Computer Graphics

Lecture 6

Rasterization

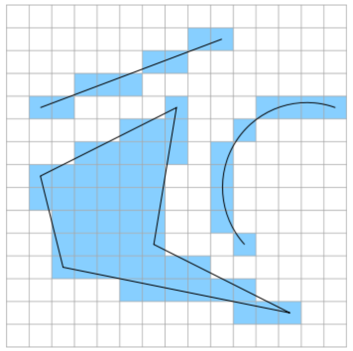
Taku Komura

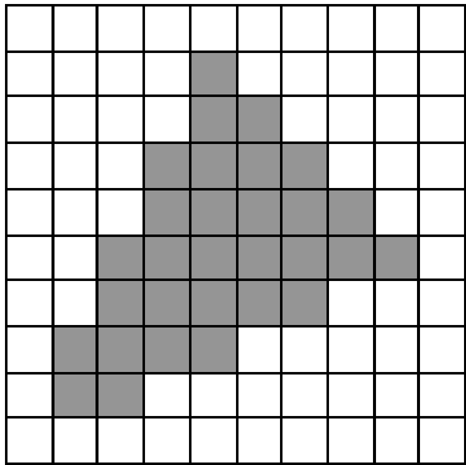
Rasterization

- 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)
- We have done this for lines already

(in lecture 4)

Let’s do it for polygons now





Overview

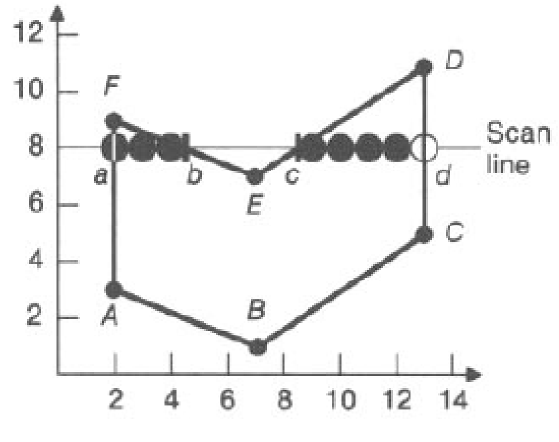
Rasterization

- Scanline algorithm
- Rasterizing triangles
  - Interpolation by barycentric coordinates
- Mean value coordinates
- Dividing polygons into triangles

Scanline algorithm

A traditional approach of rasterization

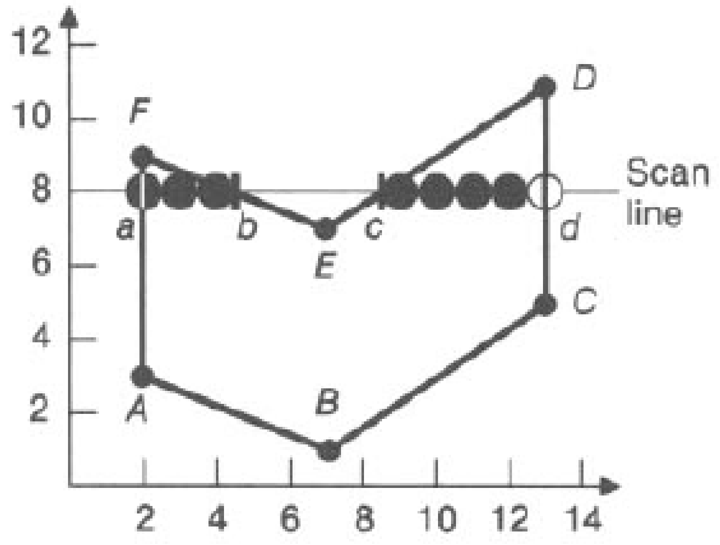
Filling in the pixels within the polygon along the scan line



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

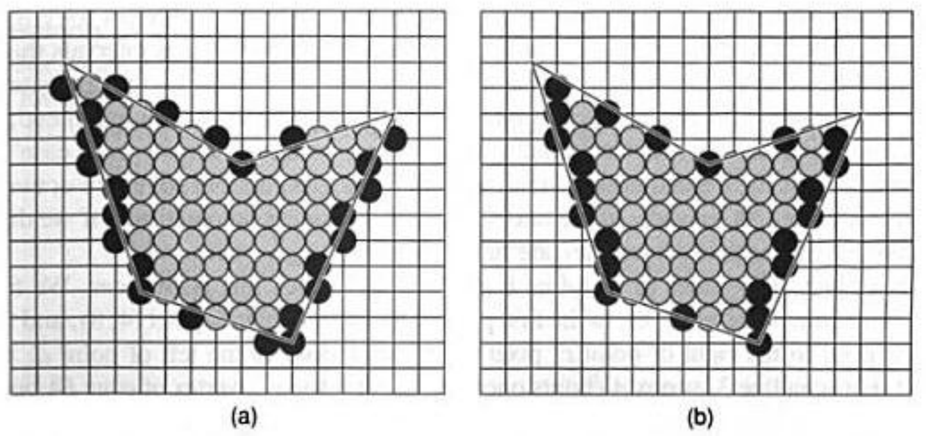
Span extrema

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,

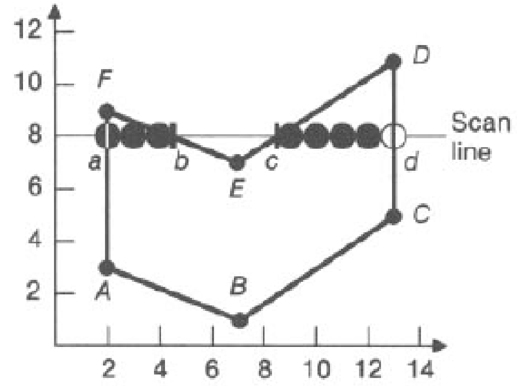
round down on the right edge



midpoint algorithm interior

Computing the Interaction of Scan Lines and Edges

- Computing the intersection of lines is expensive
- Also requires floating point arithmatic
- Make use of edge coherence;
  - Edges that intersected the previous scan line are likely to intersect the next scan line



