THE RENDERING PIPELINE

Vertices and attributes

- Vertex Shader
  - Modelview transform
  - Per-vertex attributes

- Rasterization
  - Scan conversion
  - Interpolation

- Per-Sample Operations
  - Depth test
  - Blending

- Vertex Post-Processing
  - Viewport transform
  - Clipping

- Fragment Shader
  - Texturing/...
  - Lighting/shading

→ Framebuffer
CLIPPING

• Clip stuff outside our view volume
• Outside includes left/right, top/bottom, far, near/front

CLIPPING

• More importantly, front/near:
CLIPPING

• More importantly, front/near:
**CLIPPING**

- Where to do it in pipeline?

**CLIPPING**

- Option 1: Before projection
- Option 2: After NDCS
- Option 3: In between?
CLIPPING

• Option 1: Before projection
  • Then it would have to know all the camera info
• Option 2: After NDCS
• Option 3: In between?

CLIPPING

• Option 1: Before projection
  • Then it would have to know all the camera info
• Option 2: After NDCS
  • Flip already occurred
  • Too many calculations
• Option 3: In between?
CLIPPING

• Option 1: Before projection
  • Then it would have to know all the camera info
• Option 2: After NDCS
  • Flip already occurred
  • Too many calculations
• Option 3: In between?

CLIPPING

• Perform clipping in clip-coordinates!
  • After projection and before dividing by w
CLIPPING
• Perform clipping in clip-coordinates!
  • After projection and before dividing by w

\[-w_c < x_c < w_c\]
\[-w_c < y_c < w_c\]
\[-w_c < z_c < w_c\]

We have not performed any divisions =>
no flip; efficiency

CLIPPING: UNDER THE HOOD
• Creates new vertices
• How?
CLIPPING: UNDER THE HOOD

- Creates new vertices
- How?
- Clip:
  - Points -> discard (easy)
  - Triangles -> clip (harder)

Bonus question: What is the maximal side number for triangle clipped by box? Bring an example
CLIPPING COORDINATES

- Eye coordinates (projected) → clip coordinates → normalized device coordinates (NDCs)
- Dividing clip coordinates \((x_c, y_c, z_c, w_c)\) by the \(w_c(=w_n)\) component (the fourth component in the homogeneous coordinates) yields normalized device coordinates (NDCs).

\[
\begin{bmatrix}
    x_c w_c \\
    y_c w_c \\
    z_c w_c \\
    w_c
\end{bmatrix} = \begin{bmatrix}
    s_x & 0 & -c_x & 0 \\
    0 & s_y & -c_y & 0 \\
    0 & 0 & f + n & -2fn \\
    0 & 0 & f - n & f - n
\end{bmatrix} \begin{bmatrix}
    x_c \\
    y_c \\
    z_c \\
    1
\end{bmatrix}
\]

VIEWPORT MATRIX

- We need a transform that maps the lower left corner to \([-0.5, -0.5]^t\) and upper right corner to \([W - 0.5, H - 0.5]^t\).

- The appropriate scale and shift can be done using the viewport matrix:

\[
\begin{bmatrix}
    x_w \\
    y_w \\
    z_w \\
    1
\end{bmatrix} = \begin{bmatrix}
    W / 2 & 0 & 0 & (W - 1) / 2 \\
    0 & H / 2 & 0 & (H - 1) / 2 \\
    0 & 0 & 1 / 2 & 1 / 2 \\
    0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
    x_n \\
    y_n \\
    z_n \\
    1
\end{bmatrix}
\]
THE RENDERING PIPELINE

- Vertices and attributes
  - Vertex Shader
    - Modelview transform
    - Per-vertex attributes
  - Rasterization
    - Scan conversion
    - Interpolation
  - Per-Sample Operations
    - Depth test
    - Blending
  - Vertex Post-Processing
    - Viewport transform
    - Clipping
    - Fragment Shader
      - Texturing/...
      - Lighting/shading

  → Framebuffer

RASTERIZATION

- This is part of the fixed function pipeline
- Input: all polygons are clipped
- Output: fragments (with varying variables interpolated)
PATH FROM VERTEX TO PIXEL

WHAT IS SCAN CONVERSION? (A.K.A. RASTERIZATION)
• screen is discrete
• one possible scan conversion

HOW TO CHECK IF A PIXEL IS INSIDE?

...
HOW TO CHECK IF A PIXEL IS INSIDE?

- Use implicit line equation:
  - \( Ax + By + C = 0 \)

- How to find A,B,C?

- Orientation?

A point is inside \( \Leftrightarrow \)

\[
A_i x + B_i y + C > 0, \quad i = 1, \ldots, 3
\]
HOW TO TREAT BOUNDARY?

• If two triangles share an edge, scan conversion should be consistent
  • No pixel drawn twice
  • No gaps

• Strategy ideas?
BONUS 2

• With the algorithm above, what’s the minimum number of pixels that will be drawn for the following triangle:

\[ P_1 = (0.5,0.5) \]
\[ P_2 = (99.5,100.5) \]
\[ P_3 = (-98.5,100.5) \]

• Proof!
• 5 first solutions accepted; +5% to final grade

NAÏVE SCAN CONVERSION

• Testing every pixel is suboptimal
• Better ideas?
SCANLINE IDEA

• Basic structure of code:
  • Setup: compute edge equations, bounding box
  • (Outer loop) For each scanline in bounding box...
  • (Inner loop) ...check each pixel on scanline, evaluating edge equations and drawing the pixel if all three are positive

SCANLINE: CODE

```c
findBoundingBox(xmin, xmax, ymin, ymax);
setupEdges (a0,b0,c0,a1,b1,c1,a2,b2,c2);

for (int y = yMin; y <= yMax; y++) {
  for (int x = xMin; x <= xMax; x++) {
    float e0 = a0*x + b0*y + c0;
    float e1 = a1*x + b1*y + c1;
    float e2 = a2*x + b2*y + c2;
    if (e0 > 0 && e1 > 0 && e2 > 0)
      Image[x][y] = TriangleColor;
  }
}
```
TRIANGLE RASTERIZATION ISSUES

• Exactly which pixels should be lit?
• A: Those pixels inside the triangle edges
• What about pixels exactly on the edge?

TRIANGLE RASTERIZATION ISSUES

Sliver

• Moving Slivers
GENERALIZATION TO POLYGONS: HOW TO TEST IF A POINT IS IN A POLYGON?

simple convex  simple concave  non-simple (self-intersection)