**SSN COLLEGE OF ENGINEERING, KALAVAKKAM**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**UCS1712 – GRAPHICS AND MULTIMEDIA LAB**

------------------------------------------------------------------------------------------------------------

**Lab Exercise 1** : Study of Basic Output Primitives in C++ using OpenGL

1. To create an output window using OPENGL and to draw the following basic output primitives – POINTS, LINES, LINE\_STRIP, LINE\_LOOP, TRIANGLES, QUADS, QUAD\_STRIP, POLYGON.
2. To create an output window and draw a checkerboard using OpenGL.
3. To create an output window and draw a house using POINTS,LINES,TRAINGLES and QUADS/POLYGON.

***Aim:***

Study of basic output primitives in c++ using openGL.

***Algorithm:***

1. Include necessary header files.
2. Create an initialization function (**myInit**) to set up OpenGL settings.
3. Define a function (**drawShape**) to draw various shapes with vertices and labels.
4. Create a function (**myDisplay**) for rendering:
   * Clear the color buffer.
   * Draw points, lines, quads, triangles, and polygons using **drawShape**.
   * Label points and lines with their coordinates using **drawShape**.
5. In the **main** function:
   * Initialize GLUT and set display mode.
   * Create a window, set the display function to **myDisplay**, and initialize OpenGL settings.
   * Enter the GLUT main loop.

***Code:***

***a.cpp:***

// #include <GLUT/glut.h> //in clg system

#include <GL/glut.h> //my laptop

#include <stdio.h>

#include <cstring>

#include <iostream>

using namespace std;

void myInit()

{

glClearColor(0.0, 0.0, 0.0, 0.0); // used for glClear: sets bitplane window

glPointSize(10);

glMatrixMode(GL\_PROJECTION); // applies the matrix operations to corresponding stack

/\*

GL\_MODELVIEW - modelview matrix stack.

GL\_PROJECTION - projection

GL\_TEXTURE - texture

GL\_COLOR - color

\*/

glLoadIdentity(); // replaces the current matrix with the identity matrix

gluOrtho2D(0.0, 640.0, 0.0, 480.0); // sets up a 2D orthographic viewing region

// for (0,0) to be at the center of the screen, put it like (-320,320,-240,240)

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*version 1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*:

// void myDisplay()

// {

// glClear(GL\_COLOR\_BUFFER\_BIT);

// glBegin(GL\_POINTS);

// glVertex2d(150, 100);

// glEnd();

// glBegin(GL\_LINES);

// glVertex2d(150, 150);

// glVertex2d(150, 200);

// glEnd();

// glBegin(GL\_QUADS);

// glColor3f(0.0f, 1.0f, 0.0f);

// glVertex2d(300, 300);

// glVertex2d(300, 350);

// glVertex2d(350, 350);

// glVertex2d(350, 300);

// glVertex2d(300, 300);

// glEnd();

// glBegin(GL\_TRIANGLES); // Each set of 3 vertices form a triangle

// glColor3f(0.0f, 0.0f, 1.0f);

// glVertex2d(400, 400);

// glVertex2d(400, 450);

// glVertex2d(450, 450);

// glVertex2d(400, 400);

// glEnd();

// glBegin(GL\_POLYGON); // These vertices form a closed polygon

// glColor3f(1.0f, 1.0f, 0.0f); // Yellow

// glVertex2d(200, 200);

// glVertex2d(200, 220);

// glVertex2d(220, 240);

// glVertex2d(240, 200);

// glVertex2d(230, 250);

// glVertex2d(250, 250);

// glVertex2d(200, 200);

// glEnd();

// glFlush();

// }

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*version 2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*:

// Function to convert GLenum value to string

string GLenumToString(GLenum mode)

{

switch (mode)

{

case GL\_POINTS:

return "GL\_POINTS";

case GL\_LINES:

return "GL\_LINES";

case GL\_LINE\_STRIP:

return "GL\_LINE\_STRIP";

case GL\_LINE\_LOOP:

return "GL\_LINE\_LOOP";

case GL\_TRIANGLES:

return "GL\_TRIANGLES";

case GL\_TRIANGLE\_STRIP:

return "GL\_TRIANGLE\_STRIP";

case GL\_TRIANGLE\_FAN:

return "GL\_TRIANGLE\_FAN";

case GL\_QUADS:

return "GL\_QUADS";

case GL\_QUAD\_STRIP:

return "GL\_QUAD\_STRIP";

case GL\_POLYGON:

return "GL\_POLYGON";

default:

return "Unknown";

}

return "";

}

// Common function to draw shapes

void drawShape(GLenum mode, double vertices[], int numVertices)

{

// glVertex2d-2d-d=double

glBegin(mode);

cout << "Drawing mode: " << GLenumToString(mode) << " (Value: " << mode << ")\n";

for (int i = 0; i < numVertices; i += 2)

