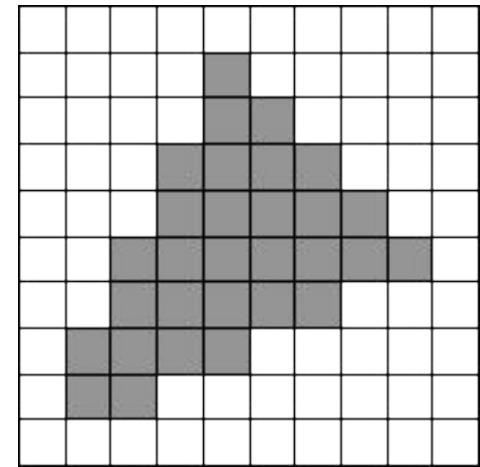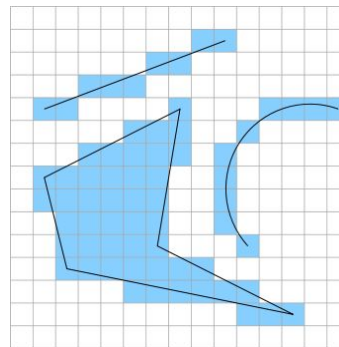# Computer Graphics

Lecture 7
Rasterization
Taku Komura

Rasterization

• After projection, the polygons are still in the continuous screen space

• We need to decide which pixels to lit how much

• This is called rasterization (or scan conversion)

Overview

Rasterization
- Line Rasterization
- Polygon Rasterization
  - Scanline Algorithm
  - Rasterizing triangles
    - Interpolation by barycentric coordinates
- Mean value coordinates
- Dividing polygons into triangles

Rasterizing Lines

Converting a continuous object in the world into a discrete object in the computer
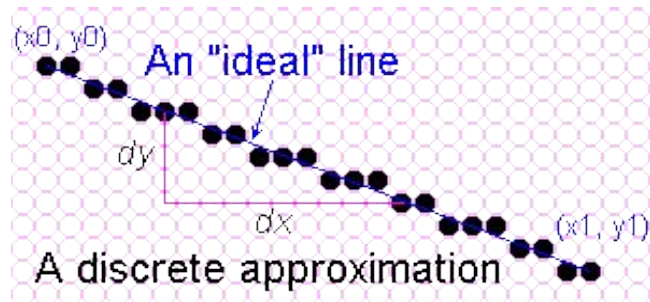
We need to lit the pixels instead of drawing a continuous line

Simple Line

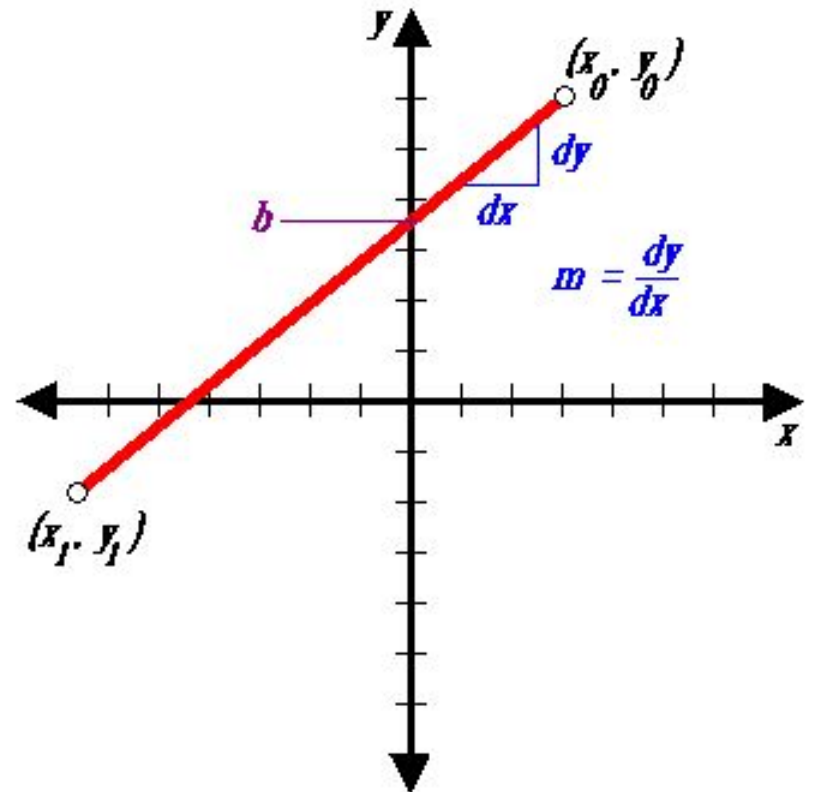From linear algebra

$$y = m \, x + n$$

Simple approach:

   increment x, calculate y

Then cast y to an integer
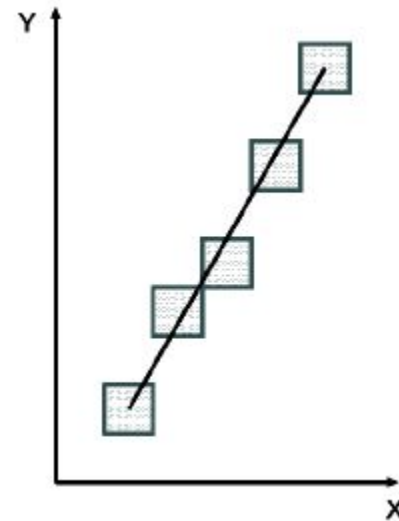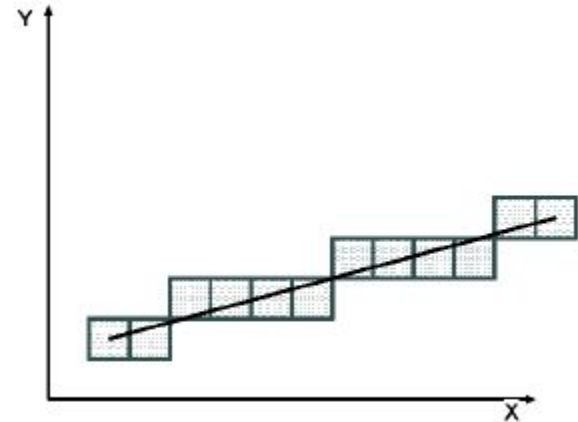
(x, (int)y)

Will this work?

Does it Work?

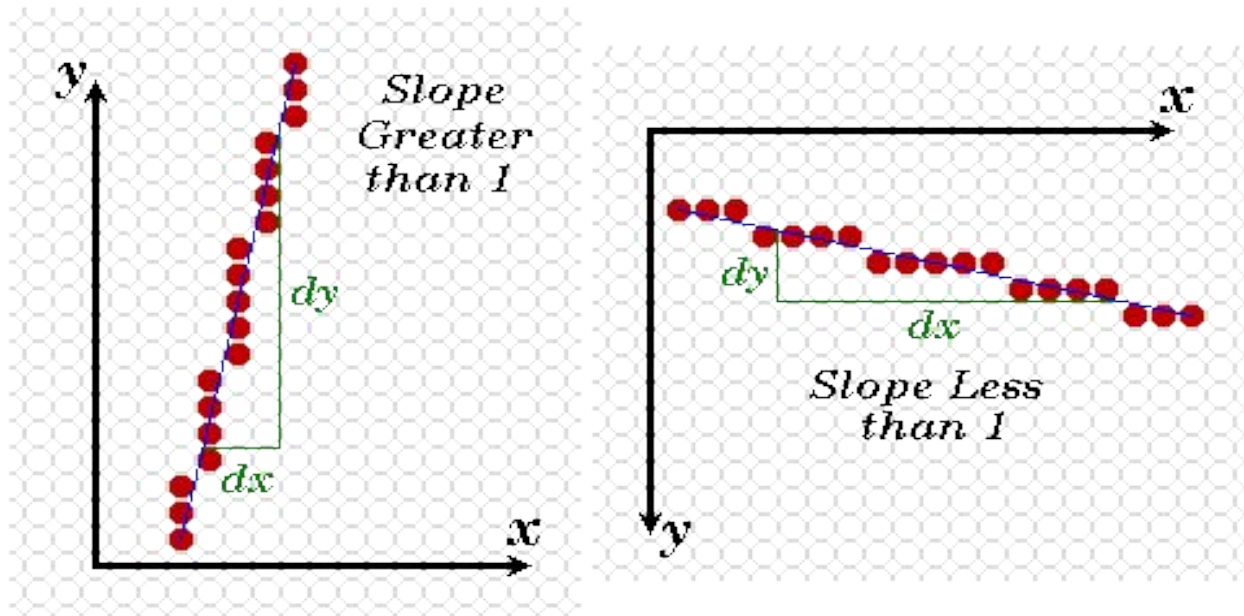It seems to work okay for lines with a slope of 1 or less,

But doesn't work well for lines with slope greater than 1

- Lines become more discontinuous in appearance
- We must add more than 1 pixel per column to make it work.

