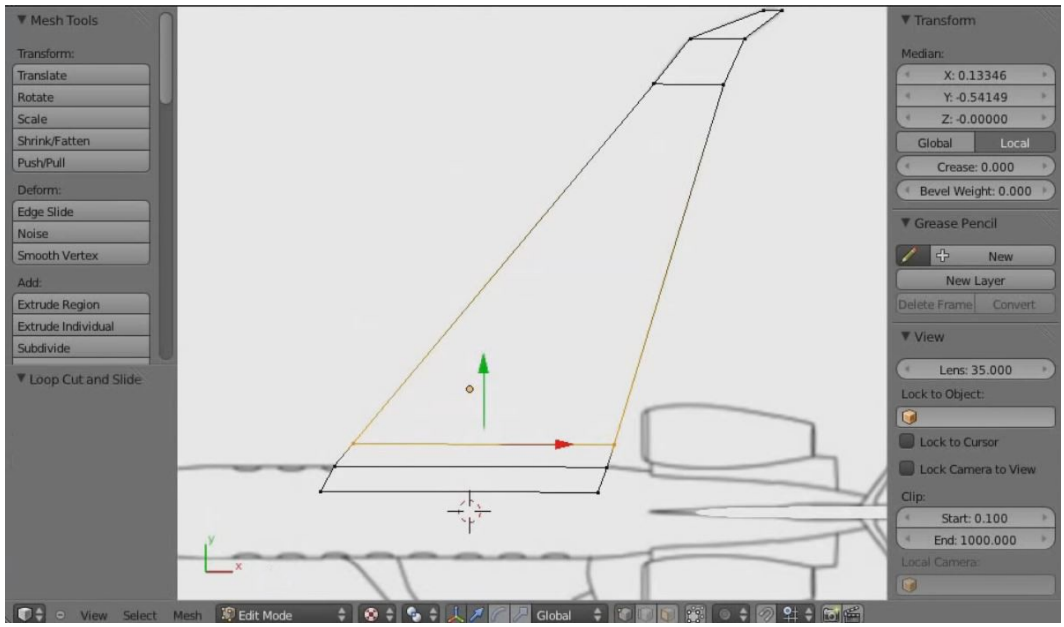
Passiamo alla parte verticale della coda che, banalmente, va realizzata come quella orizzontale, quindi ricapitolando: un Plane, alcune suddivisioni, eventualmente qualche estrusione di spigoli per ottenere delle sporgenze, quindi estrusione in profondità per lo spessore, taglio centrale, *Set Smooth*, Subdivision Surface e, per risolvere eventuali problemi di smussatura, un nuovo taglio con Ring Cut o, come visto con la parte orizzontale, Mean Crease sugli spigoli da mantenere nella configurazione originale.



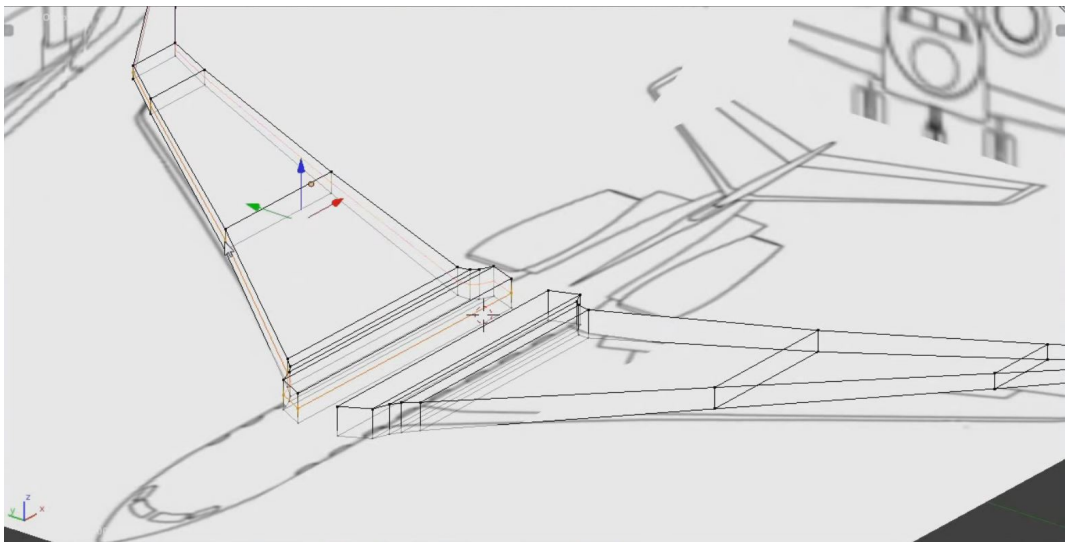

*Realizzare la parte verticale della coda (fase iniziale)*

É giunta l'ora di passare alle ali. L'ho detto e lo ripeto: non sono un grande fan di Mirror, almeno quando ci sono pochi vertici (come in questo caso), per cui ho realizzato prima un'ala e poi l'altra mediante duplicazione (e anche qui, duplicazione in Edit Mode, non copia linkata, mea culpa)... sicuramente ho perso un po' di tempo inutilmente, in questo modo, ma alla fin fine i vertici erano pochi e le geometrie chiare, quindi ci può stare.

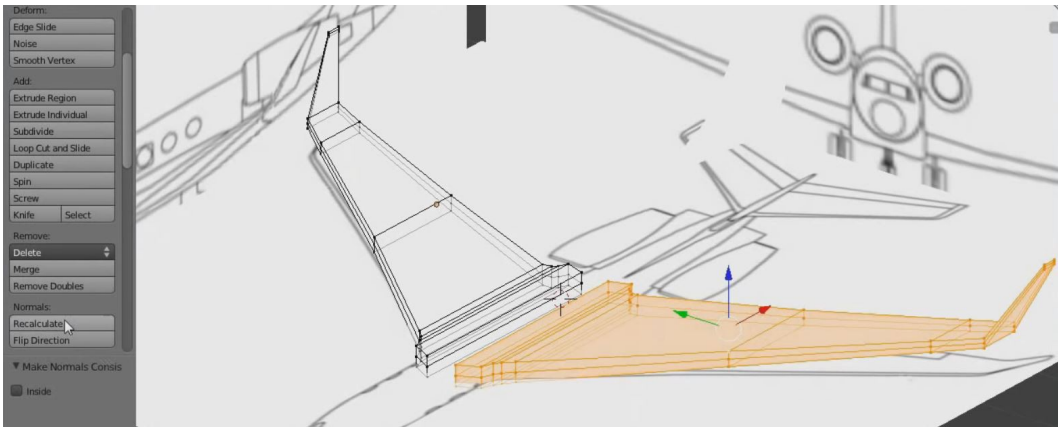
La modellazione dell'ala avviene, comunque, come quella degli elementi della coda: Plane, Ring Cut, traslazioni, estrusioni, eccetera... nient'altro da dire su questo punto.



*Inizio lavori sulle ali*


*Altra fase del lavoro sulle ali*

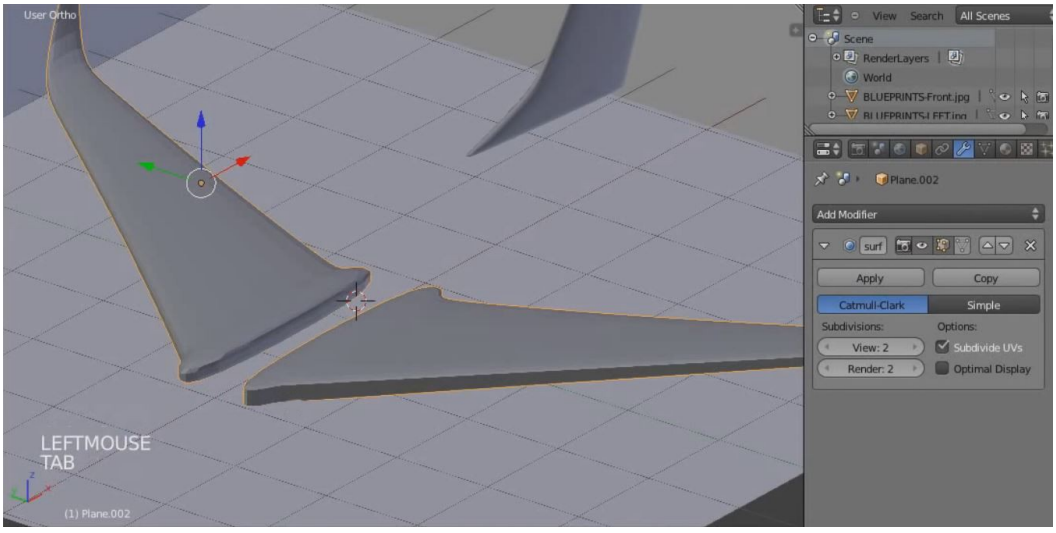
Nell'estrudere la copia dell'ala mi sono accorto che questa copia aveva le Normali invertite (si capisce ad un tratto nel video in quanto, in modalità *Textured*, si riesce a vedere attraverso il modello: questo avviene appunto quando le Normali non sono impostate correttamente).
Siccome questa parte di modello è chiusa e Blender può capire qual è il dentro e qual è il fuori, è sufficiente selezionare tale parte (click su un vertice, poi *L*, per Linked) e cliccare su *Recalculate Normals*, nella Tool Shelf.



*Recalculate Normals*

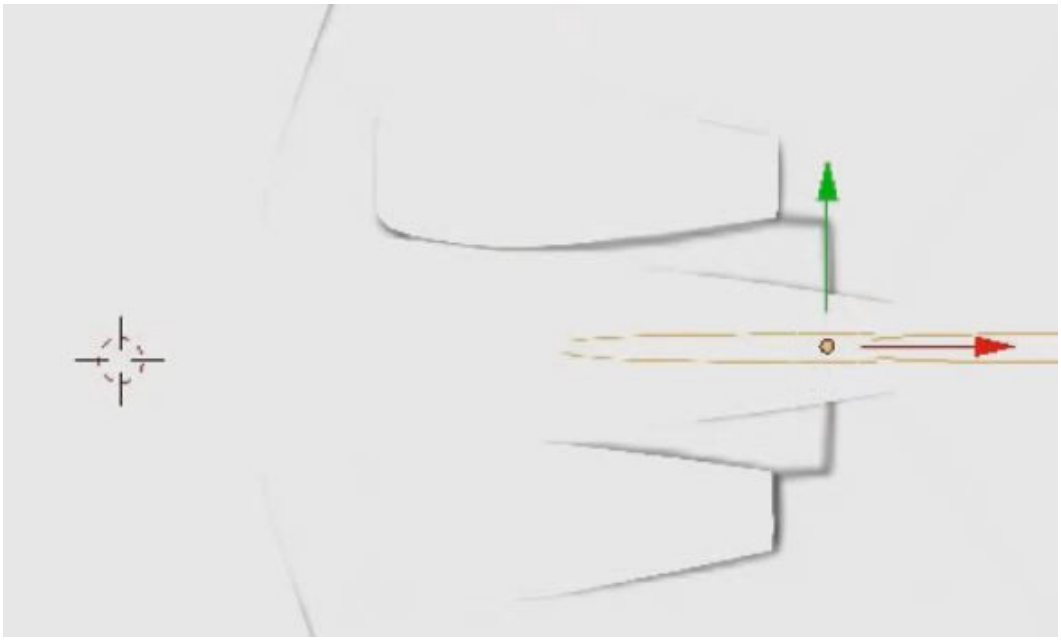
Segue la fase di traslazione e scaling dei vari elementi; da notare che le parti terminali delle ali del G650 vanno verso l'alto, come mostrato nelle Blueprints.

Tanto per cambiare, Subdivision Surface e Set *Smooth*. Qui ho preferito non mettere tagli aggiuntivi o usare Mean Crease, perché il risultato non mi piaceva: la sezione del modello non è comunque quella reale di un'ala, ok, ma almeno lasciamola smussata e levigata!



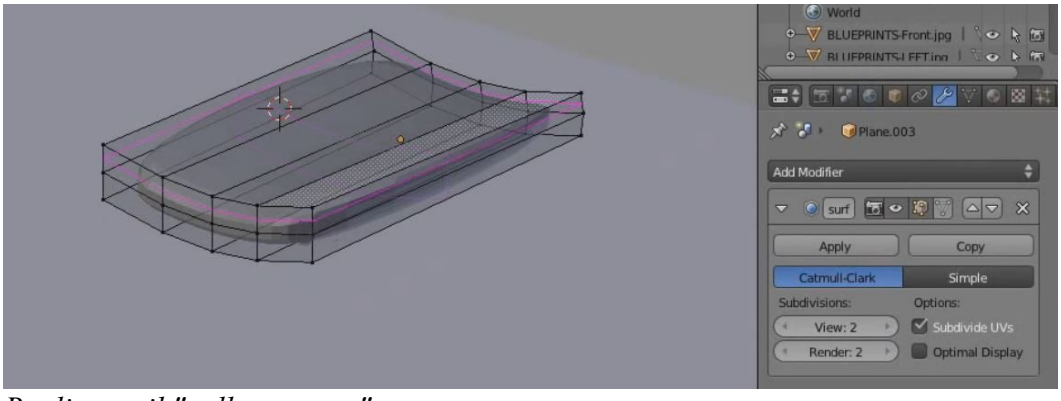
*Smooth e Subsurf sulle ali*

Qui ero convinto di aver finito e stavo osservando il modello, quando mi sono accorto di non aver fatto il pezzo di collegamento tra fusoliera e reattori...



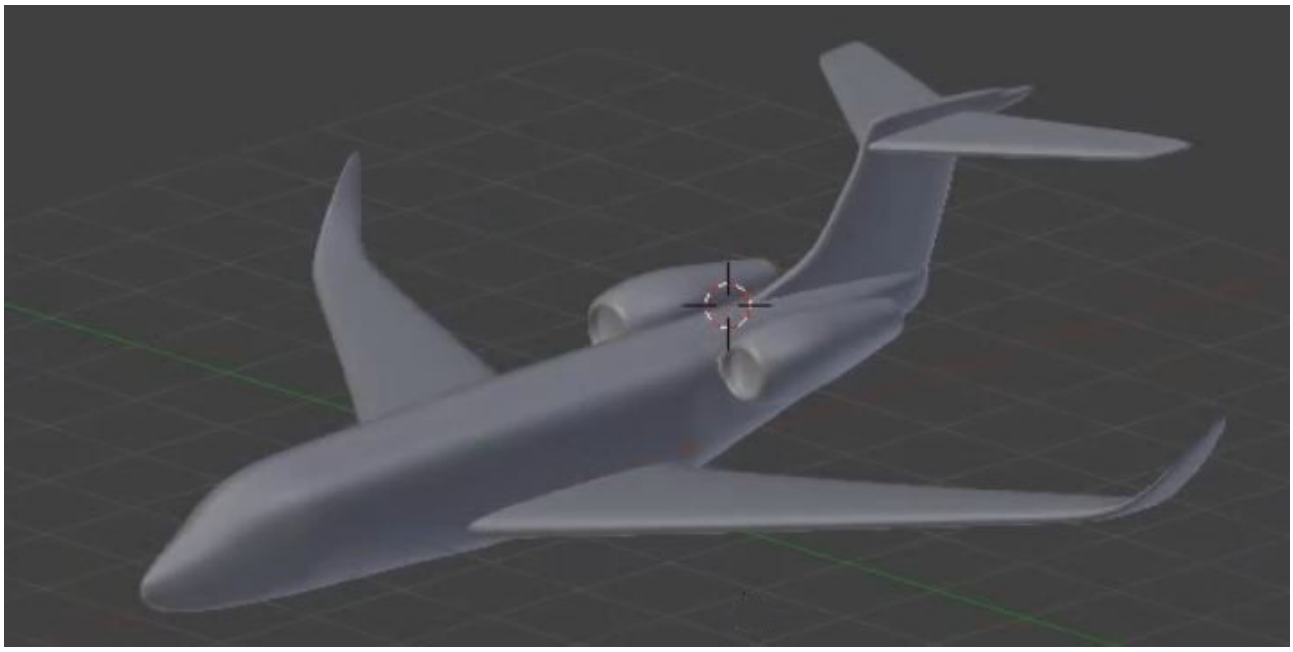
*Manca il "collegamento" tra fusoliera e reattori*

… niente di particolarmente complicato, comunque: un Plane (o anche un Cubo, a questo punto), estrusioni, scaling, tagli, traslazioni e Subdivision Surface; questo è davvero un elemento semplice e non ha importanza se qualche pezzo finisce dentro la fusoliera o il reattore, non lo stiamo progettando per la realizzazione reale o la stampa 3D, possiamo prenderci qualche libertà!




*Realizzare il "collegamento"*

Per la stampa 3D, in fondo, basta fondere il tutto, ma, come ho detto in molti tutorial pubblicati precedentemente, l'operazione va fatta con il modificatore *Boolean Union*, non con il semplice Join, che invece va bene se volete esportare questo modello per usarlo ad esempio in un Game Engine; il semplice Join, tra l'altro, lascia separati i vari elementi, cosa che vi consentirà di selezionarli, unwrapparli e texturizzarli con facilità.



*Il risultato finale*

*  *  *

# Copie radiali con Spin, Pivot Point e Array intorno a Empty

In questo tutorial vedremo tre modi per disporre copie di un oggetto intorno ad un centro di rotazione, ossia come ottenere copie "radiali", in Blender 3D; parleremo, in particolare, di Spin, Pivot Point e Array, trattando anche le copie "linkate".

Per disposizione radiale intendiamo il posizionamento e l'orientamento degli oggetti intorno a un punto, detto generalmente perno (Pivot Point). Oltre alla traslazione, è coinvolta anche la rotazione, perché gli elementi devono orientarsi verso il perno.

In genere, per un lavoro simile si può procedere con Spin, che però va fatto in Edit Mode, con i pro e i contro che vedremo, o con una rotazione intorno al 3D Cursor, utilizzato come perno della rotazione. Nel caso di Spin, si va in Edit Mode, si va in una vista perpendicolare al piano di rotazione degli elementi, si posiziona il 3D Cursor proprio nel centro della rotazione e si clicca su Spin, nella Tool Shelf; poi, in basso in tale pannello, si impostano il numero di copie (Steps) e l'angolo di apertura.

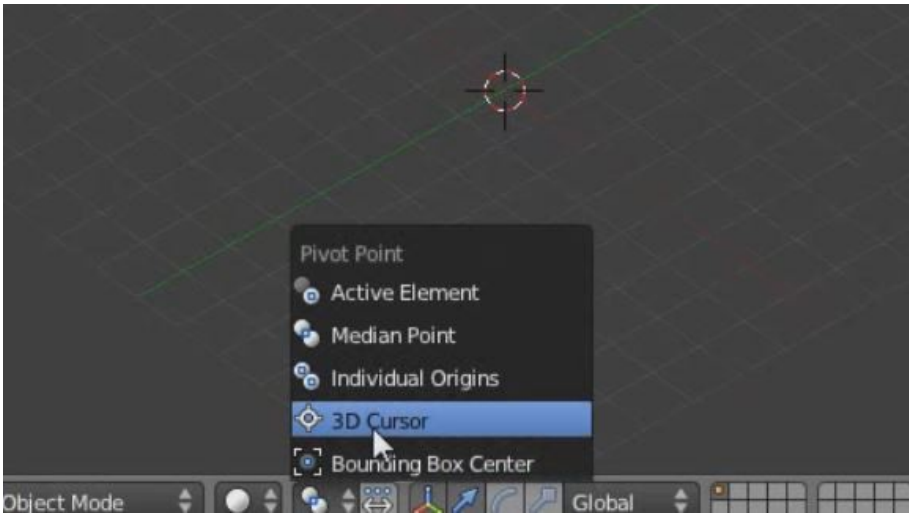
*Utilizzo di SPIN*

Di buono, questo strumento ha che è preciso, veloce e lavora in Edit Mode, quindi l'oggetto finale farà parte dell'originale; di negativo, ha parecchie cose: una volta fatte, le copie non dipendono più dall'originale, quindi una modifica apportata ad un punto non si rifletterà nelle copie, inoltre i nuovi vertici non sono istanze del modello originale ma elementi a parte, quindi il modello (e il file della scena) occuperanno più spazio in memoria.

Passiamo al caso della rotazione intorno a un Pivot Point esterno, che vedremo tra l'altro con le copie linkate.

Anche qui la posizione del 3D Cursor gioca un ruolo importante: è il centro della rotazione, quindi per prima cosa posizioniamolo in un punto preciso mediante le voci della sezione *3D Cursor Location*, nella Transform Window.

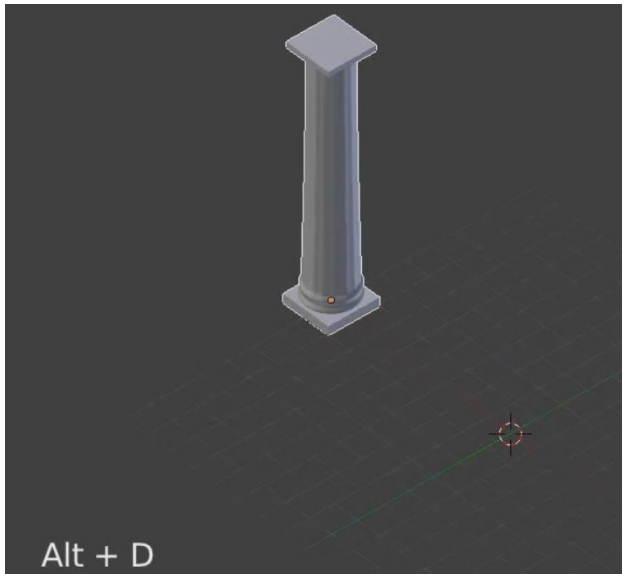
Adesso, nell'header della 3D View, scegliamo *3D Cursor* dal menù *Pivot Point*.




*Impostare il 3D Cursor come Pivot Point (perno delle trasformazioni)*

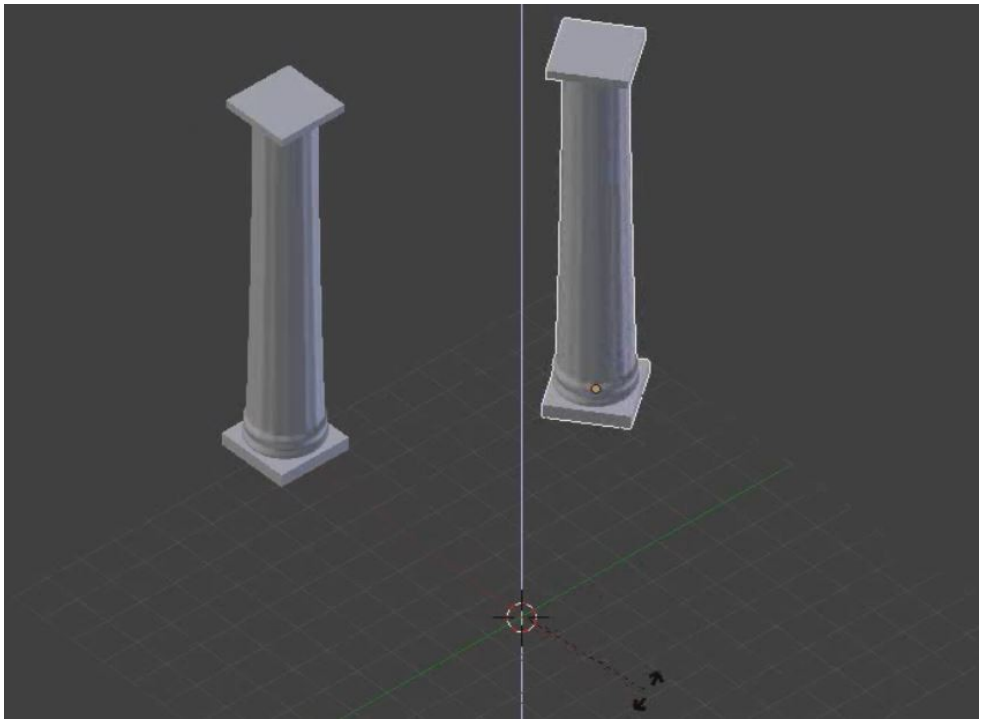
Da qui in poi dobbiamo stare attenti a non fare click col tasto sinistro del mouse in qualche punto della 3D View, pena lo spostamento del 3D Cursor e, quindi, del centro di rotazione (non è gravissimo, basta riposizionare il 3D Cursor ed eventualmente annullare copie errate con CTRL Z, ma si tratta comunque di una fastidiosa perdita di tempo).

Facciamo una copia sul posto dell'oggetto, magari una copia-linkata o istanza, quindi con l'oggetto selezionato premiamo ALT D e subito dopo Invio. Di default, avremo la copia selezionata nella 3D View.



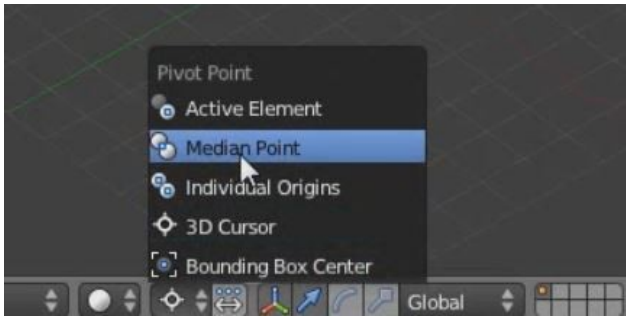
*La copia linkata (ALT D) sul posto*

Il Pivot Point è il 3D Cursor, quindi premendo ad esempio R Z potremo ruotare la copia intorno a quel punto, e magari specificando un valore numerico dopo (ad esempio R Z 45) potremo fare una rotazione precisa, parametrica.


*Rotazione della copia linkata intorno al 3D Cursor (qui utilizzato come Pivot Point)*

Lati positivi: è un sistema facile, veloce, che non fa uso di oggetti esterni come le Empty (che useremo con Array), inoltre le copie sono linkate, quindi una modifica effettuata sulla prima mesh trasformerà anche le copie; di contro, se si devono fare molte copie con valori precisi, o se ancora il numero di copie (e, quindi, le loro posizioni e gli orientamenti) non sono definitivi, si può perdere del tempo nelle modifiche, in seguito.

Veniamo quindi alla terza via: l'utilizzo dello strumento Array in modalità radiale.



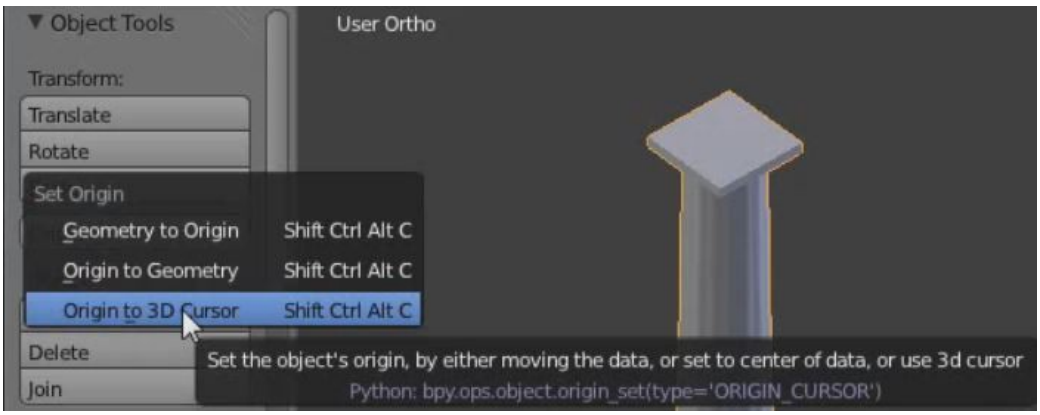
*Riportare il Pivot Point a "Median Point"*

Riportiamo il *Pivot Point* a *Median Point*, come di default.

Array è un Modifier che generalmente viene utilizzato per effettuare copie lungo una direzione, ma può essere utilizzato anche rispetto ad un oggetto di riferimento esterno, tipicamente una Empty, e in questo caso le copie possono essere trasformate in posizione, orientamento e scaling con una notevole flessibilità; di contro, c'è da perdere un po' di tempo per la sua corretta configurazione.

In particolare, l'Origin dell'oggetto iniziale deve trovarsi nel perno della rotazione; in quel punto deve trovarsi anche una Empty; infine, l'oggetto originale deve avere le trasformazioni di rotazione e scaling applicate. Procediamo con ordine.

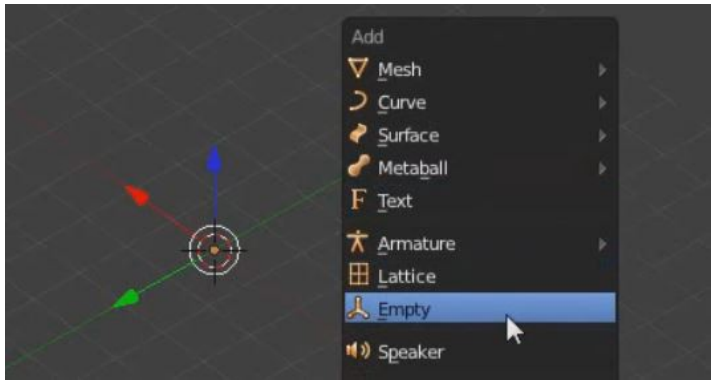
Posizioniamo il 3D Cursor nel centro della rotazione. Con la mesh selezionata, clicchiamo su Origin, nella Tool Shelf, e scegliamo "*Origin to 3D Cursor*".



*Origint to 3D Cursor, nel centro della rotazione*

Saltiamo il passaggio dell'applicazione delle trasformazioni: ve lo ricorderete meglio quando vedrete che errori si hanno se non lo si applica... e quindi vi ricorderete della soluzione.

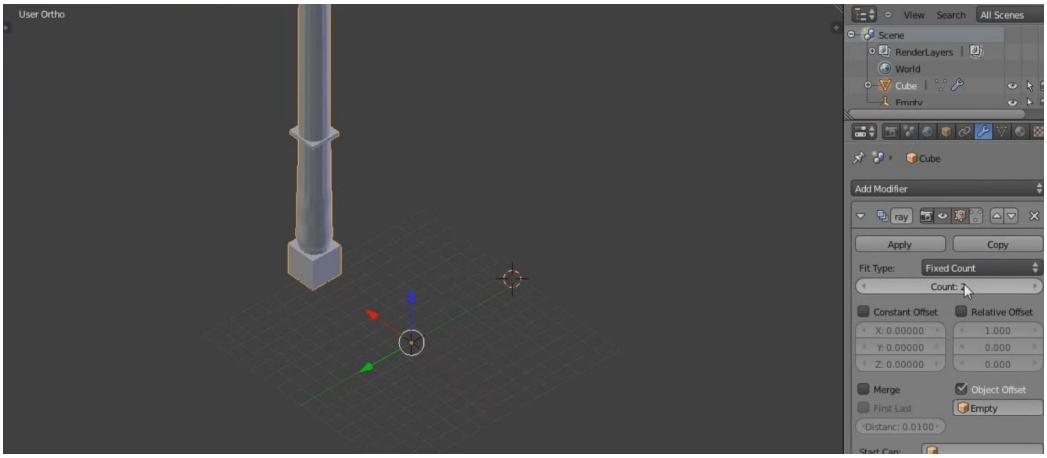
Sempre con il 3D Cursor nel centro della rotazione, aggiungiamo un oggetto Empty alla scena, che quindi verrà inserito proprio in quel punto. Da qui in poi possiamo spostare il 3D Cursor dove vogliamo, senza problemi: ora il testimone passa alla Empty.




*Posizionamento della Empty*

Adesso selezioniamo di nuovo la mesh originale, apriamo il pannello Modifiers nella Properties Window e aggiungiamo un modificatore *Array*. Impostiamo ad esempio 10 copie, ma in ogni caso potremo tornare su questo valore anche in seguito, quindi non si tratta di un'impostazione definitiva... almeno, se non si applica il Modifier!

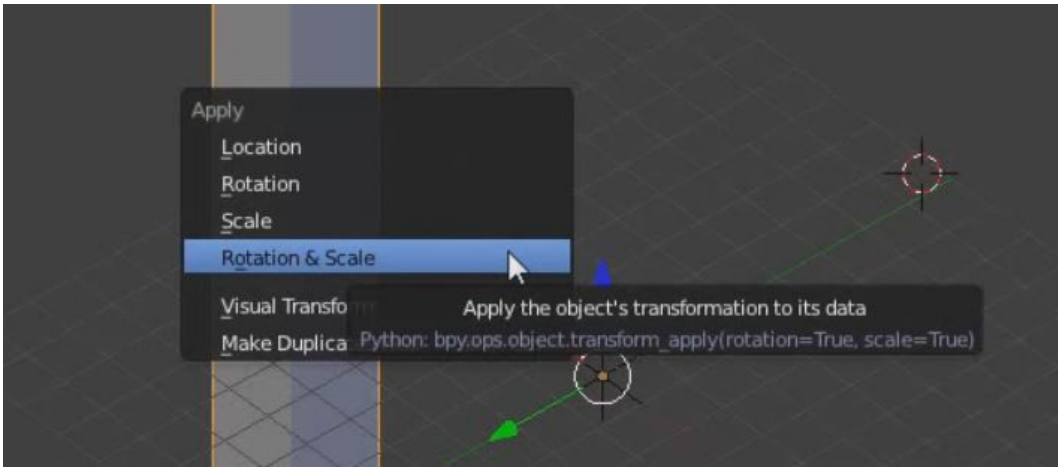
Deselezioniamo la voce *Relative Offset*, selezionata di default, e selezioniamo invece *Object Offset*, specificando la Empty nel campo Object subito sotto.




*Object Offset, con la Empty e... un errore*

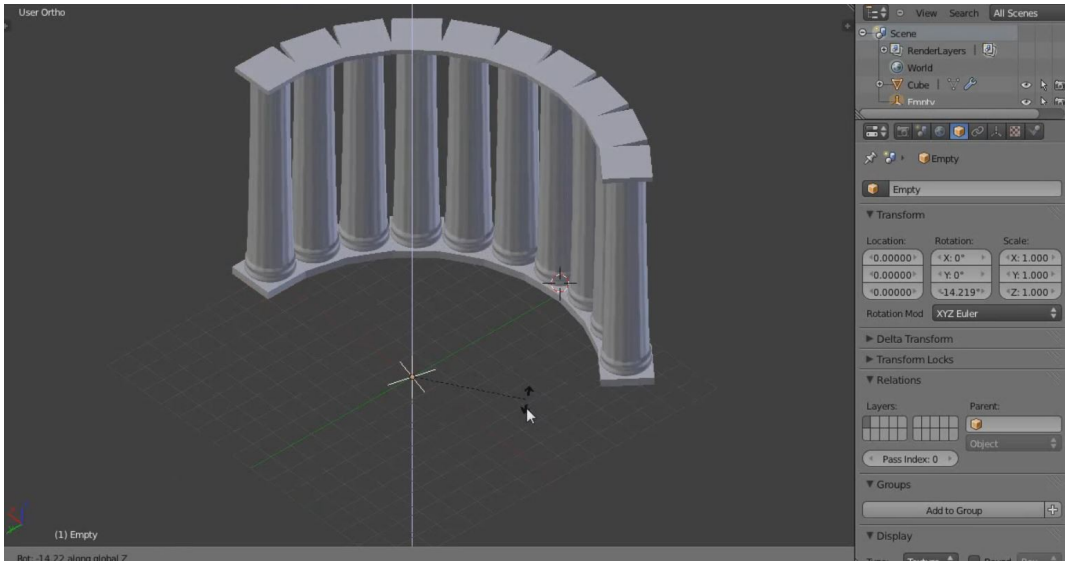
Ecco visibile l'errore di cui vi parlavo: la copia ha una forma strana e comunque è spostata rispetto all'originale, mentre con la Empty nella configurazione iniziale dovrebbe coincidere con l'originale.

Quello che bisogna fare è premere CTRL A, mentre il cursore del mouse è sulla 3D View, e scegliere di applicare sia la Rotazione che lo Scaling.



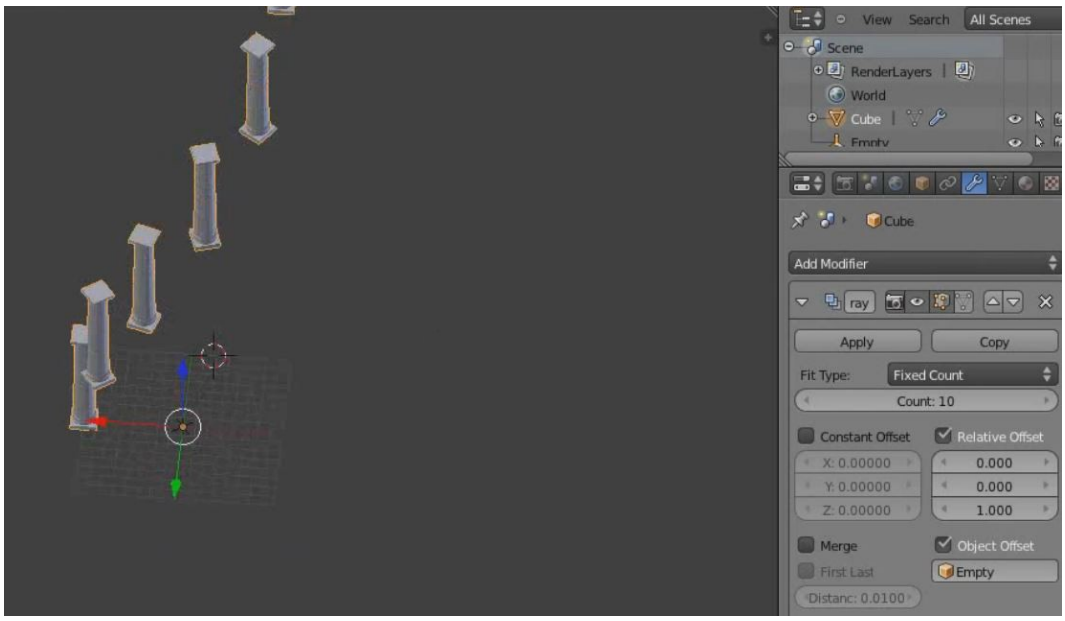

*Applicazione delle trasformazioni di rotazione e scaling*

Ora ci siamo: non dobbiamo far altro che selezionare la Empty e trasformarla, cosa che nel nostro caso significa ruotarla intorno all'asse Z (ad esempio con R Z 36, cioè 360 diviso il numero di copie), ma ovviamente siete liberi di provare altre trasformazioni, come traslazioni o scaling, o trasformazioni complesse attivando anche *Constant* o *Relative Offset* nella scheda del Modifier.




*Rotazione della Empty intorno all'asse Z (verticale)*

Ad esempio, attiviamo *Relative Offset* e impostiamo XYZ a 0, 0, 1, in modo da avere uno spostamento relativo in altezza: se il risultato vi fa pensare a una scala a chiocciola, complimenti per l'intuizione: quel tipo di oggetti si può ottenere facilmente così.




*Utilizzo di Relative Offset, insieme ad Object Offset*

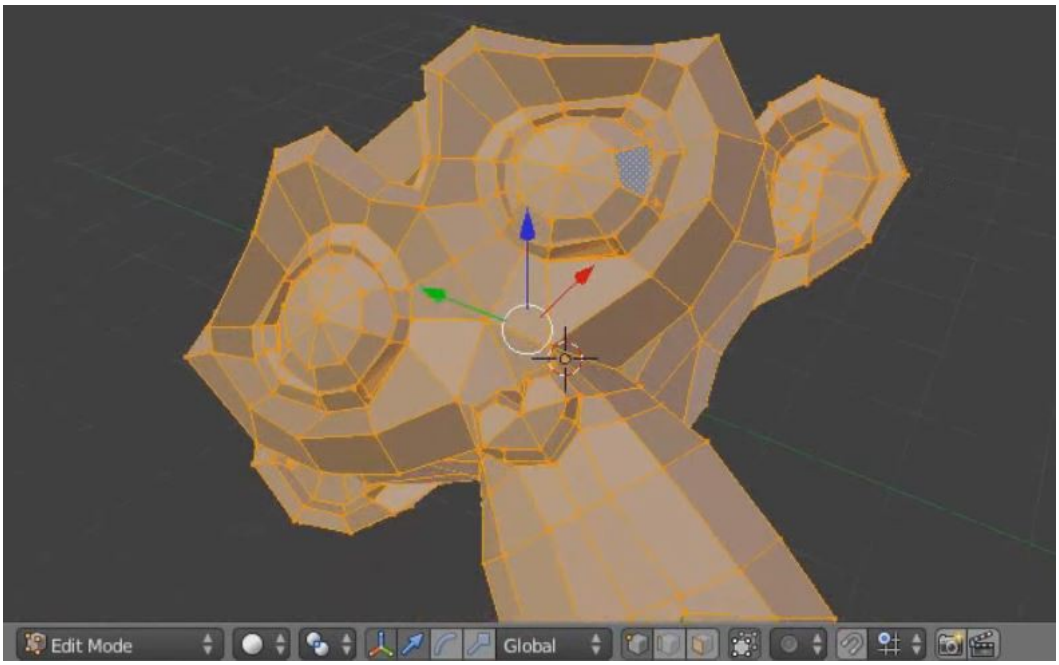
Di buono, quindi, abbiamo la possibilità di combinare più offset su tutti i tipi di trasformazione, cosa che appunto ci consente di realizzare anche oggetti apparentemente complessi come una scala a chiocciola, inoltre (a meno di non applicare il modificatore) le copie sono istanze dell'originale, quindi una modifica apportata alla prima verrà applicata anche alle altre e in generale il file *.blend* della scena occuperà poco; di contro, il tutto richiede un po' di tempo per la configurazione, visto che abbiamo a che fare anche con Empty, spostamento dell'Origin della mesh e applicazione di Rotation e Scale.

