Intro to Graphics Spring 2017 – Lab 2

Due Date: January 25th

Lab Objective

- Implement very simple ray tracing and rasterization.
- Visualize barycentric coordinates.

In this lab, you will experiment with two different ways to render a triangle. First, you will render a triangle using ray tracing. Second, you will render a triangle using rasterization. For both experiments, you will give color to the triangle using barycentric coordinates.

Lab Setup

To use Qt Creator in the labs, you can use the following instructions:

1. Download the lab2_qtproj.zip file from the Connex assignment.
2. Unzip and open lab2.qbs inside it using Qt Creator.
3. Compile and run! ☺

If you're using Visual Studio, you can do the same thing as above with lab2_vs2015.zip.

If you want to build the code with another development environment than Qt Creator or Visual Studio:

1. Download main.cpp and stb_image_write.h from the Connex assignment.
2. Put them together in the same folder.
3. Build main.cpp (eg. run g++ on it.)

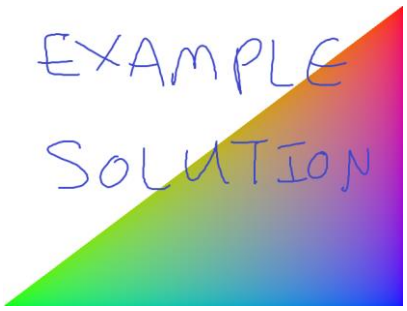
Submission Guidelines

Implement the missing code in main.cpp. Submit your code and a screenshot of your results to Connex.

You only need to submit main.cpp and the screenshot, not any project files or executables.

Expected Result

If you implement the code correctly, the program should output the following image (without text):



The program is configured to automatically open the output image when it finishes running.

Lab Instructions

The lab is in two parts: The ray tracer, and the rasterizer.

Part 1: Ray Tracer

To work on this part of the project, uncomment the line "raytrace();" and comment the line "rasterize();" inside main().

Ray tracing can be used to render an image by computing the geometric intersection of a ray and a primitive. One ray is generated per pixel, and intersected with the primitives. In this case, a simple triangle is found by shooting rays orthographic to the image plane.

You must implement the TODOs within the intersect() function inside main.cpp.

This function should intersect a ray with a triangle.

- If the ray **does not** intersect the triangle, return false.
- If the ray **does** intersect the triangle:
    1. Write the barycentric coordinate of the intersection in *bary
    2. Return true.

Consult slide 3 and 12 of the ray3.pdf slide deck in the course notes for the math.

Part 2: Rasterizer

To work on this part of the project, uncomment the line "rasterize();" and comment the line "raytrace();" inside main().

Rasterization exploits the linear relationship between screen space and the barycentric coordinates of a projected triangle in order to implement an efficient triangle coverage test and attribute interpolation. In this case, a single triangle is projected orthographically onto the screen.

You must implement the TODOs within the rasterize() function inside main.cpp.

First, you must compute the double projected area of the triangle. Note that this can be accomplished with the following formula:

$$2 * \text{Area}(\Delta ABC) = \|(B - A) \times (C - A)\|$$

Next, you must compute the barycentric coordinates at the top-left of the screen. (Yes, even though that location is outside the triangle!) Use the "Area Method" in slide 12 of ray3.pdf. You may have to flip the sign of the xyz of this barycentric coordinate, depending on the order of your cross product.

Finally, you must compute the screen-space derivatives of the barycentric coordinates. To do this, you must compute $\frac{d\alpha}{dx}, \frac{d\alpha}{dy}$ and $\frac{d\beta}{dx}, \frac{d\beta}{dy}$ and $\frac{d\gamma}{dx}, \frac{d\gamma}{dy}$. These can be obtained by finding the partial derivatives of the barycentric coordinates. For your convenience, here they are:

$$\frac{d\alpha}{dx} = (tr.b - tr.a)_y$$

$$\frac{d\beta}{dx} = (tr.c - tr.b)_y$$

$$\frac{d\gamma}{dx} = (tr.a - tr.c)_y$$

$$\frac{d\alpha}{dy} = -(tr.b - tr.a)_x$$

$$\frac{d\beta}{dy} = -(tr.c - tr.b)_x$$

$$\frac{d\gamma}{dy} = -(tr.a - tr.c)_x$$

Note:

- The variable "dbarydx" is a vec3 containing the first 3 equations in its x/y/z components.
- The variable "dbarydy" is a vec3 containing the second 3 equations in its x/y/z components.
- Make sure you divide dbarydx and dbarydy by $2 * \text{Area}(\Delta ABC)$.
- Like before, you might have to flip the signs depending on the order of your cross products.