19CSE433 Computer Graphics & Visualization

Professional Elective 1

5^{th} Semester,2021-22 Odd

2019-22 Batch, BTech CSE

DR.S.PADMAVATHI,

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,

AMRITA SCHOOL OF ENGINEERING, COIMBATORE

Output Primitives

Points

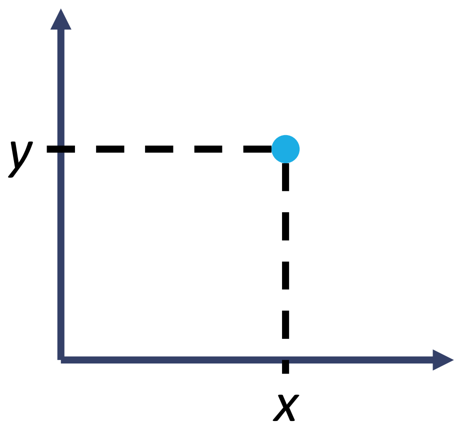
Lines
- DDA Algorithm
- Bresenham's Algorithm

Polygons
- Scan-Line Polygon Fill
- Inside-Outside Tests
- Boundary-Fill Algorithm

Points

Single Coordinate Position

- Set the bit value(color code) corresponding to a specified screen position within the frame buffer

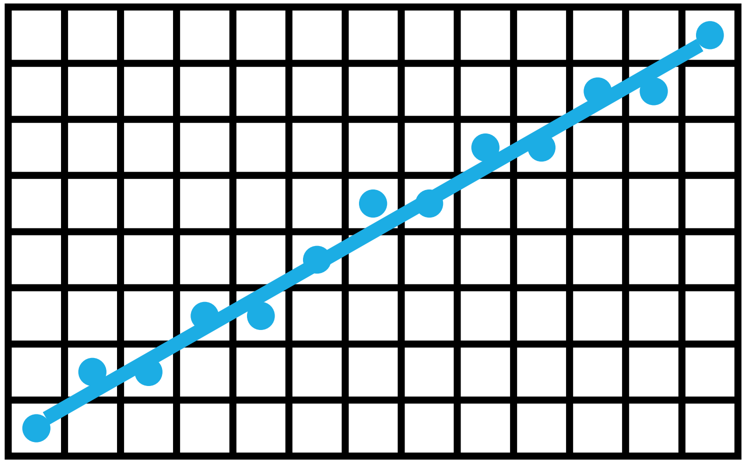


setPixel (*x*, *y*)

Lines

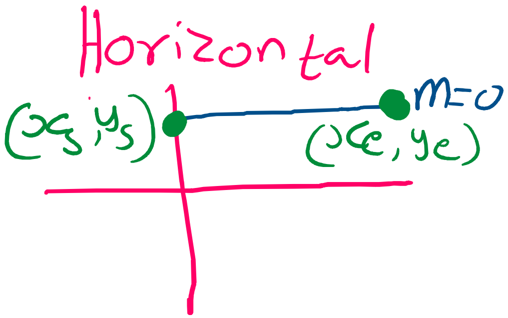
Intermediate Positions between Two Endpoints

