Perspective Projection (continued)
Perspective Projection

\[
P = \begin{pmatrix}
n & 0 & 0 & 0 \\
0 & n & 0 & 0 \\
0 & 0 & n + f & -fn \\
0 & 0 & 1 & 0 \\
\end{pmatrix}
\]

\[M_{per} = M_{orth}P\]

This does not preserve \(z\) completely, but it preserves \(z = n, f\) and is monotone (preserves ordering) with respect to \(z\)
One-Point Perspective

- One principal face parallel to projection plane
- One vanishing point for cube
Two-Point Perspective

- On principal direction parallel to projection plane
- Two vanishing points for cube
Three-Point Perspective

• No principal face parallel to projection plane
• Three vanishing points for cube
Occlusion

“painter’s algorithm” draw primitives in back-to-front order
Occlusion

“painter’s algorithm”
draw primitives in back-to-front order

problem:
triangle intersection

who’s in front of whom?
Occlusion

“painter’s algorithm”
draw primitives in back-to-front order

**problem:**
occlusion cycle

also, sort primitives by depth is **slow**
Use a z-buffer for hidden surface removal

at each pixel, record distance to the closest object that has been drawn in a depth buffer
Use a **z-buffer** for hidden surface removal

at each pixel, record distance to the closest object that has been drawn in a *depth* buffer

- assume both spheres of the same size, red drawn last
Use a z-buffer for hidden surface removal

- done in the fragment blending phase
  - each fragment must carry a depth
Use a z-buffer for hidden surface removal

http://www.beyond3d.com/content/articles/41/
Backface culling: another way to eliminate hidden geometry
Hidden Surface Removal in OpenGL

```c
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);

glEnable(GL_DEPTH_TEST);

glEnable(GL_CULL_FACE);

Note: For a perspective transformation, there is more in the depth buffer for z-values closer to the near plane
```
Clipping
What’s the equation for the plane through $q$ with normal $N$?
implicit line equation:

\[ f(X) = N \cdot (X - X_0) = 0 \]

\[ X_0 = (x_0, y_0) \]
Clipping against a plane

What’s the equation for the plane through \( q \) with normal \( \mathbf{N} \)?

\[
f(p) = ? = 0
\]

<whiteboard>
Clipping against a plane

What’s the equation for the plane through \( q \) with normal \( N \)?

\[
f(p) = N \cdot (p - q) = 0
\]

\[
f(p) = N \cdot p + D = 0
\]
Intersection of line and plane

How can we distinguish between these cases?
Intersection of line and plane

\[ f(a)f(b) \geq 0 \]

\[ f(a)f(b) < 0 \]
Intersection of line and plane

How can we find the intersection point?

<whiteboard>
Clipping against the viewing volume
Orthographic projection viewing volume
Perspective projection
viewing volume

clip coordinates
Clipping against the viewing volume

\[ t = \frac{N \cdot a + D}{N \cdot (a - b)} \]

N is particularly simple for the orthographic viewing volume in NDC.
Clipping against the viewing volume

\[ s = \frac{N \cdot c + d}{N \cdot (c - b)} \]

\[ t = \frac{N \cdot a + D}{N \cdot (a - b)} \]

N is particularly simple for the orthographic viewing volume in NDC
Clipping against the viewing volume

\[ s = \frac{N \cdot c + d}{N \cdot (c - b)} \]

\[ t = \frac{N \cdot a + D}{N \cdot (a - b)} \]

need to generate new triangles

N is particularly simple for the orthographic viewing volume in NDC.
Clipping

- Removing the unseen geometry
- Direct (brute-force) solution - solve simultaneous equations for intersections of lines/edges at window edges

A point or vertex is visible if
x_{left} < x < x_{right}
and
y_{bot} < y < y_{top}
Clipping lines

Pipeline, clip each edge of the window separately:
Clip the vertices that are outside of the window and create new vertices at window border.

Result is still a single polygon but may have more vertices and an odd shape.
Clipping polygons
Clipping polygons

**Bounding box** - surrounds each polygon
Cohen-Sutherland Algorithm

- Region Checks: Trivially reject or accept for clipping
- Good for large or small windows (all is in or out of window, respectively)
- Each vertex is assigned an 4-bit outcode

<table>
<thead>
<tr>
<th></th>
<th>1001</th>
<th>1000</th>
<th>1010</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0001</td>
<td>0000</td>
<td>0010</td>
</tr>
<tr>
<td></td>
<td>0101</td>
<td>0100</td>
<td>0110</td>
</tr>
</tbody>
</table>

A line can be **trivially accepted** if both endpoints have an outcode of 0000.
A line can be **trivially rejected** if any of the same two bits in the outcodes are both equal to 1 (both endpoints are left, right, above, below the window)
Clipping 3D

Adds far and near clipping planes for 3D viewing volume