Computer Graphics

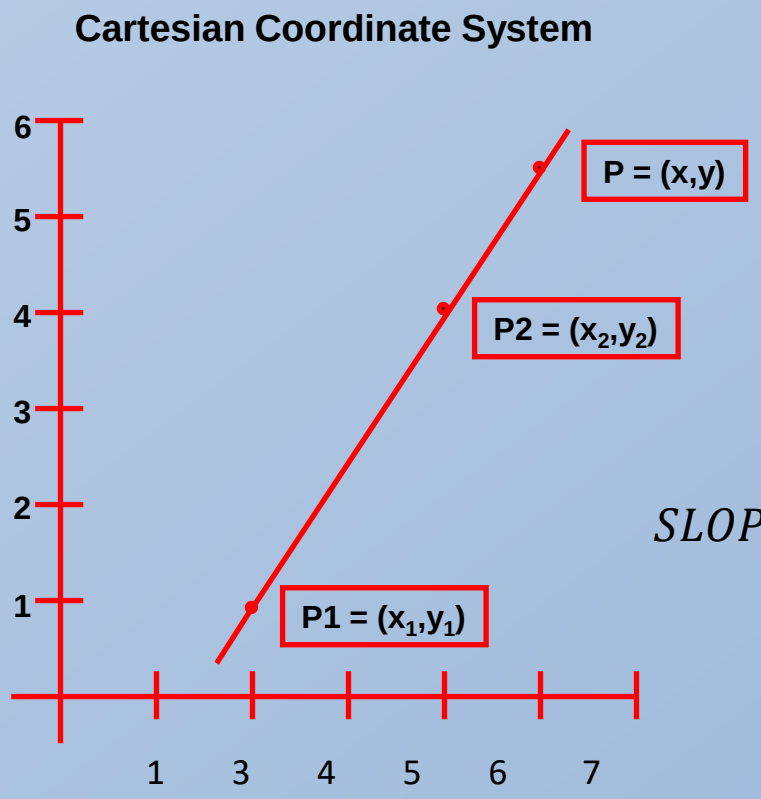
Lecture 3

By

Kareem Ahmed

Based on Computer Graphics by Hearn & Baker

Basic Math Review



$$SLOPE = \frac{RISE}{RUN} = \frac{y_2 - y_1}{x_2 - x_1}$$

Basic Math Review

Slope-Intercept Formula For A Line

Given a third point on the line:

$$P = (x, y)$$

$$Slope = \frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$$

Solving For y

$$y = y_1 + \left(\frac{y_2 - y_1}{x_2 - x_1}\right)x - \left(\frac{y_2 - y_1}{x_2 - x_1}\right)x_1$$

Therefore

$$y = Mx + B$$

Where

$$M = \frac{y_2 - y_1}{x_2 - x_1}$$

$$B = y_1 - \left(\frac{y_2 - y_1}{x_2 - x_1}\right)x_1$$

Other Helpful Formulas

Length of line segment between P_1 and P_2 :

$$L = sqrt[(x_2 - x_1)^2 + (y_2 - y_1)^2]$$

Midpoint of a line segment between P_1 and P_3:

$$P_2 = \left(\frac{x_1 + x_3}{2}, \quad \frac{y_1 + y_3}{2}\right)$$

Two lines are **perpendicular** iff

$$M_1 = \frac{-1}{M_2}$$

or Cosine of the angle between them is 0.

