Line Rasterization

MIT EECS 6.837

Frédo Durand and Seth Teller

Administrative

- Prof. Teller office hours today: 5-6 in 4-035
- Assignment 2
  - Due Friday 27 at 5pm
  - Model a scene using iv files from assignt 1
  - Lighting: at least one spot and one other light
  - You are welcome to add new geometry
  - Pay attention to the requirements!
  - Shadows are challenging! (and no required)

Review of previous lecture?

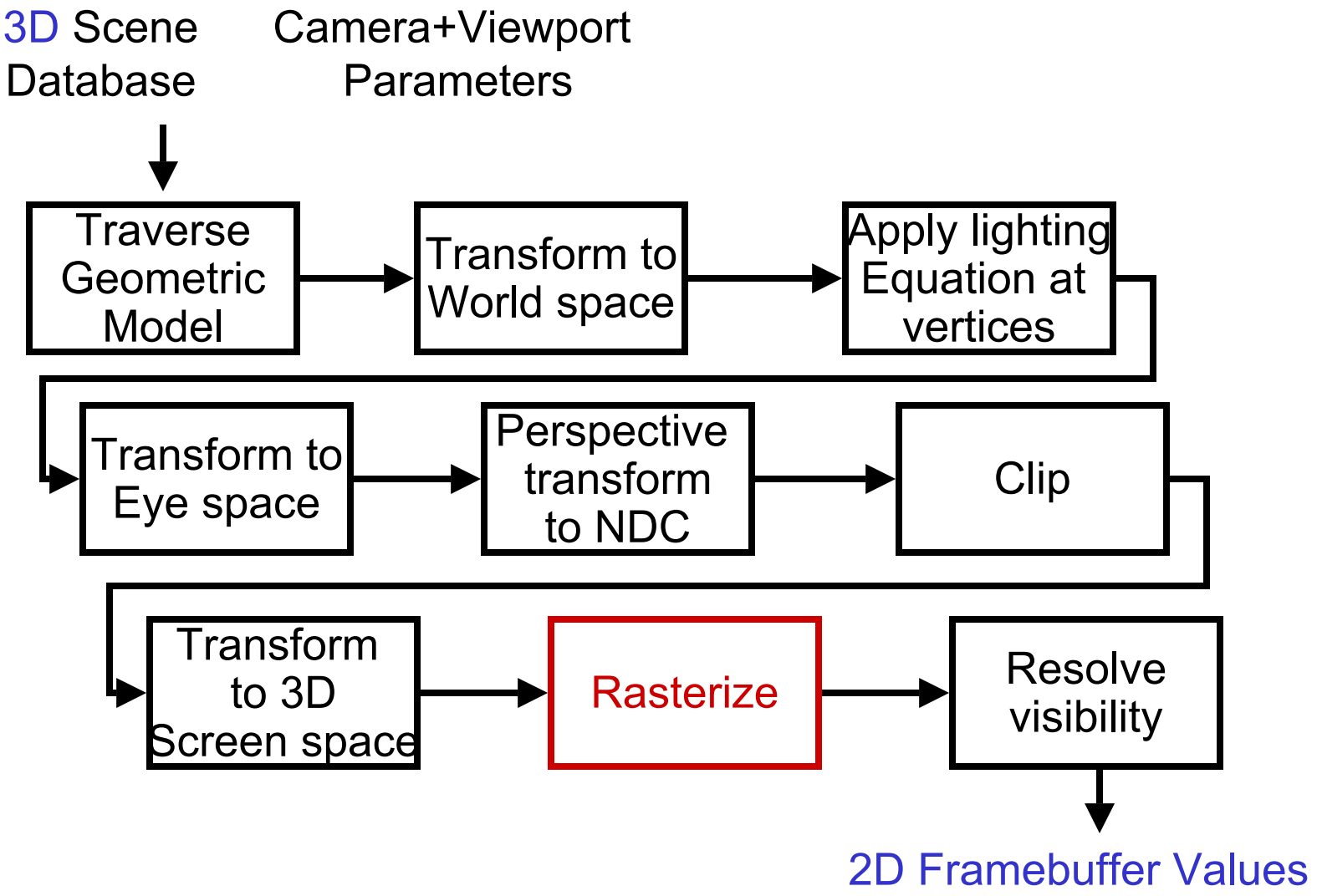
- What did we learn?
- Where was the “magic”? What were the insights?

Review of previous lecture?

- What did we learn?
- Where was the "magic"? What were the insights?
  - Graphics is a lot about interpolation/extrapolation
  - Convexity is important
  - Planes as homogeneous coordinates & dot product
  - Outcodes
  - Orienting edges of polygons

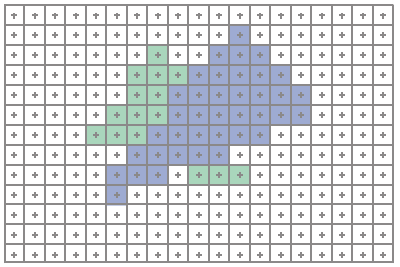
- Questions?

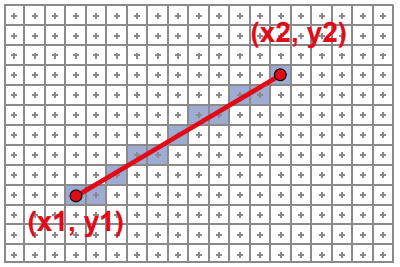
Overview of graphics pipeline?

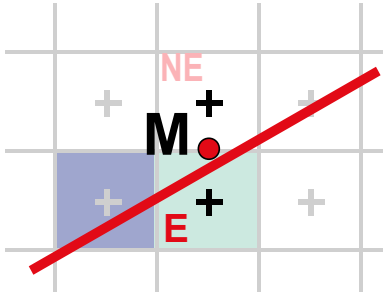


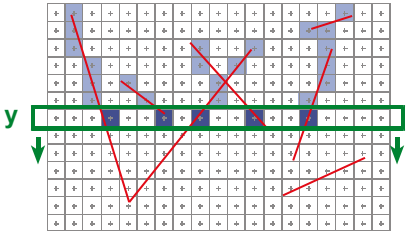
Today's lecture: line rasterization

- Overview of rasterization
- Naïve line rasterization
- Bresenham DDA
- Scanline and active edge list



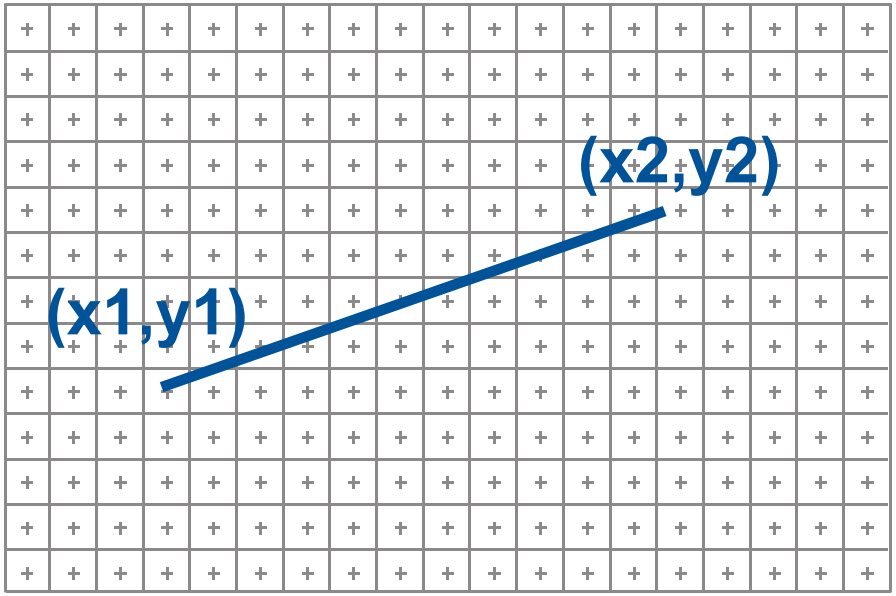






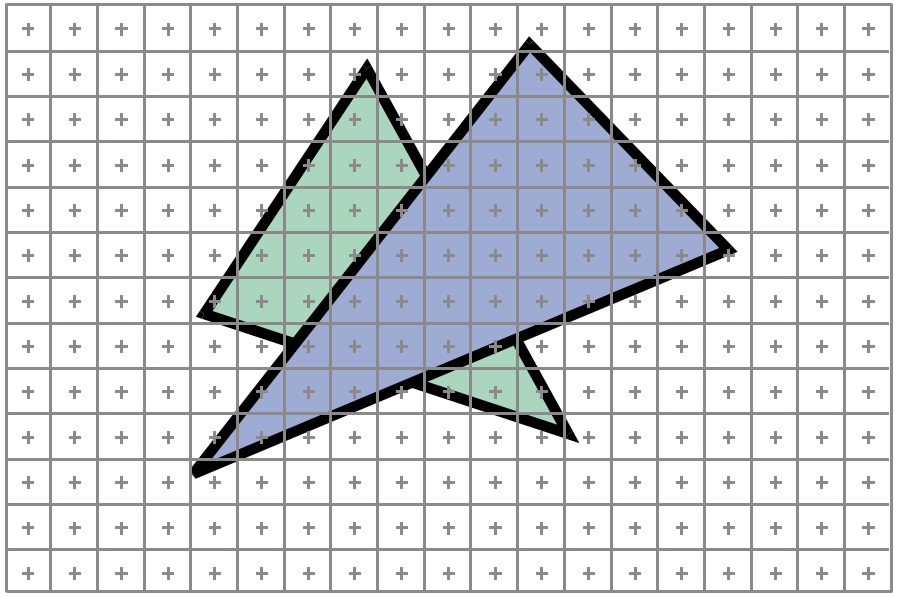
Framebuffer Model

- Raster Display: 2D array of picture elements (pixels)
- Pixels individually set/cleared (greyscale, color)
- Window coordinates: pixels centered at integers



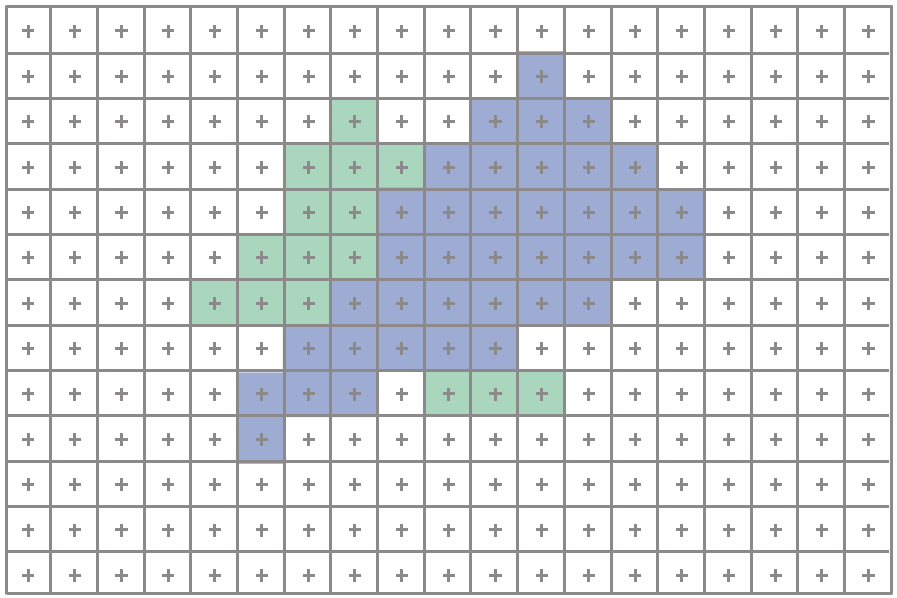
2D Scan Conversion

- Geometric primitive
  - 2D: point, line, polygon, circle...
  - 3D: point, line, polyhedron, sphere...
- Primitives are continuous; screen is discrete



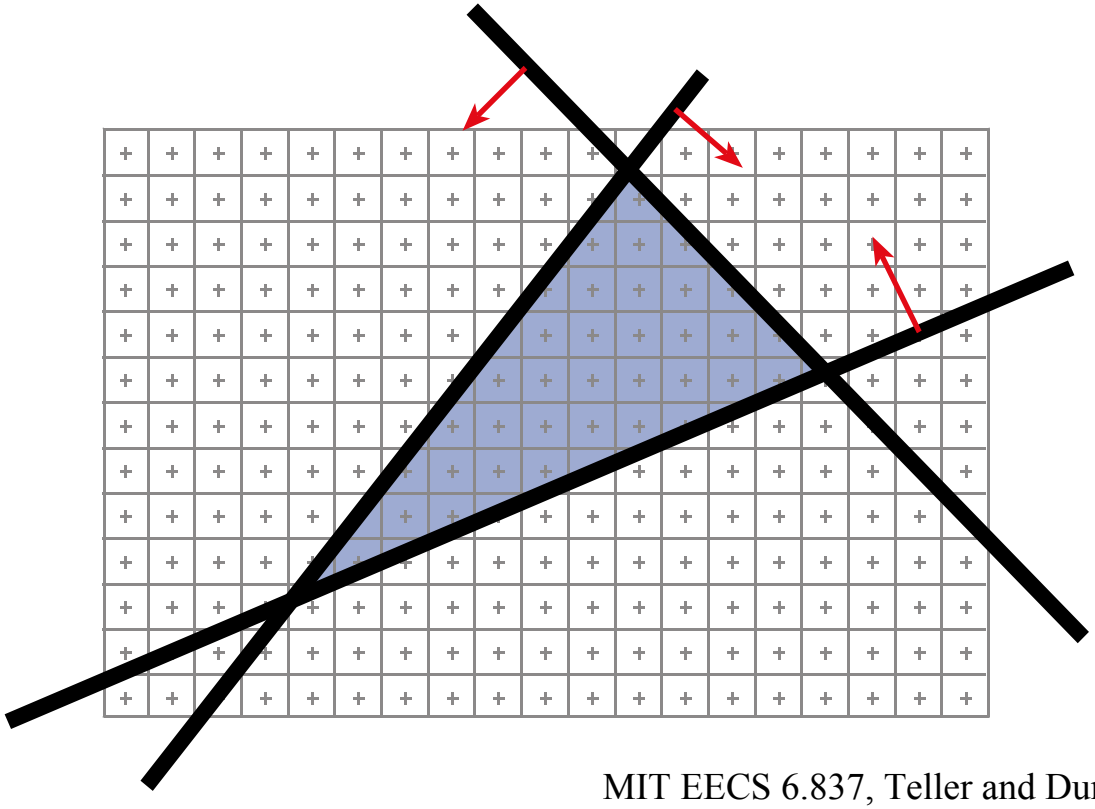
2D Scan Conversion

- Solution: compute discrete approximation
- Scan Conversion:
  algorithms for efficient generation of the samples comprising this approximation



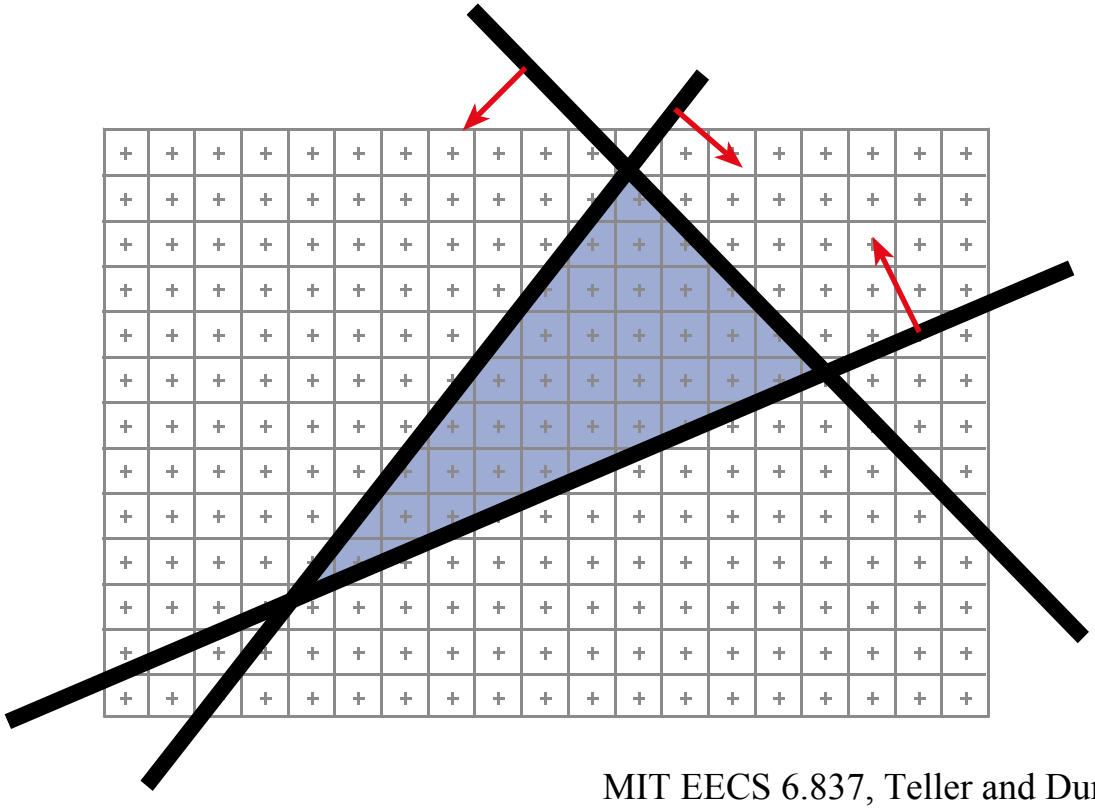
Brute force solution for triangles

- ?



