Rasterization

CMSC 161: Interactive Computer Graphics

2nd Semester 2014-2015

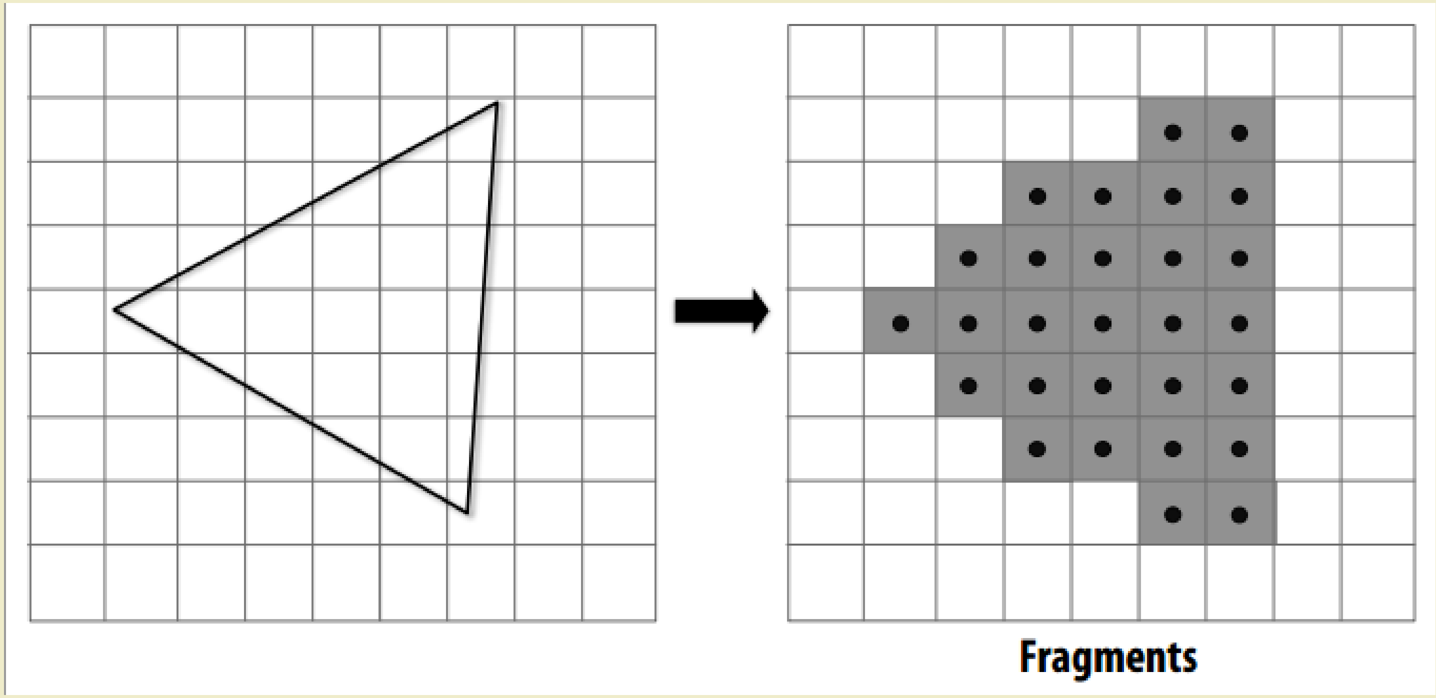
Institute of Computer Science

University of the Philippines – Los Baños

Lecture by James Carlo Plaras

Rasterization/Scan Conversion

Primitives are broken down into **fragments**



Rasterization/Scan Conversion

Clipping and culling of primitives are already processed

Rasterization/Scan Conversion

Treated as **independent process** from displaying the contents on the screen

One process for storing values at frame buffer

Another concurrent process for displaying contents at a required rate

Frame Buffer

Part of the memory where pixels are stored

Color Buffer, Depth Buffer, Stencil Buffer

Color Buffer

Storage of color information per pixel

Can be viewed as an 2D array of pixels

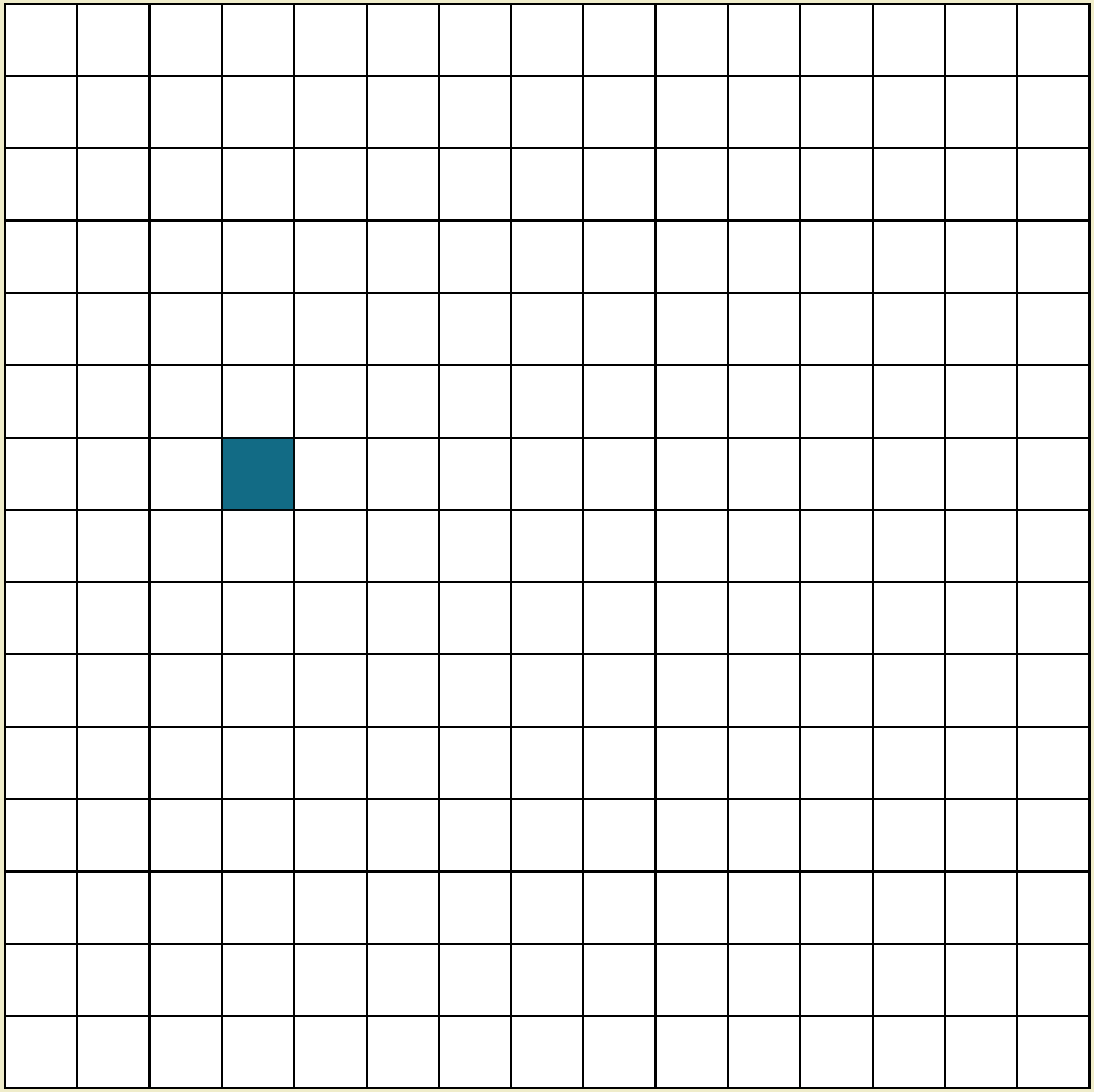
Color Buffer

Pixel color value can be set by specifying its corresponding indices

```
drawPixel(int x, int y, color_rgb value) {
        colorbuffer[x][y] = value
}
```

Color Buffer

It is inherently discrete, **only integer values for indices**



LINE DRAWING ALGORITHMS

Line Drawing

Must be as straight as possible

Pass through the end points

Smooth as possible

Naïve Line Drawing Algorithm

Simplest way

Based from slope-intercept equation

Assumption: Left to right order of end points

$$(x_1, y_1)\ and\ (x_2, y_2), where\ x_1 \leq x_2$$

Naïve Line Drawing Algorithm

```
Find slope-intercept equation y=mx+b
Let x = x1,
While x <= x2
     Let ytrue = m*x + b
     Let y = Round(ytrue)
     drawPixel(x,y,color)
     x = x + 1
End while
```

Naïve Line Drawing Algorithm Problems

Requires floating point numbers

Floating point error prone

Slower than integer operations

Naïve Line Drawing Algorithm Problems

Lines must be expressed in slope-intercept form

Naïve Line Drawing Algorithm Problems

At most 3 floating point operations per iteration

Digital Differential Analyzer Algorithm

DDA Algorithm

Improvement of the naïve version

Uses incremental calculations to reduce floating point calculations

Digital Differential Analyzer Algorithm

```
Let m = (y2-y1)/(x2-x1)
Let x = x1
Let y = y1
While x <= x2
    drawPixel(x, Round(y), color)
    y = y + m
    x = x + 1
End while
```

Example: DDA Algorithm

Line from (5,5) to (10,8)

$$m = (8-5)/(10-5) = 3/5 \quad 0.6$$

Iteration	X	Y	Round(Y)
0	5	5	5
1			
2			
3			
4			
5			

Example: DDA Algorithm

Line from (5,5) to (10,8)

$$m = (8-5)/(10-5) = 3/5 \quad 0.6$$

Iteration	X	Y	Round(Y)
0	5	5	5
1	6	5.6	6
2			
3			
4			
5			

Example: DDA Algorithm

Line from (5,5) to (10,8)

$$m = (8-5)/(10-5) = 3/5 \quad 0.6$$

Iteration	X	Y	Round(Y)
0	5	5	5
1	6	5.6	6
2	7	6.2	6
3			
4			
5			

Example: DDA Algorithm

Line from (5,5) to (10,8)

$$m = (8-5)/(10-5) = 3/5 \quad 0.6$$

Iteration	X	Y	Round(Y)
0	5	5	5
1	6	5.6	6
2	7	6.2	6
3	8	6.8	7
4			
5			

Example: DDA Algorithm

Line from (5,5) to (10,8)

$$m = (8-5)/(10-5) = 3/5 \quad 0.6$$

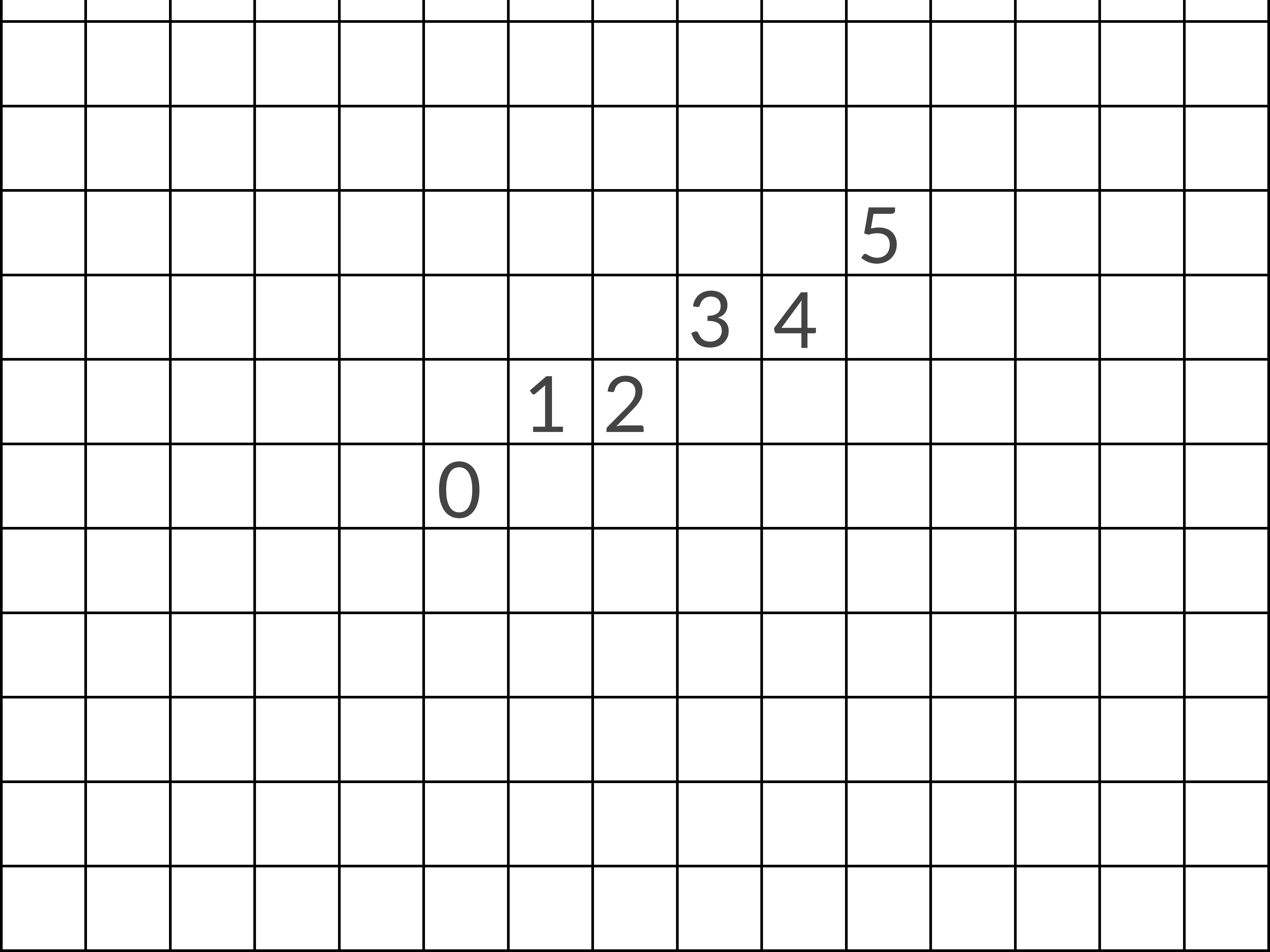
Iteration	X	Y	Round(Y)
0	5	5	5
1	6	5.6	6
2	7	6.2	6
3	8	6.8	7
4	9	7.4	7
5			

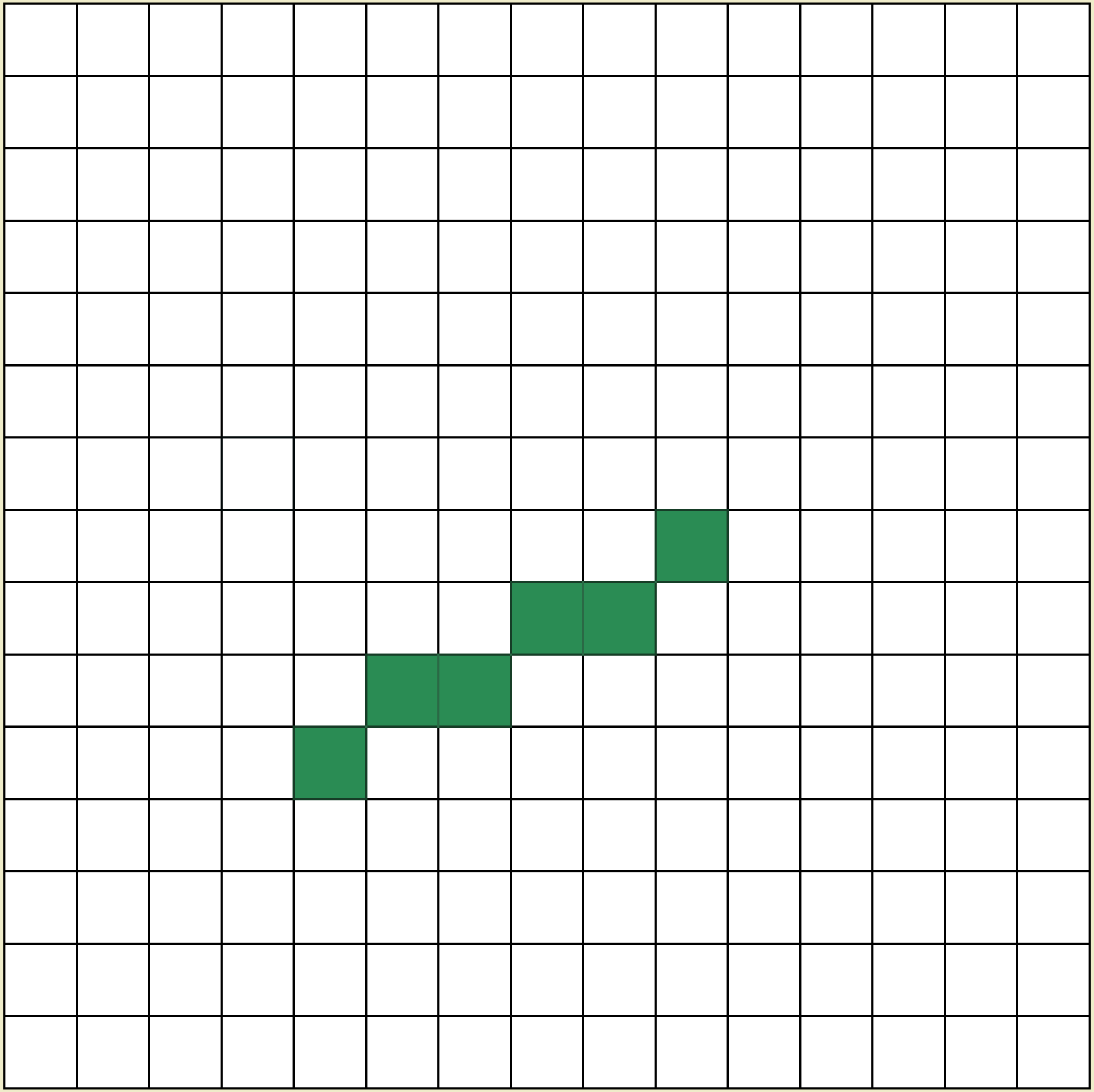
Example: DDA Algorithm

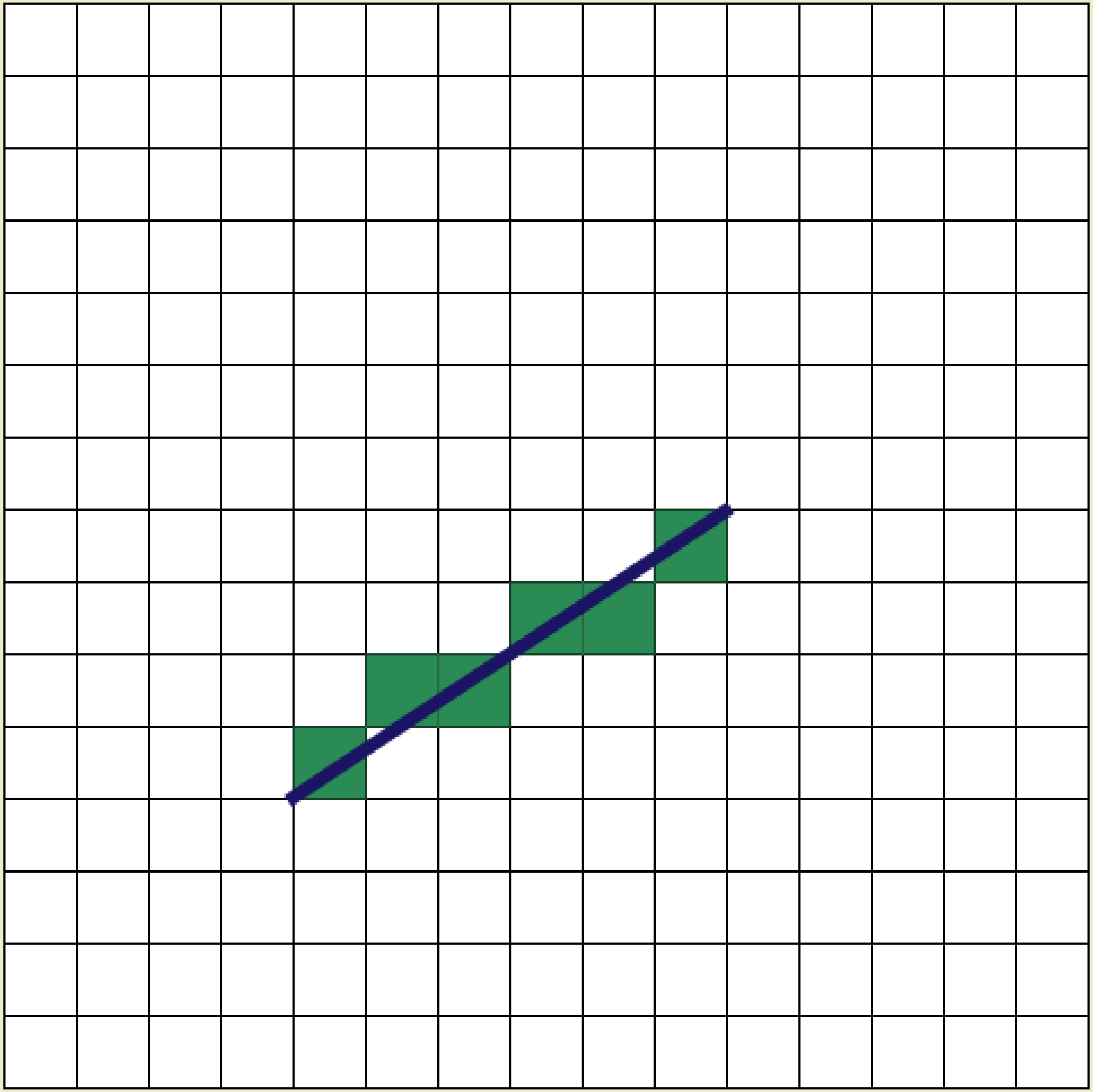
Line from (5,5) to (10,8)

