521140S Computer Graphics Programming Assignment I

General information

In this assignment, you will explore the OpenGL implementation of affine transformations, geometry, and inputs.

The assignment should be finished alone. Feel free to discuss about the assignment with other students but sharing code is not allowed.

What to return

Return the finished code **CG_assignment1.py** and a **PDF** report with the requested screenshots before the deadline.

Tasks

In this assignment, you are expected to render a simple scene, where you implement the code to transform a tetrahedron and the use of keyboard and mouse inputs.

1. Get the scene to work. Open the *CG_assignment1.py* file. Run the code. If there is any problem, please refer to Tutorial 0. Take a screenshot of this scene. (1 p)

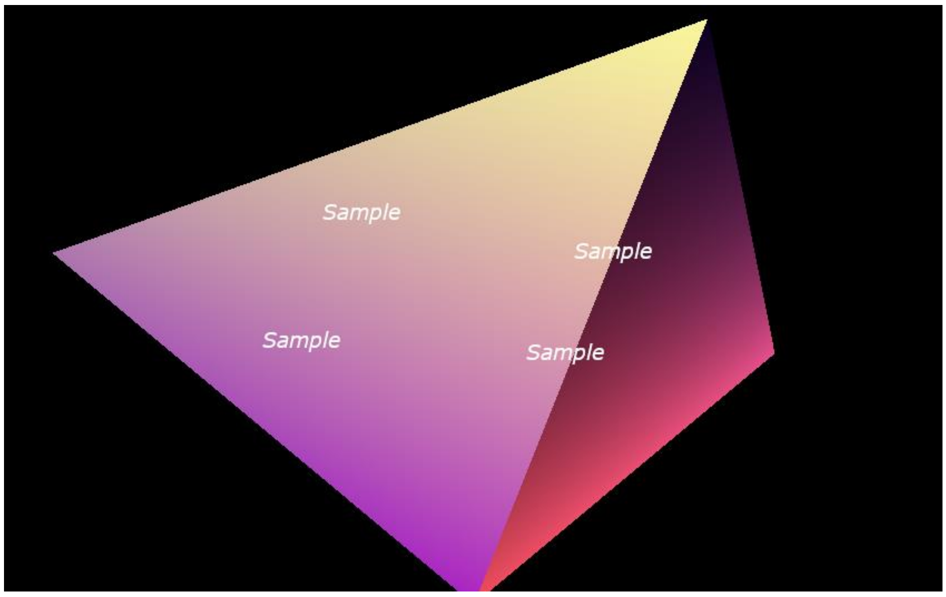

Figure 1

The code consists of five total files.

- The *.glsl* files refer to shader files (no need to modify)
- The *glut_window.py* is the base window application (no need to modify)
- The *mvp_controller.py* is responsible for managing the camera, view and projection matrices. (Tasks 2, 3 and 4 are here)

- The *CG_assignment1.py* is the main file where the data buffering and drawing occurs. (Task 1 is here)

Have a look around the code to better understand how the vertices defined in *CG_assignment1.py* end up being drawn on the screen.

2. Render the scene with rotation. Find the `calc_model` function in the `Win` class. Change the model matrix in such a way that it rotates over its x-axis continuously. For rotation use the `glm` library. To make the rotation continuous use the `time` library.

   Run the code to get the result below (see the gif from assignment1 slides). Take a screenshot in the report. (2 p)

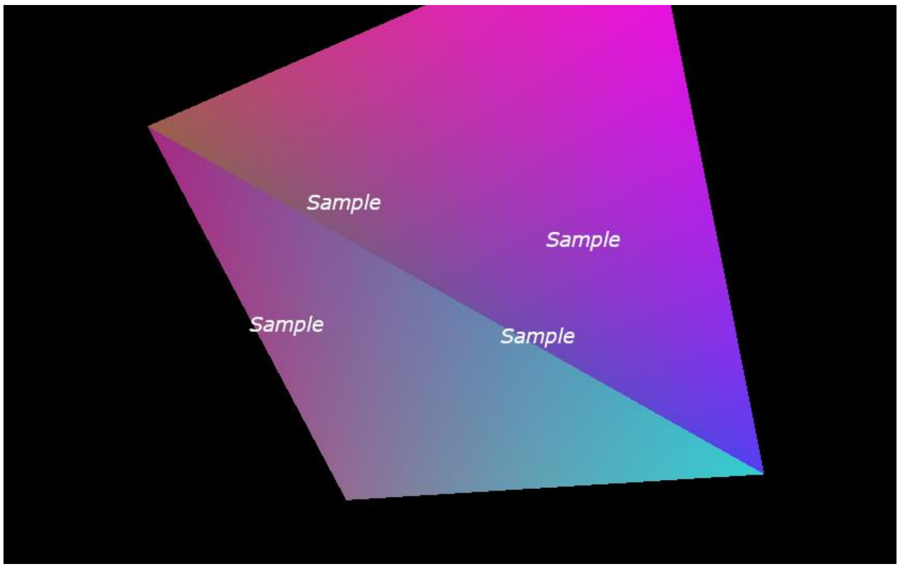

Figure 2

3. Camera view matrix. Go through the tutorial https://learnopengl.com/Getting-started/Camera and find out how the `direction`, `up` and `right` vectors are implemented using Euler angles. The tutorial uses C++ so remember to change the syntax to python. Implement the corresponding matrices in the *mvp_controller.py* function `calc_view_projection`.

