

Bresenham Circles

CS5600 Intro to Computer Graphics

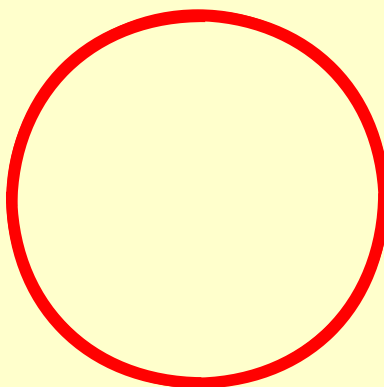
From Rich Riesenfeld

Spring 2009

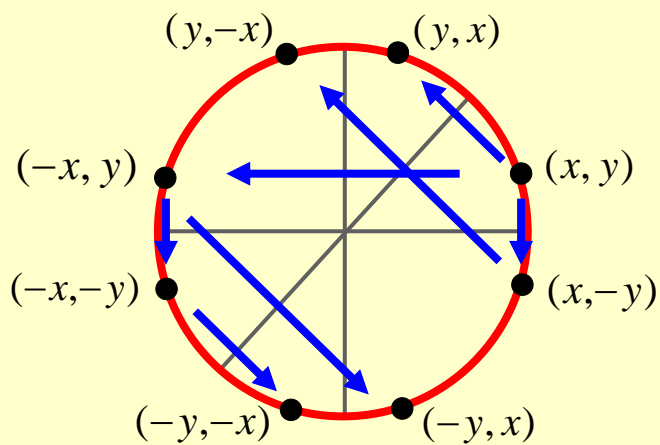
More Raster Line Issues

- 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...

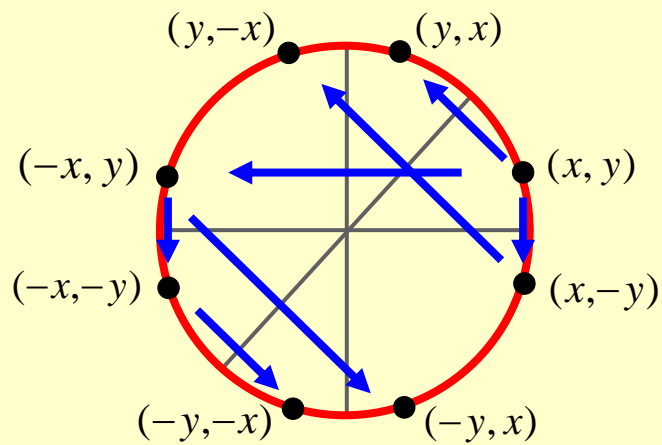
Generating Circles



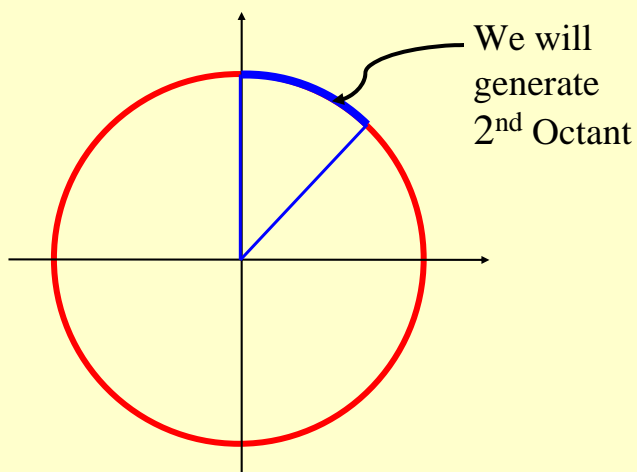
Exploit 8-Point Symmetry



Once More: 8-Pt Symmetry



Only 1 Octant Needed



Generating pt (x,y) gives

the following 8 pts by symmetry:

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

2nd Octant Is a Good Arc

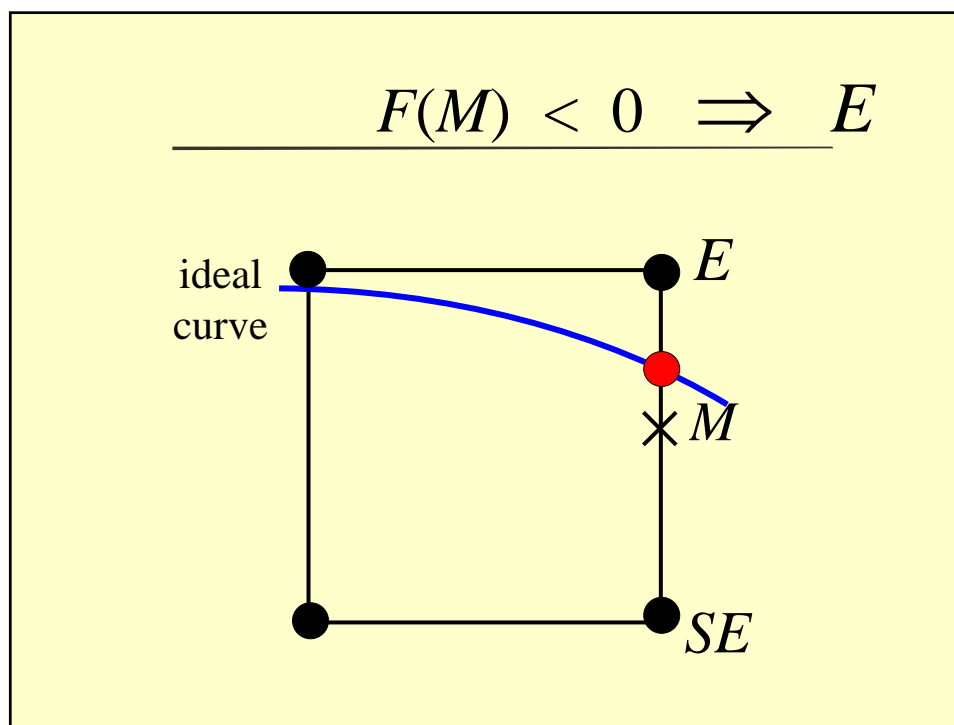
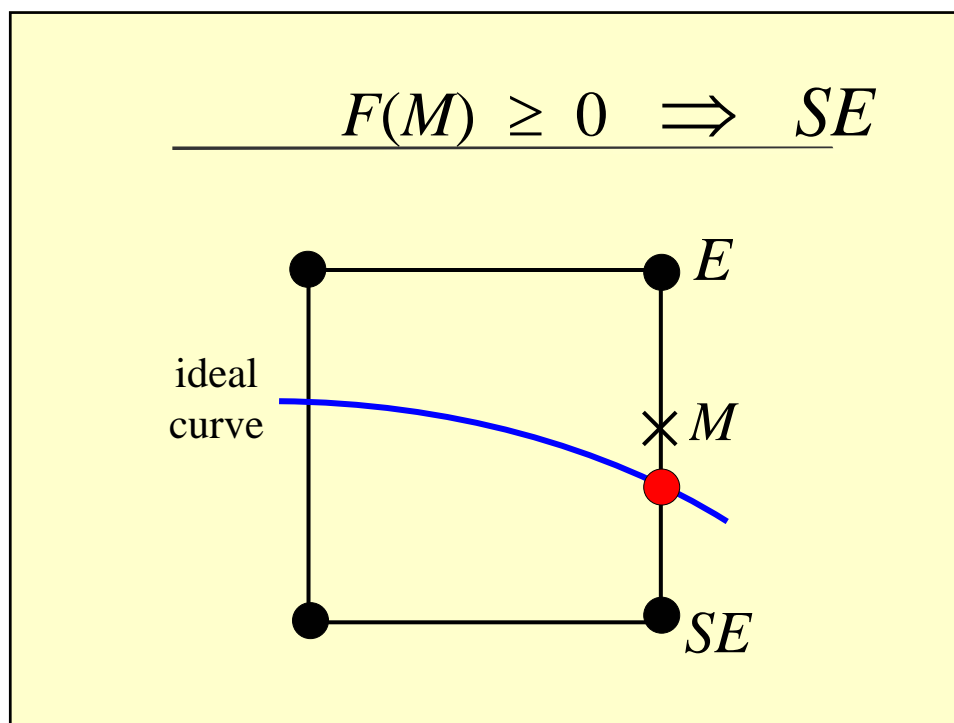
- It is a function in this domain
 - single-valued
 - no vertical tangents: $|slope| \leq 1$
- Lends itself to Bresenham
 - only need consider E or SE