Brute force solution for triangles

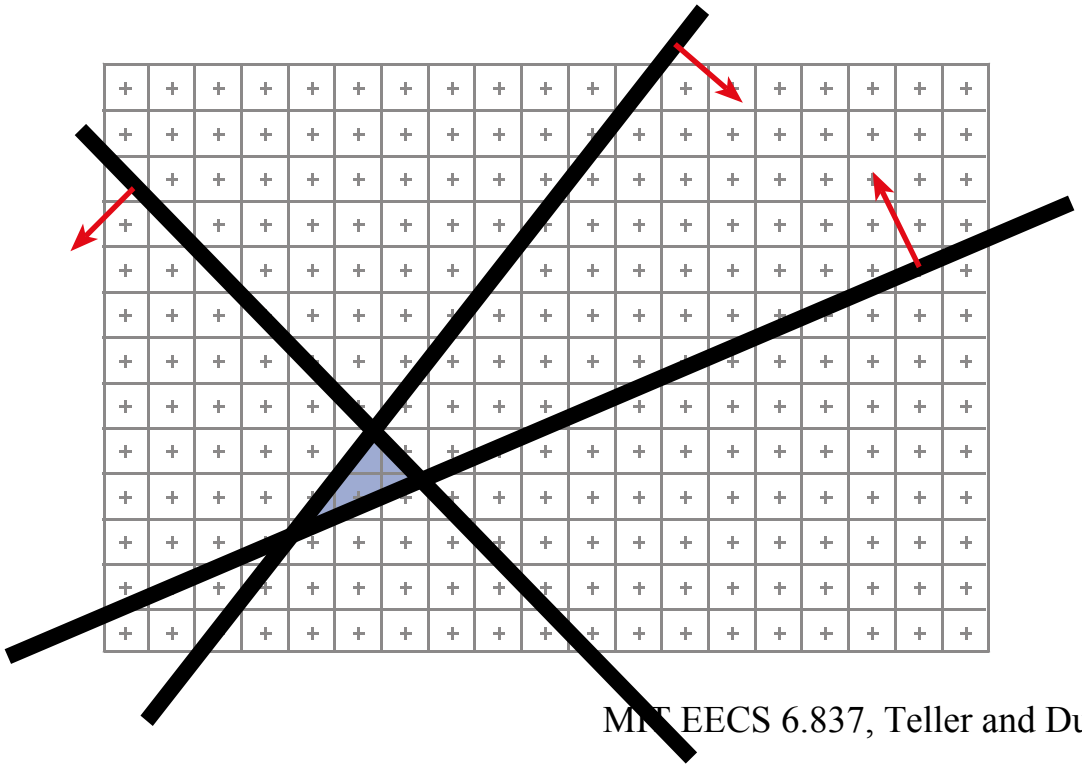
- For each pixel
  - Compute line equations at pixel center
  - "clip" against the triangle



Problem?

Brute force solution for triangles

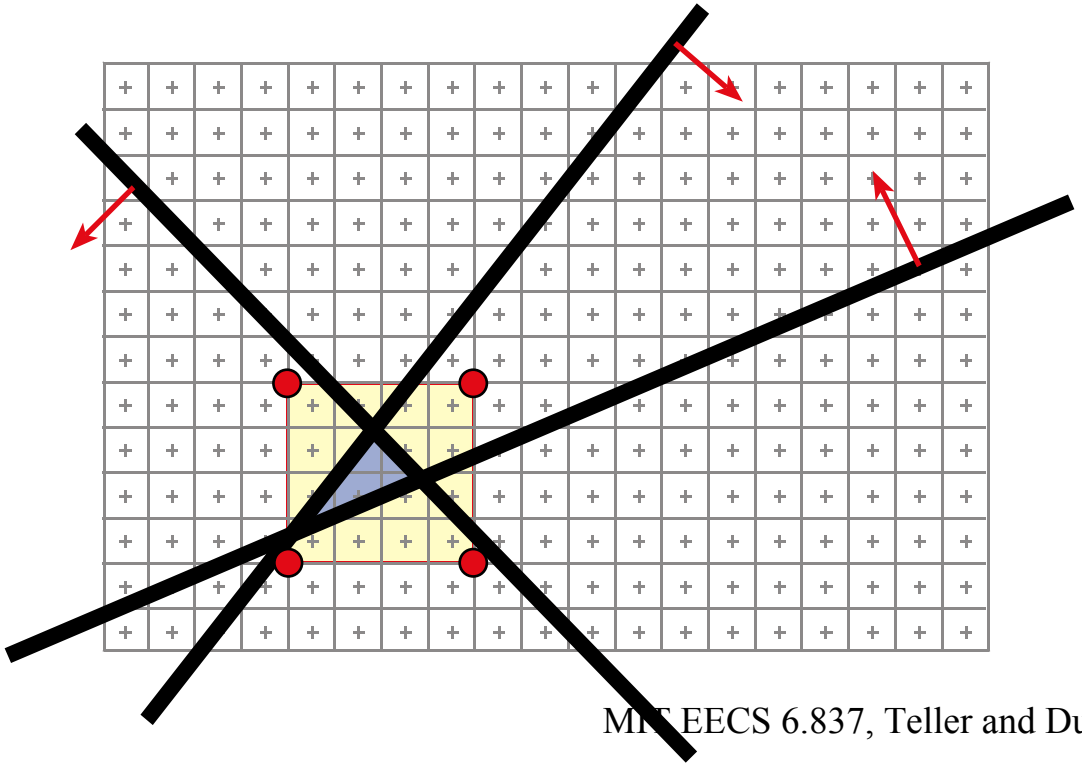
- For each pixel
  - Compute line equations at pixel center
  - "clip" against the triangle



Problem?
If the triangle is small, a lot of useless computation

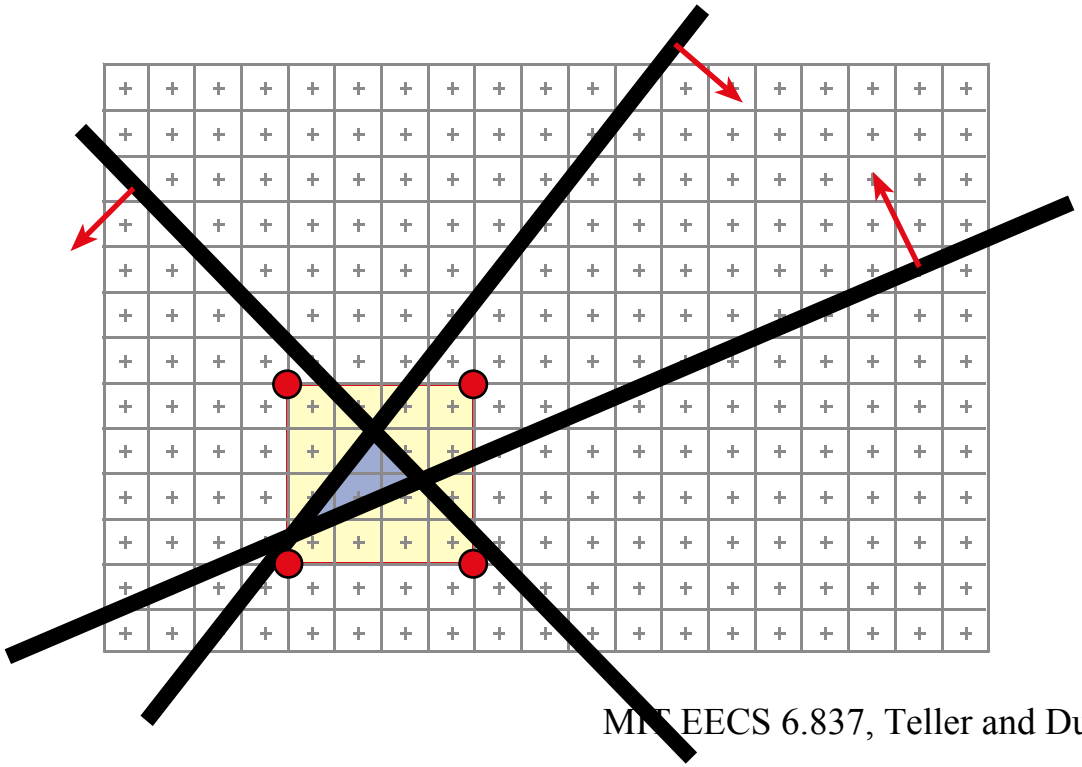
Brute force solution for triangles

- Improvement:
  - Compute only for the screen **bounding box** of the triangle
  - How do we get such a bounding box?



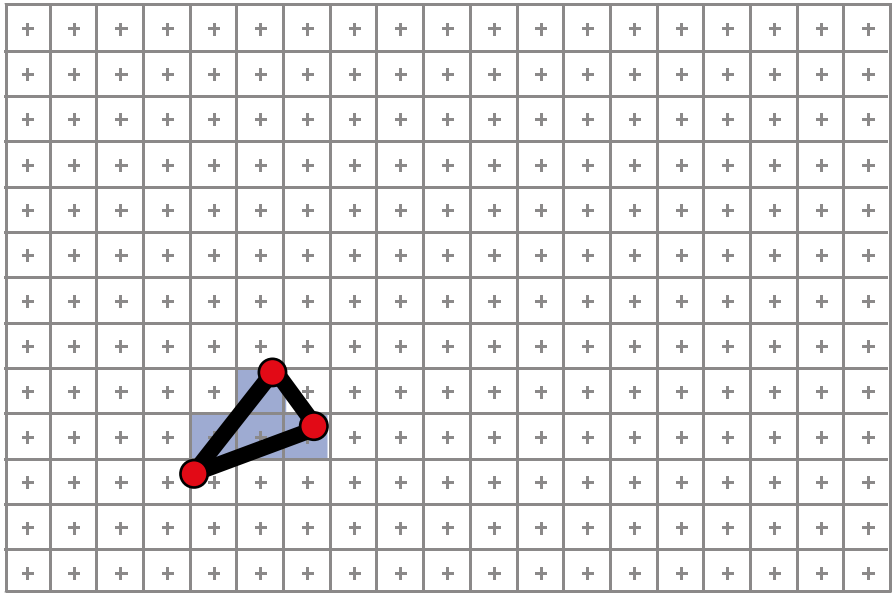
Brute force solution for triangles

- Improvement:
  - Compute only for the screen **bounding box** of the triangle
  - Xmin, Xmax, Ymin, Ymax of the triangle vertices



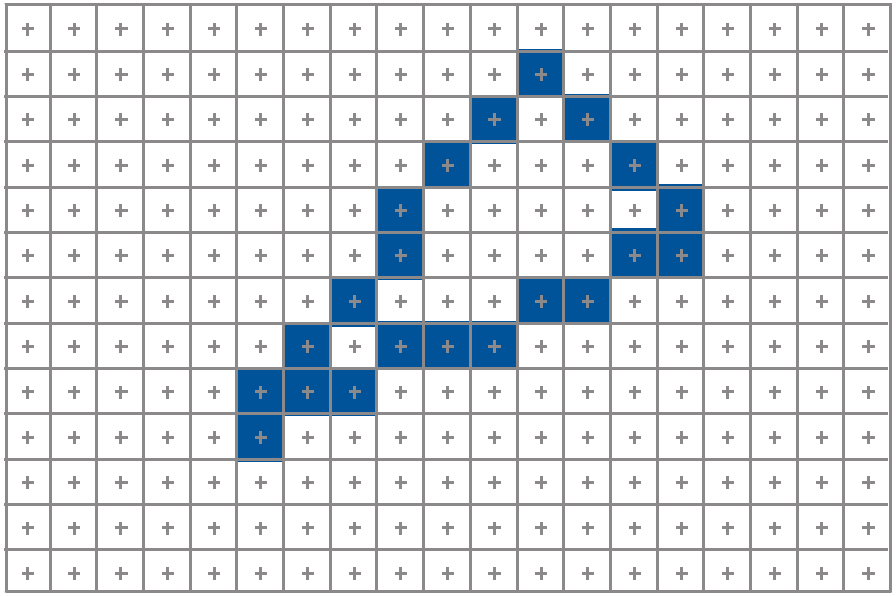
Can we do better? Yes!

- Plus, how do we rasterize lines using this approach?
- Efficient rasterization is the topic of the following two lectures



Rasterization

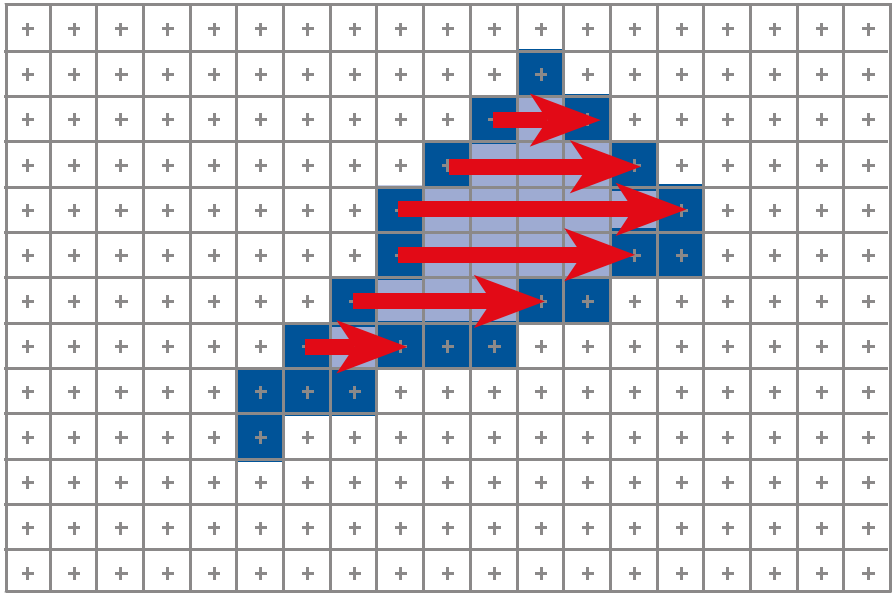
- Line rasterization
  - This lecture



Shirley page 55

Rasterization

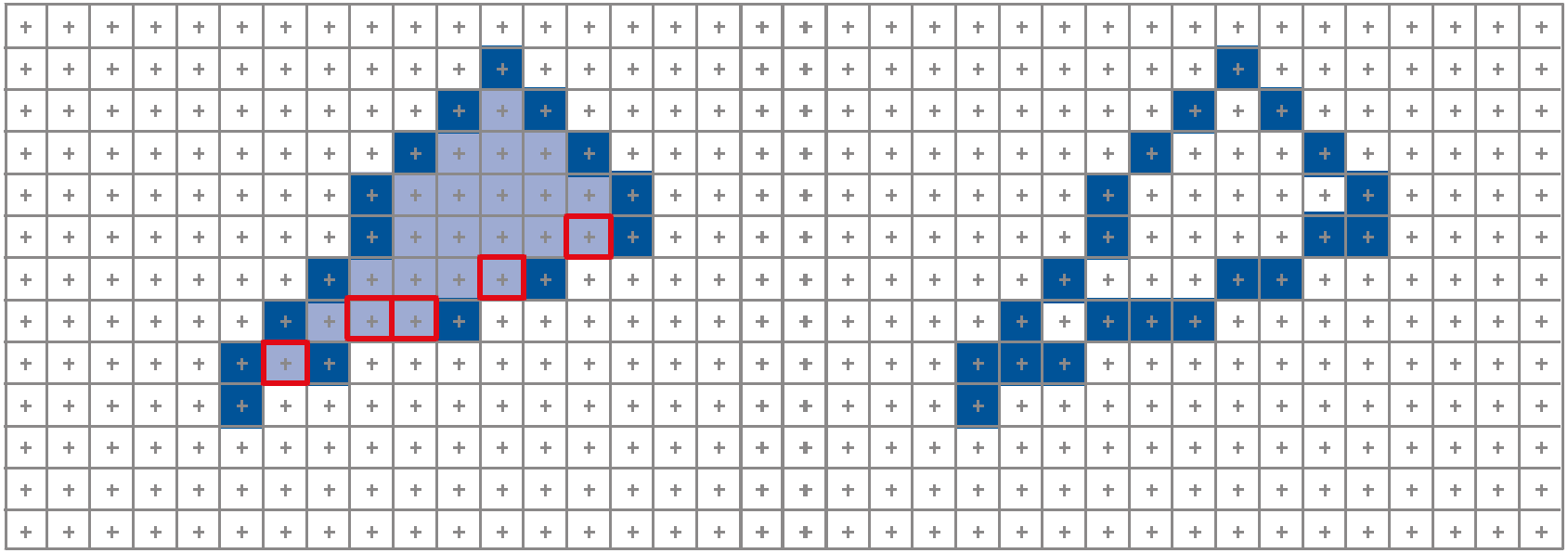
- Line rasterization
  - This lecture
- Polygon filling, interpolation, visibility
  - Tuesday



Shirley page 55

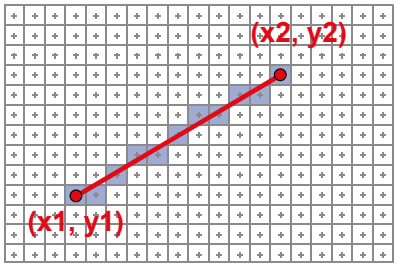
Rasterization

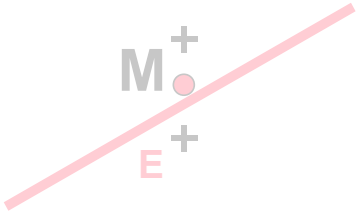
- There are subtle differences between rasterizing a segment, and rasterizing the boundary of a polygon
- Important for assignment 3 & 4

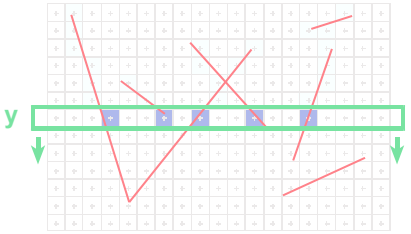


Today's lecture: line rasterization

- Overview of rasterization
- Naïve line rasterization
- Bresenham DDA
- Scanline and active edge list

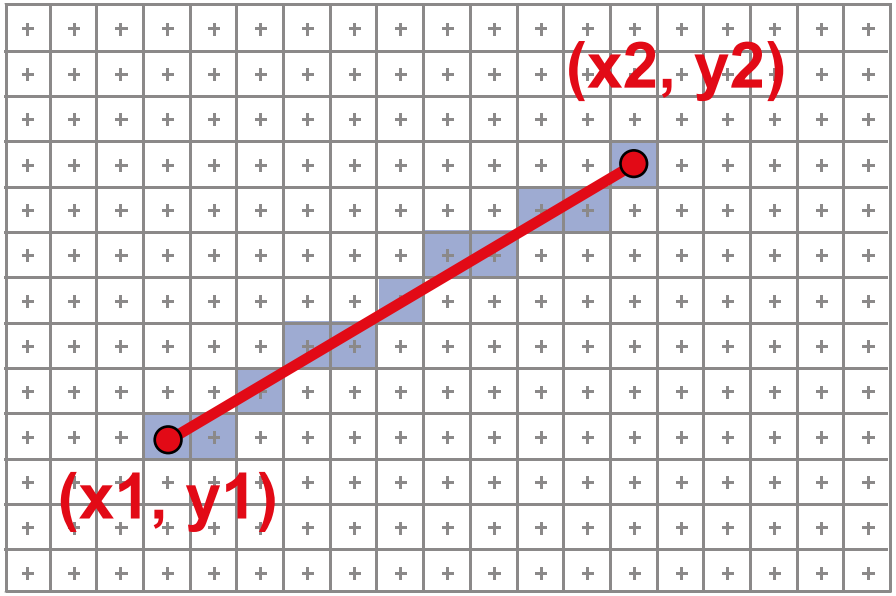






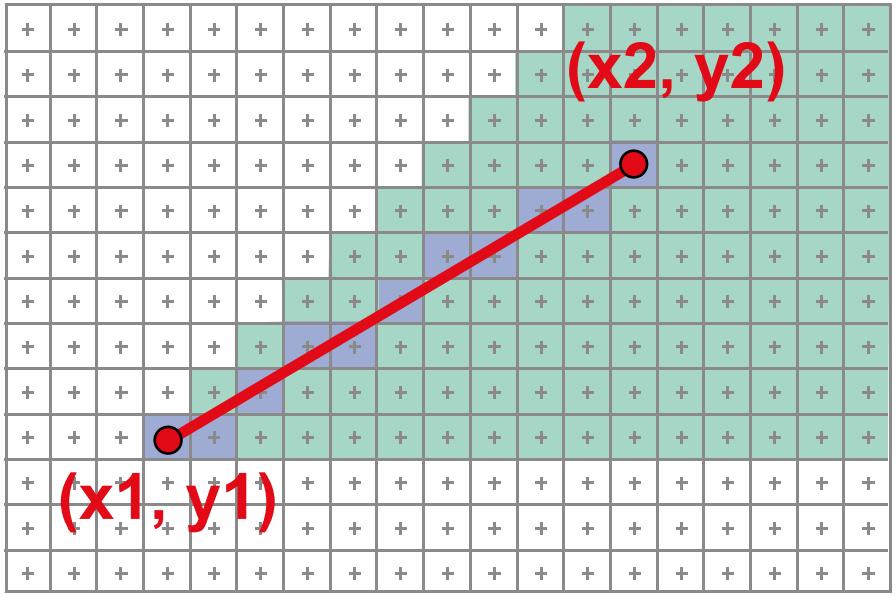
Scan Converting 2D Line Segments

- Given:
  - Segment endpoints (integers x1, y1; x2, y2)
- Identify:
  - Set of pixels x; y to “light up” for segment



