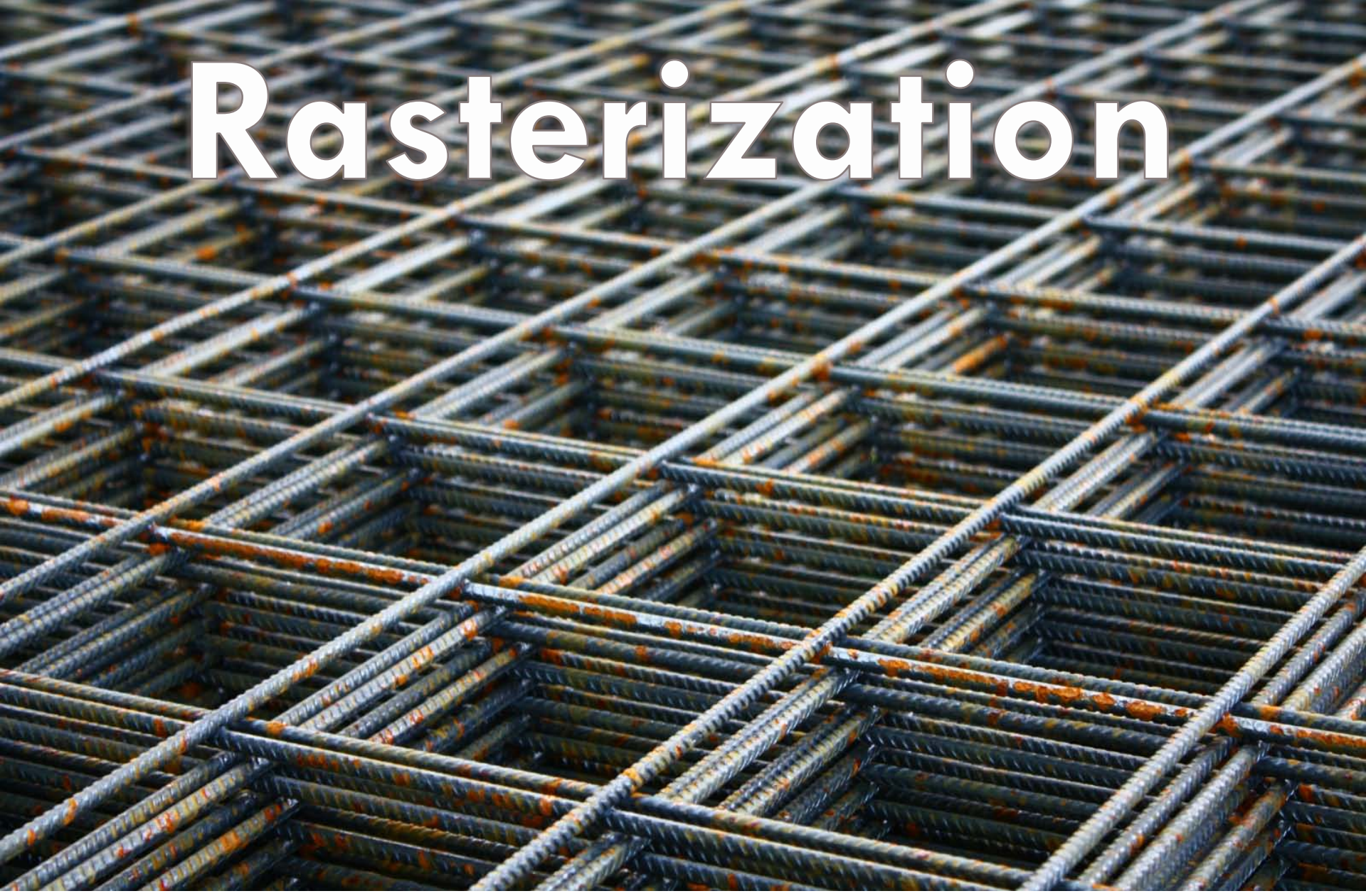
Rasterization



LESSON 8

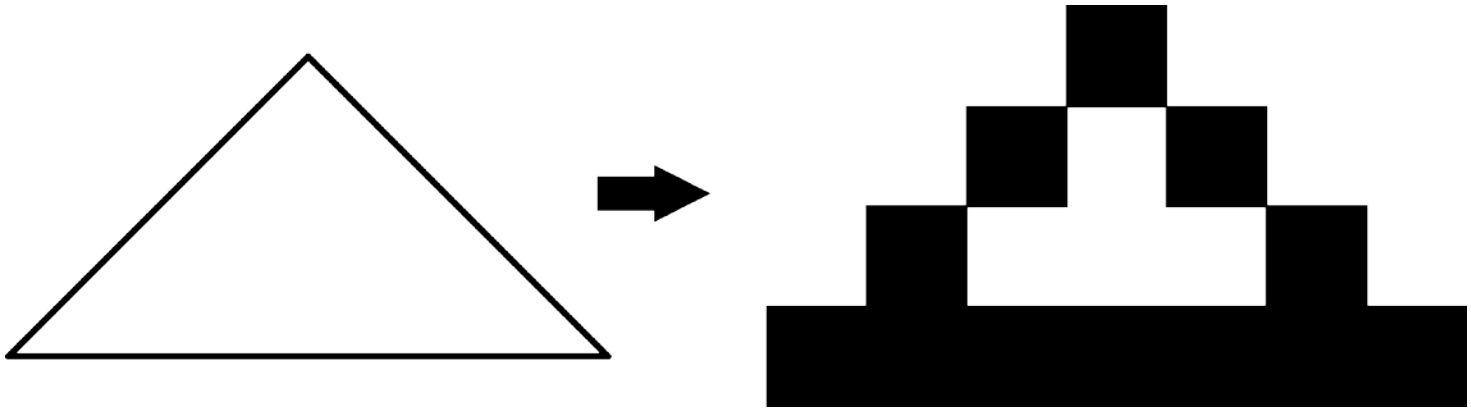
Computer Graphics 1

Outline

- What is rasterization
- Line rasterization
- Curve rasterization
- Polygon rasterization
- Filling algorithms

Rasterization

- One of the most important tasks in computer graphics
- Mapping from continuous space to discrete space
- Converting vector graphics to raster image
- Represent object by a set of pixels
- Output on display or printer (write to frame buffer)

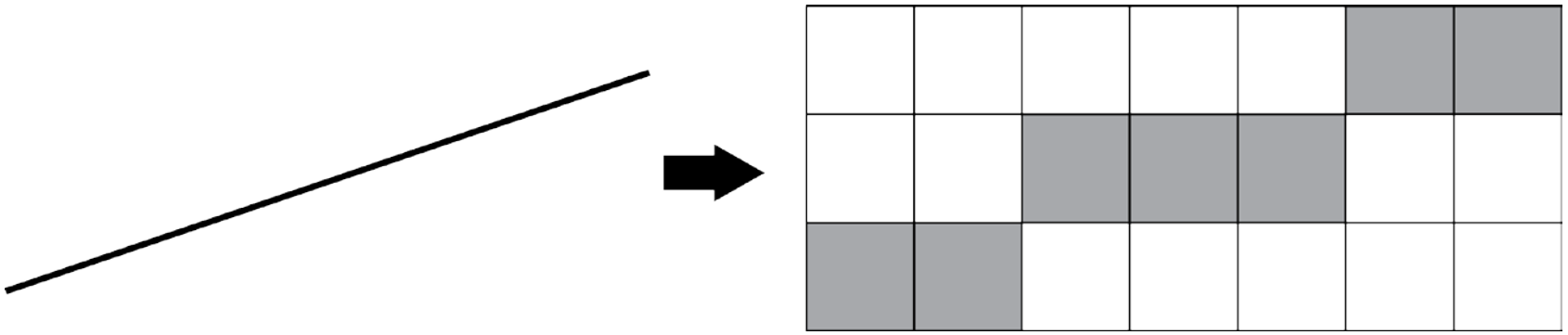


Line Rasterization

DDA, Bresenham's algorithm, midpoint algorithm

Line Rasterization

- Very frequent
- Basic operation
- Displaying of complex object can be reduced to drawing a lot of lines
- Generate satisfactory set of points

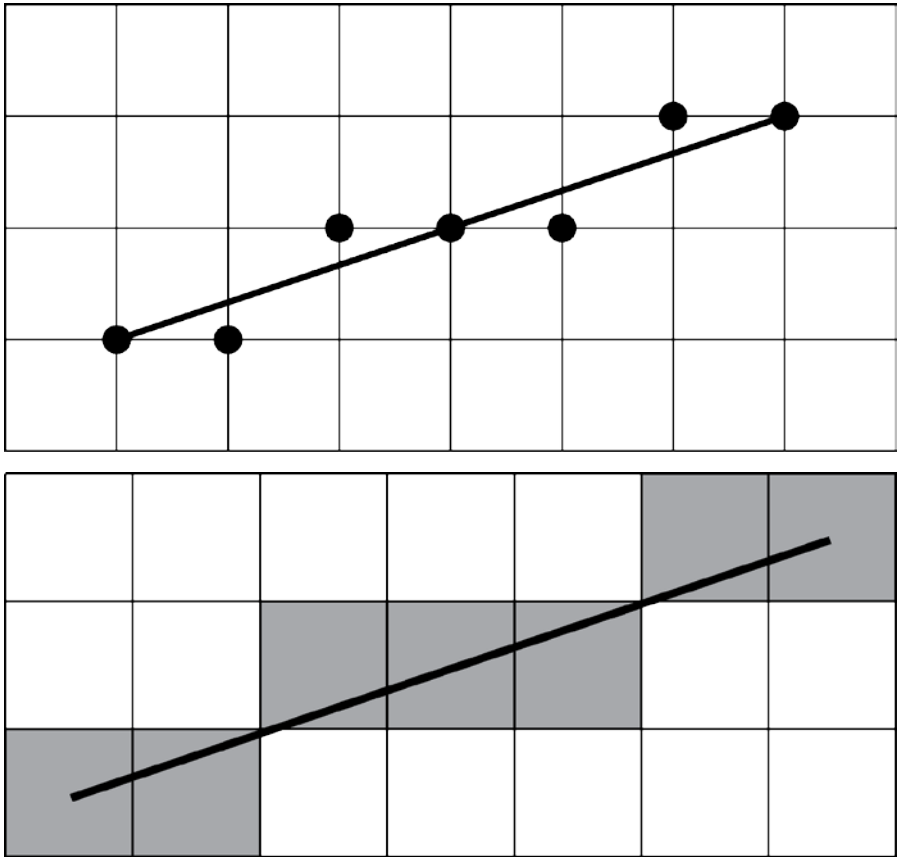


Line Rasterization - Criteria

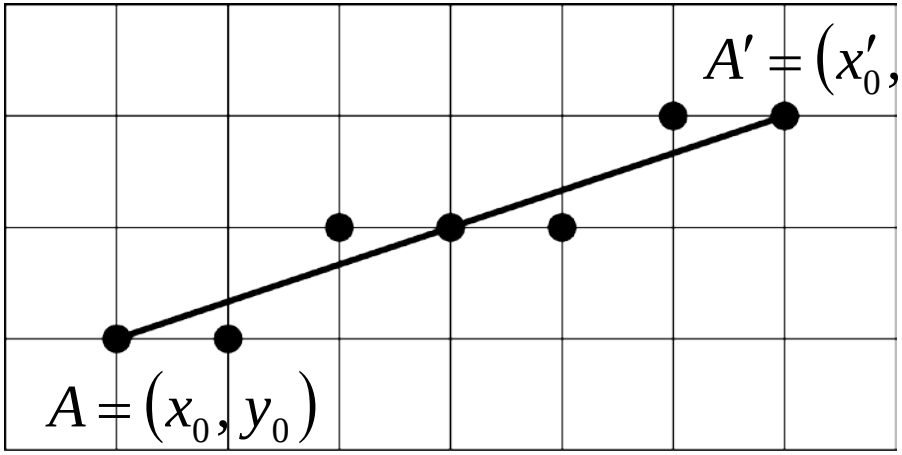
- Visually as straight as possible
- Accurate end and start
  - Avoid gaps between lines
- Even visual thickness
  - Constant density
  - Thickness independent of slope and length
- Fast rasterization

DDA

- Digital differential analyzer
- Linear interpolation of values between endpoints



DDA



$$A = (x_0, y_0)$$
$$A' = (x_0', y_0')$$
$$\Delta x = x_0' - x_0$$
$$\Delta y = y_0' - y_0$$

$$y = mx + b$$
$$m = \frac{\Delta y}{\Delta x} = \frac{dy}{dx}$$
$$dx = 1 \Rightarrow dy = mdx = m$$

DDA - Algorithm

- Draw (x_0, y_0)
- Iteration:

$$x_{i+1} = x_i + 1$$

$$y_{i+1} = y_i + m$$

- Draw $(x_{i+1}, round(y_{i+1}))$

DDA - Algorithm

- Draw (x_0, y_0)
- Iteration:

$$x_{i+1} = x_i + 1$$

$$y_{i+1} = y_i + m$$

- Draw $(x_{i+1}, round(y_{i+1}))$

- What if $\Delta y > \Delta x$?

