ASSIGNMENT -3

COMPUTER GRAPHICS

* **Shubham Tiwari**

**101916126**

**3CS12**

1. Midpoint circle algorithm

#include <GL\glut.h>

#include <iostream>

#include <windows.h>

using namespace std;

int rx, ry;

int cx, cy;

int x, y, r;

void myInit()

{

glClearColor(0, 0.2, 0.2, .5);

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

glColor3f(1.0, 0.3, 0.9);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 400.0, 0.0, 400.0);

}

void draw\_pixel(int x, int y)

{

glBegin(GL\_POINTS);

glVertex2i(x+cx, y+cy);

glEnd();

}

void midPointCircle(int x0, int y0, int r)

{

int P = 1 - r;

int x = 0;

int y = r;

while (x < y)

{

draw\_pixel(x0 + x, y0 + y);

draw\_pixel(x0 + y, y0 + x);

draw\_pixel(x0 - y, y0 + x);

draw\_pixel(x0 - x, y0 + y);

draw\_pixel(x0 - x, y0 - y);

draw\_pixel(x0 - y, y0 - x);

draw\_pixel(x0 + y, y0 - x);

draw\_pixel(x0 + x, y0 - y);

if (P < 0)

P = P + 2 \* x + 1;

else

{

y--;

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

}

x++;

}

}

void myDisplay()

{

midPointCircle(x, y, r);

glFlush();

}

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

{

cout << "Enter the center(x,y) and Radius(r): ";

cin >> x >> y >> r;

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("Mid Point Circle Generating Algorithm");

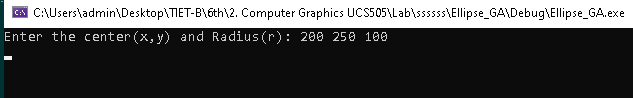
glutDisplayFunc(myDisplay);

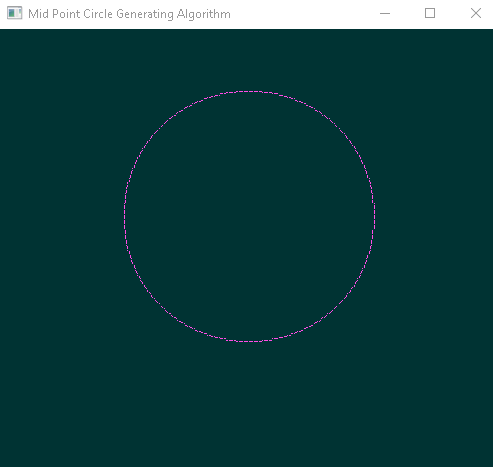
myInit();

glutMainLoop();

return 0;

}





2.Ellipse Generating Algorithm

#include <GL\glut.h>

#include <iostream>

#include <windows.h>

using namespace std;

int rx, ry;

int cx, cy;

void myInit()

{

glClearColor(0, 0.2, 0.2, .5);

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

glColor3f(1.0, 0.3, 0.9);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 400.0, 0.0, 400.0);

}

void draw\_pixel(int x, int y)

{

glBegin(GL\_POINTS);

glVertex2i(x+cx, y+cy);

glEnd();

}

void midptellipse(int rx, int ry, int xc, int yc)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

float dx, dy, d1, d2, x, y;

x = 0;

y = ry;

// Initial decision parameter of region 1

d1 = (ry \* ry) - (rx \* rx \* ry) +

(0.25 \* rx \* rx);

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

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

// For region 1

while (dx < dy)

{

// Print points based on 4-way symmetry

draw\_pixel(x, y);

draw\_pixel(-x, y);

draw\_pixel(x, -y);

draw\_pixel(-x, -y);

// Checking and updating value of

// decision parameter based on algorithm

if (d1 < 0)

{

x++;

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

d1 = d1 + dx + (ry \* ry);

}

else

{

x++;

y--;

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

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

d1 = d1 + dx - dy + (ry \* ry);

}

}

// Decision parameter of region 2

d2 = ((ry \* ry) \* ((x + 0.5) \* (x + 0.5))) +

((rx \* rx) \* ((y - 1) \* (y - 1))) -

(rx \* rx \* ry \* ry);

// Plotting points of region 2

while (y >= 0)

{

// Print points based on 4-way symmetry

draw\_pixel(x, y);

draw\_pixel(-x, y);

draw\_pixel(x, -y);

draw\_pixel(-x, -y);

// Checking and updating parameter

// value based on algorithm

if (d2 > 0)

{

y--;

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

d2 = d2 + (rx \* rx) - dy;

}

else

{

y--;

x++;

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

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

d2 = d2 + dx - dy + (rx \* rx);

}

}

}

void myDisplay()

{

midptellipse(rx, ry, cx, cy);

//midptellipse(100, 50, 300, 300);

glFlush();

}

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

{

cout << "Enter the Rx, Ry and Cx and Cy : ";

cin >> rx >> ry >> cx >> cy;

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutInitWindowPosition(0, 0);

glutCreateWindow("Ellipse Generating Algorithm");

glutDisplayFunc(myDisplay);

myInit();

glutMainLoop();

return 0;

}

