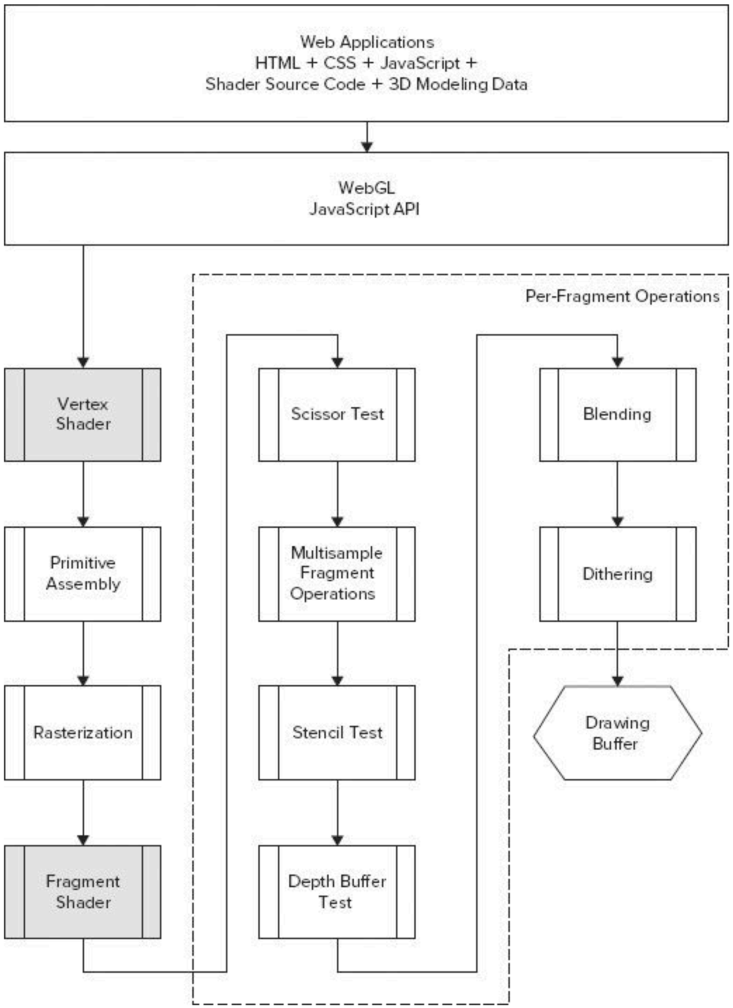
CS 418: Interactive Computer Graphics

Rasterization

Eric Shaffer

Based on John Hart's CS 418 Lecture Slides

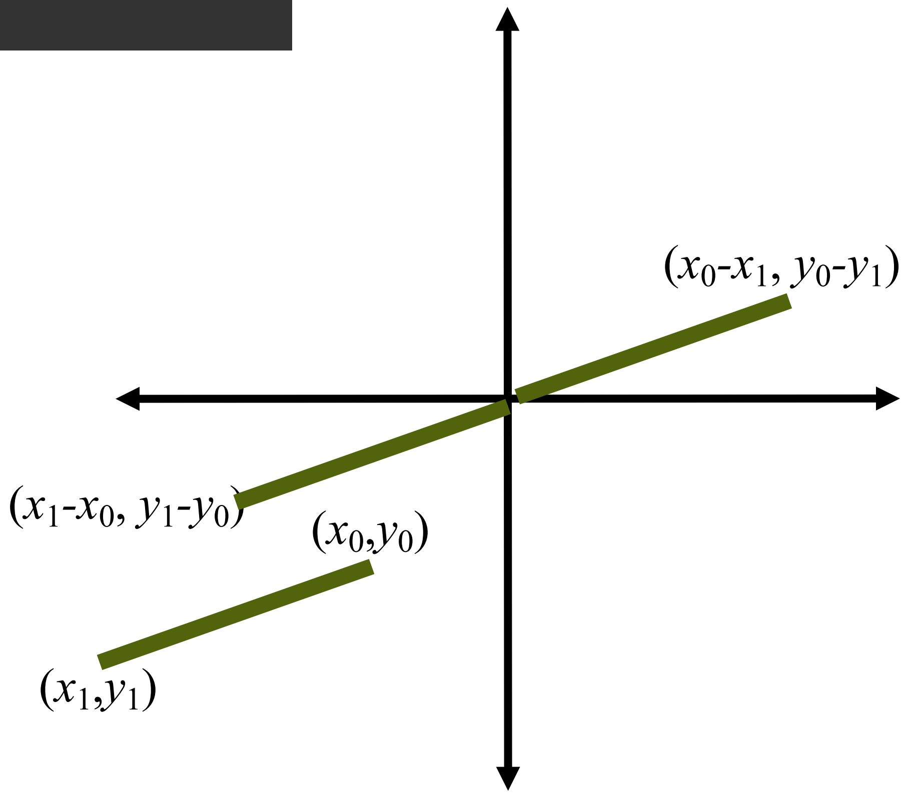
Fragment Pipeline

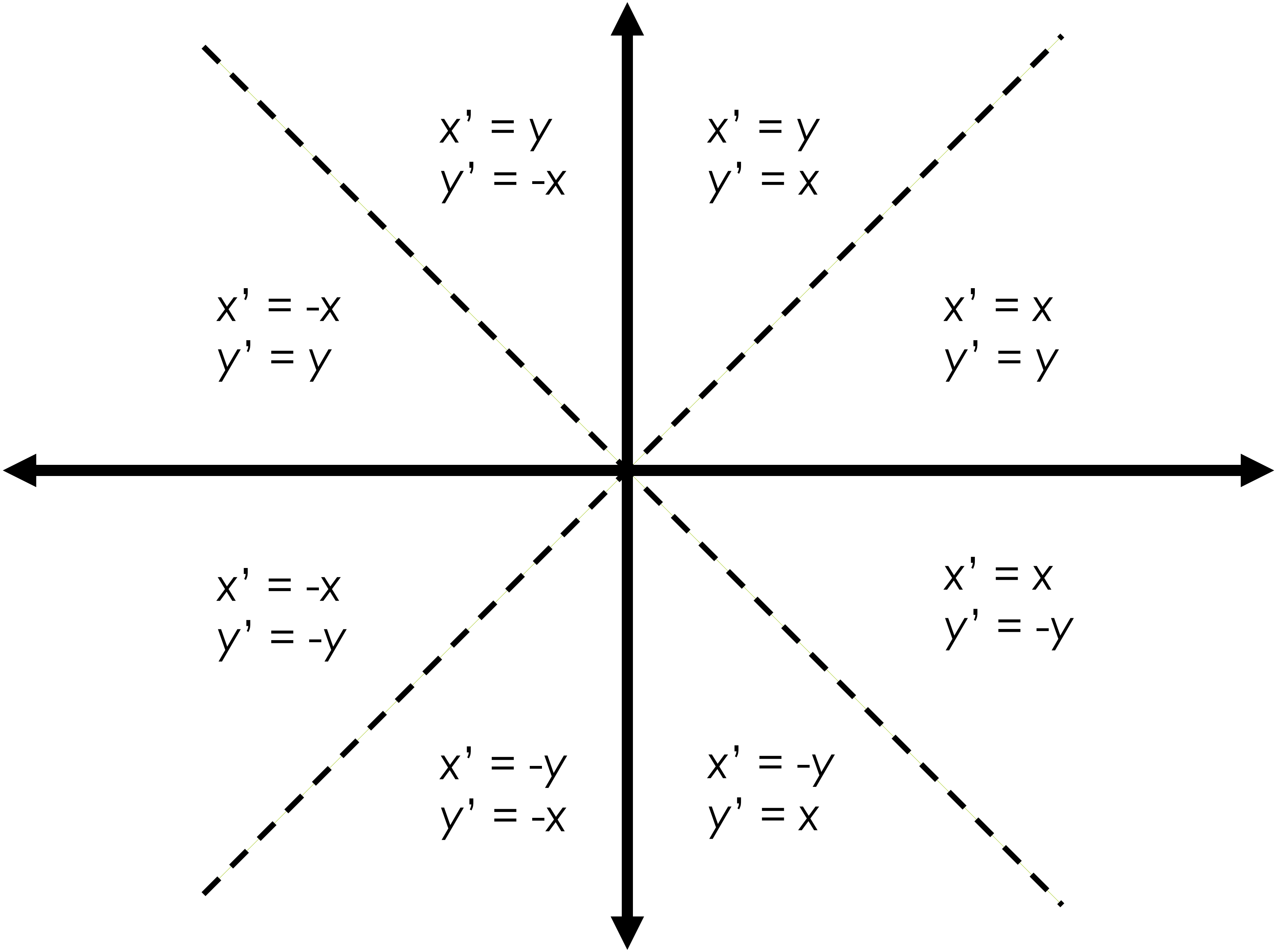


Rasterization is the process of converting vector graphics-style geometric primitives into a set of pixels

How to Draw a Line

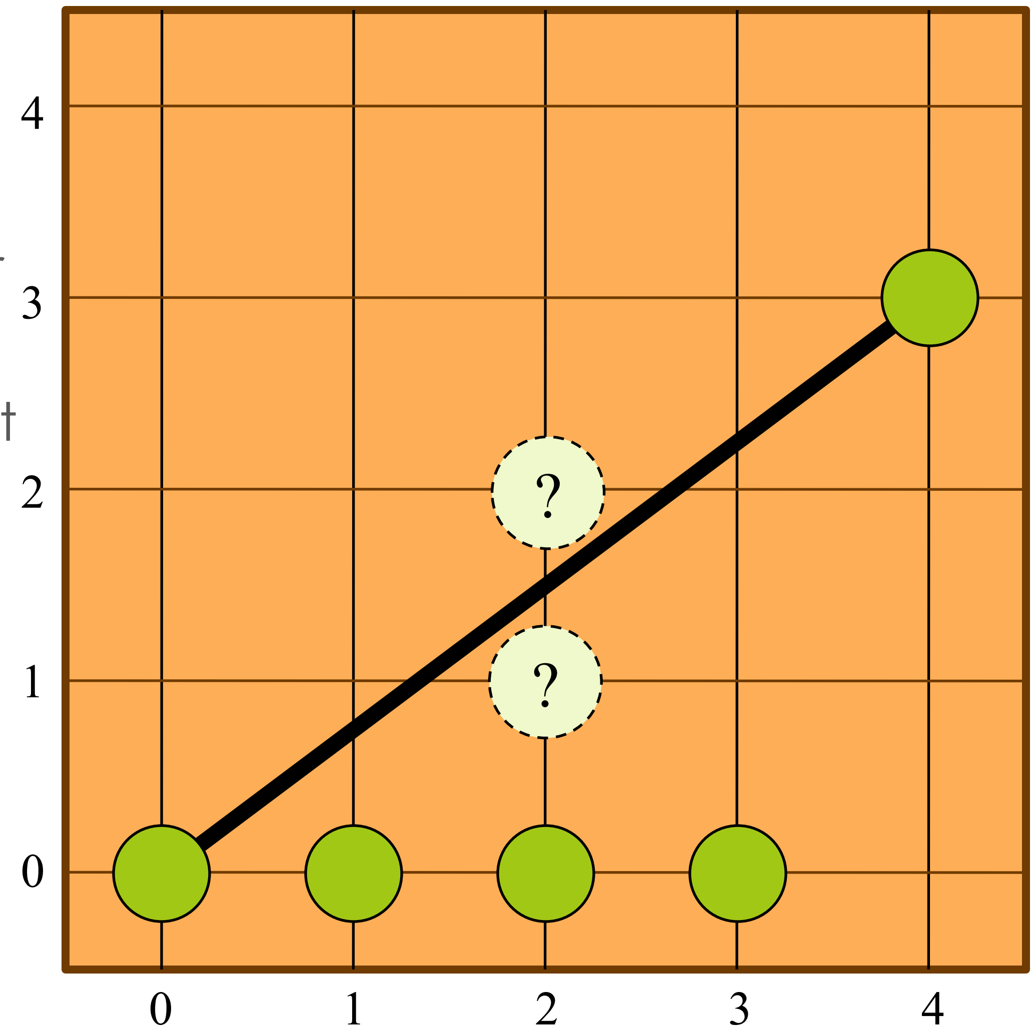
- Jack Bresenham's Algorithm
- Given a line from point (x_0, y_0) to (x_1, y_1)
- How do we know which pixels to illuminate to draw the line?
- First simplify the problem to the first octant
  - Translate start vertex to origin
  - Reflect end vertex into first octant




x' = y
y' = -x
x' = y
y' = x
x' = -x
y' = y
x' = x
y' = y
x' = -x
y' = -y
x' = x
y' = -y
x' = -y
y' = -x
x' = -y
y' = x

