Overview

- Rendering topics
  - Z-buffering
  - Back-Face Culling
  - Ray Tracing (Ray Casting)
  - Radiosity

Mesh/Faceted Model

Back-Face Culling

- Assumptions:
  - Object approximated as closed polyhedron
  - Polyhedron interior is not exposed by the front cutting plane
  - Eye-point not inside object
  - Right-hand vertex ordering defines outward normal
  - Polygons not facing the viewer called Back-Facing
- Back-Face Culling is a technique for eliminating polygons for these kinds of models
- On average eliminates half of the polygons
  - Could be done for performance reasons

Back-Face Culling

- After canonical transformation, examine normal \( \mathbf{N} = (x_k, y_k, z_k) \) to the face.
- If \( z_k < 0 \), face is a Back-Face - don’t draw it
  - More general test looks at \( \mathbf{N} \cdot \mathbf{V} \)
  - \( \mathbf{V} \) - View vector
- The only test necessary for a single convex polyhedron

Normal for Triangle

\[
\text{plane } \mathbf{n} \cdot (\mathbf{p} - \mathbf{p}_0) = 0 \\
\mathbf{n} = (\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_2 - \mathbf{p}_0) \\
\text{normalize } \mathbf{n} \leftarrow \mathbf{n} / |\mathbf{n}| \\
\mathbf{p}_0
\]

Note that right-hand rule determines outward face
Z-buffering

- Z-buffering (depth-buffering) is a visible surface detection algorithm
- Implementable in hardware and software
- Requires data structure (z-buffer) in addition to frame buffer.
- Z-buffer stores values $[0 \ldots ZMAX]$ corresponding to depth of each point.
- If the point is closer than one in the buffers, it will replace the buffered values

```cpp
for (y = 0; y < YMAX; y++)
    for (x = 0; x < XMAX; x++) {
        F[x][y] = BACKGROUND_COLOR;
        Z[x][y] = ZMIN;
    }
for (each polygon)
    for (each pixel in polygon’s projection)
        pz = polygon’s z-value at pixel coordinates (x,y)
        if (pz > Z[x][y]) { /* New point is closer */
            Z[x][y] = pz;
            F[x][y] = polygon’s color at pixel coordinates (x,y)
        }
```

Z Interpolation

- We can simplify the calculation of $z$ by exploiting the fact that triangle is planar.
  - Interpolate $z$ values along the edges
  - Interpolate $z$ values along scan line
- Special cases: horizontal edge, degenerate triangle & single vertex

![Z Interpolation Diagram](https://example.com/z_interpolation.png)
Z Interpolation

- \( z_a = z_1 + \left( \frac{|P_a - P_1|}{|P_2 - P_1|} \right) (z_2 - z_1) \)
- \( z_b = z_1 + \left( \frac{|P_b - P_1|}{|P_3 - P_1|} \right) (z_3 - z_1) \)
- \( z_p = z_a + \left( \frac{|P_p - P_a|}{|P_b - P_a|} \right) (z_b - z_a) \)

\[ P_1 = (x_1, y_1) \]
\[ P_2 = (x_2, y_2) \]
\[ P_3 = (x_3, y_3) \]

Back-Face Culled & Z-Buffered Wire-Frame

See the Difference

Depth Cueing
- Objects that are closer are brighter
- Objects farther away are darker
- Color = BaseColor*(z - far)/(near - far)