

Computer Graphics Lab-02

Draw 2D objects with OpenGL

Lê Minh Hoàng - 20125030

1 Shapes

I input all shapes first and then draw each one of them on the screen. I also implemented an event handler to switch between shapes when I click the Previous, Next, Redraw, and Check button.

1.1 Line using DDA

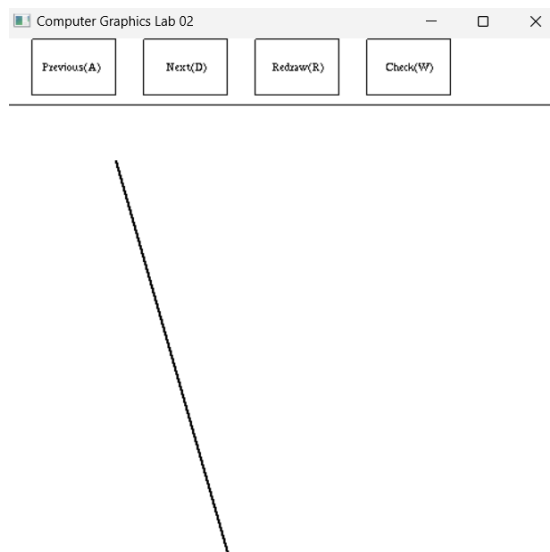


Figure 1: Line drawn using DDA

1.2 Line using Bresenham

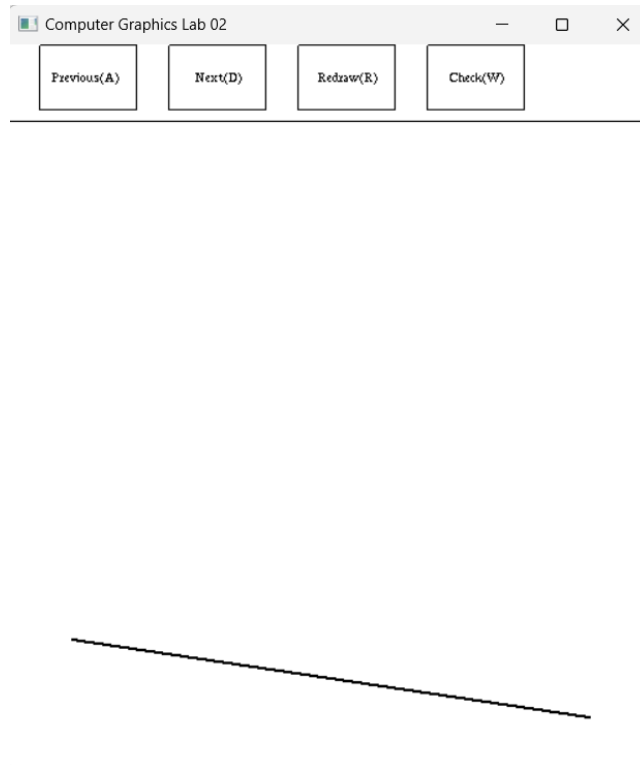


Figure 2: Line drawn using Bresenham

1.3 Circle using Midpoint

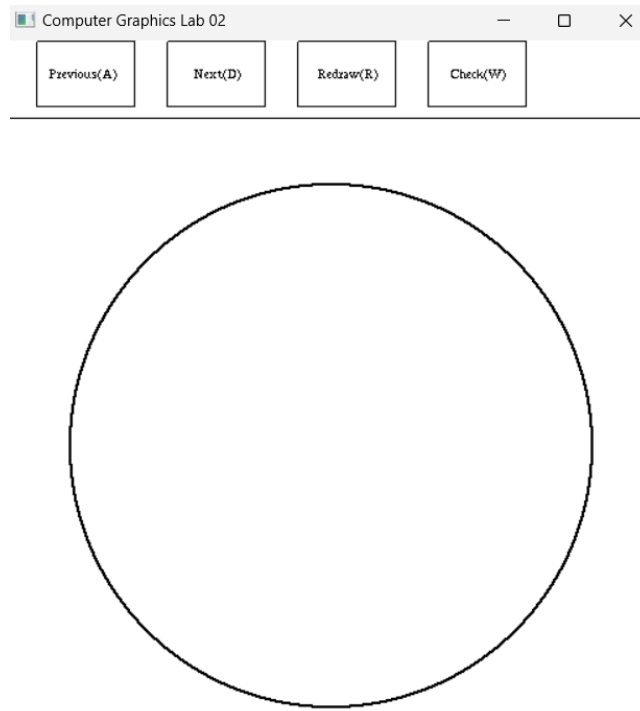


Figure 3: Circle drawn using Midpoint

1.4 Ellipse using Midpoint

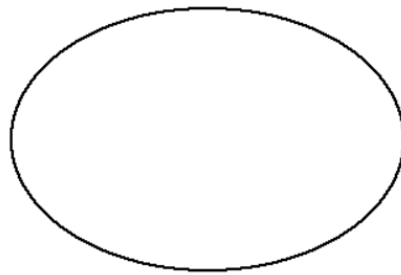
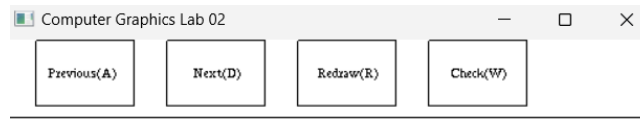


