Bresenham's Algorithm

Lecture 31

Robb T. Koether

Hampden-Sydney College

Mon, Nov 27, 2017

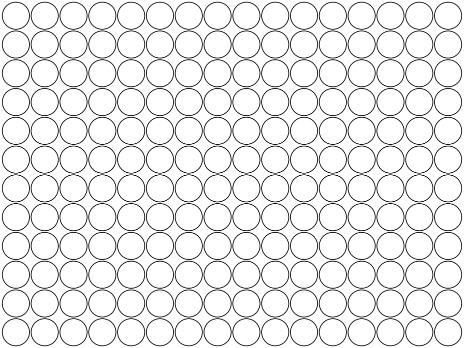
Outline

1. Pixel Coordinates
2. Bresenham's Midpoint Algorithm
   - The Decision Function
   - The Algorithm
   - Examples
3. Assignment

Outline

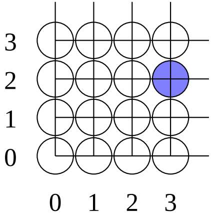
Pixels and Screen Coordinates

- We think of the viewport as consisting of a rectangular array of pixels.

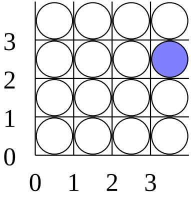


Pixels and Screen Coordinates

- A basic question is, what are the coordinates of the pixels?
- Are they integers, e.g., $(3, 2)$?
- Or, are they half-integers, e.g. $\left(3\frac{1}{2}, 2\frac{1}{2}\right)$?



Integers



Half-integers

Pixels and Screen Coordinates

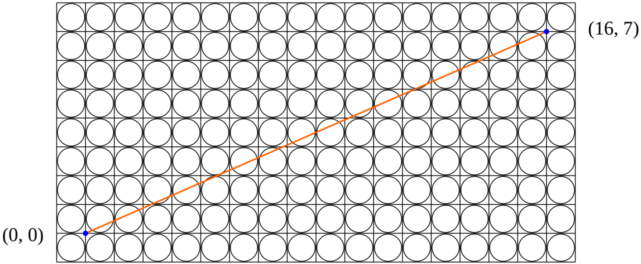
- OpenGL adopts the view that they are half-integers.
- We will conform to OpenGL's view, so that our results will match OpenGL's results.

Outline

1. Pixel Coordinates
2. Bresenham's Midpoint Algorithm
   - The Decision Function
   - The Algorithm
   - Examples
3. Assignment

Rasterizing Lines

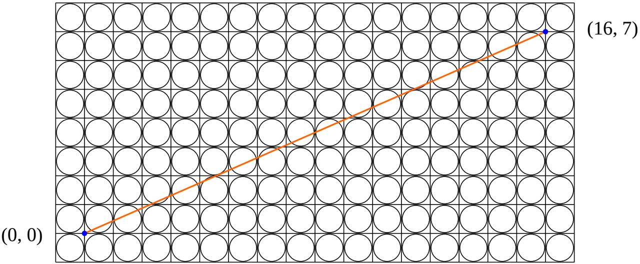
- All lines will go from one intersection of grid lines to another.



Rasterizing Lines

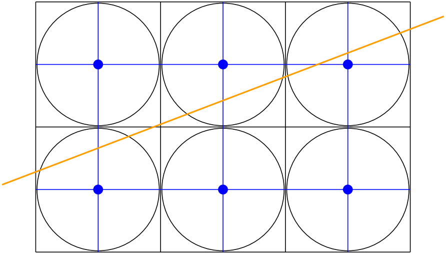
- In what follows, we will assume that the line has a slope between 0 and 1.
- That is, the change in x is greater than the change in y.
- The other cases are “easily” adapted from this case.

Rasterizing Lines



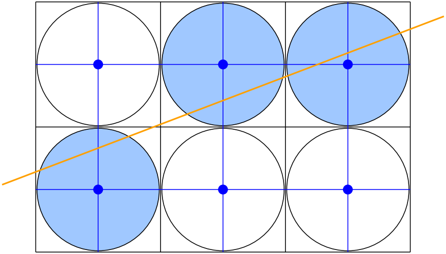
When drawing a line on the screen, how do we decide which pixels to color?