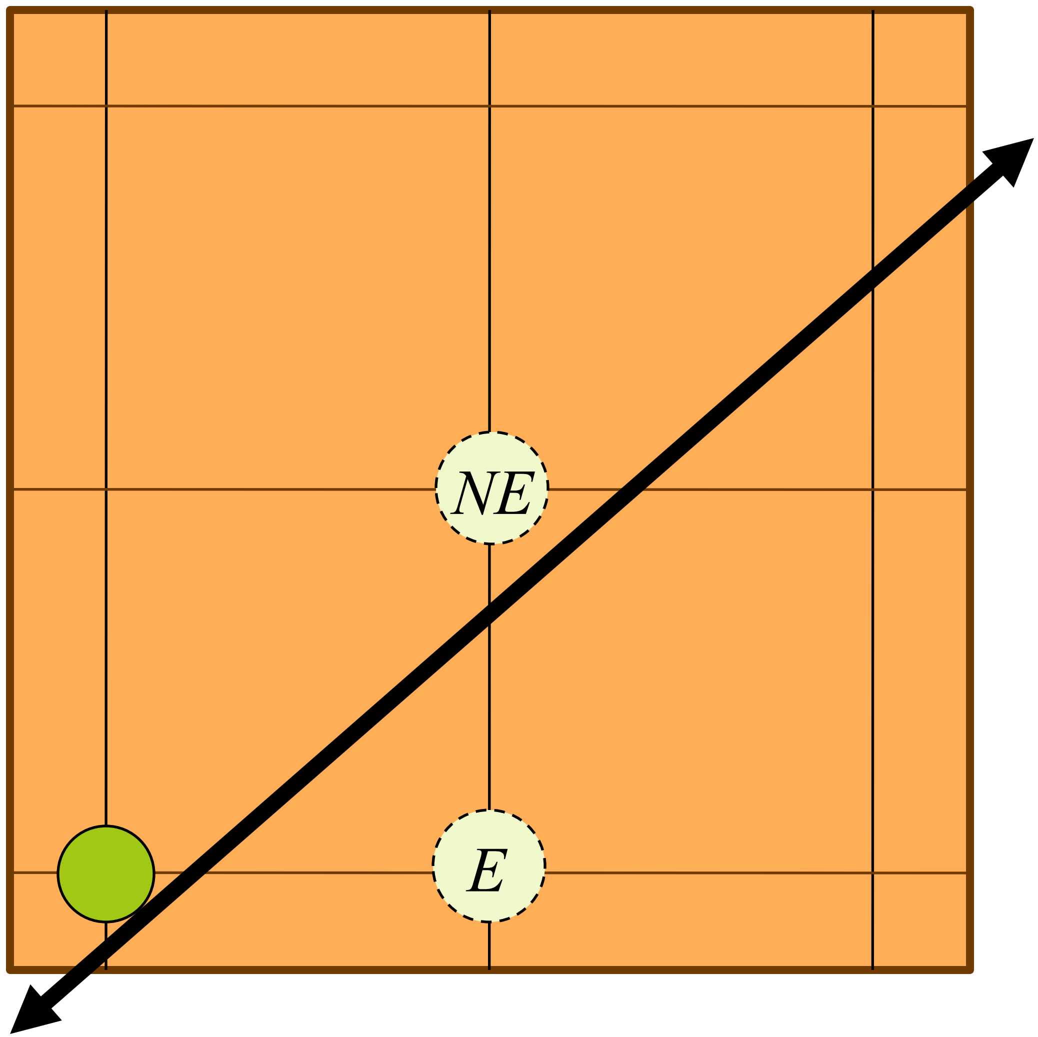
Line Rasterization

- How to rasterize a line from (0,0) to (4,3)
- Pixel (0,0) and (4,3) easy
- One pixel for each integer x-coordinate
- Pixel's y-coordinate closest to line
- If line equal distance between two pixels, pick one arbitrarily but consistently



Midpoint Algorithm

- Which pixel should be plotted next?
  - East?
  - Northeast?



Midpoint Algorithm

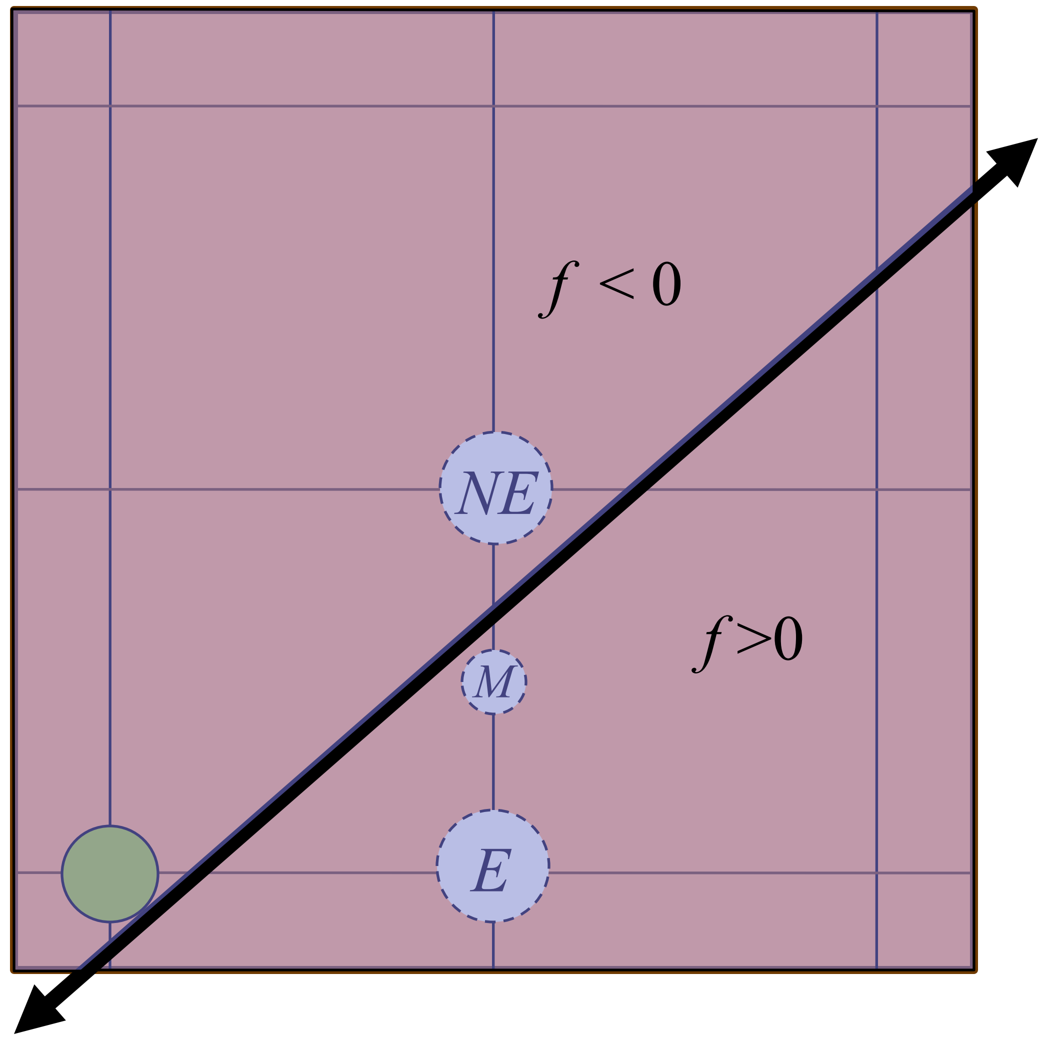
- Which pixel should be plotted next?
  - East?
  - Northeast?
- Line equation

