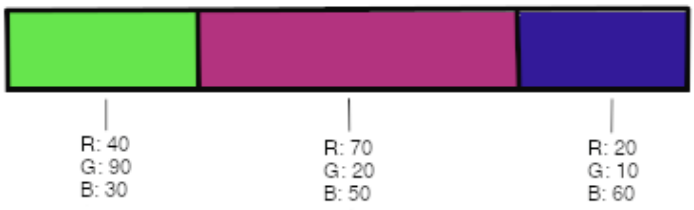
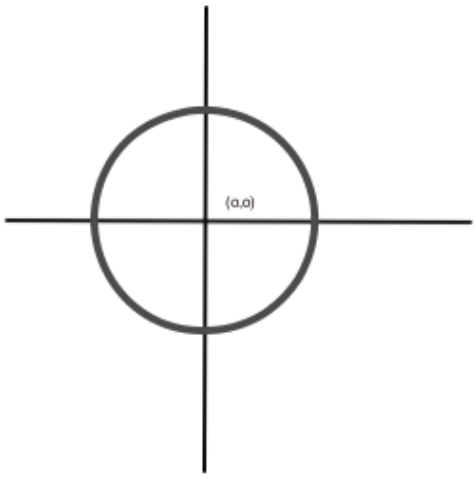
Color to Polyhedron conversion:

Each row contains many colors, each of which is made up of a combination of Red, Green, and Blue values, as well as an Alpha value for transparency (which I am ignoring). These colors take up varying spans of each row based on the interactions and trajectories taking place. Below is a segment of a typical row.




*All colors are made up of various combinations of Red, Green and Blue.*

To make a polygon, all we need are coordinates to act as vertices and draw lines connecting those vertices. These RGB values provide just that, allowing us to create 3-sided polygons.




*This image shows a circle with a radius of 1 and centered at the origin (0,0).*

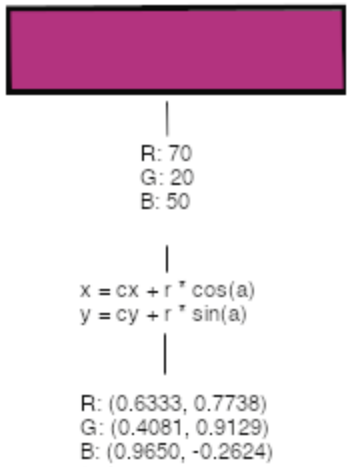
Given a known angle (in degrees), we can use the following parametric equation[1] of a circle to find what x,y coordinate on the circle that angle would reach.

$$x = cx + r*\cos(a)$$
$$y = cy + r*\sin(a)$$

Using the RGB values as the angle (‘a’) yields 3 coordinates.[2] So, taking the middle color from the earlier image, the conversion would proceed as follows:

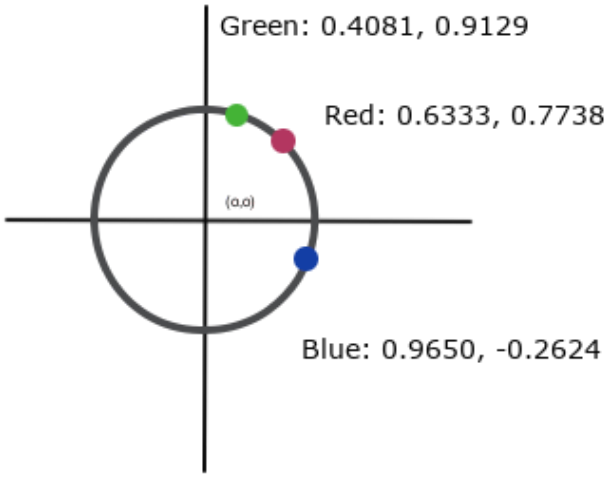
[1] Wikipedia contributors. (2023, July 2). Parametric equation. In *Wikipedia, The Free Encyclopedia*. Retrieved 20:38, August 21, 2023, from https://en.wikipedia.org/w/index.php?title=Parametric_equation&oldid=1163066620

[2] *To make this easier, I first mapped the (0-255) RGBA range onto the (0-360) degree range.*




*Shows conversion from previously used color to coordinates*

These new coordinates are now used as the vertices of a polygon.




*Shows the newly created coordinates on an x,y plane.*

The net conversion being:

*Shows final conversion from color to polygon*

Additionally, since different colors can take up different span lengths on a given row, I multiply the resulting coordinates by a factor based on its span length within the row. This results in larger or smaller polygons.[3] The results can be seen in the zoomed-in early portions of a vitaglyph run in Figure 5-9.

*[3] In an effort to keep the texture completely facing the viewer, I also converted all coords to positive x-values. The Blender Python code for this implementation can be found in Appendix-06.*