**LAB 19**

//opengl app for illustration of orthogonal projection

#include <GL/glut.h>

int win\_width = 800;

int win\_height = 600;

void drawCube() {

glBegin(GL\_QUADS);

// Front face

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

glVertex3f(-0.5f, -0.5f, 0.5f);

glVertex3f(0.5f, -0.5f, 0.5f);

glVertex3f(0.5f, 0.5f, 0.5f);

glVertex3f(-0.5f, 0.5f, 0.5f);

// Back face

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

glVertex3f(-0.5f, -0.5f, -0.5f);

glVertex3f(0.5f, -0.5f, -0.5f);

glVertex3f(0.5f, 0.5f, -0.5f);

glVertex3f(-0.5f, 0.5f, -0.5f);

// Left face

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

glVertex3f(-0.5f, -0.5f, -0.5f);

glVertex3f(-0.5f, -0.5f, 0.5f);

glVertex3f(-0.5f, 0.5f, 0.5f);

glVertex3f(-0.5f, 0.5f, -0.5f);

// Right face

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

glVertex3f(0.5f, -0.5f, -0.5f);

glVertex3f(0.5f, -0.5f, 0.5f);

glVertex3f(0.5f, 0.5f, 0.5f);

glVertex3f(0.5f, 0.5f, -0.5f);

// Top face

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

glVertex3f(-0.5f, 0.5f, 0.5f);

glVertex3f(0.5f, 0.5f, 0.5f);

glVertex3f(0.5f, 0.5f, -0.5f);

glVertex3f(-0.5f, 0.5f, -0.5f);

// Bottom face

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

glVertex3f(-0.5f, -0.5f, 0.5f);

glVertex3f(0.5f, -0.5f, 0.5f);

glVertex3f(0.5f, -0.5f, -0.5f);

glVertex3f(-0.5f, -0.5f, -0.5f);

glEnd();

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(1.5, 1.5, 1.5, 0, 0, 0, 0, 1, 0);

drawCube();

glutSwapBuffers();

}

void reshape(int w, int h) {

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-1.5, 1.5, -1.5, 1.5, -10.0, 10.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(win\_width, win\_height);

glutInitWindowPosition(100, 100);

glutCreateWindow("LAB-19 Niraj Maharjan");

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

glEnable(GL\_DEPTH\_TEST);

glutDisplayFunc(display);

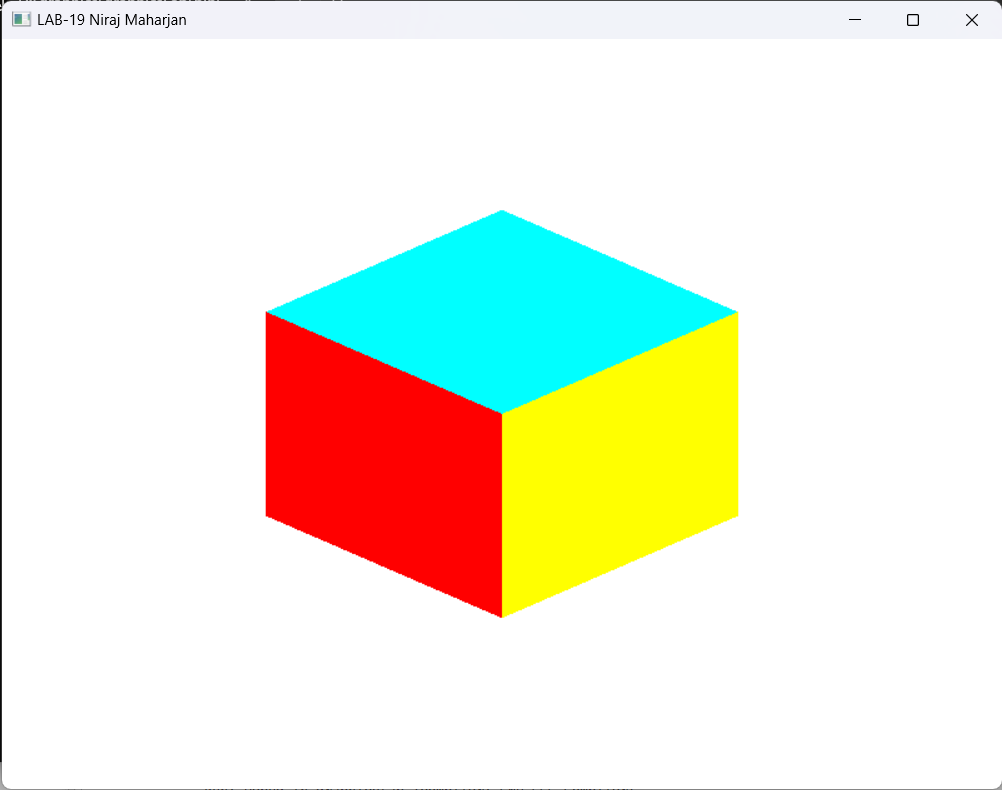
glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}

**OUTPUT**

****

**LAB 20**

//opengl app tp show clipping using orthogonal project

#include <GL/glut.h>

int win\_width = 800;

int win\_height = 600;