Edge Coherence

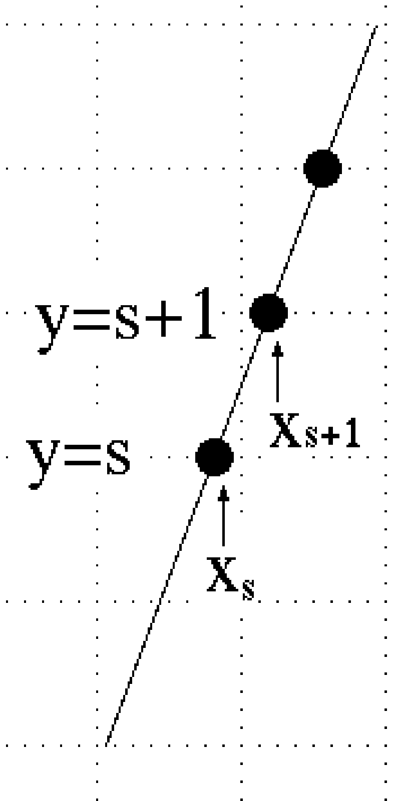
Use a method similar to the midpoint algorithm

$$y = mx + b, \qquad x = \frac{y-b}{m}$$

At $y = s,\ x_s = \frac{s-b}{m}$

At $y = s+1,\ x_{s+1} = \frac{s+1-b}{m} = x_s + \frac{1}{m}$

Incremental calculation: $x_{s+1} = x_s + \frac{1}{m}$



How to avoid floating point arithmatic?

- $x_{s+1} = x_s + \frac{1}{m} = x_s + \frac{x_{max} - x_{min}}{y_{max} - y_{min}}$
- We can keep track of the integer part of x_s (defined as X) as well as the numerator part (defined as N)
- Increase the X every time there is an overflow (N > $y_{max} - y_{min}$)

Example

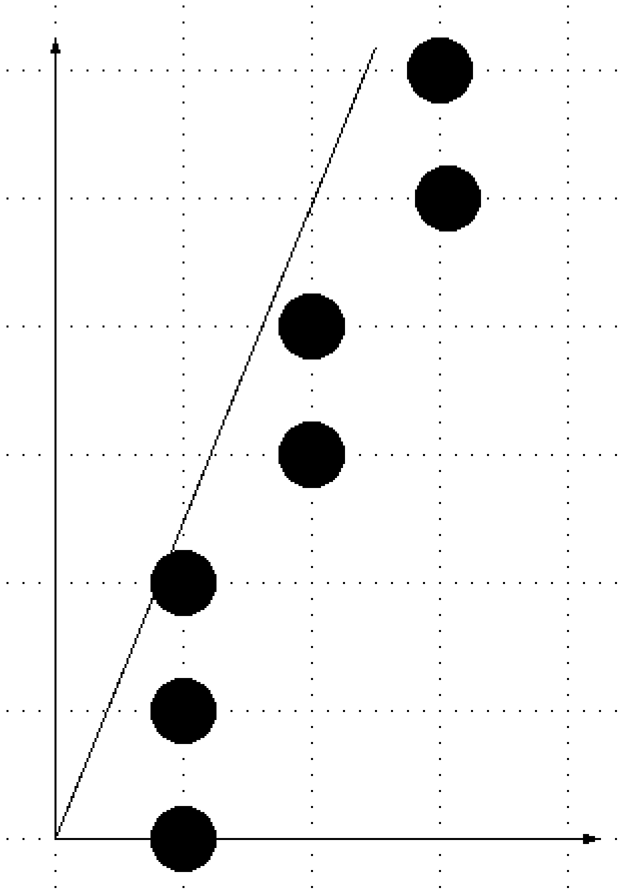
- $(x_{min}, y_{min}) = (0,0)$
- $(x_{max}, y_{max}) = (2,5)$