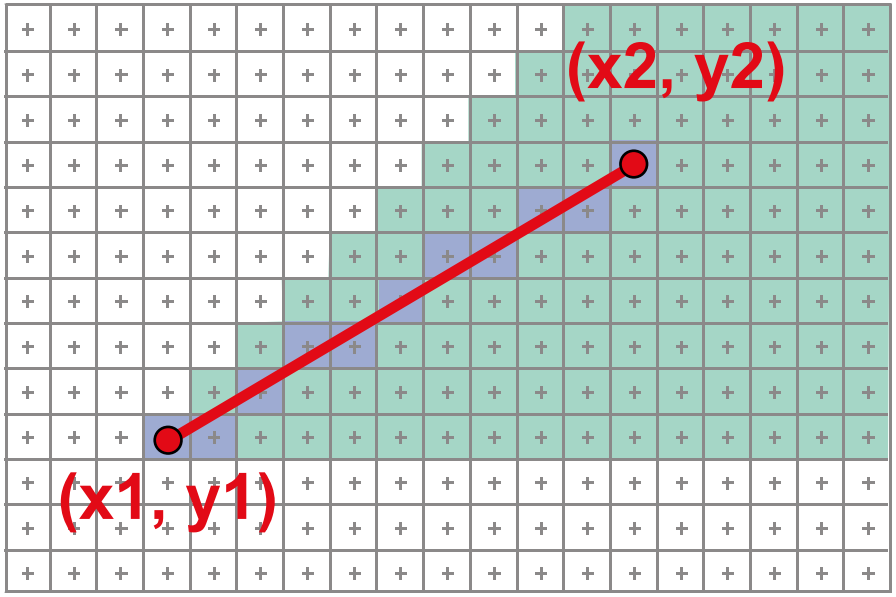
Algorithm Design Choices

- For m = dy/dx
- Assume: 0 < m < 1
- Why?



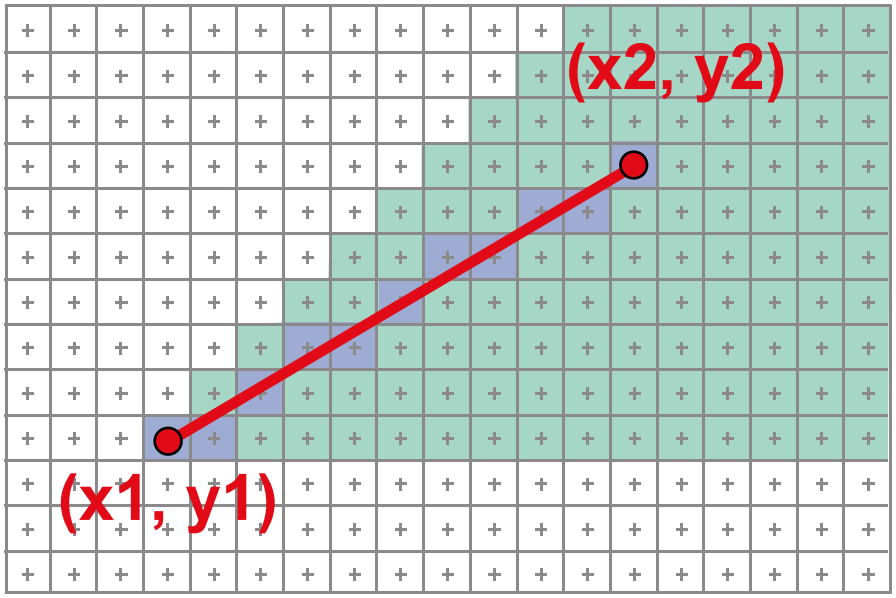
Algorithm Design Choices

- Exactly one pixel per column
  - Why?



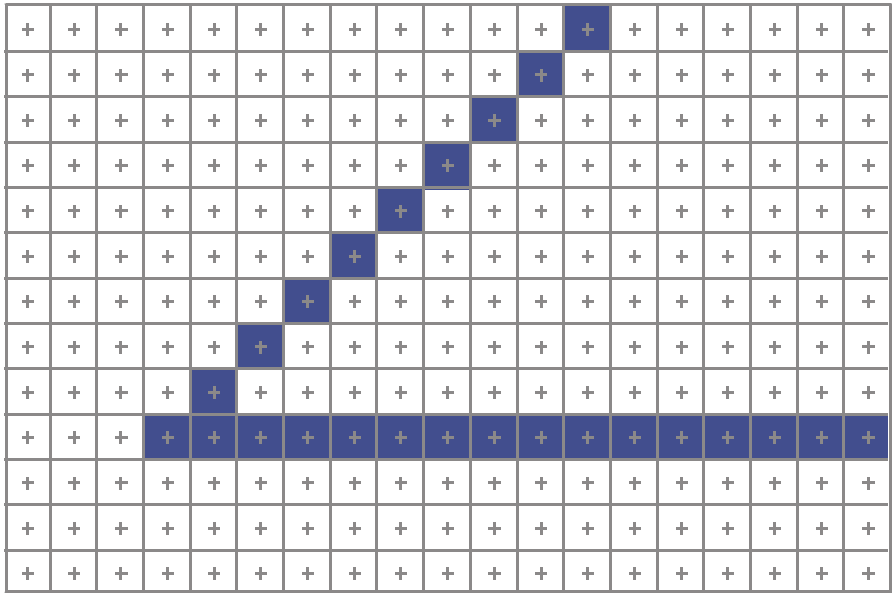
Algorithm Design Choices

- Exactly one pixel per column
  - fewer => disconnected
  - more =>too thick
  - “connectedness" with just 1 pixel per column



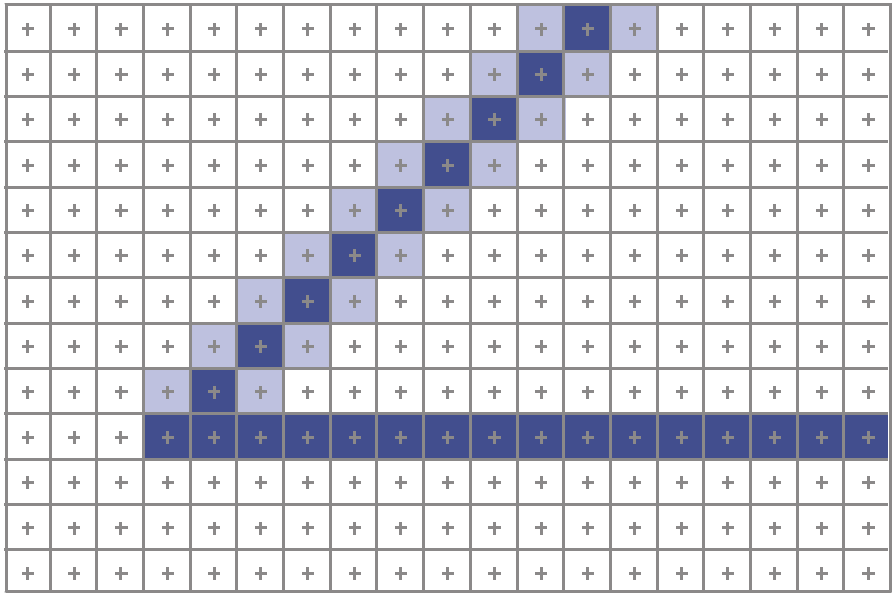
Algorithm Design Choices

- Note: brightness can vary with slope
  - What is the maximum variation?



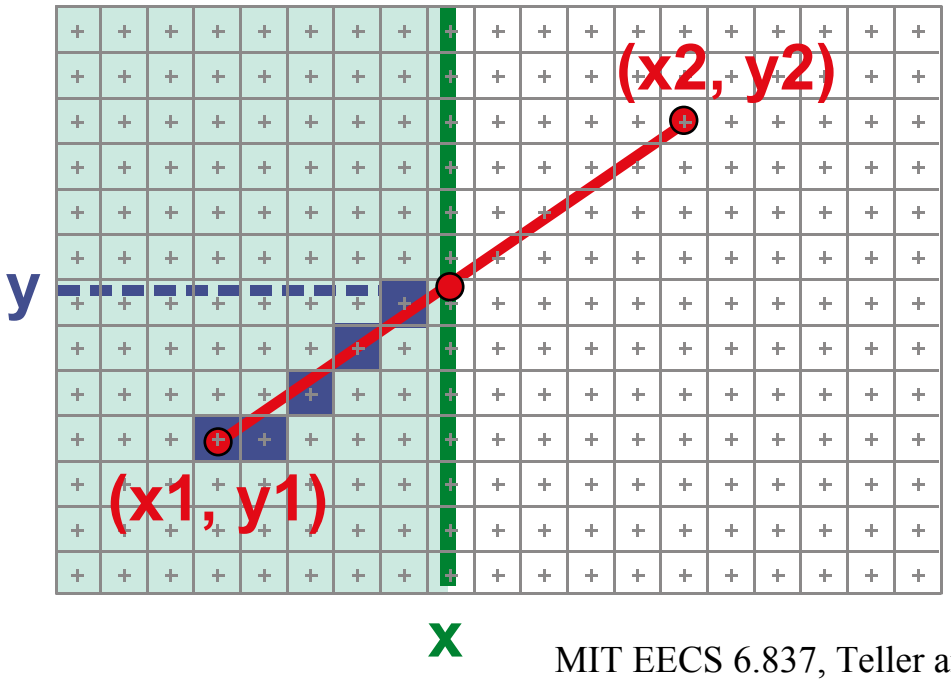
Algorithm Design Choices

- Note: brightness can vary with slope
  - What is the maximum variation? $\sqrt{2}$
- How could we compensate for this?
  - Answer: antialiasing (wait until November…)



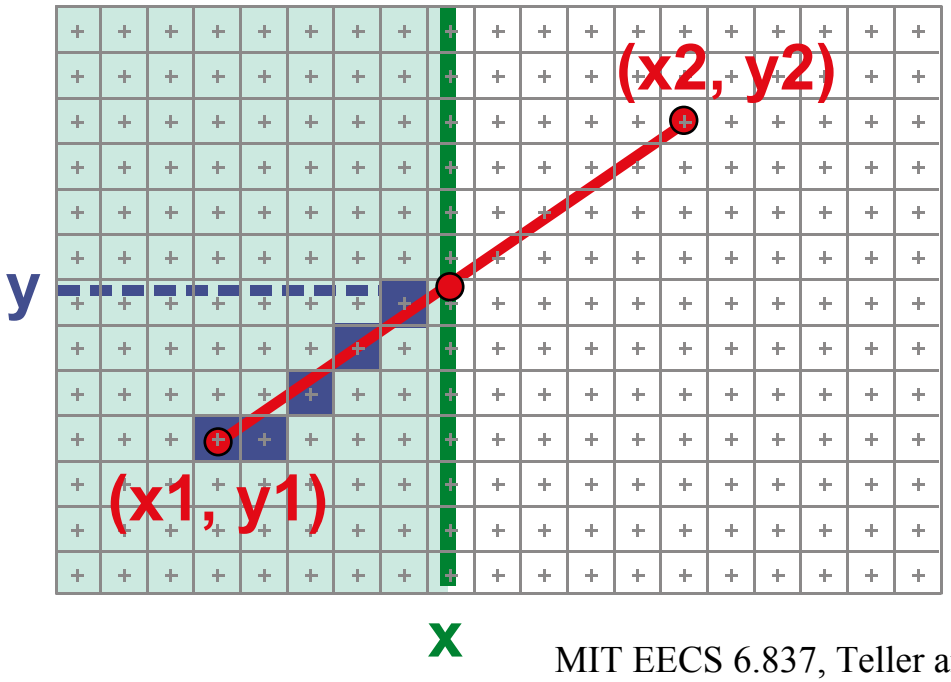
Naive algorithm, 1st attempt

- Simply compute y as a function of x
  - Conceptually: move vertical scan line from x1 to x2
  - What is the expression of y as function of x ?



Naive algorithm, 1st attempt

- Simply compute y as a function of x
  - Conceptually: move vertical scan line from x1 to x2
  - What is the expression of y as function of x ?



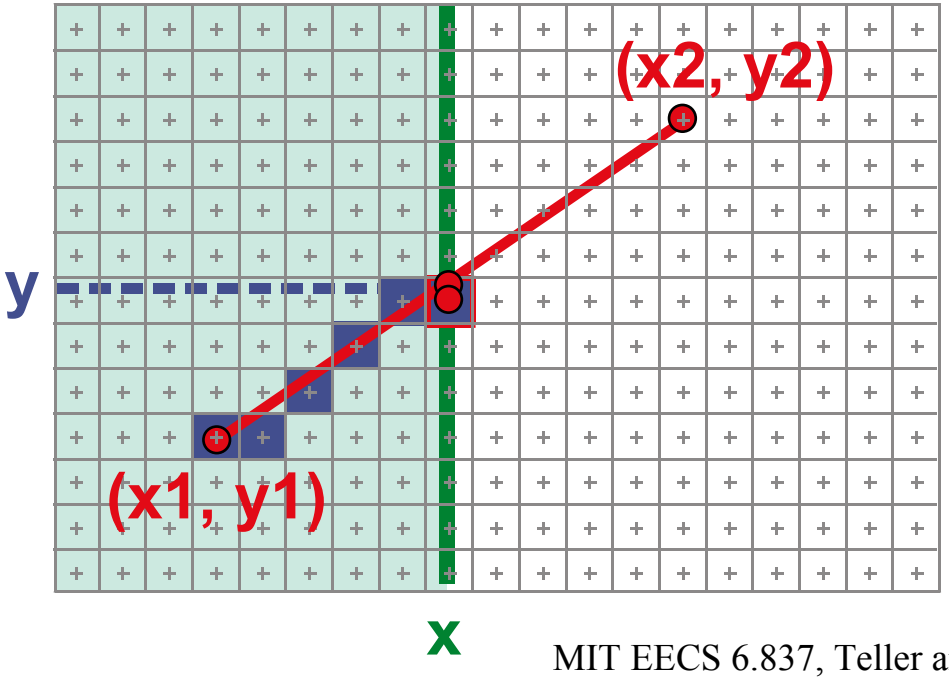
$$y = y1 + \frac{x - x1}{x2 - x1}(y2 - y1)$$

$$= y1 + m(x - x1)$$

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

Naive algorithm, 1st attempt

- Simply compute y as a function of x
- Round y (Why?)
- Set pixel (x, rnd(y(x)))



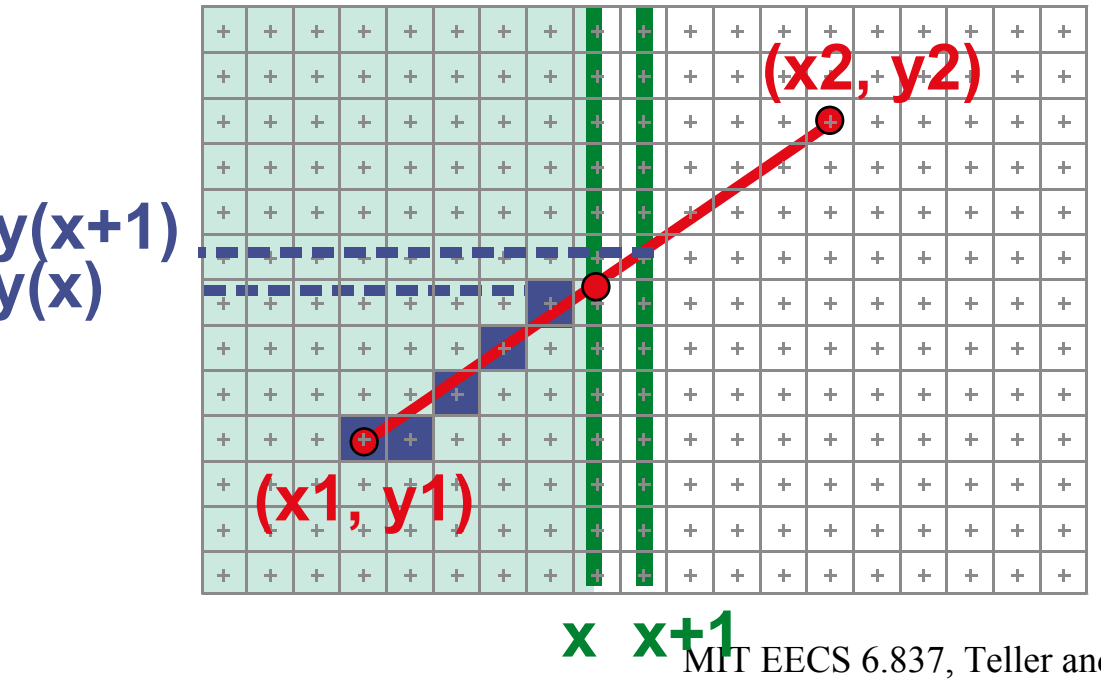
$$y = y1 + \frac{x - x1}{x2 - x1}(y2 - y1)$$