$$m = (8-5)/(10-5) = 3/5 \quad 0.6$$

Iteration	X	Y	Round(Y)
0	5	5	5
1	6	5.6	6
2	7	6.2	6
3	8	6.8	7
4	9	7.4	7
5	10	8	8


5
3
4
1
2
0





Problems with DDA

Does not work well with slope greater than 1

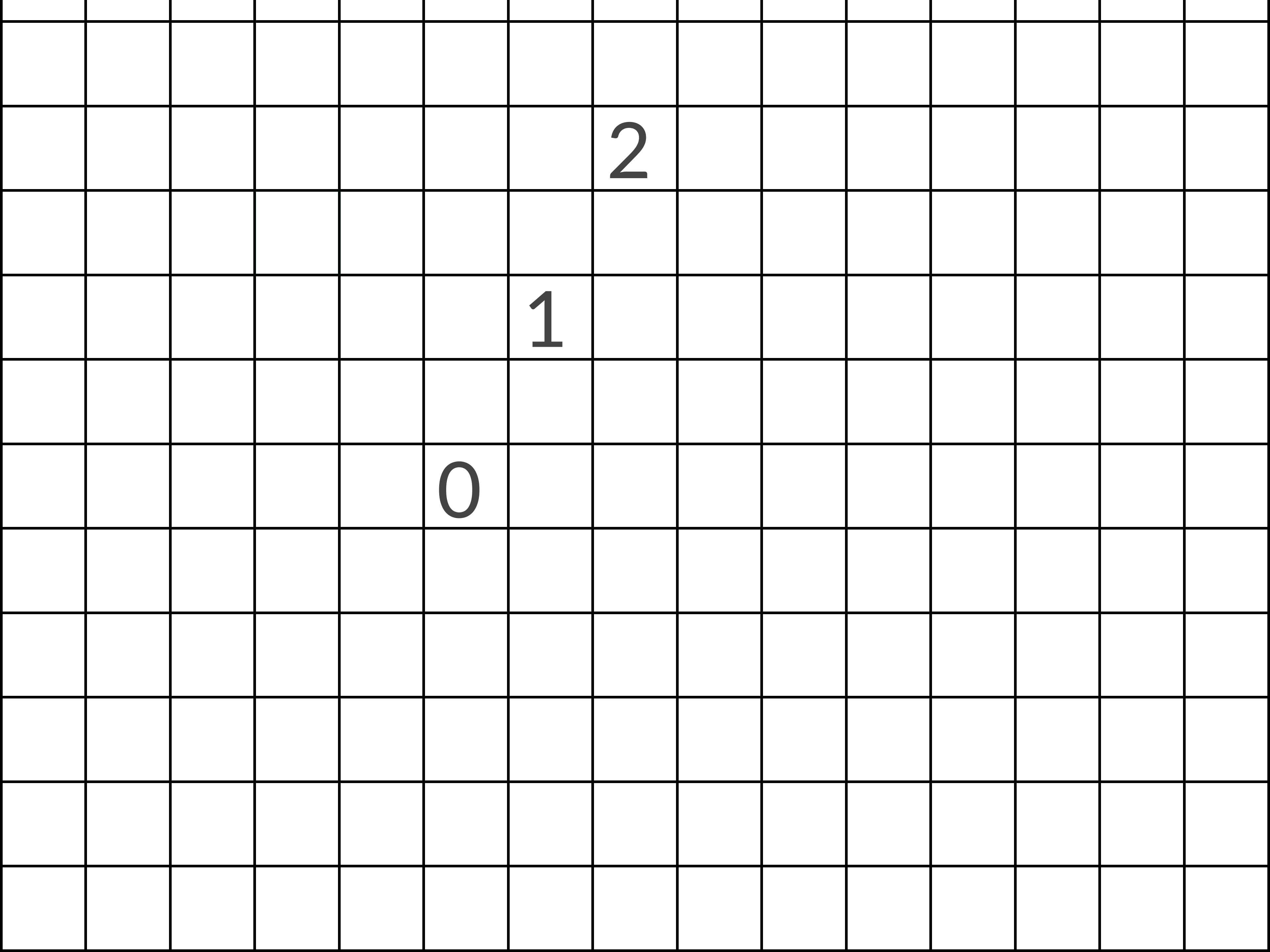
Still requires floating point numbers

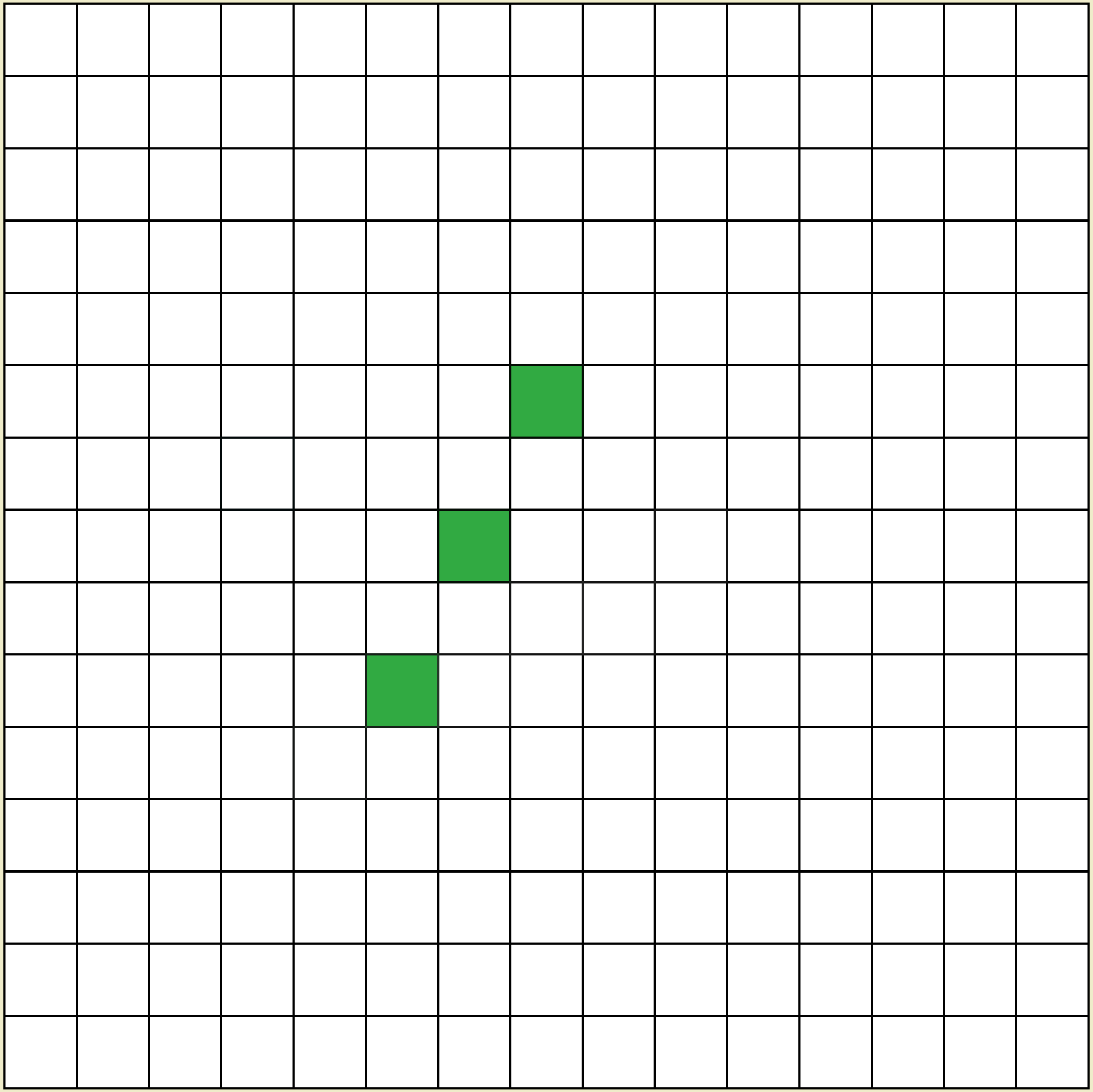
Example: DDA Algorithm

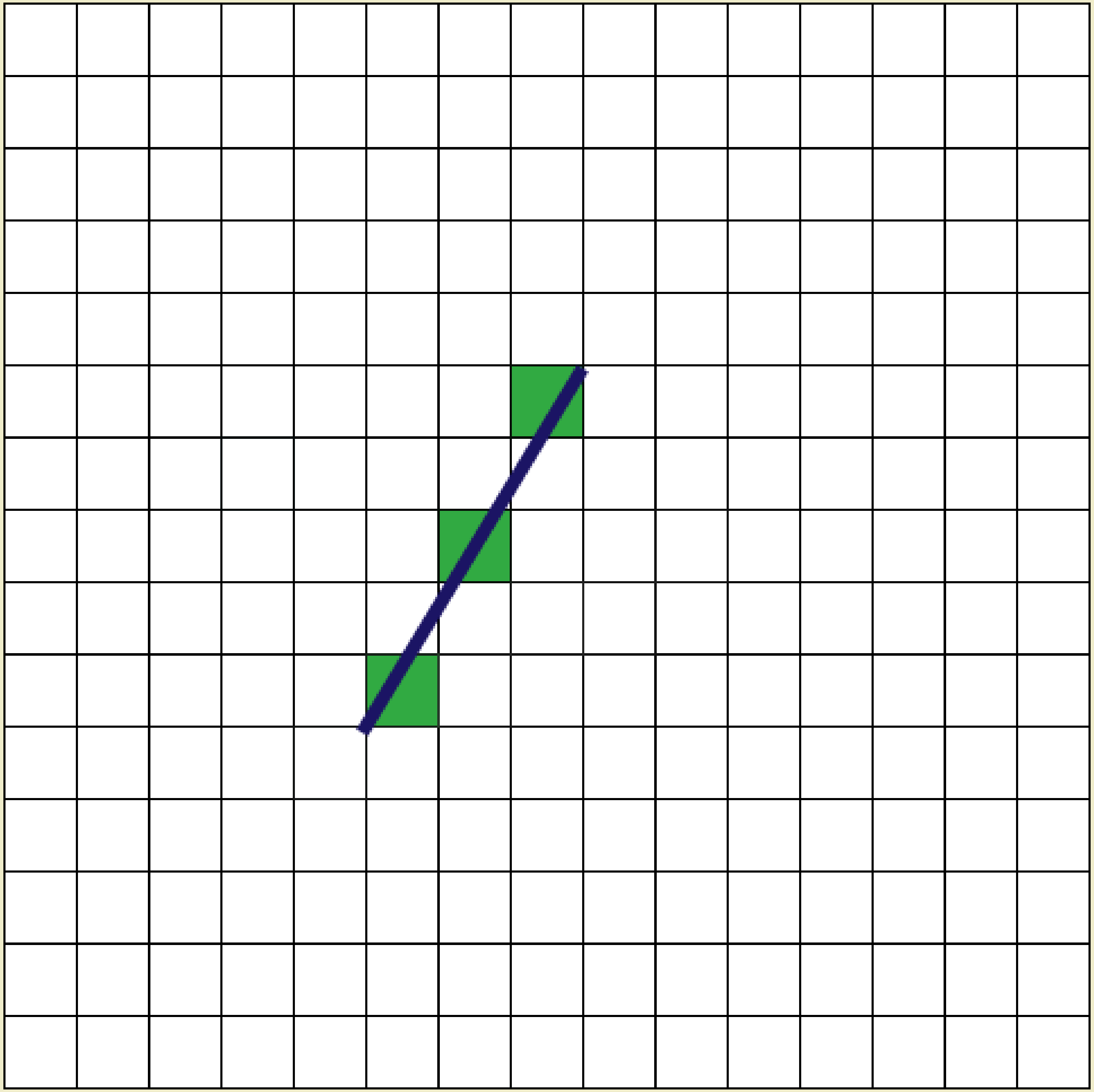
Line from (5,5) to (7,9)

$$m = (9 - 5)/(7 - 5) = 4/2 = 2$$

Iteration	X	Y	Round(Y)
0	5	5	5
1	6	7	7
2	7	9	9


2
1
0





Bresenham’s Line Algorithm

Jack Elton Bresenham, IBM, 1962

Bresenham’s Line Algorithm

Improvement of the DDA Algorithm

Avoids all floating point computations

Bresenham's Line Algorithm

Initial assumptions:

Left to right order of end points

$$0 \leq m \leq 1$$

Example: DDA Algorithm

Line from (5,5) to (10,8)

$m = (8-5)/(10-5) = 3/5 \quad 0.6$

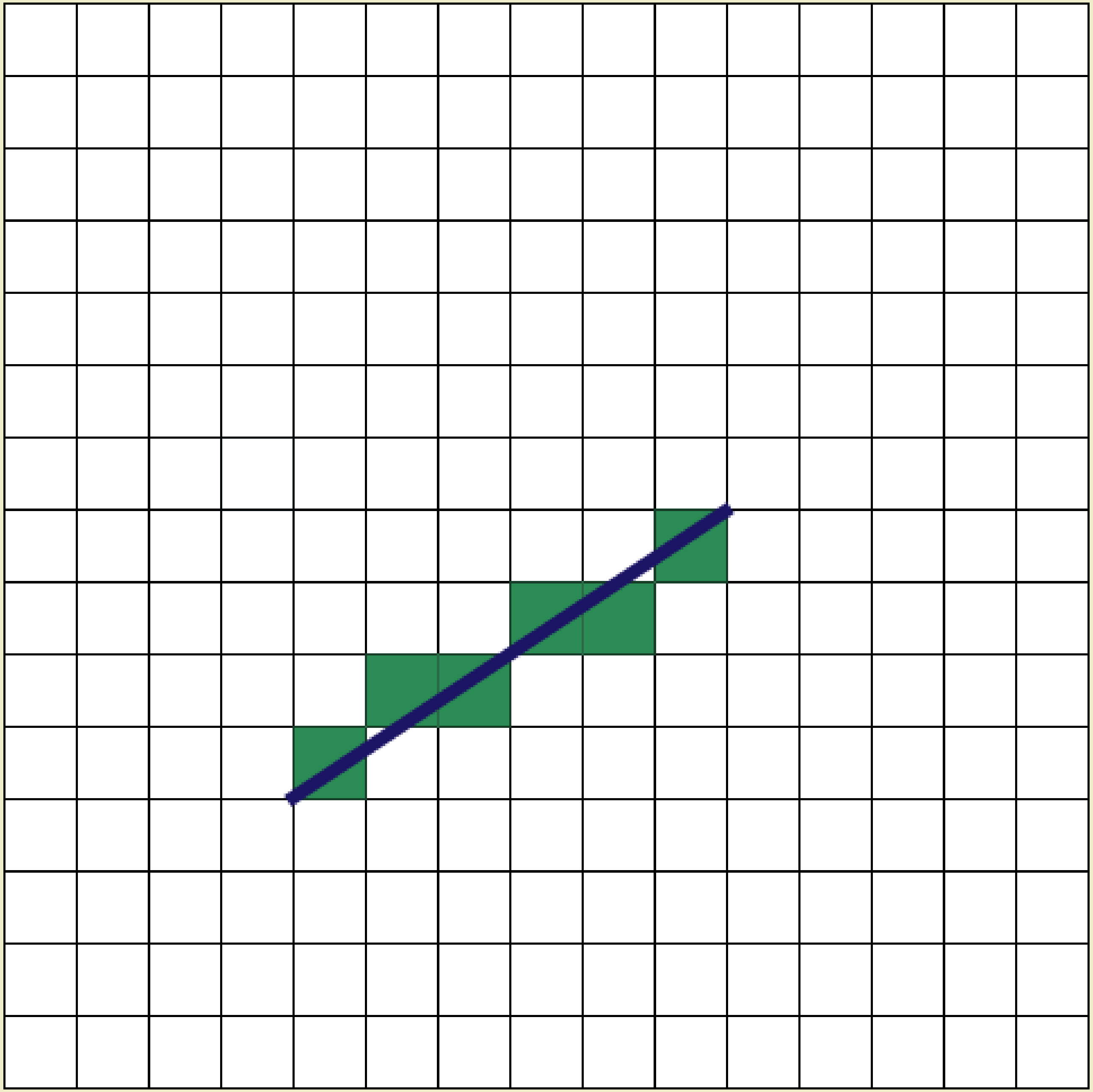
Iteration	X	Y	Σm	Round(Y)
0	5	5	0	5
1	6	5.6	0.6	6
2	7	6.2	1.2	6
3	8	6.8	1.8	7
4	9	7.4	2.4	7
5	10	8	3.0	8

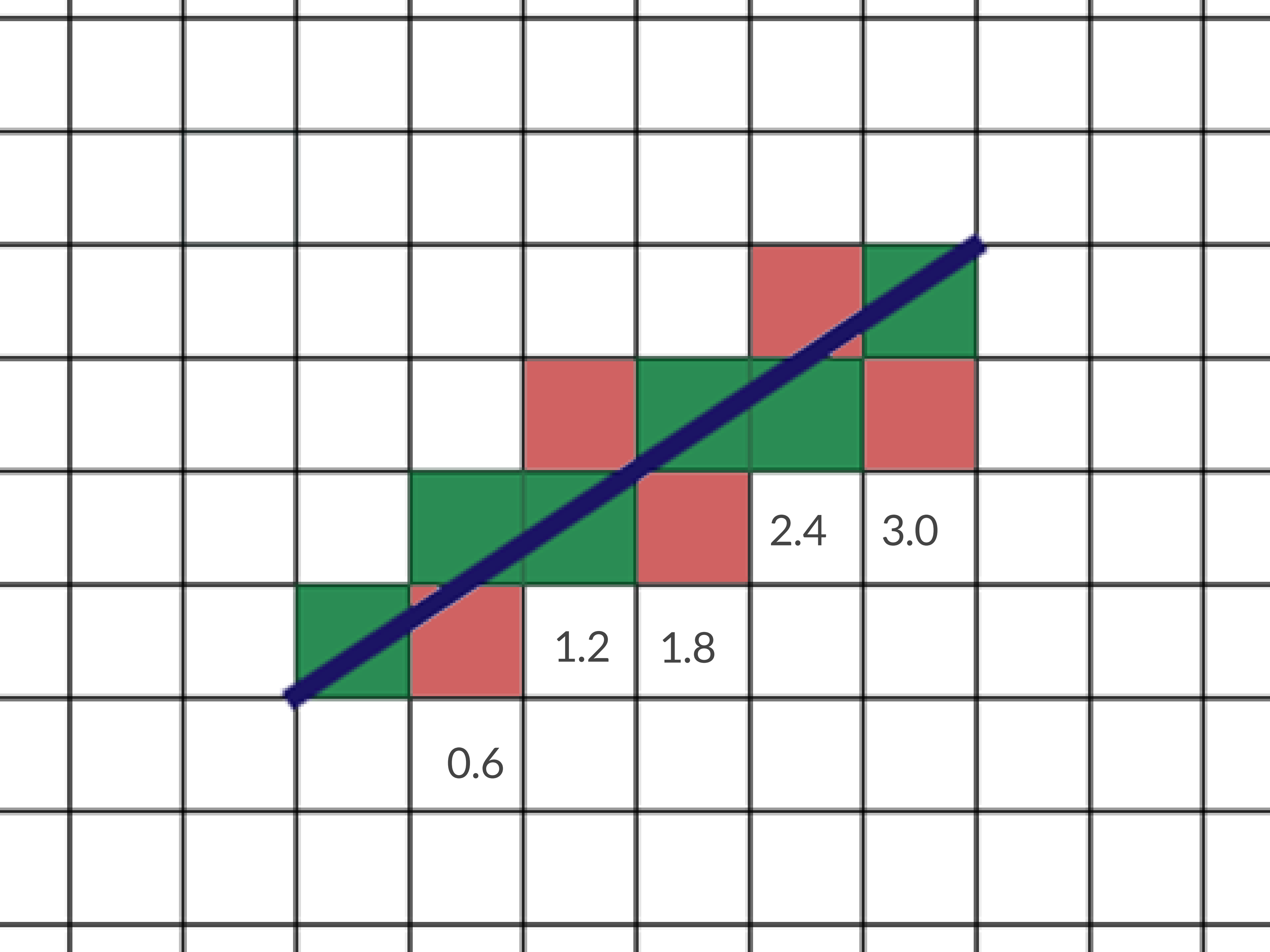
Example: DDA Algorithm

Line from (5,5) to (10,8)

$m = (8-5)/(10-5) = 3/5 \cong 0.6$

Iteration	X	Y	Σm	Round(Y)
0	5	5	0	5
1	6	5.6	**0.6 > 0.5**	**6**
2	7	6.2	1.2	6
3	8	6.8	**1.8 > 1.5**	**7**
4	9	7.4	2.4	7
5	10	8	3.0 > 2.5	8




2.4
3.0
1.2
1.8
0.6

Bresenham Algorithm

```
Let dx = x2-x1, dy=y2-y1, m = dy/dx
Let x = x1
Let y = y1
Let e = 0          //accumulated slope
Let t = 0.5        //threshold
While x <= x2
      drawPixel(x, y, color)
      e = e + abs(m)    //update accumulated slope
      if e >= t         //if greater than threshold
            y = y + 1   //select pixel above
            t = t + 1   //update new threshold
      end if
End while
```

Example: Bresenham Algorithm

Line from (5,5) to (10,8)

$$m = \frac{dy}{dx} = \frac{8-5}{10-5} = \frac{3}{5} \cong \mathbf{0.6}$$

Iteration	X	e	t	Y
0	5	0	0.5	
1	6			
2	7			
3	8			
4	9			
5	10			

Example: Bresenham Algorithm

Line from (5,5) to (10,8)

$$m = \frac{\boldsymbol{dy}}{\boldsymbol{dx}} = \frac{8-5}{10-5} = \frac{3}{5} \cong \mathbf{0.6}$$

Iteration	X	e	t	Y
0	5	0	0.5	**5**
1	6			
2	7			
3	8			
4	9			
5	10			

Example: Bresenham Algorithm

Line from (5,5) to (10,8)

$$m = \frac{dy}{dx} = \frac{8-5}{10-5} = \frac{3}{5} \cong \mathbf{0.6}$$

Iteration	X	e	t	Y
0	5	0	0.5	5
1	6	0.6	0.5	
2	7			
3	8			
4	9			
5	10			

Example: Bresenham Algorithm

Line from (5,5) to (10,8)

$$m = \frac{\boldsymbol{dy}}{\boldsymbol{dx}} = \frac{8-5}{10-5} = \frac{3}{5} \cong \mathbf{0.6}$$

Iteration	X	e	t	Y
0	5	0	0.5	**5**
1	6	0.6	0.5	**6**
2	7	1.2	1.5	
3	8			
4	9			
5	10			

Example: Bresenham Algorithm

Line from (5,5) to (10,8)

$$m = \frac{\boldsymbol{dy}}{\boldsymbol{dx}} = \frac{8-5}{10-5} = \frac{3}{5} \cong \mathbf{0.6}$$

