Bases de BPY 3 (Blender 3.x + Python) : comment exporter des UV LAYOUTS sous forme d'images PNG

Salut!

Dans ce tutoriel, nous allons voir comment utiliser la fonction export_layout dans Blender 3.0 (Alpha) pour exporter les UV maps (ou : Layouts) de tous les objets MESH disponibles dans une scène.

Ce script peut être utile si vous devez fournir les UV maps sous forme d'images PNG, par exemple si vous souhaitez publier votre scène 3D sur certains sites de stock 3D.

La fonction bpy.ops que nous allons utiliser est assez simple, en fait : elle nécessite deux paramètres (le chemin du fichier de l'image à créer et la taille UV, en pixels), mais nous devons l'utiliser pour tout le MESH objets disponibles dans la scène, nous allons donc utiliser une instruction FOR sur les objets MESH.

Tout d'abord, ouvrons un fichier BLEND ; vous pouvez également commencer avec un tout nouveau fichier, mais vous devrez l'enregistrer quelque part, car nous mettrons les fichiers PNG dans le même dossier que le projet.

Faisons quelques importations de base:

```
import bpy

from bpy import *
```

Comme je l'ai déjà dit, la fonction export_layout nécessite deux paramètres, définissons donc deux variables (nous pourrions écrire les valeurs dans la fonction elle-même, mais je préfère les définir ici par souci de clarté):

```
UVpath = bpy.path.abspath(“//”) + “UV-LAYOUT---”

UVsize = (2048, 2048)
```

Comme vous pouvez le voir, « bpy.path.abspath » donnera le chemin absolu sur le disque du fichier BLEND ; nous allons ajouter un préfixe ("UV-LAYOUT---") à tous les fichiers image, mais vous pouvez le modifier ou le supprimer.

Le paramètre UVsize est un tuple de deux valeurs numériques, qui sont les dimensions – en pixels – des images à créer ; J'écris 2048, créant ainsi une image 2k.

Afin d'exécuter la fonction sur tous les objets MESH disponibles dans une scène, nous devons créer une instruction FOR, afin que nous puissions écrire:

```
for o in bpy.data.objects:
```

ensuite, dans l'instruction FOR (donc : n'oubliez pas d'indenter le texte), on peut ajouter un « filtre » sur le type de l'objet:

```
if(o.type==”MESH”):
```

La condition est claire : le type du bpy.data.object courant doit être MESH.

Les déclarations suivantes doivent être placées à l'intérieur du IF, donc – encore une fois – n'oubliez pas d'indenter le texte !

```
bpy.context.view_layer.objects.active = o

bpy.ops.uv.export_layout(filepath = UVpath + o.name, size = UVsize)
```

L'affectation définira l'objet ACTIVE de la scène, affectant le maillage actuel à context.view_layer.objects.active ; nous devons le faire car l'OPS (opération Blender Python) suivante sera effectuée sur le seul et unique objet ACTIF (rappel : vous pouvez sélectionner plusieurs objets, mais il n'y a qu'un seul objet actif à la fois).

L'instruction OPS exportera la mise en page de l'objet ACTIVE dans le chemin de fichier donné (fait par la variable UVpath, avec le préfixe "UV-LAYOUT---", plus le nom de l'objet ACTIVE) et avec la taille donnée, en pixels .

Avant de fermer ce tutoriel, je vais vous donner un devoir : éditez ce script afin d'exporter à la fois les UV Layouts 2k et 4k des mesh disponibles dans vos scènes, en fournissant aux images deux préfixes différents ("UV-2K- --" et "UV-4k---", par exemple).

C'est tout! J'espère que vous avez apprécié ce tutoriel!

À bientôt!