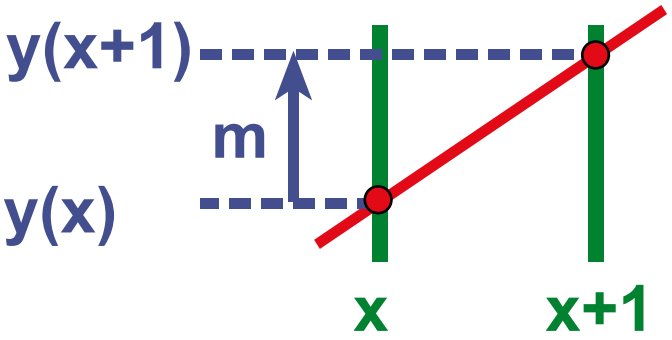
$$= y1 + m(x - x1)$$

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

Efficiency

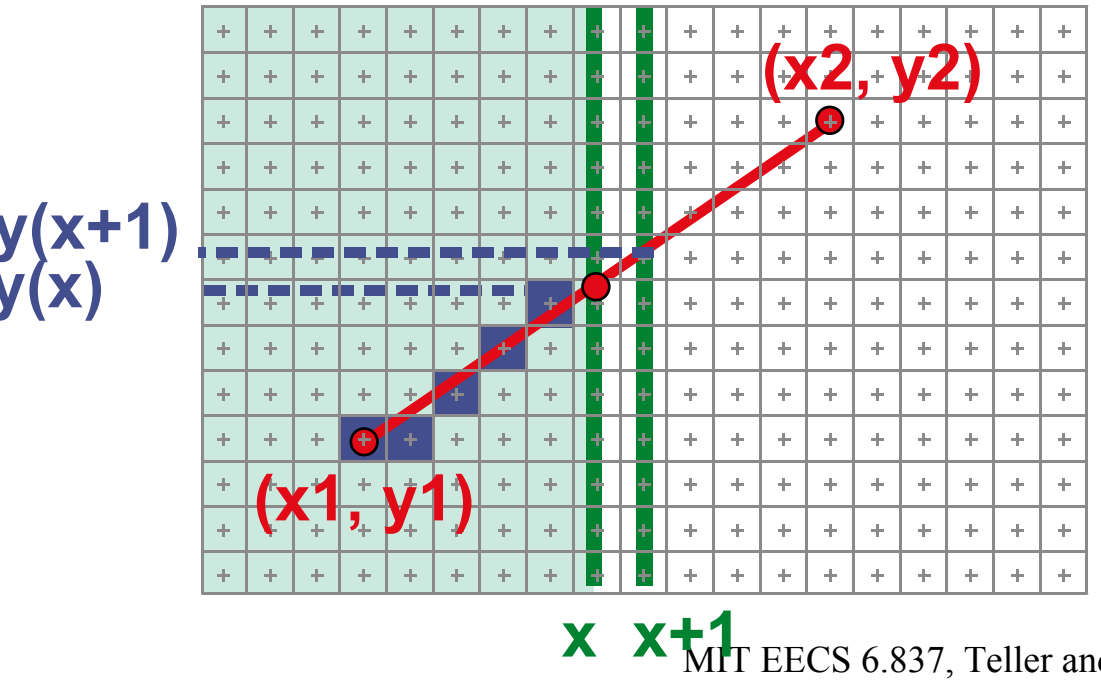
- Computing y values is expensive $y = y1 + m(x - x1)$
- Observe: y += m at each x step (m = dy/dx)
- First example of spatial coherence

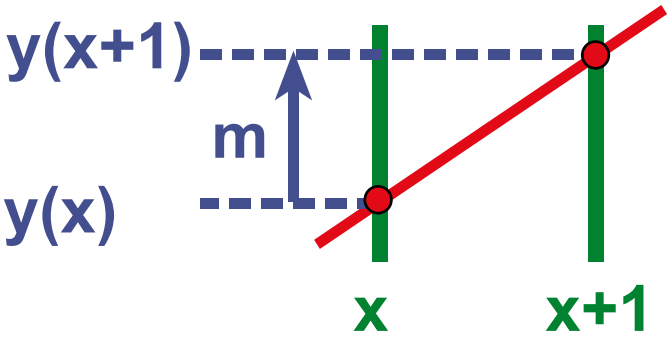




Incremental algorithm

- Start at (x1; y1)
- Thereafter, increment y value by slope *m*
- Note: *x* integer, but *y* floating point

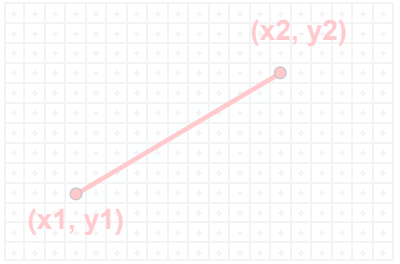


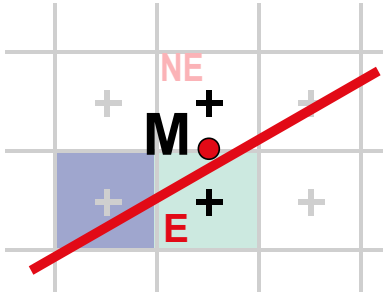


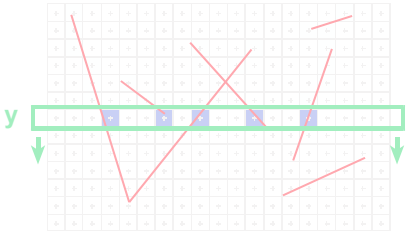
Questions?

Today's lecture: line rasterization

- Overview of rasterization
- Naïve line rasterization
- Bresenham DDA
- Scanline and active edge list

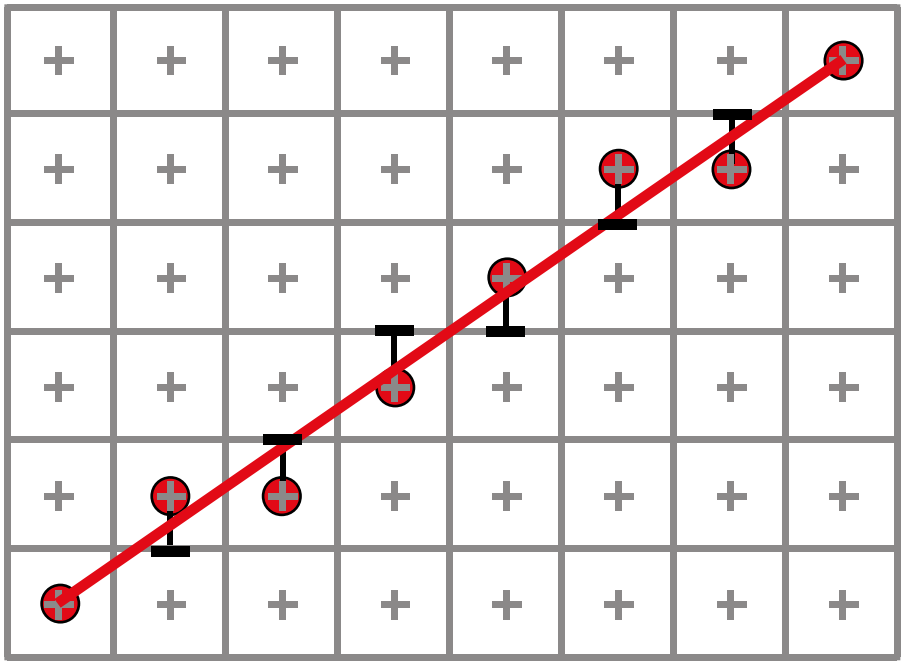






Bresenham's algorithm

- Select pixel vertically closest to segment
- Justification: intuitive, efficient
- Also: pixel center always within 0.5 vertically

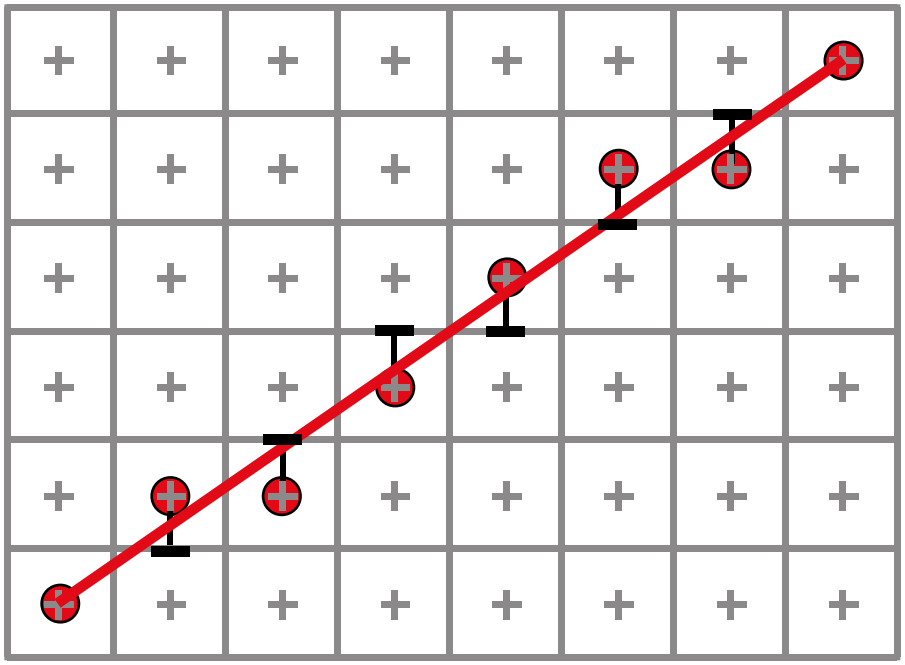


•Is selection criterion well-defined ?

•What other rules could we have used?

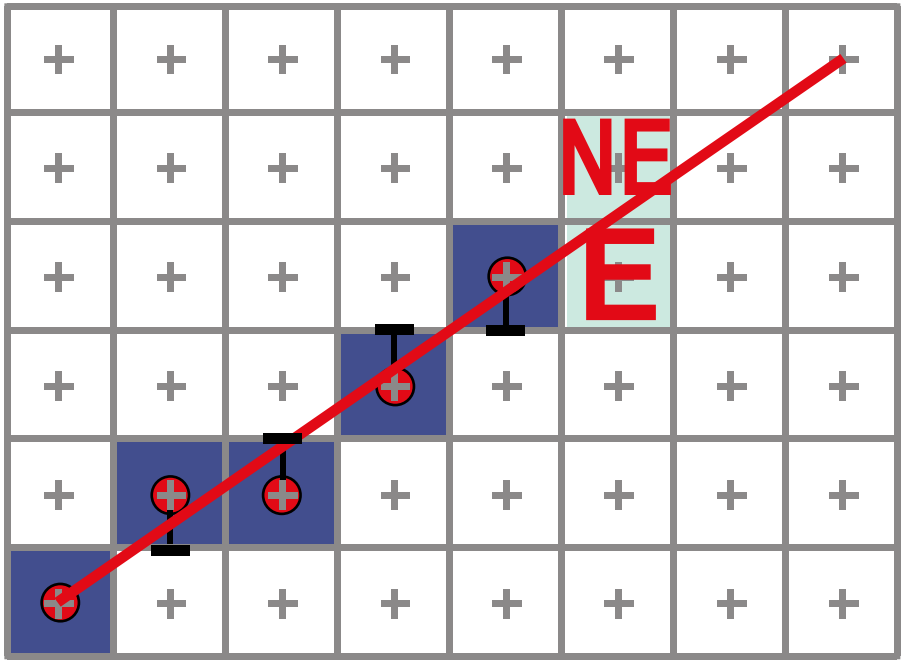
Bresenham DDA

- DDA = Digital Differential Analyzer
- Another scan algorithm
- Same output as naive & incremental algorithms



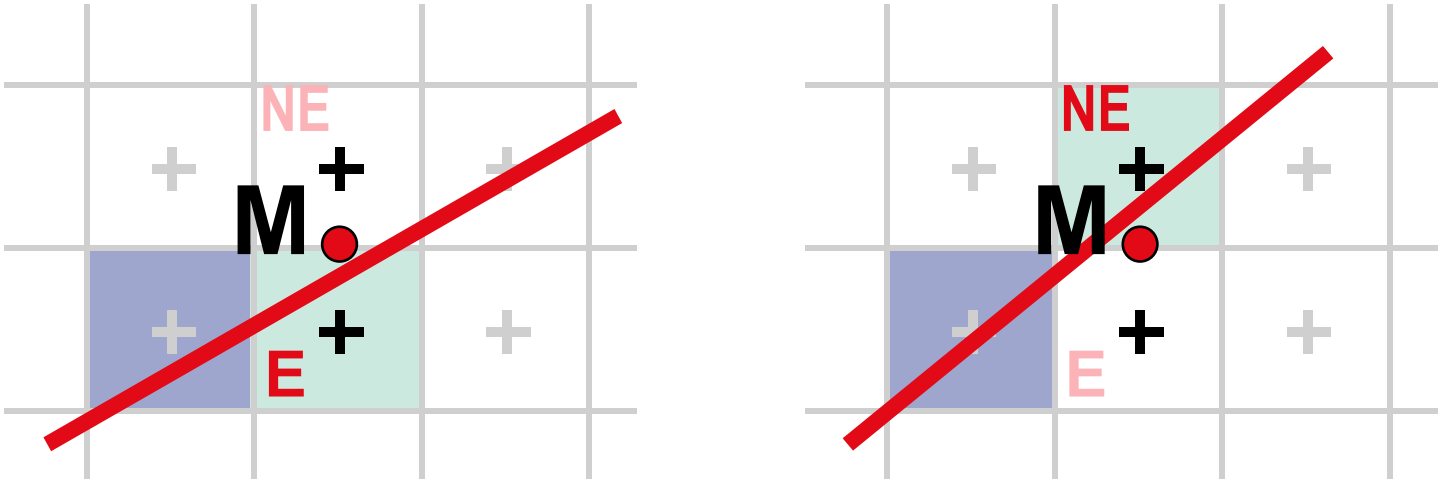
Bresenham DDA

- Observation: after pixel P at (xp , yp) next pixel must be either E or NE
  - Why?



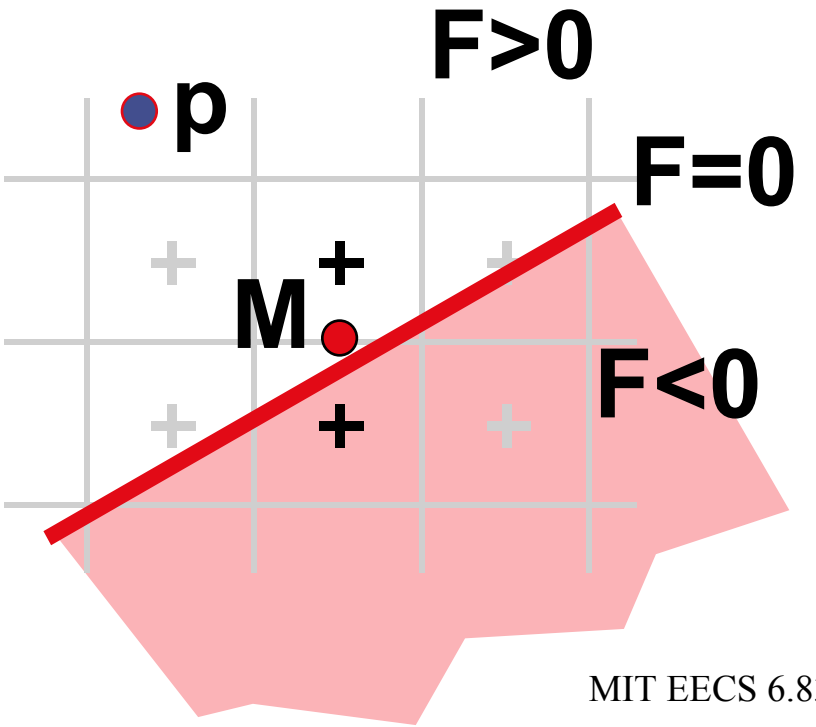
Bresenham Step

- Which pixel to choose: E or NE?
  - Choose E if segment passes below or through middle point M
  - Choose NE if segment passes above M



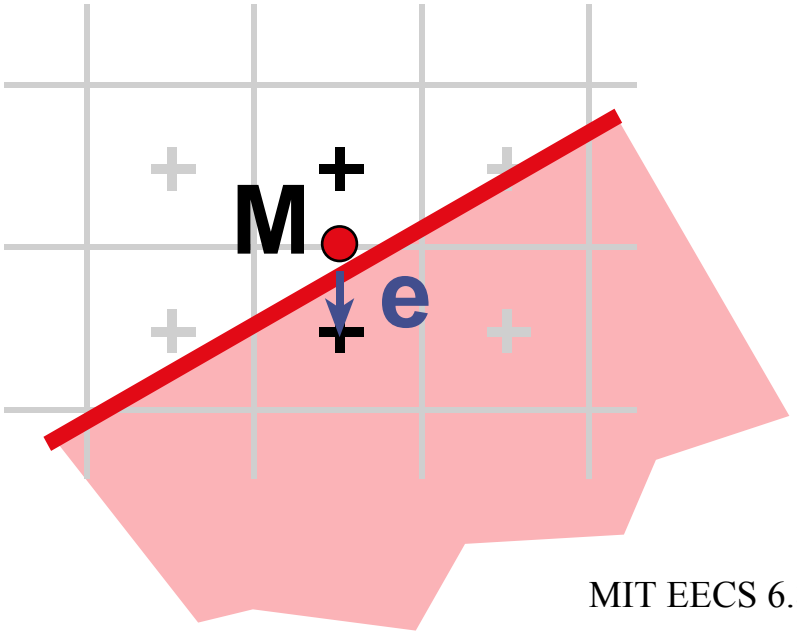
Bresenham Step

- Use implicit equation F for underlying line L:
  - F(x; y) = 0; where F(x; y) = y-mx-b (Why?)
  - F positive above L, zero on L, negative below L
- What is the meaning of the value of F(p) ?



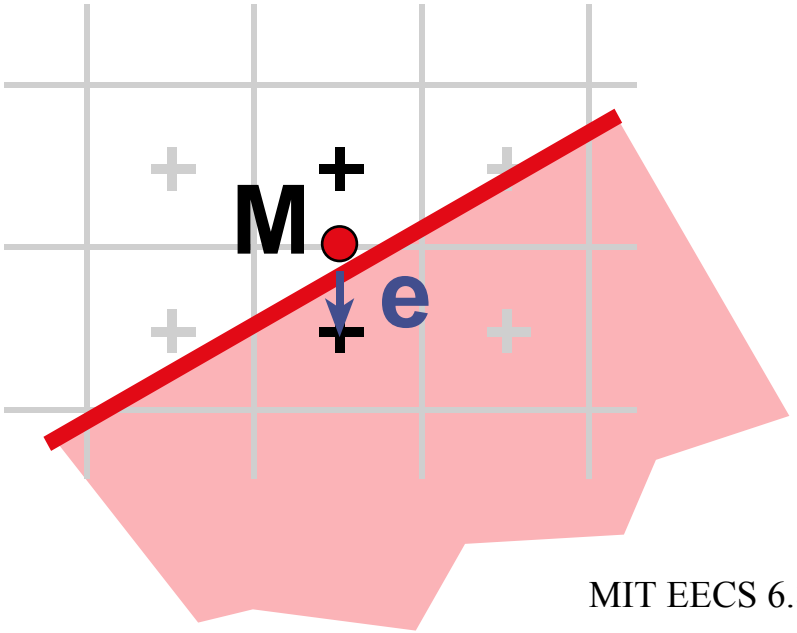
Bresenham Step

- Use implicit equation F for underlying line L
- Define an error term *e* as *e* = -F(x,y)
  - Choose NE if ?????



