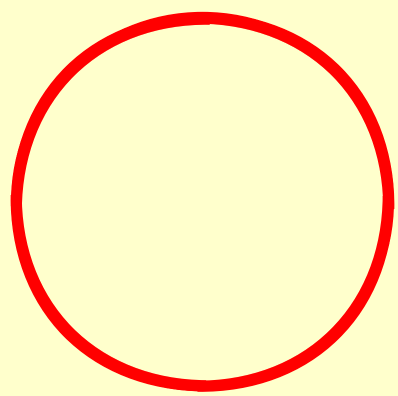
Rasterization: Bresenham Circles

CS4600 *Intro to Computer Graphics*
From Rich Riesenfeld
Fall 2015

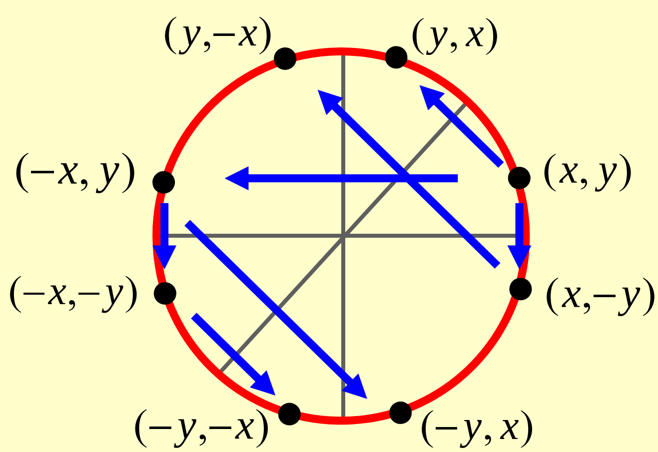
More Raster Line Issues

- Fat lines with multiple pixel width
- Symmetric lines
- End point geometry – how should it look?
- Generating curves, e.g., circles, etc.
- Jaggies, staircase effect, aliasing...

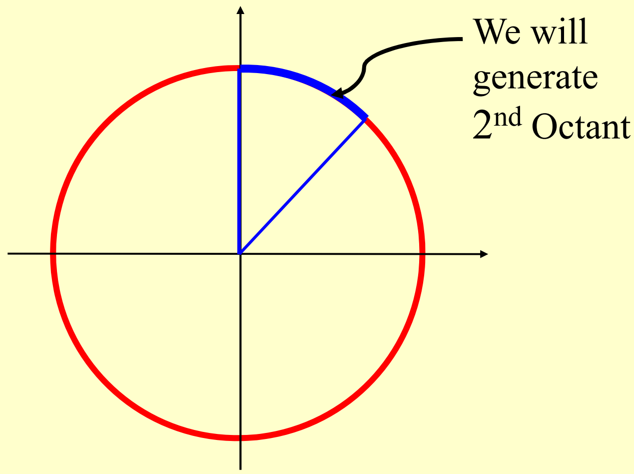
Generating Circles



Exploit 8-Point Symmetry



Only 1 Octant Needed



Generating pt (x,y) gives

the following 8 pts by symmetry:

$\{(x,y), (-x,y), (-x,-y), (x,-y),$
$(y,x), (-y,x), (-y,-x), (y,-x)\}$

2^{nd} Octant Is a Good Arc

- It is a function in this domain
  - single-valued
  - no vertical tangents: $|slope| \leq 1$
- Lends itself to Bresenham
  - only need consider ?
  - E or SE

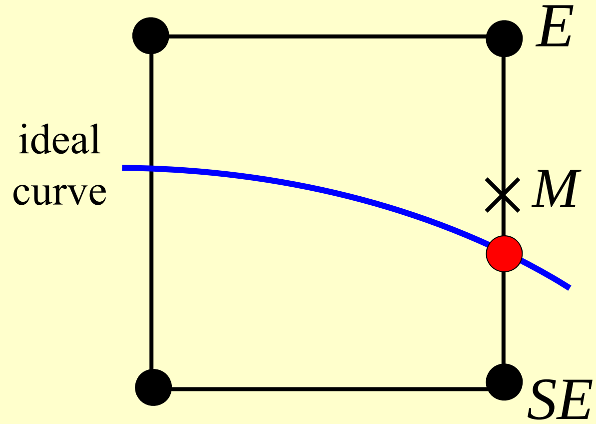
Implicit Eq's for Circle

- Let $F(x,y) = x^2 + y^2 - r^2$
- For (x,y) on the circle, $F(x,y) = 0$
- So, $F(x,y) > 0 \Rightarrow (x,y)$ *Outside*
- And, $F(x,y) < 0 \Rightarrow (x,y)$ *Inside*

