CSO351: Computer Graphics

Lab Assignment 2.Circle.(d):
Circle Generation using Midpoint

Objective:

Write a program in C/C++ for implementation of cirlce generation using midpoint algorithm.

Algorithm:

- **Step 1:** Get input of the coordinates and the radius of the circle.
- **Step 2:** Assign the starting point coordinates (x_0, y_0):

```
x0 = 0
y0 = 0
```

- **Step 3:** Calculate the value of the initial decision parameter P_0:

```
p0 = 5/4 - r
```

- **Step 4:** Based on the decision parameter, the next points are calculated using the following condition and repeat until x >= y:

```
if (pk < 0)
{
      xk+1 = xk + 1;
      yk+1 = yk;
      pk+1 = pk + 2*xk+1 + 1;
}
else
{
      xk+1 = xk + 1;
      yk+1 = yk - 1;
      pk+1 = pk + 2*(xk+1 - yk+1) + 1;
}
plot(xc + 1, yc + 1);//adding center coordinates
```

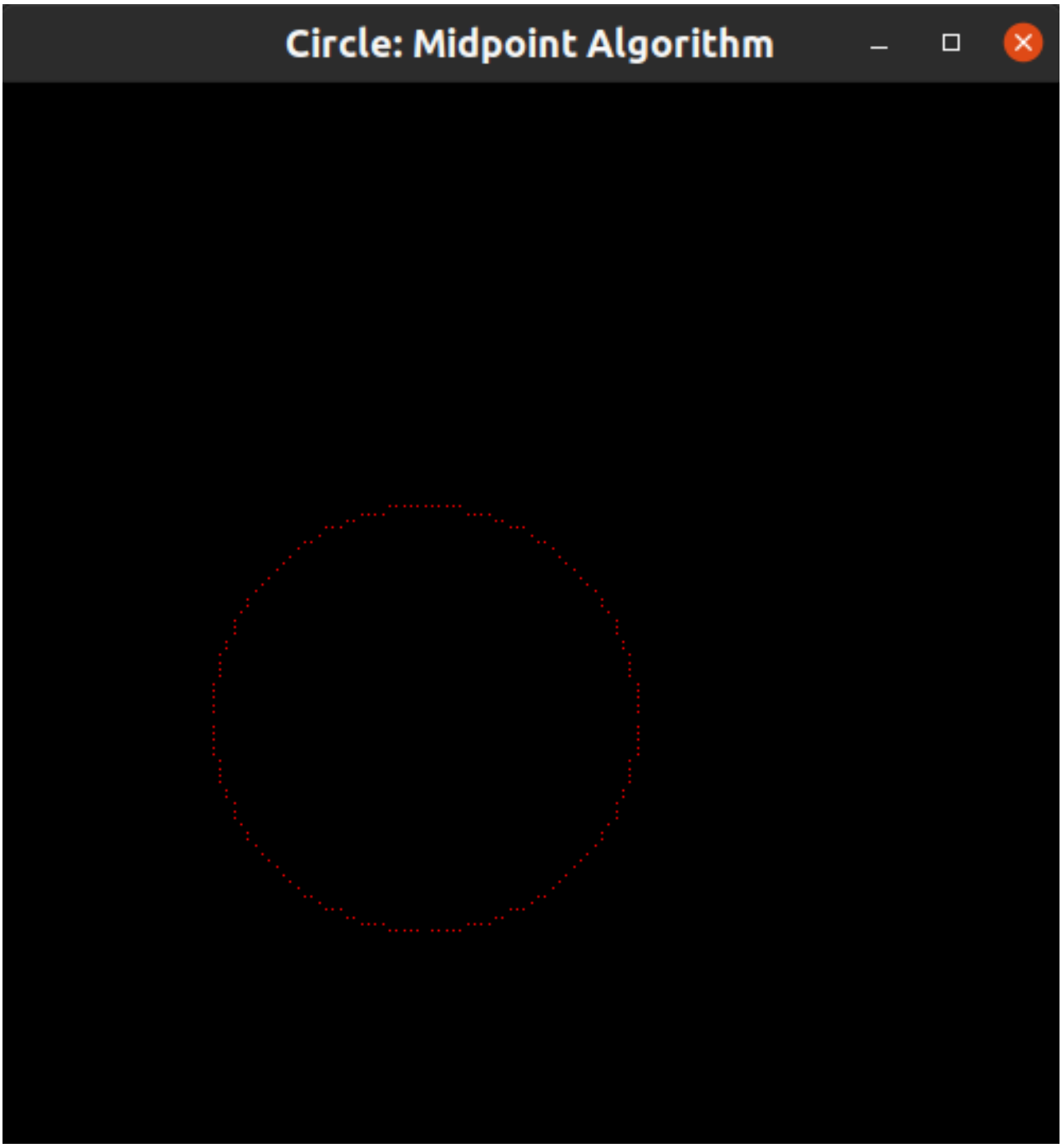
The obtained points will be of one octant. The remaining 7 octant points can be obtained using symmetry.

Result:

Input:

```
swaraj@shiv-raj-75:~/Documents/Assignments/Sem5/CG$ ./2.circle.d
Enter the center: 10 10
Enter radius : 30
```

Output:



Conclusion:

- Mid-Point Circle generation Algorithm is used to generate curves on raster displays and the algorithm is based on the simple equation of circle:

$$x^2 + y^2 = r^2$$

- The accuracy of the generating points is an issue in this algorithm.
- It is time-consuming and the circle generated is not smooth.

Appendix: Code

```
#include <stdio.h>
#include <iostream>
#include <GL/glut.h>

using namespace std;
int pntX1, pntY1, r;

void plot(int x, int y)
{
	glBegin(GL_POINTS);
	glVertex2i(x+pntX1, y+pntY1);
	glEnd();
}

void init(void)
{
	glClearColor(0.0, 0.0, 0.0, 0.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(-50, 100, -50, 100);
}

void midPointCircleAlgo()
{
	int x = 0;
	int y = r;
	float decision = 5/4 - r;
	plot(x, y);
	while (y > x)
	{
		if (decision < 0)
		{
			x++;
			decision += 2*x+1;
		}
		else
		{
			y--;
			x++;
```

```
                decision += 2*(x-y)+1;
            }
            plot(x, y);
            plot(x, -y);
            plot(-x, y);
            plot(-x, -y);
            plot(y, x);
            plot(-y, x);
            plot(y, -x);
            plot(-y, -x);
        }
}

void display(void)
{
        glClear (GL_COLOR_BUFFER_BIT);
        glColor3f (1.0, 0.0, 0.0);
        glPointSize(1.0);
        midPointCircleAlgo();
        glFlush ();
}

int main(int argc, char** argv)
{
        cout << "Enter the center: ";
        cin >> pntX1 >> pntY1;
        cout << "Enter radius : ";
        cin >> r;
        glutInit(&argc, argv);
        glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize (500, 500);
        glutInitWindowPosition (0, 0);
        glutCreateWindow ("Circle: Midpoint Algorithm");
        glutDisplayFunc(display);
        init ();
        glutMainLoop();
        return 0;
}
```