Java 2D&3D Graphics Project2

m5271506 Kiyohiro Murai

2023-2-2

Contents

1 Classes

Please see code for details. Here it will be written an overview of each class.

1.1 ViewerMain.java

A class with a main method. Has a JFrame. It composes the UI used by users.

1.2 ViewerPanel.java

A panel for displaying 3D models. The root object, Behavior, and Light are set. If you specify the file path and view mode, the specified 3D shape will be displayed.

1.3 MeshManager.java

Handles vertices, triangles, and graphs based on the triangle information in the obj file. Calculates the shortest path between vertices and normal vectors.

1.4 Face.java

Class that handles triangular meshes

1.5 Vector.java

Class that handles vertices

1.6 Dijkstra.java

A class that uses Dijkstra's method to find the shortest path

1.7 Graph.java

Graph class used when searching for the shortest path

1.8 PointCloud.java

Class that displays PointCloud. When a vertex is selected, it has a method that calculates the geodesic distance.

1.9 WireFrame.java

Class that displays WireFrame

1.10 FilledWireFrame.java

Class that displays FilledWireFrame

1.11 FlatShading.java

Class that displays FlatShading

1.12 SmoothShading.java

Class that displays SmoothShading. When a vertex is selected, it has a method that calculates the geodesic distance.

1.13 ColorMap.java

Class that handles colormaps

1.14 ColorMapManager.java

A class that imports and manages colormaps from csv

1.15 ObjReader.java

Class that reads obj files

2 FlatShading algorithm

3 SmoothShading algorithm

4 Geodesic Distance algorithm

4.1 For data with triangle mesh information

A graph was constructed based on the information about the edges of the triangular mesh. Starting from the vertex selected by the user, the shortest path to all other vertices is calculated using Dijkstra's algorithm. Since all vertices are assigned a distance from the starting point, a color is set from the color map based on that.

4.2 For PointCloud with only vertex information

A graph was created based on the k-nearest neighbor algorithm. As the number of vertices increases, the amount of calculation increases and it takes more time. With the test data provided, it took a huge amount of time to run.

5 Problems

5.1 For PointCloud with only vertex information

The k-nearest neighbor algorithm took an enormous amount of time to run on the test data provided. The use of kd tree can be considered for improvement. But I couldn't implement it.

