Ex. No. 4 **Midpoint Circle Drawing Algorithm** **Karthik D**

September 18, 2022 **UCS1712 - Graphics and Multimedia Lab** **195001047**

Aim

To develop a C++ program using the OpenGL framework to implement the Midpoint Circle Drawing algorithm, and demonstrate all its output cases.

Question

A. Plot points that make up the circle with center (x_c, y_c) and radius r using the midpoint circle drawing algorithm. Demonstrate the following cases:

Case 1:	With center (0,0)
Case 2:	With center (xc,yc)

B. Draw any object using line and circle drawing algorithms.

Note that both four cases of circle drawing must be given as test cases.

Midpoint Circle Drawing Algorithm

```
Procedure plotCircleMidpoint(xc, yc, r :integer);

    var
    x, y, p: integer;

    Begin
        x := 0;
        y := r;
        p := 1 - r;
        setPixel (x, y, 1);

        While x >= y
        Begin
            x:= x+1;
            If p < 0 then
            Begin
                setPixel (xc + x, yc + y, 1);
                setPixel (xc - x, yc + y, 1);
                setPixel (xc + x, yc - y, 1);
                setPixel (xc - x, yc - y, 1);
                setPixel (xc + y, yc + x, 1);
                setPixel (xc - y, yc + x, 1);
                setPixel (xc + y, yc - x, 1);
                setPixel (xc - y, yc - x, 1);
                p := p + 2*(x+1) + 1
            Else
                y := y - 1;
                setPixel (xc + x, yc + y, 1);
                setPixel (xc - x, yc + y, 1);
                setPixel (xc + x, yc - y, 1);
                setPixel (xc - x, yc - y, 1);
                setPixel (xc + y, yc + x, 1);
                setPixel (xc - y, yc + x, 1);
                setPixel (xc + y, yc - x, 1);
                setPixel (xc - y, yc - x, 1);
                p := p + 2*(x+1) + 1 - 2*(y-1)
            End
        End

    End {lotCircleMidpoint}
```

Implementation using C++ Program Code

1. main.cpp - Driver and Handler to render the circle using the midpoint algorithm for given center coordinates and radius length

   Function *plotCircleMidpoint()* implements the midpoint circle algorithm

```cpp
#include <GL/glut.h>
#include <stdio.h>


void renderSpacedBitmapString(float x, float y, void *font, char
*string) {
   char *c;
   int x1 = x;
   for (c = string; *c != '\0'; c++) {
       glRasterPos2f(x1, y);
       glutBitmapCharacter(font, *c);
       x1 = x1 + glutBitmapWidth(font, *c);
   }
}
```

```
void markString(char *string, int x, int y, int x_offset, int y_offset)
{
   glColor3f(255.0, 0, 0.0); // red color
   renderSpacedBitmapString(x+x_offset, y+y_offset,
GLUT_BITMAP_HELVETICA_12, string);
   glFlush();
}


void plotDivisionLines()    {
   glBegin(GL_LINES);
   glVertex2d(-320, 0);
   glVertex2d(320, 0);
   glVertex2d(0, -240);
   glVertex2d(0, 240);
   glEnd();
}


void plotPoint(int x, int y, int x_offset, int y_offset)    {
   glBegin(GL_POINTS);
   glVertex2d(x + x_offset, y + y_offset);
   glEnd();
}


void plotLine(int start_x, int start_y, int end_x, int end_y)   {
   glBegin(GL_LINES);
   glVertex2d(start_x, start_y);
   glVertex2d(end_x, end_y);
   glEnd();
}


void plotAtAllOctants(int x, int y, int x_offset, int y_offset) {
   plotPoint(x, y, x_offset, y_offset);
```