Choose E or SE

- Function is $x^2 + y^2 - r^2 = 0$
- So, $F(M) \geq 0 \Rightarrow SE$

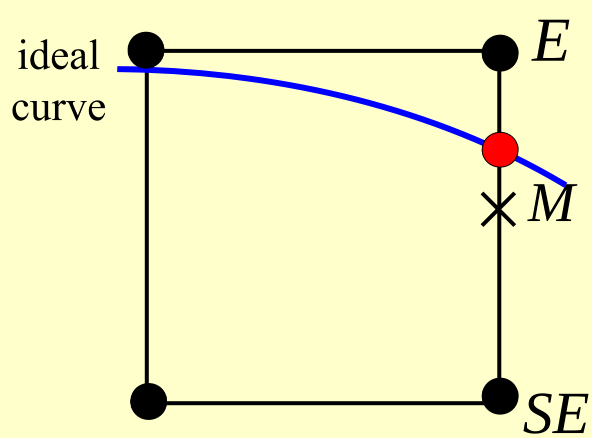
$F(M) \geq 0 \Rightarrow SE$



Choose E or SE

- Function is $x^2 + y^2 - r^2 = 0$
- So, $F(M) \geq 0 \Rightarrow SE$
- And, $F(M) < 0 \Rightarrow E$

$F(M) < 0 \Rightarrow E$

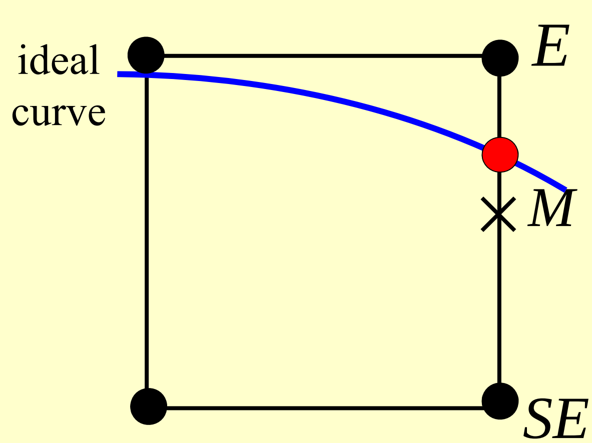


Decision Variable d

Again, we let,

$$d = F(M)$$

Look at Case 1: E



$$d_{old} < 0 \Rightarrow E$$

$$d_{old} = F(x_p + 1, y_p - \tfrac{1}{2})$$

$$= (x_p + 1)^2 + (y_p - \tfrac{1}{2})^2 - r^2$$

$$d_{old} < 0 \Rightarrow E$$

$$d_{new} = F(x_p + 2, y_p - \tfrac{1}{2})$$

$$= (x_p + 2)^2 + (y_p - \tfrac{1}{2})^2 - r^2$$

$$\underline{d_{old} < 0 \Rightarrow E}$$

$$d_{new} = d_{old} + (2x_p + 3)$$

Since, $d_{new} - d_{old}$

$$(x_p+2)^2 - (x_p+1)^2 = (x_p^2 + 4x_p + 4) - (x_p^2 + 2x_p + 1)$$
$$= 2x_p + 3$$

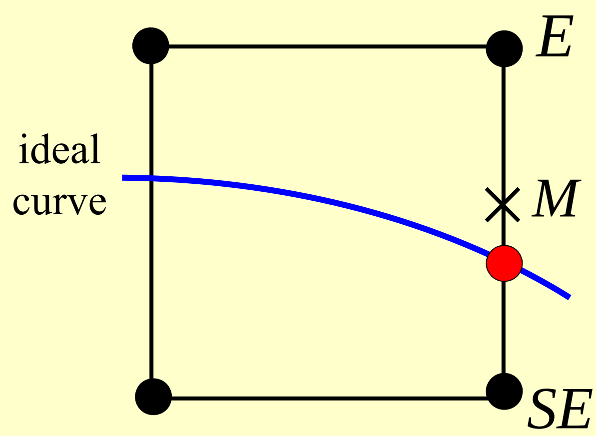
$$\underline{d_{old} < 0 \Rightarrow E}$$

$$d_{new} = d_{old} + \Delta_E \ ,$$

where,

$$\boxed{\Delta_E = 2x_p + 3}$$

Look at Case 2: *SE*



$d_{old} \geq 0 \Rightarrow SE$

$$d_{new} = F(x_p + 2, y_p - \tfrac{3}{2})$$

$$= (x_p + 2)^2 + (y_p - \tfrac{3}{2})^2 - r^2$$

$$d_{new} = d_{old} + (2x_p - 2y_p + 5)$$

Because,…, straightforward manipulation

$$\underline{d_{old} \geq 0 \quad \Rightarrow \quad SE}$$

$$d_{new} - d_{old} =$$

$$(x_p + 2)^2 + (y_p - \frac{3}{2})^2 - r^2 - \left[(x_p + 1)^2 + (y_p - \frac{1}{2})^2 - r^2 \right]$$