$$y = \frac{5}{2}x$$

As s increases, Xs will be

$0, \frac{2}{5}, \frac{4}{5}, 1\frac{1}{5}, 1\frac{3}{5}, 2, ..$

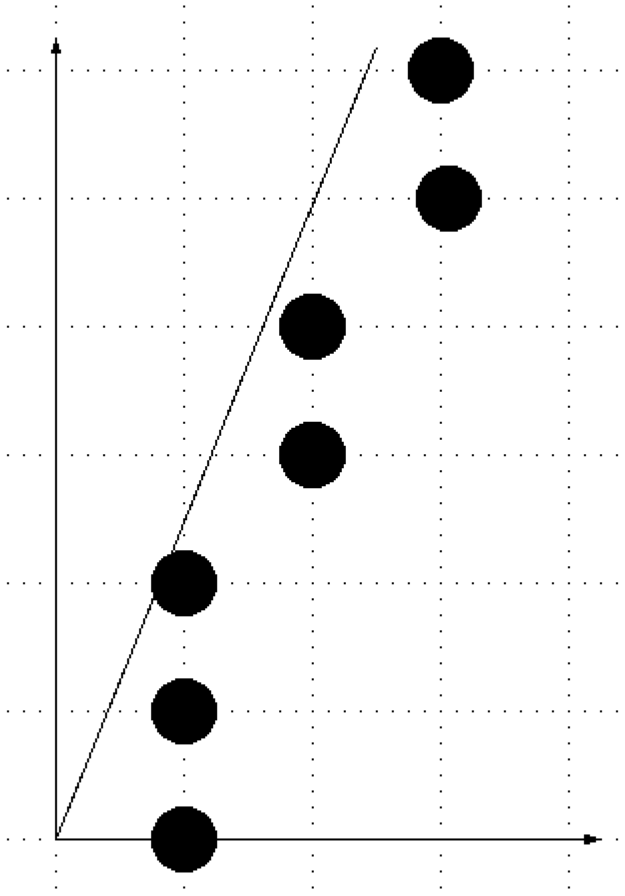
$x_1 = 1, x_2 = 1,$



Pseudo code of computing the left span extrema

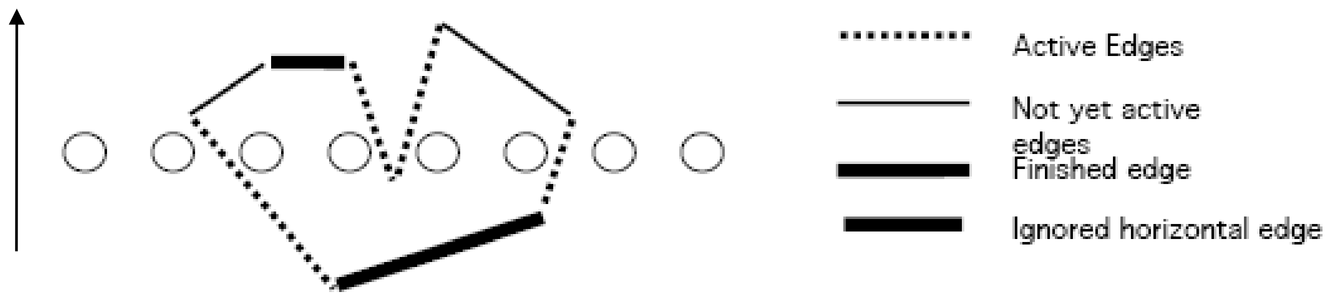
$$\frac{1}{m} = \frac{x_{\max} - x_{\min}}{y_{\max} - y_{\min}}$$

```
int $numerator = x_{\max} - x_{\min}$
int $denomenator = y_{\max} - y_{\min}$
int $N = denomenator$
for ($y = y_{\min}; y < y_{\max}; y{+}{+}$){
    WritePixel ($x,y$);
    $N$ += $numerator$;
    if ($N > denomenator$){
        /*Overflow*/
          $X{+}{+}$;
          $N$ -= $denomenator$;
    }
}
```



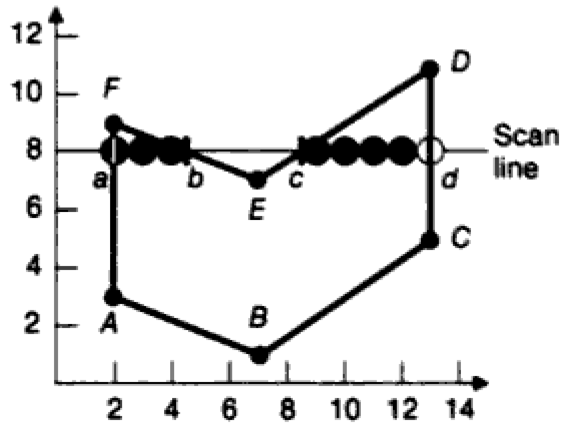
Managing the Edges

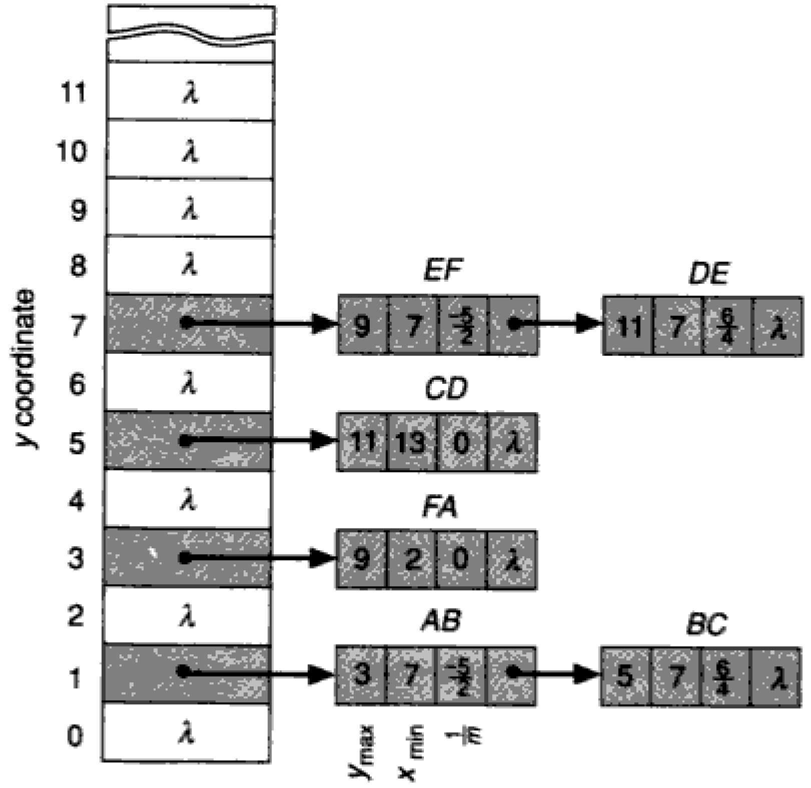
- We need to manage the edges as they are used to determine the area to be lit
- We prepare two lists of edges for this purpose
  - Global Edge Table (all the egdes in the scene)
  - Active Edge Table (the edges in the current scan line)