5

*

Use Symmetry

Rotate and Rename coordinate axes

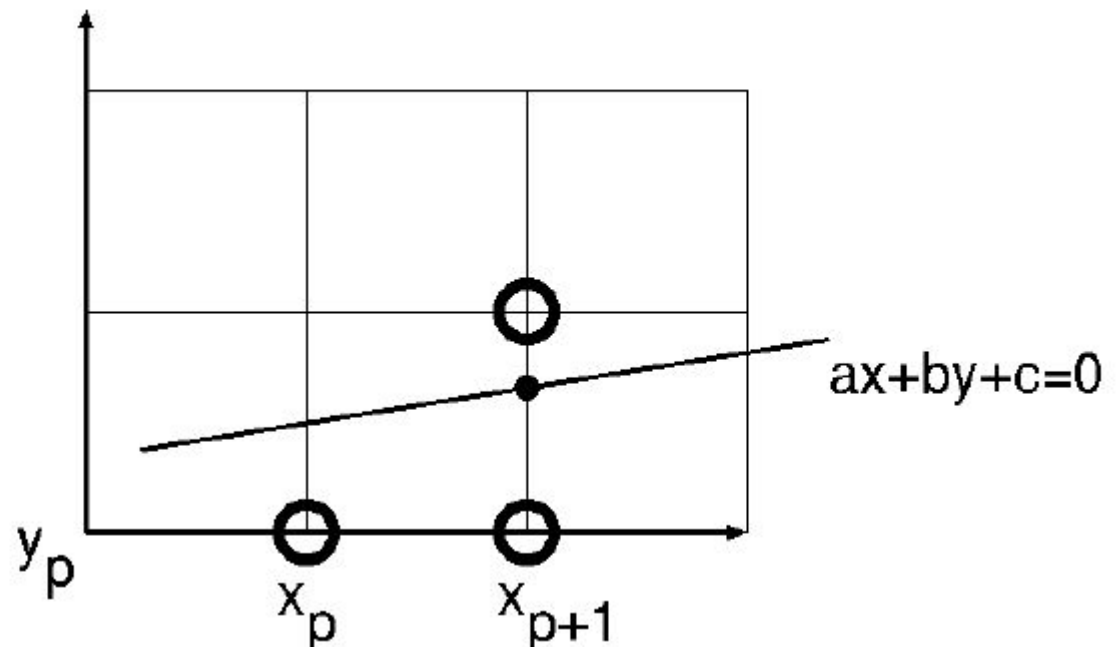Increment along x-axis if dy<dx else increment along y-axis

.But still we need to do a lot of floating point arithmetic!

*

Midpoint Algorithm: Overview

- At step p, assume pixel (x_p, y_p) was lit

- Check where the line intersects with $x = x_{p+1}$

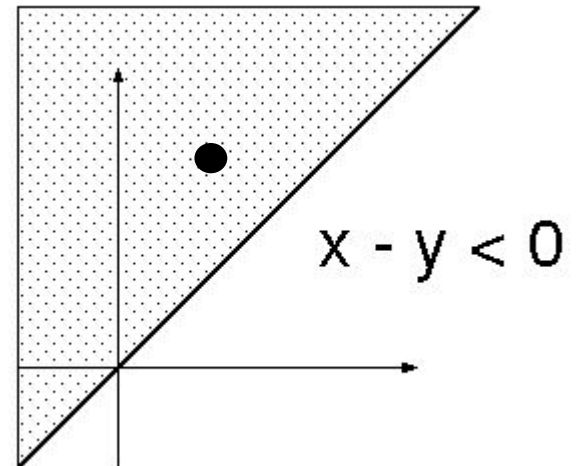- Lit (x_{p+1}, y_p) or $(x_{p+1}, y_p + 1)$, whichever is closer to the intersection

Testing above or below a line

$$F(x,y) = ax + by + c = 0.$$

If $b < 0$, $F > 0$ if the line is above the point $F < 0$ if the line is below the point

x - y < 0

Testing for the side of a line.
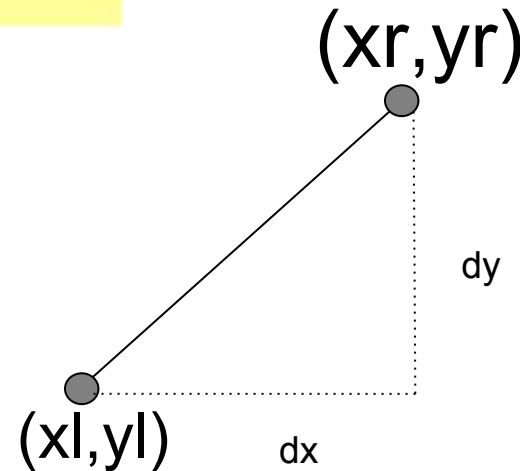
- Assume the line is connecting $(x_l, y_l), (x_r, y_r)$

- The slope will be $\frac{dy}{dx}$ where $dx = x_r - x_l,$

$dy = y_r - y_l$

$$y = mx + n \text{ and so } y = \frac{dy}{dx}x + n$$

(xr,yr)

$$F(x, y) = ax + by + c = 0$$

dy

$$F(x, y) = dy.x - dx.y + c = 0$$

(xl,yl)

dx

Decision variable.

Let's assume dy/dx < 1 (we can use symmetry)

Evaluate F at point M

$$d = F(x_p + 1, y_p + \frac{1}{2}) = a(x_p + 1) + b(y_p + \frac{1}{2}) + c$$

Referred to as *decision variable*   If d ≤ 0, select E as the next pixel
otherwise select NE as the next pixel

NE

M

E

Previous
Pixel
(x_p, y_p)
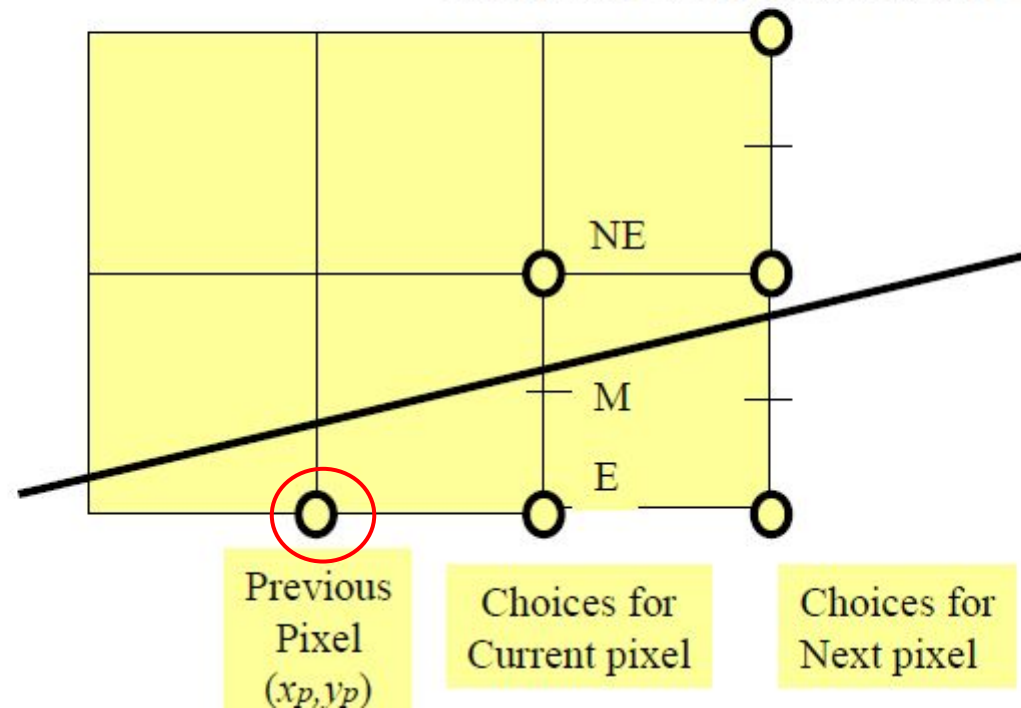
Choices for
Current pixel

Choices for
Next pixel

Decision variable.

Let's assume dy/dx < 1 (we can use symmetry)

Evaluate F at point M

$$d = F(x_p + 1, y_p + \frac{1}{2}) = a(x_p + 1) + b(y_p + \frac{1}{2}) + c$$

Referred to as *decision variable*   If $d \leq 0$, select E as the next pixel
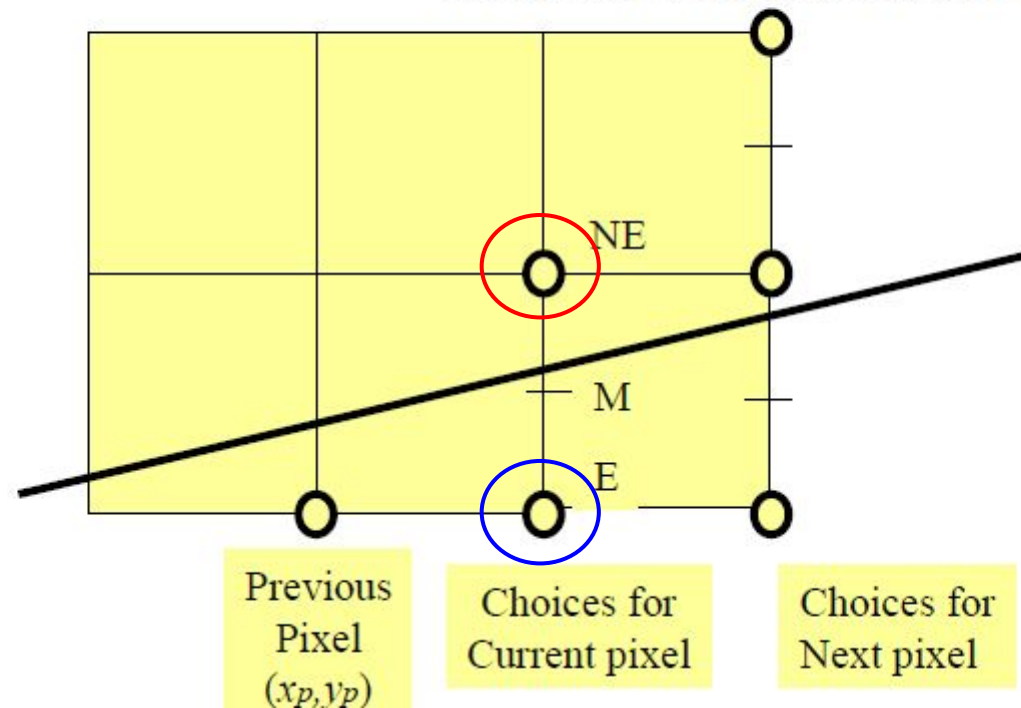otherwise select NE as the next pixel