Rasterizing Lines



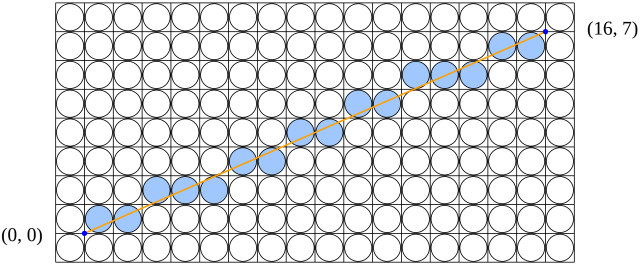
In each vertical column, choose the pixel whose center is closest to the line.

Rasterizing Lines



In each vertical column, choose the pixel whose center is closest to the line.

Rasterizing Lines



In each vertical column, choose the pixel whose center is closest to the line.

Choosing the Pixel

- Let the equation of the line be

$$y = mx + b.$$

- Given that we have just colored pixel (p_x, p_y), the next pixel to color is either $(p_x + 1, p_y)$ or $(p_x + 1, p_y + 1)$.
- Why?

Choosing the Pixel

- What is the most efficient way to determine which point to color?
- Based on the equation of a line $y = mx + b$, we could compute $m(p_x + 1) + b$ and then compare it to p_y and $p_y + 1$ to see which is closer.
- Why is this not a good idea?

The Decision Function

- We need a function which, when evaluated, will tell us whether to increment y or to keep y the same.
- Let the line go from $A = (a_x, a_y)$ to $B = (b_x, b_y)$.
- The equation of the line can be written

$$y - a_y = m(x - a_x)$$
$$y - a_y = \left(\frac{b_y - a_y}{b_x - a_x}\right)(x - a_x),$$

or

$$(b_x - a_x)(y - a_y) = (b_y - a_y)(x - a_x).$$

Outline

The Decision Function

- Let $W = b_x - a_x$ (the width).
- Let $H = b_y - a_y$ (the height).
- Note that $W \geq H \geq 0$ by our assumption.
- Then the equation can be written

$$-W(y - a_y) + H(x - a_x) = 0.$$

- Let

$$F(x, y) = -2W(y - a_y) + 2H(x - a_x).$$

- The factor of 2 is introduced to avoid the fraction $\frac{1}{2}$ later.

The Decision Function

- We will call $F(x, y)$ the decision function.

The Decision Function

- We will call $F(x, y)$ the decision function.
- Note that if we increment x by 1, then $F(x, y)$ *increases* by $2H$.

The Decision Function

- We will call $F(x, y)$ the decision function.
- Note that if we increment x by 1, then $F(x, y)$ *increases* by $2H$.
- And if we increment y by 1, then $F(x, y)$ *decreases* by $2W$.

The Decision Function

- We will call $F(x, y)$ the decision function.
- Note that if we increment x by 1, then $F(x, y)$ *increases* by $2H$.
- And if we increment y by 1, then $F(x, y)$ *decreases* by $2W$.
- And if we do both, then $F(x, y)$ *decreases* by $2(W - H)$.

The Decision Function

- Note that
  - If $F(x, y) < 0$, then the line is below (x, y).
  - If $F(x, y) = 0$, then the line passes through (x, y).
  - If $F(x, y) > 0$, then the line is above (x, y).

Bresenham's Midpoint Algorithm

- Suppose we have shaded pixel $P = (p_x, p_y)$ (centered at $(p_x + \frac{1}{2}, p_y + \frac{1}{2})$).
- Now, in the next column, we need to decide whether to shade the
  - The lower pixel $L = (p_x + 1\frac{1}{2}, p_y + \frac{1}{2})$, or
  - The upper pixel $U = (p_x + 1\frac{1}{2}, p_y + 1\frac{1}{2})$.
- Let M be the point midway between U and L.
- Then $M = \left(p_x + 1\frac{1}{2}, p_y + 1\right)$.

- If the line is above M, then we should color U.
- If the line is below M, then we should color L.
- That is,
  - If $F\left(p_x + 1\frac{1}{2}, p_y + 1\right) > 0$, then color U.
  - If $F\left(p_x + 1\frac{1}{2}, p_y + 1\right) < 0$, then color L.

