18. Do the same thing 17 without using opengl transformation methods and see if the results are same.

Source Code

#include <GL/glut.h>

#include <cmath>

float x, y;

float width, height;

int win\_width = 800;

int win\_height = 600;

void drawRectangle(float x, float y, float width, float height) {

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

glLineWidth(2.0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(x, y);

glVertex2f(x + width, y);

glVertex2f(x + width, y + height);

glVertex2f(x, y + height);

glEnd();

}

void matrixMultiply(float matrix[3][3], float& x, float& y) {

float tempX = matrix[0][0] \* x + matrix[0][1] \* y + matrix[0][2];

float tempY = matrix[1][0] \* x + matrix[1][1] \* y + matrix[1][2];

x = tempX;

y = tempY;

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

// Draw the original rectangle

drawRectangle(x, y, width, height);

// Calculate the center of rotation

float centerX = 150.0f;

float centerY = 125.0f;

// Create rotation matrix

float angle = 30.0 \* M\_PI / 180.0;

float cosA = cos(angle);

float sinA = sin(angle);

float rotationMatrix[3][3] = {

{cosA, -sinA, centerX \* (1 - cosA) + centerY \* sinA},

{sinA, cosA, centerY \* (1 - cosA) - centerX \* sinA},

{0, 0, 1}

};

// Rotate the vertices of the rectangle using matrix multiplication

float x1 = x;

float y1 = y;

float x2 = x + width;

float y2 = y;

float x3 = x + width;

float y3 = y + height;

float x4 = x;

float y4 = y + height;

matrixMultiply(rotationMatrix, x1, y1);

matrixMultiply(rotationMatrix, x2, y2);

matrixMultiply(rotationMatrix, x3, y3);

matrixMultiply(rotationMatrix, x4, y4);

// Draw the rotated rectangle

glBegin(GL\_LINE\_LOOP);

glVertex2f(x1, y1);

glVertex2f(x2, y2);

glVertex2f(x3, y3);

glVertex2f(x4, y4);

glEnd();

glFlush();

}

void reshape(int w, int h) {

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-100, win\_width, -100, win\_height);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

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

x = 100;

y = 100;

width = 200;

height = 50;

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(win\_width, win\_height);

glutInitWindowPosition(100, 100);

glutCreateWindow("Rotate Rectangle Using Matrix Multiplication-Atullya");

glClearColor(1.0, 1.0, 1.0, 1.0);

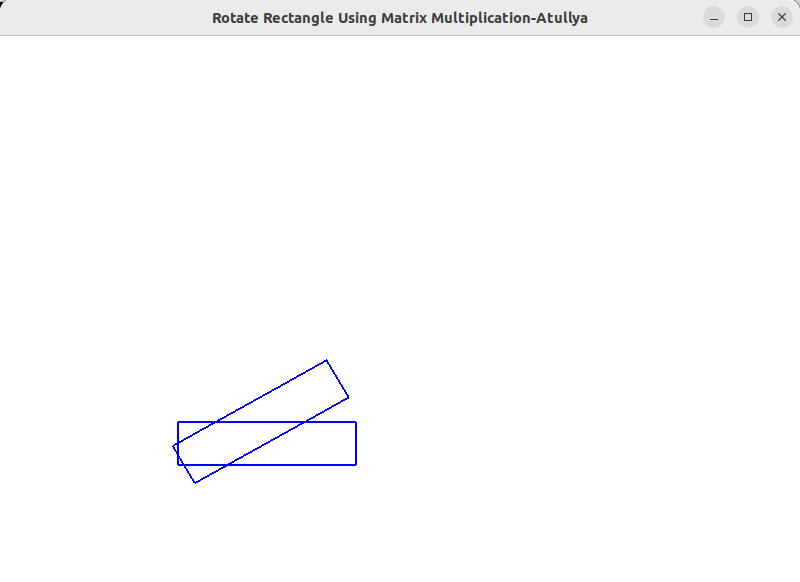
glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}