Global Edge Table

– Edges are bucket sorted in the Global Edge Table according to their minimum Y



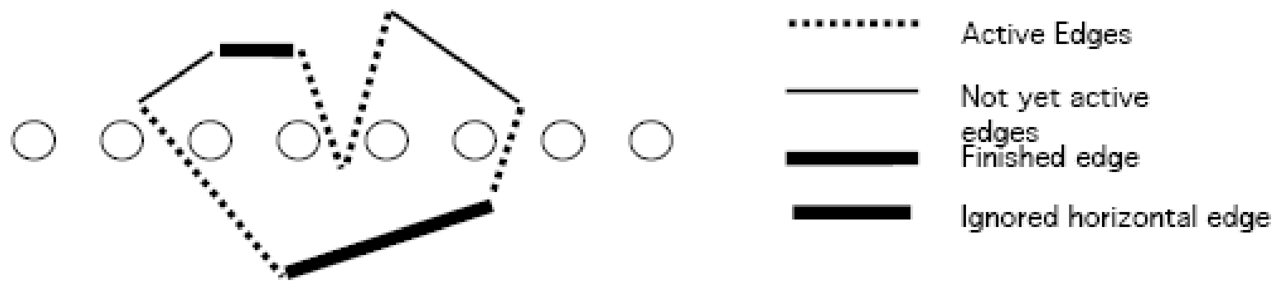


Global Edge Table

- When the current scan line reaches the lower endpoint of an edge (y_{min}) it becomes active.
  - Added into the Active Edge Table
- When the current scan line moves above the upper endpoint (y_{max}), the edge becomes inactive.
  - Removed from the Active Edge Table

Active Edge Table

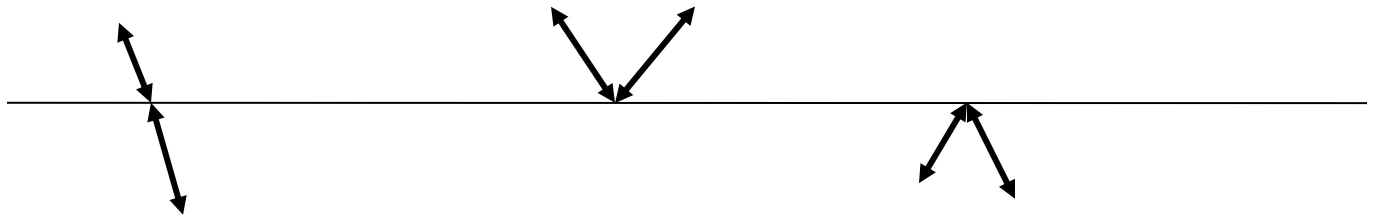
- Active edges are sorted according to increasing X.
- Filling in pixels between left most edge intersection and stops at the second.
- Restarts at the third intersection and stops at the fourth.



Polygon fill rules
(To ensure consistency)

Horizontal edges: Do not include in edge table

Vertices: If local max or min, then count twice, else count once.



If pixel is on edge, only draw left / bottom edges

Rasterizing Triangles

- Triangles are much easier to handle
  - Always convex
  - Always on a plane
  - Never self-intersects
  - Easy to interpolate data
- In many implementations, polygons with more than three edges are divided into triangles first

Triangle Rasterization by Barycenteric 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

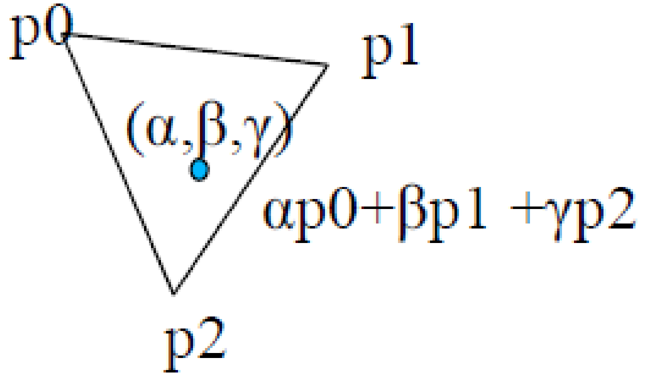
Triangle Rasterization

Consider a 2D triangle with vertices $\boldsymbol{p_0}, \boldsymbol{p_1}, \boldsymbol{p_2}$.

Let $\boldsymbol{p}$ be any point in the plane, it can be expressed by

$$\begin{aligned}\boldsymbol{p} &= \boldsymbol{p_0} + \beta(\boldsymbol{p_1} - \boldsymbol{p_0}) + \gamma(\boldsymbol{p_2} - \boldsymbol{p_0}) \\ &= (1 - \beta - \gamma)\boldsymbol{p_0} + \beta\,\boldsymbol{p_1} + \gamma\boldsymbol{p_2} \\ &= \alpha\boldsymbol{p_0} + \beta\,\boldsymbol{p_1} + \gamma\boldsymbol{p_2}\end{aligned}$$

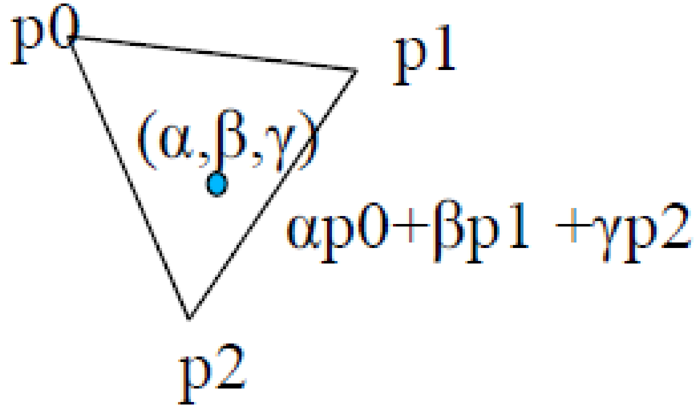
$$\alpha + \beta + \gamma = 1, \alpha, \beta, \gamma \in \mathcal{R}$$



Barycentric Coordinates

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

We call the α, β, γ the **barycentric coordinates** of ***p***.



