import bpy

# Set the camera

camera\_name = "Camera.008"

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

bpy.context.scene.camera = camera # Ensure Blender uses the correct camera

# Store original render settings

original\_engine = bpy.context.scene.render.engine

# \*\*Disable overlays (hides lights, cameras, grid, etc.)\*\*

for area in bpy.context.screen.areas:

if area.type == 'VIEW\_3D':

for space in area.spaces:

if space.type == 'VIEW\_3D':

space.overlay.show\_overlays = False # Hide all UI overlays

# ---------------- Render Color Map ---------------- #

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

bpy.context.scene.view\_settings.view\_transform = 'Filmic'

bpy.context.scene.use\_nodes = False # Disable compositing

bpy.context.scene.render.filepath = r"D:\Uni\Work\3rd Year\R&D\OutPut Textures\Render\_Colour.png"

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

# Restore original engine after rendering

bpy.context.scene.render.engine = original\_engine

# ---------------- Render Depth Map ---------------- #

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

bpy.context.view\_layer.use\_pass\_z = True # Enable Z-depth pass

bpy.context.scene.use\_nodes = True # Enable compositor

# Access Compositing Nodes

tree = bpy.context.scene.node\_tree

nodes = tree.nodes

links = tree.links

# Clear existing nodes

for node in nodes:

nodes.remove(node)

# Add Render Layers Node (Depth Input)

render\_layers = nodes.new(type="CompositorNodeRLayers")

render\_layers.location = (-400, 200)

# Add Map Range Node (Better depth normalization)

map\_range = nodes.new(type="CompositorNodeMapRange")

map\_range.location = (-200, 200)

map\_range.inputs[1].default\_value = 0.1 # Input Min (Closer objects)

map\_range.inputs[2].default\_value = 15.0 # Input Max (Far objects)

map\_range.inputs[3].default\_value = 1.0 # Output Min (White)

map\_range.inputs[4].default\_value = 0.0 # Output Max (Black)

# Add File Output Node

file\_output = nodes.new(type="CompositorNodeOutputFile")

file\_output.location = (400, 200)

file\_output.base\_path = r"D:\Uni\Work\3rd Year\R&D\OutPut Textures"

file\_output.file\_slots[0].path = "Render\_Depth.png"

# Connect Nodes

links.new(render\_layers.outputs["Depth"], map\_range.inputs[0]) # Connect Depth to Map Range

links.new(map\_range.outputs[0], file\_output.inputs[0]) # Connect Map Range to Output

# Render Depth Pass

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

# \*\*Remove compositor nodes after depth render\*\*

for node in nodes:

nodes.remove(node)

# ---------------- Render Normal Map (Last) ---------------- #

bpy.context.scene.render.engine = 'BLENDER\_WORKBENCH'

# \*\*Make sure MatCap Normal Map is applied\*\*

for area in bpy.context.screen.areas:

if area.type == 'VIEW\_3D':

for space in area.spaces:

if space.type == 'VIEW\_3D':

space.shading.type = 'SOLID'

space.shading.light = 'MATCAP'

space.shading.studio\_light = "check\_normal+y.exr" # Make sure this MatCap exists

space.shading.color\_type = 'SINGLE' # Correct color mode

space.shading.show\_shadows = False # Disable shadows to avoid interference

# \*\*Render Normal Map using OpenGL\*\*

bpy.context.scene.render.filepath = r"D:\Uni\Work\3rd Year\R&D\OutPut Textures\Render\_Normals.png"

bpy.ops.render.opengl(write\_still=True) # Uses OpenGL render for Workbench mode

# Restore original engine after rendering

bpy.context.scene.render.engine = original\_engine

# \*\*Restore overlays after rendering\*\*

for area in bpy.context.screen.areas:

if area.type == 'VIEW\_3D':

for space in area.spaces:

if space.type == 'VIEW\_3D':

space.overlay.show\_overlays = True # Restore UI overlays

# Restore original render settings

bpy.context.scene.render.engine = original\_engine