EX-04 - Midpoint Circle Drawing Algorithm in C++ using OpenGL

09/08/2021

Venkataraman Nagarajan, CSE - C
18500192

AIM

To implement Midpoint circle drawing algorithm in C++.

SPECIFICATION

1. To plot points that make up the circle with center (x_c, y_c) and radius r using Midpoint circle drawing algorithm. Give atleast 2 test cases.
    - **Case 1:** With center $(0, 0)$
    - **Case 2:** With center (x_c, y_c)
2. To draw any object using line and circle drawing algorithms.

PROGRAM - 01

Draw circles for both the cases

```
1  // To plot points that make up the circle with center (xc,yc) and radius r↩
       using Midpoint circle drawing
2  // algorithm. Give atleast 2 test cases.
3
4  //     Case 1: With center (0,0)
5  //     Case 2: With center (xc,yc)
6
7  #include<bits/stdc++.h>
8  #include<GL/glut.h>
9
10 using namespace std;
11 using ld = long double;
12 using ll = long long;
13
14 const int WINDOW_WIDTH = 850;
15 const int WINDOW_HEIGHT = 850;
16
17 void myInit();
18 void myDisplay();
19
20 void printCircles();
21 void printMidpointCircle(ll x, ll y, ll r);
22
23 const ld PADDING = 0;
24 const ld STEP = 1;
25 const ld SCALE = 1;
26
27 int main(int argc,char* argv[]) {
28     glutInit(&argc,argv);
29     glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
30     glutInitWindowSize(WINDOW_WIDTH,WINDOW_HEIGHT);
31     glutCreateWindow("Midpoint Circle Drawing Algorithm");
32     glutDisplayFunc(myDisplay);
33     myInit();
34     glutMainLoop();
35     return 1;
36 }
37
38 void myInit() {
39     glClearColor(1.0,1.0,1.0,0.0);
40     glColor3f(0.0f,0.0f,0.0f);
41     glPointSize(2.0);
42     glMatrixMode(GL_PROJECTION);
```

```
43      glLoadIdentity();
44      gluOrtho2D(-300.0, 300.0, -300.0, 300.0);
45  }
46
47  void myDisplay() {
48      glClear(GL_COLOR_BUFFER_BIT);
49
50      printCircles();
51
52      glFlush();
53  }
54
55
56  void printCircles() {
57      glBegin(GL_POINTS);
58
59      // Case 1: With center (0,0)
60      glColor3f(1.0f,0.0f,0.0f);
61      printMidpointCircle(0, 0, 50);
62
63      // Case 2: With center (xc,yc)
64      glColor3f(0.5f,0.5f,0.0f);
65      printMidpointCircle(175, 25, 100);
66
67      glEnd();
68  }
69
70  void printMidpointCircle(ll x0, ll y0, ll r) {
71
72      ld pad = PADDING, scale = SCALE;
73
74      x0 = x0*scale + pad;
75      y0 = y0*scale + pad;
76      r  = r*scale + pad;
77
78      int x = 0, y = r;
79      int p = 1 - r; // Decision parameter
80
81      //Plot the centre
82      glVertex2d(x0 , y0 );
83
84      //Draw the circle
85      while (x < y) {
86          // Plotting symmetrically in all 8 octants
87          glVertex2d(x0 + x, y0 + y);
88          glVertex2d(x0 + y, y0 + x);
89          glVertex2d(x0 - x, y0 + y);
```

```
90          glVertex2d(x0 - y, y0 + x);
91          glVertex2d(x0 - x, y0 - y);
92          glVertex2d(x0 - y, y0 - x);
93          glVertex2d(x0 + x, y0 - y);
94          glVertex2d(x0 + y, y0 - x);
95
96          if (p < 0) {
97              x += 1;
98              p += 2 * x + 1;
99          }
100         else {
101             x += 1;
102             y -= 1;
103             p = p + 2 * x + 1 - 2 * y;
104         }
105     }
106
107 }
```

