Viewing II

Viewing Transformations

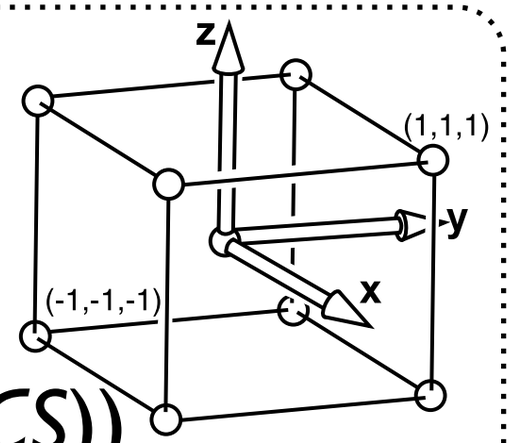
Introduction to Computer Graphics
CSE 533 / 333

Ojaswa Sharma

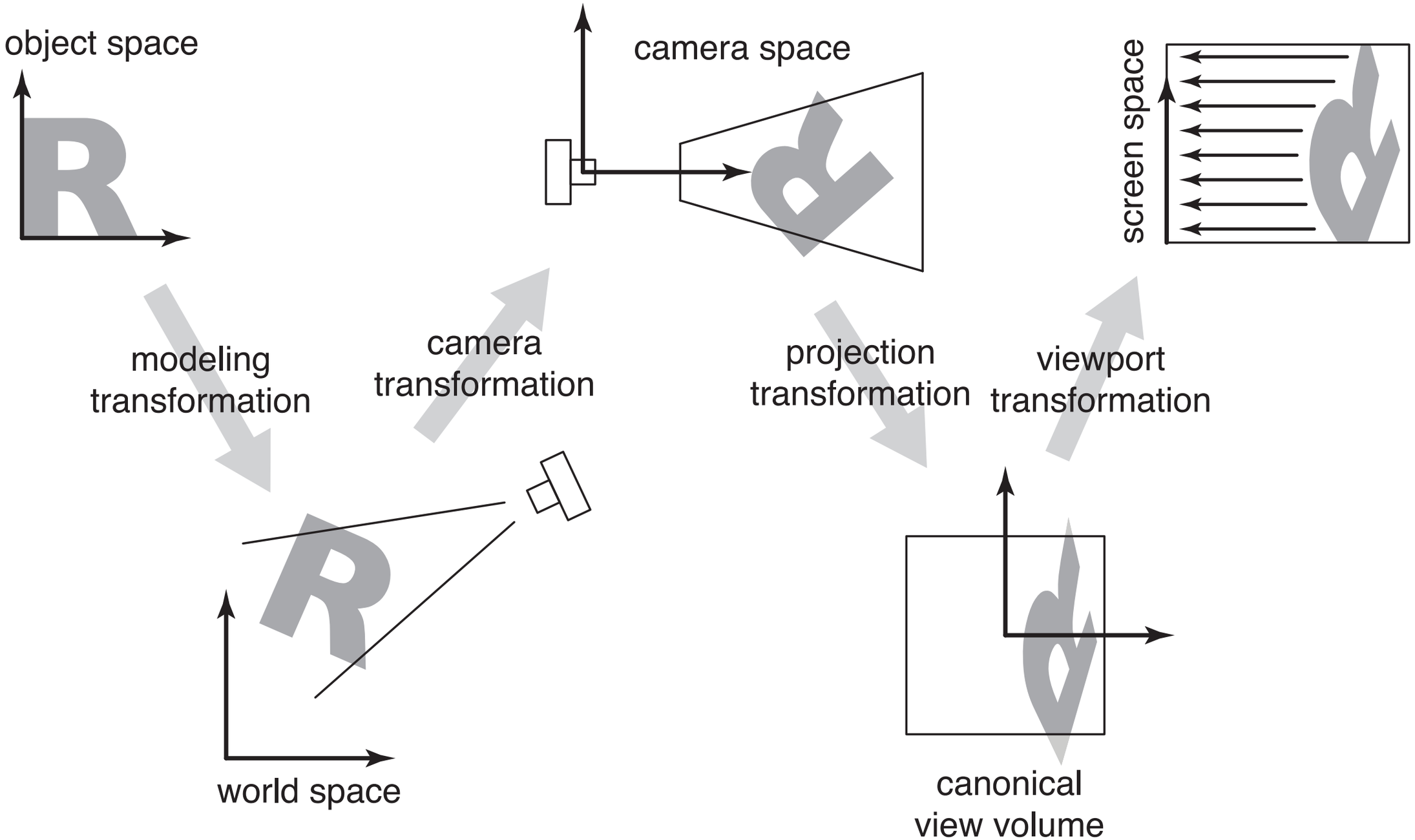
Viewing Transformations

- Mapping 3D locations in the *canonical coordinate system* to coordinates in the image. Composed of:
  - A camera transformation or eye transformation
  - A projection transformation
  - A viewport transformation or windowing transformation

Canonical View Volume is a cube containing all 3D points whose Cartesian coordinates are between -1 and +1, i.e., $(x, y, z) \in [-1, +1]^3$.
(also *Normalized Device Coordinate System (NDCS)*)