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, $F(M) \geq 0 \Rightarrow SE$
- And, $F(M) < 0 \Rightarrow E$

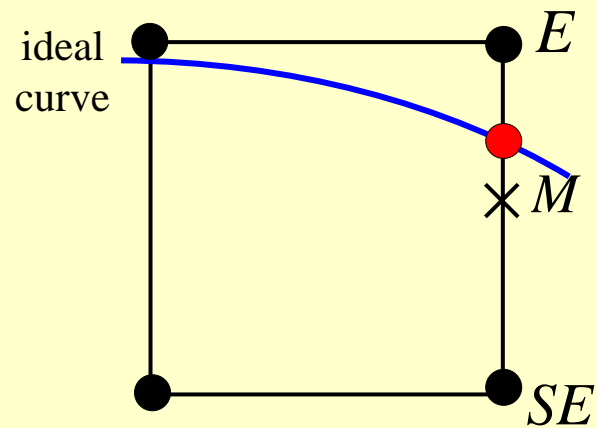


Decision Variable d

Again, we let,

$$d = F(M)$$

Look at Case 1: E



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

$$\begin{aligned}d_{old} &= F(x_p + 1, y_p - 1/2) \\ &= (x_p + 1)^2 + (y_p - 1/2)^2 - r^2\end{aligned}$$

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

$$\begin{aligned}d_{new} &= F(x_p + 2, y_p - 1/2) \\ &= (x_p + 2)^2 + (y_p - 1/2)^2 - r^2\end{aligned}$$

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

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

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

$$\begin{aligned} (x_p + 2)^2 - (x_p + 1)^2 &= (x_p^2 + 4x_p + 4) - (x_p^2 + 2x_p + 1) \\ &= 2x_p + 3 \end{aligned}$$

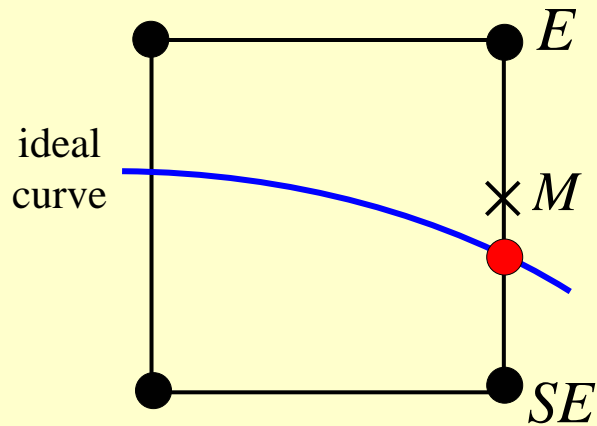
$$\underline{d_{old} < 0 \Rightarrow E}$$

$$d_{new} = d_{old} + \Delta E ,$$

where,

$$\Delta E = 2x_p + 3$$

Look at Case 2: *SE*



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

$$\begin{aligned} d_{new} &= F(x_p + 2, y_p - 3/2) \\ &= (x_p + 2)^2 + (y_p - 3/2)^2 - r^2 \\ d_{new} &= d_{old} + (2x_p - 2y_p + 5) \end{aligned}$$