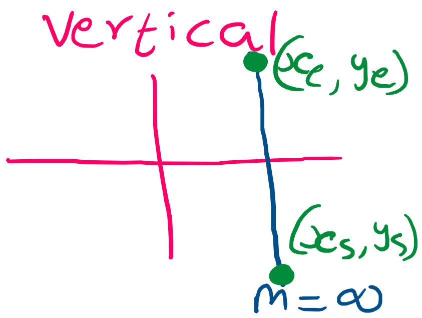
- DDA, Bresenham's line algorithms

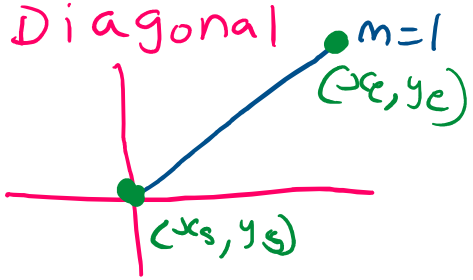


Jaggies = Aliasing

Line drawing- special case







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

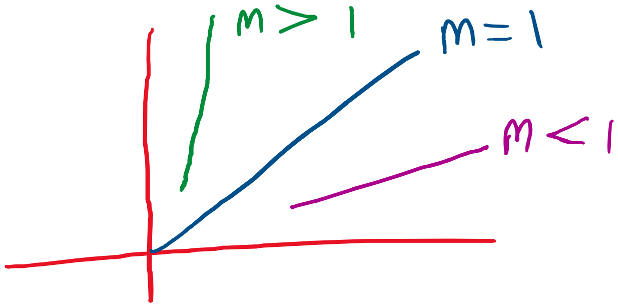
Special case

y same
x increment from x_s to x_e

x same
y: increment from y_s to y_e

Increment x or y
& calculate other as $y = x$ or $x = y$

Rasterization of lines



$m > 1$: nearly vertical $\rightarrow \Delta y > \Delta x$

Increment y & Compute x

$$x = \frac{1}{m}(y - c)$$

$$y = mx + c$$

$$x = \frac{y - c}{m}$$

$$m = \frac{\Delta y}{\Delta x} = \frac{y_e - y_s}{x_e - x_s}$$

$m < 1$: nearly horizontal $\rightarrow \Delta y < \Delta x$

Increment x & compute y

$$y = mx + c$$

DDA Algorithm

Nearly Horizontal

Digital Differential Analyzer

- $0 <$ Slope $<= 1$
  - Unit x interval = 1

$m < 1$

$\frac{\Delta y}{\Delta x} < 1$

$\Delta y < \Delta x$

Increment x and compute y

$y = mx + c$

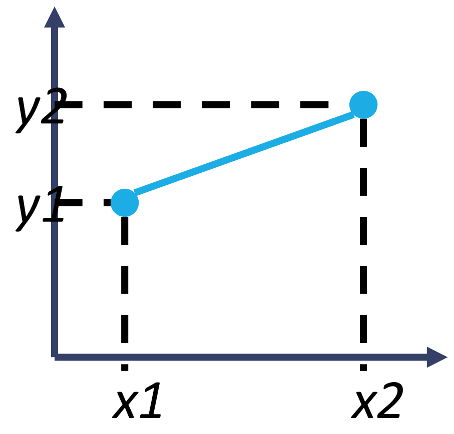
k^{th} iteration : $y_k = m x_k + c$

$k+1^{th}$ iteration : $y_{k+1} = m x_{k+1} + c$

$x_{k+1} = x_k + 1$

$\therefore y_{k+1} = m x_k + m + c$

$= y_k + m$



$$y_{k+1} = y_k + m$$

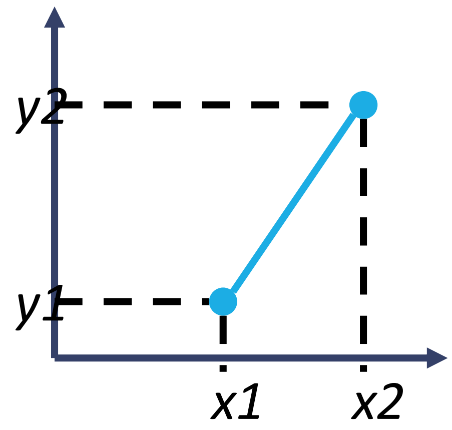
DDA Algorithm

Nearly vertical

$m > 1 \Rightarrow \frac{\Delta y}{\Delta x} > 1 \Rightarrow \Delta y > \Delta x$

Digital Differential Analyzer

- 0 < Slope <= 1
  - Unit x interval = 1
- Slope > 1
  - Unit y interval = 1



Increment y

Calculate x

$$x_k = \frac{1}{m}(y_k - c) = \frac{1}{m} y_k - \frac{c}{m}$$

$$x_{k+1} = \frac{1}{m}(y_{k+1} - c)$$

$$y_{k+1} = y_k + 1$$

$$\therefore x_{k+1} = \frac{1}{m} y_k + \frac{1}{m} - \frac{c}{m} = x_k + \frac{1}{m}$$

$$x_{k+1} = x_k + \frac{1}{m}$$

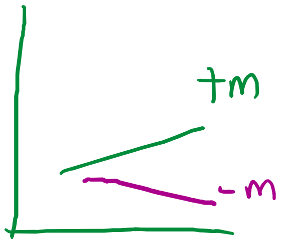
DDA Algorithm

Negative slope, Nearly horizontal

Increment x, compute y

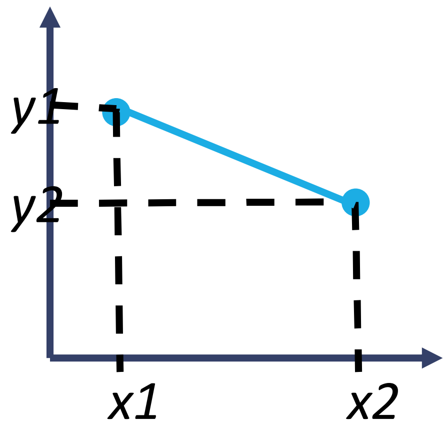
Digital Differential Analyzer

- 0 < Slope <= 1
  - Unit x interval = 1
- Slope > 1
  - Unit y interval = 1
- -1 <= Slope < 0
  - Unit x interval = -1



