11. Implement midpoint algorithm to draw ellipse given its primary, secondary axis and center.

#include <GL/glut.h>

#include <iostream>

int WIN\_WIDTH = 800;

int WIN\_HEIGHT = 600;

void drawPoints(int x, int y, int x1, int y1) {

glBegin(GL\_POINTS);

glColor3d(0, 0, 0);

glVertex2i(x1 + x, y1 + y);

glVertex2i(x1 - x, y1 + y);

glVertex2i(x1 + x, y1 - y);

glVertex2i(x1 - x, y1 - y);

glEnd();

}

// Midpoint Circle Algorithm

void drawCircle() {

int x1, y1, a, b;

std::cout << "Enter the coordinates for ellipse:" << std::endl;

std::cout << "x1 = "; std::cin >> x1;

std::cout << "y1 = "; std::cin >> y1;

std::cout << "a = "; std::cin >> a;

std::cout << "b = "; std::cin >> b;

float p, x = 0, y = b;

p = b \* b - a \* a \* b + 0.25 \* a \* a;

while (2 \* b \* b \* x < 2 \* a \* a \* y) {

drawPoints(x, y, x1, y1);

x++;

if (p < 0) {

p += b \* b \* (2 \* x + 1);

} else {

y--;

p += b \* b \* (2 \* x + 1) - a \* a \* (2 \* y);

}

}

p = b \* b \* (x + 0.5) \* (x + 0.5) + a \* a \* (y - 1) \* (y - 1) - a \* a \* b \* b;

while (y >= 0) {

drawPoints(x, y, x1, y1);

y--;

if (p > 0) {

p += a \* a \* (1 - 2 \* y);

} else {

x++;

p += b \* b \* (2 \* x + 1) + a \* a \* (1 - 2 \* y);

}

}

glFlush();

}

void myInit() {

glutInitWindowSize(WIN\_WIDTH, WIN\_HEIGHT);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutCreateWindow(" (SAHIL BHATTARAI) - Drawing an ellipse using midpoint algorithm");

glClearColor(1, 1, 1, 1);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3d(0, 0, 0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-WIN\_WIDTH, WIN\_WIDTH, -WIN\_HEIGHT, WIN\_HEIGHT);

glMatrixMode(GL\_MODELVIEW);

}

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

glutInit(&argc, argv);

myInit();

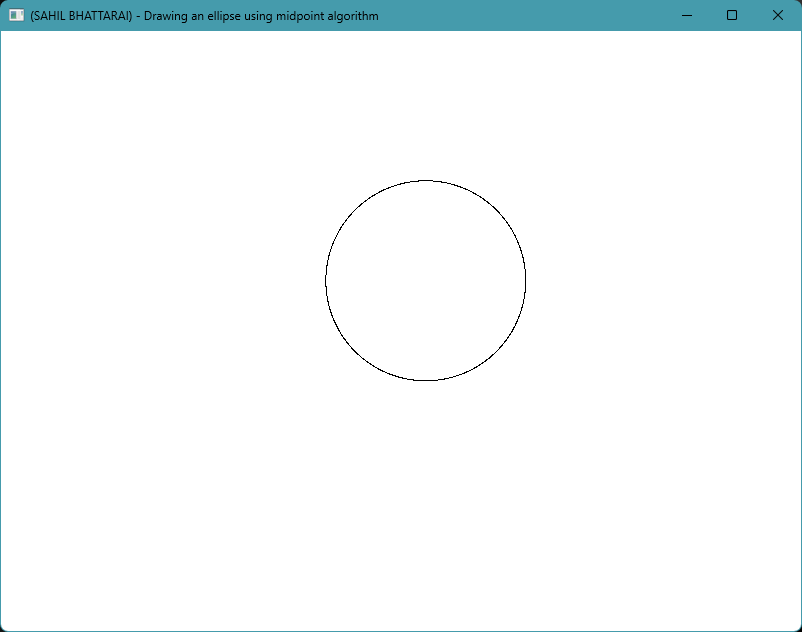
glutDisplayFunc(drawCircle);

glutMainLoop();

return 0;

}

**OUTPUT:**



13. Write an OPENGL app to rotate a rectangle about origin with one point being (100, 100), width 200 and height 50 by 30 degree anticlockwise without using opengl function.