DDA - Algorithm(2)

$$\Delta y > \Delta x \Rightarrow m > 1$$

- Solution: $dy = 1$

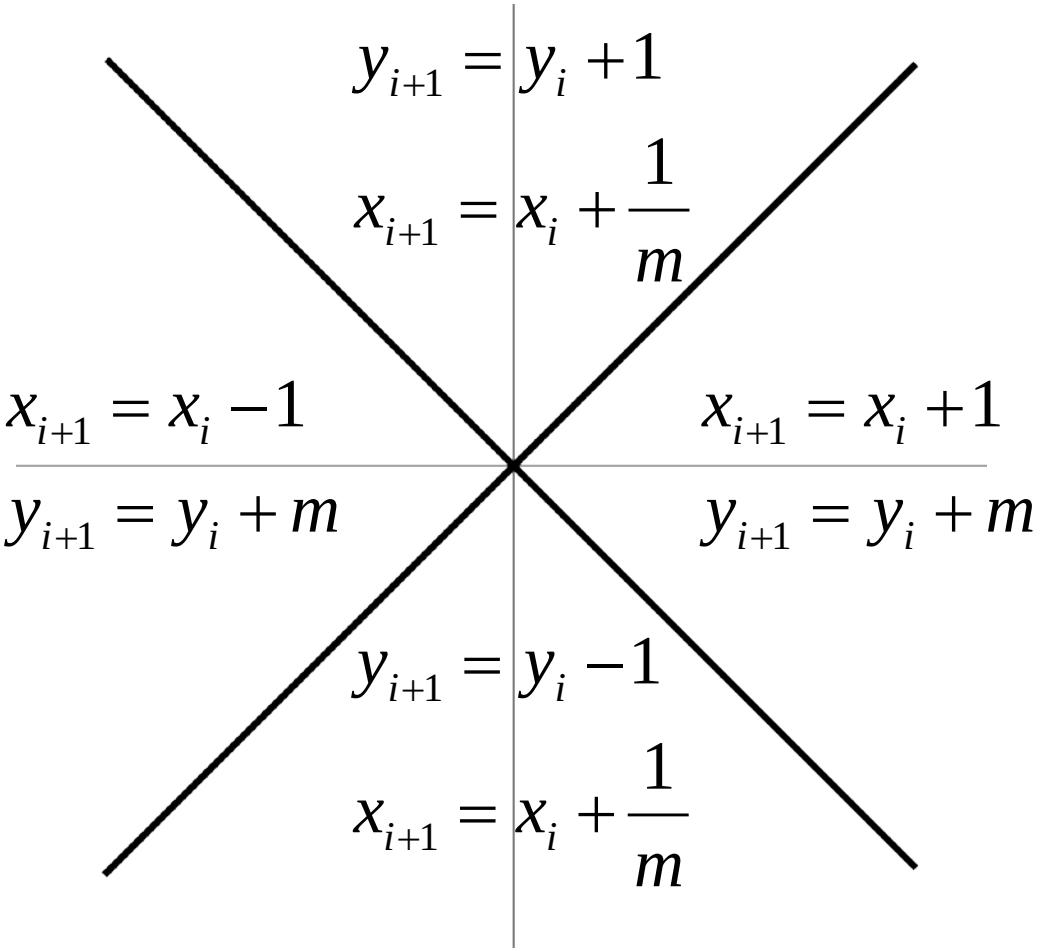
$$dx = \frac{dy}{m} = \frac{1}{m}$$

- Iteration: $y_{i+1} = y_i + 1$

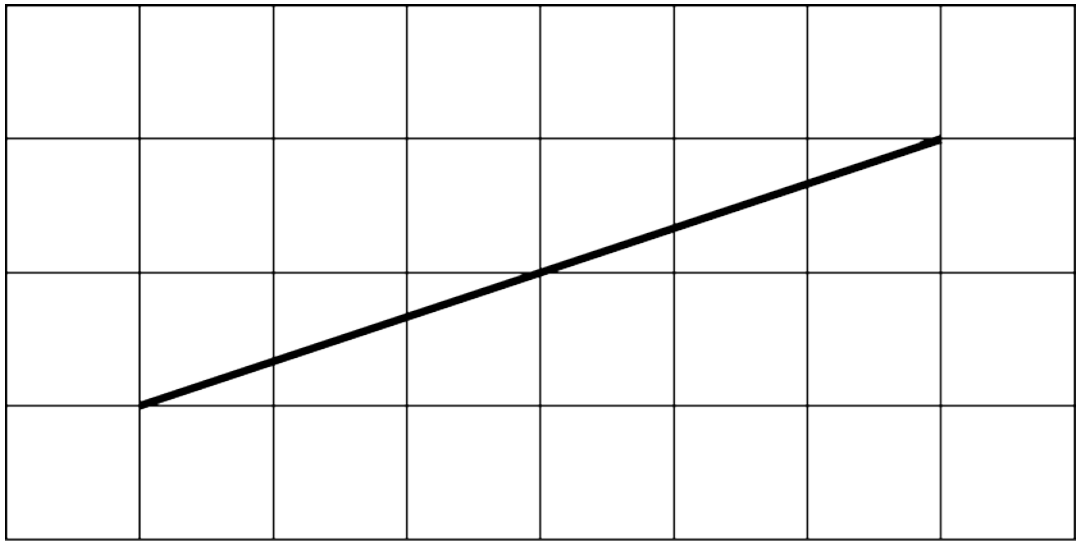
$$x_{i+1} = x_i + \frac{1}{m}$$

- Draw $\left(round\left(x_{i+1}\right), y_{i+1}\right)$

Line Rasterization - Quadrants



DDA - Example



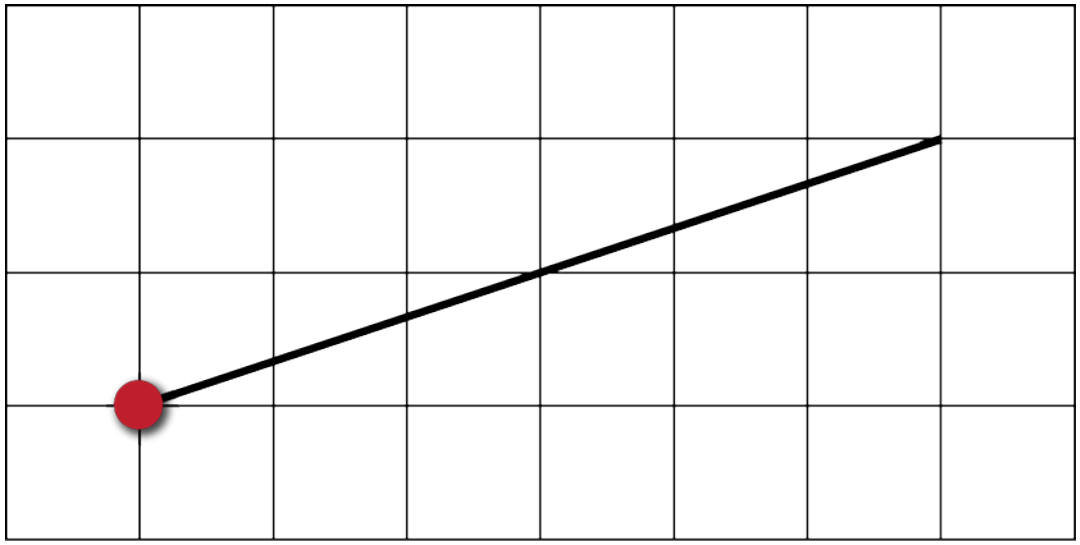
i	(x_i, y_i)	draw	error

$$A = (1,1) \qquad A' = (7,3)$$

$$\Delta x = 6 \qquad \Delta y = 2$$

$$m = 1/3 \qquad x_{i+1} = x_i$$

DDA - Example



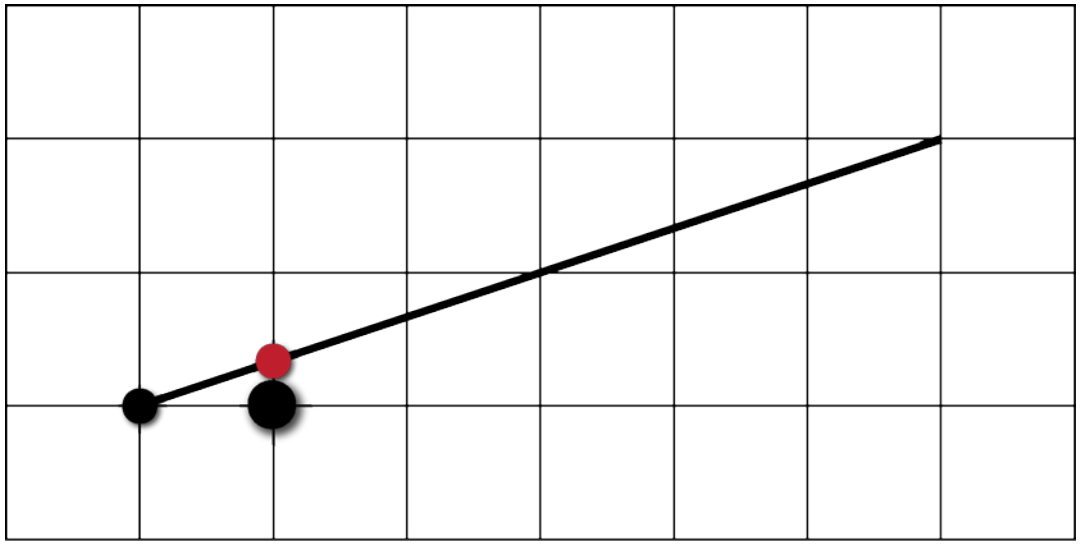
i	(x_i, y_i)	draw	error
0	(1,1)	(1,1)	0

$$A = (1,1) \qquad A' = (7,3)$$

$$\Delta x = 6 \qquad \Delta y = 2$$

$$m = 1/3 \qquad x_{i+1} = x_i$$

DDA - Example



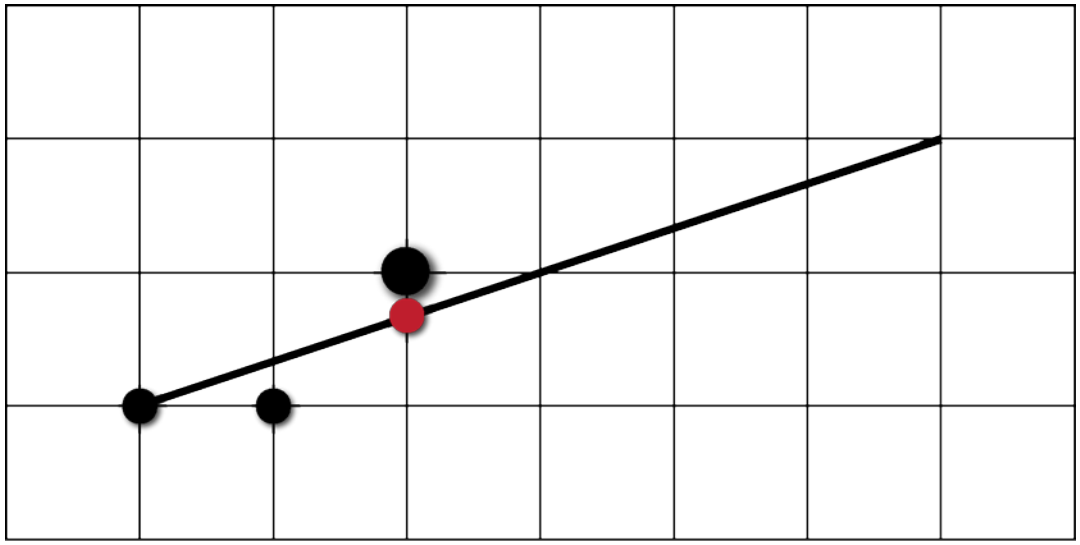
i	(x_i, y_i)	draw	error
0	(1,1)	(1,1)	0
1	(2,1 1/3)	(2,1)	1/3

$$A = (1,1) \qquad A' = (7,3)$$

$$\Delta x = 6 \qquad \Delta y = 2$$

$$m = 1/3 \qquad x_{i+1} = x_i$$

DDA - Example



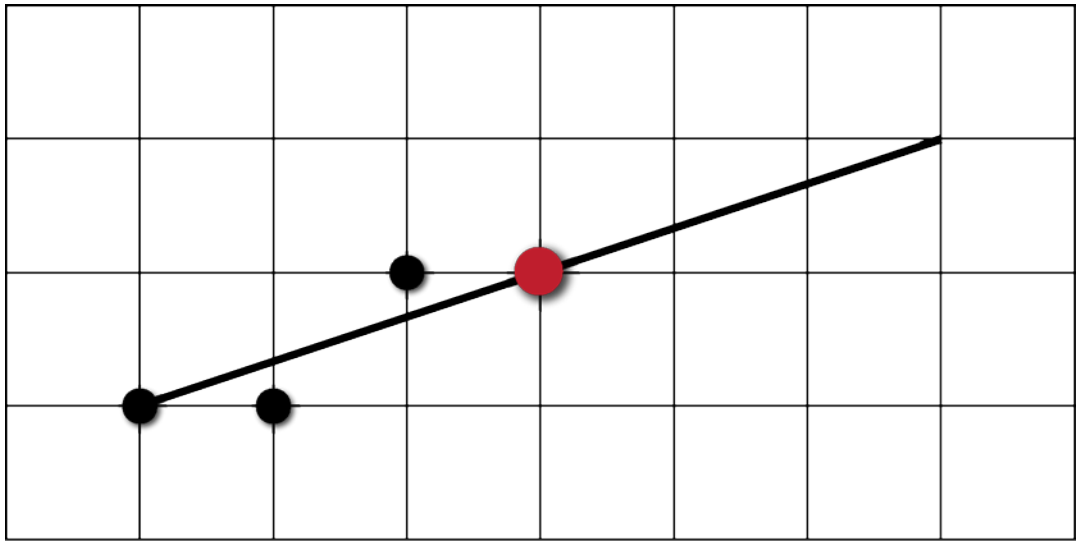
i	(x_i, y_i)	draw	error
0	(1,1)	(1,1)	0
1	(2,1 1/3)	(2,1)	1/3
2	(3,1 2/3)	(3,2)	-1/3

