SSN COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1712 – GRAPHICS AND MULTIMEDIA LAB

Name : Rahul Ram M
Reg .No : 185001121
Date : 01/08/2021

EX NO: 4

Midpoint Circle Drawing Algorithm

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

Aim:

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 ocatants

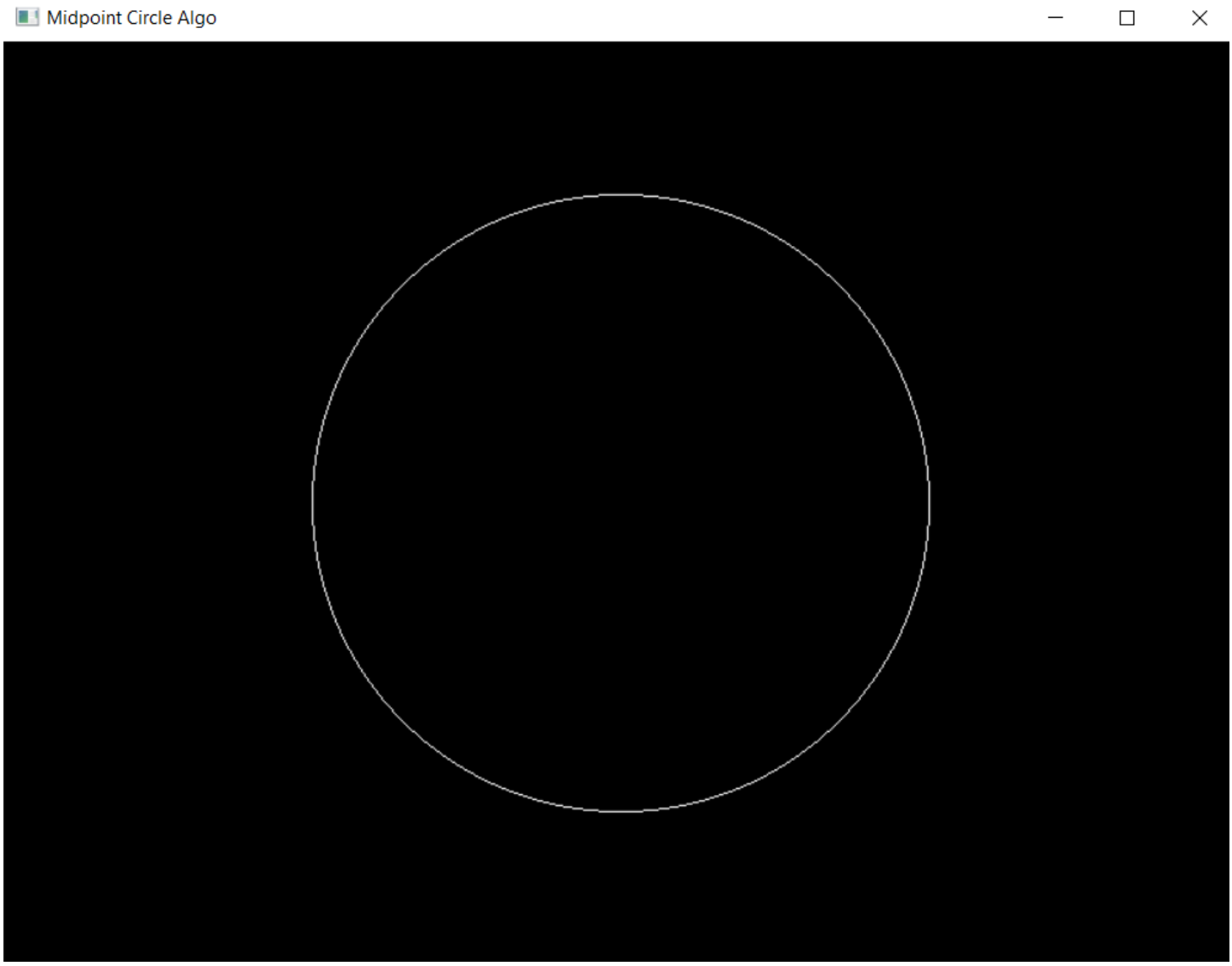
Code:

```
#include<windows.h>
#include<gl/glut.h>
```

```
#include<cstdlib>
#include<iostream>
using namespace std;

int X1, Y1, r;

void myInit()
{
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glPointSize(1);
    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 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 x = 0;
    int y = r;
    float p = 5 / 4 - r;
    plotOctants(x, y);

    while (y > x)
    {
```

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

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    midPointCircle();
    glFlush();
}

int main(int argc, char** argv)
{
    cout << "X coordinate: "; cin >> X1;
    cout << "Y coordinate: "; cin >> Y1;
    cout << "Radius: "; cin >> r;

    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("Midpoint Circle Algo"); // 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;
}