Iteration	X	e	t	Y
0	5	0	0.5	**5**
1	6	0.6	0.5	**6**
2	7	1.2	1.5	**6**
3	8	1.8	1.5	
4	9			
5	10			

Example: Bresenham Algorithm

Line from (5,5) to (10,8)

$$m = \frac{\boldsymbol{dy}}{\boldsymbol{dx}} = \frac{8-5}{10-5} = \frac{3}{5} \cong \mathbf{0.6}$$

Iteration	X	e	t	Y
0	5	0	0.5	**5**
1	6	0.6	0.5	**6**
2	7	1.2	1.5	**6**
3	8	1.8	1.5	**7**
4	9	2.4	2.5	
5	10			

Example: Bresenham Algorithm

Line from (5,5) to (10,8)

$$m = \frac{dy}{dx} = \frac{8-5}{10-5} = \frac{3}{5} \cong \mathbf{0.6}$$

Iteration	X	e	t	Y
0	5	0	0.5	5
1	6	0.6	0.5	6
2	7	1.2	1.5	6
3	8	1.8	1.5	7
4	9	2.4	2.5	7
5	10	3.0	2.5	

Example: Bresenham Algorithm

Line from (5,5) to (10,8)

$$m = \frac{\boldsymbol{dy}}{\boldsymbol{dx}} = \frac{8-5}{10-5} = \frac{3}{5} \cong \mathbf{0.6}$$

Iteration	X	e	t	Y
0	5	0	0.5	**5**
1	6	0.6	0.5	**6**
2	7	1.2	1.5	**6**
3	8	1.8	1.5	**7**
4	9	2.4	2.5	**7**
5	10	3.0	2.5	**8**

Bresenham Floating Point Removal

Floating point operations are very slow

Change Bresenham computations to handle non-floating point values

Bresenham Algorithm: Remove Float

```
Let dx = x2-x1, dy=y2-y1
Let m = abs(dy/dx * ( 2 * dx))
Let x = x1
Let y = y1
Let e = 0
Let t = 0.5 * ( 2 * dx)
While x <= x2
      drawPixel(x, y, color)
      e = e + m
      if e >= t
            y = y + 1
            t = t + (1 * ( 2 * dx))
      end if
End while
```

Bresenham Algorithm Floating Point Removal

Line from (5,5) to (10,8)

$$m = \frac{\boldsymbol{dy}}{\boldsymbol{dx}} \times 2 = \frac{8-5}{10-5} \times 2 = \frac{3}{5} \times 2 \cong \mathbf{0.6} \times \mathbf{2} = \mathbf{1.2}$$

Iteration	X	e	t	2t	Y
0	5	0	0.5	1	5
1	6	1.2	0.5	1	6
2	7	2.4	1.5	3	6
3	8	3.6	1.5	3	7
4	9	4.8	2.5	5	7
5	10	6.0	2.5	5	8

Bresenham Algorithm Floating Point Removal

Line from (5,5) to (10,8)

$$dx = 5 \; dy = 3$$

$$m = \boldsymbol{dx} \times \left(\frac{dy}{dx}\right) \times 2 = dy \times 2 = 6$$

Iteration	X	e	2t	2t * dx	Y
0	5	0	1	5	5
1	6	6	1	5	6
2	7	12	3	15	6
3	8	18	3	15	7
4	9	24	5	25	7
5	10	30	5	25	8

Bresenham Algorithm: Remove Float

```
Let dx = x2-x1, dy=y2-y1
Let m = 2 * abs(dy)
Let x = x1
Let y = y1
Let e = 0
Let t = dx
Let t_inc = 2 * dx
While x <= x2
      drawPixel(x, y, color)
      e = e + m
      if e >= t
            y = y + 1
            t = t + t_inc
      end if
End while
```

Bresenham Algorithm Floating Point Removal

Line from (5,5) to (10,8)

$$dx = 5$$
$$dy = 3$$
$$m = dy \times 2 = 6$$
$$t_{inc} = 2 \times dx$$

Iteration	X	e	t	Y
0	5	0	5	
1	6			
2	7			
3	8			
4	9			
5	10			

Bresenham Algorithm Floating Point Removal

Line from (5,5) to (10,8)

$$dx \ = \ 5$$

$$dy \ = \ 3$$

$$m = \ dy \times \ 2 \ = \ 6$$

$$t_{inc} = 2 \times dx$$

Iteration	X	e	t	Y
0	5	0	5	**5**
1	6	6	5	
2	7			
3	8			
4	9			
5	10			

Bresenham Algorithm Floating Point Removal

Line from (5,5) to (10,8)

$$dx = 5$$
$$dy = 3$$
$$m = dy \times 2 = 6$$
$$t_{inc} = 2 \times dx$$

Iteration	X	e	t	Y
0	5	0	5	**5**
1	6	6	5	**6**
2	7	12	15	
3	8			
4	9			
5	10			

Bresenham Algorithm Floating Point Removal

Line from (5,5) to (10,8)

$$dx = 5$$
$$dy = 3$$
$$m = dy \times 2 = 6$$
$$t_{inc} = 2 \times dx$$

Iteration	X	e	t	Y
0	5	0	5	**5**
1	6	6	5	**6**
2	7	12	15	**6**
3	8	18	15	
4	9			
5	10			

Bresenham Algorithm Floating Point Removal

Line from (5,5) to (10,8)

$$dx = 5$$
$$dy = 3$$
$$m = dy \times 2 = 6$$
$$t_{inc} = 2 \times dx$$

Iteration	X	e	t	Y
0	5	0	5	**5**
1	6	6	5	**6**
2	7	12	15	**6**
3	8	18	15	**7**
4	9	24	25	
5	10			

Bresenham Algorithm Floating Point Removal

Line from (5,5) to (10,8)

$$dx = 5$$

$$dy = 3$$

$$m = dy \times 2 = 6$$

$$t_{inc} = 2 \times dx$$

Iteration	X	e	t	Y
0	5	0	5	**5**
1	6	6	5	**6**
2	7	12	15	**6**
3	8	18	15	**7**
4	9	24	25	**7**
5	10	30	25	

Bresenham Algorithm Floating Point Removal

Line from (5,5) to (10,8)

$$dx = 5$$

$$dy = 3$$

$$m = dy \times 2 = 6$$

$$t_{inc} = 2 \times dx$$

Iteration	X	e	t	Y
0	5	0	5	**5**
1	6	6	5	**6**
2	7	12	15	**6**
3	8	18	15	**7**
4	9	24	25	**7**
5	10	30	25	**8**

General Bresenham's Line Algorithm

Utilize the concept of symmetry

– $m > 1$ (swap roles of x and y)

– $m < 0$ (decrease y instead of increasing it)

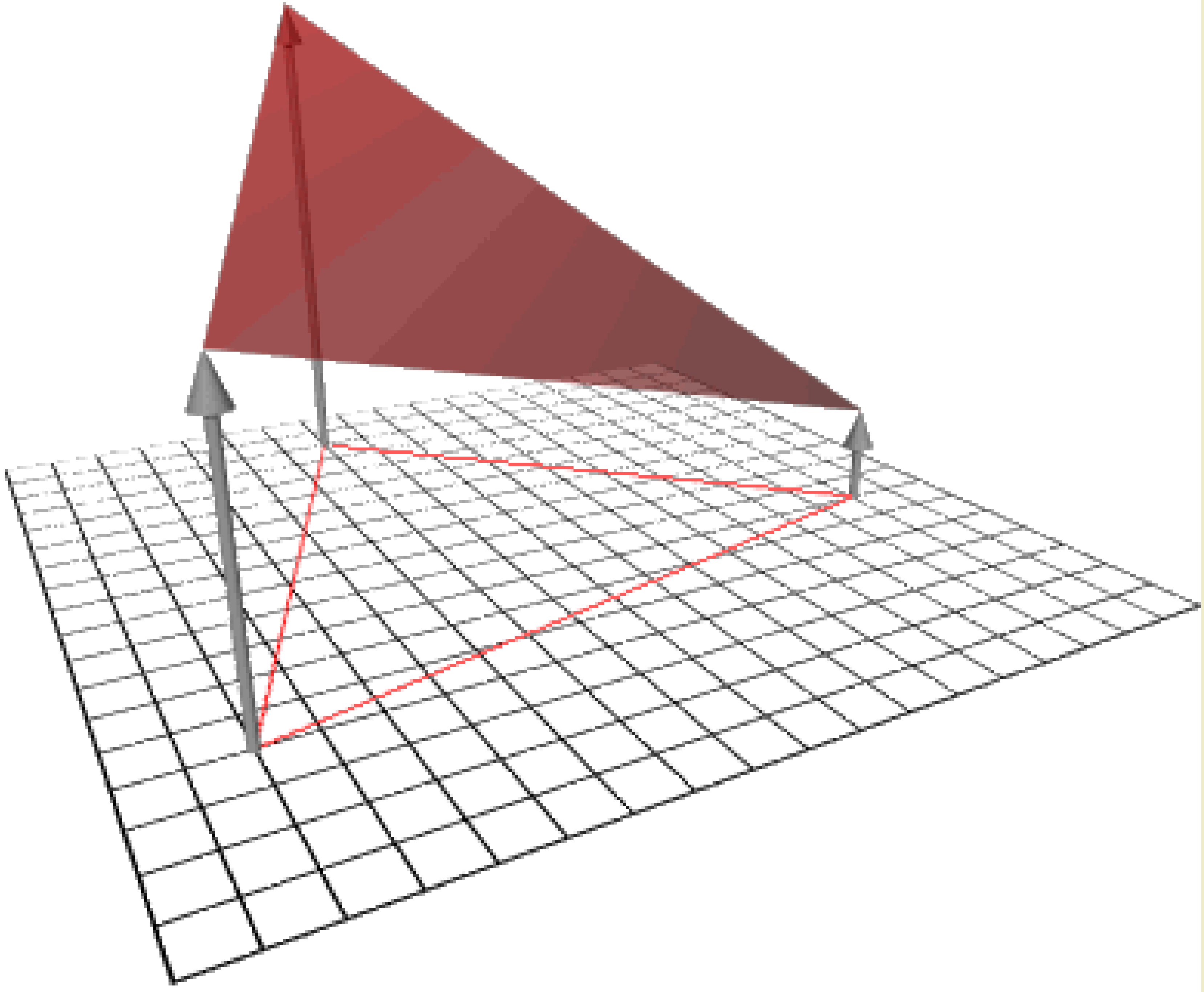
General Bresenham's Line Algorithm

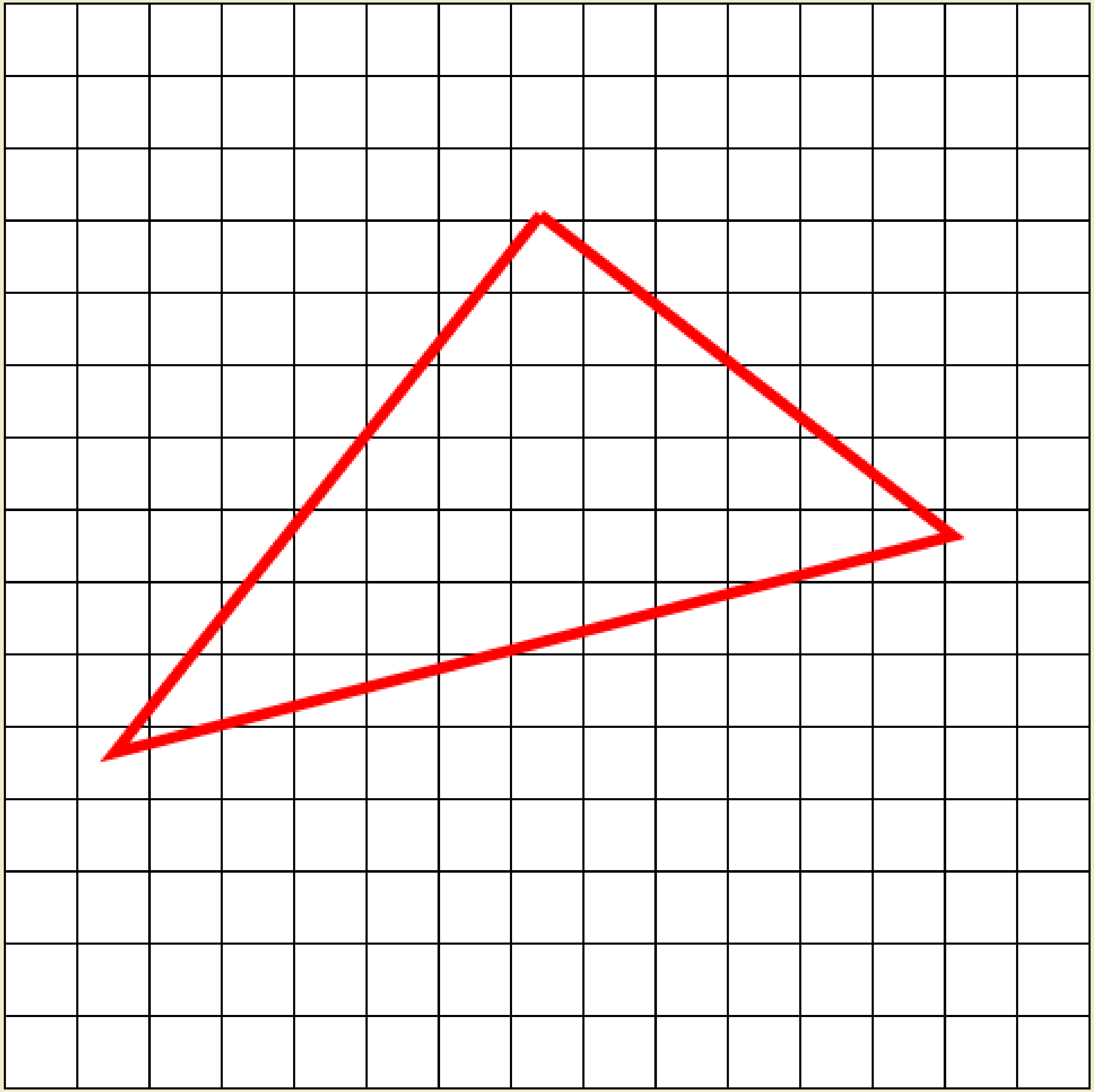
```
… //initializations
Check if steep: m > 1
         swap(x1,y1),swap(x2,y2)
Check if negative-slope: m < 0
While x <= x2
         if steep
                  drawPixel(y, x, color)
         else
                  drawPixel(x, y, color)
         End if

         e = e + m
         if e >= t
                  if negative-slope:
                           y = y - 1
                  else
                           y = y + 1
                  end if

                  t = t + t_inc
         end if
End while
```

POLYGON RASTERIZATION

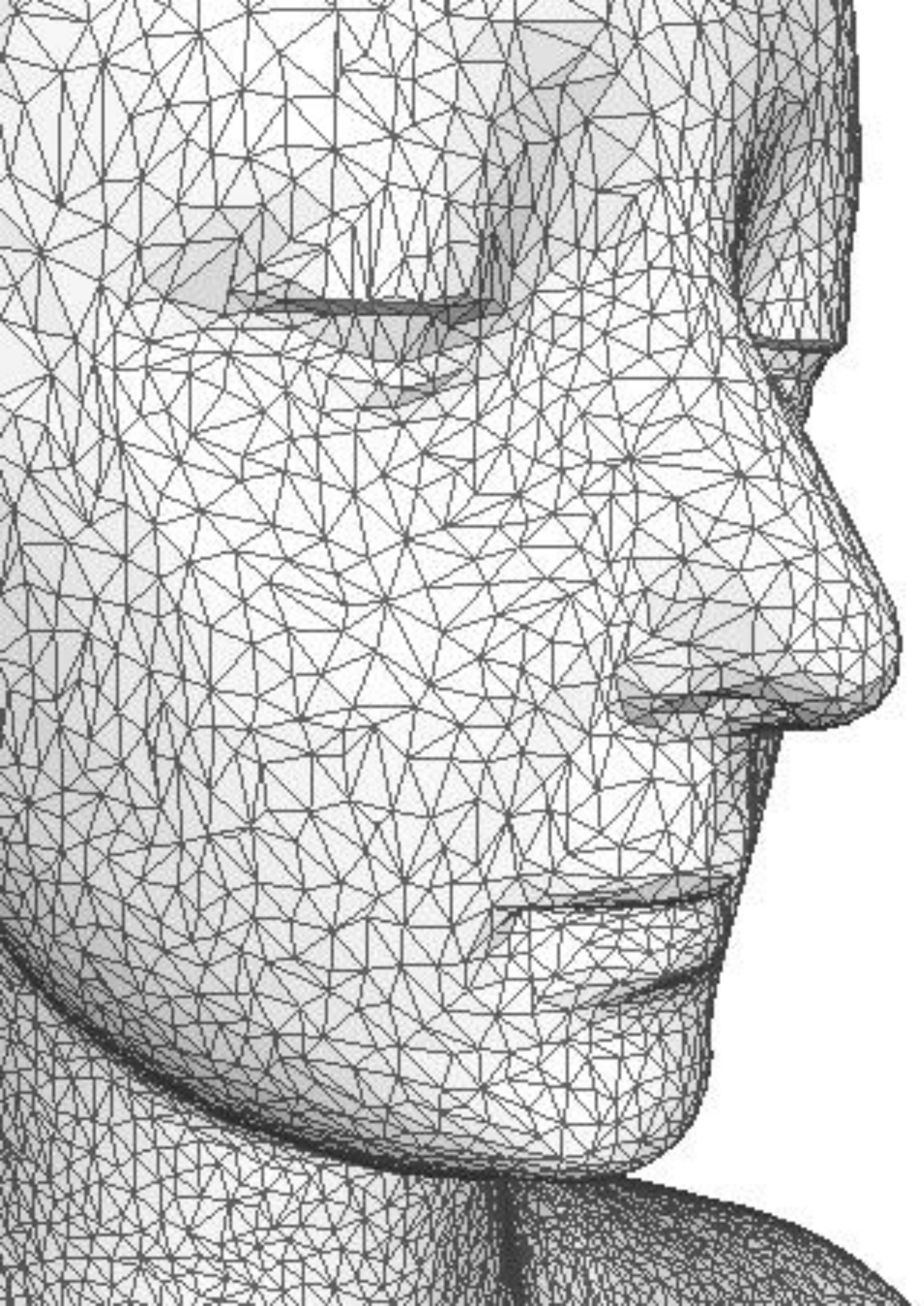




Polygon Rasterization

Complex polygons can be broken down into triangles

Enforced by WebGL!



