

[Report Lab01]- Draw Basic Geometric Shapes Using Algorithms and OpenGL Library

*CS411 — Computer Graphics

NGUYEN TRUNG HAU

Advance Program in Computer Science

University of Science

Ho Chi Minh, Vietnam

nthau18@apcs.vn -18125129

I. INTRODUCTION

In this lab, students are required implementing algorithms to draw 2D objects using OpenGL.

Objects asked to be performed consist of line, circle, ellipse, parabola, hyperbola. These shapes must be displayed on Glut window by executing algorithms such as DDA, Bresenham, MidPoint. After displaying the shapes, running time will be reported. On top of that, students are also asked to constructs their code in OOP methodology.

II. INPUT

The input file is a txt file having N lines. Each represents 1 object, that have M values separated by space. First value indicates object type, while the others indicate object parameters (all distances are pixel length). These lines follow the format below.

A. Input Rule

• Line draw by DDA algorithm:

0 X1 Y1 X2 Y2

(X1, Y1): begin point

(X2, Y2): end point

• Line draw by Bresenham algorithm:

1 X1 Y1 X2 Y2

(X1, Y1): begin point

(X2, Y2): end point

• Circle draw by MidPoint algorithm:

2 XT YT R

(XT, YT): center point

R: radius

• Ellipse draw by MidPoint algorithm:

3 XT YT A B

(XT, YT): center point

A: $\frac{1}{2}$ major length

B: $\frac{1}{2}$ minor length

• Parabola draw by MidPoint algorithm:

4 XT YT P

(XT, YT): center point

P: focal length

• Hyperbola draw by MidPoint algorithm:

5 XT YT A B

(XT, YT): center point

A: major length

B: minor length

B. Sample Input File

These numbers are saved in a text file.

0 5 60 40 40

1 -50 50 70 70

2 1 1 80

3 40 40 16 9

4 -30 -30 10

5 20 20 4 9

The above input are also used to experiment.

III. ALGORITHMS

A. DDA drawing Line

From 2 given Points (bPoint and ePoint), DDA will generate a line connecting them. To do this, start from the beginning point (bPoint), then find the next point by the differences of dx and dy.

$$dx = abs(ePoint.getX() - bPoint.getX()); \quad (1)$$

$$dy = abs(ePoint.getY() - bPoint.getY()); \quad (2)$$

The step is the larger value between dx and dy;

$$step = max(dx, dy); \quad (3)$$

Then we repeat untill reach the ePoint:

```
for (int k = 1; k != d; k++):  
    x += dx;  
    y += dy;  
    putPixelAt(round(x), round(y));
```

B. Bresenham drawing Line

Bresenham algorithm can be resolve "round" values in DDA by computing and comparing the value of p:

$$p = 2 * dy - dx; \quad (4)$$

C. Midpoint

The key idea of this algorithm is to find the next pixel(x_{i+1}) base on the current pixel (x_i) when increasing the x:

$$x_{i+1} = x_i + 1; \quad (5)$$

We compare y to y_i and y_{i+1} to decide putPixel at which one between them.

IV. EXPERIMENT

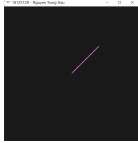
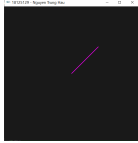



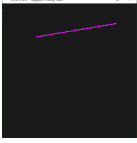
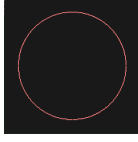
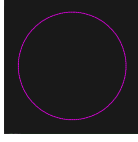
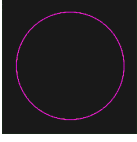
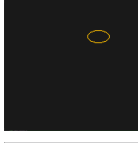
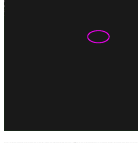
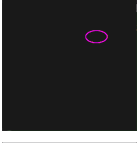




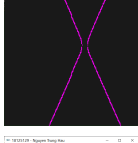
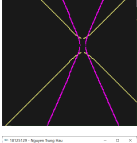



	Algorithm	Equation	Comparison
Line DDA			
Line Bresenham			
Circle MP			
Ellipse MP			
Parabol MP			
Hyperbol MP			
All			

Table Of Running time (milisecond)

	Line	Circle	Ellipse	Parabol	Hyperbol
DDA	0	-	-	-	-
Equation	0	-	-	-	-
Bresenham	0	-	-	-	-
Equation	0	-	-	-	-
MidPoint	-	0	0	0	0
Equation	-	0	0	1000	0

- <https://stackoverflow.com/questions/5886628/efficient-way-to-draw-ellipse-with-opengl-or-d3d>
- <https://stackoverflow.com/questions/22444450/drawing-circle-with-opengl>

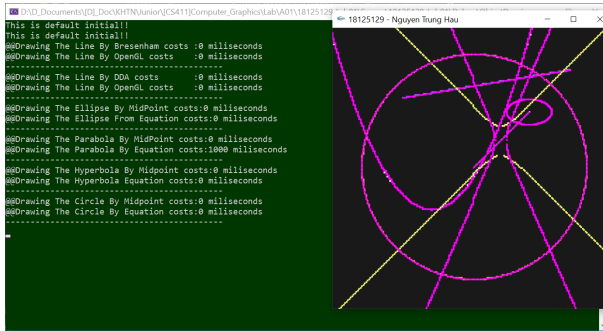


Fig. 1. Proof when running code

V. CONCLUSION

As the figures and pictures above show, drawing by taught algorithms such as DDA, Bresenham, Mi-Point offers results relatively correct to other algorithms which are deduced from equations such as:

$$\text{line} : Y = a * X + b \quad (6)$$

$$\text{circle} : x^2 + y^2 = r^2 \quad (7)$$

$$\text{ellipse} : x^2/a^2 + y^2/b^2 = 1 \quad (8)$$

$$\text{hyperbola} : x^2/a^2 - y^2/b^2 = 1 \quad (9)$$

$$\text{parabola} : y = a * x^2 \quad (10)$$

However, the hyperbola drawn by Midpoint has big error.

VI. REFERENCE

- <https://www.geeksforgeeks.org/mid-point-circle-drawing-algorithm/>
- <https://www.javatpoint.com/computer-graphics-bresenhams-line-algorithm>
- <https://www.youtube.com/watch?v=2Til5nxVYIc>
- <https://community.khronos.org/t/drawing-hyperbola-in-opengl/63123/4>