Experiment 3

Drawing a Circle and an Ellipse [Done on OpenGL]

(# Take the value of radius, major axis and minor axis as input from the user.)

1. Draw the circle with the help of polar equations

**Code:**

#include <GL/glut.h>

#include <cmath>

#include <iostream>

float radius; // Variable to store user-input radius

// Function to draw the circle

void drawCircle() {

// Set the color to white

glColor3f(1.0, 1.0, 1.0);

// Start drawing the circle

glBegin(GL\_LINE\_LOOP); // Use GL\_LINE\_LOOP to draw a circular line

// Number of segments to approximate the circle

int numSegments = 100;

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

float theta = 2.0f \* 3.1415926f \* float(i) / float(numSegments); // Angle in radians

float x = radius \* cosf(theta); // X coordinate

float y = radius \* sinf(theta); // Y coordinate

glVertex2f(x, y); // Output vertex

}

glEnd();

glFlush(); // Ensure all OpenGL commands are executed

}

// Callback for rendering

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT); // Clear the screen

drawCircle(); // Draw the circle

}

// Initialization function for OpenGL settings

void init() {

// Set the background color to black

glClearColor(0.0, 0.0, 0.0, 1.0);

// Set the projection mode to 2D orthographic view

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-100, 100, -100, 100); // Set the coordinate system range

}

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

// Ask the user for the radius

std::cout << "Enter the radius of the circle: ";

std::cin >> radius;

// Initialize GLUT and create the window

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500); // Set window size

glutInitWindowPosition(100, 100); // Window position on screen

glutCreateWindow("Circle Using Polar Coordinates(Shalu Priya)"); // Window title

init(); // Initialize OpenGL settings

// Register the display callback function

glutDisplayFunc(display);

