Lab Exercise
CSC 471
Introduction to Computer Graphics

CAL POLY
SAN LUIS OBISPO

LAB 1 and 2

Two parts lab 1 and 2

Deadline for both parts the same, but both are small compared to later labs. A good opportunity to gain some course credits!

Exercise Software Rasterizer: Single Triangle

For your first assignment, you will be writing a program to render (draw) an indexed face set (aka polygonal mesh of triangles) as an image via software rasterization. We are continuing to build some basic tools for the assignment.

Goals:

Read in three vertices which represent a triangle. Read in different color values to each vertex. Compute the bounding box of the triangle using your lab code. Paint a red triangle.

Compute the barycentric coordinates for every pixel in the bounding box. For any pixel that falls in the triangle, color that pixel with the interpolated color using the barycentric coordinates to weight the vertex colors. Write out those pixels as a PNG image.

Lab 1 goal: A red-filled triangle.

Lab 2 goal: A color blurred triangle.

Prepare:

On all OS, make sure to be able to see the file endings!
Download “Rasterbase code” (either way CMAKE or Visual Studio) from PolyLearn.
It will be base for both lab parts.

Visual Studio:

Install Visual Studio 2019 IDE “Community” (not Code) on your PC/laptop.
It’s free: https://visualstudio.microsoft.com/
During installation, make sure to check that Visual C++ is checked!
Log in with your Cal Poly account.
Open (double click) the Lab02.sln file with Visual Studio.

CMAKE:

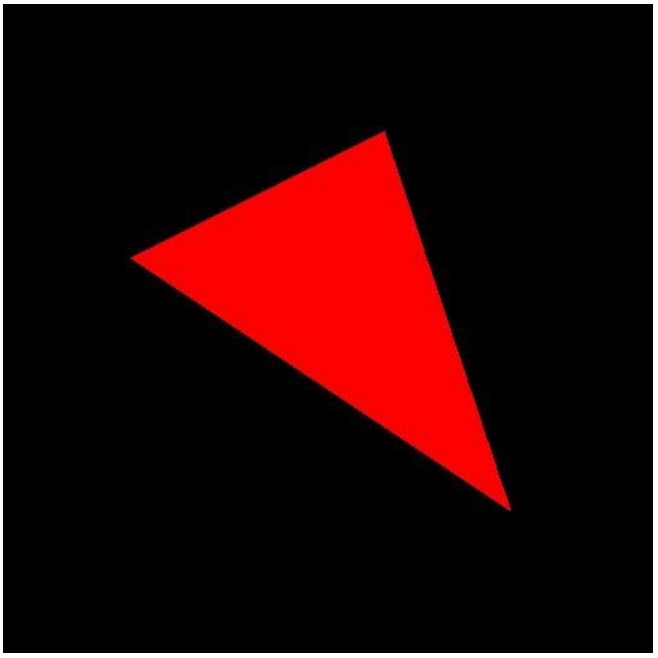
Copy over CMakeLists.txt and the src folder from the lab code on PolyLearn.
Change the name of the project in the CMake file. Build and run the executable as before. E.g., with command line,
> mkdir build > cd build > cmake .. > make -j4 > ./Lab02
With Xcode:
> mkdir build > cd build > cmake
-G Xcode .. > # Open
Lab02.xcodeproj with Xcode

Steps:

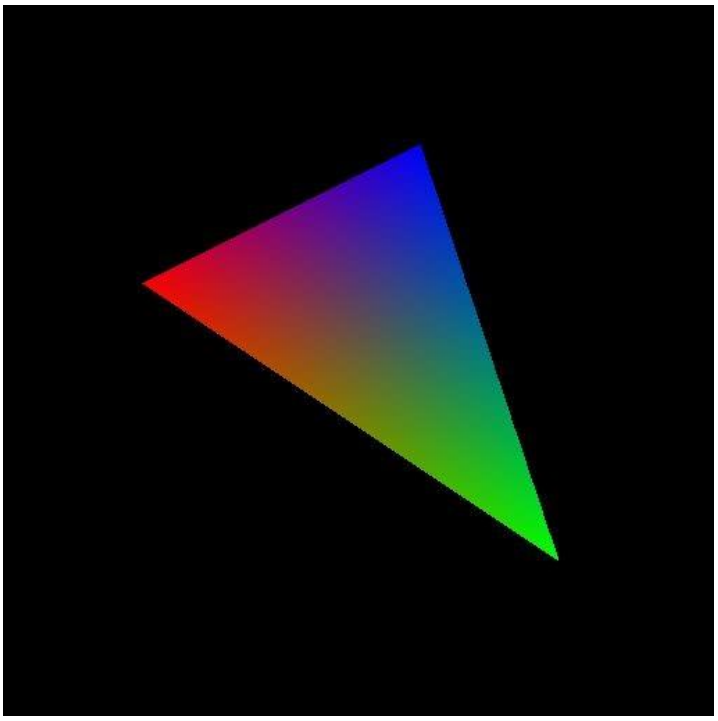
The input is a triangle.OBJ file. (It will be discussed in the lecture)
The output will be a PNG file.

- Map the triangle coordinates into image space.
- Compute the bounding box of the triangle.
- For each pixel inside the bounding box, check if you are inside the triangle or not. Set a red pixel in case of true.

The output should be (and we check this):



For any pixel that will be drawn, its color should be a blend of the colors specified at the triangle vertices.
Choose different colors for each vertex. The easiest would be red/green and blue.
Blend the colors using \alpha, \beta, and \gamma that were computed in the **barycentric coordinate computation** for each pixel. See the example figure below for an idea of what your rasterized triangle should look like:



Lab Check :

100% points for the red triangle .. lab 1.
100% points for the colored triangle .. lab 2.
You must work individually.