CS6610 PROJECT 2 – Transformations
Varsha Alangar

Implemented features:

All the three steps of the requirements listed in the Project 2 including the F6 additional requirement for CS 6610 students has been implemented.

1. The vertex data from the teapot.obj file is used. This data was read and a VAO was generated and bound using GLEW. The *glewInit* call was used after the OpenGL window creation.
2. A vertex buffer was generated and the contents of the buffer was drawn.
3. Next, the GLSL Shaders were written and compiled into a GLSL program to be used in the draw call.
4. The vertex shader multiplies the vertex position by 0.05 and I get the entire teapot.
5. Now, I create the camera matrix and multiply it with the vertices to obtain a transformed teapot.
6. *Mvp_matrix = Perspective_matrix * view_matrix * model_matrix* This matrix was sent as a uniform to the vertex shader.
7. The left and right mouse button features were implemented. (See the section “Using the above implementations” on mouse button usage)
8. GLSL Shader recompilation code is written for key press.

Additional functionalities:

I have tried to implement the orthogonal transformation on pressing key P. I have also made sure it is not case sensitive. However, my teapot vanishes on the key press when trying to use orthogonal projection and reappears for perspective.

Using the above implementations:

Left mouse button: Hold the left mouse button down and drag the cursor to change the camera angle (rotation)

Right mouse button: Hold the right mouse button down and drag the cursor to change the camera distance (zoom-in and zoom-out)

F6: Recompile GLSL Shader

P or p: Toggle between perspective and orthogonal projection.

OS Used: Windows 10

IDE: Visual Studio 2013

Libraries and dependencies:

All the libraries used in the project are included under the *lib* folder within the zip file. They include: opengl32.lib, glu32.lib and freeglut.lib

All the header files are included within the GL folder contained in an include folder within the zip file. They have been included as *#include <GL/gl.h>*, *#include <GL/freeglut.h>,*

#include<GL/vmath.h>, #include<GL/glfw3.h> in the code. All the header files from cyCodeBase recommended for use are included in the cyCodeBase folder in the same include folder that holds the GL folder.

All the DLLs required are placed in the Debug folder of the zip file.

The source code itself was created and compiled in Visual Studio and is available as *main.cpp* in Transformations folder along with the solution.

The executable is available in the Debug folder.

Requirements to compile the project:

Unzip the project zip file and open the solution in Visual Studio.

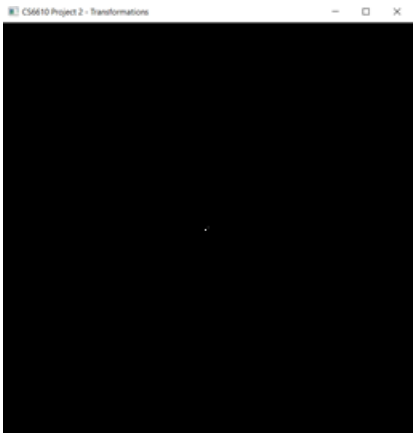
In the properties of the project, link to the libs, dlls and header files. Make sure to choose “All configurations” in the properties window before adding the dependencies.

As mentioned earlier, all the required libraries, dlls and header files are available in the lib, Debug and include folders respectively of the zip file.

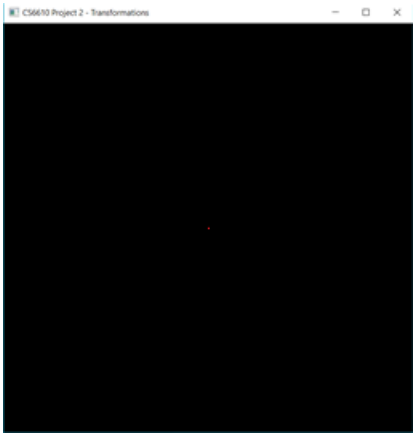
Please let me know if there is any issue in running the code.

Screenshot of code and output:

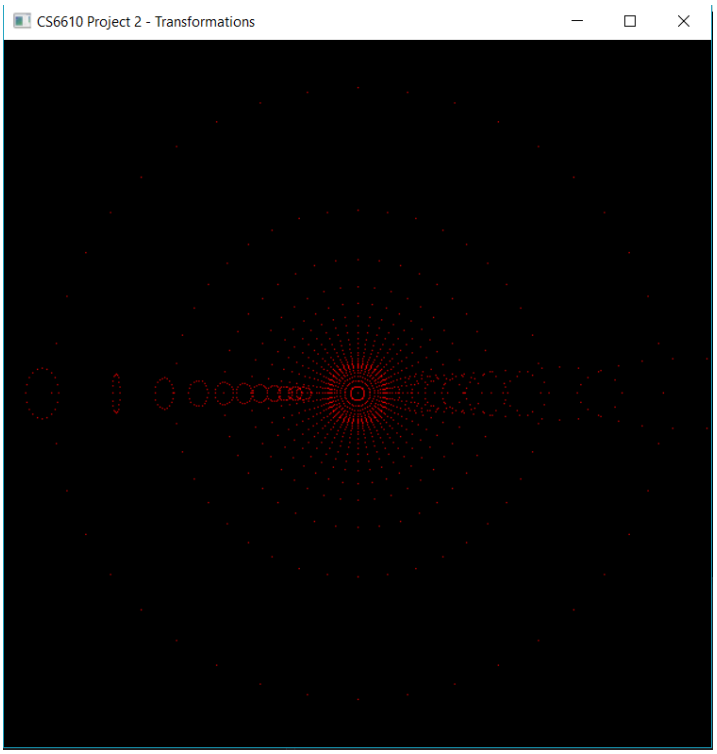
Output after step1:



Output after step2 (shader simply uses the vertex data)

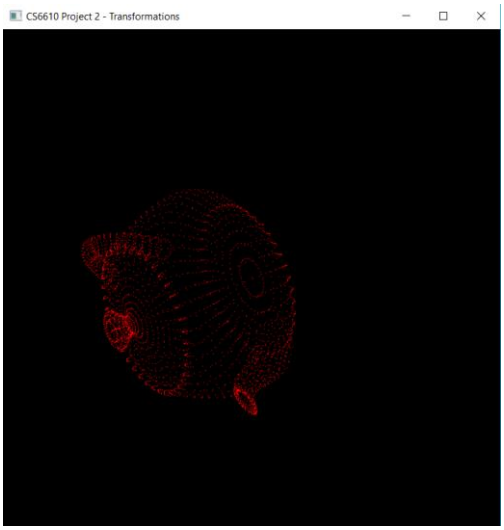


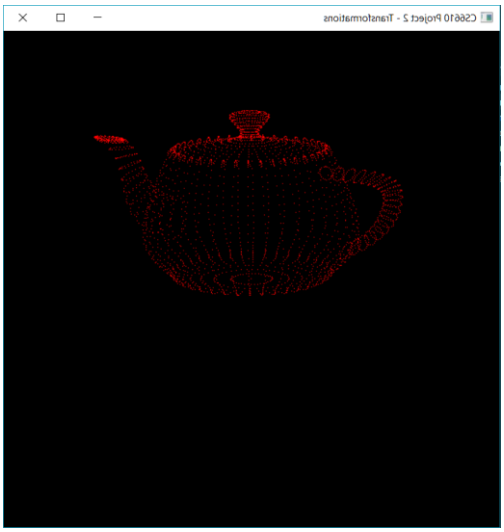
Shader uses mvp * vertex data:



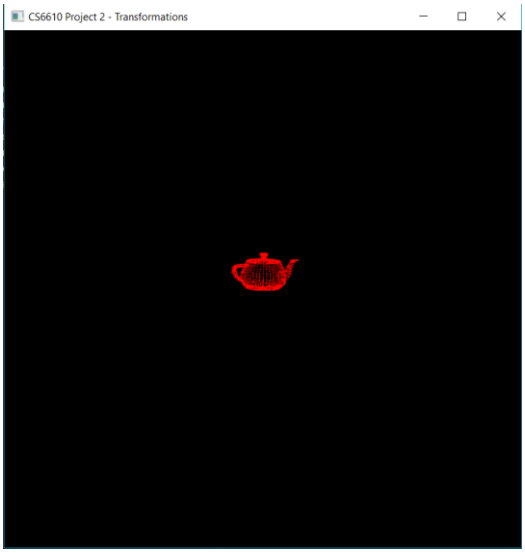
Shader multiplies each vertex by 0.05 and then multiplies it with mvp:

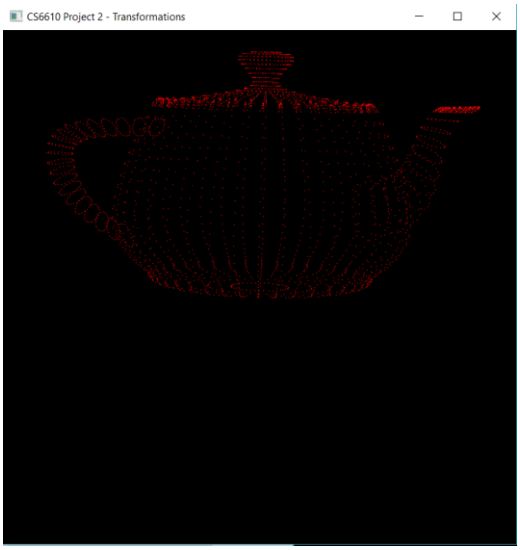
Transformations on teapot (Rotation)





Changing camera distance (Zoom out and Zoom in)





Additional requirement screenshot of code:

```
void special_keys(int key, int x, int y)
{
    switch (key)
    {
        case GLUT_KEY_F6:   glsl_program_obj.BuildFiles("Shaders/simple_vs_2.vert", "Shaders/simple_fs_2.frag", NULL, NULL, NULL, &std::cout);
                break;
    }
}
```

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv); //to initialize GLUT
    glutInitDisplayMode(GLUT_RGBA);
    glutInitWindowSize(window_height, window_width);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("CS6610 Project 2 - Transformations");
    glewInit();
    glutDisplayFunc(render);
    glutKeyboardFunc(keys);
    glutSpecialFunc(special_keys);
    glutMouseFunc(mouseButtonPress);
    glutMotionFunc(mouseMove);
    glutIdleFunc(idle);
    initialize();
    setShaders();
    glutMainLoop();
    return 0;
}
```