Computer Graphics Lab Manual

Lab: Introduction to OpenGL (Python with PyOpenGL)

Author: Md. Manirujjaman, Lecturer, Dept. of CSE, BAUST

1. Introduction

OpenGL (Open Graphics Library) is a cross-platform, open standard API used for rendering 2D and 3D graphics.

Key Points:
- Provides a set of functions to interact with the GPU (Graphics Processing Unit).
- Widely used in games, simulations, CAD software, and visualizations.
- Works across platforms (Windows, Linux, macOS, etc.).

Why do we need OpenGL?
- Hardware acceleration: Uses GPU for faster rendering than CPU.
- Cross-platform compatibility: Same code runs on different operating systems.
- Real-time graphics: Essential for games, animations, VR, etc.
- Industry standard: Used in education, research, and commercial applications.

2. Required Libraries

```
from OpenGL.GL import *       # Core OpenGL functions for drawing, transformations, colors, etc.
from OpenGL.GLUT import *     # Utility Toolkit for window management and input handling
from OpenGL.GLU import *      # Utility library for advanced operations like projections
```

3. Basic OpenGL Window

```
def show():
    glClear(GL_COLOR_BUFFER_BIT)
# Clear the screen. Think of your window as a whiteboard. Before drawing #anything new, you usually want to wipe the board clean.That's what this #command does → it clears the screen to the background color (default is black)
    glutSwapBuffers()
 # Swap buffers to display content. You have two canvases: Front buffer = the one #currently being shown on the screen. Back buffer = the hidden one where OpenGL is #drawing the next frame.
```

```
# Main setup
glutInit()                                  # Initialize GLUT
glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE) # Set RGBA color and
double buffering
glutInitWindowSize(500, 500)                # Set window size
glutCreateWindow(b"Simple OpenGL Window")   # Create window with title
glutDisplayFunc(show)                       # Set display callback
function
glutMainLoop()                              # Start main event loop
```

4. Setting Background Color

```
glClearColor(0.2, 0.6, 1.0, 1.0)  # RGBA (Red, Green, Blue, Alpha)
```

5. Setting Coordinate System (Graph Paper)

```
glLoadIdentity()                    # Reset transformations
glOrtho(0, 500, 0, 500, -1, 1)      # Set 2D coordinate system from 0
to 500 in X and Y
```

6. Drawing Shapes

6.1 Drawing a Square (Yellow)

```
glColor3f(1, 1, 0)                # Set color to yellow
glBegin(GL_QUADS)                 # Start drawing quadrilateral
for v in [(100,100),(200,100),(200,200),(100,200)]:
    glVertex2f(*v)                # Specify vertices
glEnd()
```

Finish drawing

6.2 Drawing a Triangle (Blue)

```
glColor3f(0, 0, 1)                    # Set color to blue
glBegin(GL_TRIANGLES)                 # Start drawing a triangle
for v in [(220,100),(320,100),(270,200)]:
    glVertex2f(*v)                    # Specify triangle vertices
glEnd()                               # Finish drawing the triangle
```

