DEERWALK INSTITUTE OF TECHNOLOGY

Tribhuvan University

Faculties of Computer Science

Bachelors of Science in Computer Science and Information Technology (BSc. CSIT)

Course: Computer Graphics (CSC209)

Year/Semester: II/III

A Lab report on:

Implementation of Mid-Point Circle Algorithm

Submitted by:
Name: Arun Mainalil
Roll: 1307

Submitted to:
Binod Sitaula
Department of Computer Science

LAB 3

OBJECTIVE:

Implementation of Midpoint Circle Drawing Algorithm:

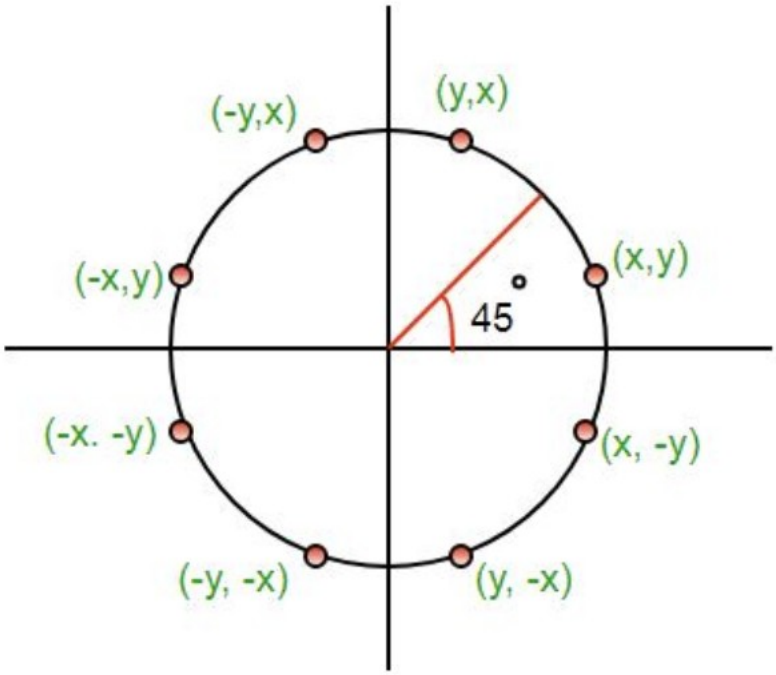
THEORY:

Midpoint Circle Algorithm:

It is based on the following function for testing the spatial relationship between the arbitrary point (x, y) and a circle of radius r centered at the origin:

$$f(x, y) = x^2 + y^2 - r^2 \qquad \begin{bmatrix} < 0 \text{ for } (x,y) \text{inside the circle} \\ = 0 \text{ for } (x,y) \text{on the circle} \\ > 0 \text{ for } (x,y) \text{outside the circle} \end{bmatrix} \text{.....equation 1}$$

Now,



consider the coordinates of the point halfway between pixel T and pixel S

This is called midpoint $(x_{i+1}, y_i - \frac{1}{2})$ and we use it to define a decision parameter:

$$P_i = f(x_{i+1}, y_i - \tfrac{1}{2}) = (x_{i+1})^2 + (y_i - \tfrac{1}{2})^2 - r^2 \text{ .....equation 2}$$

If P_i is -ve $\Longrightarrow$midpoint is inside the circle and we choose pixel T

If P_i is+ve $\Longrightarrow$midpoint is outside the circle (or on the circle)and we choose pixel S.

The decision parameter for the next step is:

$P_{i+1}=(x_{i+1}+1)^2+(y_{i+1}-\frac{1}{2})^2- r^2$.....equation 3

Since $x_{i+1}=x_{i+1}$, we have

$$P_{i+1} - P_i = ((x_i + 1) + 1)^2-(x_i + 1)^2+(y_{i+1} - \tfrac{1}{2})^2-(y_i - \tfrac{1}{2})^2$$

$$= x_i^2 + 4 + 4x_i - x_i^2 + 1 - 2x_i + y_{i+1}^2 + \tfrac{1}{4} - y_{i+1} - y_i^2 - \tfrac{1}{4} - y_i$$

$$= 2(xi+1) +1+ (y_{i+1}^2 - y_i^2)-(y_{i+1} - y_i)$$

$$P_{i+1} = P_i + 2(x_i + 1) + 1 + (y_{i+1}^2 - y_i^2) - (y_{i+1} - y_i)$$.....equation 4

If pixel T is choosen $\Longrightarrow P_i<0$

We have $y_{i+1}=y_i$

If pixel S is choosen $\Longrightarrow P_i\geq 0$

We have $y_{i+1}=y_i-1$

Thus, $$P_{i+1} = \begin{bmatrix} P_i + 2(x_i + 1) + 1, & \text{if } P_i < 0 \\ P_i + 2(x_i + 1) + 1 - 2(y_i - 1), & \text{if } P_i \geq 0 \end{bmatrix}$$.....equation 5

We can continue to simplify this in n terms of (x_i,y_i) and get

$$P_{i+1} = \begin{bmatrix} P_i + 2x_i + 3, & \text{if } P_i < 0 \\ P_i + 2(x_i - y_i) + 5, & \text{if } P_i \geq 0 \end{bmatrix}$$.....equation 6

