Lab 1-1

Young Min Kim

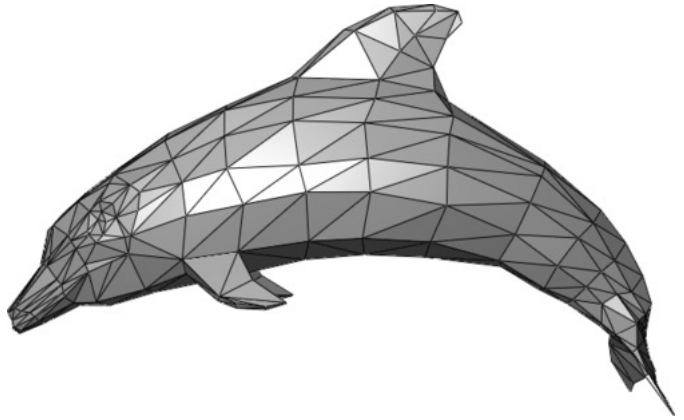
2022.07.13.

3D LAB VISION

Table of Contents

1. Mesh Data Structure

- In this part, you will create a cube with triangle mesh from scratch.
- Triangle Mesh is consists of vertices and faces.
- In other words, you can make a cube by creating vertices and faces.



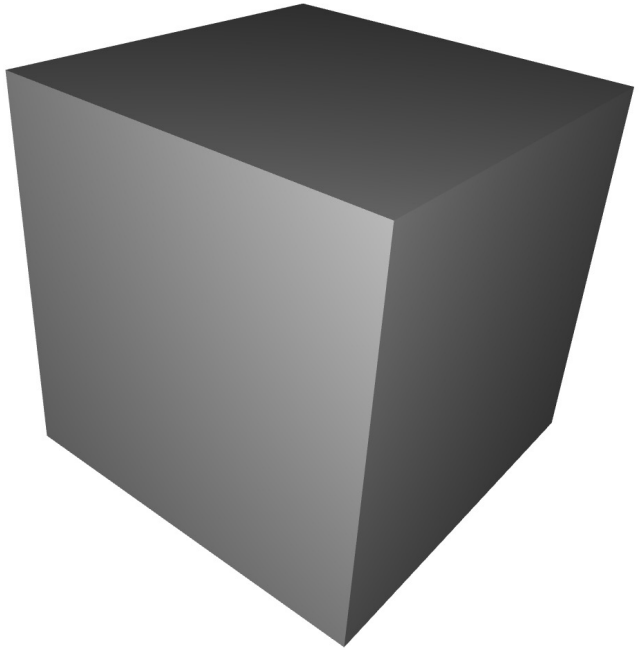
1. Mesh Data Structure

- Open '`lab1-1/python/mesh_data_structure.py`'
- In this example, you will make a zero-centered cube with length 2.
- Fill the appropriate values in the vertices array and triangles array.

```
cube = o3d.geometry.TriangleMesh()
############################################################
#########################  TODO  ###########################
############################################################
vertices = np.array([[],
                     [],
                     [],
                     [],
                     [],
                     [],
                     [],
                     []])

triangles = np.array([[],
                      [],
                      [],
                      [],
                      [],
                      [],
                      [],
                      [],
                      [],
                      [],
                      [],
                      []], dtype=np.int32)
############################################################
############################################################
```

1. Mesh Data Structure

- If you enter the correct answer, you can see a cube on your screen.



Expected Output

```
############################################################
#######################  TODO  #############################
############################################################
vertices = np.array([[-1., -1., -1.],
                     [1., -1., -1.],
                     [-1., -1., 1.],
                     [1., -1., 1.],
                     [-1., 1., -1.],
                     [1., 1., -1.],
                     [-1., 1., 1.],
                     [1., 1., 1.]])

triangles = np.array([[4, 7, 5],
                      [4, 6, 7],
                      [0, 2, 4],
                      [2, 6, 4],
                      [0, 1, 2],
                      [1, 3, 2],
                      [1, 5, 7],
                      [1, 7, 3],
                      [2, 3, 7],
                      [2, 7, 6],
                      [0, 4, 1],
                      [1, 4, 5]], dtype=np.int32)
############################################################
############################################################
```

