Forum: Python & Plugins

Topic: Questions création d'interface pour un script

Subject: Re: Questions création d'interface pour un script

PostÃ© par: vivelesnains

Contribution le : 23/11/2017 14:30:35

Bonjour à tous et surpris de n'avoir eu aucune réponse

Bref, j'ai progressé, mais j'ai toujours des problèmes pour utiliser autre chose que des strings ou des button. J'arrive à afficher ce qui m'intéresse mais impossible de les utiliser.
J'aimerais utiliser mon BoolShading pour afficher ou cacher mes planes lights à la place de mon operator Reveal/Hide.

Pareil, si quelqu'un pouvait m'expliquer également comment utiliser un IntProperty ou un float property pour changer (notamment des translations) en temps réel des objets dans ma scène.

En vous remerciant d'avance,

Cdlt,
Snains

```
################################
import bpy
from bpy.props import *

bpy.context.scene.render.engine = &#039;CYCLES&#039;
bpy.context.scene.cursor_location=(0,0,0)
bpy.context.scene.layers=[i==0 for i in range(20)]

def initSceneProperties(scn):
    bpy.types.Scene.MyString = StringProperty(
        name="Object",
        description="Obj u want to texturize")
    scn[&#039;MyString&#039;] = "Click & Drop obj"

    bpy.types.Scene.BoolShading = BoolProperty(
        name="Visibility",
        description="Show/Hide ShadersLights")
    scn[&#039;BoolShading&#039;] = "Toggle visibility"

    return

initSceneProperties(bpy.context.scene)
```

```
class ToolsPanel(bpy.types.Panel):
    bl_label = "Shading Light"
    bl_space_type = "VIEW_3D"
    bl_region_type = "TOOLS"

    def draw(self, context):
        layout = self.layout
        scn=context.scene
        layout.prop(scn, &#039;MyString&#039;)
        layout.operator("add.lightshading")
        layout.operator("hide.lightshading")
        layout.prop(scn, &#039;BoolShading&#039;)
        layout.operator("delete.lightshading")

class AddLightShading (bpy.types.Operator):
    bl_idname = "add.lightshading"
    bl_label = "Create"

    def execute(self, context):
        scn = context.scene
        bpy.ops.object.select_all(action=&#039;DESELECT&#039;)
        bpy.data.objects[scn[&#039;MyString&#039;]].select=True
        bpy.context.scene.objects.active=bpy.data.objects[scn[&#039;MyString&#039;]]
        CubeTest = bpy.context.active_object
        pos=CubeTest.location
        bpy.context.scene.layers=[i==14 for i in range(20)]
        for obj in bpy.context.visible_objects:
            obj.select=True
        bpy.ops.object.delete()
        LightMat = bpy.data.materials.get("shaders Light Mat")
        if LightMat is None :
            LightMat=bpy.data.materials.new(name="shaders Light Mat")
            LightMat.use_nodes = True
            LightMat.node_tree.nodes.remove(LightMat.node_tree.nodes.get(&#039;Diffuse
BSDF&#039;))
            emission=LightMat.node_tree.nodes.new(type=&#039;ShaderNodeEmission&#039;)
            LightMat.node_tree.links.new(LightMat.node_tree.nodes.get(&#039;Material
Output&#039;).inputs[0], emission.outputs[0])
        bpy.ops.mesh.primitive_plane_add(location=(2,2,2))
        PlaneLight = bpy.context.active_object
        PlaneLight.name=&#039;shaders_lightPlane&#039;
        bpy.context.object.rotation_euler[0]=-((3.14*45)/180)
        bpy.context.object.rotation_euler[2]=-((3.14*45)/180)
        PlaneLight.active_material = LightMat
        bpy.ops.object.origin_set(type=&#039;ORIGIN_CURSOR&#039;)
        bpy.ops.object.transform_apply(location=True, rotation=True, scale=True)
        bpy.ops.object.modifier_add(type=&#039;MIRROR&#039;)
        bpy.context.object.modifiers["Mirror"].use_y=True
        bpy.ops.transform.translate(value=pos)
```

```
        bpy.context.scene.layers[0]=True
        bpy.context.scene.BoolShading = True
        return{&#039;FINISHED&#039;}

class HideLightShading (bpy.types.Operator):
    bl_idname = "hide.lightshading"
    bl_label = "Reveal/Hide"

    def execute(self, context):
        bpy.ops.object.select_all(action=&#039;DESELECT&#039;)
        bpy.data.objects[&#039;shaders_lightPlane&#039;].select=True
        bpy.context.scene.objects.active=bpy.data.objects[&#039;shaders_lightPlane&#039;]

        if bpy.data.objects[&#039;shaders_lightPlane&#039;].hide==True :
            bpy.context.object.hide = False
        else :
            bpy.context.object.hide = True
        return{&#039;FINISHED&#039;}

class DeleteLightShading (bpy.types.Operator):
    bl_idname = "delete.lightshading"
    bl_label = "Delete"

    def execute(self, context):
        bpy.ops.object.select_all(action=&#039;DESELECT&#039;)
        bpy.data.objects[&#039;shaders_lightPlane&#039;].select=True
        bpy.context.scene.objects.active=bpy.data.objects[&#039;shaders_lightPlane&#039;]
        bpy.ops.object.delete()
        return{&#039;FINISHED&#039;}



bpy.utils.register_module(__name__)
print ("fin")
############################################################
```