Previous
Pixel
(x_p, y_p)
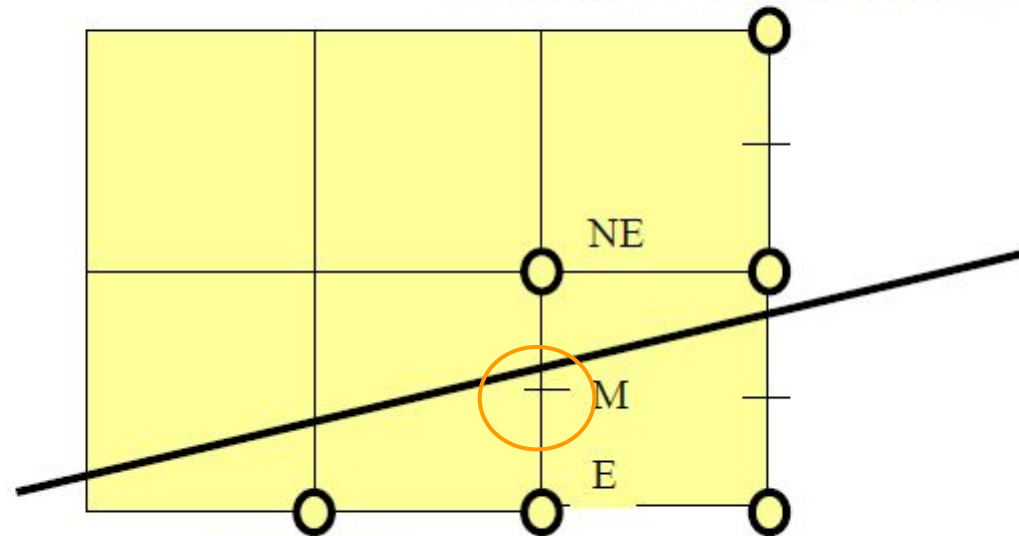
Choices for
Current pixel

Choices for
Next pixel

Decision variable.

Let's assume dy/dx < 1 (we can use symmetry)

Evaluate F at point M

$$d = F(x_p + 1, y_p + \frac{1}{2}) = a(x_p + 1) + b(y_p + \frac{1}{2}) + c$$

Referred to as *decision variable*   If d ≤ 0, select E as the next pixel
otherwise select NE as the next pixel

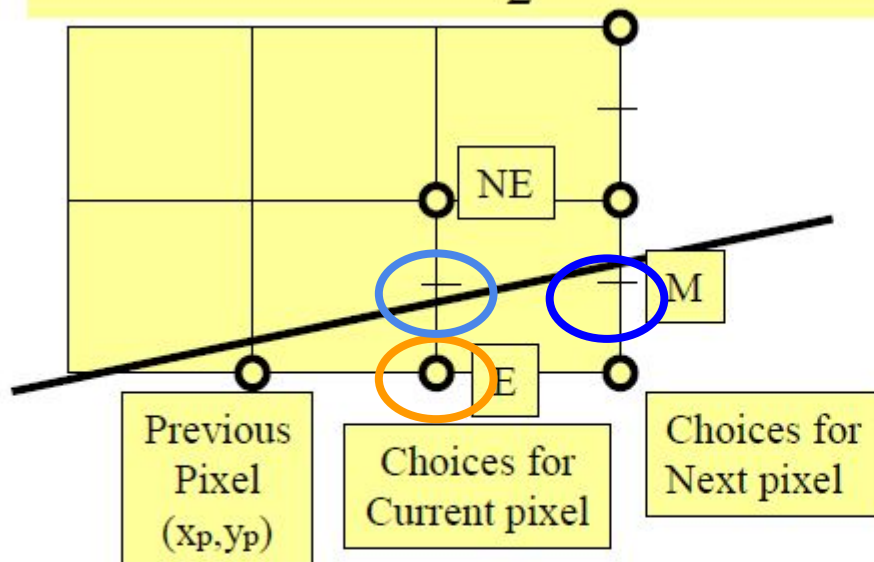

Previous
Pixel
(x_p, y_p)

Choices for
Current pixel

Choices for
Next pixel

Updating the decision variable

Evaluate d for next pixel, Depends on whether E or NE Is chosen :

If E chosen :

$$d_{new} = F(x_p + 2, y_p + \frac{1}{2}) = a(x_p + 2) + b(y_p + \frac{1}{2}) + c$$

But recall :

$$d_{old} = F(x_p + 1, y_p + \frac{1}{2})$$
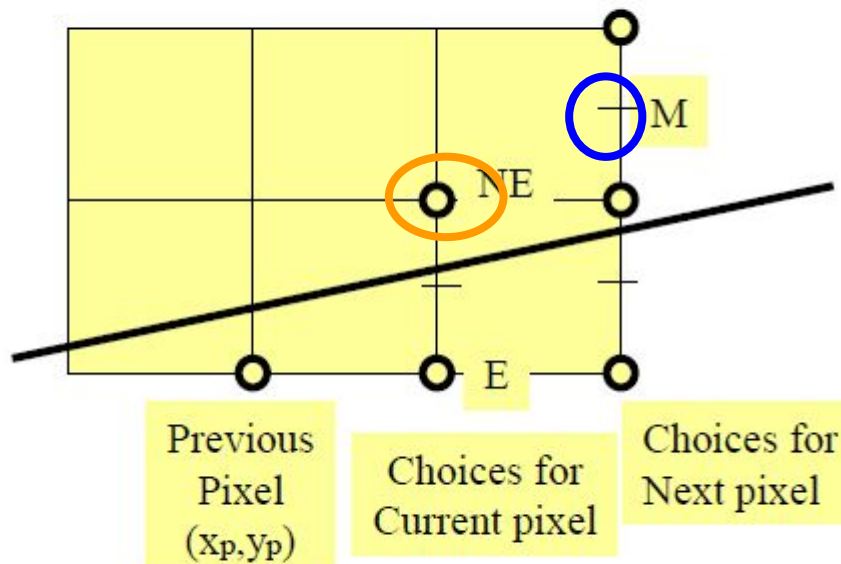
$$= a(x_p + 1) + b(y_p + \frac{1}{2}) + c$$

So :

$$d_{new} = d_{old} + a$$

$$= d_{old} + dy$$

NE

M

E

Previous
Pixel
(xp,yp)

Choices for
Current pixel

Choices for
Next pixel

14

Decision variable.

If NE was chosen :

$$d_{new} = F(x_p + 2, y_p + \frac{3}{2}) = a(x_p + 2) + b(y_p + \frac{3}{2}) + c$$

Previous Pixel (x_p, y_p)
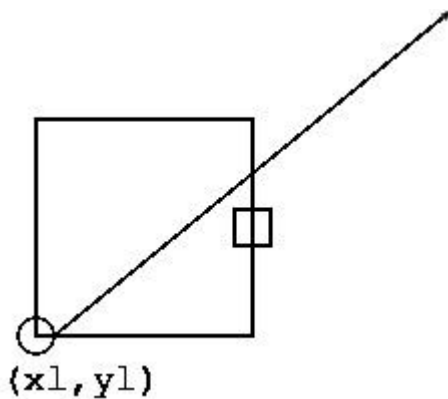
Choices for Current pixel

Choices for Next pixel

But recall :

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

$$= a(x_p + 1) + b(y_p + \frac{1}{2}) + c$$

So :

$$d_{new} = d_{old} + a + b$$

$$= d_{old} + dy - dx$$

15

Initial value of d.

Start point is (x_1, y_1)

(x1,y1)

$$d_{start} = F(x_1 + 1, y_1 + \frac{1}{2}) = a(x_1 + 1) + b(y_1 + \frac{1}{2}) + c$$

$$= ax_1 + by_1 + c + a + \frac{b}{2}$$

$$= F(x_1, y_1) + a + \frac{b}{2}$$
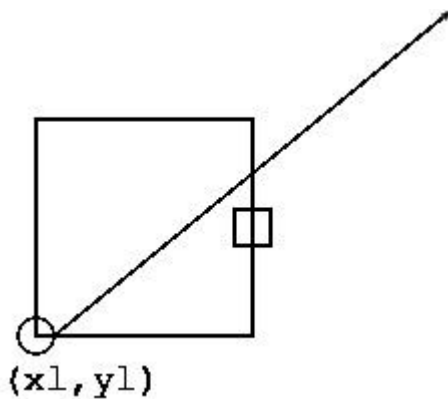
But (x_1, y_1) is a point on the line, so $F(x_1, y_1) = 0$

$$d_{start} = dy - dx / 2$$

Initial value of d.

Start point is (x_1, y_1)

(x1, y1)

$$d_{start} = F(x_1 + 1, y_1 + \frac{1}{2}) = a(x_1 + 1) + b(y_1 + \frac{1}{2}) + c$$

$$= ax_1 + by_1 + c + a + \frac{b}{2}$$

$$= F(x_1, y_1) + a + \frac{b}{2}$$

But (x_1, y_1) is a point on the line, so $F(x_1, y_1) = 0$

$$d_{start} = dy - dx/2$$

Multiply by 2 to avoid floating point operations
(We are only using the sign of the decision variable)

Decision variable : summary

- New equation for decision variable

$$d = F(x, y) = 2(ax + by + c)$$

- Initialization

$$d_{start} = 2dy - dx$$

- If E point is selected

$$d_{new} = d_{old} + 2dy$$

- If NE point is selected

$$d_{new} = d_{old} + 2dy - 2dx$$

Only requires integer operations

Summary of mid-point algorithm

- Start point is simply first endpoint (x_l, y_l).
- Calculate the initial value for d
- Choose between 2 pixels at each step based upon the sign of a decision variable.
- Update the decision variable based upon which pixel is chosen.

