Assignment 2

This project is designed to gain experience with modern OpenGL, creating a triangle editor which allows the user to create, translate, delete, color, scale, rotate, and animate objects on screen. This project utilizes the Eigen library to help facilitate matrix creation and calculations.

Specifics

For this project, it was decided that the triangles created by the user will be stored in a Eigen::MatrixXf, which will start as a 6x1 matrix, and to a 6x3 as the user creates the triangle. It should be note that the matrix must be stored in column order to function properly. The first three rows of a column will refer to the x, y, and z vertices of a single triangle vertex, and the next three rows will correlate to the r, g, and b values of the color of that vertex. Each matrix (triangle) will be coupled with its own VBO to ease the shading process. This coupling will be stored in a std::pair, which will be an element in a std::vector. There are no storage conflicts with a std::vector and a dynamic Eigen::MatrixXf.

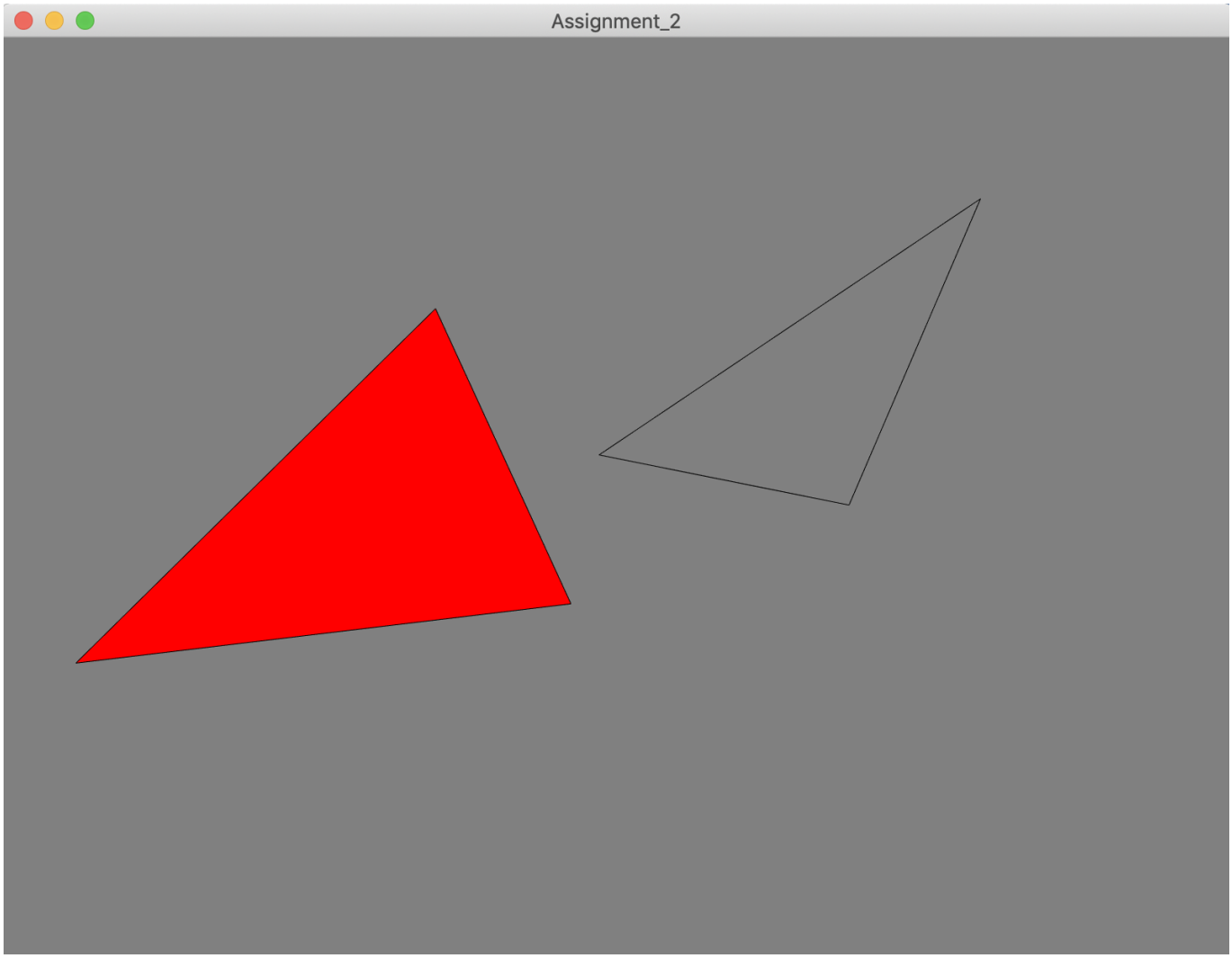
main.cpp contains the window initialization with `glfw`, and also contains the implementation for the key call back, which is mostly just a process of registering/unregistering different mouse button and cursor position call back functions whose implementation can be found in the header files. The decision on whether to register or unregister is based on global Boolean values for each type of interaction a user can have with the program. main.cpp also contains the main loop which renders the scene by looping over the aforementioned vector of VBO's and their respective matrix and deciding how to render them with glDrawArray.