#include <GL/glut.h>

#include <iostream>

#include <cmath>

using namespace std;

// Rotation of Square - without OpenGL methods

int ROT\_DEGREE = 30;

int WIN\_WIDTH = 800;

int WIN\_HEIGHT = 400;

void display() {

float ROT\_SIN\_DEGREE = sin(ROT\_DEGREE);

float ROT\_COS\_DEGREE = cos(ROT\_DEGREE);

glClear(GL\_COLOR\_BUFFER\_BIT);

glClearColor(1, 1, 1, 1);

int x = 100, y = 100, width=200, height=50;

// Original square

glColor3f(0, 0, 0);

glBegin(GL\_POLYGON);

glVertex2d(x, y);

glVertex2d(x + width, y);

glVertex2d(x + width, y + height);

glVertex2d(x, y + height);

glEnd();

// Translated square

glColor3f(1, 0, 0);

glBegin(GL\_POLYGON);

glVertex2d((x)\*ROT\_COS\_DEGREE - (y)\*ROT\_SIN\_DEGREE,

(x)\*ROT\_SIN\_DEGREE + (y)\*ROT\_COS\_DEGREE);

glVertex2d((x + width)\*ROT\_COS\_DEGREE - (y)\*ROT\_SIN\_DEGREE,

(x + width)\*ROT\_SIN\_DEGREE + (y)\*ROT\_COS\_DEGREE);

glVertex2d((x + width)\*ROT\_COS\_DEGREE - (y + height)\*ROT\_SIN\_DEGREE,

(x + width)\*ROT\_SIN\_DEGREE + (y + height)\*ROT\_COS\_DEGREE);

glVertex2d((x)\*ROT\_COS\_DEGREE - (y + height)\*ROT\_SIN\_DEGREE,

(x)\*ROT\_SIN\_DEGREE + (y + height)\*ROT\_COS\_DEGREE);

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB);

glutInitWindowSize(WIN\_WIDTH, WIN\_HEIGHT);

glutCreateWindow(" (SAHIL BHATTARAI) - Rotation without OpenGL");

glClearColor(1, 1, 1, 1);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-WIN\_WIDTH, WIN\_WIDTH, -WIN\_HEIGHT, WIN\_HEIGHT);

glMatrixMode(GL\_MODELVIEW);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

OUTPUT:

A screenshot of a computer

Description automatically generated

14. Write an OPENGL app to rotate a rectangle about origin with one point being (100, 100), width 200 and height 50 by 30 degree anticlockwise and scale it by (2,2) using opengl function.

#include <GL/glut.h>

#include <cmath>

float rectangleVertices[4][2] = {{100.0, 100.0}, {300.0, 100.0}, {300.0, 150.0}, {100.0, 150.0}};

float angle = 30.0; // Rotation angle in degrees

void drawRectangle() {

glColor3f(0.0, 1.0, 0.0); // Green color for the rectangle

glBegin(GL\_POLYGON);

for (int i = 0; i < 4; i++) {

glVertex2fv(rectangleVertices[i]);

}

glEnd();

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glLoadIdentity();

drawRectangle();

// Translate the rectangle to its origin (100, 100)

glTranslatef(-100.0, -100.0, 0.0);

// Rotate the rectangle by the specified angle

glRotatef(angle, 0.0, 0.0, 1.0);

// Translate the rectangle back to its original position

glTranslatef(100.0, 100.0, 0.0);

drawRectangle();

glScalef(2.0, 2.0, 1.0);

drawRectangle();

glFlush();