$$= (2x_p + 3) + y_p^2 - 3y_p + \frac{9}{4} - \left[y_p^2 - y_p + \frac{1}{4} \right]$$

$$\underline{d_{old} \geq 0 \quad \Rightarrow \quad SE}$$

$$d_{new} - d_{old} =$$

$$(x_p + 2)^2 + (y_p - \frac{3}{2})^2 - \cancel{r}^2 - \left[(x_p + 1)^2 + (y_p - \frac{1}{2})^2 - \cancel{r}^2 \right]$$

$$= (2x_p + 3) + y_p^2 - 3y_p + \frac{9}{4} - \left[y_p^2 - y_p + \frac{1}{4} \right]$$

$$d_{old} \geq 0 \quad \Rightarrow \quad SE$$

$$d_{new} - d_{old} =$$

$$(x_p + 2)^2 + (y_p - \frac{3}{2})^2 - \cancel{r^2} - \left[(x_p + 1)^2 + (y_p - \frac{1}{2})^2 - \cancel{r^2} \right]$$

$$= \ (2x_p + 3) + \cancel{y_p^2} - 3y_p + \frac{9}{4} - \left[\cancel{y_p^2} - y_p + \frac{1}{4} \right]$$

$$d_{old} \geq 0 \quad \Rightarrow \quad SE$$

$$d_{new} - d_{old} =$$

$$\underbrace{(2x_p + 3)}_{\text{From } \Delta_E \text{ calculation}} + \underbrace{(-3y_p + \frac{9}{4})}_{\text{From } new \text{ y-coordinate}} - \underbrace{(-y_p + \frac{1}{4})}_{\text{From } old \text{ y-coordinate}}$$

$$\underline{d_{old} \geq 0 \quad \Rightarrow \quad SE}$$

$$d_{new} - d_{old} =$$

$$\underbrace{(2x_p + 3)}_{\text{From } \Delta_E \text{ calculation}} + \underbrace{(-3y_p + \tfrac{9}{4})}_{\text{From } new \text{ y-coordinate}} - \underbrace{(-y_p + \tfrac{1}{4})}_{\text{From } old \text{ y-coordinate}}$$

$$\underline{d_{old} \geq 0 \quad \Rightarrow \quad SE}$$

$$d_{new} - d_{old} =$$

$$\underbrace{(2x_p + 3)}_{\text{From } \Delta_E \text{ calculation}} + \underbrace{(-3y_p + \tfrac{9}{4})}_{\text{From } new \text{ y-coordinate}} - \underbrace{(-y_p + \tfrac{1}{4})}_{\text{From } old \text{ y-coordinate}}$$

$$\underline{d_{old} \geq 0 \quad \Rightarrow \quad SE}$$

$$d_{new} - d_{old} =$$

$$\underbrace{(2x_p + 3)}_{\text{From } \Delta_E \text{ calculation}} + \underbrace{(-3y_p + \tfrac{9}{4})}_{\text{From \textit{new} y-coordinate}} - \underbrace{(-y_p + \tfrac{1}{4})}_{\text{From \textit{old} y-coordinate}}$$

$$\Delta_{SE} = 2x_p - 2y_p + 5$$

$$\underline{d_{old} \geq 0 \quad \Rightarrow \quad SE}$$

I.e.,

$$d_{new} = d_{old} + (2x_p - 2y_p + 5)$$

$$= d_{old} + \Delta_{SE}$$

$$\boxed{\Delta_{SE} = 2x_p - 2y_p + 5}$$

Note: Δ´s Not Constant

Δ_E *and* Δ_{SE}

depend on values of x_p and y_p

Summary

- Δ´s are no longer constant over entire line
- Algorithm structure is *exactly* the same
- Major difference from the line algorithm
  - Δ is re-evaluated at each step
  - Requires *real* arithmetic

Initial Condition

- Let *r* be an integer. Start at $(0, r)$
- Next midpoint *M* lies at $(1, r - \frac{1}{2})$
- So, $F(1, r - \frac{1}{2}) = 1 + (r^2 - r + \frac{1}{4}) - r^2$

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

Ellipses

- Evaluation is analogous
- Structure is same
- Have to work out the Δ´s

Getting to Integers

- Note the previous algorithm involves *real* arithmetic
- Can we modify the algorithm to use integer arithmetic?

