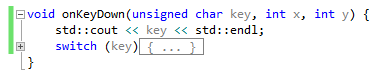
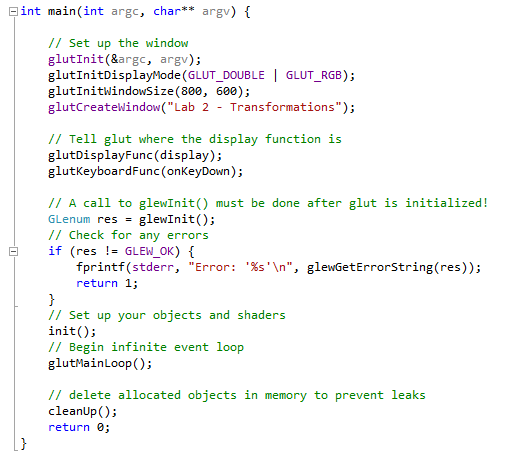
Lab 2 - Transformations  
Computer Graphics CS4052  
Edmond O’ Flynn  
12304742

“Keyboard Control”

I modified main.cpp to include a callback function that gets invoked on keypress. The binding occurs in main via glut. For transformations, I leveraged Anton’s maths classes for matrix arithmetic.



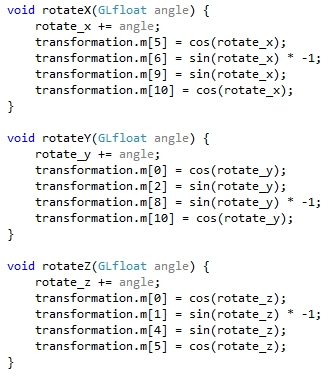
Callback function for invoking functions on keypress



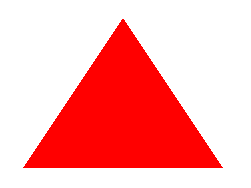
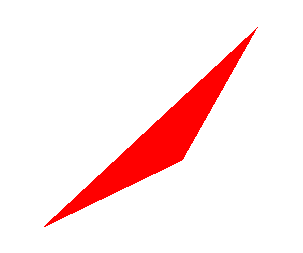
glutKeyboardFunc() takes a delegate onKeyDown for its callback

“Keypress to show: Rotation around x-, y-, and z- axes”

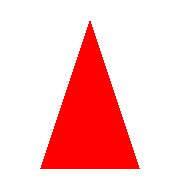
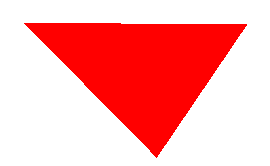
For this lab, I made the transform into an object in order to more easily have multiple objects and transformations in the scene for the same buffer. My functions for rotation take an incrementation parameter to allow for rotation in both directions.



Implementations for x, y, and z rotation for buffer objects

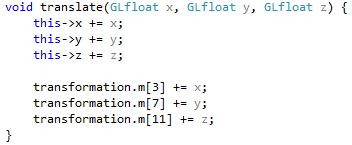
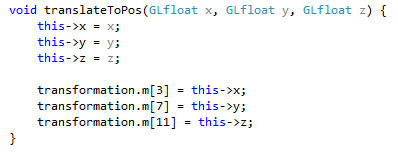
Regular triangle Combined x, y & z transformations