$$y = mx + b$$

$$m = (y_1 - y_0)/(x_1 - x_0)$$

$$b = y_0 - mx_0$$

$$f(x,y) = mx + b - y$$



Midpoint Algorithm

- Which pixel should be plotted next?
  - East?
  - Northeast?
- Line equation

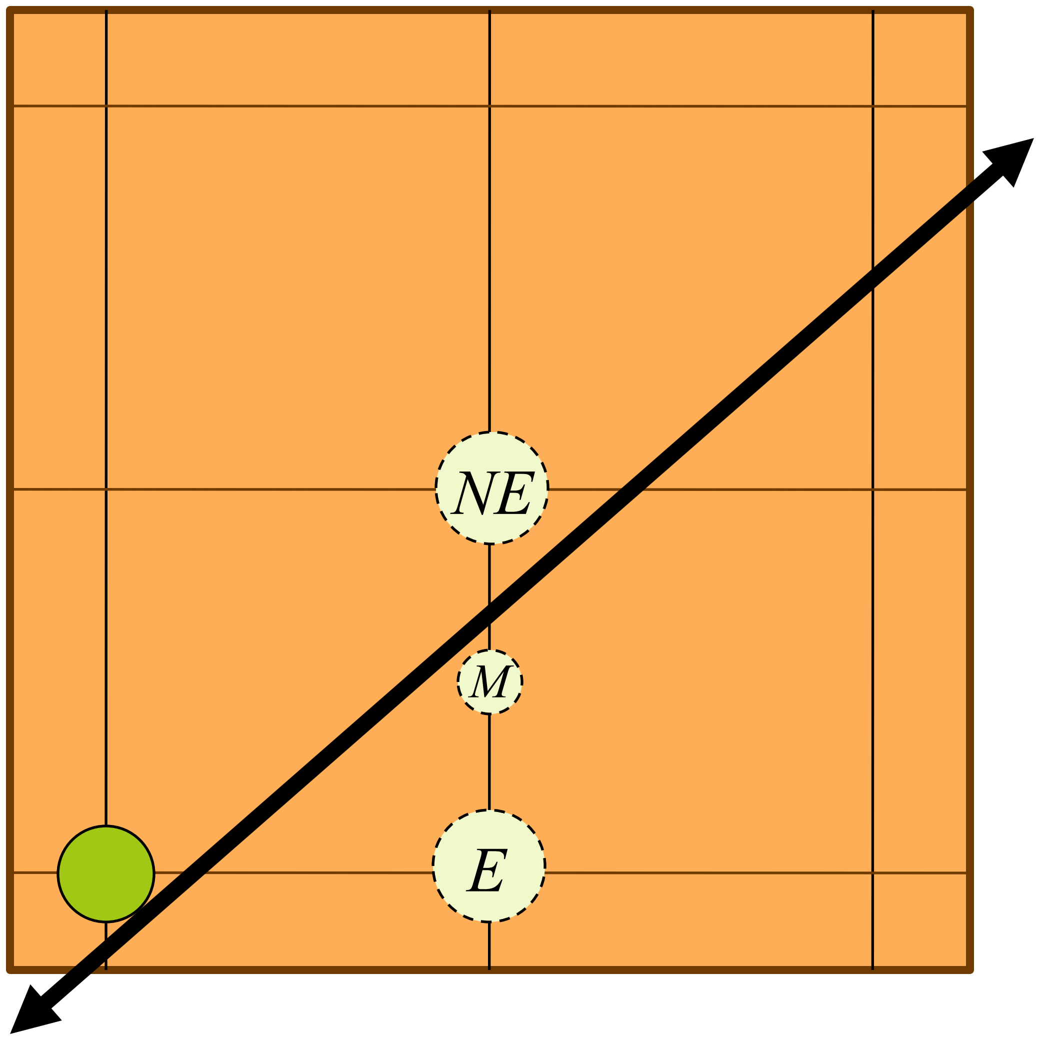
$$y = mx + b$$

$$m = (y_1 - y_0)/(x_1 - x_0)$$

$$b = y_0 - mx_0$$

$$f(x,y) = mx + b - y$$

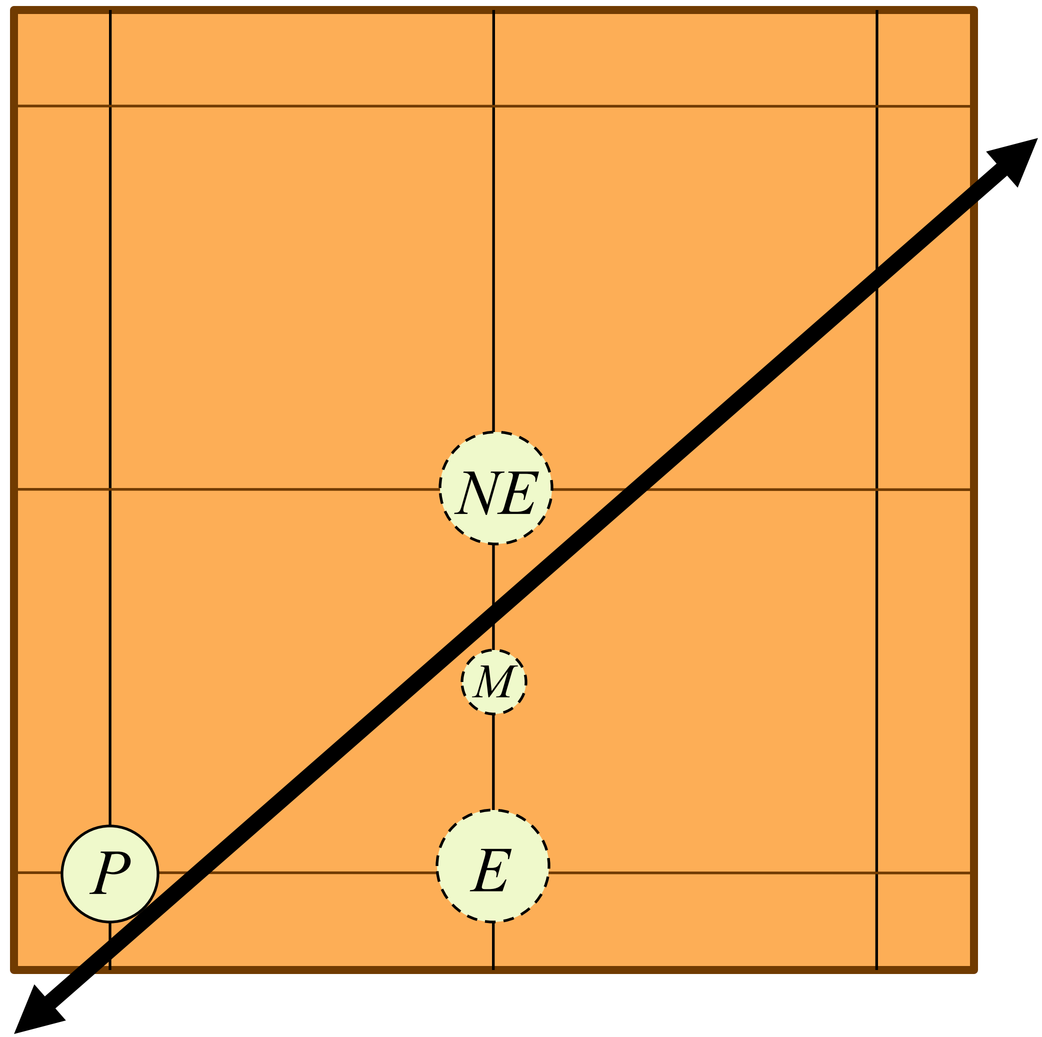
- $f(M) < 0 \rightarrow E$
- $f(M) \geq 0 \rightarrow NE$



Computing $f(M)$ Fast

If you know $f\,(P)$, then you can compute $f\,(M)$ easily:

$$
\begin{aligned}
f(x,y) &= mx + b - y \\
M &= P + (1,\tfrac{1}{2}) \\
f(M) &= f(x+1, y+\tfrac{1}{2}) \\
&= m(x+1) + b - (y+\tfrac{1}{2}) \\
&= mx + m + b - y - \tfrac{1}{2} \\
&= mx + b - y + m - \tfrac{1}{2} \\
&= f(P) + m - \tfrac{1}{2}
\end{aligned}
$$



Preparing next $f(P)$

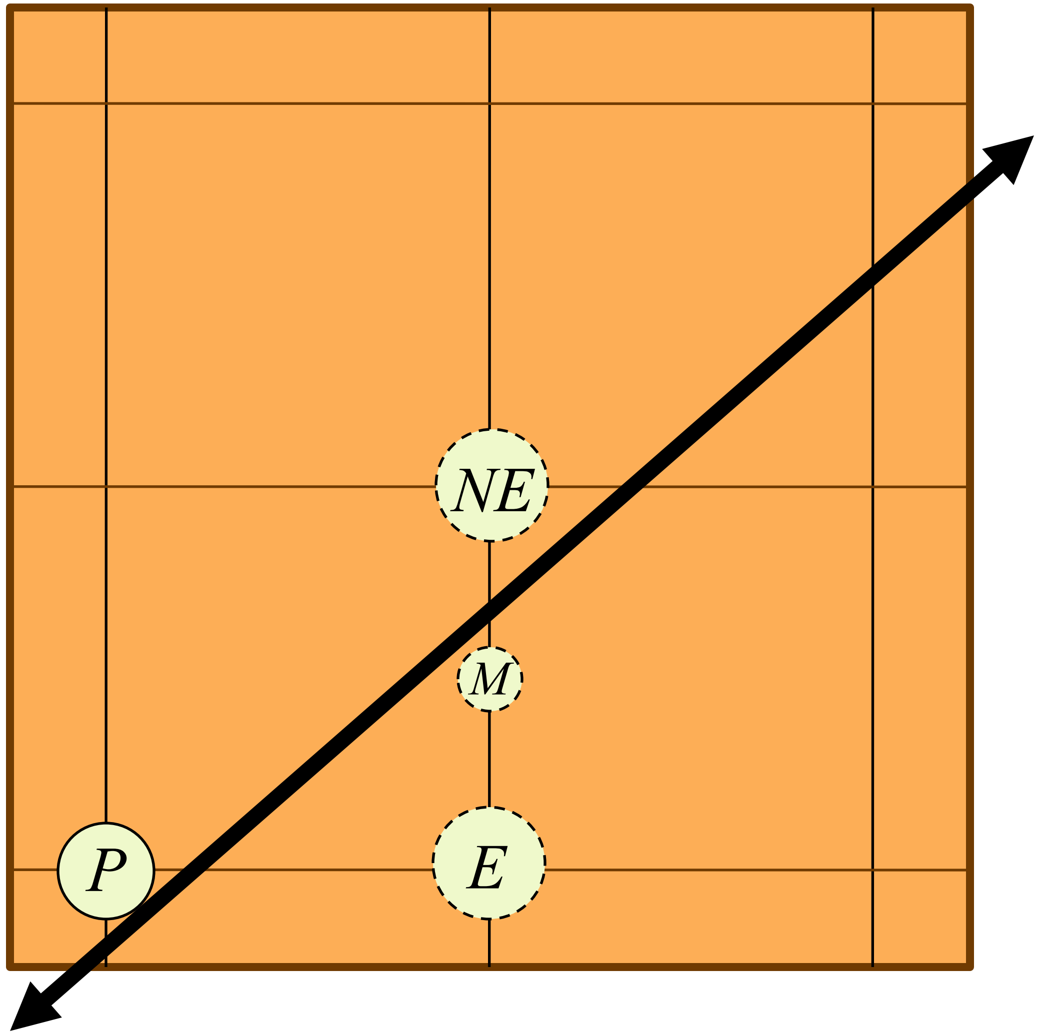
$f(x,y) = mx + b - y$

The next iteration's $f(P)$ is $f(E)$ or $f(NE)$ of this iteration

$$
\begin{aligned}
f(E) &= f(x+1,y) \\
&= f(P) + m \\
f(NE) &= f(x+1,y+1) \\
&= f(P) + m - 1
\end{aligned}
$$

Also need $f(P)$ at start point:

$$f(0,0) = b$$



Midpoint Increments

$f(M) = f(P) + m - ½$

If choice is E, then next midpoint is M_E

$f(M_E) = f(x+2, y+½)$

$= m(x+2) + b - (y+½)$

$= f(P) + 2m - ½$

$= f(M) + m$

Otherwise next midpt. is M_{NE}

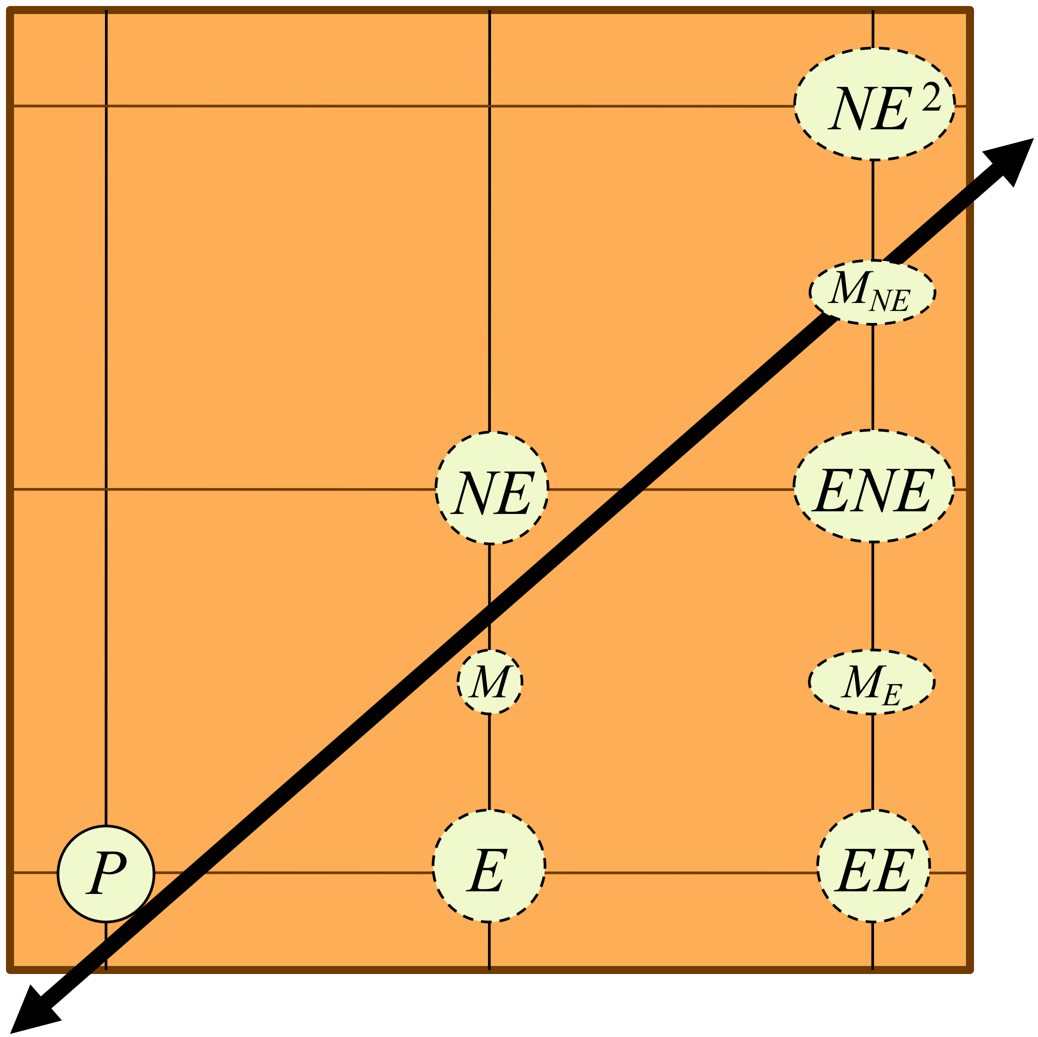
$f(M_{NE}) = f(x+2, y+1½)$

$= m(x+2) + b - (y+1½)$

$= f(P) + 2m - 1½$

$= f(M) + m - 1$

Initialize: $f(1, ½) = m + b - ½$



Integer Math

$f(M_E) = f(M) + m$

$f(M_{NE}) = f(M) + m - 1$

$f(1, \tfrac{1}{2}) = m + b - \tfrac{1}{2}$

$b = 0$

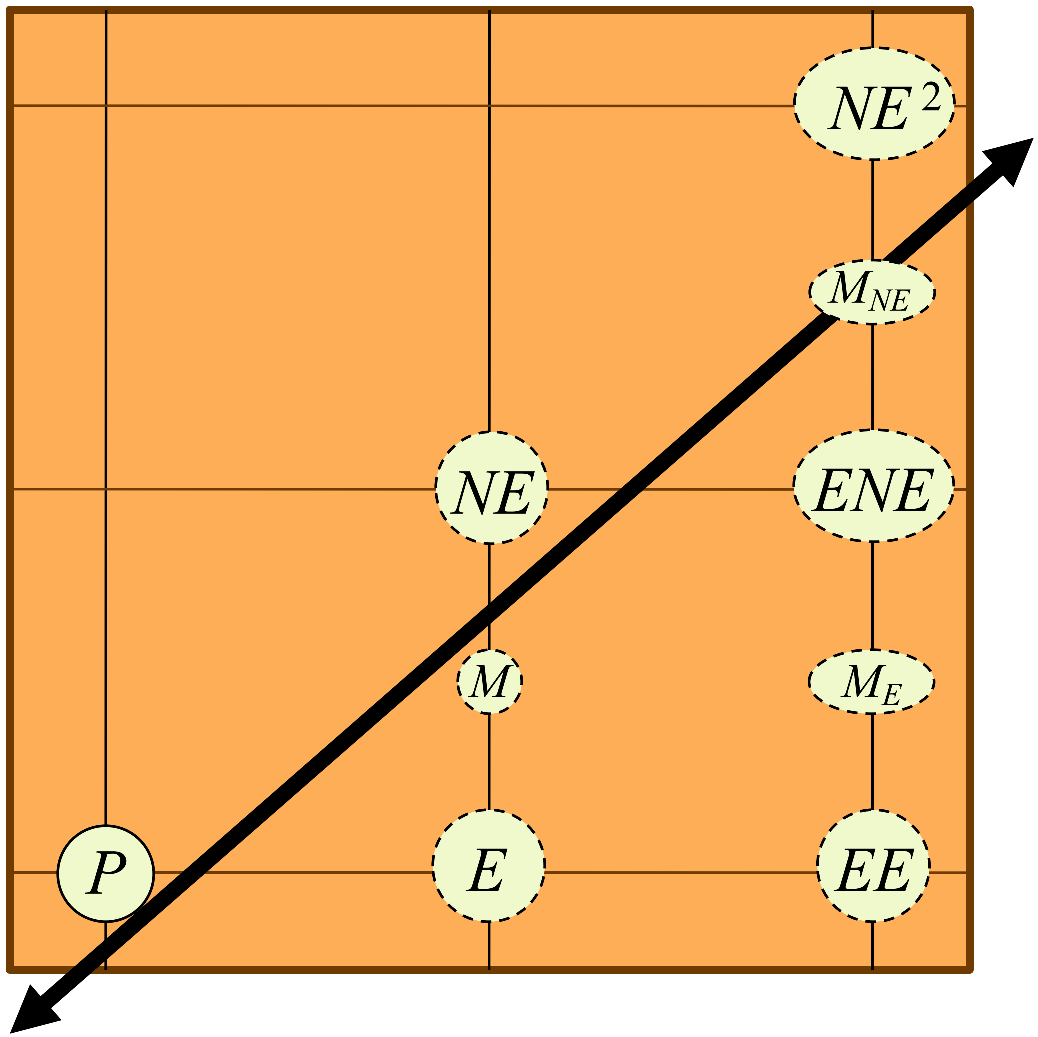
$m = (y_1 - y_0)/(x_1 - x_0)$

$= \Delta y / \Delta x$

$\Delta x\, f(M_E) = \Delta x\, f(M) + \Delta y$

$\Delta x\, f(M_{NE}) = \Delta x\, f(M) + \Delta y - \Delta x$

$\Delta x\, f(1, \tfrac{1}{2}) = \Delta y - \tfrac{1}{2}\, \Delta x$



Integer Math

$f(M_E) = f(M) + m$

$f(M_{NE}) = f(M) + m - 1$

$f(1, ½) = m + b - ½$

$b = 0$

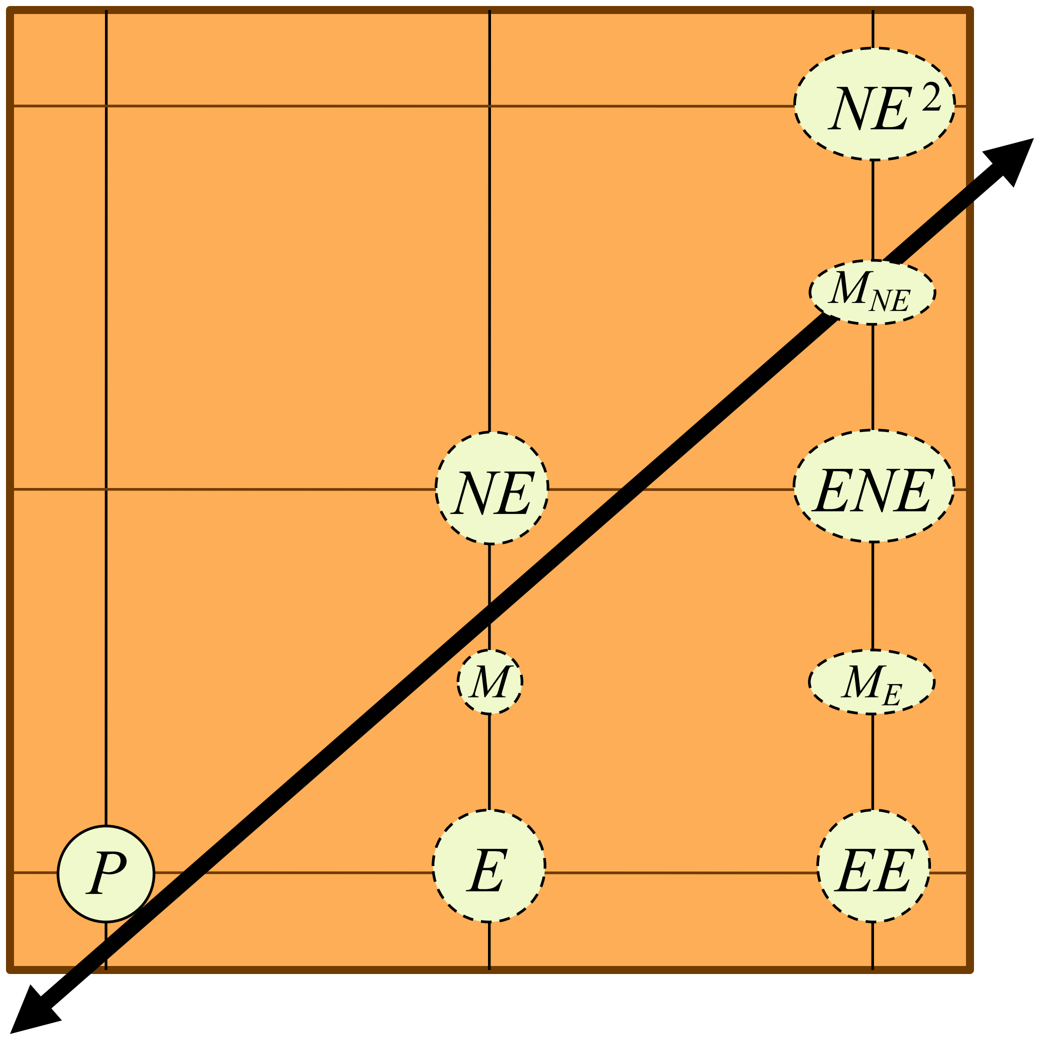
$m = (y_1 - y_0)/(x_1 - x_0)$

$= \Delta y/\Delta x$

$2\Delta x f(M_E) = 2\Delta x\, f(M) + 2\Delta y$

$2\Delta x f(M_{NE}) = 2\Delta x\, f(M) + 2\Delta y - 2\Delta x$

$2\Delta x f(1, ½) = 2\Delta y - \Delta x$



Integer Math

$f(M_E) = f(M) + m$

$f(M_{NE}) = f(M) + m - 1$

$f(1, ½) = m + b - ½$

$b = 0$

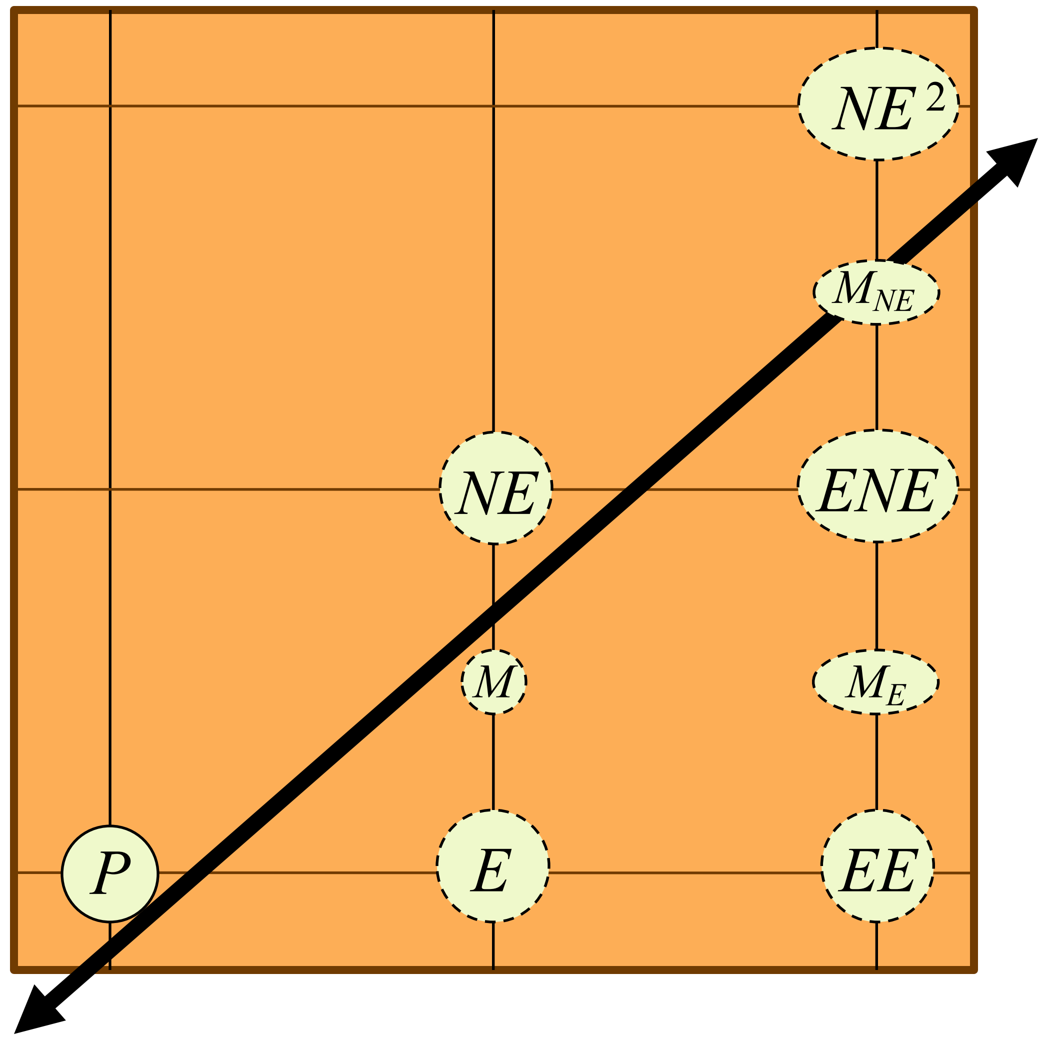
$m = (y_1 - y_0)/(x_1 - x_0)$

$= \Delta y/\Delta x$

$F(M_E) = F(M) + 2\Delta y$

$F(M_{NE}) = F(M) + 2\Delta y - 2\Delta x$

$F(1,½) = 2\Delta y - \Delta x$



Integer Math

$f(M_E) = f(M) + m$

$f(M_{NE}) = f(M) + m - 1$

$f(1, ½) = m + b - ½$

$b = 0$

$m = (y_1 - y_0)/(x_1 - x_0)$

$= \Delta y/\Delta x$

$F(M_E) = F(M) + 2\Delta y$

$F(M_{NE}) = F(M) + 2\Delta y - 2\Delta x$

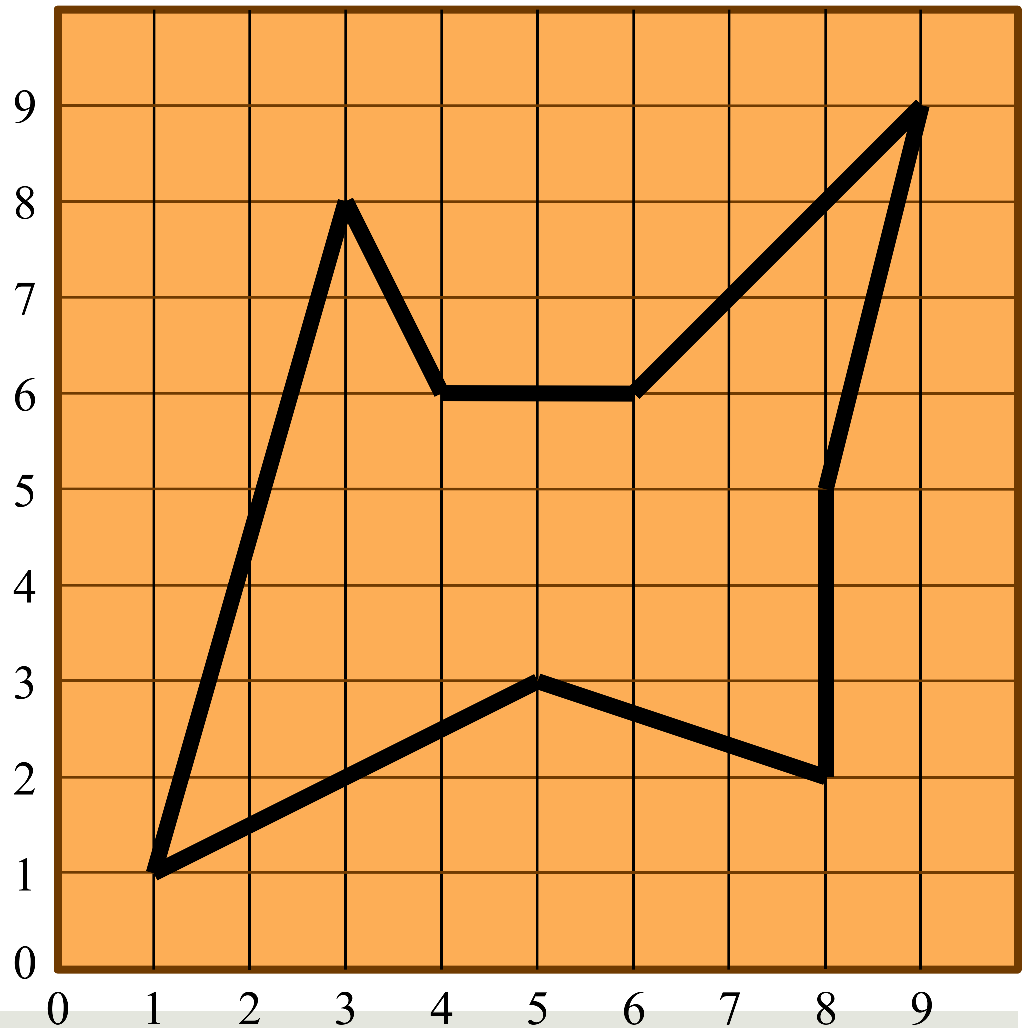
$F(1,½) = 2\Delta y - \Delta x$

Bresenham Line Algorithm

```
line(int x0,int y0,int x1,int y1)
{
  int dx = x1 – x0;
  int dy = y1 – y0;

  int F = 2*dy – dx;

  int dFE = 2*dy;
  int dFNE = 2*dy – 2*dx;

  int y = y0;
  for (int x = x0, x < x1; x++) {
    plot(x,y);
    if (F < 0) {
      F += dFE;
    } else {
      F += dFNE; y++;
    }
  }
}
```

