Department of Computer Science and Engineering

| Course Code: CSC 422 | Credits: 0.50 |
|---|---|
| Course Name: Computer Graphics | |

# Lab 01
## Midpoint Line Drawing Algorithm

**Topic Overview:**

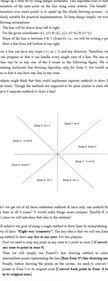
The students were introduced to the DDA line drawing algorithm in the theory class. Given the coordinates of any two points on the line, $(x1, y1)$ & $(x2, y2)$, the scheme works. It calculates the slope of the line and depending on the slope, it increments the coordinates of pixels where every pixel has integer coordinates. The DDA algorithm uses floats and also does floating point arithmetic.

Therefore, we would be introducing another significant line drawing algorithm in today's class speed-things a little bit by using integer arithmetic. The algorithm tries to find the best approximation of a line. It is fast and it does not use floating point. The idea of best approximation on raster points is so good up the whole drawing process looks perfectly suitable for practical implementations. To keep things simple, we would consider only the positive slope.

- The line will be drawn from left to right.
- For the given coordinates $(x1, y1)$ & $(x2, y2)$ $x1 < x2$ & $y1 < y2$
- Slope of the line is between 0 & 1 (Exactly, i.e., we will be writing a program to draw a line from left bottom to top right.

However, a line can have any slope. It can be in any of four directions. Therefore, we need to allow our program so that it can handle every single case of a line. We can consider a line with slope 0 to 1 in the first quadrant. We can use the same logic for the other quadrants. Implementing midpoint line drawing algorithm, we can modify the code of our program so that it can draw any line in any quadrant.

The students might think that they could implement separate methods to draw lines in 8 different zones. Though the methods are supposed to be quite similar to each other, it's a lot to know 8 separate methods to draw a line!



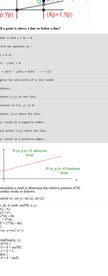

Couldn't we get all of these redundant methods & have only one method that would handle lines in all 8 zones? It would make things more compact. Handle 8 zones. Handle 4 zones or will handle 2 zones that cover the whole screen.

We will achieve our goal of using a single method to draw lines by implementing a special function. We would be using the function to swap the coordinates. We would be using the function to draw a line in any zone. **Swap is the key.**

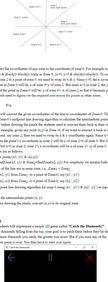
- First we need to map any point in any zone to a point in zone 1**(Convert point in any zone to point in zone 1)**
- Then we will simply use Bresenham's line drawing method to calculate the intermediate points on the line in zone 1 (Draw a line in zone 1)
- Finally, before drawing the pixels on the screen, we need to convert back the point in Zone 1 to original zone **(Convert back point in Zone 1 to a point in its original zone)**

**Swap is the key.**

Consider a line in zone 2 and we want to draw a line in zone 1. We can do it by swapping x and y coordinates. Similarly, we can swap x and y coordinates for any point in zone 2 to map it to zone 1. Suppose, we want to map a point in zone 3 to zone 1. We can do it by swapping x and y and then taking the negative of both the coordinates. Suppose, we want to map a point in zone 4 to zone 1. We can do it by taking the negative of the point's x coordinate. Suppose, we want to map a point in zone 5 to zone 1. We can do it by taking the negative of both the coordinates. Suppose, we want to map a point in zone 6 to zone 1. We can do it by swapping x and y and then taking the negative of the point's x coordinate. Suppose, we want to map a point in zone 7 to zone 1. We can do it by taking the negative of the point's y coordinate. Suppose, we want to map a point in zone 8 to zone 1. We can do it by swapping x and y and then taking the negative of the point's y coordinate.

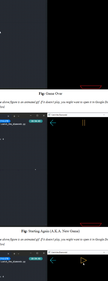



Now, we have to find out the slope of the line. Suppose, we have a line in zone 1 with slope 0 to 1. We can find out the next point on the line. Suppose, we have a line in zone 2 with slope 0 to 1. We can find out the next point on the line. Suppose, we have a line in zone 3 with slope 0 to 1. We can find out the next point on the line. Suppose, we have a line in zone 4 with slope 0 to 1. We can find out the next point on the line. Suppose, we have a line in zone 5 with slope 0 to 1. We can find out the next point on the line. Suppose, we have a line in zone 6 with slope 0 to 1. We can find out the next point on the line. Suppose, we have a line in zone 7 with slope 0 to 1. We can find out the next point on the line. Suppose, we have a line in zone 8 with slope 0 to 1. We can find out the next point on the line.




