3D Viewing

Dr. Zhigang Deng

Viewing, backward and forward

- So far have used the backward approach to viewing
  - start from pixel
  - ask what part of scene projects to pixel
  - explicitly construct the ray corresponding to the pixel
- Next will look at the forward approach
  - start from a point in 3D
  - compute its projection into the image
- Central tool is matrix transformations
  - combines seamlessly with coordinate transformations used to position camera and model
  - ultimate goal: single matrix operation to map any 3D point to its correct screen location.

Forward viewing

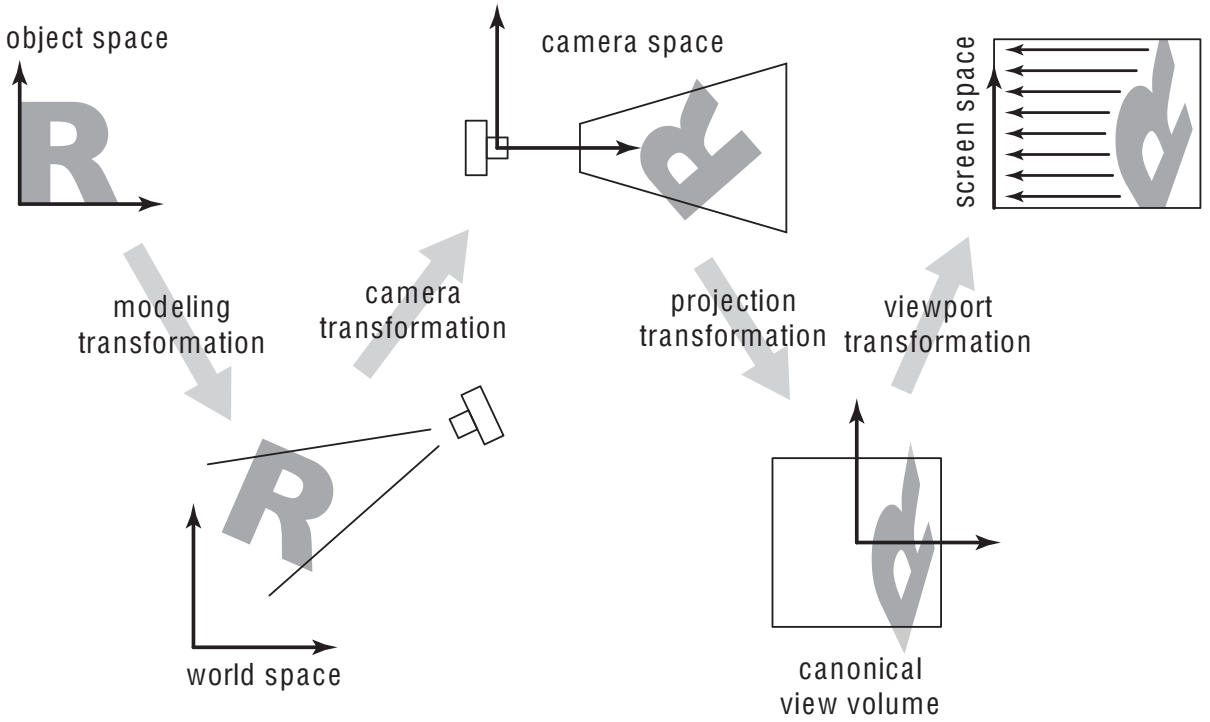
- Would like to just invert the ray generation process
- Problem 1: ray generation produces rays, not points in scene
- Inverting the ray tracing process requires division for the perspective case

Mathematics of projection

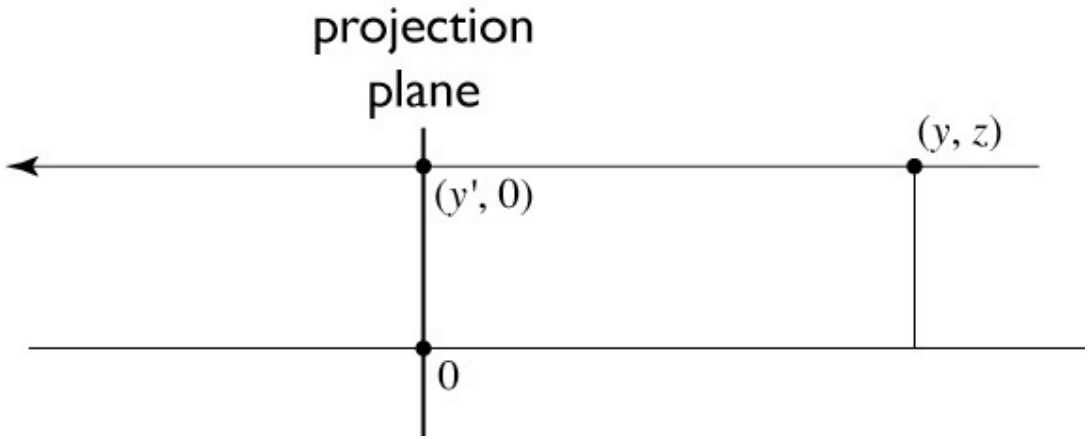
- Always work in eye coords
  - assume eye point at 0 and plane perpendicular to *z*
- Orthographic case
  - a simple projection: just toss out *z*
- Perspective case: scale diminishes with *z*
  - and increases with *d*