Parametric Form Of The Equation Of A 2D Line Segment

Given points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$

$$x = x_1 + t(x_2 - x_1)$$

$$y = y_1 + t(y_2 - y_1)$$

t is called the parameter. When

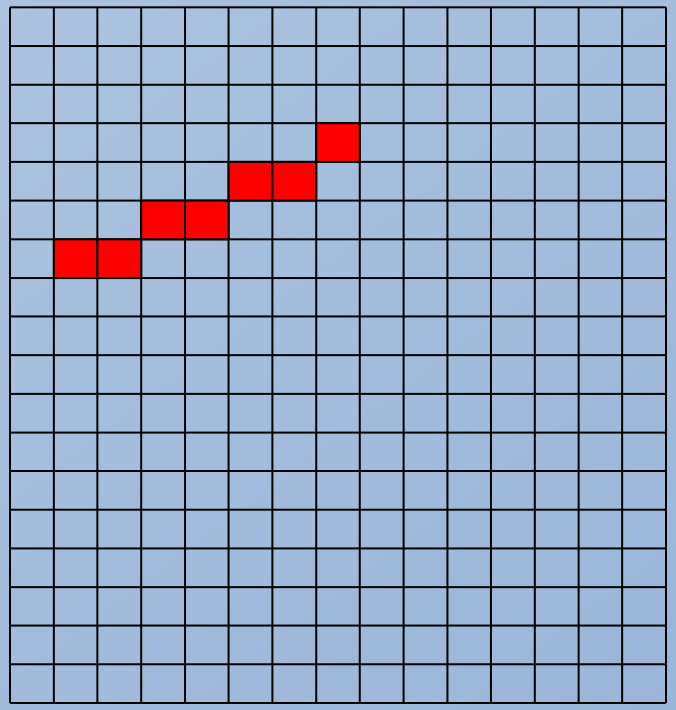
t = 0 we get (x_1, y_1)

t = 1 we get (x_2, y_2)

As $0 < t < 1$ we get all the other points on the line segment between (x_1, y_1) and (x_2, y_2).

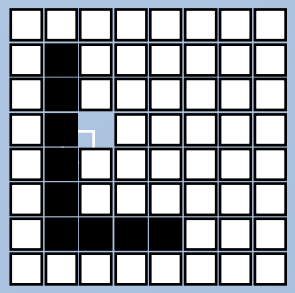
How does computer draw line?

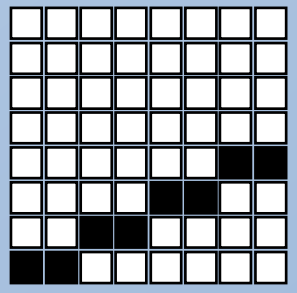
- Screen made of pixels
- High-level language specifies line
- System must color pixels

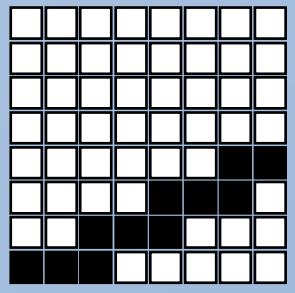


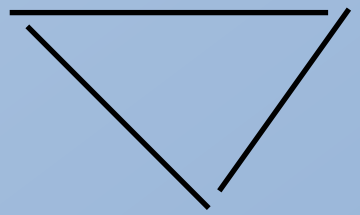
Basic Line and Circle Algorithms

1. Must compute integer coordinates of pixels which lie on or near a line or circle.
2. Pixel level algorithms are invoked hundreds or thousands of times when an image is created or modified.
3. Lines must create visually satisfactory images.
    - Lines should appear straight
    - Lines should terminate accurately
    - Lines should have constant density
    - Line density should be independent of line length and angle.
4. Line algorithm should always be defined.









DDA Algorithm

- Start with starting and ending coordinates of the line:

$$(x_0,\ y_0) \text{ and } (x_1,\ y_1)$$

- Color first pixel (round to nearest integer)
- Suppose $x_1 - x_0 > y_1 - y_0$ (gentle slope)
  - There will be $x_1 - x_0$ steps (# pixels to be colored)
- Set $x = x_0$, $y = y_0$
- At each step,
  - Increment x by $\frac{(x_1 - x_0)}{Num\_of\_steps}$
  - Increment y by $\frac{(y_1 - y_0)}{Num\_of\_steps}$
- For each step, round off x and y to nearest integer, and color pixel