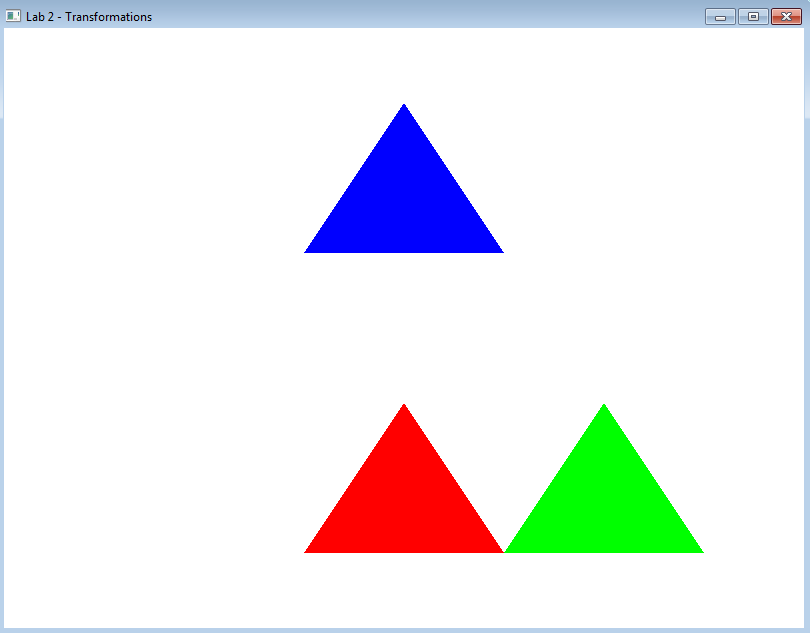
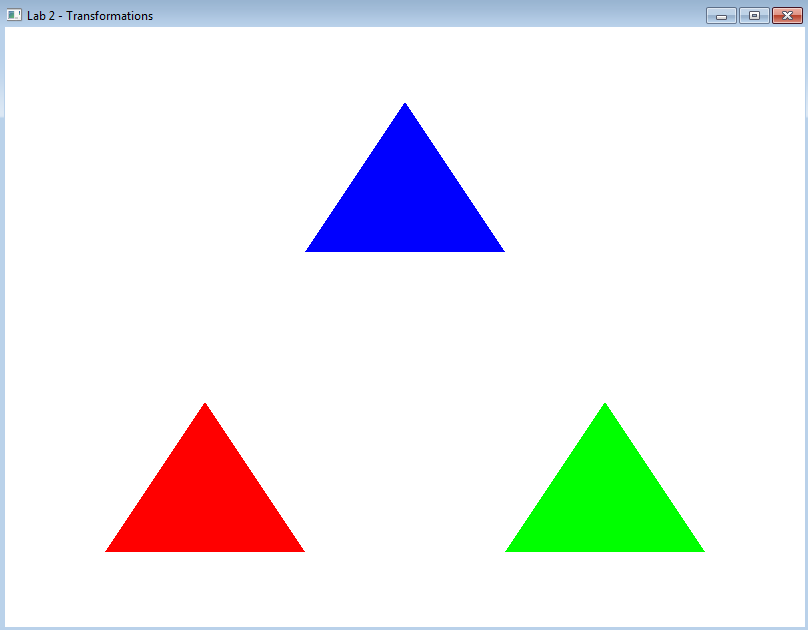
Individual x, y, & z rotations

“Keypress to show: Translation in the x-, y- and z- directions”