Results

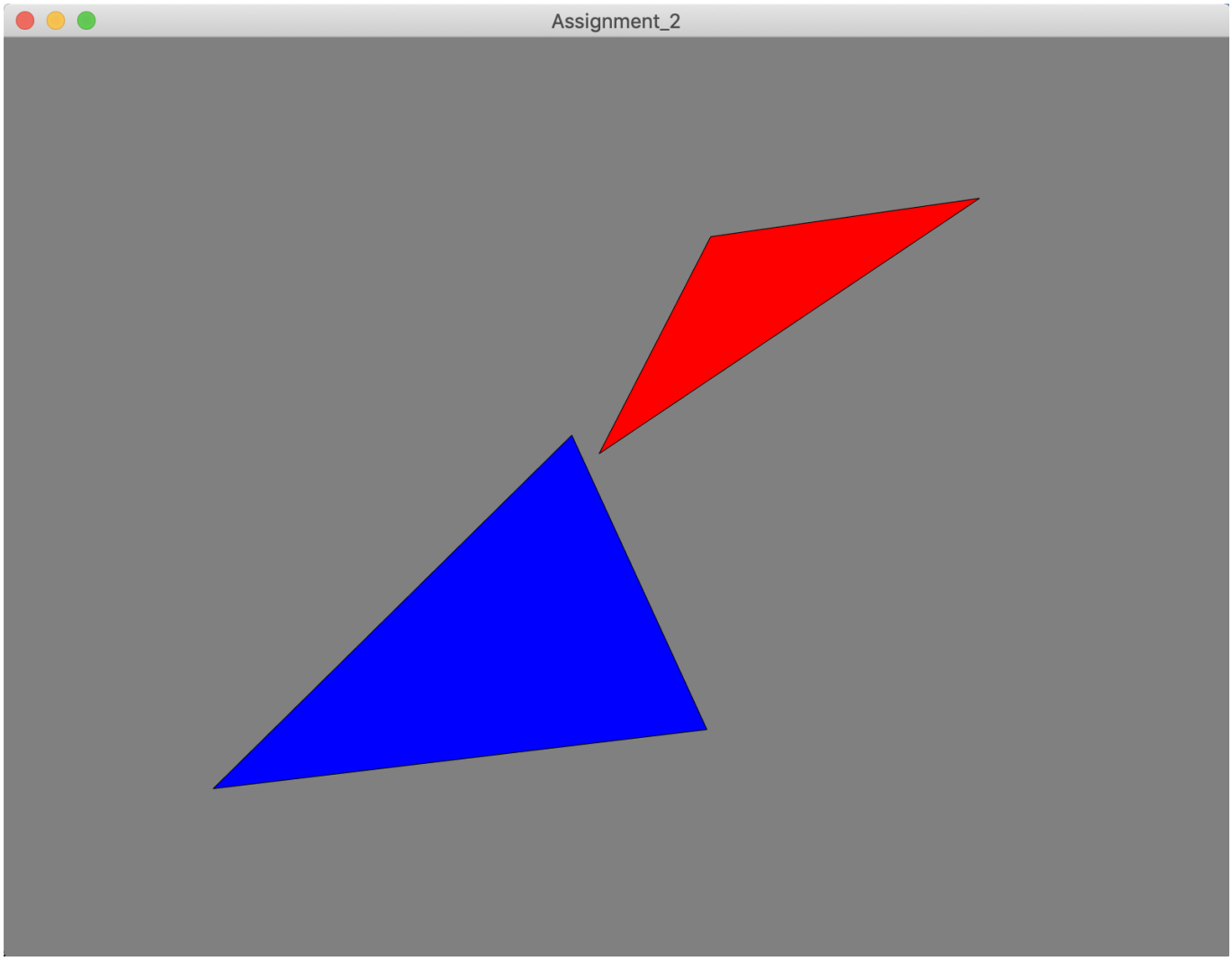
1.1 Triangle Soup

Can insert MAX_TRIANGLE amount of triangles, which is defaulted to 25.

