CS241 Project 3

Synopsis

Implement the **Triangle Scanline** algorithm with **Z-buffer** depth testing.

Instructions

Use the previous project (the clipper) as a starting point for this project. At each call to **DrawScene** (in **render.cpp**)

- ensure you have a Z-buffer of the size specified by **DrawScene's width** and **height** parameters.
- Extract all the polygons from the scene,
- transform vertices (from Project 1),
- clip to the view frustum (Project 2),
- transform to pixel coordinates,
  $$(x, y, z, w) \rightarrow (x/w * width/2 + width/2,\ \ y/w * height/2 + height/2,\ \ z/w),$$
- split polygon into triangles, and
- render each triangle to the screen via the Z-buffer/scanline algorithm.

Several modifications will need to be made to the current framework to put OpenGL into a pixel drawing mode rather than a polygon drawing mode. (See the provided example **render.cpp** for details.)

- Use **glDisable(GL_DEPTH_TEST)** to disable OpenGL's depth testing.
- Use **gluOrtho2D(0,width,0,height)** to setup individual pixel addressing.
- Use **glBegin(GL_POINTS)** to specify pixel drawing.
- Use **glPointSize(1.0)** to specify that points will be a single pixel in size.
- Use **glMaterialfv(...)** and **glNormal3fv(...)** to setup the lighting parameters.
- Use **glVertex2f(px, py)** to draw a single pixel at pixel coordinate (px,py).

Project Report

As always, submit a **succinct** project report with your project, describing anything I need in order to understand your implementation. Place your project report in a text file named **report.txt** (or **report.doc** or **report.odt** if you (unnecessarily) insist on producing a formatted document).

To submit

Create a zip file containing only your report, and your source code ***.cpp, *.h** and ***.vcproj** files ***and nothing else.*** Submit the **zip** file via the Moodle class web page.

Grading Basis

Grading will be on a 100 point basis, with the following distribution:

- **Accuracy:** 60%, judgment based on looking at the results on the screen.
- **Efficiency:** 20%, judgment based on frame rate.
- **Code:** 10%, judgment based on looking at the code for proper implementation and reasonable coding and commenting practices.
- **Project report:** 10%, judgment based on how succinctly and accurately the project report describes your implementation.