Bresenham Step

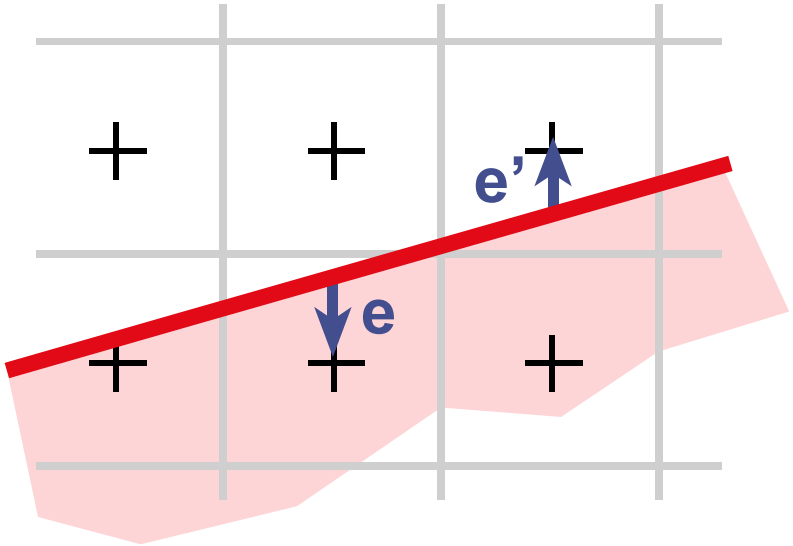
- Use implicit equation F for underlying line L
- Define an error term *e* as *e* = -F(x,y)
  - Choose NE if **e > 0.5** otherwise choose E



Using the Error Term

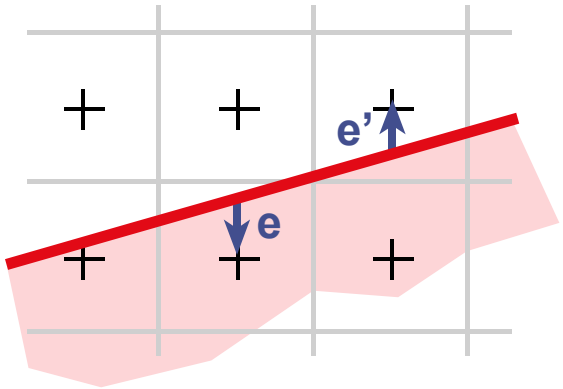
- Compute *e'* under (unconditional) *x* increment

  $e'=e+m$ (remember why?)
- Under what condition should we choose E?
- Under what condition should we choose NE?
  - In this case, how should *e'* be computed?



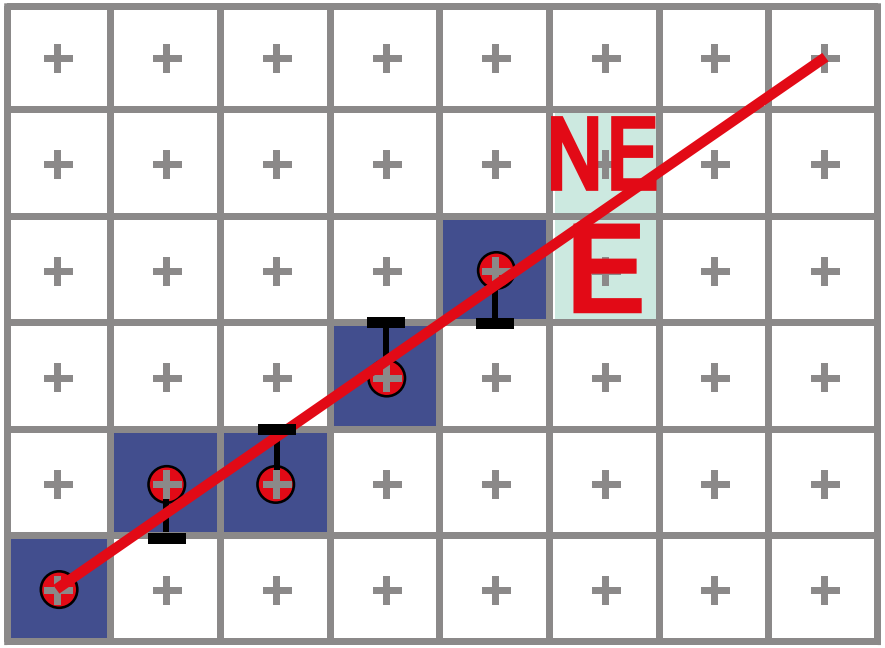
Using the Error Term

- Compute *e'* under *x* increment *e'=e+m*
- If e'<0.5
  - Choose E
- Otherwise
  - Choose NE?
  - Then decrement e' by 1



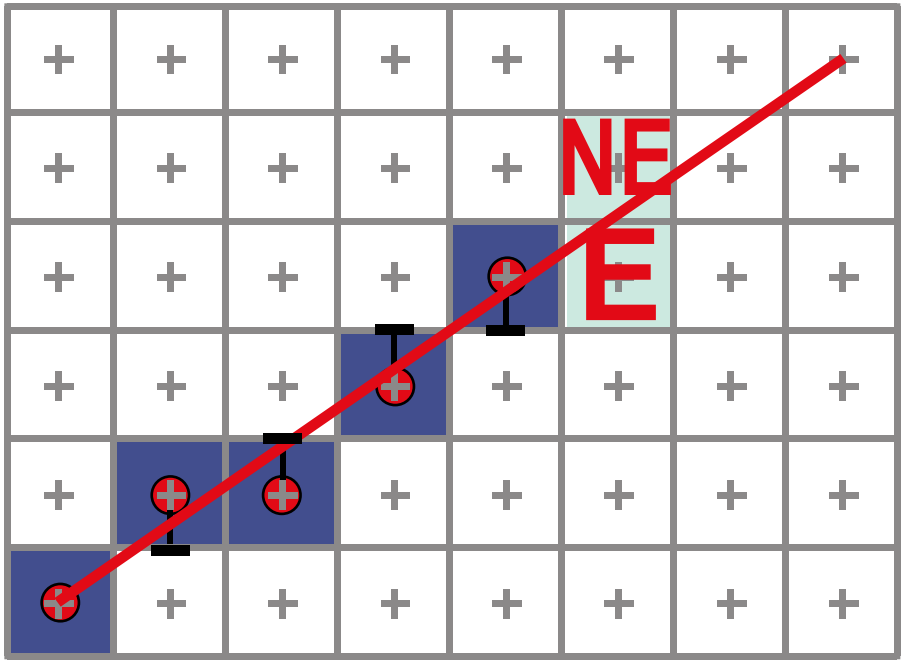
Summary of Bresenham

- Initialize x, y, e
- Loop over x1::x2
  - Plot
  - Update e
  - If e>0.5
    - Increment y
    - Decrement 1 from e



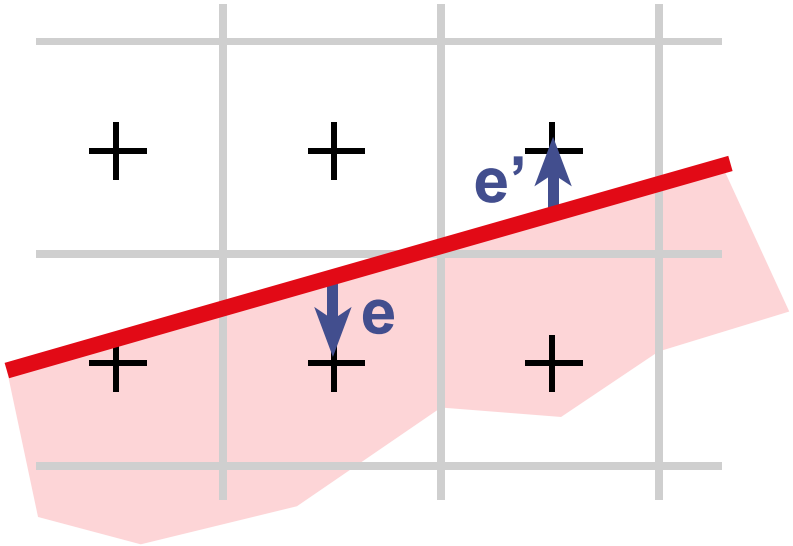
Bresenham Implementation

- We've sketched case in which x1 < x2, m <= 1
- Handle all eight octants (using symmetry)
- Endpoints not necessarily integer coordinates



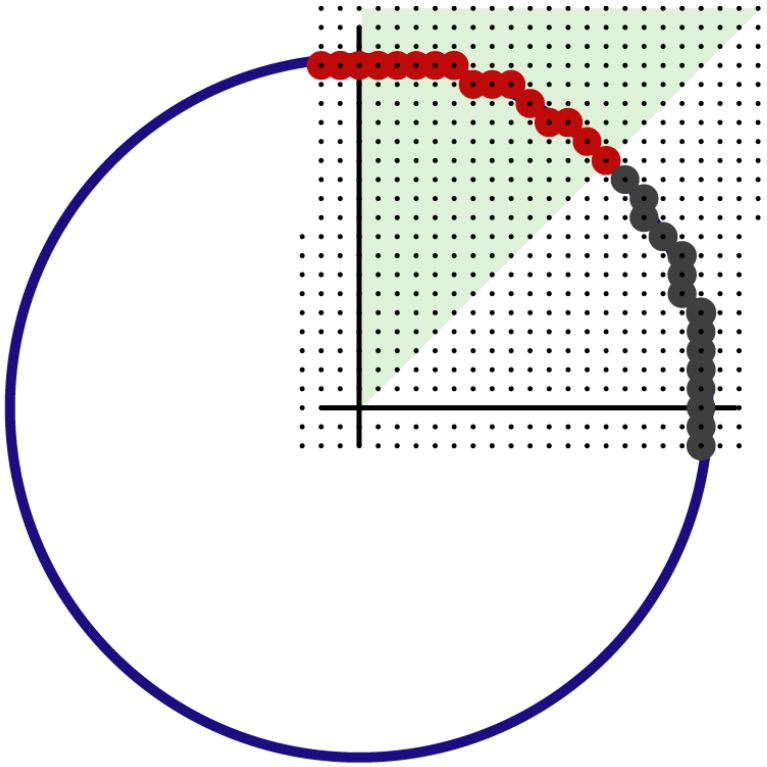
Using only integer arithmetic

- Multiply everything by dx=(x2-x1)
- Initialize *dx e=0*
  - *dx e' = dx e + m dx*
  - *dx e' = dx e +dy*
  - Choose NE if *dx e > [(dx+1) / 2]*



Circle Scan Conversion

- Need generate pixels for 2nd octant only
- Slope progresses from 0 to –1
- Analog of Bresenham Segment Algorithm

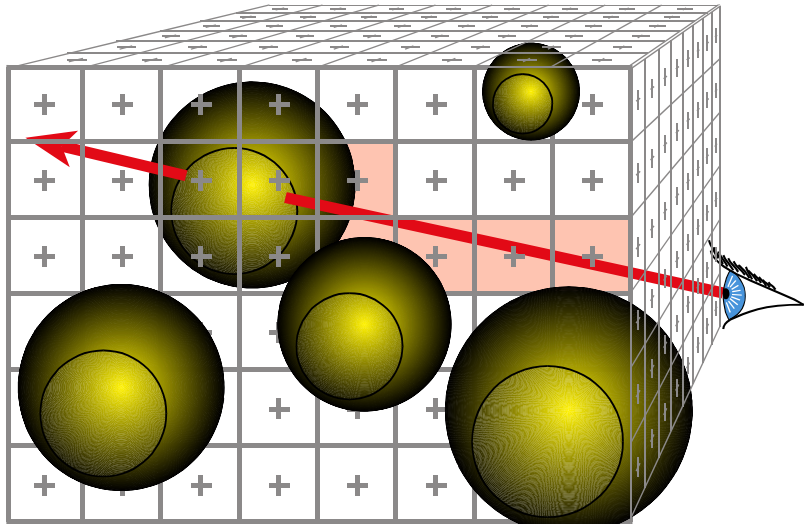


Questions?

"Anytime, Slim."

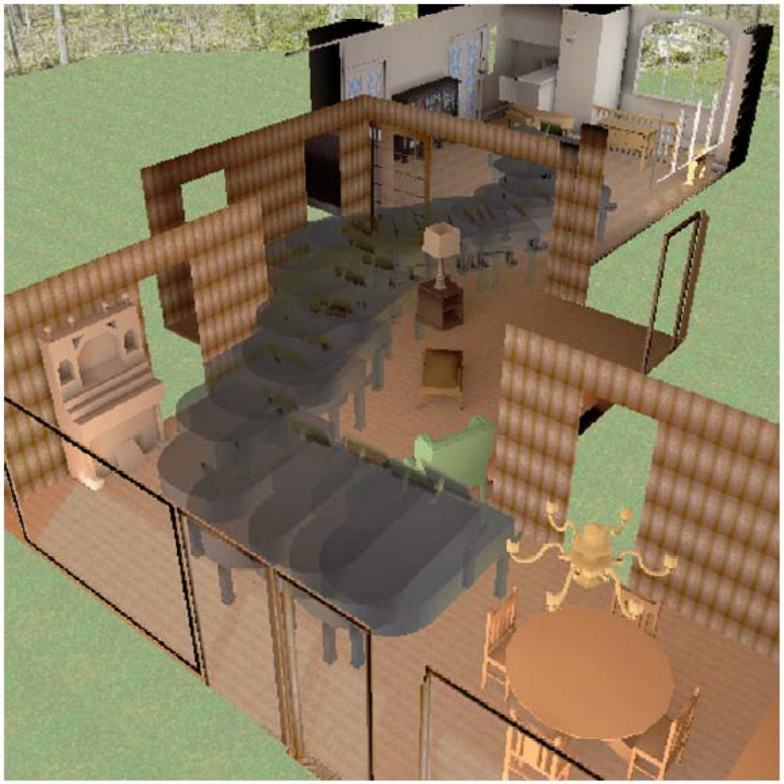
Discussion: Why learn Bresenham?

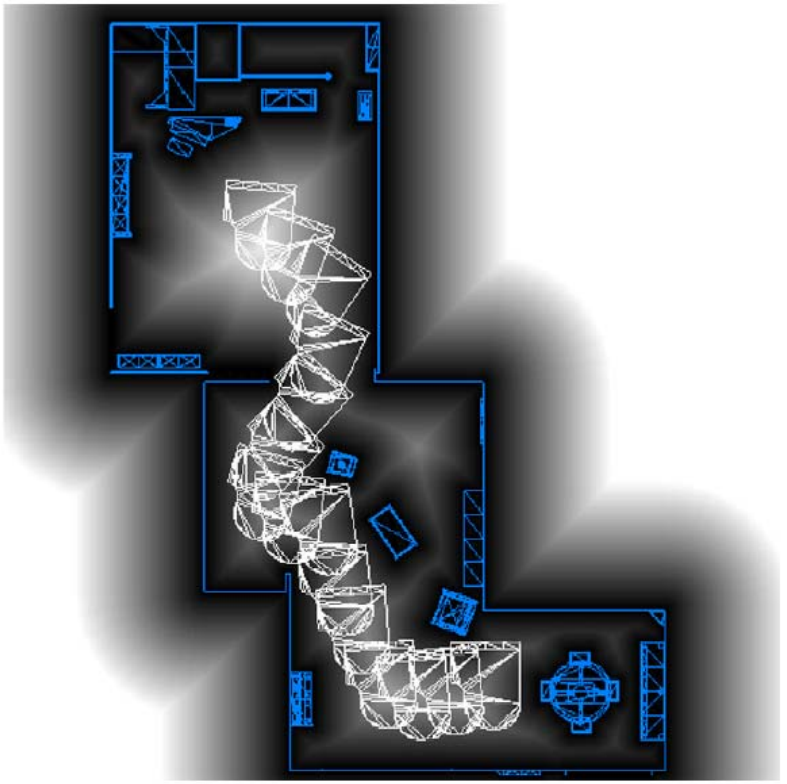
- You might think: “All modern API do it for me, why do I need to learn Bresenham?”
- Well, somebody has to code it!
- Useful extensions
  - E.g. ray marching for ray tracing
  - Remember the key insights
  - Bresenham is a “paradigm”



Discussion: Why learn Bresenham?

