import bpy

import os

import math

print("Script started")

# Set output directory

output\_dir = r"D:\Uni\Work\3rd Year\R&D\OutPut Textures\Cubemap"

if not os.path.exists(output\_dir):

os.makedirs(output\_dir)

print("Output directory created")

# Get the camera

camera\_name = "Cubemap\_Camera"

camera = bpy.data.objects.get(camera\_name)

if not camera:

print(f"ERROR: Camera '{camera\_name}' not found!")

exit()

# Set render settings

scene = bpy.context.scene

scene.camera = camera

scene.render.engine = 'BLENDER\_EEVEE\_NEXT'

scene.render.resolution\_x = 1024

scene.render.resolution\_y = 1024

scene.render.image\_settings.file\_format = 'PNG'

scene.render.use\_file\_extension = True

# Define camera rotations for each frame

camera\_rotations = {

1: (0, 0, 0),

2: (0, math.radians(180), 0),

3: (0, math.radians(90), 0),

4: (0, math.radians(-90), 0),

5: (math.radians(-90), 0, 0),

6: (math.radians(90), 0, 0)

}

# Apply keyframes for each frame (camera rotation)

for frame, rotation in camera\_rotations.items():

camera.rotation\_euler = rotation

camera.keyframe\_insert(data\_path="rotation\_euler", frame=frame)

print(f"Keyframe set for frame {frame}")

# Ensure compositor is active and reset it for normal rendering

scene.use\_nodes = True

tree = scene.node\_tree

# Remove old nodes (compositor reset)

for node in tree.nodes:

tree.nodes.remove(node)

# Add the Render Layer node to connect with the compositor

render\_layer\_node = tree.nodes.new(type="CompositorNodeRLayers")

render\_layer\_node.location = (-200, 0)

# Composite output node

composite\_node = tree.nodes.new(type="CompositorNodeComposite")

composite\_node.location = (600, 0)

composite\_node.use\_alpha = True

# Link the Render Layer node to the composite node

tree.links.new(render\_layer\_node.outputs["Image"], composite\_node.inputs["Image"])

# Render each frame

for frame in range(1, 7):

scene.frame\_set(frame)

bpy.context.view\_layer.update()

# Set file path for each rendered frame

output\_path = os.path.join(output\_dir, f"Render\_Frame\_{frame}.png")

scene.render.filepath = output\_path

print(f"Rendering frame {frame} to {output\_path}...")

# Render and save

bpy.ops.render.render(write\_still=True)

print("ALL cubemap images rendered successfully!")