Computing Barycentric Coordinates

The triangle is composed of 3 points

***p0** (x0,y0), **p1** (x1, y1), **p2**(x2,y2).*

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

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

where

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

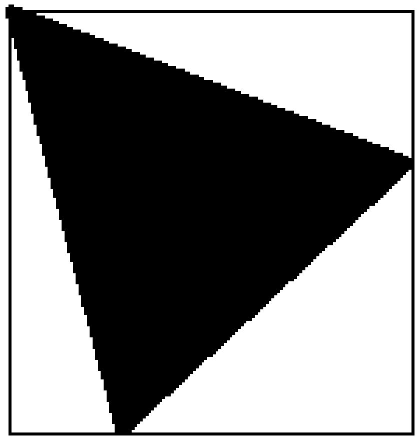
for $a, b \in \{0, 1, 2\}$.

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)



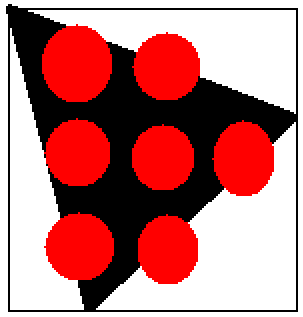
Scanning inside the triangle

Once we've identified the bounding box, we loop over each pixel in the box.

For each pixel, we first compute the corresponding (x, y) coordinates in the canonical view volume

Next we convert these into barycentric coordinates for the triangle being drawn.

Only if the barycentric coordinates are within the range of [0,1], we plot it

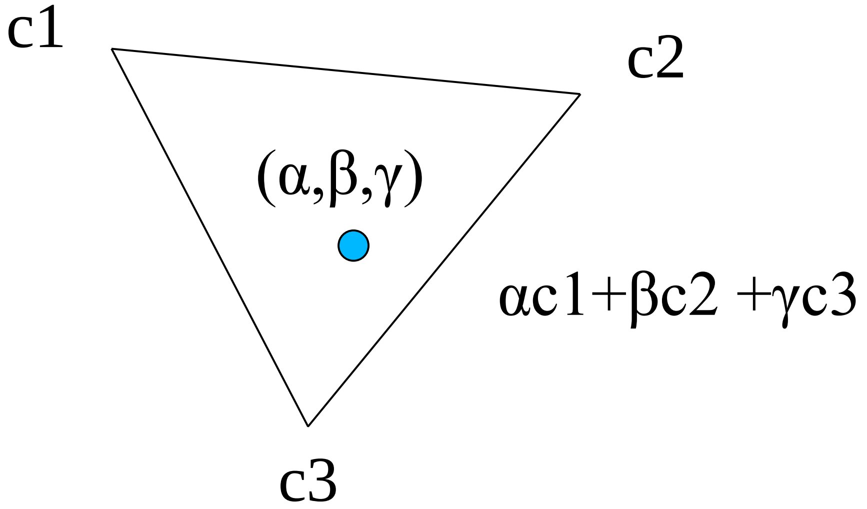


Interpolation by Barycentric Coordinates

We can use the barycentric coordinates to interpolate attributes of the triangle vertices