}

void init() {

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-700.0, 700.0, -700.0, 700.0);

glMatrixMode(GL\_MODELVIEW);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(400, 400);

glutCreateWindow("SAHIL BHATTARAI - Rectangle Scaling with OpenGL ");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

OUTPUT:

A screenshot of a computer

Description automatically generated

15. Do the same thing in Question no. 14 without using opengl scaling function.

#include <GL/glut.h>

#include <iostream>

#include <cmath>

using namespace std;

// Scaling of Rectangle - without OpenGL methods

int SCALE\_X = 2;

int SCALE\_Y = 2;

int WIN\_WIDTH = 1024;

int WIN\_HEIGHT = 768;

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glClearColor(1, 1, 1, 1);

int x = 100, y = 100, width=200, height=50;

// Original square

glColor3f(0, 0, 0);

glBegin(GL\_POLYGON);

glVertex2d(x, y);

glVertex2d(x + width, y);

glVertex2d(x + width, y + height);

glVertex2d(x, y + height);

glEnd();

// Scaled Square

glColor3f(1, 0, 0);

glBegin(GL\_POLYGON);

glVertex2d(x \* SCALE\_X, y \* SCALE\_Y);

glVertex2d((x + width) \* SCALE\_X, y \* SCALE\_Y);

glVertex2d((x + width) \* SCALE\_X, (y + height)\* SCALE\_Y);

glVertex2d(x \* SCALE\_X, (y + height)\* SCALE\_Y);

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB);

glutInitWindowSize(WIN\_WIDTH, WIN\_HEIGHT);

glutCreateWindow(" (SAHIL BHATTARAI) - Scaling without using OpenGL functions");

glClearColor(1, 1, 1, 1);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-WIN\_WIDTH, WIN\_WIDTH, -WIN\_HEIGHT, WIN\_HEIGHT);

glMatrixMode(GL\_MODELVIEW);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

OUTPUT:

A screenshot of a computer

Description automatically generated

16. Use the rectangle in Question no. 12 to rotate it by 30 degree anticlockwise about a fixed point (150,125) and see how the result is different from Question no. 12 by using OpenGL transformation methods.

#include <GL/glut.h>

#include <cmath>

float rectangleVertices[4][2] = {{100.0, 100.0}, {300.0, 100.0}, {300.0, 150.0}, {100.0, 150.0}};

float angle = 30.0; // Rotation angle in degrees

void drawRectangle() {

glColor3f(1.0, 1.0, 0.0); // Yellow color for the rectangle

glBegin(GL\_POLYGON);

for (int i = 0; i < 4; i++) {

glVertex2fv(rectangleVertices[i]);

}

glEnd();

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glLoadIdentity();

drawRectangle();

glTranslatef(-100.0, -100.0, 0.0);

glTranslatef(150.0, 125.0, 0.0);

glRotatef(angle, 0.0, 0.0, 1.0);

glTranslatef(-150.0, -125.0, 0.0);

drawRectangle();

glFlush();

}

void init() {

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-400.0, 400.0, -400.0, 400.0);

glMatrixMode(GL\_MODELVIEW);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(512, 512);

glutCreateWindow(" SAHIL BHATTARAI -Rectangle Rotation About Fixed Point ");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

OUTPUT:

A screenshot of a computer

Description automatically generated

18. Write an OPENGL app to illustrate orthogonal projection.

#include <GL/glut.h>

void drawCube() {

glBegin(GL\_QUADS);

// Front face

glVertex3f(-1.0, -1.0, 1.0);

glVertex3f(1.0, -1.0, 1.0);

glVertex3f(1.0, 1.0, 1.0);

glVertex3f(-1.0, 1.0, 1.0);

// Back face

glVertex3f(-1.0, -1.0, -1.0);

glVertex3f(1.0, -1.0, -1.0);

glVertex3f(1.0, 1.0, -1.0);

glVertex3f(-1.0, 1.0, -1.0);

