CSSE 351
Computer Graphics

Triangle fill & interpolation

Session schedule

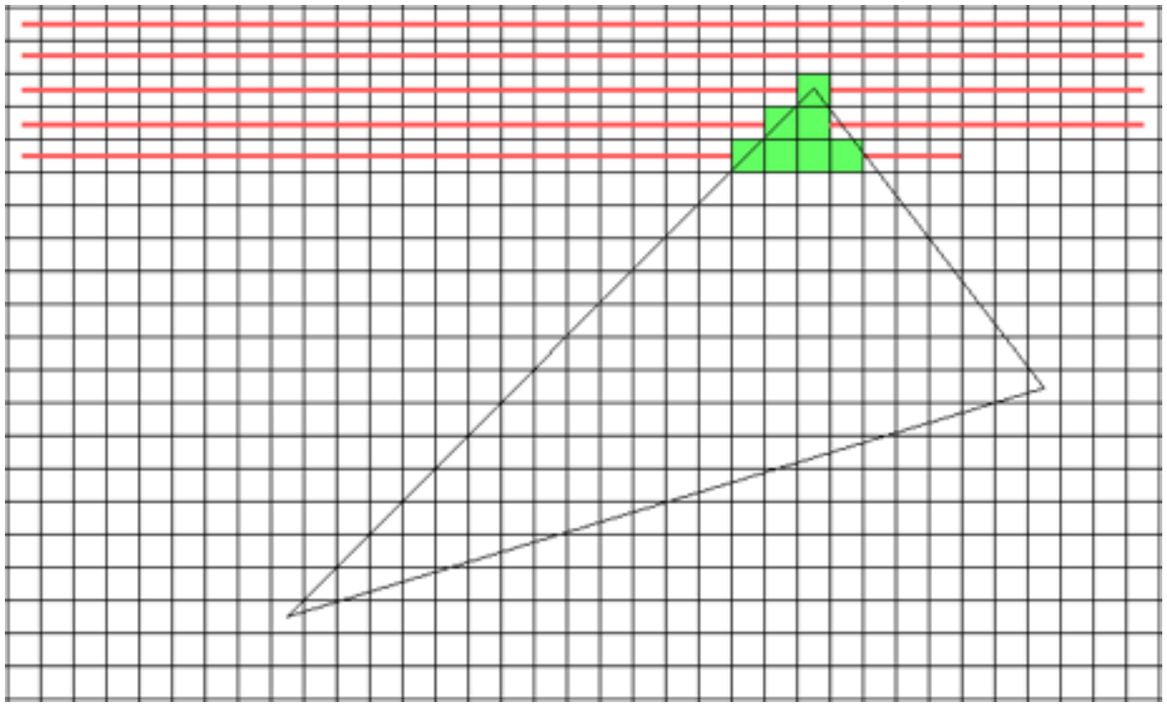
- Review
- Triangle fill
- Barycentric coordinates
  - Interpolation
- Scanline fill