Pipeline of transformations

- Standard sequence of transforms



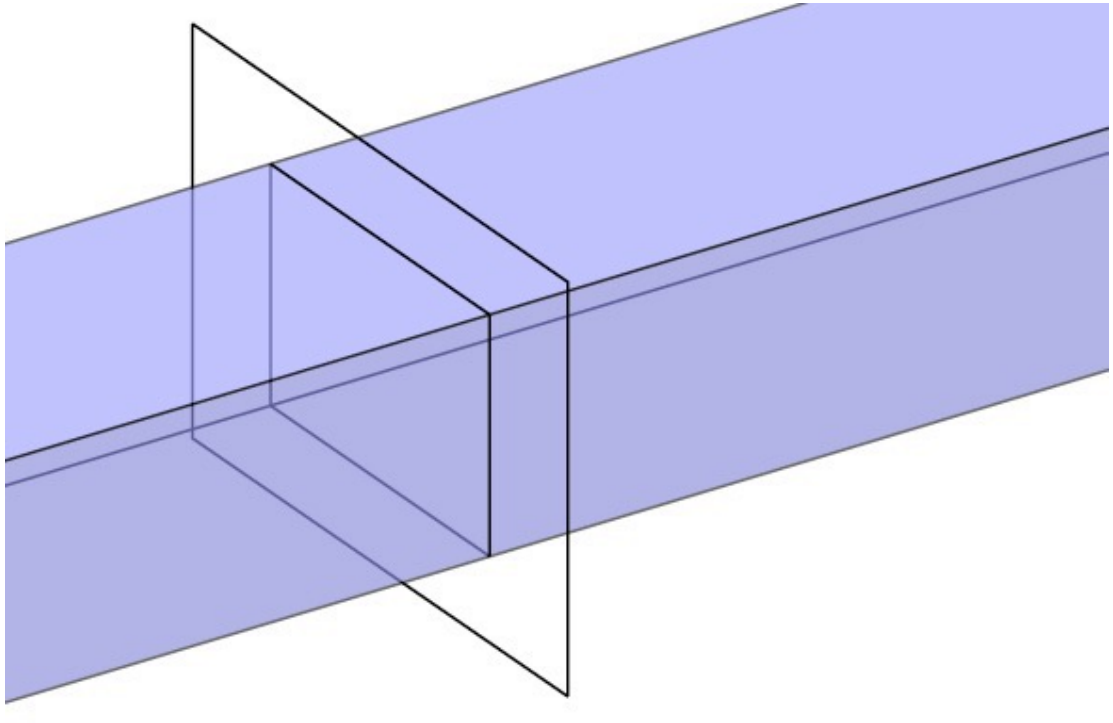
Parallel projection: orthographic



to implement orthographic, just toss out *z*:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