Integer Circle Algorithm

- Define a shift decision variable

$$h = d - \frac{1}{4}$$

- In the code, plug in $d = h + \frac{1}{4}$

Integer Circle Algorithm

- Now, the initialization is $h = 1 - r$
- So the initial value becomes

$$F(1, r - \frac{1}{2}) - \frac{1}{4} = (\frac{5}{4} - r) - \frac{1}{4}$$
$$= 1 - r$$

Integer Circle Algorithm

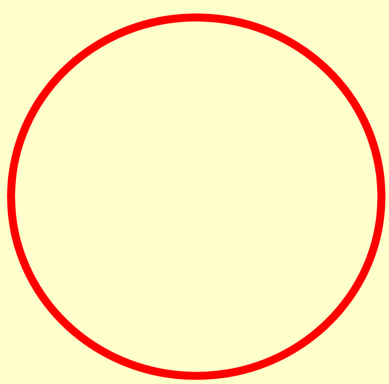
- Then, $d < 0$ becomes $h < -\frac{1}{4}$
- Since h an integer

$$\boxed{h < -\frac{1}{4} \quad \Leftrightarrow \quad h < 0}$$

Integer Circle Algorithm

- But, h begins as an integer
- And, h gets incremented by integer
- Hence, we have an integer circle algorithm
- Note: Sufficient to test for $h < 0$

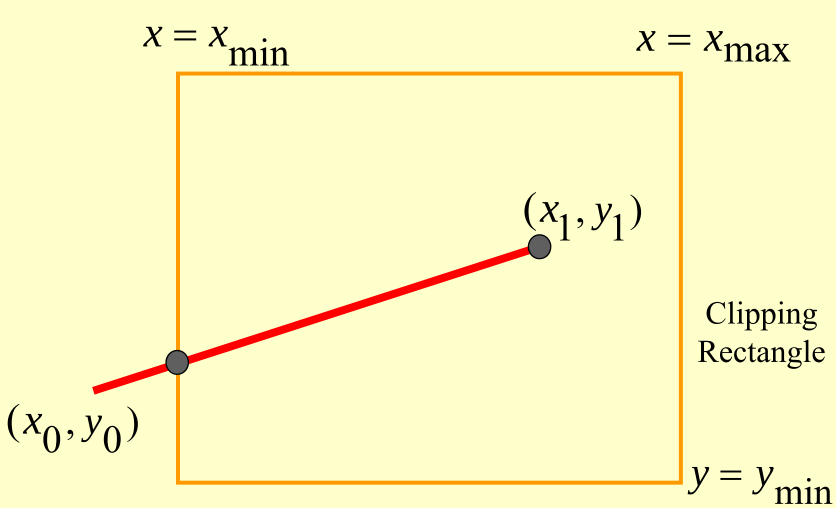
End of Bresenham Circles



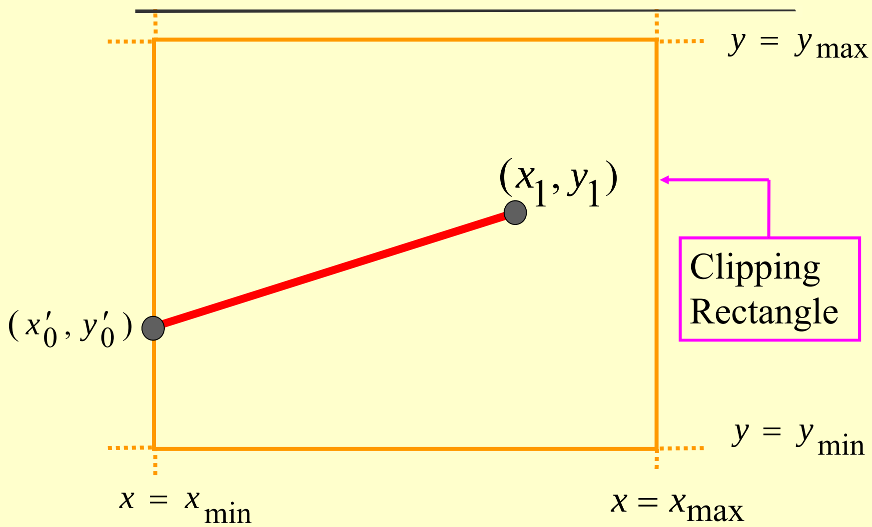
Another Digital Line Issue

- Clipping Bresenham lines
- The integer slope is not the true slope
- Have to be careful
- More issues to follow