Una nota a parte, che può servirvi da spunto per fare pratica con Array: i parametri di un Modifier Array sono tutti animabili, così com'è animabile la forma della mesh originale o posizione, orientamento e scaling della Empty usata come riferimento; questo apre la strada alla realizzazione di animazioni interessanti, anche artistiche se vogliamo: basta avere chiaro in mente cosa si vuole ottenere e, quindi, tradurlo in valori dei parametri e keyframes.

*  *  *

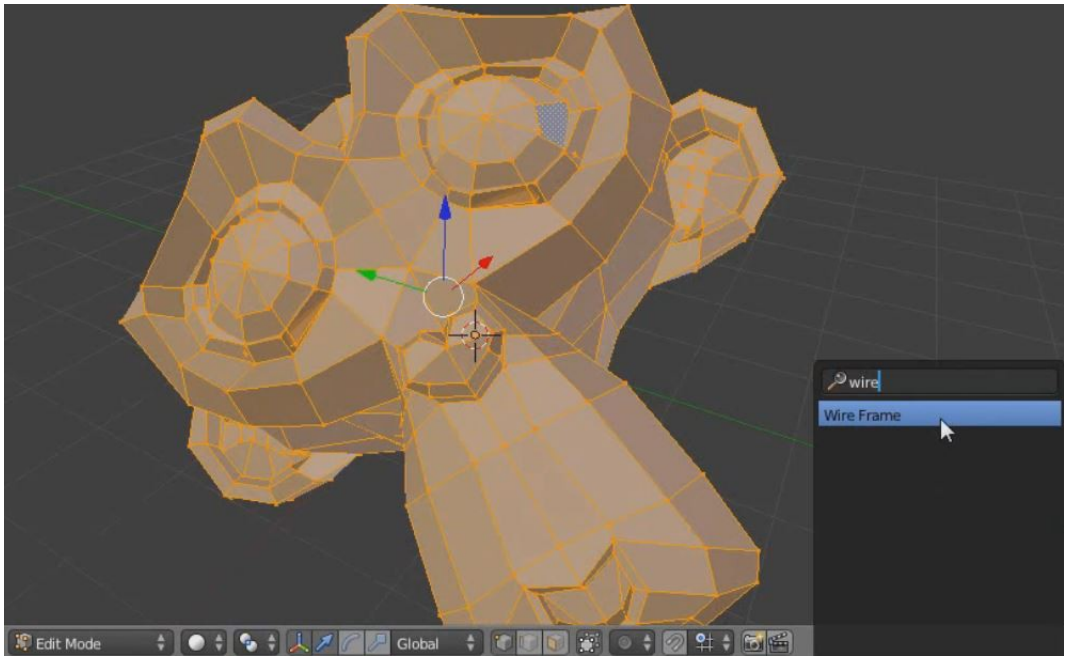
# Il nuovo strumento Wireframe per le mesh

Questo più che un tutorial è un tips & tricks su una nuova funzionalità di Blender 3D 2.64: lo strumento Wire Frame, che consente di ottenere la mesh dei soli spigoli, resi come prismi, di un modello 3D. Per fare un esempio pratico di utilizzo dello strumento, che può essere applicato a qualsiasi mesh della scena, utilizziamo Suzanne, visto che è una primitiva che mette già a disposizione un bel po' di spigoli.



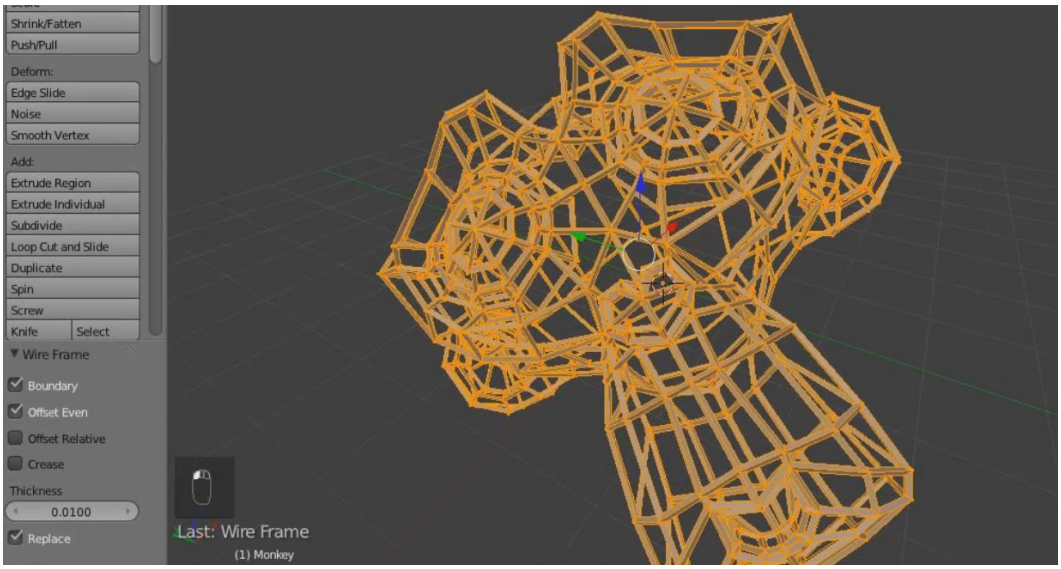
*La scena di partenza per questo tutorial*

Con tale mesh selezionata, passiamo in Edit Mode, premiamo la barra spazio mentre il cursore del mouse è su una 3D View e digitiamo, nella casella di help veloce che apparirà, “Wire Frame”, con lo spazio tra Wire e Frame, quindi scegliamo tale strumento nella casella.



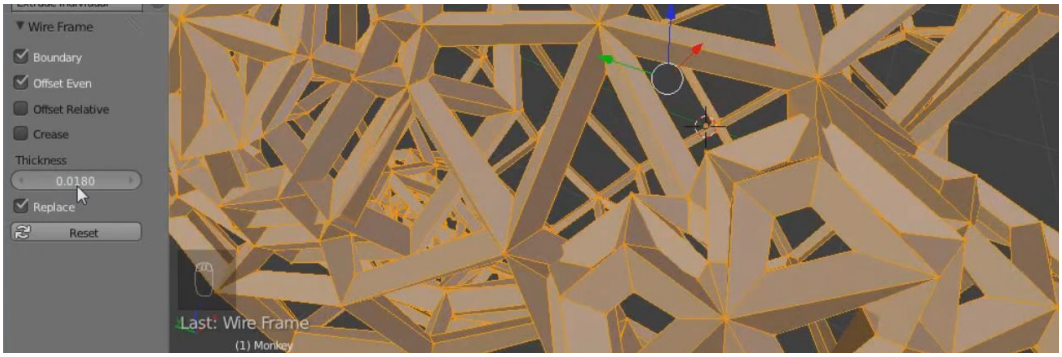

*Scelta dello strumento Wire Frame, dalla casella di ricerca veloce*

L'effetto è visibile subito: della mesh originale, restano solo gli spigoli, realizzati però come mesh, dei prismi, con una sezione variabile, non come singoli edge, spigoli di collegamento tra vertici.



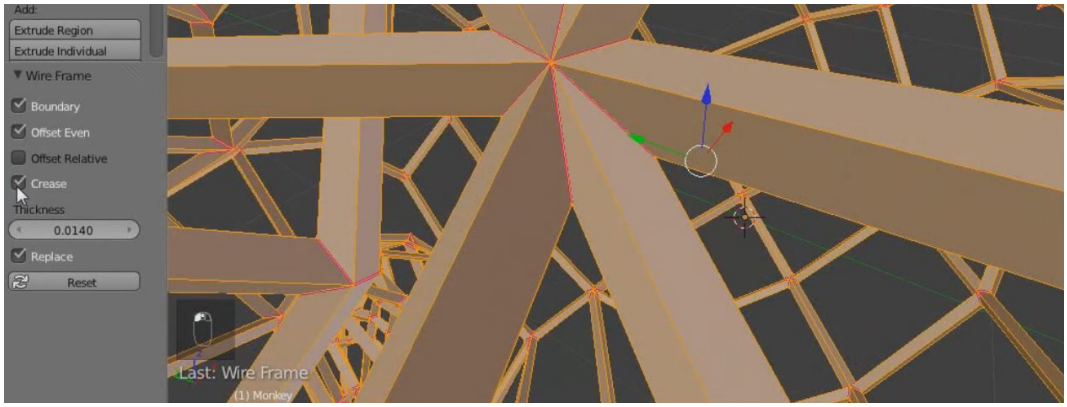

*Primi effetti di Wire Frame*

La dimensione dei collegamenti può essere modificata: è sufficiente modificare il valore del parametro *Thickness*, letteralmente spessore, nel pannello dello strumento Wire Frame che appare in basso nella Tool Shelf (apribile con T in una qualsiasi 3D View) subito dopo aver applicato lo strumento; il valore 0.0 produce in pratica gli spigoli, mentre aumentandolo abbiamo dei prismi con sezioni di varie dimensioni.




*Wire Frame Thickness, nella Tool Shelf*

Altro parametro importante è *Crease* che, se attivato, aumenta il Mean Crease degli spigoli nelle giunture; ho parlato di Mean Crease e del suo utilizzo con Subdivision Surface in altri tutorial, ad esempio in quello della modellazione del Gulfstream G650: detto brevemente, Mean Crease è un valore associato agli spigoli di una mesh (ogni spigolo può avere un valore Mean Crease diverso dagli altri, non per forza tutti uguali) e tanto maggiore è tale valore (da 0 a 1, nella Transform Window con lo spigolo selezionato), minore sarà l'effetto di un eventuale Subdivision Surface su quello spigolo, quindi minore la distorsione introdotta da Subdivision Surface in quella zona.

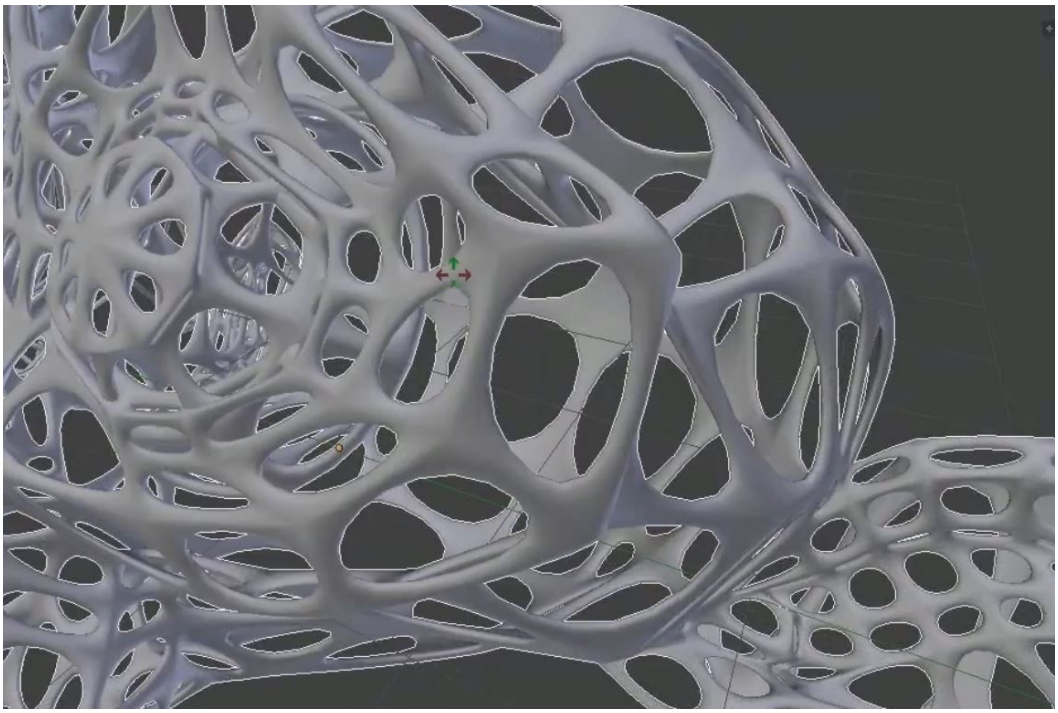



*Utilizzo di Crease per Wire Frame*

In un certo senso, gli spigoli con *Crease* a 1 restano congelati, non vengono modificati da Subdivision Surface.

*Crease*, in Wire Frame, serve quindi a tenere fermi, per così dire, gli spigoli delle giunture, in previsione dell'utilizzo di un modificatore Subdivision Surface sulla mesh così realizzata; in questo modo, mantenendo ferme le giunture, la forma generale della mesh verrà preservata.

Personalmente penso che Wire Frame seguito da Subdivision Surface SENZA opzione Crease attiva produca risultati davvero particolari... sarebbe interessante provare a stampare i modelli 3D così realizzati con una stampante 3D: potrebbero venire fuori simpatici oggetti decorativi o d'arredo, ma questo è un altro discorso!



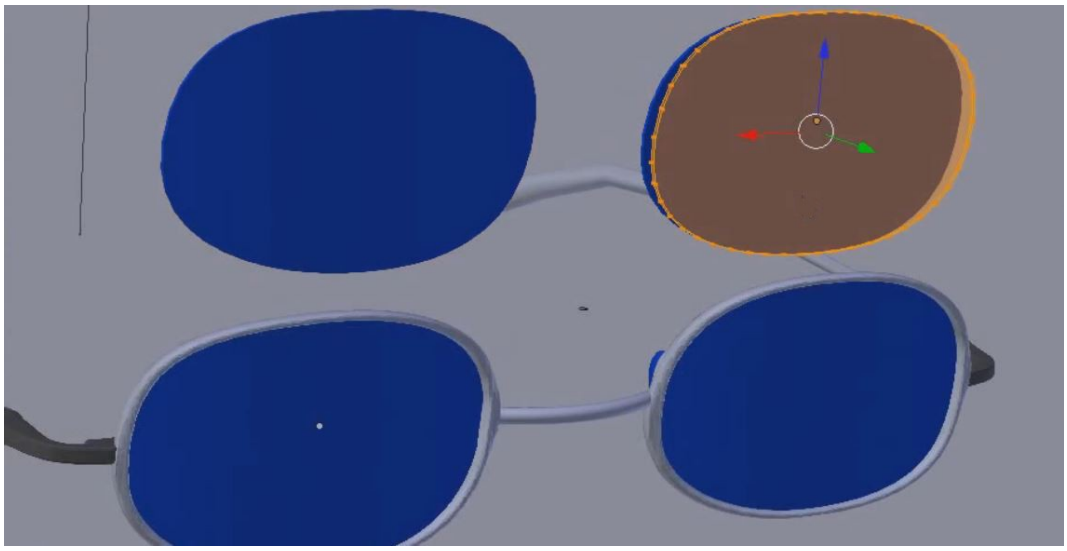
*Esempio di utilizzo di Wire Frame + Subsurf senza Crease*

*  *  *

# Le lenti degli occhiali: tips & tricks di modellazione e rendering

Questo più che un tutorial è un piccolo tips & tricks riguardante alcuni aspetti della modellazione in Blender 3D (qui versione 2.64) e, in particolare, sulle topologie delle mesh e dell'importanza delle stesse in vista del rendering, effettuato in questo caso con Cycles.

La scena che utilizzeremo è, in effetti, quella che ha fatto emergere il problema e che ha dato lo spunto per questo tutorial. Si tratta del primo lavoro in Blender 3D di "Happosai", che, come saprete, ogni tanto mi aiuta a realizzare qualche tutorial o videotutorial su Photoshop, quindi più che altro grafica 2D, ma che ora sta muovendo i primi passi con Blender.

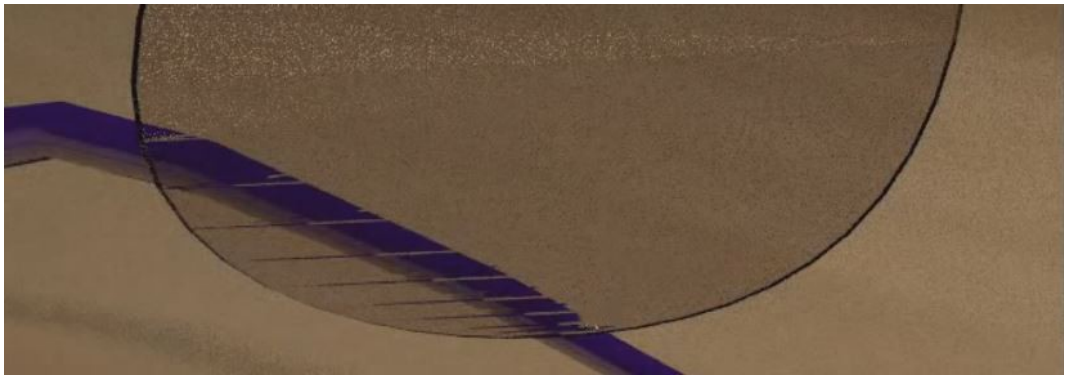


*La scena utilizzata per questo tutorial*

Il problema, in particolare, sta nelle lenti: inizialmente, sono state realizzate partendo da Curves, curve Bezier 2D, quindi piatte, poi trasformate in mesh, dotate di spessore e piegate mediante l'utilizzo di Simple Deform – *Bend*.

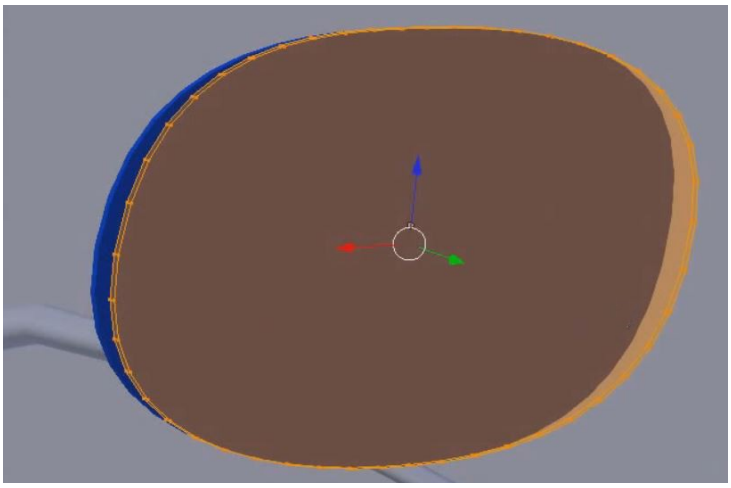
Il riempimento delle lenti può portare, a seconda delle geometrie utilizzate per implementarlo, a effetti molto diversi.

Se ci si basa sul riempimento dato dalla Bezier Curve convertita in mesh o su quello di una mesh inizialmente senza riempimento e poi riempita con il metodo "tradizionale", ossia *ALT F*, si avranno delle brutte strisce, storte, dovute al riempimento implementato con facce triangolari di collegamento tra i vari vertici del contorno dell'oggetto.



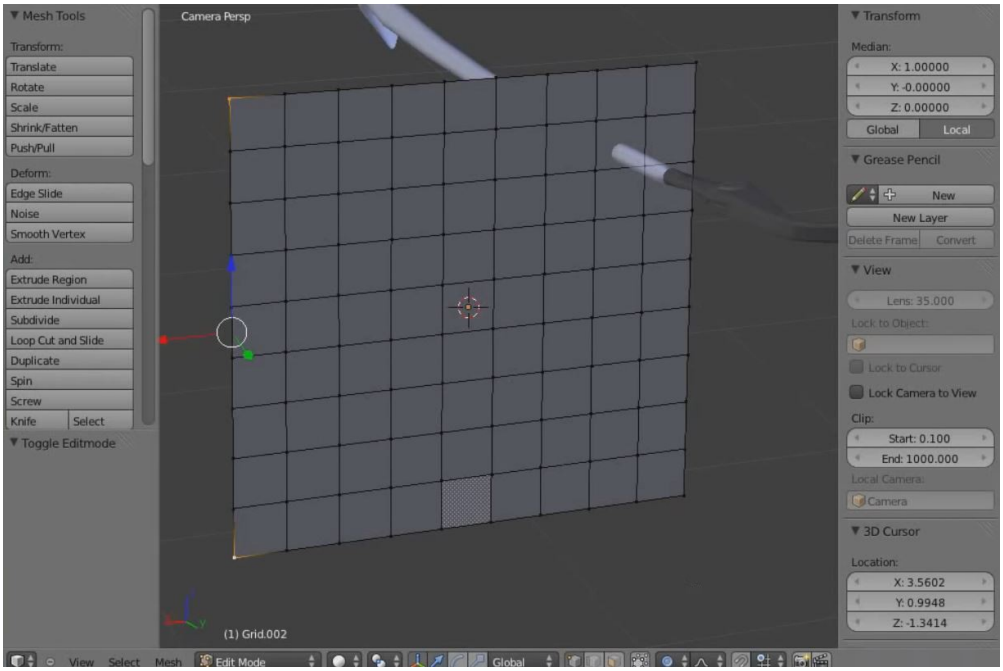
*Riempimento con ALT F*

La situazione non migliora se si effettua il riempimento con *F*, disponibile dalla versione 2.63 di Blender con modellazione Bmesh: ora si vedono delle strisce orizzontali, magari dritte, ma sempre di strisce si tratta... questo perché lì per lì la visualizzazione della superficie mostra solo una faccia, è vero, ma all'atto pratico, quando si passa al rendering, i collegamenti tra i vertici vengono comunque realizzati, ed ecco saltar fuori le strisce.



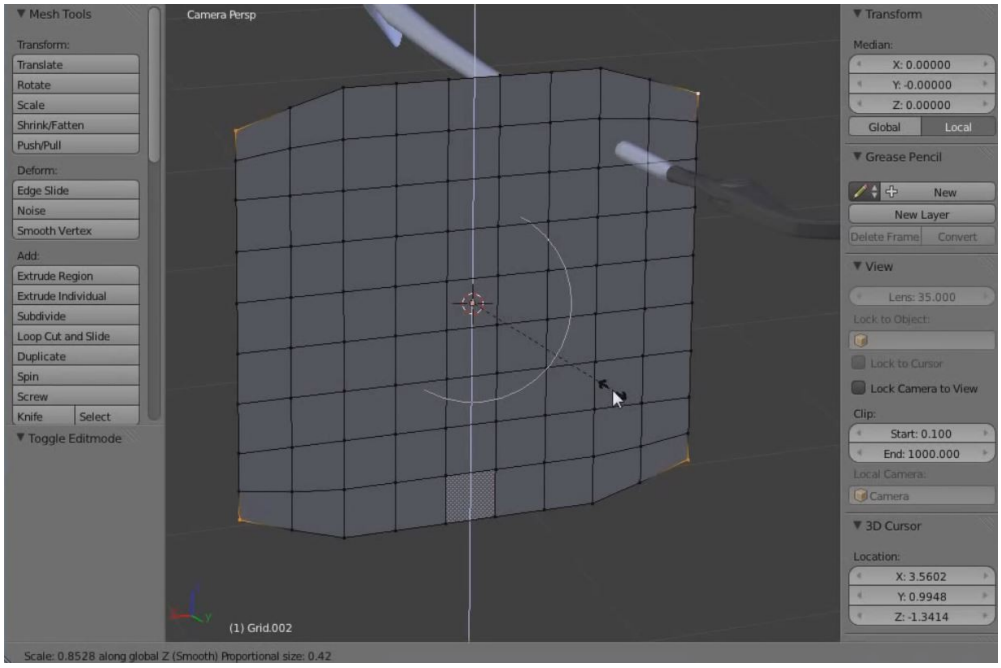
*Riempimento con F (BMESH)*

L'ideale sarebbe, quindi, una superficie fatta di facce quadrangolari, e più piccole saranno le facce minori saranno le righe e le distorsioni introdotte.
Per modellare questa superficie, anziché partire dal semplice Plane aggiungiamo un oggetto Grid, che ha già un buon numero di suddivisioni.




*Utilizzo di una GRID*

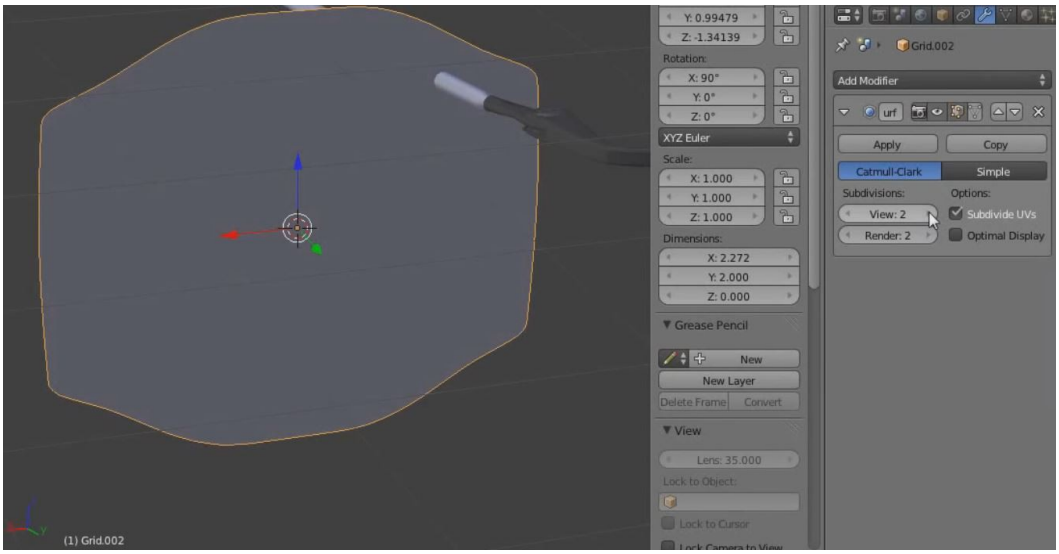
Per curvare il profilo della Grid, possiamo utilizzare (in Edit Mode) il Proportional Editing (shortcut: O), il cui raggio d'azione va regolato subito dopo aver premuto il tasto della trasformazione, ad esempio G o S, con la rotellina del mouse; in particolare, per avvicinare verticalmente i vertici agli angoli possiamo selezionarli con una selezione multipla e, con il Proportional Editing attivo, premere S Z, perché lo scaling sull'asse Z (quello verticale globale) avrà l'effetto di avvicinarli solo verticalmente.




*Utilizzo del Proportional Editing*

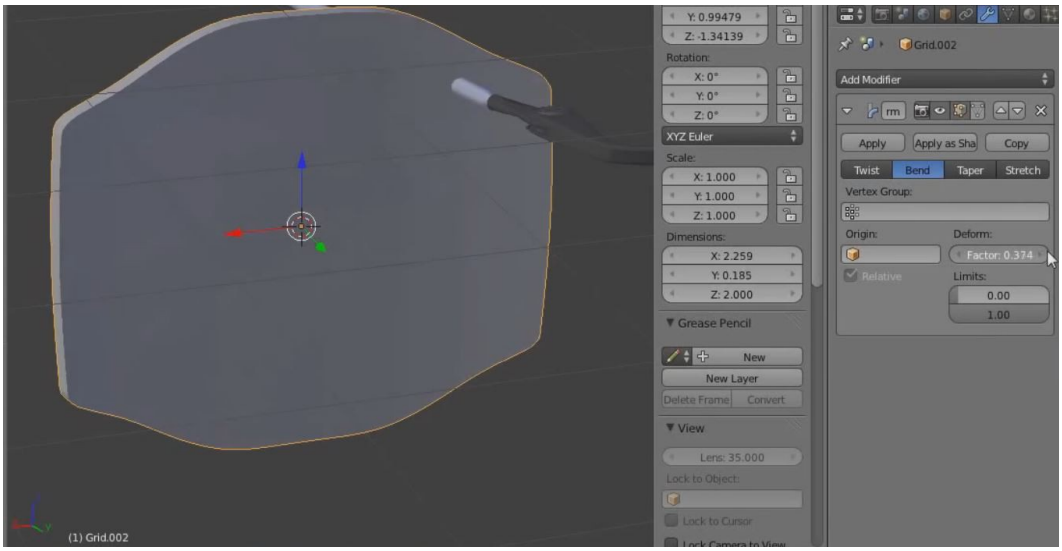
Possiamo quindi procedere con le distorsioni per seguire il profilo della lente.

L'importante, alla fine, è fare un buon numero di suddivisioni o, ancora meglio, utilizzare Subdivision Surface, che leviga l'oggetto e non crea "davvero" i vertici, almeno fino a quando non viene applicato definitivamente alla mesh (piccola nota: per mantenere fermi i bordi della Grid, selezionare gli spigoli del perimetro e impostare Mean Crease a 1 nella Transform Window).




*Utilizzo di Subsurf*

Al termine, estrudiamo o comunque creiamo spessore, ad esempio con il Modifier *Solidify*, quindi pieghiamo (ad esempio con Simple Deform in modalità Bend) e impostiamo lo Shading *Smooth*.




*Operazioni finali: Solidify, Bend, Smooth*

*  *  *

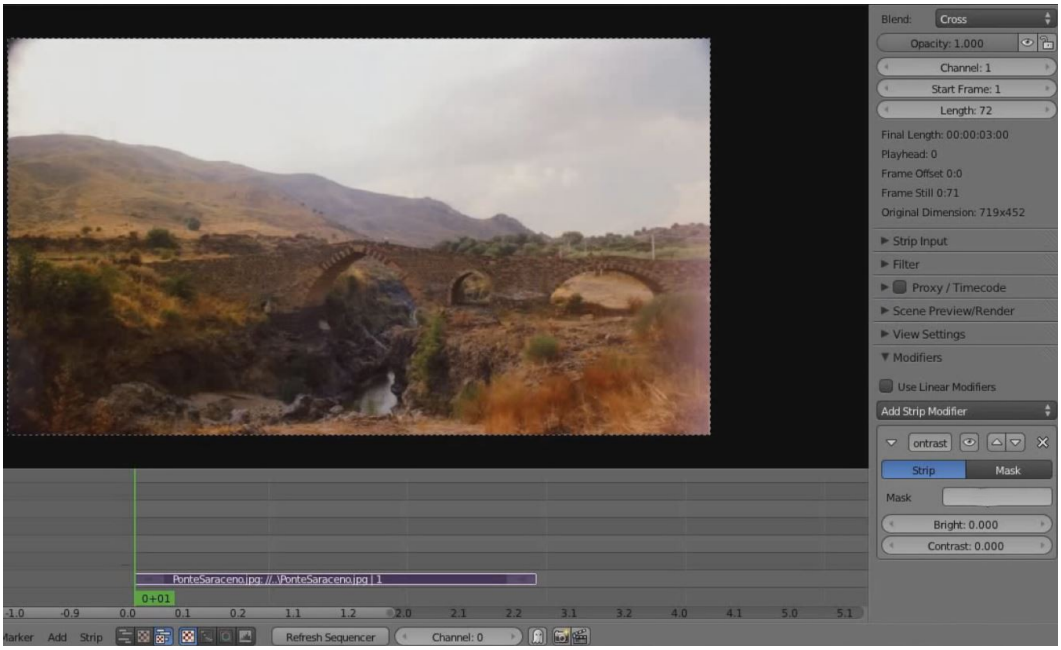
# Filtri di fotoritocco e post-produzione animabili nel VSE

Questo più che un tutorial è un piccolo tips & tricks sui filtri di correzione colore, tonalità, contrasto, eccetera, messi a disposizione dal Video Sequence Editor di Blender 3D 2.64.
È infatti possibile utilizzare Blender anche per gestire una fase di post-produzione o ritocco di un'immagine o filmato, senza dover passare da una scena 3D o dal Compositing dei Nodes.

Anticipo che questo tutorial è di livello medio-avanzato, visto che è data per scontata la conoscenza di certi effetti di compositing, del VSE e del sistema di animazione con keyframes e IPO di Blender.
Per questo esempio sto utilizzando un'immagine (ho aggiunto una striscia *Image* al Video Sequence Editor), ma va da sé che è possibile provare queste tecniche su interi filmati, importandoli con *Add – Movie* anziché *Add – Image*, come ho fatto io per importare l'immagine, nell'header della finestra.

*L'immagine che utilizzeremo per i nostri esempi, nel VSE*

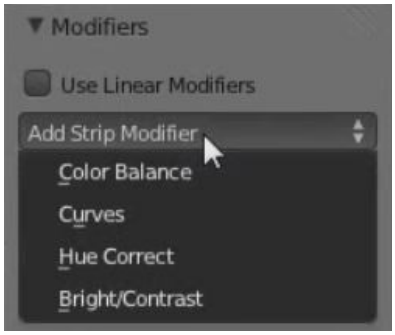
Assicuriamoci di aver selezionato la striscia da modificare, visto che i filtri fanno riferimento, appunto, all'elemento selezionato, quindi premiamo N mentre il cursore del mouse è sulla finestra per aprire la Transform Window della stessa e cerchiamo, in basso, la sezione Modifiers.




*La Transform Window e i modificatori per la Clip selezionata*

Il menù *Add Strip Modifier*, letteralmente "aggiungi modificatore" (per la striscia selezionata), mette a disposizione quattro voci:

- *Brightness / Contrast*, per regolare luminosità e contrasto;
- *Hue Correct*, per variare saturazione, tonalità e intensità per i vari canali-colore;
- *Curves*, per agire sulle curve RGB e composita, come avviene in molti programmi di fotoritocco, come Gimp o Photoshop;
- infine, *Color Balance*, per agire su ombre, mezzitoni e alte luci, definendo per ciascun canale la tonalità e l'intensità.




*Le voci del menù "Add Strip Modifier"*

Da notare che gli effetti non sono esclusivi: per ciascuna strip possiamo aggiungere, ad esempio, sia *Curves* che *Color Balance*, e agire liberamente nelle due finestre.

In questo brevissimo tutorial non mi dilungherò sul significato delle varie voci, per le quali rimando piuttosto ai tutorial sul Nodes Compositing e al relativo ebook, visto che comunque questi strumenti sono disponibili anche come Nodes e quindi sono stati trattati nelle risorse prima citate, oppure a qualche tutorial sui programmi di fotoritocco, come Gimp o Photoshop, perché anche lì abbiamo ad

esempio le *Curves*, luminosità e contrasto, eccetera... solo un appunto finale: come quasi tutte le voci presenti in Blender, anche i parametri di questi modificatori sono animabili.

È possibile, quindi, inserire due keyframes per il campo *Brightness* in due momenti diversi per animare la variazione di luminosità nel tempo, oppure cliccare sul valore di intensità del Gamma o del Lift di *Color Balance* e registrare i valori inserendo dei keyframes con I (mentre il cursore del mouse si trova sul parametro per il quale registrare il keyframe), per variare l'intensità di mezzitoni o ombre nel corso del tempo, e questo anche su immagini statiche, non solo videoclip.

*Registrare dei Keyframes per animare i filtri / modificatori (in questo caso, il box del colore di un Color Balance)*

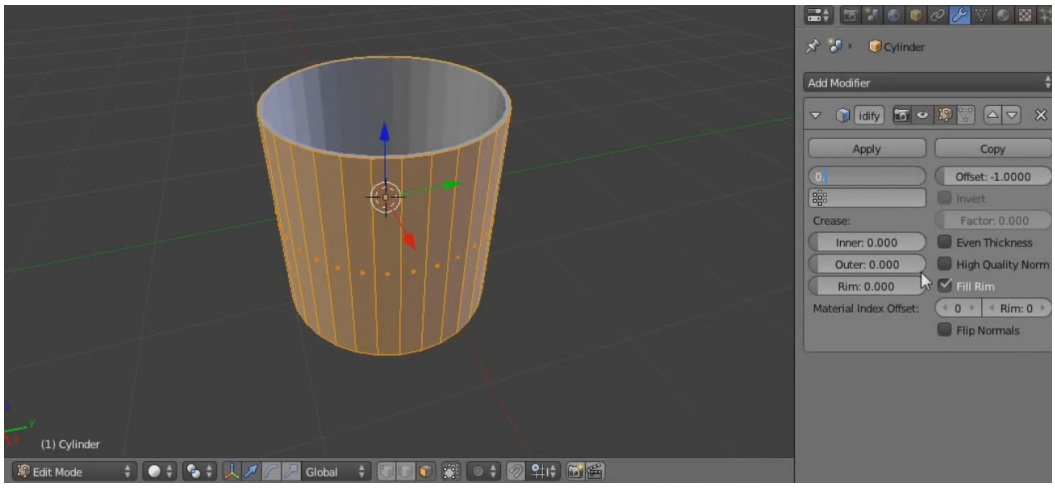
Vi invito quindi a sperimentare con i vari filtri e a provare vari effetti su immagini, filmati o rendering della scena corrente (infatti è possibile scegliere anche *Add – Scene* dall'header per effettuare della post-produzione "al volo", con i Modifiers da associare alla strip del rendering della scena 3D corrente).

*  *  *

# Cell Fracture Tool e animazione fisica col Game Engine

In questo tutorial vedremo come realizzare l'animazione di un oggetto che cade e si rompe mediante il plugin *Cell Fracture Tool* e la simulazione fisica del Game Engine in Blender 3D.
Un avviso: si tratta di un tutorial di difficoltà media, non di base, dove do per scontate operazioni di modellazione e UV-Mapping; nella prima parte del tutorial, che riguarda la modellazione della tazza e il Texturing, andrò quindi velocemente, per passare poi all'argomento vero e proprio del tutorial.

Sto modellando la tazza partendo da una primitiva cilindro privata, mediante selezione in modalità *Face Select* e cancellazione Faces, della faccia superiore. Fornisco un po' di spessore mediante il modificatore *Solidify*, che applico subito, quindi in Edit Mode aggiungo delle suddivisioni sulla superficie laterale esterna dell'oggetto con CTRL R (Ring Cut) e Invio.

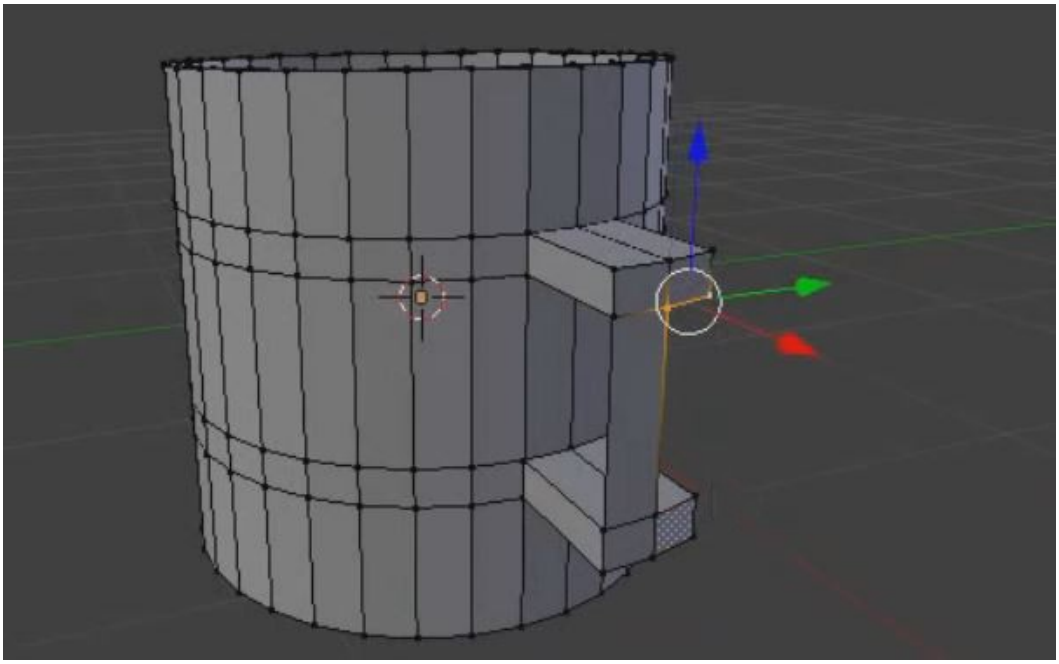



*Modellazione della tazza (l'oggetto utilizzato per la simulazione, in questo tutorial)*

Con CTRL R, il primo click, con l'anello color magenta, serve a definire la direzione di taglio, quindi con il mouse posso spostare “l'anello” da creare e col secondo click confermare l'operazione.

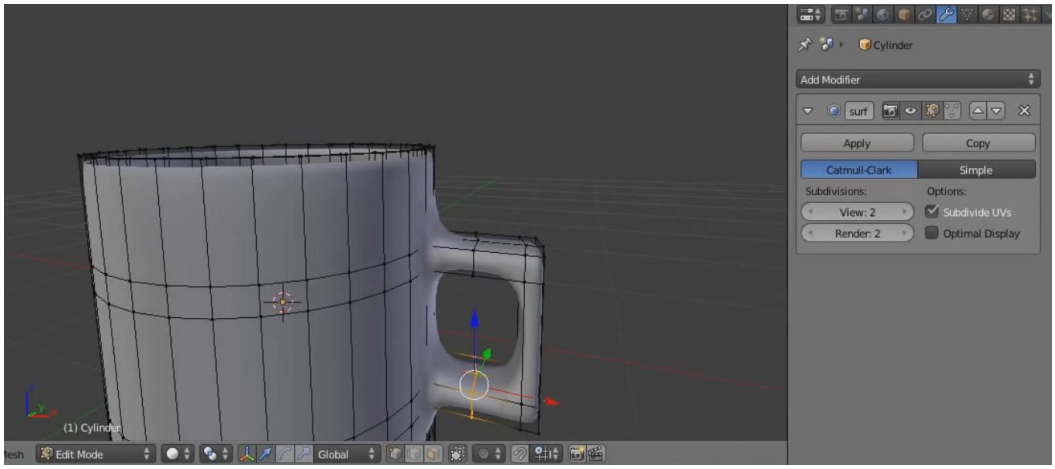
Qui sto andando a occhio, nel definire il manico, da creare mediante estrusioni, creazioni di spigoli e facce con *F*, quindi non stupitevi delle strane proporzioni!