Example Code

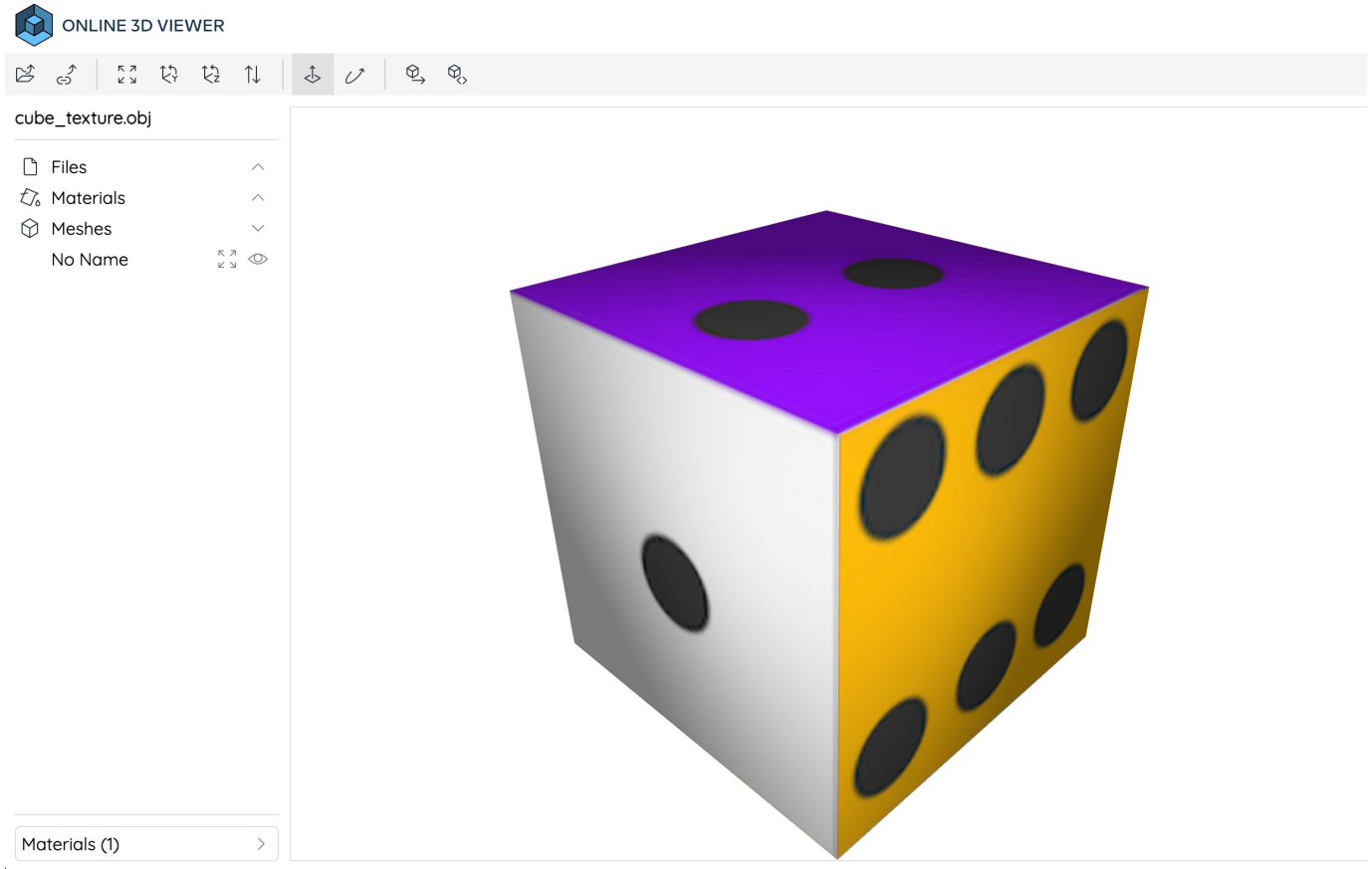
1. Mesh Data Structure

- Now, let's add a texture to our cube.
- One can map the 2D texture to triangle surface. (i.e. uv mapping)
- Fill in the v_uv array which contains the uv coordinate of each vertices in triangles.
- (The size of array should be 3|F| x 2, when |F| is number of triangles.)

```
"""
2) Add texture to a mesh
"""
text = cv2.imread('../data/cube_texture.png')

################################################################
#########################  TODO  ###############################
################################################################

v_uv = np.array([])

################################################################
################################################################
```