DDA Pseudo-code

```
// assume that slope is gentle
DDA(float x0, float x1, float y0, float y1) {
   float x, y;
   float xinc, yinc;
   int numsteps;

   numsteps = Round(x1) – Round(x0);
   xinc = (x1 – x0) / numsteps;
   yinc = (y1 – y0) / numsteps;
   x = x0;
   y = y0;
   ColorPixel(Round(x),Round(y));

   for (int i=0; i<numsteps; i++) {
      x += xinc;
      y += yinc;
      ColorPixel(Round(x),Round(y));
   }
}
```

Q: For each step, how many floating point operations are there?
Answer: 4

Q: For each step, how many integer operations are there?
Answer: 2

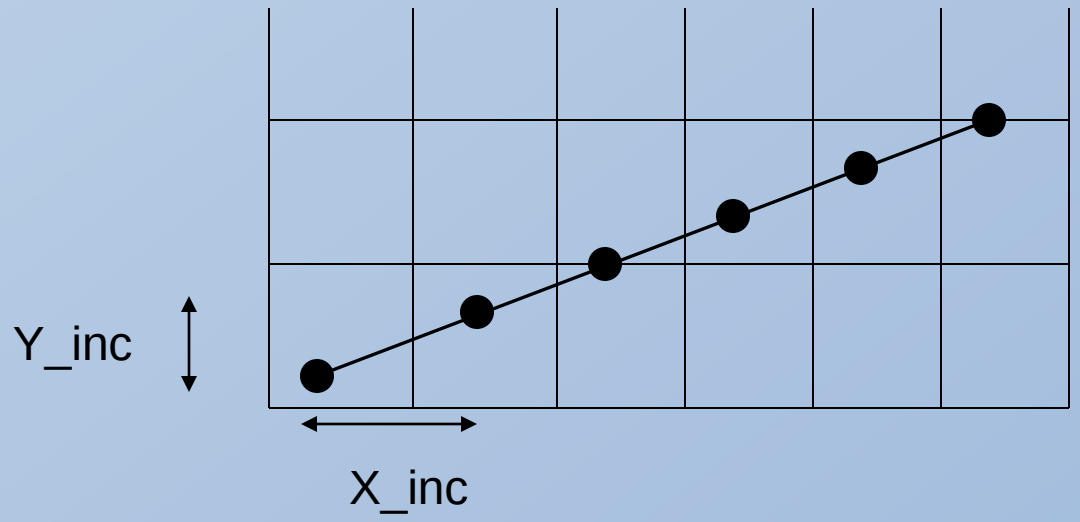
DDA Example

```
numsteps = 12 - 2 = 10
xinc = 10/10 = 1.0
yinc = 5/10 = 0.5
```

- Suppose we want to draw a line starting at pixel (2,3) and ending at pixel (12,8).
- What are the values of the variables x and y at each timestep?
- What are the pixels colored, according to the DDA algorithm?

t	x	y	R(x)	R(y)
0	2	3	2	3
1	3	3.5	3	4
2	4	4	4	4
3	5	4.5	5	5
4	6	5	6	5
5	7	5.5	7	6
6	8	6	8	6
7	9	6.5	9	7
8	10	7	10	7
9	11	7.5	11	8
10	12	8	12	8

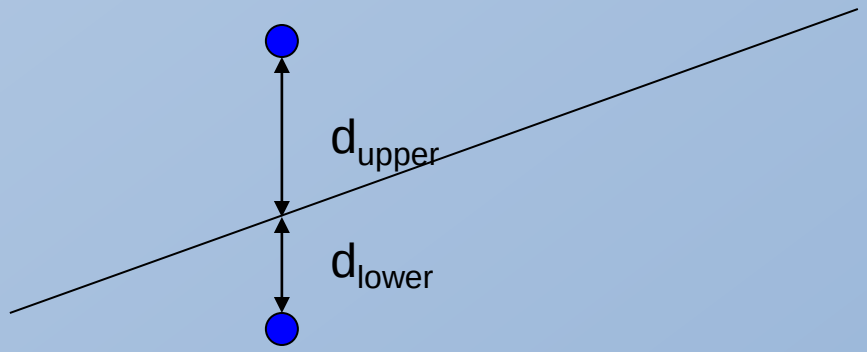
DDA Example



- … but floating point operations and rounding operations are expensive

Bresenham's Algorithm

- Uses only integer calculations
- Uses distance between ideal y-coordinate and the upper and lower pixel (assuming gentle slope)



Bresenham's Algorithm
General idea how Bresenham works

- Suppose that the line is gently sloping upwards from left to right.
- Start by coloring the left-most pixel.
- Then, for the next column (that is, for each x value), we have to figure out whether we color the same y or y+1.