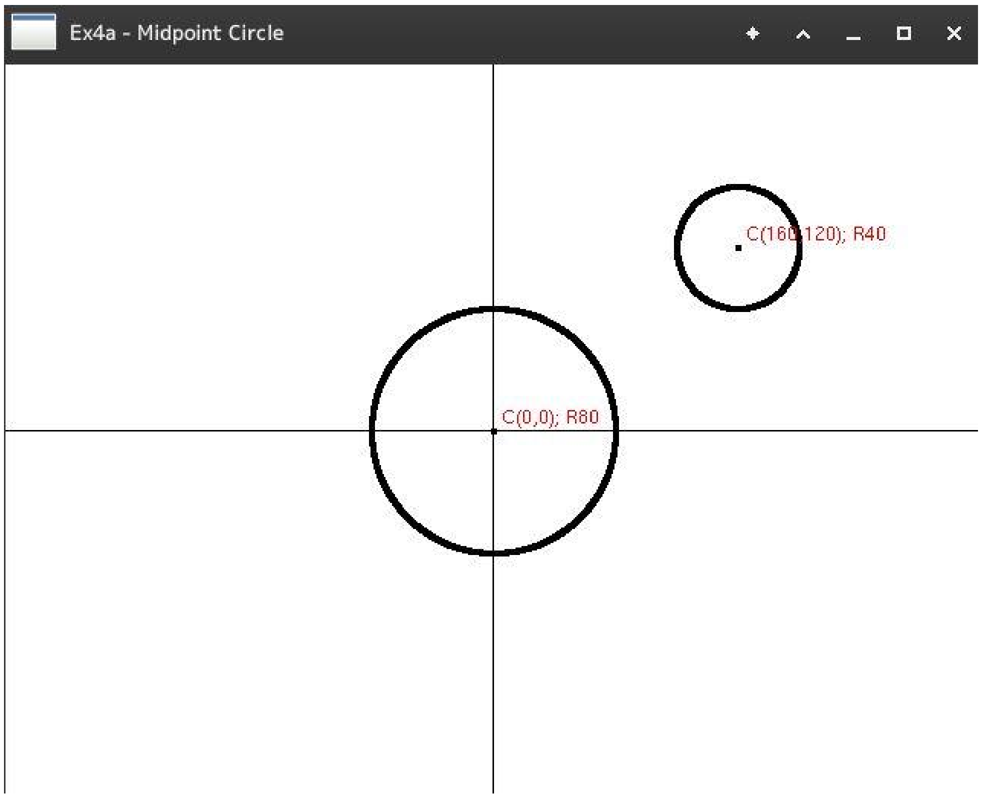
```
    plotPoint(y, x, x_offset, y_offset);
    plotPoint(x, -y, x_offset, y_offset);
    plotPoint(y, -x, x_offset, y_offset);
    plotPoint(-x, y, x_offset, y_offset);
    plotPoint(-y, x, x_offset, y_offset);
    plotPoint(-x, -y, x_offset, y_offset);
    plotPoint(-y, -x, x_offset, y_offset);
}


void plotCircle(int center_x, int center_y, int radius) {
    int x_k = 0;
    int y_k = radius;
    int p_k = 1 - radius;

    plotPoint(center_x, center_y, 0, 0);
    plotAtAllOctants(x_k, y_k, center_x, center_y);
    while(x_k <= y_k)   {
        if(p_k < 0) {
            plotAtAllOctants(x_k + 1, y_k, center_x, center_y);
            x_k += 1;
            p_k += (2*x_k) + 1;
        }
        else{
            plotAtAllOctants(x_k + 1, y_k - 1, center_x, center_y);
            x_k += 1;
            y_k += -1;
            p_k += (2*x_k) - (2*y_k) + 1;
        }
    }
}


void display_figure()   {
    glClear(GL_COLOR_BUFFER_BIT);
    plotCircle(0, 110, 30);
    plotCircle(0, 20, 60);
    // right-limb 1
```

```
   plotLine(60, 40, 110, 20);
   plotLine(110, 20, 140, 40);
   // left-limb 1
   plotLine(-60, 40, -110, 20);
   plotLine(-110, 20, -140, 40);
   // right-limb 2
   plotLine(60, 0, 110, -20);
   plotLine(110, -20, 140, 0);
   // left-limb 2
   plotLine(-60, 0, -110, -20);
   plotLine(-110, -20, -140, 0);
   // right-limb 3
   plotLine(40, -30, 90, -50);
   plotLine(90, -50, 120, -35);
   // left-limb 3
   plotLine(-40, -30, -90, -50);
   plotLine(-90, -50, -120, -35);
   // right-antenna
   plotLine(15, 140, 30, 180);
   // left-antenna
   plotLine(-15, 140, -30, 180);
   glFlush();
}


void display_circle()   {
   glClear(GL_COLOR_BUFFER_BIT);
   plotDivisionLines();

   plotCircle(0, 0, 80);
   markString("C(0,0); R80", 0, 0, 5, 5);

   glColor3f(0.0f, 0.0f, 0.0f);
   plotCircle(160, 120, 40);
   markString("C(160,120); R40", 160, 120, 5, 5);
   glFlush();
}
```