Review

- Rasterization
- DDAs
- Line drawing

Triangle fill

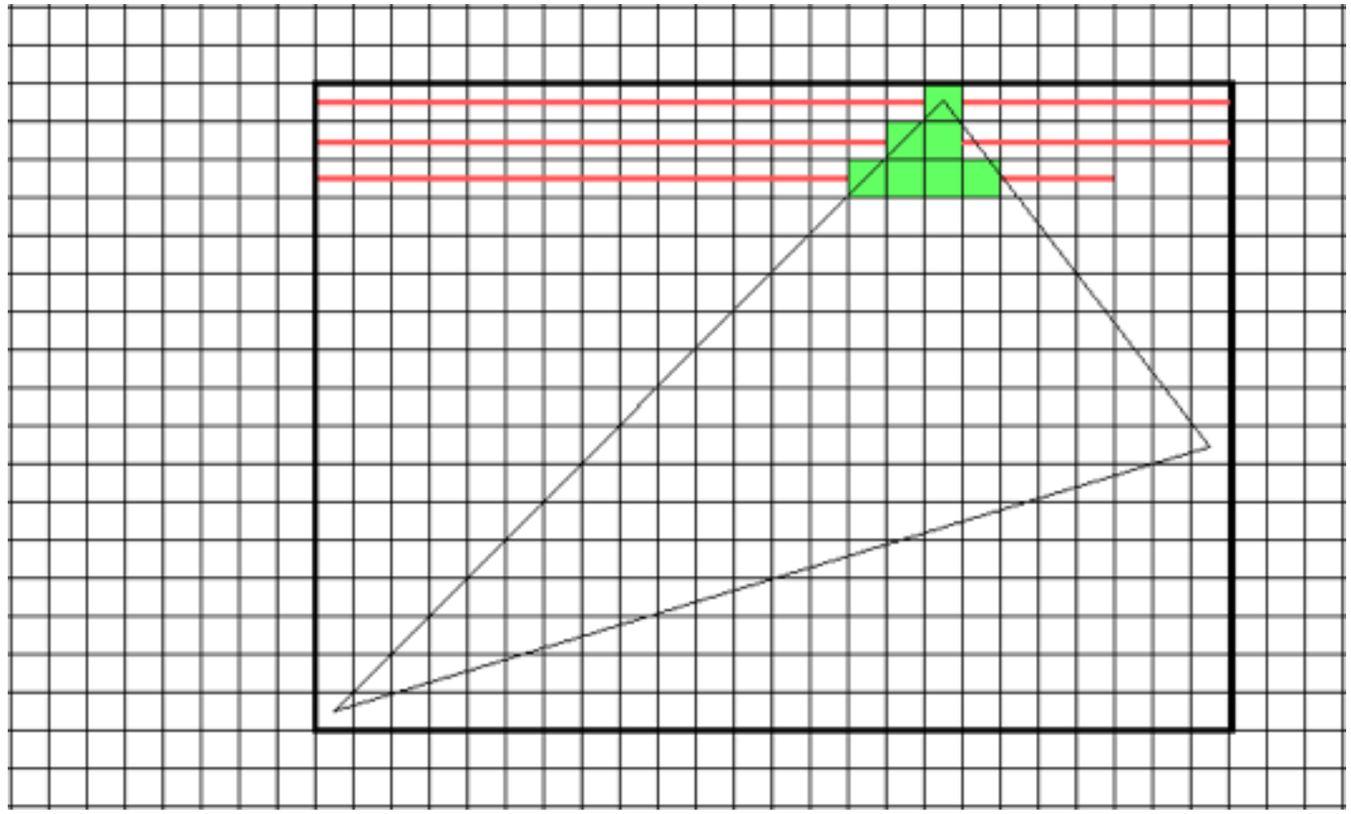
- Test if each pixel is inside each triangle



```
for x, y
    if inside(tri, x, y)
        draw(x, y)
```

Triangle fill

- Better to bound fill



```
maxX = tri.maxX
minX = tri.minX
maxY = tri.maxY
minY = tri.minY
for y = minY to maxY
for x = minX to maxX
    if inside(tri, x, y)
        draw(x, y)
```

Barycentric coordinates

- In graphics, we need to
  - interpolate over triangle surface
  - test if a point is inside a triangle