$$A = (1,1) \qquad A' = (7,3)$$

$$\Delta x = 6 \qquad \Delta y = 2$$

$$m = 1/3 \qquad x_{i+1} = x_i$$

DDA - Example



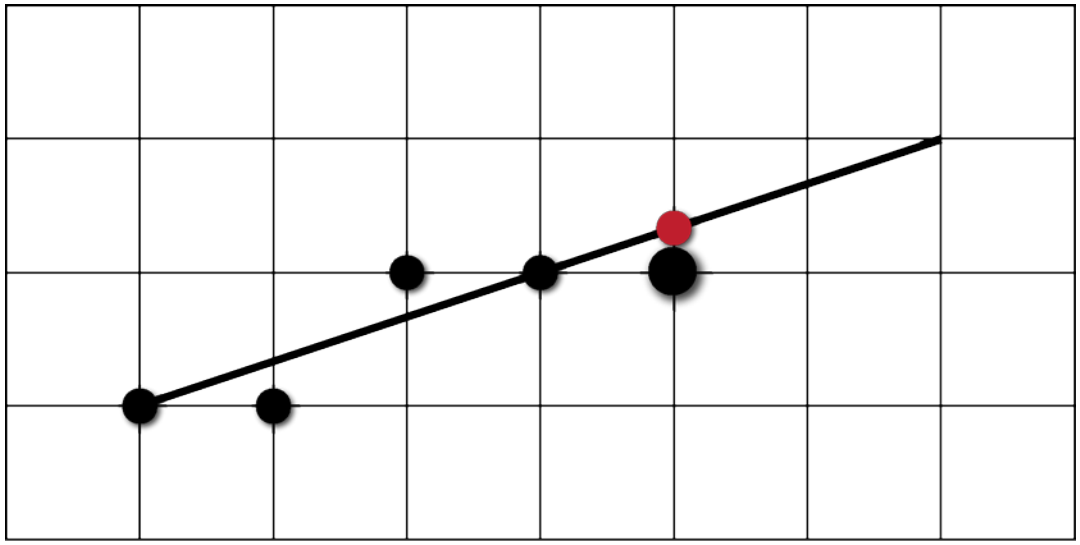
i	(x_i, y_i)	draw	error
0	(1,1)	(1,1)	0
1	(2,1 1/3)	(2,1)	1/3
2	(3,1 2/3)	(3,2)	-1/3
3	(4,2)	(4,2)	0

$$A = (1,1) \qquad A' = (7,3)$$

$$\Delta x = 6 \qquad \Delta y = 2$$

$$m = 1/3 \qquad x_{i+1} = x_i$$

DDA - Example



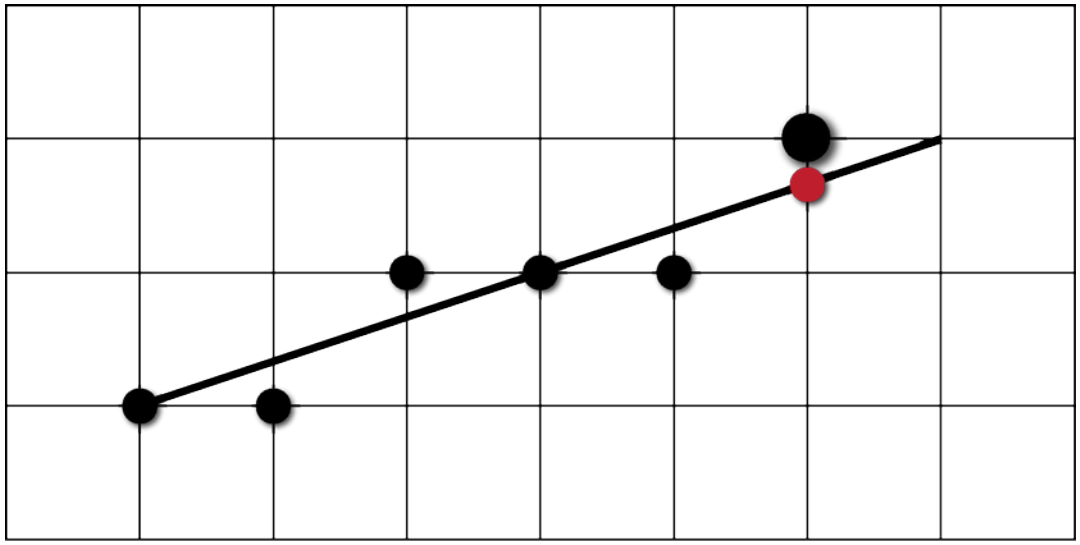
i	(x_i, y_i)	draw	error
0	(1,1)	(1,1)	0
1	(2,1 1/3)	(2,1)	1/3
2	(3,1 2/3)	(3,2)	-1/3
3	(4,2)	(4,2)	0
4	(5,2 1/3)	(5,2)	1/3

$$A = (1,1) \qquad A' = (7,3)$$

$$\Delta x = 6 \qquad \Delta y = 2$$

$$m = 1/3 \qquad x_{i+1} = x_i$$

DDA - Example



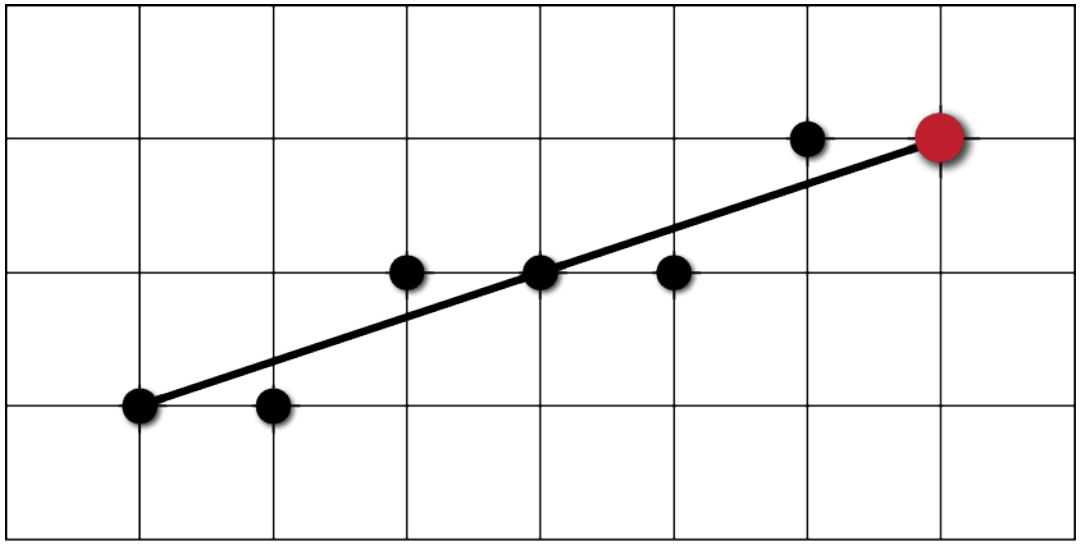
i	(x_i, y_i)	draw	error
0	(1,1)	(1,1)	0
1	(2,1 1/3)	(2,1)	1/3
2	(3,1 2/3)	(3,2)	-1/3
3	(4,2)	(4,2)	0
4	(5,2 1/3)	(5,2)	1/3
5	(6,2 2/3)	(6,3)	-1/3

$$A = (1,1) \qquad A' = (7,3)$$

$$\Delta x = 6 \qquad \Delta y = 2$$

$$m = 1/3 \qquad x_{i+1} = x_i$$

DDA - Example



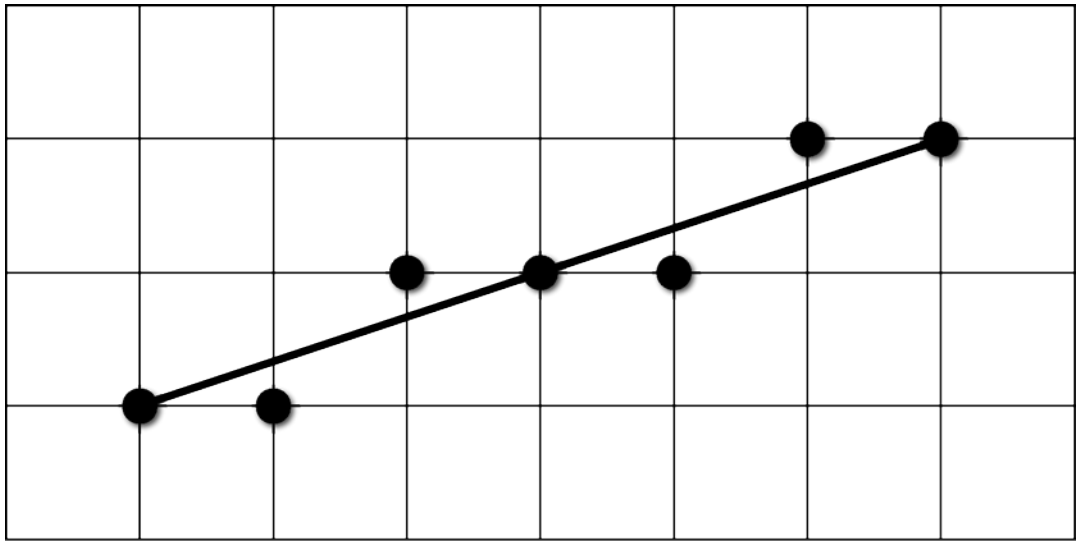
$$A = (1,1) \qquad A' = (7,3)$$

$$\Delta x = 6 \qquad \Delta y = 2$$

$$m = 1/3 \qquad x_{i+1} = x_i$$

i	(x_i, y_i)	draw	error
0	(1,1)	(1,1)	0
1	(2,1 1/3)	(2,1)	1/3
2	(3,1 2/3)	(3,2)	-1/3
3	(4,2)	(4,2)	0
4	(5,2 1/3)	(5,2)	1/3
5	(6,2 2/3)	(6,3)	-1/3
6	(7,3)	(7,3)	0

DDA - Example



$A = (1,1) \qquad A' = (7,3)$

$\Delta x = 6 \qquad \Delta y = 2$

$m = 1/3 \qquad x_{i+1} = x_i$

i	(x_i, y_i)	draw	error
0	(1,1)	(1,1)	0
1	(2,1 1/3)	(2,1)	1/3
2	(3,1 2/3)	(3,2)	-1/3
3	(4,2)	(4,2)	0
4	(5,2 1/3)	(5,2)	1/3
5	(6,2 2/3)	(6,3)	-1/3
6	(7,3)	(7,3)	0

Problem with DDA

- Floating point arithmetic
  - Computationally expensive (problem of old computers)
  - Inaccuracy

- Decimal

  0.1

- Binary

  e = - 4

  s = 1100110011001100110011001100110011...

- Decimal

  0.100000001490116119384765625

Bresenham's Algorithm

- Fast drawing algorithm
- Integer arithmetic
- Following the drawn point (x_i, y_i) while storing the error $\frac{p_i}{q_i}$

- q_i is constant ($q_{i+1} = q_i$)
- Iteration: $(x_i, y_i), p_i \rightarrow (x_{i+1}, y_{i+1}), p_{i+1}$

Bresenham -Iteration

$$(x_i, y_i), p_i \rightarrow (x_{i+1}, y_{i+1}), p_{i+1}$$

$$x_{i+1} = x_i + 1$$

$$p_{i+1} = p_i + \Delta y$$

$$\frac{p_{i+1}}{q_{i+1}} = \frac{p_i}{q_i} + \frac{\Delta y}{\Delta x} = \frac{p_i + \Delta y}{q_i}$$

- If $p_i > \frac{q_i}{2} = \frac{\Delta x}{2}$ then $y_{i+1} = y_i + 1$ $p_{i+1} = p_i - \Delta x$

  else $y_{i+1} = y_i$

Bresenham -Iteration

$$(x_i, y_i), p_i \rightarrow (x_{i+1}, y_{i+1}), p_{i+1}$$

$$x_{i+1} = x_i + 1$$

$$p_{i+1} = p_i + \Delta y \qquad \frac{p_{i+1}}{q_{i+1}} = \frac{p_i}{q_i} + \frac{\Delta y}{\Delta x} = \frac{p_i + \Delta y}{q_i}$$

- If $p_i > \frac{q_i}{2} = \frac{\Delta x}{2}$ then $y_{i+1} = y_i + 1 \quad p_{i+1} = p_i - \Delta x$

  else $y_{i+1} = y_i$

- Problem: $\frac{\Delta x}{2}$ (integer arithmetic)

Bresenham -Iteration

$$(x_i, y_i), p_i \rightarrow (x_{i+1}, y_{i+1}), p_{i+1}$$

$$x_{i+1} = x_i + 1 \qquad \frac{p_{i+1}}{q_{i+1}} = \frac{p_i}{q_i} + \frac{\Delta y}{\Delta x} = \frac{p_i + \Delta y}{q_i}$$

$$p_{i+1} = p_i + \Delta y$$

- If $p_i > \frac{q_i}{2} = \frac{\Delta x}{2}$ then $y_{i+1} = y_i + 1 \quad p_{i+1} = p_i - \Delta x$

  else $y_{i+1} = y_i$

- Problem: $\frac{\Delta x}{2}$ (integer arithmetic)
- Solution: multiply the whole computation by 2

Bresenham – Correction (1)