Figure 4: Ellipse drawn using Midpoint

1.5 Parabola using Midpoint

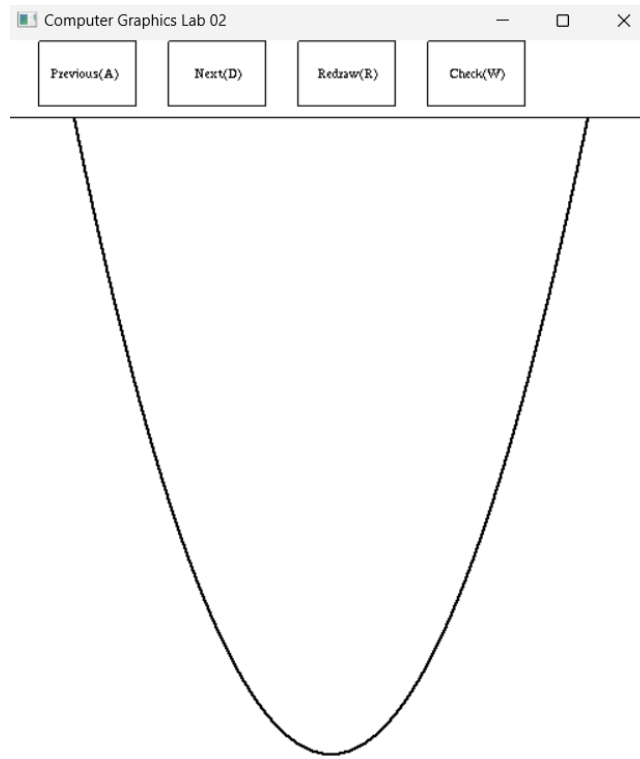


Figure 5: Parabola drawn using Midpoint

1.6 Hyperbola using Midpoint

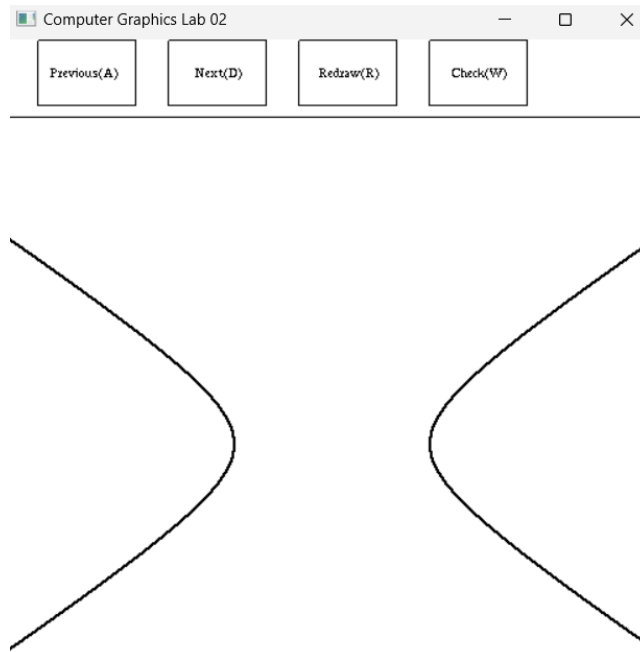
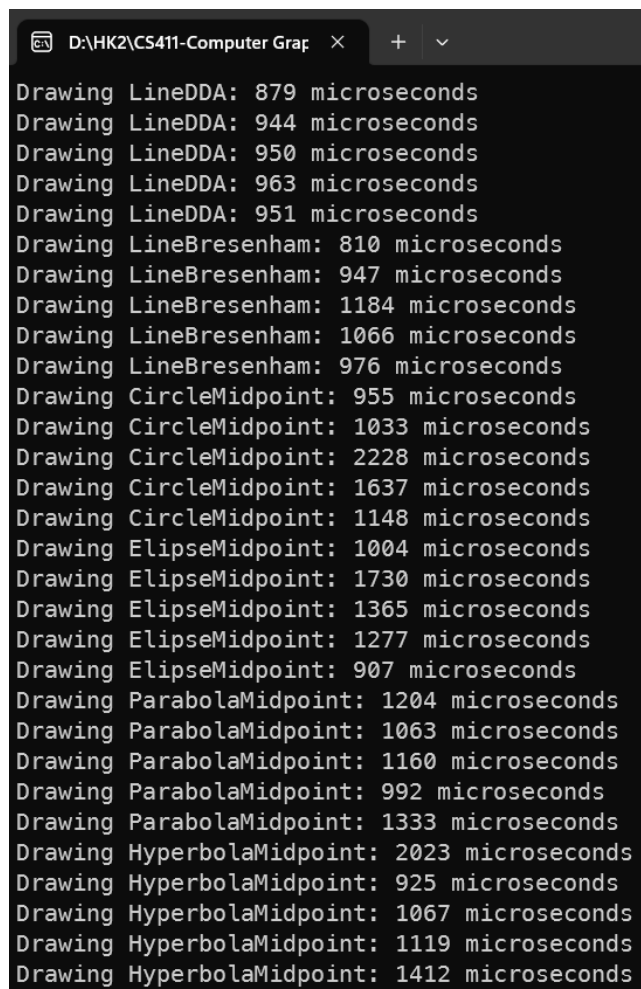


Figure 6: Hyperbola drawn using Midpoint

1.7 Log

A screenshot of a log window titled "D:\HK2\CS411-Computer Gra...". The log contains 30 entries, each starting with "Drawing" followed by a shape name and its execution time in microseconds. The shapes are grouped by type: 5 lines (LineDDA), 5 lines (LineBresenham), 5 circles (CircleMidpoint), 5 ellipses (EllipseMidpoint), and 5 parabolas (ParabolaMidpoint). The last 5 entries are for hyperbolas (HyperbolaMidpoint).

```
Drawing LineDDA: 879 microseconds
Drawing LineDDA: 944 microseconds
Drawing LineDDA: 950 microseconds
Drawing LineDDA: 963 microseconds
Drawing LineDDA: 951 microseconds
Drawing LineBresenham: 810 microseconds
Drawing LineBresenham: 947 microseconds