●**No floating point arithmetic needed**

Midpoint algorithm

```
void MidpointLine(int
x1,y1,x2,y2)
   {
   int dx=x2-x1;
   int dy=y2-y1;
   int d=2*dy-dx;
   int increE=2*dy;
   int incrNE=2*(dy-dx);
   x=x1;
   y=y1;
   WritePixel(x,y);
```

```
while (x < x2) {
    if (d<= 0) {
        d+=incrE;
        x++
    } else {
        d+=incrNE;
        x++;
        y++;
    }
    WritePixel(x,y);
}
}
```
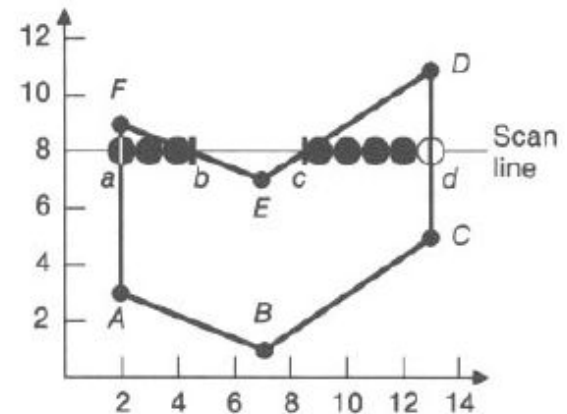
*

Overview

Rasterization
- Line Rasterization
- **Polygon Rasterization**
  - Scanline Algorithm
  - Rasterizing triangles
    - Interpolation by barycentric coordinates
- Mean value coordinates
- Dividing polygons into triangles

Scanline algorithm

A traditional approach of rasterization
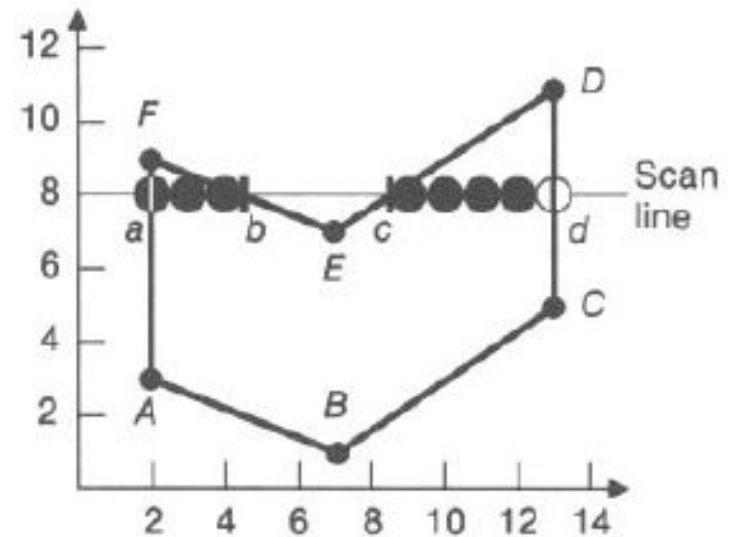Filling in the pixels within the polygon along
the scan line

Scanline algorithm

For each scan line:

1.Find the intersections of the scan line with all edges of the polygon.
2.Sort the intersections in the increasing order of the x coordinate.
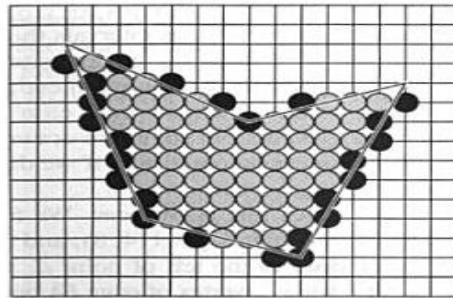3.Fill in all pixels between pairs of intersections.
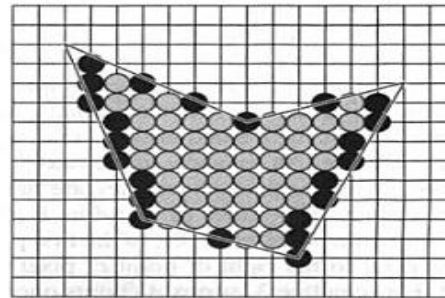
Can also deal with concave polygons

Span extrema

Only turn on pixels whose centers are *interior* to the polygon:
Otherwise will intrude other adjacent polygons
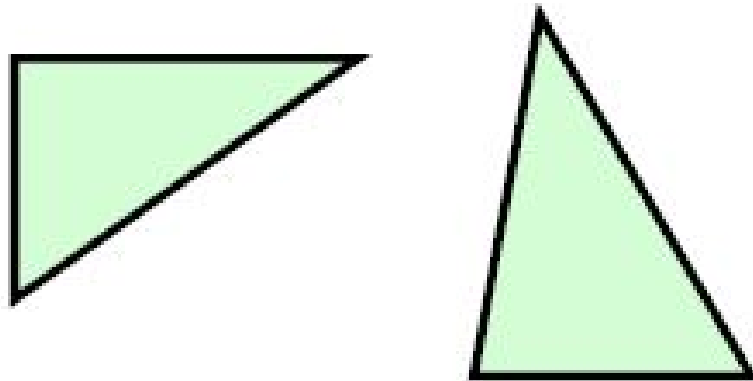round up values on the left edge of a span, round down on the right edge

(a)   (b)

Scanline Algorithm
Cons and Pros

- Simple

- Very serial (cannot be efficiently parallelized)
- Special cases require exception handling

• sliver: not even a single pixel wide

Triangle Rasterization by Barycentric Coordinates

Barycentric coordinates
- Can check whether a pixel is inside / outside the triangle
- Can interpolate the attributes at the vertices
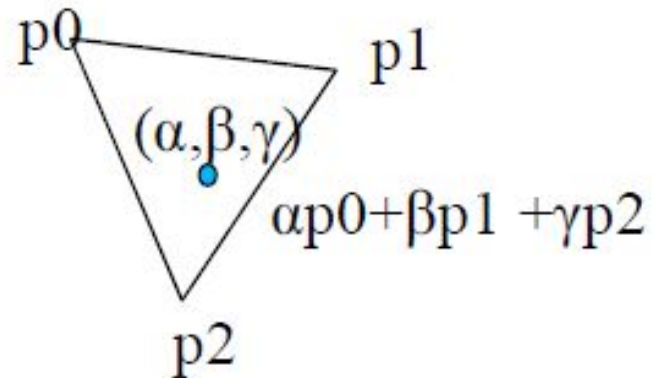- Often used in modern graphics cards
- Can be easily parallelized

Triangle Rasterization

Consider a 2D triangle with vertices p_0, p_1, p_2.
Let p be any point in the plane, it can be expressed by

$$p = p_0 + \beta(p_1 - p_0) + \gamma(p_2 - p_0)$$
$$= (1 - \beta - \gamma)p_0 + \beta p_1 + \gamma p_2$$
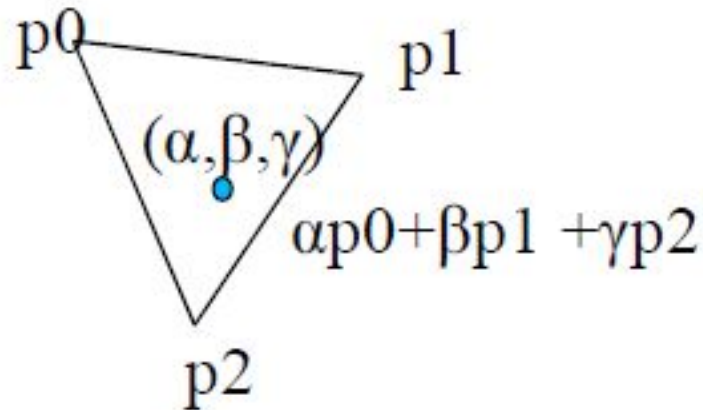$$= \alpha p_0 + \beta p_1 + \gamma p_2$$

$$\alpha + \beta + \gamma = 1, \alpha, \beta, \gamma \in \mathcal{R}$$

Barycentric Coordinates

We will have $\alpha, \beta, \gamma \in [0,1]$ if and only if $\boldsymbol{p}$ is inside the triangle.

We call the α, β, γ the **barycentric coordinates** of $\boldsymbol{p}$.

Computing Barycentric Coordinates

The triangle is composed of 3 points

$$p_0\ (x0,y0),\ p_1\ (x1,\ y1),\ p_2(x2,y2).$$

For point (x,y), its barycentric coordinates can be computed by

where

$$\alpha = \frac{f_{12}(x,y)}{f_{12}(x_0,y_0)} \quad \beta = \frac{f_{20}(x,y)}{f_{20}(x_1,y_1)} \quad \gamma = \frac{f_{01}(x,y)}{f_{01}(x_2,y_2)}$$

$$f_{ab}(x,y) \overset{\text{def}}{=} (y_a - y_b)\,x + (x_b - x_a)\,y + x_a y_b - x_b y_a$$

for $a,b \in \{0,1,2\}$.

