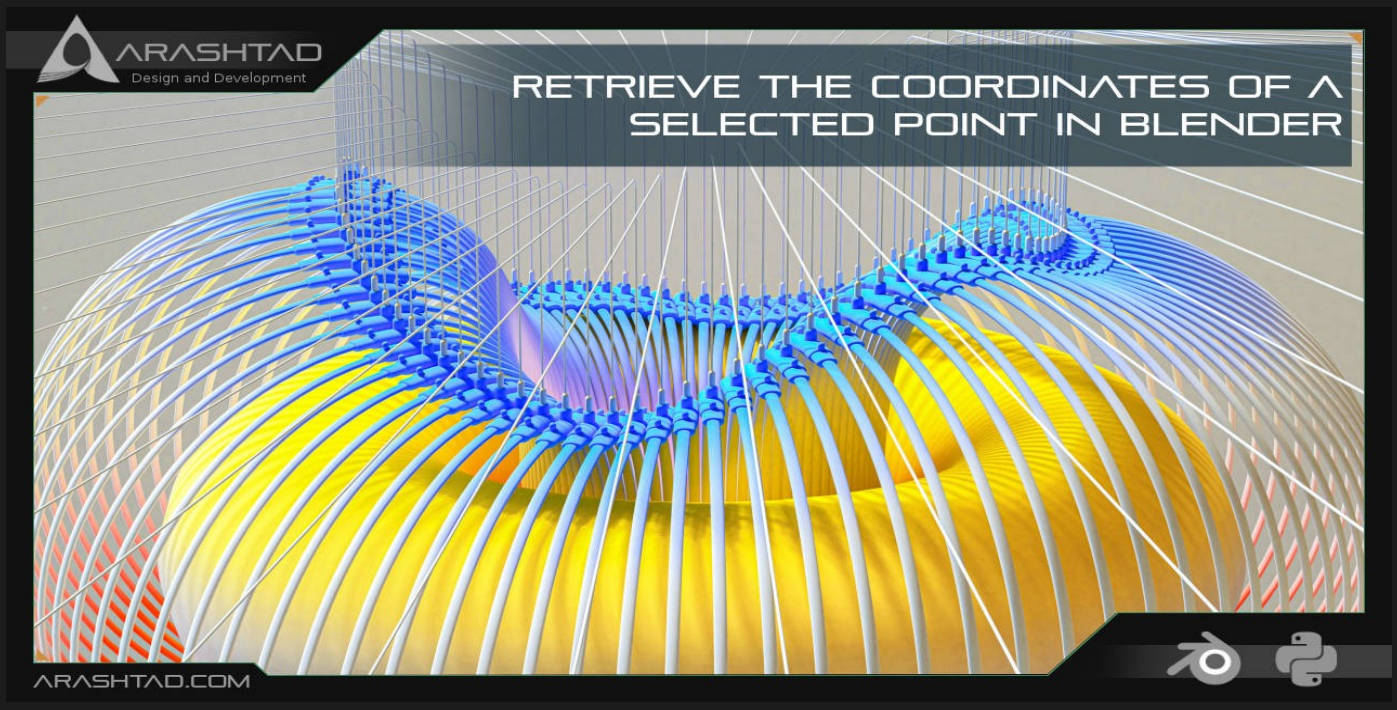
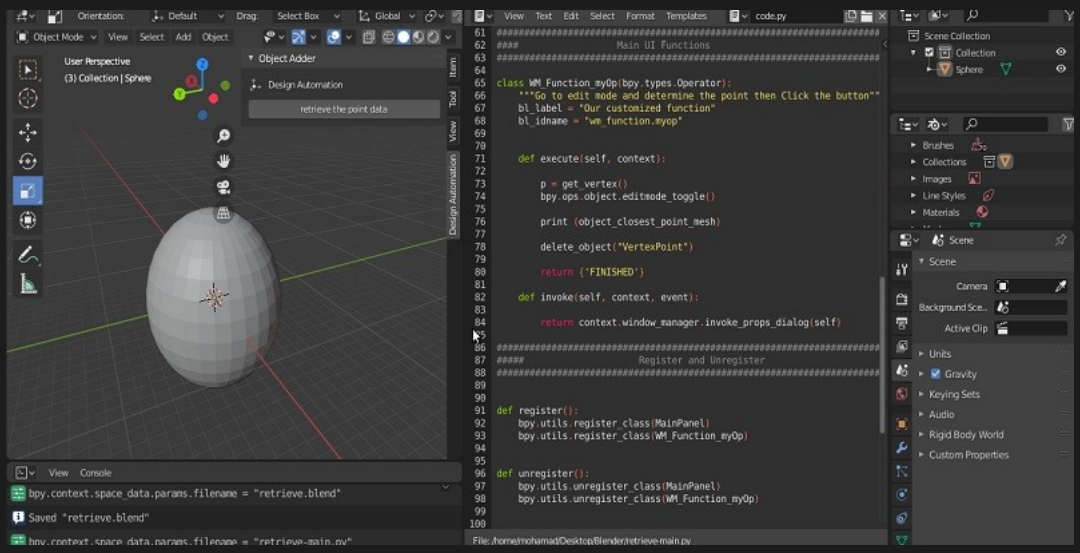
Title	RETRIEVE THE COORDINATES OF A SELECTED POINT IN
Description	Intro
Date	June 29, 2022
Author	Arashtad
Author URI	https://Arashtad.com