Now, we know the coordinates of any zone to the coordinates of zone 1. For example in Zone 2 we can find out the next point on the line in Zone 2 and then we can map it to Zone 1. Suppose, we want to draw a line in zone 2 with slope 0 to 1. We can find out the next point on the line in Zone 2 and then we can map it to Zone 1. Suppose, we want to draw a line in zone 1 with slope 0 to 1. We can find out the next point on the line in Zone 1. Suppose, we want to draw a line in zone 3 with slope 0 to 1. We can find out the next point on the line in Zone 3 and then we can map it to Zone 1. Suppose, we want to draw a line in zone 4 with slope 0 to 1. We can find out the next point on the line in Zone 4 and then we can map it to Zone 1. Suppose, we want to draw a line in zone 5 with slope 0 to 1. We can find out the next point on the line in Zone 5 and then we can map it to Zone 1. Suppose, we want to draw a line in zone 6 with slope 0 to 1. We can find out the next point on the line in Zone 6 and then we can map it to Zone 1. Suppose, we want to draw a line in zone 7 with slope 0 to 1. We can find out the next point on the line in Zone 7 and then we can map it to Zone 1. Suppose, we want to draw a line in zone 8 with slope 0 to 1. We can find out the next point on the line in Zone 8 and then we can map it to Zone 1.

Now, to convert the coordinates of any zone to the coordinates of zone 1. For example in Zone 2 we can find out the next point on the line in Zone 2 and then we can map it to Zone 1. Suppose, we want to draw a line in zone 2 with slope 0 to 1. We can find out the next point on the line in Zone 2 and then we can map it to Zone 1. Suppose, we want to draw a line in zone 1 with slope 0 to 1. We can find out the next point on the line in Zone 1. Suppose, we want to draw a line in zone 3 with slope 0 to 1. We can find out the next point on the line in Zone 3 and then we can map it to Zone 1. Suppose, we want to draw a line in zone 4 with slope 0 to 1. We can find out the next point on the line in Zone 4 and then we can map it to Zone 1. Suppose, we want to draw a line in zone 5 with slope 0 to 1. We can find out the next point on the line in Zone 5 and then we can map it to Zone 1. Suppose, we want to draw a line in zone 6 with slope 0 to 1. We can find out the next point on the line in Zone 6 and then we can map it to Zone 1. Suppose, we want to draw a line in zone 7 with slope 0 to 1. We can find out the next point on the line in Zone 7 and then we can map it to Zone 1. Suppose, we want to draw a line in zone 8 with slope 0 to 1. We can find out the next point on the line in Zone 8 and then we can map it to Zone 1.



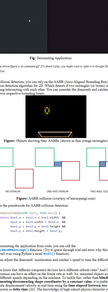

**Steps:**

1. Swap
2. Convert (x1, y1) from Zone1 to point (Zone1, x0, y0) // Convert point in any zone to point in zone 1
3. Convert (x2, y2) from Zone1 to point (Zone1, x0, y0) // Convert point in any zone to point in zone 1
4. Draw a line in zone 1 (Zone1, x0, y0, x1, y1)
5. Convert (x0, y0) from Zone1 to point (Zone1, x0, y0) // Convert back point in zone 1 to point in original zone
6. Convert (x1, y1) from Zone1 to point (Zone1, x0, y0) // Convert back point in zone 1 to point in original zone
7. Draw a line in original zone (x0, y0, x1, y1)

```
Swap(x1, y1)
Swap(x2, y2)
Zone = Find_Zone(x1, y1, x2, y2)
Point1 = Convert_Point_To_Zone_1(Zone, x1, y1)
Point2 = Convert_Point_To_Zone_1(Zone, x2, y2)
Draw_Line_In_Zone_1(Point1.x, Point1.y, Point2.x, Point2.y)
Point1 = Convert_Point_From_Zone_1_To_Original_Zone(Zone, Point1.x, Point1.y)
Point2 = Convert_Point_From_Zone_1_To_Original_Zone(Zone, Point2.x, Point2.y)
Draw_Line_In_Original_Zone(Point1.x, Point1.y, Point2.x, Point2.y)
```