Bresenham's Algorithm
General idea how Bresenham works

- How do we decide?

When going from one column to the next, add an error value. If the error value is more than 0.5, we should color y+1 and reset the error value. Otherwise, color y and accumulate the error value.

Bresenham's Algorithm
General idea how Bresenham works

- However, it seems like we're still using floating point

Solution, multiply both sides by 2 so that we use integer comparisons instead.

Bresenham's Algorithm

1. Input the two line endpoints and store left endpoint as (x_0,y_0)
2. Pre-calculate the values dx, dy, 2dy and 2dy - 2dx
3. Color pixel (x_0, y_0)
4. Let $P_0 = 2dy – dx$
5. At each x_k along the line, starting with k=0:

   If $P_k<0$, then

   the next point to plot is (X_{k+1}, Y_k),

   $P_{k+1} = P_k + 2dy$

   Else

   the next point to plot is (X_{k+1}, Y_{k+1}),

   $P_{k+1} = P_k + 2dy – 2dx$
6. Repeat Step-5 dx times

Bresenham's Algorithm

- Switch Point 0 and Point 1 if necessary
- If negative slope, reflect
- If steep slope, flip y and x

Q: In each step, how many floating point operations are there?
A: 0

Q: In each step, how many integer operations are there?
A: 3 or 4

Bresenham's Algorithm

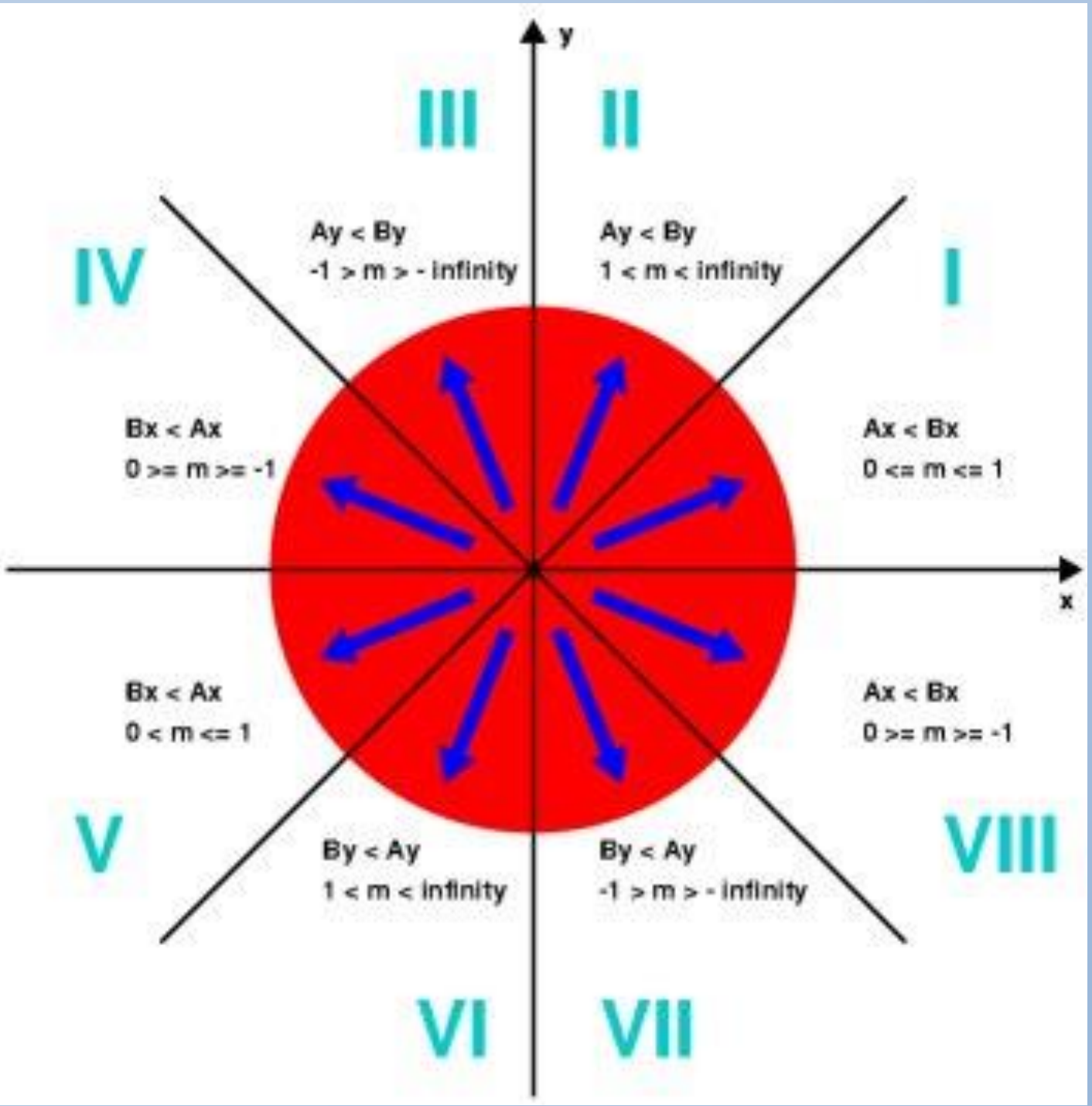
dx = 12 – 2 = 10
dy = 8 – 3 = 5
P_0 = 2dy – dx = 0

