

BCT 2405 Computer Graphical systems: Assignment One

Due on 23rd Feb 2025

A). Various Image Formats

1. AI (Adobe Illustrator Artwork)

- **Abbreviation:** Adobe Illustrator.
 - **History:** Developed by Adobe Systems for representing single-page vector-based drawings in EPS or PDF formats. It is the native format for Adobe Illustrator.
 - **Raster or Vector:** Vector.
 - **Typical Application:** Used for logos, illustrations, and print media due to its scalability and precision.
-

2. WMF (Windows Metafile)

- **Abbreviation:** Windows Metafile.
 - **History:** Introduced by Microsoft for storing vector and raster graphics. It is used in Windows applications for clip art and other graphics.
 - **Raster or Vector:** Can contain both vector and raster elements.
 - **Typical Application:** Commonly used in Windows applications for clip art and printing.
-

3. CMX (CorelDRAW Metafile Exchange)

- **Abbreviation:** CorelDRAW Metafile Exchange.
 - **History:** Developed by Corel for use in CorelDRAW. It is a metafile format that supports both vector and raster graphics.
 - **Raster or Vector:** Can contain both vector and raster elements.
 - **Typical Application:** Used for exchanging graphics between CorelDRAW and other applications.
-

4. CGM (Computer Graphics Metafile)

- **Abbreviation:** Computer Graphics Metafile.
- **History:** An ISO-standardized format for 2D vector graphics, raster graphics, and text. It was widely used in technical illustration and engineering.
- **Raster or Vector:** Primarily vector, but can include raster elements.
- **Typical Application:** Technical drawings, engineering diagrams, and aviation industry documentation.

5. SVG (Scalable Vector Graphics)

- **Abbreviation:** Scalable Vector Graphics.
 - **History:** Developed by the World Wide Web Consortium (W3C) as an open standard for vector graphics on the web.
 - **Raster or Vector:** Vector.
 - **Typical Application:** Web graphics, icons, and interactive graphics due to its scalability and small file size.
-

6. ODG (OpenDocument Graphics)

- **Abbreviation:** OpenDocument Graphics.
 - **History:** Part of the OpenDocument standard developed for open-source office suites like LibreOffice and Apache OpenOffice.
 - **Raster or Vector:** Vector.
 - **Typical Application:** Used for creating and editing vector graphics in open-source office applications.
-

7. EPS (Encapsulated PostScript)

- **Abbreviation:** Encapsulated PostScript.
 - **History:** Developed by Adobe for storing vector graphics and is widely used in professional printing.
 - **Raster or Vector:** Primarily vector, but can include raster elements.
 - **Typical Application:** Print media, logos, and illustrations due to its high-quality output.
-

8. DXF (Drawing Exchange Format)

- **Abbreviation:** Drawing Exchange Format.
 - **History:** Developed by Autodesk for enabling data interoperability between AutoCAD and other CAD programs.
 - **Raster or Vector:** Vector.
 - **Typical Application:** CAD (Computer-Aided Design) drawings, engineering, and architectural designs.
-

9. BMP (Bitmap Image File)

- **Abbreviation:** Bitmap.
 - **History:** Introduced by Microsoft for storing bitmap images. It is a simple and widely supported format.
 - **Raster or Vector:** Raster.
 - **Typical Application:** Storing uncompressed images, simple graphics, and icons.
-

10. JPEG (Joint Photographic Experts Group)

- **Abbreviation:** Joint Photographic Experts Group.
 - **History:** Developed in 1992 as a standard for compressing photographic images. It uses lossy compression to reduce file size.
 - **Raster or Vector:** Raster.
 - **Typical Application:** Digital photography, web images, and sharing images online due to its balance of quality and file size.
-

11. GIF (Graphics Interchange Format)

- **Abbreviation:** Graphics Interchange Format.
 - **History:** Developed by CompuServe in 1987. It supports animation and uses lossless compression.
 - **Raster or Vector:** Raster.
 - **Typical Application:** Web animations, simple graphics, and low-color images.
-

12. TIFF (Tagged Image File Format)

- **Abbreviation:** Tagged Image File Format.
 - **History:** Developed by Aldus (now Adobe) in 1986 for storing high-quality images. It supports lossless compression.
 - **Raster or Vector:** Raster.
 - **Typical Application:** Professional photography, publishing, and archiving high-quality images.
-