The purpose of this article is to find an easy way to obtain the coordinates and the normal of a point in Blender specified by the user. Using the obtained point data, we can automate the translation of the objects that are going to be transferred to that certain point and rotated according to the normal directions of that point. The use cases of this kind of object translation are for 3D modeling of complex objects as well as placing the internal lattice structures.

OBTAINING THE COORDINATES OF ANY POINT IN BLENDER

Finding the coordinates of a selected point is a key function for a lot of other important tasks, such as translating another part to that point and so on. Here, we write some functions in addition to an interface for you to get the coordinates of a selected point on an object and print the data of the location and the direction of the mesh normal.



Notice that the user should go to edit mode while selecting the object and click on the edge that they want to get the coordinates from. As you know edges have no normals so we have to find the closest mesh to the edge that has been selected by the user.

Using the below python scripts in the scripts section of Blender, we will define some utility functions as well as the main execute class that is going to apply the utility functions with a certain sequence.

```
import  bpy
import bmesh

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

def object_closest_point_mesh(p, obj):

    result, location, normal, face_index =
obj.closest_point_on_mesh(p)
    assert result, "Can't find closest point on mesh"

    location = location.to_tuple()
    normal = normal.to_tuple()
```

The above function will find the closest point on the mesh for us. The function itself uses another function that is built in Blender and finds the closest vertex on a mesh based on the given point and returns the location and the normal data of the said vertex. Finally, it returns the 2 sets of data.

```
def get_vertex():

        bm = bmesh.new()
        ob = bpy.context.active_object
        bm = bmesh.from_edit_mesh(ob.data)

        points = []
        for v in bm.verts:
            if (v.select == True):
                obMat = ob.matrix_world
                points.append(obMat @ v.co)
        for p in points:
            pOb = bpy.data.objects.new("VertexPoint", None)
            bpy.context.collection.objects.link(pOb)
            pOb.location = p
        return p
```

The above function will get the vertex that has been selected by the user in the edit mode and returns the coordinates of the said vertex.

```
def delete_object(objName):
    bpy.ops.object.select_all(action='DESELECT')
    bpy.data.objects[objName].select_set(True) # Blender 2.8x
    bpy.ops.object.delete()
```

The above function will delete any given object. To delete an object, we need to first deselect all the other objects and then select the object that we specified its name in the function and finally delete the selected object.

```
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 its name. Meaning that it will select according to the name given.

