**DEERWALK INSTITUTE OF TECHNOLOGY**

**Tribhuvan University**

**Faculties of Computer Science**

**A logo of a sea creature

Description automatically generated**

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

**Course: Computer Graphics (CSC214)**

**Year/Semester: II/III**

**A Lab report on:**

**Implementation of Midpoint Circle Algorithm**

Submitted by:

Name: Parth Poudyal

Roll: 1317

Submitted to:

Binod Sitaula

Department of Computer Science

**Theory**

For a circle, it can be represented using the equation:

Where,

r is the radius of the given circle, and (x, y) be the points on the circle respectively.

Now, to understand:

if for values of (x, y)

: the point lies beyond the outlines the circle

: the point lies inside the outlines of the circle.

While progressing through the circle from point ( to the next point on the circle points can be either

To determine the next point in the sequence of points, we take the midpoint of the two points given above:

Fitting the given midpoint in the given equation of the circle we get:

if be the point next on the circle. The point after that be :

to determine the next point on the sequence, we find the difference between and to get :

For different values of

In the equation of the circle with midpoint:

if

**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**

#include <iostream>

#include <graphics.h>

using namespace std;

// Function to draw all eight octants of the circle

void drawCircle(int xc, int yc, int x, int y)

{

putpixel(xc + x, yc + y, WHITE);

putpixel(xc - x, yc + y, WHITE);

putpixel(xc + x, yc - y, WHITE);

putpixel(xc - x, yc - y, WHITE);

putpixel(xc + y, yc + x, WHITE);

putpixel(xc - y, yc + x, WHITE);

putpixel(xc + y, yc - x, WHITE);

putpixel(xc - y, yc - x, WHITE);

}

void bresenhamCircle(int xc, int yc, int r)

{

int x = 0, y = r;

int d = 3 - 2 \* r;

drawCircle(xc, yc, x, y);

while (y >= x)

{

x++;

if (d > 0)

{

y--;

d = d + 2 \* (x - y) + 5;

}

else

{

d = d + 2 \* x + 3;

}

drawCircle(xc, yc, x, y);

}

}

int main()

{

int xc, yc, r;

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

cin >> xc >> yc;

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

cin >> r;

int gd = DETECT, gm;

initgraph(&gd, &gm, (char\*)"");

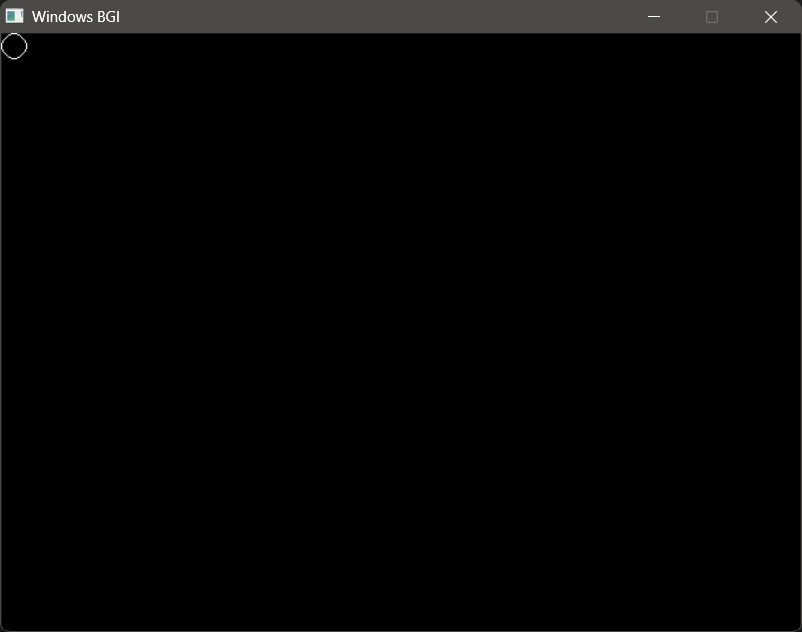
bresenhamCircle(xc, yc, r);

getch();

closegraph();

return 0;

}

**OUTPUT **

**Conclusion**

From this project, I go to understand the application of the Brasenham’s midpoint circle drawing algorithm.