

EX-02 - DDA LINE DRAWING ALGORITHM IN C++ USING OPENGL

26/07/2021

Venkataraman Nagarajan, CSE - C
18500192

AIM

To implement DDA Line drawing algorithm in C++.

SPECIFICATION

To plot points that make up the line with endpoints (x_0, y_0) and (x_n, y_n) using DDA line drawing algorithm.

- **Case 1:** $+ve$ slope Left to Right line
- **Case 2:** $+ve$ slope Right to Left line
- **Case 3:** $-ve$ slope Left to Right line
- **Case 4:** $-ve$ slope Right to Left line

Each case has two subdivisions

(i) $|m| \leq 1$

(ii) $|m| > 1$

Note that all four cases of line drawing must be given as test cases.

PROGRAM - 01

Draw lines for all eight cases

```
1 // Q: To plot points that make up the line with endpoints (x0,y0) and (xn,↵
   yn) using DDA line drawing algorithm.
2
3 // Case 1: +ve slope Left to Right line
4 // Case 2: +ve slope Right to Left line
5 // Case 3: -ve slope Left to Right line
6 // Case 4: -ve slope Right to Left line
7
8 // Each case has two subdivisions
9 // (i) |m| <= 1
10 // (ii) |m| > 1
11 // Note that all four cases of line drawing must be given as test ↵
   cases.
12
13 #include<bits/stdc++.h>
14 #include<GL/glut.h>
15
16 using namespace std;
17 using ld = long double;
18
19 const int WINDOW_WIDTH = 850;
20 const int WINDOW_HEIGHT = 700;
21
22 void myInit();
23 void myDisplay();
24
25 void printLines();
26 void printDDALine(ld x1, ld y1, ld x2, ld y2);
27
28 const ld PADDING = 250;
29 const ld STEP = 1;
30 const ld SCALE = 5;
31
32 int main(int argc, char* argv[]) {
33     glutInit(&argc, argv);
34     glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
35     glutInitWindowSize(WINDOW_WIDTH, WINDOW_HEIGHT);
36     glutCreateWindow("DDA - Line Drawing Algorithm");
37     glutDisplayFunc(myDisplay);
38     myInit();
39     glutMainLoop();
40     return 1;
41 }
```

```

42
43 void myInit() {
44     glClearColor(1.0,1.0,1.0,0.0);
45     glColor3f(0.0f,0.0f,0.0f);
46     glPointSize(2.0);
47     glMatrixMode(GL_PROJECTION);
48     glLoadIdentity();
49     gluOrtho2D(0.0,640.0,0.0,480.0);
50 }
51
52 void myDisplay() {
53     glClear(GL_COLOR_BUFFER_BIT);
54
55     printLines();
56
57     glFlush();
58 }
59
60
61 void printLines() {
62     glBegin(GL_POINTS);
63
64     // Case 1: +ve slope Left to Right line
65     glColor3f(1.0f,0.0f,0.0f);
66     // | m | > 1
67     printDDALine((ld)3,(ld)2, (ld)15,(ld)10);
68     // | m | < 1
69     printDDALine((ld)2,(ld)3, (ld)10,(ld)15);
70
71     // Case 2: +ve slope Right to Left line
72     glColor3f(0.5f,0.5f,0.0f);
73     // | m | > 1
74     printDDALine((ld)-3,(ld)-2, (ld)-15,(ld)-10);
75     // | m | < 1
76     printDDALine((ld)-2,(ld)-3, (ld)-10,(ld)-15);
77
78     //Case 3: -ve slope Left to Right line
79     glColor3f(0.0f,1.0f,0.0f);
80     // | m | > 1
81     printDDALine((ld)3,(ld)-2, (ld)15,(ld)-10);
82     // | m | < 1
83     printDDALine((ld)2,(ld)-3, (ld)10,(ld)-15);
84
85     //Case 4: -ve slope Right to Left line
86     glColor3f(0.0f,0.5f,0.5f);
87     // | m | > 1
88     printDDALine((ld)-3,(ld)2, (ld)-15,(ld)10);

```

```

89     // | m | < 1
90     printDDALine((ld)-2,(ld)3, (ld)-10,(ld)15);
91
92     glEnd();
93 }
94
95 void printDDALine(ld x1, ld y1, ld x2, ld y2) {
96
97     ld pad = PADDING, scale = SCALE;
98
99     x1 = x1*scale + pad;
100    x2 = x2*scale + pad;
101    y1 = y1*scale + pad;
102    y2 = y2*scale + pad;
103
104    ld dx, dy, steps;
105    ld xInc, yInc, x, y;
106
107    dx = (x2-x1);
108    dy = (y2-y1);
109
110    if(abs(dx) > abs(dy))    steps = abs(dx);
111    else                    steps = abs(dy);
112
113    xInc = dx/steps;
114    yInc = dy/steps;
115
116    x = x1; y = y1;
117    glVertex2d(x, y);
118
119    for(long i=1;i<=steps;i++) {
120        x += xInc;
121        y += yInc;
122
123        glVertex2d(x, y);
124    }
125 }