Because, ..., straightforward manipulation

$$\underline{d_{old} \geq 0 \implies SE}$$

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

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

$$\underline{d_{old} \geq 0 \implies SE}$$

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

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

$$\underline{d_{old} \geq 0 \implies SE}$$

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

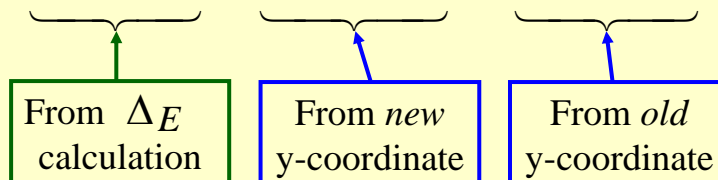
$$(x_p + 2)^2 + (y_p - \frac{3}{2})^2 - r^2 - \left[(x_p + 1)^2 + (y_p - \frac{1}{2})^2 - r^2 \right]$$

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

$$\underline{d_{old} \geq 0 \implies SE}$$

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

$$(2x_p + 3) + (-3y_p + \frac{9}{4}) - (-y_p + \frac{1}{4})$$



$$\text{I.e., } \frac{d_{old} \geq 0 \implies SE}{d_{new} = d_{old} + (2x_p - 2y_p + 5)}$$
$$= d_{old} + \Delta_{SE}$$

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

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

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

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

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

Integer Circle Algorithm

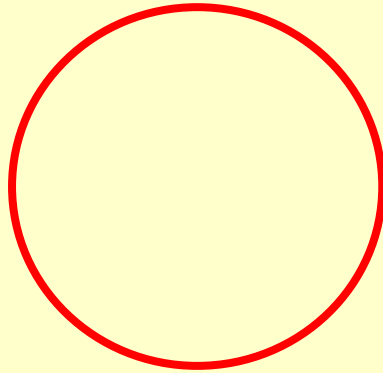
- Then, $d < 0$ becomes $h < -\frac{1}{4}$
- Since h an integer

