Computer Graphics

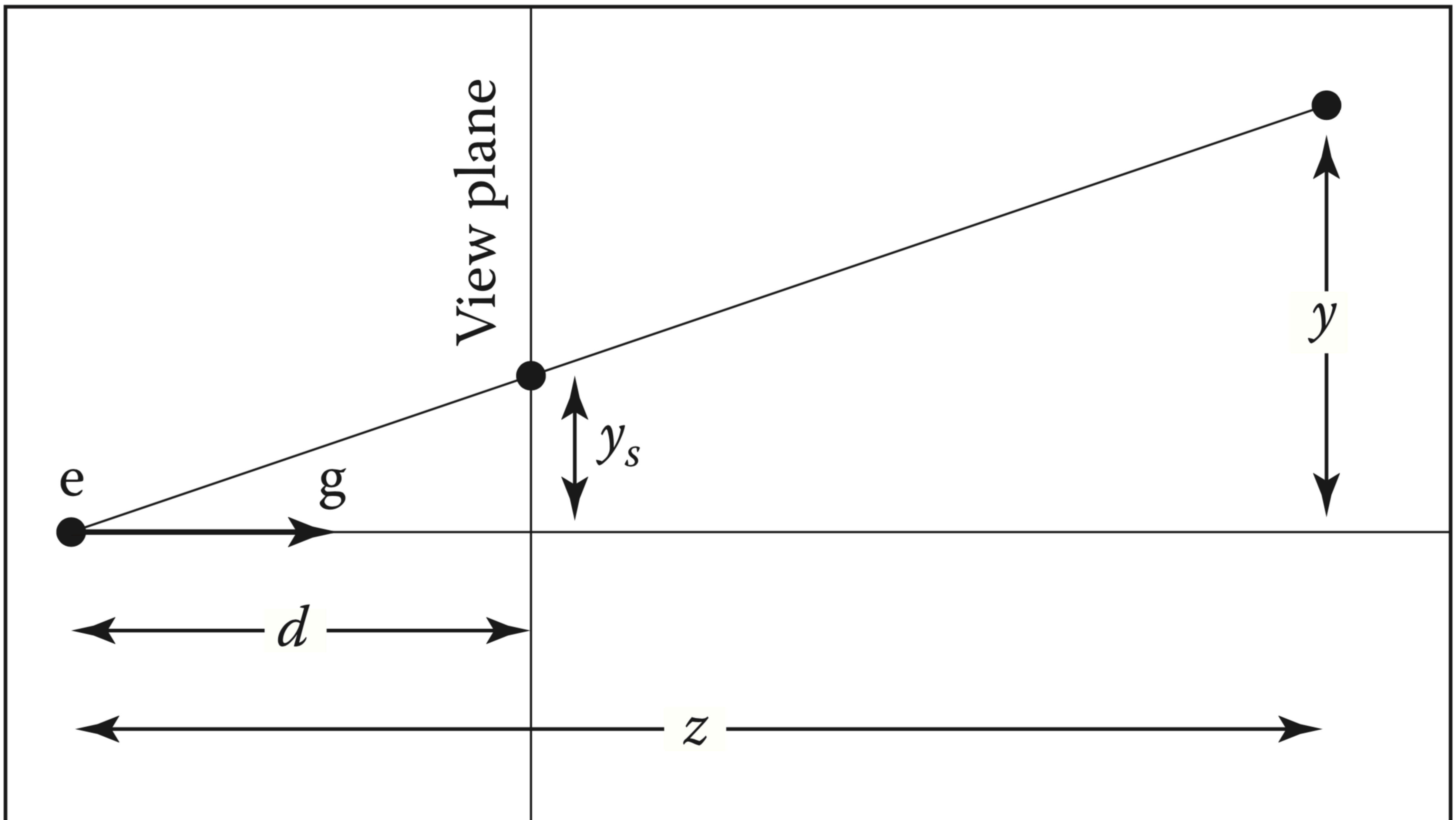
Lecture #8

Projective Transformations

Relative Size Based on Distance

- Key idea of perspective: the size of an object on the screen is proportional to 1/z

$$y_s = \frac{d}{z}\,y$$



Problem: How to divide by z?