```

SAMPLE I/O



Figure 1: Each color represents different cases for DDA algorithm

PROGRAM - 02

Drawing a line, given two end points

```
1 // Q: To plot points that make up the line with endpoints (x0,y0) and (xn,↵
   yn) using DDA line drawing algorithm.
2 // I/P: Point (x1,y1), Point (x2,y2)
3 // O/P: Line joining (x1,y1) - (x2,y2)
4
5 #include<bits/stdc++.h>
6 #include<GL/glut.h>
7
8 using namespace std;
9 using ld = long double;
10
11 const int WINDOW_WIDTH = 850;
12 const int WINDOW_HEIGHT = 700;
13
14 void myInit();
15 void myDisplay();
16
17 void printLines();
18 void printDDALine(ld x1, ld y1, ld x2, ld y2);
19
20 const ld PADDING = 250;
21 const ld STEP = 1;
22 const ld SCALE = 5;
23
24 int main(int argc, char* argv[]) {
25     glutInit(&argc,argv);
26     glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
27     glutInitWindowSize(WINDOW_WIDTH,WINDOW_HEIGHT);
28     glutCreateWindow("DDA - Line Drawing Algorithm");
29     glutDisplayFunc(myDisplay);
30     myInit();
31     glutMainLoop();
32     return 1;
33 }
34
35 void myInit() {
36     glClearColor(1.0,1.0,1.0,0.0);
37     glColor3f(0.0f,0.0f,0.0f);
38     glPointSize(2.0);
39     glMatrixMode(GL_PROJECTION);
40     glLoadIdentity();
41     gluOrtho2D(0.0,640.0,0.0,480.0);
42 }
```

```

43
44 void myDisplay() {
45     glClear(GL_COLOR_BUFFER_BIT);
46
47     printLines();
48
49     glFlush();
50 }
51
52
53 void printLines() {
54     glBegin(GL_POINTS);
55
56     ld x1, x2, y1, y2;
57     cout << "\n-----\n";
58     cout << "\t\t DDA Line drawing Algorithm \n";
59     cout << "\nStarting point (x1, y1) \n";
60     cout << "\tx1 : "; cin>>x1;
61     cout << "\ty1 : "; cin>>y1;
62
63     cout << "\nEnding point (x2, y2) \n";
64     cout << "\tx2 : "; cin>>x2;
65     cout << "\ty2 : "; cin>>y2;
66
67     printDDALine(x1,y1, x2,y2);
68
69     printf("\nLine between (%.2Lf,%.2Lf) & (%.2Lf, %.2Lf) is drawn.. \n\n"↵
        , x1,y1,x2,y2);
70
71     glEnd();
72 }
73
74 void printDDALine(ld x1, ld y1, ld x2, ld y2) {
75     // m : slope;
76     ld pad = PADDING, scale = SCALE;
77
78     x1 = x1*scale + pad;
79     x2 = x2*scale + pad;
80     y1 = y1*scale + pad;
81     y2 = y2*scale + pad;
82
83     ld dx, dy, steps;
84     ld xInc, yInc, x, y;
85
86     dx = (x2-x1);
87     dy = (y2-y1);
88

```

```
89     if(abs(dx) > abs(dy))    steps = abs(dx);
90     else                    steps = abs(dy);
91
92     xInc = dx/steps;
93     yInc = dy/steps;
94
95     x = x1; y = y1;
96     glVertex2f(x, y);
97
98     for(long i=1;i<=steps;i++) {
99         x += xInc;
100        y += yInc;
101
102        glVertex2f(x, y);
103    }
104 }
```

SAMPLE I/O

```
Multimedia and graphics Lab/EX02 - DDA Line Drawing Algorithm(main) > g++ "02-DdaUser.cpp" -o userLines.out -lGL -lGLU -lglut && ./userLines.out

-----
          DDA Line drawing Algorithm

Starting point (x1, y1)
    x1 : 1
    y1 : 1

Ending point (x2, y2)
    x2 : 20
    y2 : 20

Line between (1.00,1.00) & (20.00, 20.00) is drawn..
```

Figure 2: Getting end points as input from terminal



Figure 3: Window displaying input line

RESULT

The code for DDA line drawing algorithm is written and output is verified.
