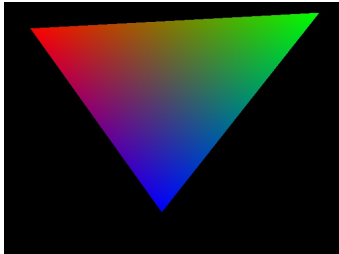


Lab 1 - Software Rasterization

You may work in pairs on this assignment. To receive credit, demonstrate your completed program during lab. Due one week from when assigned.

In lab 1, we'll be exploring how triangles are drawn to the screen using a process called rasterization. You will adapt the provided code to request input from the user for three points that will define a triangle along with three colors associated with each point. You must draw the triangle and output the result to an image file. When drawing the triangle, you must smoothly transition the color between each of the points. Here is an example of the required behavior:

<pre># ./triangle Enter 3 points (enter a point as x,y:r,g,b): 50,50:1,0,0 600,20:0,1,0 300,400:0,0,1 You entered: 50, 50 : 1, 0, 0 600, 20 : 0, 1, 0 300, 400 : 0, 0, 1 Output triangle.jpg</pre>	<p>triangle.jpg</p> 
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------

I suggest breaking the assignment into the following parts.

Part 1 - Bounding box

Retrieve three points (just x,y) from the user. Color can wait until Part 3. Once you have the points, calculate their bounding box. Loop over every pixel in the bounding box and set them to white (white is full strength red, green and blue so 1,1,1).

Part 2 - Barycentric coordinates

Compute barycentric coordinates for each pixel in the bounding box. Instead of turning every pixel in the bounding box on, we'll use barycentric coordinates to decide whether or not the current pixel is in the triangle. If it is, set the pixel to white, otherwise move on to the next pixel. Wikipedia is a great resource for learning more about barycentric coordinates.

Part 3 - Color

Modify your code that retrieves input to include color values that range from 0 - 1 for a red, green and blue component for each point. Modify your loop to use the barycentric coordinates as weights and compute a color for the current pixel.

Things To Notice

Everything about each pixel (whether it's on or off, it's color, location, etc.) is calculated independently of every other pixel. All of this information could be calculated in any order and in parallel. The ability to highly parallelize graphics computations is what makes graphics hardware so powerful.

Recommended Reading

[Wikipedia: Barycentric Coordinates On Triangles](#)

[Wikipedia: RGB Color Model - Numeric Representations](#)