- Inicialization: (x_0, y_0) $p_0 = 0$

$$x_{i+1} = x_i + 1$$

$$p_{i+1} = p_i + 2\Delta y$$

- If $p_i > \Delta x$ then $y_{i+1} = y_i + 1$ $p_{i+1} = p_i - 2\Delta x$

  else $y_{i+1} = y_i$

Bresenham – Correction (1)

- Inicialization: (x_0, y_0) $p_0 = 0$

$$x_{i+1} = x_i + 1$$

$$p_{i+1} = p_i + 2\Delta y$$

- If $p_i > \Delta x$ then $y_{i+1} = y_i + 1$ $p_{i+1} = p_i - 2\Delta x$

  else $y_{i+1} = y_i$

- Problem: double modification of p_{i+1}

Bresenham – Correction (1)

- Inicialization: (x_0, y_0) $p_0 = 0$

$$x_{i+1} = x_i + 1$$

$$p_{i+1} = p_i + 2\Delta y$$

- If $p_i > \Delta x$ then $y_{i+1} = y_i + 1$ $p_{i+1} = p_i - 2\Delta x$

  else $y_{i+1} = y_i$

- Problem: double modification of p_{i+1}
- Solution: move the modification to the previous step

Bresenham – Correction (2)

- Inicialization: (x_0, y_0) $p_0 = 2\Delta y$

$$x_{i+1} = x_i + 1$$

- If $p_i > \Delta x$ then $y_{i+1} = y_i + 1$ $\quad p_{i+1} = p_i + 2\Delta y - 2\Delta x$

  else $y_{i+1} = y_i$ $\quad p_{i+1} = p_i + 2\Delta y$

Bresenham – Correction (2)

- Inicialization: (x_0, y_0) $p_0 = 2\Delta y$

$$x_{i+1} = x_i + 1$$

- If $p_i > \Delta x$ then $y_{i+1} = y_i + 1$ $\quad p_{i+1} = p_i + 2\Delta y - 2\Delta x$

  else $y_{i+1} = y_i$ $\quad p_{i+1} = p_i + 2\Delta y$

- Final correction: compare p_{i+1} with 0

Bresenham – Final Corrections

- Inicialization: (x_0, y_0) $p_0 = 2\Delta y - \Delta x$

$$x_{i+1} = x_i + 1$$

- If $p_i > 0$ then $y_{i+1} = y_i + 1$ $p_{i+1} = p_i + 2\Delta y - 2\Delta x$

  else $y_{i+1} = y_i$ $p_{i+1} = p_i + 2\Delta y$

Bresenham – Final Algorithm

- Inicialization: (x_0, y_0) $p_0 = 2\Delta y - \Delta x$

$$x_{i+1} = x_i + 1$$

- If $p_i > 0$ then $y_{i+1} = y_i + 1$ $p_{i+1} = p_i + 2\Delta y - 2\Delta x$

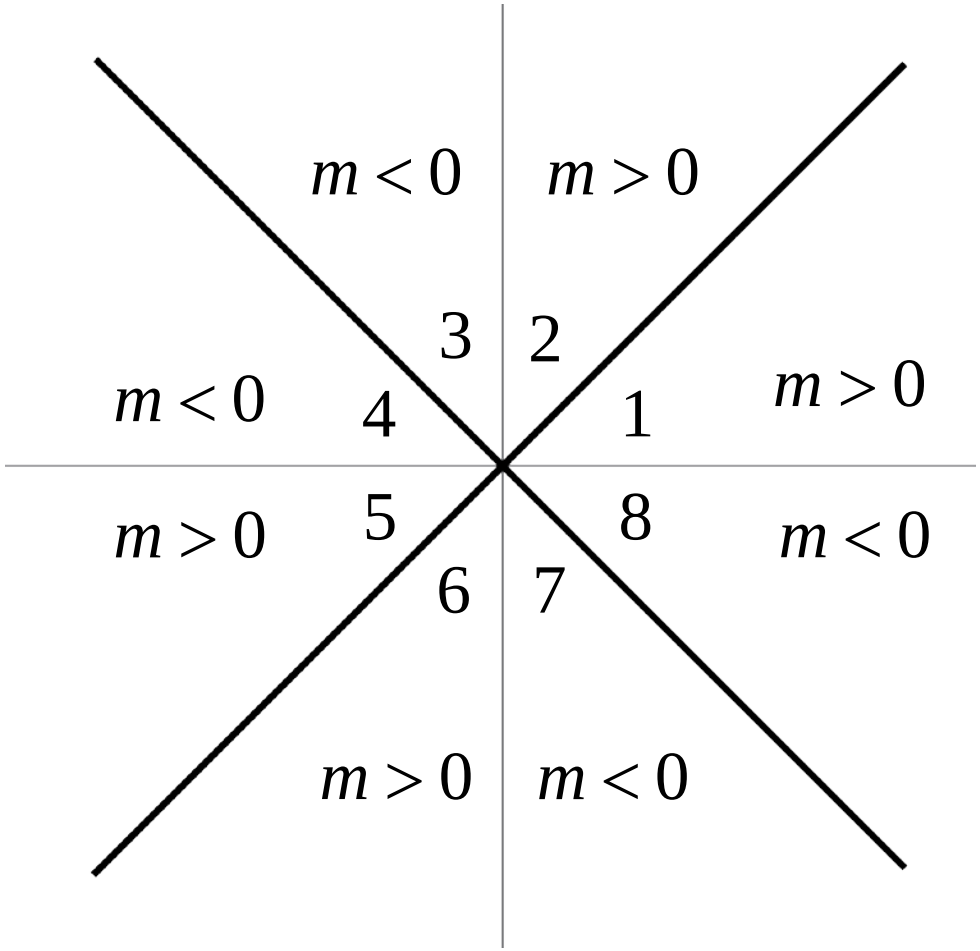
  else $y_{i+1} = y_i$ $p_{i+1} = p_i + 2\Delta y$

- Precompute: $2\Delta y - \Delta x = p_0$

  $2\Delta y$

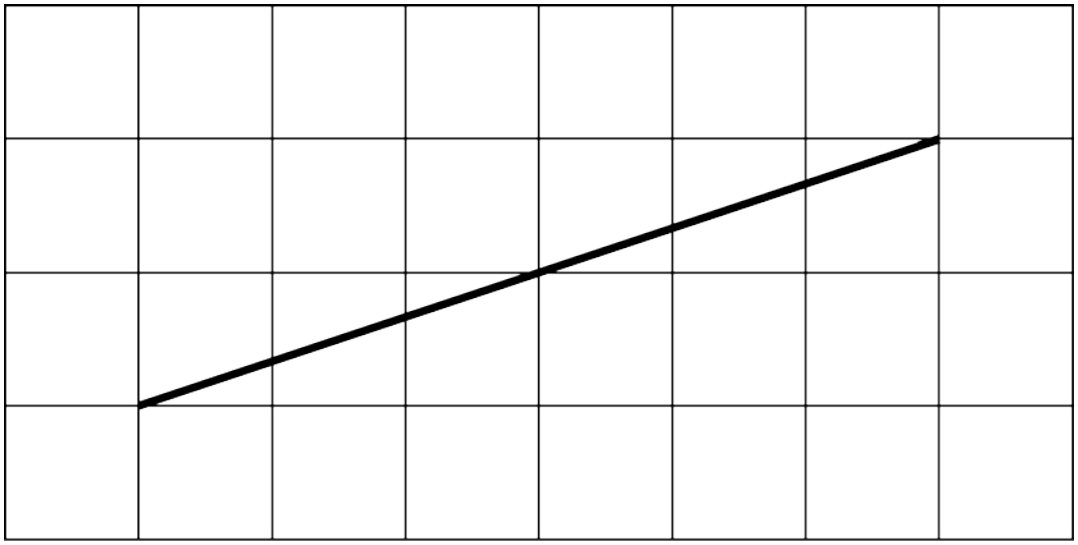
  $2\Delta y - 2\Delta x$

Bresenham - Octants



- Previous algorithm works only for the first octant

Bresenham - Example



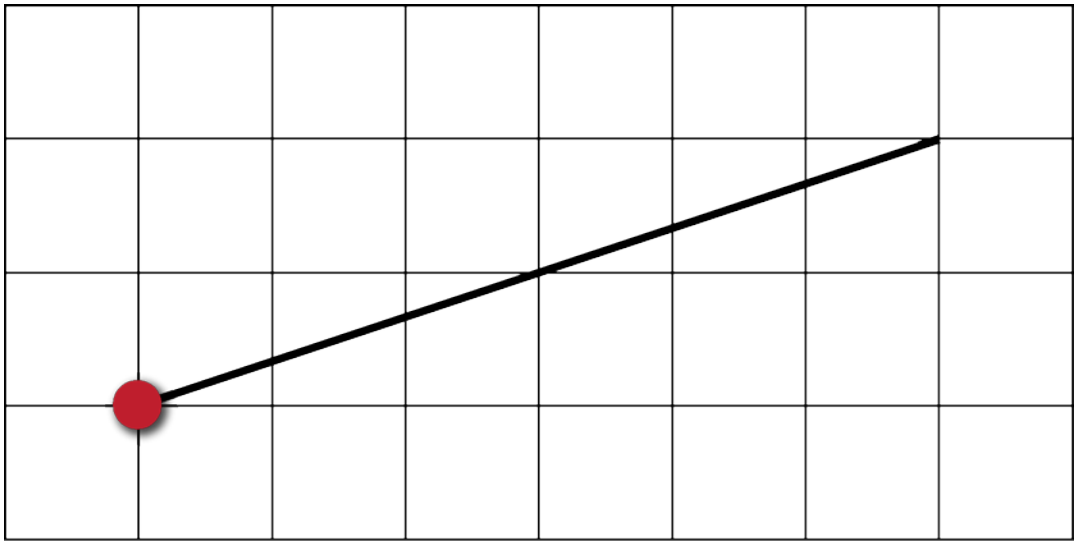
i	draw	p_i

$A = (1,1) \qquad A' = (7,3)$

$\Delta x = 6 \qquad \Delta y = 2$

$2\Delta y = 4 \qquad 2\Delta y - 2\Delta x = -8 \qquad 2\Delta y - \Delta x = -2$

Bresenham - Example



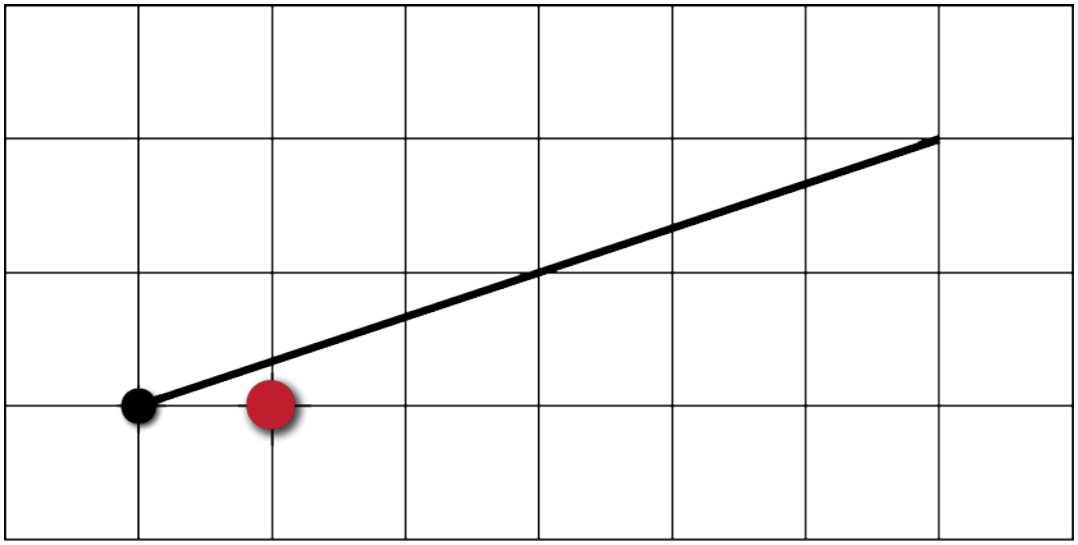
i	draw	p_i
0	(1,1)	-2

$A = (1,1) \qquad A' = (7,3)$

$\Delta x = 6 \qquad \Delta y = 2$

$2\Delta y = 4 \qquad 2\Delta y - 2\Delta x = -8 \qquad 2\Delta y - \Delta x = -2$

Bresenham - Example



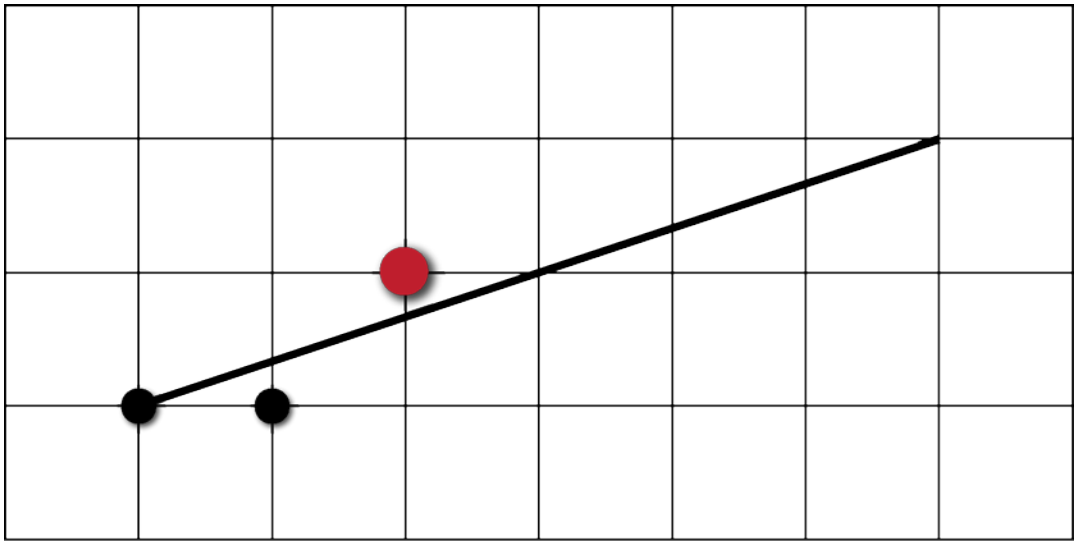
i	draw	p_i
0	(1,1)	-2
1	(2,1)	2

$A = (1,1) \qquad A' = (7,3)$

$\Delta x = 6 \qquad \Delta y = 2$

$2\Delta y = 4 \qquad 2\Delta y - 2\Delta x = -8 \qquad 2\Delta y - \Delta x = -2$