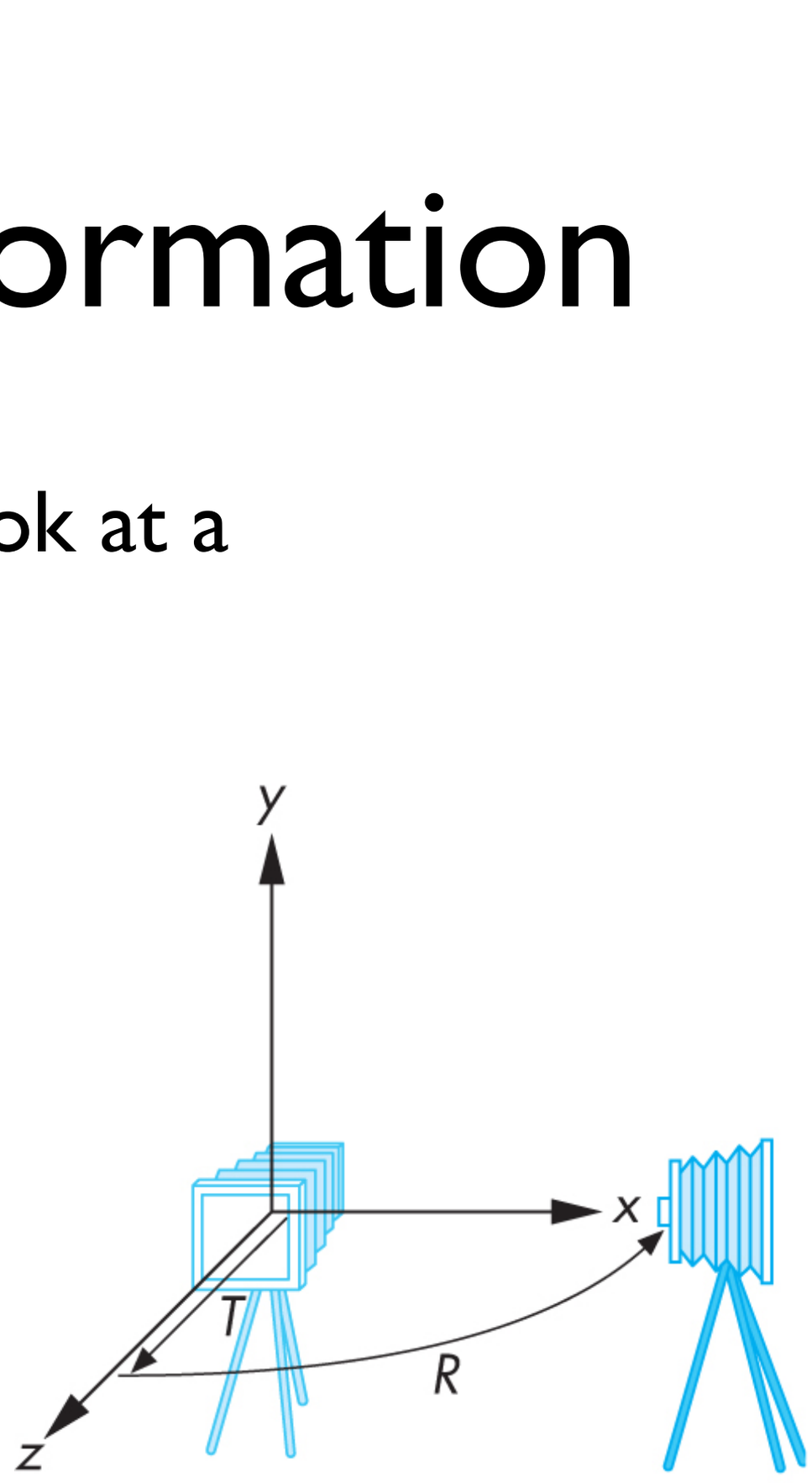


Viewing Transformations



Camera Transformation

- Positioning the camera to look at a particular direction
- the eye position *e*,
- the gaze direction *g*,