y increases from start to end

y decreases



$$y_{k+1} = y_k - m$$

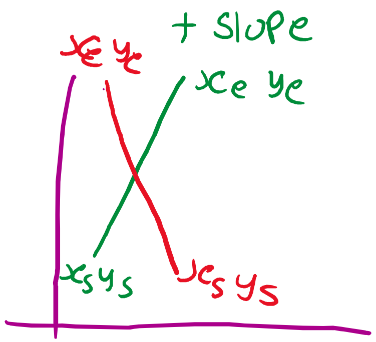
DDA Algorithm

Nearly vertical, & negative slope

Increment y, compute x

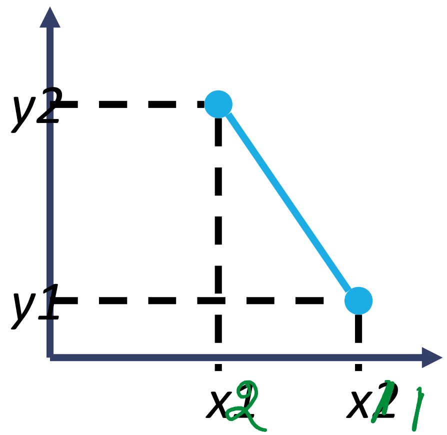
Digital Differential Analyzer

- Slope >= 1
  - Unit x interval = 1
- 0 < Slope < 1
  - Unit y interval = 1
- -1 <= Slope < 0
  - Unit x interval = -1
- Slope < -1
  - Unit y interval = -1



y increases x ↓

y ↑ x ↑ ⇒ +ve slope



$$x_{k+1} = x_k - \frac{1}{m}$$

DDA Algorithm

```
#define ROUND(a) ((int)(a+0.5))

void lineDDA (int xa, int ya, int xb, int yb)
{
  int dx = xb - xa, dy = yb - ya, steps, k;
  float xIncrement, yIncrement, x = xa, y = ya;

  if (abs (dx) > abs (dy)) steps = abs (dx);
  else steps = abs  dy);
  xIncrement = dx / (float) steps;
  yIncrement = dy / (float) steps;

  setPixel (ROUND(x), ROUND(y));
  for (k=0; k<steps; k++) {
    x += xIncrement;
    y += yIncrement;
    setPixel (ROUND(x), ROUND(y));
  }
}
```

Exercise:

Digitize the line from (20,10) to (30,18)

(2,10)to (5,18)

(15,30) to(30,25)

(20,10) to (15,25)

Bresenham's Line Algorithm

Accurate and Efficient

- Use only incremental integer calculations
- Test the sign of an integer parameter

nearly Horizontal

Case) Positive Slope Less Than 1

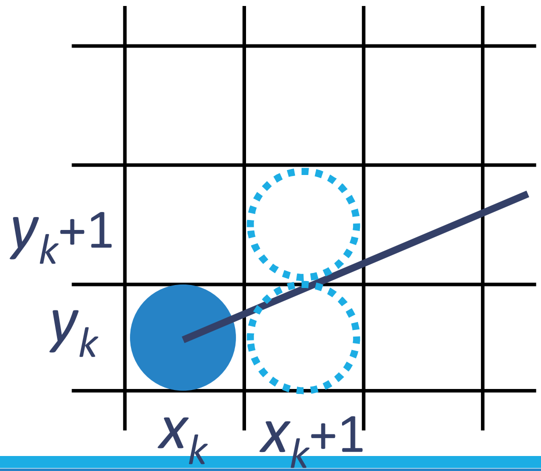
$m<1$ $\Delta y<\Delta x$

- After the pixel (x_k, y_k) is displayed, next which pixel is decided to plot in column x_{k+1}?

Increment x

Decide y_k or y_k+1

➔ (x_k+1, y_k) or (x_k+1, y_k+1)



Bresenham's Algorithm(cont.)