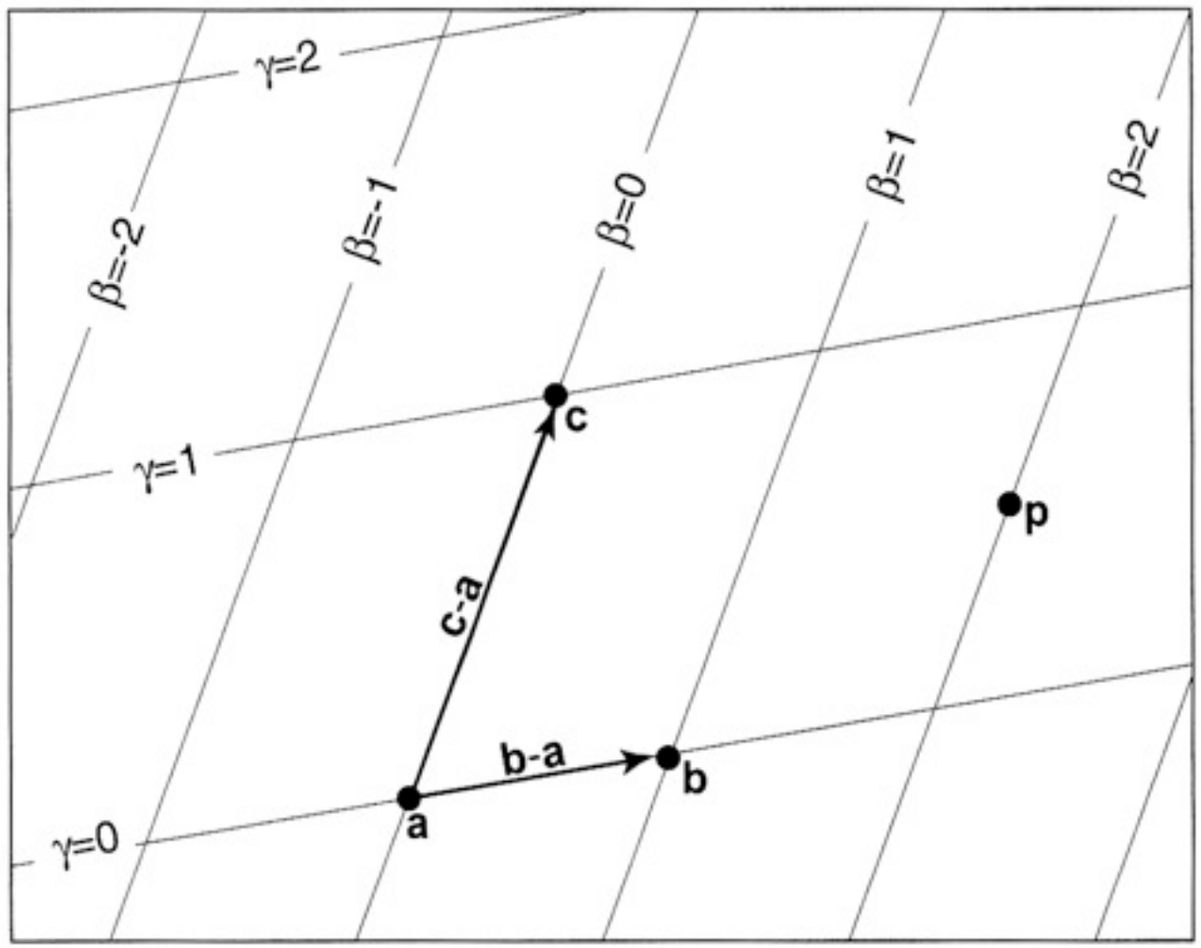
- Barycentric coordinates can do both!

Barycentric coordinates

- Relation of distances from vertices
- Defined over triangle plane
- Can be used for interpolation
- Can be used for containment test