- Linear transformations: $x' = ax + by + cz$

Problem: How to divide by z?

- Linear transformations: $$x' = ax + by + cz$$
- Affine transformations: $$x' = ax + by + cz + d$$

Problem: How to divide by z?

- Linear transformations: $$x' = ax + by + cz$$

- Affine transformations: $$x' = ax + by + cz + d$$

- Our trick: using w in homogeneous coordinates as a denominator: $$x' = \frac{a_1 x + b_1 y + c_1 z + d_1}{ex + fy + gz + h}$$

Problem: How to divide by z?

- Linear transformations:

$$x' = ax + by + cz$$

- Affine transformations:

$$x' = ax + by + cz + d$$

- Our trick: using w in homogeneous coordinates as a denominator:

- Same denominator for all coordinates.

$$\begin{aligned} x' &= \frac{a_1 x + b_1 y + c_1 z + d_1}{ex + fy + gz + h} \\ y' &= \frac{a_2 x + b_2 y + c_2 z + d_2}{ex + fy + gz + h} \\ z' &= \frac{a_3 x + b_3 y + c_3 z + d_3}{ex + fy + gz + h} \end{aligned}$$

Projective Transformations, or Homographies

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \widetilde{w} \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \\ e & f & g & h \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Where we reinterpret coordinates by diving by w:

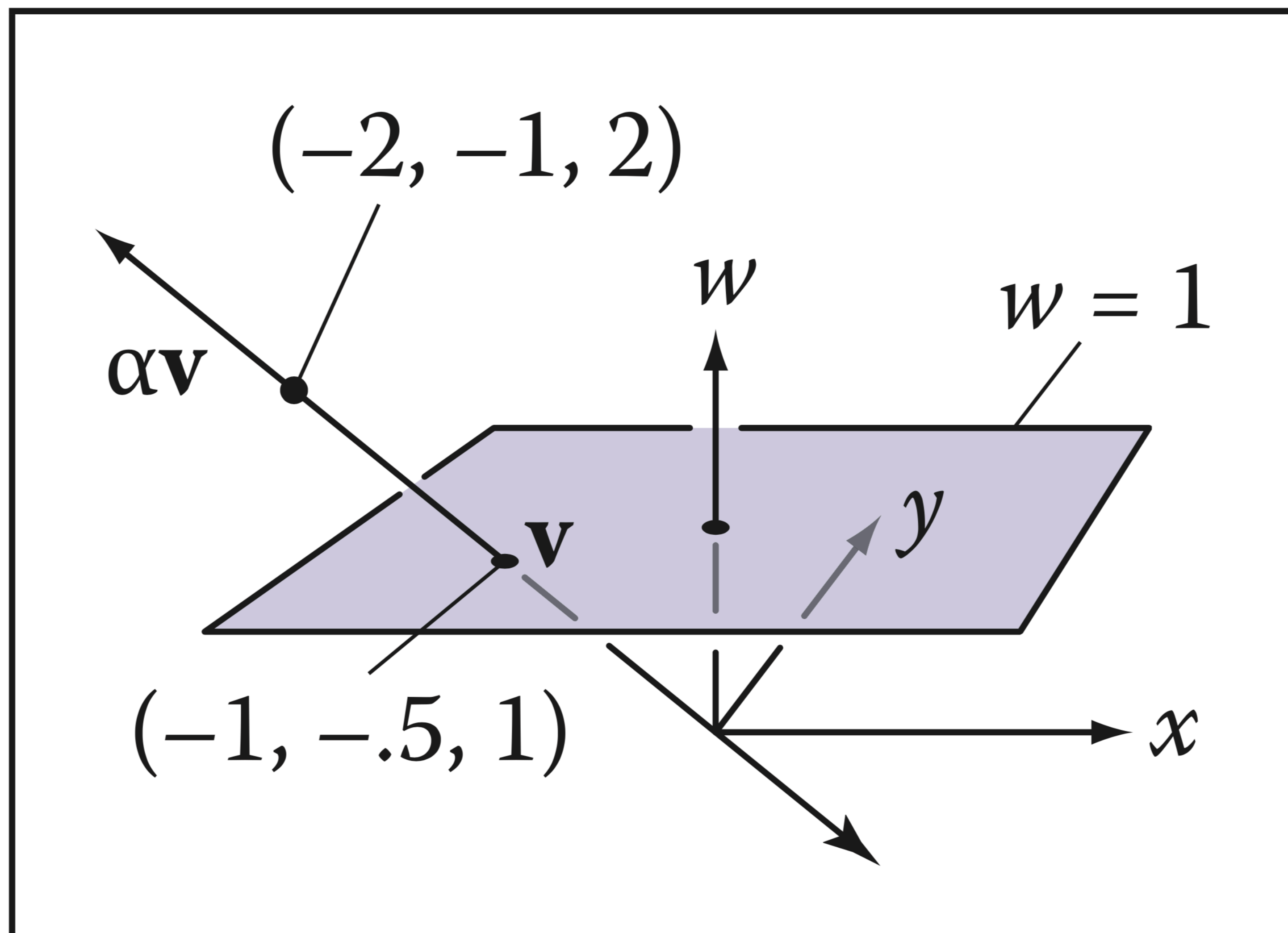
$$\left(x', y', z'\right) = (\tilde{x}/\widetilde{w}, \tilde{y}/\widetilde{w}, \tilde{z}/\widetilde{w})$$