5.2 Mouse translation problem

I wanted to use the mouse to move a shape in translation, but for some reason the behavior didn't work properly. The officially stated default key (shift key) was assigned a rotation function, but the movement function did not work correctly.
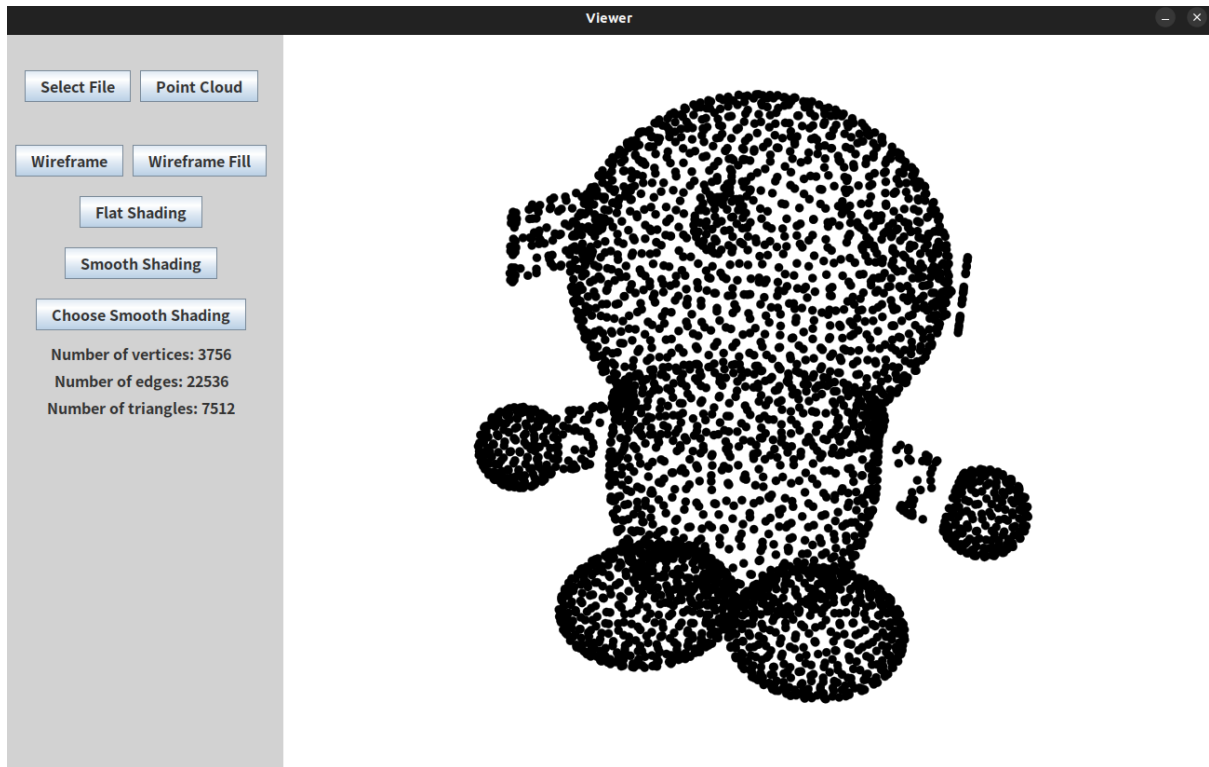
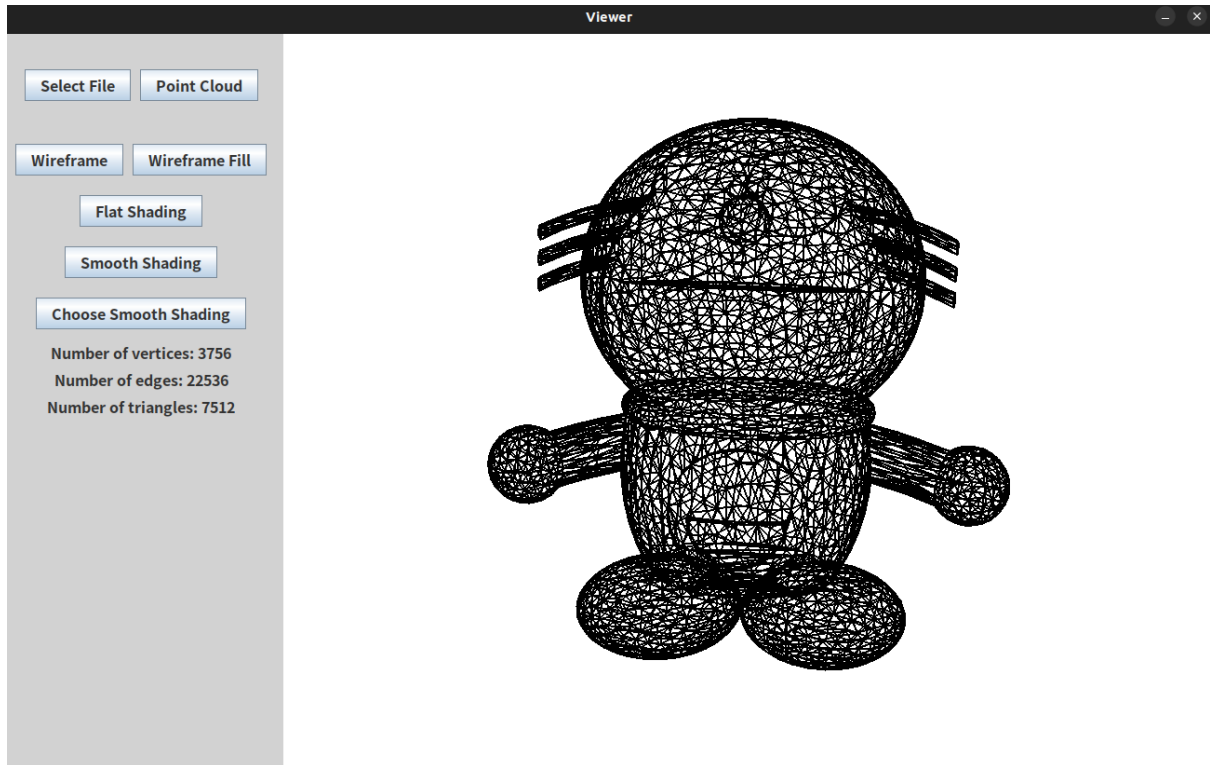# 6 Screenshots