13. PICT (Picture)

- **Abbreviation:** Picture.
 - **History:** Developed by Apple in 1984 for use on Macintosh systems. It can contain both vector and raster data.
 - **Raster or Vector:** Can contain both vector and raster elements.
 - **Typical Application:** Legacy Macintosh graphics and screen captures.
-

14. PNG (Portable Network Graphics)

- **Abbreviation:** Portable Network Graphics.
- **History:** Developed in the mid-1990s as an open alternative to GIF. It supports lossless compression and transparency.
- **Raster or Vector:** Raster.
- **Typical Application:** Web graphics, logos, and images requiring transparency.

✓ B). LINE DRAWING ALGORITHMS

1. Xiaolin Wu's Line Algorithm

Solution

Xiaolin Wu's Line Algorithm is an anti-aliased line drawing technique that smooths the edges of a line by adjusting pixel intensities. Instead of turning pixels fully on or off, the algorithm interpolates intensities to create a smoother appearance.

Step 1: Compute ΔX , ΔY , and Gradient

Given the points:

- $(x_0, y_0) = (1, 1)$
- $(x_n, y_n) = (3, 5)$

Calculate:

$$\Delta X = x_n - x_0 = 3 - 1 = 2$$

$$\Delta Y = y_n - y_0 = 5 - 1 = 4$$

$$M = \frac{\Delta Y}{\Delta X} = \frac{4}{2} = 2.0$$

Step 2: Compute Step Size and Apply Anti-Aliasing

1. Determine the number of steps:

- If $|\Delta X| > |\Delta Y|$, steps = $|\Delta X|$.
- Else, steps = $|\Delta Y|$.

Since $|\Delta Y| > |\Delta X|$, we take **steps = 4**.

2. Apply Anti-Aliasing:

- Calculate the fractional and integer parts of pixel positions.
- Blend pixel brightness based on distance to the ideal line.

Step 3: Iterative Calculation of Points

Using Xiaolin Wu's anti-aliasing method, we compute successive points:

x_p	y_p	Fractional Part y	Pixel 1 Intensity	Pixel 2 Intensity
1.0	1.0	0.00	1.00	0.00
1.5	2.0	0.50	0.50	0.50
2.0	3.0	0.00	1.00	0.00
2.5	4.0	0.50	0.50	0.50
3.0	5.0	0.00	1.00	0.00

OpenGL Implementation and plot



2. Gupta-Sproull Line Algorithm

Solution

The **Gupta-Sproull Algorithm** is an anti-aliased line drawing technique that smooths edges by adjusting pixel intensities based on the distance from the ideal line.

Step 1: Compute ΔX , ΔY , and Gradient

Given the points:

- **Starting coordinates:** $(X_0, Y_0) = (-2, 3)$
- **Ending coordinates:** $(X_n, Y_n) = (1, 4)$

Calculate:

$$\begin{aligned}\Delta X &= X_n - X_0 = 1 - (-2) = 3 \\ \Delta Y &= Y_n - Y_0 = 4 - 3 = 1 \\ M &= \frac{\Delta Y}{\Delta X} = \frac{1}{3} \approx 0.33\end{aligned}$$

Step 2: Compute Initial Decision Parameter and Apply Anti-Aliasing

Using the **Gupta-Sproull method**:

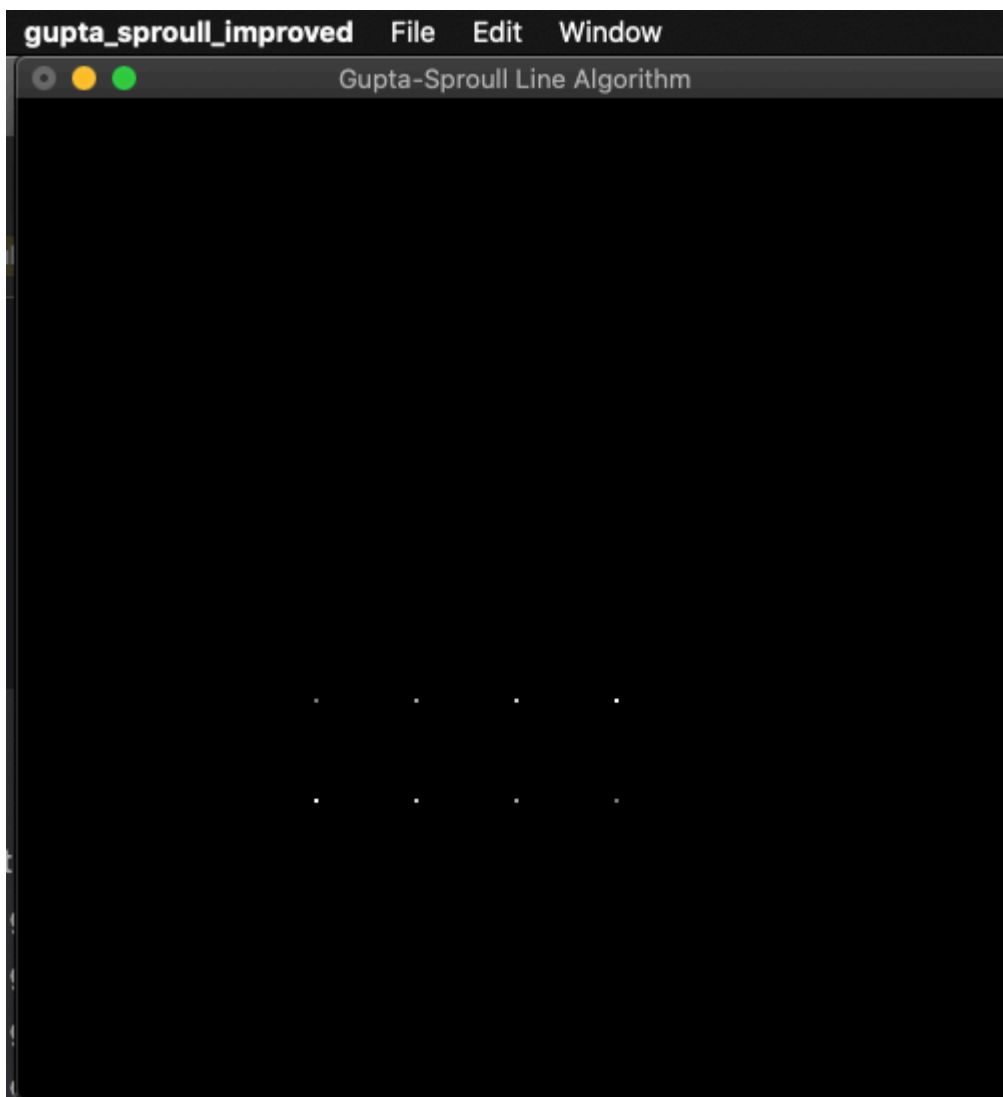
- Compute the perpendicular distance to the ideal line.
- Adjust pixel brightness based on its proximity to the true line.

Step 3: Iterative Calculation of Points

Using distance-based anti-aliasing, we compute successive points:

x_p	y_p	Distance from Line	Pixel Intensity
-2.0	3.0	0.00	1.00
-1.0	3.3	0.30	0.70
0.0	3.6	0.60	0.40
1.0	4.0	0.00	1.00

OpenGL Implementation



3. Midpoint Algorithm

Solution

Given the starting and ending coordinates of a line, the Midpoint Line Drawing Algorithm determines which intermediate pixels should be turned on to approximate the line on a raster display.

Step 1: Compute ΔX and ΔY

Given the points:

- $(x_0, y_0) = (-3, 4)$
- $(x_n, y_n) = (5, -2)$

Calculate:

$$\Delta X = x_n - x_0 = 5 - (-3) = 8$$

$$\Delta Y = y_n - y_0 = -2 - 4 = -6$$

Step 2: Compute Initial Decision Parameter

Define the initial decision parameter $D_{initial}$:

$$D_{initial} = 2\Delta Y - \Delta X = 2(-6) - 8 = -12 - 8 = -20$$

$$\Delta D_{east} = 2\Delta Y = 2(-6) = -12$$

$$\Delta D_{northeast} = 2(\Delta Y - \Delta X) = 2(-6 - 8) = 2(-14) = -28$$

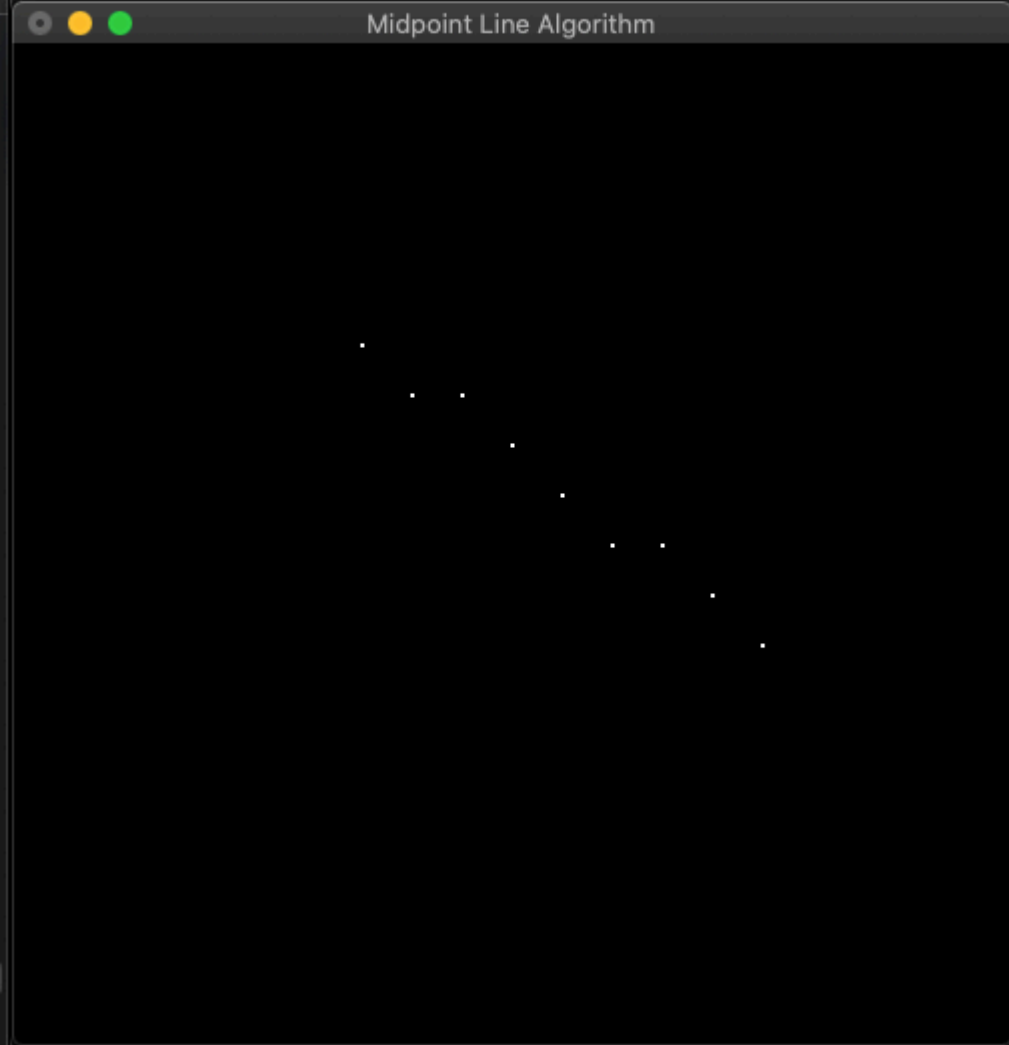
Step 3: Iterative Calculation of Points

Using the decision parameter updates, we compute successive points:

$D_{initial}$	D_{new}	X_{k+1}	Y_{k+1}
-20	-32	-2	4
-32	-44	-1	4
-44	-56	0	4
-56	-68	1	4
-68	-80	2	3
-80	-92	3	3
-92	-104	4	3
-104	-116	5	2

OpenGL Implementation

```
macbook@macbooks-MacBook-Pro Desktop % g++ midpoint_line.cpp  
-o midpoint_line -framework OpenGL -framework GLUT  
macbook@macbooks-MacBook-Pro Desktop % ./midpoint_line
```



✓ 4. Bresenham's Line Drawing Algorithm Calculation

Solution

Bresenham's Line Drawing Algorithm is an efficient raster graphics algorithm for drawing a straight line between two given points.