19. Write an OPENGL app to illustrate orthogonal projection

Source Code

#include <GL/glut.h>

#include <iostream>

int windowWidth = 800;

int windowHeight = 600;

void drawScene() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

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

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// Set the background color to white

glClearColor(1.0, 1.0, 1.0, 1.0);

// Clear the color buffer

glClear(GL\_COLOR\_BUFFER\_BIT);

// Set the square color to black

glColor3f(0.0, 0.0, 0.0);

// Draw a square

glBegin(GL\_QUADS);

glVertex2f(-0.5, -0.5);

glVertex2f(0.5, -0.5);

glVertex2f(0.5, 0.5);

glVertex2f(-0.5, 0.5);

glEnd();

glutSwapBuffers();

}

void reshape(int width, int height) {

glViewport(0, 0, width, height);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(windowWidth, windowHeight);

glutCreateWindow("Orthogonal Projection Example-Atullya");

glutDisplayFunc(drawScene);

glutReshapeFunc(reshape);

glClearColor(1.0, 1.0, 1.0, 1.0);

glutMainLoop();

return 0;

}



20. Write an OPENGL app to show clipping using orthogonal projection.

Source Code

#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("Clipping Using Orthogonal Projection-Atullya");

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

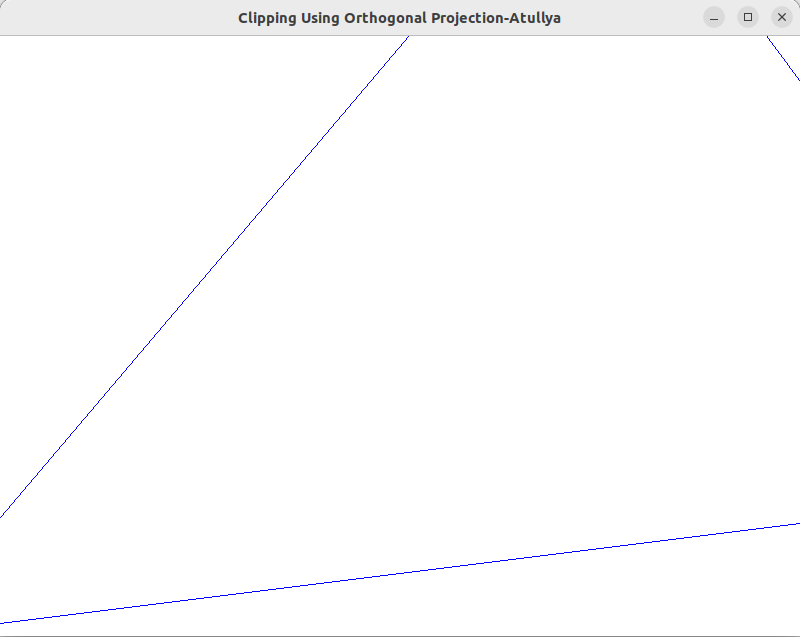
glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}



22. Implement sutherland cohen clipping algorithm to clip the given line.

Source Code

#include <GL/glut.h>

#include <iostream>

int wx\_max = 100, wy\_max = 150, wx\_min = 50, wy\_min = 50;

int x\_1, x\_2, y\_1, y\_2; // Variables to store user input

int getCode(int x, int y)

{

int code = 0;

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 x1, int y1, int x2, 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(x1, y1);

glVertex2i(x2, y2);

glEnd();

glFlush();

}

void cohen\_sutherland()

{

// ... (unchanged)

}

void timer(int value)

{

cohen\_sutherland();

glutPostRedisplay();

glutTimerFunc(1000, timer, 0);

}

void display()

{

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("Line Clipping - Cohen-Sutherland-Atullya");

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

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

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

glMatrixMode(GL\_MODELVIEW);

glutDisplayFunc(display);

// User input

std::cout << "Enter x1 y1 x2 y2: ";

std::cin >> x\_1 >> y\_1 >> x\_2 >> y\_2;

glutTimerFunc(1000, timer, 0); // Start the timer

glutMainLoop();

return 0;

}