// Left face

glVertex3f(-1.0, -1.0, 1.0);

glVertex3f(-1.0, -1.0, -1.0);

glVertex3f(-1.0, 1.0, -1.0);

glVertex3f(-1.0, 1.0, 1.0);

// Right face

glVertex3f(1.0, -1.0, 1.0);

glVertex3f(1.0, -1.0, -1.0);

glVertex3f(1.0, 1.0, -1.0);

glVertex3f(1.0, 1.0, 1.0);

// Top face

glVertex3f(-1.0, 1.0, 1.0);

glVertex3f(1.0, 1.0, 1.0);

glVertex3f(1.0, 1.0, -1.0);

glVertex3f(-1.0, 1.0, -1.0);

// Bottom face

glVertex3f(-1.0, -1.0, 1.0);

glVertex3f(1.0, -1.0, 1.0);

glVertex3f(1.0, -1.0, -1.0);

glVertex3f(-1.0, -1.0, -1.0);

glEnd();

}

void display() {

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

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-2.0, 2.0, -2.0, 2.0, -10.0, 10.0); // Orthogonal projection

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(3.0, 3.0, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); // Camera position TO BE COMMENTED

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

drawCube();

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 | GLUT\_DEPTH);

glutInitWindowSize(800, 600);

glutCreateWindow("SAHIL BHATTARAI -ORTHOGONAL PROJECTION");

glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // Set background color to white

glEnable(GL\_DEPTH\_TEST);

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}

OUTPUT:

A blue hexagon on a white background

Description automatically generated

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

#include <GL/glut.h>

GLfloat vertices[][2] = {

{50.0, 50.0},

{200.0, 50.0},

{200.0, 200.0},

{50.0, 200.0}

};

void init() {

glClearColor(1.0, 1.0, 1.0, 1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0, 300, 0, 300, -1, 1);

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 0.0, 1.0);

glBegin(GL\_POLYGON);

for (int i = 0; i < 4; i++) {glVertex2fv(vertices[i]);}

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(400, 400);

glutCreateWindow("SAHIL BHATTARAI -Clipping using Orthogonal Projection ");

glutDisplayFunc(display);

init();

glutMainLoop();

return 0;

}

OUTPUT:

A screenshot of a computer screen

Description automatically generated

20. Write an OPENGL app to show rectangle in two different viewports.

#include <GL/glut.h>

void displayViewport1() {

glViewport(0, 0, 400, 400);

glClearColor(0.0, 0.0, 0.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 0.0, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(-0.5, -0.5);

glVertex2f(0.5, -0.5);

glVertex2f(0.5, 0.5);

glVertex2f(-0.5, 0.5);

glEnd();

glFlush();

}

void displayViewport2() {

glViewport(400, 0, 400, 400);

glClearColor(0.0, 0.0, 0.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0, 0.0, 1.0);

glBegin(GL\_POLYGON);

glVertex2f(-0.5, -0.5);

glVertex2f(0.5, -0.5);

glVertex2f(0.5, 0.5);

glVertex2f(-0.5, 0.5);

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(1024, 768);

glutCreateWindow("SAHIL BHATTARAI -Two Viewports ");

glutDisplayFunc(displayViewport1);

glutDisplayFunc(displayViewport2);

glutMainLoop();

return 0;

}

OUTPUT:

A blue square on a black background

Description automatically generated

21. Implement Sutherland Cohen clipping algorithm to clip the given line the output must contain the unclipped and clipped view.

#include <GL/glut.h>

#include <iostream>

const int LEFT = 1;

const int RIGHT = 2;

const int BOTTOM = 4;

const int TOP = 8;

float x\_min = 100.0, y\_min = 100.0, x\_max = 300.0, y\_max = 300.0;

float x1, y1, x2, y2;

int computeCode(float x, float y) {

int code = 0;

if (x < x\_min)

code |= LEFT;

if (x > x\_max)

code |= RIGHT;

if (y < y\_min)

code |= BOTTOM;

if (y > y\_max)

code |= TOP;

return code;