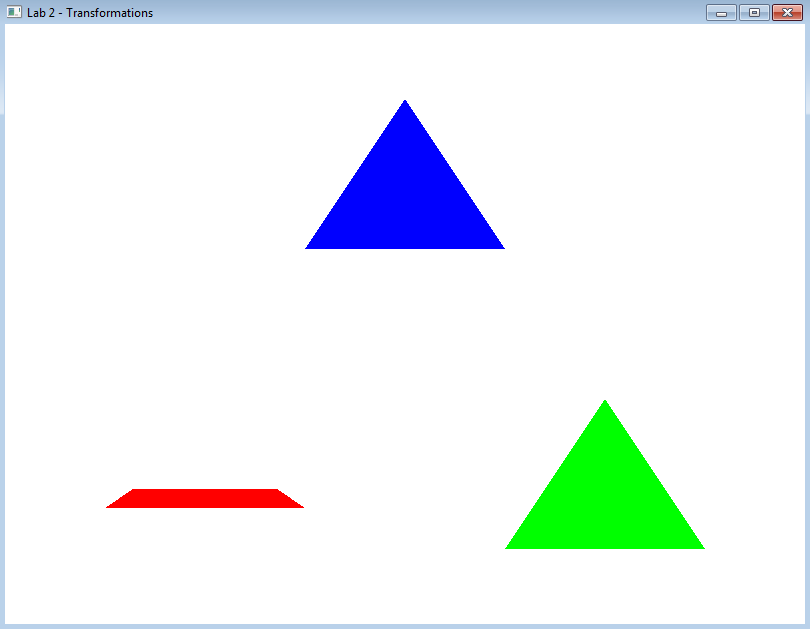
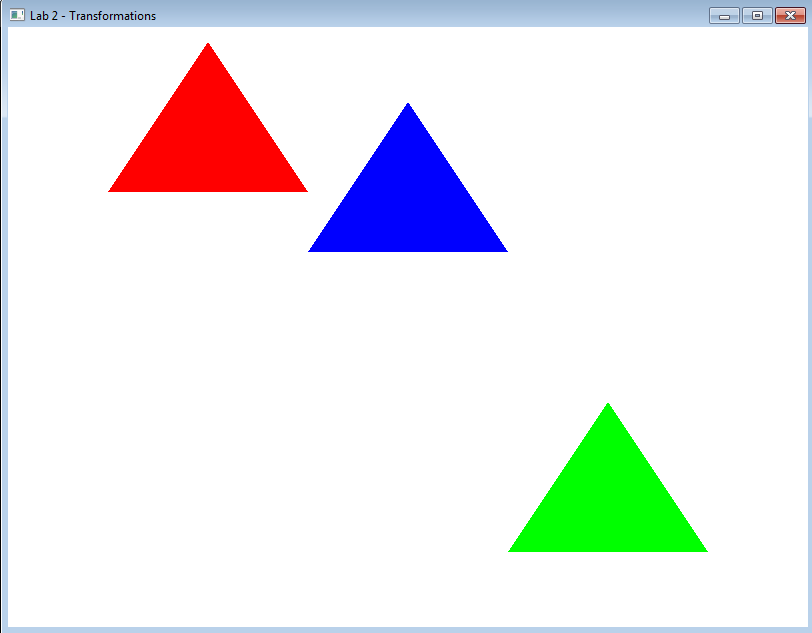
On keypress, the red triangle has an operation performed onto its transformation matrix which was then fed into the vertex shader and multiplied with its position vector.



Functions for translating to a specific position and to transform by incrementing a parameter

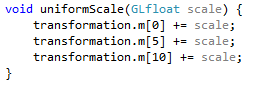


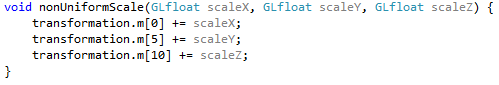
Untransformed triangles Red triangle transformed in x direction

