A Quick Way to Automatically Remove Loose Parts in Blender

No comments

There are times that you face some unwanted extra objects all over your mesh such as when you get a raw 3d scan object that has been converted from point cloud to mesh. And, you want to quickly get rid of them all. In this tutorial, we are going to see how we can remove these loose parts in Blender and how we can create a button to quickly do that for us.

How to Remove Loose Parts in Blender

There are times that you face some unwanted extra objects all over your mesh such as when you get a raw 3d scan object that has been converted from point cloud to mesh and you want to quickly get rid of them all. In this tutorial, we are going to see how we can get away with these small useless specks of dust using Blender python and how we can create a button to quickly do that for us.

IMPORTANT NOTE:

Remember that the scripts we are using here are related to Blender version 2.83 and if you are working with any other versions, it is probable that the scripts might differ a little bit, but we will show you ways to find the proper functions if there are any differences at all.
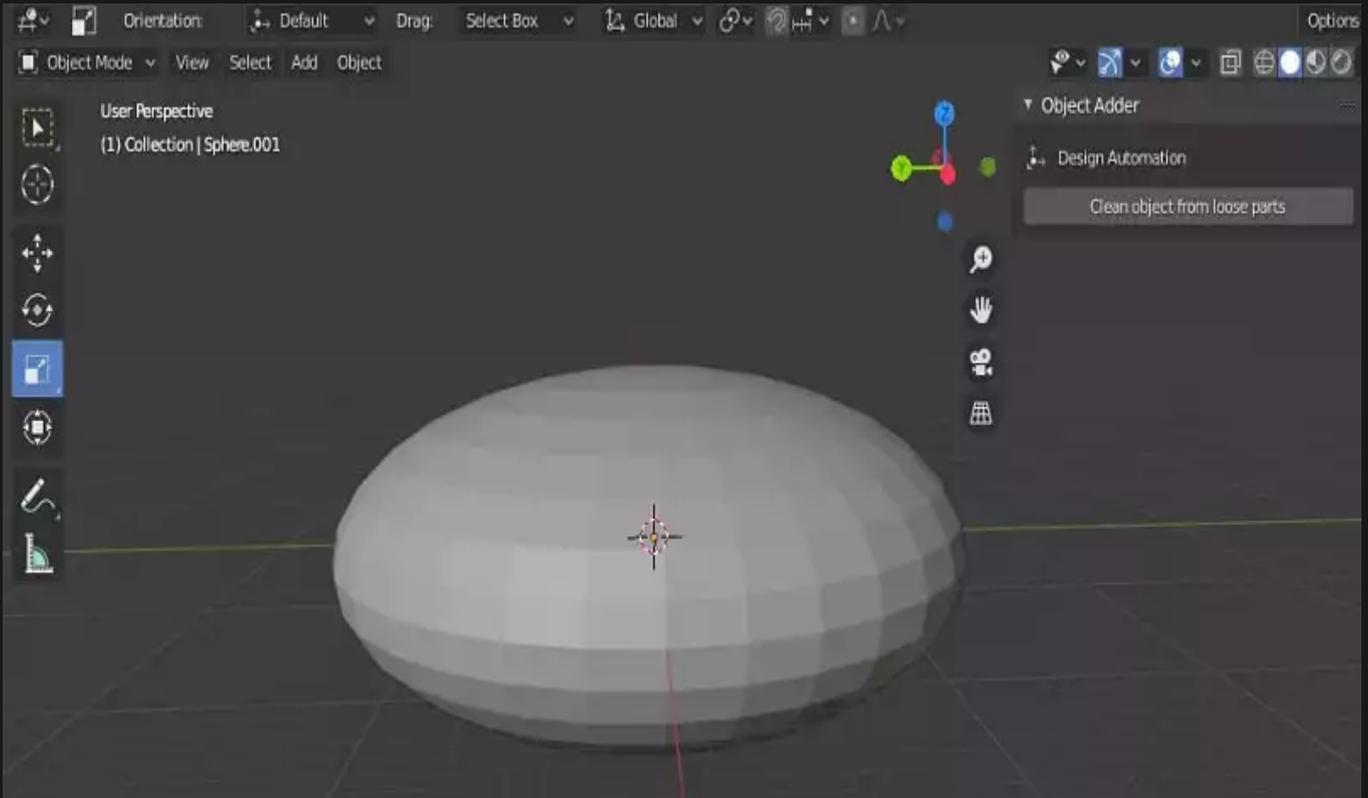
Python Scripts

First of all, we write our utility functions and then apply them in the main panel.

```
import  bpy


###################################################################
#####                   Utility Functions
###################################################################


def clean_from_loose_parts(obj):
    deselect_objects()

    # separate the mesh
    if obj.type == 'MESH':
        curr_obj_name = obj.name

        obj.select_set(True)
        bpy.ops.mesh.separate(type='LOOSE')

        # separated_objects = tuple(obj.name for obj in
bpy.context.selected_objects))
        separated_objects_count = len(bpy.context.selected_objects)
```

```
            print("Mesh object {} separated in to {} object(s)"
.format(curr_obj_name,
                separated_objects_count))

        if separated_objects_count > 1:
            # remove mesh objects which have less than max vertices
            obj_max_vertices = get_max_vertices_object_from_selection()
            obj_max_vertices.select_set(False)
            bpy.ops.object.delete()
            # restore input name of the object
            obj_max_vertices.name = curr_obj_name

        else:
            obj.select_set(False)
```