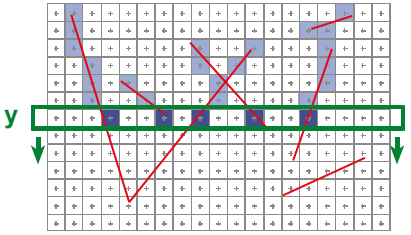
- Understand what the hardware does
  - E.g. use the hardware for other calculations
    - E.g. Piano mover problem [Hoff et al. SIGGRAPH 99]





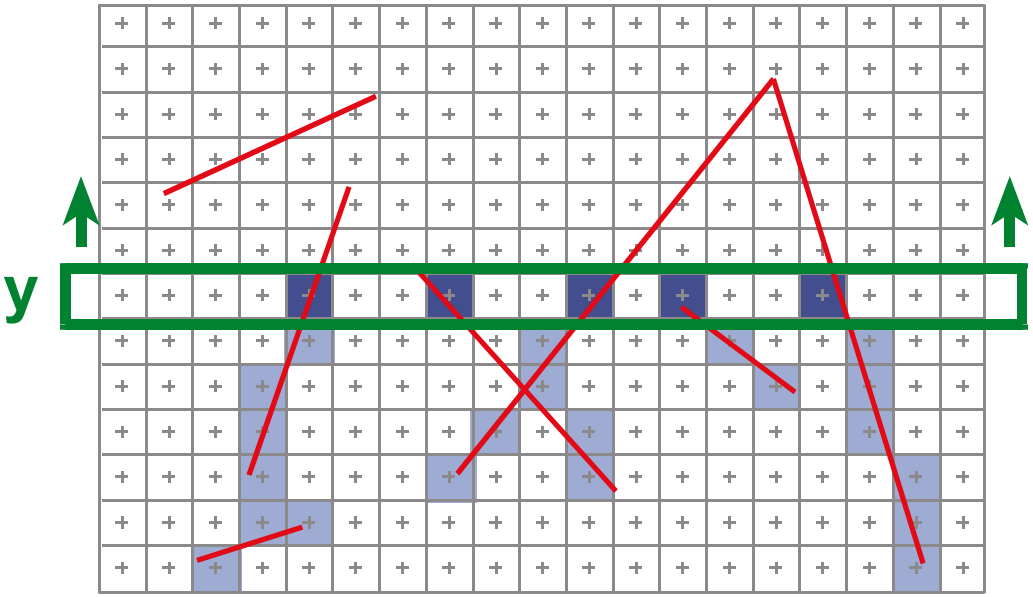
Today's lecture: line rasterization

- Overview of rasterization
- Naïve line rasterization
- Bresenham DDA
- Scanline and active edge list



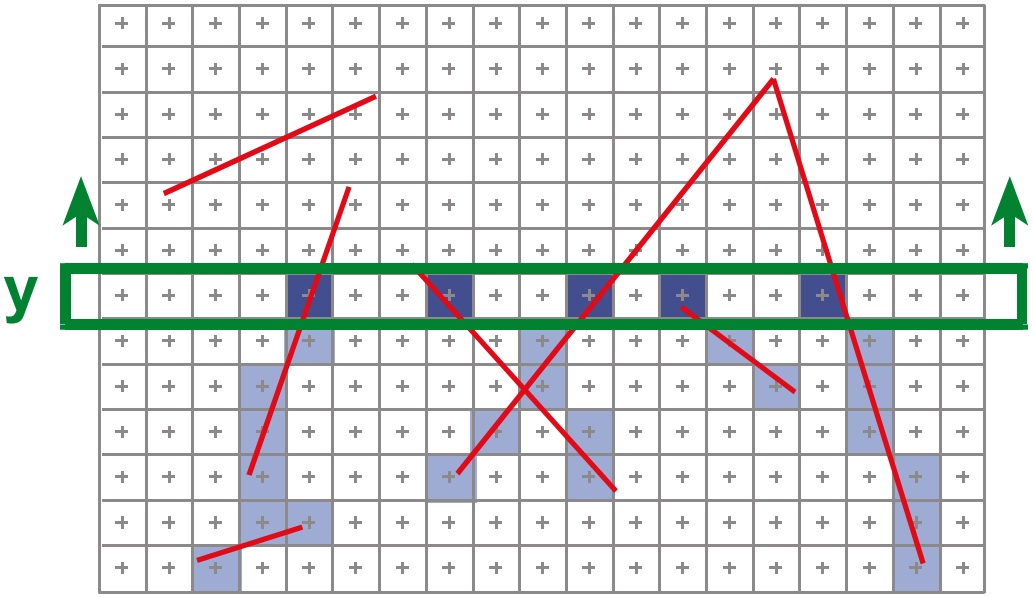
Scan Line rasterization

- So far, we have drawn one segment at a time
- Now, we are going to handle all the segments together pixel row by pixel row



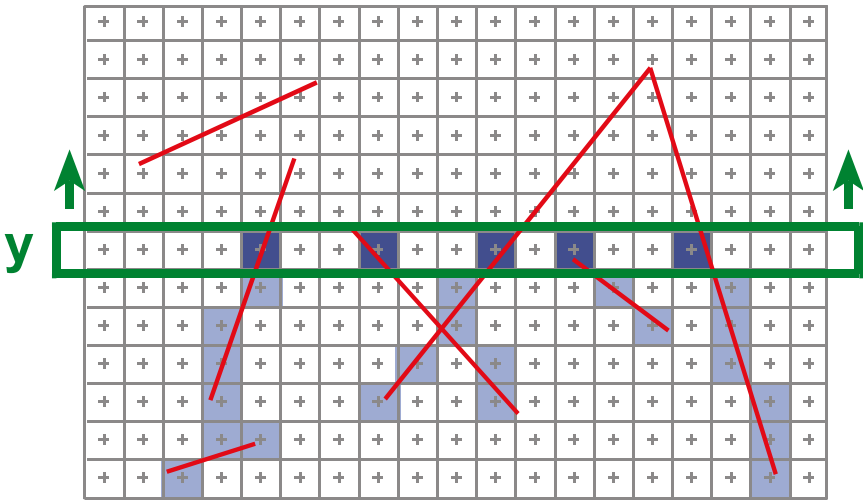
Why Scanline?

- Not enough memory
  - First framebuer (512) cost $80,000 (late '60s)!
- Access to one row at a time (some printers)
- Will simplify our treatment of polygons



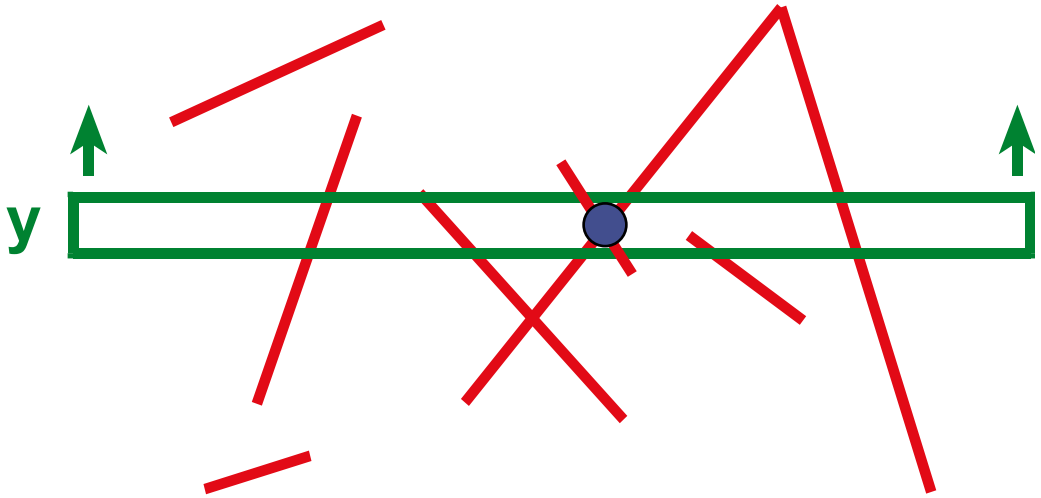
For assignment 3

- Access through class Raster
  - Constructor (int width, *environment)
  - Init () *provides a clear new scanline*
  - setPixel(x, color) *modifies current scaline*
  - write(y) *write entire scanline*



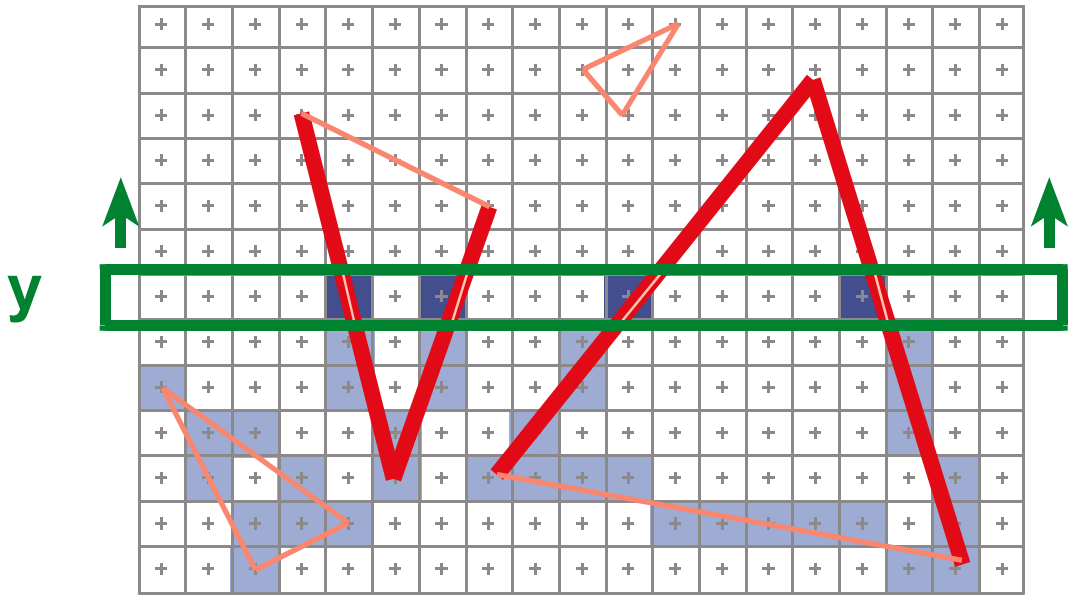
Why learn scanline?

- Basic principle of **sweep** algorithm.
  - Spatial coherence:
    Computation at previous scanline is reused
  - E.g. efficient segment intersection
  - E.g. sweep construction of 4.5D visibility structure



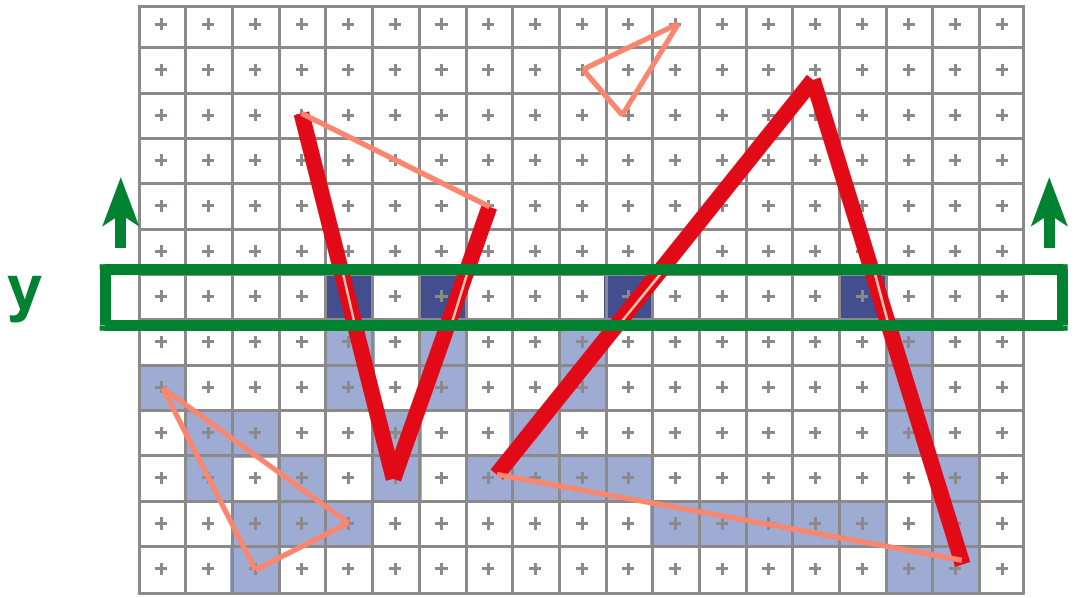
Scan Line and Active Edge lists

- Assume polygons are convex
  - Exactly 0 or 2 edges at each scanline. Why?



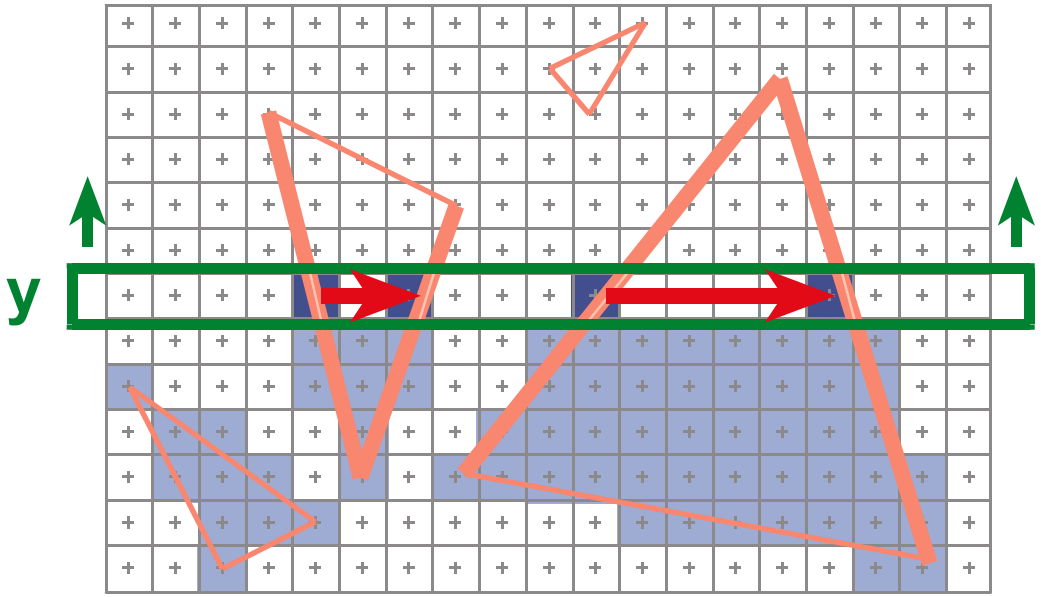
Scan Line : Principle

- Proceed row by row
- Maintain Active Edge List (AEL) (EdgeRecList)
- Edge Table (ET) for new edges at y (EdgeRecTable)



Scan Line : Principle

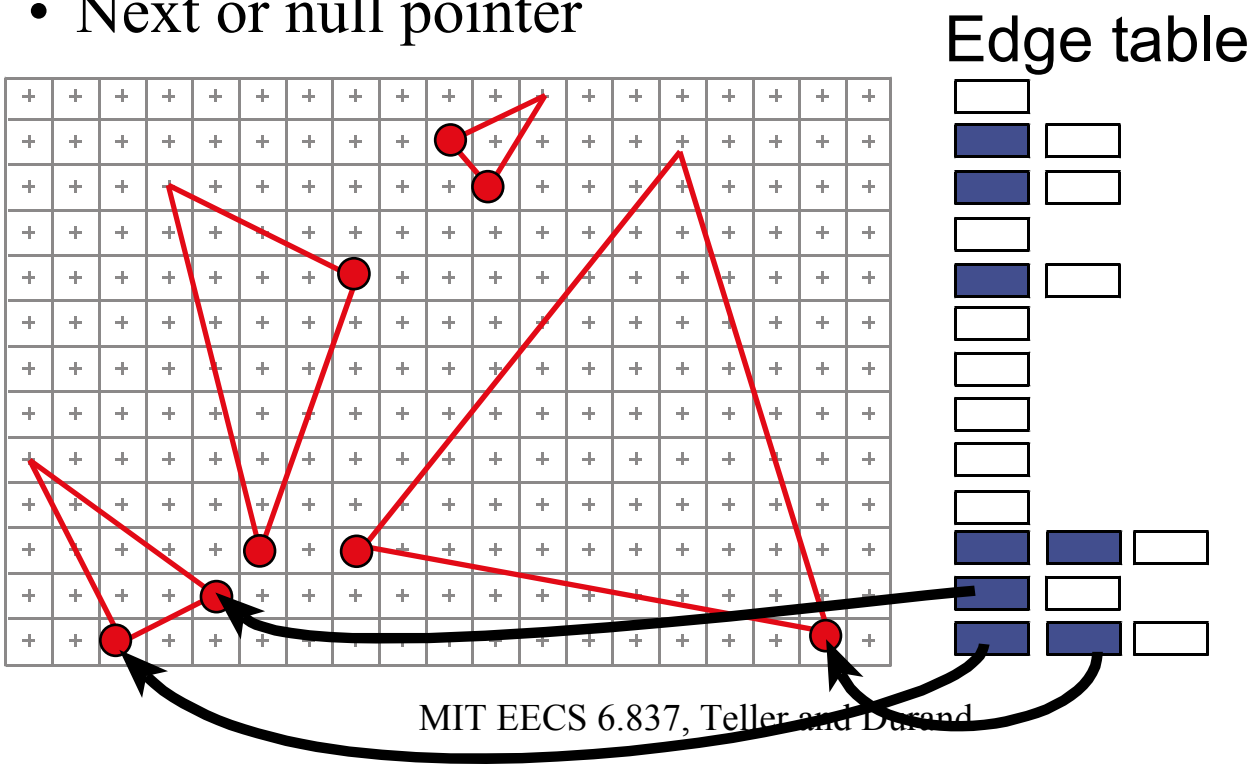
- Proceed row by row
- Maintain Active Edge List (AEL) (EdgeRecList)
- Edge Table (ET) for new edges at y (EdgeRecTable)



Will prove important for span filling

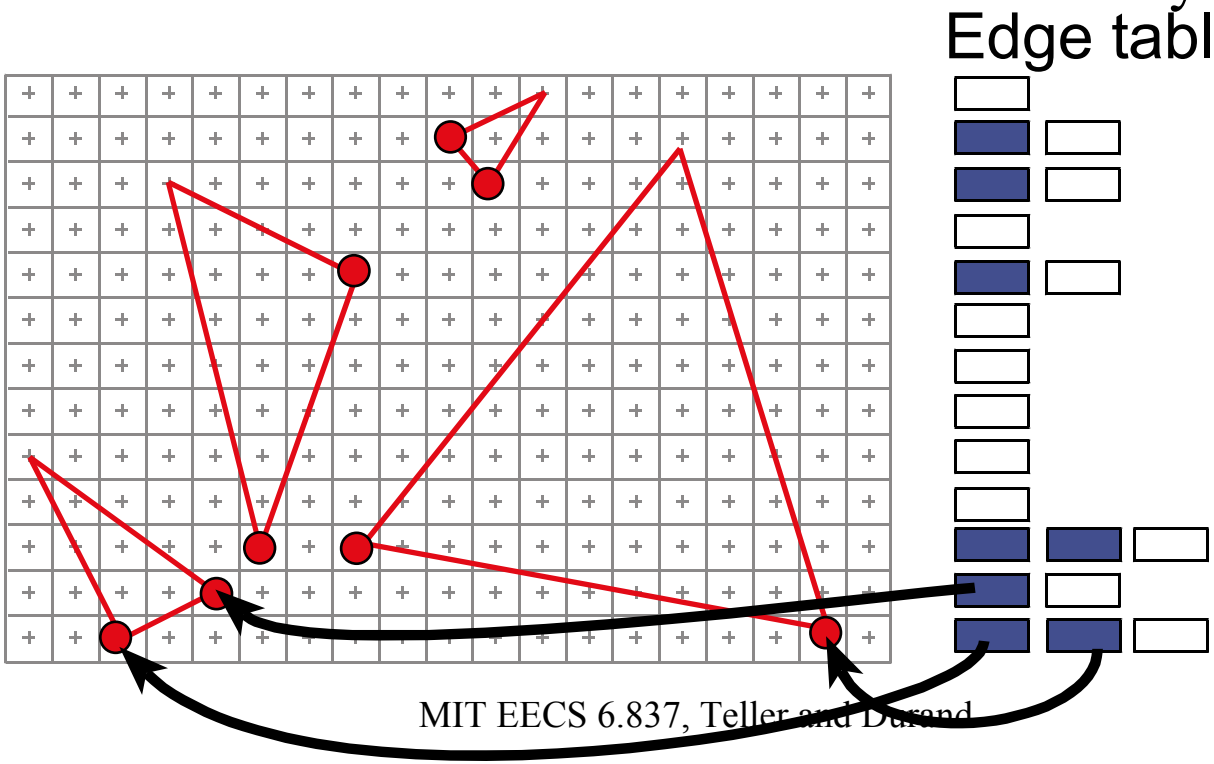
Precompute: Edge Table (EdgeRecTable)