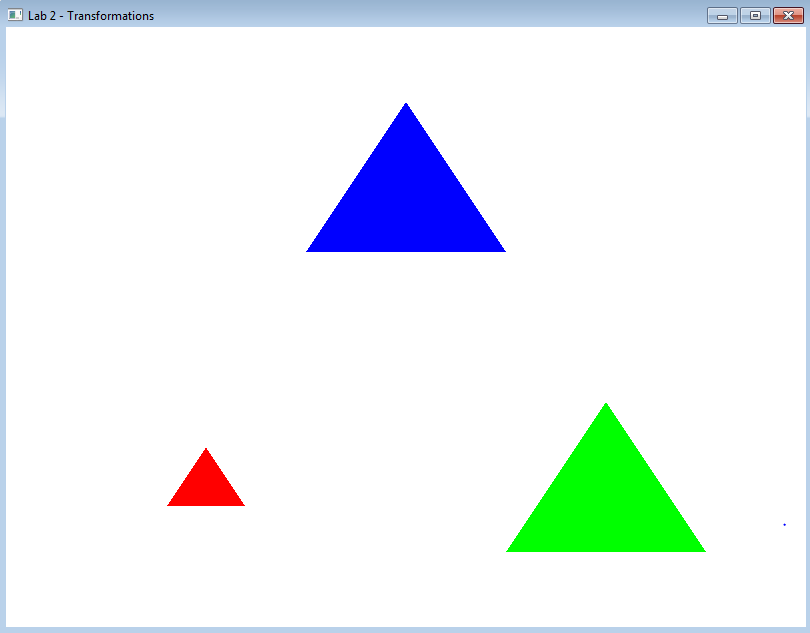
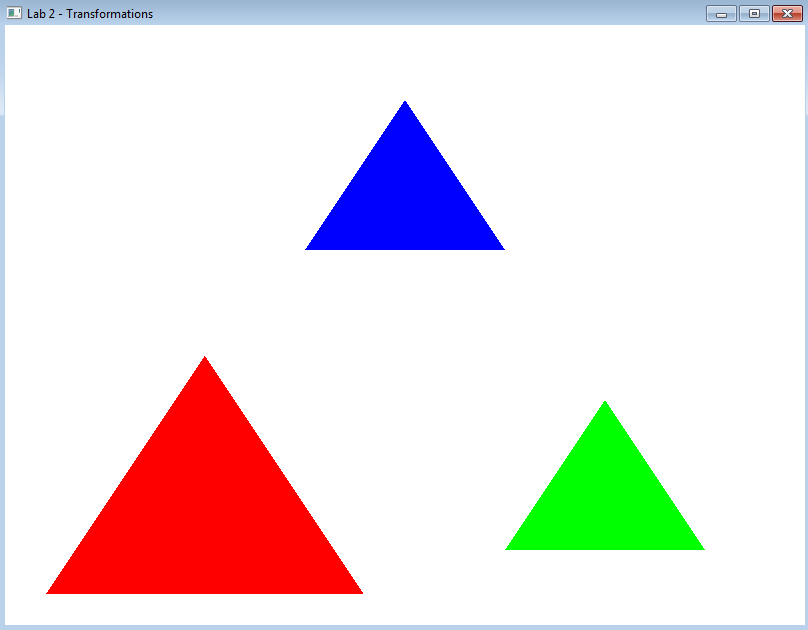