Step 1: Compute ΔX and ΔY

Given the points:

- $(x_0, y_0) = (1, 5)$
- $(x_n, y_n) = (2, 8)$

Calculate:

$$\Delta X = x_n - x_0 = 2 - 1 = 1$$

$$\Delta Y = y_n - y_0 = 8 - 5 = 3$$

Step 2 Compute Decision Parameter

$$P_k = 2\Delta Y - \Delta X = 2(3) - 1 = 6 - 1 = 5$$

$$\Delta P_k = 2\Delta Y = 2(3) = 6$$

$$\Delta P_{\text{northeast}} = 2(\Delta Y - \Delta X) = 2(3 - 1) = 2(2) = 4$$

Step 3: Iterative Calculation of Points

The iteration follows the decision rule:

- **Case 1:** If $P_k \geq 0$, update as:

$$P_{k+1} = P_k + \Delta P_{\text{northeast}}$$

$$x_{k+1} = x_k + 1, \quad y_{k+1} = y_k + 1$$

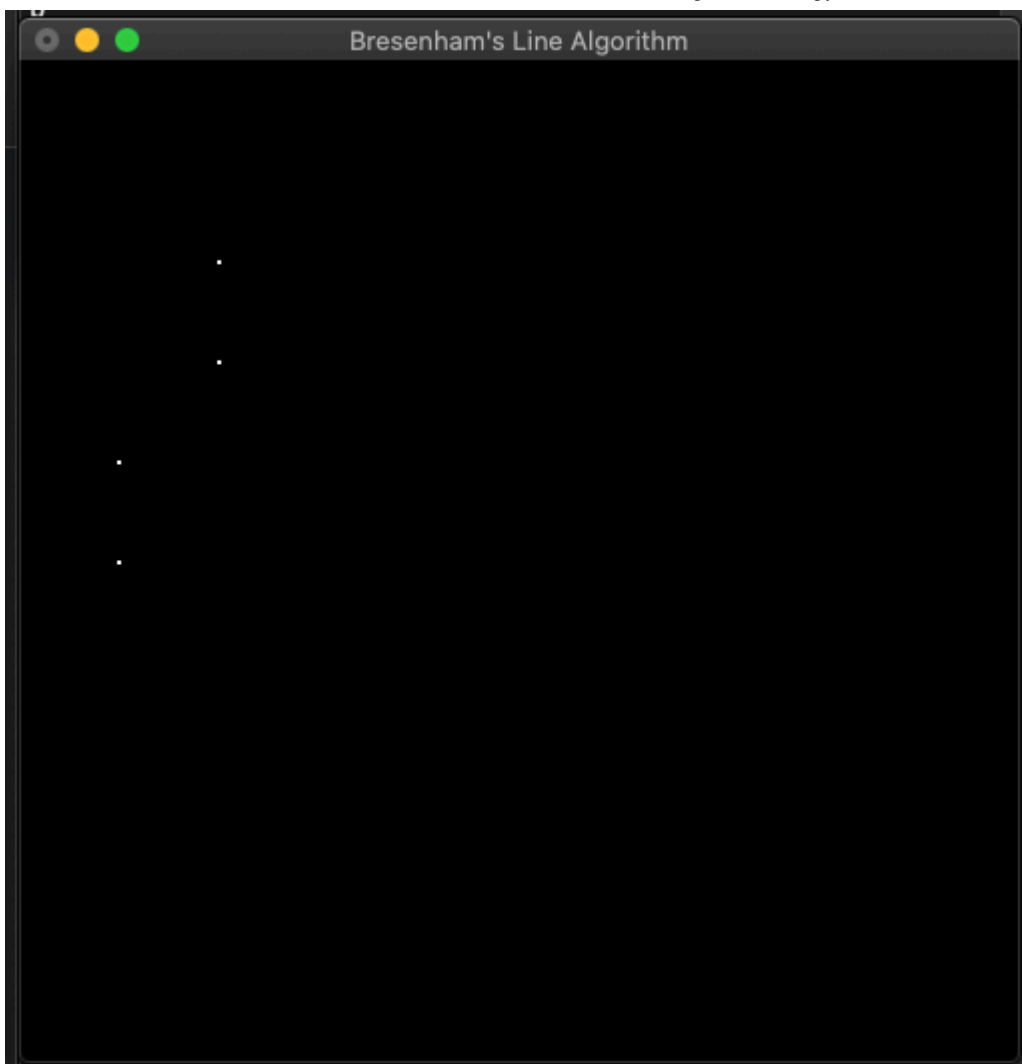
- **Case 2:** If $P_k < 0$, update as:

$$P_{k+1} = P_k + \Delta P_{\text{east}}$$

$$x_{k+1} = x_k + 1, \quad y_{k+1} = y_k$$

P_k	P_{k+1}	X_{k+1}	Y_{k+1}
5	9	2	6
9	13	3	7
13	17	4	8

OpenGL Implementation



5. Midpoint Line Drawing Algorithm

Solution

Given the starting and ending coordinates of a line, the **Midpoint Line Drawing Algorithm** attempts to generate the points between them by evaluating a decision parameter to determine the next pixel.

Step 1: Compute ΔX and ΔY

Given the points:

- **Starting coordinates:** $(X_0, Y_0) = (0, 2)$
- **Ending coordinates:** $(X_n, Y_n) = (-1, 4)$

Calculate:

$$\begin{aligned}\Delta X &= X_n - X_0 = -1 - 0 = -1 \\ \Delta Y &= Y_n - Y_0 = 4 - 2 = 2\end{aligned}$$

Step 2: Compute Initial Decision Parameter and ΔD

Calculate the initial decision parameter:

$$D_{initial} = 2\Delta Y - \Delta X = 2(2) - (-1) = 4 + 1 = 5$$

Calculate the incremental change in decision parameter:

$$\Delta D = 2(\Delta Y - \Delta X) = 2(2 - (-1)) = 2(3) = 6$$

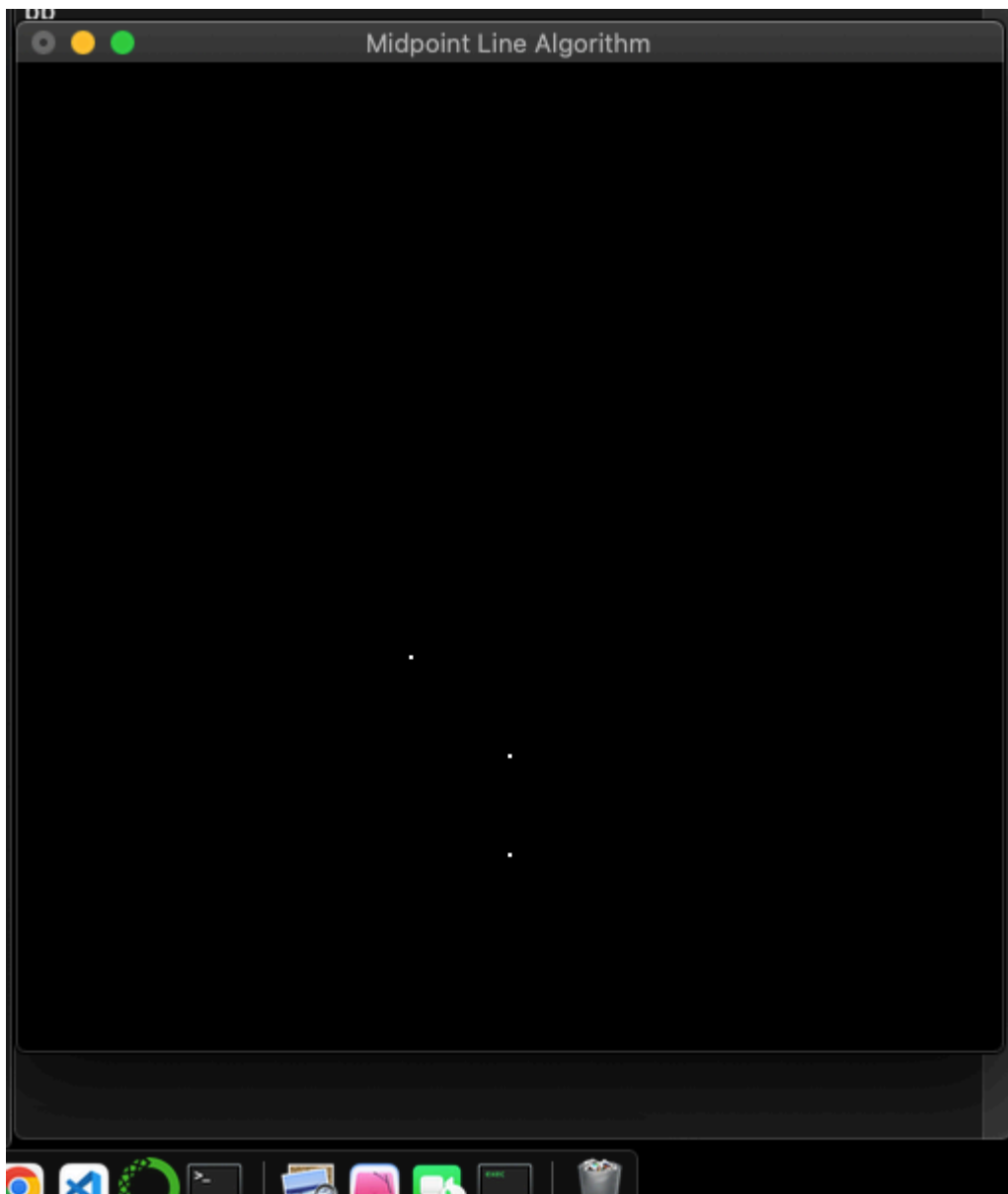
Step 3: Iterative Calculation of Points

The decision whether to increment X or Y coordinate depends on the value of $D_{initial}$.

- If $D_{initial} \geq 0$, move diagonally.
- Otherwise, move horizontally.

$D_{initial}$	D_{new}	X_{k+1}	Y_{k+1}
5	11	-1	3
11	17	-1	4

OpenGL Implementation



6. DDA Line Drawing Algorithm

Solution

The Digital Differential Analyzer (DDA) algorithm is used for rasterizing a line by calculating intermediate points between the start and end points.