- In the first column of pixels, the midpoint is $M_1 = \left(a_x + \frac{1}{2}, a_y + 1\right)$.
- The line passes below M_1.
- Indeed, the line passes below the center of the pixel $\left(a_x + \frac{1}{2}, a_y + \frac{1}{2}\right)$.
- For that first midpoint M_1,

$$F_1 = F\left(a_x + \frac{1}{2}, a_y + 1\right) = -2W + H < 0.$$

Bresenham's Midpoint Algorithm

- In the second column of pixels, the midpoint is $M_2 = \left(a_x + 1\frac{1}{2}, a_y + 1\right)$.
- The value of F is

$$\begin{aligned} F_2 &= F\left(a_x + 1\tfrac{1}{2}, a_y + 1\right) \\ &= F_1 + 2H \\ &= 3H - 2W, \end{aligned}$$

  which may be positive or negative.
- We compute it, test it, and decide whether to shade U or L.

Bresenham's Midpoint Algorithm

- To find a recursive formula for the decision, consider what happens as we go from pixel $P_i = (p_x, p_y)$ to pixel P_{i+1}.
- That is, from $x = p_x + \frac{1}{2}$ to $x = p_x + 1\frac{1}{2}$.
- There are two cases:
  - We shaded L in the previous step.
  - We shaded U in the previous step.

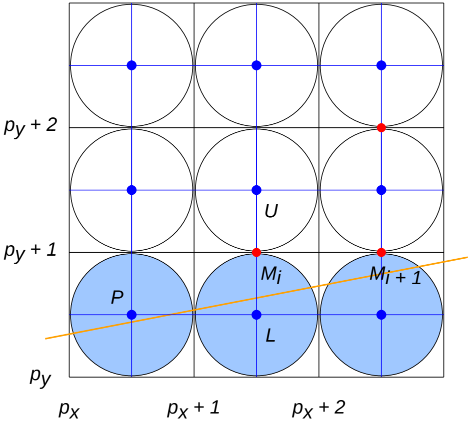
Bresenham's Midpoint Algorithm

- Suppose first that we shaded L.
  - Then the midpoint considered was $M_i = \left(p_x + \frac{1}{2}, p_y + 1\right)$ and $F_i < 0$.
  - Because we shaded L, the next midpoint to consider is $M_{i+1} = \left(p_x + 1\frac{1}{2}, p_y + 1\right)$.
  - We calculate

$$F_{i+1} = F\left(p_x + 1\tfrac{1}{2}, p_y + 1\right) = F_i + 2H,$$

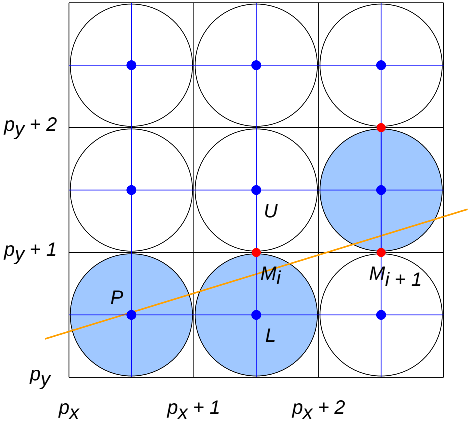
which may be positive or negative.

Bresenham's Midpoint Algorithm



Right - Right

Bresenham's Midpoint Algorithm



Right - Up

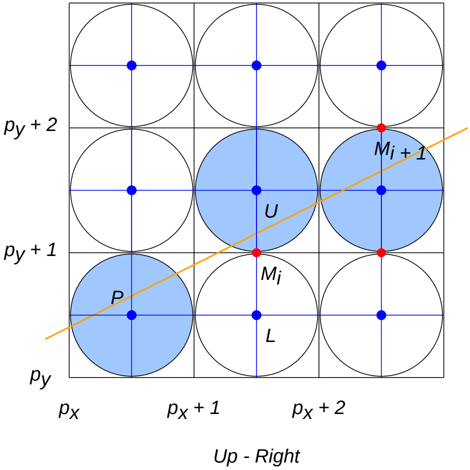
Bresenham's Midpoint Algorithm

- Now suppose that we shaded U.
  - Then the midpoint considered was $M_i = \left(p_x + \frac{1}{2}, p_y + 1\right)$ and $F_i > 0$.
  - Because we shaded U, the next midpoint to consider is $M_{i+1} = \left(p_x + 1\frac{1}{2}, p_y + 2\right)$.
  - We calculate

