**SSN College of Engineering**

**Department of Computer Science and Engineering**

## UCS1712 – GRAPHICS AND MULTIMEDIA LAB

**EX NO: 4 – Midpoint Circle Drawing Algorithm**

SATHEESH KUMAR G R

185001136

**1]AIM:**

To write a C++ program using OpenGL to implement a Midpoint Circle drawing algorithm with radius and a centre given as user input.

**ALGORITHM:**

1. Here we are dividing the circle into 8 octants.

2. And find the points for one octant and by modifying the values get the points for the remaining octants.

3. Creating a function that plots points for all of the octants given x and y coordinate for the first octant.

4. Find p = 5/4 - 4

5. Run a while loop till x == y

a. Increment x value

b. If p is less than 0

i. Update the p value as p += 2 \* x + 1

ii. Else decrement y value and update p value as p += 2 \* (x - y) + 1

c. Plot the octants.

**CODE:**

#include<GL/glut.h>

#include<iostream>

using namespace std;

int R, X, Y;

void pointplot(int x, int y)

{

    glBegin(GL\_POINTS);

    glColor3f(1.0f, 1.0f, 1.0f);

    glVertex2i(x+X, y+Y);

    glPointSize(15);

    glEnd();

}

void display()

{

    glClear(GL\_COLOR\_BUFFER\_BIT);

    int x=0, y=R;

    int p=5/4-R;

    while(x<y)

    {

        x++;

        if(p<=0)

        {

            p += 2\*(x+1) + 1;

        }

        else

        {

            y--;

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

        }

        pointplot(x, y);

        pointplot(x, -y);

        pointplot(-x, y);

        pointplot(-x, -y);

        pointplot(y, x);

        pointplot(y, -x);

        pointplot(-y, x);

        pointplot(-y, -x);

    }

    glFlush();

}

int main(int argc, char \*\* argv)

{

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowPosition(700, 0);

    glutInitWindowSize(500, 500);

    glutCreateWindow("EX4-Circle-Midpt");

    glClearColor(0,0,0,1);

    glMatrixMode(GL\_PROJECTION);

    gluOrtho2D(-250, 250, -250, 250);

    printf("Enter radius: \t");

    scanf("%d",&R);

    printf("\nEnter center (X,Y):\n");

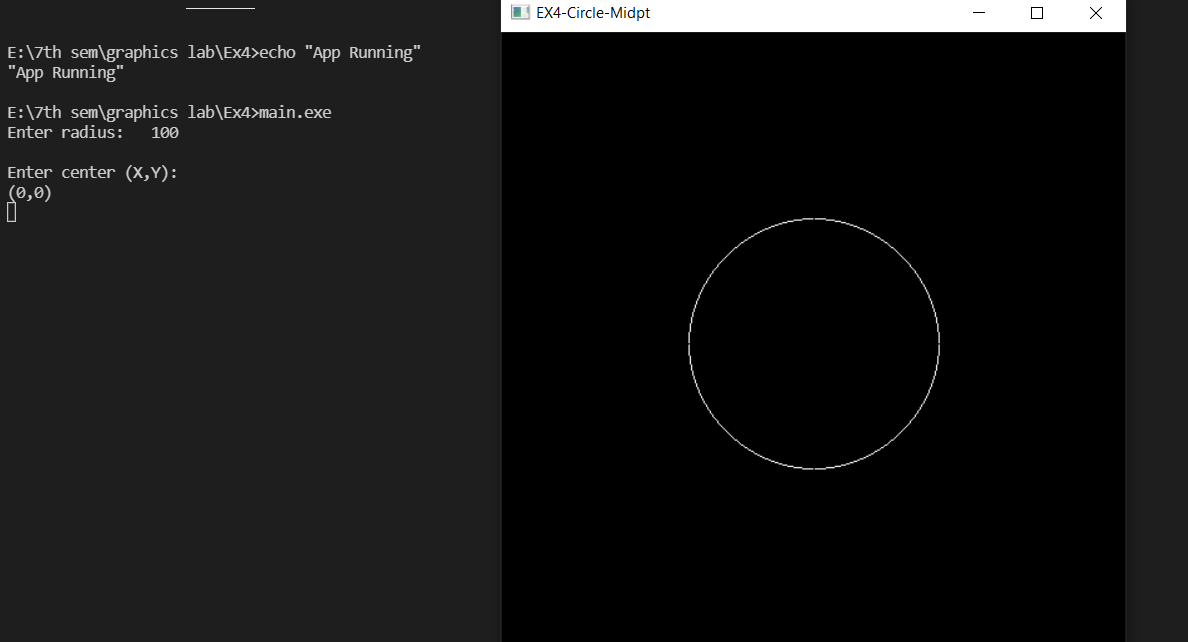
    scanf(" (%d,%d)",&X,&Y);

    glutDisplayFunc(display);

    glutMainLoop();

}

OUTPUT:



**RESULT:**

Thus a circle is drawn using Midpoint Circle Drawing Algorithm.

**2]AIM:**

To write a C++ program using OPENGL to replicate any circular object with the help of the Midpoint Circle algorithm.

**ALGORITHM:**

1. Here we are dividing the circle into 8 octants.

2. And find the points for one octant and by modifying the values get the points for the remaining octants.

3. Creating a function that plots points for all of the octants given x and y coordinate for the first octant.

4. Find p = 5/4 - 4

5. Run a while loop till x == y

a. Increment x value

b. If p is less than 0

i. Update the p value as p += 2 \* x + 1

ii. Else decrement y value and update p value as p += 2 \* (x - y) + 1

c. Plot the octants.