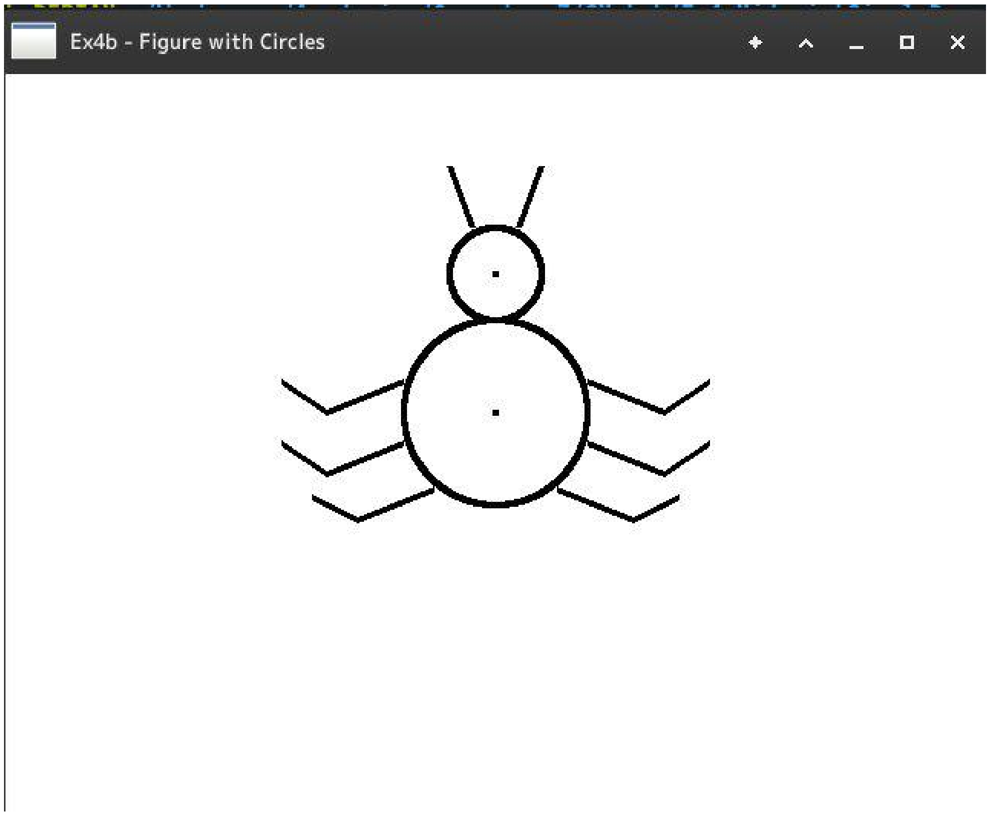
```c
void init() {
   glClearColor(1.0, 1.0, 1.0, 0.0);
   glColor3f(0.0f, 0.0f, 0.0f);
   glPointSize(4);
   glLineWidth(4);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(-320.0, 320.0, -240.0, 240.0);
}


int main(int argc, char **argv)  {
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize(640, 480);

   // circle
   // glutCreateWindow("Ex4a - Midpoint Circle");
   // glutDisplayFunc(display_circle);

   // figure
   glutCreateWindow("Ex4b - Figure with Circles");
   glutDisplayFunc(display_figure);

   init();
   glutMainLoop();
   return 1;
}
```

Sample Output

- **Both the cases for circle drawing**



- **Figure using circles and lines**



Learning Outcomes

Through this implementation of Midpoint Circle Drawing algorithm using the OpenGL framework and C++ programming language, the following concepts were learnt:

1. The working of the midpoint circle drawing algorithm.
2. General understanding of the OpenGL framework and its APIs.