Polygon Rasterization

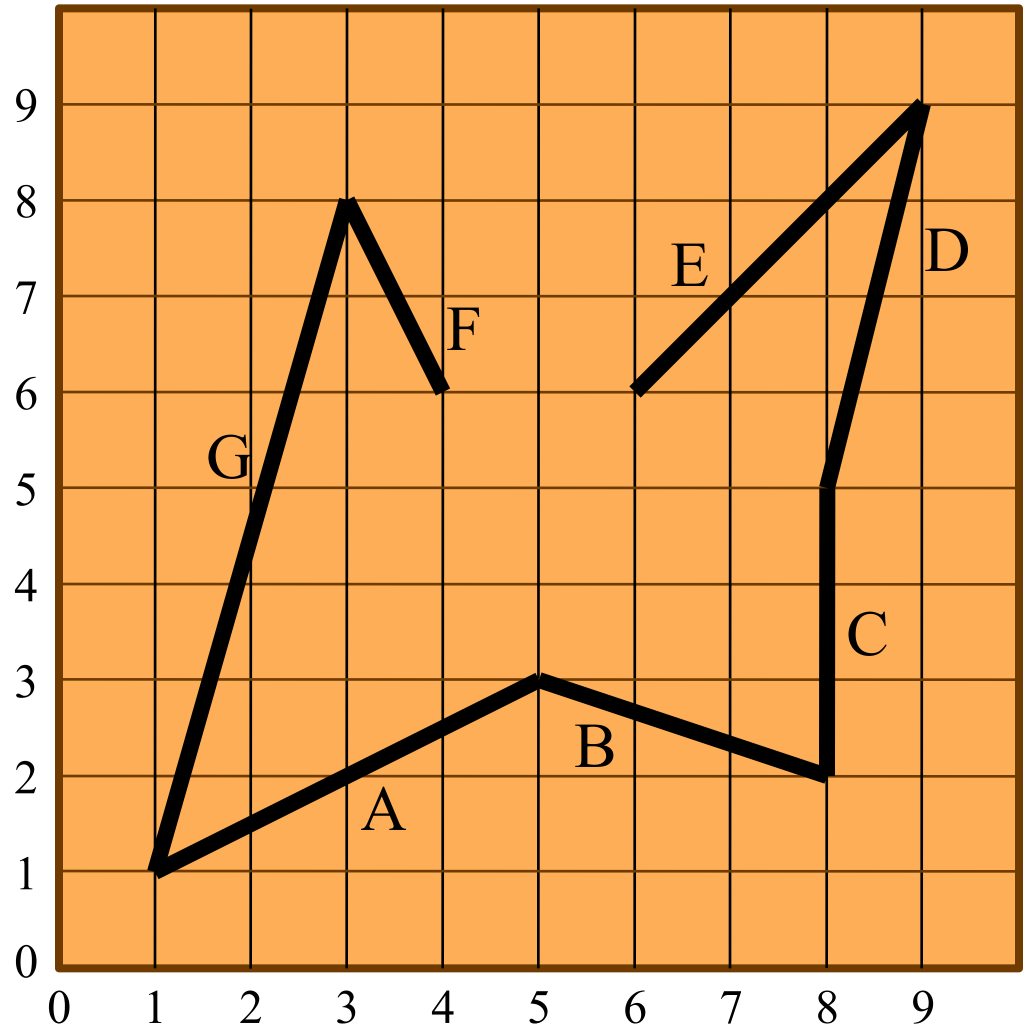
- Ignore horizontal lines



Polygon Rasterization

- Ignore horizontal lines
- Sort edges by smaller y coordinate

Edge	ymin
A	1
G	1
B	2
C	2
D	5
E	6
F	6

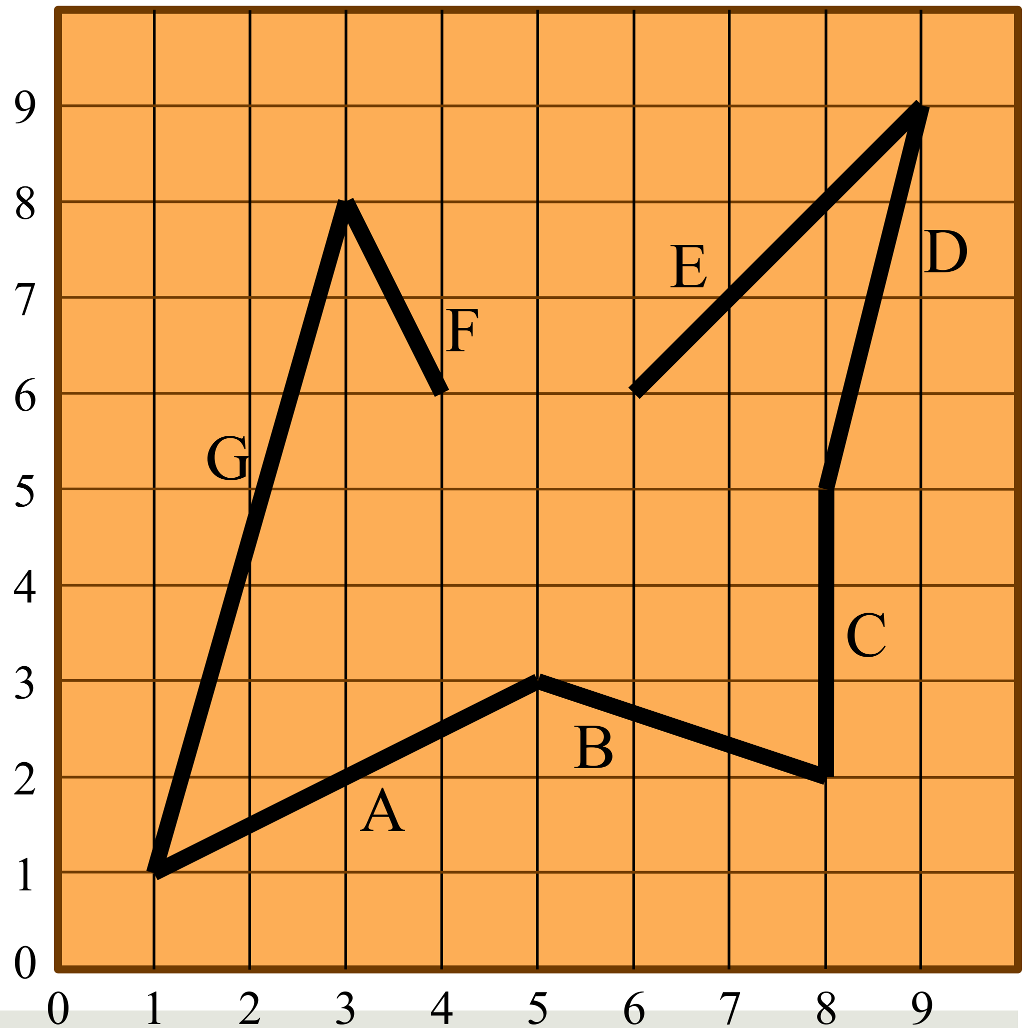


Polygon Rasterization

Edge	x	dx/dy	ymax

- For each scanline…
- Add edges where y = ymin
- Sorted by x
- Then by dx/dy

Edge	ymin
A	1
G	1
B	2
C	2
D	5
E	6
F	6



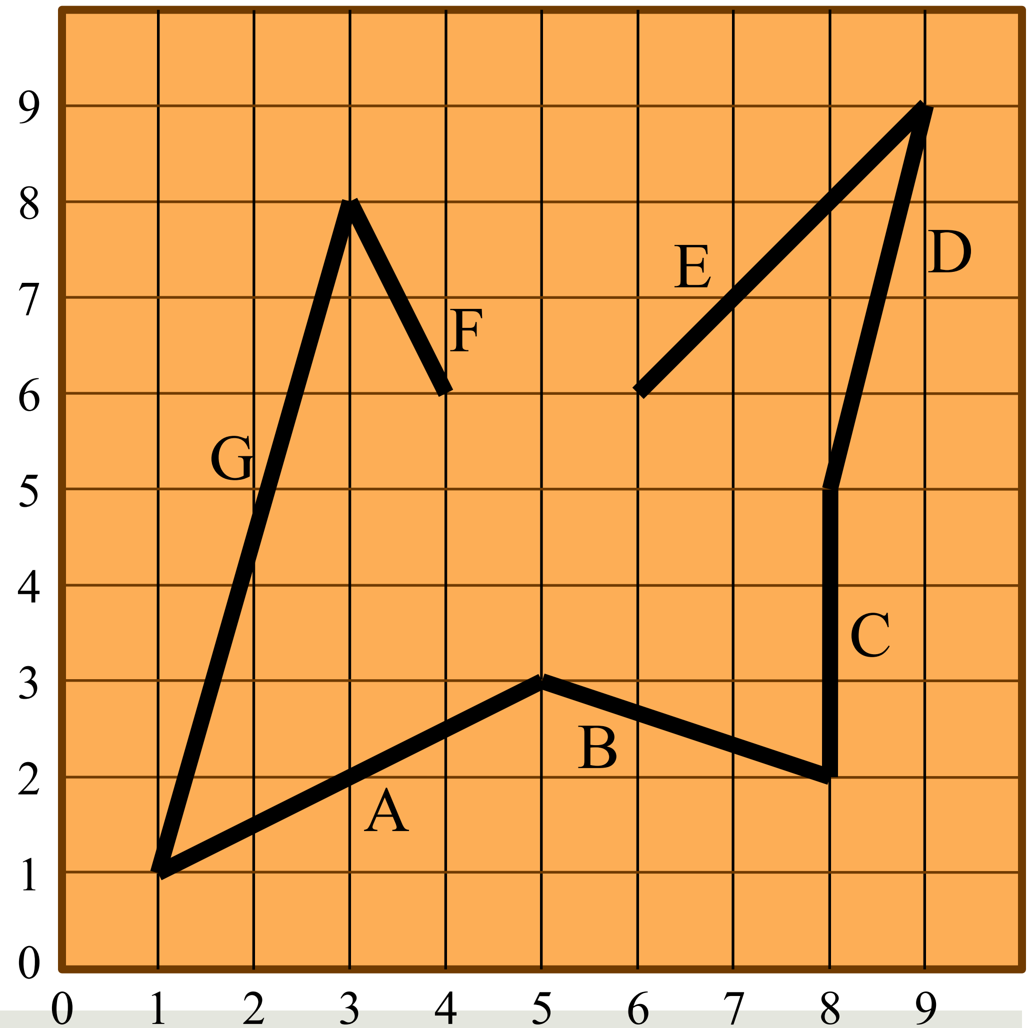
Polygon Rasterization

Edge	x	dx/dy	ymax

Plotting rules for when segments lie on pixels

1. Plot lefts
2. Don't plot rights
3. Plot bottoms
4. Don't plot tops

Edge	ymin
A	1
G	1
B	2
C	2
D	5
E	6
F	6

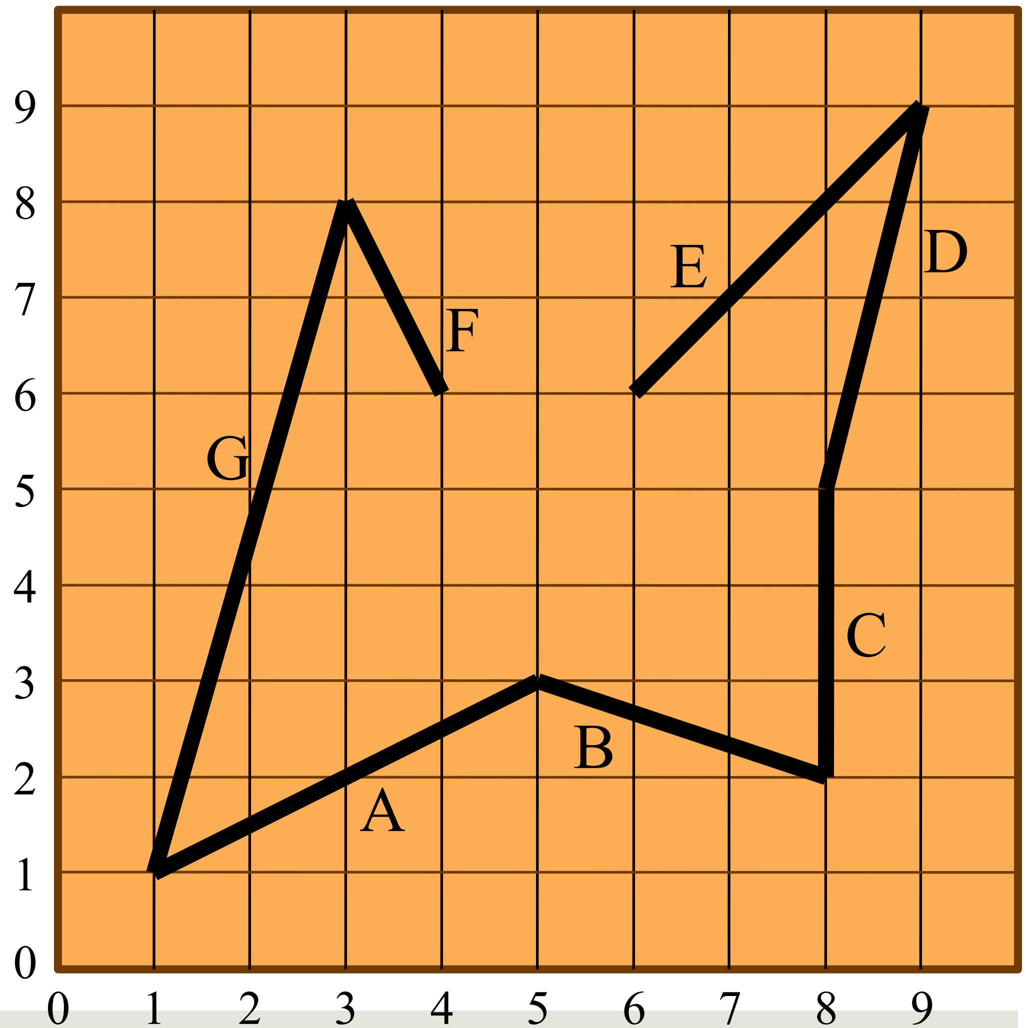


Polygon Rasterization