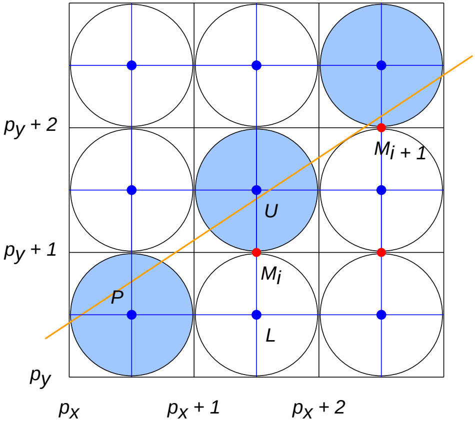
$$F_{i+1} = F\left(p_x + 1\tfrac{1}{2}, p_y + 2\right) = F_i - 2(W - H),$$

    which may or may not be positive.

Bresenham's Midpoint Algorithm



Bresenham's Midpoint Algorithm



Up - Up

Outline

Bresenham's Midpoint Algorithm

- Summarizing the two cases:
  - If the previous value of F was negative, then the next value is $2H$ larger.
  - If the previous value of F was positive, then the next value is $2(W - H)$ smaller.
- This allows us to move quickly from one decision to the next, since we need only add or subtract a constant.
- We do not need to evaluate $F(x, y)$ each time.

Bresenham's Midpoint Algorithm

- To start the process, the first midpoint to be considered is

$$M_2 = \left(a_x + 1\tfrac{1}{2}, a_y + 1\right).$$

- The value of $F\left(a_x + 1\frac{1}{2}, a_y + 1\right)$ is $3H - 2W$.
- Therefore, the initial value of the decision function is $3H - 2W$.

Bresenham's Midpoint Algorithm

Bresenham's Midpoint Algorithm

- Initialize $x = a_x$, $y = a_y$, $F = 3H - 2W$.
- Repeat until $x = b_x$.
  - Increment x.
  - If $F < 0$, then
    - Add $2H$ to F.
  - Else if $F > 0$, then
    - Subtract $2(W - H)$ from F.
    - Increment y.

Bresenham's Midpoint Algorithm

- This algorithm produces the coordinates of the grid point to the lower left of the pixel to be shaded.

Outline

Example

- Rasterize the line from $(0, 0)$ to $(16, 7)$.
  - $W = 16$, $H = 7$.
  - $2H = 14$, $2(W - H) = 18$, $3H - 2W = -11$.

x	F	Δy	y	x	F	Δy	y	x	F	Δy	y
0				6				12			
1				7				13			
2				8				14			
3				9				15			
4				10				16			
5				11							

Example

- Rasterize the line from $(0,0)$ to $(16,7)$.
  - $W = 16$, $H = 7$.
  - $2H = 14$, $2(W - H) = 18$, $3H - 2W = -11$.

x	F	Δy	y	x	F	Δy	y	x	F	Δy	y
0	-11			6	9			12	-3		
1	3			7	-9			13	11		
2	-15			8	5			14	-7		
3	-1			9	-13			15	7		
4	13			10	1			16	-11		
5	-5			11	-17						

Example

- Rasterize the line from $(0, 0)$ to $(16, 7)$.
  - $W = 16$, $H = 7$.
  - $2H = 14$, $2(W - H) = 18$, $3H - 2W = -11$.

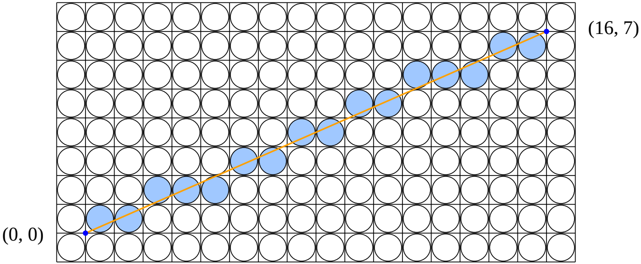
x	F	Δy	y	x	F	Δy	y	x	F	Δy	y
0	-11	0		6	9	1		12	-3	0	
1	3	1		7	-9	0		13	11	1	
2	-15	0		8	5	1		14	-7	0	
3	-1	0		9	-13	0		15	7	1	
4	13	1		10	1	1		16	-11	0	
5	-5	0		11	-17	0					