Equivalence of Points

- Key idea: all scalar multiples of a vector are the same!
- Equivalently: we're treating points as lines in one dimension higher

$$\mathbf{x} \sim \alpha\mathbf{x}$$
$$\text{for all } \alpha \neq 0$$

We will only divide by w when we want the Cartesian coordinates

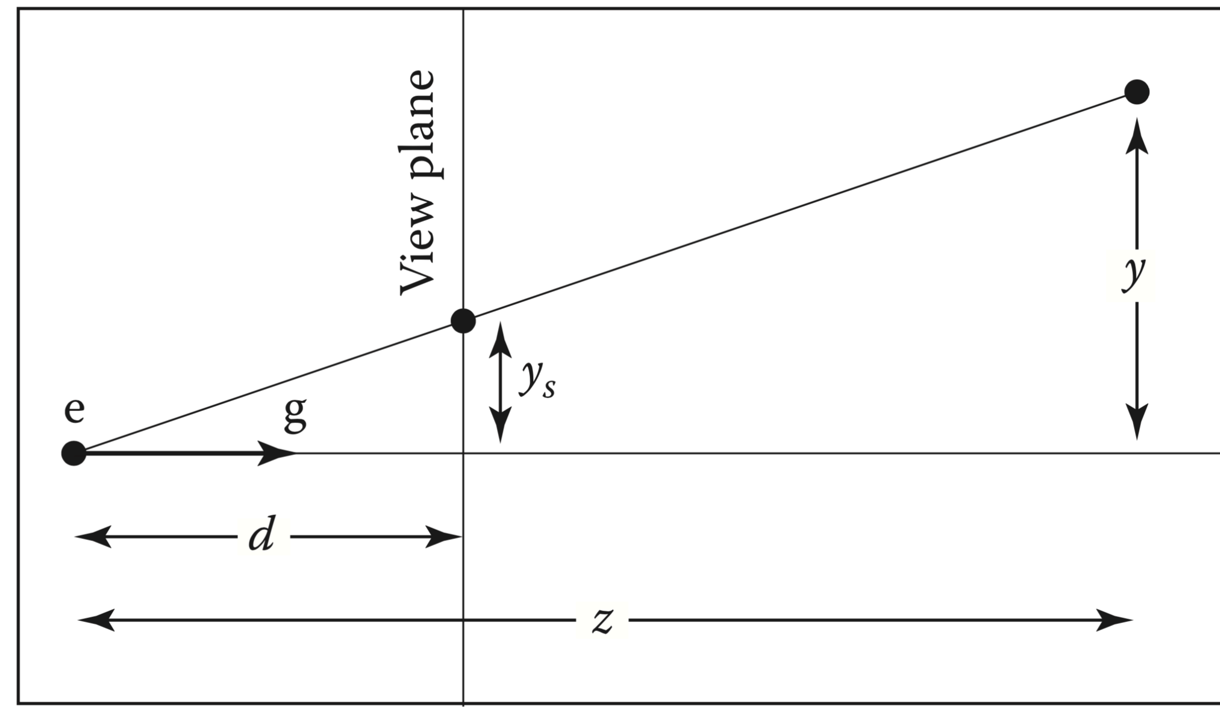


