Script Export tri blender, identifier les informations concernant les positions des vertices, faces et texturage.

Le script associé sera trouvable dans le wiki de la forge : https://forge.clermont-universite.fr/projects/polytech-ge-sous-traitance/wiki/Raydium_Blender/Script_identifier.py

Le mesh est lu dans bpy.data.meshes['Objet'], ici Objet = Cube

```
me = bpy.data.meshes['Cube']
```

- Détermination de la position dans l'espace des vertices :

Les vertices sont stockées dans : bpy.data.meshes.vertices[i]
On peut obtenir les coordonnées de chaque vertices avec ___.vertices[i].co[0]...[1]...[2] (x,y,z)

```
# génération matrice des coordonées des vertices :
vts = []
for vert in me.vertices :
    vts.append([vert.co[0],vert.co[1],vert.co[2]])
vts_mtx = np.matrix(vts)
```

- La triangulation du mesh :

La triangulisation se réalise avec la fonction bpy.data.meshes.calc_looptriangles()

```
me.calc_looptriangles()
```

Pour obtenir les vertices de chaque triangle :
bpy.data.meshes.loop_triangles.loop_triangle.vertices[0]..[1]..[2]

- Génération de la matrice des vertices des triangles issus de la triangulation :

```
for triangle in me.loop_triangles:
    trgl.append(list(triangle.vertices))

trgl_mtx = np.matrix(trgl)
```

- **Lecture face par face du mesh, et identification des vertices par face :**

Les faces sont obtenues dans :
bpy.data.meshes['Objet'].polygons

Les vertices liés à chaque face sont obtenues dans :
me.data.polygons[0].vertices (si on souhaite obtenir les vertices de la face 0)

```
for face in me.polygons :
    print('Face', face.index, 'vertices', list(face.vertices))
```

- **Lecture des coordonnées x,y des textures du mesh :**

IL FAUT IMPERATIVEMENT PASSER PAR LA METHODE .to_mesh()
bpy.context.scene.objects['Objet'].to_mesh()

Donc :

```
obj = bpy.context.scene.objects['Cube']

mm = obj.to_mesh()
```

!!!! attention : noter bien la différence entre mm et me, me peut utiliser la méthode .calc_looptriangles(), mm quand à lui peut obtenir les coordonnées x et y de l'image. mm ne peut pas utiliser calc_looptriangles() et vice-versa me ne peut obtenir les coordonnées x et y de l'image texturé !!!

Pour lire les coordonnées de chaque point :
bpy.data.meshes.uv_layers[i].data[i].uv.xy

```
for uv_layer in mm.uv_layers :
     for coord in uv_layer.data :

          print('x:',coord.uv.xy[0],'y:',coord.uv.xy[1])
```

Lorsque que l'on exécute le script pour un cube, voici le résultat dans la console :

```
Voici les caractéristiques de votre mesh nommé : Cube
Matrice des coordonnées des vertices : bpy.data.meshes.vertices[i].vertices[i].co[0]...[1]...[2] x=0,y=1,z=2
Vert_0 [[1. 1. 1.]]
Vert_1 [[ 1.  1. -1.]]
Vert_2 [[ 1. -1.  1.]]
...
Vert_7 [[-1. -1. -1.]]


CALCULE DE LA TRIANGULISATION DU MESH : bpy.data.meshes.calc_looptriangles()
Matrice de la triangulisation du mesh : bpy.data.meshes.loop_triangles.loop_triangle.vertices[0]..[1]..[2]
Triangle_0 vertices [[0 4 6]]
Triangle_1 vertices [[0 6 2]]
Triangle_2 vertices [[3 2 6]]
...
Triangle_11 vertices [[5 0 1]]


Affichage faces par faces des vertices de chaques faces
Face 0 vertices [0, 4, 6, 2]
Face 1 vertices [3, 2, 6, 7]
Face 2 vertices [7, 6, 4, 5]
Face 3 vertices [5, 1, 3, 7]
Face 4 vertices [1, 0, 2, 3]
Face 5 vertices [5, 4, 0, 1]
Les coordonnees des textures : bpy.data.meshes.uv_layers[i].data[i].uv.xy


Face 0
Vert_0 : x= 0.625 y= 0.5
Vert_1 : x= 0.875 y= 0.5
Vert_2 : x= 0.875 y= 0.75
Vert_3 : x= 0.625 y= 0.75


Face 1
Vert_4 : x= 0.375 y= 0.75
Vert_5 : x= 0.625 y= 0.75
Vert_6 : x= 0.625 y= 1.0
Vert_7 : x= 0.375 y= 1.0


Face 2
Vert_8 : x= 0.375 y= 0.0
Vert_9 : x= 0.625 y= 0.0
Vert_10 : x= 0.625 y= 0.25
Vert_11 : x= 0.375 y= 0.25


Face 3
Vert_12 : x= 0.125 y= 0.5
Vert_13 : x= 0.375 y= 0.5
Vert_14 : x= 0.375 y= 0.75
Vert_15 : x= 0.125 y= 0.75


Face 4
Vert_16 : x= 0.375 y= 0.5
Vert_17 : x= 0.625 y= 0.5
Vert_18 : x= 0.625 y= 0.75
Vert_19 : x= 0.375 y= 0.75


Face 5
Vert_20 : x= 0.375 y= 0.25
Vert_21 : x= 0.625 y= 0.25
Vert_22 : x= 0.625 y= 0.5
Vert_23 : x= 0.375 y= 0.5
```