Hidden Surface Removal

Reminder - Pipeline

Hidden Surface Removal

Hidden Surface Removal Algorithms

Back Face Culling
Object Space

In a closed polyhedron back faces are not visible
The Normal Vector

\[ \mathbf{n} = (\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1) \]

Back Face Culling
Determining back faces

- In a closed polyhedron back faces are not visible
- Assume normals point out

face visible iff: \[ -\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2} \]

iff \( \cos \theta \geq 0 \)

iff \( \mathbf{v} \cdot \mathbf{n} = |\mathbf{v}| |\mathbf{n}| \cos \theta \geq 0 \)

Back Face Culling
When will it work?

- Closed, convex
- Open
- Closed, non convex

Back Face Culling
When will it work?

- Closed convex objects
  - All back faces invisible
  - All front faces visible
  \rightarrow Visibility problem solved
- Open objects
  - Back faces possibly visible
- Closed non-convex objects
  - Invisible back faces
  - Front faces can be visible, invisible or partially visible

Painter’s Algorithm
Object space

- Which color to draw?
- Draw everything \rightarrow which order to draw?

Painter’s Algorithm
When will this fail?

- Sort polygons by depth values
- Paint back to front
- When will this fail?
Painter’s Algorithm

- Works in special cases
  - E.g. polygons with constant z
  - Where do we have polygons with constant z?

Painter’s Algorithm

How do we fix it?

- Depth sort per polygon doesn’t work
- Depth sort per pixel
  - Image space algorithm

Z-Buffer Algorithm

Image Space

Resolve visibility at the pixel level
Store color + current z per pixel
Put new color only if new z < current z

Z-Buffer Algorithm

The Z-Buffer

The Algorithm

For every pixel \((x, y)\)
\[\text{putZ}(x, y, \text{MaxZ})\]

For each polygon \(P\)
\[Q = \text{project}(P)\]
for each pixel \((x, y)\) in \(Q\)
\[z = \text{depth}(Q, x, y)\]
if \(z < \text{getZ}(x, y)\)
\[\text{putColor}(x, y, \text{col}(P))\]
\[\text{putZ}(x, y, z)\]
end
end
end

Z-Buffer Algorithm

The Algorithm

For every pixel \((x, y)\)
\[\text{putZ}(x, y, \text{MaxZ})\]

For each polygon \(P\)
\[Q = \text{project}(P)\]
for each pixel \((x, y)\) in \(Q\)
\[z = \text{depth}(Q, x, y)\]
if \(z < \text{getZ}(x, y)\)
\[\text{putColor}(x, y, \text{col}(P))\]
\[\text{putZ}(x, y, z)\]
end
end
end
Z-Buffer Algorithm

The Algorithm

For every pixel \((x, y)\)
\[ \text{put} z(x, y, \text{MaxZ}) \]

For each polygon \(P\)
\[ \text{Q} = \text{project}(P) \]
for each pixel \((x, y)\) in \(Q\)
\[ z = \text{depth}(Q, x, y) \]
if \(z < \text{get} z(x, y)\)
\[ \text{putColor}(x, y, \text{col}(P)) \]
\[ \text{put} z(x, y, z) \]
end
end
end

Computing \(\text{depth}(Q, x, y)\)

- Have \(z\) coordinate at 3 vertices
- How do we compute \(z\) at pre-image of projected point?

Computing \(\text{depth}(Q, x, y)\)

- Linear transformations preserve straight lines

1. Compute \(z_{12}\) and \(p_{12}\)
2. Express \(p\) with \(p_3\) and \(p_{12}\)
3. Compute \(z\) in the same way from \(z_3\) and \(z_{12}\)

Linear Interpolation

On a line

- Input: 2 points, 2 values
- Output: value at any point \(p\) on the line \(L(t)\) between them

\[ L(t) = \text{tp}_1 + (1 - t)p_0 \quad t \in [0, 1] \]

\[ L(0) = p_0 \quad L(1) = p_1 \quad L(0.5) = \frac{p_0 + p_1}{2} \]

\[ t_{12} = \frac{|p_3 - p_{12}|}{|p_3 - p_2|} \quad t_{12} = \frac{|p_3 - p_{12}|}{|p_2 - p_1|} \]

\[ f(t) = f(t) = t_{12} + (1 - t_{12})z_1 \]

Apply linear interpolation on a line twice:
\[ z_{12} = t_{12}z_1 + (1 - t_{12})z_2 \]

\[ z_{13} = t_{13}z_1 + (1 - t_{13})z_3 \]

\[ \alpha = \beta + \gamma \]

\[ f = \alpha p_1 + \beta p_2 + \gamma p_3 \]

\[ a_1 + a_2 + a_3 = 1 \]

\[ a_i \geq 0 \]

\[ f(p) = \alpha f_1 + \beta f_2 + \gamma f_3 \]

But \( a_i = ? \)

Z-Buffer – depth (\( Q, x, y \))

Barycentric Coordinates

\[ a_1 = \frac{A_1}{A_1 + A_2 + A_3} \]

\[ B.C. \text{ of all interior points are } a_i > 0. \]

\[ \text{Triangle centroid } = (x_1/3, y_1/3, z_1/3). \]

Z-Buffer Project (\( P \))

Z-Buffer Algorithm

- Image space algorithm
- Data structure: Array of depth values
- Implemented in hardware due to simplicity
- Depth resolution of 32 bits is common
- Scene may be updated on the fly, adding new polygons

Copyright
C. Gotsman, G. Elber, M. Ben-Chen
Computer Science Dept., Technion
Z Fighting

When Z-buffer has low precision and/or $\alpha$ is not chosen correctly

Transparency Z-Buffer

How can we emulate transparent objects?

Transparency Z-Buffer

- Extension to the basic Z-buffer algorithm
- Save all pixel values
- At the end – have list of polygons & depths (order) for each pixel
- Simulate transparency by weighting the different list elements, in order
- Do we really need to store all pixel values?

The Graphics Pipeline

- Hardware implementation of screen Z-buffer:
  - Polygons sent through pipeline one at a time
  - Display updated to reflect each new polygon

Scan-Line Z-Buffer Algorithm

- In software implementations - amount of memory required for screen Z-buffer may be prohibitive

Scan-Line Z-Buffer Algorithm

- Scan-line Z-buffer algorithm:
  - Render the image one line at a time
  - Take into account only polygons affecting this line

- Combination of polygon scan-conversion & Z-buffer algorithms
- Only Z-buffer the size of scan-line is required.
- Entire scene must be available in advance
- Image cannot be updated incrementally
Scan-Line Z-Buffer Algorithm

```
ScanLineZBuffer(Scene)
    Scene2D := Project(Scene);
    Sort Scene2D into buckets of polygons P in increasing YMin(P) order;
    A := EmptySet;
    for y := YMin(Scene2D) to YMax(Scene2D) do
        for each pixel (x, y) in scanline Y=y do
            SetZ(x, MaxZ);
        end;
        A := A + {P in Scene : YMin(P) <= y};
        A := A - {P in A : YMax(P) < y};
        for each polygon P in A
            for each pixel (x, y) in P's span(s) on the scanline
                z := Depth(P, x, y);
                if (z < GetZ(x)) then
                    PutColor(x, y, Col(P));
                    PutZ(x, z);
                end;
            end;
        end;
    end;
```

Line and Polygon Clipping Algorithms

- Cohen-Sutherland
- Sutherland-Hodgman
- Liang-Barksy
- Cyrus-Beck