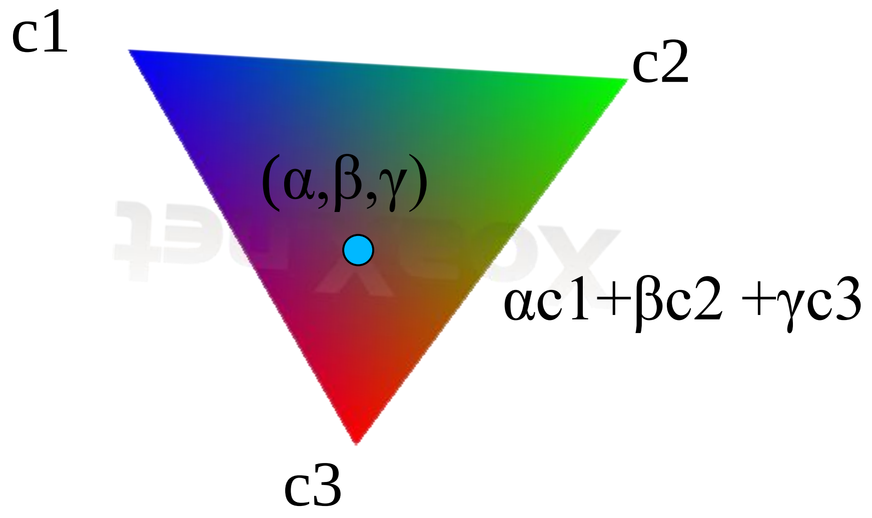
- color, depth, normal vectors, texture

coordinates



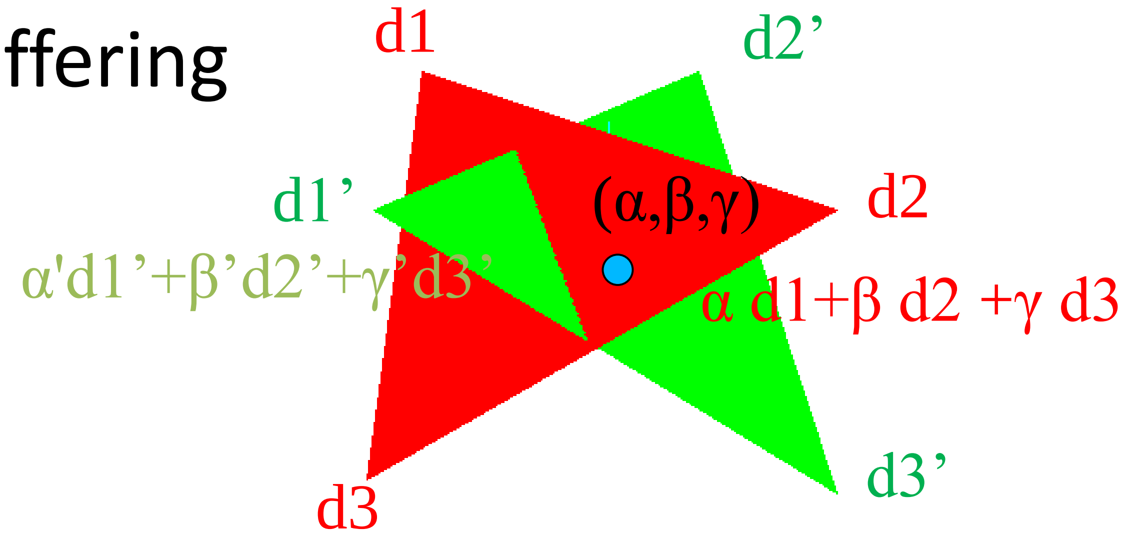
Interpolation of Color

- 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

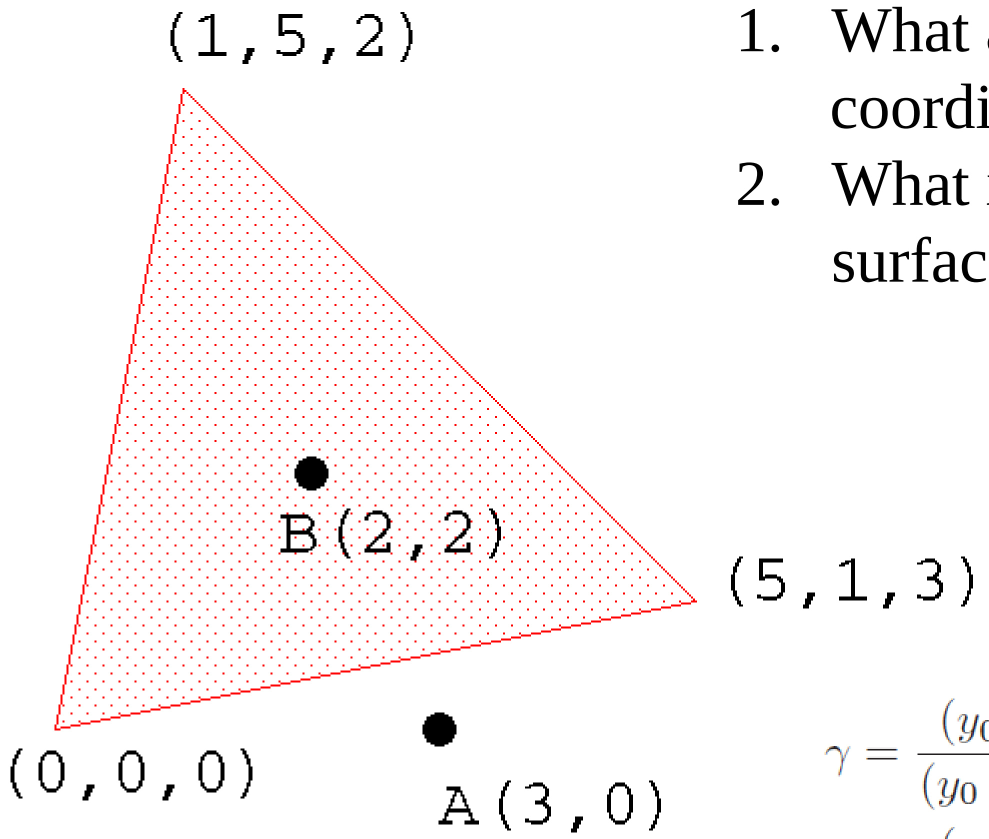


Interpolation of Depth

- 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



Exercise



1. What are the barycentric coordinates at point A and B?
2. What is the depth of the triangle surface at point B?

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

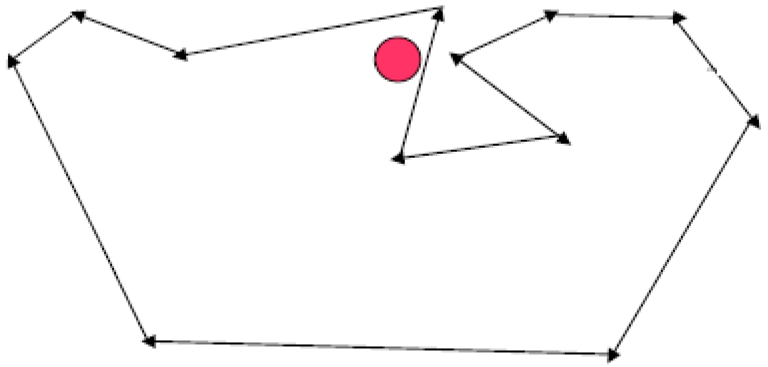
$$\beta = \frac{(y_1 - y_2)x + (x_2 - x_0)y + x_0 y_2 - x_2 y_0}{(y_0 - y_2)x_2 + (x_2 - x_0)y_2 + x_0 y_2 - x_2 y_0}$$

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

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

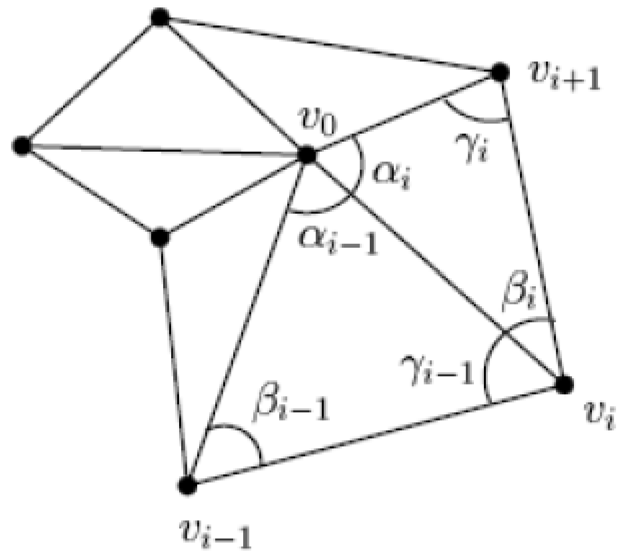
  Harmonic coordinates

  (generalized barycentric coordinates)