**Bonus:**

- Draw a circle based at the bottom of the player screen, which can be revived horizontally using **left and right arrow keys** (press and hold down the arrow keys) **up** out of the screen.
- The downside left vertically from the top of the screen. For a diamond to be regarded as "satisfied", the catcher need to right towards it at the right moment (crossing the catcher and the diamond vertical lines at the same pixel). Then it will be one diamond falling at a time on the screen.
- If a diamond is "satisfied", your score will increase by 1, and the current score should be printed on the console. After that, the new diamond will start falling from the top. The color of the diamond will be random. The horizontal position of it will be random as well.
- If a diamond is not "satisfied" and it reaches the bottom of the screen, the game ends. Your score should be printed on the console. Also, the game should ask if you want to restart the game, and then it will start and automat all round when. In the console, "Game Over" should be printed, and the game should end.
- **The speed of the diamond falling will gradually increase with time to ramp up the difficulty.**
- Time will be a billable bonus on the top of the screen (all-down using sprites/text).
  - A bright red colored bonus the ball is left in the shape of a left arrow. Clicking this will restart the game (and the speed is over or not). Your score and diamond speed will also be reset. A new diamond will also start falling. You can also click on the "bonus" to be crucial. Don't forget to revive back the catcher if it is out of the screen.
  - An extra colored bonus the ball is in the shape of a plus or green icon. Clicking this will ignite a power-up that will make the diamond only fall vertical on the ones or only. Which is. The power-ups ends when the catcher is pulled out of the screen. Clicking on the power-up when it is in the screen will pause the game. After falling diamond falls down, and you can be the catcher to hide out the pause. Clicking the power-up when it is in the screen will resume the game.
  - And a red colored button on the right is the shape of a cross. Clicking this will pop up "Goodbye" along with your score in the console, and terminate the application.
- Text/Bonus Text should be displayed on the screen. Text/Bonus Text should be displayed using "Ui" in **client mode**, the game will **play automatically by moving the catcher left and right without mirroring any falling diamond** even (Text need to implement this). Text/Bonus Text should be displayed using "Ui" in **server mode**, the game will **not play and just teleport the catcher thereto.** The client mode will **add** Text **some buttons** in personal sight.
- **You have to show everything on screen using the midpoint line drawing algorithm. With that being said, you're only allowed to use the DrawPixel() prototype type.**
- Your code should be well-commented, and it should be well-organized. Clicking on the catcher's length, it shouldn't be too long that it gets **stuck** in the middle of the screen. **The shape of the diamond and the catcher should be exactly what's shown in the figure. diamond will free-fall/spin/bare and catcher with another free-adjustment bars, maintaining the shape shown in the figure.**
- Your code should be well-commented, the entire should be highly-arranged, and it's uniform with the background.




*Please Note: The drawing shown is an example of 2D diamond (plus, you might want to assume to draw Bonus Text).*




*Please Note: The drawing shown is an example of 2D diamond (plus, you might want to assume to draw Bonus Text).*




*Please Note: The drawing shown is an example of 2D diamond (plus, you might want to assume to draw Bonus Text).*




*Please Note: The drawing shown is an example of 2D diamond (plus, you might want to assume to draw Bonus Text).*

*Please Note: The drawing shown is an example of 2D diamond (plus, you might want to assume to draw Bonus Text).*

**Tips:**

- For collision detection, you can rely on the AABB (Axis-Aligned Bounding Box) collision detection algorithm (for 2D). Which directly corresponds to the rectangular shape of the diamond and the catcher. You can consider the diamond and catcher to have their own rectangular bounding boxes.

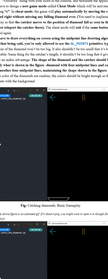



**Figure:** AABB collision (courtesy of Immersyncat.com)

Here is the pseudocode for AABB collision detection:

```
function boxesIntersect(a, b): -- assumed to exist
    return box1.x + box1.width > box2.x and
           box1.x < box2.x + box2.width and
           box1.y + box1.height > box2.y and
           box1.y < box2.y+box2.height end,
```

- For terminating the application from exit, you can call the function exit(0) from the "stdlib.h" library. Please read and note why this is preferred over using Debug.Quit() and Application.Quit().
- You can adjust the diamonds' acceleration and catcher's speed to turn the difficulty of the game.

Do you know that billionaire computer-the-icon have billion-dollar-valued ideas? And CPU performance can have an effect on the frame rate as well. So, automated sharpen might be a billion-dollar-valued idea. Automated sharpen is a **Deep Learning** based **Image Sharpening/Deconvolving Image contributions by a constant value.** It is preferable to use Deep Learning to sharpen the image. Deep Learning is a subset of Machine Learning. It also known as **Artificial Intelligence**. The knowledge of high-school physics literature will be helpful for the Deep Learning. Deep Learning is a subset of Machine Learning with no Python-based. English Wikipedia article on Deep Learning here.

- You can't use any other OpenGL functions except the types mentioned above. Any other OpenGL functions will be considered as cheating and zero in Bug. Further, the marks will be penalized by 50% mark in the whole assignment.

- Keep in mind that 50% mark will depend on the midpoint line drawing implementation, and 50% will be on the game. Anyone making the game without midpoint line, hacking will be awarded with 0.