- One entry per scan line (where edge begins)
- Each entry is a linked list of EdgeRecs, sorted by x
  - yend: y of top edge endpoint
  - xcurr, x: current x intersection, delta wrt y
  - Next or null pointer



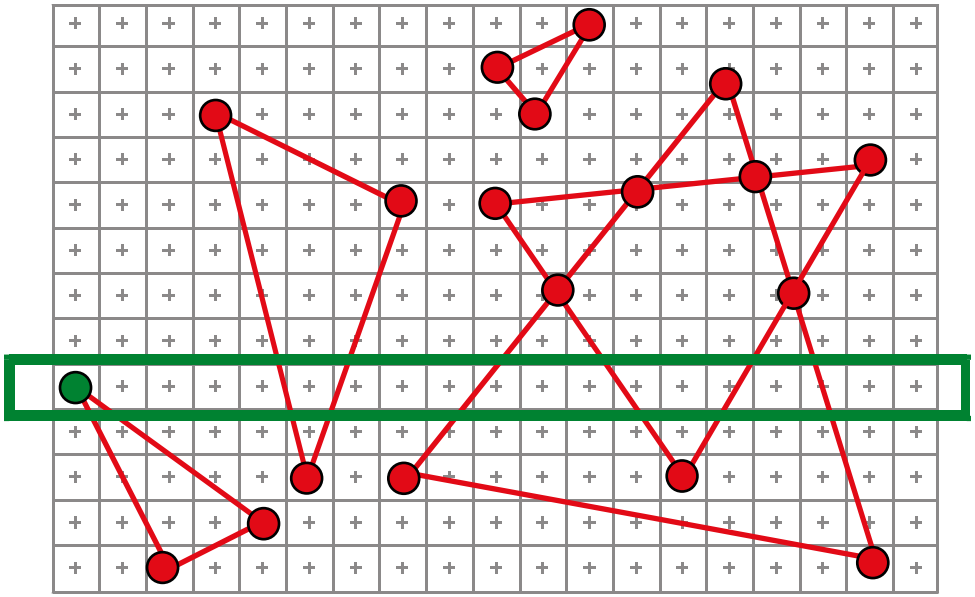
Initialization: events

- Edge Table
  - List of EdgeRecs, sorted by x
    - yend
    - xcurr, delta wrt y
- Active edge list (AEL)
  - Will be maintained
  - Store all active edges intersecting scanline
  - Ordered by x



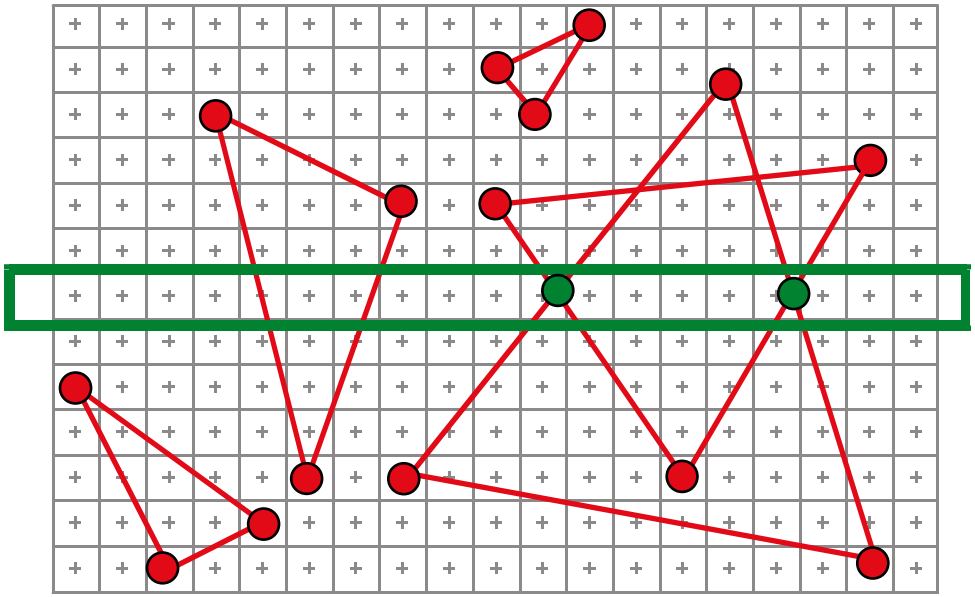
When Does AEL Change State?

- When a vertex is encountered
  - When an edge begins
    - All such events pre-stored in Edge Table
  - When and edge ends
    - Can be deduced from current Active Edge List



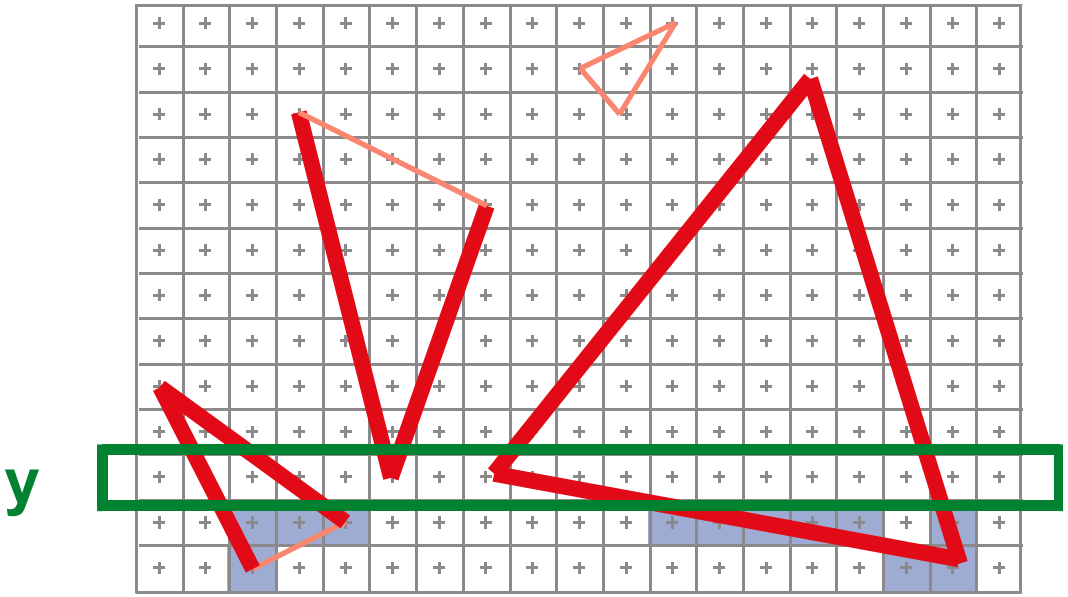
When Does AEL Change State?

- When a vertex is encountered
- When two edges change order along a scanline
  - I.e., when edges cross each other!
  - How to detect this efficiently?



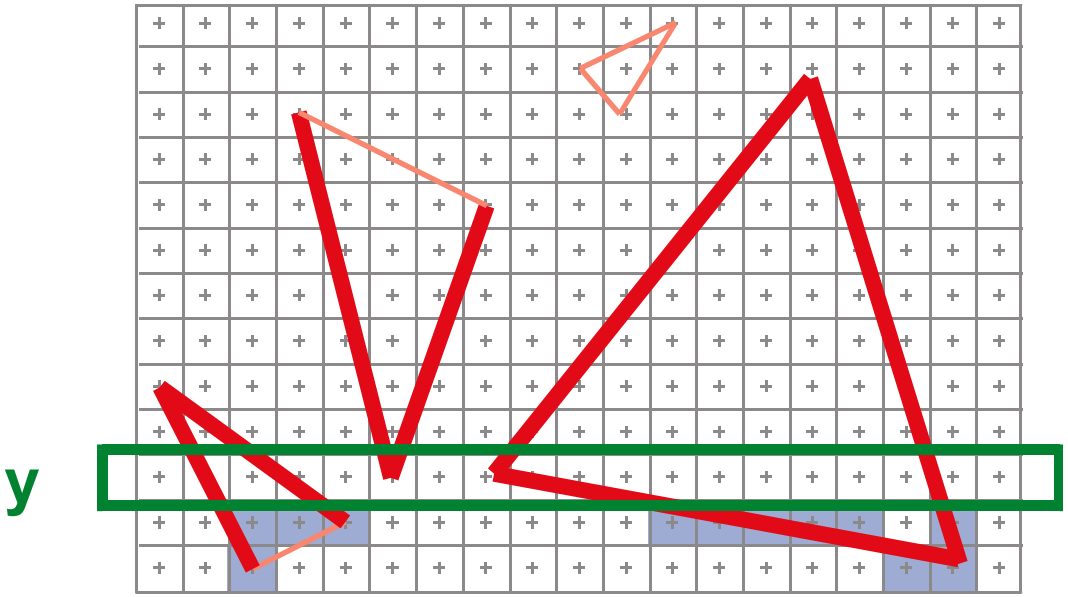
Processing a Scanline

- Add any edges which start in [y; y + 1[
  - Using Edge Table (EdgeRecTable)
  - How can x, y pre-sorting be exploited ?



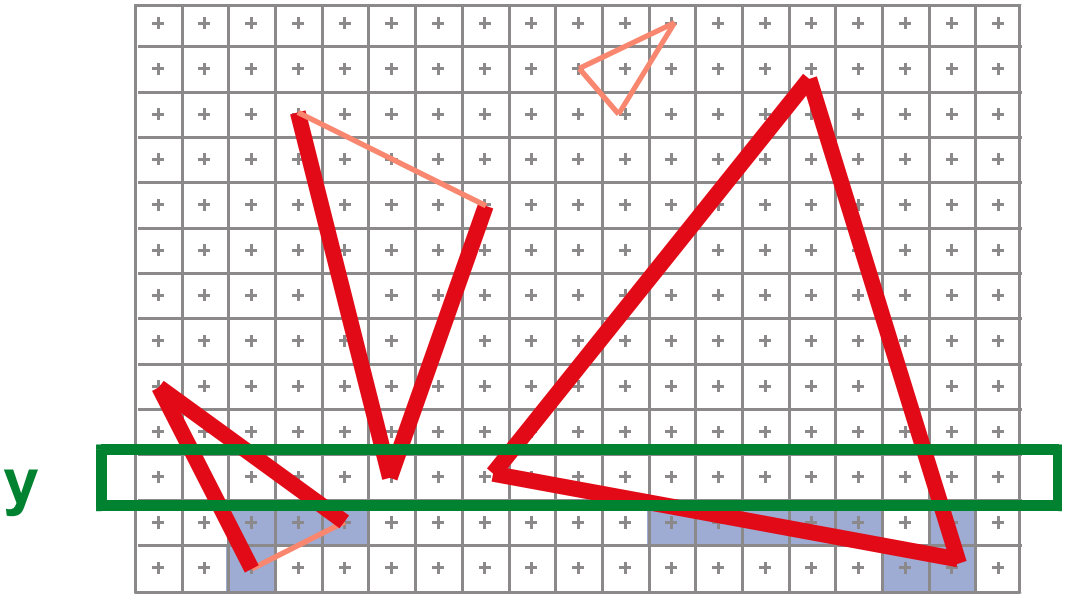
Processing a Scanline

- Add any edges which start in [y; y + 1[
- For each edge in AEL (EdgeRecList):
  - If edge ends in (y ,y-1) delete it from AEL
  - Otherwise, for next y: update x



Processing a Scanline

- Add any edges which start in [y; y + 1[
- For each edge in AEL:
  - If edge ends in (y ,y-1) delete it from AEL
  - Otherwise, for next y: update x



Write pixels]x(y), x(y+1)[

As you update x

Using Raster.setPixel(x')

Be careful with boundary conditions

Processing a Scanline

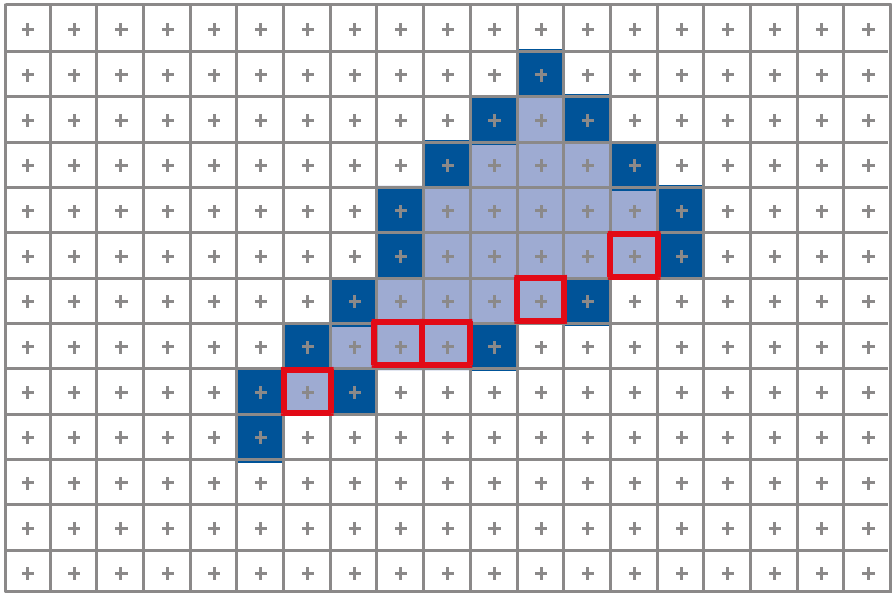
Write pixels]x(y), x(y+1)[

As you update x

Be careful with boundary conditions

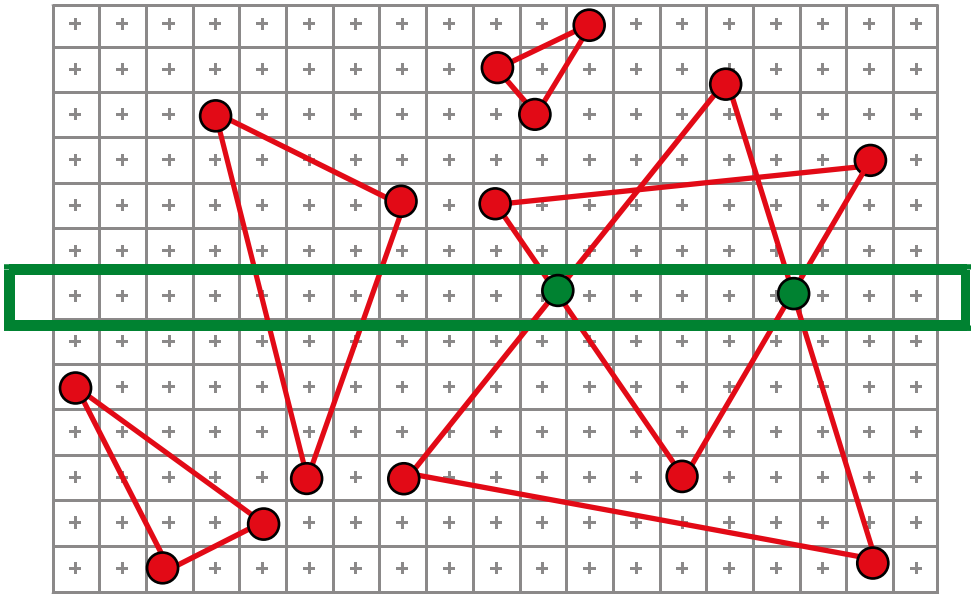
Advice: use setPixel(x(y))

Then loop on x(y)+1::x(y+1)-1



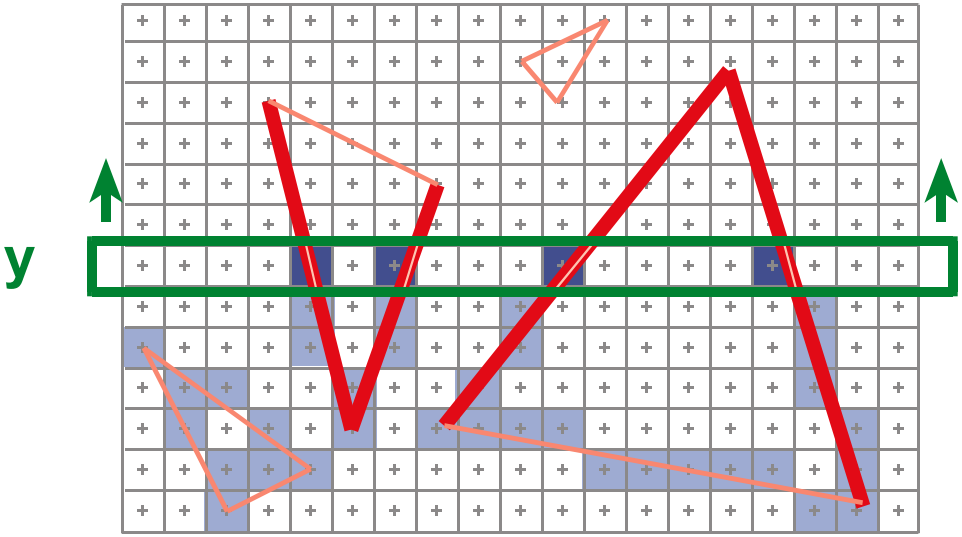
Processing a Scanline (continued)

- Once after each scanline is processed:
  - Sort AEL by x (mostly sorted, so...)
  - Increment scanline variable y
  - Raster.Write()



Scanline algorithm summary

- Initialize Raster, Polygons, Edge Table, AEL
- For each scanline y
  - Update AEL (insert edges from EdgeTable[y])
  - Assign raster of pixels from AEL
  - Update AEL (delete, increment, resort)



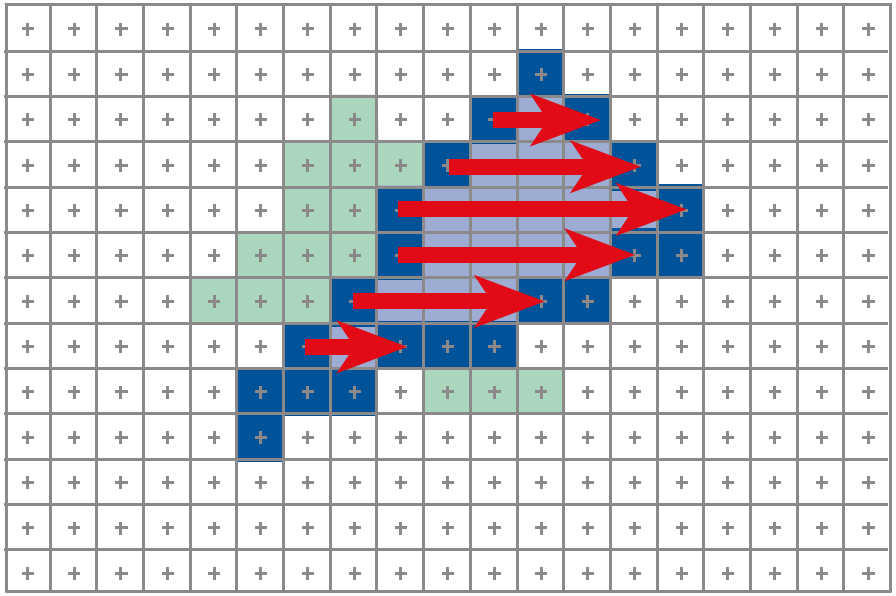
Scanline algorithm summary

- Initialize Raster(width), Polygons, Edge Table, AEL
- For each scanline y
  - Raster.init()
  - Update AEL (insert edges from EdgeTable[y])
  - For each edge
    - Update x
    - Assign raster of pixels from AEL
      - Using setPixel(x), on segment
      - Pay attention to boundary
    - If ends, delete
  - Resort AEL
  - Raster.write()

Questions?