Triangles

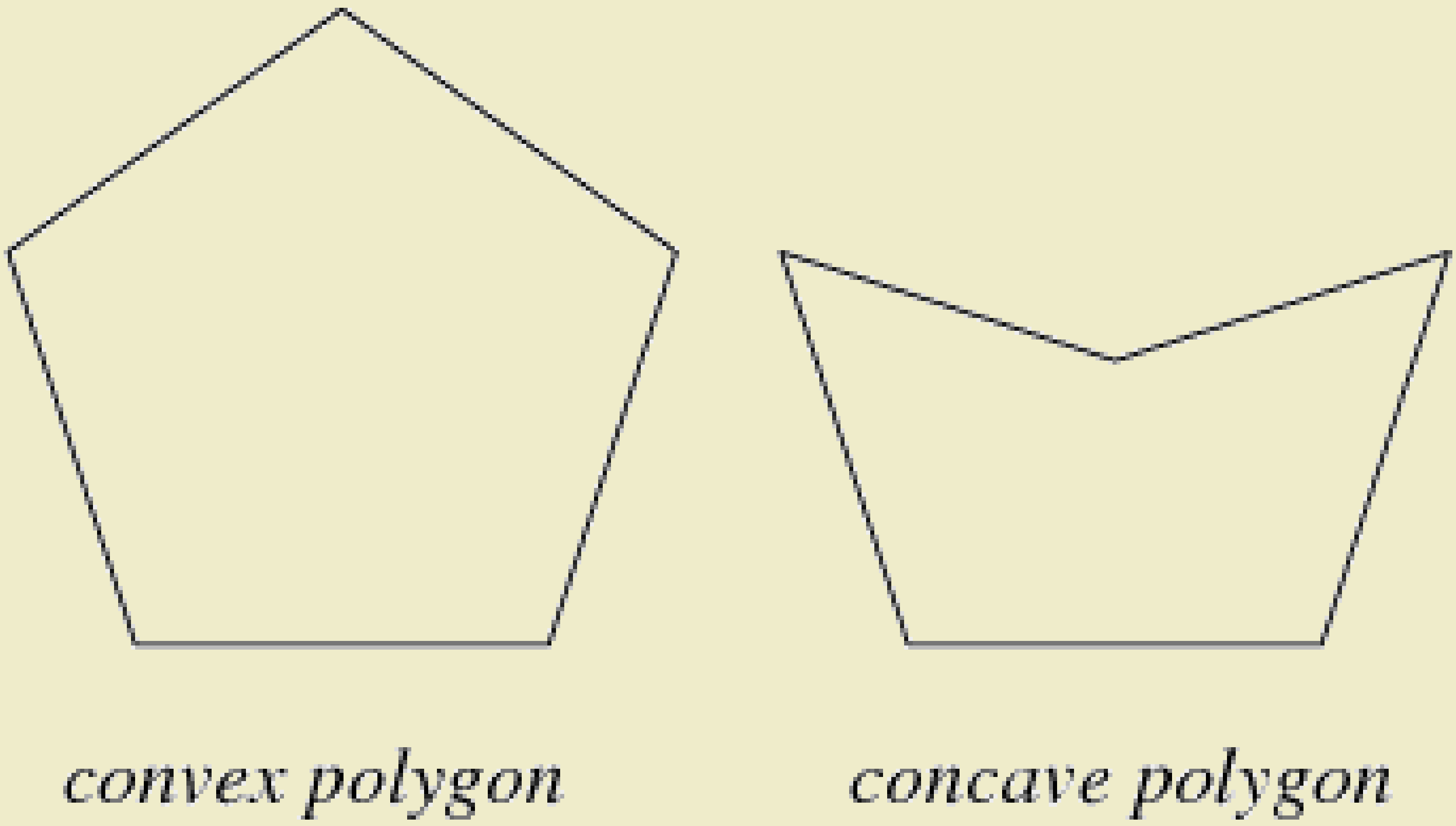
Polygon of minimum number of vertices

Can approximate any 2D shape or 3D surface

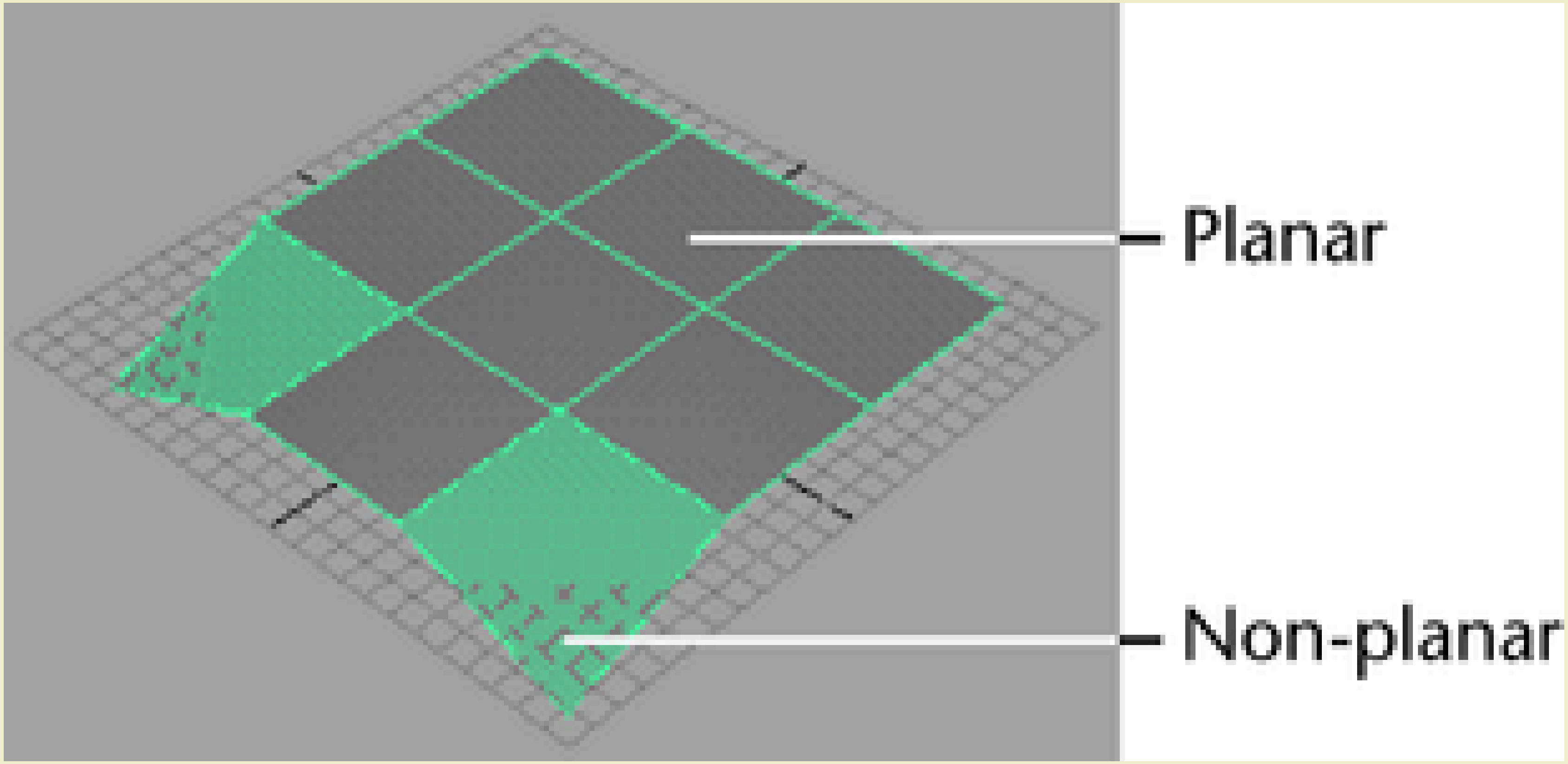
Triangles are convex:Easier to rasterize

Triangles are planar:Easier to rasterize

Convex and Concave



Planar and Non-Planar Polygon



Methods for Rasterizing Triangles

Edge Walking

Edge Equation

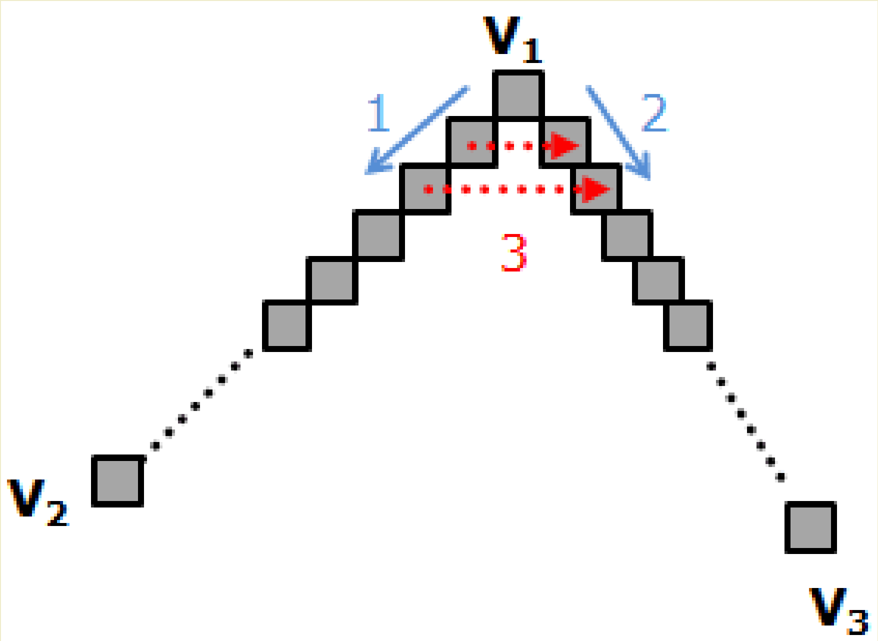
Barycentric Coordinates

Recursive Subdivision

EDGE WALKING METHOD

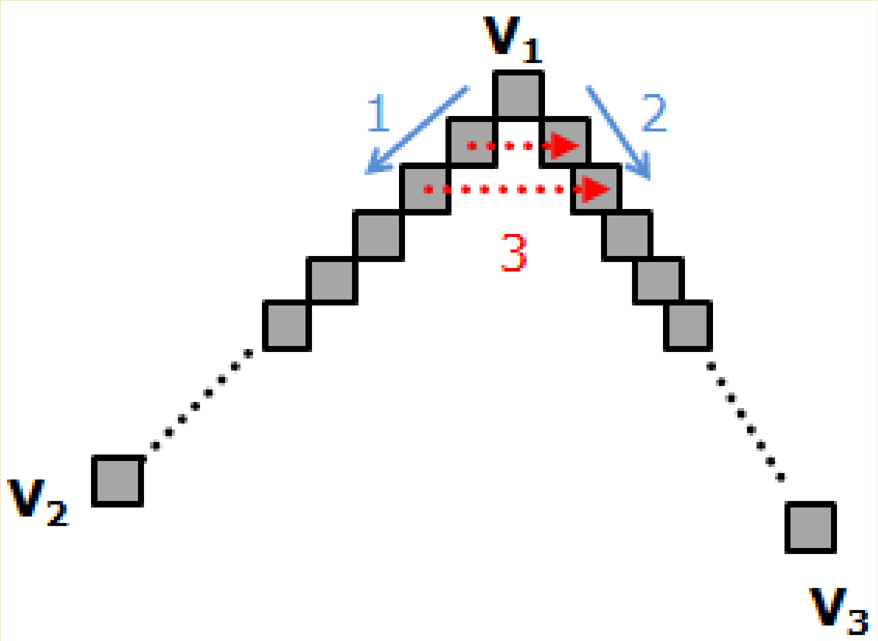
Edge Walking Method

1. Draw edges from each vertices using a line drawing algorithm (i.e. Bresenham)
   - Interpolate the color from one vertex to another



Edge Walking Method

2. Draw all horizontal lines from the **left edge pixel** to the **right edge pixel**
   – Interpolate the color from one vertex to another



Edge Walking Method

Advantages

Can be optimized to be fast

Low memory usage

Edge Walking Method

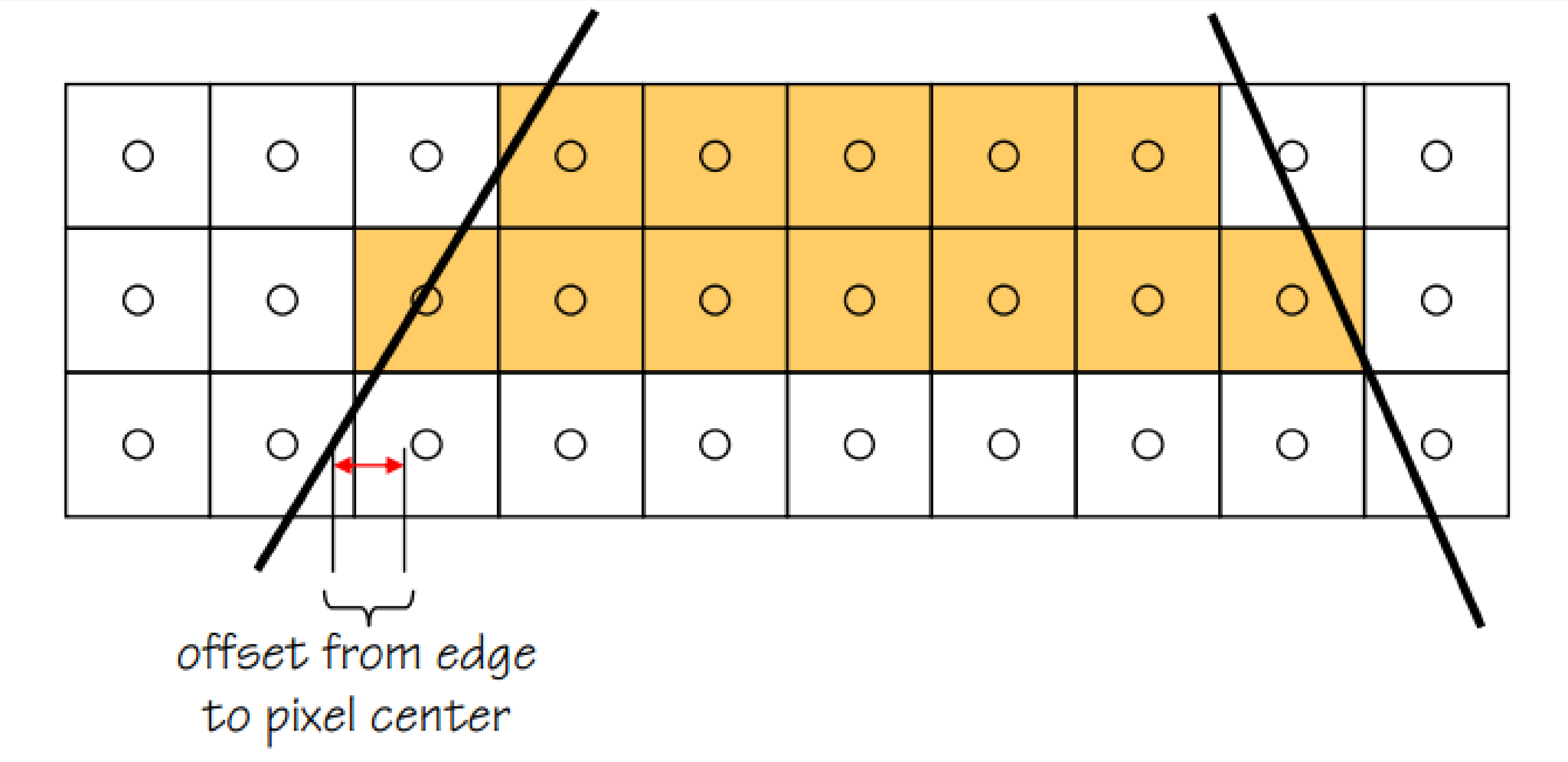
Disadvantages

Non straight forward scaling on large scenes

Problem on fractional offsets

Special Cases: Degenerate Triangles

Fractional Offsets



EDGE EQUATION METHOD

Edge Equation Method

Each edge of the triangle is a line

Each has a form of $Ax + By + C$

Edge Equation Method

A point can be categorized into 3:

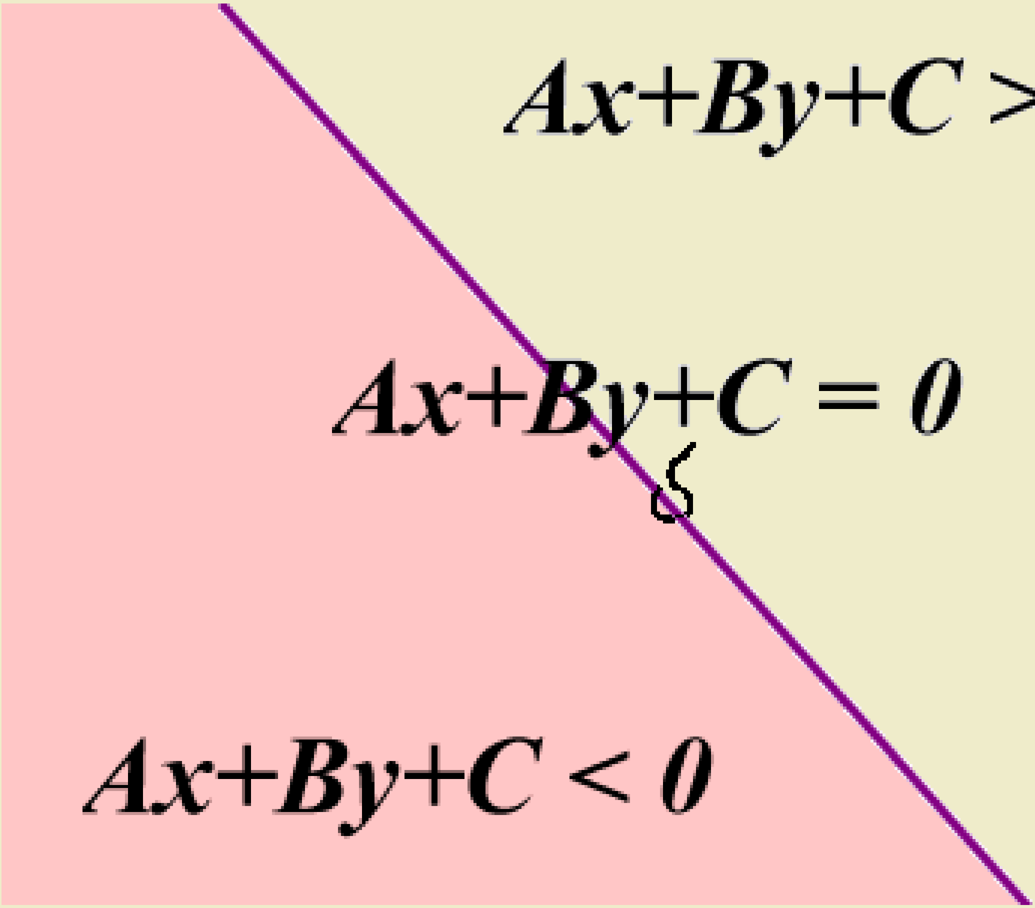
On the line: $Ax + By + C = 0$

"*Inside*" the line: $Ax + By + C > 0$

"*Outside*" the line: $Ax + By + C > 0$

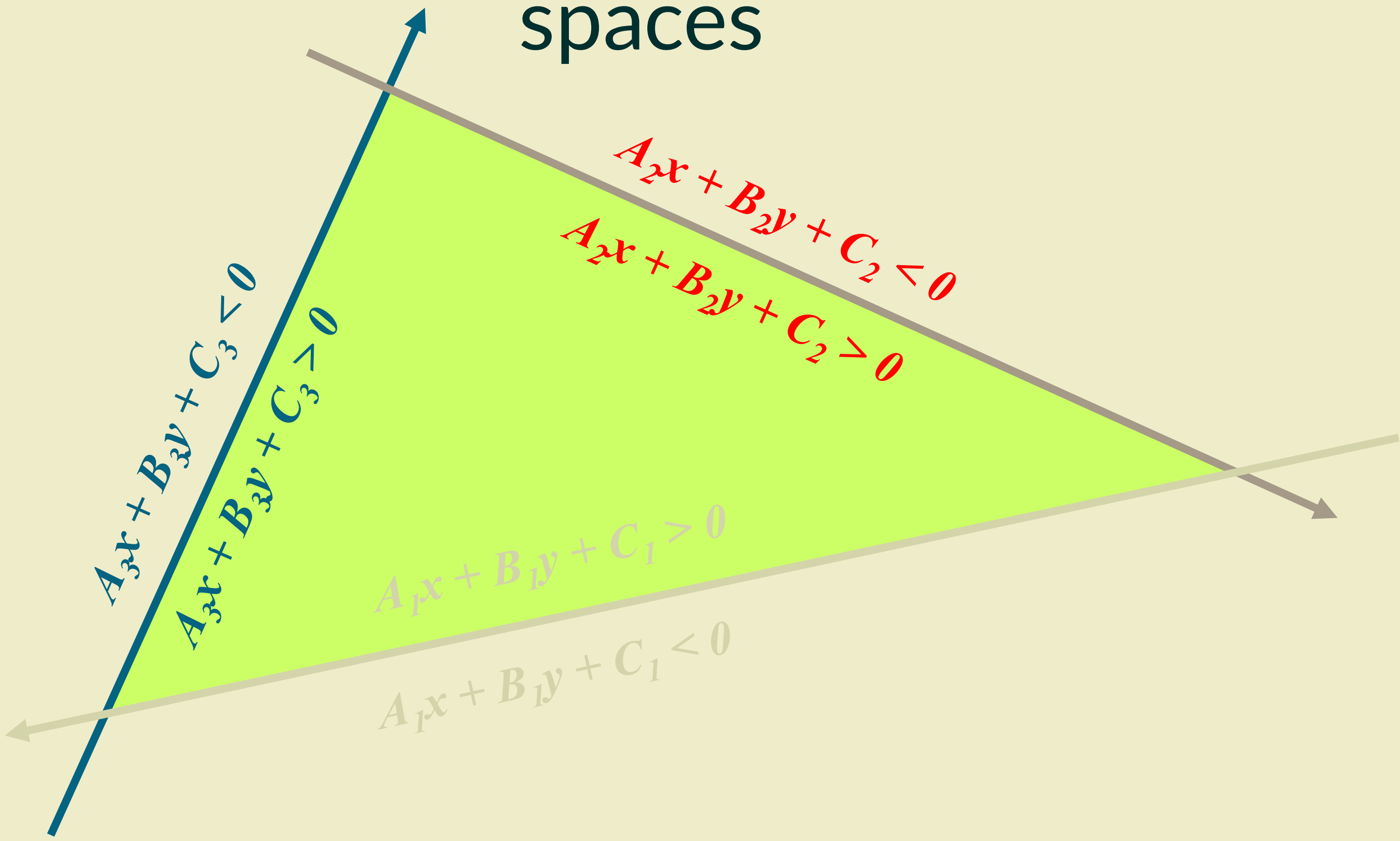
Edge Equation Method: Half space

Positive half-space and negative half-space



Edge Equation Method

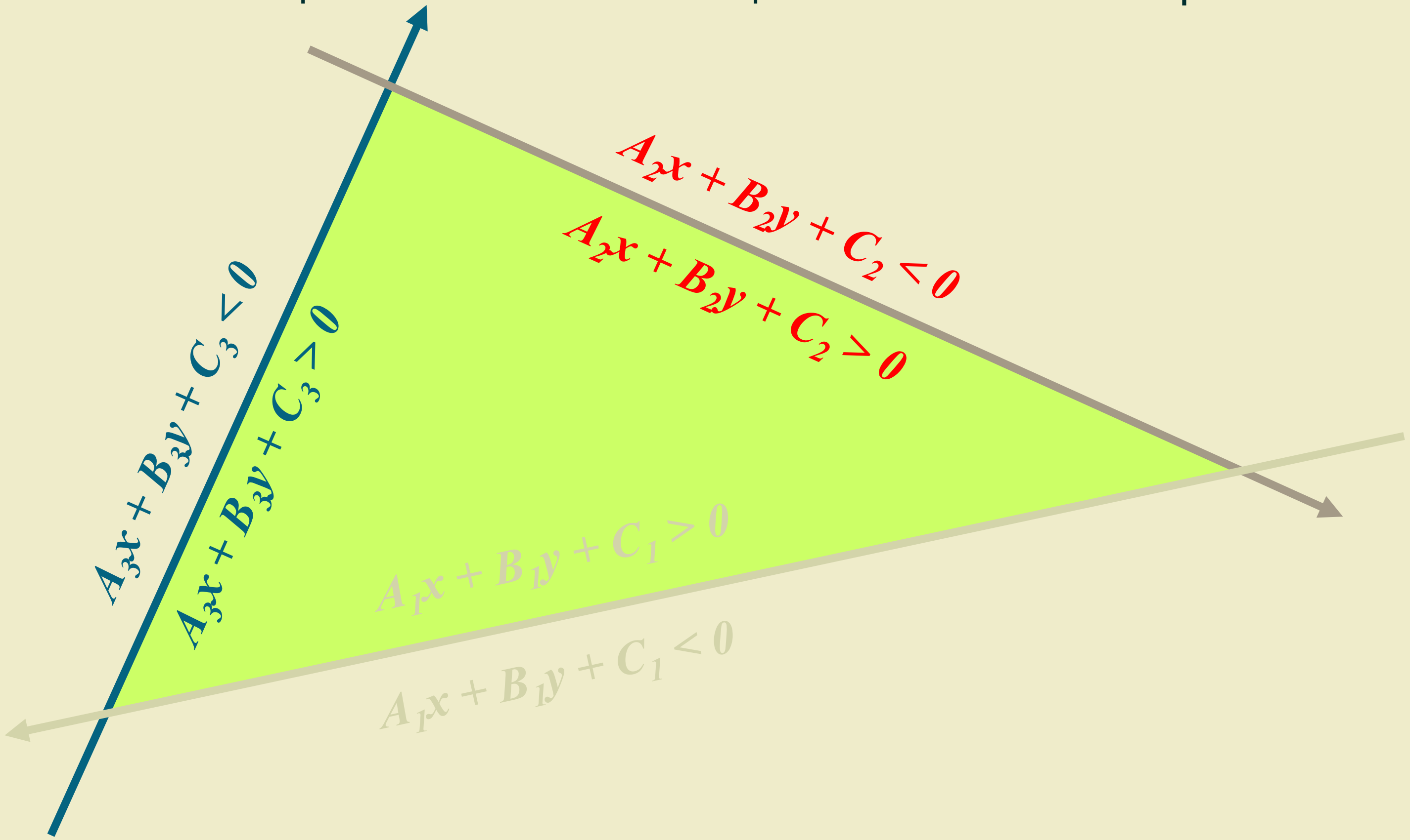
Triangle as intersection of 3 positive half spaces