- the view-up vector *t*



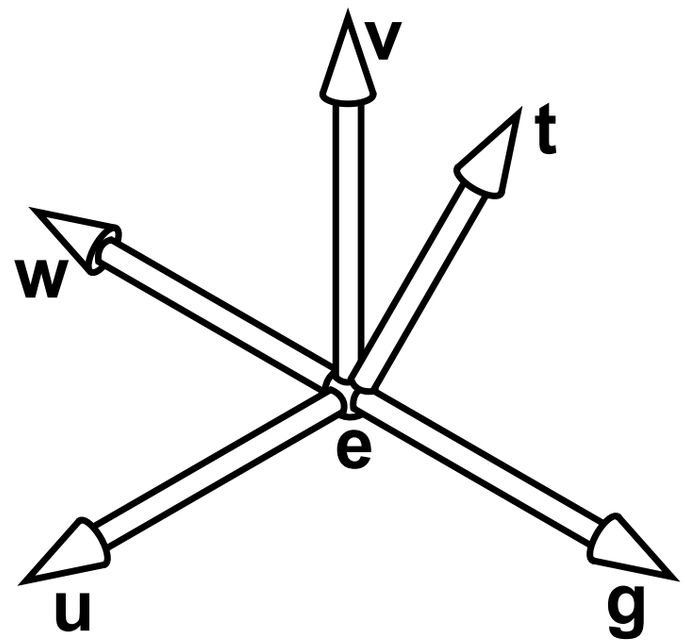
Camera Transformation

- The user specifies viewing as a triple (*e*, *g*, *t*)
- We can set up a coordinate system with origin e and *uvw* basis