Now, initial value of P_i (0,r)from equation 2

$$P_1 = (0 + 1)^2+(r - \tfrac{1}{2})^2-r^2$$

$$= 1 + \tfrac{1}{4} - r^2 = \tfrac{5}{4} - r$$

We can put $\frac{5}{4} \cong 1$
$\therefore$r is an integer
So, P_1=1-r

ALGORITHM:

Step1: Put x =0, y =r in equation 2. We have p=1-r

Step2: Repeat steps while x ≤ y
Plot (x, y)
If (p<0), then set p = p + 2x + 3

Else
p = p + 2(x-y)+5
y =y - 1 (end if)
x =x+1 (end loop)

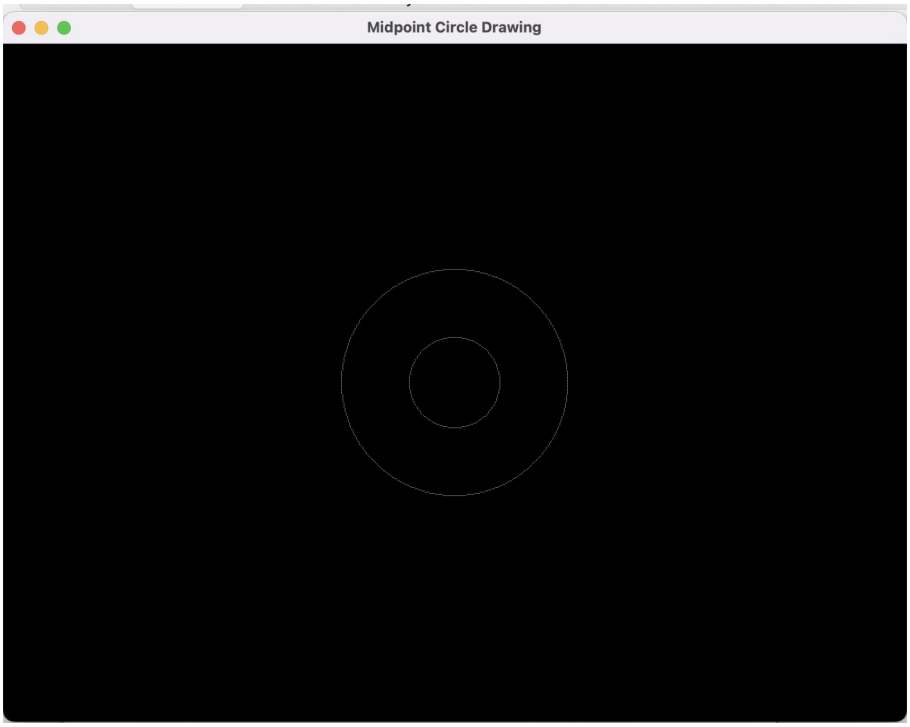
Step3: End

PROGRAM CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <GLFW/glfw3.h>

const int WIDTH = 800, HEIGHT = 600;

void putpixels(GLFWwindow* window, float x, float y) {
   float x_ndc = (2.0f * x) / WIDTH - 1.0f;
   float y_ndc = 1.0f - (2.0f * y) / HEIGHT;

   glBegin(GL_POINTS);
   glVertex2f(x_ndc, y_ndc);
   glEnd();
}

void midPointCircleDraw(GLFWwindow* window, int xc, int yc, int r) {
   int x = r, y = 0;
   int P = 1 - r;

   while (x >= y) {
      putpixels(window, xc + x, yc + y);
      putpixels(window, xc - x, yc + y);
      putpixels(window, xc + x, yc - y);
      putpixels(window, xc - x, yc - y);
      putpixels(window, xc + y, yc + x);
      putpixels(window, xc - y, yc + x);
      putpixels(window, xc + y, yc - x);
      putpixels(window, xc - y, yc - x);
```

```
        y++;
        if (P <= 0)
            P = P + 2*y + 1;
        else {
            x--;
            P = P + 2*y - 2*x + 1;
        }
    }
}

int main() {
    if (!glfwInit()) {
        printf("GLFW initialization failed!\n");
        return -1;
    }

    GLFWwindow* window = glfwCreateWindow(WIDTH, HEIGHT, "Midpoint Circle Drawing",
NULL, NULL);
    if (!window) {
        printf("Window creation failed!\n");
        glfwTerminate();
        return -1;
    }
    glfwMakeContextCurrent(window);

    glViewport(0, 0, WIDTH, HEIGHT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-1, 1, -1, 1, -1, 1);
    glMatrixMode(GL_MODELVIEW);

    while (!glfwWindowShouldClose(window)) {
        glClear(GL_COLOR_BUFFER_BIT);
        glLoadIdentity();

        midPointCircleDraw(window, WIDTH / 2, HEIGHT / 2, 100);
        midPointCircleDraw(window, WIDTH / 2, HEIGHT / 2, 40);

        glfwSwapBuffers(window);
        glfwPollEvents();
    }

    glfwDestroyWindow(window);
    glfwTerminate();
    return 0;
}
```

SAMPLE OUTPUT:



CONCLUSION:

From this lab, we were able to draw a circle using Midpoint Circle Drawing Algorithm and display it using C graphic functions.