Step 1: Compute ΔX , ΔY , and M

Given the points:

- $(x_0, y_0) = (5, 2)$
- $(x_n, y_n) = (10, 3)$

Calculate:

$$\begin{aligned}\Delta X &= x_n - x_0 = 10 - 5 = 5 \\ \Delta Y &= y_n - y_0 = 3 - 2 = 1 \\ M &= \frac{\Delta Y}{\Delta X} = \frac{1}{5} = 0.2\end{aligned}$$

Step 2: Compute number of steps or points

To determine the number of steps:

$$\begin{aligned}\text{If } |\Delta X| > |\Delta Y|, \text{ then } S &= |\Delta X| \\ \text{Else, } S &= |\Delta Y|\end{aligned}$$

Applying this:

$$\begin{aligned}\text{Since } |\Delta X| = 5 \text{ and } |\Delta Y| = 1, \text{ we choose:} \\ S &= |\Delta X| = 5\end{aligned}$$

Now, compute the increments for x and y:

$$\begin{aligned}I_x &= \frac{\Delta X}{S} = \frac{5}{5} = 1.0 \\ I_y &= \frac{\Delta Y}{S} = \frac{1}{5} = 0.2\end{aligned}$$

Case Selection Based on M

- **Case 1:** If $0 < M < 1$, increment x_p by 1 and y_p by M .
- **Case 2:** If $M > 1$, increment y_p by 1 and x_p by $1/M$.
- **Case 3:** If $M = 1$, increment both x_p and y_p by 1.

In this case, since $M = 0.2$, **Case 1 is satisfied**.

Step 3: Iterative Calculation of Points

Using the computed increments, generate successive points:

x_p	y_p	x_{p+1}	y_{p+1}	Round(x_{p+1}, y_{p+1})
5.0	2.0	6.0	2.2	(6,2)
6.0	2.2	7.0	2.4	(7,2)
7.0	2.4	8.0	2.6	(8,3)
8.0	2.6	9.0	2.8	(9,3)
9.0	2.8	10.0	3.0	(10,3)

OpenGL Implementation