Per creare spigoli e facce di collegamento, seleziono 4 vertici e premo F, quindi estrudo; il problema è che dietro, nella parte interna del manico, non vengono create certe facce, per cui sto tornando in quel punto per inserirle manualmente con il metodo appena descritto.



*Creazione delle facce con F (Fill, "riempi") per il manico*

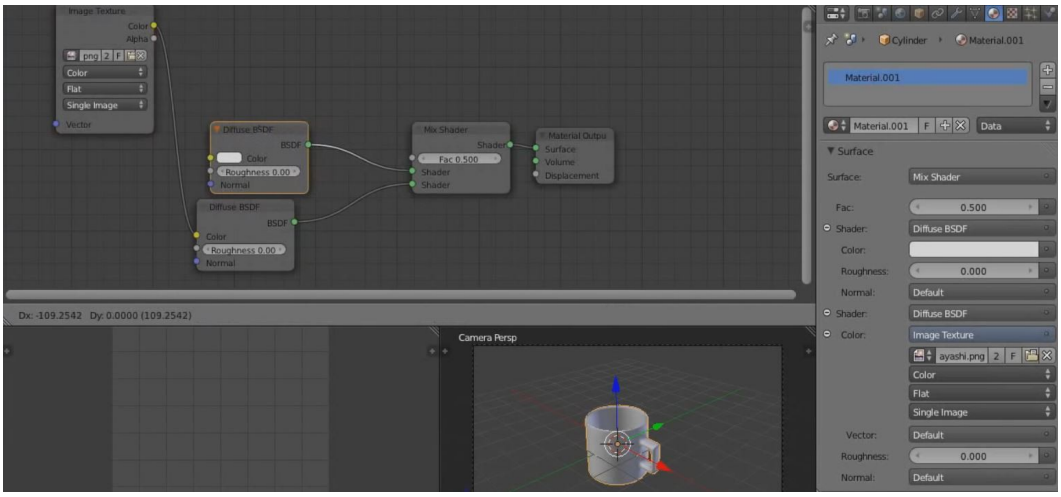
Per smussare il tutto applico Subdivision Surface e *Set Smooth*, nella Tool Shelf, quindi aggiungo, dove necessario, altri “anelli”, con CTRL R, per ammorbidire la mesh in certi punti.




*Levigare con Set Smooth e Subsurf*

Con la mesh base ci siamo: passiamo ora all'UV Mapping di una texture immagine.

L'immagine che sto per utilizzare è di tipo PNG con canale Alpha, ossia è trasparente ovunque tranne dove c'è la scritta, come vedremo, quindi in Cycles creiamo un Material *Mix*, con due Nodi *Diffuse* da mixare, solo che per il secondo utilizzeremo il file immagine e prenderemo l'uscita Alpha dell'Image come Factor di miscelazione del *Mix*, per definire le zone di trasparenza (se siete completamente a digiuno su Cycles, vi consiglio il relativo ebook).



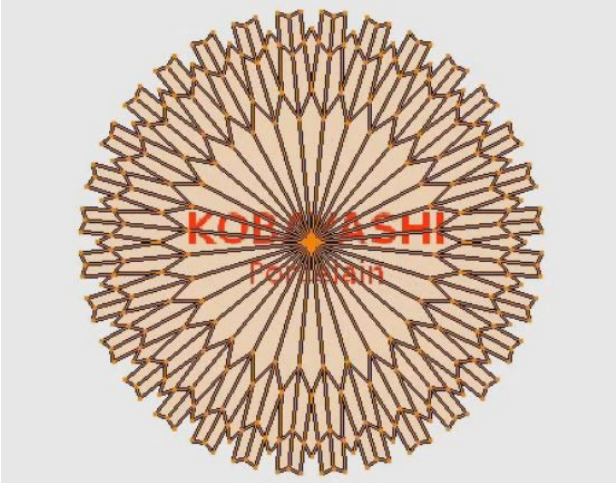

*Definizione del Material (texturizzato) per la tazza*

Possiamo cambiare il colore di base della tazza cambiando quello del primo Nodo *Diffuse*, attivo nelle zone trasparenti dell'oggetto; il Material c'è, ma l'immagine non è stata mappata (con mappatura UV) sull'oggetto, quindi dobbiamo occuparci proprio di questo aspetto.

Selezionando la mesh e andando in Edit Mode, con tutti i vertici selezionati, potremo fare l'Unwrap (ad esempio, con *U* e *Project From View*, per la prima fase) e posizionare tutti i vertici della mesh in una parte bianca dell'immagine (Kobayashi.png), in quanto quella zona, che qui vediamo bianca, è in realtà trasparente, come detto all'inizio.

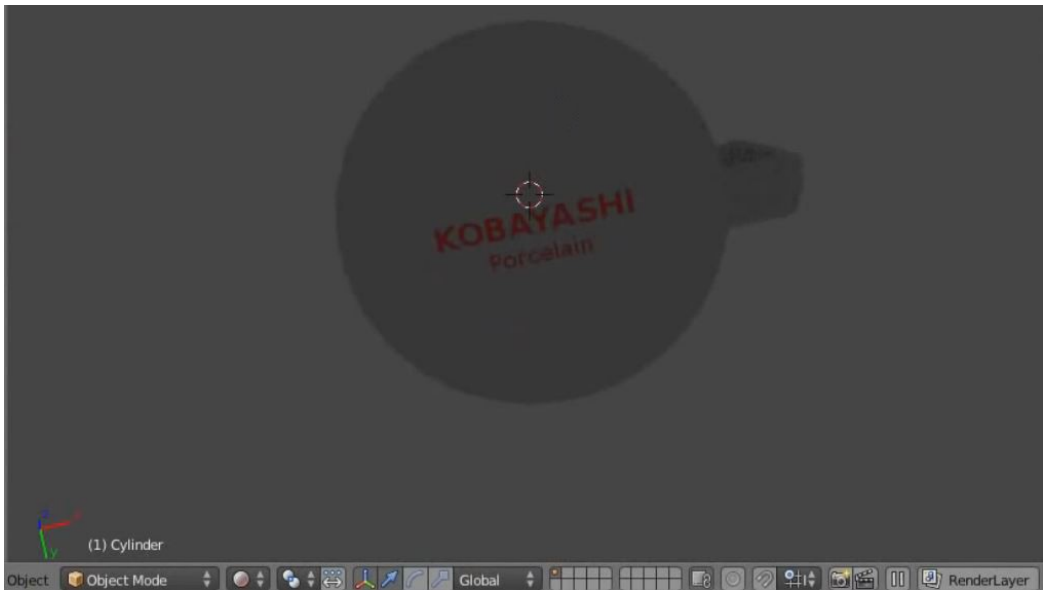
*Primo Unwrap della tazza*

Dopo, dovremo selezionare la parte inferiore della tazza, possibilmente da una vista Bottom Ortho per fare un unwrap veloce con *Project from View*, e mappare questa parte in modo tale da includere la scritta... beh, io sto mettendo la scritta nella parte inferiore, ma ovviamente nessuno vi vieta di mettere immagini, loghi o altro sui lati, sul manico, e così via.




*Unwrap e mappatura della parte inferiore della tazza*

Con le operazioni preliminari abbiamo finito: diamo un'occhiata in modalità *Rendered* per osservare il risultato ed eventualmente cambiamo illuminazione, colore World con una *Sky Textures*, e così via, quantomeno per non avere una tazza grigia su sfondo grigio, come all'inizio!

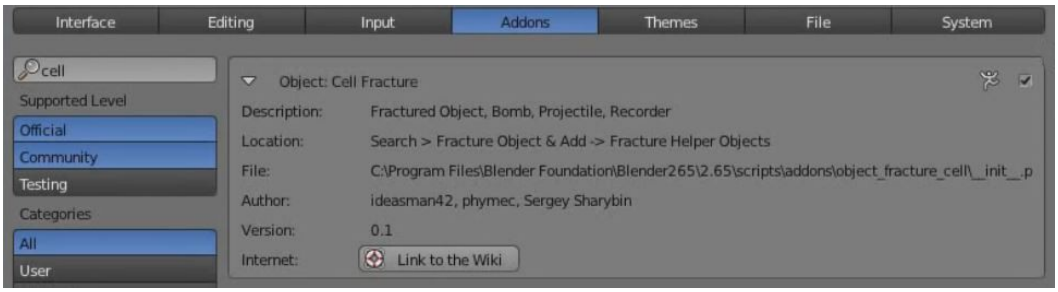



*Anteprima Rendered*

É giunto finalmente il momento di utilizzare il *Cell Fracture Tool*!

L'Add-on è presente di serie nella versione 2.65 di Blender ma, di default, è disattivato, per cui per prima cosa dobbiamo trovarlo nella scheda Add-ons della finestra User Preferences e attivarlo.

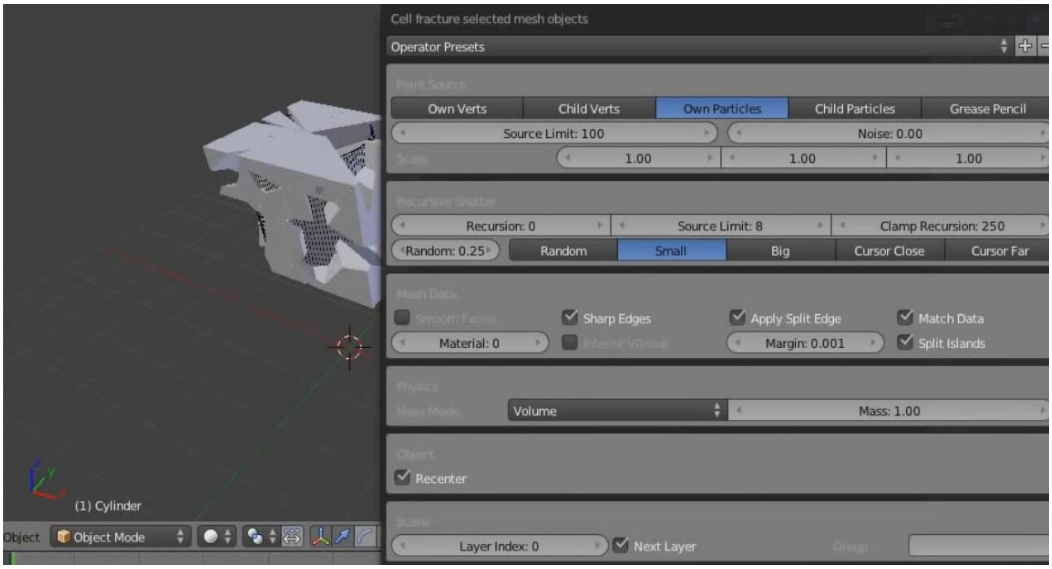
La descrizione del plugin ci dice che *Cell Fracture* va ricercato nella casella di ricerca del programma e attivato da lì, non ci sono quindi voci nella Properties Window o nella 3D View.




*Attivazione dell'Add-On Cell Fracture Tool*

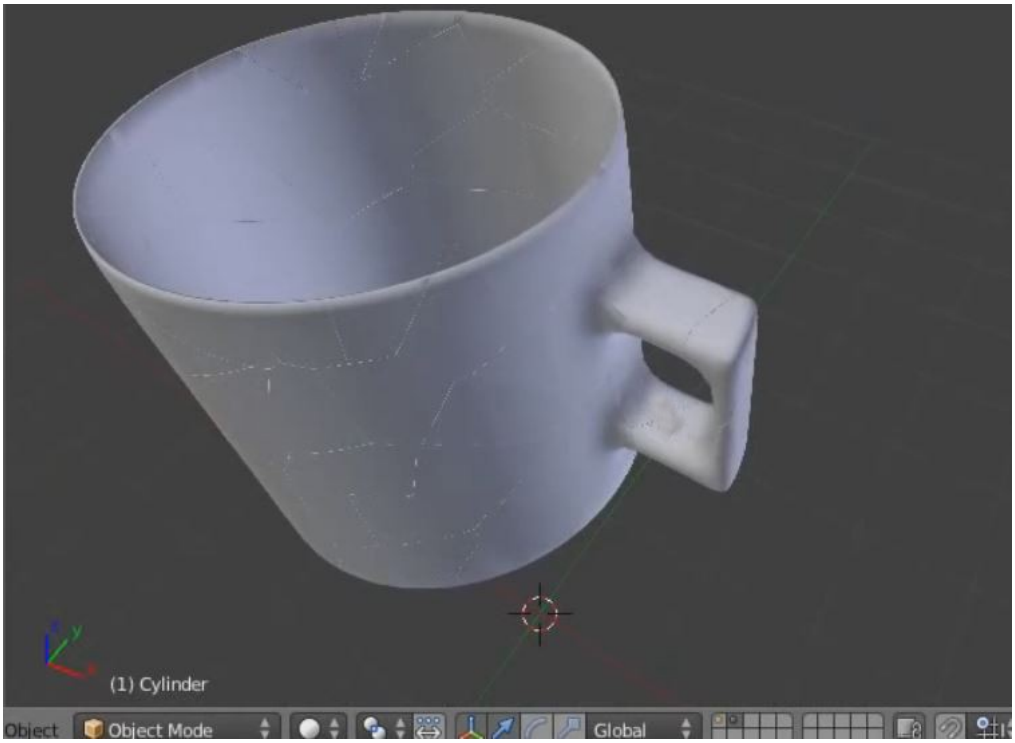
L'Add-on presenta molti comandi, ma in genere le impostazioni di default vanno più che bene, per cui qui mi sto limitando a premere “Ok” per avviare le operazioni con i parametri originali.

Nella 3D View, possiamo vedere Cell Fracture all'opera. A seconda della complessità della mesh, questa operazione può richiedere poco o molto tempo.



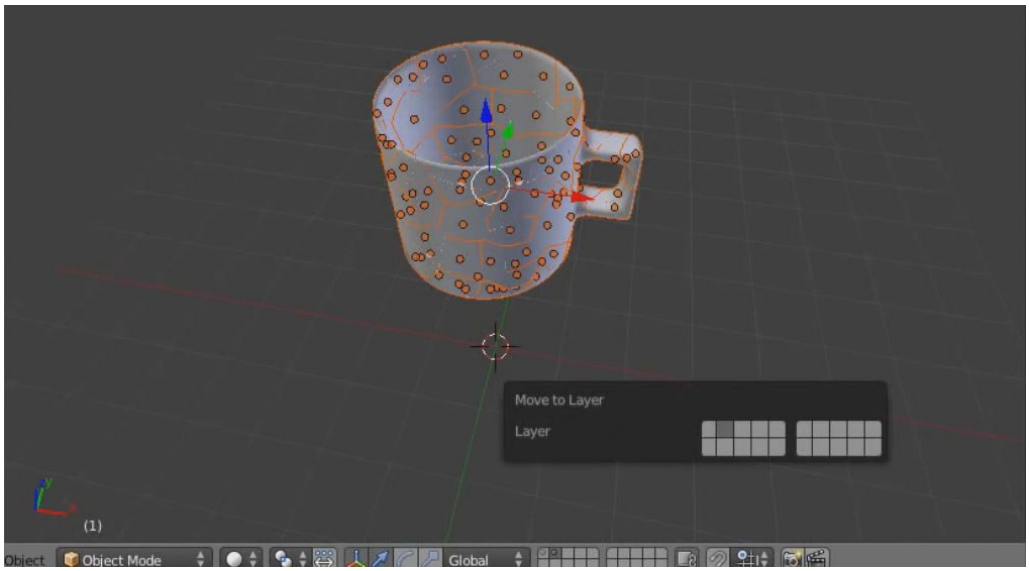

*Il Cell Fracture Tool all'opera*

Al termine dell'operazione, comunque, avremo di nuovo la mesh originale, quindi apparentemente non è successo nulla; in realtà, le nuove mesh si trovano nel secondo Layer di scena.



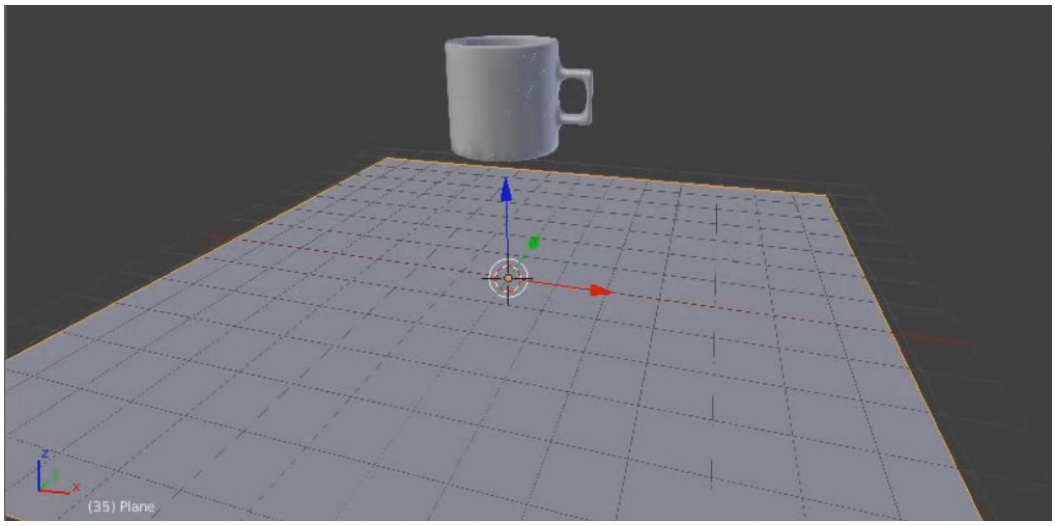

*Le nuove mesh nel secondo Layer della scena*

Dal momento che il risultato ottenuto va benissimo, sto cancellando dal primo Layer l'oggetto originale e sto passando i nuovi elementi dal secondo al primo Layer (selezionandoli tutti e premendo *M* per far apparire il menù di cambio Layer, quindi click sulla prima casella).




*Spostare gli oggetti da un Layer ad un altro con M*

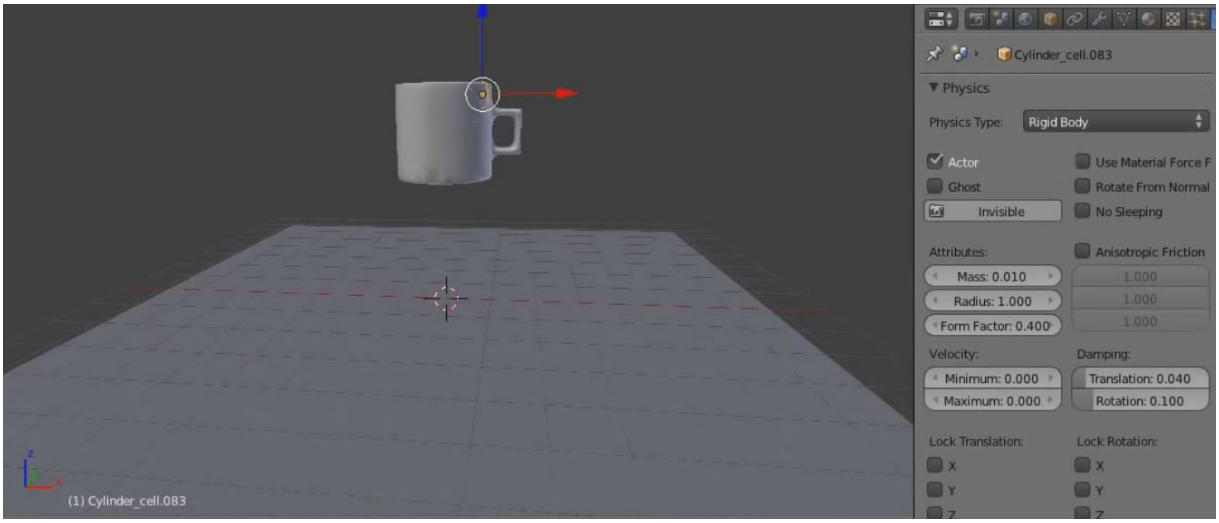
Adesso sto aggiungendo nella scena un Plane, come “pavimento” per rompere la tazza, tuttavia lanciando un'anteprima dell'animazione notiamo che in realtà... non succede niente.




*Con il Plane appena inserito... non succede niente*

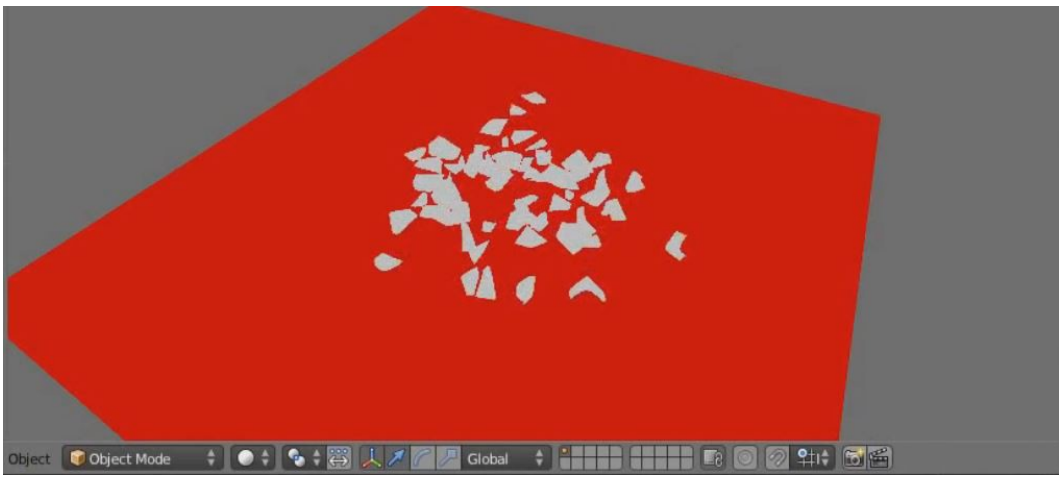
Passiamo al motore Blender Game, dal selettore “Engine” in alto al centro, nell'editor Info.

Per realizzare l'animazione, infatti, utilizzeremo il Game Engine di Blender, che contiene un simulatore fisico che fa al caso nostro e che ci consente di registrare l'animazione ottenuta mediante simulazione fisica e riprodurla poi negli altri motori di rendering sotto forma di animazione IPO degli oggetti. Il Plane-pavimento sarà di tipo Static, mentre le varie mesh che compongono la tazza potranno essere o di tipo Rigid Body o Dynamic.



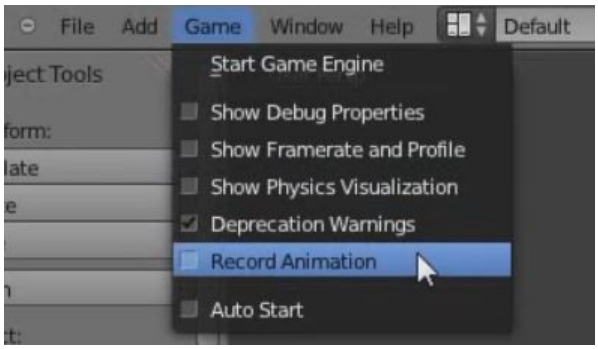

*Impostazioni per Plane e tazza nel Game Engine di Blender*

Avviando ora il gioco con i, vedremo la tazza cadere e rompersi al contatto col Plane.



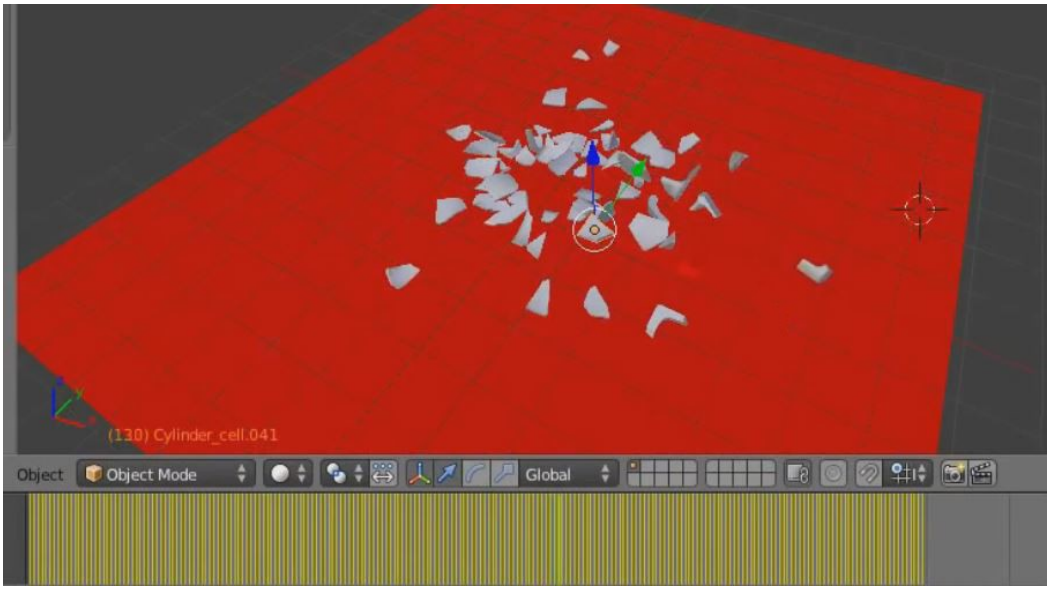

*In modalità esecuzione del "gioco", la tazza cade e si rompe*

Per registrare questa animazione come IPO, usciamo dalla modalità gioco con ESC, apriamo il menù Game e selezioniamo la voce "*Record Animation*" (registra animazione, appunto).




*Record Animation*

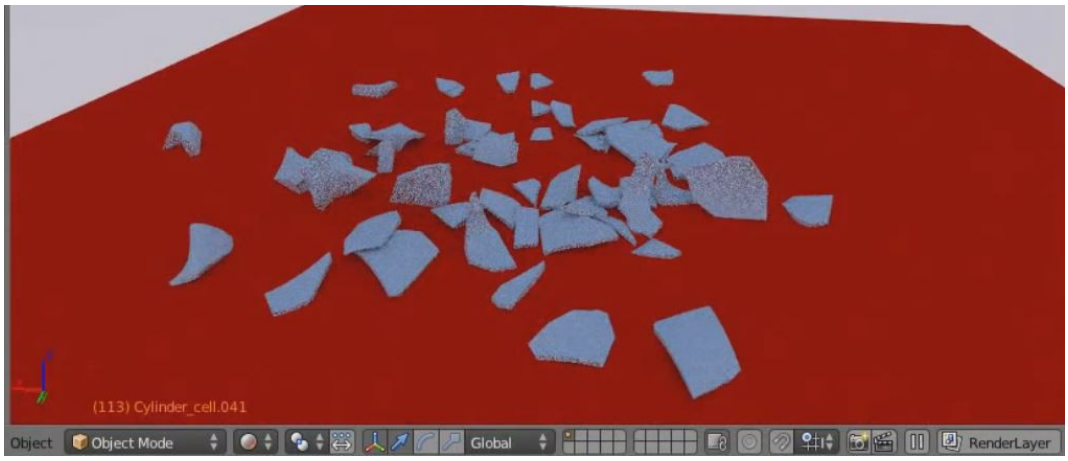
Torniamo in modalità gioco con *P*, per avviare la simulazione e, quando questa termina, usciamo con ESC. Selezionando le mesh della tazza, vedremo l'animazione IPO nella Timeline.




*L'animazione registrata*

Possiamo tornare a Cycles.

Come potete vedere, abbiamo ancora l'animazione IPO degli oggetti e, cosa ancora più interessante, il *Cell Fracture* ha mantenuto i Materials con la mappatura *UV* della Texture immagine per tutti i pezzi, quindi quando realizzeremo l'animazione renderizzata avremo i vari pezzi con la Texture originale, più l'animazione realizzata con il Game Engine, registrata e utilizzata in Cycles.




*Anteprima del risultato finale in Cycles*

… certo, il risultato, nel mio caso, sarebbe stato molto più simpatico con la scritta in bella vista, ma proprio quei cocci sono rimasti poggiati; d'altronde, erano sul fondo! Con decorazioni, scritte o loghi otterrete sicuramente risultati più interessanti.

Ricapitolando: si modella l'oggetto, si usa il *Fracture Tool* per scomporlo, si realizza l'animazione nel Game Engine con *Record Animation*, si torna in Cycles o Blender Render ed ecco il risultato finale. Per questo tutorial, quindi, è tutto!

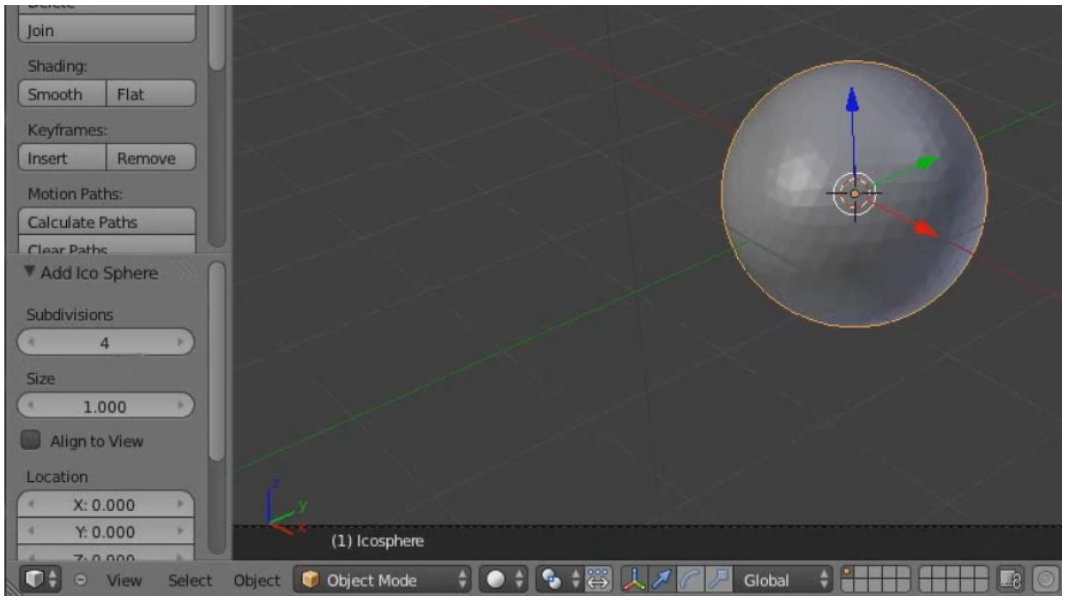
*  *  *

# Come modellare una pallina da golf con Icosphere e Bevel

In questo tutorial di modellazione di base, realizzato con la versione 2.66 di Blender 3D (quindi con modellazione bmesh ed ngons), vedremo come modellare una pallina da golf con pochi semplici passi partendo da una *Icosphere* e alcuni strumenti di selezione e modifica delle geometrie.

Il risultato non sarà magari una pallina da golf regolamentare, corretta al 100%, ma penso che possa andare bene per semplici rendering o videoclip e, in ogni caso, il risultato dipende anche da certi valori, quindi siete liberi di sperimentare altre soluzioni!

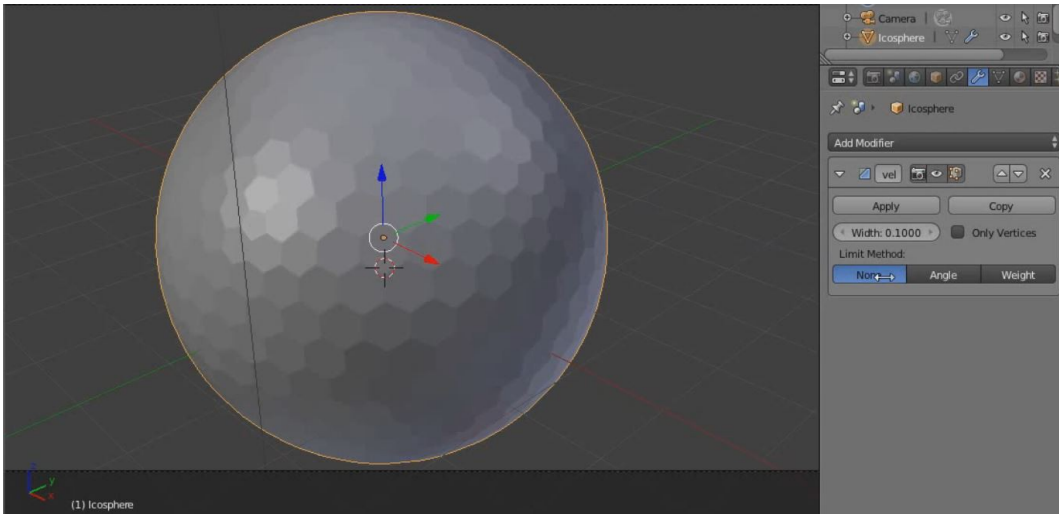
Iniziamo creando una *Icosphere* nella scena, quindi SHIFT A – Mesh - *Icosphere*; senza premere altri tasti, nella Tool Shelf (richiamabile con T, mentre il cursore del mouse è in una 3D View) impostiamo, in basso, 4 suddivisioni.




*Creazione della Icosphere con 4 suddivisioni*

A questo punto apriamo il pannello Modifiers, nella Properties Window, per aggiungere un modificatore Bevel.

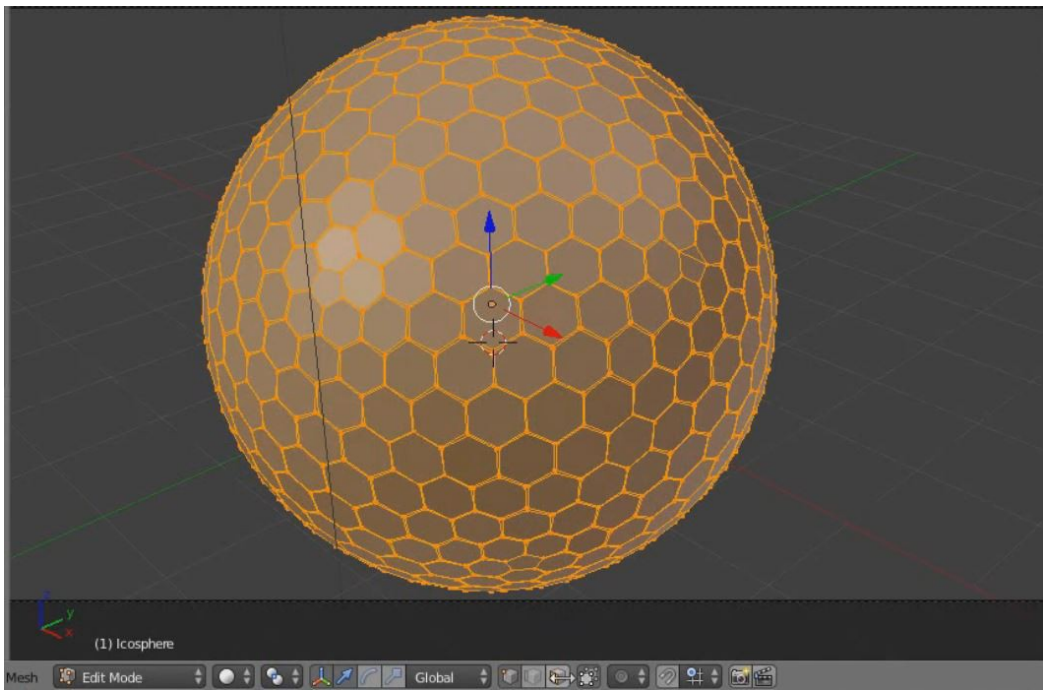
Le impostazioni di default, ossia Method None e Width 0.1, dovrebbero andare più che bene.




*Aggiungere un modificatore Bevel, da applicare con le impostazioni di default*

Applichiamo il modificatore.

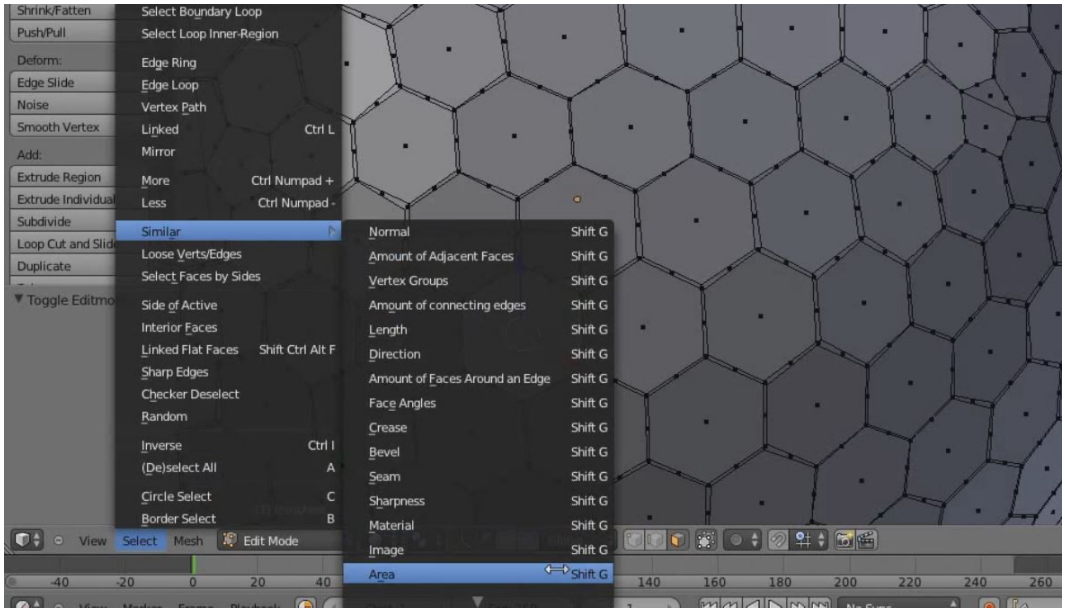
Adesso, andiamo in Edit Mode (con TAB o con l'apposito selettore nella 3D View): notiamo che ci sono delle facce esagonali e, ai bordi, altre piccole facce, che separano gli esagoni.




*L'oggetto in Edit Mode*

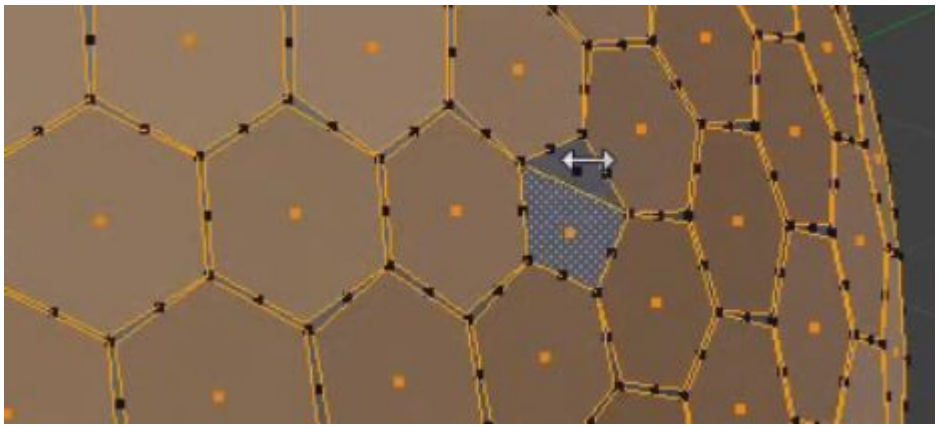
Passiamo in modalità selezione facce mediante l'apposito selettore nell'header, quindi selezioniamo UNA faccia esagonale con un click destro del mouse.

Ecco la magia: nell'header della 3D View, scegliamo *Select – Similar - Area*.

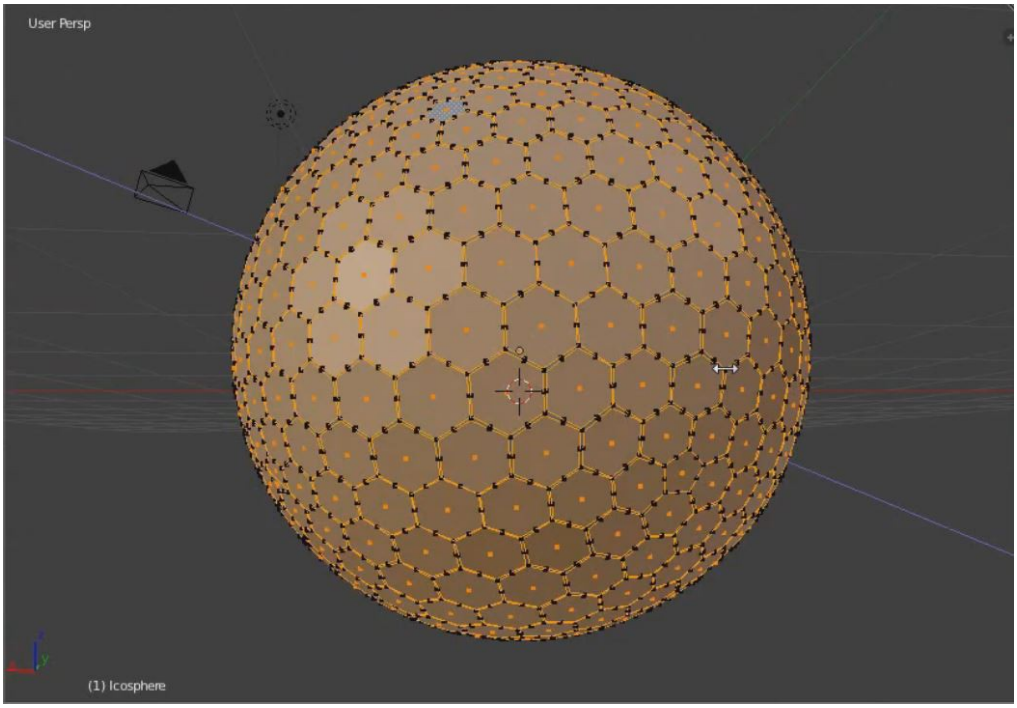


*Select - Similar - Area, con una delle facce esagonali selezionata*

Blender selezionerà per noi tutte le facce esagonali, ignorando quelle dei bordi.
In base alle impostazioni, potrebbero esserci un paio di facce triangolari o comunque non esagonali non selezionate, in giro per la mesh: giriamo intorno all'oggetto, molto velocemente, e se ne troviamo aggiungiamole alla selezione corrente con SHIFT e click destro su tali elementi.