$$w = -\frac{g}{\|g\|}$$

$$u = \frac{t \times w}{\|t \times w\|}$$

$$v = w \times u$$



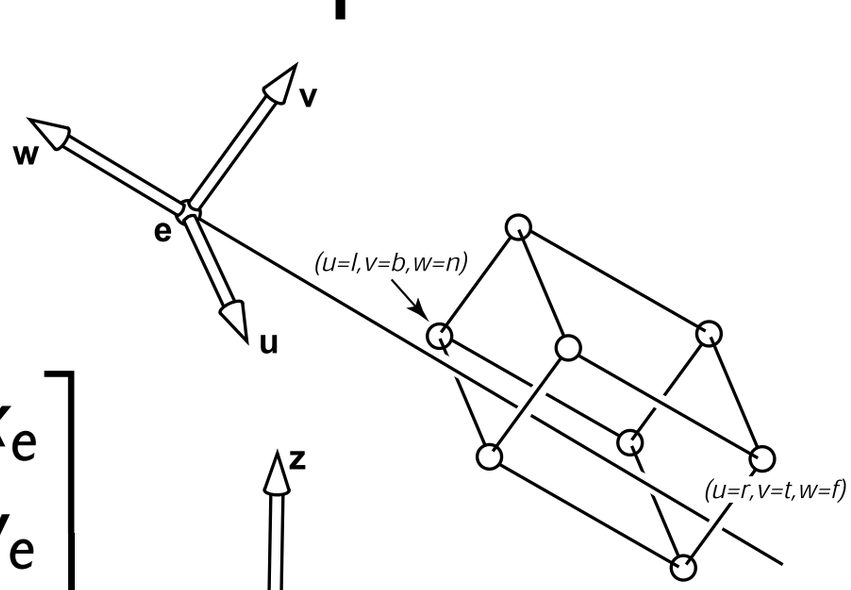
The camera in a right handed coordinate system is placed looking negative z axis since this will make the x-axis point right and the y-axis point up.

Camera Transformation

- Model coordinates are stored in terms of the canonical (or world) origin *o* with basis *xyz*
- Convert coordinates from xyz to uvw space

$$M_{Cam} = \begin{bmatrix} u & v & w & e \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

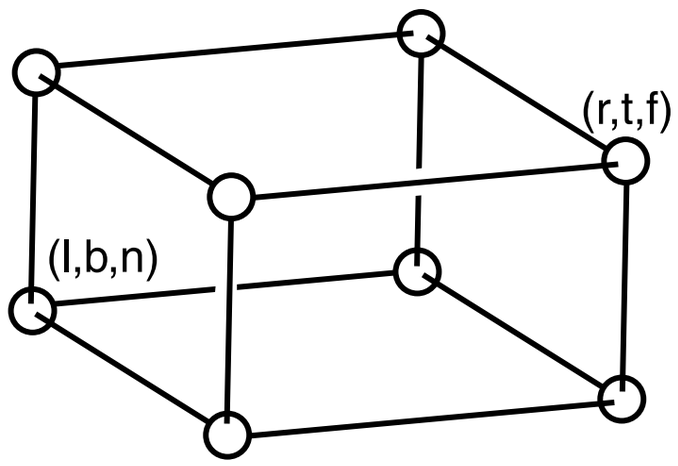


Projection Transformation

- Projectors point from camera space to the canonical view volume (retaining z-coordinate)
- Two fundamental projections:
  - Orthographic
  - Perspective

Orthographic Projection

- Camera looking along -z, with +y up
- View volume is an axes-aligned box called the *orthographic view volume*: $[l, r] \times [b, t] \times [f, n]$