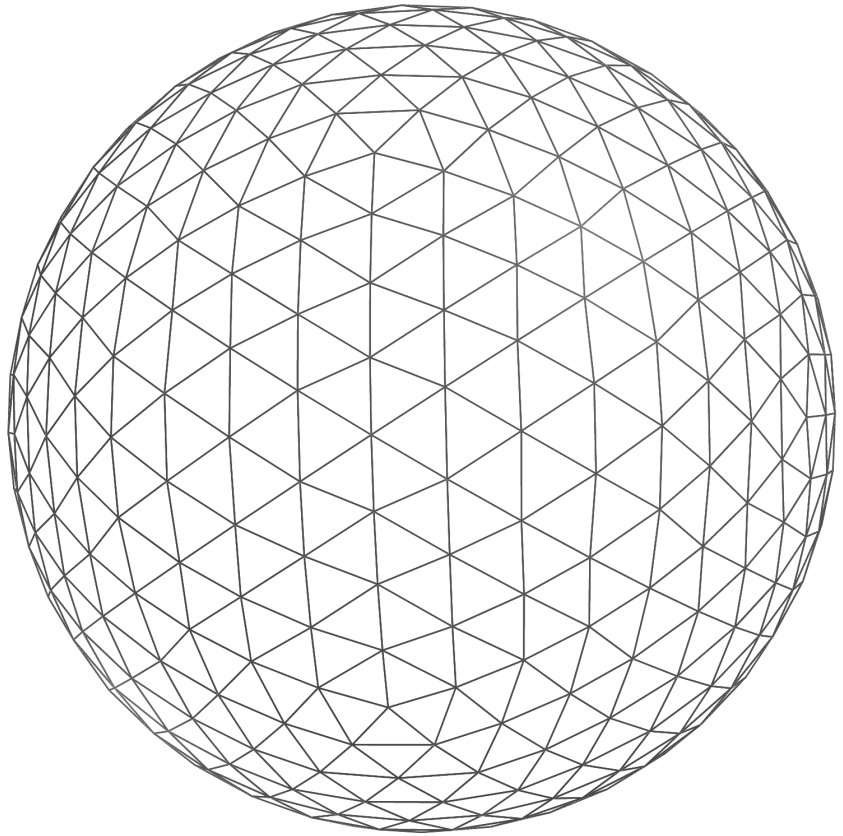
1. Mesh Data Structure

- You can obtain `cube_texture.mtl`, `cube_texture_0.png`, `cube_texture.obj` in `lab1-1/data` directory by `running mesh_data_structure.py`
- You can see the textured cube by online 3D visualization tool. (https://3dviewer.net)
- Drag and Drop the three files you created



2. Point Cloud Sampling from Mesh

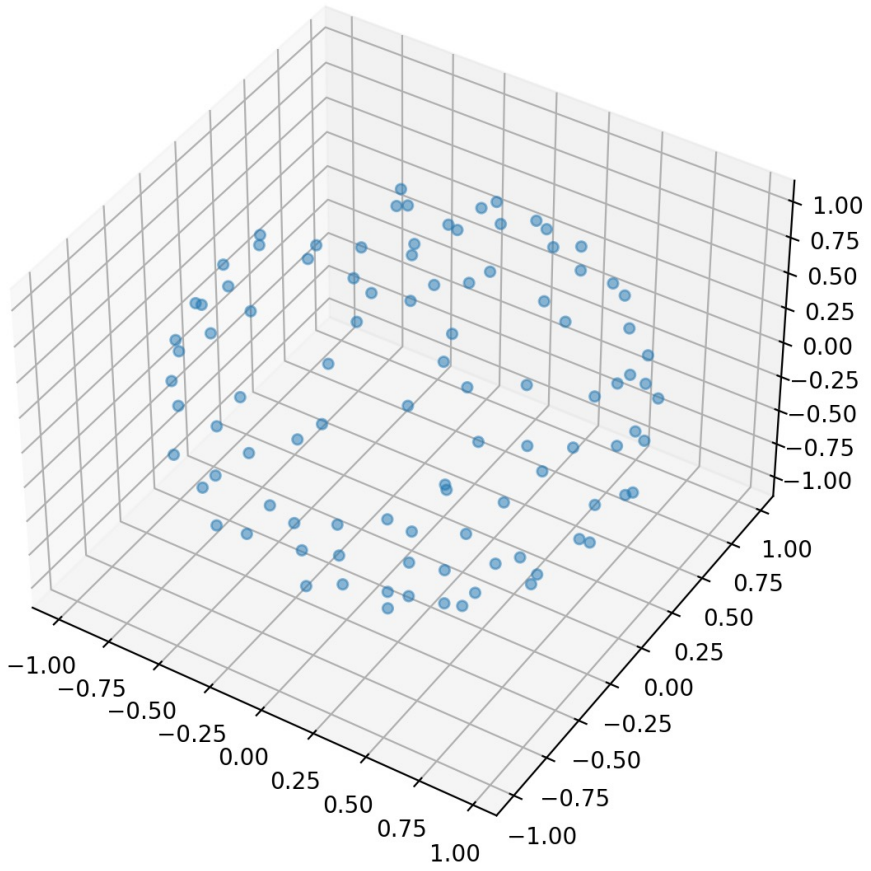
- In this example we will obtain point cloud by sampling points from a triangle mesh of sphere.
  - Sample Point Cloud from Vertices
  - Sample Point Cloud from Surfaces (Triangles)



