Visibility

Determining which objects / triangles / pixels can be seen

1. view volume culling
2. view volume clipping
3. backface culling
4. occlusion: z-buffer test
5. occlusion: object culling

Raytracing

For each pixel in image, compute a ray and see what it hits in the scene.
(1) View Volume Culling (for triangles)

Idea: If vertices are outside view volume then cull.

Revised Idea:

Cull iff all vertices are outside w.r.t. the same view volume plane.

(1) View Volume Culling (for objects)

Idea: Fast test to cull entire object

Bounding sphere:
Cull if \( \text{dist}(C_{p}, \text{plane}) < -r \)

Bounding box:
Cull if all 8 vertices are "outside" w.r.t. any one of the view volume planes.
(2) View Volume Clipping

General polygon clipping:
Clip against each of the 6 planes in turn.

For triangles with bounding-box scan conversion:

Clipping in VCS

Plane equations

Orthographic View Volume

- left: \( x - \text{left} = 0 \)
- right: \(-x + \text{right} = 0 \)
- bottom: \( y - \text{bottom} = 0 \)
- top: \(-y + \text{top} = 0 \)
- front: \(-z + \text{near} = 0 \)
- back: \( z + \text{far} = 0 \)

Perspective View Volume

- left: \( x + \text{left} \times z/\text{near} = 0 \)
- right: \(-x - \text{right} \times z/\text{near} = 0 \)
- top: \(-y - \text{top} \times z/\text{near} = 0 \)
- bottom: \( y + \text{bottom} \times z/\text{near} = 0 \)
- front: \(-z - \text{near} = 0 \)
- back: \( z + \text{far} = 0 \)
Clipping in NDCS (?)

NDCS:

\[ \begin{align*}
\text{left:} & \quad x + h = 0 \\
\text{right:} & \quad -x + h = 0 \\
\text{bot:} & \quad y + h = 0 \\
\text{top:} & \quad -y + h = 0 \\
\text{near:} & \quad z + h = 0 \\
\text{far:} & \quad -z + h = 0
\end{align*} \]

Clipping in CCS

NDCS: 
-1 \leq x_{\text{NDCS}} \leq 1

CCS:
-h_{\text{CCS}} \leq x_{\text{CCS}} \leq +h_{\text{CCS}}

canonical plane equations:

- \text{left: } x + h = 0
- \text{right: } -x + h = 0
- \text{bot: } y + h = 0
- \text{top: } -y + h = 0
- \text{near: } z + h = 0
- \text{far: } -z + h = 0

would be convenient, because the plane equations remain fixed
Line-Plane intersection

**Plane equation**

\[ \mathbf{N} = (\mathbf{P}_2 - \mathbf{P}_0) \times (\mathbf{P}_1 - \mathbf{P}_0) \]

\[ A x + B y + C z + D = 0 \]

\[ \mathbf{N} \cdot \mathbf{P} + D = 0 = F(\mathbf{P}) \]

where \( D = -\mathbf{N} \cdot \mathbf{P}_i \)

\[ \mathbf{N} \cdot (\mathbf{P}_a + t(\mathbf{P}_b - \mathbf{P}_a)) + D = 0 \]

\[ \mathbf{N} \cdot \mathbf{P}_a + t[(\mathbf{N} \cdot \mathbf{P}_b - \mathbf{N} \cdot \mathbf{P}_a)] + D = 0 \]

\[ d = \frac{d_1}{d_1 + d_2} \]

**Line equation**

\[ \mathbf{P}(t) = \mathbf{P}_a + t(\mathbf{P}_b - \mathbf{P}_a) \]

\[ t = -\frac{\mathbf{N} \cdot \mathbf{P}_a - D}{-\mathbf{N} \cdot \mathbf{P}_b - \mathbf{N} \cdot \mathbf{P}_a} = \frac{-F(\mathbf{P}_a)}{F(\mathbf{P}_b) - F(\mathbf{P}_a)} \]

**(3) Backface Culling in VCS**

Idea: cull if \( N_z < 0 \)

but should be culled

\[ N_z > 0 \]

above

Correct VCS culling rule:

Cull if \( \mathbf{P}(0,0,0) \) is below the plane of the polygon,

where "above" = side of the Normal

\[ \mathbf{N} \cdot \mathbf{P} + D = 0 = F(x,y,z) \]

Cull if \( F(0,0,P) < 0 \) i.e. cull if \( D < 0 \)
(3) Backface Culling in NDCS

Cull if $N_z$ in NDCS is $>0$

Transforming Normals

Using $h=0$

Problem (occurs for non-uniform scaling)

Scale $\left(\frac{2}{1}, 1\right)$

$N(1,1)$

$N(2,1)$ wrong answer

$\frac{2}{1}$

$\frac{1}{h}$

$0.0$
Transforming Normals

Consider a plane, before and after transformation:

\[ M : \text{transforms points in the usual way} \]
\[ Q : \text{transforms normals \text{-what should it be?}} \]

Let's write a plane equation for \( \mathbf{A} \):
\[
Ax + By + Cz + D = 0
\]
or \[
\begin{bmatrix}
A & B & C & D
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix} = 0
\]

Let's write this as \( \mathbf{N}^T \mathbf{p} = 0 \)

Similarly, for \( \mathbf{B} \), we can write \( \mathbf{N}^T \mathbf{p}' = 0 \)

Now substitute for \( \mathbf{N}' ; \mathbf{p}' = (\mathbf{Q} \mathbf{N})^T (\mathbf{M} \mathbf{p}) = 0 \)

\[
\Rightarrow \mathbf{N}^T \mathbf{Q}^T \mathbf{M} \mathbf{p} = 0 \quad \mathbf{Q}^T \mathbf{M} = \mathbf{I} \quad \Rightarrow \mathbf{Q} = (\mathbf{M}^{-1})^T
\]

(4) Occlusion: Z-buffer

View occluded by objects in front of a given pixel or polygon?

- **Image space algorithms:**
  - operate on pixels or scan-lines
  - visibility resolved to the precision of the display
  - e.g.: Z-buffer

- **Object space algorithms:**
  - explicitly compute visible portions of polygons
  - painter's algorithm: depth-sorting, BSP trees

Simple painter's algorithm will not work for these two triangles

Sort and draw from back-to-front.
store \( (r,g,b,z) \) for each pixel

\[
\text{for all } i,j \{
    \text{Depth}[i,j] = \text{MAX DEPTH}
    \text{Image}[i,j] = \text{BACKGROUND_COLOUR}
\}
\]

\[
\text{for all polygons } P \{
    \text{project vertices into screen-space, i.e., DCS}
    \text{for all pixels in } P \{
        \text{if } (Z_{\text{pixel}} < \text{Depth}[i,j]) \{ \text{ // closer?}
            \text{Image}[i,j] = C_{\text{pixel}} \text{ // overwrite pixel}
            \text{Depth}[i,j] = Z_{\text{pixel}} \text{ // overwrite } z
        \}
    \}
\}
\]

Z-buffer

- hardware support
- extra memory
- jaggies, i.e., steps along intersections
- poor performance for high depth complexity scenes;
  - use occlusion culling to mitigate this

-resolution issues in \( z \): "\( z \)-fighting"
-result can still depend on rendering order
-for cases of identical depths for a fragment
(5) Occlusion: Object culling

- **occlusion queries**
  - virtual render of bounding box

- **precomputed visibility tables**
  - *store a list of visible cells*

- **horizon maps**
  - *for terrain models*

---

**Visibility in Practice:**

**WebGL, OpenGL**

Commonly supported by hardware & OpenGL / DirectX

- view volume culling (for triangles)
- view volume clipping
- backface culling
- z-buffer occlusion test

Software, i.e., on your own

- view volume culling (for objects)
- occlusion culling
Raytracing

**alternative to projective rendering**

- for each pixel $p$
  - construct ray $r$ from eye through $p$
  - intersect $r$ with all polygons or objects
  - color $p$ according to closest surface