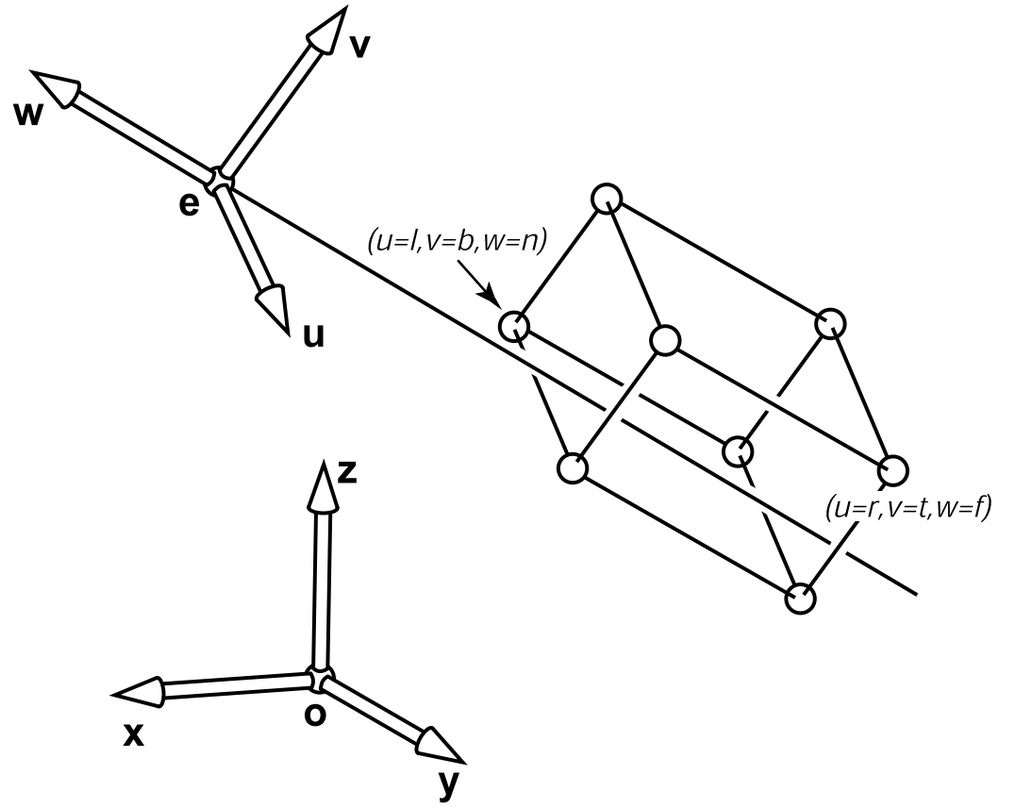


l	left plane
r	right plane
b	bottom plane
t	top plane
n	near plane
f	far plane

Orthographic Projection

- Orthographic to canonical view volume is just another windowing transform:

$$M_{orth} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Projective Transformations

Key observation:

- Size of an object on the screen is proportional to 1/z for an eye at origin looking up the negative z-axis:

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

Projective Transformations

- Linear transformation

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

- Affine transformation

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

- Projective transformation / homography

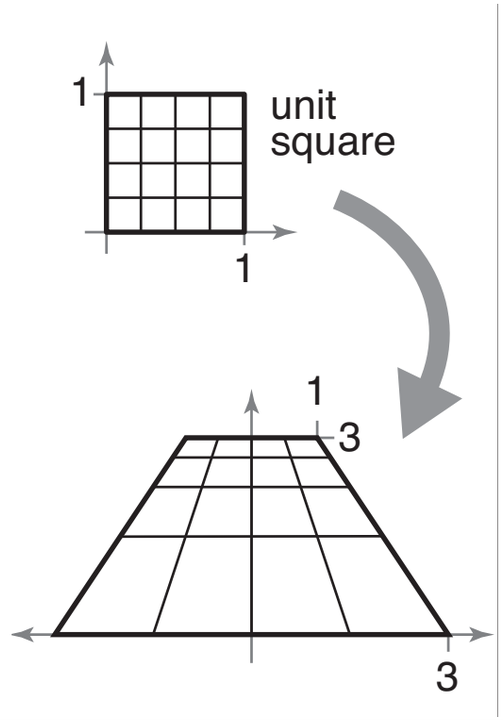
$$x' = \frac{ax + by + cz + d}{ex + fy + gz + h}$$

Projective Transformations

- Division by z cannot be achieved using affine transformations!
- Use **homogeneous coordinates** to generalise the matrix machinery to *projective transformations*

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

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



Perspective Projection

- Camera at origin, facing -z direction
- Near/far planes limit the view volume
- Near plane used as the projection plane (image plane distance is -*n*)