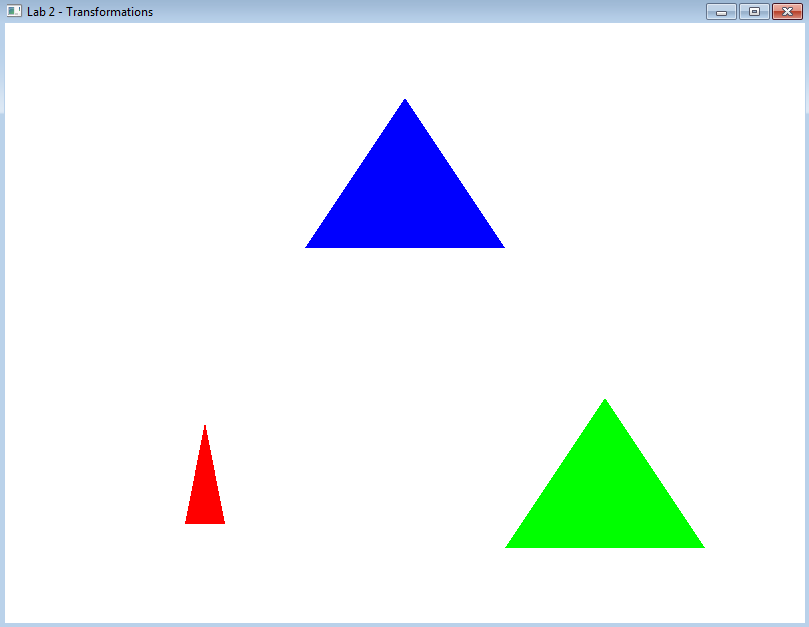
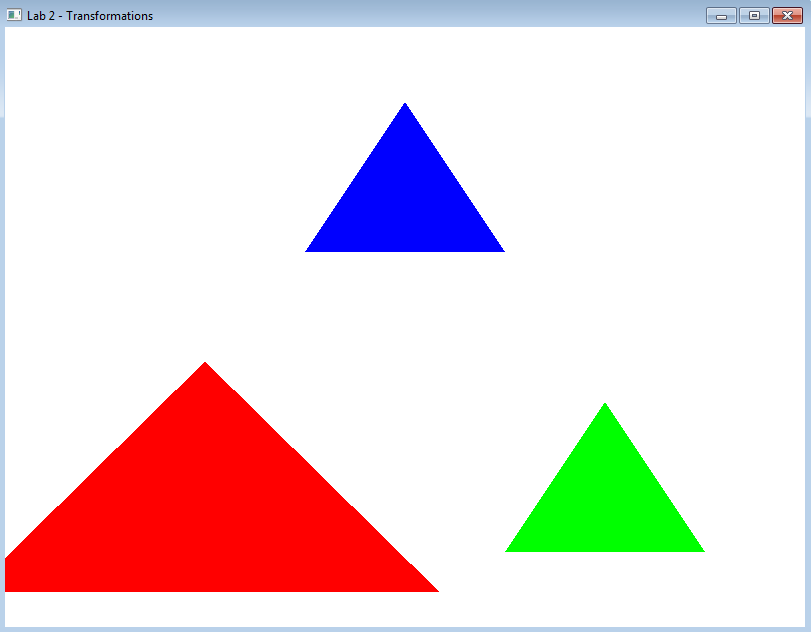
Transformed in the y direction Transformed in the z direction (with rotation)

“Keypress to show: Uniforms and non-uniform scaling”