SAMPLE I/0

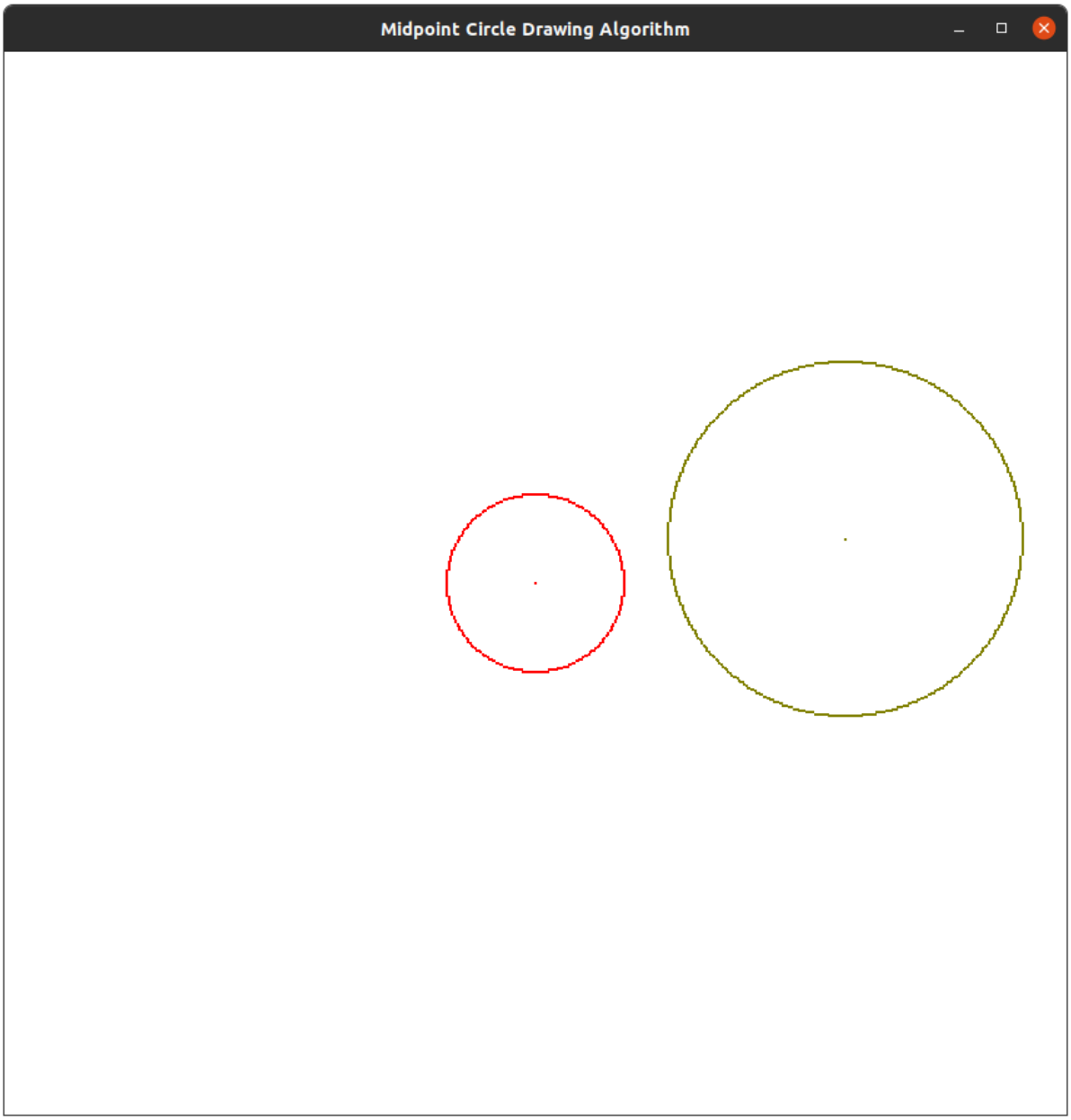

Figure 1: Each color represents different cases for Midpoint circle drawing algorithm