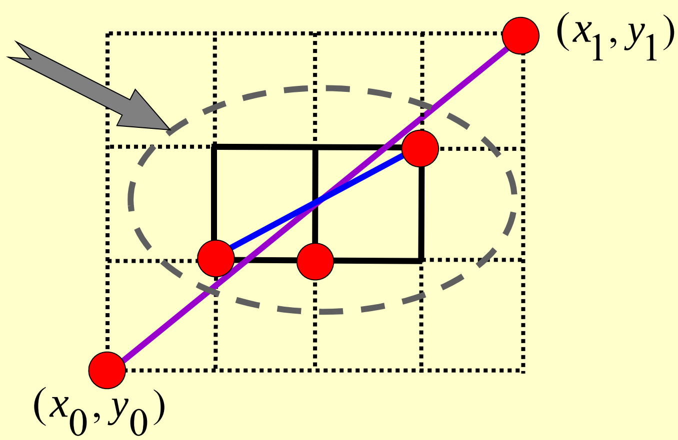
Line Clipping Problem



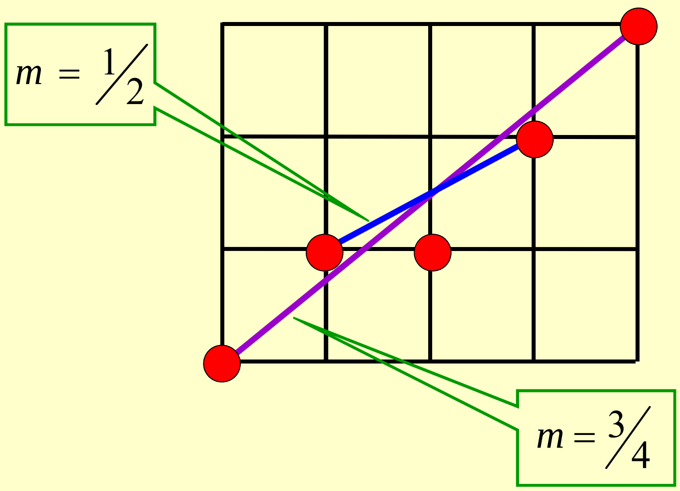
Clipped Line



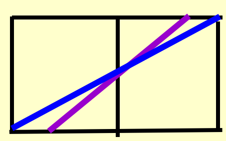
Drawing Clipped Lines



Clipped Line Has Different Slope !

