Preface

Program that creates triangles between user created vertices using the Delaunay Triangulation Algorithm. Vertices have a set RGB color and randomized height. Triangle fill determined by average color of its vertices. Height used to create a visually stimulating experience from finished artwork.

CONTROLS

Colors: 1 - Black

2 - White

3 - Red

4 - Green

5 - Blue

Toggles: T - Triangles

P - Vertices

O - Circle Boundaries

C - Click Color-Selector

M - 3D

Up/Down - Increase/Decrease Background Darkness

Left/Right - Change 3D **direction of rotation**

Space to **reset** project

Enter to **save** image

Escape to **exit**

This project was inspired by several different recent projects I had encountered, most notably, a video displaying an individual recreating the Mona Lisa using only ellipses, and then a separate individual that photoshopped images into poly-meshes with each shape consisting of the original region's color average. With these two projects in mind, I set out with the intention of creating something that displayed a clear and simple message – drawing is simple geometry. While so much more can be said for it, both good artists and poor have the same lines and curves at their disposal. The art is in the combination that turns the simple into the complex. My program functions by allowing the user to individualize their simple, triangle based artwork through simple color and position choices for the vertices, and then turns the simple 2-dimensional work into an impressive 3-dimensional arrangement.

While creating the project I learned a great deal about processing and various algorithms that I was unaware of prior. Processing proves to be excellent for creating programs that are simple yet beautiful, but it has certain failings when handling more robust processes such as the Delaunay Triangulation Algorithm used in my program. Inability to treat shapes as objects with unique instances and a lack of foresight in type compatibility, such as 2D primitive shapes in a 3D environment, greatly increased the code necessary to produce my tool. Another non-processing specific issue that arose was the lack of a similar program's existence. My attempts to find information I needed to create my program, such as algorithms and geometric relationships was fairly limited, and the few examples I found were often needlessly complex. I will take a moment to credit the two most important sources here - http://introcs.cs.princeton.edu/java/36inheritance/Delaunay.java.html, an open-source code for Delaunay Triangulation in Java from Princeton and Paul Bourke's section on "Equation of a Circle from 3 Points (2 dimensions)", found on http://paulbourke.net/geometry/circlesphere/.

My users displayed the intended qualities of the program without issue, and further went on to perform in ways that highlighted the idea of how something simple can become beautiful. One user limited himself to only base color changes and mouse-clicks, eliminating the use of many additional features, yet still creating something intriguing. The other user took time to test all features to their fullest and ended up showing some interesting results of color pairings that were otherwise unexpected.