CS7GV6 – Computer Graphics Lab 1.2 Transformation in OpenGL

Shao Yuzhou (19322035)

1. Create a transformation matrix to move your triangle object around the scene.

Firstly, I configured the environment variable to add **freeglut** and **glew** libraries.

Secondly, in order to create a transformation matrix, **glm** (GL mathematic) library should be downloaded separately and added in the project properties. And add glm header file to load the library.

```
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp>
```

Next, based on the main.cpp code from the previous assignment Lab1.1, write the code for transform the position. The code is as below.

```
static const char* pVS = "                                                    \n\
#version 330                                                                  \n\
                                                                              \n\
in vec3 vPosition;                                                            \n\
in vec4 vColor;                                                               \n\
out vec4 color;                                                               \n\
uniform mat4 transform;                                                       \n\
                                                                              \n\
void main()                                                                   \n\
{                                                                             \n\
    gl_Position = transform * vec4(vPosition.x, vPosition.y, vPosition.z, 1.0); \n\
    color = vColor;                         \n\
}";
```

In display function write glUniformMatrix4fv function to make the call to draw the geometry in the currently activated vertex buffer. The code is as below.

```
GLint transformLoc = glGetUniformLocation(shaderProgramID, "transform");
// NB: Make the call to draw the geometry in the currently activated vertex buffer.
glUniformMatrix4fv(transformLoc, 1, GL_FALSE, glm::value_ptr(vec));
glDrawArrays(GL_TRIANGLES, 0, 3);
glutSwapBuffers();
```

In init function sets the initial status on shader of the triangle:

```
    // Set up the shaders
    GLuint shaderProgramID = CompileShaders();
    // Put the vertices and colors into a vertex buffer object
    generateObjectBuffer(vertices, colors);
    // Link the current buffer to the shader
    linkCurrentBuffertoShader(shaderProgramID);}
```

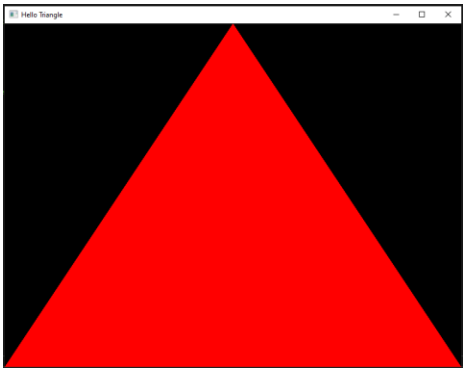
Similarly, for rotate and scaling, the code is added on display function:

```
trans = glm::rotate(trans, glm::radians(90.0f), glm::vec3(0.0, 0.0, 1.0));//
trans = glm::scale(trans, glm::vec3(0.5, 0.5, 0.5));//
```

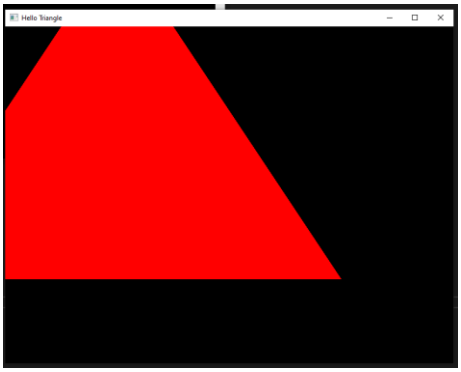
Then, write process_keyboard function to set key letters to control its corresponding movements of the triangle. Inside the function, I have used if and else if loop to set different key buttons to its corresponding changing to the triangle, to show Rotation around the x- y- and z-axis, Translation in the x- y- and z- direction ,Uniform and non-uniform Scaling Combined Transformations and Multiple triangle in the scene.

Some screenshots of outputs:

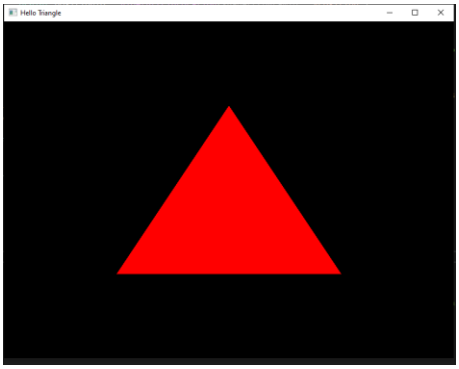
Original Triangle



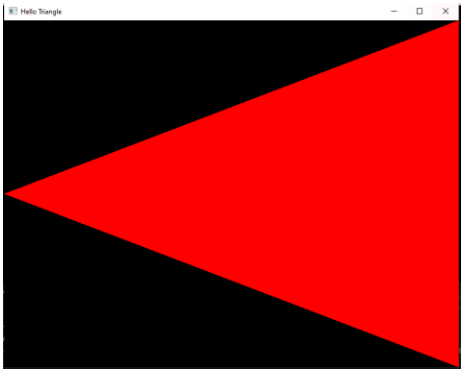
A transformation output example after press keys w s a or d



Uniform and non-uniform Scaling



Rotation



Acknowledge:

I have requested technical help from my classmate Shail, Bharat and Li Chaoyi for correct the errors of my code.

Referenced the code from the website *LearnOpenGL-Transformations*. And Youtube *tutorial* OpenGL/C++ 3D Tutorial 16 - Model Matrix (Movement, Rotation and Scaling)