Perspective Projection

Using Homographies for Perspective

- We can now replace:

$$y_s = \frac{d}{z} y$$



- With:

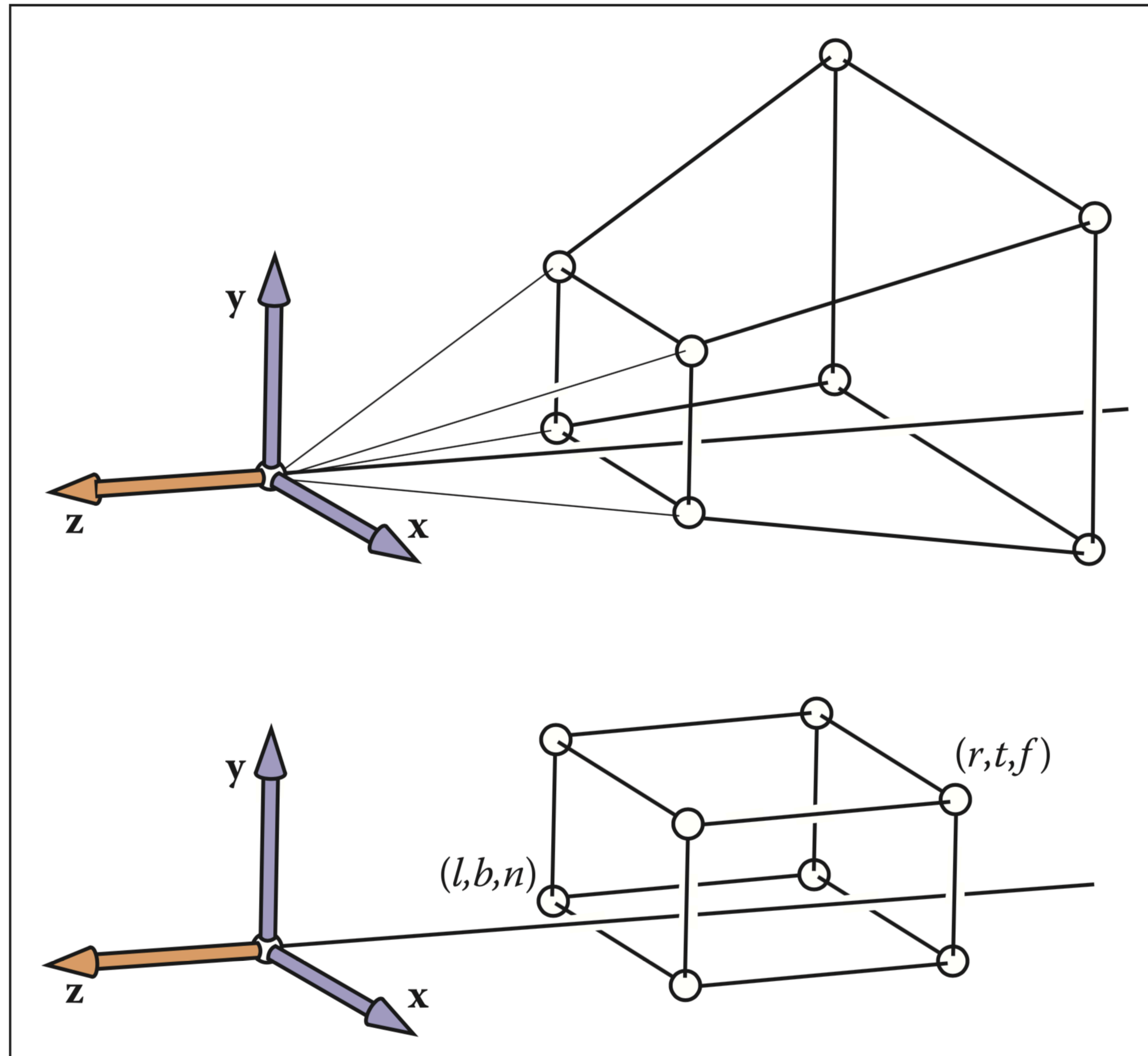
$$\begin{bmatrix} y_s \\ 1 \end{bmatrix} \sim \begin{bmatrix} d & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} y \\ z \\ 1 \end{bmatrix}$$