2dy = 10
2dy – 2dx = -10

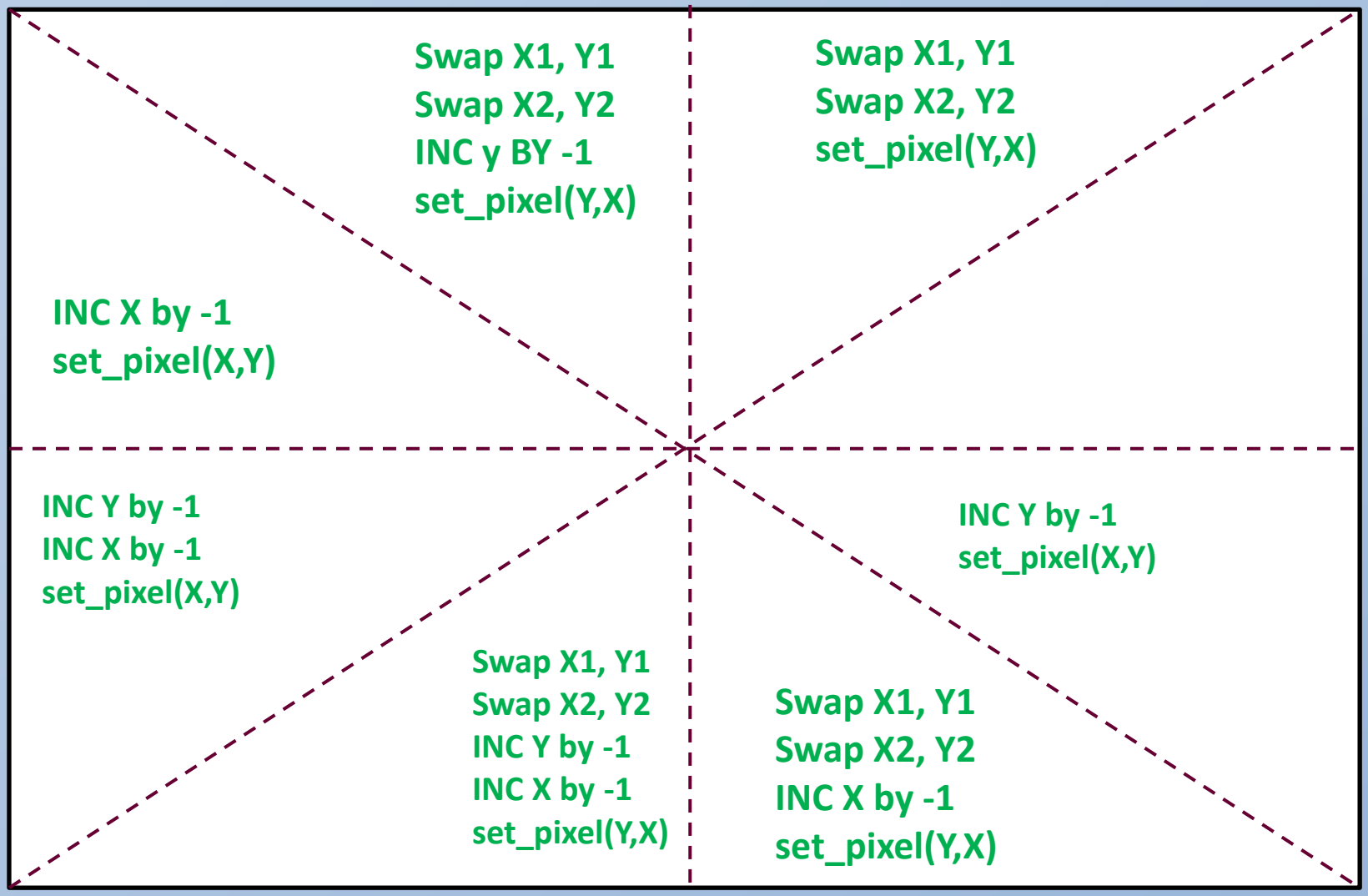
- Suppose we want to draw a line starting at pixel (2,3) and ending at pixel (12,8).
- What are the values of p0, dx and dy?
- What are the values of the variable p at each timestep?
- What are the pixels colored, according to Bresenham's algorithm?

t	p	P(x)	P(y)
0	0	2	3
1	-10	3	4
2	0	4	4
3	-10	5	5
4	0	6	5
5	-10	7	6
6	0	8	6
7	-10	9	7
8	0	10	7
9	-10	11	8
10	0	12	8

Bres


y
x
III
II
IV
I
V
VIII
VI
VII
Ay < By
-1 > m > - infinity
Ay < By
1 < m < infinity
Bx < Ax
0 >= m >= -1
Ax < Bx
0 <= m <= 1
Bx < Ax
0 < m <= 1
Ax < Bx
0 >= m >= -1
By < Ay
1 < m < infinity
By < Ay
-1 > m > - infinity

Bresenham's Algorithm



First Octant

First Point (32, 3) Second Point (48, 8)

$$\text{Slope} = \frac{8-3}{48-32} = 0.3125$$

Since $x_1 < x_2$ (32 < 48)

The line is in the first octant

```
dx = 16
dy = 5
2dy = 10
2dy-2dx = -22
P0 = 2dy-dx
```

t	P	X	Y
0	-6	32	3
1	4	33	3
2	-18	34	4
3	-8	35	4
4	2	36	4
5	-20	37	5
6	-10	38	5
7	0	39	5
8	-22	40	6
9	-12	41	6
10	-2	42	6
11	8	43	6
12	-14	44	7
13	-4	45	7
14	6	46	7
15	-16	47	8
16	-6	48	8

Second Octant

First Point (3, 32) Second Point (8, 48)

$$\text{Slope} = \frac{48-32}{8-3} = 3.2$$

Since $y_1 < y_2$ (32 < 48)