The above function will do the main job of removing small loose parts for us.

```
def get_max_vertices_object_from_selection():
    max_vert_count = 0
    obj_max_vertices = None
    for obj in bpy.context.selected_objects:
        vert_count = len(obj.data.vertices)
        if max_vert_count < vert_count:
            max_vert_count = vert_count
            obj_max_vertices = obj

    return obj_max_vertices
```

The above function will count the vertices of the selected objects and return the one that has the maximum amount of vertices.

```
def get_object_by_name(obj_name):
    assert obj_name in bpy.data.objects, "Error
 getting object by name:{}".format(obj_name)
    obj = bpy.data.objects[obj_name]

    return obj
```

The above function will get the object by name and store it in a variable.

```
def deselect_objects():
    bpy.ops.object.select_all(action='DESELECT')
```

The above function will deselect the active object. Let's write the code of our main panel.

```python
#################################################################
########              Main Panel
#################################################################

class MainPanel(bpy.types.Panel):
    bl_label = "Object Adder"
    bl_idname = "VIEW_PT_MainPanel"
    bl_space_type = 'VIEW_3D'
    bl_region_type = 'UI'
    bl_category = 'Design Automation'

    def draw(self, context):
        layout = self.layout
        layout.scale_y = 1.2

        row = layout.row()
        row.label(text= "Design Automation", icon= 'OBJECT_ORIGIN')
        row = layout.row()
        row.operator("wm_function.myop", text=
"Clean object from loose parts")

#################################################################
####                    Main UI ?Functions
#################################################################

class WM_Function_myOp(bpy.types.Operator):
    """Click to apply our customized function"""
    bl_label = "Our customized function"
    bl_idname = "wm_function.myop"


    def execute(self, context):

        obj = get_object_by_name('Sphere')
        clean_from_loose_parts(obj)

        return {'FINISHED'}

    def invoke(self, context, event):

        return context.window_manager.invoke_props_dialog(self)

#################################################################
#####                        Register and Unregister
#################################################################
```

```python
def register():
    bpy.utils.register_class(MainPanel)
    bpy.utils.register_class(WM_Function_myOp)


def unregister():
    bpy.utils.unregister_class(MainPanel)
    bpy.utils.unregister_class(WM_Function_myOp)



if __name__ == "__main__":
    register()
```

In the above main panel script, we use the utility function `clean_from_loose_parts(obj)`. The rest is related to the GUI of the panel.

Testing the Scripts

Now, it is finally time to run our code using `CTRL + P` or the run button at the top of the scripting window and we will see that a panel appears with a button using which we can remove extra loose parts.

And as you can see, the small loose parts are removed. Consequently, we can have a clean object from noises. You can apply this tool to any raw 3D scan and make the edition of it much faster.

Getting Rid of Loose Parts in Blender: Wrapping Up

In this tutorial, we have managed to create a panel that can remove the noises and loose parts from all the objects that have them in Blender. Most of the time we face the issue of having loose parts when we receive a raw 3D scan from a scanner device. Having such a tool is very useful and saves a lot of time for you.

Join Arashtad Community

Follow Arashtad on Social Media

We provide variety of content, products, services, tools, tutorials, etc. Each social profile according to its features and purpose can cover only one or few parts of our updates. We can not upload our videos on SoundCloud or provide our eBooks on Youtube. So, for not missing any high quality original content that we provide on various social networks, make sure you follow us on as many social networks as you're active in. You can find out Arashtad's profiles on different social media services.
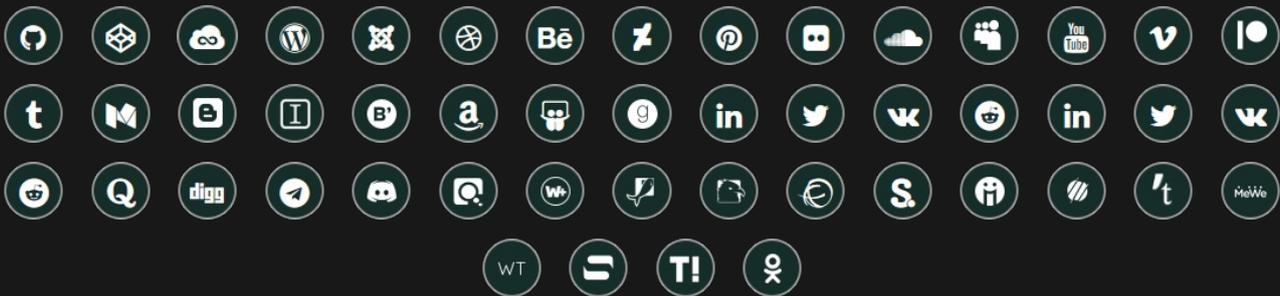
Get Even Closer!

Did you know that only one universal Arashtad account makes you able to log into all Arashtad network at once? Creating an Arashtad account is free. Why not to try it? Also, we have regular updates on our newsletter and feed entries. Use all these benefitial free features to get more involved with the community and enjoy the many products, services, tools, tutorials, etc. that we provide frequently.

SIGN UP   NEWSLETTER   RSS FEED