// 300 300 100
```

Output Screenshot:



Point - (400, 300) Radius - 200

Result:

A circle is drawn successfully with the help of the midpoint circle algorithm.

2. Write a C++ program using OPENGL to replicate any circular object with the help of the Midpoint Circle algorithm. Use the necessary colours and elements to show details.

Aim:

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

Algorithm:

1. Drawing button using midpoint circle algorithm.
2. We fill the color for a shape (eg: a circle filled with color) by drawing multiple circles with radius 0 to the given radius and by keeping the size of the point by 2 to prevent any gap in the shapes.
3. Outer part and inner part of the button is drawn by drawing multiple circles with one color from radius 0 to 160 and other color from radius 160 to 200.
4. Four while circles are created by choosing a different midpoint and radius as 50.

Code:

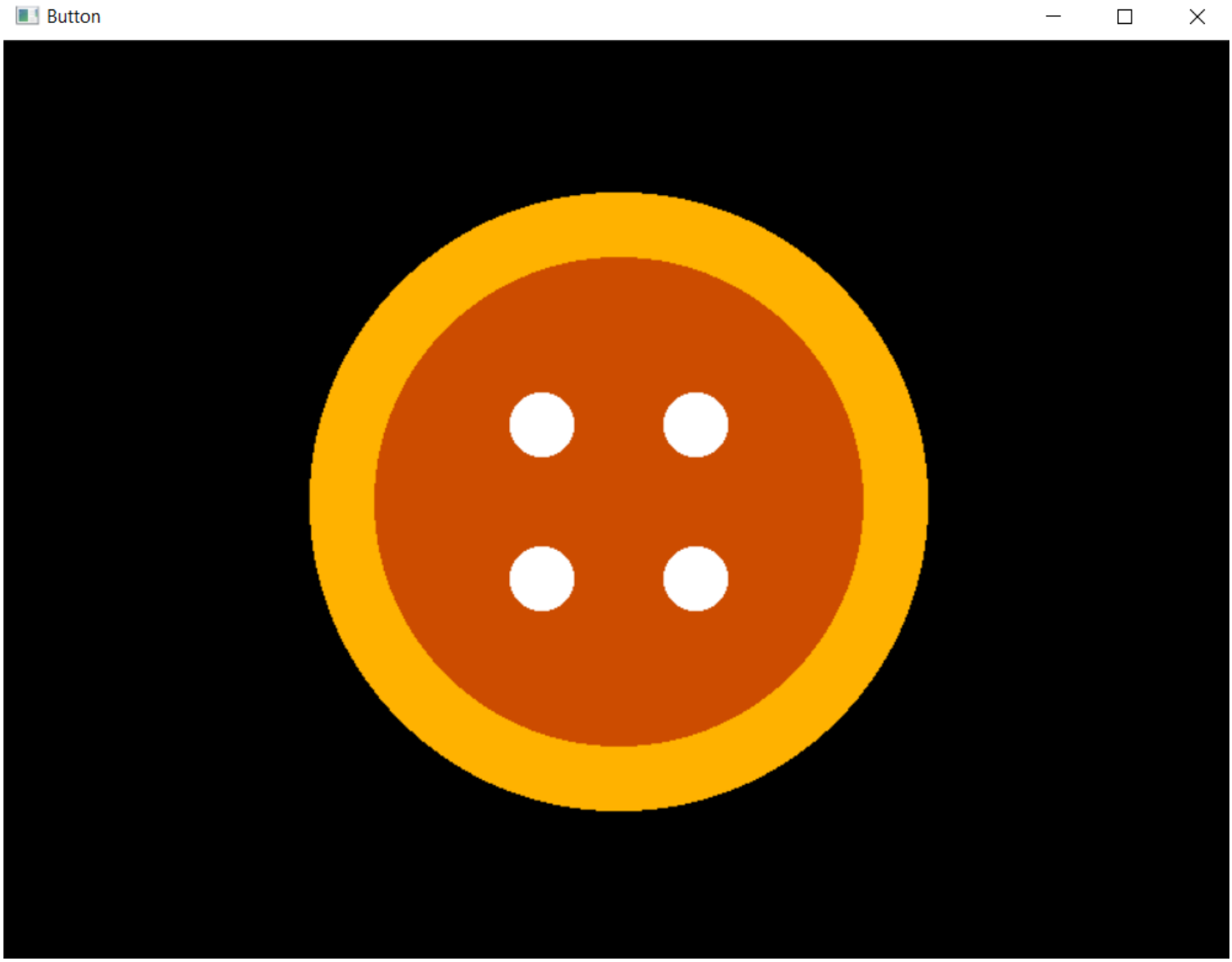
```
#include<windows.h>
#include<gl/glut.h>
#include<cstdlib>
#include<iostream>
using namespace std;


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(0.8, 0.3, 0.0);
    while (temp <= r)
```

```
{
    midPointCircle(400, 300, temp);
    temp++;
}

temp = 160;
r = 200;
glColor3f(1.0, 0.7, 0.0);
while (temp <= r)
{
    midPointCircle(400, 300, temp);
    temp++;
}

temp = 0;
r = 20;
glColor3f(1.0, 1.0, 1.0);
while (temp <= r)
{
    midPointCircle(450, 350, temp);
    temp++;
}

temp = 0;
r = 20;
glColor3f(1.0, 1.0, 1.0);
while (temp <= r)
{
    midPointCircle(450, 250, temp);
    temp++;
}

temp = 0;
r = 20;
glColor3f(1.0, 1.0, 1.0);
while (temp <= r)
{
    midPointCircle(350, 350, temp);
    temp++;
}

temp = 0;
r = 20;
```

```
    glColor3f(1.0, 1.0, 1.0);
    while (temp <= r)
    {
        midPointCircle(350, 250, temp);
        temp++;
    }

    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("Button"); // 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 Screenshot:



Result:

A button shape is drawn successfully using the midpoint circle algorithm.