PROGRAM - 02

Drawing an object with lines and circles

```
// To draw any object using line and circle drawing algorithms.

#include<bits/stdc++.h>
#include<GL/glut.h>

using namespace std;
using ld = long double;
using ll = long long;

const int WINDOW_WIDTH = 850;
const int WINDOW_HEIGHT = 850;

void myInit();
void myDisplay();

void drawOmnitrix();
void printDDALine(ld x1, ld y1, ld x2, ld y2);
void printMidpointCircle(ll x, ll y, ll r);

ld radian(ll x);

const ld PADDING = 0;
const ld STEP = 1;
const ld SCALE = 1;
const ld PI = 3.141592653589793238;

int main(int argc,char* argv[]) {
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(WINDOW_WIDTH,WINDOW_HEIGHT);
    glutCreateWindow("Omnitrix Scribble");
    glutDisplayFunc(myDisplay);
    myInit();
    glutMainLoop();
    return 1;
}

void myInit() {
    glClearColor(1.0,1.0,1.0,0.0);
    glColor3f(0.0f,0.0f,0.0f);
    glPointSize(2.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
```

```
    gluOrtho2D(-300.0, 300.0, -300.0, 300.0);
}

void myDisplay() {
    glClear(GL_COLOR_BUFFER_BIT);

    drawOmnitrix();

    glFlush();
}

ld radian(ll x) {
    return x * PI / 180.0;
}

void drawOmnitrix() {
    //Draw circle frame
    glColor3f(0.184f, 0.310f, 0.310f);
    for(ll radius = 150; radius >= 135; radius --)
        printMidpointCircle(0, 0, radius);

    //Print Omnitrix "X" Frame
    glColor3f(0.000f, 1.000f, 0.000f);
    for(ll point = 30; point >= 20; point -- ) {
        ll x = point;
        ll y = point+65;
        printDDALine(x, 0, y,y);
        printDDALine(x, 0, y, -y);


        printDDALine(-x, 0, -y, y);
        printDDALine(-x, 0, -y, -y);
    }

    //Print Omnitrix Handle Frame
    glColor3f(0.294f, 0.000f, 0.510f);
    for(ld angle = 145; angle >= 35; angle-=0.1) {
        ll x = 150*cos(radian(angle));
        ll y = 150*sin(radian(angle));

        printDDALine(x, y, x, y + rand()%10+100);
        printDDALine(x, -y, x, -(y + rand()%10+100));
    }
 }

void printDDALine(ld x1, ld y1, ld x2, ld y2) {

```

```
91      glBegin(GL_POINTS);
92
93      ld pad = PADDING, scale = SCALE;
94
95      x1 = x1*scale + pad;
96      x2 = x2*scale + pad;
97      y1 = y1*scale + pad;
98      y2 = y2*scale + pad;
99
100     ld dx, dy, steps;
101     ld xInc, yInc, x, y;
102
103     dx = (x2-x1);
104     dy = (y2-y1);
105
106     if(abs(dx) > abs(dy))   steps = abs(dx);
107     else                    steps = abs(dy);
108
109     xInc = dx/steps;
110     yInc = dy/steps;
111
112     x = x1; y = y1;
113     glVertex2d(x, y);
114
115     for(long i=1;i<=steps;i++) {
116         x += xInc;
117         y += yInc;
118
119         glVertex2d(x, y);
120     }
121
122     glEnd();
123 }
124
125
126 void printMidpointCircle(ll x0, ll y0, ll r) {
127
128     glBegin(GL_POINTS);
129
130     ld pad = PADDING, scale = SCALE;
131
132     x0 = x0*scale + pad;
133     y0 = y0*scale + pad;
134     r  = r*scale + pad;
135
136     int x = 0, y = r;
137     int p = 1 - r; // Decision parameter
```

```
138
139     //Plot the centre
140     glVertex2d(x0 , y0 );
141
142     //Draw the circle
143     while (x < y) {
144         // Plotting symmetrically in all 8 octants
145         glVertex2d(x0 + x, y0 + y);
146         glVertex2d(x0 + y, y0 + x);
147         glVertex2d(x0 - x, y0 + y);
148         glVertex2d(x0 - y, y0 + x);
149         glVertex2d(x0 - x, y0 - y);
150         glVertex2d(x0 - y, y0 - x);
151         glVertex2d(x0 + x, y0 - y);
152         glVertex2d(x0 + y, y0 - x);
153
154         if (p < 0) {
155             x += 1;
156             p += 2 * x + 1;
157         }
158         else {
159             x += 1;
160             y -= 1;
161             p = p + 2 * x + 1 - 2 * y;
162         }
163     }
164
165     glEnd();
166
167 }
```