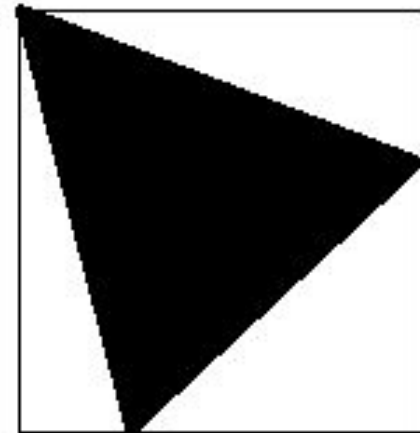http://www.vis.uni-stuttgart.de/~kraus/LiveGraphics3D/cagd/chap3fig5.html

Bounding Box of a Triangle

Calculate a tight bounding box for a triangle: simply calculate pixel coordinates for each vertex, and find the minimum/maximum for each axis
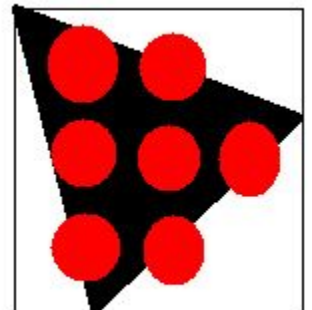min (x_0, x_1, x_2), max (x_0, x_1, x_2)
min (y_0, y_1, y_2), max (y_0, y_1, y_2)

Scanning inside the triangle

- For each pixel, compute the barycentric coordinates
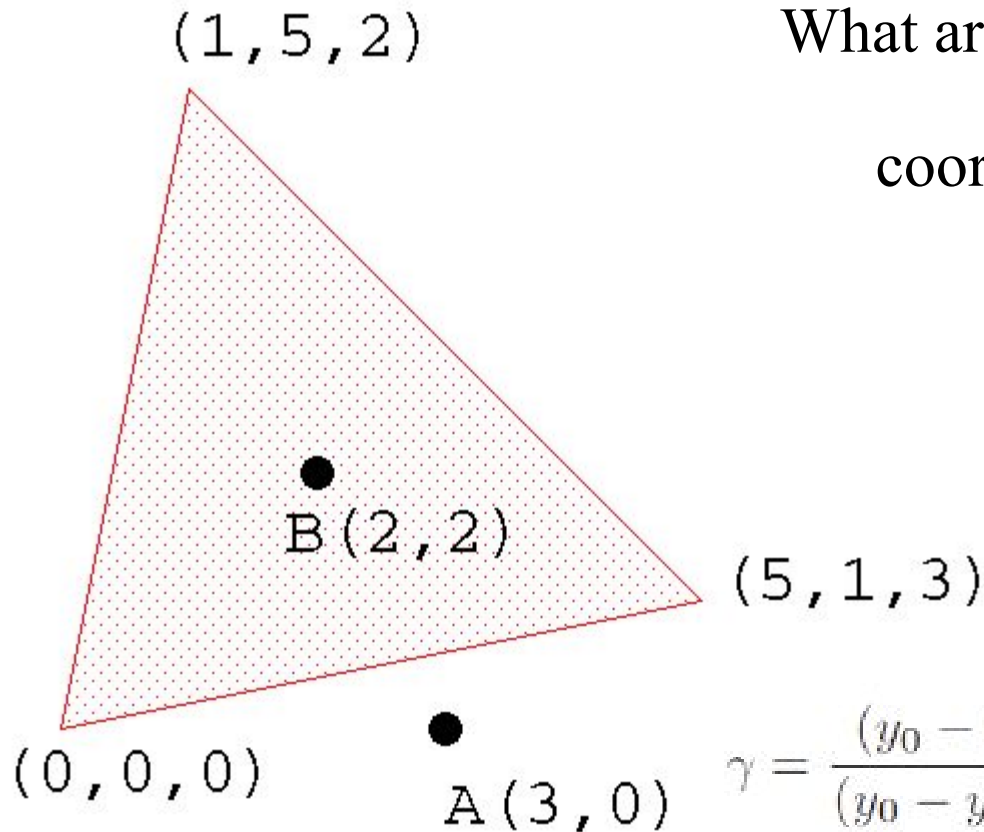- Color it if all the three values are in the range of [0,1]

Exercise
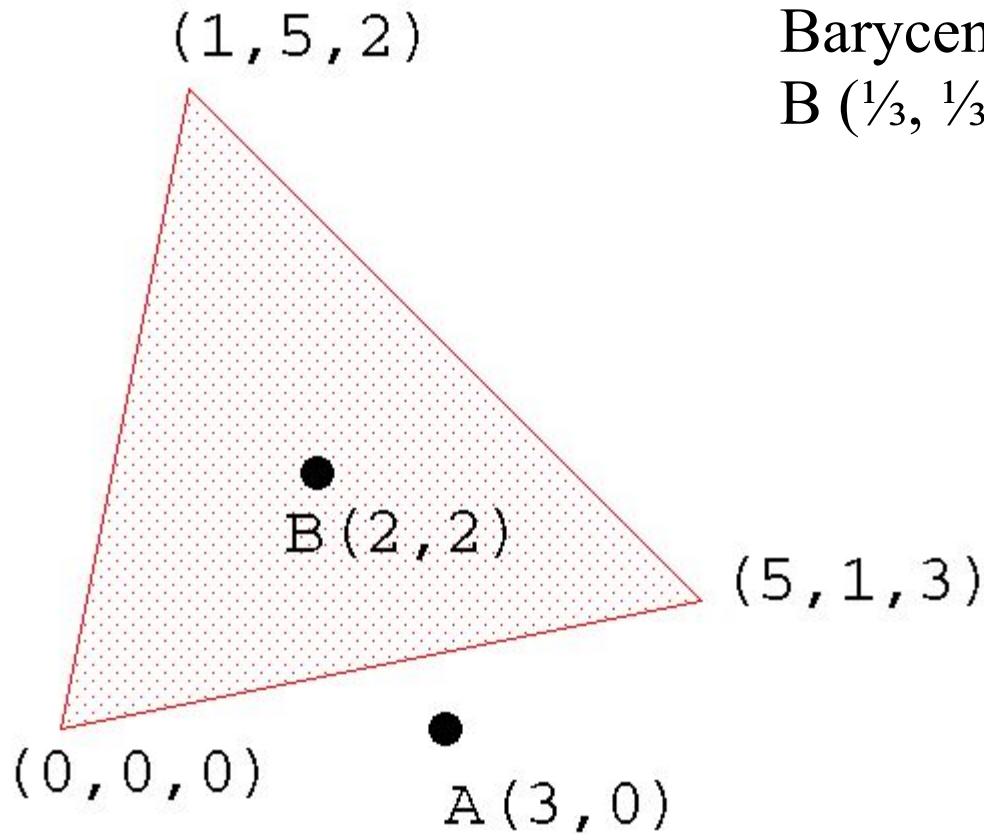
What are the barycentric coordinates at point A and B?

(1,5,2)

B(2,2)

(5,1,3)

(0,0,0)

A(3,0)

$$\gamma = \frac{(y_0 - y_1)x + (x_1 - x_0)y + x_0 y_1 - x_1 y_0}{(y_0 - y_1)x_2 + (x_1 - x_0)y_2 + x_0 y_1 - x_1 y_0}$$

$$\beta = \frac{(y_0 - y_2)x + (x_2 - x_0)y + x_0 y_2 - x_2 y_0}{(y_0 - y_2)x_2 + (x_2 - x_0)y_2 + x_0 y_2 - x_2 y_0}$$
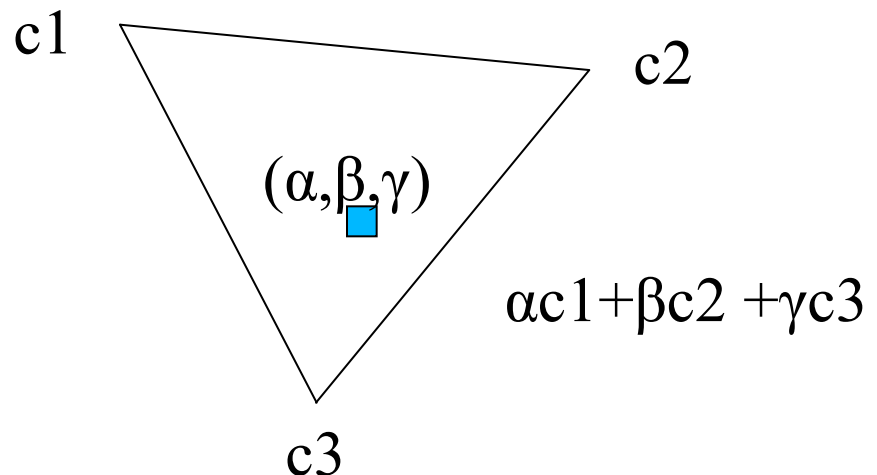
Exercise

(1,5,2)

Barycentric coordinates
B ($\frac{1}{3}$, $\frac{1}{3}$, $\frac{1}{3}$),   A($\frac{1}{2}$, $\frac{5}{8}$, -$\frac{1}{8}$)

B(2,2)

(5,1,3)

(0,0,0)

A(3,0)

Interpolation by Barycentric Coordinates

We can use the barycentric coordinates to interpolate attributes of the triangle vertices
- color, depth, normal vectors, texture coordinates

c1

c2

(α,β,γ)

$\alpha c1 + \beta c2 + \gamma c3$

c3

Interpolation of Color

- We can compute the color at the vertices (computed using the lighting equation) and interpolate the color on the surface of the triangle

- This is called Gouraud shading

c1

c2

(α,β,γ)