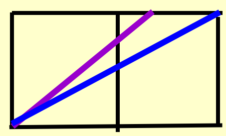


Pick Right Slope to Reproduce Original Line Segment



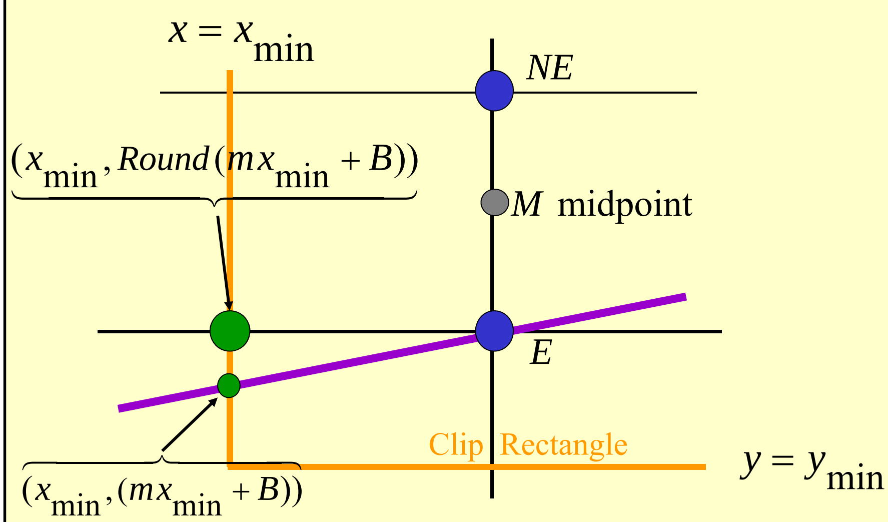
Zoom of previous situation

Pick Right Slope to Reproduce Original Line Segment

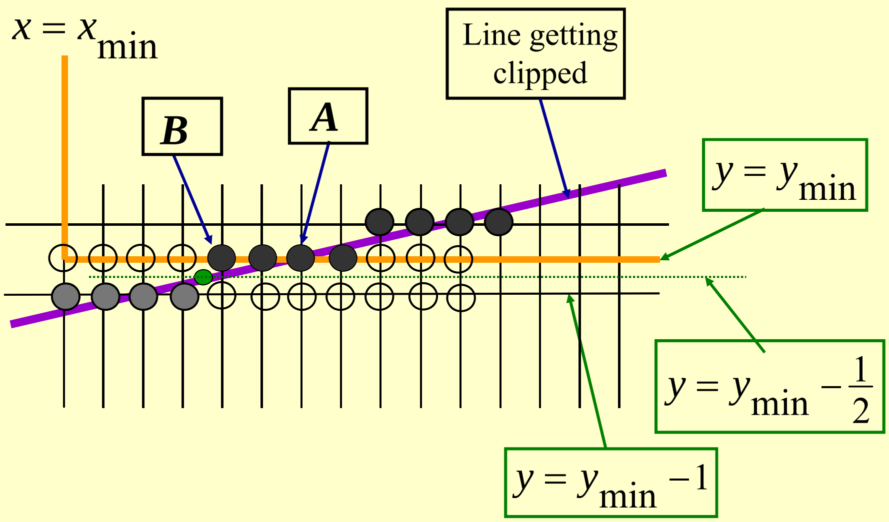


Zoom of previous situation

Clipping Against $x = x_{min}$



Clipping Against $y = y_{min}$



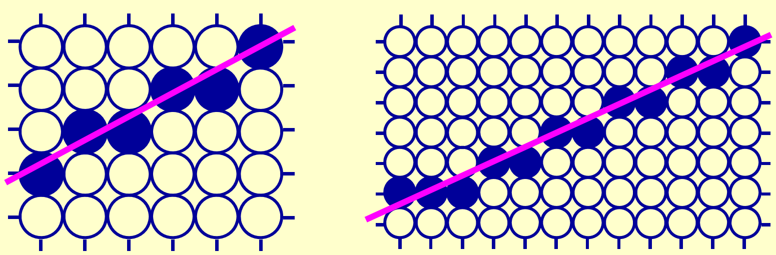
Clipping Against $y = y_{min}$

- Situation is complicated
- Multiple pixels involved at $(y = y_{min})$
- Want all of those pixels as "*in*"
- Analytic $\cap$, rounding x gives A
- We want point B

Clipping Against $y = y_{min}$

- Use $\boxed{Line} \cap \boxed{y = y_{min} - \frac{1}{2}}$
- Round *up* to nearest integer x
- This yields point B, the desired result

Jaggies-Manifestation of Aliasing



Added resolution helps, but does not directly address underlying issue of *aliasing*

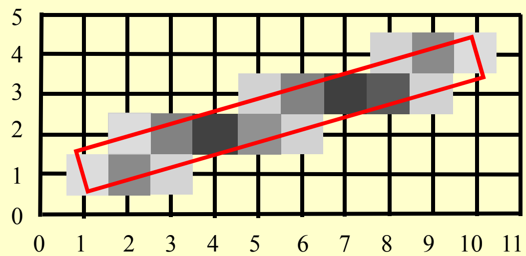
Jaggies and Aliasing

- To represent a line with discrete pixel values is to sample finitely a continuous function
- Jaggies are visual manifestation, artifacts, resulting from information loss
- The term aliasing is a complicated, unintuitive phenomenon which will be defined later

Jaggies and Aliasing

- Doubling resolution in x and y reduces the effect of the problem, but does not fix it
- Doubling resolution costs 4 times memory, memory bandwidth and scan conversion time!

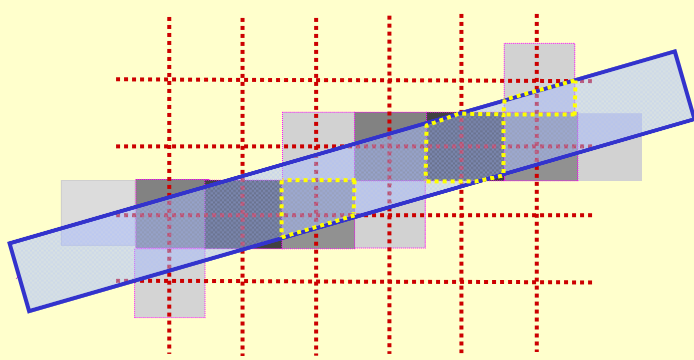
Anti-aliasing



Pixel Space

Pixel intensity (darkness, in this case) is proportional to area covered by line

Anti-aliasing



Pixel Space

Pixel intensity (darkness, in this case) is proportional to area covered by line

Anti-aliasing

- Set each pixel's intensity value proportional to its area of overlap (i.e. sub-area) covered by primitive
- Not more than 1 pixel/column for lines with

$$0 < slope < 1$$

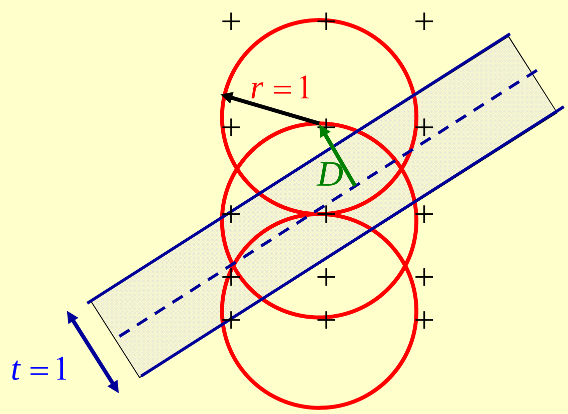
Gupta-Sproull Algorithm -1

- Standard Bresenham chooses E or NE
- Incrementally compute distance D from chosen pixel to center of line
- Vary pixel intensity by value of D
- Do this for line above and below

Gupta-Sproull Algorithm -2

- Use coarse (4-bit, say) lookup table for intensity : $Filter\,(D, t)$
- Note, $Filter$ value depends *only* on D and t, not the slope of line**!** (Very clever)
- For *line_width* $t = 1$ geometry and associated calculations greatly simplify