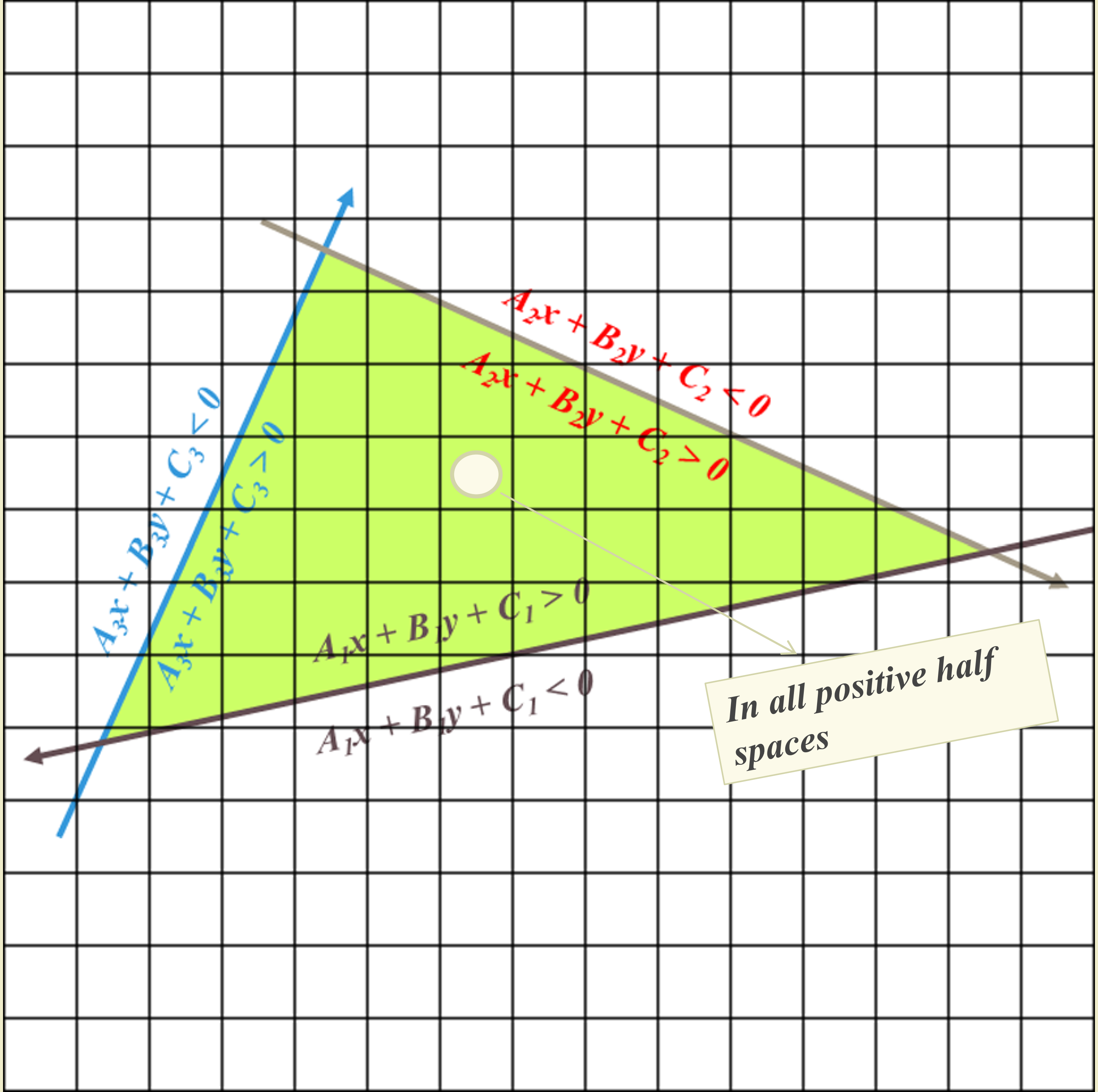


Edge Equation Method

For all candidate points

Check if the point if is in all positive half spaces




$A_2x + B_2y + C_2 < 0$
$A_2x + B_2y + C_2 > 0$
$A_3x + B_3y + C_3 < 0$
$A_3x + B_3y + C_3 > 0$
$A_1x + B_1y + C_1 > 0$
$A_1x + B_1y + C_1 < 0$
In all positive half spaces

Edge Equation Method

Coefficients can be computed using linear algebra

$$A = y_0 - y_1$$

$$B = x_1 - x_0$$

$$C = x_0 y_1 - x_1 y_0$$

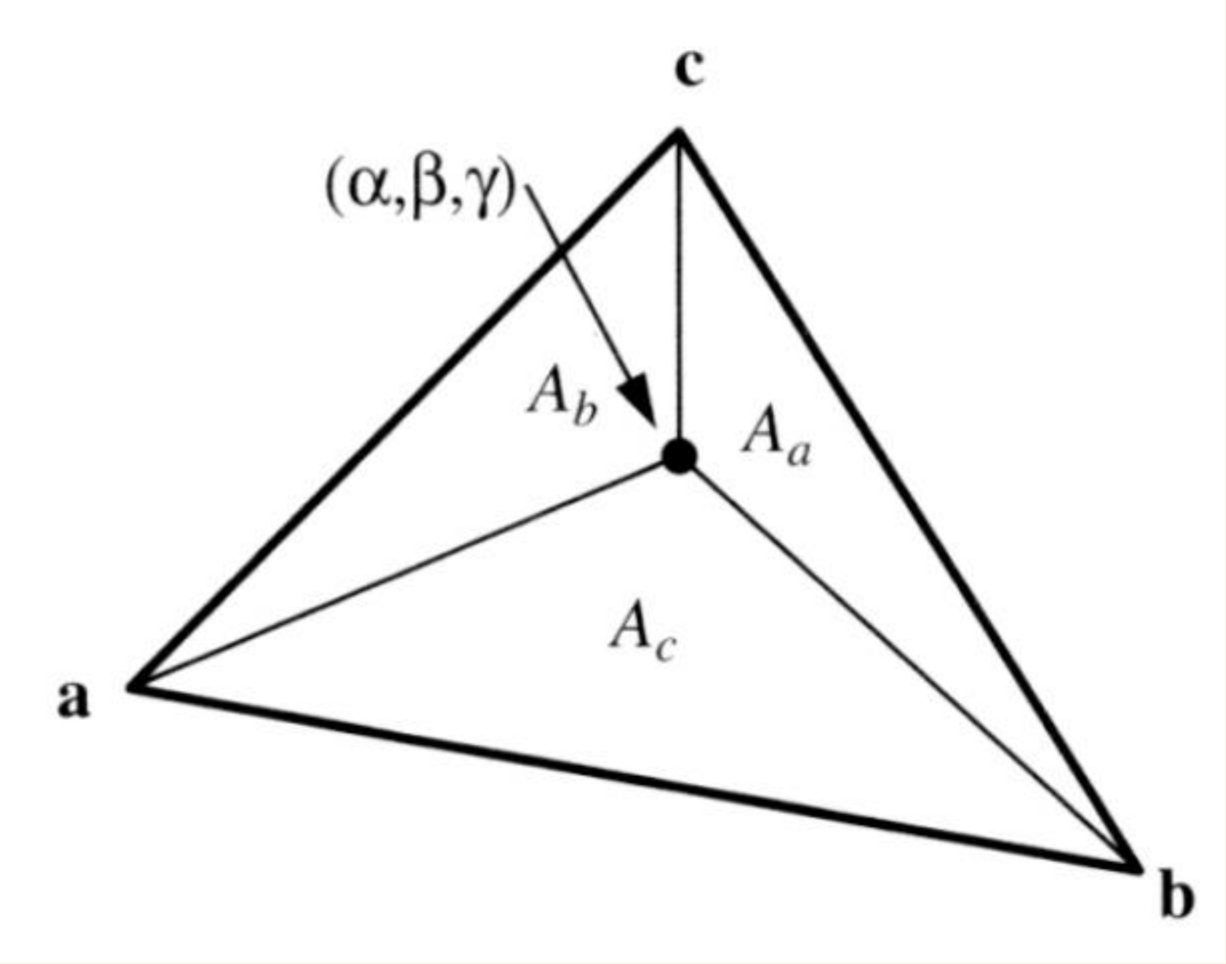
Edge Equation Method

Disadvantage:

May suffer from numerical precision errors

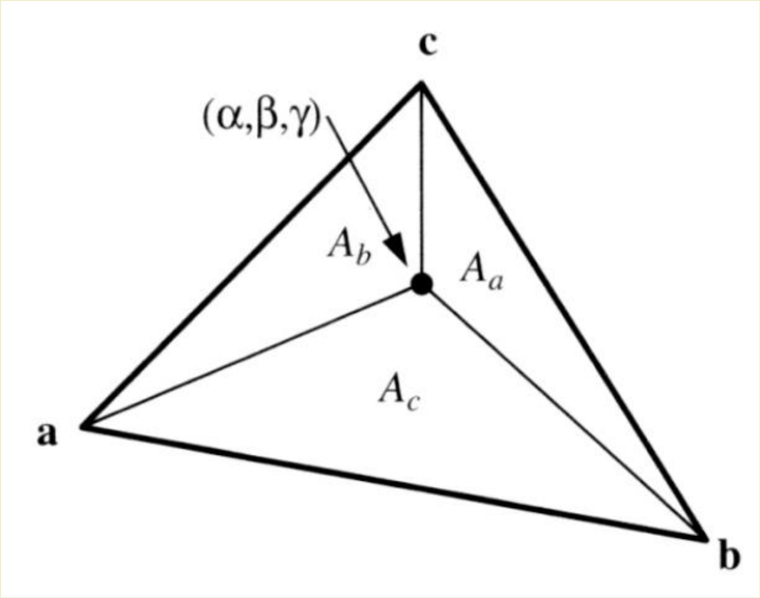
BARYCENTRIC ALGORITHM

Barycentric Coordinates



Barycentric Coordinates

Coordinate system for triangles based on affine geometry



Barycentric Coordinates

Properties:

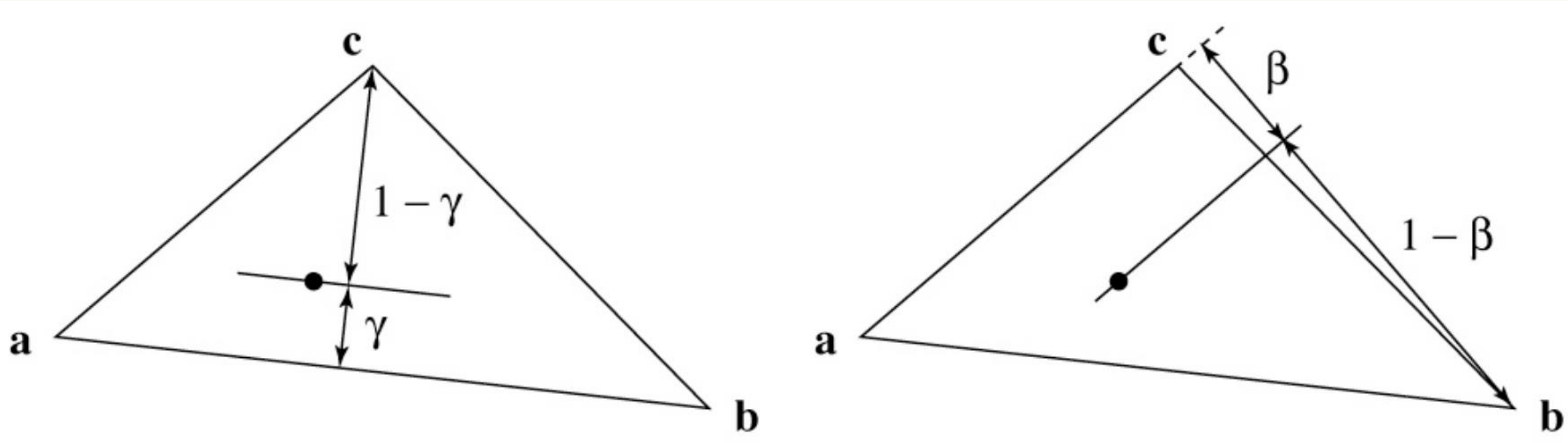
$$p(\alpha, \beta, \gamma) = \ \alpha A + \ \beta B + \gamma C$$

$$\alpha + \ \beta + \ \gamma = 1$$

Point $p(\alpha, \beta, \gamma)$ is **inside the triangle** if

$$0 \leq \alpha, \beta, \gamma \leq 1$$

Barycentric Coordinates



Barycentric Method

For each candidate point, compute the corresponding barycentric coordinate

Using determinants

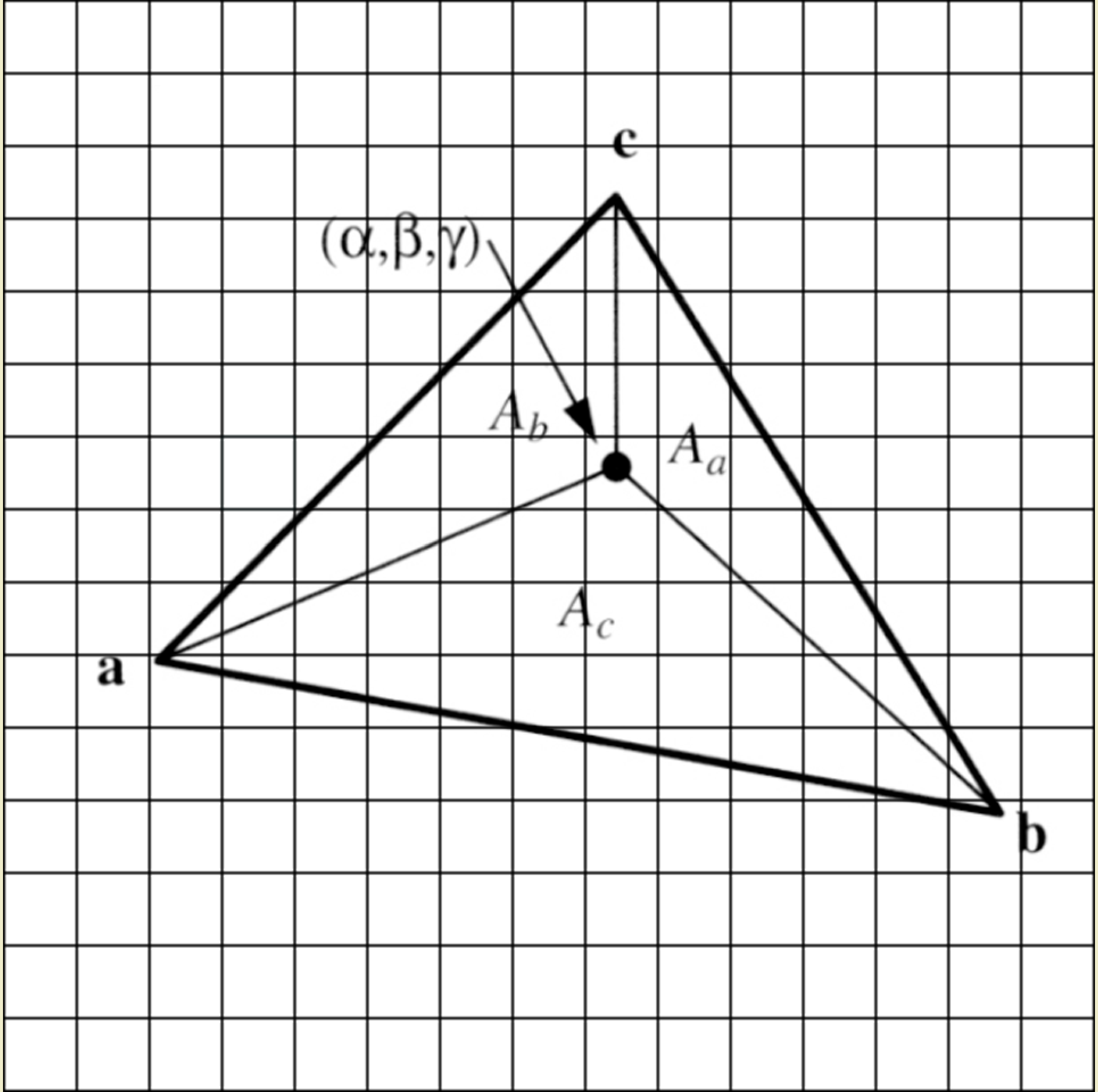
Affine Geometry Concepts

Trigonometric concepts

Barycentric Method

Use inside property

$$0 \leq \alpha, \beta, \gamma \leq 1$$


c
(α,β,γ)
A_b
A_a
A_c
a
b

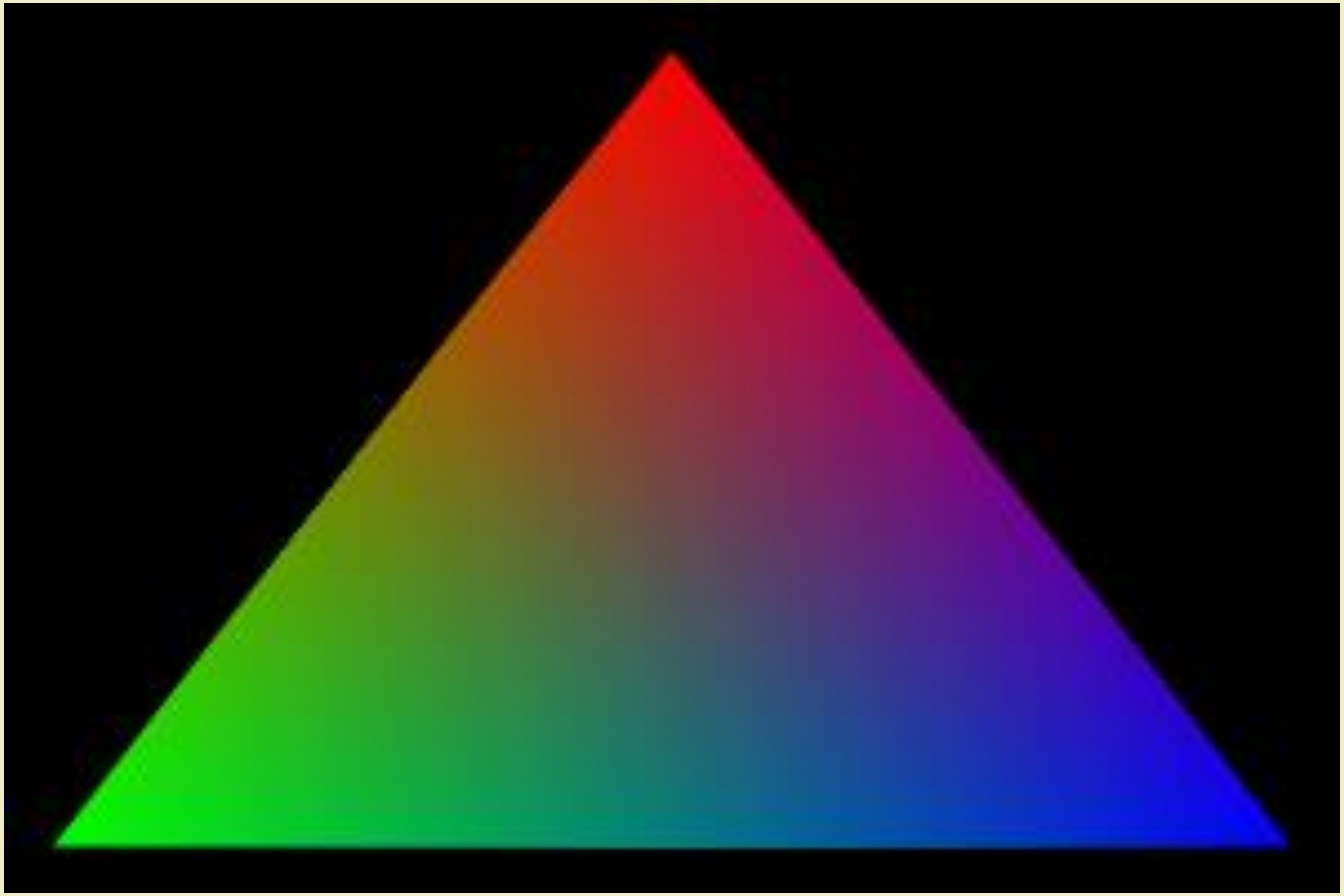
Sample Cartesian to Barycentric

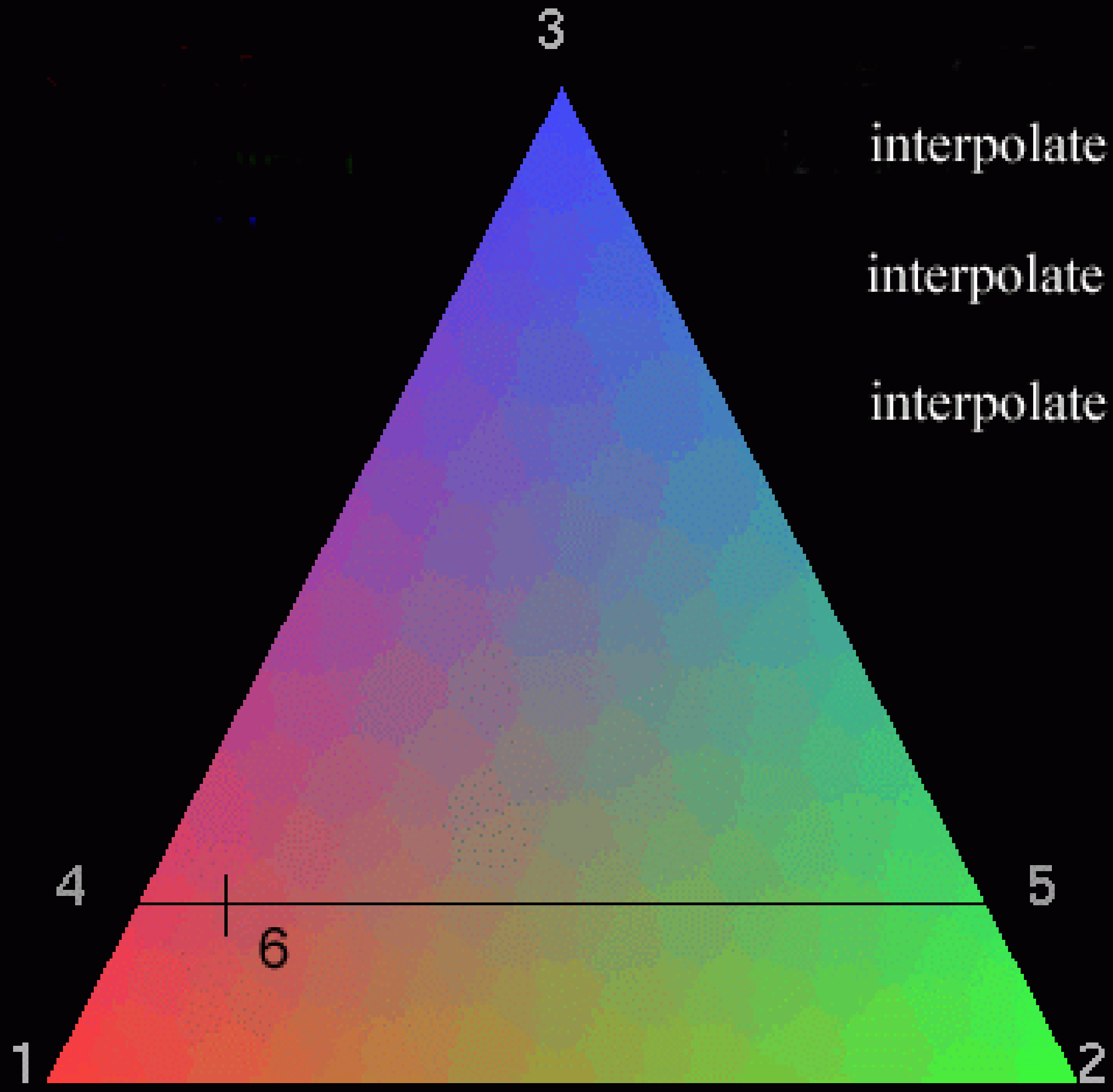
$$\alpha = \frac{\left(B_y - C_y\right)(\boldsymbol{x} - C_x) + (C_x - B_x)(\boldsymbol{y} \ - C_y)}{\left(B_y - C_y\right)(A_x - C_x) + (C_x - B_x)(A_x \ - C_y)}$$

$$\beta = \frac{\left(C_y - A_y\right)(\boldsymbol{x} - C_x) + (A_x - C_x)(\boldsymbol{y} \ - C_y)}{\left(B_y - C_y\right)(A_x - C_x) + (C_x - B_x)(A_x \ - C_y)}$$

$$\gamma = 1 - \alpha - \beta$$

Barycentric Coordinates: Interpolation of Color





interpolate between 1,3 to get 4

interpolate between 2,3 to get 5

interpolate between 4,5 to get 6

GENERAL POLYGON RASTERIZATION

General Polygon Rasterization

What if the system doesn't suuport polygon triangulation?