Bresenham - Example



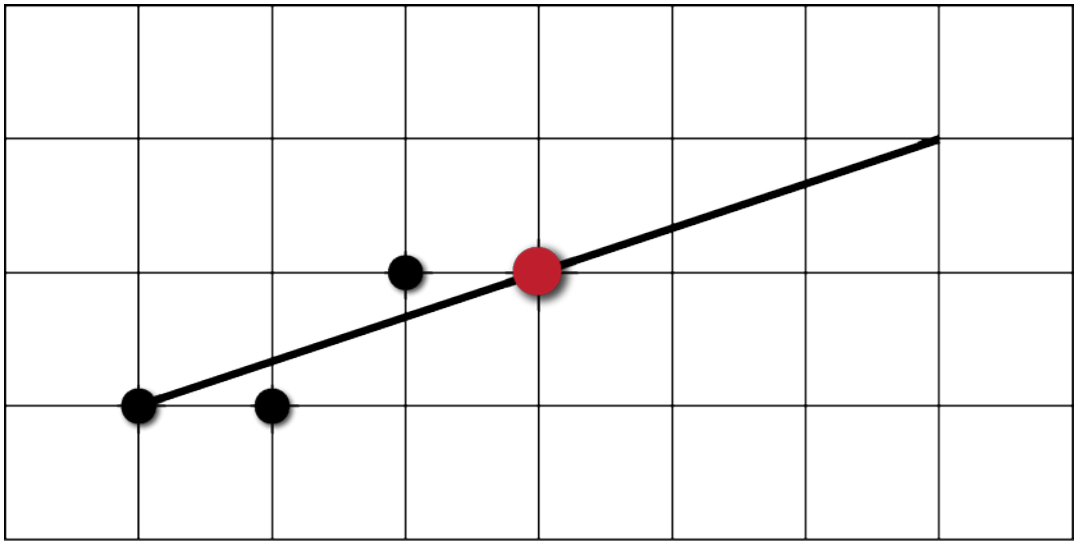
i	draw	p_i
0	(1,1)	-2
1	(2,1)	2
2	(3,2)	-6

$A = (1,1) \qquad A' = (7,3)$

$\Delta x = 6 \qquad \Delta y = 2$

$2\Delta y = 4 \qquad 2\Delta y - 2\Delta x = -8 \qquad 2\Delta y - \Delta x = -2$

Bresenham - Example



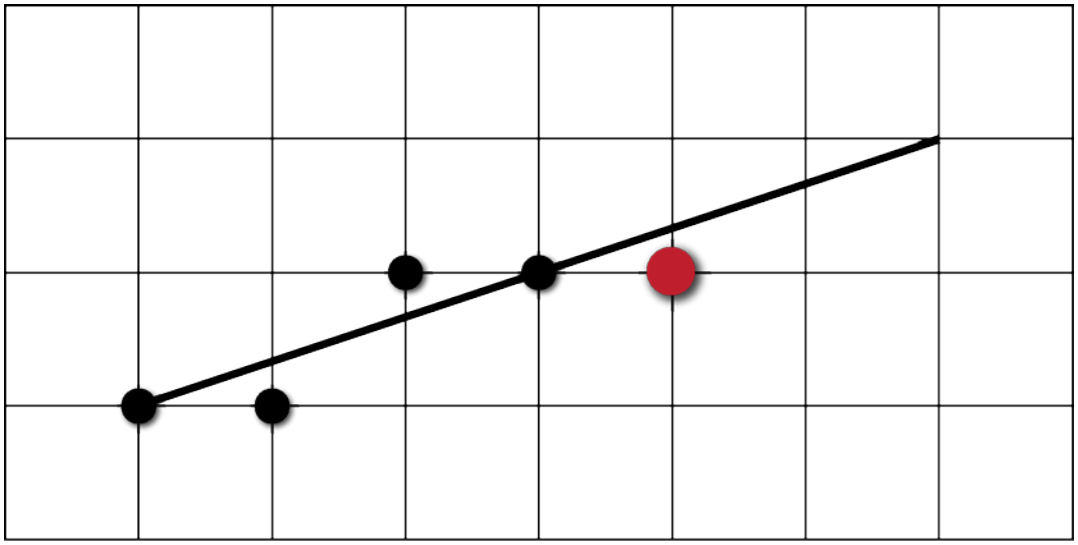
i	draw	p_i
0	(1,1)	-2
1	(2,1)	2
2	(3,2)	-6
3	(4,2)	-2

$A = (1,1) \qquad A' = (7,3)$

$\Delta x = 6 \qquad \Delta y = 2$

$2\Delta y = 4 \qquad 2\Delta y - 2\Delta x = -8 \qquad 2\Delta y - \Delta x = -2$

Bresenham - Example



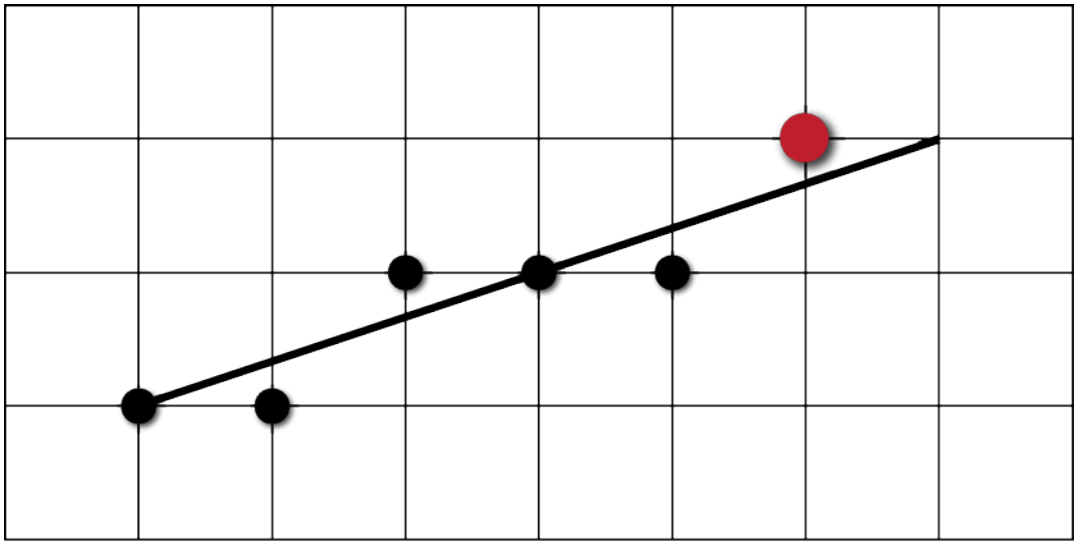
i	draw	p_i
0	(1,1)	-2
1	(2,1)	2
2	(3,2)	-6
3	(4,2)	-2
4	(5,2)	2

$A = (1,1) \qquad A' = (7,3)$

$\Delta x = 6 \qquad \Delta y = 2$

$2\Delta y = 4 \qquad 2\Delta y - 2\Delta x = -8 \qquad 2\Delta y - \Delta x = -2$

Bresenham - Example



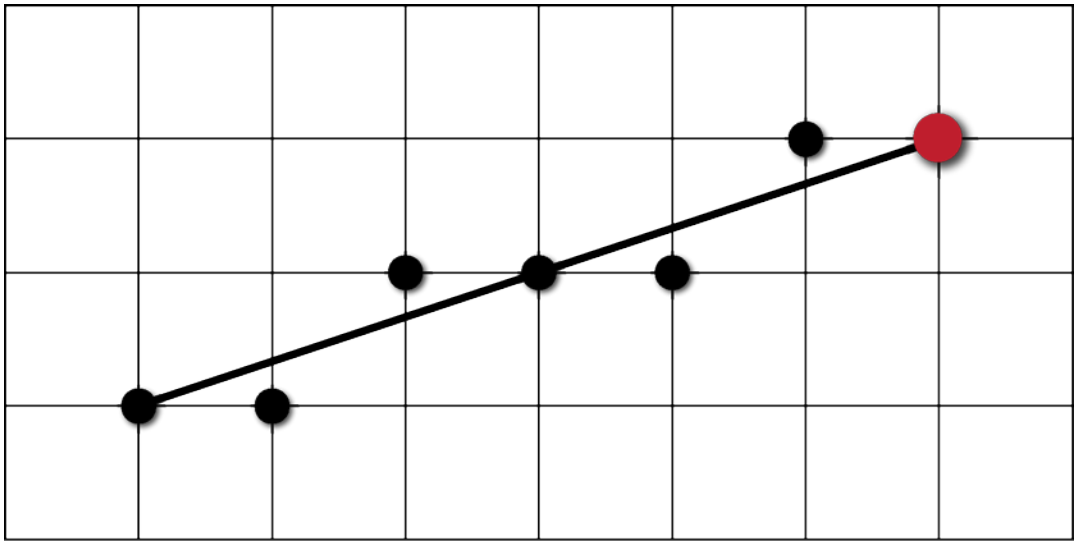
i	draw	p_i
0	(1,1)	-2
1	(2,1)	2
2	(3,2)	-6
3	(4,2)	-2
4	(5,2)	2
5	(6,3)	-8

$A = (1,1) \qquad A' = (7,3)$

$\Delta x = 6 \qquad \Delta y = 2$

$2\Delta y = 4 \qquad 2\Delta y - 2\Delta x = -8 \qquad 2\Delta y - \Delta x = -2$

Bresenham - Example



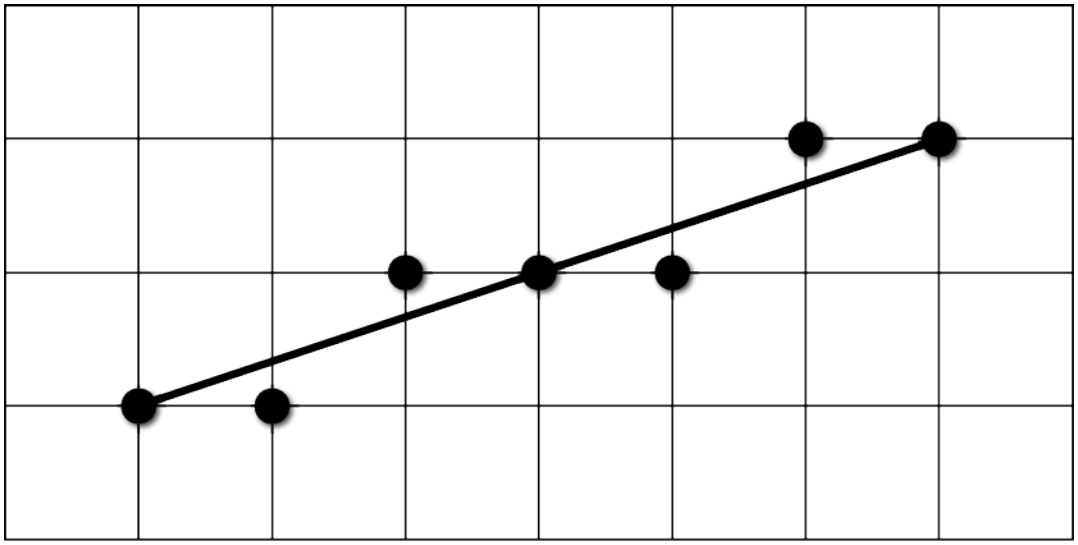
i	draw	p_i
0	(1,1)	-2
1	(2,1)	2
2	(3,2)	-6
3	(4,2)	-2
4	(5,2)	2
5	(6,3)	-8
6	(7,3)	-2

$A = (1,1) \qquad A' = (7,3)$

$\Delta x = 6 \qquad \Delta y = 2$

$2\Delta y = 4 \qquad 2\Delta y - 2\Delta x = -8 \qquad 2\Delta y - \Delta x = -2$

Bresenham - Example



i	draw	p_i
0	(1,1)	-2
1	(2,1)	2
2	(3,2)	-6
3	(4,2)	-2
4	(5,2)	2
5	(6,3)	-8
6	(7,3)	-2

$A = (1,1) \qquad A' = (7,3)$

$\Delta x = 6 \qquad \Delta y = 2$

$2\Delta y = 4 \qquad 2\Delta y - 2\Delta x = -8 \qquad 2\Delta y - \Delta x = -2$

Midpoint Algorithm

- Alternative construction of Bresenham's algorithm
  - Produces the same results
- Sometimes preferred to previous approach
  - Better applicable for curve drawing algorithms