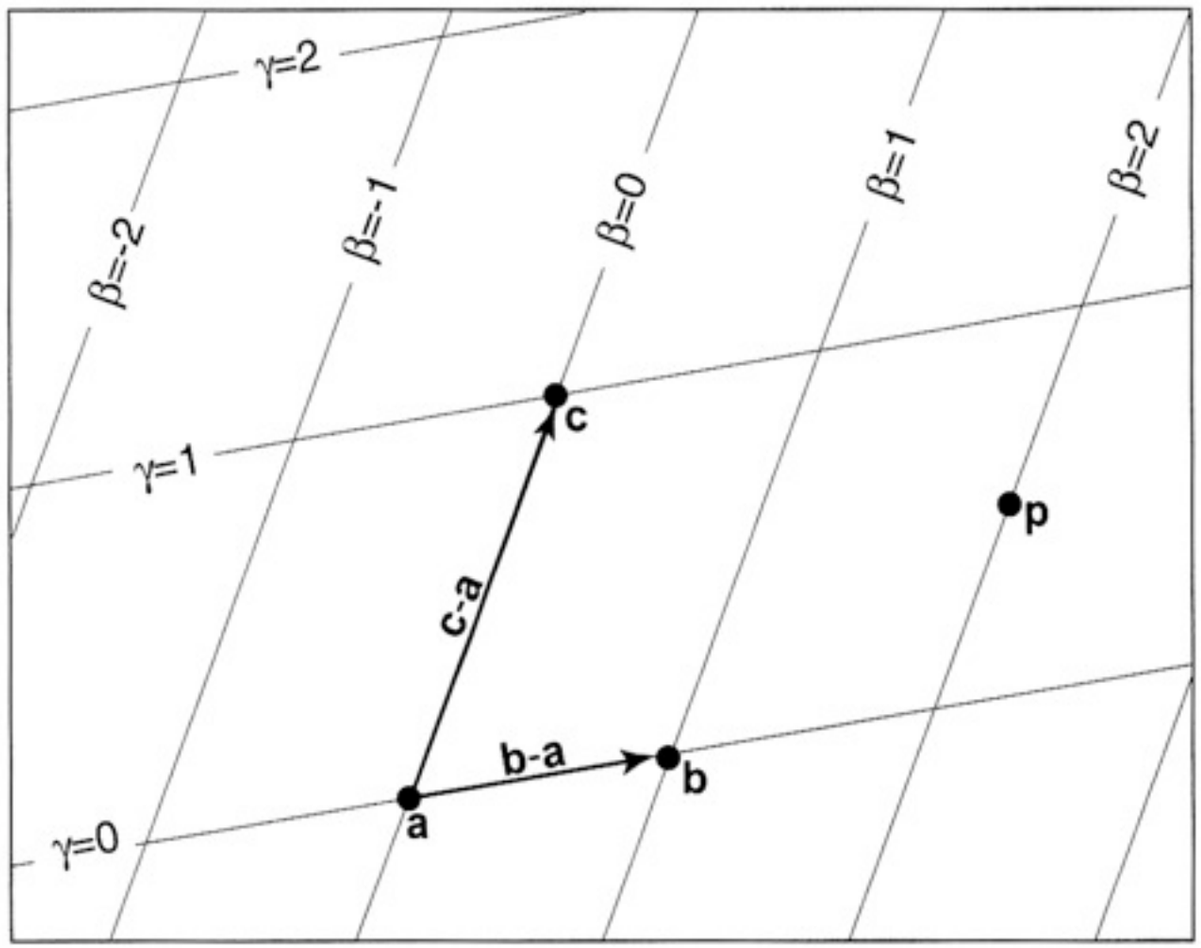
Barycentric coordinates

- Given triangle of **a**, **b**, **c**
- Compute barycentric basis
- Can then locate point **p** in triangle's barycentric coordinates



Barycentric coordinates

- Pick triangle point as origin (**a**)
- Form bases vectors (**c-a**), (**b-a**)
- Compute point **p** offsets from basis origin
  α, β, γ



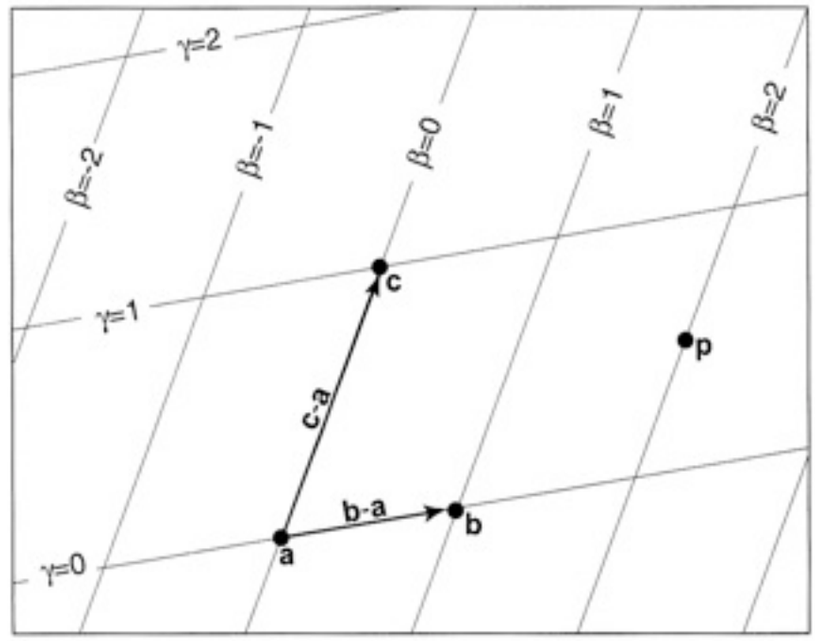
Barycentric coordinates

- So, **p** can be defined:

$$\mathbf{p} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$$

- Rearrange:

$$\mathbf{p} = (1 - \beta - \gamma)\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$$