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

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 $h < 0$

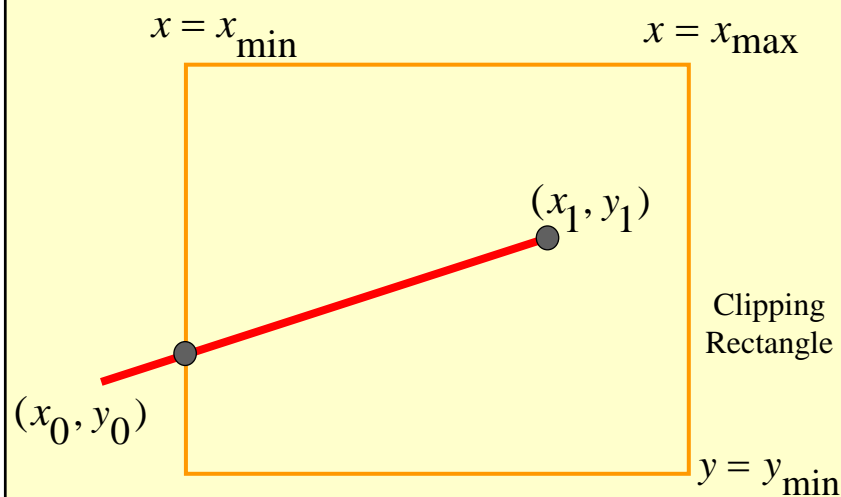
End of Bresenham Circles



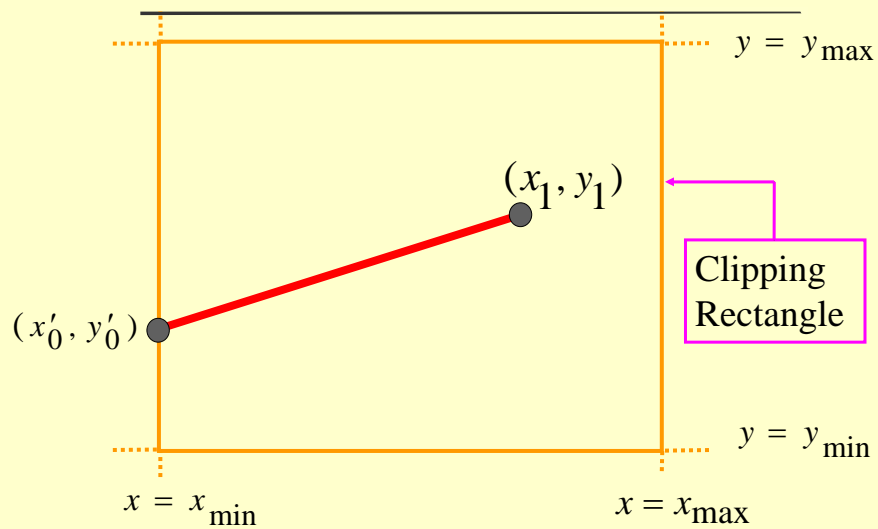
Another Digital Line Issue

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

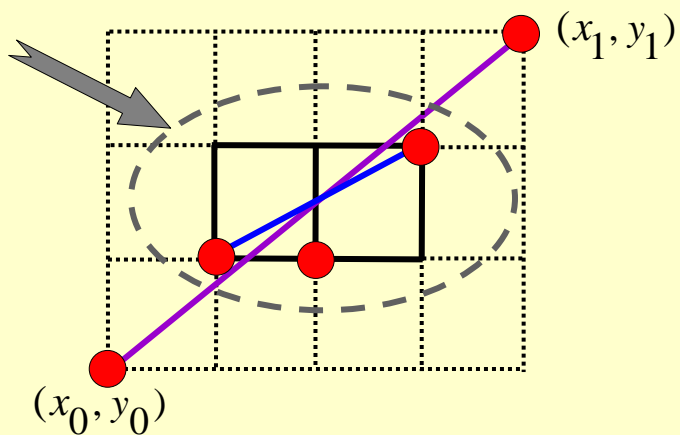
Line Clipping Problem



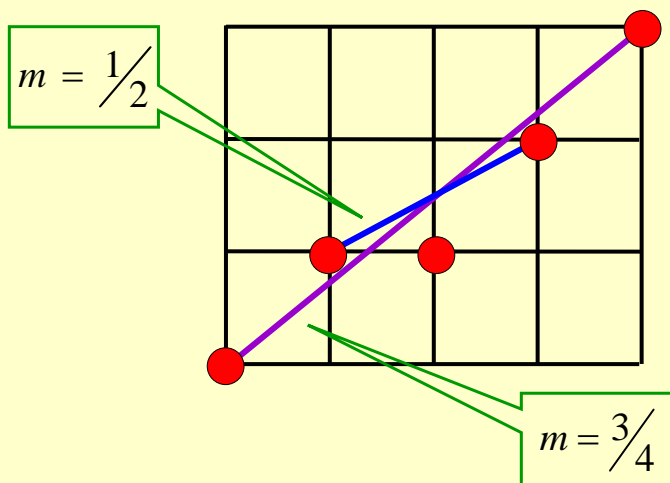
Clipped Line



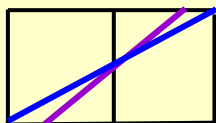
Drawing Clipped Lines



Clipped Line Has Different Slope !

