CS 184 - Computer Graphics

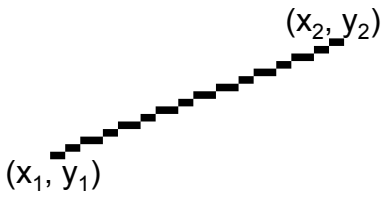
Lecture # 10: Scan Conversion

Today

- Line Drawing
- Triangle Rasterization

Line Drawing

- Given two end points (x_1, y_1) and (x_2, y_2)
  - Draw reasonable approximation of the line
  - Often limits to integer coordinate



Implicit Line Equation

- (x, y) on the line (x_1, y_1)---(x_2, y_2) if:

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$$

$$(x_2 - x_1)(y - y_1) = (y_2 - y_1)(x - x_1)$$

$$(x_2 - x_1)y - x_2y_1 - x_1y_1 = (y_2 - y_1)x - x_1y_2 + x_1y_1$$

$$f(x, y) = (x_2 - x_1)y + (y_1 - y_2)x + x_1y_2 - x_2y_1 = 0$$

Implicit Line Equation

- Interpretation of f(x,y)

$$f(x,y) = (x_2 - x_1)y + (y_1 - y_2)x + x_1y_2 - x_2y_1$$

  – Scaled signed distance from the line

$$f(x,y) = Ax + By - C = \begin{bmatrix} A \\ B \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} - C$$

  - A = y_1-y_2
  - B = x_2-x_1

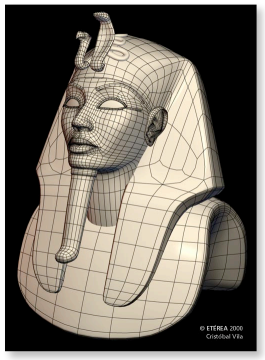
  $\begin{bmatrix} A \\ B \end{bmatrix}$ is a vector perpendicular to the line

  - C = x_2y_1-x_1y_2 ⟸ Scaled signed distance from the origin to the line

  – Will be signed distance if $\begin{bmatrix} A \\ B \end{bmatrix}$ is normalized

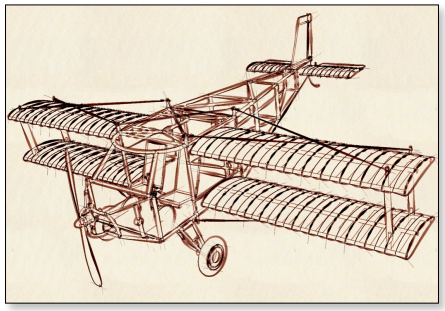
Drawing a line

- Basically, it's easy .. but for the details
- Lines are a basic primitive that needs to be done well...



Drawing a line

- Basically, it's easy .. but for the details
- Lines are a basic primitive that needs to be done well...



From "A Procedural Approach to Style for NPR Line Drawing from 3D models," by Grabli, Durand, Turquin, Sillion

Drawing Line (Implicit EQ)

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

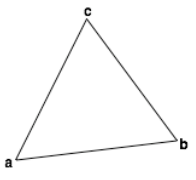
- Assume 0 < m <= 1
  - For other range of m, need modification
  - More "run" than "rise"
    - Draw 1 pixel per integer x between x_1 and x_2

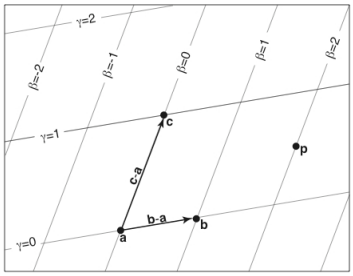
Drawing a line

- See line drawing slides

Triangle Rasterization

- A triangle is defined by 2D points, **a,b,c**
  - Defines non-orthogonal coordinate system



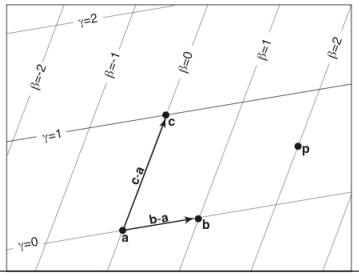


Barycentric Coordinate

- Points on triangle satisfy equation

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

$\beta, \gamma \in [0,1]$ and $\beta + \gamma <= 1$



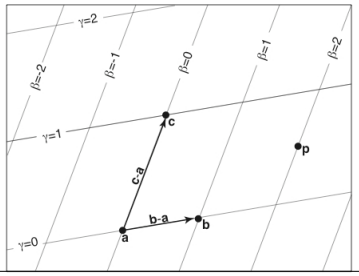
Barycentric Coordinate

- Equivalently,

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

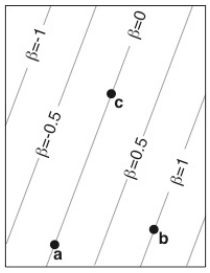
$\alpha, \beta, \gamma \in [0,1]$

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



Barycentric Coordinate

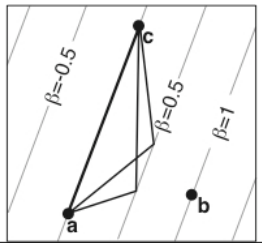
- Geometric Interpretation
  - Scaled sign distance
    - 1 at corresponding vertex
    - 0 on opposite edge



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

Barycentric Coordinate

- Geometric Interpretation
  - Ratio of area
    - Eg. β = Area of Δapc / Area of Δabc
    - 1 at corresponding vertex
    - 0 on opposite edge