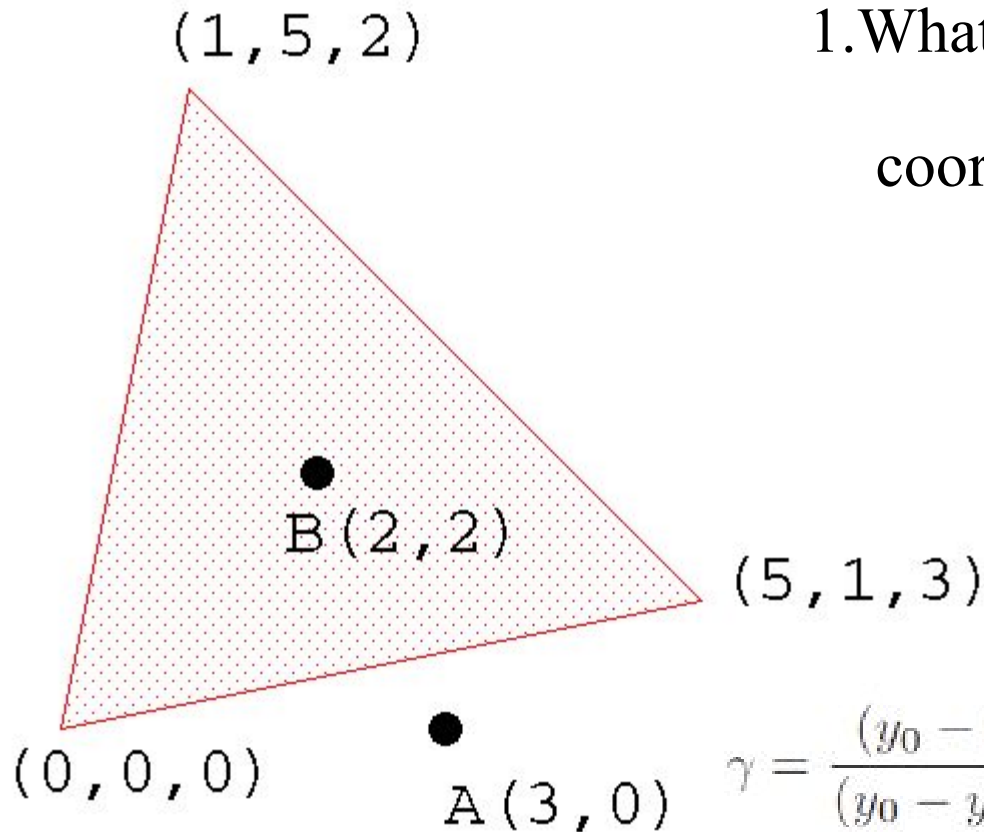$\alpha c1 + \beta c2 + \gamma c3$

c3
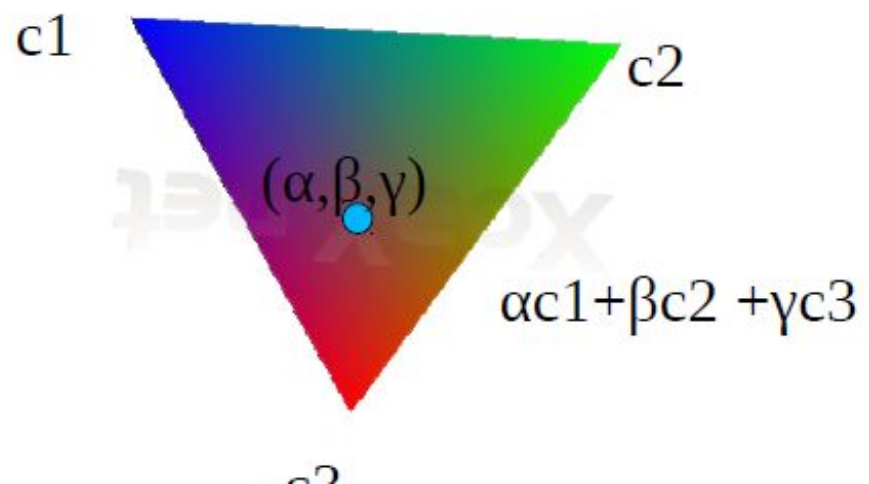
Interpolation of Depth

- •When triangles are overlapped, need to compute the depth at each pixel
- •Can be computed by barycentric coordinates
- •Compare the depth of the pixel at different triangles and only show the closest one
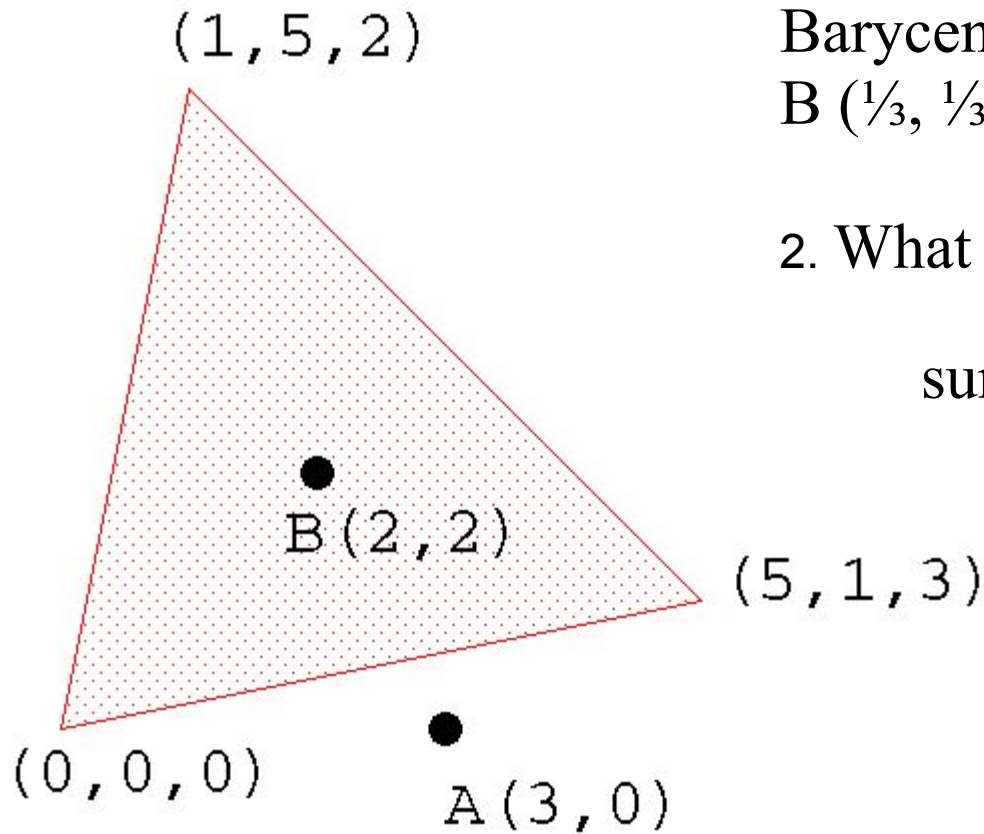- •This is called Z-buffering

d1   d2'

d1' (α,β,γ) d2

$\alpha'd1'+\beta'd2'+\gamma'd3'$

$\alpha\ d1+\beta\ d2+\gamma\ d3$

d3   d3'
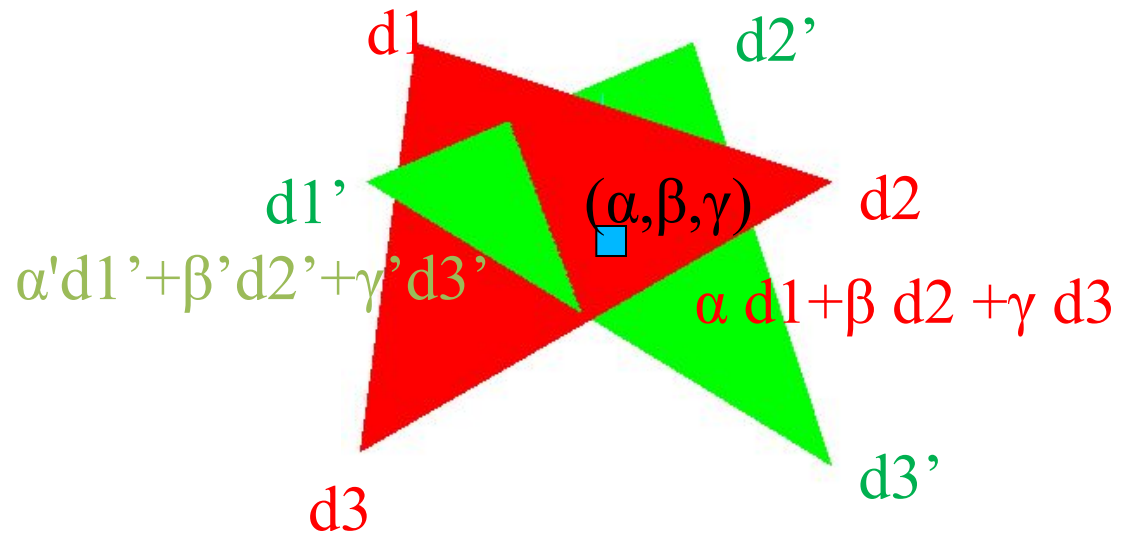
Exercise

1. What are the barycentric coordinates at point A and B?

$(1,5,2)$

$B(2,2)$

$(5,1,3)$

$(0,0,0)$

$A(3,0)$

$$\gamma = \frac{(y_0 - y_1)x + (x_1 - x_0)y + x_0 y_1 - x_1 y_0}{(y_0 - y_1)x_2 + (x_1 - x_0)y_2 + x_0 y_1 - x_1 y_0}$$

$$\beta = \frac{(y_0 - y_2)x + (x_2 - x_0)y + x_0 y_2 - x_2 y_0}{(y_0 - y_2)x_2 + (x_2 - x_0)y_2 + x_0 y_2 - x_2 y_0}$$
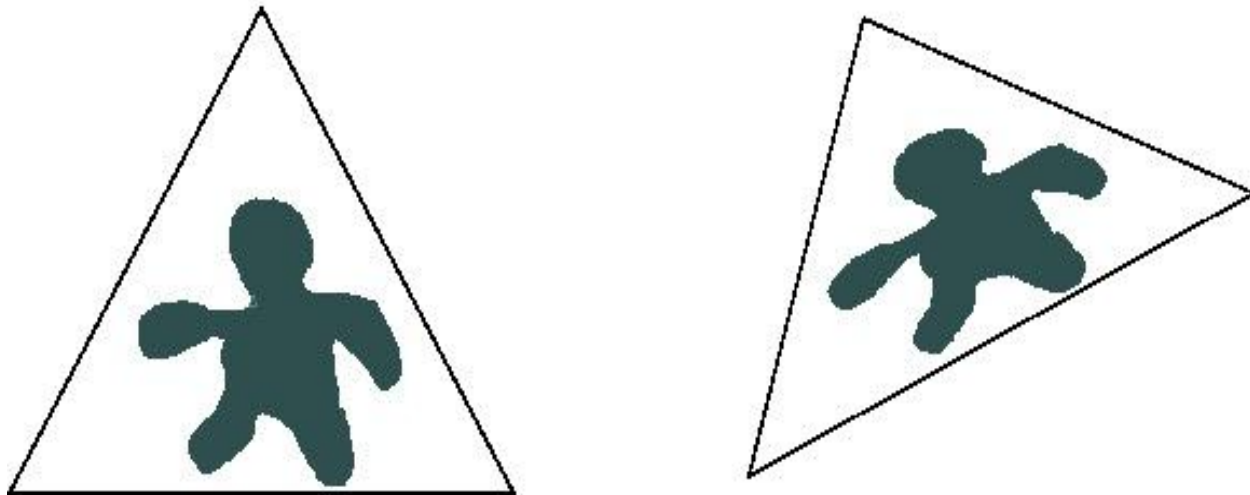
Exercise

(1,5,2)

Barycentric coordinates
B (⅓, ⅓, ⅓),   A(½, ⅝, -⅛)

2. What is the depth of the triangle

surface at point B?