Perspective Matrix

- Our matrix:

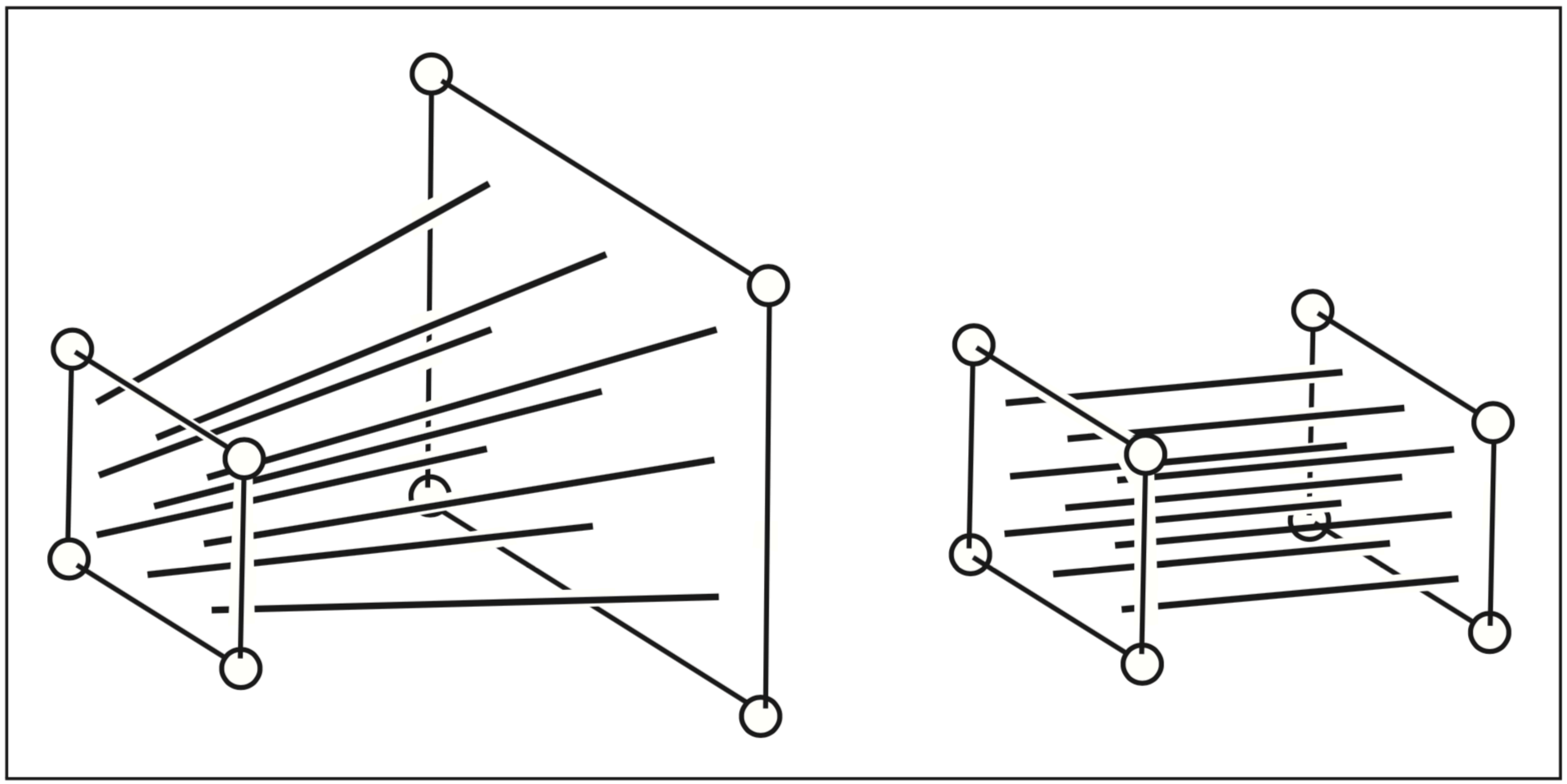
$$\mathbf{P} = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- Keeps near plane fixed, maps far plane to back of the box