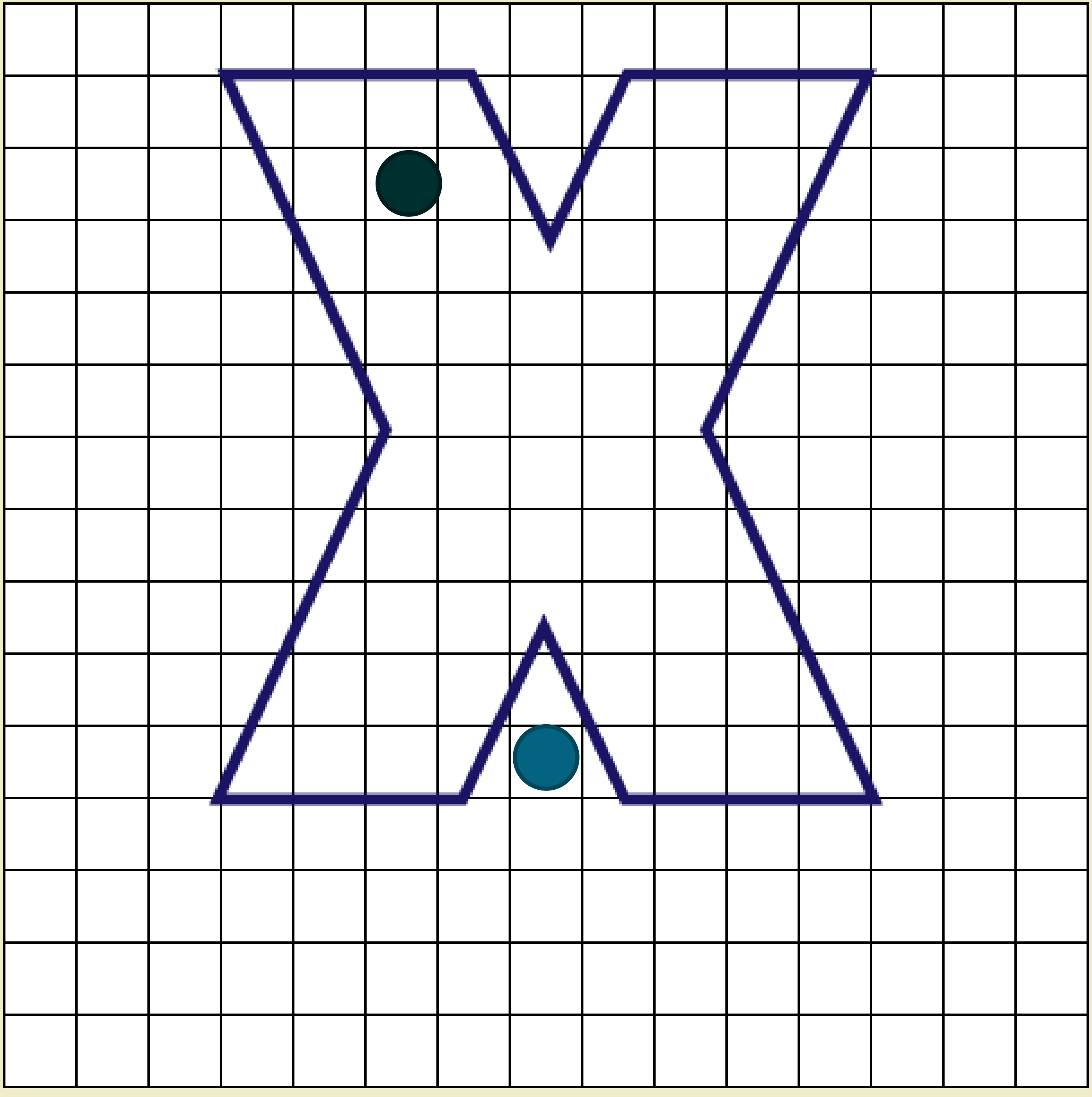
Polygon cannot be easily broken down into triangles

Inside or Outside?

