UCS1712 - GRAPHICS AND MULTIMEDIA LAB

EX - 4: Midpoint Circle Drawing Algorithm

NAME: KEERTHANA T
REGISTER NUMBER: 185001074
CLASS-SEC: CSE-B
DATE: 12/08/2021

AIM :

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

Practice question:
2. Write a C++ program using OpenGL to replicate any circular object with the help of the Midpoint Circle algorithm. Use the necessary colors and elements to show details

ALGORITHM :

- Start
- Import GL library as a header file
- Create a function void myInit()
- Provide the default conditions
- Create a function void plot(x,y)
- Plot the points using GL_POINTS
- Create a function void vada()
- Apply the midpoint circle algorithm and use plot() function to plot the determined points
- Create function midPointCircleAlgo()

- Call the vada function with the radius and the necessary customization
- Create the main function
- Get the center point and the radius from the user
- Give basic details for the output window in the main function
- Apply the midpoint algorithm on the user input by calling the myDisplay function.
- End

CODE :

1.

```
#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 myInit(void)
{
	glClear(GL_COLOR_BUFFER_BIT);
	glClearColor(1.0, 1.0, 1.0, 0.0);
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(-200, 200, -200, 200);
}
```

```
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 myDisplay(void)
{
```

```
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(0.0, 0.0, 0.0);
	glPointSize(1.0);

	glColor3f(1.0,0.5,0.0);
	glPolygonMode(GL_FRONT, GL_FILL);
	midPointCircleAlgo();

	glFlush();
}

void main(int argc, char** argv)
{
	cout << "Enter the coordinates of the center:\n\n" << endl;

	cout << "X-coordinate   : "; cin >> pntX1;
	cout << "\nY-coordinate : "; cin >> pntY1;
	cout << "\nEnter radius : "; cin >> r;


	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(400, 400);
	glutInitWindowPosition(0, 0);
	glutCreateWindow("Sambhar vadai");
	glutDisplayFunc(myDisplay);
	myInit();
	glutMainLoop();

}
```

2.

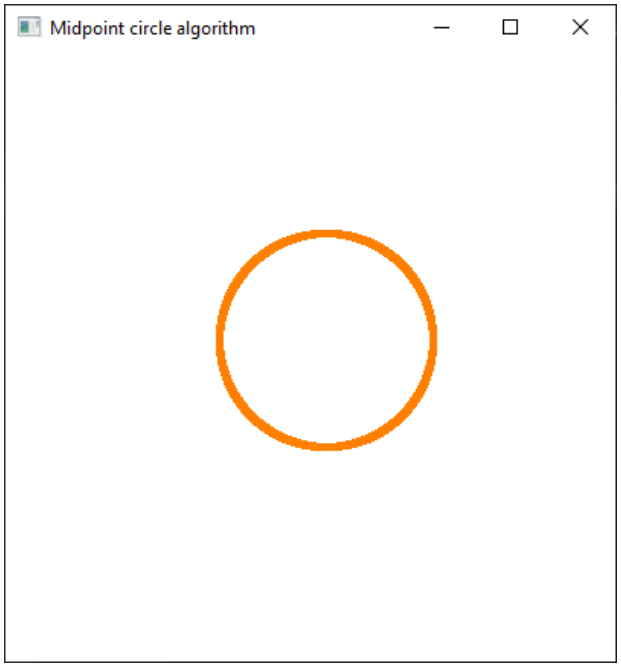
```
#include <stdio.h>
#include <iostream>
#include <GL/glut.h>
using namespace std;

int pntX1, pntY1, r;

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


void myInit(void)
{
	glClear(GL_COLOR_BUFFER_BIT);
	glClearColor(1.0, 1.0, 1.0, 0.0);
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(-200, 200, -200, 200);
}

void vada(int r)
{
	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 midPointCircleAlgo()
{
        glPointSize(18.0);
        glColor3f(0.9, 0.5, 0.1);
        vada(r);
        glPointSize(8.0);
        glColor3f(1, 0, 1);
        glPolygonMode(GL_FRONT, GL_FILL);
        vada(r - 10);
}

void myDisplay(void)
{
```

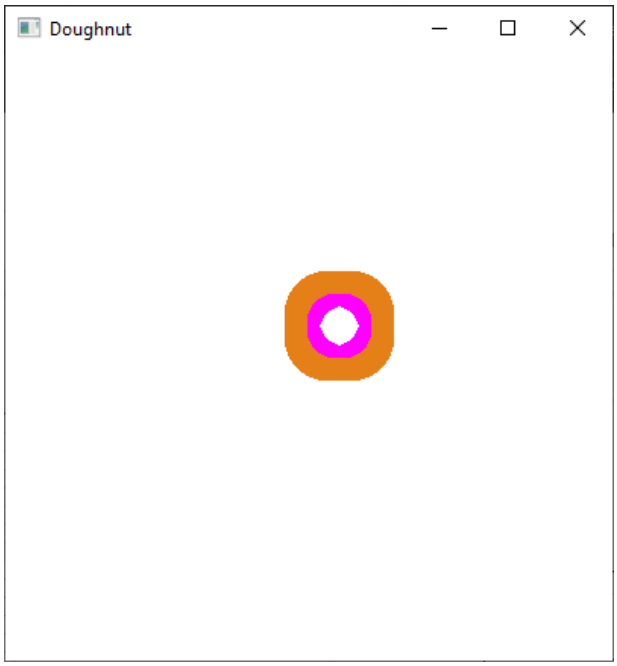
```
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(0.0, 0.0, 0.0);

	midPointCircleAlgo();

	glFlush();
}

void main(int argc, char** argv)
{
	cout << "Enter the coordinates of the center:\n\n" << endl;

	cout << "X-coordinate   : "; cin >> pntX1;
	cout << "\nY-coordinate : "; cin >> pntY1;
	cout << "\nEnter radius : "; cin >> r;


	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(400, 400);
	glutInitWindowPosition(0, 0);
	glutCreateWindow("Doughnut");
	glutDisplayFunc(myDisplay);
	myInit();
	glutMainLoop();

}
```

OUTPUT:

1.



2.



RESULT:

Thus we have successfully implemented the midpoint circle algorithm to draw circles in OpenGL.