}

void drawLine(float x1, float y1, float x2, float y2) {

glLineWidth(3);

glBegin(GL\_LINES);

glVertex2f(x1, y1);

glVertex2f(x2, y2);

glEnd();

}

void clipLine() {

int code1 = computeCode(x1, y1);

int code2 = computeCode(x2, y2);

bool accept = false;

while (true) {

if (!(code1 | code2)) {

// Both endpoints inside the window, accept the line

accept = true;

break;

} else if (code1 & code2) {

// Both endpoints outside in the same region, reject the line

break;

} else {

// Calculate intersection points

float x, y;

int codeOut = code1 ? code1 : code2;

if (codeOut & TOP) {

x = x1 + (x2 - x1) \* (y\_max - y1) / (y2 - y1);

y = y\_max;

} else if (codeOut & BOTTOM) {

x = x1 + (x2 - x1) \* (y\_min - y1) / (y2 - y1);

y = y\_min;

} else if (codeOut & RIGHT) {

y = y1 + (y2 - y1) \* (x\_max - x1) / (x2 - x1);

x = x\_max;

} else if (codeOut & LEFT) {

y = y1 + (y2 - y1) \* (x\_min - x1) / (x2 - x1);

x = x\_min;

}

if (codeOut == code1) {

x1 = x;

y1 = y;

code1 = computeCode(x1, y1);

} else {

x2 = x;

y2 = y;

code2 = computeCode(x2, y2);

}

}

}

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 0.0, 0.0); // Red color for the unclipped line

drawLine(x1, y1, x2, y2);

if (accept) {

glColor3f(0.0, 1.0, 0.0); // Green color for the clipped line

drawLine(x1, y1, x2, y2);

}

glFlush();

}

void display() {

glClearColor(1.0, 1.0, 1.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0, 0.0, 0.0); // Black color for the window boundary

glBegin(GL\_LINE\_LOOP);

glVertex2f(x\_min, y\_min);

glVertex2f(x\_max, y\_min);

glVertex2f(x\_max, y\_max);

glVertex2f(x\_min, y\_max);

glEnd();

clipLine();

glFlush();

}

void init() {

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 400.0, 0.0, 400.0);

}

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

std::cout << "Enter the coordinates of the line (x1 y1 x2 y2): ";

std::cin >> x1 >> y1 >> x2 >> y2;

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(400, 400);

glutCreateWindow("SAHIL BHATTARAI -Cohen-Sutherland Line Clipping ");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

OUTPUT:

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

17. Do the same thing 16 without using OpenGL transformation methods and see if the results are same.

#include <GL/glut.h>

#include <iostream>

using namespace std;

int TRANSLATE = 20;

int WIN\_WIDTH = 800;

int WIN\_HEIGHT = 400;

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glClearColor(1, 1, 1, 1);

int x = 100, y = 100, width=200, height=50;

// Original square

glColor3f(0, 0, 0);

glBegin(GL\_POLYGON);

glVertex2d(x, y);

glVertex2d(x + width, y);

glVertex2d(x + width, y + height);

glVertex2d(x, y + height);

glEnd();

// Translated square

glColor3f(1, 0, 0);

glBegin(GL\_POLYGON);

glVertex2d(x + TRANSLATE, y + TRANSLATE);

glVertex2d(x + width + TRANSLATE, y + TRANSLATE);

glVertex2d(x + width + TRANSLATE, y + height + TRANSLATE);

glVertex2d(x + TRANSLATE, y + height + TRANSLATE);

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB);

glutInitWindowSize(WIN\_WIDTH, WIN\_HEIGHT);

glutCreateWindow("(SAHIL BHATTARAI)-Translation without using OpenGL");

glClearColor(1, 1, 1, 1);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-WIN\_WIDTH, WIN\_WIDTH, -WIN\_HEIGHT, WIN\_HEIGHT);

glMatrixMode(GL\_MODELVIEW);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

OUTPUT:

A screenshot of a computer

Description automatically generated