Edge	x	dx/dy	ymax
G	1	2/7	8
A	1	4/2	3

- y = 1
- Delete y = ymax edges
- Update x
- Add y = ymin edges
- For each pair x_0,x_1, plot from ceil(x_0) to ceil(x_1) – 1

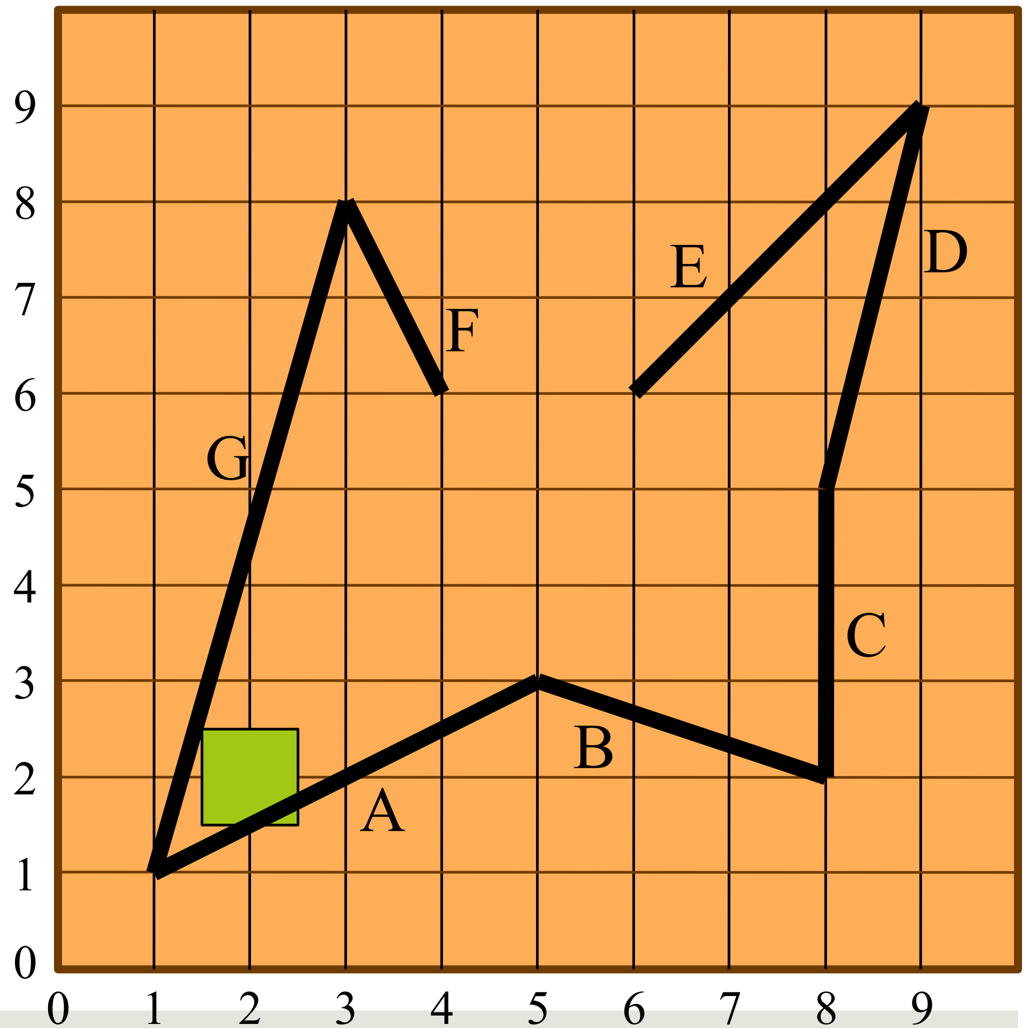
Edge	ymin
A	1
G	1
B	2
C	2
D	5
E	6
F	6



Polygon Rasterization

- y = 2
- Delete y = ymax edges
- Update x
- Add y = ymin edges
- For each pair x_0, x_1, plot from ceil(x_0) to ceil(x_1) – 1

Edge	x	dx/dy	ymax
G	1 2/7	2/7	8
A	3	4/2	3
B	8	-3/1	3
C	8	0/3	5



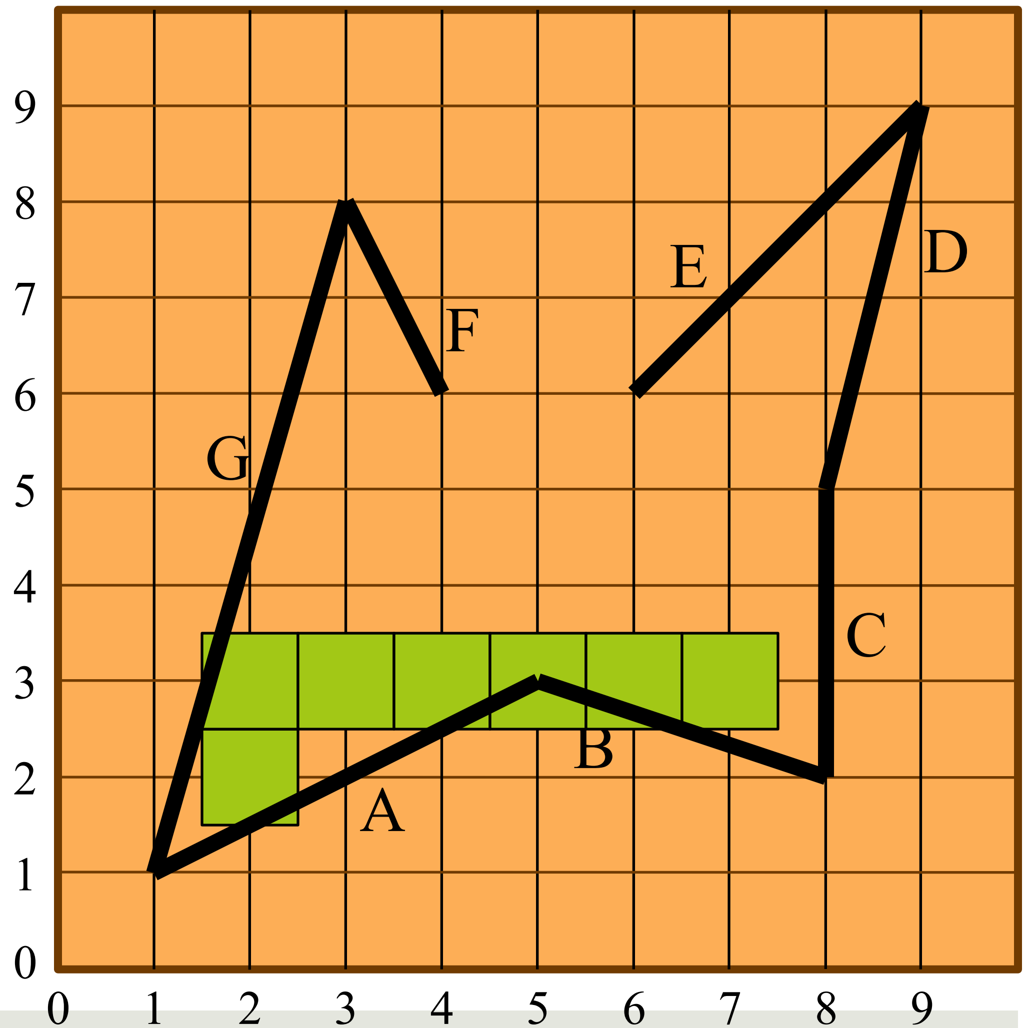
Edge	ymin
A	1
G	1
B	2
C	2
D	5
E	6
F	6

Polygon Rasterization

Edge	x	dx/dy	ymax
G	1 4/7	2/7	8
C	8	0/3	5

- y = 3
- Delete y = ymax edges
- Update x
- Add y = ymin edges
- For each pair x_0,x_1, plot from ceil(x_0) to ceil(x_1) – 1