View volume: orthographic



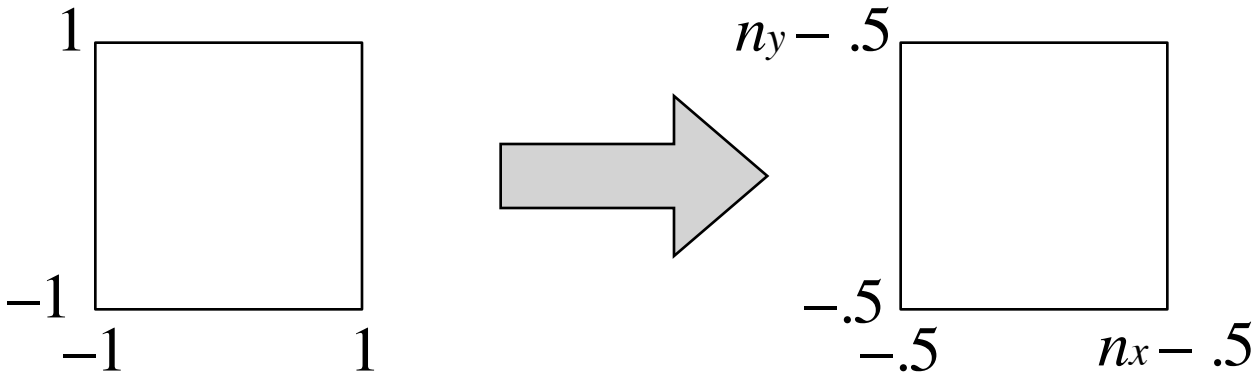
Viewing a cube of size 2

- Start by looking at a restricted case: the *canonical view volume*
- It is the cube $[0,1]^3$, viewed from the z direction
- Matrix to project it into a square image in $[0,1]^2$ is trivial:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

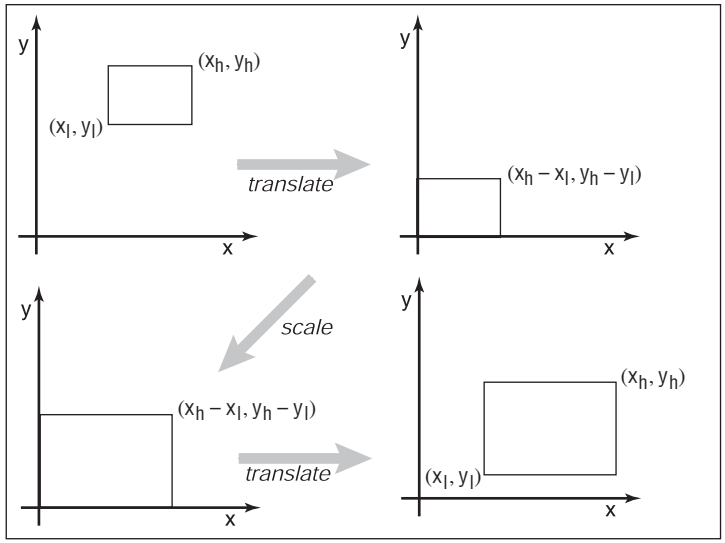
Viewing a cube of size 2

- To draw in image, need coordinates in pixel units, though
- Exactly the opposite of mapping (*i*,*j*) to (*u*,*v*) in ray generation



Windowing transforms

- This transformation is worth generalizing: take one axis-aligned rectangle or box to another
  - a useful, if mundane, piece of a transformation chain

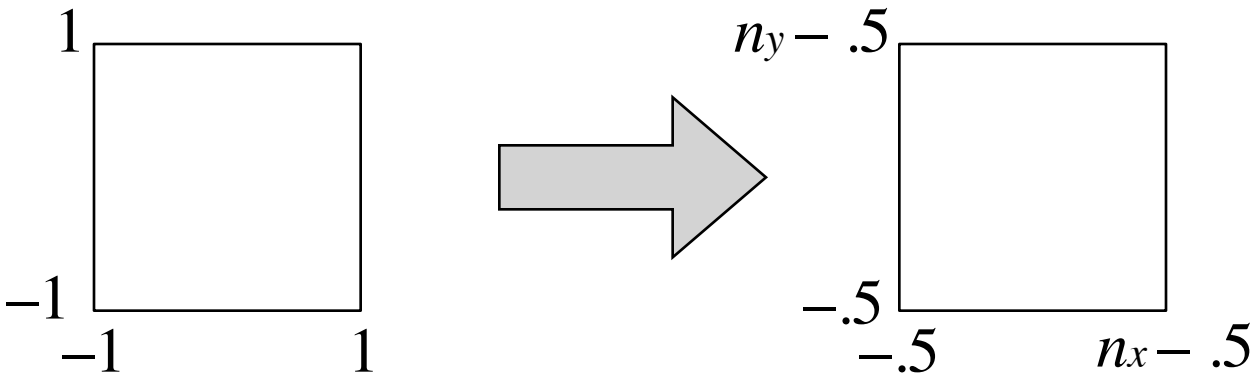


$$\text{window} = \text{translate}(x_l, y_l)\ \text{scale}\ \frac{x_h - x_l}{x_h - x_l}, \frac{y_h - y_l}{y_h - y_l}\ \text{translate}(-x_l, -y_l)$$