{

// glVertex2d(vertices[i], vertices[i + 1]);

double x = vertices[i];

double y = vertices[i + 1];

cout << "Vertex (" << x << ", " << y << ")" << endl; // Print coordinates

glVertex2d(x, y);

}

glEnd();

cout << endl;

}

// Function to draw text at a given position

void drawText(double x, double y, int a = 0, int b = 0)

{

char text[20];

snprintf(text, 20, "(%0.0f,%0.0f)", x, y);

glRasterPos2d(x + a, y + b);

for (int i = 0; text[i] != '\0'; i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10, text[i]);

memset(text, 0, sizeof(text));

}

void myDisplay()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

double pointVertices[] = {150, 100};

double lineVertices[] = {150, 150, 150, 200};

double quadVertices[] = {300, 300, 300, 350, 350, 350, 350, 300, 300, 300};

double triangleVertices[] = {400, 400, 400, 450, 450, 450, 400, 400};

double polygonVertices[] = {200, 200, 200, 220, 220, 240, 240, 200, 230, 250, 250, 250, 200, 200};

// Draw points

drawShape(GL\_POINTS, pointVertices, sizeof(pointVertices) / sizeof(pointVertices[0]));

drawText(pointVertices[0], pointVertices[1], 10, -5);

// Draw lines

drawShape(GL\_LINES, lineVertices, sizeof(lineVertices) / sizeof(lineVertices[0]));

drawText(lineVertices[0], lineVertices[1], 10, -5);

drawText(lineVertices[2], lineVertices[3], -55,-5);

// Draw quads

glBegin(GL\_QUADS);

glColor3f(0.0f, 1.0f, 0.0f);

drawShape(GL\_QUADS, quadVertices, sizeof(quadVertices) / sizeof(quadVertices[0]));

glEnd();

// Draw triangles

glColor3f(0.0f, 0.0f, 1.0f);

drawShape(GL\_TRIANGLES, triangleVertices, sizeof(triangleVertices) / sizeof(triangleVertices[0]));

// Draw polygon

glBegin(GL\_POLYGON);

glColor3f(1.0f, 1.0f, 0.0f); // Yellow

drawShape(GL\_POLYGON, polygonVertices, sizeof(polygonVertices) / sizeof(polygonVertices[0]));

glEnd();

glFlush();

}

int main(int argc, char \*argv[])

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640, 480);

glutCreateWindow("1-a");

glutDisplayFunc(myDisplay);

myInit();

glutMainLoop();

return 1;

}

***b.cpp:***

// #include <GLUT/glut.h> //in clg system

#include <GL/glut.h> //my laptop

int windowWidth = 800;

int windowHeight = 800;

void myInit()

{

glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

// glPointSize(10);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, windowWidth, 0.0, windowHeight);

// why?

// glMatrixMode(GL\_MODELVIEW);

}

void drawCheckerboard()

{

// glVertex2i-2i-i=int

int rows = 8;

int cols = 8;

int squareSize = windowWidth / cols;

glColor3f(1.0f, 1.0f, 1.0f);

glBegin(GL\_QUADS);

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

((i + j) & 1) ? glColor3f(0.0f, 0.0f, 0.0f) : glColor3f(1.0f, 1.0f, 1.0f);

glVertex2i(j \* squareSize, i \* squareSize);

glVertex2i((j + 1) \* squareSize, i \* squareSize);

glVertex2i((j + 1) \* squareSize, (i + 1) \* squareSize);

glVertex2i(j \* squareSize, (i + 1) \* squareSize);

}

}

glEnd();

glFlush();

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

drawCheckerboard();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(windowWidth, windowHeight);

glutCreateWindow("1-b");

glutDisplayFunc(display);

myInit(); // glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

glutMainLoop();

return 1;

}

***c.cpp:***

// #include <GLUT/glut.h> //in clg system

#include <GL/glut.h> //my laptop

int windowWidth = 800;

int windowHeight = 600;

void myInit()

{

glClearColor(1.0f, 1.0f, 1.0f, 0.0f); //White BG

// glPointSize(10);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, windowWidth, 0.0, windowHeight);

// why?