Figure 1: Screenshot of PointCloud
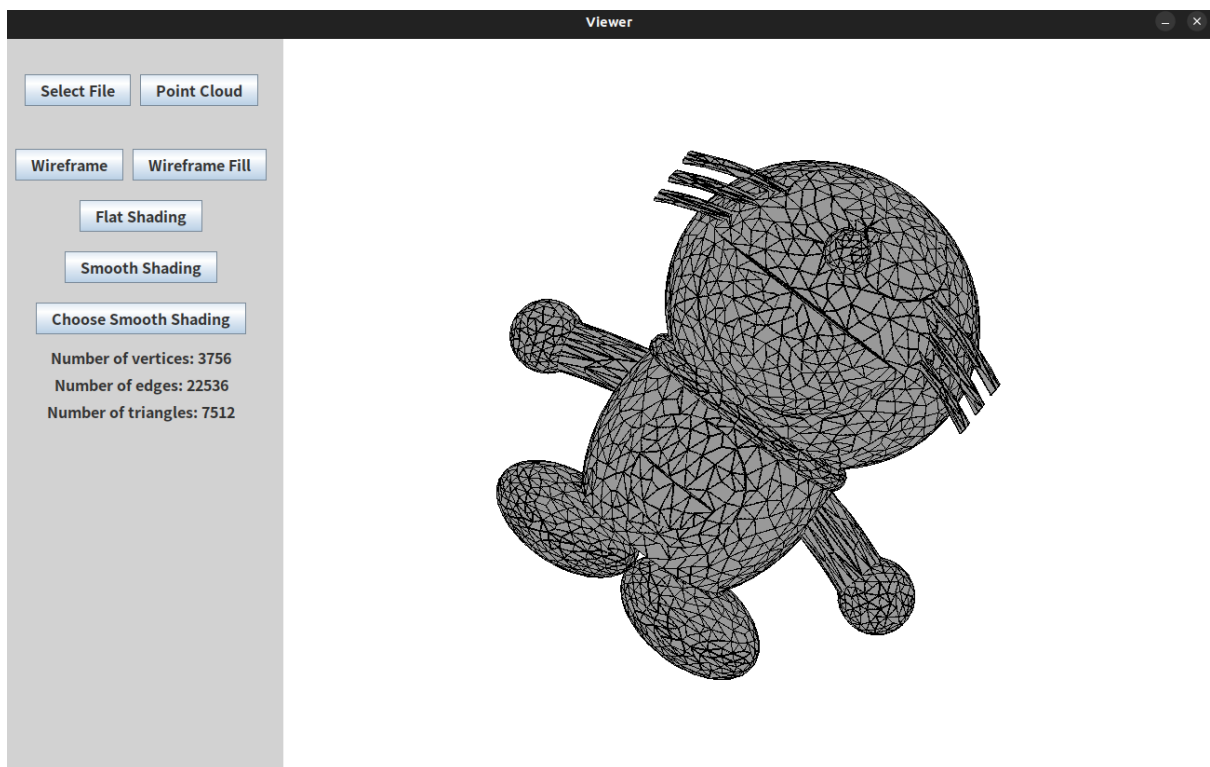
Figure 2: Screenshot of WireFrame