- Assume nonvertical line (first quadrant)

$$f(x,y) = ax + by + c = 0 \qquad -b \geq a \geq 0$$

- Positive for points above the line
- Negative for lines below the line

Midpoint Algorithm

- Increase the x-coordinate : $x_{i+1} = x_i + 1$
- Determine only the y coordinate based on the sign:

$$d_i = f(x_i + 1, y_i + 0.5)$$

- If $d_i \leq 0$ then

$$d_{i+1} = f(x_i + 2, y_i + 0.5) = a(x_i + 2) + b(y_i + 0.5) + c =$$
$$= d_i + a$$

- If $d_i > 0$ then

$$d_{i+1} = f(x_i + 2, y_i + 1.5) = a(x_i + 2) + b(y_i + 1.5) + c =$$
$$= d_i + a + b$$

Midpoint Algorithm

- Next value of d_i can be computed by simple additions
- Initial value:

$$d_0 = f(x_0 + 1, y_0 + 0.5) = f(x_0, y_0) + a + b/2$$

- Integer computation:

$$F(x, y) = 2f(x, y) = 2(ax + by + c) = 0$$

Midpoint algorithm

- Equation of the line:

$$f(x,y) = 2(dy)x - 2(dx)y + 2c = 0$$

$$d_0 = f(x_0, y_0) + a + b/2 = \boxed{2dy - dx}$$

$$d_i \le 0 : d_{i+1} = d_i + a = d_i + \boxed{2dy}$$

$$d_i > 0 : d_{i+1} = d_i + a + b = d_i + \boxed{2dy - 2dx}$$

- d_i is equivalent to the p_i in Bresenham's algorithm

Curve Rasterization

Midpoint circle, Bresenham circle

Midpoint Circle Algorithm

- Circle:

$$f(x,y) = x^2 + y^2 - r^2 = 0$$

$$f(x,y) > 0 \quad \text{outside}$$

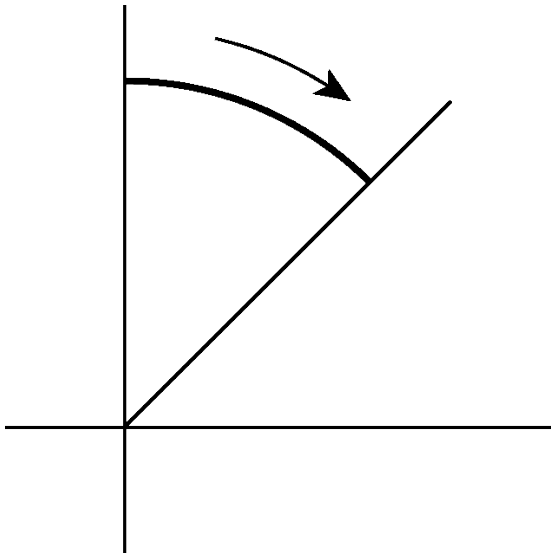
$$f(x,y) < 0 \quad \text{inside}$$

- Draw 2. octant of the circle

$$x_{i+1} = x_{i+1} + 1$$

$$d_i \geq 0 : y_{i+1} = y_i - 1$$

$$d_i < 0 : y_{i+1} = y_i$$



- Test : $d_i = f\left(x_i + 1, y_i - \frac{1}{2}\right)$

Midpoint Circle Algorithm - Step

- Determine d_i

$$d_{i+1} - d_i = f\left(x_i + 2, y_{i+1} - \tfrac{1}{2}\right) - f\left(x_i + 1, y_i - \tfrac{1}{2}\right) =$$

$$= \left(x_i + 2\right)^2 + \left(y_{i+1} - \tfrac{1}{2}\right)^2 - r^2 - \left(x_i + 1\right)^2 - \left(y_i - \tfrac{1}{2}\right)^2 + r^2 =$$

$$= 2x_i + 3 + \left(y_{i+1}{}^2 - y_{i+1}\right) - \left(y_i{}^2 - y_i\right)$$

$$y_{i+1} = y_i \Rightarrow d_{i+1} - d_i = 2x_i + 3$$

$$y_{i+1} = y_i - 1 \Rightarrow d_{i+1} - d_i = 2x_i + 3 + \left(y_i{}^2 - 2y_i + 1 - y_i + 1\right) - \left(y_i{}^2 - y_i\right) =$$

$$= 2x_i - 2y_i + 5$$

Midpoint Circle Algorithm- Initialization

- Compute d_0

$$x_0 = 0,\ y_0 = r$$

$$d_0 = \left(1, r - \tfrac{1}{2}\right) = 1 + \left(r - \tfrac{1}{2}\right)^2 - r^2 = 1 + r^2 - r + \tfrac{1}{4} - r^2 = \tfrac{5}{4} - r$$

- Integer computation

$$d_0 = round\left(\tfrac{5}{4} - r\right)$$

- If r is integer value

$$d_0 = 1 - r$$

Midpoint Circle Algorithm - Final

- Initialization: $(x_0, y_0) = (0, \mathrm{r}) \quad d_0 = round\left(\frac{5}{4} - r\right)$

$$x_{i+1} = x_i + 1$$

- If $d_i \geq 0$ then $y_{i+1} = y_i - 1 \quad d_{i+1} = d_i + 2x_i - 2y_i + 5$

  else $y_{i+1} = y_i \quad d_{i+1} = d_i + 2x_i + 3$

Bresenham Circle Algorithm

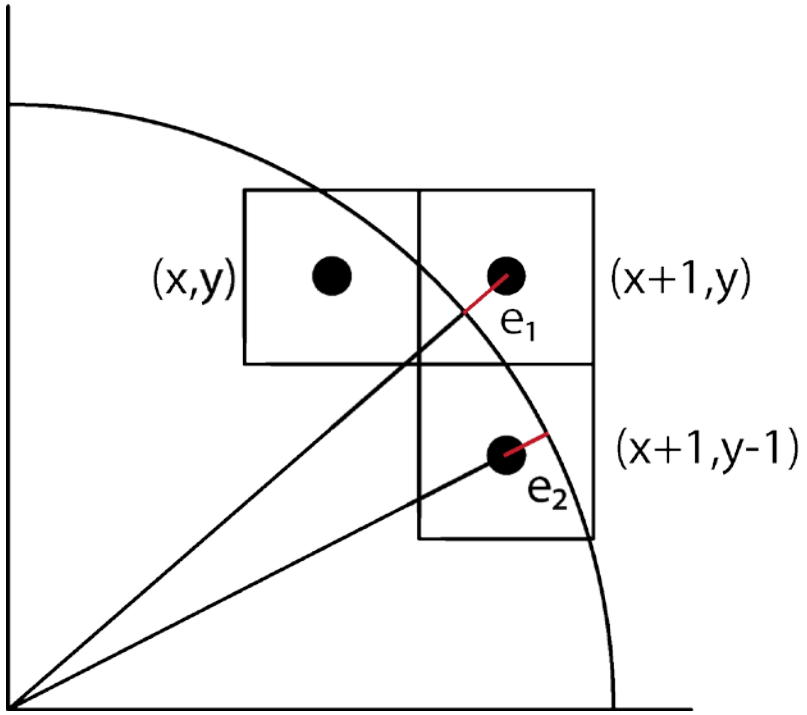
- Generate 2. octant
- Determine if $y_{i+1} = y_i$ or $y_{i+1} = y_i - 1$

- Based on sign of error:

$$p_i = e_{i1} + e_{i2}$$

$$e_{i1} = (x_i + 1)^2 + y_i^{\,2} - r^2$$

$$e_{i2} = (x_i + 1)^2 + (y_i - 1)^2 - r^2$$



Bresenham Circle Algorithm - Step

- Determine change in p

$$p_i = e_{i1} + e_{i2} = 2(x_i + 1)^2 + y_i^2 + (y_i - 1)^2 - 2r^2$$

$$p_{i+1} = 2(x_i + 2)^2 + y_{i+1}^2 + (y_{i+1} - 1)^2 - 2r^2$$

$$p_{i+1} - p_i = 2\left((x_i + 2)^2 - (x_i + 1)^2\right) + \left(y_{i+1}^2 - y_i^2\right) + \left((y_{i+1} - 1)^2 - (y_i - 1)^2\right)$$

$$p_{i+1} = p_i + 4x_i + 6 + \left(y_{i+1}^2 - y_i^2\right) + \left((y_{i+1} - 1)^2 - (y_i - 1)^2\right)$$

Bresenham Circle Algorithm - Step

- Determine change in p

$$p_i = e_{i1} + e_{i2} = 2(x_i + 1)^2 + y_i^2 + (y_i - 1)^2 - 2r^2$$

$$p_{i+1} = 2(x_i + 2)^2 + y_{i+1}^2 + (y_{i+1} - 1)^2 - 2r^2$$

$$p_{i+1} - p_i = 2\left((x_i + 2)^2 - (x_i + 1)^2\right) + \left(y_{i+1}^2 - y_i^2\right) + \left((y_{i+1} - 1)^2 - (y_i - 1)^2\right)$$

$$p_{i+1} = p_i + 4x_i + 6 + \left(y_{i+1}^2 - y_i^2\right) + \left((y_{i+1} - 1)^2 - (y_i - 1)^2\right)$$

- If $p_i \le 0$ then $y_{i+1} = y_i$

$$p_{i+1} = p_i + 4x_i + 6$$

- If $p_i > 0$ then $y_{i+1} = y_i - 1$

$$p_{i+1} = p_i + 4(x_i - y_i) + 10$$

Bresenham Circle Algorithm - Initialization

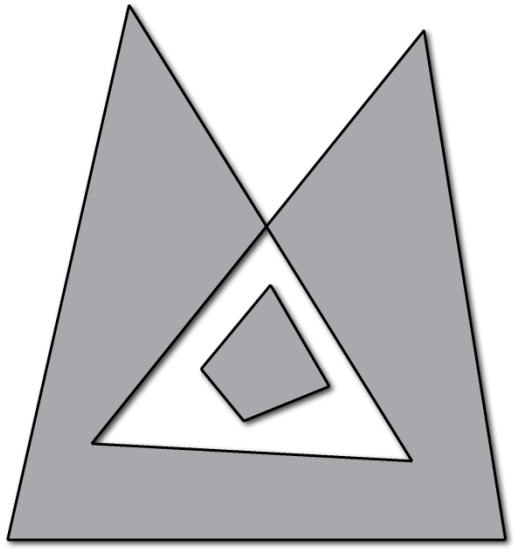
- Compute p_0

$$p_0 = 2 + r^2 + (r-1)^2 - 2r^2 = 2 + r^2 + r^2 + 1 - 2r - 2r^2 = 3 - 2r$$

Polygon Rasterization

Polygon Rasterization

- Draw boundary
  - Draw polygon edges
- Draw filled polygon
  - Draw edges and fill the inside
- Problem
  - Need to determine which areas to fill
  - Slow
- Solution
  - Specialized algorithm



Polygon Rasterization

- Using scan line
  - Scanning from top to bottom (or bottom to top)
- Computing intersections
  - Using coherence
  - Computing intersections incrementally
- Compute intersections only with lines that intersect the scan line
  - Store active edge list (AEL)

Polygon Rasterization: Algorithm - Initialization

- Associate an empty bucket to each scan line
- Find the larges y value for each edge and put it into corresponding scan line's bucket
  - Sort according to x
- For each edge store
  - x – x-coordinate of the highest point
  - y – the lowest y-coordnate of the edge
  - dx – change in x between scan lines (slope)
- Initialize the AEL to empty
- Set y to the height of the top scan line

Polygon Rasterization: Algorithm - Iteration

- Add all edges in the bucket for y to the AEL
  - AEL is always sorted according to x
- Compute intersection for each edge in AEL
  - Intersections are sorted because AEL is sorted
- Fill areas between intersections
- Remove any edge from AEL for which y is equal to the y-coordinate of the scan line
- Process next scan line
  - Exit if all scan lines are processed

Polygon Rasterization: Intersections

- Compute intersection based on the intersection with previous scan line

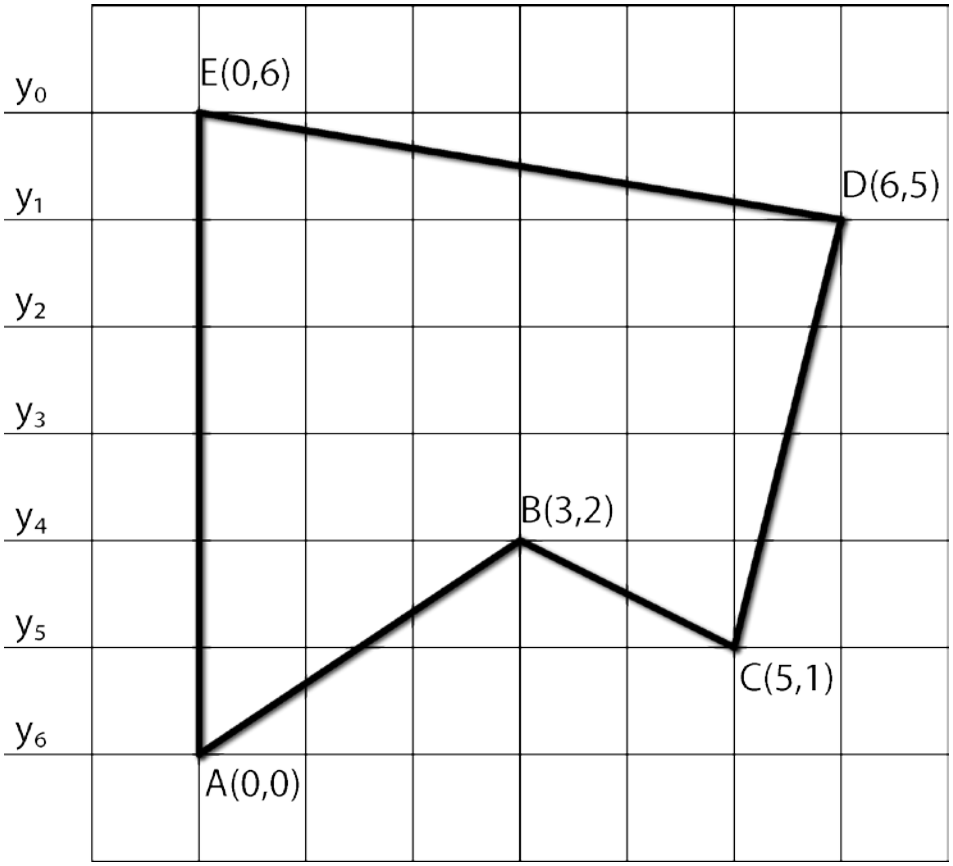
$$x_{k+1} = x_k + dx = x_k + \frac{\Delta x}{\Delta y}$$