2. Point Cloud Sampling from Mesh

- We can see triangular mesh as a graph. (vertex -> node, edge -> edge)
- We will collect vertex with Farthest Point Sampling algorithm.
  1. Add an arbitrary vertex into Point Set
  2. Perform Dijkstra's algorithm
  3. Add Farthest Node (Vertex) into Point Set
  4. Iterate
- You can use `networkx` library to perform Dijkstra's algorithm

2. Point Cloud Sampling from Mesh



Expected Output

```python
points = []

##################################################################
#########################  TODO  #################################
##################################################################

# 1) start with an arbitrary vertex.
points.append(0)

# 2) perform Dijkstra's algorithm.
for it in range(vertex_sample_num):
    path_length = nx.multi_source_dijkstra_path_length(g, points)
    nodes = np.fromiter(path_length.keys(), dtype=np.int32)
    lengths = np.fromiter(path_length.values(), dtype=np.int32)

    # 3) add farthest node into points.
    points.append(nodes[np.argmax(lengths)])

##################################################################
##################################################################

sampled_points = sphere.vertices[points]
```

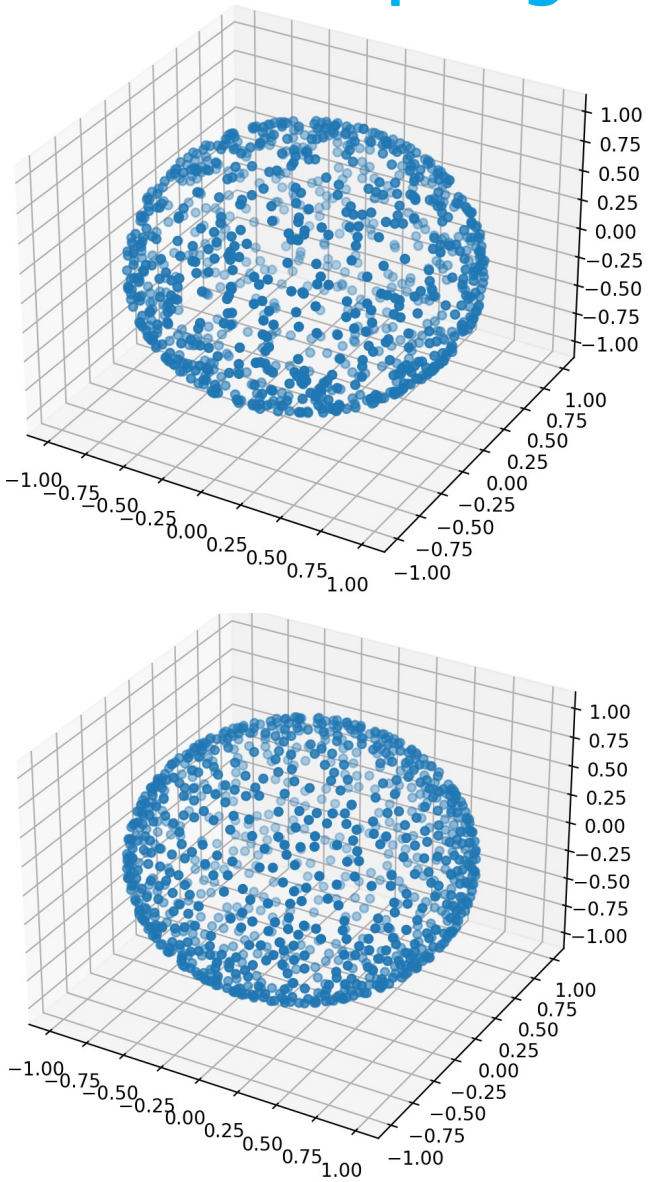
Example Code

2. Point Cloud Sampling from Mesh

1. Random Sampling from Surface
2. Uniform Sampling from Surface

- You can use surface sampling function from `trimesh` library.

2. Point Cloud Sampling from Mesh



Expected Output

```python
# 2. Random Sample from Triangle
random_points = trimesh.sample.sample_surface(sphere, surface_sample_num)[0]
# 3. Sample evenly.
uniform_points = trimesh.sample.sample_surface_even(sphere, surface_sample_num)[0]
```

Example Code