- Rename **a** coefficient:

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

- Final form:

$$\mathbf{p}(\alpha, \beta, \gamma) = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$$

Barycentric coordinates

- Nice properties
  - Components sum to 1

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

- Inside triangle, components bound (0,1)
- On edge, components bound [0,1]
- Varies smoothly over surface

Barycentric coordinates

- Similarities to implicit line equation

$$f(x,y) \equiv (y_0 - y_1)x + (x_1 - x_0)y + x_0y_1 - x_1y_0 = 0$$

  - 0 on line
  - Smoothly vary +/- off line

- Can compute barycentric coordinate coefficients using line equation
  - Define line through triangle points
  - Measure point's offset from line
  - Normalize to triangle size

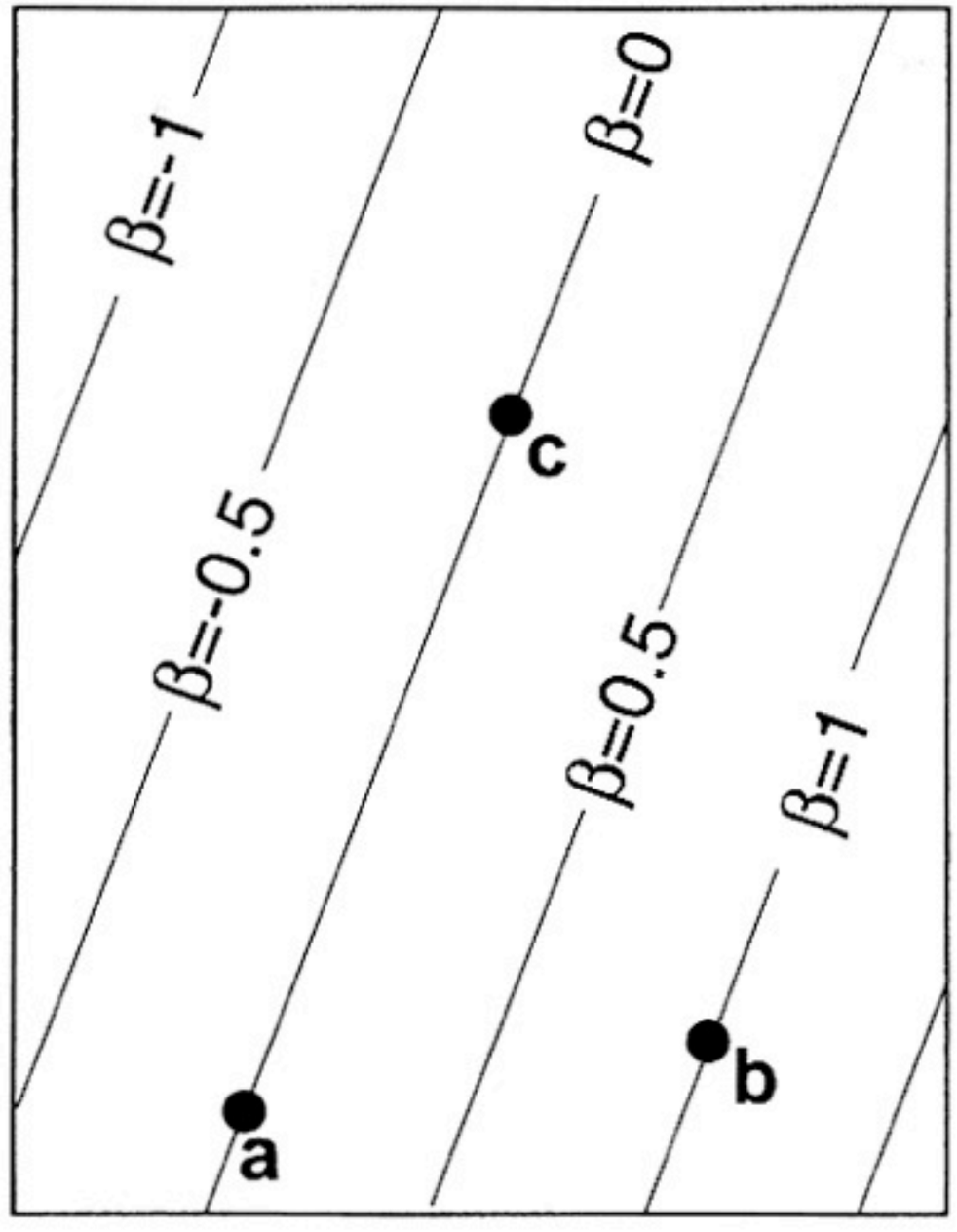
Barycentric coordinates

- Computing for point p
- Use line equation $f(x_p, y_p)$