Drawing LineBresenham: 1184 microseconds
Drawing LineBresenham: 1066 microseconds
Drawing LineBresenham: 976 microseconds
Drawing CircleMidpoint: 955 microseconds
Drawing CircleMidpoint: 1033 microseconds
Drawing CircleMidpoint: 2228 microseconds
Drawing CircleMidpoint: 1637 microseconds
Drawing CircleMidpoint: 1148 microseconds
Drawing EllipseMidpoint: 1004 microseconds
Drawing EllipseMidpoint: 1730 microseconds
Drawing EllipseMidpoint: 1365 microseconds
Drawing EllipseMidpoint: 1277 microseconds
Drawing EllipseMidpoint: 907 microseconds
Drawing ParabolaMidpoint: 1204 microseconds
Drawing ParabolaMidpoint: 1063 microseconds
Drawing ParabolaMidpoint: 1160 microseconds
Drawing ParabolaMidpoint: 992 microseconds
Drawing ParabolaMidpoint: 1333 microseconds
Drawing HyperbolaMidpoint: 2023 microseconds
Drawing HyperbolaMidpoint: 925 microseconds
Drawing HyperbolaMidpoint: 1067 microseconds
Drawing HyperbolaMidpoint: 1119 microseconds
Drawing HyperbolaMidpoint: 1412 microseconds
```