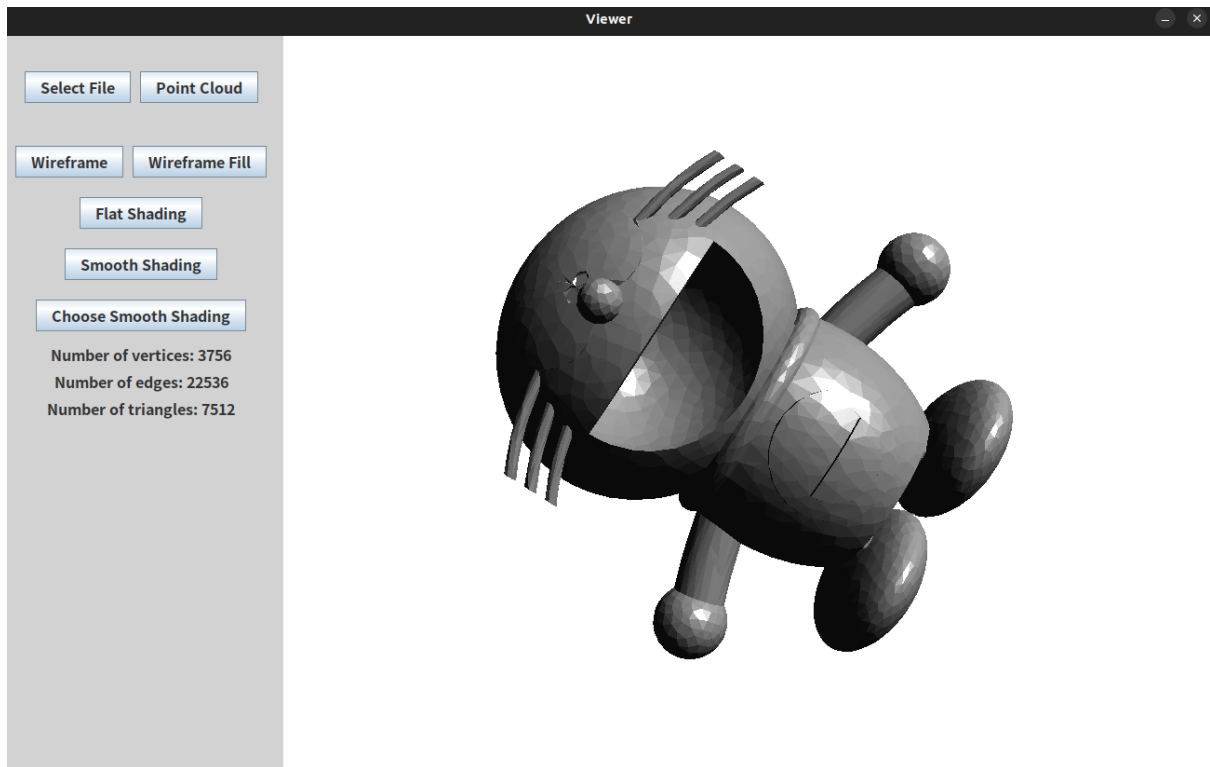Figure 3: Screenshot of FilledWireFrame

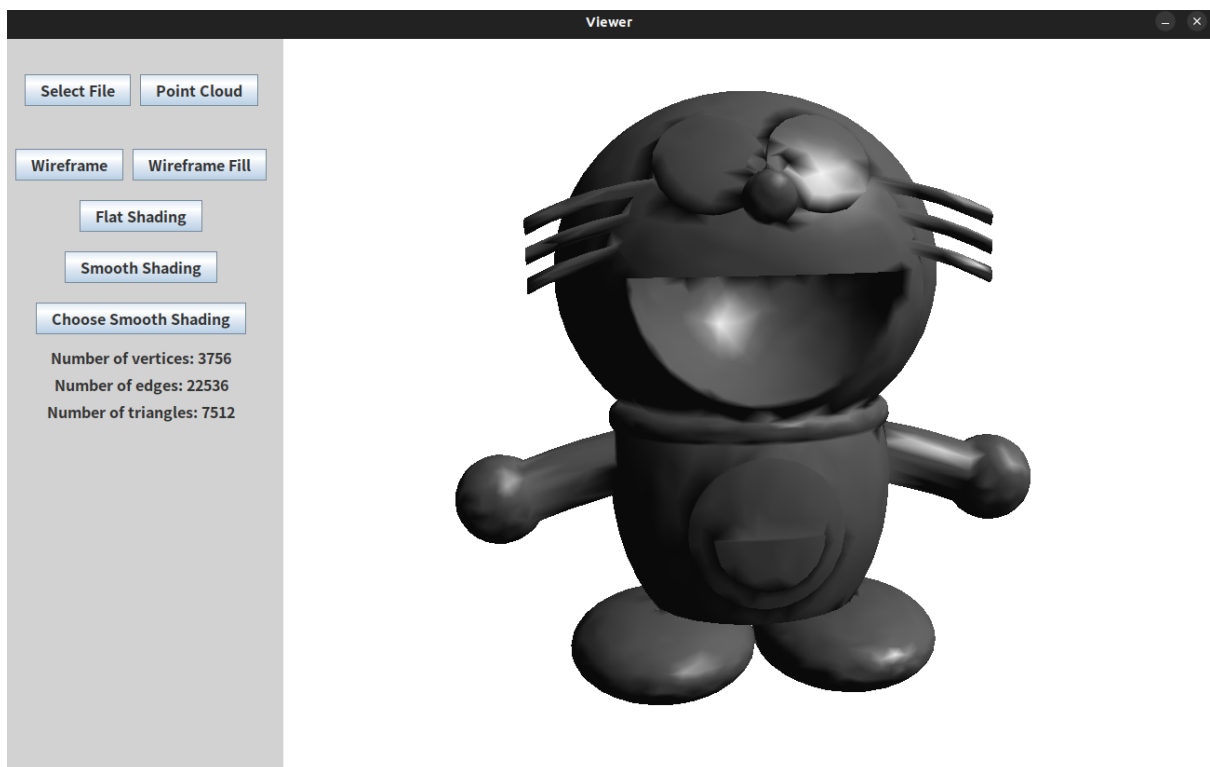