SAMPLE I/0

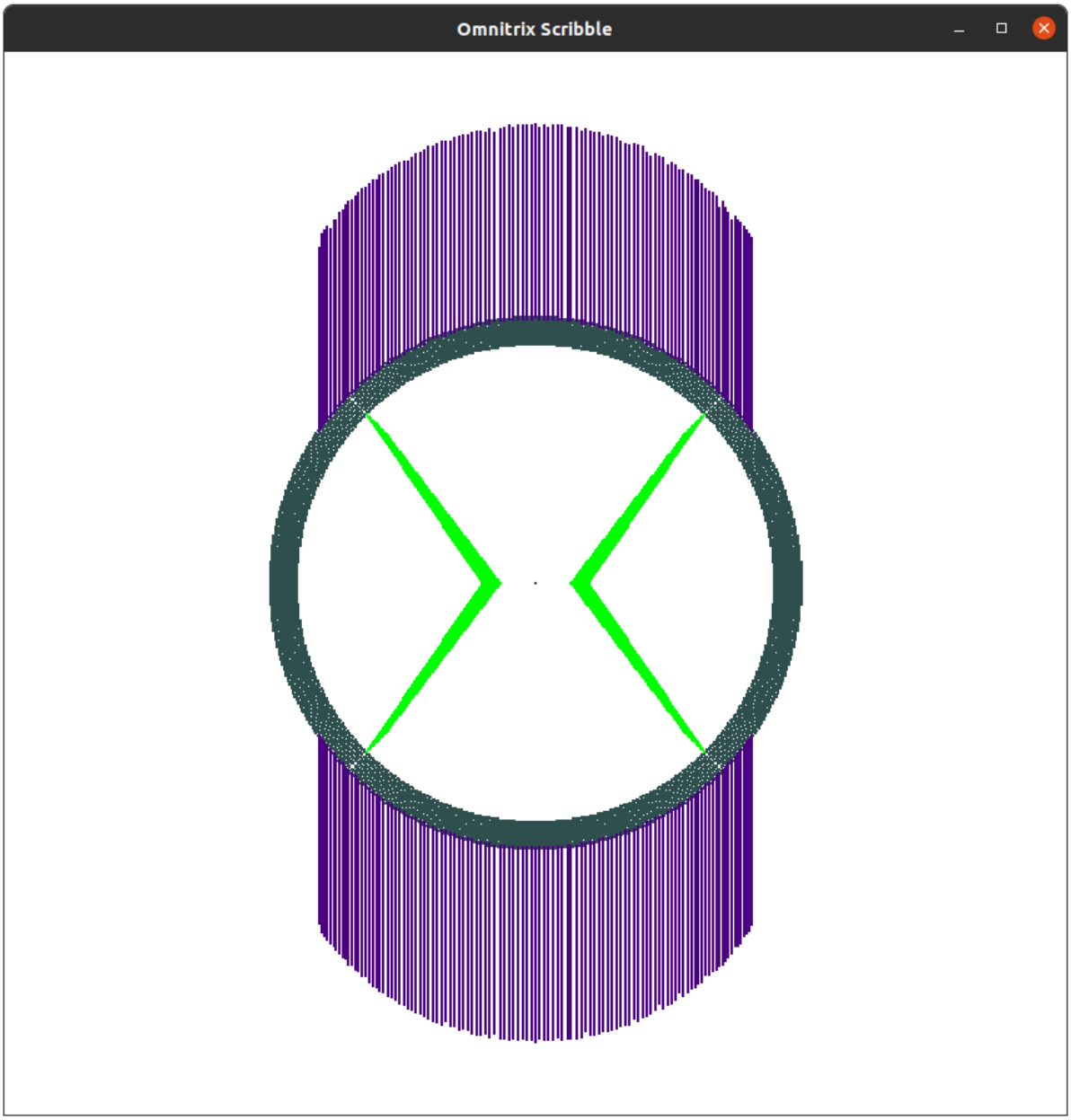

Figure 2: A simple omnitrix using lines and circles

RESULT

The code for Midpoint circle drawing algorithm is written and output is verified.