Figure 7: Log after drawing 5 of each shape

2 Result

For OpenGL functions, I can only find a function to draw a line in OpenGL. Because I could not find any builtin function to draw shapes in OpenGL, I instead use parametric equations to draw other shapes.

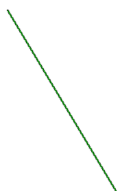
2.1 Time

Algorithm	Time(ms)	Time OpenGL(ms)
LineDDA	0.9808	0.7410
LineBresenham	0.9000	0.9206
CircleMidpoint	1.4764	32.5570
EllipseMidpoint	1.0490	30.8504
ParabolaMidpoint	1.5182	104.1822
HyperbolaMidpoint	0.6920	82.5874

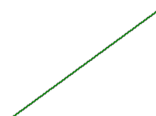
2.2 Accuracy

I test the accuracy of my algorithm by overlaying the shapes drawn by parametric equations on top and compare the pixels.

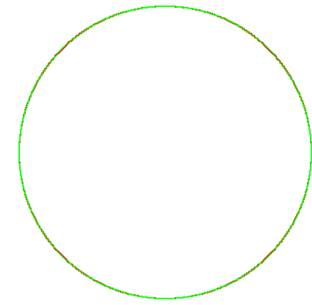
- A green pixel means both my algorithm and parametric equations draw that pixel.
- A red pixel means the parametric equations draw that pixel but my algorithm does not.
- A black pixel means my algorithm draw that pixel but the parametric equations does not.



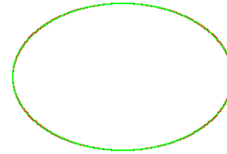
Line drawn using DDA



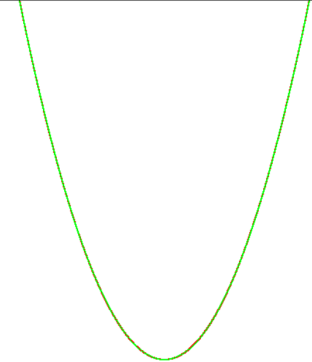
Line drawn using Bresenham



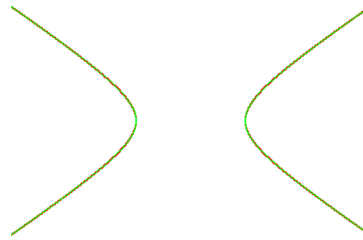
Circle drawn using Midpoint



Ellipse drawn using Midpoint



Parabola drawn using Midpoint



Hyperbola drawn using Midpoint

Comment: There are a lot of green and red pixels and there are no black pixels. The reason why there are red pixels is because for example, in regions where $dx > dy$, parametric functions can draw multiple pixels y pixels for the same x while DDA, Bresenham and Midpoint does not. This means the shapes drawn by parametric equations are thicker than my algorithms. The important thing is that there are no black pixels, which means my algorithm does not draw pixels that are not on the correct line.