Edge	ymin
A	1
G	1
B	2
C	2
D	5
E	6
F	6

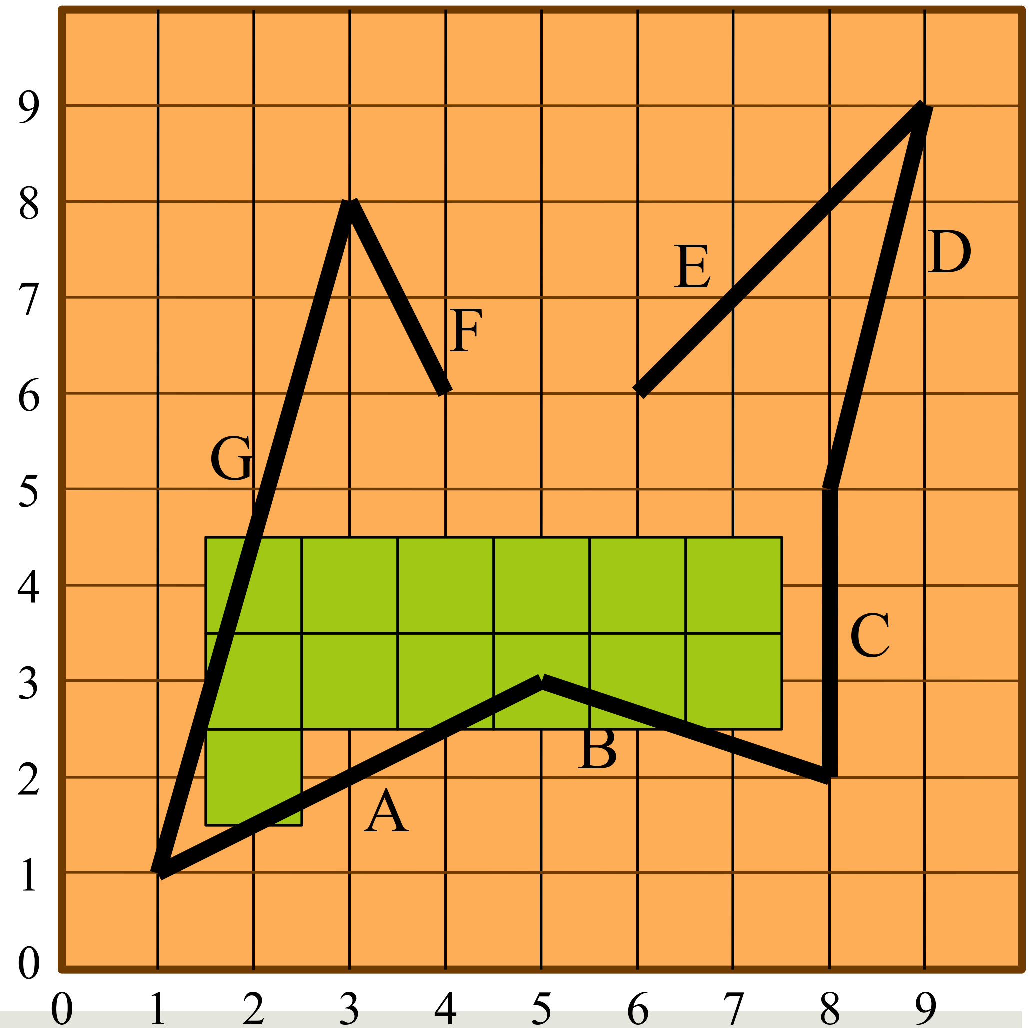


Polygon Rasterization

Edge	x	dx/dy	ymax
G	1 6/7	2/7	8
C	8	0/3	5

- y = 4
- Delete y = ymax edges
- Update x
- Add y = ymin edges
- For each pair x_0,x_1, plot from ceil(x_0) to ceil(x_1) – 1

Edge	ymin
A	1
G	1
B	2
C	2
D	5
E	6
F	6

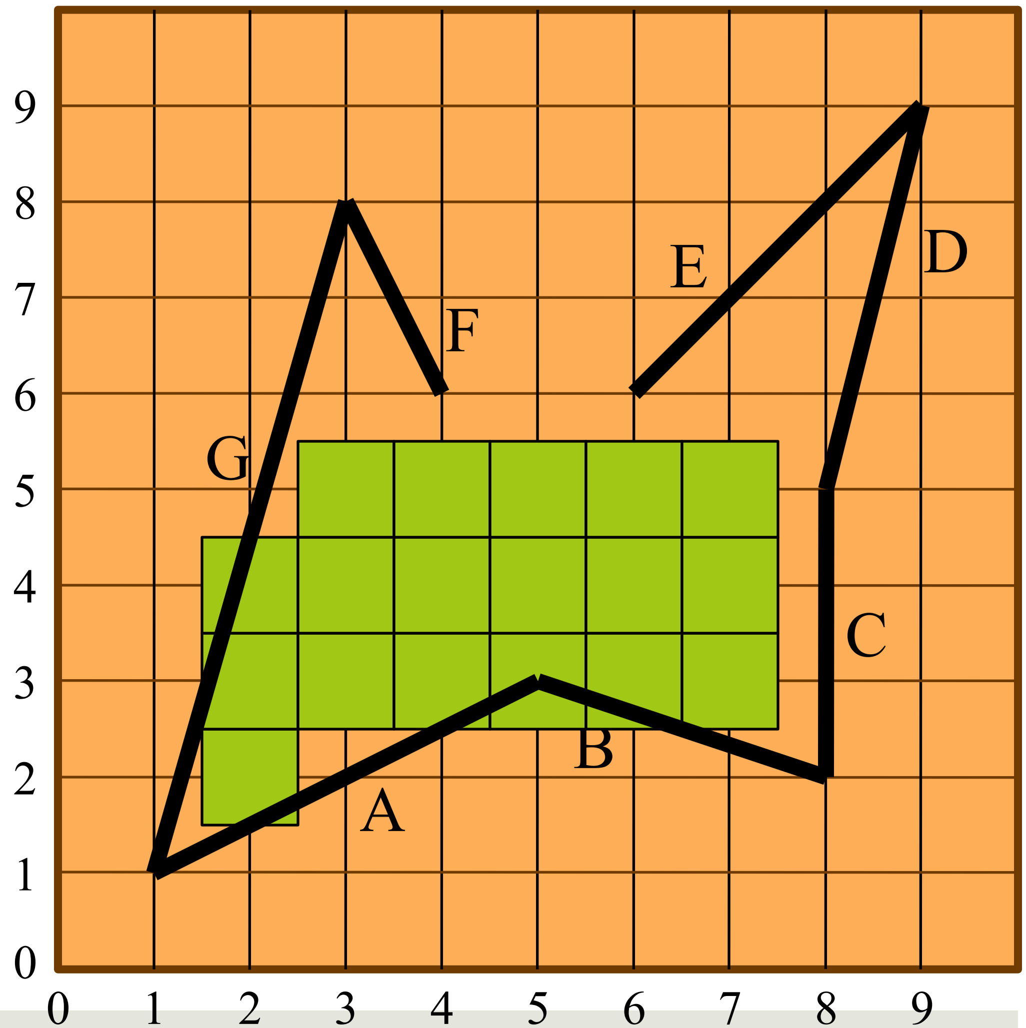


Polygon Rasterization

Edge	x	dx/dy	ymax
G	2 1/7	2/7	8
D	8	1/4	9

- y = 5
- Delete y = ymax edges
- Update x
- Add y = ymin edges
- For each pair x_0,x_1, plot from ceil(x_0) to ceil(x_1) – 1

Edge	ymin
A	1
G	1
B	2
C	2
D	5
E	6
F	6

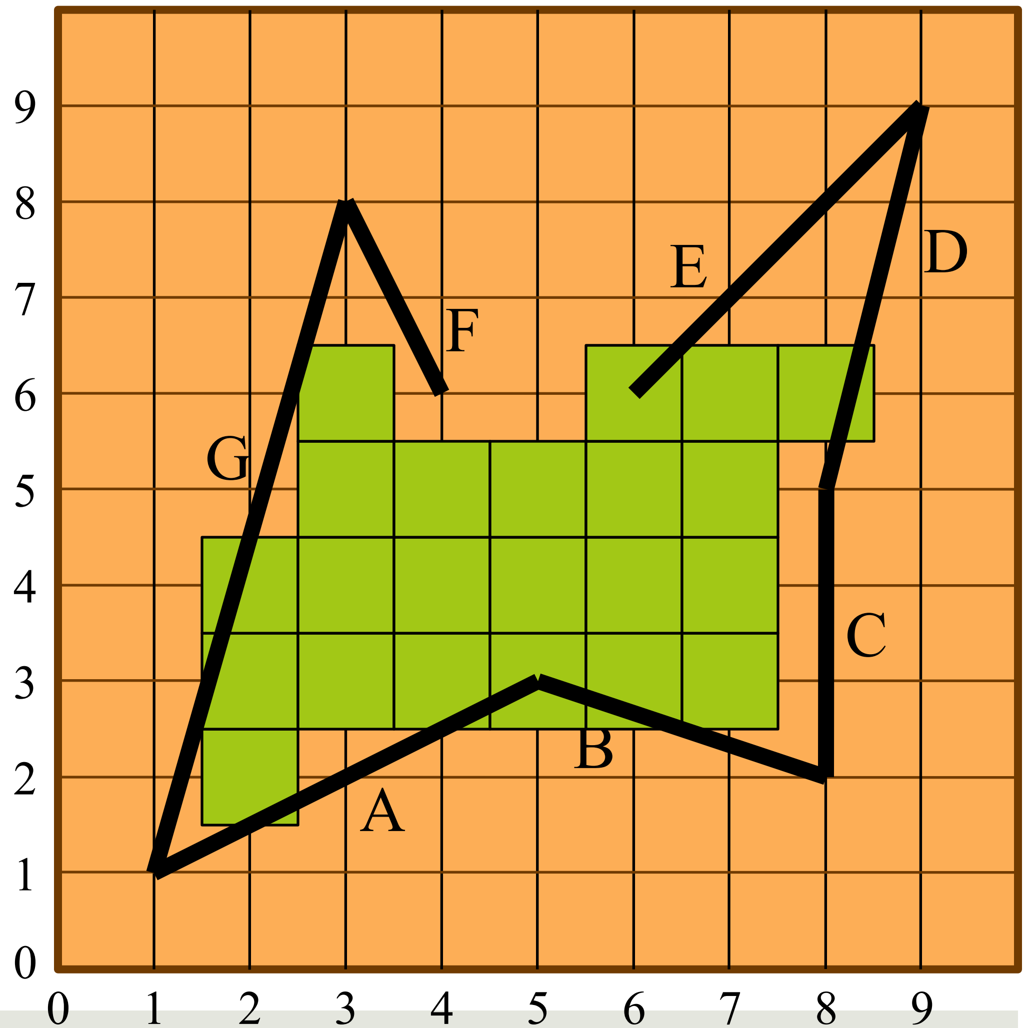


Polygon Rasterization

Edge	x	dx/dy	ymax
G	2 3/7	2/7	8
F	4	-1/2	8
E	6	1/1	9
D	8 1/4	1/4	9

- y = 6
- Delete y = ymax edges
- Update x
- Add y = ymin edges
- For each pair x_0,x_1, plot from ceil(x_0) to ceil(x_1) – 1