Case) Positive Slope Less Than 1

- y at sampling position x_k

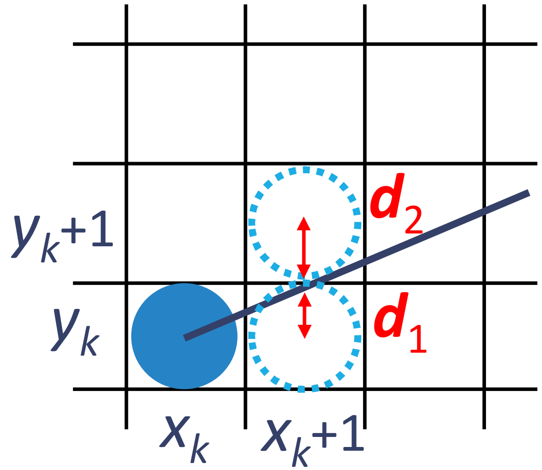
$$y = m(x_k + 1) + b$$

- Difference

$$d_1 = y - y_k = m(x_k + 1) + b - y_k$$
$$d_2 = y_k + 1 - y = y_k + 1 - m(x_k + 1) - b$$

- Decision parameter

$d_1 - d_2 < 0 \Rightarrow (x_k+1, y_k)$
$d_1 - d_2 > 0 \Rightarrow (x_k+1, y_k+1)$



$$p_k = \Delta x(d_1 - d_2)$$
$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + 2\Delta y + \Delta x(2b - 1)$$
$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$

Bresenham’s Algorithm(cont.)

Case) Positive Slope Less Than 1

- Decision parameter

$$p_{k+1} - p_k = \left(2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + c\right) - \left(2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c\right)$$
$$= 2\Delta y\left(x_{k+1} - x_k\right) - 2\Delta x\left(y_{k+1} - y_k\right)$$

$$\therefore p_{k+1} = p_k + 2\Delta y - 2\Delta x\left(y_{k+1} - y_k\right)$$

- Decision parameter c

$$p_0 = 2\Delta y \cdot x_0 - 2\Delta x \cdot y_0 + 2\Delta y + \Delta x\left(2b - 1\right)$$
$$= 2\Delta y \cdot x_0 - 2\Delta x \cdot \left(mx_0 + b\right) + 2\Delta y + \Delta x\left(2b - 1\right)$$
$$= 2\Delta y \cdot x_0 - 2\Delta y \cdot x_0 - 2b\Delta x + 2\Delta y + 2b\Delta x - \Delta x$$

$$\therefore p_0 = 2\Delta y - \Delta x$$

Bresenham's Algorithm(cont.)

Algorithm for $0<m<1$

- Input the two line endpoints and store the left end point in (x_0, y_0)
- Load (x_0, y_0) into the frame buffer; that is, plot the first point
- Calculate constants Δx, Δy, $2\Delta y$, and $2\Delta y - 2\Delta x$, and obtain the starting value for the decision parameter as

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

- At each x_k along the line, start at $k = 0$, perform the following test:
  - If $p_k < 0$, the next point to plot is (x_k+1, y_k) and

$$p_{k+1} = p_k + 2\Delta y$$

  - Otherwise, the next point to plot is (x_k+1, y_k+1) and

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$

- Repeat step 4 Δx times

Bresenham's Line-Drawing Algorithm for $|m| < 1$

1. Input the two line endpoints and store the left endpoint in (x_0, y_0).
2. Load (x_0, y_0) into the frame buffer; that is, plot the first point.
3. Calculate constants Δx, Δy, $2\Delta y$, and $2\Delta y - 2\Delta x$, and obtain the starting value for the decision parameter as

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

4. At each x_k along the line, starting at $k = 0$, perform the following test: If $p_k < 0$, the next point to plot is $(x_k + 1, y_k)$ and

$$p_{k+1} = p_k + 2\Delta y$$

   Otherwise, the next point to plot is $(x_k + 1, y_k + 1)$ and

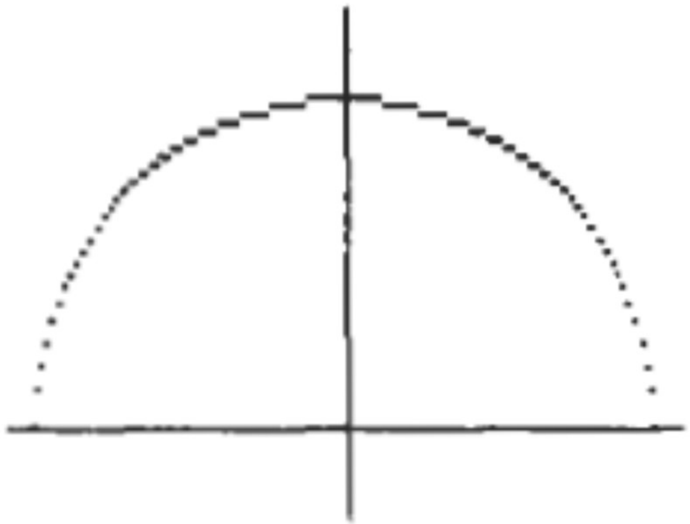
$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$