Mean Value Coordinates

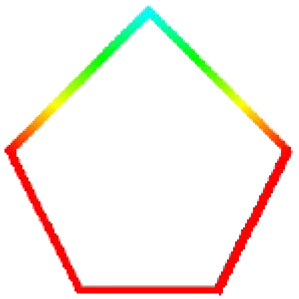
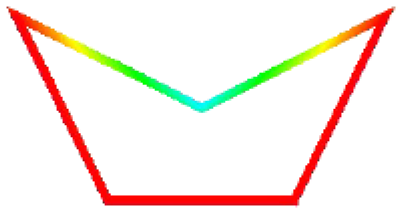
- A good and smooth barycentric coordinates that can
- smoothly interpolate the boundary values
- Also works with concave polygons
- There is also a 3D version

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



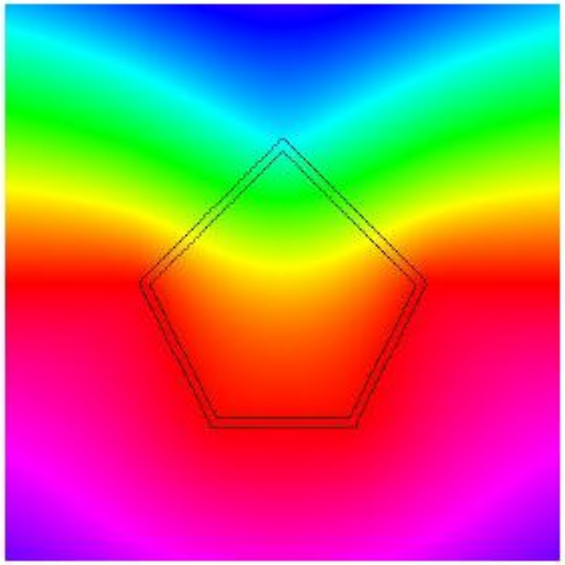
Mean Value Coordinates

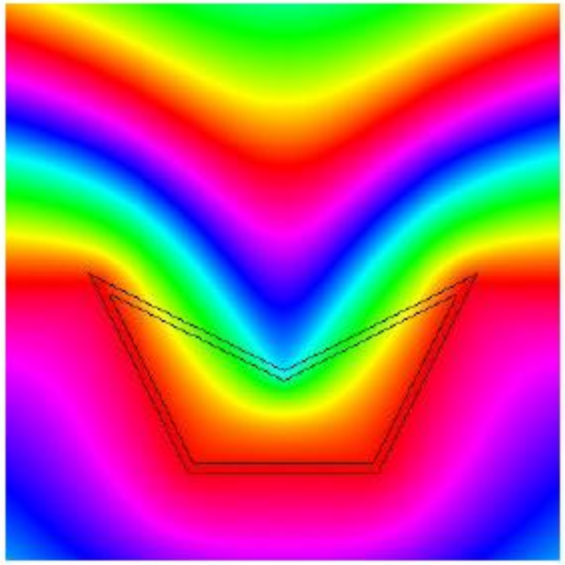
- Can interpolate convex and concave polygons
- Smoothly interpolate the interior as well as exterior

Mean Value Coordinates

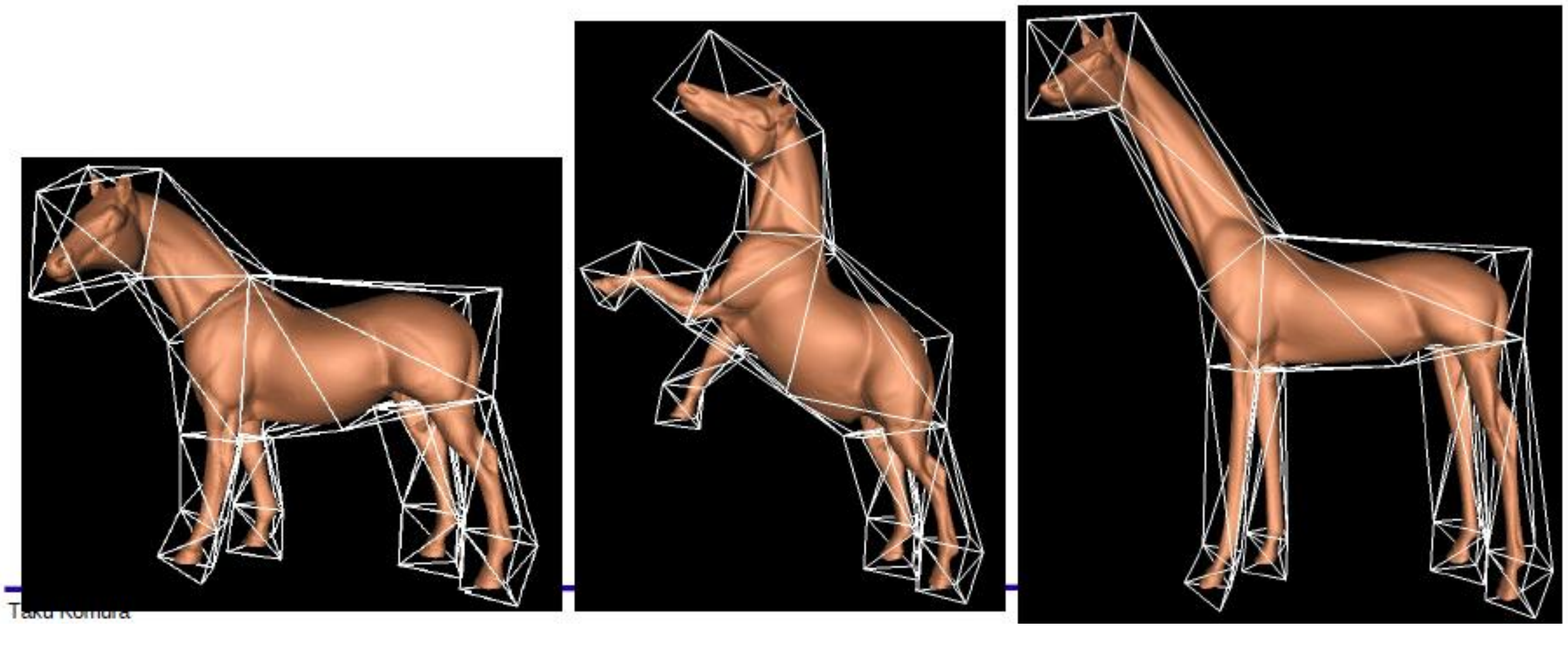
- Can interpolate convex and concave polygons
- Smoothly interpolate the interior as well as exterior





