CS130 : Computer Graphics

Rasterizing Triangles and Graphics Pipeline (cont.)

Tamar Shinar
Computer Science & Engineering
UC Riverside
Triangles
barycentric coordinates
barycentric coordinates
barycentric coordinates

\[ \beta = -1 \quad \beta = 0 \quad \beta = 1 \]
barycentric coordinates

\[ \gamma = 2 \]

\[ \gamma = 1 \]

\[ \gamma = 0 \]

\[ \gamma = -1 \]
barycentric coordinates
barycentric coordinates

\[ p = \alpha a + \beta b + \gamma c \]

What are \((\alpha, \beta, \gamma)\) ?

<whiteboard>
Triangle rasterization
Which pixels should be used to approximate a triangle?
Triangle rasterization issues
Which pixels should be used to approximate a triangle?

Which should fill in shared edge?
Which pixels should be used to approximate a triangle?

Which should fill in shared edge?

- triangle that contains pixel center
- still have some ties!
- neither? both?
- want a unique assignment
Which pixels should be used to approximate a triangle?

Use Midpoint Algorithm for edges and fill in?
Which pixels should be used to approximate a triangle?

Use an approach based on barycentric coordinates.
Advantage: we can easily interpolate attributes using barycentric coordinates

\[ c = \alpha c_0 + \beta c_1 + \gamma c_2 \]

Gouraud shading

(Gouraud, 1971)

http://jtibble.dyndns.org/graphics/eecs487/eecs487.html
Triangle rasterization algorithm

for all \( x \) do
  for all \( y \) do
    compute \((\alpha, \beta, \gamma)\) for \((x,y)\)
    if \((\alpha \in [0, 1] \text{ and } \beta \in [0, 1] \text{ and } \gamma \in [0, 1])\) then
      \[ c = \alpha c_0 + \beta c_1 + \gamma c_2 \]
      drawpixel\((x,y)\) with color \( c \)
Triangle rasterization algorithm

for all x do
    for all y do
        compute \((\alpha, \beta, \gamma)\) for \((x,y)\)
        if \((\alpha \in [0, 1] \text{ and } \beta \in [0, 1] \text{ and } \gamma \in [0, 1])\) then
            \[c = \alpha c_0 + \beta c_1 + \gamma c_2\]
            drawpixel\((x,y)\) with color \(c\)
        else
            the rest of the algorithm is to make the steps in red more efficient
Triangle rasterization algorithm

use a bounding rectangle

\[
\begin{align*}
&\text{for } x \text{ in } [x_{\text{min}}, x_{\text{max}}] \\
&\quad \text{for } y \text{ in } [y_{\text{min}}, y_{\text{max}}] \\
&\quad \text{compute } (\alpha, \beta, \gamma) \text{ for } (x, y) \\
&\quad \text{if } (\alpha \in [0, 1] \text{ and } \beta \in [0, 1] \text{ and } \gamma \in [0, 1]) \text{ then} \\
&\quad \quad c = \alpha c_0 + \beta c_1 + \gamma c_2 \\
&\quad \quad \text{drawpixel}(x, y) \text{ with color } c
\end{align*}
\]
for $x$ in $[x_{\text{min}}, x_{\text{max}}]$
  for $y$ in $[y_{\text{min}}, y_{\text{max}}]$
    $\alpha = \frac{f_{bc}(x, y)}{f_{bc}(x_a, y_a)}$
    $\beta = \frac{f_{ca}(x, y)}{f_{ca}(x_b, y_b)}$
    $\gamma = \frac{f_{ab}(x, y)}{f_{ab}(x_c, y_c)}$
    if $(\alpha \in [0, 1] \text{ and } \beta \in [0, 1] \text{ and } \gamma \in [0, 1])$ then
      $c = \alpha c_0 + \beta c_1 + \gamma c_2$
      drawpixel($x, y$) with color $c$

<whiteboard>
Triangle rasterization algorithm

Optimizations?

\[
\begin{align*}
\text{for } x \text{ in } [x_{\text{min}}, x_{\text{max}}] & \text{ for } y \text{ in } [y_{\text{min}}, y_{\text{max}}] \\
\alpha &= \frac{f_{bc}(x, y)}{f_{bc}(x_a, y_a)} \\
\beta &= \frac{f_{ca}(x, y)}{f_{ca}(x_b, y_b)} \\
\gamma &= \frac{f_{ab}(x, y)}{f_{ab}(x_c, y_c)} \\
\text{if } (\alpha \in [0, 1] \text{ and } \beta \in [0, 1] \text{ and } \gamma \in [0, 1]) & \text{ then} \\
& \quad c = \alpha c_0 + \beta c_1 + \gamma c_2 \\
& \quad \text{drawpixel}(x, y) \text{ with color } c
\end{align*}
\]
Triangle rasterization algorithm