Cone Filter for Weighted Area Sampling



Unit thickness line intersects no more than 3 pixels

Observations

- Lines are complicated
- Many aspects to consider
- We omitted many
- What about intensity of

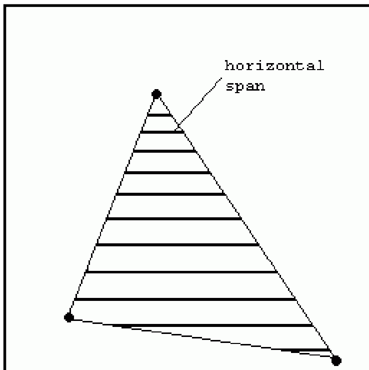
$y = x$ vs $y = 0$?

Rasterization: Triangles

CS4600 *Intro to Computer Graphics*

From Rich Riesenfeld

Fall 2014



Rasterize This!

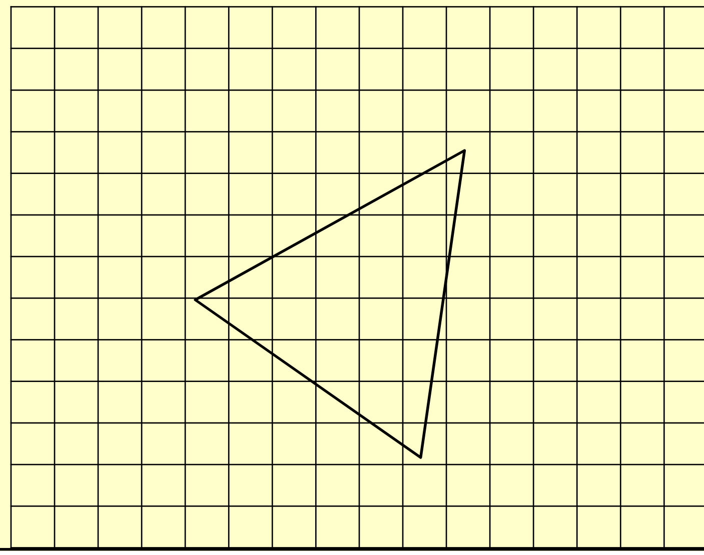
(Rasterization intuition)

- When we render a triangle we want to determine if a pixel is within a triangle. (barycentric coords)
- Calculate the color of the pixel (use barycentric coors).
- Draw the pixel.
- Repeat until the triangle is appropriately filled.

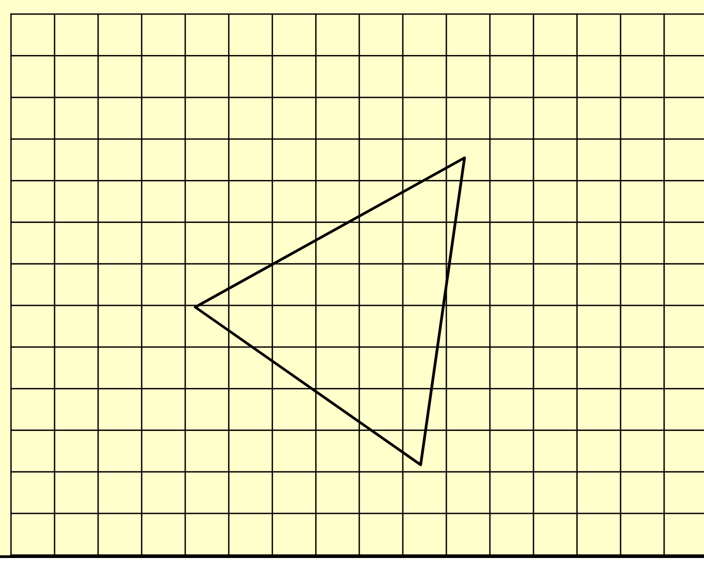
Rasterization Pseudo Code

```
drawTriangle2D(x_a, y_a, x_b, y_b, x_c, y_c)
{
   for all x in screen_x
      for all y in screen_y
         compute(α, β, γ) for (x, y)
            if(α ∈ [0, 1] and β ∈ [0, 1] and γ ∈ [0, 1])
               color = compute_color(...)
                  put_pixel(x, y, color)
}
```