  
Uniform scaling with parameter

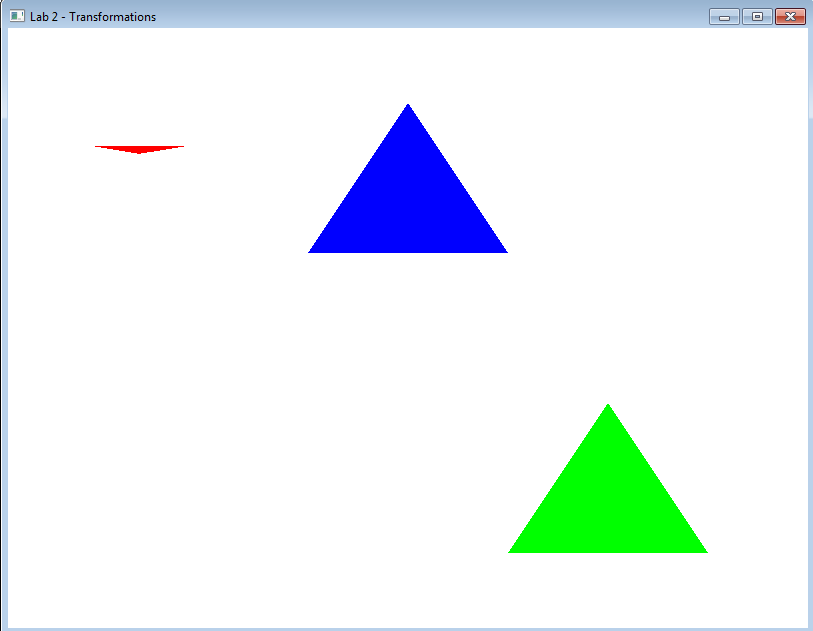
  
Non-uniform scaling with multiple scale parameters

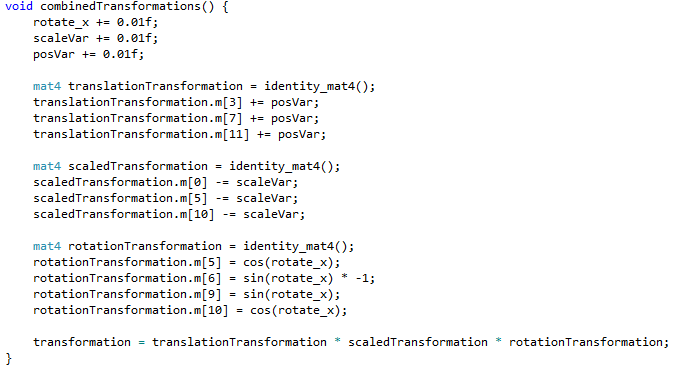
  
Uniform scaling up and down

  
Non-uniform scaling up and down

“Keypress to show: Combined transformations”

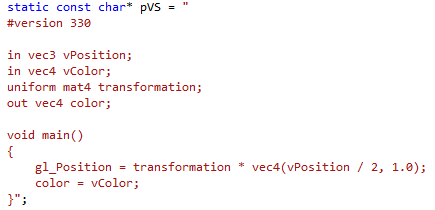
As I was feeding in a transformation matrix into my vertex, all of my matrix calculations were done in a function that was directly fed into the vertex shader.

  
Red triangle has a translation, scale and rotation applied in one transformation

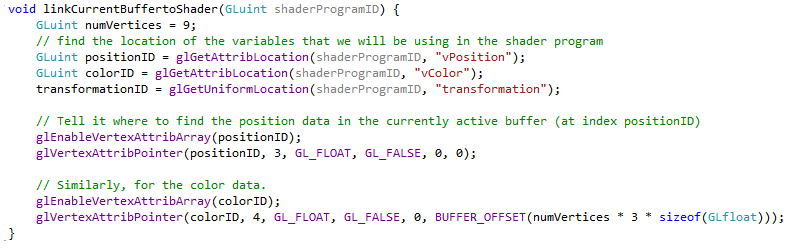
  
Operation performed with multiple transforms in one

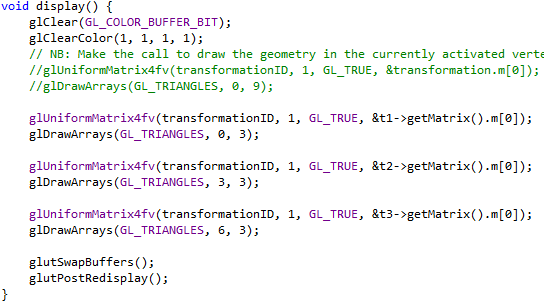
“Multiple triangles in the scene using the same buffer but creating a new transformation matrix for each one”

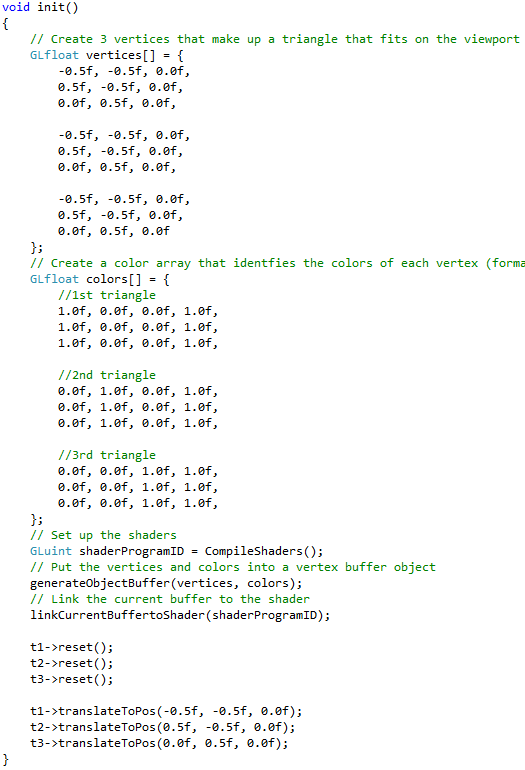
As I had to also modify the vertex shader to feed in another variable in order to pass a transformation in, I had to create a new variable, register it with an ID, find it, and then pass data into it.