// glMatrixMode(GL\_MODELVIEW);

}

void drawHouse()

{

// Clear the screen

// Draw the house using different primitive shapes

// Draw the base of the house using a quad

glColor3f(0.59f, 0.85f, 0.71f); // Gray color

glBegin(GL\_QUADS);

glVertex2i(100, 100);

glVertex2i(500, 100);

glVertex2i(500, 400);

glVertex2i(100, 400);

glEnd();

// Draw the roof using triangles

glColor3f(1.0f, 0.0f, 0.0f); // Red color

glBegin(GL\_TRIANGLES);

glVertex2i(100, 400);

glVertex2i(300, 600);

glVertex2i(500, 400);

glEnd();

// Draw the door using quads

glColor3f(1.0f, 1.0f, 1.0f); // Blue color

glBegin(GL\_QUADS);

glVertex2i(250, 100);

glVertex2i(350, 100);

glVertex2i(350, 300);

glVertex2i(250, 300);

// glEnd();

glColor3f(0.36f, 0.05f, 0.05f);

// glBegin(GL\_QUADS);

glVertex2i(250, 100);

glVertex2i(330, 130);

glVertex2i(330, 300);

glVertex2i(250, 300);

// glEnd();

// Draw the windows using quads

glColor3f(0.06f, 0.22f, 0.45f); // Green color

// glBegin(GL\_QUADS);

glVertex2i(150, 200);

glVertex2i(200, 200);

glVertex2i(200, 250);

glVertex2i(150, 250);

glVertex2i(400, 200);

glVertex2i(450, 200);

glVertex2i(450, 250);

glVertex2i(400, 250);

glEnd();

glFlush();

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

drawHouse();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(windowWidth, windowHeight);

glutCreateWindow("Drawing a House using OpenGL");

glutDisplayFunc(display);

myInit(); // glClearColor(1.0f, 1.0f, 1.0f, 0.0f); // White background

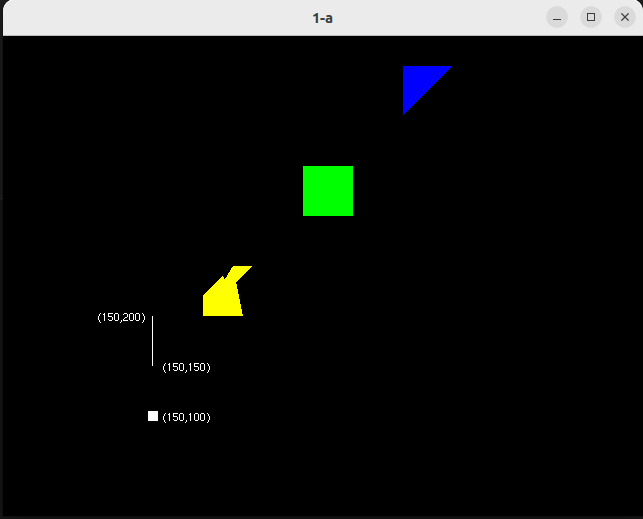
glutMainLoop();

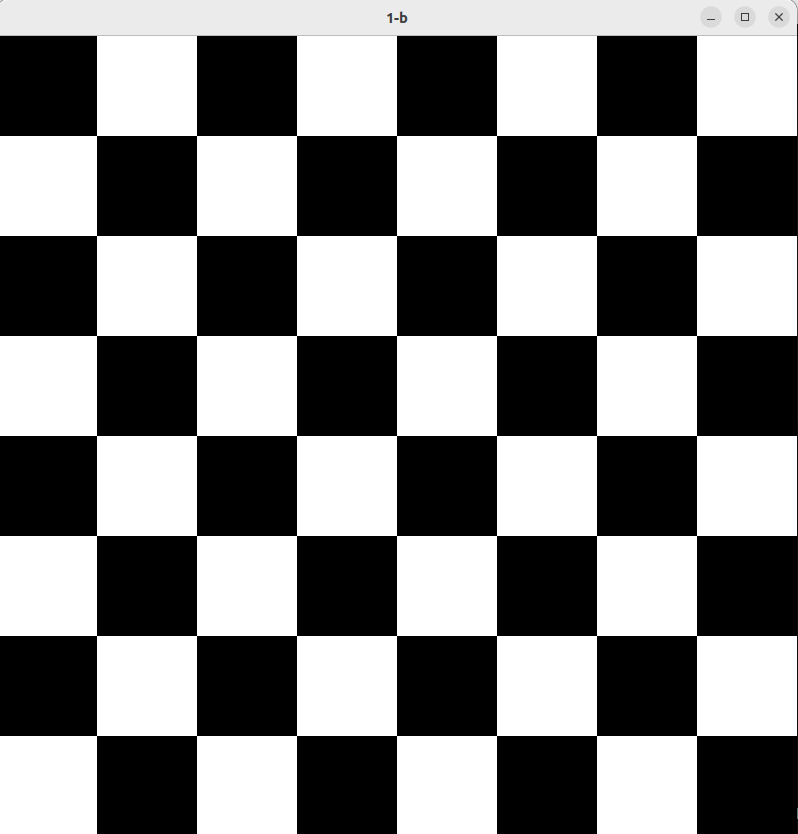
return 0;

}

***run.sh:***g++ a.cpp -lGL -lglut -lGLU

./a.out  
  
***Sample I/O:***







***Learning Outcomes:***

Thus, the following shapes/objects have been created using OpenGL primitive