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

Perspective Projection

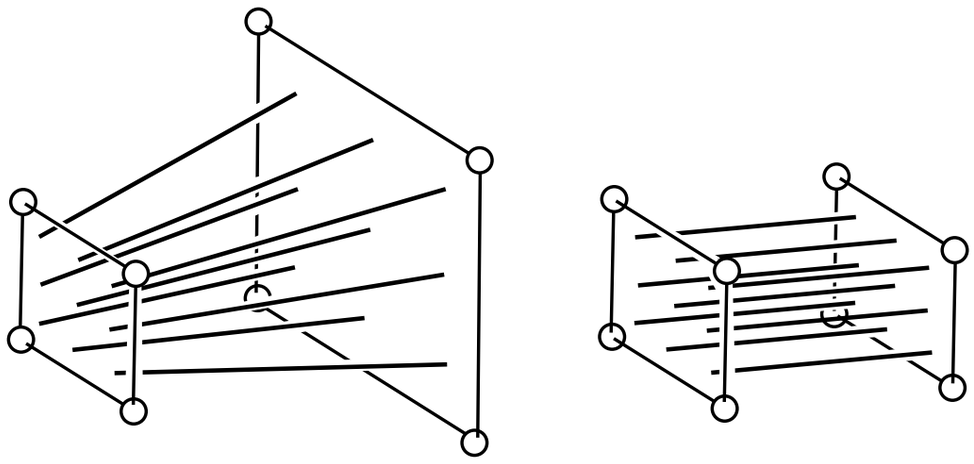
- *P* leaves points on ($z = n$)-plane entirely alone, but "squish" points on ($z = f$)-plane in x and y appropriately

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

Exercise: prove that P preserves the relative order of z values between z = n and z=f.

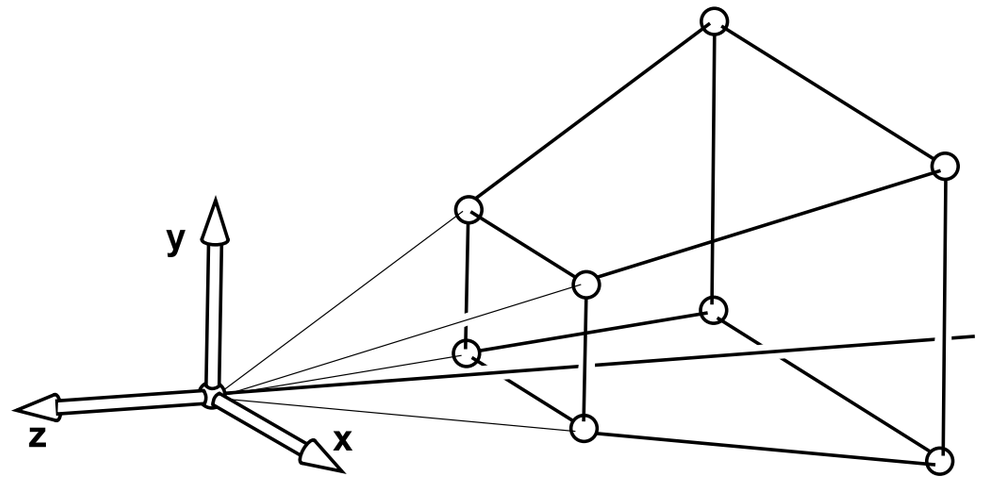
Perspective Projection

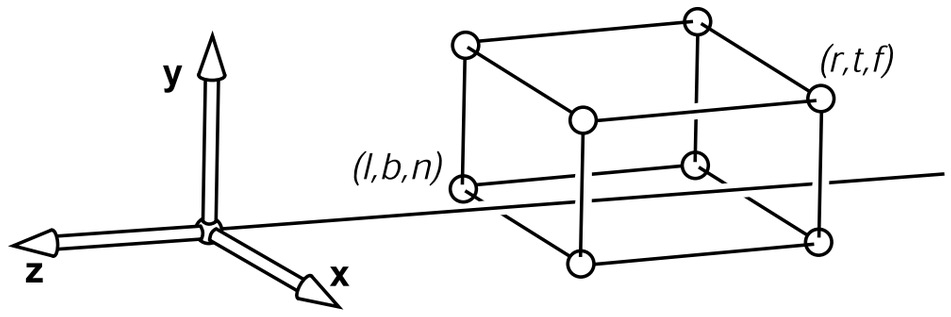
- *P* leaves points on ($z = n$)-plane entirely alone, but "squish" points on ($z = f$)-plane in x and y appropriately



