TUTORIAL 8 – GENERATING AND MANIPULATING 3D CONTENTS WITH VARIOUS EFFECTS

Objectives:

- Learn how to implement 3D effects
- Add the effects and animate objects

Preparation:

- Fully install Blender 4 or higher

Here's a Blender Python script that creates five moving objects with a focus on shading, 3D optical illusions, terrain effects, and digital sculpting. The script generates a terrain-like object (mountain), two spheres, and two cubes, each showcasing various effects and animations.

Blender Python Script

1. **Open Blender.**
2. **Switch to the "Scripting" workspace.**
3. **Create a new text file in the text editor.**
4. **Copy and paste the following script:**

```
import bpy
import bmesh
import math
import random

# Function to create a terrain-like object (mountain)
def create_terrain(location):
    bpy.ops.mesh.primitive_plane_add(size=5, location=location)
    plane = bpy.context.object

    # Subdivide the plane to create more geometry
    bpy.ops.object.mode_set(mode='EDIT')
    bpy.ops.mesh.subdivide(number_cuts=20)
    bpy.ops.object.mode_set(mode='OBJECT')
    # Modify vertex heights to create terrain
    bpy.ops.object.mode_set(mode='EDIT')
    bm = bmesh.from_edit_mesh(plane.data)
    for v in bm.verts:
        v.co.z += math.sin(v.co.x * 2) * 0.5 + random.uniform(-0.5,
0.5)  # Terrain effect
    bmesh.update_edit_mesh(plane.data)
    bpy.ops.object.mode_set(mode='OBJECT')

    # Create a material with shading
    material = bpy.data.materials.new(name="TerrainMaterial")
    material.use_nodes = True
    nodes = material.node_tree.nodes
    links = material.node_tree.links

    # Clear default nodes
    for node in nodes:
        nodes.remove(node)

    # Create shader nodes
    output_node = nodes.new(type='ShaderNodeOutputMaterial')
    bsdf_node = nodes.new(type='ShaderNodeBsdfPrincipled')
    noise_node = nodes.new(type='ShaderNodeTexNoise')

    # Configure noise texture for terrain shading
    noise_node.inputs['Scale'].default_value = 3.0
    noise_node.inputs['Detail'].default_value = 2.0

    # Link nodes
    links.new(noise_node.outputs['Fac'], bsdf_node.inputs['Base
Color'])
    links.new(bsdf_node.outputs['BSDF'],
output_node.inputs['Surface'])

    # Assign material to the terrain
    plane.data.materials.append(material)
    return plane
```

```
# Function to create a sculpted sphere with optical illusion
def create_sculpted_sphere(location):
    bpy.ops.mesh.primitive_uv_sphere_add(radius=0.5,
location=location)
    sphere = bpy.context.object

    # Enter Edit Mode to modify the mesh
    bpy.ops.object.mode_set(mode='EDIT')
    bm = bmesh.from_edit_mesh(sphere.data)

    # Modify vertex heights using random displacement
    for v in bm.verts:
        v.co.z += random.uniform(-0.2, 0.2)

    bmesh.update_edit_mesh(sphere.data)
    bpy.ops.object.mode_set(mode='OBJECT')

    # Create a material for the sphere
    material = bpy.data.materials.new(name="SphereMaterial")
    material.use_nodes = True
    nodes = material.node_tree.nodes
    links = material.node_tree.links

    # Clear default nodes
    for node in nodes:
        nodes.remove(node)

    # Create shader nodes
    output_node = nodes.new(type='ShaderNodeOutputMaterial')
    bsdf_node = nodes.new(type='ShaderNodeBsdfPrincipled')
    wave_node = nodes.new(type='ShaderNodeTexWave')

    # Configure wave texture for optical illusion
    wave_node.inputs['Scale'].default_value = 10.0
    wave_node.inputs['Distortion'].default_value = 5.0

    # Link nodes
    links.new(wave_node.outputs['Color'], bsdf_node.inputs['Base
Color'])
```

```
    links.new(bsdf_node.outputs['BSDF'],
output_node.inputs['Surface'])

    # Assign material to the sphere
    sphere.data.materials.append(material)
    return sphere

# Function to create a sculpted cube
def create_sculpted_cube(location):
    bpy.ops.mesh.primitive_cube_add(size=1, location=location)
    cube = bpy.context.object

    # Enter Edit Mode to modify the mesh
    bpy.ops.object.mode_set(mode='EDIT')
    bm = bmesh.from_edit_mesh(cube.data)

    # Modify vertex heights using random displacement
    for v in bm.verts:
        v.co.z += random.uniform(-0.5, 0.5)

    bmesh.update_edit_mesh(cube.data)
    bpy.ops.object.mode_set(mode='OBJECT')

    # Create a material for the cube
    material = bpy.data.materials.new(name="CubeMaterial")
    material.use_nodes = True
    nodes = material.node_tree.nodes
    links = material.node_tree.links

    # Clear default nodes
    for node in nodes:
        nodes.remove(node)

    # Create shader nodes
    output_node = nodes.new(type='ShaderNodeOutputMaterial')
    bsdf_node = nodes.new(type='ShaderNodeBsdfPrincipled')
    noise_node = nodes.new(type='ShaderNodeTexNoise')

    # Configure noise texture for shading
    noise_node.inputs['Scale'].default_value = 5.0

    # Link nodes
    links.new(noise_node.outputs['Fac'], bsdf_node.inputs['Base
Color'])
    links.new(bsdf_node.outputs['BSDF'],
output_node.inputs['Surface'])

    # Assign material to the cube
    cube.data.materials.append(material)
    return cube

# Clear existing objects
bpy.ops.object.select_all(action='DESELECT')
bpy.ops.object.select_by_type(type='MESH')
bpy.ops.object.delete()

# Create the terrain and other objects
terrain = create_terrain(location=(0, 0, 0))
objects = [terrain]

# Create spheres and cubes
for i in range(1, 5):
    if i % 2 == 0:
        obj = create_sculpted_sphere(location=(i * 2, 0, 0))
    else:
        obj = create_sculpted_cube(location=(i * 2, 0, 0))
    objects.append(obj)

# Animate the objects
for obj in objects:
    frame_start = 1
    frame_end = 100
    obj.keyframe_insert(data_path="location", frame=frame_start)
    obj.location.y += 2 * math.sin(math.radians(obj.location.x * 10))
# Sinusoidal motion
    obj.keyframe_insert(data_path="location", frame=frame_end)

# Set scene properties for animation playback
```

```
bpy.context.scene.frame_start = 1
bpy.context.scene.frame_end = 100
```

How to Run the Script

1. After pasting the code, click on "Run Script" in the Blender text editor.
2. The script will create five moving objects: a terrain-like structure, two spheres, and two cubes, each demonstrating various effects.

Explanation of the Code

- **Creating the Terrain**: The `create_terrain` function generates a subdivided plane to simulate terrain, applying vertex displacements for height variations and assigning a noise texture for shading.
- **Creating Sculpted Objects**: The `create_sculpted_sphere` and `create_sculpted_cube` functions create spheres and cubes with random height displacements, mimicking digital sculpting.
- **Optical Illusions and Shading**: The spheres use wave textures for optical effects, while cubes use noise textures for shading.
- **Animation**: All objects move in a sinusoidal pattern along the Y-axis, creating dynamic motion. Keyframes are set for their positions at the start and end of the animation.
- **Clear Existing Objects**: The script removes any existing mesh objects to ensure a clean workspace.