The line is in the second octant

- Switch x1 and y1 (32, 3)
- Switch x2 and y2 (48, 8)
- Apply Bressenham's algorithm
- Switch X and Y

```
dx = 16
dy = 5
2dy = 10
2dy-2dx = -22
P0 = 2dy-dx
```

t	P	X	Y
0	-6	32	3
1	4	33	3
2	-18	34	4
3	-8	35	4
4	2	36	4
5	-20	37	5
6	-10	38	5
7	0	39	5
8	-22	40	6
9	-12	41	6
10	-2	42	6
11	8	43	6
12	-14	44	7
13	-4	45	7
14	6	46	7
15	-16	47	8
16	-6	48	8

Y	X
3	32
3	33
4	34
4	35
4	36
5	37
5	38
5	39
6	40
6	41
6	42
6	43
7	44
7	45
7	46
8	47
8	48

Third Octant

First Point (8, 32) Second Point (3, 48)

$$\text{Slope} = \frac{48-32}{3-8} = -3.2$$

Since $y_1 < y_2$ (32 < 48)

The line is in the Third octant

- Swap x1 and y1 (32, 8)
- Swap x2 and y2 (48, 3)
- Set dy = -dy
- Decrement Y
- Switch X and Y

```
dx = 16
dy = -5   → dy = 5
2dy =  10
2dy-2dx = -22
P0 = 2dy - dx
```

t	P	X	Y
0	-6	32	8
1	4	33	8
2	-18	34	7
3	-8	35	7
4	2	36	7
5	-20	37	6
6	-10	38	6
7	0	39	6
8	-22	40	5
9	-12	41	5
10	-2	42	5
11	8	43	5
12	-14	44	4
13	-4	45	4
14	6	46	4
15	-16	47	3
16	-6	48	3

Y	X
8	32
8	33
7	34
7	35
7	36
6	37
6	38
6	39
5	40
5	41
5	42
5	43
4	44
4	45
4	46
3	47
3	48

Fourth Octant

First Point (48, 3) Second Point (32, 8)

$$\text{Slope} = \frac{8-3}{32-48} = -0.3125$$

Since $x_1 > x_2$ (48 > 32)

The line is in the Fourth octant

- Set dx = -dx
- Decrement x

- dx = -16 → dx = 16
- dy = 5
- 2dy = 10
- 2dy-2dx = -22
- P0 = 2dy-dx

t	P	X	Y
0	-6	48	3
1	4	47	3
2	-18	46	4
3	-8	45	4
4	2	44	4
5	-20	43	5
6	-10	42	5
7	0	41	5
8	-22	40	6
9	-12	39	6
10	-2	38	6
11	8	37	6
12	-14	36	7
13	-4	35	7
14	6	34	7
15	-16	33	8
16	-6	32	8

Fifth Octant

First Point (48, 8) Second Point (32, 3)

$$\text{Slope} = \frac{3-8}{32-48} = 0.3125$$

Since $x_1 > x_2$ (48 > 32)

The line is in the Fifth octant

- Set dx = -dx
- Set dy = -dy
- Decrement x
- Decrement y

- dx = -16 → dx = 16
- dy = -5 → dy = 5
- 2dy = 10
- 2dy-2dx = -22
- P0 = 2dy - dx

t	P	X	Y
0	-6	48	8
1	4	47	8
2	-18	46	7
3	-8	45	7
4	2	44	7
5	-20	43	6
6	-10	42	6
7	0	41	6
8	-22	40	5
9	-12	39	5
10	-2	38	5
11	8	37	5
12	-14	36	4
13	-4	35	4
14	6	34	4
15	-16	33	3
16	-6	32	3

Sixth Octant

First Point (8, 48) Second Point (3, 32)

$$\text{Slope} = \frac{32-48}{3-8} = 3.2$$

Since $y_1 > y_2$ (48 > 32)

The line is in the sixth octant

- Swap x1 and y1 (48, 8)
- Swap x2 and y2 (32, 3)
- Set dx = -dx
- Set dy = -dy
- Decrement x
- Decrement y
- Swap X and Y

dx = -16 → dx = 16
dy = -5 → dy = 5
2dy = 10
2dy-2dx = -22
P0 = 2dy-dx

t	P	X	Y
0	-6	48	8
1	4	47	8
2	-18	46	7
3	-8	45	7
4	2	44	7
5	-20	43	6
6	-10	42	6
7	0	41	6
8	-22	40	5
9	-12	39	5
10	-2	38	5
11	8	37	5
12	-14	36	4
13	-4	35	4
14	6	34	4
15	-16	33	3
16	-6	32	3