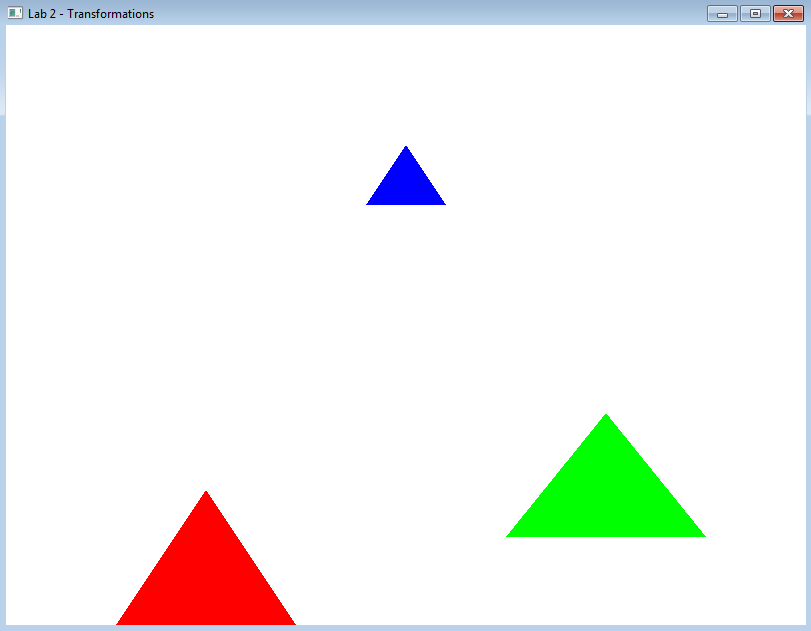
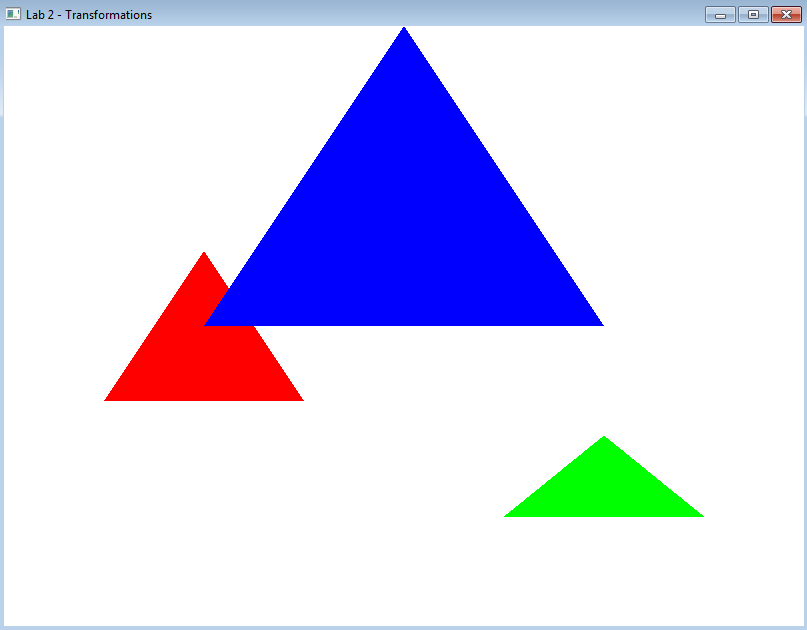


New vertex shader with additional uniform mat4 transformation

  
Assigning transformationID a value via glGetUniformLocation()

  
Passing in a transformation matrix via class object to the vertex uniform variable

  
init function feeding in 3 triangles into one buffer with setting up of positional data



Each triangle concurrently having its own transformation applied in its matrix in the same buffer