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

// Set up the clipping region using glOrtho

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-0.4, 0.4, -0.4, 0.4, -1.0, 1.0);

// Draw the original non-filled triangle

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

glBegin(GL\_LINE\_LOOP);

glVertex2f(-0.5f, -0.4f);

glVertex2f(0.7f, -0.2f);

glVertex2f(0.2f, 0.7f);

glEnd();

glutSwapBuffers();

}

void reshape(int w, int h) {

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-1.0, 1.0, -1.0, 1.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(win\_width, win\_height);

glutInitWindowPosition(100, 100);

glutCreateWindow("LAB-20 Niraj Maharjan");

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

glutDisplayFunc(display);

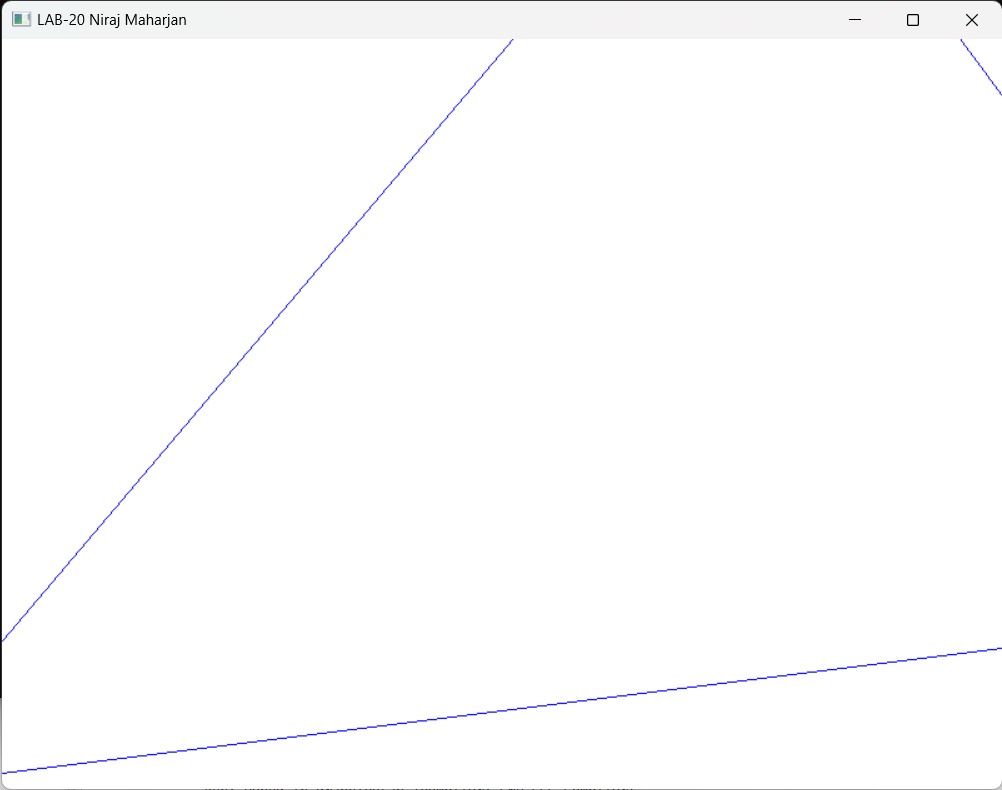
glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}

**OUTPUT**

****

**LAB 21**

//OPENGL app to show rectangle in two different viewports

#include <GL/glut.h>

int win\_width = 800;

int win\_height = 600;

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

// Set up viewport for the left rectangle

glViewport(0, 0, win\_width / 2, win\_height);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// Draw the left rectangle

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

glBegin(GL\_POLYGON);

glVertex2f(-0.8f, -0.8f);

glVertex2f(-0.2f, -0.8f);

glVertex2f(-0.2f, 0.8f);

glVertex2f(-0.8f, 0.8f);

glEnd();

// Set up viewport for the right rectangle

glViewport(win\_width / 2, 0, win\_width / 2, win\_height);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// Draw the right rectangle

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

glBegin(GL\_POLYGON);

glVertex2f(0.2f, -0.8f);

glVertex2f(0.8f, -0.8f);

glVertex2f(0.8f, 0.8f);

glVertex2f(0.2f, 0.8f);

glEnd();

glutSwapBuffers();

}

void reshape(int w, int h) {

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-1.0, 1.0, -1.0, 1.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(win\_width, win\_height);

glutInitWindowPosition(100, 100);

glutCreateWindow("LAB-21 Niraj Maharjan");

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

glutDisplayFunc(display);