   Once implemented properly you should be able to move the camera freely with your mouse by holding the right button down. Take a screenshot of this. (3 p)

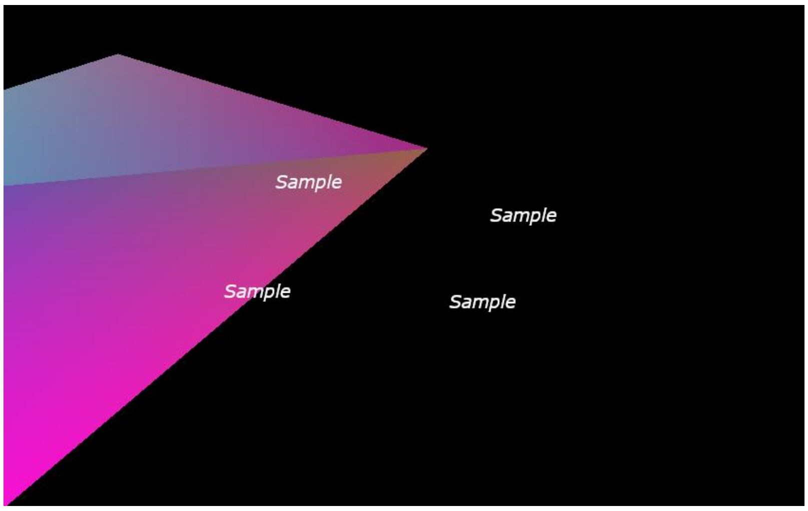

Figure 3

4. Keyboard movement. Add keyboard movement to the `on_keyboard` function in *mvp_controller.py*. Implement the following moves: forward, left, right, backwards, up and down. You can choose to use whichever keys you prefer, or to use the following: w, a, d, s, e and r, respectively. You should manipulate the `self.position` matrix with some of the matrices defined in `calc_view_projection` function.

   Note that the `key`'s type is `bytes` not `str`!

   Once implemented properly you should be able to move around freely around the scene. Take a screenshot of this. (2 p)

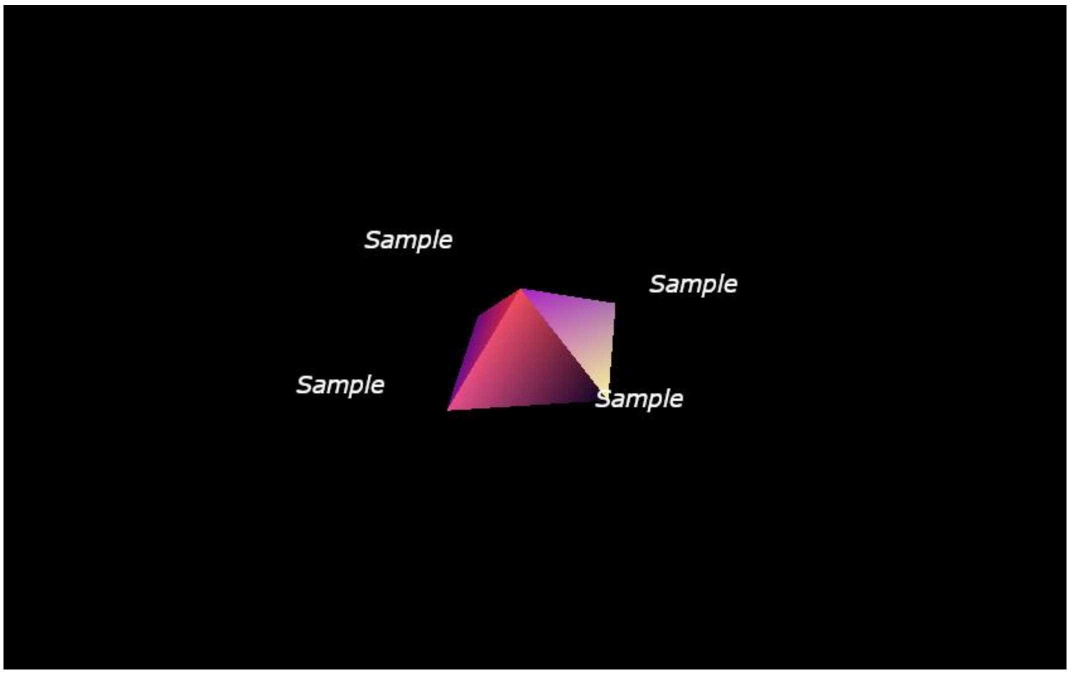

Figure 4

5. Additional tasks. These tasks are not mandatory. These tasks do not provide additional points. They can be done in any order
    a. Add translation and scaling to the object (remember the correct order!)
    b. Change the object to a cube
    c. Set the color to red by using the fragment shader
    d. Add zooming to the keyboard inputs
    e. Implement a second object with its own model matrix
    f. Rather than defining the model by hand, load an object file