What is rasterization?

- input: primitives, output: fragments
- enumerate the pixels covered by a primitive
- interpolate attributes across the primitive
- output 1 fragment per pixel covered by the primitive

Figure 1. Block diagram of OpenGL.
Triangles
barycentric coordinates
barycentric coordinates

\[ p = f(a, b, c) \]
\[ p = \alpha a + \beta b + \gamma c \]

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?

Who should fill in shared edge?

but who should fill in pixels for a shared edge?
Who should fill in shared edge?

give to triangle that contains pixel center
– but we have some ties
why can’t neither/both triangles draw the pixel?
  neither: gaps
  both: indeterminacy (due to indeterminate drawing order), incorrect, e.g., if both triangles are partially transparent
we want a unique assignment
Which pixels should be used to approximate a triangle?

Use Midpoint Algorithm for edges and fill in?

That could be one possibility but we use a different approach based on barycentric coordinates.
Which pixels should be used to approximate a triangle?

Use an approach based on barycentric coordinates

For each pixel, we compute its barycentric coordinates. If the coordinates are all $\geq 0$, then the pixel is covered by the triangle.
We can interpolate attributes using barycentric coordinates

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

Gouraud shading
(Gouraud, 1971)

Using barycentric coordinates also has the advantage that we can easily interpolate colors or other attributes from triangle vertices
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

\begin{align*}
\text{for all } x \text{ do} \\
\hspace{1cm} \text{for all } y \text{ do} \\
\hspace{2cm} \text{compute } (\alpha, \beta, \gamma) \text{ for } (x,y) \\
\hspace{3cm} \text{if } (\alpha \in [0,1] \text{ and } \beta \in [0,1] \text{ and } \gamma \in [0,1]) \text{ then} \\
\hspace{4cm} c = \alpha c_0 + \beta c_1 + \gamma c_2 \\
\hspace{4cm} \text{drawpixel}(x,y) \text{ with color } c
\end{align*}

the rest of the algorithm is to make the steps in red more efficient
Triangle rasterization algorithm

use a bounding rectangle

for x in \([x_{\text{min}}, x_{\text{max}}]\)
  for y in \([y_{\text{min}}, y_{\text{max}}]\)
    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 x in [x_min, x_max]
  for y in [y_min, y_max]
    \[ \alpha = \frac{f_{bc}(x, y)}{f_{bc}(x_a, y_a)} \]
    \[ \beta = \frac{f_{ac}(x, y)}{f_{ac}(x_b, y_b)} \]
    \[ \gamma = \frac{f_{ab}(x, y)}{f_{ab}(x_c, y_c)} \]
    if (\alpha \in [0, 1] and \beta \in [0, 1] and \gamma \in [0, 1]) then
      \[ c = \alpha c_0 + \beta c_1 + \gamma c_2 \]
      drawpixel(x, y) with color c

<whiteboard>

<whiteboard> : computing alpha, beta, and gamma
Triangle rasterization algorithm

Optimizations?

for x in [x_min, x_max]
  for y in [y_min, y_max]
    \[ \alpha = \frac{f_{bc}(x, y)}{f_{bc}(x_a, y_a)} \]
    \[ \beta = \frac{f_{ac}(x, y)}{f_{ac}(x_b, y_b)} \]
    \[ \gamma = \frac{f_{ab}(x, y)}{f_{ab}(x_c, y_c)} \]
    if (\alpha \in [0, 1] and \beta \in [0, 1] and \gamma \in [0, 1]) then
      \[ c = \alpha c_0 + \beta c_1 + \gamma c_2 \]
      drawpixel(x, y) with color c

1. can make computation of bary. coords. **incremental**
   - f(x,y) = Ax+By+C
   - f(x+1,y) = f(x,y) + A
2. **color** computation can also be made **incremental**
3. alpha > 0 and beta > 0 and gamma > 0 (if true => they are also less than one)
Triangle rasterization algorithm

dealing with shared triangle edges

for \( x \) in \([x_{\text{min}}, x_{\text{max}}]\)
  for \( y \) in \([y_{\text{min}}, y_{\text{max}}]\)
    \[
    \alpha = \frac{f_{bc}(x, y)}{f_{bc}(x_a, y_a)}
    \]
    \[
    \beta = \frac{f_{ac}(x, y)}{f_{ac}(x_b, y_b)}
    \]
    \[
    \gamma = \frac{f_{ab}(x, y)}{f_{ab}(x_c, y_c)}
    \]
    \[
    \text{if } (\alpha \geq 0 \text{ and } \beta \geq 0 \text{ and } \gamma \geq 0) \text{ then}
    \]
    \[
    \text{if } (\alpha > 0 \text{ or } f_{12}(p_0)f_{12}(r) > 0) \text{ and}
    \]
    \[
    (\beta > 0 \text{ or } f_{20}(p_1)f_{20}(r) > 0) \text{ and}
    \]
    \[
    (\gamma > 0 \text{ or } f_{01}(p_2)f_{01}(r) > 0)
    \]
    \[
    c = \alpha c_0 + \beta c_1 + \gamma c_2
    \]
    \[
    \text{drawpixel}(x, y) \text{ with color } c
    \]

- compute \( f_{12}(r), f_{20}(r) \) and \( f_{01}(r) \) and make sure \( r \) doesn’t hit a line