- Can be altered to use only integer arithmetic

- Parallel implementation
  - Compute intersection for each scan line separately

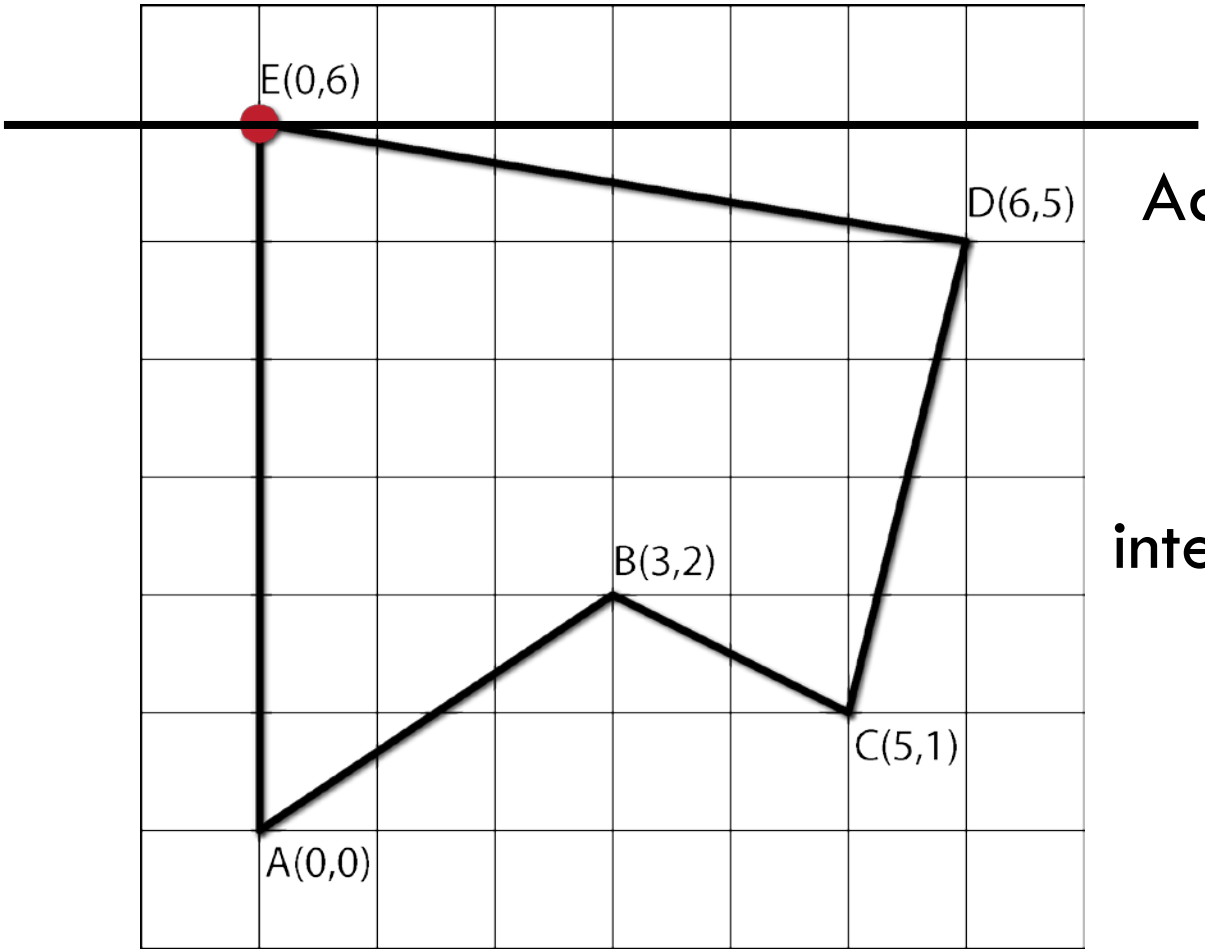
$$x_{k+1} = x_0 + kdx$$

Polygon Rasterization: Edge List



	EA	ED
0	|0|0|0	|0|5|6
1	|6|1|-1/4	
2	DC	
3		
	BA	BC
4	|3|0|-3/2	|3|1|2
5		
6		

Polygon Rasterization: Example

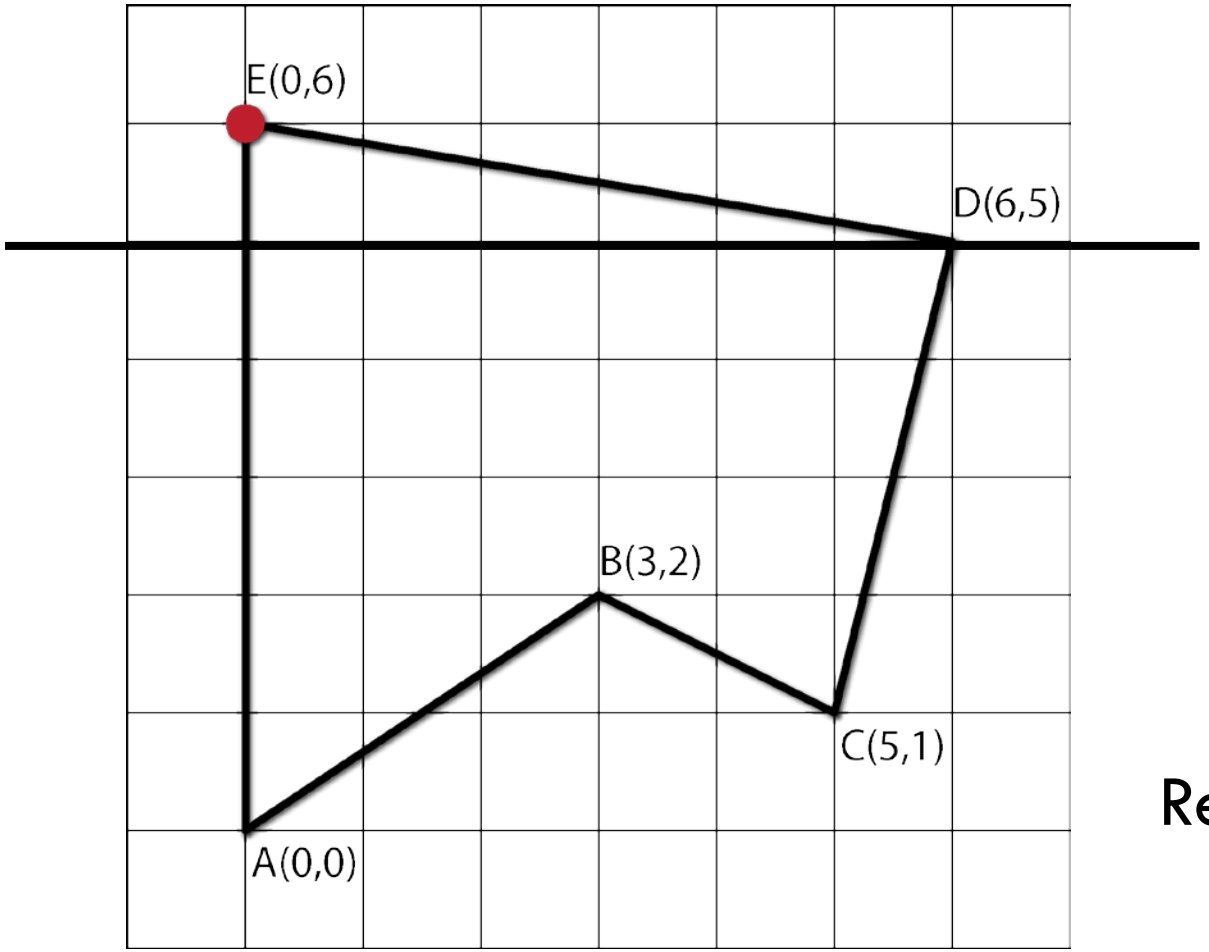


Add EA and ED to AEL

AEL: EA, ED

intersections: (0,6), (0,6)

Polygon Rasterization: Example



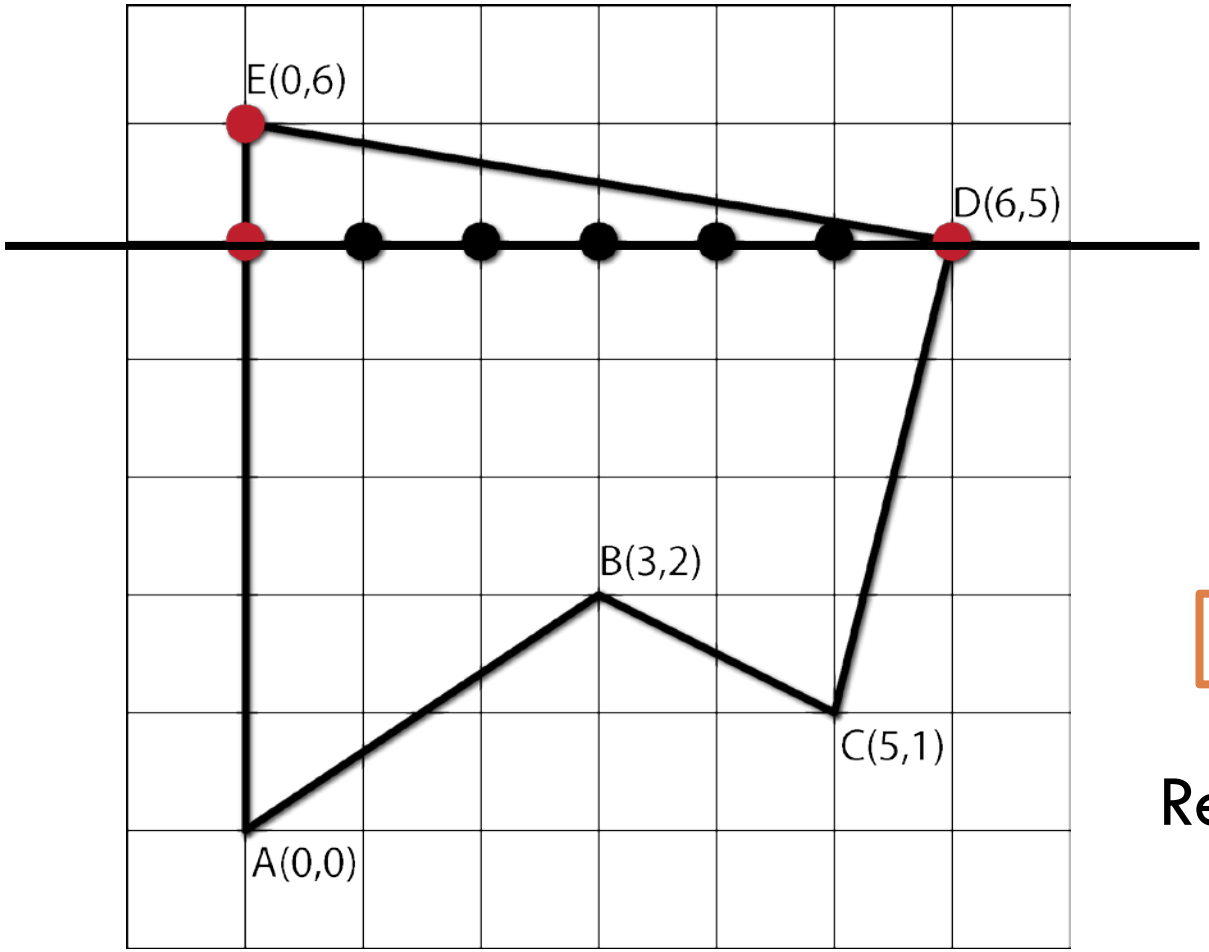
Add DC to AEL

AEL: EA, ED, DC

intersections:
(0,5), (6,5), (6,5)