$$\beta = \frac{f_{ac}(x_p, y_p)}{f_{ac}(x_b, y_b)}$$

$$\gamma = \frac{f_{ba}(x_p, y_p)}{f_{ba}(x_c, y_c)}$$

$$\alpha = \frac{f_{cb}(x_p, y_p)}{f_{cb}(x_a, y_a)}$$



Barycentric fill

- Inside test
  - Check if all coefficients in range [0,1]
- Interpolate
  - Use barycentric coordinates

Barycentric fill example

- For colors c1, c2, c3

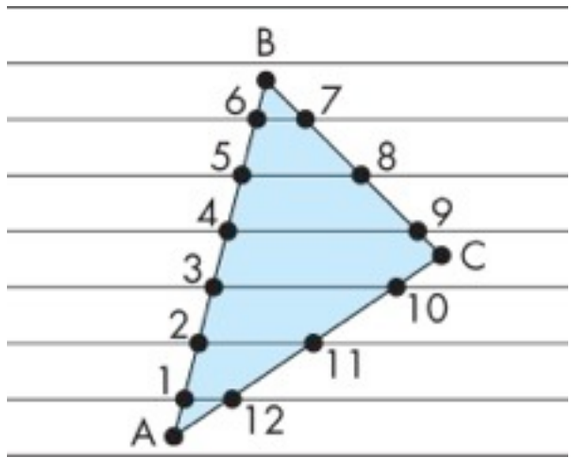
```
for y = minY to maxY
   for x = minX to maxX
      a, b, c = tri.computeBarycentric(x,y)
      if a in [0,1] and b in [0,1] and c in [0,1]
         color = a*c0 + b*c1 + c*c2
         draw(x, y, color)
```

Scanline fill

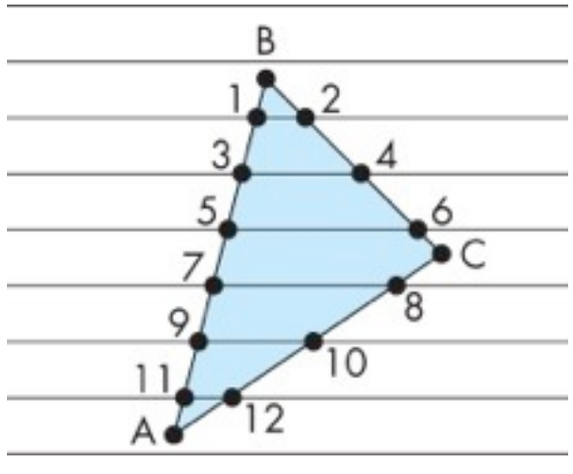
- Similar to line drawing
- Instead of drawing line
  - Compute interpolation values along lines
  - Fill in middle by interpolating horizontal lines

Scanline fill

- Order vertices conveniently
  - Often sort by vertical height
- Interpolate along edges
- Fill interior by horizontal rows