Example

- Rasterize the line from $(0,0)$ to $(16,7)$.
  - $W = 16$, $H = 7$.
  - $2H = 14$, $2(W - H) = 18$, $3H - 2W = -11$.

x	F	Δy	y	x	F	Δy	y	x	F	Δy	y
0	-11	0	0	6	9	1	2	12	-3	0	5
1	3	1	0	7	-9	0	3	13	11	1	5
2	-15	0	1	8	5	1	3	14	-7	0	6
3	-1	0	1	9	-13	0	4	15	7	1	6
4	13	1	1	10	1	1	4	16	-11	0	7
5	-5	0	2	11	-17	0	5				

Lines and Pixels



Another Example

- Rasterize the line from $(3, 5)$ to $(16, 14)$.
  - $W = 13$, $H = 9$.
  - $2H = 18$, $2(W - H) = 8$, $3H - 2W = 1$.

x	F	Δy	y	x	F	Δy	y	x	F	Δy	y
3				8				13			
4				9				14			
5				10				15			
6				11				16			
7				12							

Another Example

- Rasterize the line from $(3, 5)$ to $(16, 14)$.
  - $W = 13$, $H = 9$.
  - $2H = 18$, $2(W - H) = 8$, $3H - 2W = 1$.

x	F	Δy	y	x	F	Δy	y	x	F	Δy	y
3	1			8	13			13	-1		
4	-7			9	5			14	17		
5	11			10	-3			15	9		
6	3			11	15			16	1		
7	-5			12	7						

Another Example

- Rasterize the line from $(3, 5)$ to $(16, 14)$.
  - $W = 13$, $H = 9$.
  - $2H = 18$, $2(W - H) = 8$, $3H - 2W = 1$.

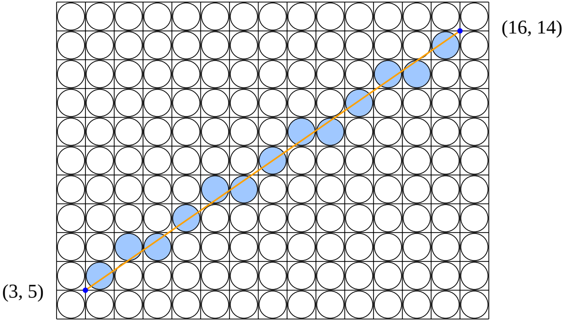
x	F	Δy	y	x	F	Δy	y	x	F	Δy	y
3	1	1		8	13	1		13	-1	0	
4	-7	0		9	5	1		14	17	1	
5	11	1		10	-3	0		15	9	1	
6	3	1		11	15	1		16	1	1	
7	-5	0		12	7	1					

Another Example

- Rasterize the line from $(3, 5)$ to $(16, 14)$.
  - $W = 13$, $H = 9$.
  - $2H = 18$, $2(W - H) = 8$, $3H - 2W = 1$.

x	F	Δy	y	x	F	Δy	y	x	F	Δy	y
3	1	1	5	8	13	1	8	13	-1	0	12
4	-7	0	6	9	5	1	9	14	17	1	12
5	11	1	6	10	-3	0	10	15	9	1	13
6	3	1	7	11	15	1	10	16	1	1	14
7	-5	0	8	12	7	1	11				

Example



Bresenham's Algorithm

Bresenham's Algorithm

```
void bresenham(Point2D A, Point2D B)
{
    int y = A.y;
    int W = B.x - A.x;
    int H = B.y - A.y;
    int H2 = 2*H;
    int WH2 = 2*(W - H);
    int F = 3*H - 2*W;
    for (int x = A.x; x < B.x; x++)
    {
        setPixel(x, y);
        if (F < 0)
            F += H2;
        else
        {
            F -= WH2;
            y++;
        }
    }
}
```

Outline

Assignment

Assignment

- Rasterize the line from $(0, 0)$ to $(8, 5)$.
- Rasterize the line from $(3, 8)$ to $(22, 39)$.
- Rasterize the line from $(0, 0)$ to $(5, 8)$ by adapting the rasterization of the line from $(0, 0)$ to $(8, 5)$.
- Rasterize the line from $(0, 0)$ to $(-5, 8)$ by adapting the rasterization of the line from $(0, 0)$ to $(8, 5)$.