Perspective Projection

- *P* simply maps the perspective view volume (*frustum*) to the orthographic view volume (axes aligned)





Perspective Projection

- Full perspective projection matrix is:

$$M_{per} = M_{orth} P$$

$$M_{per} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{l+r}{l-r} & 0 \\ 0 & \frac{2n}{t-b} & \frac{b+t}{b-t} & 0 \\ 0 & 0 & \frac{f+n}{n-f} & \frac{2fn}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

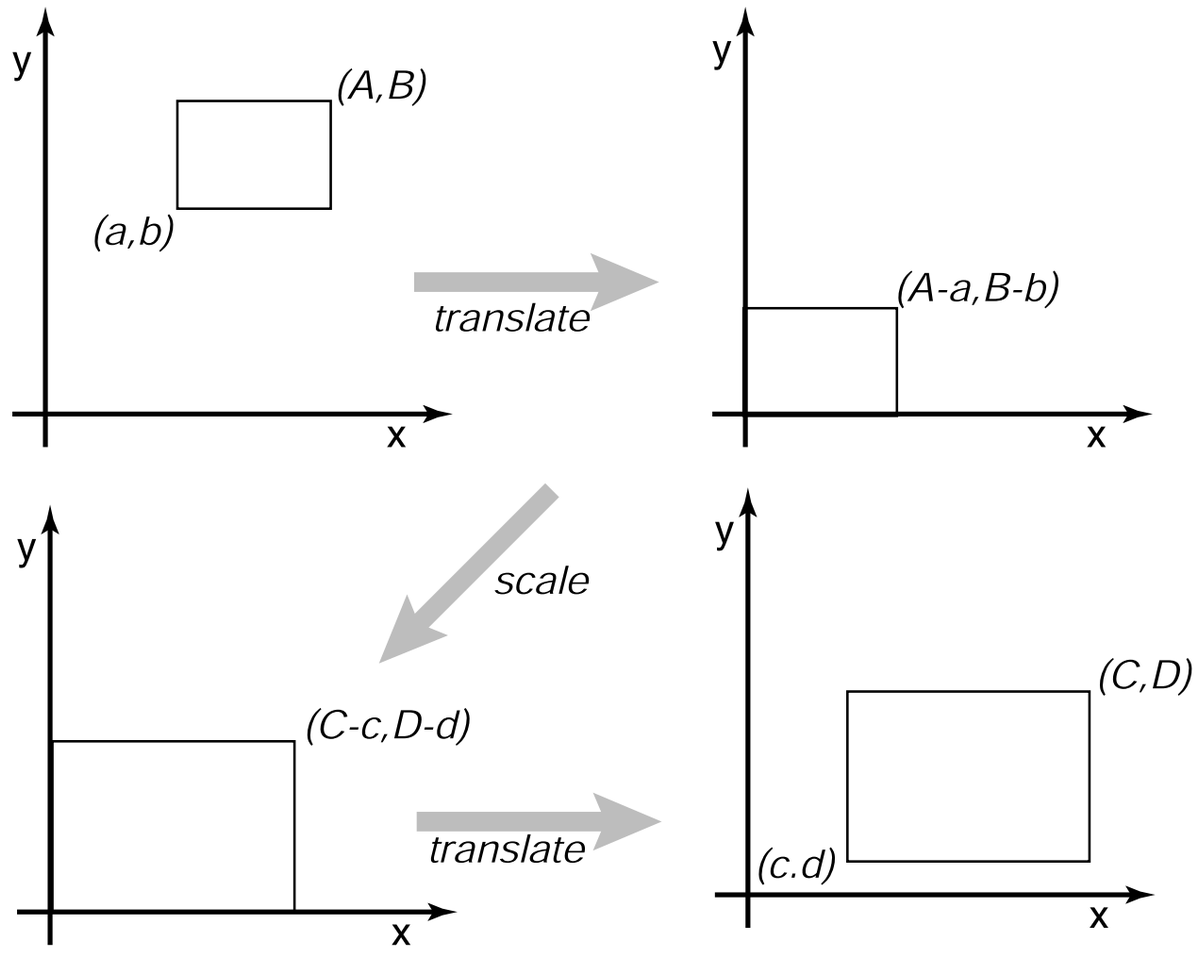
Viewport Transformation

- Geometry that we want to view is in canonical view volume
- Each pixel owns a unit square centred at integer coordinates
- Image boundaries have a half-unit overshoot from the pixel centres
- To draw to a window/image of size n_x x n_y, map

$$[-1, 1]^2 \mapsto [-0.5, n_x - 0.5] \times [-0.5, n_y - 0.5]$$

Windowing Transformation (2D)

A transformation to map points in $[x_l, x_h] \times [y_l, y_h]$ to $[x'_l, x'_h] \times [y'_l, y'_h]$



Windowing Transformation (2D)

Sequence of operations:

1. move (x_l, y_l) to origin
2. Scale the rectangle to be the same size as target rect
3. Move the origin to point (x'_l, y'_l)

Windowing Transformation (2D)

window

$$= \text{translate}\,(x_l', y_l')\ \text{scale}\left(\frac{x_h' - x_l'}{x_h - x_l}, \frac{y_h' - y_l'}{y_h - y_l}\right)\ \text{translate}\,(-x_l, -y_l)$$