Optimizations? don’t need to check upper bound

for x in [x_min, x_max]
  for y in [y_min, y_max]
    \( \alpha = \frac{f_{bc}(x, y)}{f_{bc}(x_a, y_a)} \)
    \( \beta = \frac{f_{ca}(x, y)}{f_{ca}(x_b, y_b)} \)
    \( \gamma = \frac{f_{ab}(x, y)}{f_{ab}(x_c, y_c)} \)
    if \((\alpha \geq 0 \text{ and } \beta \geq 0 \text{ and } \gamma \geq 0)\) then
      \( c = \alpha c_0 + \beta c_1 + \gamma c_2 \)
    drawpixel(x,y) with color c
Triangle rasterization algorithm

Optimizations? compute bary. coord. and colors incrementally

for x in [x_min, x_max]
  for y in [y_min, y_max]
    \( \alpha = \frac{f_{bc}(x, y)}{f_{bc}(x_a, y_a)} \)
    \( \beta = \frac{f_{ca}(x, y)}{f_{ca}(x_b, y_b)} \)
    \( \gamma = \frac{f_{ab}(x, y)}{f_{ab}(x_c, y_c)} \)
    if (\( \alpha \geq 0 \) and \( \beta \geq 0 \) and \( \gamma \geq 0 \)) then
      \[ c = \alpha c_0 + \beta c_1 + \gamma c_2 \]
      drawpixel(x, y) with color c
Triangle rasterization algorithm

dealing with shared triangle edges

for \( x \) in \([x_{\text{min}}, x_{\text{max}}]\)
    for \( y \) in \([y_{\text{min}}, y_{\text{max}}]\)
        \( \alpha = f_{bc}(x, y) / f_{bc}(x_a, y_a) \)
        \( \beta = f_{ac}(x, y) / f_{ac}(x_b, y_b) \)
        \( \gamma = f_{ab}(x, y) / f_{ab}(x_c, y_c) \)
        if \((\alpha \geq 0 \text{ and } \beta \geq 0 \text{ and } \gamma \geq 0)\) then
            if \((\alpha > 0 \text{ or } f_{bc}(a) f_{bc}(r) > 0)\) and
                \((\beta > 0 \text{ or } f_{ca}(b) f_{ca}(r) > 0)\) and
                \((\gamma > 0 \text{ or } f_{ab}(c) f_{ab}(r) > 0)\) then
                \( c = \alpha c_0 + \beta c_1 + \gamma c_2 \)
                drawpixel(x, y) with color \( c \)
Graphics Pipeline (cont.)
Graphics Pipeline

Geometric Pipeline

Transform → Project → Clip → Rasterizer → Frame buffer

Pixel Pipeline

OpenGL application program → Transform → Pixel operations → Rasterizer
Transform
“Modelview” Transformation

Object coordinates → World coordinates → Eye coordinates

Model

View
Project
Projection: map 3D scene to 2D image
Orthographic projection
Orthographic projection

- parallel lines appear parallel
- equal length lines appear equal length
OpenGL Orthogonal Viewing

\texttt{glOrtho(left, right, bottom, top, near, far)}

View volume for an orthographic projection is a rectangular box
Perspective projection
OpenGL Perspective Viewing

\[ \text{glFrustum}(x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}, \text{near}, \text{far}) \]

View volume for a perspective projection is a frustum.
Clip
Clip against view volume
Clipping against a plane

What’s the equation for the plane through \( q \) with normal \( N \)?
Implicit line equation:

$$f(X) = N \cdot (X - X_0) = 0$$

$$X_0 = (x_0, y_0)$$
Clipping against a plane

What’s the equation for the plane through $\mathbf{q}$ with normal $\mathbf{N}$?

$$f(p) = \mathbf{n} \cdot \mathbf{q} = 0$$
Clipping against a plane

What’s the equation for the plane through \( q \) with normal \( N \)?

\[
f(p) = N \cdot (p - q) = 0
\]
Intersection of line and plane

How can we distinguish between these cases?
Intersection of line and plane

\[ f(a)f(b) \geq 0 \]

\[ f(a)f(b) < 0 \]
Intersection of line and plane

How can we find the intersection point?

<whiteboard>
Clip against view volume

\[ s = \frac{N \cdot (q - c)}{N \cdot (b - c)} \]
\[ t = \frac{N \cdot (q - a)}{N \cdot (b - a)} \]

need to generate new triangles