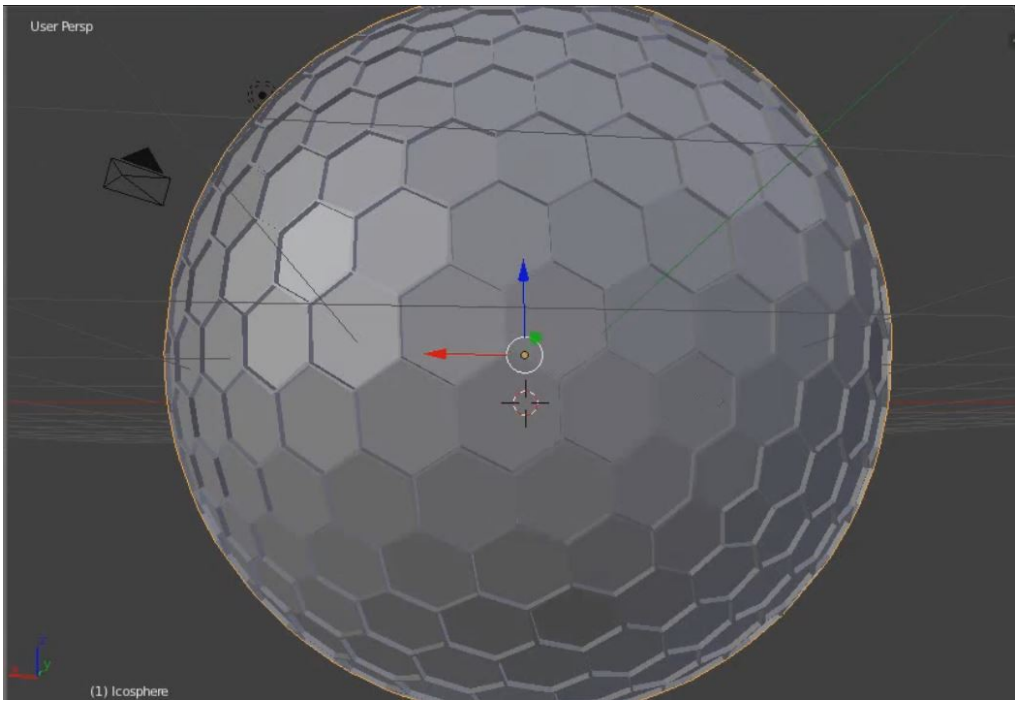

*Selezionare le facce tralasciate da "Select Similar", con SHIFT e click destro*

A questo punto premiamo *E*, per l'estrusione, seguito subito dopo da Invio, per attuare l'operazione "sul posto", senza spostare gli oggetti.



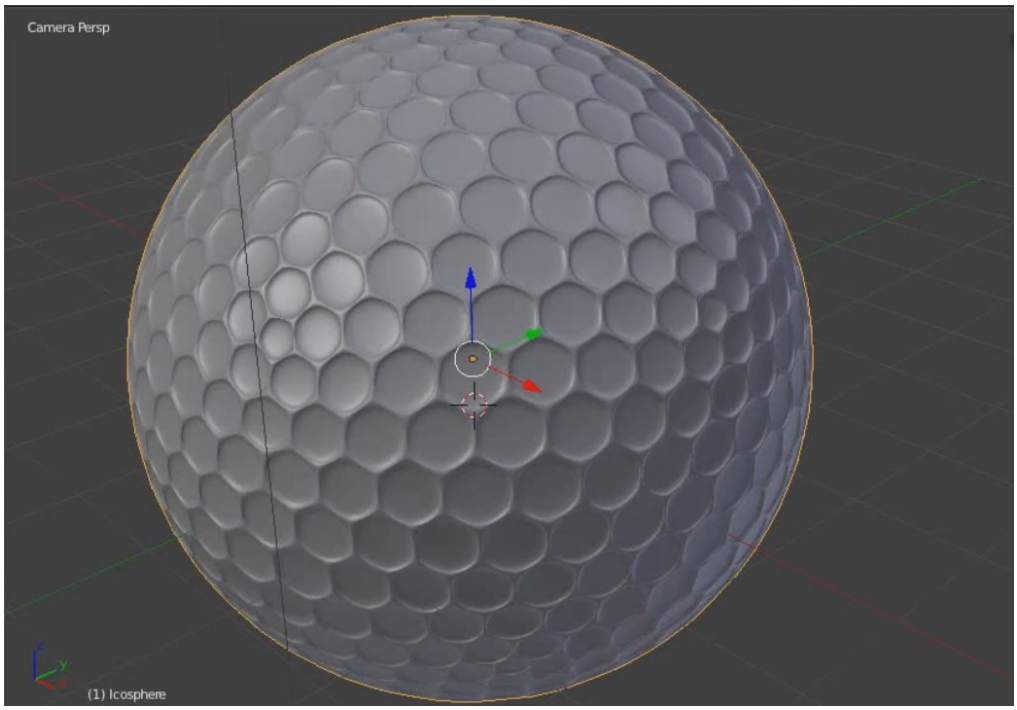

*Estrusione sul posto*

Senza premere altri tasti, quindi con le facce dell'estrusione selezionate, premiamo *S*, per Scaling, quindi digitiamo ad esempio 0.99 e premiamo Invio per confermare lo scaling, scalando le facce e rientrandole un po' nella mesh (scalando su tutta la selezione le avvicineremo, di fatto facendole rientrare rispetto alla superficie di partenza).



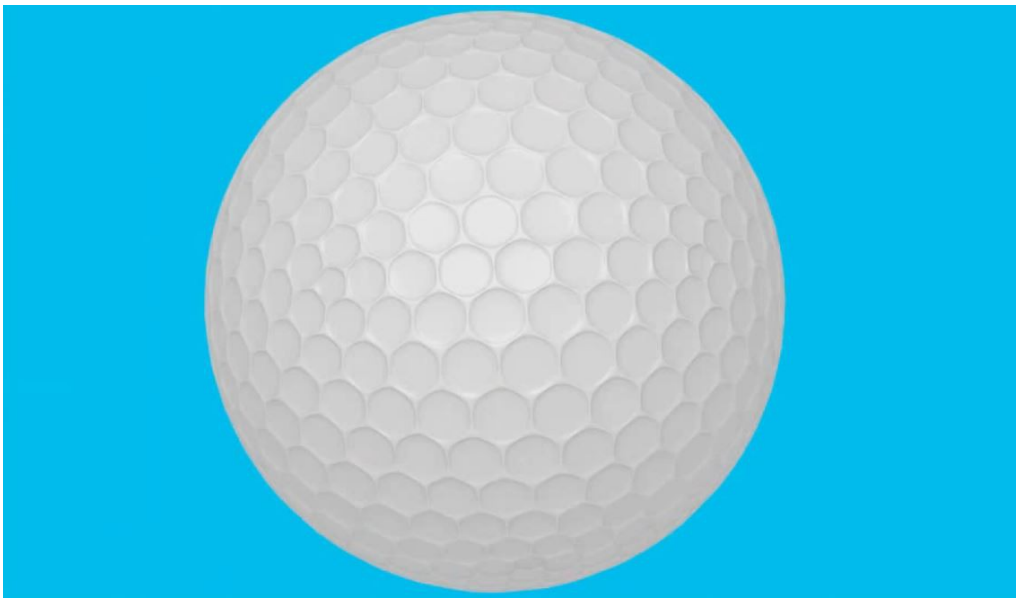

*Scaling dell'intera selezione, portando le facce estruse all'interno della mesh*

Adesso, per arrotondare e levigare le fossette, due operazioni: aggiungiamo e applichiamo un Modifier Subdivision Surface con 2 suddivisioni e, nella Tool Shelf, clicchiamo su Shading *Smooth*, per impostare un'ombreggiatura dolce, non sfaccettata.




*Levigare l'oggetto con Subsurf, qui prima di applicare anche Shading Smooth*

Ricapitolando: Icosphere, Bevel per ottenere le facce esagonali, *Select Similar Area*, Estrusione, Scale, Subdivision, Smooth, ed ecco la nostra pallina da golf!



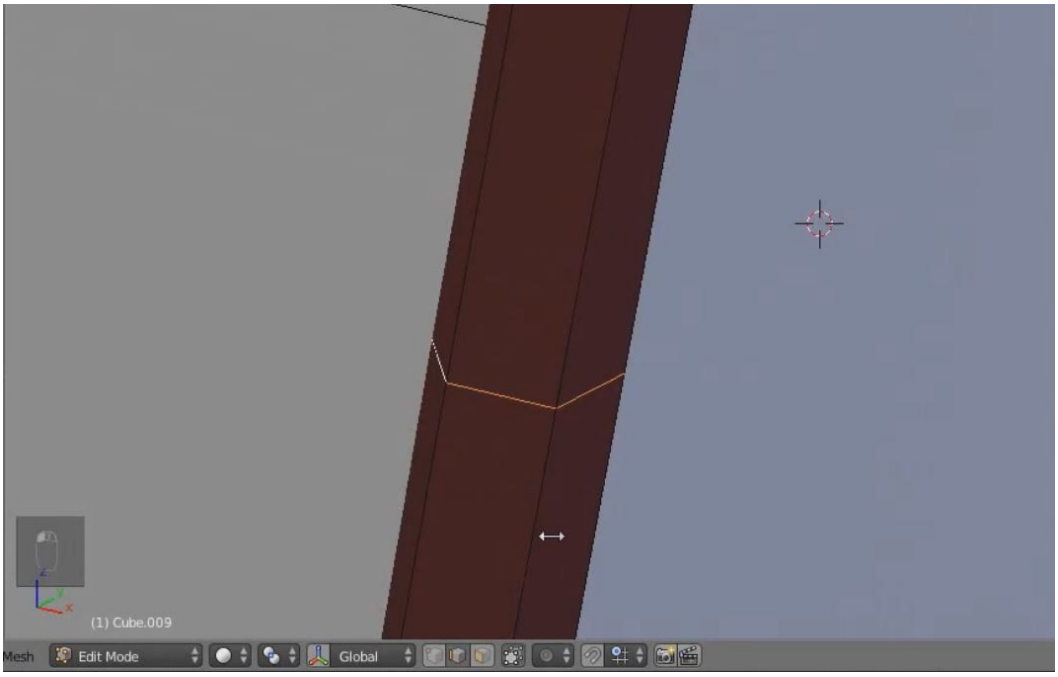
*Il risultato finale*

*  *  *

# Tips & Tricks: Edge Slide

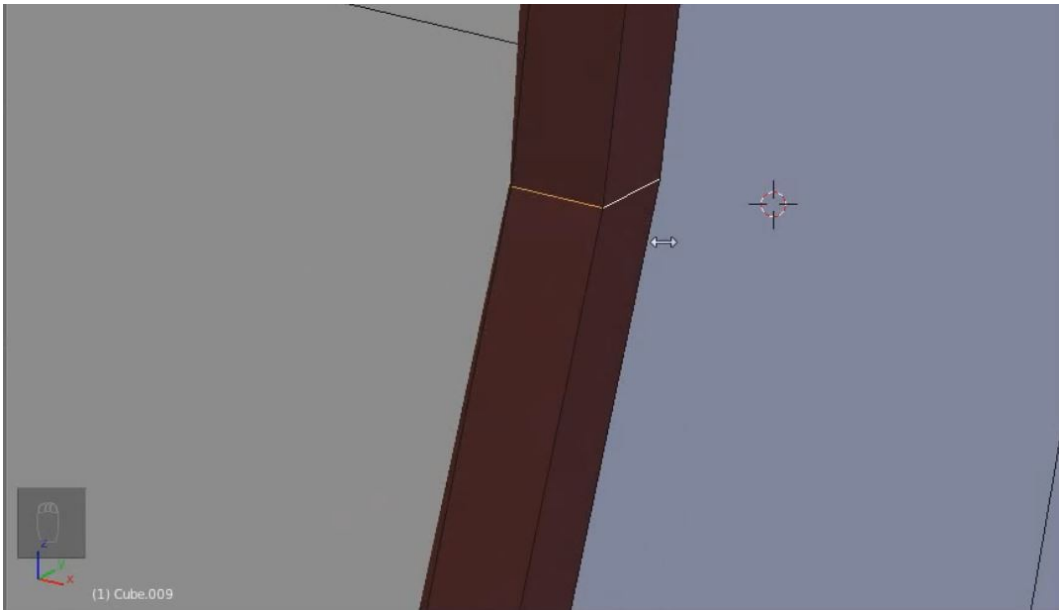
In questo brevissimo tutorial parlerò di uno strumento di modellazione di base di Blender 3D: *Edge Slide*, che serve a far scivolare (letteralmente) degli spigoli su un oggetto.
Si tratta di uno strumento semplice ma di importanza fondamentale per risolvere facilmente certi problemi di modellazione; vediamo a che serve con un esempio pratico.

Nello scenario visibile nell'immagine seguente abbiamo dei segmenti che rappresentano un "taglio" su un'asse di legno inclinata e, per qualche motivo, vogliamo traslare questi spigoli sulla mesh, portandoli più in alto.



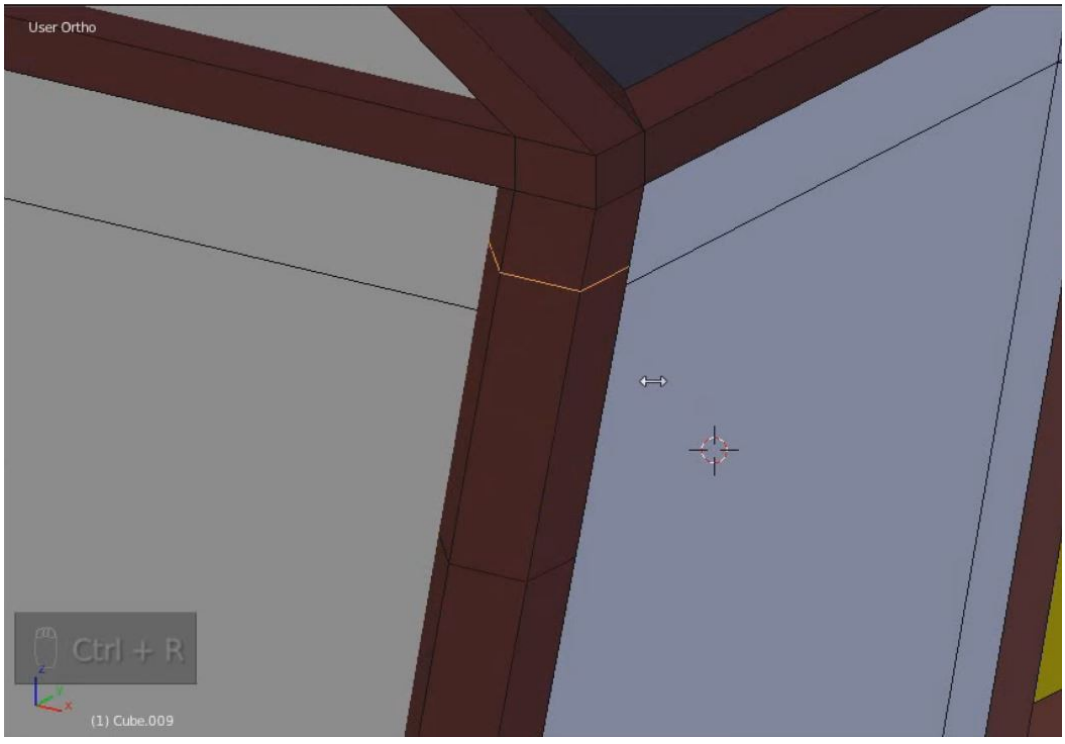

*La situazione di partenza*

La soluzione che viene subito in mente consiste nel selezionare l'anello di spigoli (cosa che si fa con ALT e click destro su uno spigolo dell'anello, in modalità selezione spigoli) e trascinare con *G*, cercando di mantenere la direzione dell'oggetto; questa però è un'operazione che si rivela poco semplice e che può introdurre brutte modifiche nella mesh.



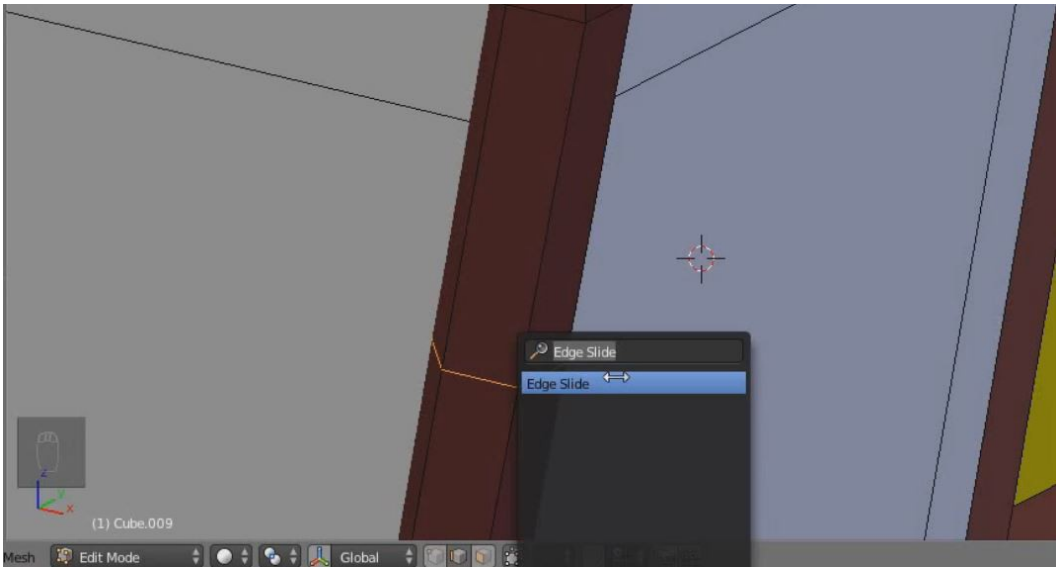

*La soluzione (errata) con G (Grab, il trascinamento)*

Una possibile via di fuga consiste nell'utilizzo di Ring Cut, ma questa soluzione, applicabile qui, non è disponibile in altre situazioni e, comunque, introduce nuovi elementi, aumentando inutilmente la complessità della mesh senza risolvere il problema principale, ossia spostare quell'insieme di spigoli!



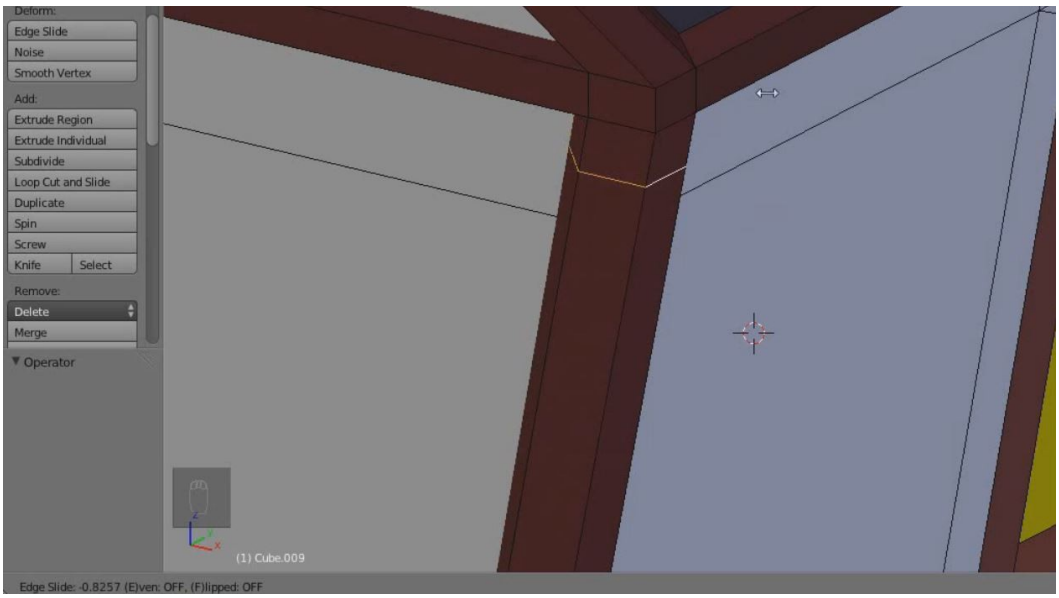

*Altra soluzione errata: inserire tagli con Ring Cut (CTRL R)*

Ecco la soluzione: selezionato l'insieme di spigoli da muovere, premiamo la barra spazio per aprire la Tool Box nella vista 3D, quindi digitiamo *Edge* (spigolo) e scegliamo *Edge Slide*, per far "scivolare" gli spigoli.




*Edge Slide (richiamato, in questo caso, dalla Tool Box)*

Nella 3D View potremo ora traslare la selezione sull'oggetto muovendo il mouse, per poi fare click col tasto sinistro nella posizione desiderata, seguendo la forma originale della mesh, senza introdurre alcuna deformazione.




*Utilizzo di Edge Slide*

Questo è chiaramente solo uno dei possibili esempi di utilizzo di *Edge Slide* che, quindi, serve a far "scivolare" dei segmenti su un oggetto seguendo l'inclinazione di quest'ultimo, senza cioè introdurre delle deformazioni che si avrebbero inevitabilmente tentando di traslare la selezione con *G* e trascinamento.

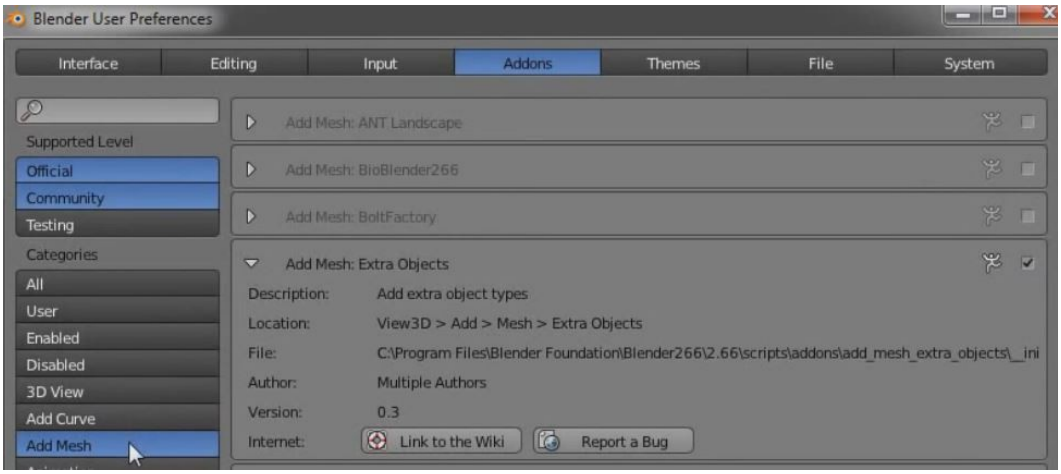
Come detto all'inizio, si tratta di una funzionalità semplice, ma davvero molto utile in parecchie situazioni... questo è solo un esempio.

*  *  *

# Math Function e Array: modellare portici

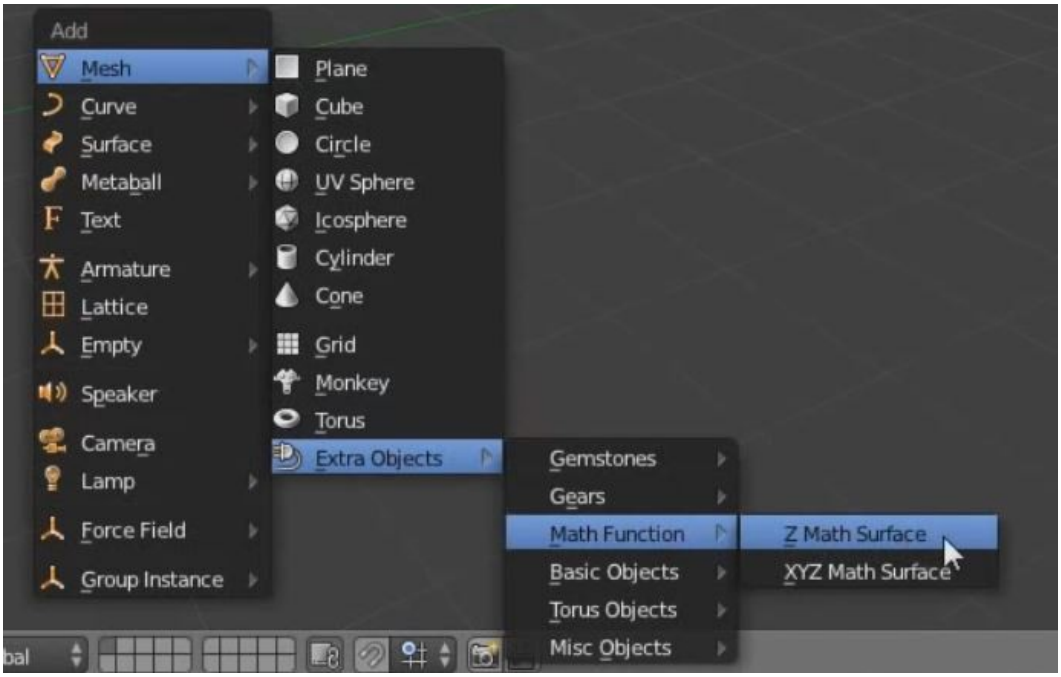
In questo brevissimo tips & tricks di modellazione in Blender 3D parleremo di un plugin poco noto ma molto interessante: la sezione *Math Function* di *Extra Objects*, che consente di modellare superfici partendo da equazioni da inserire direttamente nella Tool Shelf.
Questa funzionalità può tornare utile o per modellare graficamente delle equazioni oppure, come vedremo in questo tutorial, per modellare oggetti "non banali" come una volta e, da lì, un porticato, con un certo grado di controllo sui parametri dell'elemento.

Per prima cosa, attiviamo l'add-on *Extra Objects*, presente di serie con Blender 3D (qui nella versione 2.66 del programma) ma disattivato, per cui apriamo la finestra User Preferences dal menù File, scegliamo il pannello Add-Ons e, nella sezione *Add Mesh*, attiviamo la voce *Extra Objects* selezionando la relativa checkbox.



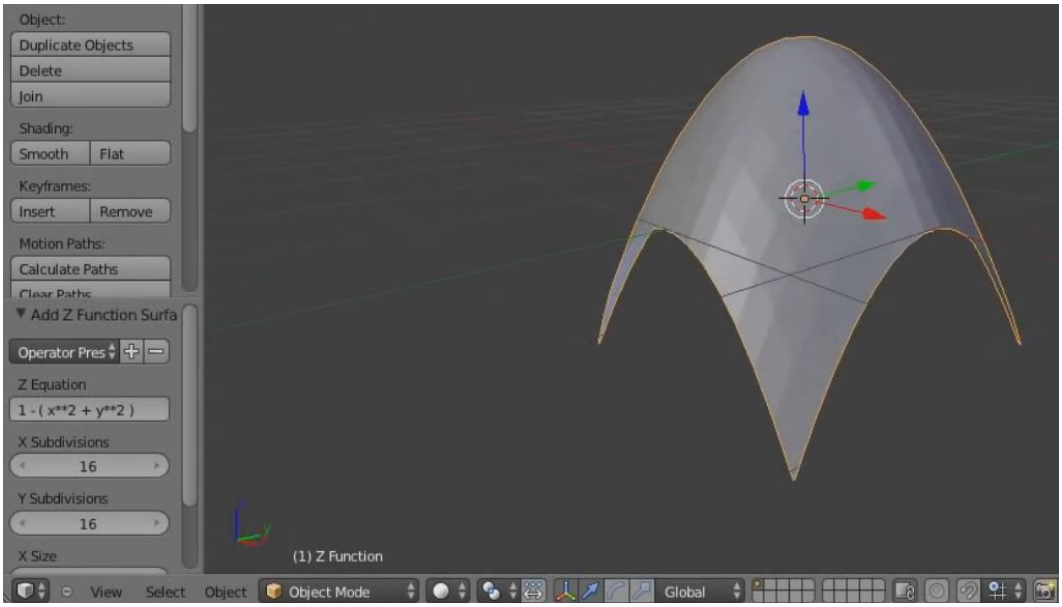
*Attivare l'Add-On Extra Objects*

Adesso, di nuovo nella 3D View, apriamo il menù Add e scegliamo *Mesh* | *Extra Objects* | *Math Functions* | *Z Surface*, cioè la modellazione di una superficie dove il valore Z, quello verticale, viene calcolato punto per punto in base ai valori dati con un'equazione, con x e y che fanno quindi riferimento alle coordinate locali dei vari vertici dell'oggetto.




*Le nuove voci Extra Objects in Add Mesh*

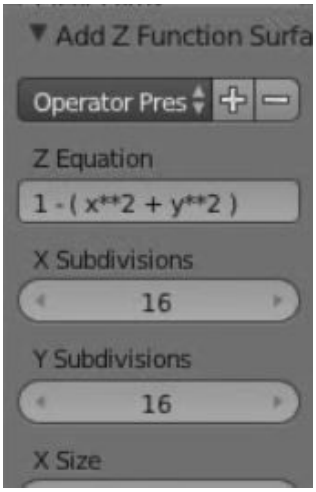
Aggiungiamo un oggetto di base, con l'equazione data di default, e torniamo su questo concetto per spiegarlo meglio, soprattutto per quanto riguarda la frase "coordinate locali".




*Oggetto di base aggiunto con Z Surface*

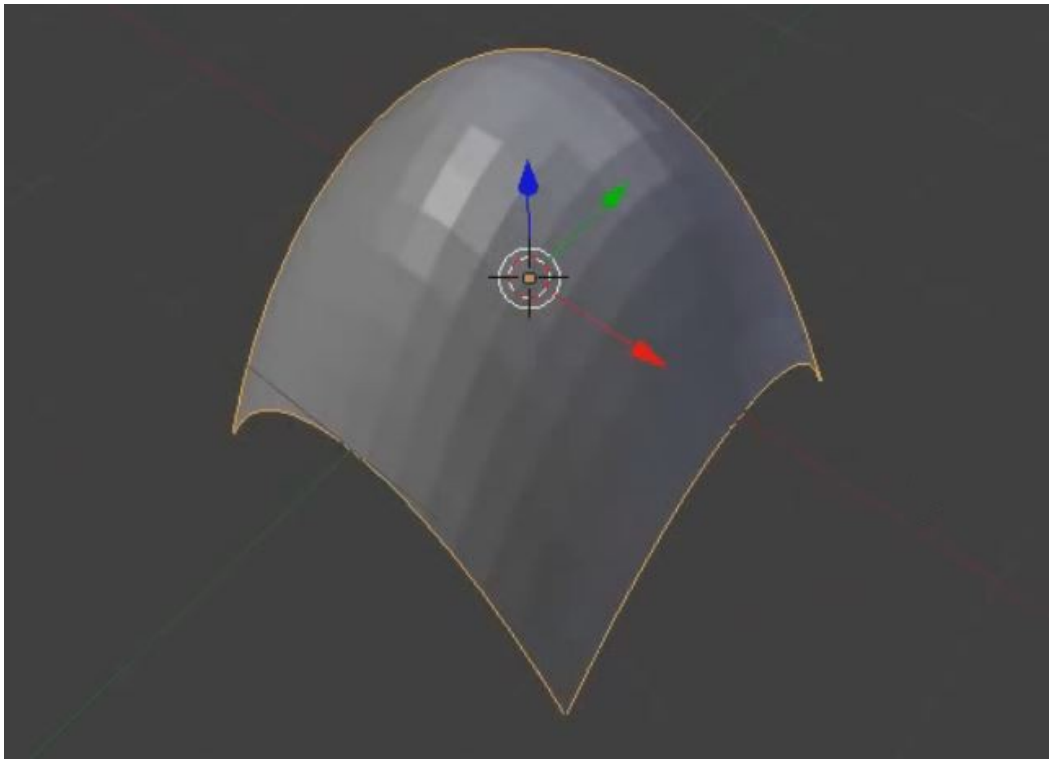
L'equazione di default è, come visibile nella Tool Shelf in basso a sinistra non appena si crea una Z Surface, *z = 1 – (x^2 + y^2)*; da notare che noi non vediamo *z* = ma in realtà il contenuto del campo Z Equation è proprio la parte destra dell'equazione. Altro appunto: l'elevamento a potenza è fatto digitando due volte asterisco, quindi *x**2* e *y**2* indicano appunto x al quadrato e y al quadrato.

Potete quindi iniziare a digitare equazioni ed equazioni per vedere i vari risultati; qui, prima di procedere con l'esempio della volta a crociera, parliamo un po' delle coordinate usate per calcolare i valori.



*L'equazione della forma, nella Tool Shelf*

Come potete vedere, l'Origin dell'oggetto è al centro dello stesso, almeno sul piano XY, il che significa che l'oggetto viene visto con un "range" di valori locali che vanno da -1 a 1 sia per la dimensione X che per quella Y, in riferimento al sistema locale, con l'origine (punto XY 0,0) che coincide con l'Origin dell'elemento, il suo centro, quindi andremo da -1 a 1 su X e da -1 a 1 su Y.

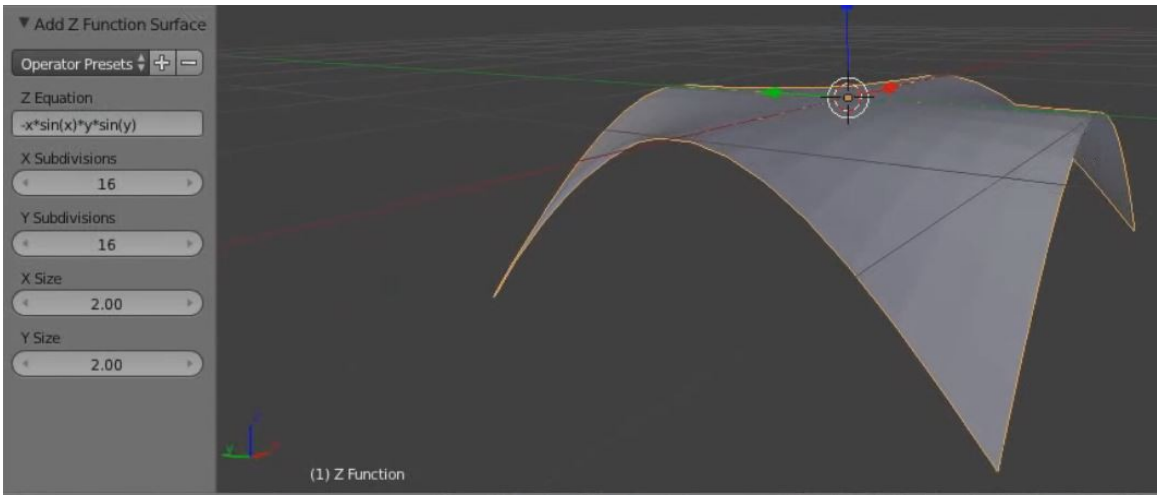

*L'Origin dell'oggetto*

In base a questo sistema di coordinate, punto per punto viene calcolato il valore Z, quello verticale, sul quale posizionare i vertici...

… già, ma quanti vertici? L'oggetto è una mesh, non una Curve o una Surface, quindi è composto da un certo numero di vertici. Tale numero va specificato nella Tool Shelf al momento della creazione della mesh, nei campi *X Subdivisions* e *Y Subdivisions*, che di default hanno valore 16, quindi si tratta di una griglia 16x16, dove punto per punto viene calcolato il valore Z e il relativo vertice viene spostato di conseguenza.

Questi valori vanno quindi specificati quando la sezione *Math Function* della Tool Shelf è ancora attiva, quindi prima di spostare o comunque trasformare in qualsiasi modo l'oggetto.

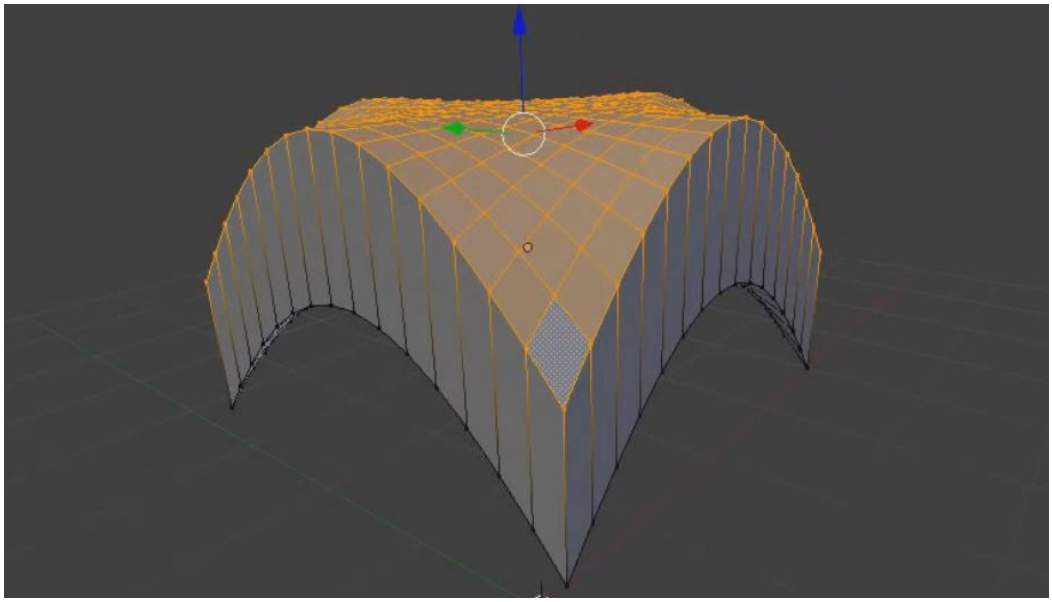
Come detto prima, potete provare tutte le equazioni che volete; qui, ne sto digitando una in particolare, nel campo *Z Equation: -x*sin(x)*y*sin(y)*, attenzione alle parentesi per specificare x e y come argomenti della funzione sin, la funzione seno.



*La funzione utilizzata in questo tutorial*

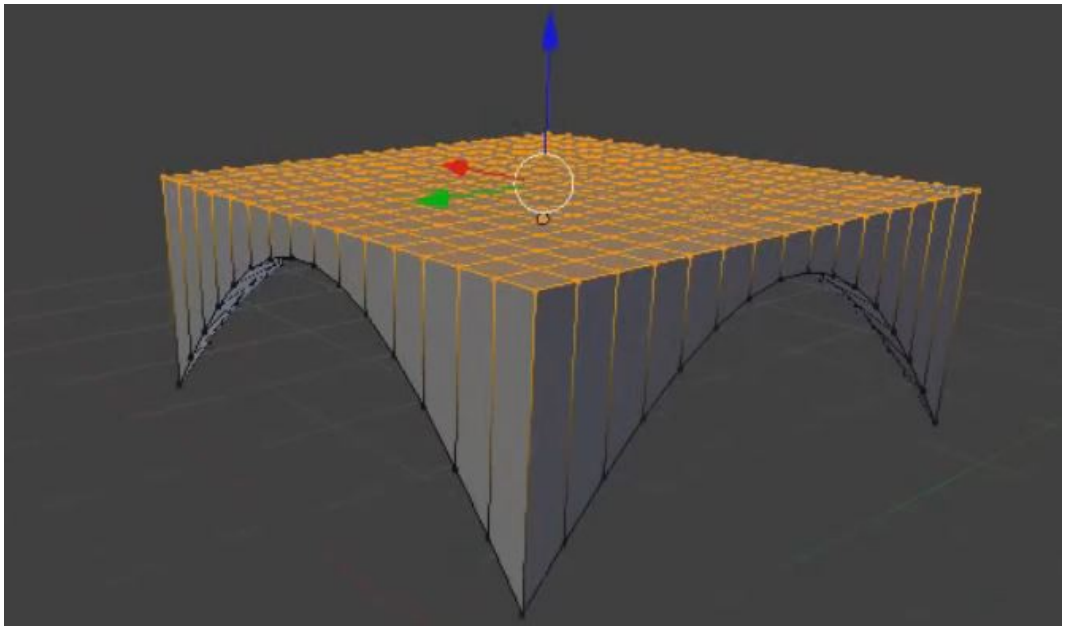
Neanche a dirlo, utilizzando la funzione Array è possibile sviluppare in lunghezza questa mesh, dopo aver fatto alcune modifiche, per ottenere un porticato.

Partiamo da queste "opportune modifiche": passiamo in Edit Mode, selezioniamo tutto e premiamo *E*, estrudendo verso l'alto.



