 Graphics I – Lab 3

# Objective

The objective of today’s lab will be to implement what was learned in lecture. You will be creating your first triangle primitive in 2D space, hence forth known as NDC (Normalized Device Coordinates). The range of NDC by default goes from -1 and +1 on the X axis, -1 and +1 on the Y axis, -1 and +1 on the Z axis. Keep this in mind when creating the positions for your triangle vertices. Understand that these labs will need to use critical thinking and some creative decisions on your part on how to setup your code. Keep in mind the power point slides give you some direction on how to setup the shaders and use them as function pointers.

# Grading Breakdown

* 25% - Code Cleanup
  + Refactor your code so that it isn’t all in your main.cpp
  + You can make header files, classes, structures. Use critical thinking on how to make your code fit together!
* 50% - Rotating triangle (lines only)
  + This step involves using the **VERTEX** shader alongside your own vertex structure.
* 75% - Rotating triangle (filled to a solid color with white border lines)
  + This step will be introducing the barycentric formula to fill the triangle.
* 100% - Rotating triangle (interpolated colors with white border lines)
  + This step will involve the barycentric formula and the **PIXEL** shader to determine which color to use.

# Above & Beyond

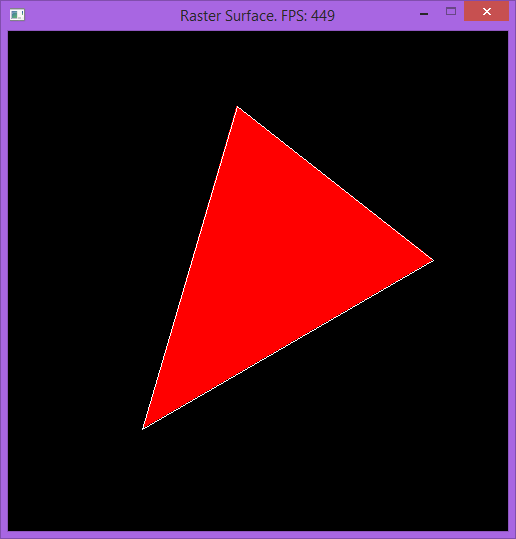
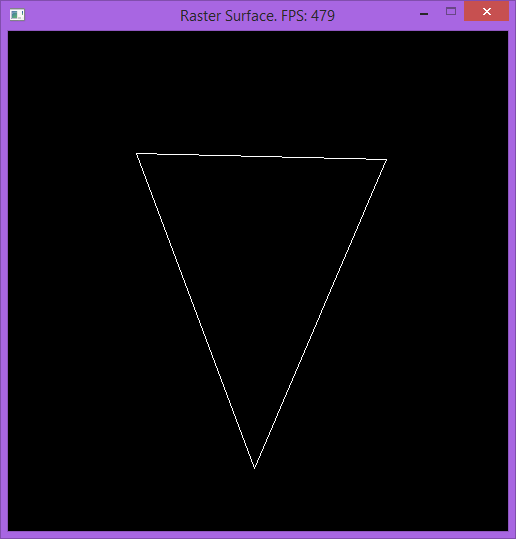
1. Use two (and only two) linear interpolations to perform your Barycentric color interpolation. Use the fixed point blend routine from Lab 1’s Above and Beyond. This can be a significant optimization.
2. Optimize the speed of your triangle fill routine even further by implementing Lari’s parametric triangle algorithm. You may implement your own as well, but it must be reasonably faster than “Better Brute” and cannot simply be plagiarized off of the internet.

# Setup

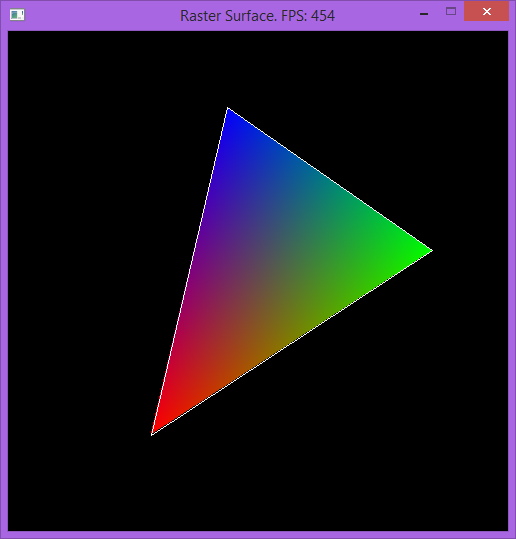
1. Clean up your code. This step involves creating header files, classes, and/or structures that contain specific functions or variables. For example, create a header file that contains your #defines, or create a class that contains your rasterizing functions.
2. Now that your code has been cleaned up, let’s go ahead and create a **VERTEX** structure. The **VERTEX** structure will contain 2 data members. The first will be an array of 4 floats. The second will be an unsigned integer for the color.
3. Create an array of 3 vertex structure objects, and define their information. Make sure to not exceed the bounds of NDC. For the color, make each verex a different value, preferable RED GREEN and BLUE.
4. Let’s keep in mind that we will need to do some coordinate conversions from NDC to SCREEN values to properly plot our pixels. This is a good time to create a function for doing so. Remember, we should keep our code neat, so we will need to place this function in one of the new header files we created.
5. Draw three lines connecting each VERTEX to make a triangle. Additionally we want to rotate each VERTEX by a rotation matrix. This rotation will be handled by a **vertex shader** you create. Continue to rotate the triangle by time. (It should spin continuously).
6. One we have the lines drawing properly, it’s time to fill the triangle. At first you can fill the triangle to a solid color but to get the 100% mark we need to interpolate the color values between all three vertices. In order to fill the triangle we will need to do the barycentric calculations in order to see if our pixel is currently inside the triangle.

# Example Images

50% 75%



100%



# Submission

Labs will be turned into student vfiler in the format of a zip file. The naming convention for the file will be ***Lastname.Firstname.Lab#.zip***. The contents of your zip file should only contain source files and shaders you have written. We would like these submissions to be as small as possible in order to accommodate space for everyone. If you fail to submit properly *you will lose points* on your grade. Even if it was a visual check-off.

# Frequently Asked Questions