5. Repeat step 4 Δx times.

Circle Drawing

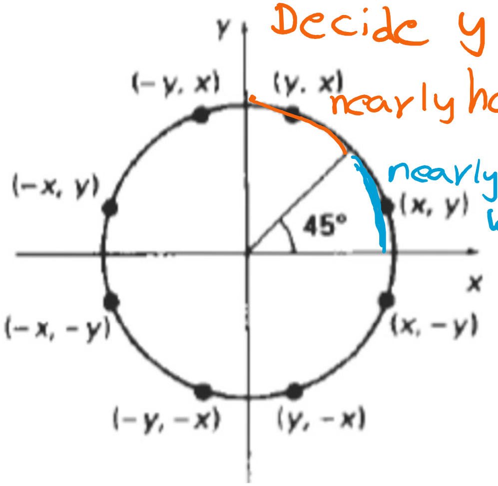
$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

$$y = y_c \pm \sqrt{r^2 - (x_c - x)^2}$$

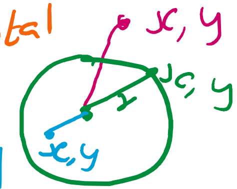


Circle drawing algorithm

Increment x
Decide y



$$f_{circle}(x, y) = x^2 + y^2 - r^2$$



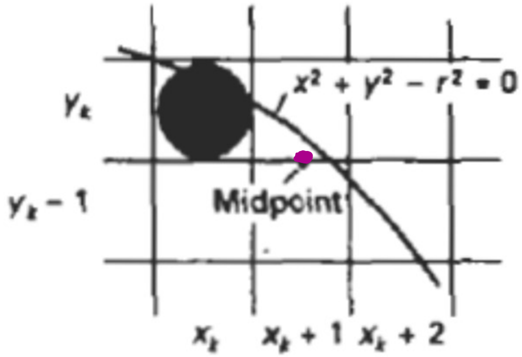
$x^2 + y^2 < r^2 \Rightarrow$ inside circle

$x^2 + y^2 > r^2 \Rightarrow$ outside circle

$x^2 + y^2 = r^2 \Rightarrow$ on the circle

$$f_{circle}(x, y) \begin{cases} < 0, & \text{if } (x, y) \text{ is inside the circle boundary} \\ = 0, & \text{if } (x, y) \text{ is on the circle boundary} \\ > 0, & \text{if } (x, y) \text{ is outside the circle boundary} \end{cases}$$

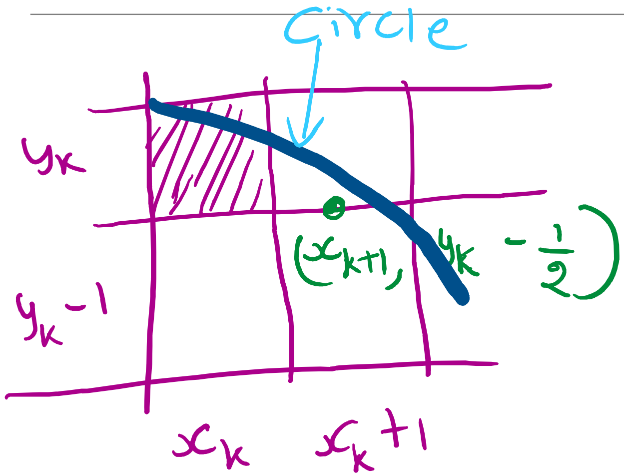
mid point of (y_k, $y_k - 1$) for $x_k + 1$