*Estrusione verso l'alto, in Edit Mode*

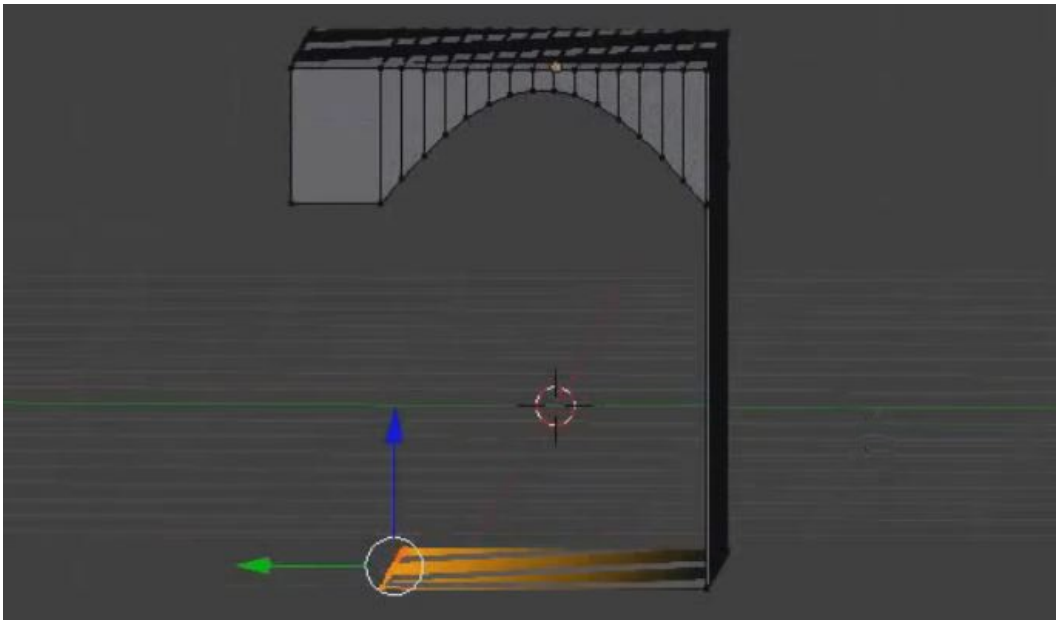
Con i vertici derivanti dall'estrusione ancora selezionati, premiamo *S Z 0*, per realizzare uno scaling a 0 sull'asse verticale, cosa che avrà l'effetto di allineare verticalmente tutti i vertici della selezione (quando si fa uno scaling vincolato a un asse, infatti, ad esempio con S Z, si possono allontanare o avvicinare gli elementi lungo quell'asse; facendo uno scaling a 0, si eliminano del tutto le differenze, col risultato di appiattire la selezione su quell'asse).



*Appiattire la selezione con uno scaling a 0 sull'asse verticale*

Adesso deselezioniamo tutto, passiamo in modalità selezione facce e selezioniamo le facce di un lato della mesh, quindi estrudiamole nel verso perpendicolare alle facce selezionate, in modo da ottenere uno spessore: tornerà utile per aggiungere le colonne.

Deselezionate tutto, passate su un lato ed estrudete, in modo da creare lo spessore per l'arco dal lato aperto e per la colonna.



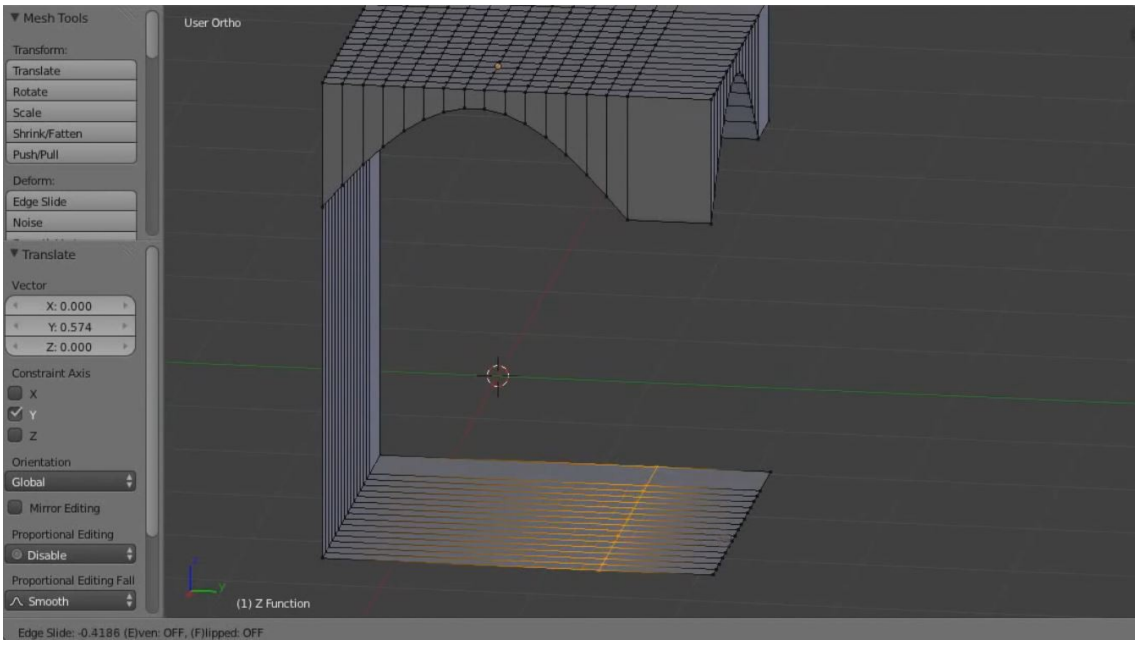
*Altre fasi della modellazione del "porticato"*

Ora sta a voi decidere se fare parete e pavimento con mesh distinte (come vi consiglio di fare, se non altro per gestire meglio finestre, porte e textures varie) o se procedere con un'unica mesh monoblocco da replicare mediante Array.

Se volete procedere con il primo metodo, allora applicate subito Array (come faremo tra un attimo), altrimenti fate così: deselezionate tutto, passate in modo selezione vertice, selezionate i vertici inferiori di un lato della mesh, estrudeteli verso il basso e, neanche a dirlo, digitate *S Z 0* seguito da Invio per allinearli, quindi spostateli fino a “terra” e, una volta lì, estrudete ma trascinate, questa volta, orizzontalmente, per realizzare il pavimento.

Le colonne potete realizzarle anche a parte, dopo, ma visto che ci siamo, se non altro per un concept e per mostrare quanto può essere utile Array, facciamo così: definiamo una semplice base e una

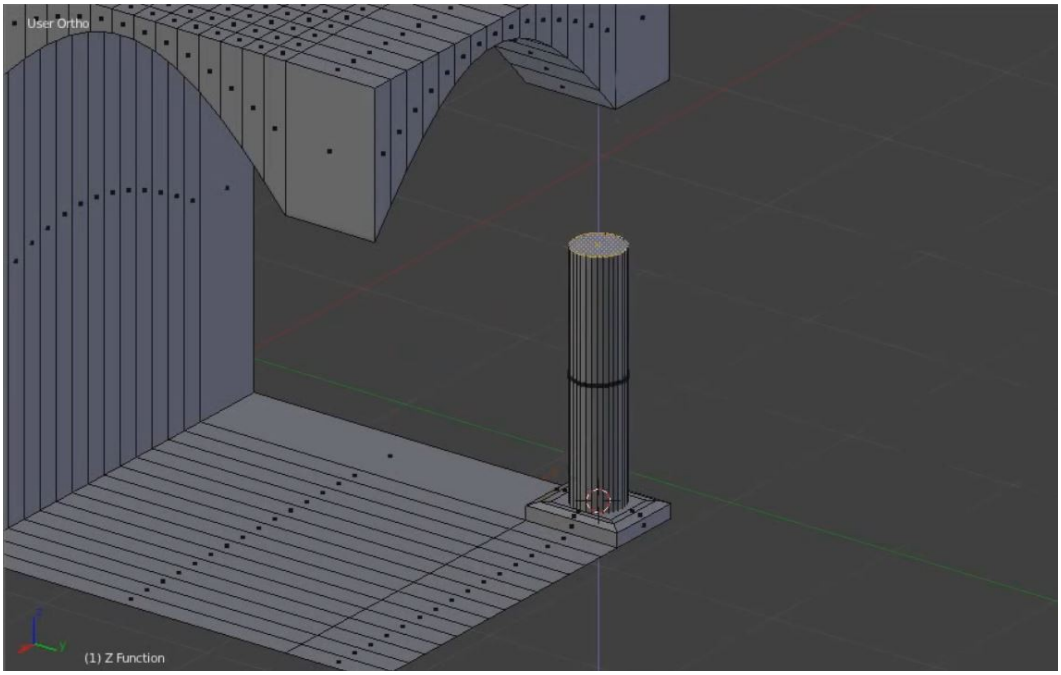
colonna una volta, partendo dal pavimento: con Ring Cut, shortcut CTRL R, sto inserendo una "fetta" nel pavimento per identificare la base della colonna (in alternativa, potete fare una doppia estrusione alla base; in questa fase, inoltre, vi consiglio di utilizzare una vista frontale ortogonale, oppure digitare valori numerici o ancora utilizzare la calamita ed altri strumenti di Snap, per un allineamento di precisione, se necessario).



*Inserimento di una "fetta" nel pavimento per creare una colonna*

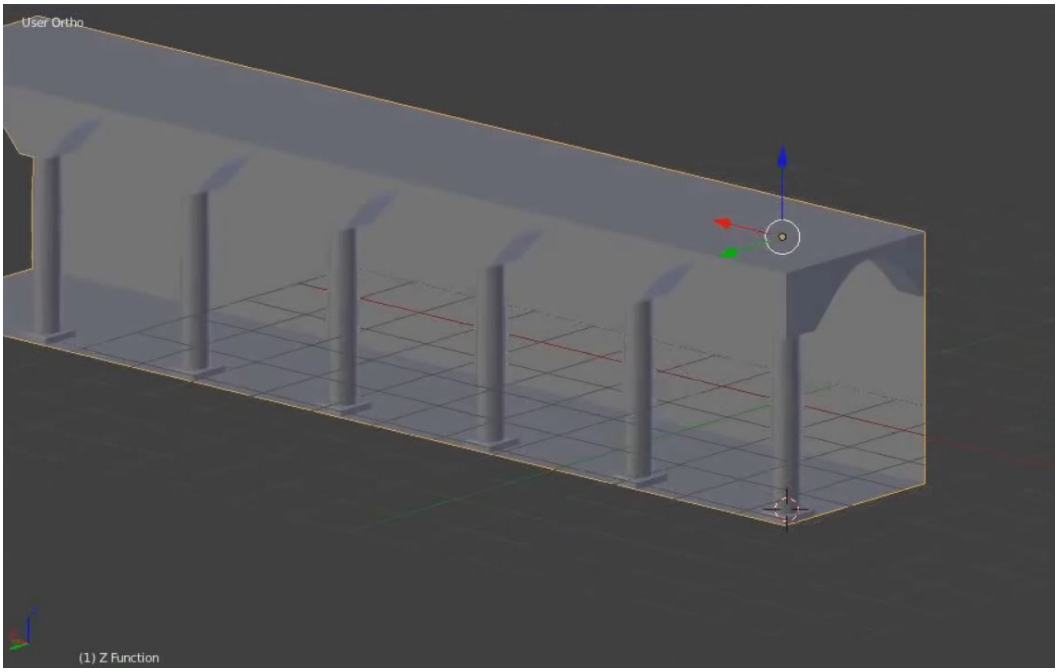
Adesso selezioniamo, magari in modalità selezione facce, la faccia che corrisponde alla base della colonna, quindi estrudiamola e; a questo punto, sta a voi modellare la colonna!

Qui davvero potete sbizzarrirvi: inserendo un cilindro in Edit Mode, questo farà comunque parte della mesh originale e verrà replicato con Array, oppure potete utilizzare la faccia quadrata iniziale ma, inserendo delle suddivisioni, potete fare una torsione su un gruppo di vertici ristretto con *Simple Deform – Twist*, o altro ancora... davvero, c'è l'imbarazzo della scelta: io sto usando un semplice cilindro giusto per sbrigarmi anche perché, in effetti, l'argomento di questo tutorial era *Math Function*, che ora sta quasi passando in secondo piano!




*Modellazione della colonna*

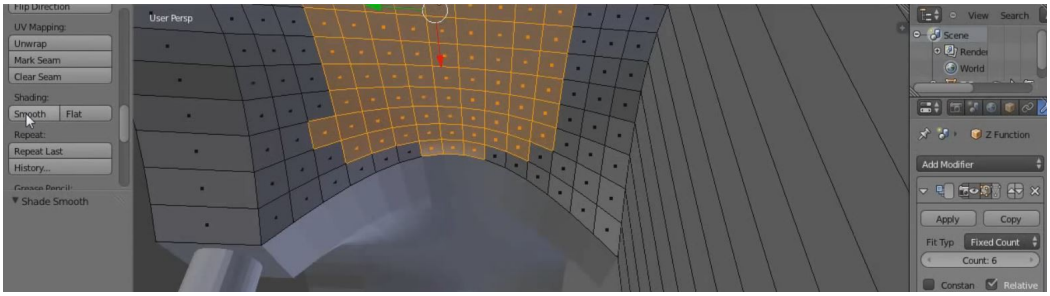
A questo punto, comunque, torniamo in Object Mode e aggiungiamo un modificatore Array, aumentando il valore del parametro Count... ed ecco qui il nostro porticato!




*Utilizzo del modificatore Array*

Punto di forza di Array è che, ora, tornando in Edit Mode, potremo modificare una parte della mesh originale e ritrovare le modifiche nelle copie; questo vale ovviamente se Array non viene applicato, quindi se dovete esportare la mesh completa o comunque dovete applicare il modificatore, fate tutte le modifiche necessarie e magari salvate una copia del file a parte, con Array non ancora applicato definitivamente, prima di cliccare su *Apply*!

Un esempio di modifica da applicare ora all'originale riguarda, ad esempio, lo Shading: per far apparire più levigate le facce, senza aumentare il numero di vertici, passiamo in Edit Mode, selezioniamo le facce inferiori dell'oggetto e, nella Tool Shelf, clicchiamo su *Shading Smooth*, per applicare questa modalità alla selezione dell'oggetto originale, ritrovandola però, come si può constatare, anche nelle copie create con Array, non ancora applicato definitivamente.




*Utilizzo di Shading Smooth su una selezione in Edit Mode*

Ricapitolando, abbiamo parlato di *Math Function* e l'abbiamo utilizzato in combinazione con Array per ottenere un risultato interessante in pochi minuti; l'esempio pratico visto è molto semplice ma gli strumenti (sia singolarmente che in combinazione) si prestano a moltissimi utilizzi.

*  *  *

# Come fermare un sistema Emitter con Damp: una galassia a spirale

In questo tutorial vedremo un modo per “congelare” nella scena 3D le particelle di un sistema *Emitter* in Blender 3D quando queste raggiungono una certa configurazione, fermandole ed evitando che si disperdano nell'universo virtuale...

… e, parlando di universo virtuale, come esempio pratico modelleremo una galassia, o comunque darò le basi per poter creare oggetti simili, poi ovviamente voi potrete sperimentare con altre mesh, altre impostazioni per i valori eccetera.

Attenzione, prima di continuare, voglio anticipare una possibile obiezione: se il nostro scopo è creare una galassia a spirale, non abbiamo bisogno di *Damp*: possiamo modellare le strutture dell'oggetto e creare i sistemi particellari con Start ed End 1 (per creare tutte le particelle in un colpo) e con *Physics* disattivato, per non farle muovere: in questo modo, con tre impostazioni otterremmo la galassia senza bisogno di *Damp* o altro... MA, questo non è un tutorial su come modellare una galassia, bensì un tutorial su come utilizzare *Damp* per fermare un sistema *Emitter* dotato di *Physics* e in questo tutorial useremo, come esempio di oggetto, una galassia... attenzione, quindi, a non confondere l'argomento del tutorial con uno dei possibili utilizzi!

Altro avvertimento: questo NON è un tutorial di base, vengono dati per noti molti argomenti di modellazione, animazione e sistemi particellari... in effetti, è un tutorial di livello medio-avanzato.

Anche se l'argomento centrale del tutorial è *Damp*, tratteremo in generale molti aspetti dei sistemi particellari *Emitter*, parlando anche dei Children, di come annullare la gravità e altro ancora.

Iniziamo pensando a cosa vogliamo ottenere: un set di particelle disposte principalmente su un piano (anche se c'è un certo spessore verticale) dove sia possibile individuare delle strutture, detti bracci di spirale, più un rigonfiamento centrale.

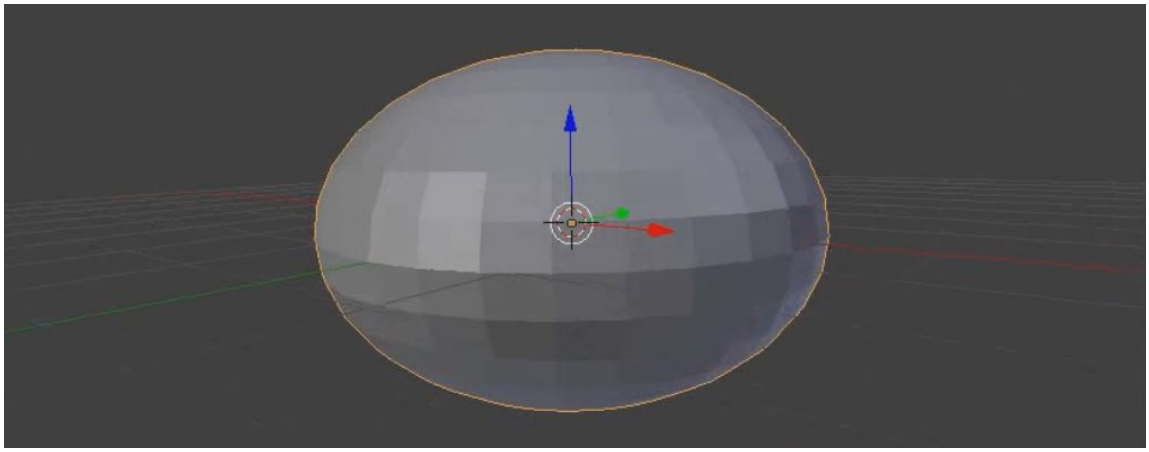
In questo tutorial modelleremo solo i sistemi particellari, non tratteremo materiali, illuminazione e compositing per il rendering.

Avremo bisogno di ben tre sistemi particellari: uno per un set di stelle generico, uno per i bracci di spirale e uno per il rigonfiamento centrale.

Tutti e tre hanno, comunque, una caratteristica comune: bisogna "fermare" le particelle di *Emitter* in una determinata posizione.

Come detto prima, una galassia può essere modellata agendo semplicemente su Start, End e *Physics*, e questo è quello che faremo con il rigonfiamento centrale (così, nel mentre, vedremo in pratica quanto detto all'inizio del tutorial... due tecniche in un solo tutorial).

Iniziamo quindi creando una sfera e schiacciandola un po' verticalmente, con S Z.



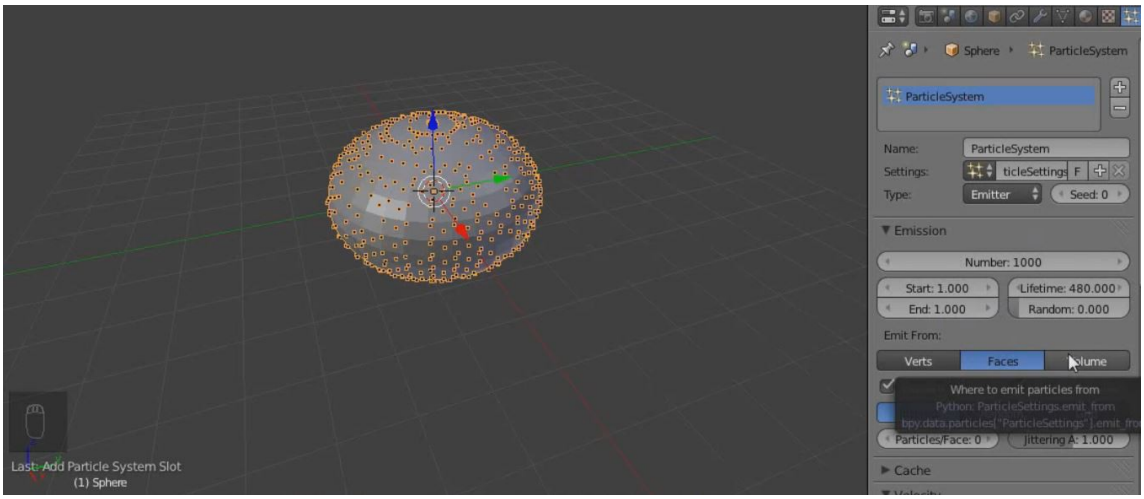
*La sfera, un po' schiacciata verticalmente*

In questo caso, vogliamo le particelle all'interno della mesh, quindi con generazione dal volume, non da facce o vertici, senza fisica, immobili nel tempo.

Aggiungiamo quindi un *Particle Emitter* per la sfera e impostiamo un numero di particelle a piacere (potete provare ad esempio 1000 e aumentarle poi, facendo dei paragoni con quelle create per disco e bracci), con Start 1, End 1 e Lifetime doppia della durata del videoclip che volete creare, vi dirò presto perché... ad esempio, 480.

Sempre in questa sezione, selezioniamo Volume (per far generare le particelle all'interno del Volume della sfera, non su facce o vertici della mesh) e Random, per una distribuzione casuale.

Nella sezione *Physics*, selezioniamo la casella None, mentre nella sezione Render deselezioniamo *Emitter*, per evitare di renderizzare la sfera generatrice.

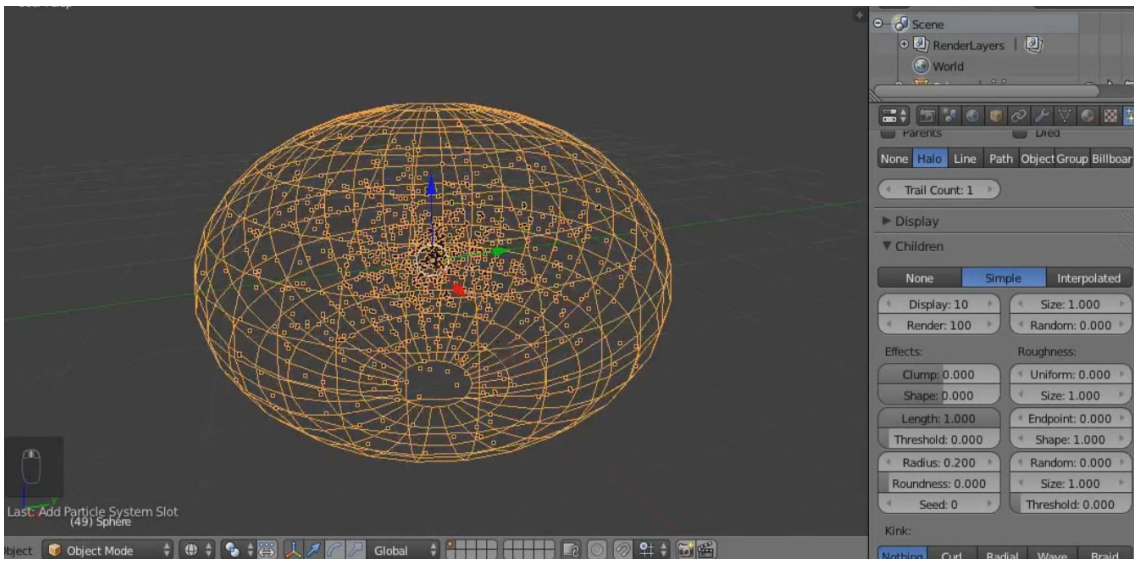


*Il Particle Emitter creato per la sfera*

Il numero di stelle nel campo Number può sembrare piccolo ma possiamo aggiungere delle particelle "figlie" (children) di quelle principali selezionando Simple in Children e impostando un valore per Render; la differenza tra i campi View e Render sta nel fatto che View indica quante particelle figlie visualizzare per ciascuna particella ADESSO, nella 3D View, in fase di editing (ma un numero elevato può appesantire troppo l'editing), mentre Render indica quante crearne una volta lanciato il rendering, per cui ad esempio possiamo lasciare 10-100 (in modo da avere un editing fluido con 10.000 particelle visualizzate ma 100.000 in fase di rendering).

Nella Timeline del progetto, impostiamo End Frame 480.

Premendo ALT A per avviare un'anteprima dell'animazione non vedremo le particelle perché queste si trovano all'interno della sfera; per visualizzarle, passiamo alla modalità di visualizzazione Wireframe: come potete vedere (immagine seguente), le particelle ci sono, con distribuzione casuale, immobili (*Physics* è disattivato) e racchiuse in quel volume.



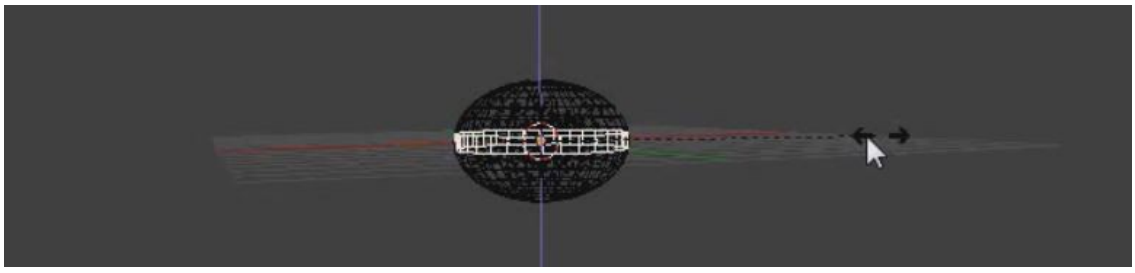

*Screenshot del risultato ottenuto fino a questo punto*

A questo punto, se il vostro scopo è modellare una galassia, potete fermare qui il tutorial, creare due mesh (una per il disco e l'altra per le strutture, come i bracci di spirale o barre e bracci, per le spirali barrate, eccetera) e creare sistemi particellari come fatto finora, con *Physics* disattivato, Start ed End 1 eccetera... qui, continuiamo con *Damp*, sia per il disco che per i bracci di spirale.

Iniziamo con il disco, che ci consente di utilizzare *Damp* senza il discorso della rotazione dei bracci.

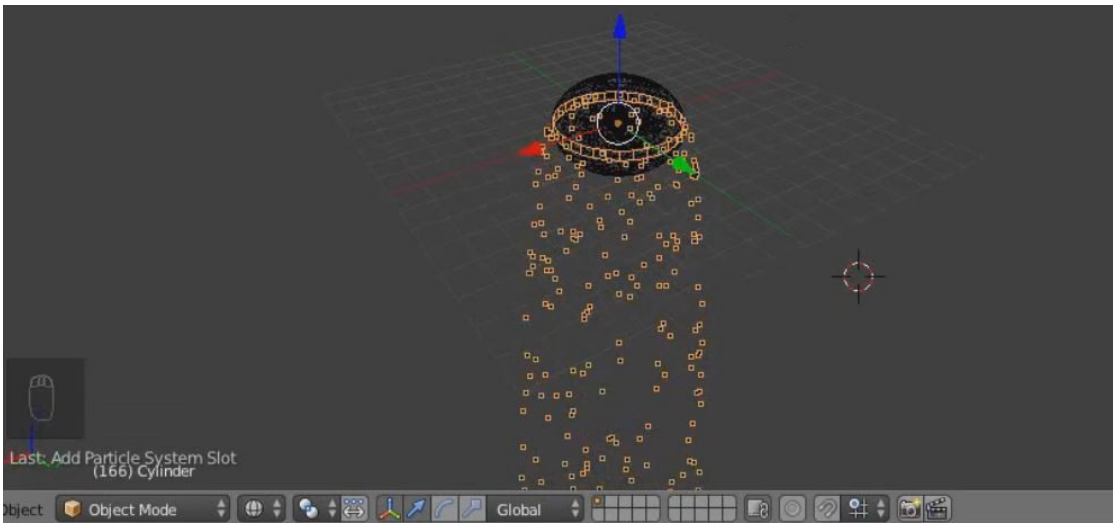
Inseriamo un cilindro nella scena, centrato nella sfera (per mettere lì il 3D Cursor, premete SHIFT S e Cursor to Selected, con la sfera selezionata), in particolare con 32 lati (come di default) ma con *Nothing* nel campo *Cap Fill Type*, per non creare le basi inferiore e superiore.

Scaliamo il cilindro per dargli il raggio della sfera, quindi scaliamolo verticalmente per appiattirlo: useremo le facce laterali per proiettare le particelle nello spazio lateralmente, creando il disco.



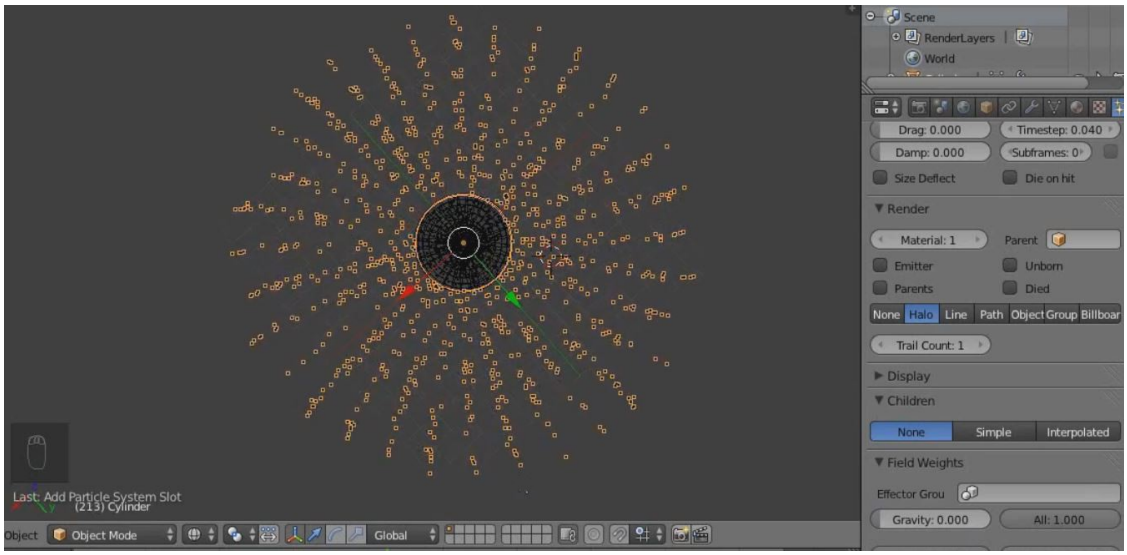
*Il cilindro, senza basi, scalato verticalmente*

Se, a questo punto, andiamo nella scheda Particles e creiamo un Emitter, impostando Start 1, End 240, lifetime 480 (vi prometto che tra poco vi dirò il perché di questi valori, con lifetime doppio di End) e avviamo l'animazione con ALT A, notiamo tre cose: le particelle cadono nel vuoto; due: a lungo andare, il numero – ridotto – di facce porta a generare non un disco ma dei raggi; infine, le particelle vagano nello spazio e anche le ultime generate non restano vicine al centro, non si fermano.



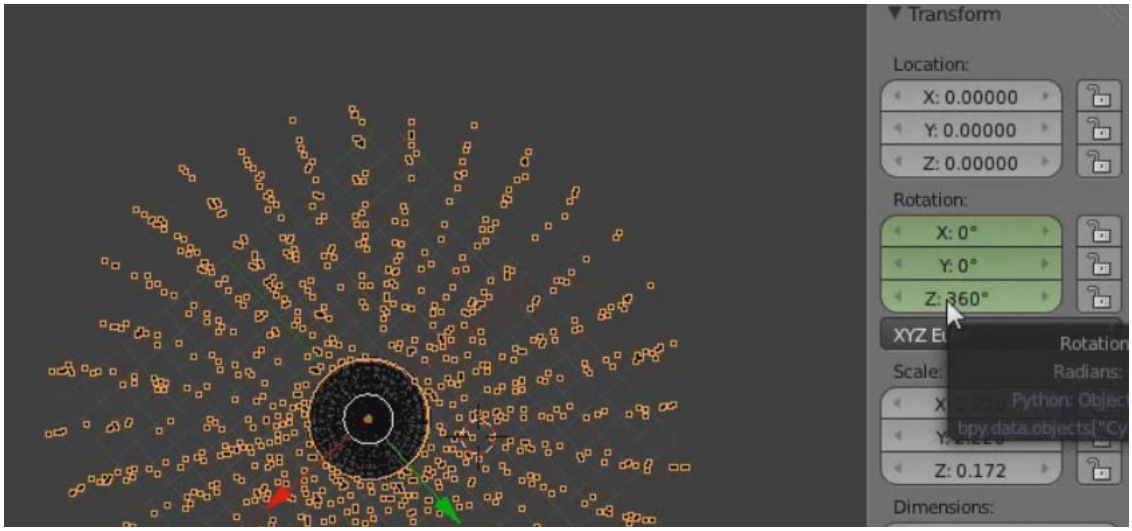

*Primo risultato del sistema particellare, con qualche problema*

Deselezionate *Emitter* nella sezione Rendering del sistema: anche se in questo tutorial non parleremo del rendering, teniamo sempre a mente che non vogliamo la mesh di emissione.
Per non far cadere le particelle, nella sezione *Field Weights* portiamo *Gravity* a 0: ora le particelle viaggeranno in direzione perpendicolare alla faccia di emissione, come visibile con ALT A.



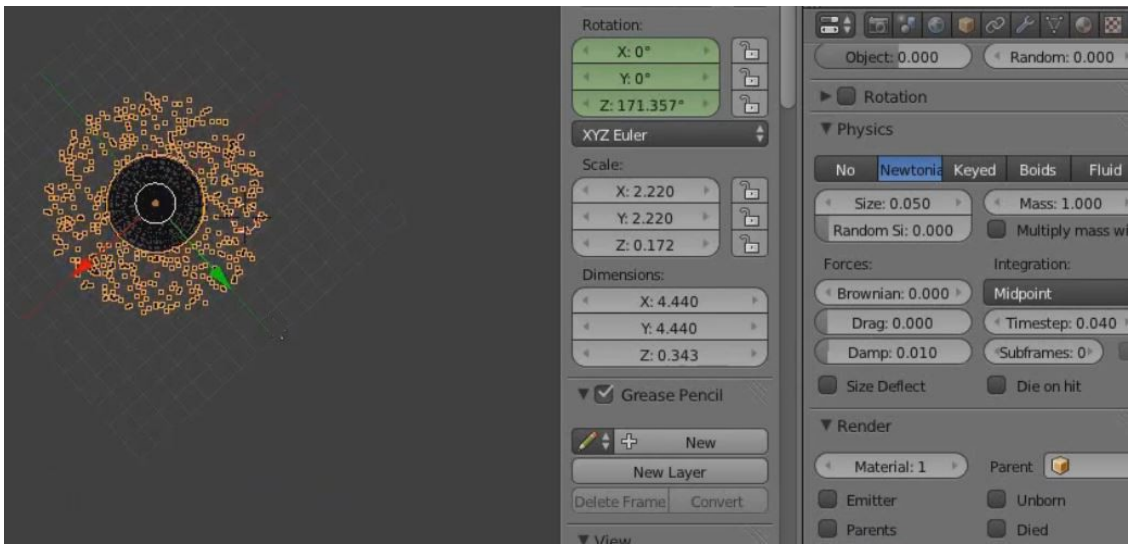

*Anteprima con Gravity a 0*

Risolvere il secondo problema, quello dei "raggi", è semplicissimo: facciamo ruotare il cilindro.
In particolare, al frame 1, con il cilindro selezionato, premiamo *I* mentre il mouse è su una vista 3D e scegliamo Rotation, per memorizzare la rotazione, quindi andiamo al frame 240 (ultimo frame dell'emissione), apriamo la Transform Window della 3D View, impostiamo *Rotation Z* (intorno all'asse verticale) ad esempio 360 e registriamo questo Keyframe con *I – Rotation*.




*Animazione della rotazione intorno all'asse Z: inserimento Keyframe*

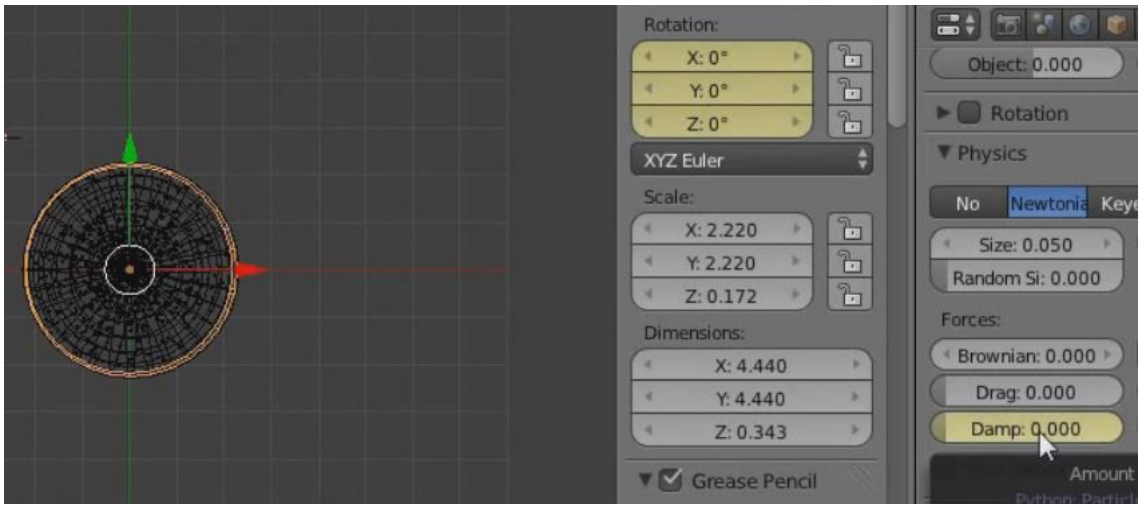
Eccoci all'argomento principale: come frenare le particelle.
Il campo *Damp*, nella sezione *Physics*, fa al caso nostro: anche un valore molto basso, come 0.01, rallenta il moto delle particelle, mentre il valore massimo (1.0) le ferma del tutto.
Proviamo, ad esempio, a impostare *Damp* 0.01 e avviare l'anteprima con ALT A: il disco è molto più ridotto, compatto, ma l'emissione finale c'è...




*Primi effetti di DAMP*

Dobbiamo, quindi, animare questo valore (è possibile inserire un keyframe sul campo mettendoci sopra il mouse e premendo *I* o click destro – *Insert Keyframe*).

Procediamo in questo modo: al primo frame, impostiamo un keyframe con *Damp* 0.0: ciò garantirà una buona espansione del disco con particelle sparse; al frame 230, mettiamo *Damp* a 0.05 e registriamo il keyframe: le particelle rallenteranno nel tempo e, nel mentre, si addenseranno al centro, perché le ultime generate saranno più lente già alla nascita; al frame 240, infine, mettiamo *Damp* a 1.0 e registriamo il keyframe, per fermare del tutto tutte le particelle.



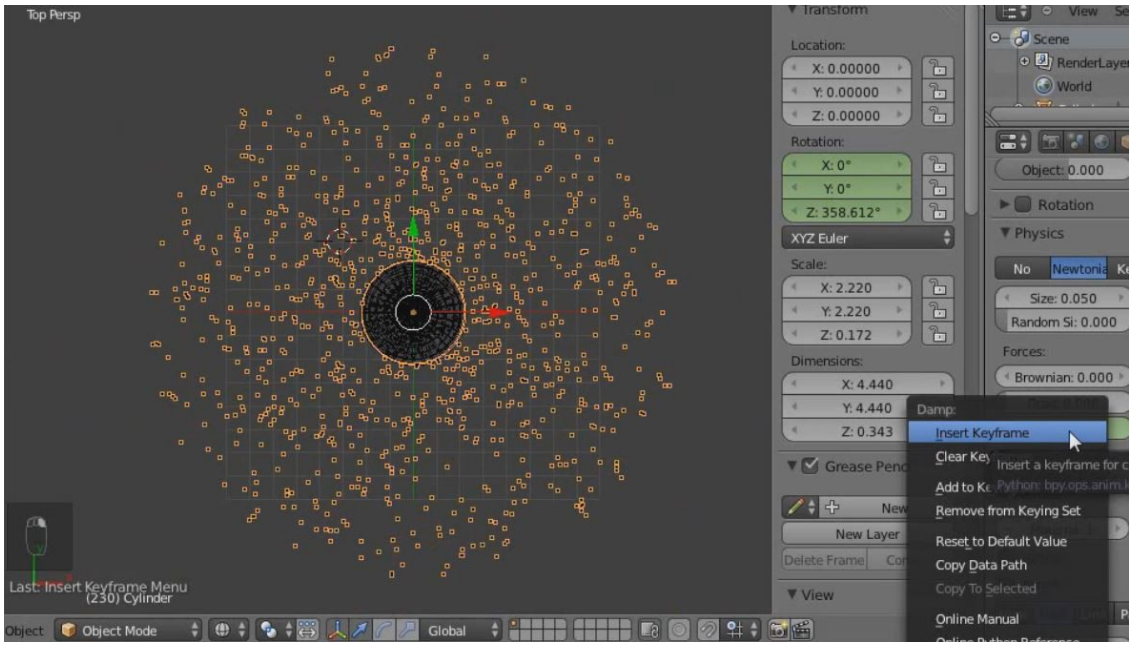

*Registrazione dei Keyframe per DAMP*

A questo punto DEVO dirvi perché l'End Time è 480 e le particelle vengono emesse fino a 240: dal momento che abbiamo scelto di generare la configurazione in maniera "fisica" e di congelare la posizione delle particelle, abbiamo bisogno di tempo per arrivare a questa configurazione, con un certo numero di frames che non andranno renderizzati, ma servono solo per impostare l'oggetto.

Per arrivare alla configurazione, useremo i primi 240 frame della Timeline: ecco perché Start è a 1, End a 240 e animeremo il *Damp* entro i primi 240 frames.