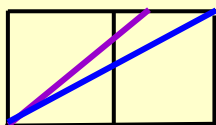


Pick Right Slope to Reproduce
Original Line Segment

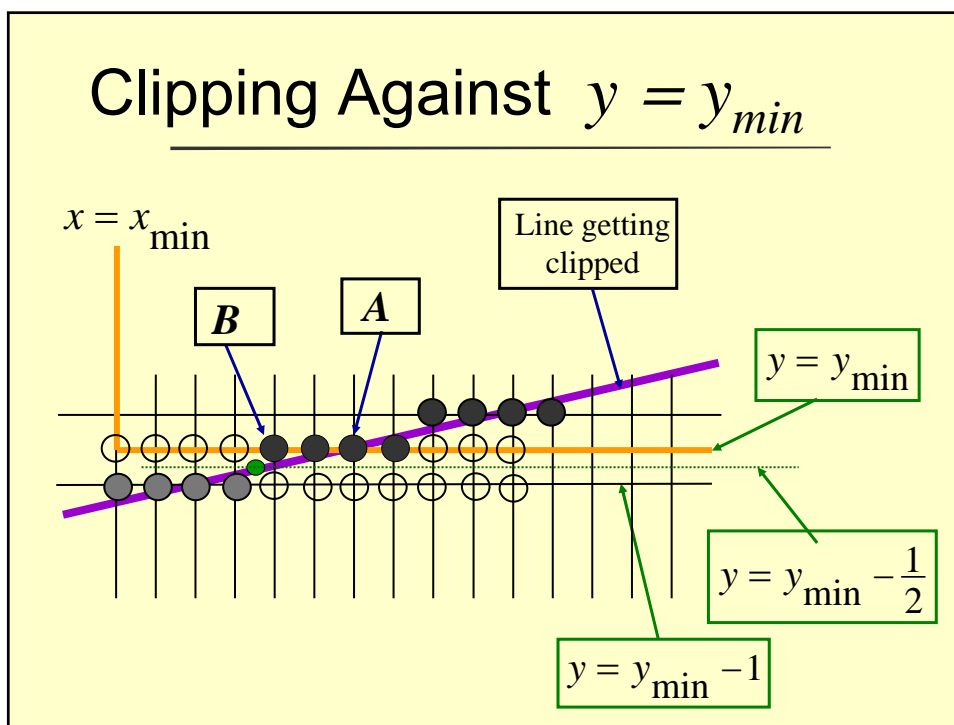
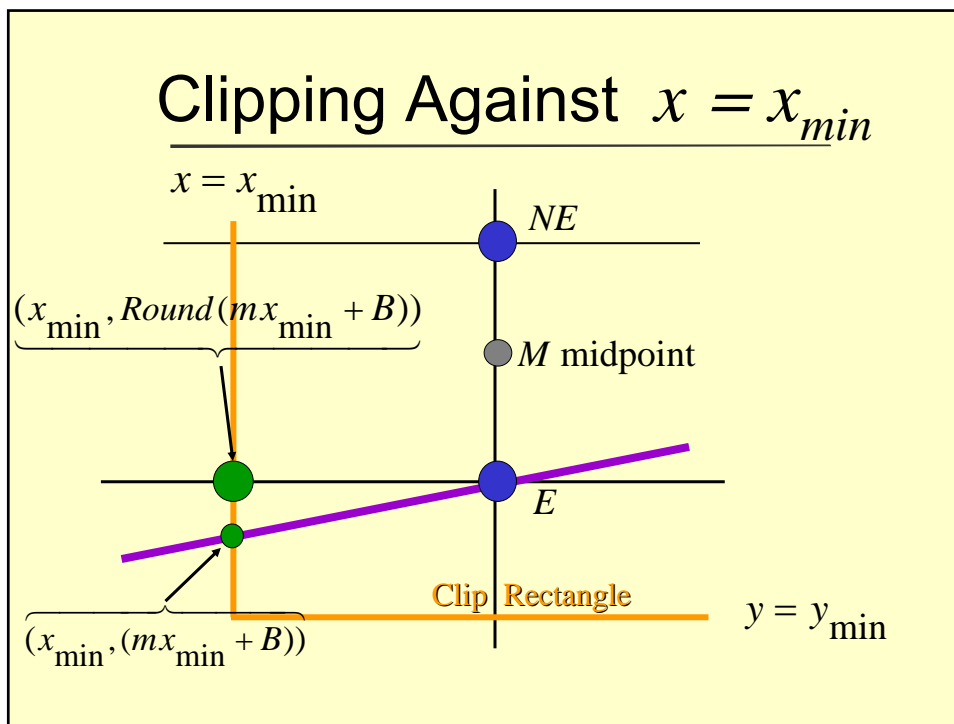