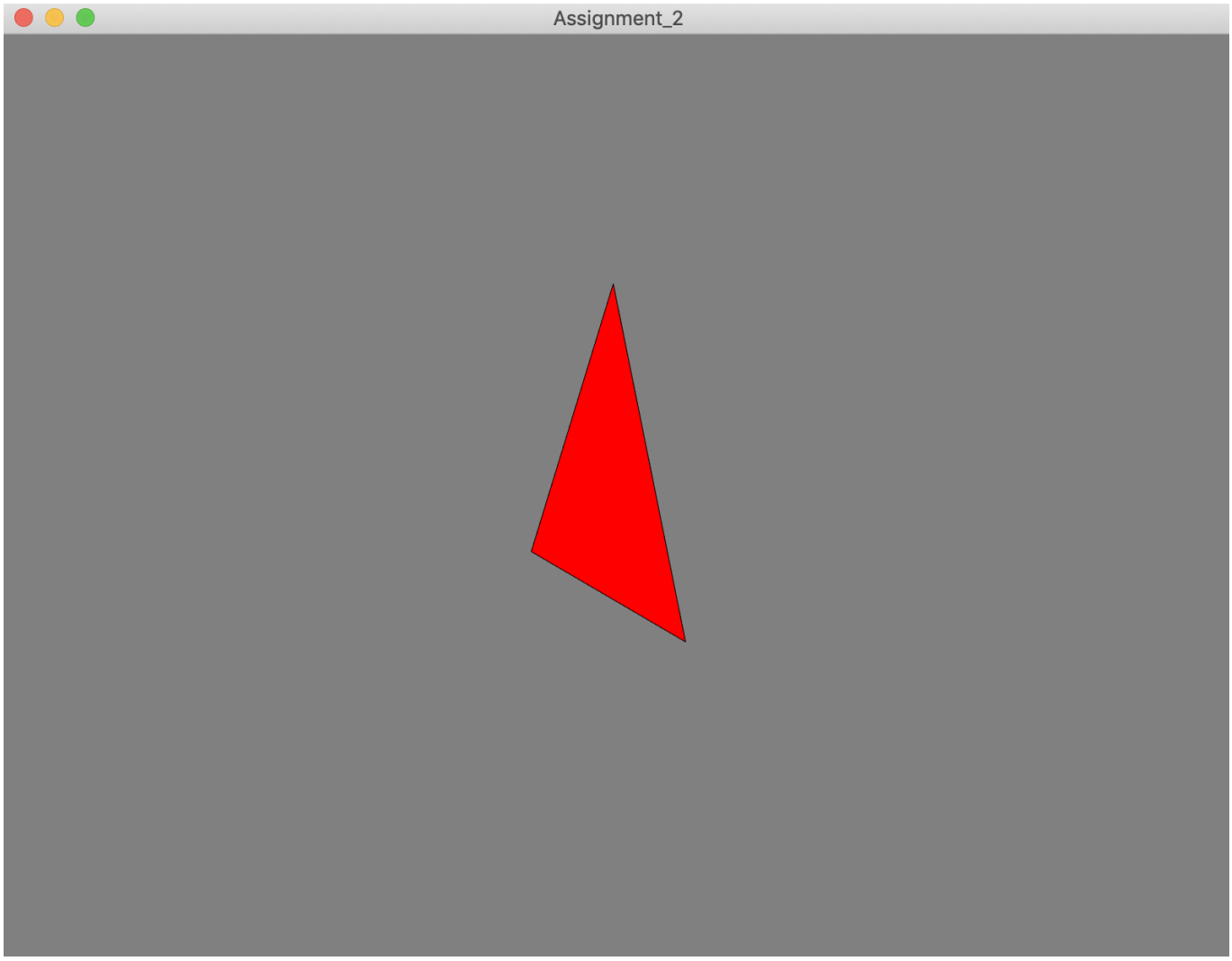
1.1 Triangle Soup Insertion Mode



1.1 Triangle Soup Translation Mode



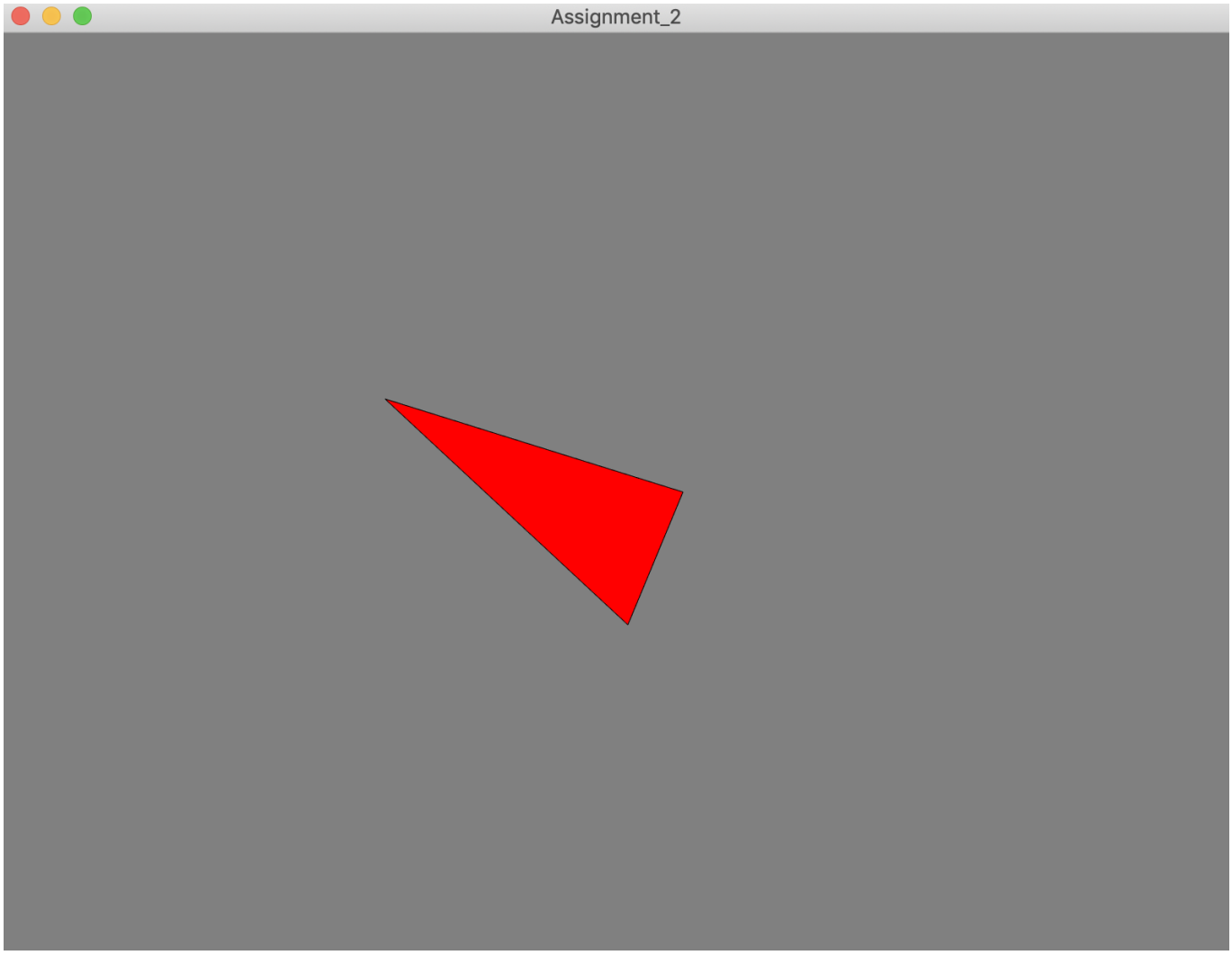