6. We draw two sets of concurrent circles using the Midpoint Circle Drawing Algorithm mentioned above.

7. Draw 8 lines from center to circumference using GL\_LINES.

**CODE:**

#include<windows.h>

#include<gl/glut.h>

#include<cstdlib>

#include<iostream>

using namespace std;

void myInit()

{

    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(0.0, 800.0, 0.0, 600.0);

}

void putPixel(int x, int y)

{

    glBegin(GL\_POINTS);

    glVertex2i(x, y);

    glEnd();

}

void plotOctants(int X1, int Y1, int x, int y)

{

    putPixel(X1 + x, Y1 + y);

    putPixel(X1 + x, Y1 - y);

    putPixel(X1 - x, Y1 + y);

    putPixel(X1 - x, Y1 - y);

    putPixel(X1 + y, Y1 + x);

    putPixel(X1 - y, Y1 + x);

    putPixel(X1 + y, Y1 - x);

    putPixel(X1 - y, Y1 - x);

}

void midPointCircle(int X1, int Y1, int r)

{

    int x = 0;

    int y = r;

    float p = 5 / 4 - r;

    plotOctants(X1, Y1, x, y);

    while (y > x)

    {

        x++;

        if (p < 0)

        {

            p += 2 \* x + 1;

        }

        else

        {

            y--;

            p += 2 \* (x - y) + 1;

        }

        plotOctants(X1, Y1, x, y);

    }

}

void display(void)

{

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glPointSize(2);

    int r = 160;

    int temp = 0;

    glColor3f(1.0, 0.0, 0.0);

    while (temp <= r)

    {

        midPointCircle(400, 300, temp);

        temp++;

    }

    glBegin(GL\_LINES);

    glColor3f(1.0, 1.0, 1.0);

    glVertex2f(370,140);

    glVertex2f(370,460);

    glEnd();

    glBegin(GL\_LINES);

    glColor3f(1.0, 1.0, 1.0);

    glVertex2f(390,140);

    glVertex2f(390,460);

    glEnd();

    glBegin(GL\_LINES);

    glColor3f(1.0, 1.0, 1.0);

    glVertex2f(410,140);

    glVertex2f(410,460);

    glEnd();

    glBegin(GL\_LINES);

    glColor3f(1.0, 1.0, 1.0);

    glVertex2f(430,140);

    glVertex2f(430,460);

    glEnd();

    glFlush();

}

int main(int argc, char\*\* argv)

{

    glutInit(&argc, argv);                 // Initialize GLUT

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(800, 600);   // Set the window's initial width & height

    glutInitWindowPosition(50, 50); // Position the window's initial top-left corner

    glutCreateWindow("Ball"); // Create a window with the given title

    myInit();

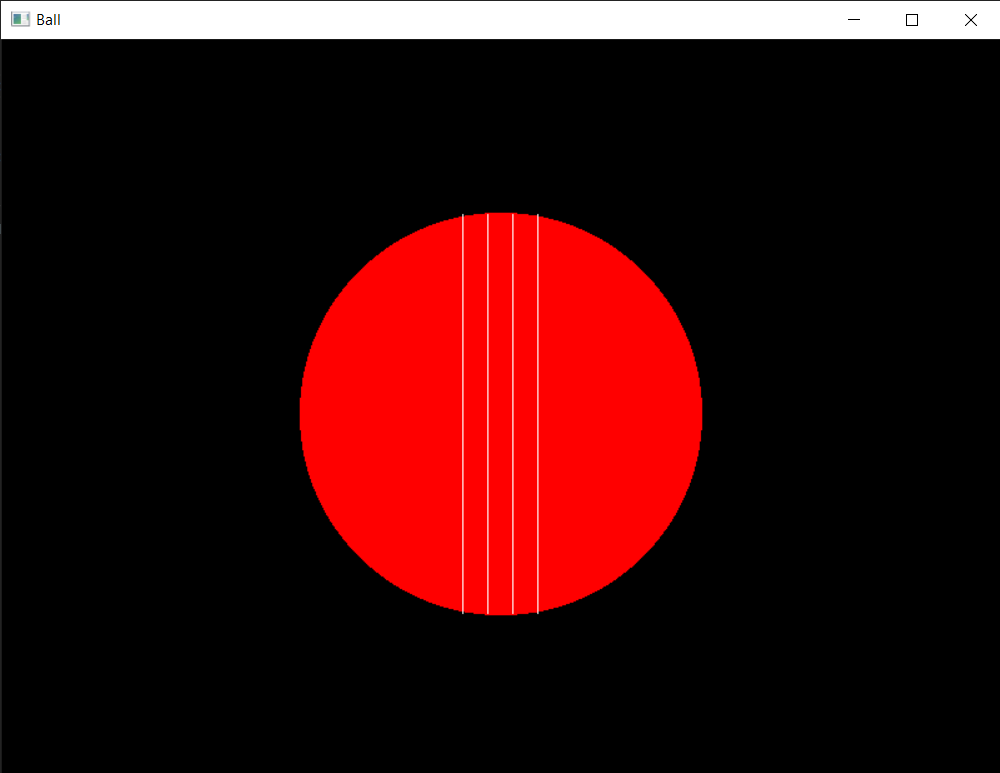
    glutDisplayFunc(display); // Register display callback handler for window re-paint

    glutMainLoop();           // Enter the infinitely event-processing loop

    return 0;

}

**OUTPUT:**

****

**RESULT:**

Thus, a wheel structure is drawn with the use of Midpoint Circle Drawing Algorithm.