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

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

Viewport transformation



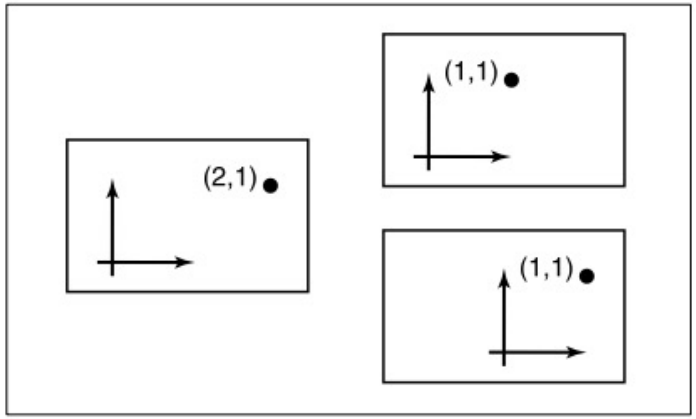
$$\begin{bmatrix} x_{\text{screen}} \\ y_{\text{screen}} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n_x}{2} & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & \frac{n_y-1}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{\text{canonical}} \\ y_{\text{canonical}} \\ 1 \end{bmatrix}$$

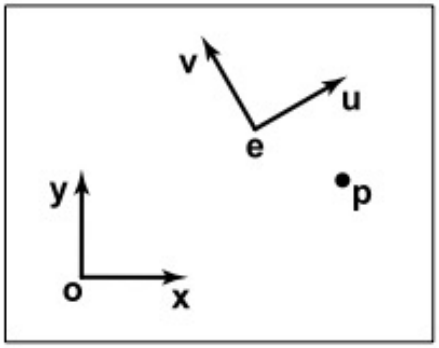
Viewport transformation

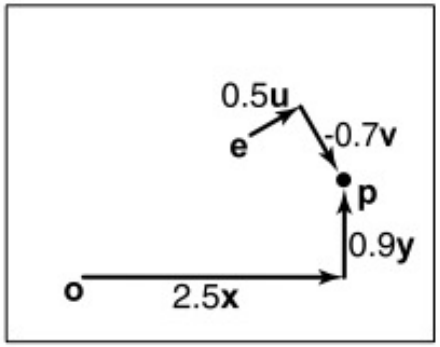
- In 3D, carry along z for the ride
  - one extra row and column

$$\mathbf{M}_{\text{vp}} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Coordinate Transformation







$$\mathbf{p} = (x_p, y_p) \equiv \mathbf{o} + x_p\mathbf{x} + y_p\mathbf{y}.$$

$$\mathbf{p} = (u_p, v_p) \equiv \mathbf{e} + u_p\mathbf{u} + v_p\mathbf{v}.$$

- Frame-to-Canonical Matrix

$$\mathbf{p}_{xy} = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{e} \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p}_{uv}.$$

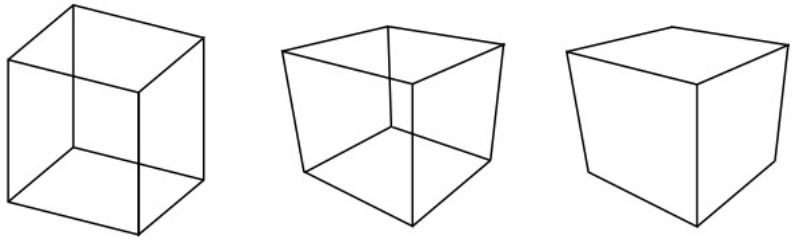
$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_e \\ 0 & 1 & y_e \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_u & x_v & 0 \\ y_u & y_v & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix} = \begin{bmatrix} x_u & x_v & x_e \\ y_u & y_v & y_e \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix}.$$

