JAVA 2D/3D
Project 2

Takuto ODAIRA　m5281036

Goals

1. Load a 3D point-cloud using the xyz file format
2. Visualize the 3D point-cloud using OpenGL
3. Implement surface reconstruction with radial basis functions (RBF)
4. Implement the Marching Cubes algorithm to convert the zero level-
5. set of the RBF to a triangle mesh
6. Implement data-structures for manipulating triangle meshes
7. Visualize the reconstructed surface using OpenGL

Repository Structure

```
project2
├ data (xyz files)
├ docs
├ src
    ├ Config.java
    ├ MainApp.java
    ├ Point3D.java
    ├PointCloudLoader.java
    ├SurfceDrawer.java
    ├SurfceDrawerKeyListener.java
```

1. Loading Point Cloud

1. In “PointCLoudLoader” class
   - load line by line
   - stores in [p_x, p_y, p_z]
   - same way for the norms

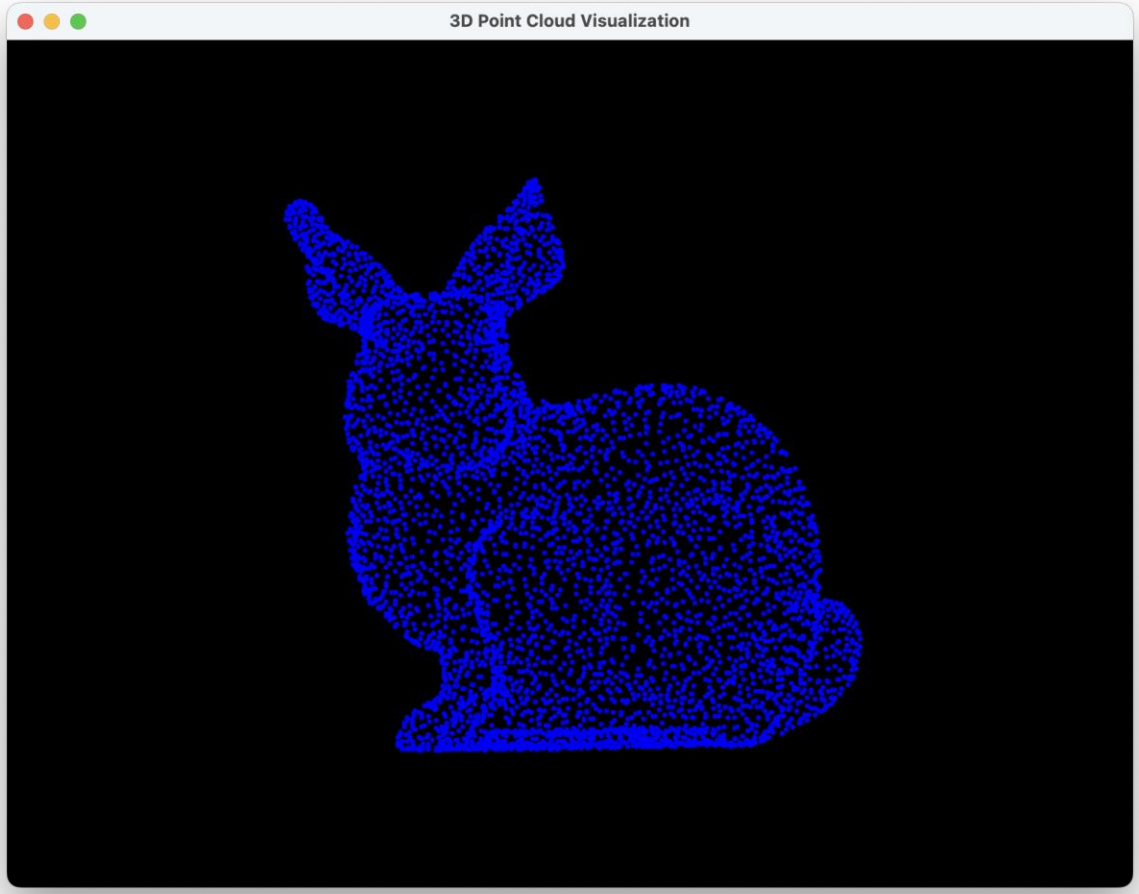
```
while ((line = br.readLine()) != null) {
    String[] coords = line.trim().split(regex:" "); // split line by space

    // Store first 3 numbers as coordinates (x, y, z)
    float p_x = Float.parseFloat(coords[0]);
    float p_y = Float.parseFloat(coords[1]);
    float p_z = Float.parseFloat(coords[2]);
    points.add(new Point3D(p_x, p_y, p_z));
}
```

2. Visualize the 3D point-cloud using OpenGL

- In “SurfaceDrawer” class
  - Display vertices in 3D space in 2D (by only using p_x and p_y)

```
// project 3D objects into 2D display
int screenX = (int) (p_x / scale + offsetX);
int screenY = (int) (-p_y / scale + offsetY);
g2d.fillOval(screenX, screenY, width:4, height:4);
```



2. Visualize the 3D point-cloud using OpenGL

- Manipulation of the vertices
  - Rotation, scaling, moving is implemented
  - Key input is read in “SurfaceDrawerKeyListener” class

```
// controlling the display of the points
for (Point3D point : pointCloud) {
    double p_x = point.getX();
    double p_y = point.getY();
    double p_z = point.getZ();

    // roatate in x axis
    double new_Y = p_y * Math.cos(rotationX) - p_z * Math.sin(rotationX);
    double new_Z = p_y * Math.sin(rotationX) + p_z * Math.cos(rotationX);
    p_y = new_Y;
    p_z = new_Z;

    // roatate in y axis
    double new_X = p_x * Math.cos(rotationY) + p_z * Math.sin(rotationY);
    p_z = -p_x * Math.sin(rotationY) + p_z * Math.cos(rotationY);
    p_x = new_X;

    // project 3D objects into 2D display
    int screenX = (int) (p_x / scale + offsetX);
    int screenY = (int) (-p_y / scale + offsetY);
    g2d.fillOval(screenX, screenY, width:4, height:4);
}
```

3. Computing RBF

- Implemented in “RBFInterpolation” class

  (not successfully done)

$$\hat{f}(\boldsymbol{x}) = p(x) + \sum_{i=1}^{n} \lambda_i \varphi\left(\left\|\boldsymbol{x} - \boldsymbol{x}_i\right\|\right)$$

$p(x)$: Linear polynomial

$\varphi(r) = r$

Conclusion

1. Load a 3D point-cloud using the xyz file format ✅
2. Visualize the 3D point-cloud using OpenGL ✅
3. Implement surface reconstruction with radial basis functions (RBF)
4. Implement the Marching Cubes algorithm to convert the zero level-set of the RBF to a triangle mesh
5. Implement data-structures for manipulating triangle meshes
6. Visualize the reconstructed surface using OpenGL

References

OpenGL methods:
https://registry.khronos.org/OpenGL-Refpages/gl4/

Manipulation of dots (shape):
http://www.maroon.dti.ne.jp/koten-kairo/works/Java3D/transform2.html (basic transformation method of objects)

AddKeyListener:
https://qiita.com/derodero24/items/9ea025b92ac61edf0aa4