Given a point P and a polygon, is P inside or outside the polygon

If inside, color the pixel in the color buffer

If outside, do not color

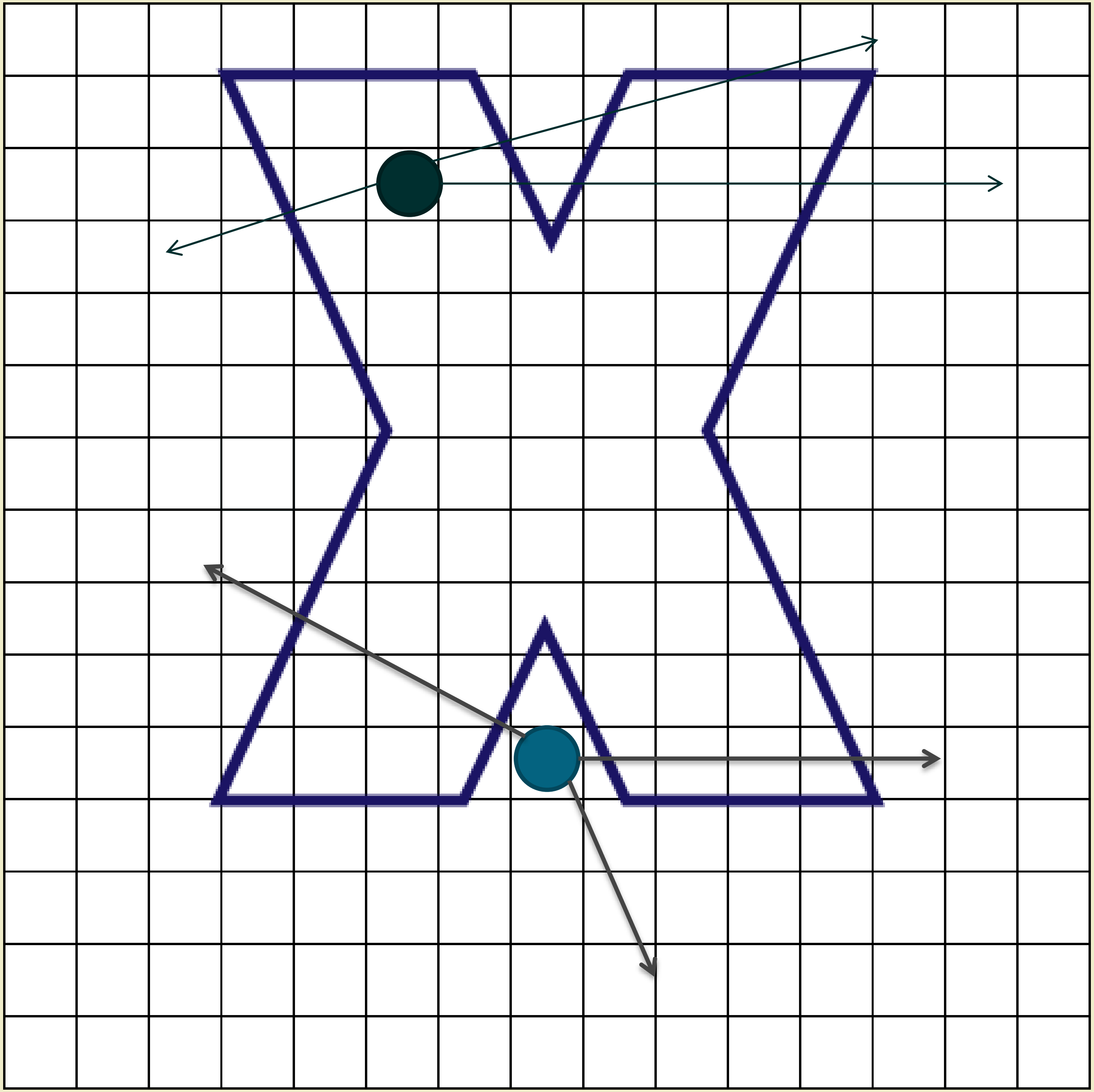


Odd-Even Testing

Draw a line from the point to infinity

If it crosses an <u>odd number</u> of edges, it is INSIDE

If it crosses an <u>even number</u> of edges, it is OUTSIDE



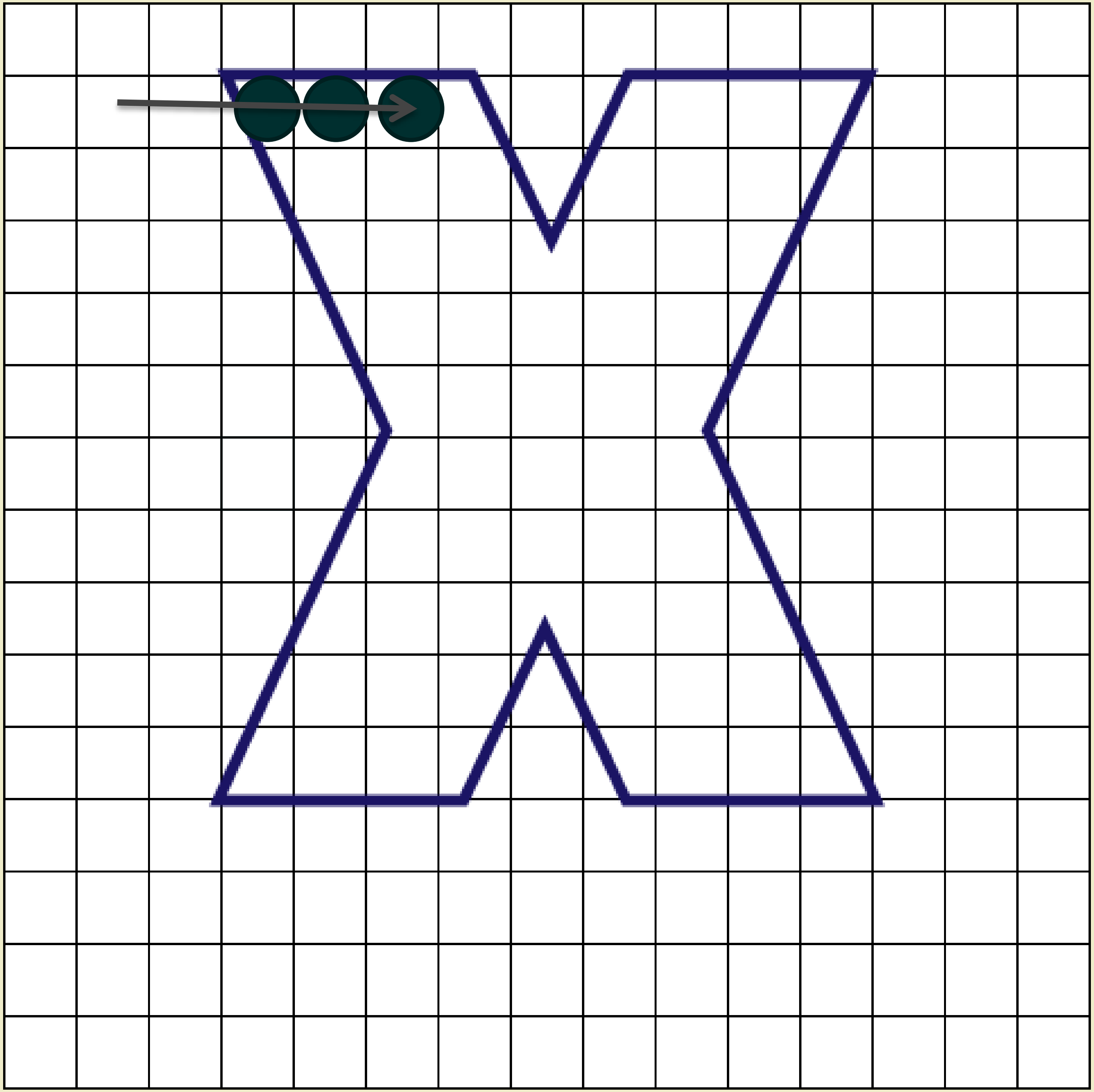
GENERAL POLYGON FILL ALGORITHMS

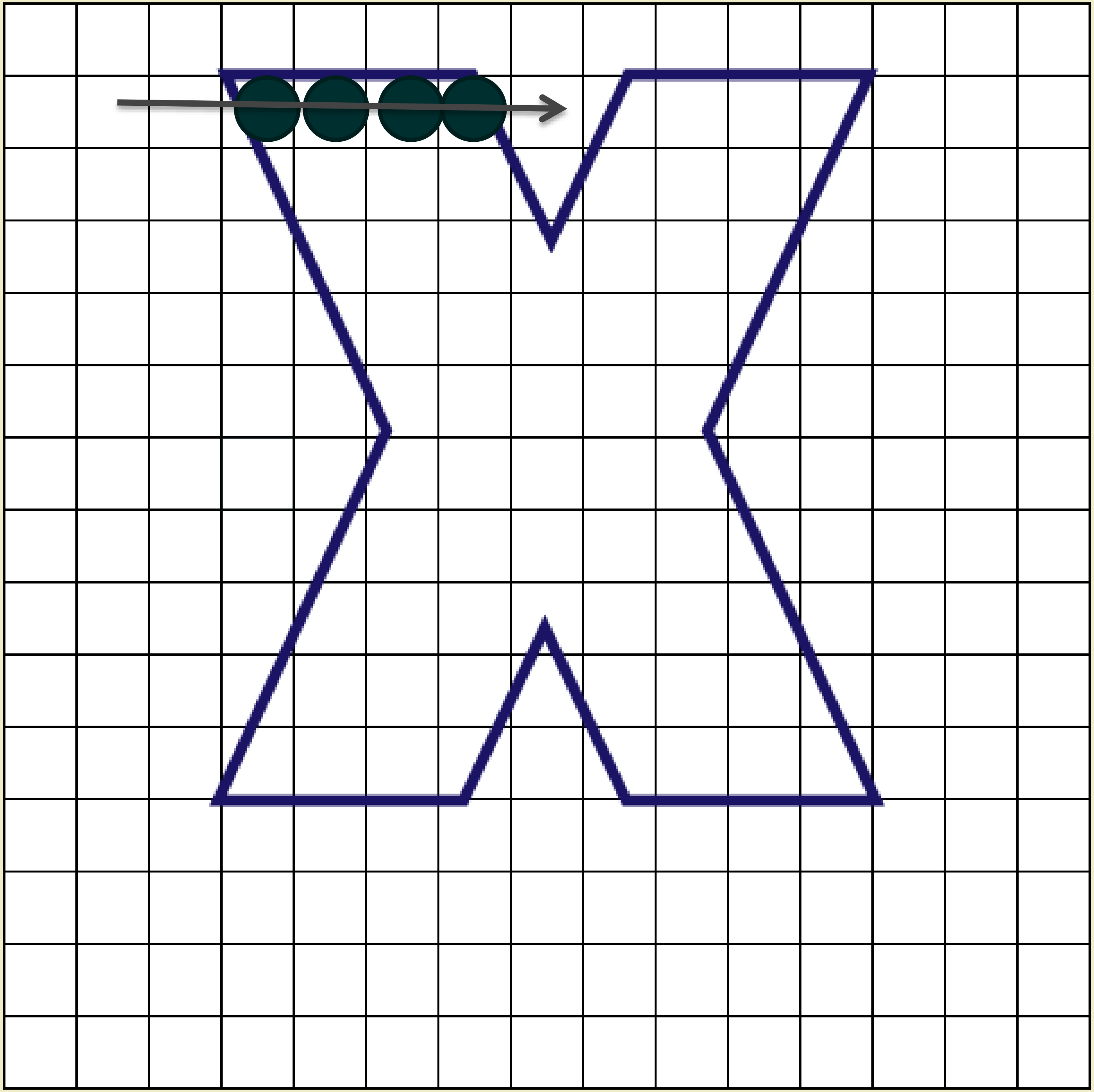
Odd-Even Fill

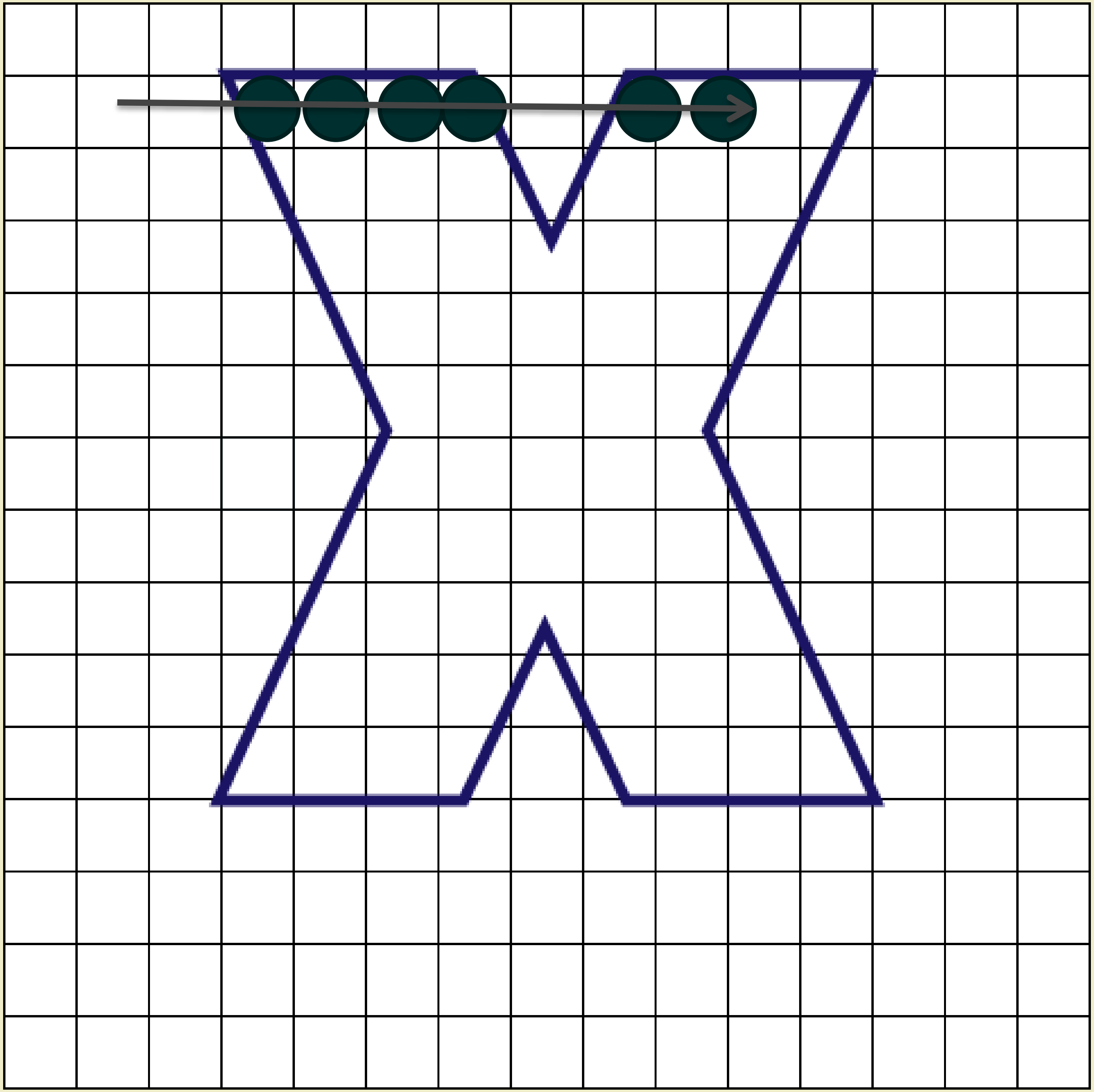
For each horizontal line,

if current intersection count is odd, start drawing

if current intersection count is even, stop drawing

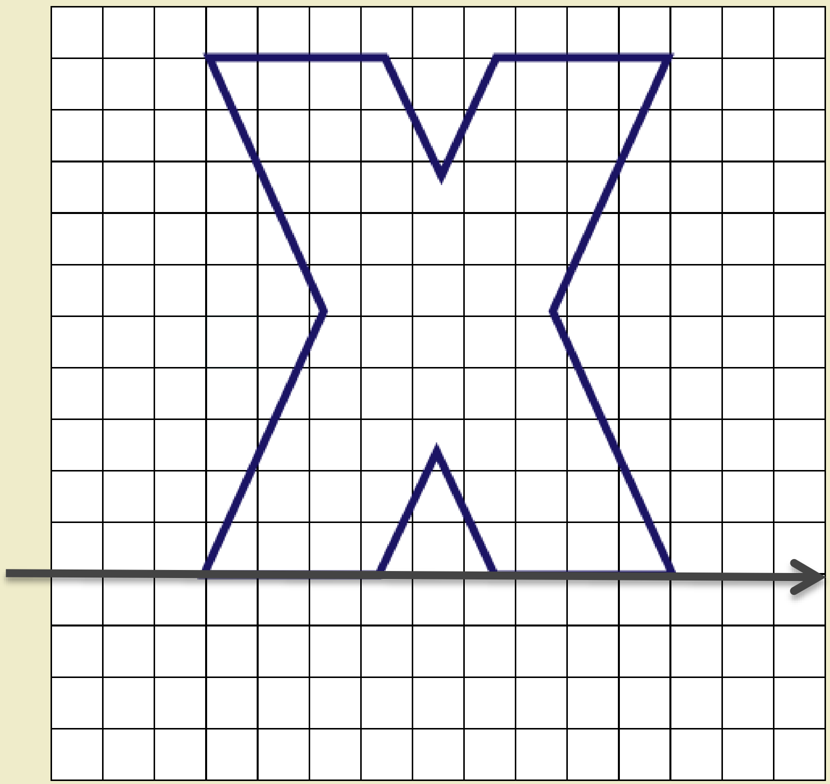






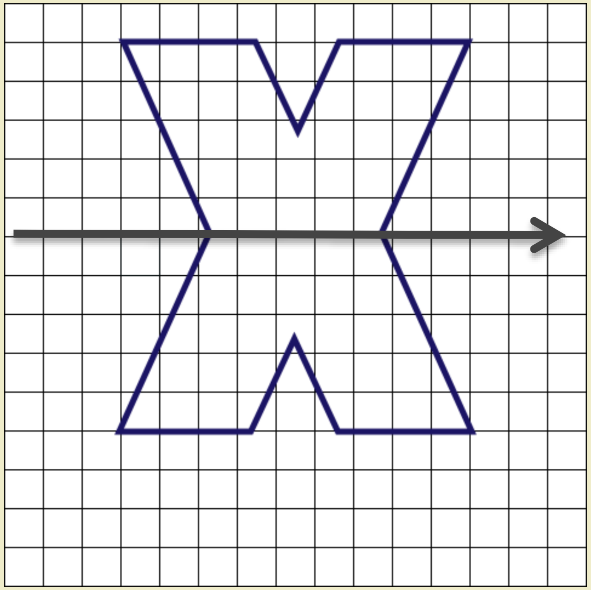
Odd-Even Fill

Scanline passing through horizontal edge



Odd-Even Fill

Scan line passing through a vertex



*Is it **even** edge crossing or **odd** edge crossing?*

Flood Fill

Draw edges of Polygon using Line Drawing Algorithms

Flood Fill

Find a seed pixel that is inside the polygon

Recursively color its neighbors

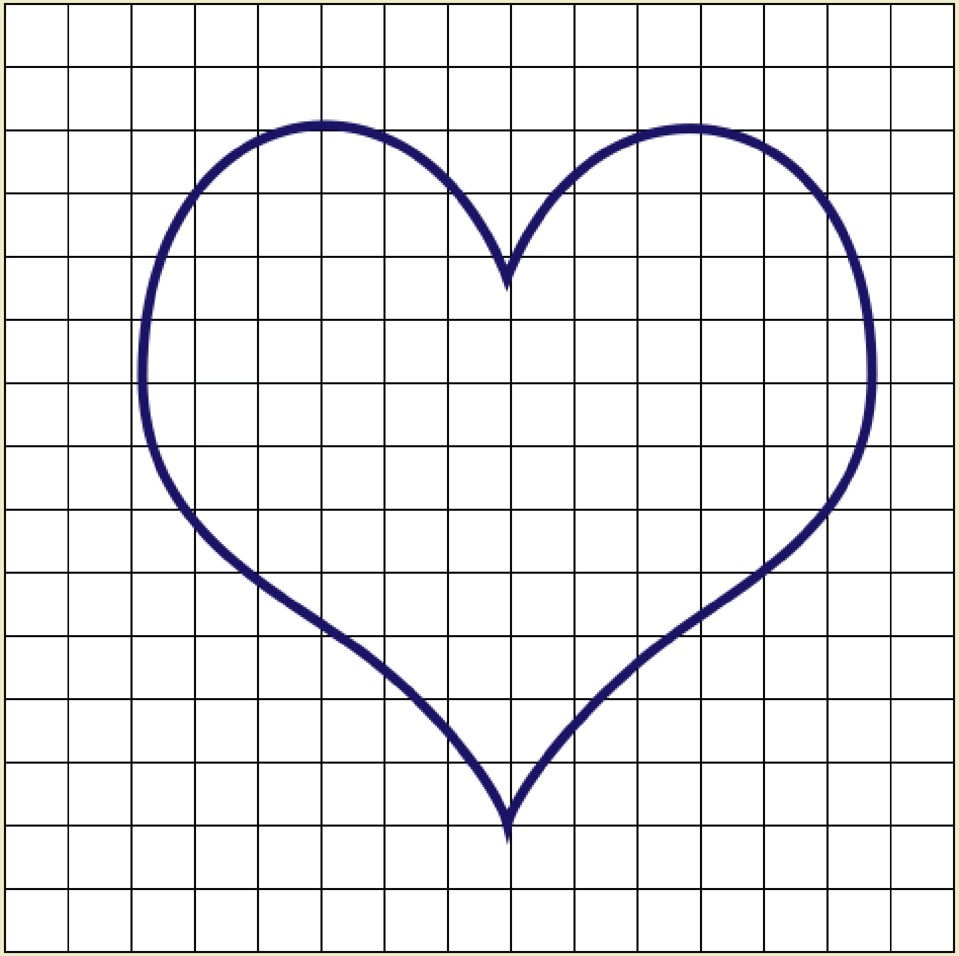
- 4-connected
- 8-connected

Flood Fill

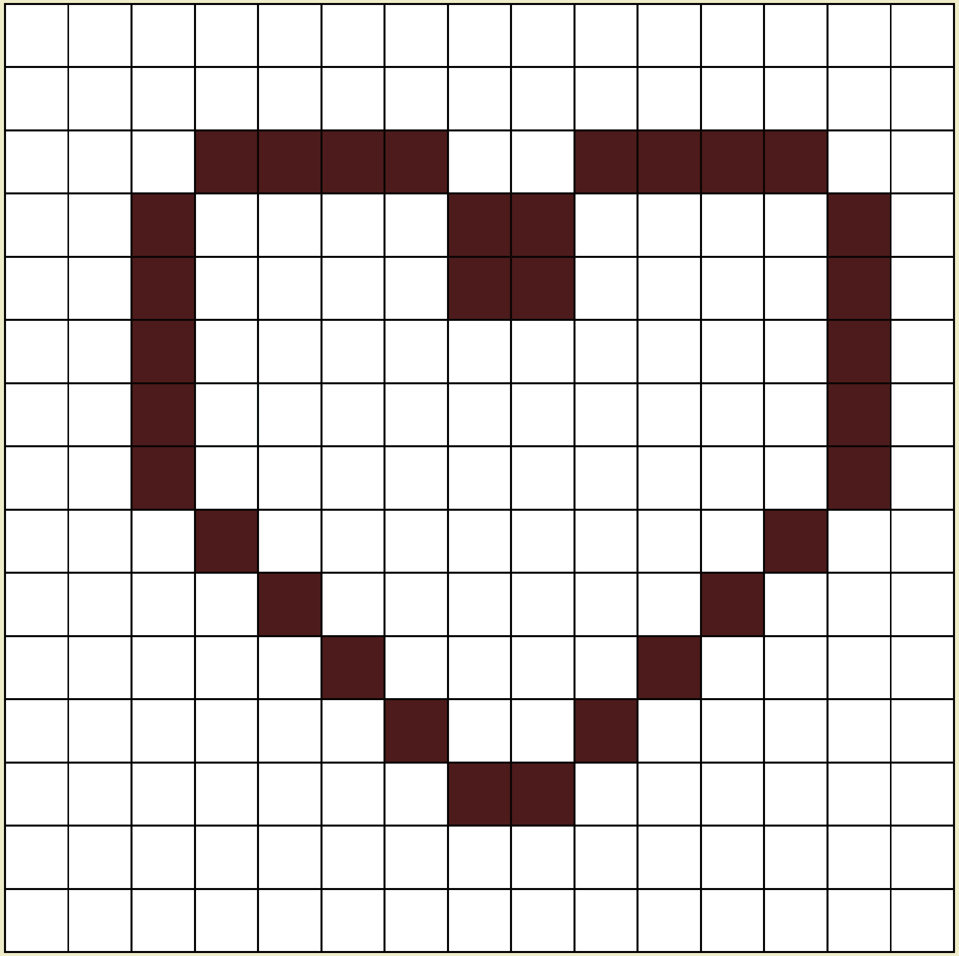
Breadth-First Search (BFS) Algorithm

applied to a grid

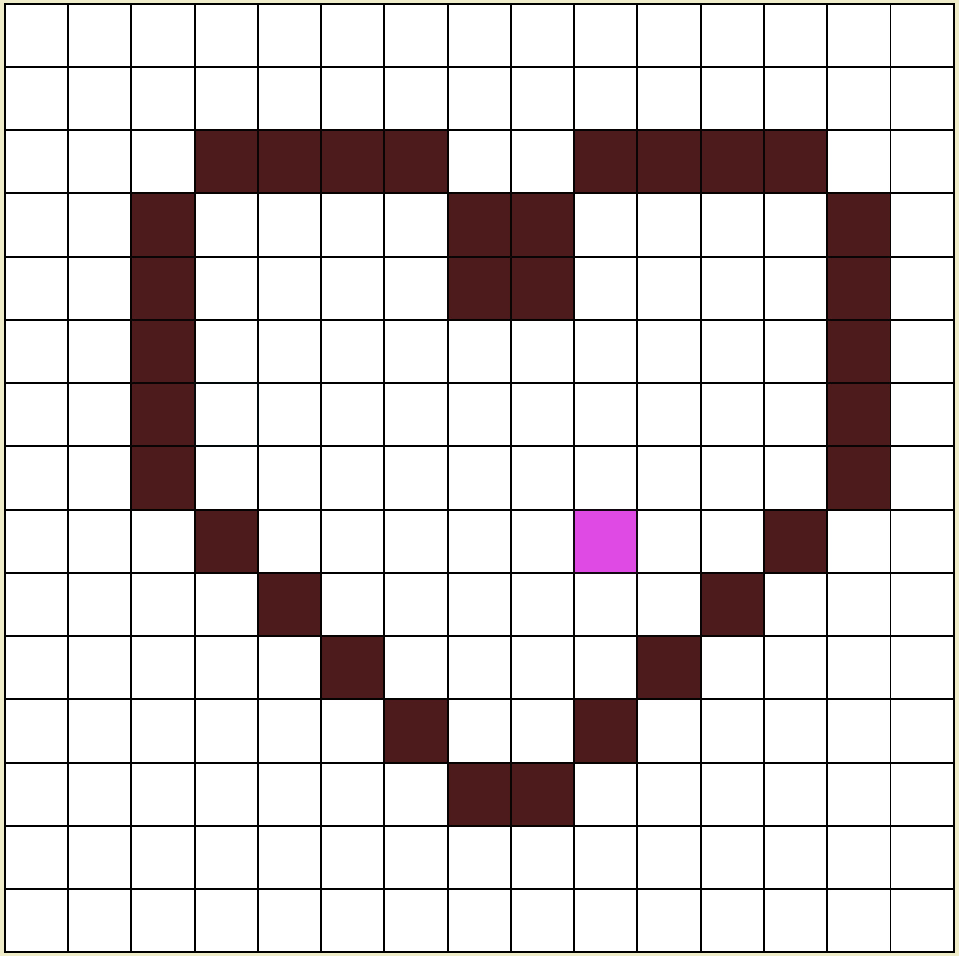
Flood Fill: 4-connected



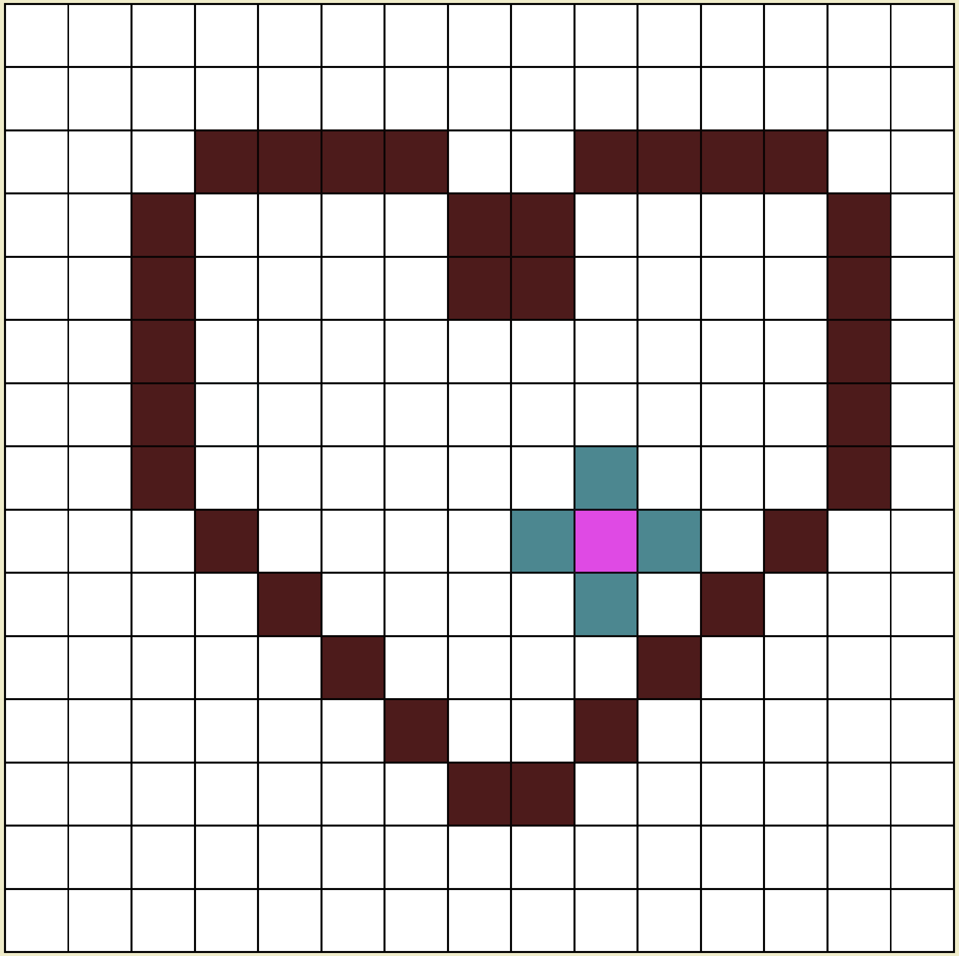
Flood Fill: 4-connected



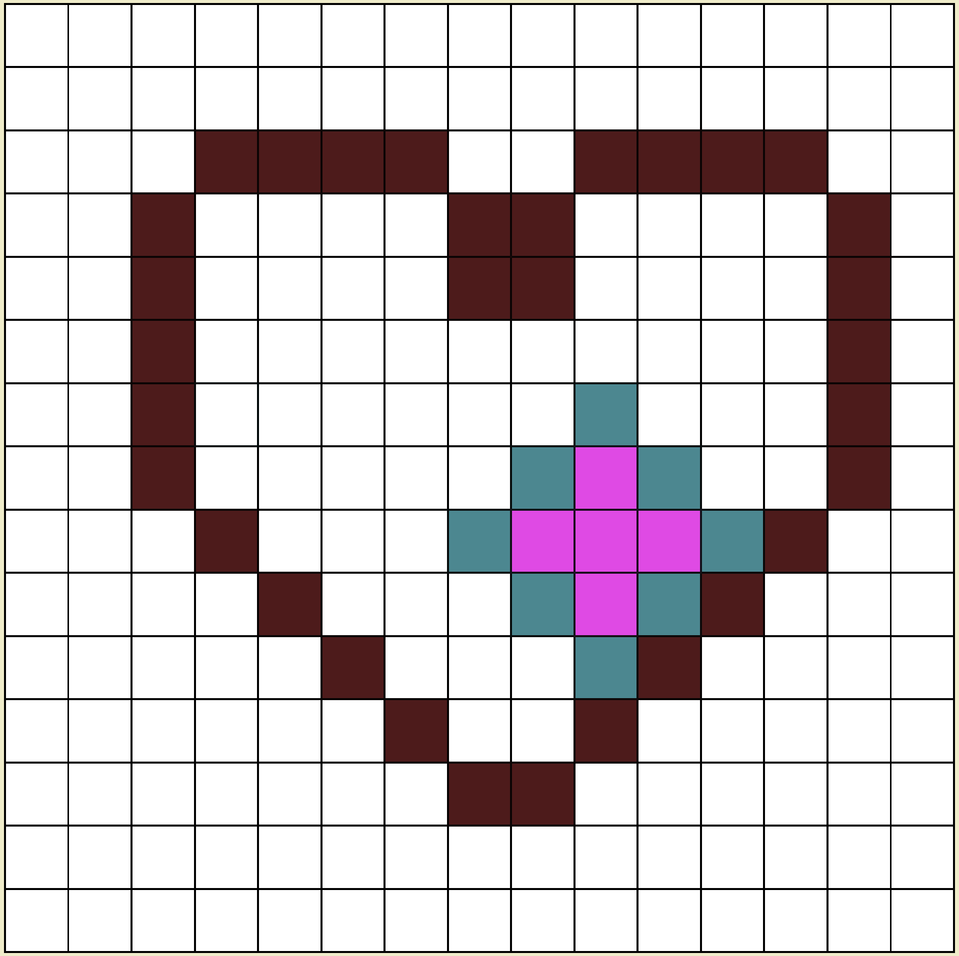
Flood Fill: 4-connected



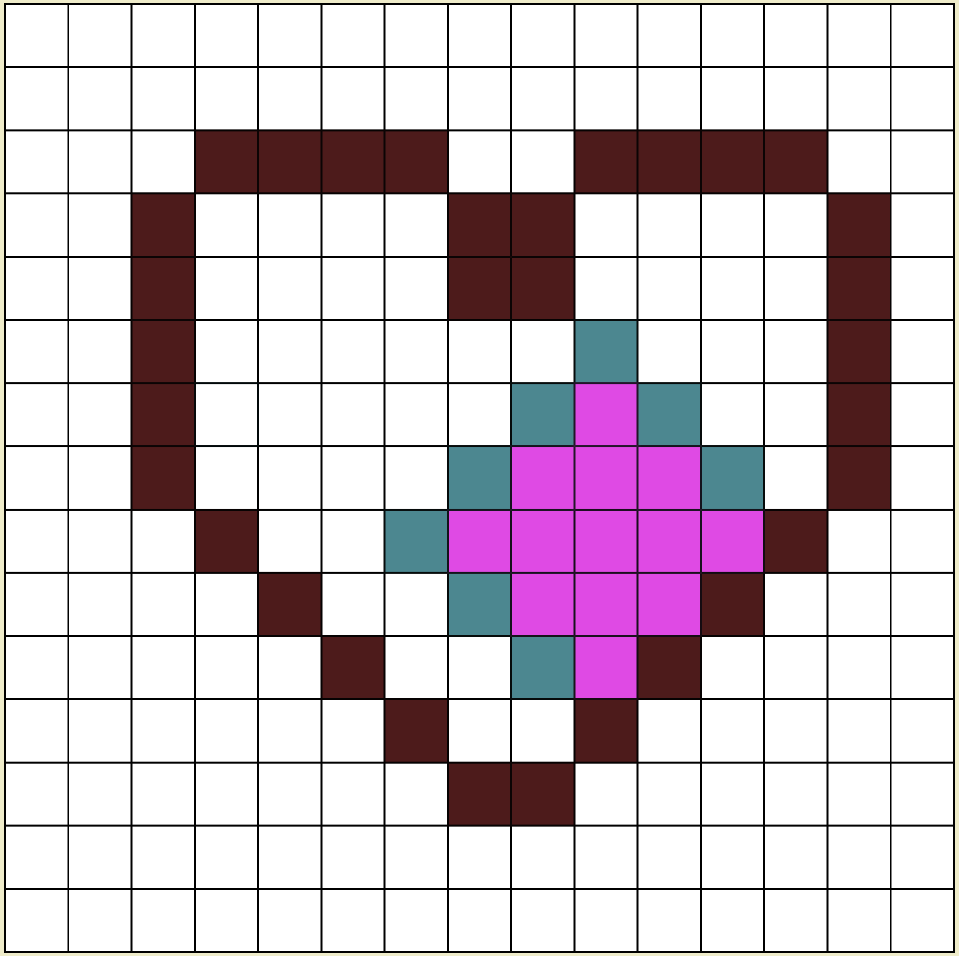
Flood Fill: 4-connected



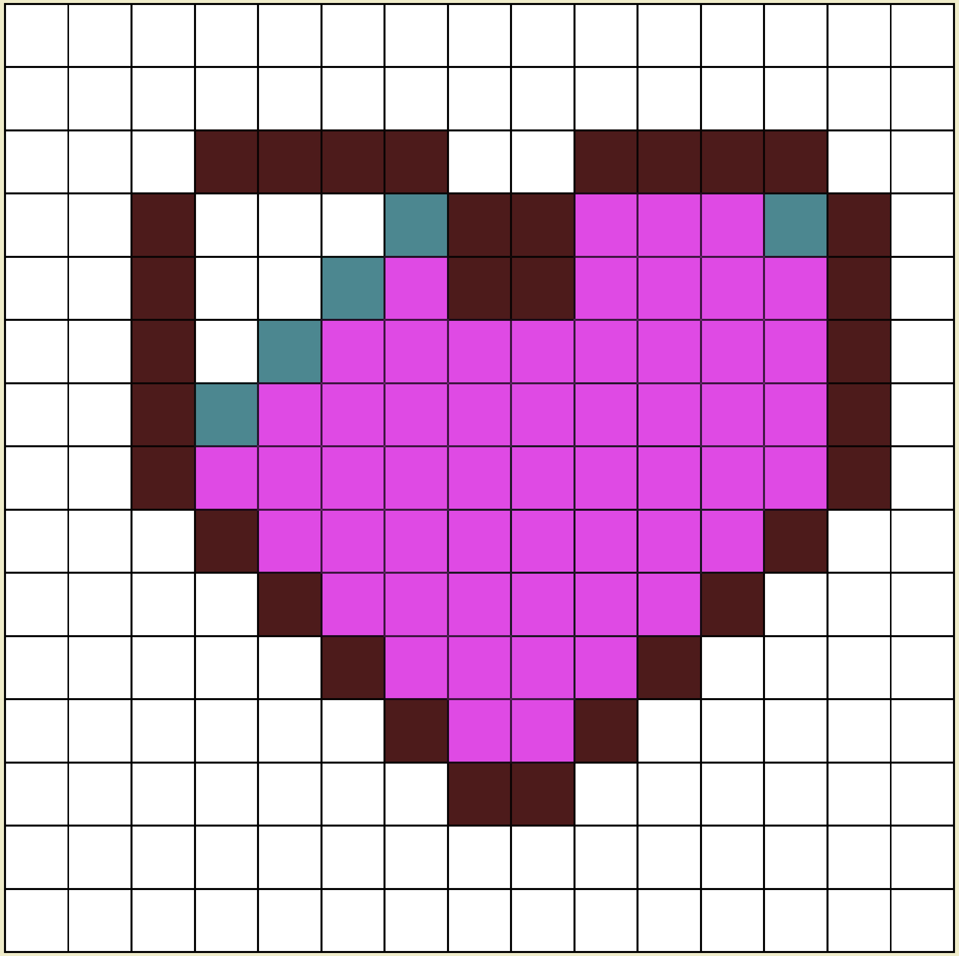
Flood Fill: 4-connected



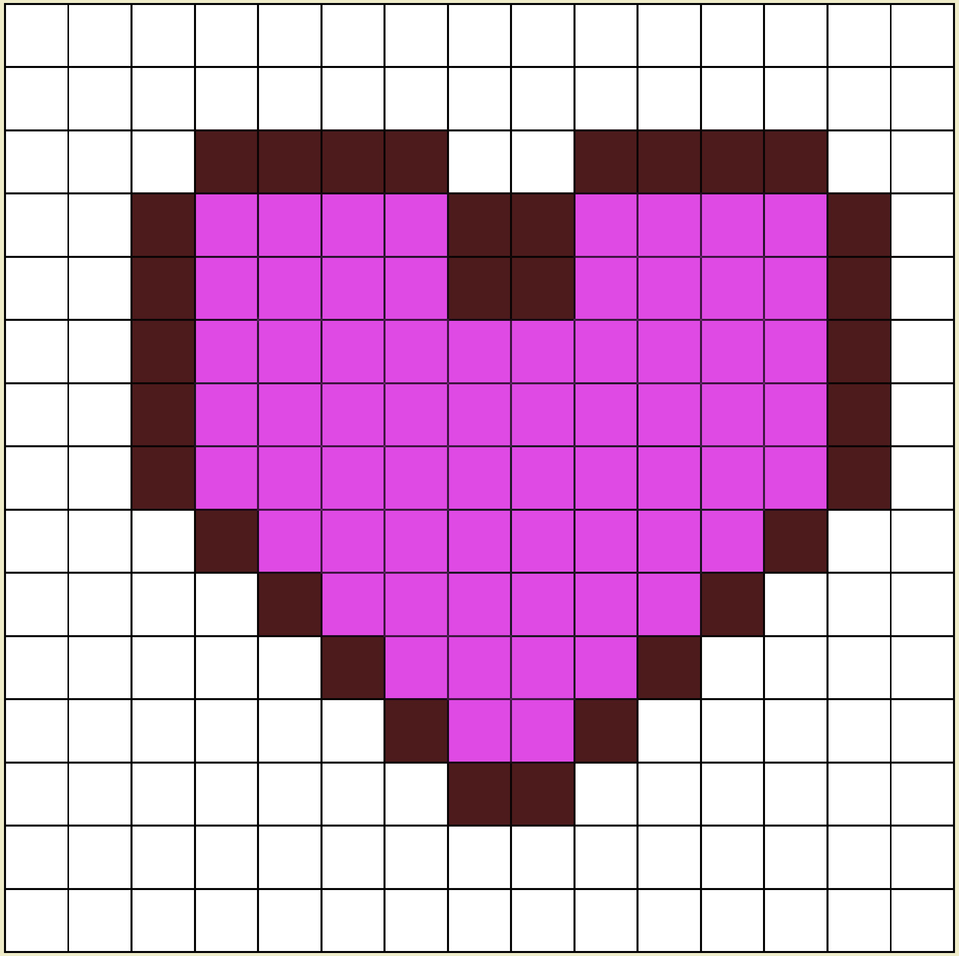
Flood Fill: 4-connected



Flood Fill: 4-connected



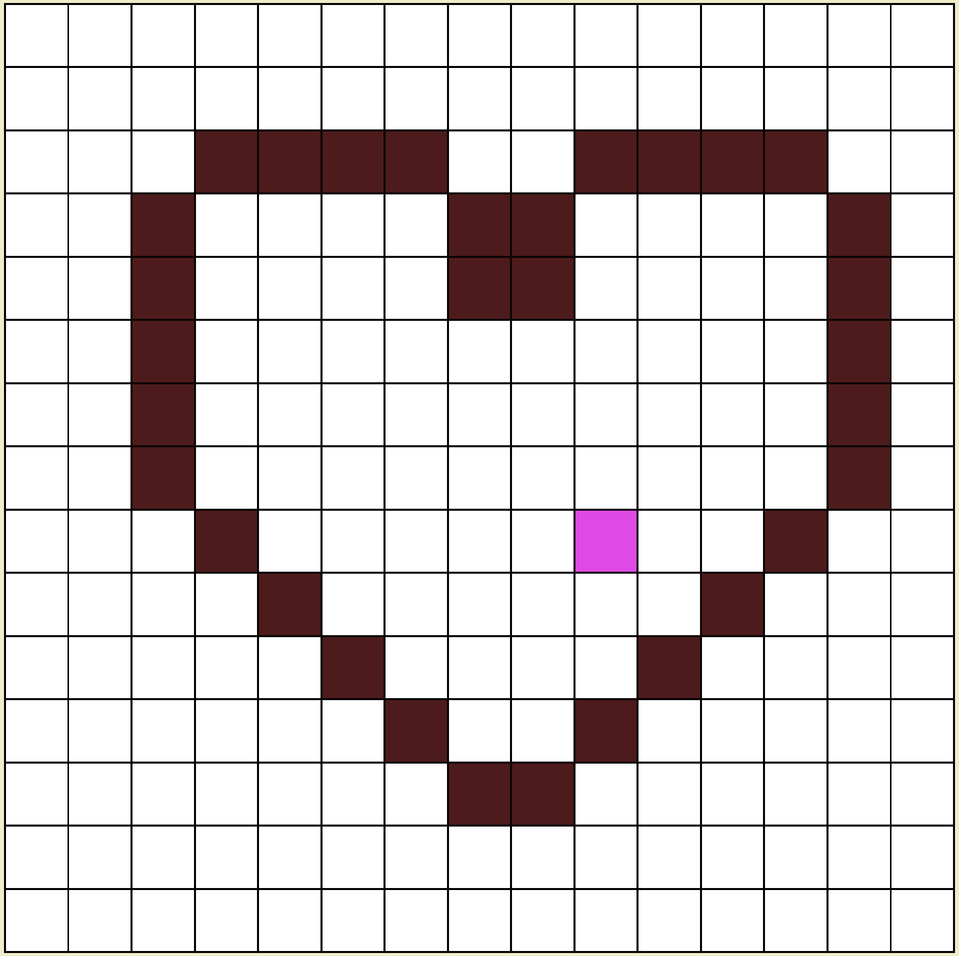
