Rasterizing triangles

• Summary
  1 evaluation of linear functions on pixel grid
  2 functions defined by parameter values at vertices
  3 using extra parameters to determine fragment set
Barycentric coordinates

• A coordinate system for triangles

\[ p = a + \beta(b - a) + \gamma(c - a) \]

Clipping to the triangle

• Interpolate three barycentric coordinates across the plane
  – each barycentric coord is 1 at one vert. and 0 at the other two
  – lies between 0 and 1 inside the triangle

• Output fragments only when all three are > 0.
Another way: Edge equations

• In plane, triangle is the intersection of 3 half spaces

\[(x - a) \cdot (b - a)^\perp > 0\]
\[(x - b) \cdot (c - b)^\perp > 0\]
\[(x - c) \cdot (a - c)^\perp > 0\]

Walking edge equations

• We need to update values of the three edge equations with single-pixel steps in \(x\) and \(y\)
• Edge equation already in form of dot product
• Components of vector are the increments
Rest of pipeline

Clipping

• Rasterizer tends to assume triangles are on screen
  – particularly problematic to have triangles crossing the plane \( z = 0 \)

• Why?
Clipping

- Eliminate geometry outside frustum

- How?
  - Clip to world space frustum
  - Clip to clip coordinates (before divide)
    - Both fine, often the latter

- After projection, before perspective divide
  - clip against the planes $x, y, z = 1, -1$ (6 planes)
  - primitive operation: clip triangle against axis-aligned plane
Clipping a triangle against a plane

- 4 cases, based on sidedness of vertices
  - all in (keep)
  - all out (discard)
  - one in, two out (one clipped triangle)
  - two in, one out (two clipped triangles)

- Intersect the edge line with the plane
  - \( x = 0 \), use y intercept (for example)

Pipeline of transformations

- Standard sequence of transforms
Hidden surface elimination

- We have discussed how to map primitives to image space
  - projection and perspective are depth cues
  - occlusion is another very important cue
Painter’s algorithm

• Simplest way to do hidden surfaces
• Draw from back to front, use overwriting in framebuffer

Painter’s algorithm

• Useful when a valid order is easy to come by
• Compatible with alpha blending
**Painter’s algorithm**

- Amounts to a topological sort of the graph of occlusions
  - that is, an edge from A to B means A sometimes occludes B
  - any sort is valid
    - BA...
    - B...A...
  - if there are cycles
    - there is no sort

- Works when valid sort is easy to come by

---

**The z buffer**

- In many (most) applications maintaining a z sort is too expensive
  - changes all the time as the view changes
  - many data structures exist, but complex
- Solution: draw in any order, keep track of closest surface
  - allocate extra channel per pixel to keep track of closest depth so far
  - when drawing, compare object’s depth to current closest depth and discard if greater

- History: memory use was a concern, not anymore.
The z buffer

- Memory-intensive brute force approach that works and has become the standard

| 0 0 0 0 0 0 0 0 | 5 5 5 5 5 5 5 5 | 5 5 5 5 5 5 5 5 |
| 0 0 0 0 0 0 0 0 | 5 5 5 5 5 5 5 5 | 5 5 5 5 5 5 5 5 |
| 0 0 0 0 0 0 0 0 | 5 5 5 5 5 5 5 5 | 5 5 5 5 5 5 5 5 |
| 0 0 0 0 0 0 0 0 | 5 5 5 5 5 5 5 5 | 5 5 5 5 5 5 5 5 |
| 0 0 0 0 0 0 0 0 | 5 5 5 5 5 5 5 5 | 5 5 5 5 5 5 5 5 |
| 5 5 5 5 5 5 5 5 | 3 4 3 5 4 3 6 5 5 | 5 5 5 5 5 5 5 5 |
| 5 5 5 5 5 5 5 5 | 4 3 5 4 3 6 5 5 | 5 5 5 5 5 5 5 5 |
| 5 5 5 5 5 5 5 5 | 5 4 3 6 5 4 3 7 6 5 4 3 | 5 5 5 5 5 5 5 5 |
| 5 5 5 5 5 5 5 5 | 6 5 4 3 7 6 5 4 3 8 7 6 5 4 3 | 5 5 5 5 5 5 5 5 |
| 5 5 5 5 5 5 5 5 | 7 6 5 4 3 8 7 6 5 4 3 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 5 5 5 5 5 5 5 5 | 8 7 6 5 4 3 0 0 0 0 0 0 0 0 0 0 |

Precision in z buffer

- The precision is distributed between the near and far clipping planes
  - this is why these planes have to exist
  - also why you can’t always just set them to very small and very large distances

- Generally use $z'$ (not world $z$) in z buffer
  - More about correct interpolation for $z$ in texturing!
Back face culling

• For closed shapes you will never see the inside
  – therefore only draw surfaces that face the camera
  – implement by checking $\mathbf{n} \cdot \mathbf{v}$