```
######################################################################
########              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= "retrieve the point
data")
```

```
################################################################################
####                  Main UI Functions
################################################################################

class WM_Function_myOp(bpy.types.Operator):
    """Go to edit mode and determine the point then Click the
button"""
    bl_label = "Our customized function"
    bl_idname = "wm_function.myop"

    def execute(self, context):

        p = get_vertex()
        bpy.ops.object.editmode_toggle()
        obj = get_object_by_name('Sphere')
        print (object_closest_point_mesh(p,obj))
        delete_object("VertexPoint")
        return {'FINISHED'}

    def invoke(self, context, event):

        return context.window_manager.invoke_props_dialog(self)
```

The above execute function is so simple. It first gets the selected vertex in the edit mode from the user, then it toggles the edit mode to object mode. After that, it prints the coordinates and the normals of the selected point. And at last, it deletes the VertexPoint object from the list of objects.

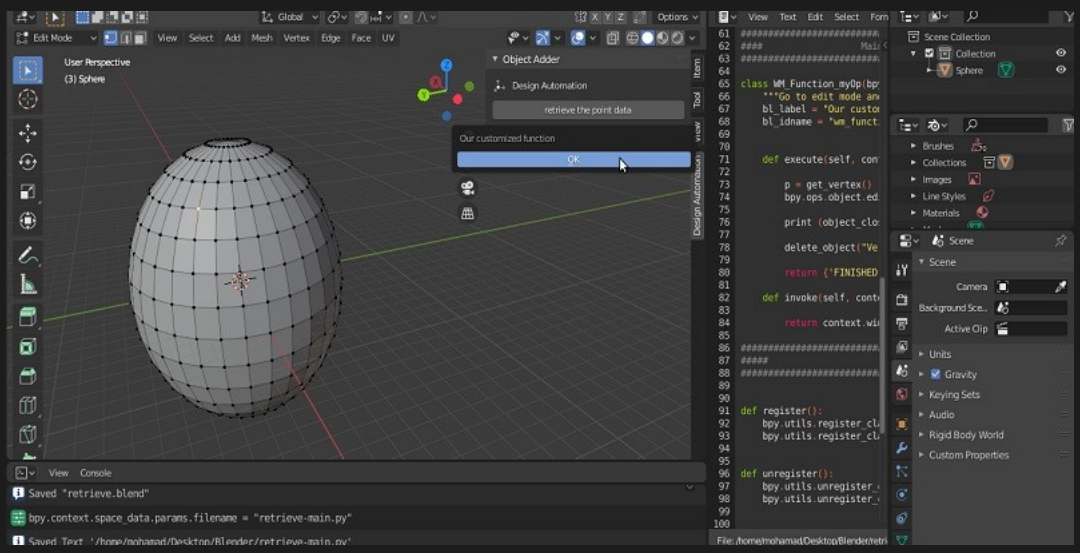
```
################################################################################
#####                        Register and Unregister
################################################################################

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()
```

Don’t forget to close the project by registering and unregistering the classes defined. If you run the scripts, you will be able to see the panel below. Click the button in the panel (retrieve the point data button) and Click OK and notice that before you click OK, you need to have determined the point you wish to select, in the edit mode of Blender.



The following is the result of retrieved data of the point:

```
(-0.46193963289260864, 0.3086579740047455, 0.8314695954322815, -
0.41729676723480225, 0.22304992377758026, 0.8809722661972046) Info:
Deleted 1 object(s)
```

The first 3 numbers are the XYZ coordinates of the point and the second 3 numbers are related to the direction of the mesh normal.

THE JOB IS DONE

In this tutorial, we have managed to propose a way to quickly obtain the coordinates and normal directions of any point on an object. This data is very useful especially when you want to operate some complex 3D modeling or translate objects to a certain point on another object.