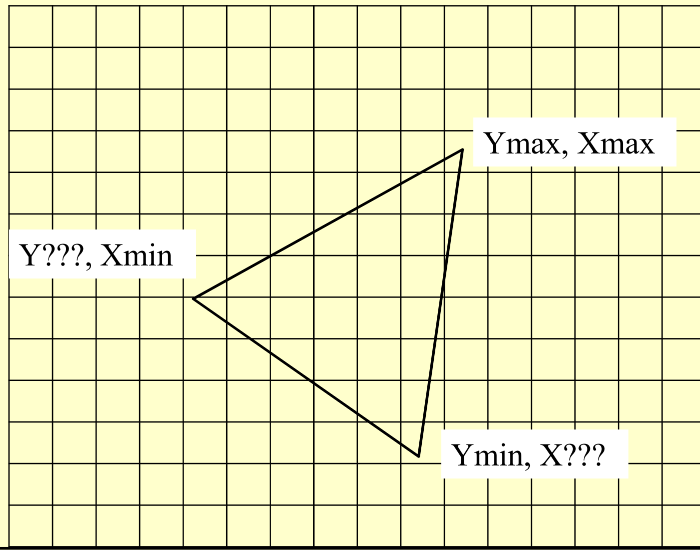
Rasterization



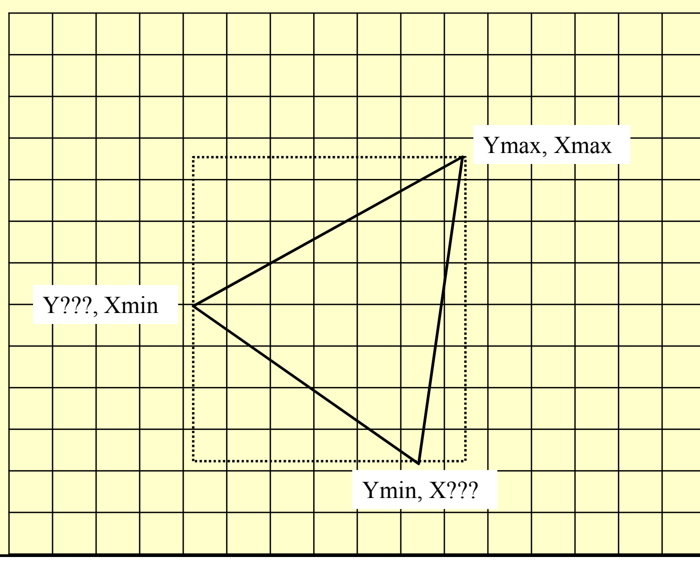
Rasterization



Rasterization



Bounding Box

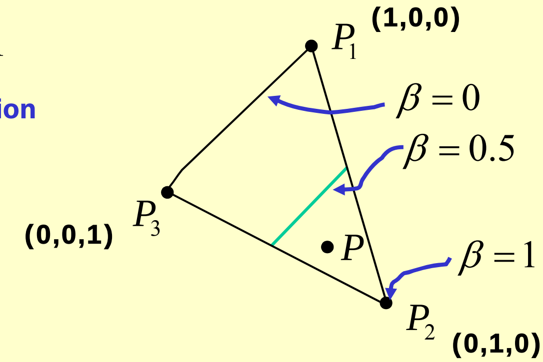


Barycentric Coordinates

• weighted combination of vertices

$$\begin{cases} P = \alpha \cdot P_1 + \beta \cdot P_2 + \gamma \cdot P_3 \\ \alpha + \beta + \gamma = 1 \\ 0 \le \alpha, \beta, \gamma \le 1 \end{cases}$$

"convex combination of points"



Barycentric Coordinates for Interpolation

• how to compute α, β, γ ?

– use bilinear interpolation or plane equations

interpolate α, β, γ

$$\alpha = a \cdot x + b \cdot y + c \cdot z + d$$
$$\beta = \ldots$$

– once computed, use to interpolate any # of parameters from their vertex values

$$x = \alpha \cdot x_1 + \beta \cdot x_2 + \gamma \cdot x_3$$
$$r = \alpha \cdot r_1 + \beta \cdot r_2 + \gamma \cdot r_3$$
$$g = \alpha \cdot g_1 + \beta \cdot g_2 + \gamma \cdot g_3$$

etc.

Interpolatation: Gouraud Shading

- need linear function over triangle that yields original vertex colors at vertices
- use barycentric coordinates for this
  - every pixel in interior gets colors resulting from mixing colors of vertices with weights corresponding to barycentric coordinates
  - color at pixels is affine combination of colors at vertices

$$Color(\alpha \cdot \mathbf{x}_1 + \beta \cdot \mathbf{x}_2 + \gamma \cdot \mathbf{x}_3) :=$$
$$\alpha \cdot Color(\mathbf{x}_1) + \beta \cdot Color(\mathbf{x}_2) + \gamma \cdot Color(\mathbf{x}_3)$$

Gouraud Shading Scanline Alg

- algorithm
  - modify scanline algorithm for polygon scan-conversion :
    - linearly interpolate colors along edges of triangle to obtain colors for endpoints of span of pixels
    - linearly interpolate colors from these endpoints within the scanline

$$\frac{X_{\max} - X_{cur}}{X_{\max} - X_{\min}} * C_{\min} + \left(1 - \frac{X_{\max} - X_{cur}}{X_{\max} - X_{\min}}\right) * C_{\max}$$