$$= \begin{bmatrix} 1 & 0 & x_l' \\ 0 & 1 & y_l' \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{x_h' - x_l'}{x_h - x_l} & 0 & 0 \\ 0 & \frac{y_h' - y_l'}{y_h - y_l} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_l \\ 0 & 1 & -y_l \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{x_h' - x_l'}{x_h - x_l} & 0 & \frac{x_l' x_h - x_h' x_l}{x_h - x_l} \\ 0 & \frac{y_h' - y_l'}{y_h - y_l} & \frac{y_l' y_h - y_h' y_l}{y_h - y_l} \\ 0 & 0 & 1 \end{bmatrix}$$

Viewport Transformation

- Geometry that we want to view is in canonical view volume
- Each pixel owns a unit square centred at integer coordinates
- Image boundaries have a half-unit overshoot from the pixel centres
- To draw to a window/image of size n_x x n_y, map

$$[-1, 1]^2 \mapsto [-0.5, n_x - 0.5] \times [-0.5, n_y - 0.5]$$

Viewport Transformation

$$\begin{bmatrix} x_{screen} \\ y_{screen} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n_x}{2} & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & \frac{n_y-1}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{canonical} \\ y_{canonical} \\ 1 \end{bmatrix}$$

$$M_{vp} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note the z-coordinate in M_{vp} is preserved. This will be used later in visibility determination.

Projection Algorithm

compute M_{vp}
compute M_{per}
compute M_{cam}
$M = M_{vp} M_{per} M_{cam}$
foreach *line segment* (a_i, b_i) **do**
    $p = Ma_i$
    $q = Mb_i$
    drawline $\left(x_p/w_p, y_p/w_p, x_q/w_q\right)$

Reading

- FCG: 7

ICG: Interactive Computer Graphics, E. Angel, and D. Shreiner, 6th ed.
FCG: Fundamentals of Computer Graphics, P. Shirley, M. Ashikhmin, and S. Marschner, 3rd ed.