Zoom of previous situation

Pick Right Slope to Reproduce
Original Line Segment



Zoom of previous situation



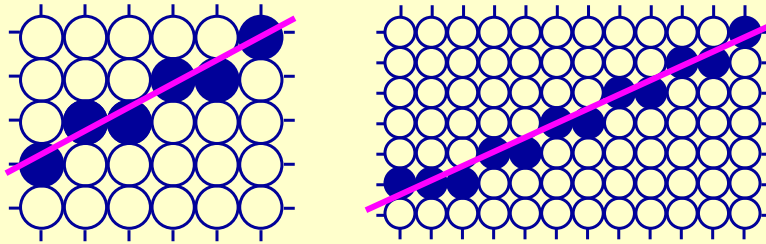
Clipping Against $y = y_{min}$

- 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 $\boxed{Line} \cap \boxed{y = y_{min} - 1/2}$
- 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*

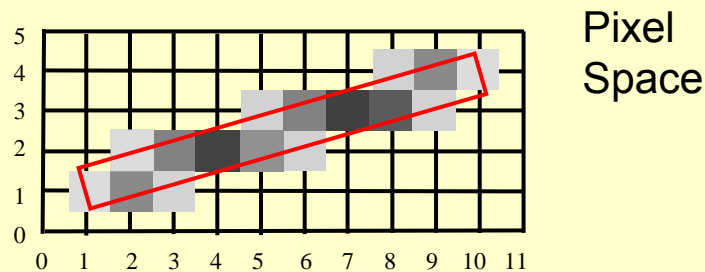
Jaggies and Aliasing

- 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

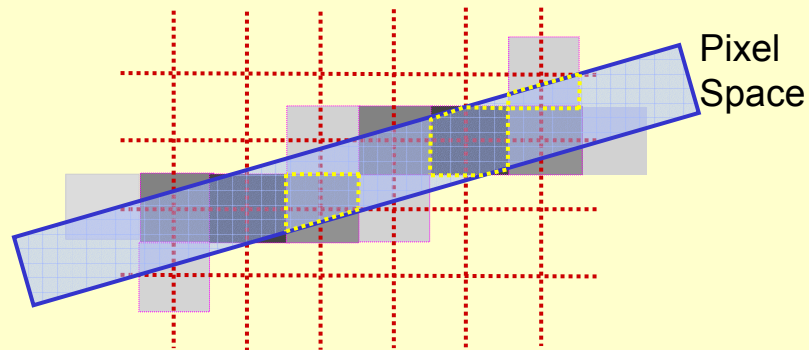
- Doubling resolution in x and y reduces the effect of the problem, but does not fix it
- Doubling resolution costs 4 times memory, memory bandwidth and scan conversion time!

Anti-aliasing



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

Anti-aliasing



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

$$0 < slope < 1$$

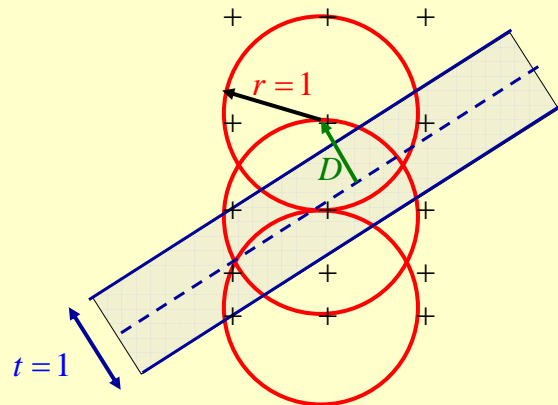
Gupta-Sproull Algorithm -1

- 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

$$y = x \quad \text{vs} \quad y = 0 \quad ?$$

The End

Bresenham Circles

Lecture Set 3