Perspective Projection

\[ y' = \frac{d}{z} y \]

Both \( x \) and \( y \) get multiplied by \( d/z \)

[Shirley, Marschner]
Simple perspective projection

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1/d & 0
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix} =
\begin{pmatrix}
x \\
y \\
z \\
z/d
\end{pmatrix}
\implies
\begin{align*}
x' &= \frac{d}{z} x \\
y' &= \frac{d}{z} y \\
z' &= \frac{d}{z} z = d
\end{align*}

This achieves a simple perspective projection onto the view plane \( z = d \)

<whiteboard>

but we’ve lost all information about \( z \)!
Perspective Projection

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

\[z' = (n + f) - \frac{nf}{z}\]

Example:

\[
\begin{align*}
n &= -1 \\
f &= -2
\end{align*}
\]
\[ M_{\text{per}} = M_{\text{orth}} P \]
OpenGL Perspective Viewing

`glFrustum(xmin, xmax, ymin, ymax, near, far)`

Clipping volume (frustrum) for a perspective projection.
Using Field of View

With `glFrustum` it is often difficult to get the desired view. `gluPerspective(fovy, aspect, near, far)` often provides a better interface.

![Diagram showing front plane and aspect ratio](image)

*front plane*

*aspect = w/h*
Clipping after the perspective transformation can cause problems.
OpenGL clips after projection and before perspective division

\[-w \leq x \leq w\]
\[-w \leq y \leq w\]
\[-w \leq z \leq w\]
Hidden Surface Removal
Hidden Surface Removal
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
Occlusion

“painter’s algorithm”

draw primitives in back-to-front order

problem:
occlusion cycle
Use a z-buffer for hidden surface removal
test depth on a pixel by pixel basis

red drawn last

without z-buffer  with z-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*
Use a *z-buffer* for hidden surface removal

**Figure 1.** Block diagram of OpenGL.
Use a **z-buffer** for hidden surface removal

[http://www.beyond3d.com/content/articles/41/](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);
```

For a perspective transformation, there is more precision in the depth buffer for z-values closer to the near plane.