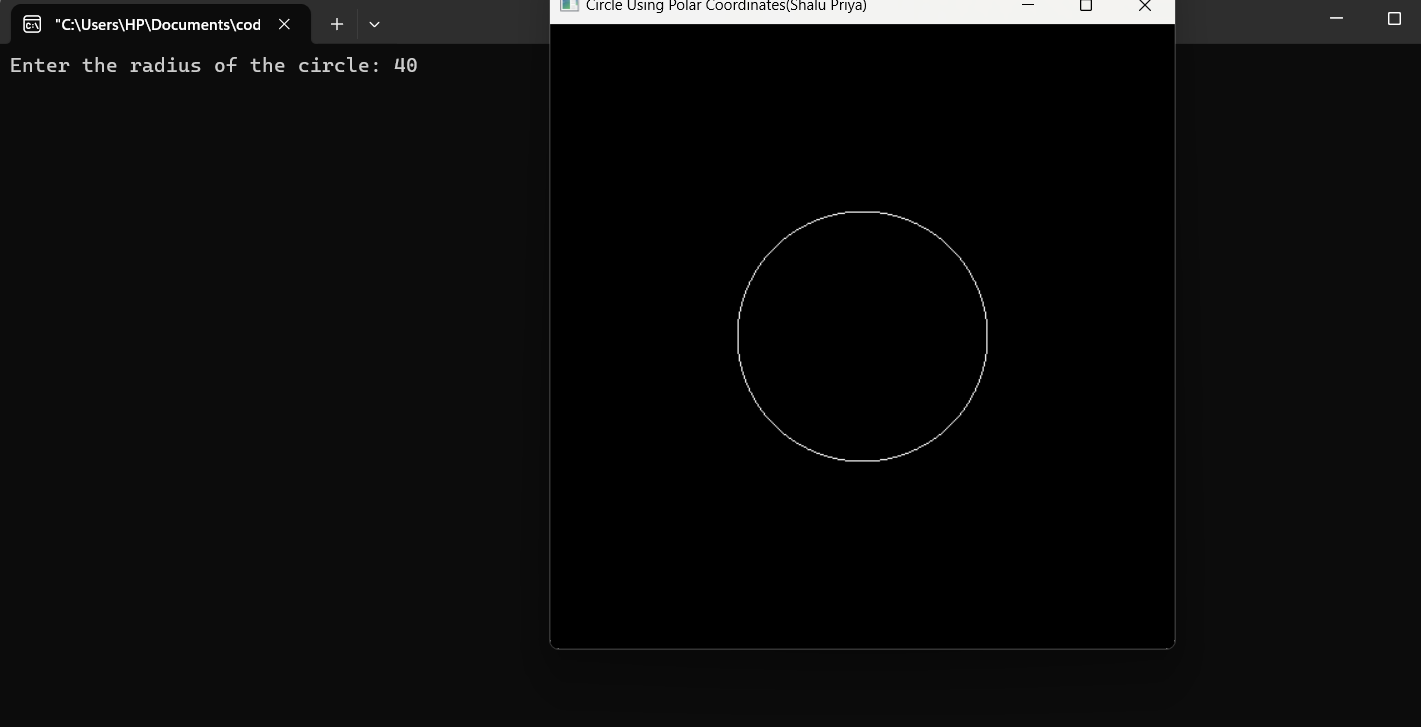
// Enter the GLUT main loop

glutMainLoop();

return 0;

}

**Output:**

****

1. Draw the circle with the help of mid-point method.

**Code :**

#include <GL/glut.h>

#include <iostream>

using namespace std;

int xc, yc, r; // Center and radius of the circle

void plot(int x, int y) {

glBegin(GL\_POINTS);

glVertex2i(xc + x, yc + y);

glVertex2i(xc - x, yc + y);

glVertex2i(xc + x, yc - y);

glVertex2i(xc - x, yc - y);

glVertex2i(xc + y, yc + x);

glVertex2i(xc - y, yc + x);

glVertex2i(xc + y, yc - x);

glVertex2i(xc - y, yc - x);

glEnd();

}

void midPointCircle() {

int x = 0, y = r;

int d = 1 - r;

plot(x, y);

while (x <= y) {

x++;

if (d < 0) {

d = d + 2 \* x + 1;

} else {

y--;

d = d + 2 \* (x - y) + 1;

}

plot(x, y);

}

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

midPointCircle();

glFlush();

}

void init() {

glClearColor(1.0, 1.0, 1.0, 1.0); // White background

glColor3f(0.0, 0.0, 0.0); // Black drawing color

glPointSize(3.0); // Set point size

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, 500, 0, 500); // Set the coordinate system

}

void getUserInput() {

cout << "Enter center of the circle (xc yc): ";

cin >> xc >> yc;

cout << "Enter radius of the circle: ";

cin >> r;

}

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

// Get user input

getUserInput();

// Initialize GLUT

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("Mid-Point Circle Drawing(shalu priya)");

// Call initialization routines

init();

// Register display callback

glutDisplayFunc(display);