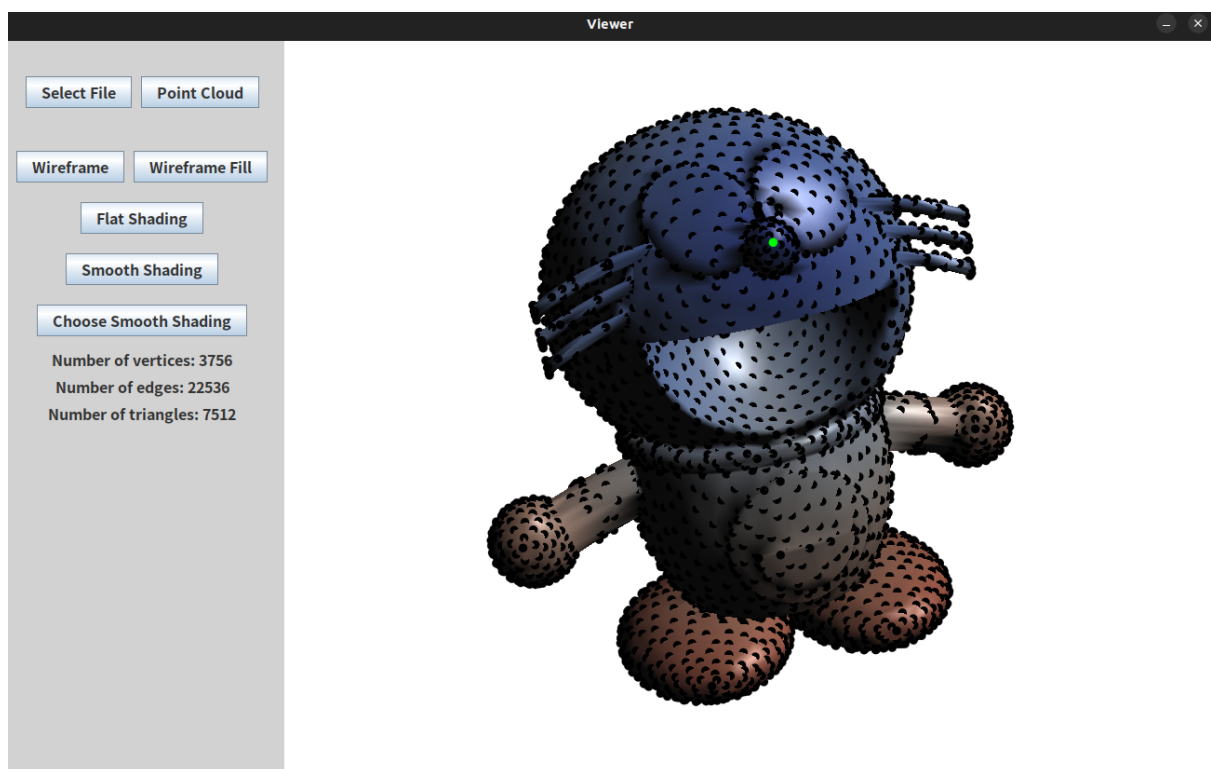Figure 4: Screenshot of FlatShading

Figure 5: Screenshot of SmoothShading

Figure 6: Screenshot of Geodesic Distance