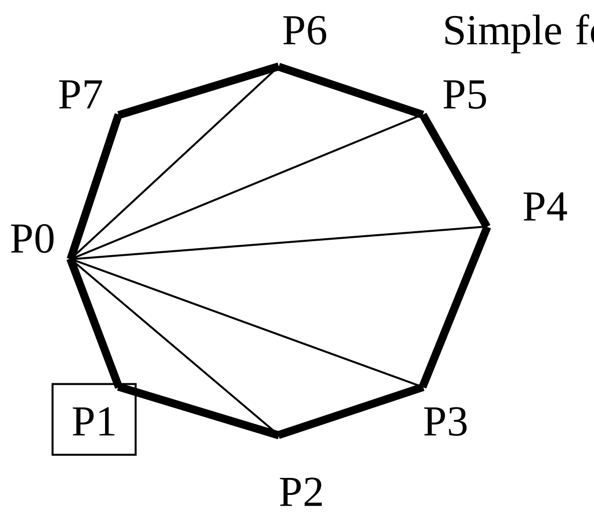
Mean Value Coordinates

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

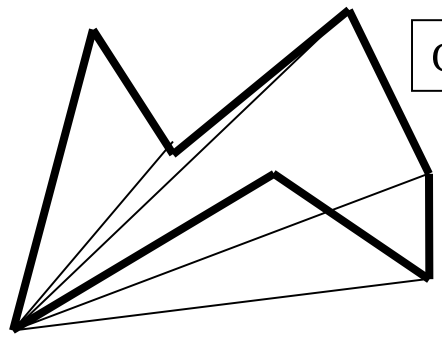


Polygon Decomposition

However, for polygons with more than four vertices, we usually decompose them into triangles



Simple for convex polygons.



Concave more difficult.

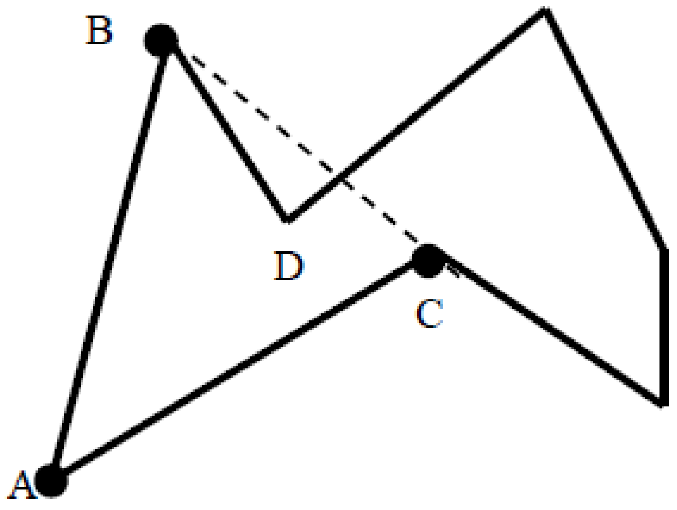
Polygon Decomposition: Algorithm

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)

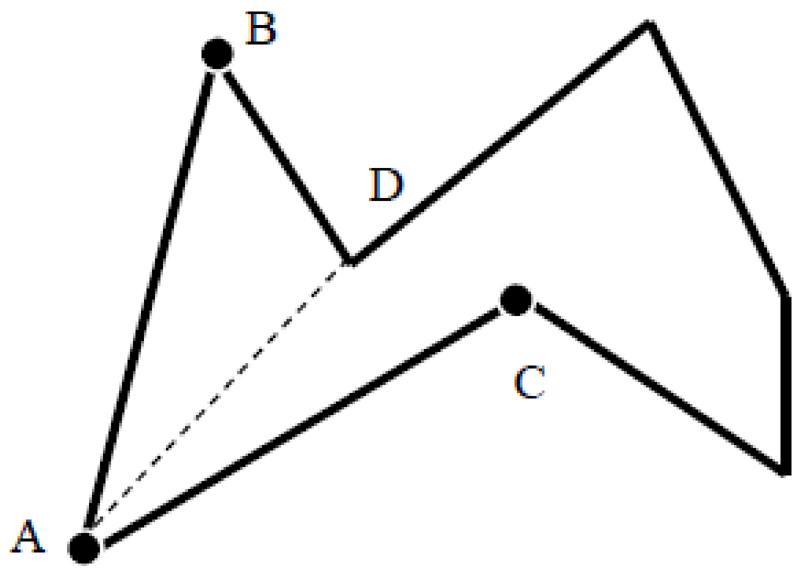
- The left most vertex A
- A triangle is formed by A and the two adjacent B and C
- Check if all the other vertices are outside the triangle



Vertex 'D' fails test.

Polygon decomposition (3)

If a vertex is inside, form a new triangle with leftmost inside vertex and point A, proceed as before.



Test ABD in same manner as before,

Reading for rasterization

- Scanline algorithm
  - Foley et al., Chapter 3.5, 3.6
- Baricentric coordinates
  - www.cs.caltech.edu/courses/cs171/barycentric.pdf
- Mean value coordinates for closed triangular meshes, SIGGRAPH 2005
- Polygon decomposition
- http://www.siggraph.org/education/materials/HyperGraph/scanline/outprims/polygon1.htm