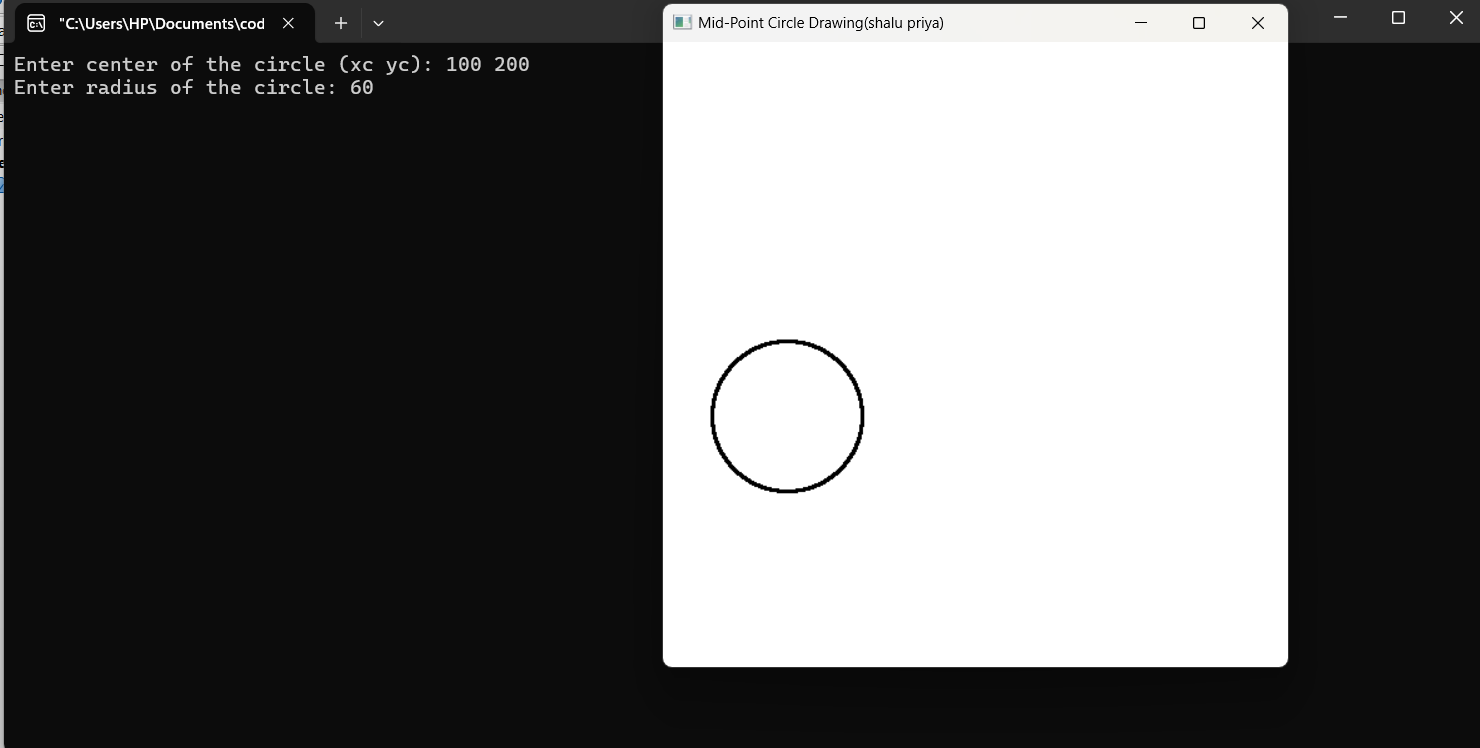
// Enter the GLUT main loop

glutMainLoop();

return 0;

}

**Output:**



1. Draw the Ellipse with the mid-point method.

**Code :**

#include <GL/glut.h>

#include <iostream>

using namespace std;

// Global variables for the center, radii, and axes of the ellipse

float xc, yc, rx, ry;

// Function to set a pixel

void setPixel(int x, int y) {

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

glFlush();

}

// Mid-point ellipse drawing algorithm

void drawEllipse(float xc, float yc, float rx, float ry) {

float x = 0;

float y = ry;

float p1 = ry \* ry - rx \* rx \* ry + (0.25 \* rx \* rx);

float dx = 2 \* ry \* ry \* x;

float dy = 2 \* rx \* rx \* y;

// Region 1

while (dx < dy) {

setPixel(xc + x, yc + y);

setPixel(xc - x, yc + y);

setPixel(xc + x, yc - y);

setPixel(xc - x, yc - y);

if (p1 < 0) {

x++;

dx = dx + (2 \* ry \* ry);

p1 = p1 + dx + (ry \* ry);

} else {

x++;

y--;

dx = dx + (2 \* ry \* ry);

dy = dy - (2 \* rx \* rx);

p1 = p1 + dx - dy + (ry \* ry);

}

}

// Region 2

float p2 = ((ry \* ry) \* ((x + 0.5) \* (x + 0.5))) + ((rx \* rx) \* ((y - 1) \* (y - 1))) - (rx \* rx \* ry \* ry);

while (y > 0) {

setPixel(xc + x, yc + y);

setPixel(xc - x, yc + y);

setPixel(xc + x, yc - y);

setPixel(xc - x, yc - y);

if (p2 > 0) {

y--;

dy = dy - (2 \* rx \* rx);

p2 = p2 + (rx \* rx) - dy;

} else {

y--;

x++;

dx = dx + (2 \* ry \* ry);

dy = dy - (2 \* rx \* rx);

p2 = p2 + dx - dy + (rx \* rx);

}

}

}

// Display function

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

drawEllipse(xc, yc, rx, ry);

}

// Initialize OpenGL

void init() {

glClearColor(1.0, 1.0, 1.0, 1.0); // Background color (white)

glColor3f(1.0, 0.0, 0.0); // Drawing color (red)

glPointSize(2.0); // Point size

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-500, 500, -500, 500); // Set the coordinate system

}

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

// Taking input from user

cout << "Enter the center of the ellipse (xc, yc): ";

cin >> xc >> yc;

cout << "Enter the radius of the major axis (rx): ";

cin >> rx;

cout << "Enter the radius of the minor axis (ry): ";

cin >> ry;

// Initialize OpenGL/Glut

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("Mid-Point Ellipse Drawing(Shalu priya)");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**Output:**