1.1 Triangle Soup Deletion Mode



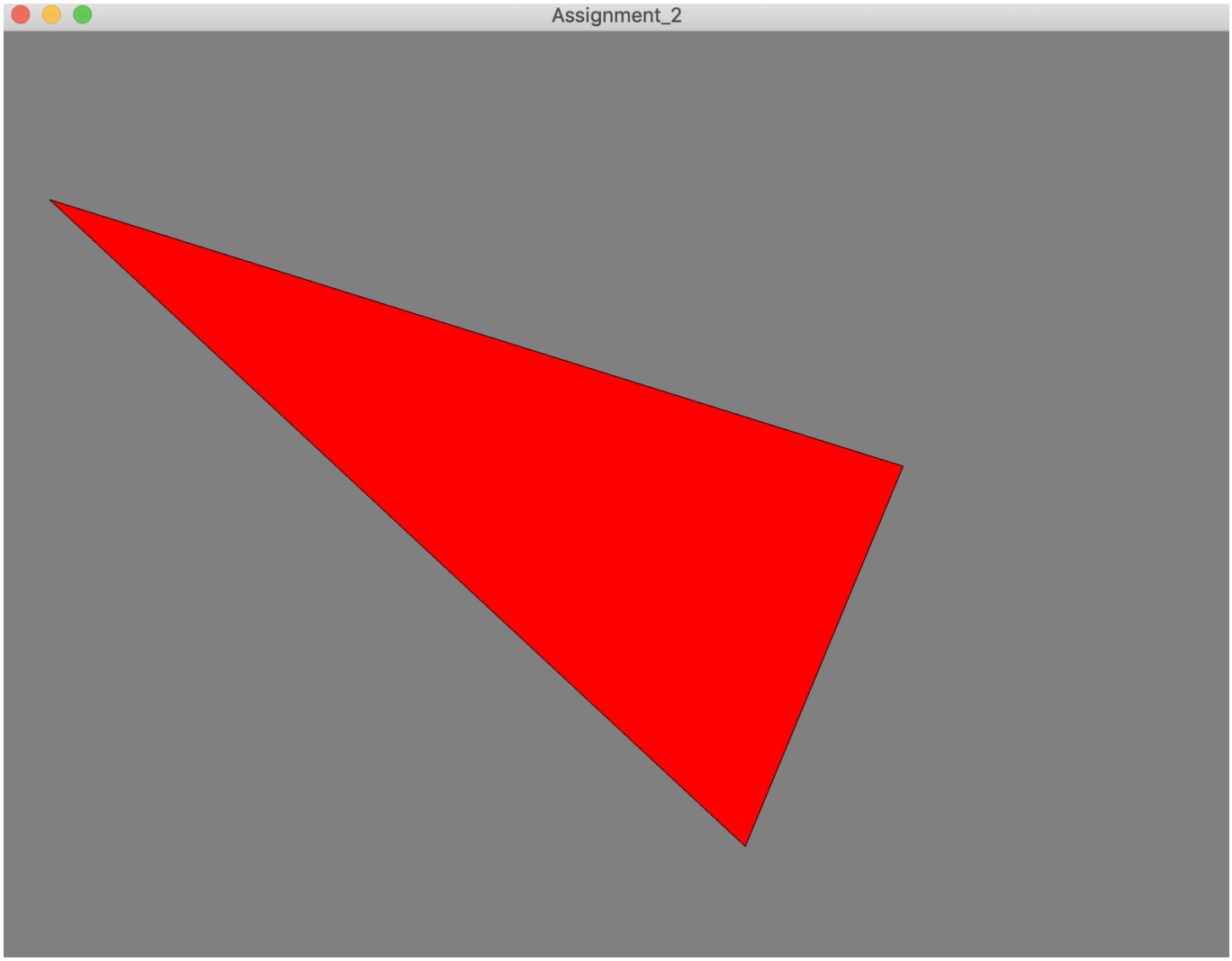
1.2

For this mode, I decided to get into the rotate/scale mode, the user must hit "R" before selecting the triangle, and then can use h, j, k, and l for rotation and scaling. Modifies the vertices before updating the VBO.