Next week: polygon rasterization

- Span filling
- Value interpolation
- Visibility

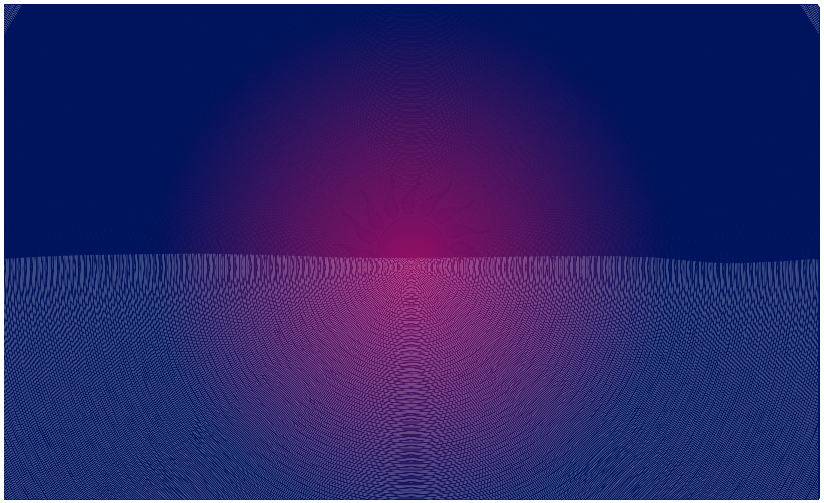


Assignment 3

- Due Friday October 4 at 5pm
- Line rasterization
- C++ only!
- Linux and SGI only!
- Scanline wireframe rendering
  - Will be used for assignment 4

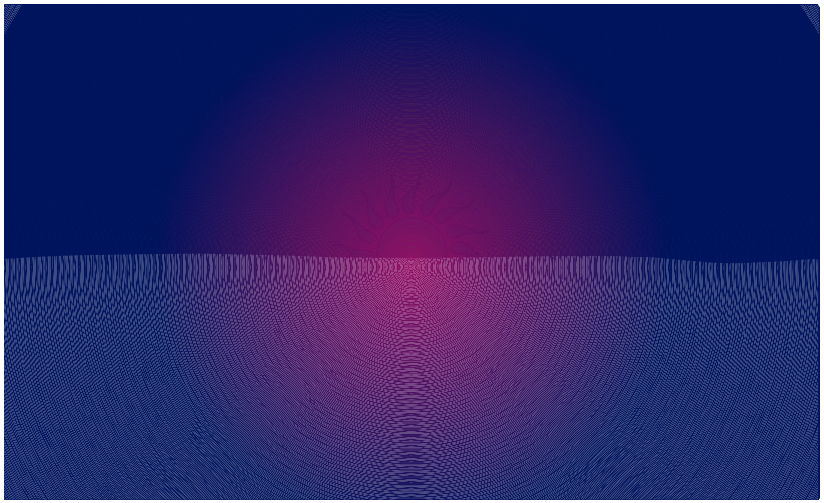
Playtime: Sunset

- We have seen why the sunset is red

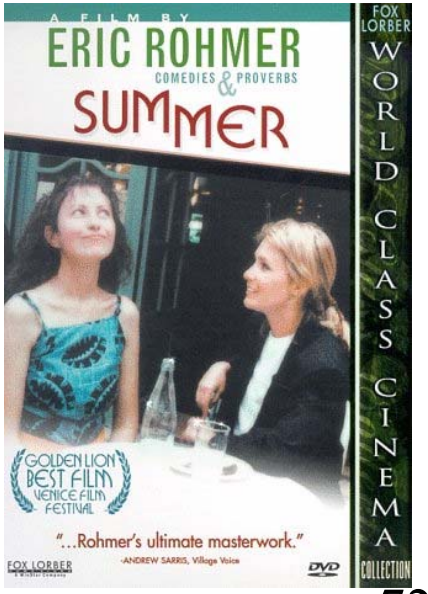


Playtime: Green flash

- The last ray of the Sun is green



- Any idea why?
- Also, novel by Jules Verne (19^{th} century)
  - Movie by Rohmer, *Summer*



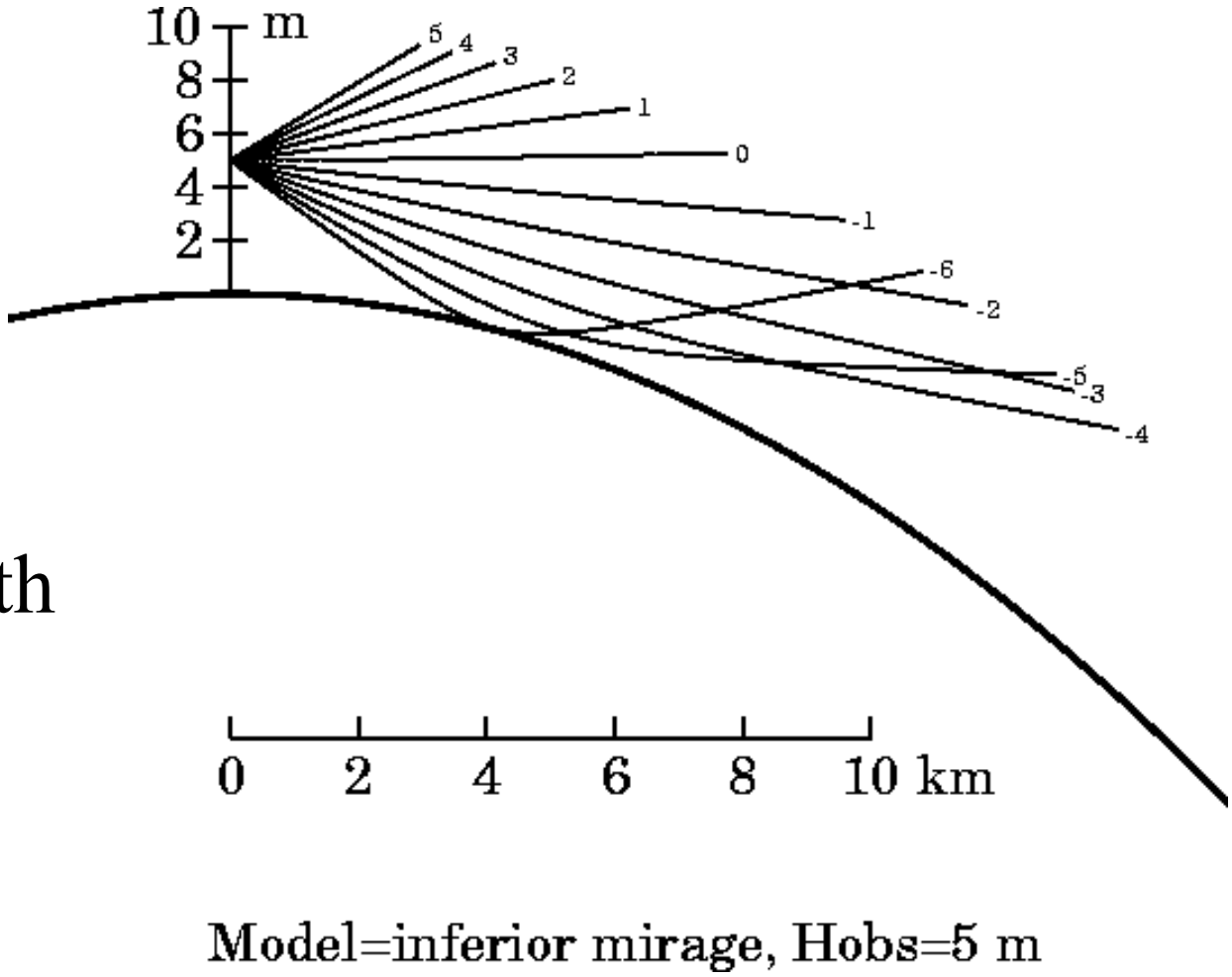
Green Flash

- **http://mintaka.sdsu.edu/GF/**
- http://www.mtwilson.edu/Tour/Lot/Green_Flash/
- http://www.intersoft.it/galaxlux/GreenFlashGallery.htm
- http://www.isc.tamu.edu/~astro/research/sandiego.html
- http://www.meteores.net/rv.html
- http://flzhgn.home.mindspring.com/grnray.htm

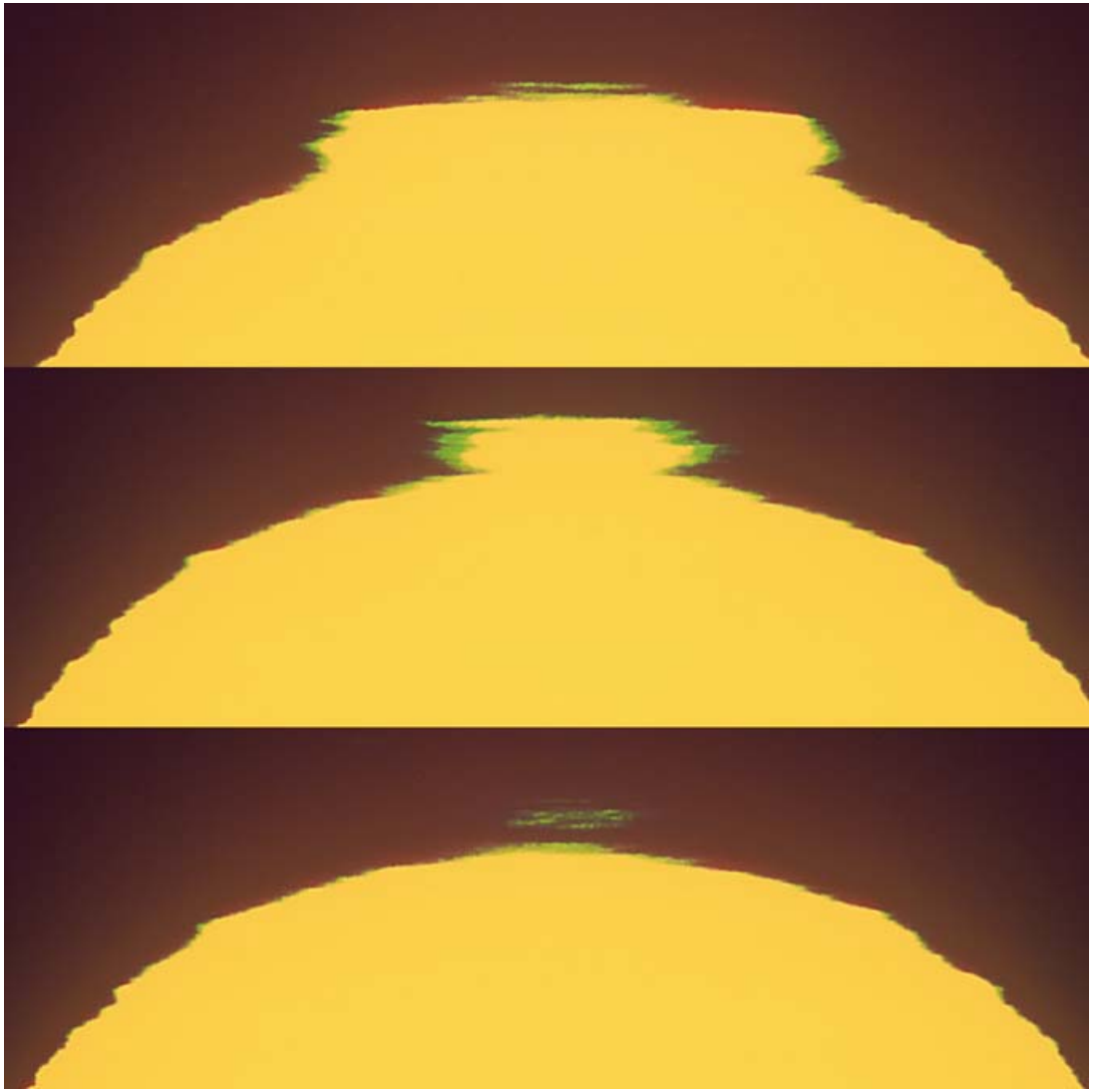
Green flash

Green flash

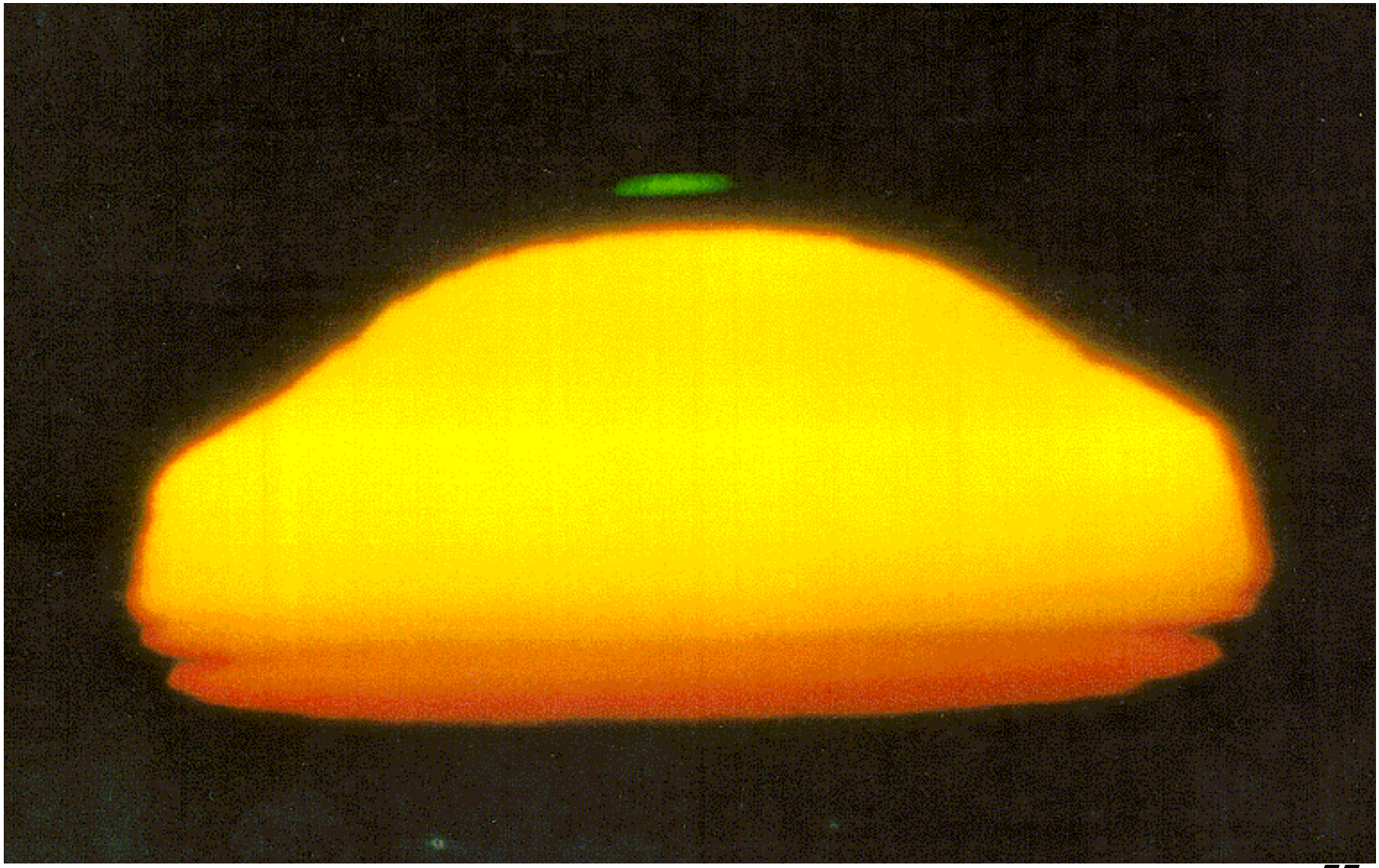
- Mirage
- Refraction
  - temperature gradient
- Ω shape
- Depends on wavelength



Model=inferior mirage, Hobs=5 m



Green flash



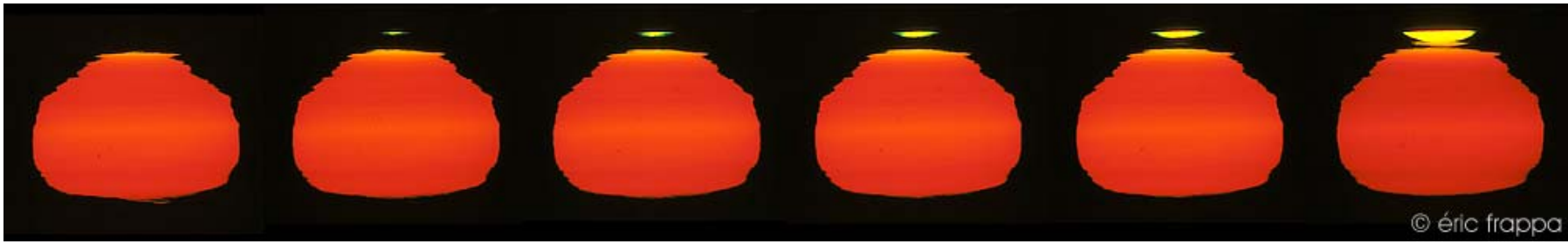
Green flash

- Simulation

Green flash

Green flash

- Why don’t you see green flash at sunrise?



Events

- Event list of buckets, 1 per scan line
  - Start/finish of an edge (vertex)
  - Occurs within interval [y; y + 1)
  - In each bucket, events sorted by x coordinate