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

Barycentric Coordinate

- Given a point P, how to find α, β, γ?
  - Solving linear system

$$\begin{bmatrix} x_b - x_a & x_c - x_a \\ y_b - y_a & y_c - y_a \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} x_p - x_a \\ y_p - y_a \end{bmatrix}$$

  - Cramer's Rule

$$\beta = \frac{\begin{vmatrix} x_p - x_a & x_c - x_a \\ y_p - y_a & y_c - y_a \end{vmatrix}}{\begin{vmatrix} x_b - x_a & x_c - x_a \\ y_b - y_a & y_c - y_a \end{vmatrix}} \qquad \gamma = \frac{\begin{vmatrix} x_b - x_a & x_p - x_a \\ y_b - y_a & y_p - y_a \end{vmatrix}}{\begin{vmatrix} x_b - x_a & x_c - x_a \\ y_b - y_a & y_c - y_a \end{vmatrix}}$$

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

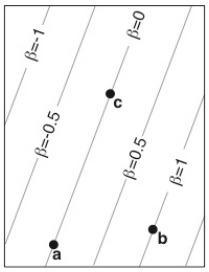
Barycentric Coordinate

- Given a point P, how to find α, β, γ?
  - Use the geometric interpretation
    - Let $f_{ac}(x,y)$ be the implicit equation for line ac

$$\beta(x,y) \propto f_{ac}(x,y)$$

$$\beta(x_b,y_b) = 1$$

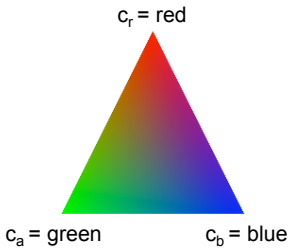
$$\beta(x,y) = \frac{f_{ac}(x,y)}{f_{ac}(x_b,y_b)}$$



    - Can do similar reasoning for α, γ

Barycentric Coordinate

- Can be used for Gouraud Shading



$$\mathbf{c} = \alpha\mathbf{c}_a + \beta\mathbf{c}_b + \gamma\mathbf{c}_c$$

- Other interpolations
  - Phong Shading
  - Texture Mapping

Triangle Rasterization

- Assume Gouraud shading

For all x do
  For all y do
    Compute (α, β, γ) for (x,y)
      If (all α, β, $\gamma \in [0,1]$)
        $\mathbf{c} = \alpha\mathbf{c}_a + \beta\mathbf{c}_b + \gamma\mathbf{c}_c$
        DrawPixel(x,y,c)

Triangle Rasterization

- Improvement

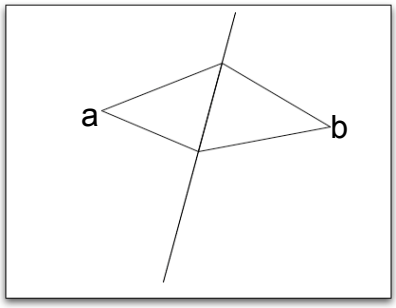
x_{min} = floor(x_i)
x_{max} = ceiling(x_i)
y_{min} = floor(y_i)
y_{max} = ceiling(y_i)
For y = y_{min} to y_{max} do
  For x = x_{min} to x_{max} do
    $\beta = f_{ac}(x,y)/f_{ac}(x_b, y_b)$
    $\gamma = f_{ab}(x,y)/f_{ab}(x_c, y_c)$
    $\alpha = 1 - \beta - \gamma$
    If (α >= 0 and β >= 0 and γ >= 0) then
      $\mathbf{c} = \alpha\mathbf{c}_a + \beta\mathbf{c}_b + \gamma\mathbf{c}_c$
      DrawPixel(x,y,c)

Triangle Rasterization

- Issues
  - Pixels on triangles edges
    - Need to draw only once
  - Still not very fast
    - Loop through many pixels not in the triangle

Triangle Rasterization

- Dealing with pixels on triangle edges
  - Two adjacent triangles sharing an edge
    - No Draw - Hole
    - Draw twice - Incorrect transparency
  - Use an off screen point to help with this
    - Draw pixel only if the off screen point is on the same side of the edge as the other vertex



. (-1,-1) - Off screen point

Triangle Rasterization

- Dealing with pixels on triangle edges

x_{min} = floor(x_i) x_{max} = ceiling(x_i)
y_{min} = floor(y_i) y_{max} = ceiling(y_i)
$f_{\alpha bc}=f_{bc}(x_a, y_a)f_{bc}(-1,-1)$
$f_{\beta ca}=f_{ca}(x_b, y_b)f_{ca}(-1,-1)$
$f_{\gamma ac}=f_{ab}(x_c, y_c)f_{ab}(-1,-1)$
For y = y_{min} to y_{max} do
  For x = x_{min} to x_{max} do
    $\beta = f_{ac}(x,y)/f_{ac}(x_b, y_b)$
    $\gamma = f_{ab}(x,y)/f_{ab}(x_c, y_c)$
    $\alpha = 1- \beta- \gamma$
    If (α >= 0 and β >= 0 and γ >= 0) then
      if (α > 0 or $f_{\alpha bc}$ > 0) and (β > 0 or $f_{\beta ca}$ > 0) and
      (γ >0 or $f_{\gamma ab}$ > 0) then
      DrawPixel(x,y,c)

Triangle Rasterization

- Fast Algorithm
  - Split triangle into 2 pieces
    - Each piece involves only 2 edges
  - Start with top 2 edges
    - For each y,
      - Compute span
      - ceil for min, floor for max
      - Draw horizontal line
      - Linearly interpolate barycoord
      - Until an edge runs out
    - If not done yet,
      - Continue with another edge