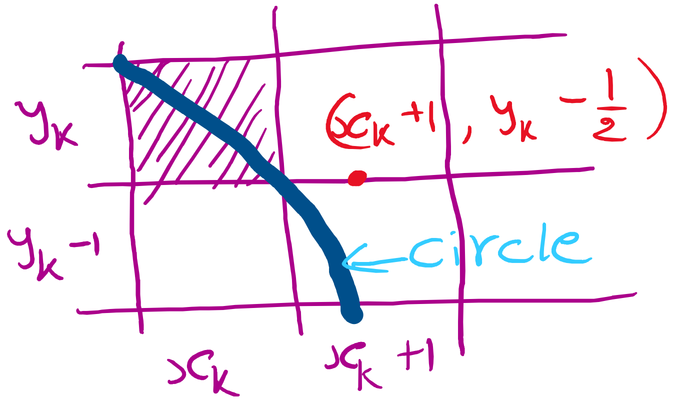


midpoint inside $\Rightarrow y_k$ closer to circle

midpoint outside $\Rightarrow y_k - 1$ closer

Midpoint inside/outside circle





If equal any one can be chosen

midpoint inside
y_k closer to circle
than $y_k - 1$

midpoint outside circle
$y_k - 1$ closer to circle
than y_k

Mid point circle drawing algorithm

$$p_k = f_{circle}\left(x_k + 1, y_k - \frac{1}{2}\right)$$

$$= (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2$$

$$p_{k+1} = f_{circle}\left(x_{k+1} + 1, y_{k+1} - \frac{1}{2}\right)$$

$$= [(x_k + 1) + 1]^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - r^2$$

or

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

$P_{k+1} = (x_k+1)^2 + 1 + 2(x_k+1) + y_{k+1}^2 + \frac{1}{4} - y_{k+1} - r^2$

$P_k = (x_k+1)^2 + y_k^2 + \frac{1}{4} - y_k - r^2$

$P_{k+1} - P_k = 2(x_k+1) + 1 + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k)$

$y_{k+1} = y_k$ or $y_k - 1$

Inside

$\therefore P_{k+1} = P_k + 2(x_k+1) + 1 \leftarrow \boxed{\text{if } y_{k+1} = y_k}$

else

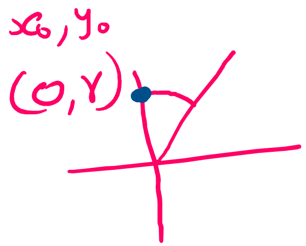
$P_{k+1} = P_k + 2(x_k+1) + 1 + (y_k^2 + 1 - 2y_k - y_k^2) - (y_k - 1 - y_k)$

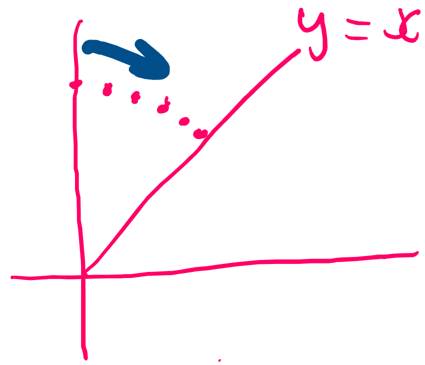
$\Leftarrow P_k + 2(x_k+1) + 1 + (1 - 2y_k) + 1$

outside

$P_k + 2(x_k+1) + 1 - 2(y_k - 1)$

Initial values





$$2x_{k+1} = 2x_k + 2$$

$$2y_{k+1} = 2y_k - 2$$

$(x_0, y_0) = (0, r)$: Initial value

x increments
y could be same
or less

$$p_0 = f_{circle}\left(1, r - \frac{1}{2}\right)$$

$$= 1 + \left(r - \frac{1}{2}\right)^2 - r^2$$

$$p_0 = \frac{5}{4} - r$$

Generate pts until $x = y$

Midpoint Circle Algorithm

1. Input radius r and circle center (x_c, y_c), and obtain the first point on the circumference of a circle centered on the origin as

$$(x_0, y_0) = (0, r)$$

2. Calculate the initial value of the decision parameter as

$$p_0 = \frac{5}{4} - r$$

midpoint
Inside circle

3. At each x_k position, starting at $k = 0$, perform the following test: If $p_k < 0$, the next point along the circle centered on (0, 0) is (x_{k+1}, y_k) and

$$p_{k+1} = p_k + 2x_{k+1} + 1$$

Otherwise, the next point along the circle is $(x_k + 1, y_k - 1)$ and

$$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$$

$$\left\{ \begin{aligned} x_{k+1} &= x_k + 1 \\ y_{k+1} &= y_k - 1 \end{aligned} \right.$$

outside circle

where $2x_{k+1} = 2x_k + 2$ and $2y_{k+1} = 2y_k - 2$.

4. Determine symmetry points in the other seven octants.
5. Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) and plot the coordinate values:

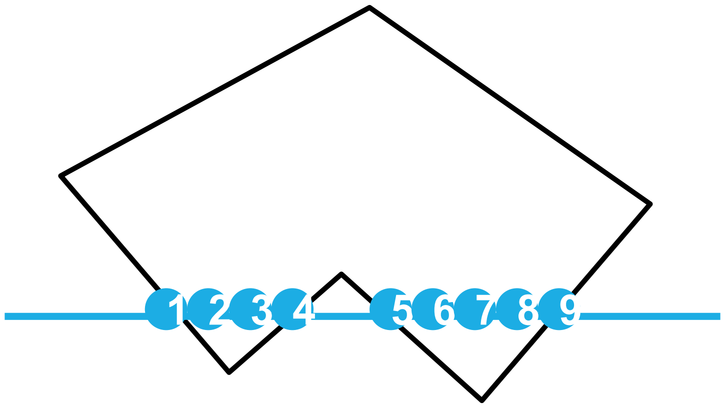
$$x = x + x_c, \qquad y = y + y_c$$

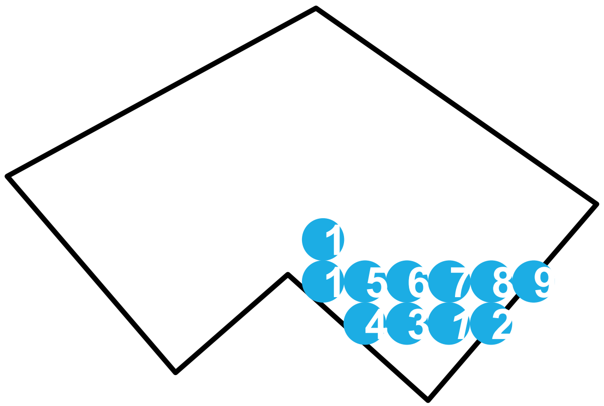
6. Repeat steps 3 through 5 until $x \geq y$.