Sort & reorder



Interpolation

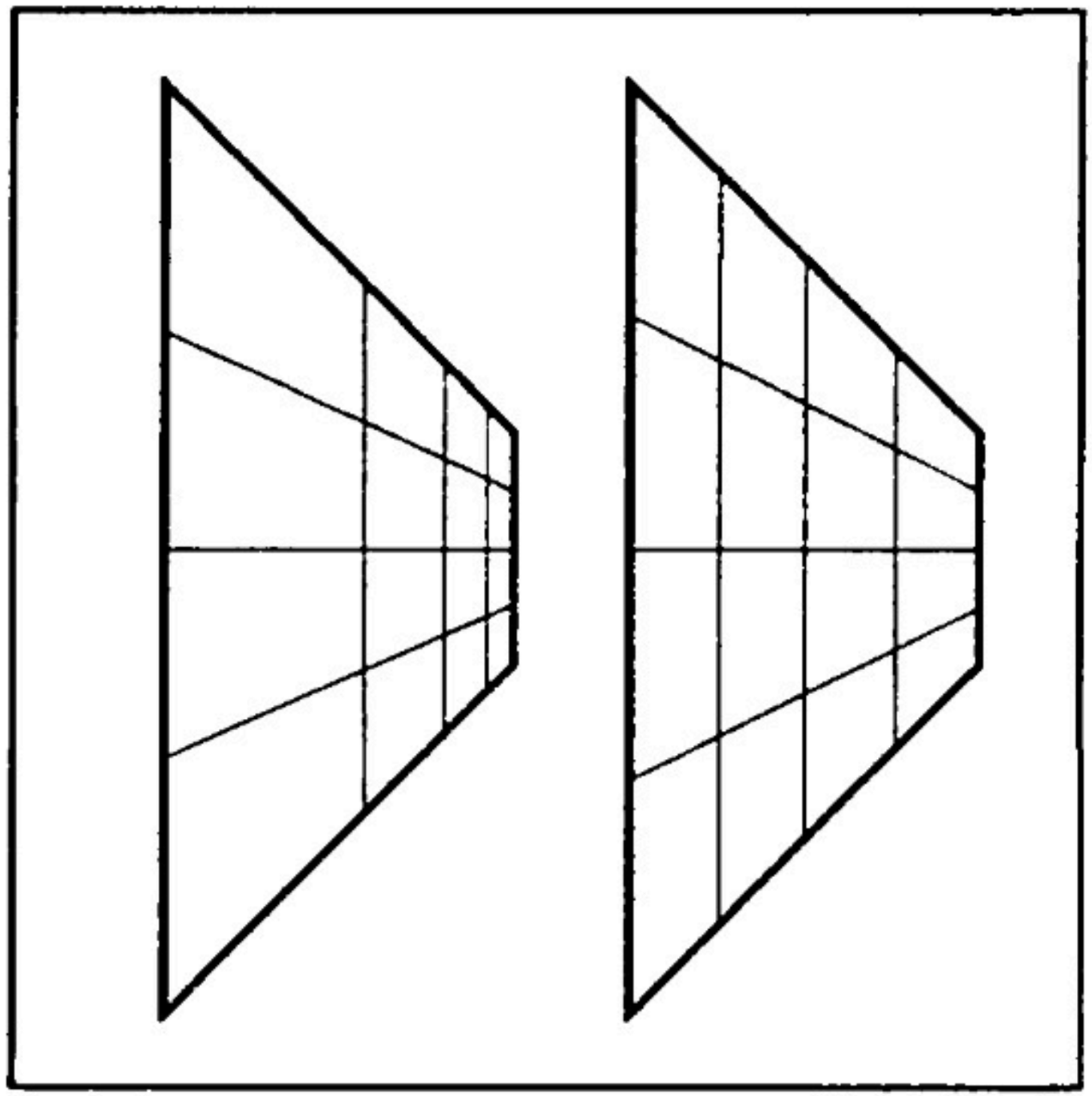
- Flat shading
  - Interpolate nothing, just fill with color
- Gouraud shading
  - Compute color at vertices
  - Interpolate color over surface
- Phong shading
  - Compute normals at vertices
  - Interpolate normals over surface
  - Compute lighting for each fragment

Interpolation

- For general shaders
- Interpolate whatever desired

  varying output from vertex
  varying input to fragment

One small problem...



Perspective Correction

- Perspective is a non-linear transform!

linear x=Az+B

perspective x=x'z

nonlinear result! z=B/(x'-A)

Perspective Correction

- Can't use linear interpolation!

  perspective line z = B / (x' - A)

  take inverse

  1/z = x(1/B) - A/B

- Linear in terms of 1/z

Perspective Correction

- So, 'homogenize' all attributes (1/z)
  - Now linear in screen space
- Then, interpolate over triangle face
  - Homogenized attributes
  - 1/z factor
- Finally, divide all homogenized attributes by 1/z
  - Undoes the perspective transform

Perspective Correction

- Book has example of correct interpolation using barycentric coordinate fill

- Can make scanline fill very fast
  - Compute 1/z gradient for x,y
  - Compute attribute gradient for x,y
  - Increment gradient along for each fragment