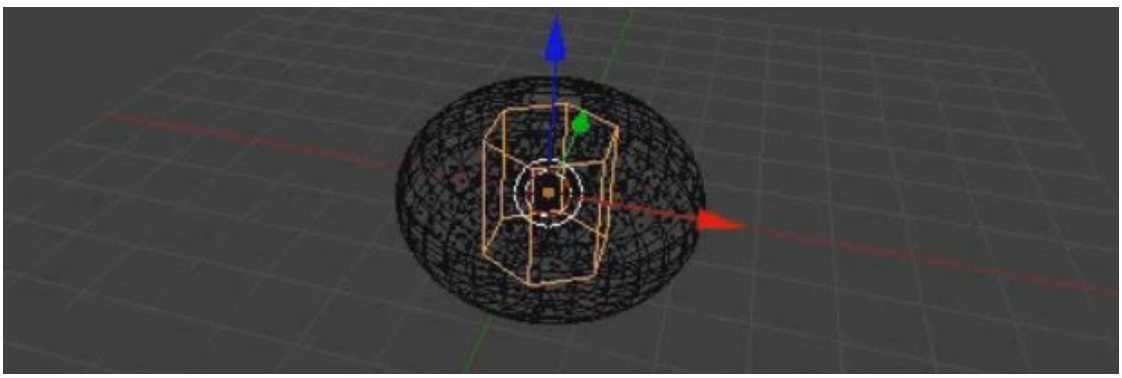
I restanti 240 frames del progetto servono invece come tempo di rendering di base, ad esempio per realizzare un videoclip di 10 secondi (a 24 frame al secondo), per cui il movimento della telecamera di rendering dovrà iniziare al 240esimo frame e, ovviamente, la scena andrà renderizzata da quel frame, non dal frame 1.

Ricapitolando: i primi 240 frame per arrivare alla configurazione desiderata... poi, un numero qualsiasi di frame per il rendering del videoclip (ovviamente, impostate la Lifetime delle particelle in modo da includere sia i frame di creazione sia quelli necessari per il videoclip vero e proprio).
Avviate ora l'anteprima dell'animazione: il disco si espande dal centro, con una disposizione abbastanza casuale delle particelle, con un'espansione che rallenta e si ferma del tutto (anzi, per la precisione, si fermano proprio le particelle, che nel mentre si addensano nella parte centrale).

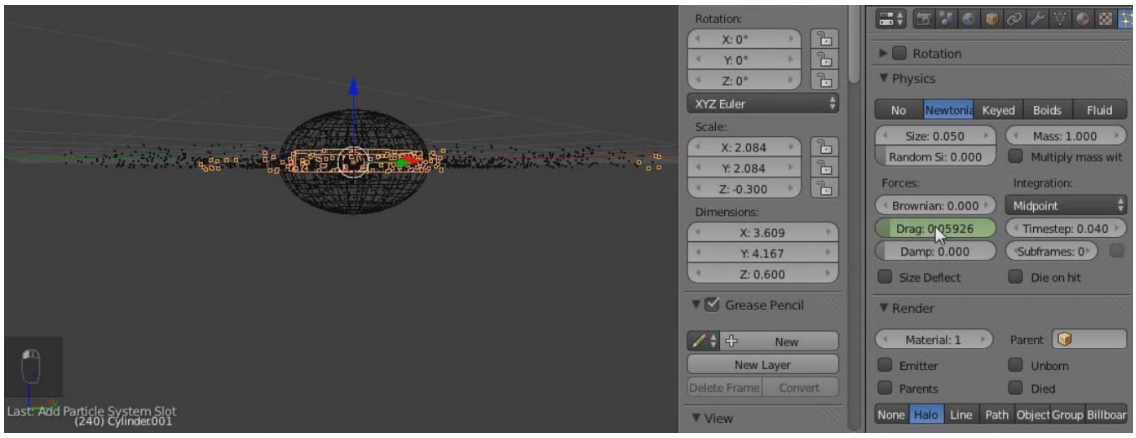


*Screenshot del risultato ottenuto fino a questo punto*

Il cuore del tutorial è tutto qui e la tecnica è la stessa da usare per i bracci di spirale, solo che in questo caso cambia qualcosa, come vedremo.
Centrando il 3D Cursor nella sfera del rigonfiamento centrale, inseriamo un nuovo cilindro, questa volta però con 6 facce laterali (potete provare anche con 4... beh, sostanzialmente, un Cube senza basi superiore e inferiore) e niente *Cap Fill* (niente basi).

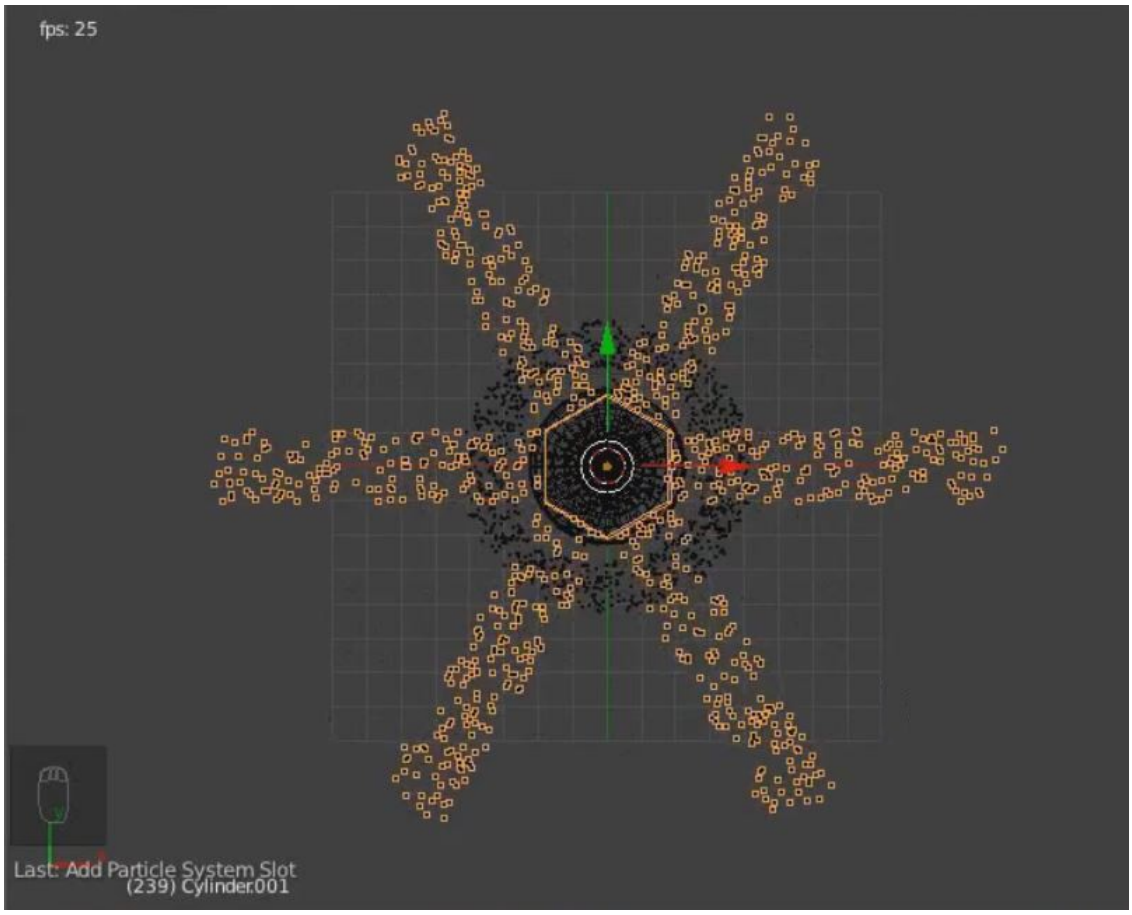

*Il prisma esagonale senza basi*

Creiamo un sistema particellare *Emitter* con Start 1, End 240, Lifetime 480, quindi *Gravity* 0.0, Render Emitter deselezionato e, come fatto per il cilindro precedente, *Damp* animato in questo modo: al frame 1 valore 0.0, al frame 230 valore 0.05, al frame 240 valore 1.0.




*Il terzo sistema particellare*

Avviamo l'anteprima dell'animazione: non preoccupatevi se le particelle del sistema precedente sembrano apparire in punti diversi, ogni tanto Blender fa di questi scherzi ma dal frame 240 in poi i sistemi dovrebbero avere la loro configurazione corretta.

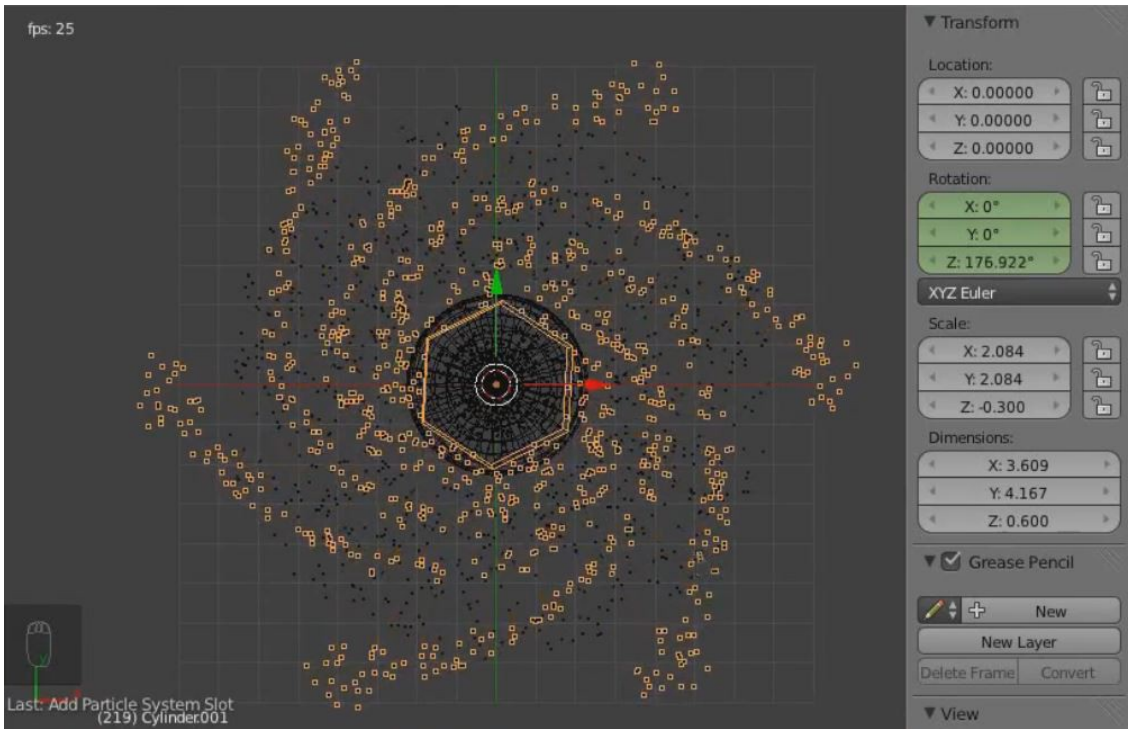



*Screenshot del risultato ottenuto fino a questo punto*

Sincronizzando il Damping e l'emissione, bracci di spirale e disco hanno la stessa estensione, ma i bracci sono... “dritti” (sono raggi di una ruota, piuttosto che bracci di spirale).

Dobbiamo quindi animare la rotazione del prisma esagonale che genera i raggi, operazione semplicissima, ma seguite il tutorial perché vedremo un piccolo inconveniente che si ha con la rotazione e come risolverlo.

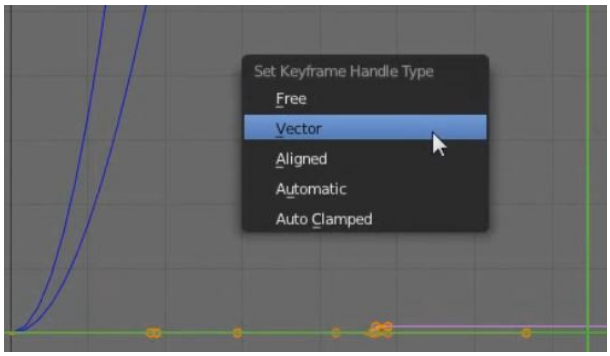
Al frame 1 della Timeline, con il prisma esagonale selezionato, inseriamo un keyframe *Rotation*, quindi andiamo al frame 240, impostiamo *Rotation Z* 180 (o 360 o un altro valore, dipende dalla forma che volete ottenere, questi sono solo degli esempi: anche il numero dei lati, 6, non è un valore esatto: potete provarne anche altri e valutare i risultati), registriamo quindi il keyframe di rotazione anche al frame 240, torniamo al frame 1 e osserviamo il risultato con ALT A.



*Screenshot dell'animazione realizzata*

L'inconveniente di cui vi parlavo è visibile alle estremità dei bracci di spirale: le estremità sono “gonfie” e poco curve.

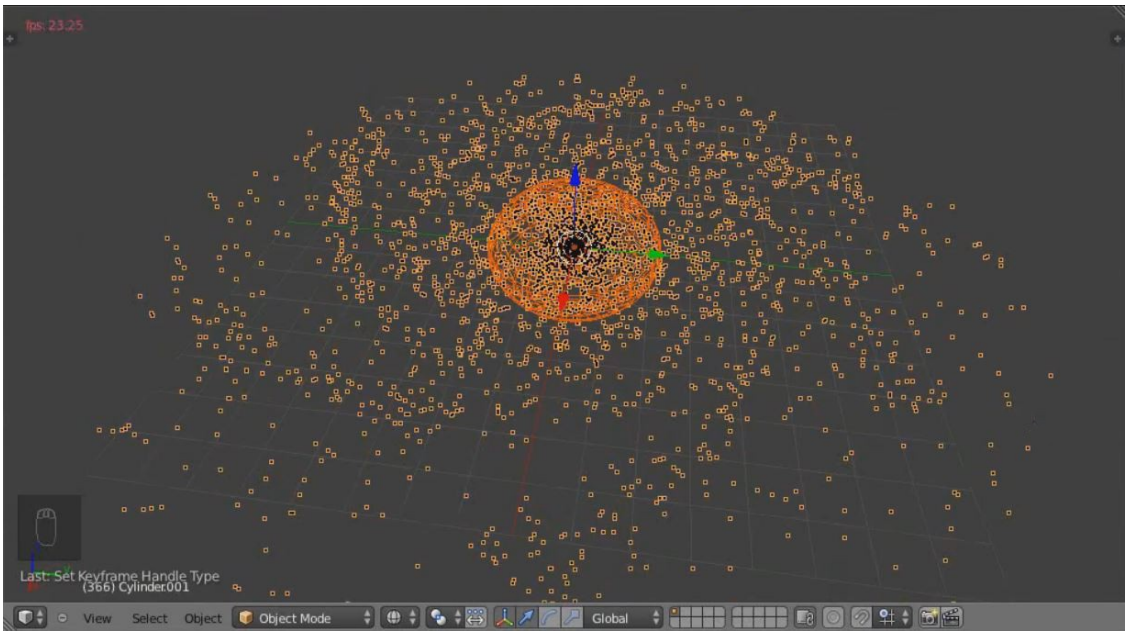
Il motivo sta nell'interpolazione dell'animazione ai keyframe iniziale e finale: le curve IPO sono smussate, cioè l'animazione parte e termina in maniera "dolce", non lineare, e questo andamento si riflette nella pratica sulla disposizione delle particelle emesse per prime e per ultime.




*Rendere vettoriali (interpolazione lineare) le curve IPO*

Per risolvere il problema, cambiamo una finestra in Graph Editor, dove (osservando la curva IPO della rotazione intorno all'asse Z) si vede chiaramente l'interpolazione Bezier, "dolce", ai keyframe 1 e 240; selezioniamo i keyframe presenti con A (per selezionare tutto), quindi premiamo *V* e scegliamo l'opzione *Vector* (vettoriale, lineare).

Torniamo nella 3D View al frame 1 e facciamo partire l'anteprima dell'animazione con ALT A: nei primi 240 frame abbiamo la generazione "dietro le quinte" (con possibili comportamenti strani da parte delle particelle, che al 240esimo frame dovrebbero raggiungere la configurazione finale), dopodiché abbiamo la struttura finale, immutabile, disponibile per essere inquadrata da una telecamera in movimento eccetera.




*Screenshot del risultato ottenuto*

Come detto, questo non è il metodo più veloce per generare una galassia con un sistema *Emitter*, ma solo un esempio per parlare di *Damp*, di come utilizzarlo in un caso pratico, di come "congelare" nel tempo un *Emitter* e risolvere un problema con le particelle emesse da oggetti in rotazione.

*  *  *

# Script Python: un piccolo planetario – I parte

In questo breve tutorial vedremo uno script in Python per Blender 3D 2.6 per inserire, nella scena 3D, delle sfere, rappresentanti delle stelle o anche altri oggetti celesti, partendo dai dati delle coordinate celesti, ossia ascensione retta, declinazione e distanza.

Per un'introduzione alla programmazione Python e, soprattutto, Python per Blender, sono disponibili un paio di miei ebook a riguardo (sezione “ebooks” nel mio sito web).

In questa prima parte vedremo alcuni elementi di teoria e la disposizione delle sfere, in base alle coordinate celesti, nella scena 3D; nella prossima puntata, un paio di “abbellimenti”, come il colore delle sfere in base ai colori reali dei relativi astri, definendo i Materials via codice Python.

Prima di mettere mano al codice, vediamo di capire cosa sono questi dati; si tratta di una piccola introduzione di geografia astronomica, cercherò di essere breve ma il più chiaro possibile.

L'immagine a lato, tratta da Wikipedia, mostra il modello della sfera celeste: un modello ideale (considerando che le stelle sono a distanze diverse) sul quale “proiettiamo” gli astri, osservandoli dalla Terra.

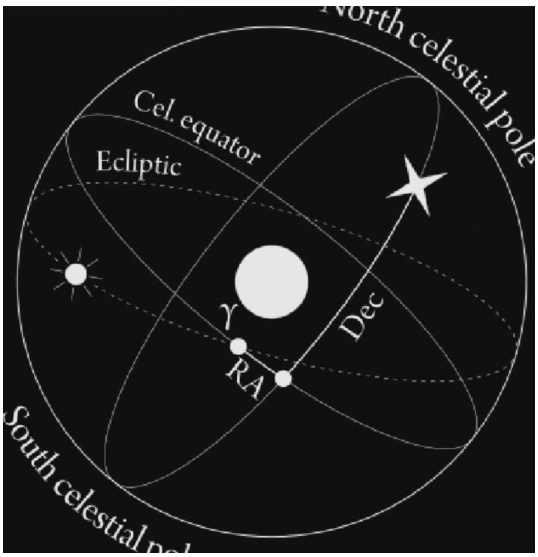


*Immagine da Wikipedia*

Come potete vedere, la Terra è inclinata, in quanto l'asse terrestre è effettivamente inclinato, attualmente, di 23°27': ecco perché l'equatore celeste non coincide col piano del sistema solare e il Sole percorre l'eclittica.

Così come sulla Terra abbiamo latitudine e longitudine, anche sulla sfera celeste abbiamo un sistema di coordinate, per individuare facilmente gli astri: si tratta di declinazione (che sarebbe la latitudine, misurata in gradi, da +90 per il Polo Nord celeste e -90 per il Polo Sud celeste) e ascensione retta, misurata in ore, minuti e secondi, quindi la nostra "longitudine" va dalle ore 00 a 24, in un giro completo.

L'argomento è abbastanza vasto ma per i nostri scopi possiamo fermarci qui, ossia: convertire le coordinate di ascensione retta e declinazione di un astro (facilmente reperibili, come vedremo) e distanza in anni luce in coordinate XYZ nell'universo virtuale di Blender.

Apriamo ad esempio la pagina Wiki della costellazione di Orione e, da qui, guardiamo ad esempio Betelgeuse, una supergigante rossa prossima (si fa per dire) alla fine dei suoi giorni: la pagina ci dice che tale stella si trova a ore 5:55, con declinazione positiva 7°24' e distanza 640 anni luce, anno luce più anno luce meno.

| Coordinate | |
|---|---|
| (all'epoca J2000.0) | |
| **Ascensione retta** | $05^{h}\ 55^{m}\ 10{,}3053^{s}$ [2] |
| **Declinazione** | +07° 24′ 25,426″[2] |
| **Lat. galattica** | 199,7872 ± 20,24[2] |
| **Long. galattica** | −08,9586 ± 11,61[2] |

*Scheda di Betelgeuse su Wikipedia*

Siccome sono pigro, per lo script che stiamo per realizzare ho fatto diverse semplificazioni rispetto al modello corretto; in particolare: ho pensato la Terra come dritta e, di conseguenza, l'equatore celeste coinciderà, nel nostro modello, con la griglia XY di Blender 3D; inoltre, anche qui per semplificare i calcoli di trigonometria, sto considerando le ore 00 (che dovrebbero coincidere con il punto d'Ariete, uno dei punti di intersezione tra eclittica ed equatore celeste, nella figura vista prima) come "+X", cioè sull'asse X positivo.

Indichiamo l'ascensione retta con AR, per cui per lo script dovremo individuare i parametri *arh*, *arm* e *ars*, ossia ore, minuti e secondi così come si trovano nella pagina wiki o sulle carte astronomiche; per la declinazione, individuiamo non solo i parametri *decg*, *decp*, *decs*, ossia la declinazione in gradi, primi e secondi, ma anche un parametro segno, che indicherà se *dec* sarà positiva o negativa, per fare i calcoli bene; resta, infine, la distanza, *dist*, in anni luce, per cui per noi un anno luce sarà un'unità nella scena 3D.

Ora, una piccola nota, prima di procedere: per fare la conversione in coordinate XYZ, ho buttato giù un paio di formule che PENSO possano andare bene, ma non me ne assumo completamente la responsabilità... anzi, si accettano commenti e suggerimenti, come commenti al video nel forum *effe12.com*; tenete conto sempre delle semplificazioni che ho detto prima, ossia equatore celeste identificato con il piano XY e ore 00 con l'asse X positivo.

In particolare, per proiettare l'astro nel 3D, prima ho fatto una conversione sul piano XY, mediante trigonometria, quindi per far entrare in gioco anche l'asse Z ho moltiplicato i valori X e Y per il coseno della declinazione, perché (ed è qui che potrei sbagliarmi, quindi vi chiedo di correggermi, in caso) il punto proiettato su XY si avvicinerà al centro in maniera proporzionale alla declinazione dell'astro, ma questa "proporzione" non è lineare ma, anche questa, con "andamento trigonometrico", per così dire, in particolare in base al coseno della declinazione.
Per la coordinata Z, invece, ho messo solo il seno della declinazione.

Le tre equazioni alle quali sono arrivato sono quindi queste:

- x = distanza * cos(ascensione retta) * cos (declinazione)
- y = distanza * sin(ascensione retta) * cos(declinazione)
- z = distanza * sin(declinazione)

Se si fa il modulo di questo vettore, si ottiene effettivamente la distanza, il che è confortante ma non so se sufficiente per dire che i calcoli sono corretti; eventualmente, commentate il video nel forum.

Ad ogni modo, i parametri "ascensione retta" e "declinazione" vanno convertiti in valori puri e in radianti, prima di essere utilizzati nello script, ma a questo punto iniziamo subito col codice, parleremo delle conversioni quando le farò.

Per prima cosa, importiamo (in una finestra script editor di Blender 3D, con un nuovo file pronto) i moduli *mathutils*, *math* e *bpy*, scrivendo:

```
import mathutils
from mathutils import *
from math import *

import bpy
from bpy import *
```

Adesso definiamo una funzione, chiamandola ad esempio *star*, che prende in input nome, tre campi per l'ascensione retta (ore, minuti, secondi), quattro campi per la declinazione (gradi, primi, secondi, segno) e distanza (in anni luce), per cui scriviamo:

```
def star(nome, ah, am, ars, decSign, dg, dp, ds, dist):
```

Le istruzioni che seguono sono con indentazione, visto che si trovano nel corpo della funzione *star*.

La prima operazione è la conversione dei dati di ascensione retta in un valore singolo trasformato, poi, in radianti; considerando che, in proporzione tra 360 gradi e 24 ore, un'ora è... "ampia", per così dire, 15 gradi, e che quindi un minuto sarà 15 diviso 60 (cioè 0,25) e un secondo sarà 15 diviso 3600 (cioè 0,0042, più o meno), possiamo effettuare la conversione scrivendo:

```
a = radians((15*ah) + (0.25*am) + (0.0042*ars))
```

Per la declinazione, sarà sufficiente sommare al valore in gradi i primi diviso 60 e i secondi diviso 3600; attenzione: qui useremo i valori in valore assoluto, poi terremo conto del segno con il relativo parametro. Scriviamo quindi:

```
dec = radians((abs(dg) + (abs(dp)/60) +
(abs(ds)/3600))*decSign)
```

A questo punto la conversione, seguendo le equazioni che vi ho detto prima, è semplice; si tratta di scrivere le seguenti tre istruzioni:

```
x = dist * cos(a) * cos(dec)
y = dist * sin(a) * cos(dec)
z = dist * sin(dec)
```

Finalmente, le due istruzioni che consentono di inserire la sfera nella scena con questi dati e di rinominarla con il nome passato come parametro:

```
bpy.ops.mesh.primitive_uv_sphere_add(size = distance/100,
location=(x, y, z))
bpy.context.scene.objects.active.name = nome
```

Come potete notare, ho messo "*size = distance/100*", il che significa che le stelle vicine avranno raggio minore e quelle lontane raggio maggiore... perché? Perché, per via della prospettiva, senza questo piccolo accorgimento le stelle lontane potrebbero risultare troppo piccole o addirittura invisibili, nella scena 3D, per cui (solo per renderle meglio visibili) imposto lo scaling come *distance/100*.

La distanza porta però un altro problema: quello del clipping.
Anche scalando le sfere, queste potrebbero risultare non visibili nella scena 3D perché Blender visualizza l'universo virtuale, sia nella 3D View che nella telecamera di rendering, solo fino a un certo punto (il clipping end); per risolvere il problema in una View 3D, impostate un valore elevato (ad esempio, 5.000 o 6.000) per *Clip End*, nella sezione *Display* della Transform Window della vista 3D; allo stesso modo, per un oggetto Camera, impostate un valore elevato in *Clip End* nella sezione Object Data, nella Properties Window.

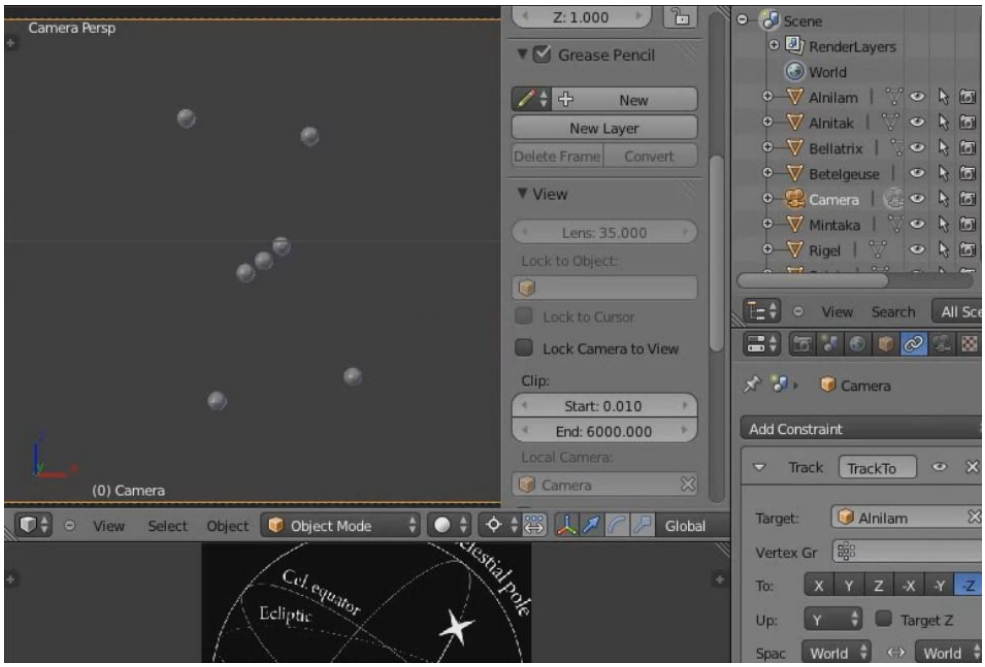
*Clip End della telecamera*

Per testare il tutto, ho recuperato i dati delle principali stelle di Orione e li sto inserendo in coda allo script, chiamando la funzione star con questi parametri (quindi le chiamate a star vanno messe senza indentazione, in quanto chiamano la funzione, non si trovano al suo interno); premendo *Run As Script*, si ha l'inserimento delle sfere nella scena.

```
4
5 import bpy
6 from bpy import *
7
8
9 def star(nome, ah, am, ars, decSign, dg, dp, ds, dist):
10     a = radians((15*ah)+(0.25*am)+(0.0042*ars))
11     dec = radians((abs(dg) + (abs(dp)/60) + (abs(ds)/3600))*decSign
12     x = dist * cos(a) * cos(dec)
13     y = dist * sin(a) * cos(dec)
14     z = dist * sin(dec)
15     bpy.ops.mesh.primitive_uv_sphere_add(size = dist/100, location=
16     bpy.context.scene.objects.active.name = nome
17
18
19 star("Betelgeuse",  5, 55, 10,   1, 7, 24, 25, 640)
20 star("Rigel",       5, 14, 32,  -1, 8, 12, 6, 800)
21 star("Mintaka",     5, 32, 0,   -1, 0, 17, 57, 900)
22 star("Alnilam",     5, 36, 18,  -1, 1, 12, 6.9, 1300)
```

*Inserimento dei dati di Orione, prima di cliccare su "Run As Script"*

Per individuarle facilmente, facciamo così: diamo alla telecamera un Constraint *Track To*, con asse verticale Y positivo e asse di puntamento -Z, quindi specifichiamo come *Target*, ad esempio, Betelgeuse; la telecamera inquadrerà la nostra costellazione, dove tutto sembrerà al posto giusto.




*La costellazione di Orione, inquadrata velocemente grazie al Constraint TRACK TO su una delle sue stelle*

Per questa prima parte è tutto; abbiamo visto in particolare come convertire le coordinate celesti di ascensione retta, declinazione e anni luce (dati che trovate facilmente online per moltissimi astri) in coordinate XYZ, risolvendo anche certi problemi di visualizzazione nella scena; nella prossima puntata faremo qualche modifica allo script per introdurre altri dati e "pensare" ad ulteriori sviluppi.

*  *  *

# Script Python: un piccolo planetario – II parte

In questa seconda e ultima puntata del minitutorial di livello medio su come posizionare, mediante script Python, degli oggetti nella scena 3D di Blender partendo dai dati astronomici, vedremo come accedere ai singoli oggetti della scena, in modo da poter effettuare delle operazioni via codice, in particolare impostando i Nodes per i Materials in Cycles, argomento di programmazione Blender Python API che non ho mai trattato né nei videotutorials né negli ebook pubblicati finora.

Per le stelle, esiste una classificazione sulla base del loro spettro, con tale classe spettrale basata sulla temperatura superficiale e, quindi, sul colore.

Le classi spettrali sono indicate con lettere, dalle temperature più calde alle più fredde (beh, "fredde" per modo di dire, in senso relativo!) secondo la sequenza O B A F G K M; un trucco mnemonico per ricordare la sequenza è la frase "*Oh be a fine girl kiss me*", comunque i colori vanno dal blu al rosso passando per bianco e giallo (infatti le stelle più calde sono di colore blu, mentre le più "fredde", sono rosse).

Le classi O B A F G K M possono quindi essere rappresentate con altrettanti Materials, nella scena di Blender, per cui per prima cosa vediamo come impostare il motore di rendering Cycles e creare 7 Materials di tipo *Emission* colorati, il tutto via codice; in seguito, vedremo come assegnare i Materials agli oggetti presenti nella scena.

Nel corso di questo tutorial utilizzeremo due diversi moduli delle librerie Python di Blender (BPY): il *Context*, per modificare il contesto di esecuzione; *data*, per accedere ai dati presenti nella scena e modificarli.

Per poter cambiare motore di rendering, dobbiamo accedere al *Context*, il contesto di esecuzione del programma.

In particolare, riprendendo la scena dal tutorial precedente (creata con un file script, che potete ampliare col caricamento di altre stelle e altre operazioni), scriviamo questa volta in un nuovo file di testo:

```
import bpy
from bpy import *
```

per importare i moduli bpy, per operare su Blender con Python, quindi diciamo al contesto di esecuzione corrente di impostare il rendering engine CYCLES con:

```
bpy.context.scene.render.engine = 'CYCLES'
```

A questo punto, per aggiungere dei Materials personalizzati nella scena e modificare le proprietà delle mesh presenti (assegnando loro i Materials), dobbiamo utilizzare il modulo *Data*, che contiene appunto i dati presenti nell'ambiente.

Per creare 7 Materials, impostando Shader e colore di base, scriviamo ad esempio:

```
O = bpy.data.materials.new("O")
```

per cui la O alla quale assegniamo il bpy data è la variabile con la quale ci riferiremo, in seguito, a questo nuovo Material, nel codice Python; la "O" tra doppi apici passata come parametro è, invece, il nome del Material nella scena, così come si troverà nella sezione Materials dell'editor Properties.

Facciamo quindi dei copia-incolla cambiando le lettere secondo lo schema O B A F G K M, per cui avremo sette variabili che si riferiranno ai relativi Materials, creati così nella scena via codice, ma a questo punto c'è un piccolo rallentamento nelle nostre operazioni...

```
bpy.context.scene.render.engine = 'CYCLES'

O = bpy.data.materials.new("O")
B = bpy.data.materials.new("B")
A = bpy.data.materials.new("A")
F = bpy.data.materials.new("F")
G = bpy.data.materials.new("G")
K = bpy.data.materials.new("K")
M = bpy.data.materials.new("M")
```

*La creazione dei Materials, prima parte*

… rallentamento dovuto al fatto che in Cycles i Materials sono definiti mediante Nodes, in particolare mediante *Nodes Trees*, alberi di Nodi, visto che più Nodes possono essere miscelati in vari modi, per cui non abbiamo le singole proprietà come in Blender Render (ad esempio, Emit, Specular_Color, ecc...) ma dobbiamo attivare l'utilizzo dei Nodes Trees e, per ciascun Node Tree, specificare almeno un Nodo per dire che tipo di Shader (non più di Material) vogliamo definire e personalizzare.

Per prima cosa, quindi, per tutti i sette Materials creati, attiviamo l'utilizzo dei Nodes, scrivendo:

```
O.use_nodes = True
```

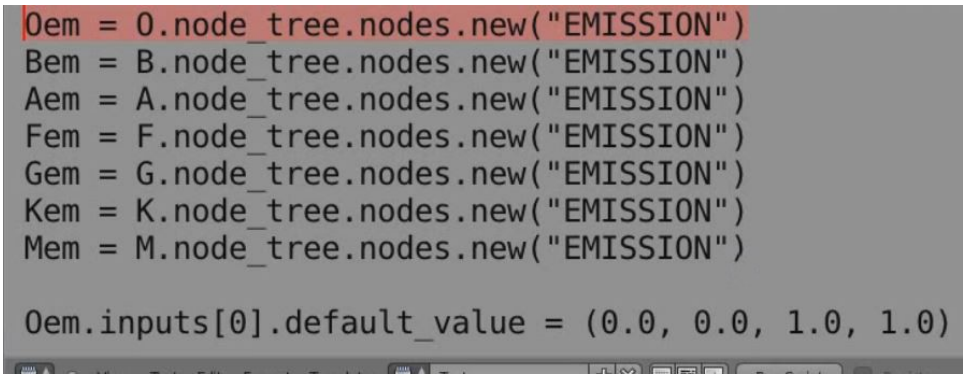
e così via con dei copia-incolla, modificando solo la lettera iniziale (ossia, la variabile che identifica il Material), per B A F G K M.

Adesso possiamo accedere al Node Tree di un Material con *.node_tree*, per cui per aggiungere un Nodo (uno Shader) di tipo Emission per il Material *O*, scriviamo:

```
Oem = O.node_tree.nodes.new("EMISSION")
```

e adesso con la variabile *Oem* possiamo accedere all'Emission del Material *O*.

Facciamo al solito dei copia-incolla con le altre lettere (B A F G K M) per avere le variabili per accedere facilmente agli Emission dei vari Materials.




*La creazione dei Materials, prima parte*

Visto che possiamo accedere agli Shader dei vari Materials, è facile cambiare qualche parametro, ad esempio il colore con:

```
Oem.inputs[0].default_value = (0.0, 0.0, 1.0, 1.0)
```

(i tre canali colore RGB più il canale di opacità Alpha), per cui per le altre variabili definite finora cambiamo i colori, ad esempio:

```
Bem.inputs[0].default_value = (0.5, 0.75, 1.0, 1.0)
Aem.inputs[0].default_value = (0.85, 0.85, 1.0, 1.0)
Fem.inputs[0].default_value = (0.65, 1.0, 1.0, 1.0)
Gem.inputs[0].default_value = (1.0, 1.0, 0.5, 1.0)
Kem.inputs[0].default_value = (1.0, 1.0, 0.3, 1.0)
Mem.inputs[0].default_value = (1.0, 0.1, 0.0, 1.0)
```

(faccio notare che quelli utilizzati qui sono solo valori d'esempio, che ho impostato a occhio, quindi potete cambiarli e sistemarli come preferite).

Non è finita qui: abbiamo creato e personalizzato uno Shader Node per ciascun Material, è vero, ma non abbiamo collegato tale Shader Node all'input Surface del Nodo Material Output del Material, quindi attualmente abbiamo ancora lo Shader di default (il Diffuse grigio chiaro).

Per effettuare il collegamento, dobbiamo: prendere il Nodo Material Output (che di default è quello di indice 0, nell'elenco *node_tree*) e passargli sulla porta d'ingresso Surface (quella di indice 0, identificata con *inputs[0]*) l'output del Nodo Emission... da notare che ad esempio nei nodi Emission, per specificare il colore, abbiamo scritto "*inputs[0].default_value*", perché per un Nodo Emission il primo input è il canale Color: questo è il senso dell'utilizzo di *inputs* e *outputs* nei nodi...

… ok, ma a livello di codice, cos'è un collegamento?
È un oggetto *link*, definito nel *node_tree* specificando due parametri: Nodo e porta di partenza, Nodo e porta di arrivo; in particolare, per il Nodo di partenza la porta sarà una porta di output, mentre per quello di arrivo sarà una porta di input.

Scriviamo quindi:

```
O.node_tree.links.new(Oem.outputs[0],
O.node_tree.nodes[0].inputs[0])
```

e così via per gli altri (potete fare dei copia-incolla sostituendo le lettere dei Materials e degli Shader Emission, visto che per il resto gli indici delle porte restano gli stessi):

```
B.node_tree.links.new(Bem.outputs[0],
B.node_tree.nodes[0].inputs[0])
A.node_tree.links.new(Aem.outputs[0],
A.node_tree.nodes[0].inputs[0])
F.node_tree.links.new(Fem.outputs[0],
F.node_tree.nodes[0].inputs[0])
G.node_tree.links.new(Gem.outputs[0],
G.node_tree.nodes[0].inputs[0])
K.node_tree.links.new(Kem.outputs[0],
K.node_tree.nodes[0].inputs[0])
M.node_tree.links.new(Mem.outputs[0],
M.node_tree.nodes[0].inputs[0])
```

Il codice trattato finora può essere ottimizzato sotto molti aspetti; ad esempio, potete creare una *list* di variabili per i vari Materials ed effettuare tutte le operazioni di utilizzo dei *nodes*, accesso ai *node_tree*, creazione del Node Emission e settaggio del colore all'interno di un ciclo *for*, quindi con poche righe di codice in un'iterazione; qui ho voluto scrivere le operazioni singolarmente per procedere passo per passo, in maniera lineare, per cui c'è poca scalabilità e un certo spreco di tempo e istruzioni, ma è proprio dalla prima bozza che si parte per ottimizzazioni e rifiniture... comunque, su questo punto tornerò a breve.