Remove ED from AEL

Polygon Rasterization: Example



Add DC to AEL

AEL: EA, ED, DC

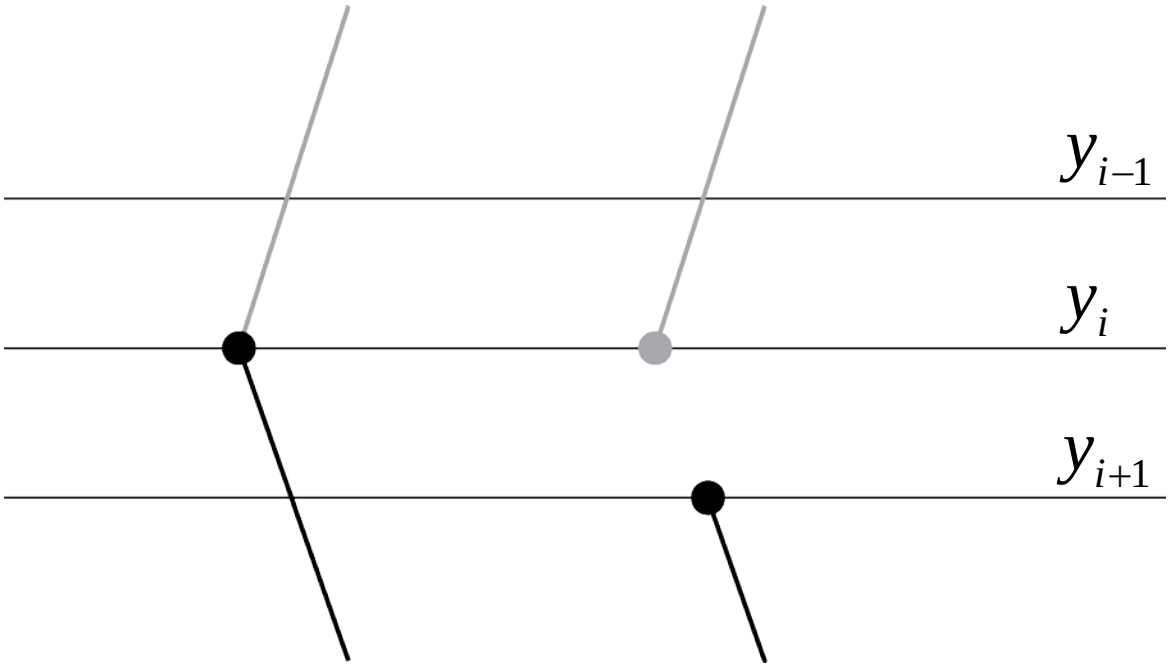
intersections:

(0,5), (6,5), (6,5)

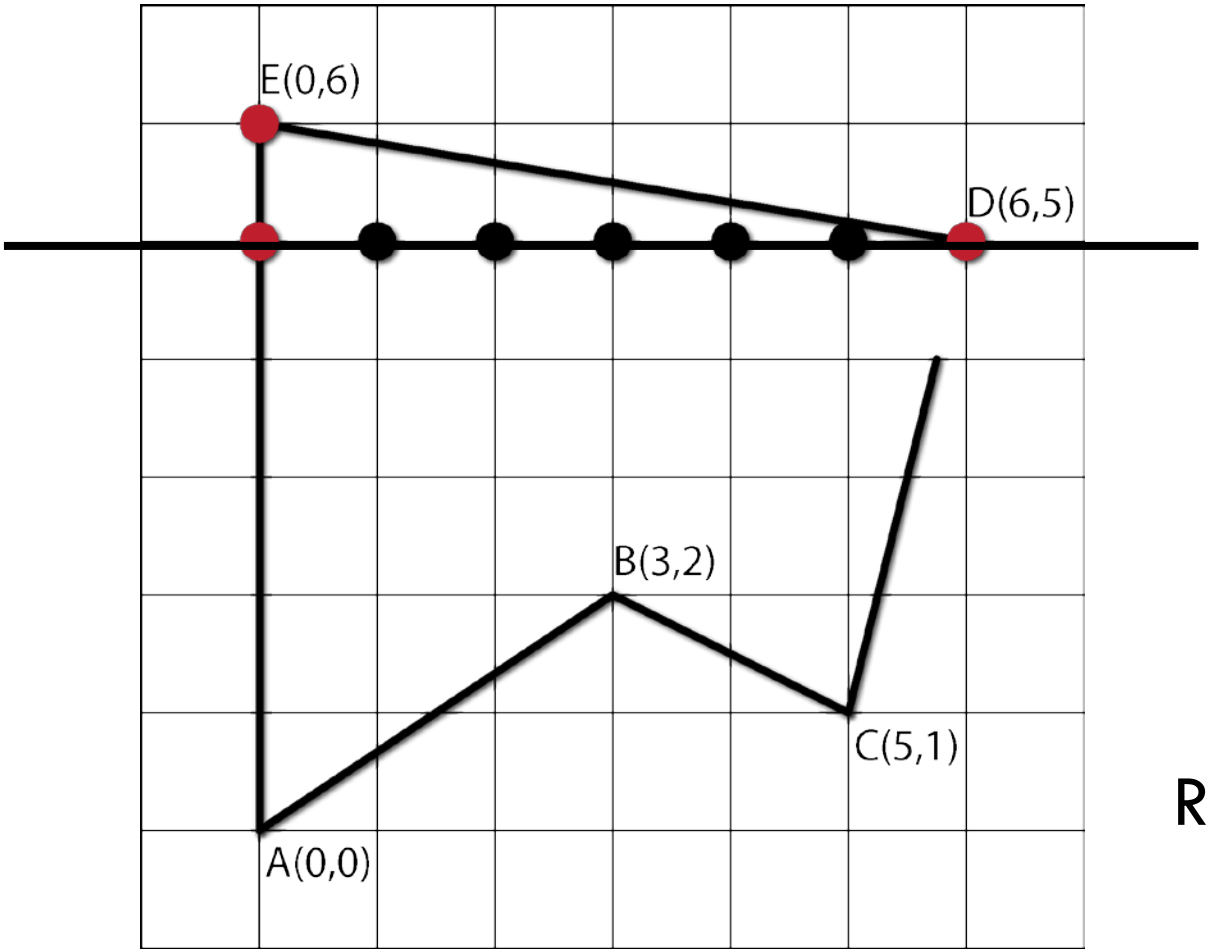
Remove ED from AEL

Polygon Rasterization - Correction

- Shorten the by moving the top endpoint of the edge
  - Do in preprocessing



Polygon Rasterization: Example



Add DC to AEL

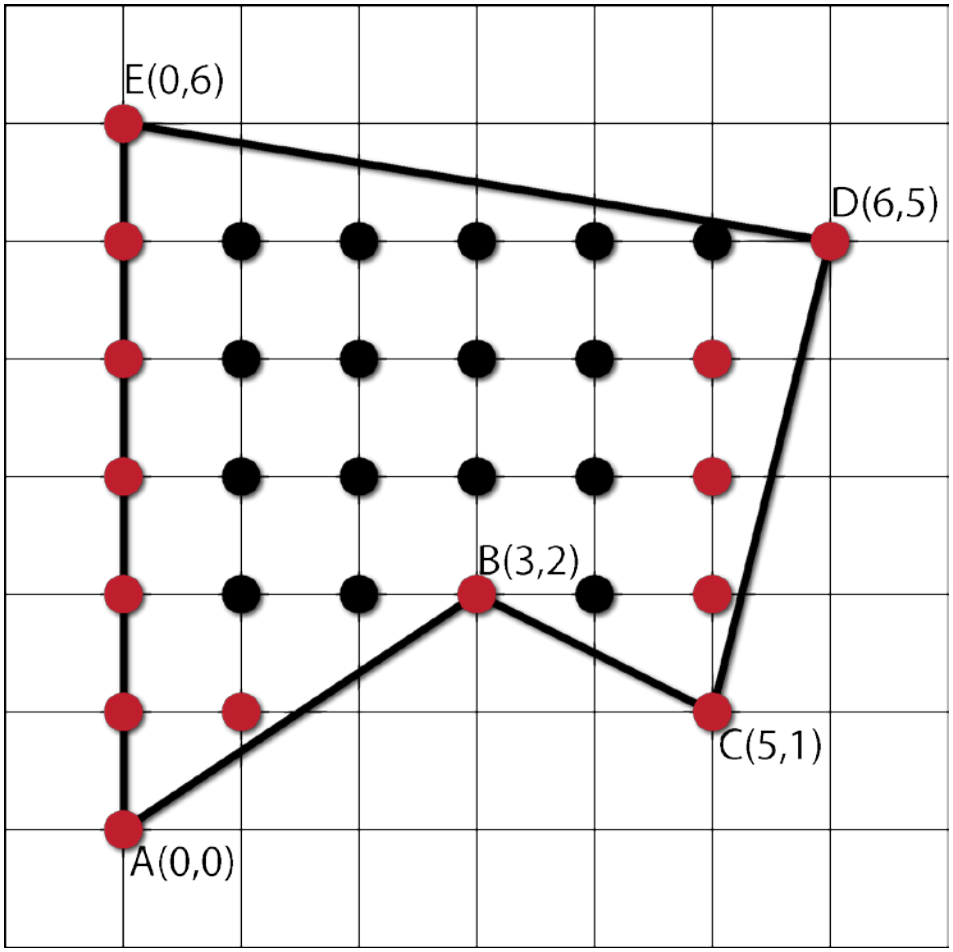
AEL: EA, ED, DC

intersections:

(0,5), (6,5)

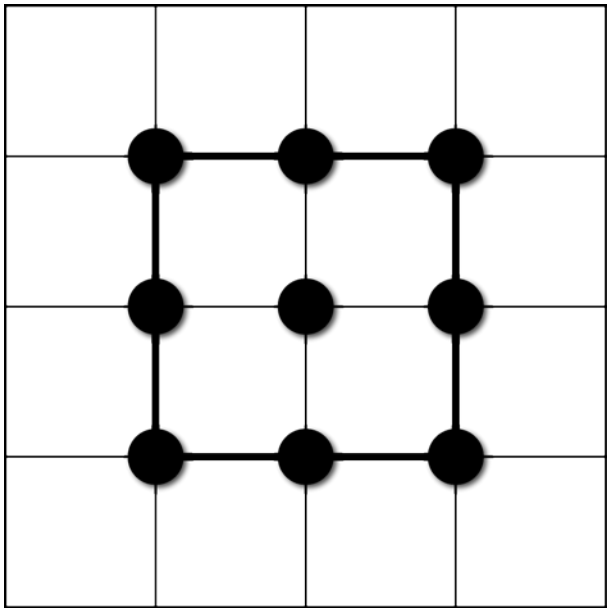
Remove ED from AEL

Polygon Rasterization: Example



Polygon Rasterization: Problem

- Area = 4
- Highlighted points = 9



- Improvement
  - Filling: omit the right point
  - Shorten the edges from top by 1

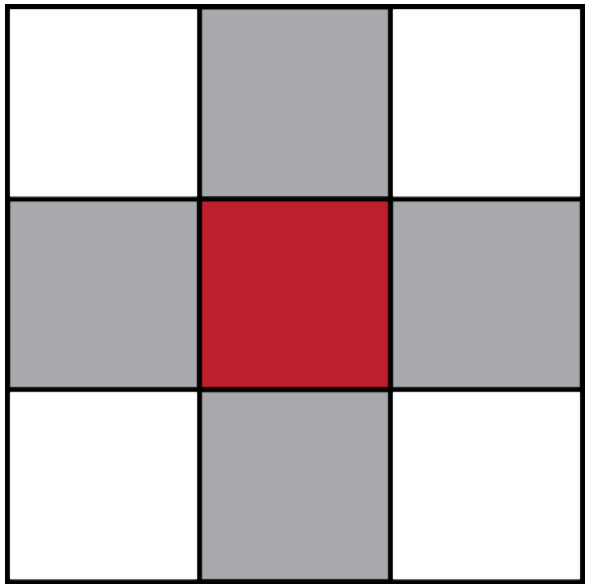
Fill Algorithms

Pixel-Based Fill Algorithms

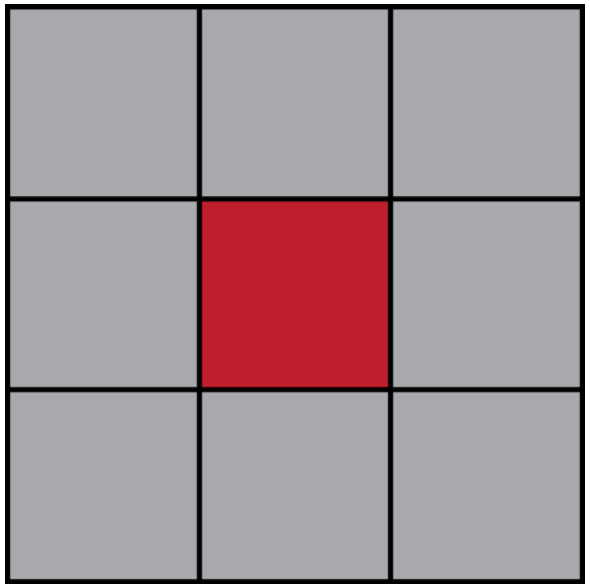
- Operate only with pixel
- Filling a specified area with a color
  - Arbitrary boundary (not only polygons)
- Specify starting point (seed)
  - Automatic or user specified
- Specify area
  - Color – flood fill
  - Boundary – boarder fill
- Dependent from adjacency

Adjacency

4-adjacency



8-adjacency



Flood Fill


```
FloodFill(x, y, new_color, old_color)
    if color(x,y) = old_color then
        color(x,y) = new_color
        FloodFill(x+1, y, new_color, old_color)
        FloodFill(x-1, y, new_color, old_color)
        FloodFill(x, y+1, new_color, old_color)
        FloodFill(x, y-1, new_color, old_color)
```

- For 8-adjucency add
  - (x+1, y+1), (x-1, y+1), (x+1, y-1), (x-1, y-1)

Boarder Fill


```
FloodFill(x, y, new_color, border_color)
    if (color(x,y) != border_color) and
          (color(x,y) != new_color) then
        color(x,y) = new_color
        FloodFill(x+1, y, new_color, border_color)
        FloodFill(x-1, y, new_color, border_color)
        FloodFill(x, y+1, new_color, border_color)
        FloodFill(x, y-1, new_color, border_color)
```

- Preprocessing
  - Delete new_color from area

Questions ???