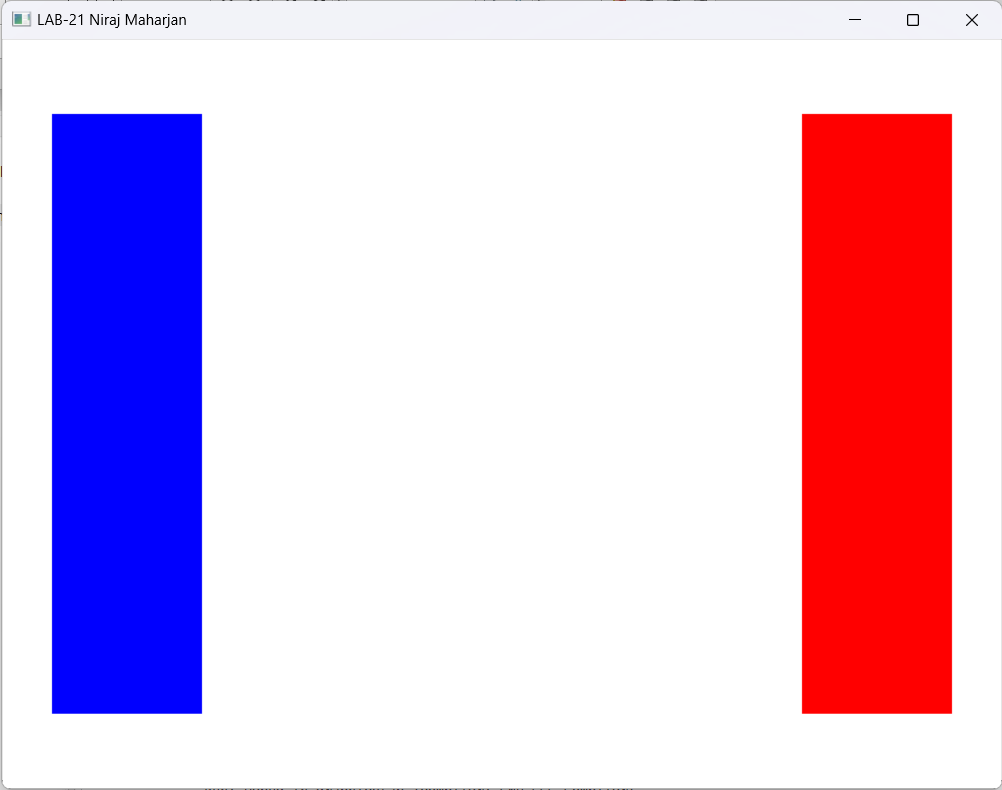
glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}

**OUTPUT**

****

**LAB-22**

//OPEN\_GL implementation of cohen shuterland algorithm

#include <GL/glut.h>

int wx\_max = 100, wy\_max = 100, wx\_min = 50, wy\_min = 50;

int x\_1 = 40, x\_2 = 100, y\_1 = 50, y\_2 = 150;

int getCode(int x, int y){

int code = 0000;

if (x < wx\_min)

code |= 1;

else if (x > wx\_max)

code |= 2;

if (y < wy\_min)

code |= 4;

else if (y > wy\_max)

code |= 8;

return code;

}

void drawline(int x\_1, int y\_1, int x\_2, int y2){

glClear(GL\_COLOR\_BUFFER\_BIT);

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

glLineWidth(2.0);

// Draw the window

glBegin(GL\_LINE\_LOOP);

glVertex2i(wx\_min, wy\_min);

glVertex2i(wx\_max, wy\_min);

glVertex2i(wx\_max, wy\_max);

glVertex2i(wx\_min, wy\_max);

glEnd();

// Draw the line

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

glBegin(GL\_LINES);

glVertex2i(x\_1, y\_1);

glVertex2i(x\_2, y2);

glEnd();

glFlush();

}

void cohen\_sutherland(){

int code1 = getCode(x\_1, y\_1);

int code2 = getCode(x\_2, y\_2);

bool accept = false;

do

{

// Check if the line is completely inside or outside the window

if ((code1 == 0) && (code2 == 0))

{

accept = true;

break;

}

else if (code1 & code2)

{

break;

}

else

{

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

// Clipping variables

int code;

float m = (float)(y\_2 - y\_1) / (float)(x\_2 - x\_1);

float c = y\_1 - m \* x\_1;

float x, y;

// Find the intersection points

if (code1 != 0)

code = code1;

else

code = code2;

if (code & 1) // Left boundary

{

y = m \* wx\_min + c;

x = wx\_min;

}

else if (code & 2) // Right boundary

{

y = m \* wx\_max + c;

x = wx\_max;

}

else if (code & 4) // Bottom boundary

{

x = (wy\_min - c) / m;

y = wy\_min;

}

else if (code & 8) // Top boundary

{

x = (wy\_max - c) / m;

y = wy\_max;

}

if (code == code1)

{

x\_1 = x;

y\_1 = y;

code1 = getCode(x\_1, y\_1);

}

else

{

x\_2 = x;

y\_2 = y;

code2 = getCode(x\_2, y\_2);

}

}

} while (accept == false);

}

void display(){

drawline(x\_1, y\_1, x\_2, y\_2);

Sleep(1000);

cohen\_sutherland();

drawline(x\_1, y\_1, x\_2, y\_2);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(800, 600);

glutCreateWindow("LAB-22 Niraj Maharjan");

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-20, 200, -20, 200);

glMatrixMode(GL\_MODELVIEW);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**OUTPUT:**