B(2,2)

(5,1,3)

(0,0,0)

A(3,0)

Another usage : Shape Editing

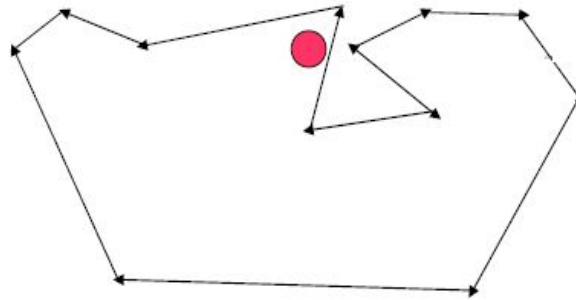We can re-apply the same barycentric coordinates within a triangle when its shape is edited
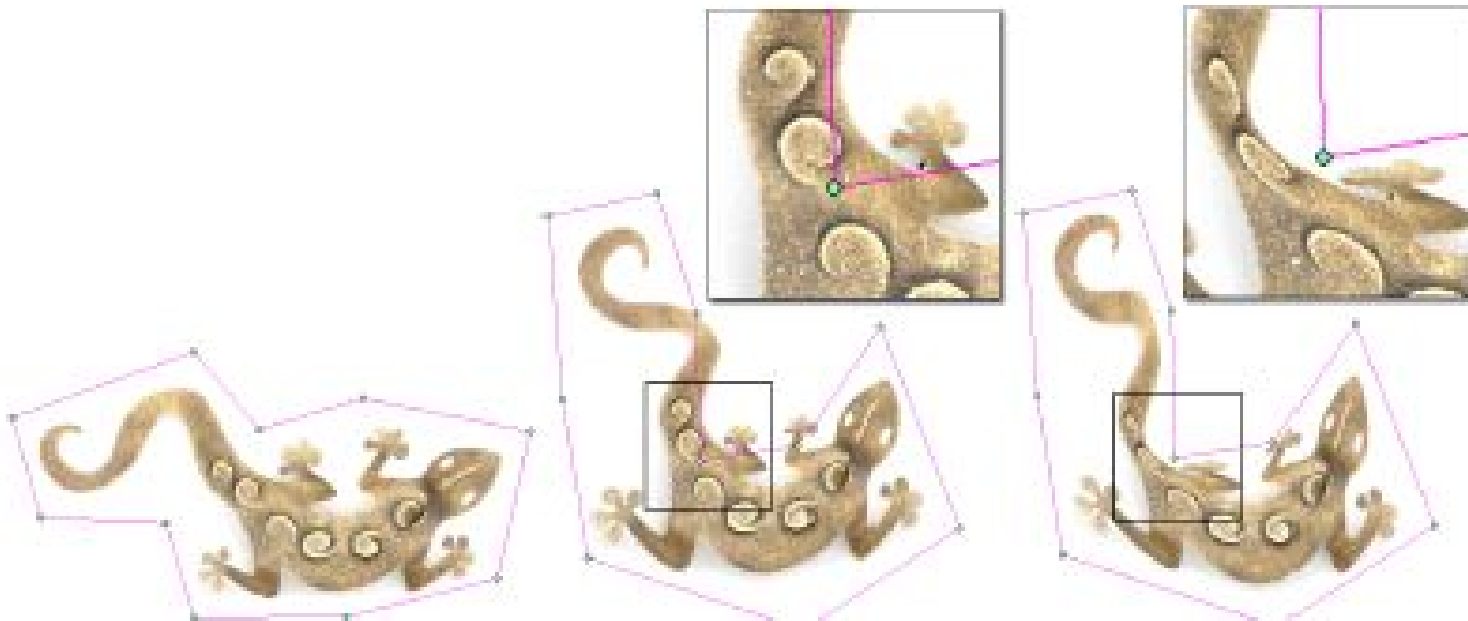
What about polygons with many vertices?

•Can we compute barycentric coordinates for polygons with more vertices?

$$v = \frac{\sum_i w_i p_i}{\sum_i w_i}$$

•Can we compute barycentric coordinates for 3D meshes?

-> Mean value coordinates
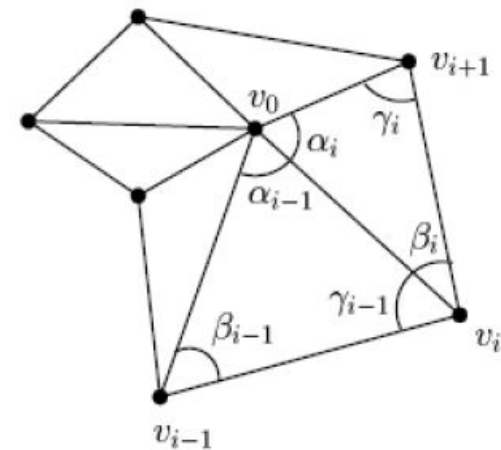Harmonic coordinates
(generalized barycentric coordinates)

Mean Value Coordinates (MVC)

- A good and smooth barycentric coordinates that can
  - smoothly interpolate the boundary values
  - Also works with concave polygons
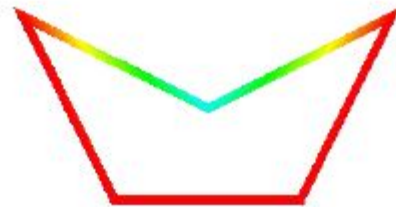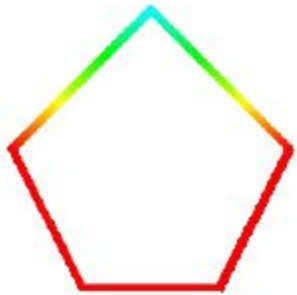 - There is also a 3D version

For a point v0 inside the polygon, the MVC for vertex vi is

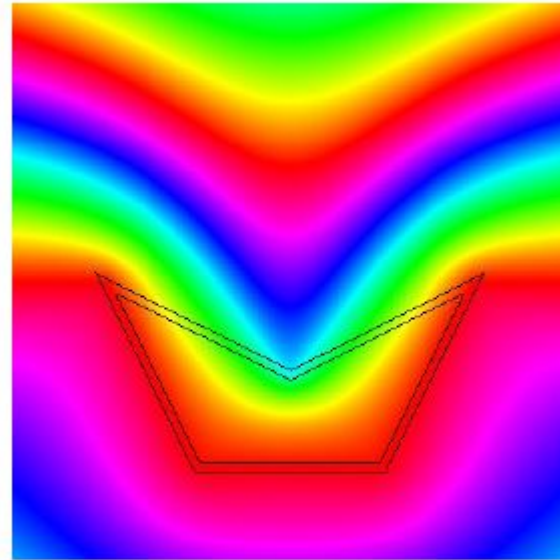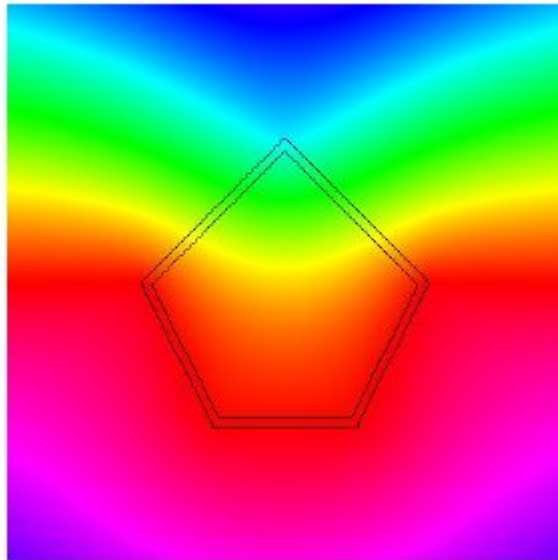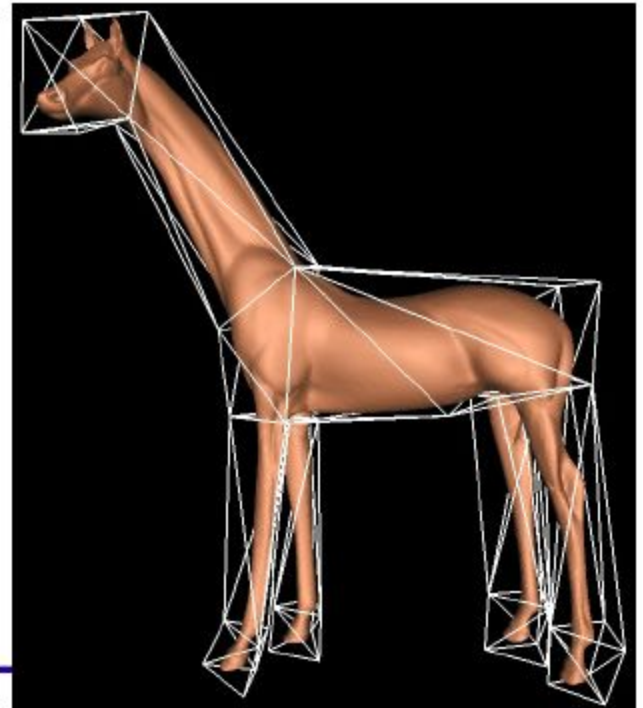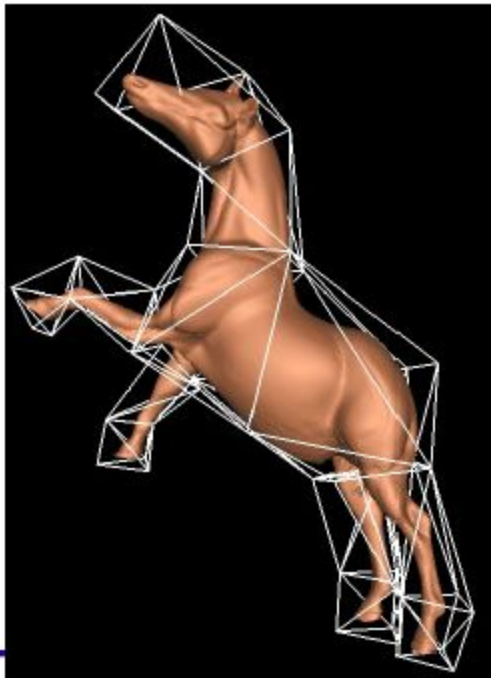$$w_i = \frac{\tan(\alpha_{i-1}/2) + \tan(\alpha_i/2)}{||v_i - v_0||}$$
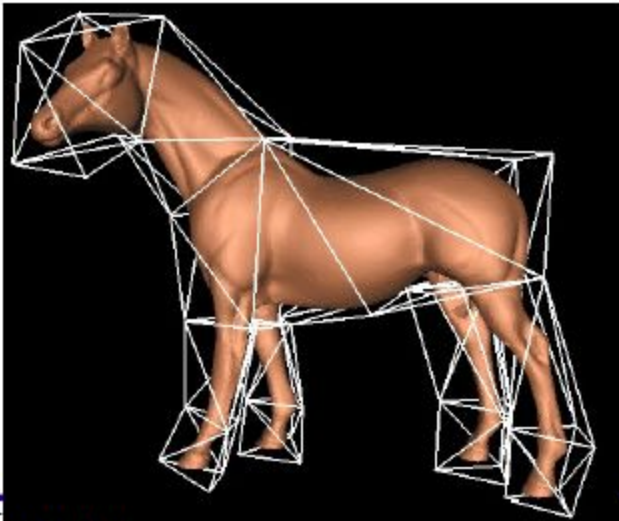
Mean Value Coordinates

