JAVARIA MALIK (191873)

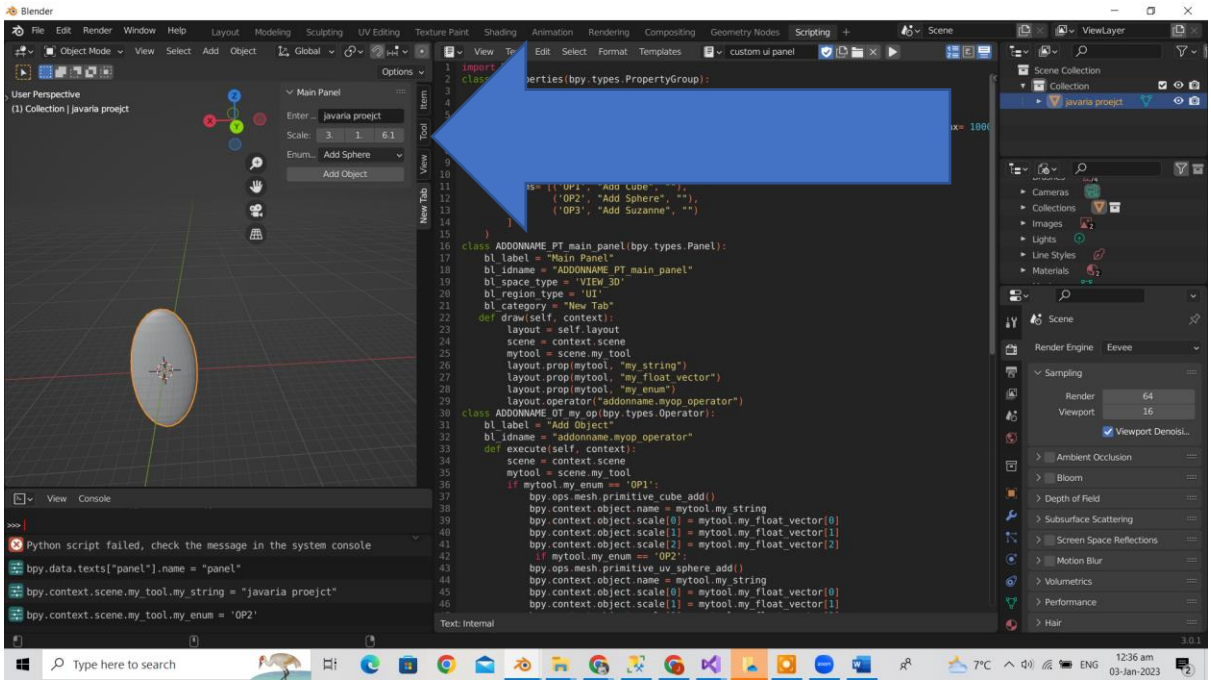
Game engine project

CUSTOM UI PANEL

First I create a panel in blender using python scritping

```
'''javaria malik project'''

'''prject for Gam engine'''


import bpy


class PANEL_CUSTOM_UI(bpy.types.Panel):
   """Create custom Panel"""
   bl_label="Panel Custom UI"
   bl_idname="OBJECT_PT_panel"
   bl_space_type='VIEW_3D'
   bl_region_type='UI'
   bl_category = "Panel Custom UI"
   def draw(self, context):
      layout = self.layout
      row=layout.row()
      row.label(text = "ROW 5454")
def register():
   bpy.utils.register_class(PANEL_CUSTOM_UI)
def unregister():
   bpy.utils.register_class(PANEL_CUSTOM_UI)
if _name_ == "_main_":
   register()


   ------------------------------------------------
```

THEN I CREATE CUSTOM PANEL WHERE I CAN ADD OBJECT AND ADD NAME OF MY COLLECTION I can ALSO SCALE the object by using panel



This code of my python script

```
import bpy

class MyProperties(bpy.types.PropertyGroup):


    my_string : bpy.props.StringProperty(name= "Enter Text")


    my_float_vector : bpy.props.FloatVectorProperty(name= "Scale", soft_min= 0, soft_max= 1000, default= (1,1,1))


    my_enum : bpy.props.EnumProperty(
        name= "Enumerator / Dropdown",
        description= "sample text",
        items= [('OP1', "Add Cube", ""),
                ('OP2', "Add Sphere", ""),
                ('OP3', "Add Suzanne", "")
        ]
    )
class ADDONNAME_PT_main_panel(bpy.types.Panel):
    bl_label = "Main Panel"
    bl_idname = "ADDONNAME_PT_main_panel"
```

```
    bl_space_type = 'VIEW_3D'
    bl_region_type = 'UI'
    bl_category = "New Tab"
   def draw(self, context):
      layout = self.layout
      scene = context.scene
      mytool = scene.my_tool
      layout.prop(mytool, "my_string")
      layout.prop(mytool, "my_float_vector")
      layout.prop(mytool, "my_enum")
      layout.operator("addonname.myop_operator")
class ADDONNAME_OT_my_op(bpy.types.Operator):
   bl_label = "Add Object"
   bl_idname = "addonname.myop_operator"
   def execute(self, context):
      scene = context.scene
      mytool = scene.my_tool
      if mytool.my_enum == 'OP1':
         bpy.ops.mesh.primitive_cube_add()
         bpy.context.object.name = mytool.my_string
         bpy.context.object.scale[0] = mytool.my_float_vector[0]
         bpy.context.object.scale[1] = mytool.my_float_vector[1]
         bpy.context.object.scale[2] = mytool.my_float_vector[2]
          if mytool.my_enum == 'OP2':
         bpy.ops.mesh.primitive_uv_sphere_add()
         bpy.context.object.name = mytool.my_string
         bpy.context.object.scale[0] = mytool.my_float_vector[0]
         bpy.context.object.scale[1] = mytool.my_float_vector[1]
         bpy.context.object.scale[2] = mytool.my_float_vector[2]
         if mytool.my_enum == 'OP3':
         bpy.ops.mesh.primitive_monkey_add()
         bpy.context.object.name = mytool.my_string
         bpy.context.object.scale[0] = mytool.my_float_vector[0]
```

```
            bpy.context.object.scale[1] = mytool.my_float_vector[1]
            bpy.context.object.scale[2] = mytool.my_float_vector[2]
            return {'FINISHED'}
classes = [MyProperties, ADDONNAME_PT_main_panel, ADDONNAME_OT_my_op]
def register():
    for cls in classes:
        bpy.utils.register_class(cls)
        bpy.types.Scene.my_tool = bpy.props.PointerProperty(type= MyProperties)

def unregister():
    for cls in classes:
        bpy.utils.unregister_class(cls)
        del bpy.types.Scene.my_tool

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