Anche le istruzioni che seguono, che servono ad accedere agli Objects in base ai loro nomi e a settare i materiali, sono ottimizzabili, comunque procediamo passo per passo: per impostare ad esempio il Material *M* per Betelgeuse, è sufficiente scrivere:

```
bpy.data.objects['Betelgeuse'].active_material = M
```

e così via per le altre mesh presenti, ad esempio:

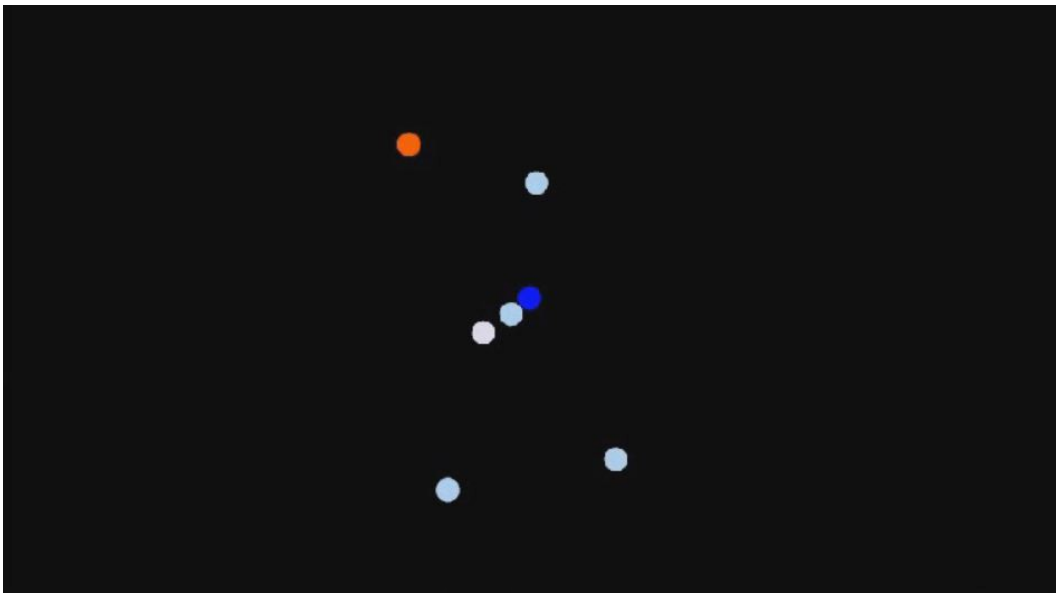
```
bpy.data.objects['Rigel'].active_material = B
bpy.data.objects['Bellatrix'].active_material = B
bpy.data.objects['Saiph'].active_material = B
bpy.data.objects['Mintaka'].active_material = O
bpy.data.objects['Alnilam'].active_material = B
bpy.data.objects['Alnitak'].active_material = A
```

Mentre ci siamo, prima di andare avanti, impostiamo a nero lo sfondo dell'universo virtuale, con:

```
bpy.context.scene.world.horizon_color = (0.0,0.0,0.0)
```

per cui stiamo utilizzando di nuovo il *context*, il contesto della scena, per accedere a World e, da lì, al colore di sfondo, *Horizon Color* (questa volta, alla vecchia maniera, come in Blender Render, mettendo un colore di sfondo su *Horizon*, senza attivare i Nodes per World Background).

Finalmente possiamo cliccare su *Run as Script*, per eseguire lo script, e avviare un rendering della scena, per osservare il risultato.



*Il rendering finale della scena corrente*

È chiaro che, per un progetto con appena 7 elementi, forse si fa prima a impostare i Materials manualmente, ma in un progetto dove in arrivo ci sono magari centinaia o migliaia di elementi, uno script del genere può fare un lavoro che richiederebbe molto tempo in un attimo, con poche

semplici istruzioni; l'importante è conoscere le librerie *bpy* per sapere come accedere ai giusti campi degli oggetti di scena ed è quello che abbiamo appena fatto!

Un paio di spunti per l'ottimizzazione: potete riprendere la funzione "*star*", vista nella puntata precedente, e aggiungere un parametro *color* per passare il riferimento al Material corrispondente già in fase di creazione della mesh, oppure effettuare la... colorazione, per così dire, in un secondo momento, definendo magari le associazioni tra le stelle e i colori in una variabile di tipo *Dict*, dizionario Python, colorando tutte le mesh con un semplicissimo ciclo *for*; questo secondo esempio vi consiglio di provarlo come esercizio: guardando il codice, si vede subito che molte operazioni vengono ripetute ma sono sostanzialmente le stesse (varia il Material al quale sono applicate), possono essere eseguite con delle iterazioni, per cui provate appunto a scrivere una lista per i Materials e un ciclo *for* per creare e personalizzare questi materiali, con poche semplici istruzioni, in maniera più compatta e scalabile, cioè adattabile per casi diversi, con un numero diverso di materiali, nomi, colori, eccetera.

Ricapitolando: i materiali vengono creati con il blocco *bpy.data.materials*, ma in Cycles un Material è in realtà una collezione di Nodi, quindi non possiamo settare direttamente parametri come colore o *emit* come avveniva con un Material nel motore interno di Blender 3D, ma dobbiamo attivare i *nodes*, accedere al *Node Tree* e, lì, creare, personalizzare e collegare un Nodo *Shader Emission* collegandolo (*link*) al Nodo *Material Output*.

*  *  *

# Hair e Strand Render. Realizzare un tappeto in B3D e Cycles

Questo è un breve tutorial dedicato al sistema particellare *Hair* e, soprattutto, al rendering di tali particelle in Blender e Cycles; si tratta di un tutorial base, introduttivo, dove come esempio pratico verrà mostrato un tappeto ma, come potrete intuire, con poche modifiche sarà possibile realizzare altri elementi, ad esempio un tappeto... d'erba, giusto per dirne uno!

Il tutorial è stato realizzato con Blender 2.67.

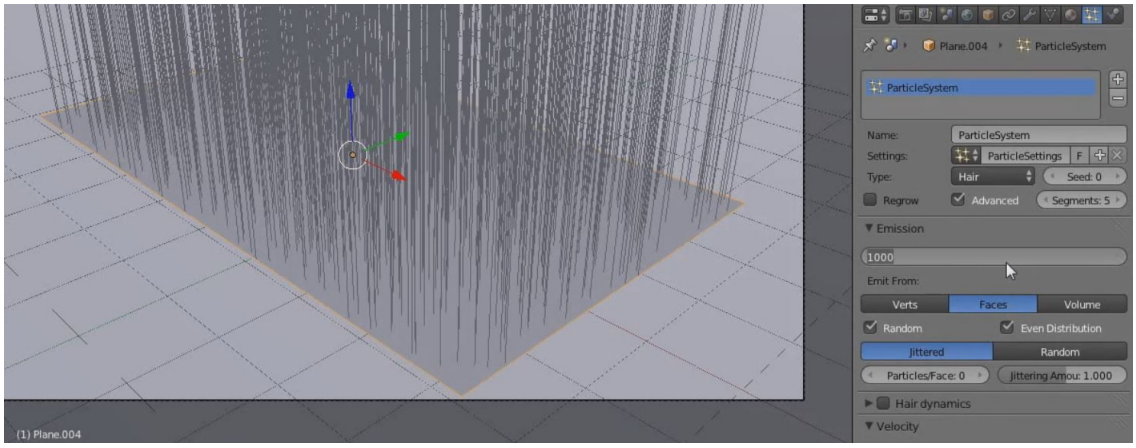
Il lavoro si divide in due fasi: le impostazioni del sistema particellare *HAIR* e impostazioni di Materials e Rendering.

Prima di procedere, impostiamo ovviamente il motore di rendering in Cycles e, nella scheda Render, nella Properties Window, scegliamo CPU ed Experimental, in quanto dovremo usare una funzionalità (*Cycles Strand Renderer*) disponibile, attualmente, solo con questa combinazione.

*CPU Experimental, in Cycles*

Iniziamo con la prima fase: le impostazioni geometriche del sistema particellare.

Con il Plane selezionato, passiamo alla scheda Particle System, nella Properties Window, ed aggiungiamo un sistema *HAIR* (letteralmente “capelli”) alla mesh (attenzione perché di default un sistema particellare appena creato è di tipo *EMITTER*).

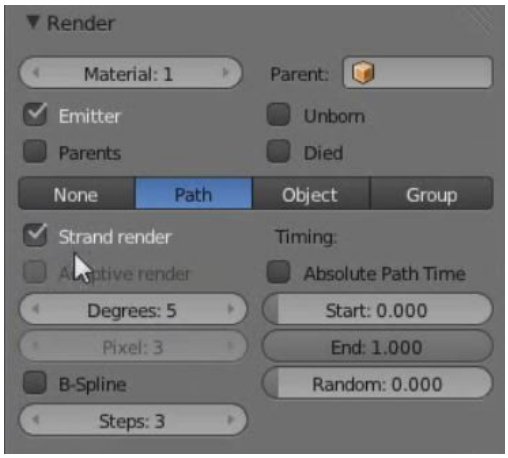



*Il sistema particellare HAIR per il Plane*

Attiviamo la checkbox *Use Advanced*, quindi impostiamo ad esempio 500 particelle; possono sembrare un po' poche, ma presto useremo i *Children* per aumentarle mantenendo bassa la complessità di modellazione e, in ogni caso, tutto dipende anche dalla scala e dalle dimensioni della scena, quindi questo è solo un valore d'esempio, non valido per tutte le stagioni!

Altro valore d'esempio, da cambiare in base alla situazione, è il *Normal* dell'Emitter, che definisce la lunghezza delle particelle; qui, mettiamo ad es. 0.01 o 0.02 (al posto del valore di default 1.0).

Spostiamoci ora nella sezione Render della scheda del sistema particellare: selezioniamo per prima cosa *Strand Render*, per il rendering su particelle *HAIR*, quindi impostiamo ad esempio End 0.9 e Random 0.2; i parametri più interessanti, comunque, si trovano nella sezione Children, letteralmente “figli”, da attivare con modalità *Interpolated*.




*Strand Render*

L'opzione Children consente di generare, per ogni particella (attualmente, ne abbiamo 500) altre particelle, “figlie”, appunto, utilizzando poche risorse per la modellazione interattiva.

Specifichiamo ad esempio 200 *Children* in Render: significa che per ogni particella delle 500 originali ne verranno create altre 200, quindi, 100.000 (in *Children View* manteniamo un valore molto più basso, ad esempio 10, così nel nostro caso verranno visualizzate appena 5.000 particelle, non rallentando troppo il sistema in questo momento e lasciando i calcoli definitivi per il Render).

Le particelle Children vengono create intorno alla particella originale e qui, per definire la loro forma, entrano in gioco gli altri parametri della scheda; ad esempio, con *Clamp* possiamo modellare i "ciuffi" (io sto mettendo 0.05; provate vari valori e osservate i cambiamenti in tempo reale nella 3D View: sono molto intuitivi e in effetti è meglio guardarli direttamente che descriverli a parole); poi, ad esempio, Shape 0.5, Length 0.9, modalità Radial, Amplitude 0.01... gli altri parametri sono trascurabili.

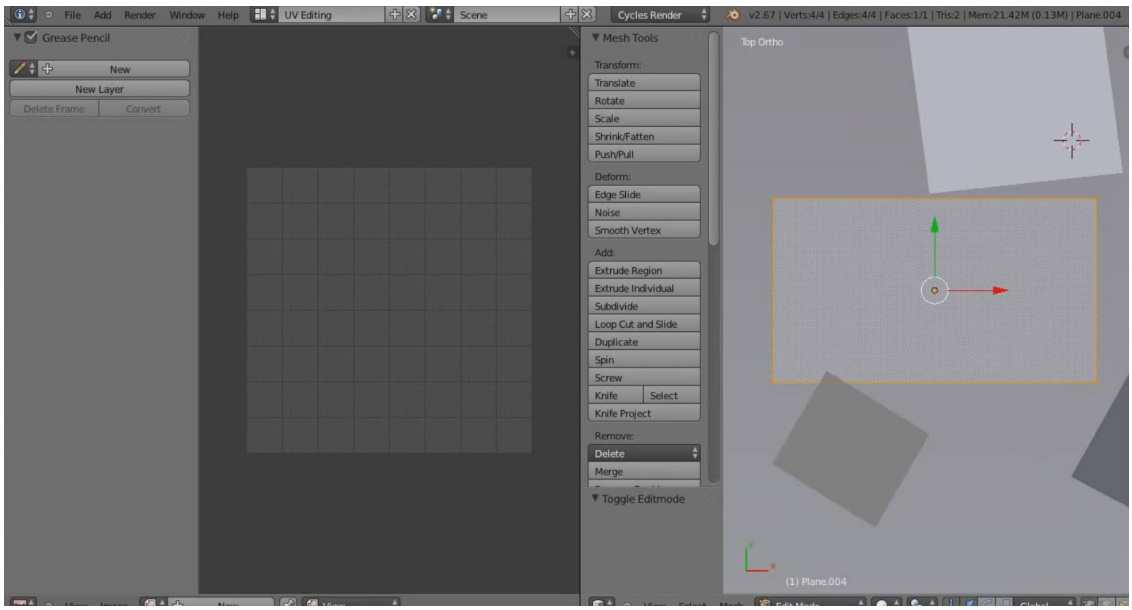
Nella sezione *Cycles HAIR Strand Render*, disponibile con *CPU Experimental*, impostiamo il metodo *True Normal* (*Smooth Curves* è, in realtà, il metodo migliore, ma è anche più lento e conviene usarlo solo per dettagli e macro); mettiamo poi *Shape* (la thickness, lo spessore lungo il pelo) ad es. -0.9, *root thickness* (spessore alla radice) 0.1, *tip* (spessore in punta) 0.02 e selezioniamo l'opzione *close tip*.

*Parametri di Cycles Hair Strand Render*

Le particelle ci sono, ma sono decisamente "troppo ordinate": per... "agitarle" un po', nella sezione *Physics* aumentiamo leggermente il valore di *Brownian*, portandolo ad esempio a 0.03 (nelle particelle *Emitter* questo parametro si riferisce al moto browniano delle particelle nel mondo reale; nel caso di *Hair*, è un fattore di casualità nella forma dei filamenti).

Adesso, seconda e ultima parte: parliamo del Material e del rendering!

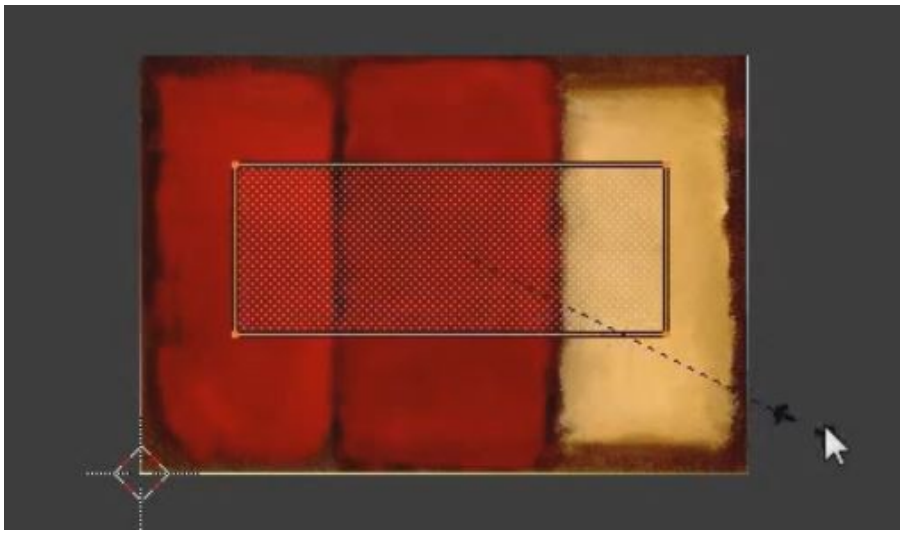
In realtà, per il tipo di tappeto che vogliamo ottenere, è sufficiente un semplice Shader Velvet, per cui scegliamo questa opzione e passiamo a un Layout *UV Editing* per unwrap e texturing del Plane.



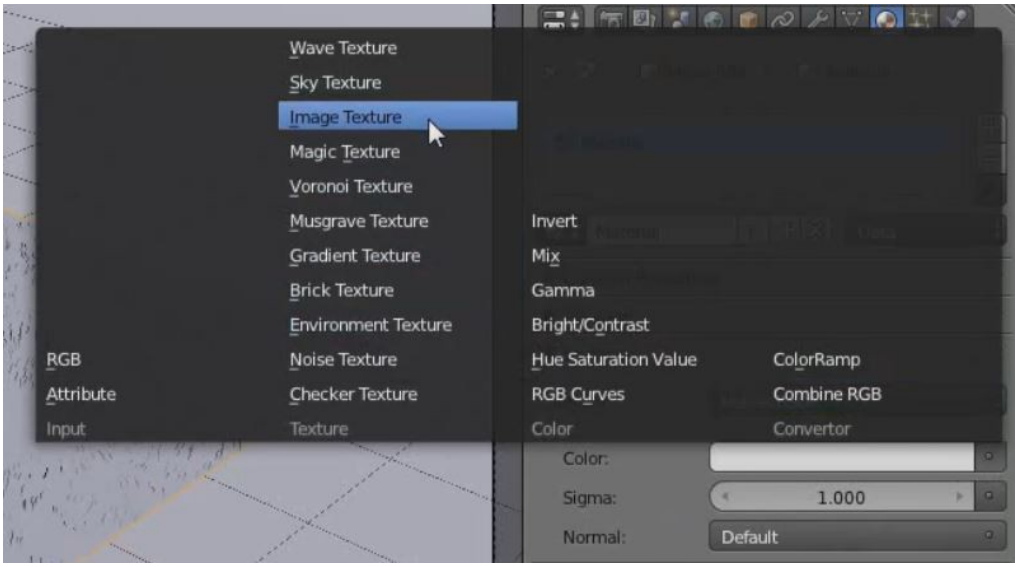
*Layout UV Editing per unwrappare e texturizzare il Plane*

Con il Plane selezionato passiamo in Edit Mode, selezioniamo tutto e facciamo, dalla vista Top Ortho (per inquadrare bene il Plane dall'alto) un unwrap (tasto U) in modalità "*Project From View*", proietta dalla vista corrente, ottenendo, nella finestra UV, la scucitura.

Nella finestra UV carichiamo quindi l'immagine da utilizzare e posizioniamo e ridimensioniamo la scucitura per adattarla all'immagine (potete selezionare, traslare, ruotare e scalare vertici e spigoli della scucitura come con vertici e spigoli di una mesh, quindi click destro, con SHIFT o B per selezioni multiple, e G, R ed S).

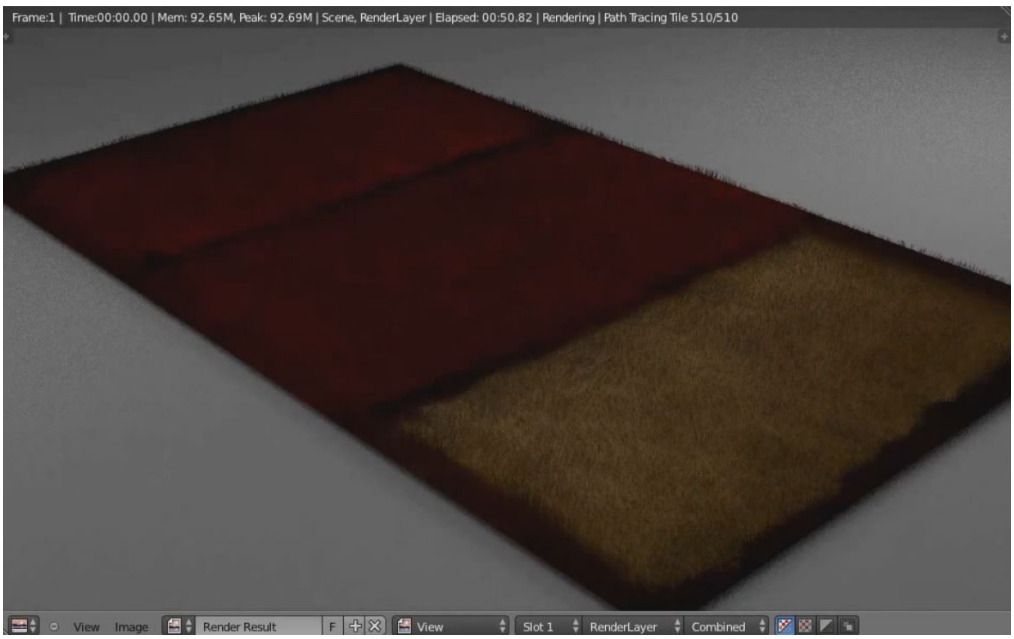

*Mappatura UV della Texture*

Questa però è solo la scucitura della mesh sull'immagine: non abbiamo applicato, ancora, l'immagine allo Shader della mesh; clicchiamo quindi sul quadrato col cerchietto accanto a Velvet, nella scheda del Material, e scegliamo *Image Texture*, specificando la stessa immagine utilizzata in fase di scucitura.




*Creazione di una Image Texture per il Material dell'oggetto*

Fatto quindi il sistema particellare, l'UV-unwrap e il Material, possiamo avviare il rendering ed osservare il risultato finale.




*Il rendering finale*

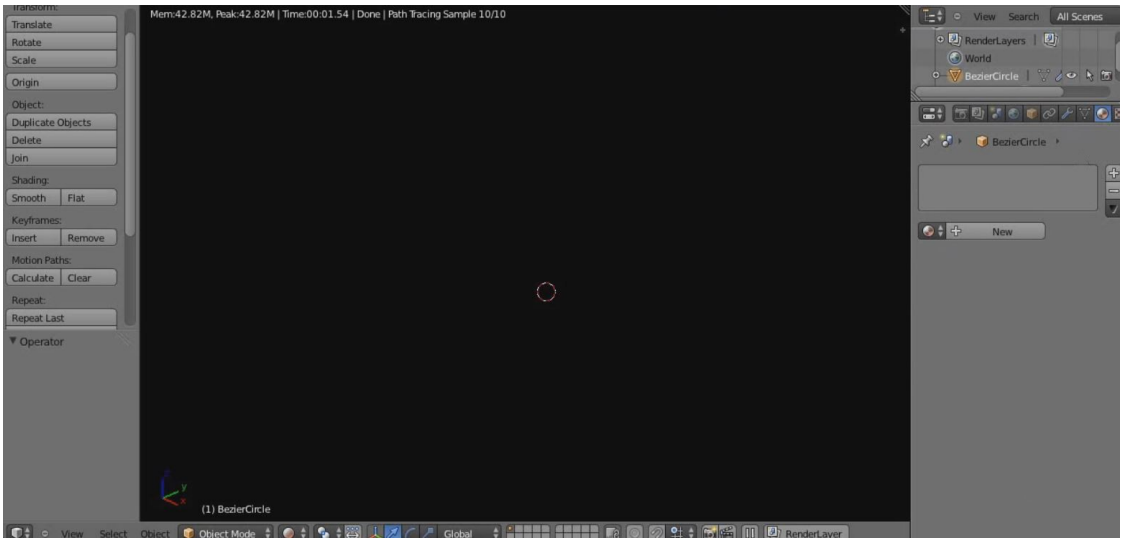
*  *  *

# Mappe ambientali su Reflection per rendering veloci in Cycles

In questo tutorial vedremo come applicare una texture ambientale *Equirectangular* direttamente su un oggetto per un'illuminazione e un rendering "da studio" di oggetti, senza ambienti circostanti, veloci ma interessanti, in Blender 3D e Cycles (con un tocco di GIMP).

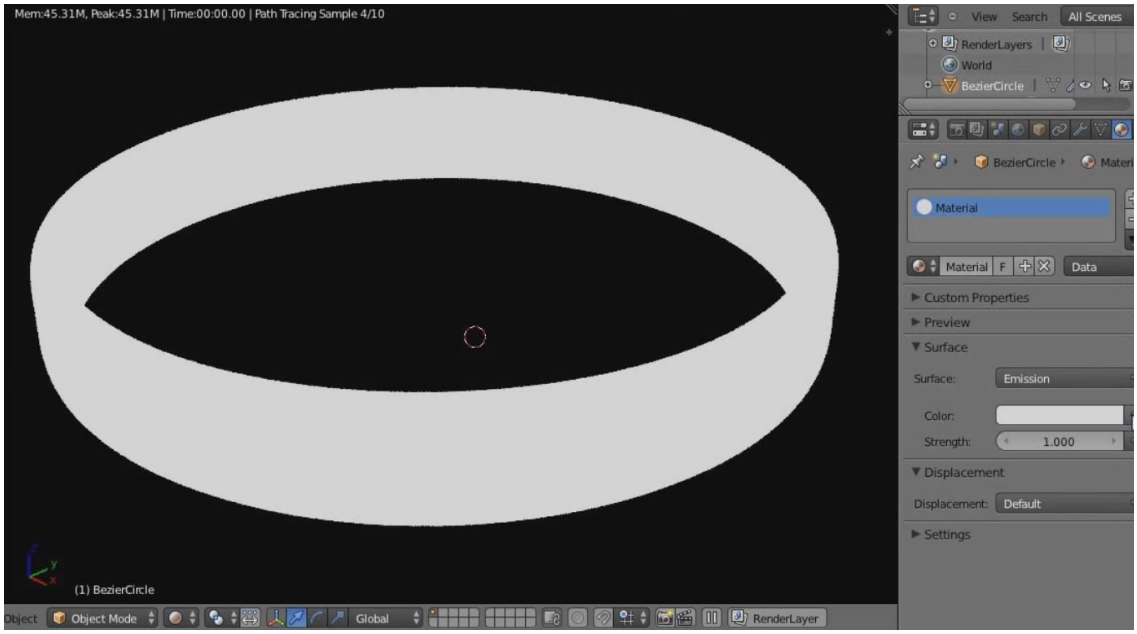
Si tratta, in particolare, di una variante dell'utilizzo delle mappe ambientali, applicate però in maniera particolare direttamente all'oggetto, senza dover coinvolgere il World Background.

Per prima cosa, andiamo in modalità visualizzazione *Rendered*: nella scena non ci sono fonti di luce e stiamo peggiorando la situazione impostando a nero il *Background Color*, nella scheda World, per cui... non si vede niente.



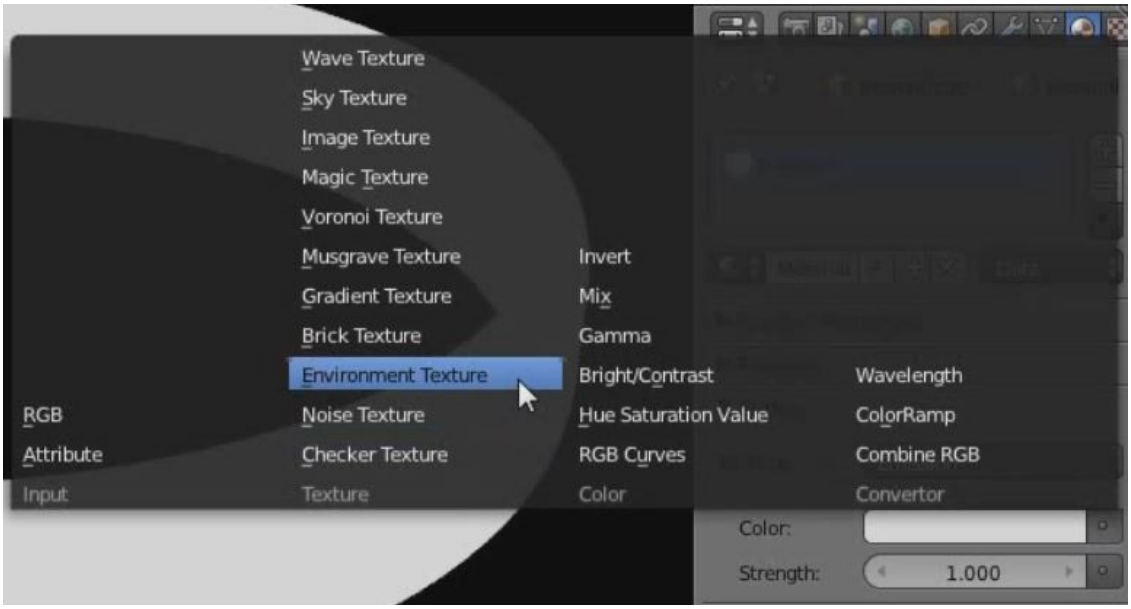

*Preview Rendered: non si vede niente*

Il trucchetto è questo: nella scheda del Material, aggiungiamo uno Shader *Emission*, emissione luminosa, che di default sarà grigio chiaro, per cui vedremo la silhouette dell'oggetto...



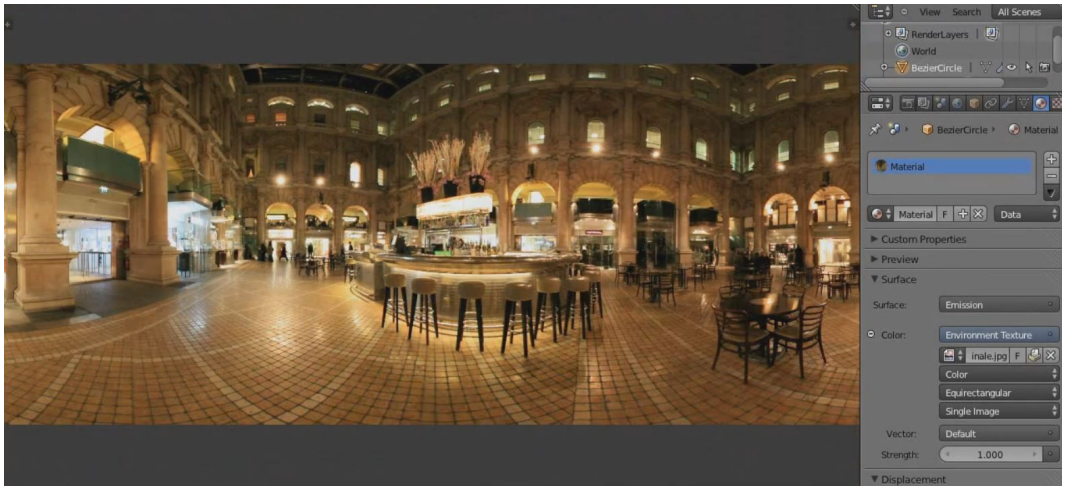

*Un Material EMISSION per l'oggetto*

… cliccando sul quadratino accanto al colore, quindi, scegliamo una *Environment Texture*, una texture "ambientale", specificando quindi un file immagine.



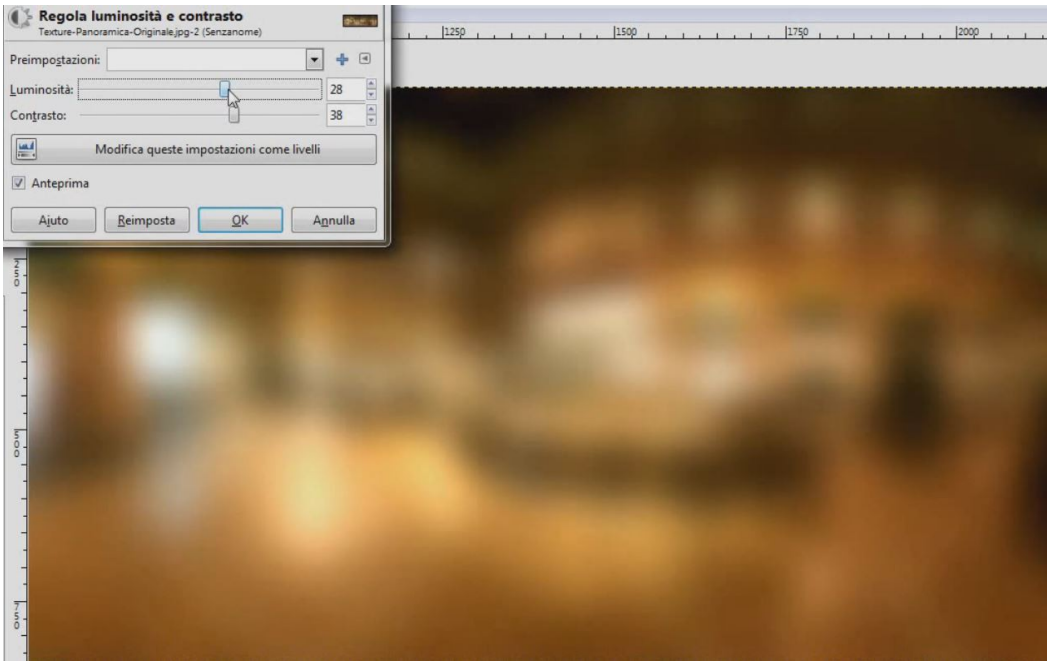

*Una Environment Texture per l'oggetto*

L'immagine utilizzata è rettangolare (un'immagine ambientale, *Environment*, in modalità *Equirectangular*, ad alta risoluzione) ma per i nostri scopi è fin troppo definita: utilizzeremo quindi GIMP per sfocarla e cambiarne il contrasto, prima di ri-applicarla all'oggetto e cambiarne la mappatura.



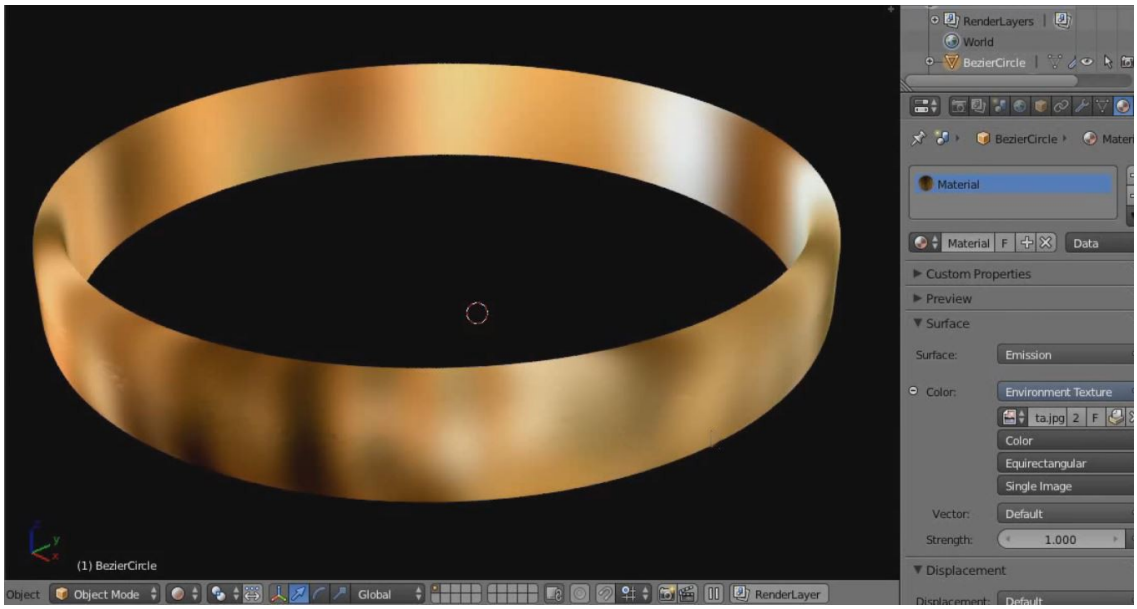

*L'immagine utilizzata è fin troppo definita*

In GIMP, o comunque in qualsiasi programma di fotoritocco, è sufficiente utilizzare un filtro di *sfocatura gaussiana*, con un valore molto alto, in modo da confondere gli oggetti, quindi possiamo procedere ad altre correzioni come correzione curve colore, contrasto, luminosità, eccetera; qui sto usando questa immagine e questi colori (con valori a caso per luminosità e contrasto) per via del tipo di materiale, oro, che vogliamo ottenere, ma ovviamente per altri tipi di oggetti (ad esempio, cromature) potrebbero essere indicate immagini panoramiche di ambienti esterni, eccetera.
Nessuno vi vieta inoltre di fare un po' di correzione colore sull'immagine originale o miscelare due o più Shaders, insieme all'Emission, in Blender; questo è solo un esempio.



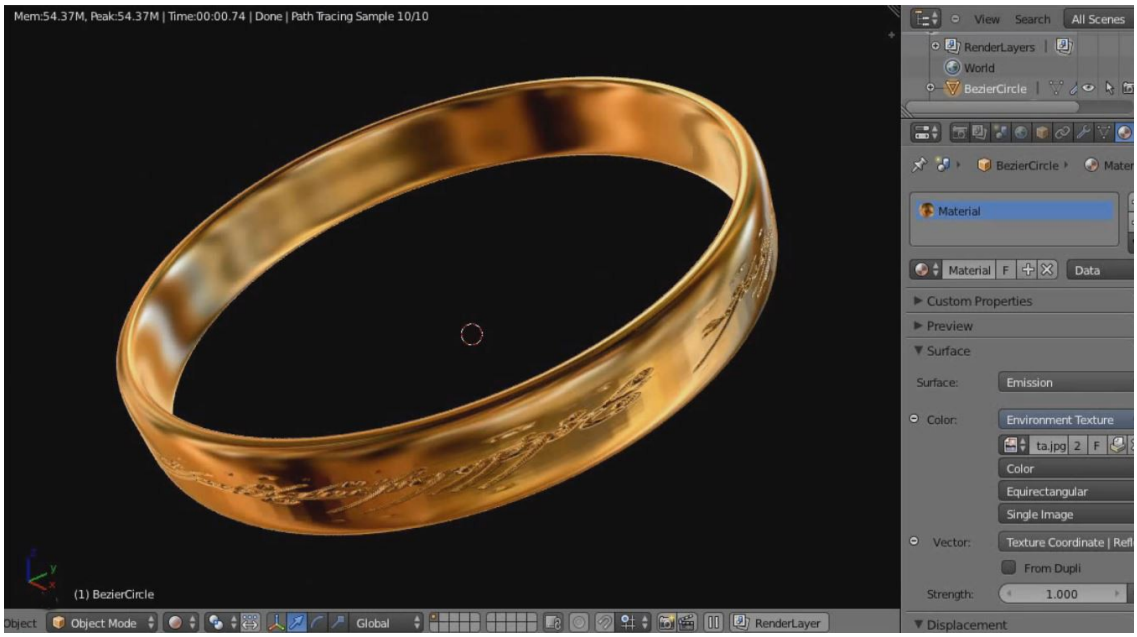

*Modificare l'immagine texture in GIMP*

Tornati in Blender, quindi, utilizziamo l'immagine ritoccata come Environment Texture applicata all'oggetto: qualcosa è cambiato, è vero, ma ancora non ci siamo...




*La nuova texture immagine, in Blender*

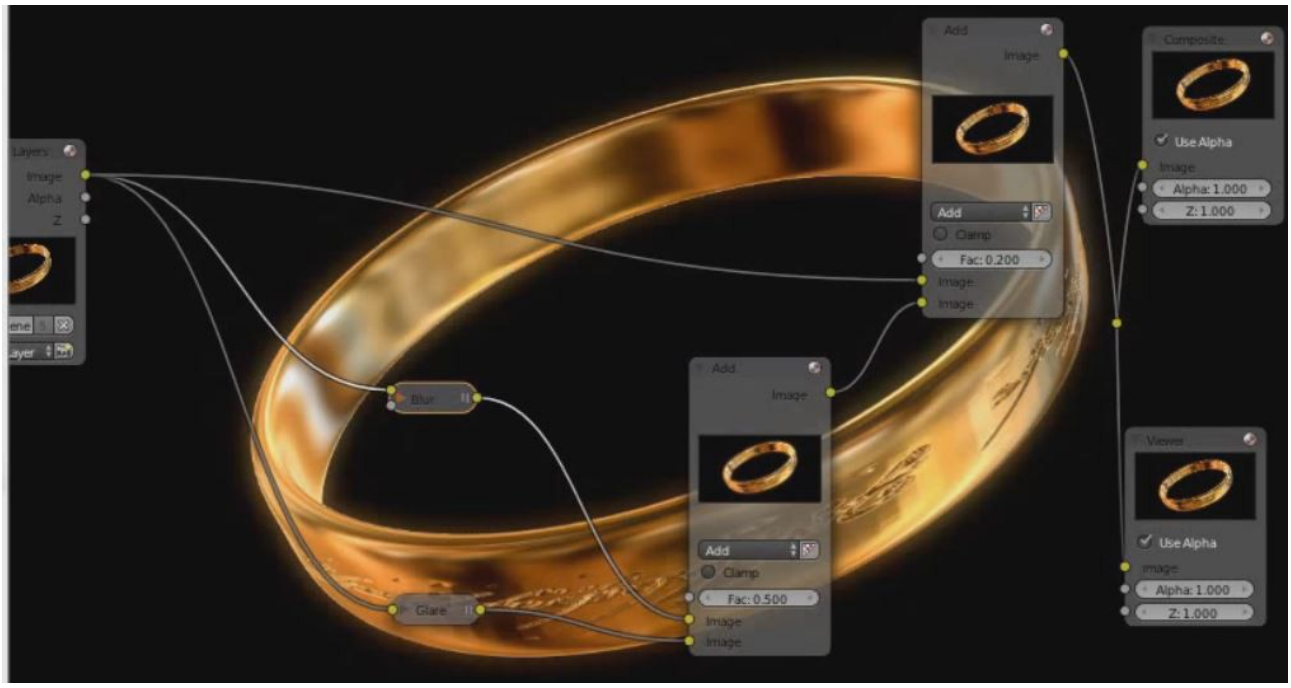
Il vero trucchetto sta nel cambiare il *Vector* della mappatura dell'immagine, nello Shader Emission, da *Default* a *Reflection*: in questo modo, sarà l'oggetto a simulare quello che avviene quando si applica una mappa *Environment Equirectangular* applicata a World Background, "incorporando" per così dire la riflessione della Texture immagine nello Shader.




*Con modalità di mappatura Vector REFLECTION*

Da notare che in questo modo le incisioni, prodotte mediante una Texture e un modificatore Displace, sono di nuovo visibili, mentre prima non lo erano.

Adesso sto andando in una finestra Nodes Compositing per aggiungere un po' di bagliore soft glow (con il *Fast Gaussian Blurring*) e hard glow (con il nodo *Glare*) al rendering finale, tuttavia non è questo l'argomento del tutorial, che riguardava invece l'utilizzo dell'*Equirectangular* direttamente sull'oggetto con *Emission* e soprattutto *Reflection*.



*Schema di Nodi per gli effetti GLOW in Compositing*

Ne approfitto per dirvi che al seguente link:

*http://www.francescomilanese.com/cg-3d/blender-3d/79-blender-3d-cycles-compositing-mappe-ambientali-su-reflection-per-rendering-veloci.html*

potete scaricare gratuitamente e senza registrazione il *pacchetto zip* contenente le immagini textures e due scene *blend* utilizzate in questo tutorial (quella iniziale e quella che sto producendo, completa quindi di schema di Nodi di Compositing per la post-produzione), in modo da esaminarle con calma (in particolare il Nodes Editor) e per ricreare personalmente quanto visto, magari modificando qualcosa...

