Lab 4: Flat Shaded Rasterizer using Edge Equations

Topics Covered

- Viewport transform
- Triangle Rasterization based on Edge Equations
- Point Sampling using Incremental Calculations
- Culling Back-Facing Triangles

Prerequisites

- Correct implementation of Lab 2
- Triangle rasterization handout and presentation deck used in class lectures

Implementation Details

1. Implement a function `Pbo::render_triangle0()` with declaration:

```
struct Pbo {
/*
other declarations here ...
static void render_linedda(int x1, int y1, int x2, int y2, Pbo::Color clr);
static void render_linebresenham( ... );
*/

  /*
  Implements a flat shaded triangle rasterizer using edge equations.
  Whether fragment (x, y) is rasterized or not is based on the results of
  point sampling the fragment at its center (x+0.5, y+0.5).
  Top left tie-breaking rule must be implemented.
  Parameters p0, p1, and p2 represent vertices of triangle whose x- and y-
  values are specified in window coordinates.
  Front-facing triangles are assumed to have counterclockwise winding.
  Back-facing triangles must be culled.
  */
  static void render_triangle0(glm::vec3 const& p0,
                               glm::vec3 const& p1,
                               glm::vec3 const& p2,
                               Pbo::Color clr);
};
```

2. In `Pbo::render()`, replace the call to `Pbo::render_linebresenham()` with a call to `Pbo::render_triangle0()`.
3. Amend `Pbo::init()` so that `DPML::parse_obj_mesh()` will parse an OBJ file named `lab4-test.obj`.

4. Declare and define `Pbo::get_backface_cnt()` to return the number of back-facing triangles culled by the emulator. This function will be used to print statistics to the window title bar in `update()` in `./src/main.cpp`.

Submission Guidelines

1. Check the assignment web page for the submission deadline.
2. ***Only*** files `pbo.h` and `pbo.cpp` must be submitted - additional files will not be used for grading. Changes to `main.cpp`, `app.cpp`, `glslshader.cpp`, and shader source files are not required nor necessary for this submission.
3. Open the rubric page (in Word format or in PDF format) and check the portions that are complete. Sign and date the bottom of the page. The document can be submitted as a PDF or JPEG or Word document. I cannot grade your submission if the signed rubrics page is missing.
4. Copy submission source files and signed rubric page into a submission folder that is named using the following convention: <**student login name>_<class>_<lab#>.zip**. Assuming your login is **foo**, the course obviously is **cs250**, the lab is **4**, then the zipped folder would be named: **foo_cs250_lab4.zip**.