1.2 Rotation Mode

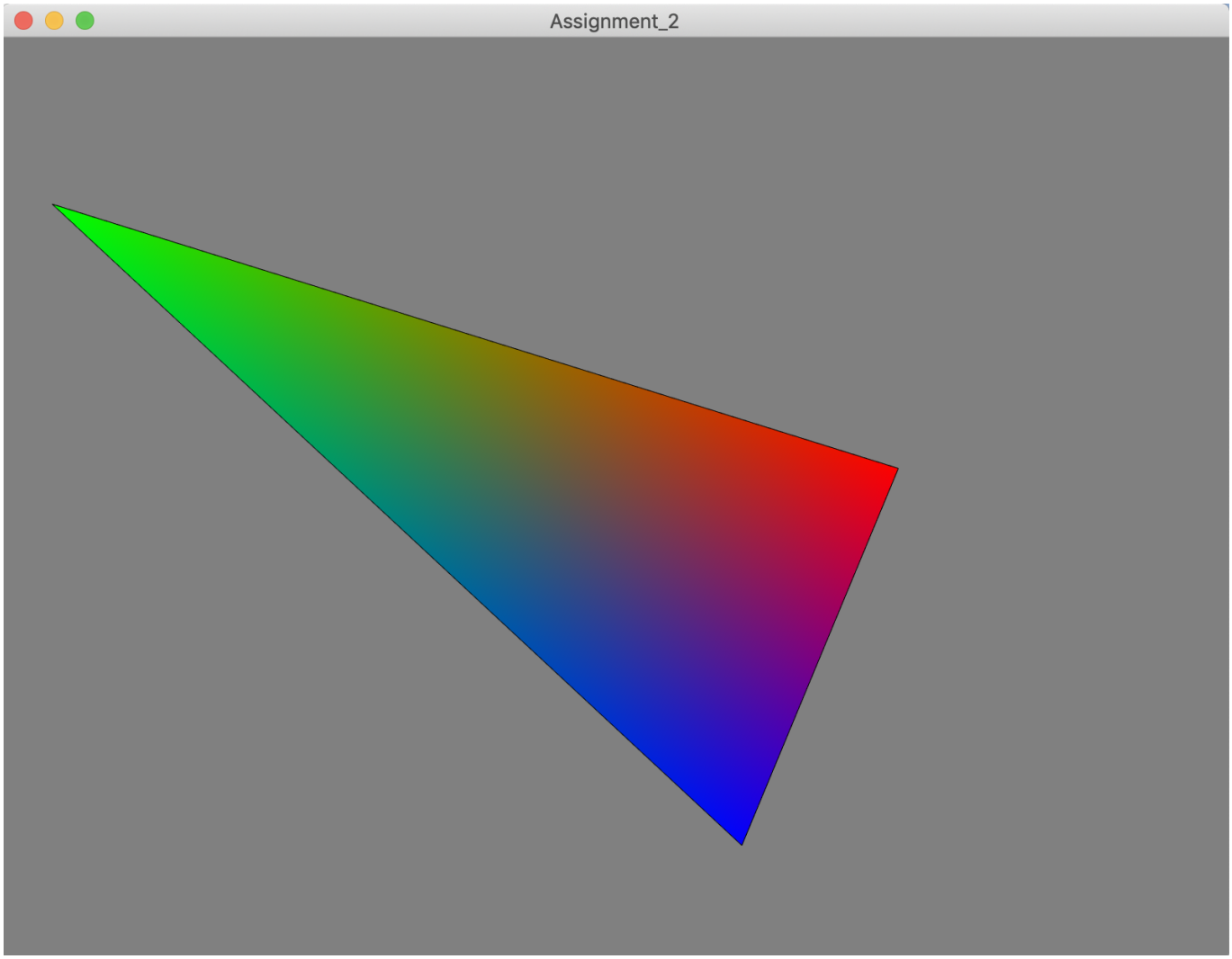


1.2 Scale Mode



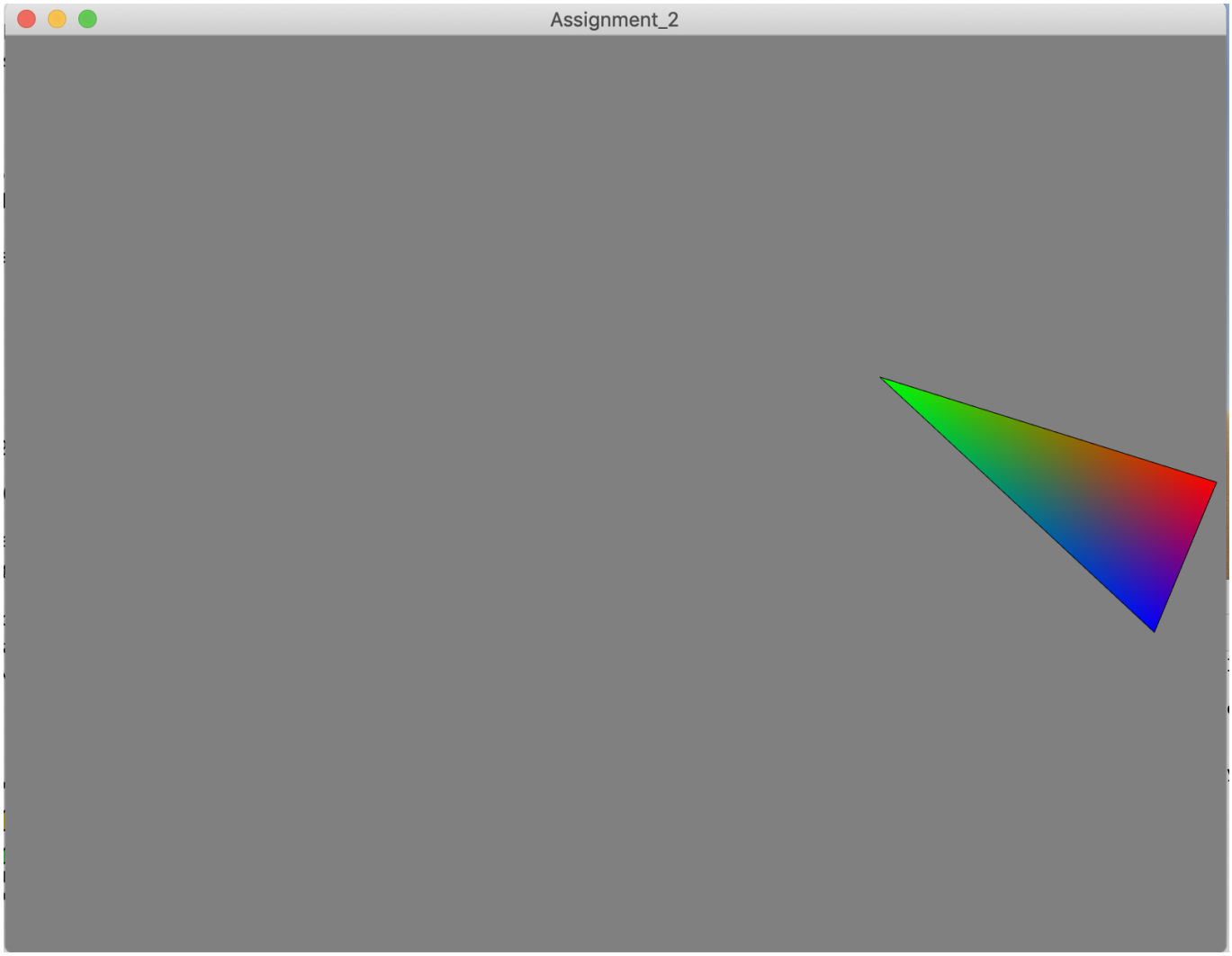
1.3 Color Mode

This mode updates the rgb values in the matrix for whichever vertex is closest to a click (if within a triangle). To change the color of a different triangle, it is recommended that you exit color mode by pressing "C" and then enter color mode again. Otherwise all triangles getting colored will change colors.



1.4 View Control

Global variables used to keep track of the modification on the original scalars (press "A" then say longitudinal increase by 0.2). This variable placed in the view matrix which be used as a uniform matrix in the vertex shader.



1.5 Animation

Not sure how to show this as a picture, but implementation relies on performing a rotation operation on each vertex of the triangle and updating the VBO on each frame.