Edge	ymin
A	1
G	1
B	2
C	2
D	5
E	6
F	6

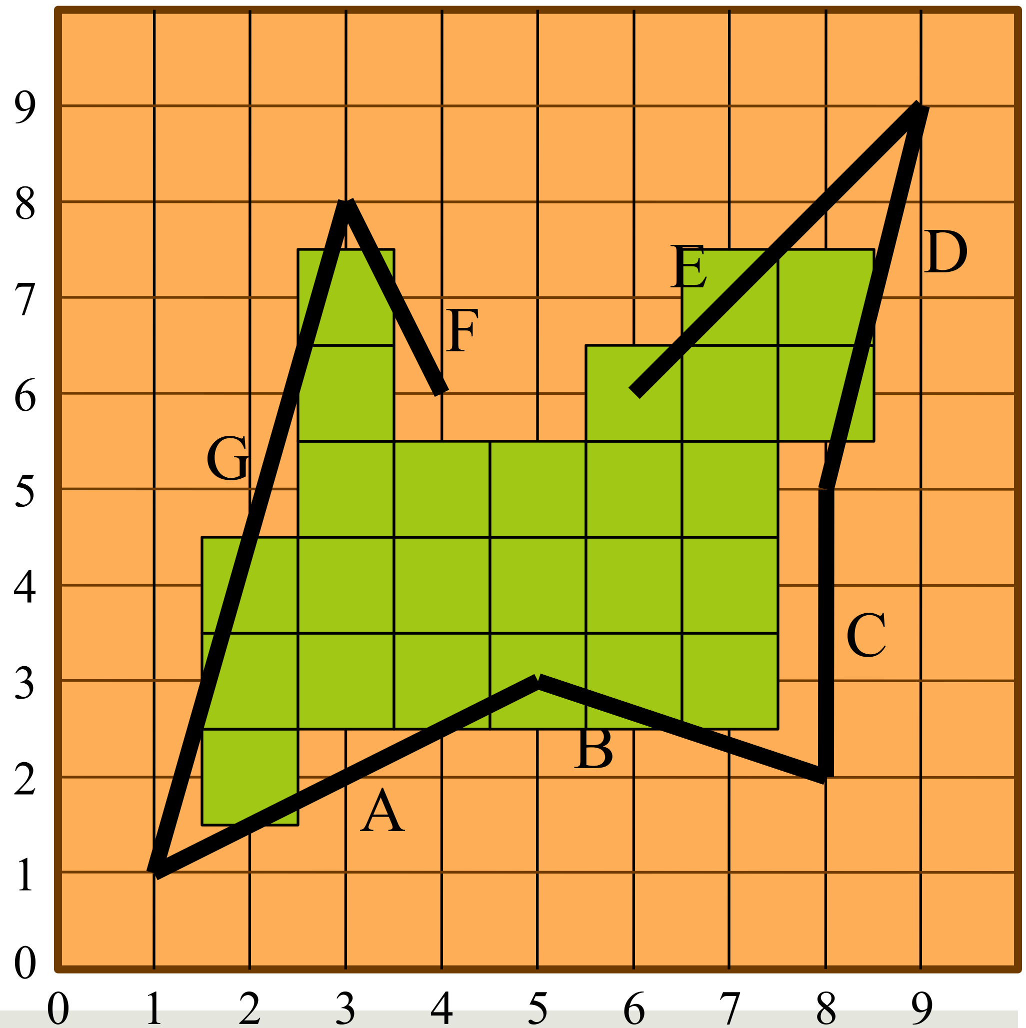


Polygon Rasterization

Edge	x	dx/dy	ymax
G	2 5/7	2/7	8
F	3 1/2	-1/2	8
E	7	1/1	9
D	8 2/4	1/4	9

- y = 7
- Delete y = ymax edges
- Update x
- Add y = ymin edges
- For each pair x_0,x_1, plot from ceil(x_0) to ceil(x_1) – 1

Edge	ymin
A	1
G	1
B	2
C	2
D	5
E	6
F	6

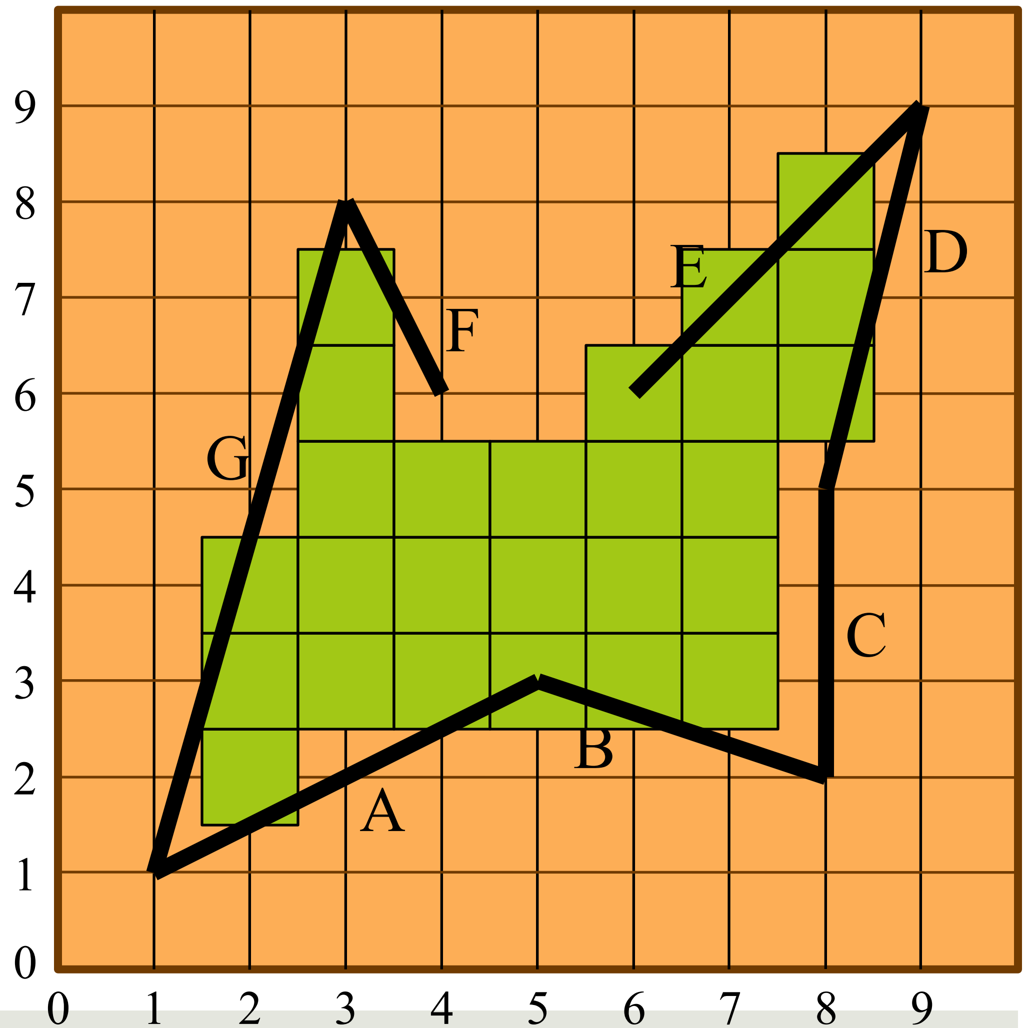


Polygon Rasterization

Edge	x	dx/dy	ymax
E	8	1/1	9
D	8 3/4	1/4	9

- y = 8
- Delete y = ymax edges
- Update x
- Add y = ymin edges
- For each pair x_0, x_1, plot from $\text{ceil}(x_0)$ to $\text{ceil}(x_1) - 1$

Edge	ymin
A	1
G	1
B	2
C	2
D	5
E	6
F	6

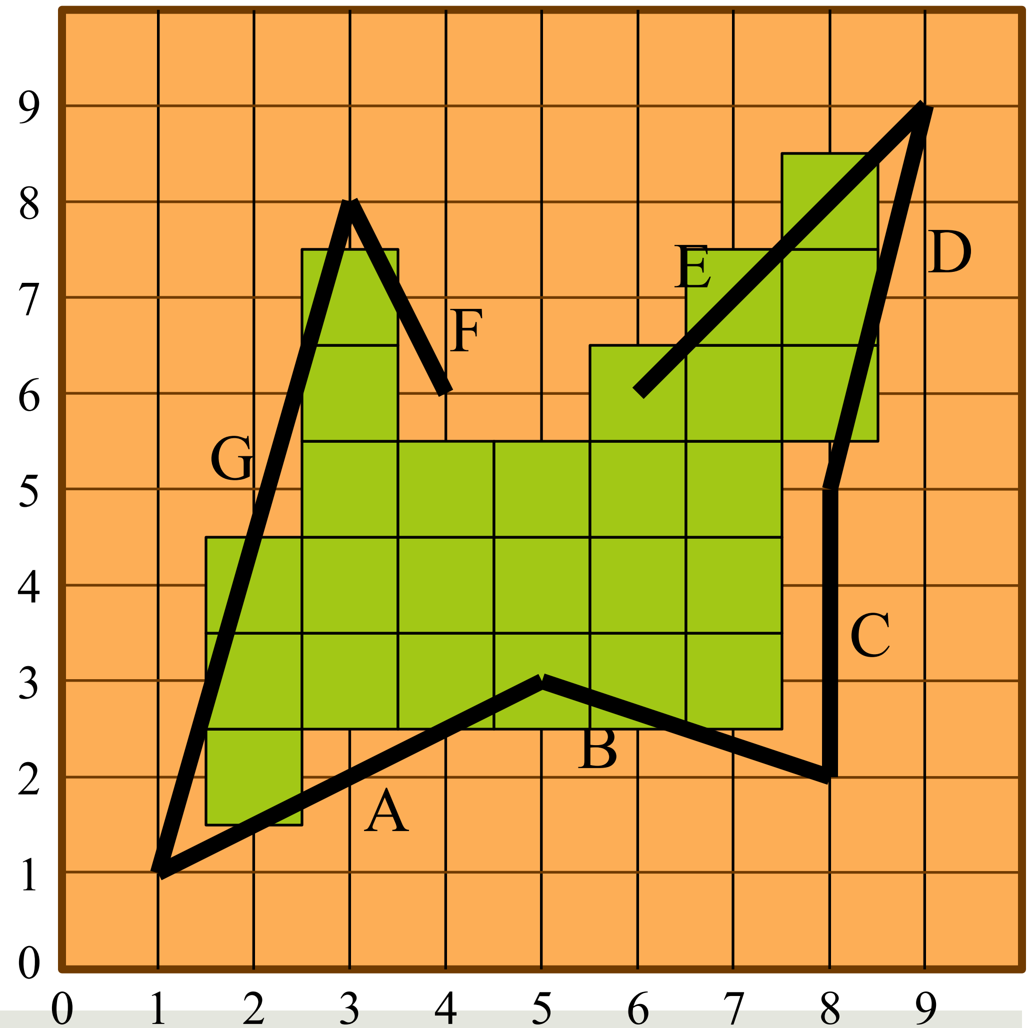


Polygon Rasterization

Edge	x	dx/dy	ymax

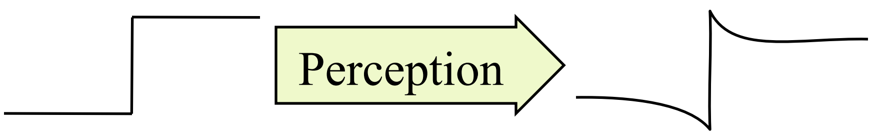
- $y = 9$
- Delete y = ymax edges
- Update x
- Add y = ymin edges
- For each pair x_0, x_1, plot from $\text{ceil}(x_0)$ to $\text{ceil}(x_1) - 1$

Edge	ymin
A	1
G	1
B	2
C	2
D	5
E	6
F	6



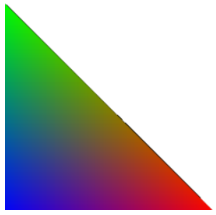
Gouraud Shading Revisited

- Flat shading
  - Per face normals
  - Color jumps across edge
  - Human visual perception accentuates edges



- Smooth shading
  - Per vertex normals
  - Colors similar across edge
  - Edges become harder to discern

Gouraud Shading Revisited



- Keep track of R, G, B at edge endpoints
- Compute dR/dy, dG/dy and dB/dy per edge
- Compute dR/dx, dG/dx and dB/dx at each scanline
- Color each pixel

  R += dR/dx

  G += dG/dx

  B += dB/dx