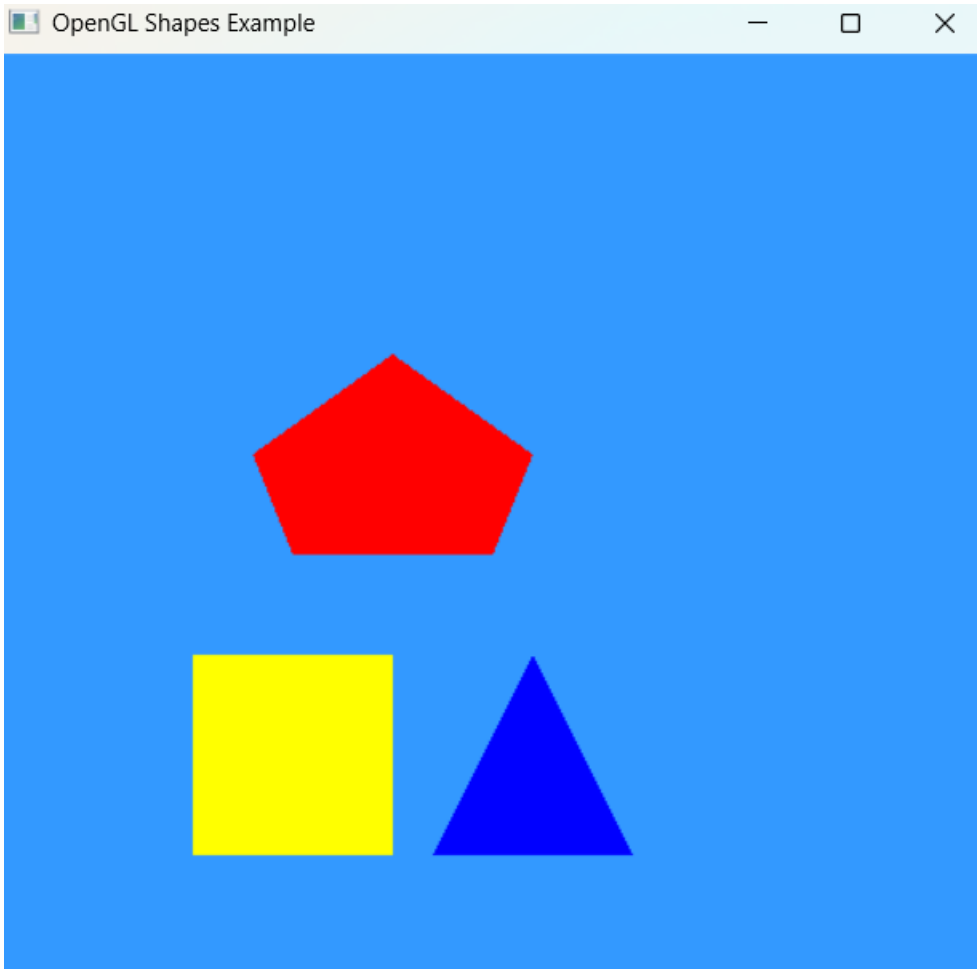
Finish drawing

6.3 Drawing a Polygon (Red)

```
# Polygon (red) - on top of both
    glColor3f(1, 0, 0)
    glBegin(GL_POLYGON)
    for v in [(150,250),(250,250),(270,300),(200,350),(130,300)]:
        glVertex2f(*v)
    glEnd()
```

7. Complete Example Code

```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *

def show():
    glClear(GL_COLOR_BUFFER_BIT)
    glLoadIdentity()
    glOrtho(0, 500, 0, 500, -1, 1)

    # Draw Yellow Square
    glColor3f(1, 1, 0)
    glBegin(GL_QUADS)
    for v in [(100,100),(200,100),(200,200),(100,200)]:
        glVertex2f(*v)
    glEnd()

    # Draw Blue Triangle
    glColor3f(0, 0, 1)
    glBegin(GL_TRIANGLES)
    for v in [(220,100),(320,100),(270,200)]:
        glVertex2f(*v)
    glEnd()

    # Draw Red Polygon
    glColor3f(1, 0, 0)
    glBegin(GL_POLYGON)
    for v in [(150,250),(250,250),(270,300),(200,350),(130,300)]:
        glVertex2f(*v)
    glEnd()

    glutSwapBuffers()

# Main setup
glutInit()
glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE)
glutInitWindowSize(500, 500)
glutCreateWindow(b"OpenGL Shapes Example")
glClearColor(0.2, 0.6, 1.0, 1.0)
glutDisplayFunc(show)
glutIdleFunc(show)
glutMainLoop()
```



8. Notes

- Double buffering prevents flickering by drawing in a hidden back buffer first.
- glLoadIdentity() ensures transformations do not accumulate.
- Shapes are drawn in the order specified: later shapes appear on top.
- Using glOrtho() allows easy 2D drawing with coordinates like graph paper.
- glutIdleFunc(show) ensures the window continuously redraws, useful for animations.