Polygons

Filling Polygons

- Scan-line fill algorithm
  - Inside-Outside tests
- Boundary fill algorithm

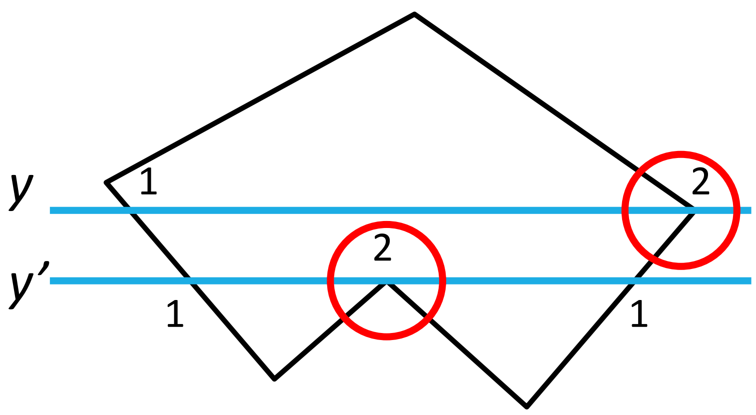


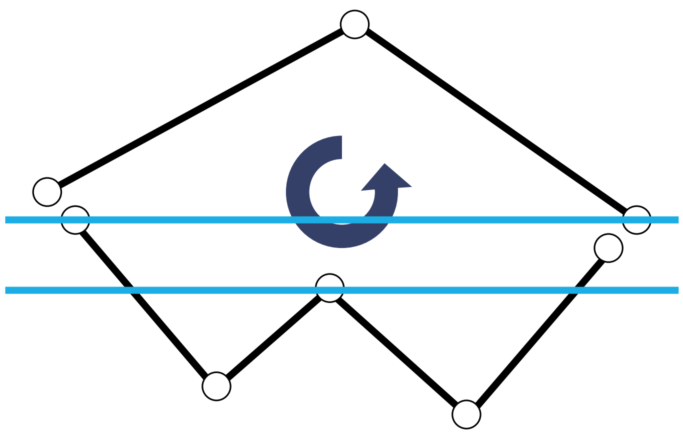


Scan-Line Polygon Fill

Topological Difference between 2 Scan lines

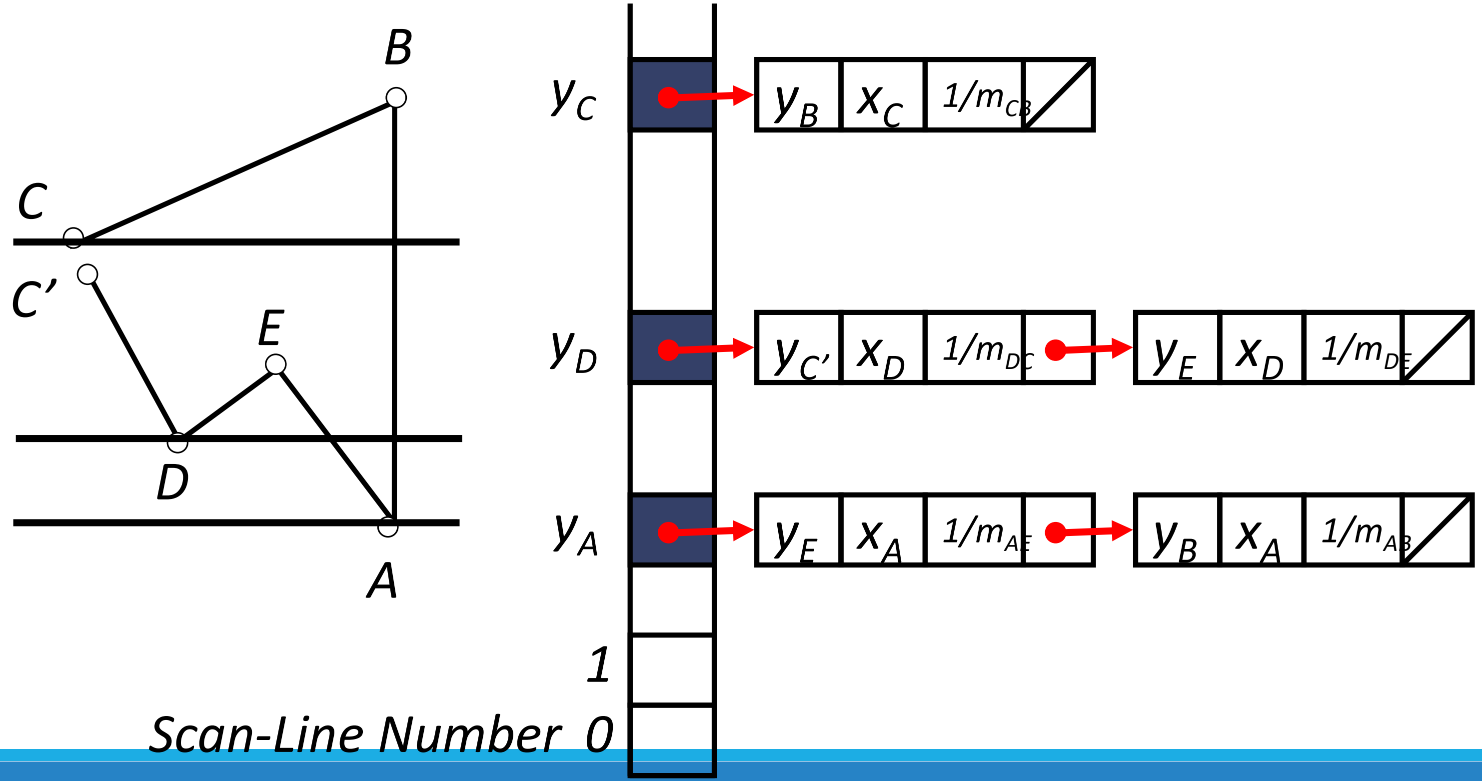
- y : intersection edges are opposite sides
- y’ : intersection edges are same side





Scan-Line Polygon Fill (cont.)

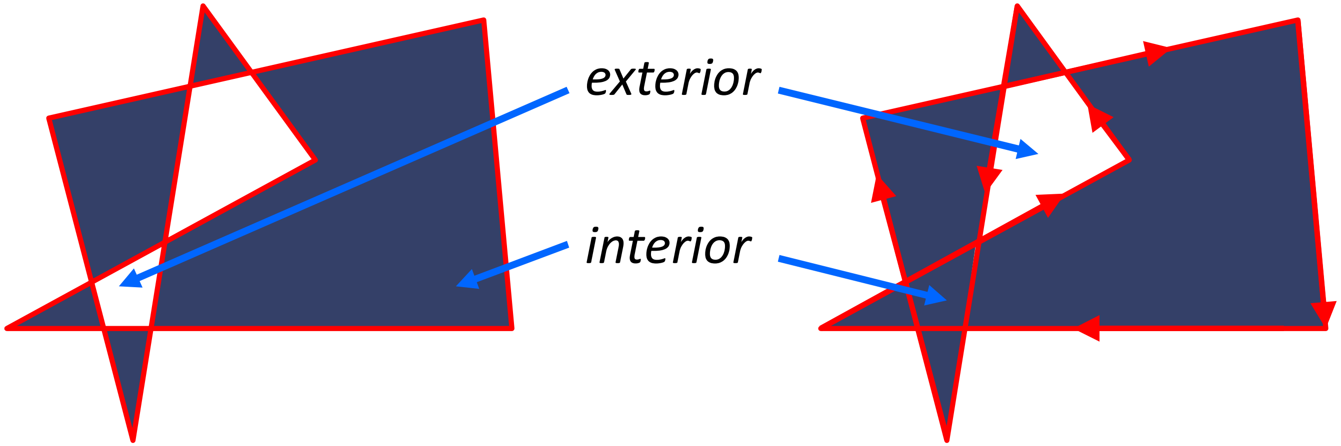
Edge Sorted Table



Inside-Outside Tests

Self-Intersections

- Odd-Even rule
- Nonzero winding number rule



Boundary-Fill Algorithm

Proceed to Neighboring Pixels

- 4-Connected
- 8-Connected