Experiment No 1

Title Install and explore the OpenGL.

Create cpp file and write your code

Run on terminal:

g++ MyProg.cpp -lGL -lGLU -lglut (for C++ program)

./a.out

Basic Code # Triangle

In C++

// A simple introductory program; its main window contains a static picture

// of a triangle, whose three vertices are red, green and blue. The program

#include <GL/glut.h>

// Clears the current window and draws a triangle.

void display() {

// Set every pixel in the frame buffer to the current clear color.

glClear(GL\_COLOR\_BUFFER\_BIT);

// Drawing is done by specifying a sequence of vertices. The way these

// vertices are connected (or not connected) depends on the argument to

// glBegin. GL\_POLYGON constructs a filled polygon.

glBegin(GL\_POLYGON);

glColor3f(1, 0, 0); glVertex3f(-0.6, -0.75, 0.5);

glColor3f(0, 1, 0); glVertex3f(0.6, -0.75, 0);

glColor3f(0, 0, 1); glVertex3f(0, 0.75, 0);

glEnd();

// Flush drawing command buffer to make drawing happen as soon as possible.

glFlush();

}

// Initializes GLUT, the display mode, and main window; registers callbacks;

// enters the main event loop.

int main(int argc, char\*\* argv) {

// Use a single buffered window in RGB mode (as opposed to a double-buffered

// window or color-index mode).

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

// Position window at (80,80)-(480,380) and give it a title.

glutInitWindowPosition(80, 80);

glutInitWindowSize(400, 300);

glutCreateWindow("A Simple Triangle");

// Tell GLUT that whenever the main window needs to be repainted that it

// should call the function display().

glutDisplayFunc(display);

// Tell GLUT to start reading and processing events. This function

// never returns; the program only exits when the user closes the main

// window or kills the process.

glutMainLoop();

}

How to run glut/ OpenGL in g++ (Ubuntu)

g++ MyProg.cpp -lGL -lGLU -lglut

OUTPUT