Y	X
8	48
8	47
7	46
7	45
7	44
6	43
6	42
6	41
5	40
5	39
5	38
5	37
4	36
4	35
4	34
3	33
3	32

Seventh Octant

First Point (3, 48)

Second Point (8, 32)

$$\text{Slope} = \frac{32-48}{8-3} = -3.2$$

Since $y_1 > y_2$ (48 > 32)

The line is in the seventh octant

- Swap x1 and y1 (48, 3)
- Swap x2 and y2 (32, 8)
- Set dx = -dx
- Decrement x
- Swap X and Y

```
dx = -16  → dx = 16
dy = 5
2dy = 10
2dy-2dx = -22
P0 = 2dy-dx
```

t	P	X	Y
0	-6	48	3
1	4	47	3
2	-18	46	4
3	-8	45	4
4	2	44	4
5	-20	43	5
6	-10	42	5
7	0	41	5
8	-22	40	6
9	-12	39	6
10	-2	38	6
11	8	37	6
12	-14	36	7
13	-4	35	7
14	6	34	7
15	-16	33	8
16	-6	32	8

Y	X
3	48
3	47
4	46
4	45
4	44
5	43
5	42
5	41
6	40
6	39
6	38
6	37
7	36
7	35
7	34
8	33
8	32

Eighth Octant

First Point (32, 8) Second Point (48, 3)

$$\text{Slope} = \frac{3-8}{48-32} = -0.3125$$

Since $x_1 < x_2$ (32 < 48)

The line is in the Eighth octant

- Set dy = -dy
- Decrement y

- dx = 16
- dy = -5 → dy = 5
- 2dy = 10
- 2dy-2dx =-22
- P0 = 2dy-dx

t	P	X	Y
0	-6	32	8
1	4	33	8
2	-18	34	7
3	-8	35	7
4	2	36	7
5	-20	37	6
6	-10	38	6
7	0	39	6
8	-22	40	5
9	-12	41	5
10	-2	42	5
11	8	43	5
12	-14	44	4
13	-4	45	4
14	6	46	4
15	-16	47	3
16	-6	48	3

Difference Between DDA Line Drawing Algorithm and Bresenhams Line Drawing Algorithm

	Digital Differential Analyzer Line Drawing Algorithm	Bresenhams Line Drawing Algorithm
Arithmetic	DDA algorithm uses floating points i.e.Real Arithmetic.	Bresenhams algorithm uses fixed points i.e. Integer Arithmetic.
Operations	DDA algorithm uses multiplication anddivision in its operations.	Bresenhams algorithm uses only subtraction and addition in its operations.
Speed	DDA algorithm is rather slowly than Bresenhams algorithm in line drawing because it uses real arithmetic (floating-point operations).	Bresenhams algorithm is faster than DDA algorithm in line drawing because it performs only addition and subtraction in its calculation and uses only integer arithmetic so it runs significantly faster.
Accuracy & Efficiency	DDA algorithm is not as accurate and efficient as Bresenham algorithm.	Bresenhams algorithm is more efficient and much accurate than DDA algorithm.
Drawing	DDA algorithm can draw circles and curves but that are not as accurate as Bresenhams algorithm.	Bresenhams algorithm can draw circles and curves with much more accuracy than DDA algorithm.
Round Off	DDA algorithm round off the coordinates to integer that is nearest to the line.	Bresenhams algorithm does not round off but takes the incremental value in its operation.
Expensive	DDA algorithm uses an enormous number of floating-point multiplications so it is expensive.	Bresenhams algorithm is less expensive than DDA algorithm as it uses only addition and subtraction.

Temp

- Last update on 21-October 2016