Lab 1. Creating an OpenGL Framework

Introduction

To write OpenGL programs we must be able to interact with the underlying graphics system. In particular, displaying output requires manipulation of the operating system's window manager. To simplify this process, we use GLUT - OpenGL Utility Toolkit – module. Through some simple initialization, GLUT will take care of all the window management as well as provide for basic user interaction via handling keyboard and mouse events. GLUT is an ***event driven*** API, meaning that we will be defining ***callback functions*** that will be executed in response to various system generated events, e.g. if the window needs to be redrawn, a mouse button has been clicked, a key has been pressed, etc.

Initialize GLUT

In order to have GLUT handle the window management tasks, typically our main program will contain the following sections:

1. initialize the library (module)
2. set the various window properties
3. create the window
4. register any needed callbacks
5. begin the (infinite) event loop

The corresponding Python/OpenGL code is given hereafter.

This code represents the structure of the Python program that will serve as a template for all the programs we will write in this course through modifying the window properties (and the objects it contains) and registering additional callbacks.

```
# Initialize GLUT
glutInit()

# Initialize the window with double buffering and RGB colors
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB)

# Set the window size to 500x500 pixels
glutInitWindowSize(500, 500)

# Create the window and give it a title
glutCreateWindow("My First OpenGL Window")

# Set the initial window position to (50, 50)
glutInitWindowPosition(50, 50)

# Define callbacks
glutDisplayFunc(display)

# Begin event loop
glutMainLoop()
```

Draw in display()

The display callback is executed every time the window needs to be displayed, including when the program first starts. Later we will see how to use this routine to perform animations and interactive scenes. We will begin by simply drawing a red square by (1) clearing the screen, (2) draw the scene (simply a red square), (3) flushing the buffer, or (4) swapping the buffer to display it on the screen (via double buffering).

The corresponding Python/OpenGL code is given hereafter.

```
# Disply callback function
def display():
    # Reset background
    glClear(GL_COLOR_BUFFER_BIT);

    # Render scene
    render_Scene();

    # Flush buffer
    glFlush()
OR
    # Swap buffers
    glutSwapBuffers();
```

Rendering the scene

To add more modularity in our code we can put all the drawing commands (rendering) in the user defined `render_Scene()` that we call from within the `display()` function (otherwise we can put them in the `display()` function itself).

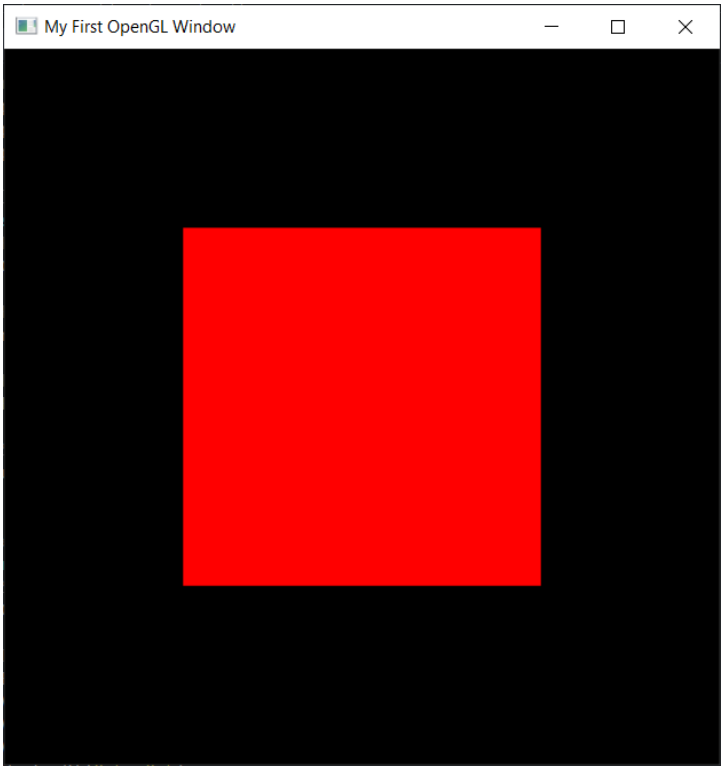
The Python/OpenGL code corresponding to drawing a square is given hereafter (we will see more details about squares drawing later in the following lab sessions).

```
# Scene render function
def render_Scene():
{
    # Set current color to red
    glColor3f(1.0,0.0,0.0);

    # Draw a square
    glBegin(GL_POLYGON)
    glVertex2f(-0.5,-0.5)
    glVertex2f(-0.5,0.5)
    glVertex2f(0.5,0.5)
    glVertex2f(0.5,-0.5)
    glEnd();
}
```

The complete template code is the following:

```
# Importing the necessary Modules
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *

# Disply callback function
def display():
    # Reset background
    glClear(GL_COLOR_BUFFER_BIT)

    # Render scene
    render_Scene()

    # Flush buffer
    glFlush()

    # Swap buffers
    glutSwapBuffers()


# Scene render function
def render_Scene():
    # Set current color to red
    glColor3f(1.0,0.0,0.0)

    # Draw a square
    glBegin(GL_POLYGON)
    glVertex2f(-0.5,-0.5)
    glVertex2f(-0.5,0.5)
    glVertex2f(0.5,0.5)
    glVertex2f(0.5,-0.5)
    glEnd()

# Initialize GLUT
glutInit()

# Initialize the window with double buffering and RGB colors
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB)

# Set the window size to 500x500 pixels
glutInitWindowSize(500, 500)

# Create the window and give it a title
glutCreateWindow("My First OpenGL Window")

# Set the initial window position to (50, 50)
glutInitWindowPosition(50, 50)

# Define callbacks
glutDisplayFunc(display)

# Begin event loop
glutMainLoop()
```

The output resulting from the execution of the previous code is the following:



Exercises

1. Search the Internet to find how we can change the background color of the window (black by default) and set it to white (or any other color of your choice).
2. Draw the previous window in the middle of the screen (horizontally and vertically) with dimensions (width and height) equal to the half of those of your computer screen.
   Hint: you can get your screen resolution using the function `size()` and the corresponding properties `width` and `height` of the Python module `pyautogui` (search how to use it on the Internet).