Perspective Distortion

- Effect on view rays / lines:



- Note that affine transformation cannot do this because it keeps parallel lines parallel

Perspective Distortion

- Perspective matrix effect on coordinates is nonlinear distortion in z:

$$\mathbf{P}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Perspective Distortion

- Perspective matrix effect on coordinates is nonlinear distortion in z:

$$\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Perspective Distortion

- Perspective matrix effect on coordinates is nonlinear distortion in z:

$$\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} nx \\ ny \\ (n+f)z - fn \\ z \end{bmatrix} \sim \begin{bmatrix} \frac{nx}{z} \\ \frac{ny}{z} \\ n + f - \frac{fn}{z} \\ 1 \end{bmatrix}$$

Perspective Distortion

- Perspective matrix effect on coordinates is nonlinear distortion in z:

$$\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} nx \\ ny \\ (n+f)z - fn \\ z \end{bmatrix} \sim \begin{bmatrix} \frac{nx}{z} \\ \frac{ny}{z} \\ n+f-\frac{fn}{z} \\ 1 \end{bmatrix}$$

- But it does, however, preserve order in the z-coordinate (which will become useful very soon)

Perspective Projection Matrix

- Concatenating the perspective matrix with the orthographic projection provides the perspective projection matrix:

$$\mathbf{M}_{\mathrm{per}} = \mathbf{M}_{\mathrm{orth}}\mathbf{P} \qquad \mathbf{M}_{\mathrm{per}} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{l+r}{l-r} & 0 \\ 0 & \frac{2n}{t-b} & \frac{b+t}{b-t} & 0 \\ 0 & 0 & \frac{f+n}{n-f} & \frac{2fn}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- We can define l, r, b, and t relative to the near plane, since we keep it fixed

Putting it all together

Equivalently:

$$\mathbf{M} = \mathbf{M}_{\text{vp}} \mathbf{M}_{\text{orth}} \mathbf{P} \mathbf{M}_{\text{cam}}$$

```
construct M_vp
construct M_per
construct M_cam
M = M_vp * M_per * M_cam
for each 3D object O {
  O_screen = M * O
  draw(O_screen)
}
```

For a given vertex $\mathbf{a} = (x, y, z)$, $\mathbf{p} = \mathbf{M}\mathbf{a}$ should result in drawing $(x_{\mathbf{p}}/w_{\mathbf{p}}, y_{\mathbf{p}}/w_{\mathbf{p}}, z_{\mathbf{p}}/w_{\mathbf{p}})$ on the screen

Lec20 Required Reading

- FOCG, Ch. 8