- Can interpolate convex and concave polygons
- Smoothly interpolate the interior as well as exterior

Mean Value Coordinates

- Can interpolate convex and concave polygons
- Smoothly interpolate the interior as well as exterior

Mean Value Coordinates

- Can be computed in 3D
- Applicable for mesh editing

Overview

Rasterization
- Line Rasterization
- Polygon Rasterization
  - Scanline Algorithm
  - Rasterizing triangles
    - Edge walking
    - Interpolation by barycentric coordinates
- Mean value coordinates
- **Dividing polygons into triangles**

Polygon Decomposition

For polygons with more than three vertices, we usually decompose them into triangles
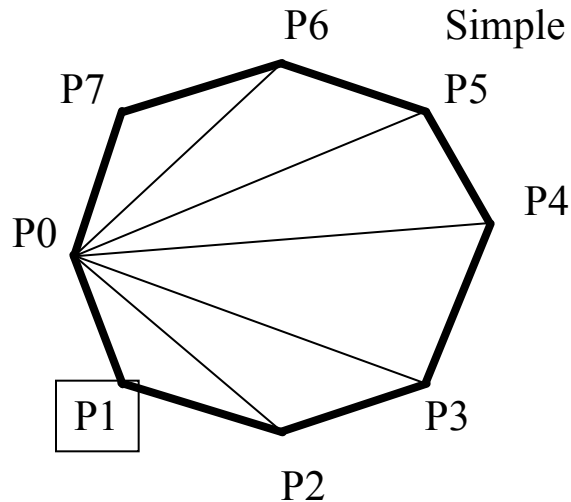

Simple for convex polygons.
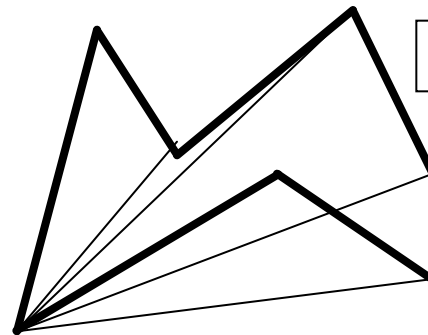
Concave more difficult.
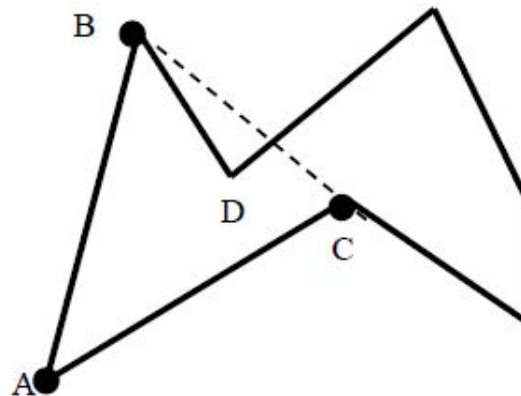
Polygon Decomposition: Algorithm

Start from the left and form the leftmost triangle:

- Find leftmost vertex (smallest x) – A

- Compose possible triangle out of A and the two adjacent vertices B and C

- Check to ensure that no other polygon point P is inside of triangle ABC

- If all other polygon points are outside of ABC then cut it off from polygon and proceed with next leftmost triangle
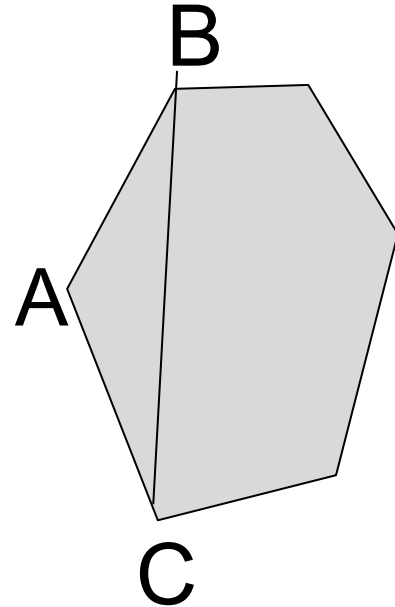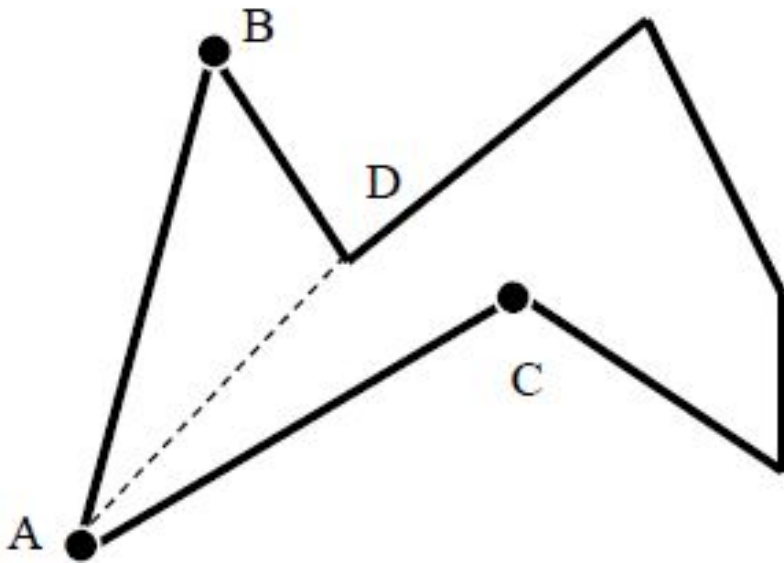
Polygon decomposition (2)

- The left most vertex A
- A triangle is formed by A and the two adjacent B and C
- Check if all the other vertices are outside the triangle

Vertex 'D' fails test.

Polygon decomposition (3)

If a vertex is inside, split the polygon by the inside vertex and point A, proceed as before.
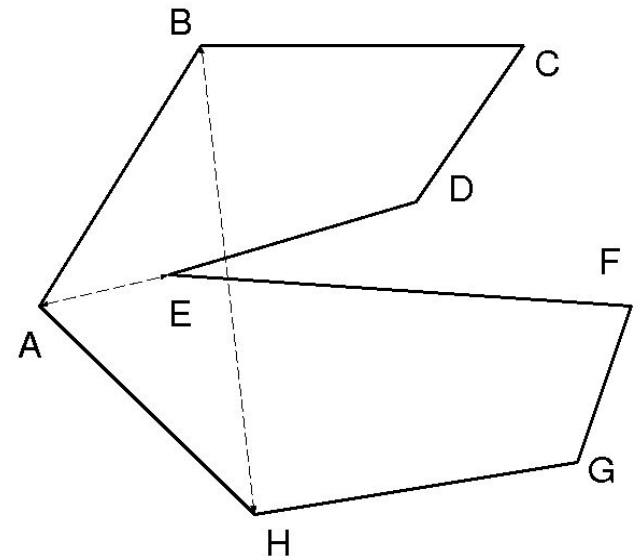

Test ABD in same manner as before,

Polygon decomposition (4)

This edge may split the polygon into two.
Then, recurse the method with each polygon

(ABCDE and AEFGH in the example below)

Summary

Rasterization

- Line Rasterization - Midpoint algorithm
- Triangle Rasterization - barycentric coordinates
- Mean value coordinates
- Dividing polygons into triangles

Reading for rasterization

Midpoint and scanline algorithm:
Foley Chapter 3.2, 3.5, 3.6

Barycentric coordinates:
Shirley Chapter 2.7

Mean value coordinates:
Floater, M. S. Mean value coordinates. Computer Aided Geometric Design, 20(1), 19–27, 2003
Ju, T., Schaefer, S., & Warren, J. Mean value coordinates for closed triangular meshes. ACM Transactions on Graphics, 24(3), 561–566, 2005.
Polygon decomposition:
http://www.siggraph.org/education/materials/HyperGraph/scanline/outprims/polygon1.htm