Flood Fill: 4-connected



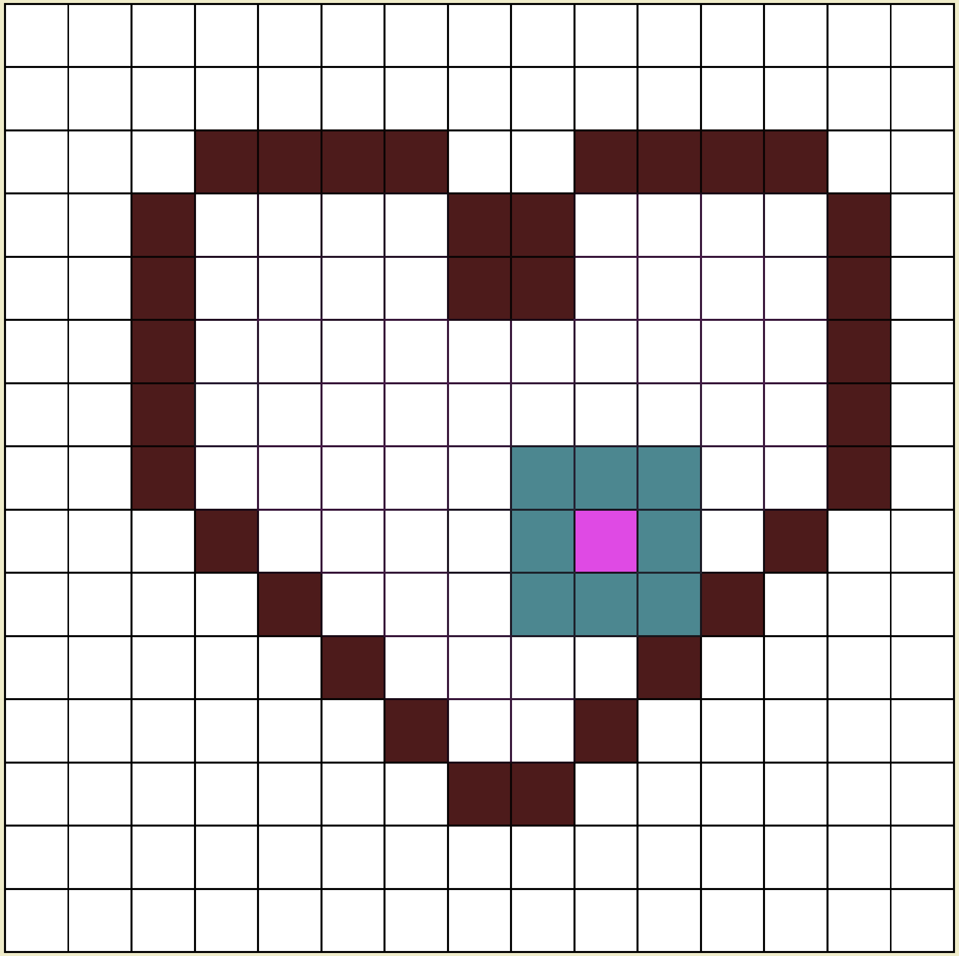
Flood Fill 4-connected: Pseudocode

```
Flood_fill(x, y,color) {
    if(colorBuffer[x,y] == EMPTY) {
        colorBuffer[x,y] =color
        Flood_fill(x-1,y)
        Flood_fill(x+1,y)
        Flood_fill(x,y-1)
        Flood_fill(x,y+1)
    }
}
```

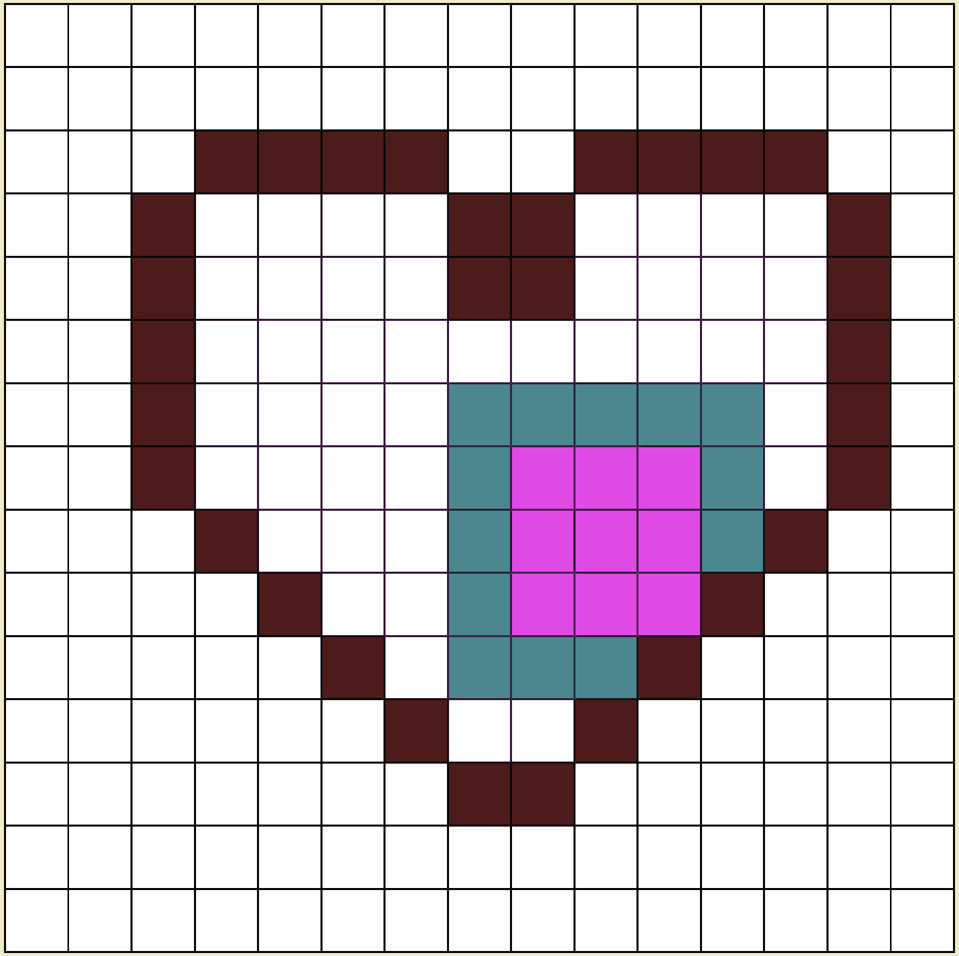
Flood Fill: 8-connected



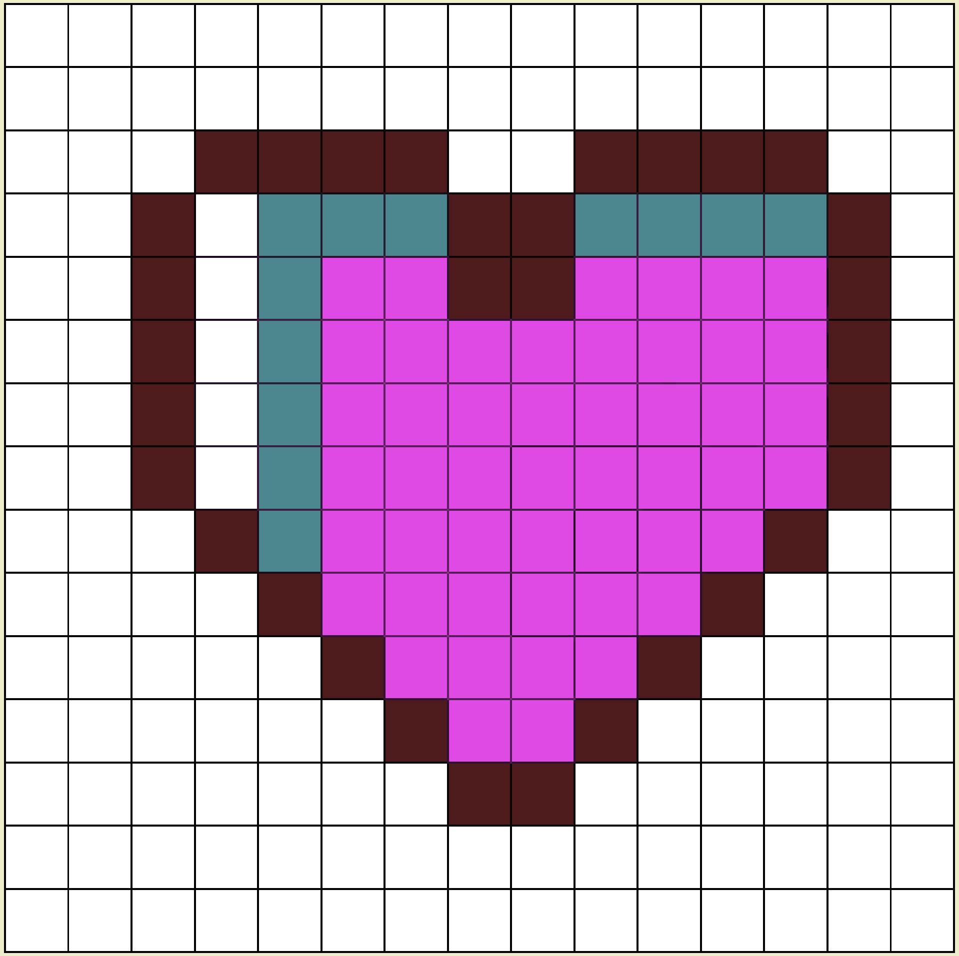
Flood Fill: 8-connected



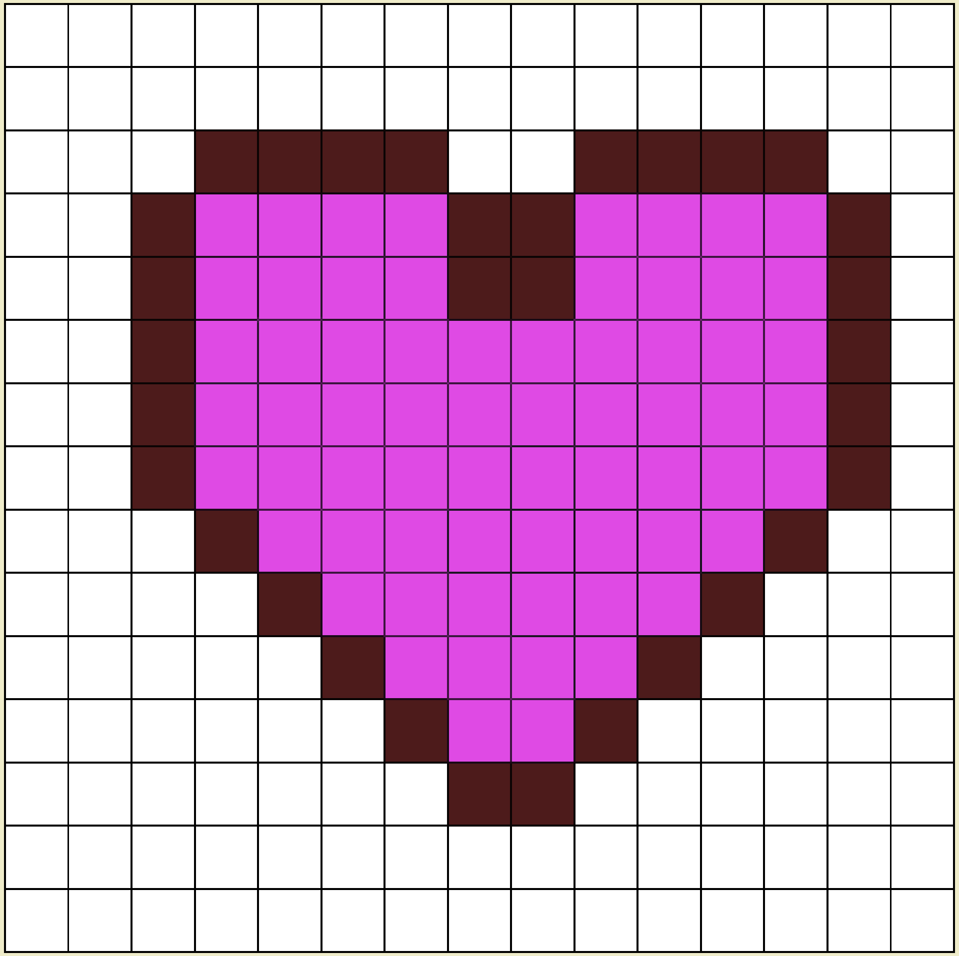
Flood Fill: 8-connected



Flood Fill: 8-connected



Flood Fill: 8-connected



References

Books

- ANGEL, E. AND SHREINER, D. 2012. Interactive computer graphics : a top-down approach with shader-based OpenGL. Addison-Wesley. 6 ed. Boston, MA.
- CANTOR, D. AND JONES, B. 2012. WebGL Beginner's Guide. Packt Publishing. Birmingham, UK.
- MATSUDA, K. AND LEA, R. 2013. WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL.. Addison-Wesley. Upper Saddle River, NJ

Lecture Slides

- BLOOMFIELD, A. Rasterization. CS 445: Introduction to Graphics Fall 2006 Lecture Slides
- LLOYD, B. Triangle Rasterization. COMP 770 (236): Computer Graphics Spring 2007 Lecture Slides

Images

- http://dev.opera.com/articles/view/raw-webgl-part1-getting-started/
- http://www.cs.virginia.edu/~asb/teaching/cs445-fall06/slides/09-rasterization.ppt
- http://mathworld.wolfram.com/images/eps-gif/ConvexPolygon_1000.gif
- http://download.autodesk.com/global/docs/maya2013/en_us/images/comp_poly_customwarpeg.png
- http://www.sunshine2k.de/coding/java/TriangleRasterization/bresenhamIdea.png
- http://stochastix.files.wordpress.com/2008/07/ippolita50k-close-up.png
- http://www.eecs.berkeley.edu/~sequin/CS184/IMGS/rgb-gouraud-triangle.gif