… ad esempio, *Displace* introduce delle sfaccettature rettangolari sul bordo esterno, altrimenti l'oggetto sarebbe liscio come la parte interna, per cui un esercizio che potete provare per conto vostro consiste nel modellare un anello liscio, senza rilievi, per poi applicare la texture della scritta (che trovate nel pacchetto zip scaricabile in alto) come *Normal Map*, non come *Displacement*; ossia: modellate l'anello liscio, senza *Displacement*, e mappategli come *Normal Map* la scritta, per verificare le differenze e valutare il risultato.

Visto che resta un po' di tempo, due parole sullo schema di Nodi: per una trattazione approfondita su cos'è il Compositing, cosa sono i Nodi e quali sono le loro funzioni e caratteristiche, rimando all'ebook sul Nodes Compositing in Blender, che trovate nella sezione ebook del mio sito web; in questa sede, invece, diciamo che sto aggiungendo un nodo *Glare*, regolando in particolare la soglia Threshold e la qualità, per aggiungere dei "raggi di luce", un effetto Hard Glow, mentre per un effetto Soft Glow sto aggiungendo un Nodo *Blur*, la sfocatura, in modalità *Fast Gaussian*, per ottenere una maschera sfocata dell'anello, che farà da *Soft Glow*.

Le due immagini elaborate, quindi, vengono sommate in cascata e all'immagine originale, in arrivo da *Render Layers*, mediante due Nodi *Mix* in modalità Add, dopodiché si lavora appunto sulla modalità e sul Factor, il fattore di influenza dei nodi, regolando l'intensità degli effetti.

*Il rendering finale senza Compositing...*

Al termine, basterà premere F12 per avviare il rendering finale.

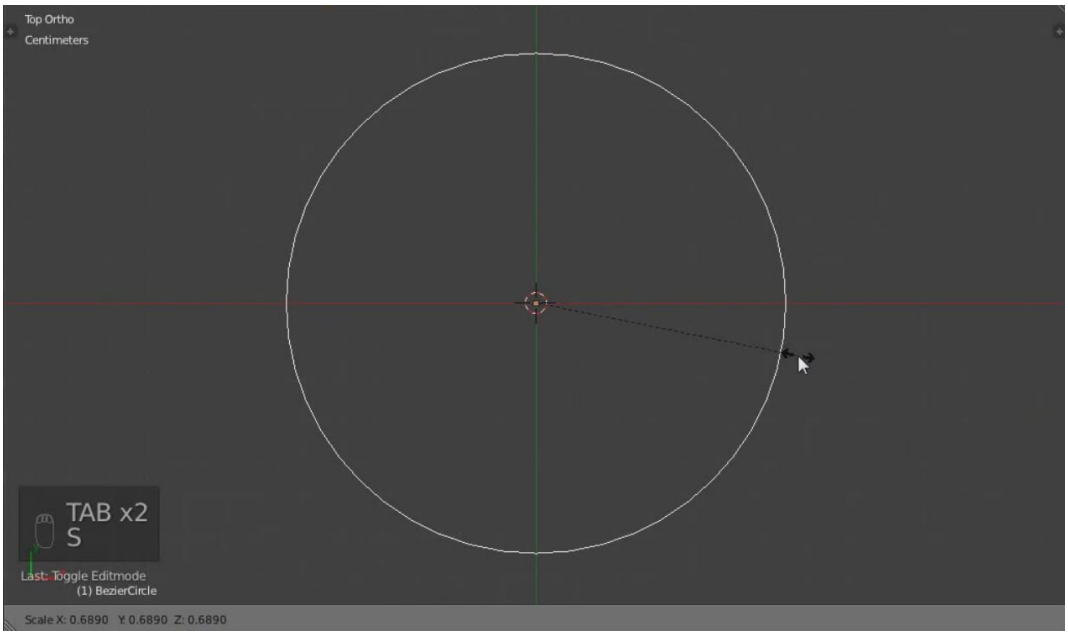
*... e il rendering finale con effetti di Compositing in Post-Produzione*

*  *  *

# Modellare una punta da trapano con Bezier Circle, Screw, Bridge

In questo breve tutorial di base vedremo come modellare una punta da trapano in Blender 3D utilizzando il modificatore *Screw* e lo strumento *Bridge&Loft*. Il tutorial è stato realizzato con la versione 2.68 di Blender 3D.
Prima di iniziare, pianifichiamo le operazioni: inizieremo modellando la sezione della punta da trapano; dopo, mediante Screw otterremo l'estrusione con rotazione (creando quindi la mesh della punta vera e propria) e infine ci occuperemo della parte cilindrica e della punta, nel primo caso mediante Bridge&Loft e nel secondo con dei semplici scaling.

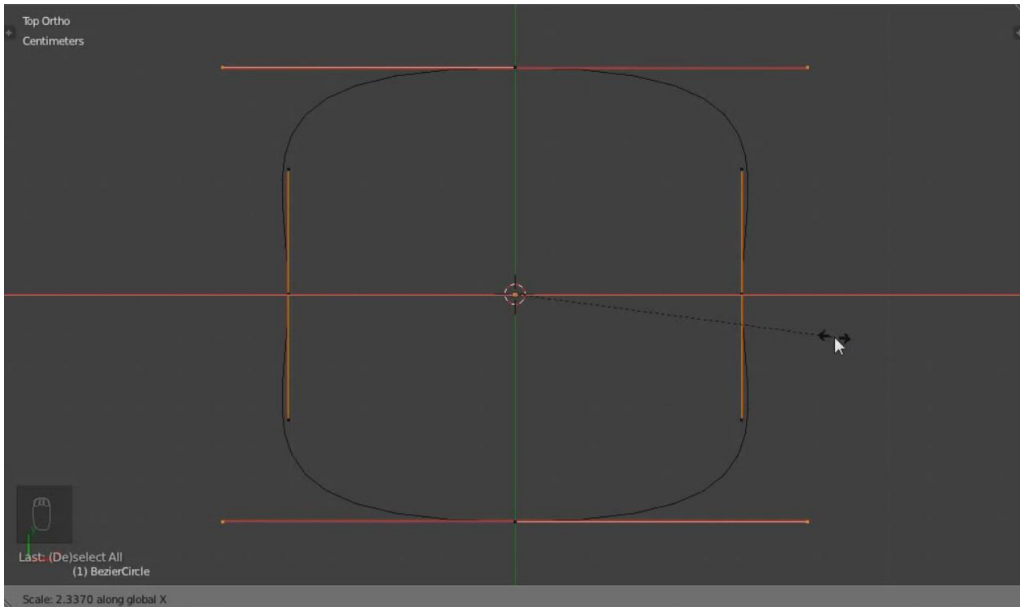
Iniziamo, in una vista TOP ORTHO, con la definizione del profilo, inserendo un Bezier Circle, che è un oggetto di tipo Curve con 4 punti di controllo, come si può verificare andando in Edit Mode: 4 punti di controllo ciascuno con 2 maniglie, per definire l'interpolazione tra i punti.




*La Bezier Circle iniziale*

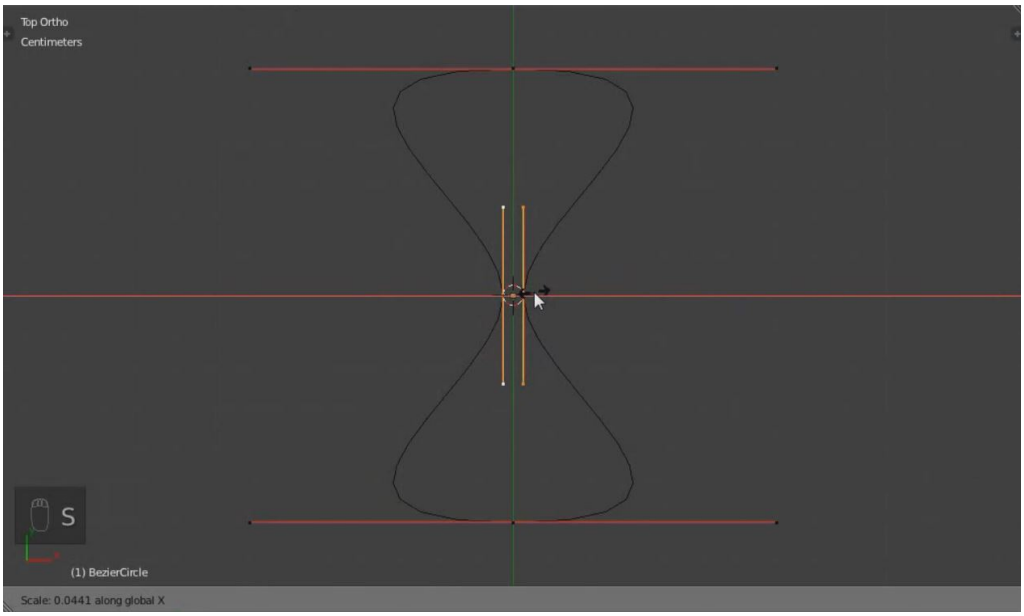
Qui non seguirò dei progetti tecnici precisi, tuttavia se li avete potete utilizzarli come Background Image e modellare il Bezier Circle ricalcando l'immagine.

Nel mio caso, sto selezionando solo le due coppie di maniglie dei due punti di controllo superiore e inferiore (attenzione: seleziono solo le maniglie, non i punti di controllo) e, mediante S X e il mouse, li sto allontanando: in pratica, con *S* entro in modalità scaling, mentre con *X* vincolo lo scaling al solo asse orizzontale della scena che, nel mio caso, è il verso orizzontale, per cui l'effetto finale è di scalare la selezione orizzontalmente, allontanando i punti.




*Modellazione della curva, in Edit Mode*

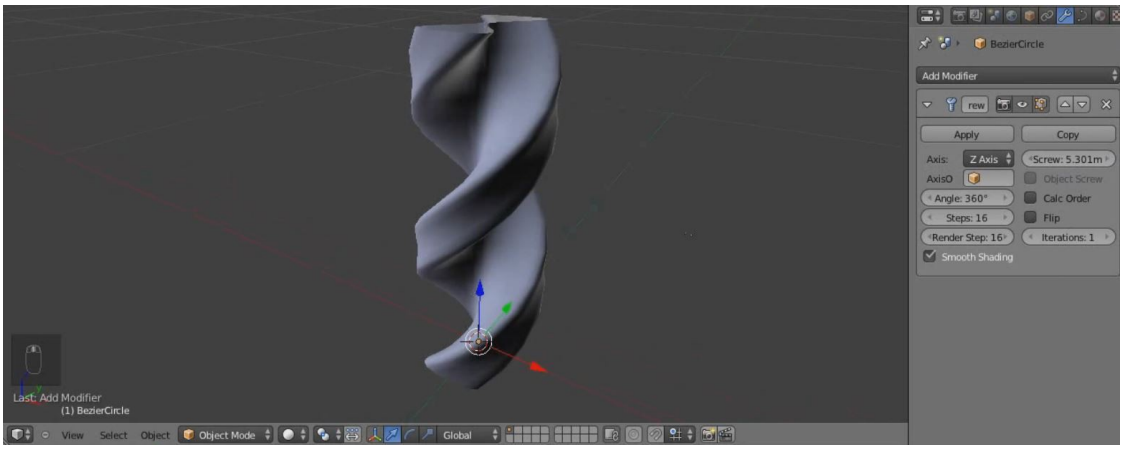
Con lo stesso principio, ma scalando verso l'interno (e, quindi, avvicinandoli), seleziono questa volta sia i punti di controllo che le maniglie al centro e, con S X e il mouse, li avvicino, ottenendo la figura visibile nell'immagine seguente.




*La forma finale*

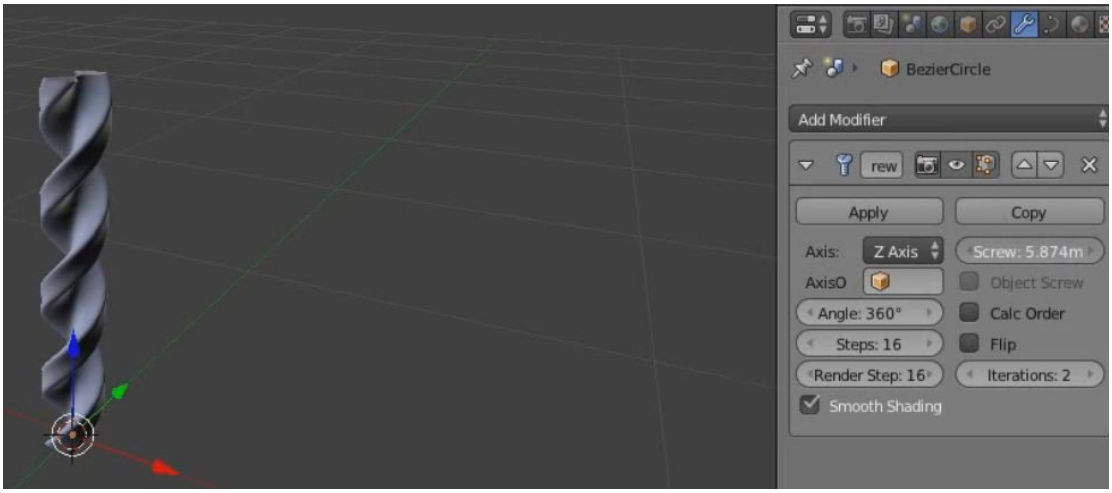
A questo punto torno in modalità oggetto e, nella scheda Modifiers, all'interno del pannello Properties Window, seleziono un modificatore Screw, letteralmente “avvita”, appunto.

Il risultato iniziale è deludente (è una specie di cerchio), ma solo perché l'avvitamento è stato eseguito senza profondità: aumentiamo il valore del campo “*Screw*” e...




*Utilizzo di Screw (aumentando il valore di "Screw", appunto)*

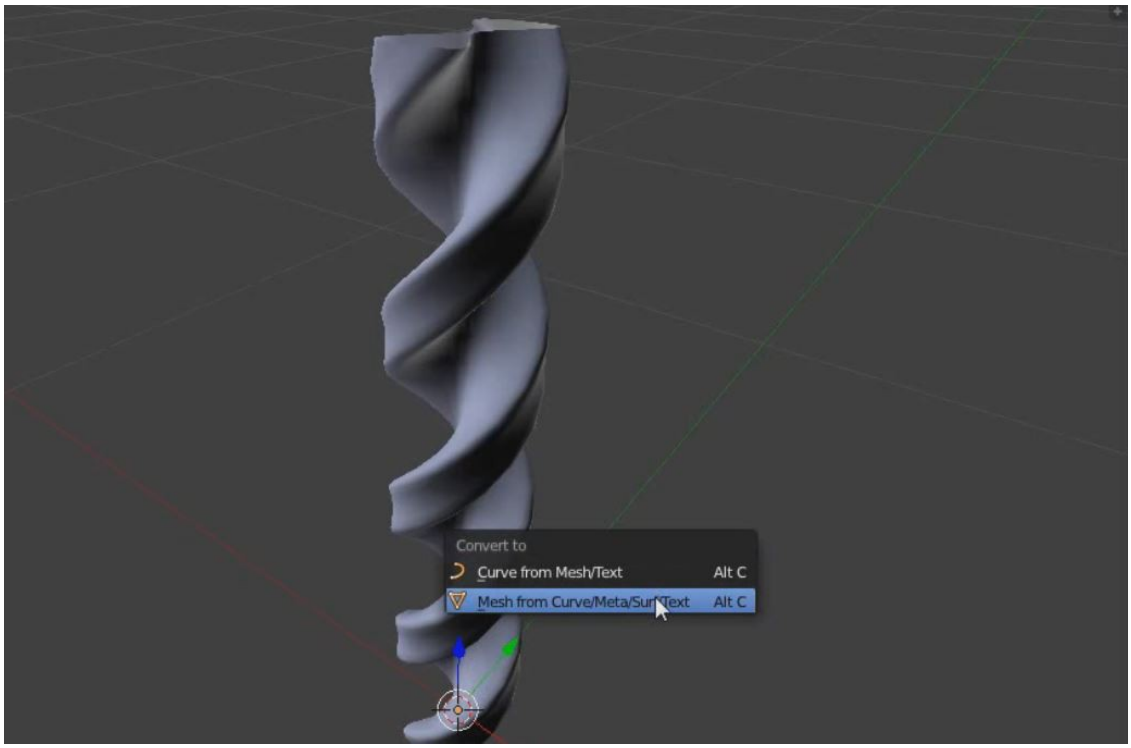
… ecco la forma che volevamo! Variamo quindi questo parametro a piacere, sistemando la qualità con *Steps* e il numero di ripetizioni con *Iterations*.




*Variare Screw, Steps e Iterations nella scheda del modificatore*

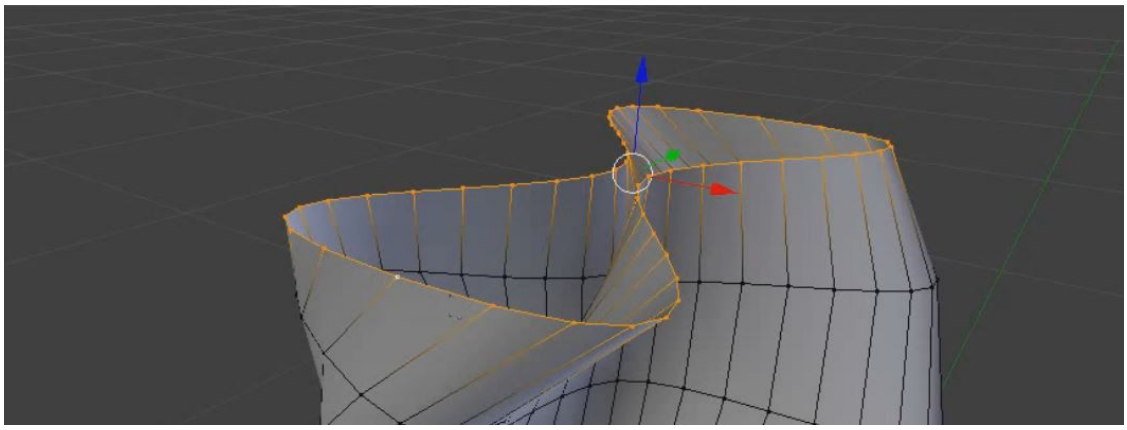
Definita questa porzione di mesh siamo già a metà dell'opera: dobbiamo sistemare la punta e la parte cilindrica... prima, però, dobbiamo applicare il modificatore per ottenere la mesh.

Non è possibile applicare Screw ad un oggetto di tipo Curve, come il Bezier Circle, per cui (con l'oggetto selezionato in Object Mode) premiamo ALT C e scegliamo “Mesh from Curve”.




*Conversione della curva in mesh (dopo aver applicato il modificatore)*

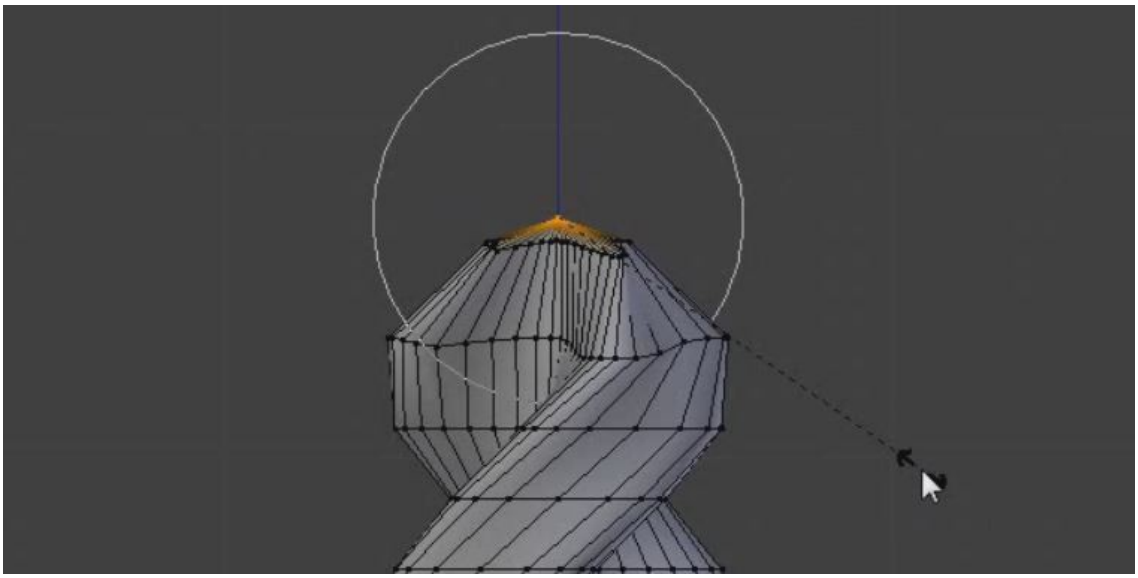
Adesso andiamo in Edit Mode e selezioniamo l'anello superiore di spigoli (Edge Loop) con ALT e Click destro su uno spigolo.



*Selezionare l'anello di spigoli superiore*

L'idea è quella di scalare la selezione a 0, facendola finire in un punto, ma per applicare l'effetto anche a un paio di Edge Loop successivi attiviamo, nell'header della 3D View, il Proportional Editing (shortcut: O), che andrà regolato con la rotellina del mouse (per visualizzare meglio i risultati conviene andare in una vista frontale o laterale in modalità ORTHO).

Con l'anello superiore selezionato e il Proportional Editing attivo, quindi, premiamo S 0, ma prima di premere Invio per confermare ruotiamo la rotellina del mouse in modo da includere, in maniera proporzionale, anche altri Edge Loop.

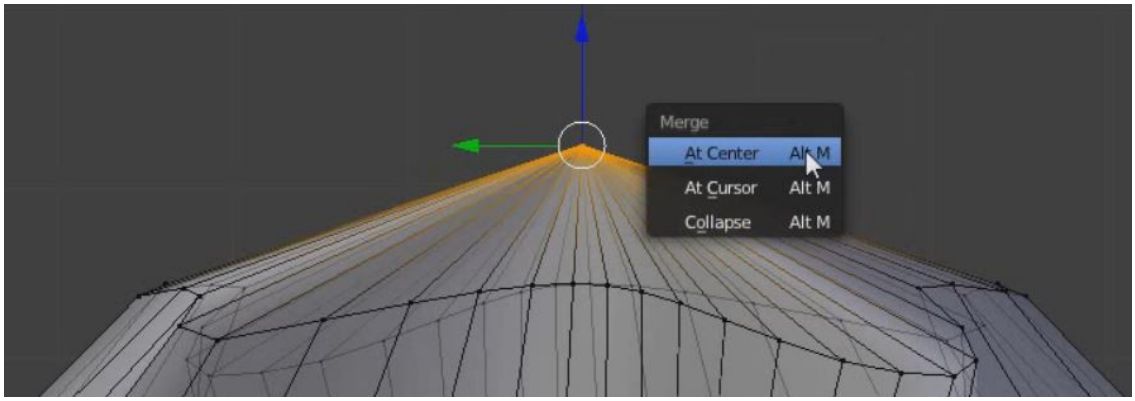


*Scaling a 0 con il Proportional Editing*

Quando il risultato ci convince, premiamo Invio per confermare.

Disattiviamo il Proportional Editing (di nuovo, shortcut: O), nell'header della finestra.

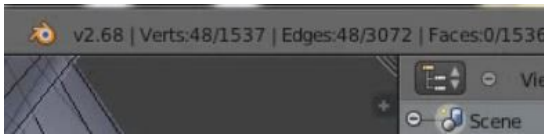
In realtà, la punta così ottenuta non è proprio "chiusa" e questo potrebbe essere un problema per la gestione delle geometrie da parte di Blender; selezioniamo quindi i vertici che la compongono (semplicemente, con B per la selezione rettangolare), premiamo W e, dal menù *Specials* che apparirà a video, selezioniamo *Merge – At Center*.




*Merge - At Center, per i vertici degli spigoli della punta*

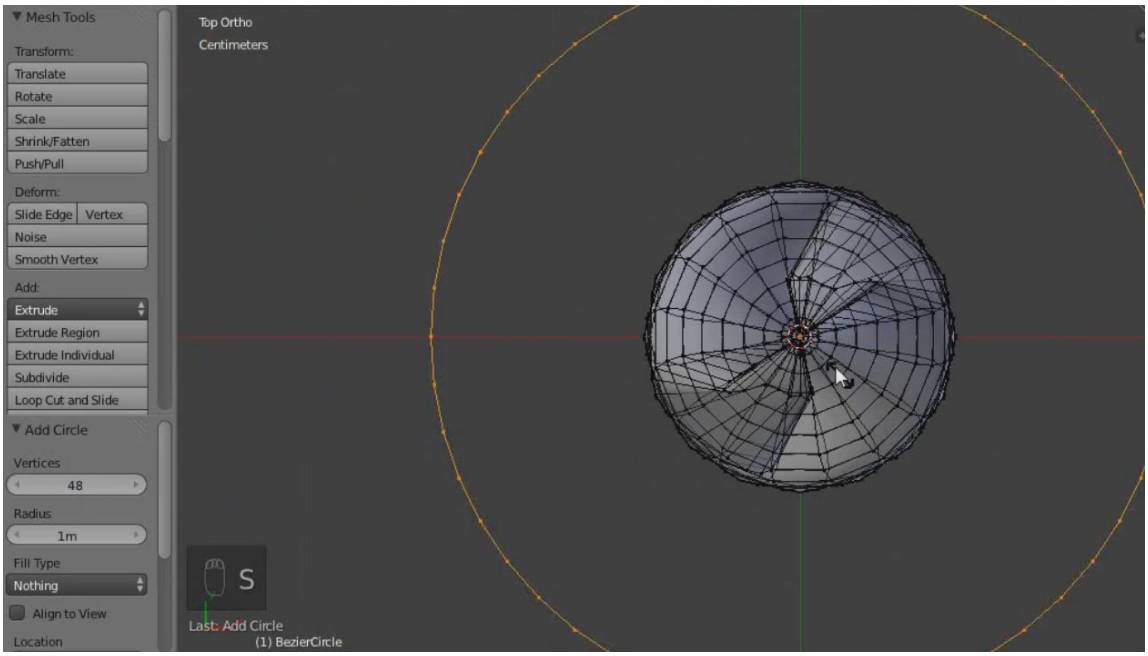
Passiamo ora alla base inferiore.

Per passare dalla forma della punta di trapano a un cerchio (sezione della base cilindrica), useremo lo strumento Bridge che prende due anelli di spigoli (Edge Loop) con lo stesso numero di vertici ed effettua le connessioni. In Edit Mode, quindi, selezioniamo l'Edge Loop della base con ALT e click destro su uno spigolo, e vediamo, in alto nella finestra di Blender, alla voce "Verts", quanti sono i vertici.



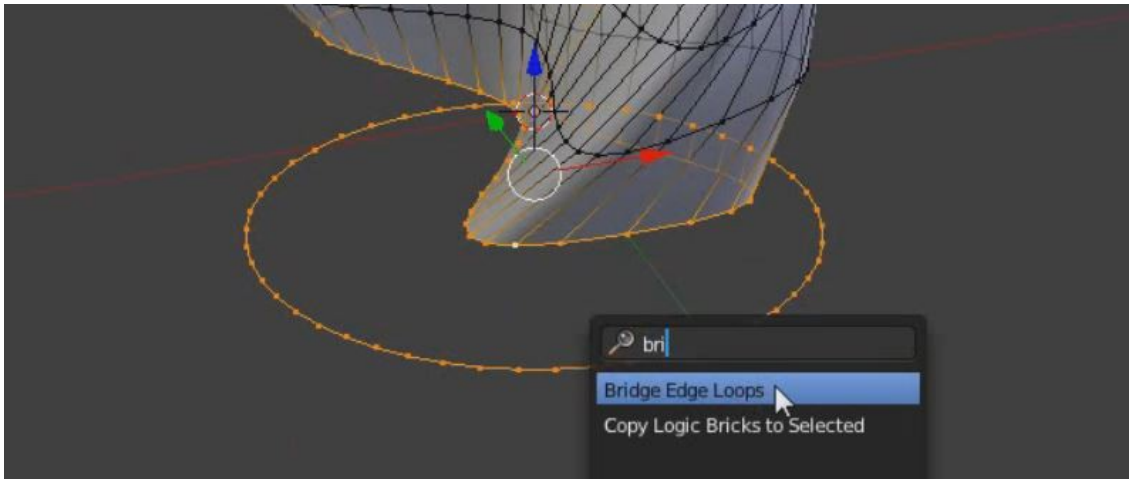

*Numero di vertici selezionati in Edit Mode, nella sezione INFO*

Sempre in Edit Mode, aggiungiamo una Mesh Circle, che farà quindi parte dell'oggetto originale, impostando lo stesso numero di vertici dell'Edge Loop, digitando questo valore nella Tool Shelf, in basso a sinistra, quando si aggiunge la mesh.



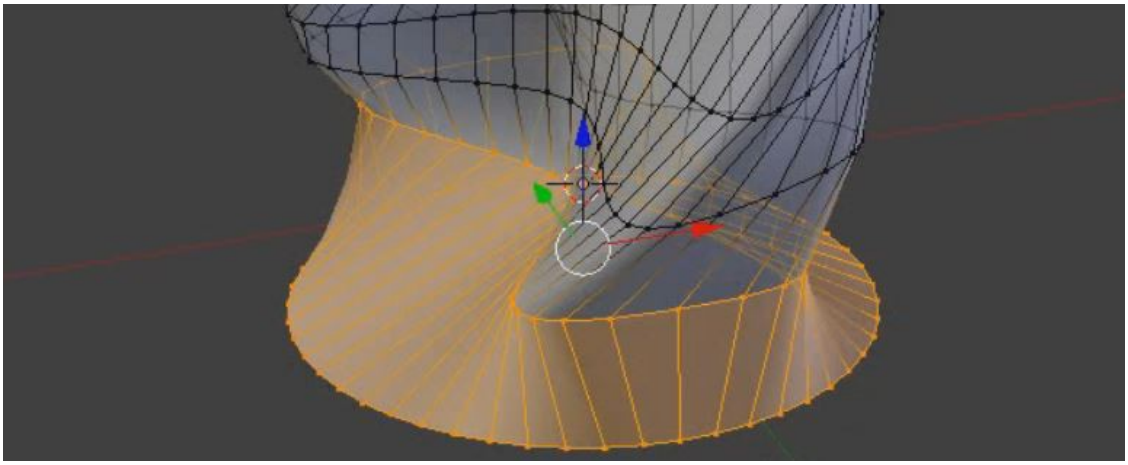

*Inserimento della mesh Circle, in Edit Mode, con il giusto numero di vertici*

Scaliamo questo cerchio per dargli il giusto diametro e posizioniamolo sotto l'Edge Loop; selezioniamo anche questo elemento (con una selezione multipla, da fare con SHIFT e ALT e click destro), quindi premiamo la barra spazio, digitiamo Bridge e selezioniamo lo strumento Bridge Edge Loops, per effettuare il collegamento: fatto!



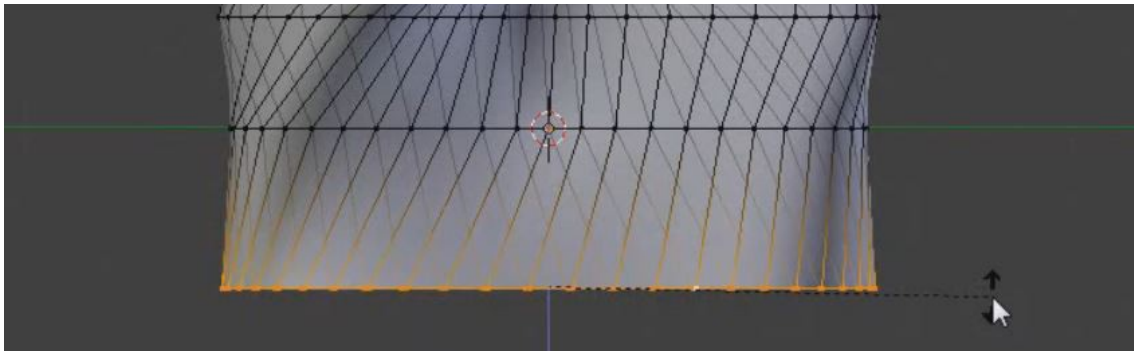

*Effettuare il collegamento con BRIDGE*

Il risultato potrebbe essere deludente, per un semplice motivo: le connessioni tendono a essere verticali, dritte, ignorando la curvatura della punta.



*Risultato dell'applicazione di BRIDGE*

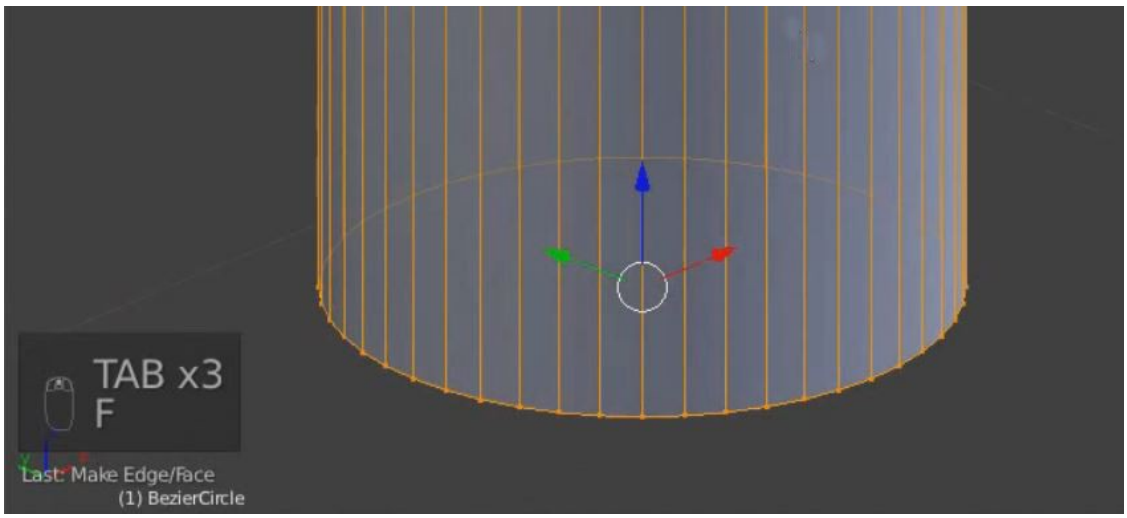
Per risolvere il problema, selezioniamo l'Edge Loop circolare alla base e ruotiamolo un po' intorno all'asse verticale, con R Z. Al termine, possiamo estruderlo verso il basso per ottenere la parte cilindrica della punta da trapano.



*Sistemare l'Edge Loop alla base, ruotandolo intorno a Z*

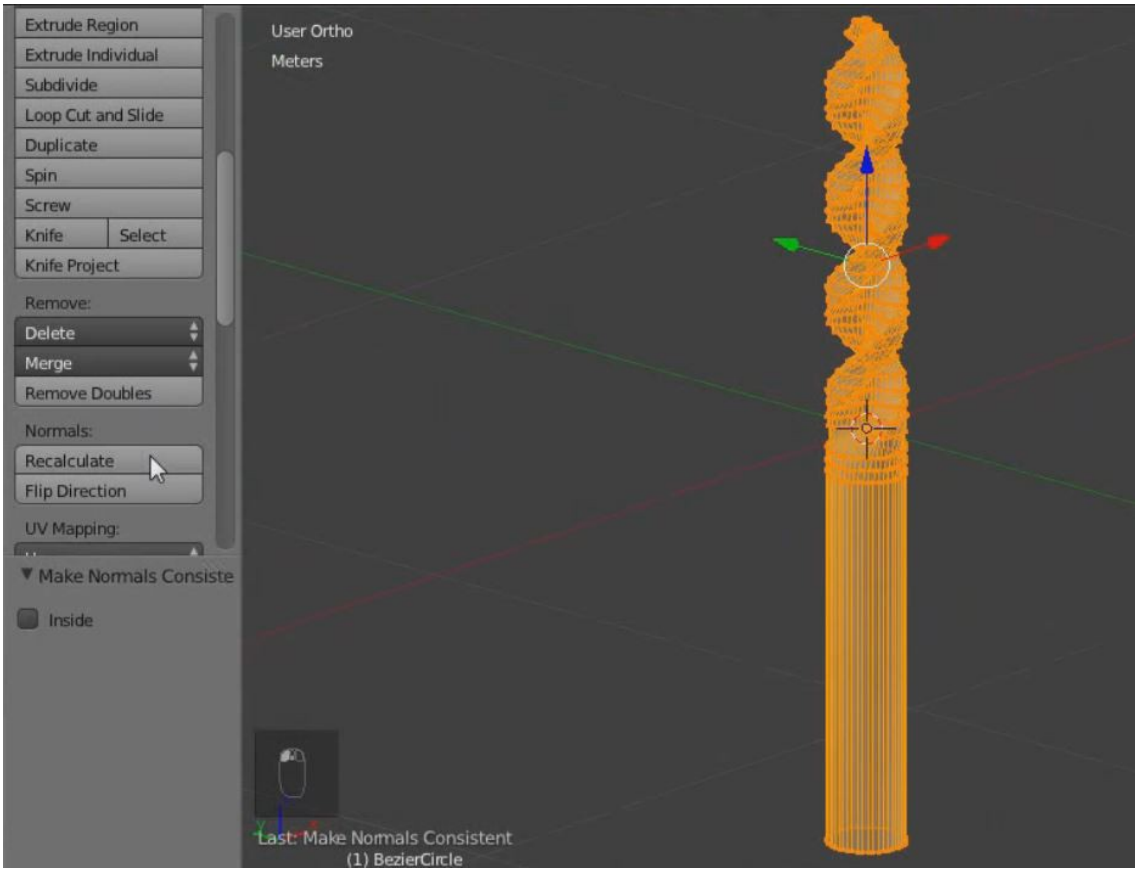
Per concludere, mancano tre piccoli passi: chiudere la base, ricalcolare le normali e applicare l'ombreggiatura Smooth.

Per chiudere la base, selezioniamo l'Edge Loop che la compone e premiamo *F* (per *Fill*, riempi).




*Chiusura dell'Edge Loop alla base, con FILL (shortcut: F)*

Per ricalcolare le normali, in Edit Mode selezioniamo tutto con *A* e clicchiamo su *Recalculate*, nella Tool Shelf (immagine nella pagina seguente).

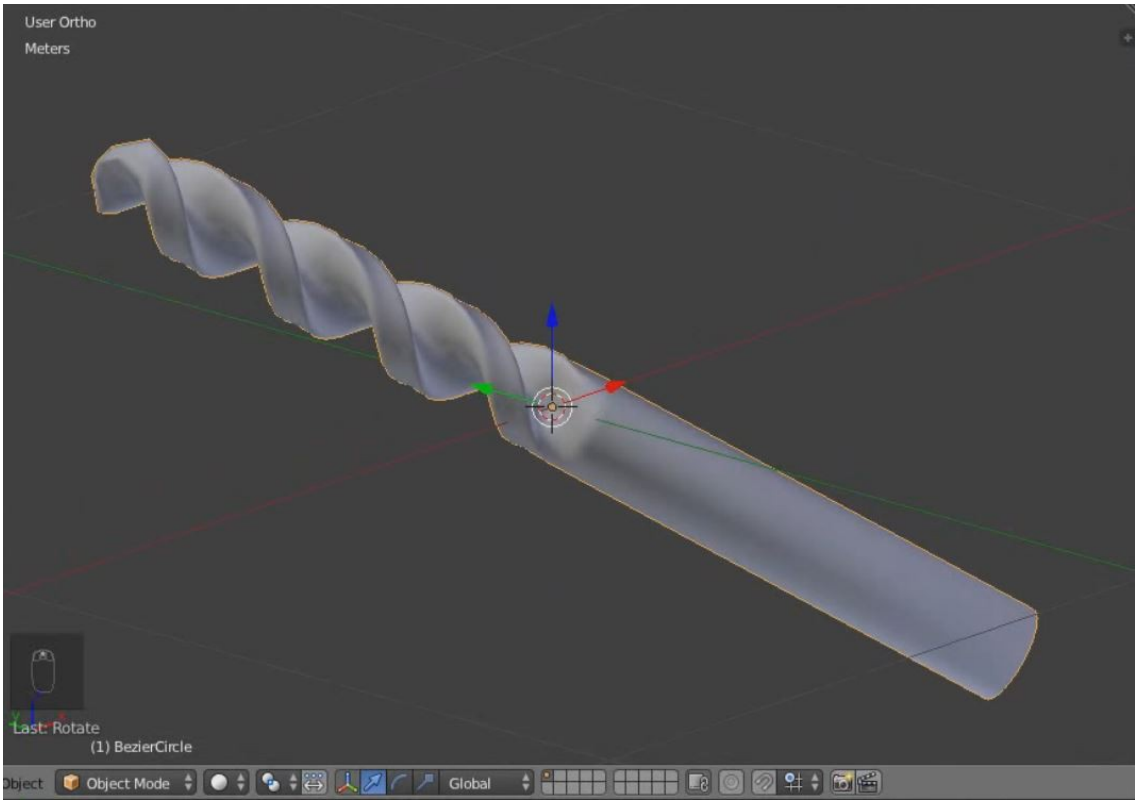



*Recalculate Normals, nella Tool Shelf, con tutti gli elementi selezionati in Edit Mode*

Per applicare l'ombreggiatura Smooth, torniamo in Object Mode e, nella Tool Shelf, clicchiamo su Shading *Smooth*.

Se volete, potete applicare altri modificatori, come Subdivision Surface, per “addolcire” ancora di più le forme, ma non dovrebbe essere necessario.

Il risultato finale del lavoro è visibile nell'immagine nella pagina seguente.




*Risultato finale, dopo aver applicato lo Shading Smooth, in Object Mode*

*  *  *

*  *  *