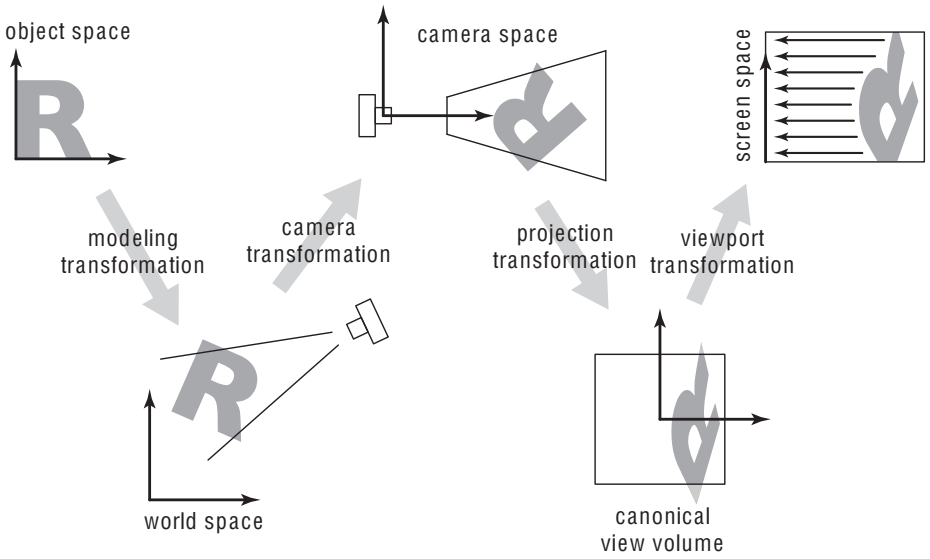
$$\mathbf{p}_{uv} = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{e} \\ 0 & 0 & 1 \end{bmatrix}^{-1} \mathbf{p}_{xy} \quad \mathbf{p}_{uv} = \begin{bmatrix} \mathbf{x}_{uv} & \mathbf{y}_{uv} & \mathbf{o}_{uv} \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p}_{xy}.$$

3D case?

View Transformation

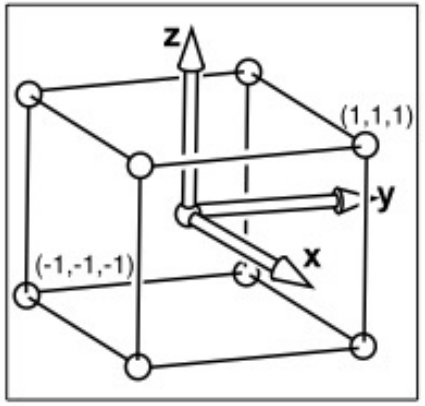
- 3D to 2D Mapping (3D points in the scene → 2D points in the image)
- Three step process
  - Camera/eye transformation
  - Project transformation
  - Viewpoint/Windowing transformation





Viewport Transformation

- $[-1,1]^2$ → [-0.5, Nx-0.5] x [-0.5, Ny-0.5]
- A specific case of Windowing transform

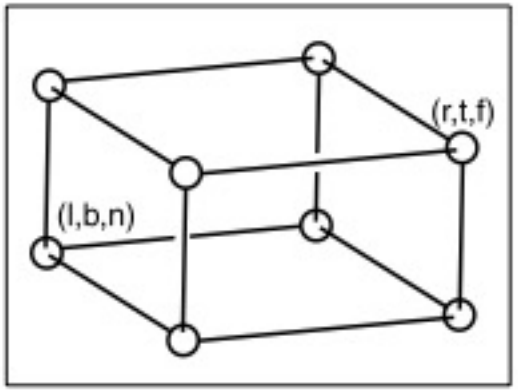


$$\begin{bmatrix} x_{\text{screen}} \\ y_{\text{screen}} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n_x}{2} & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & \frac{n_y-1}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{\text{canonical}} \\ y_{\text{canonical}} \\ 1 \end{bmatrix}$$

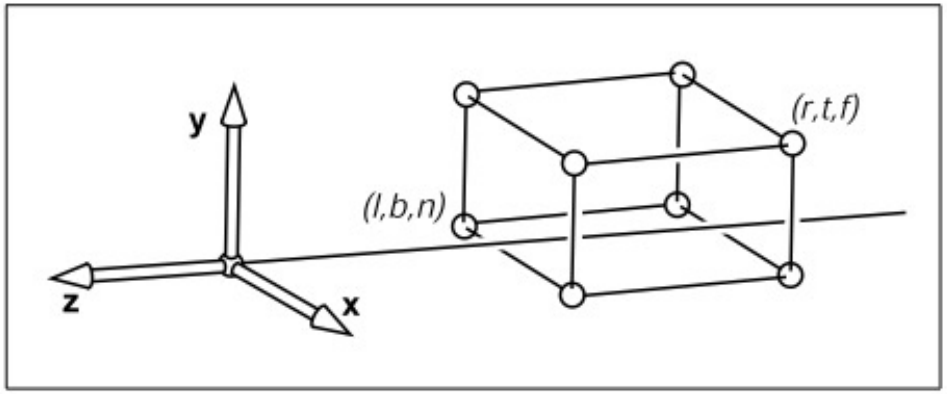
$$M_{\text{vp}} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Orthographic Projection

- Just another window (3D) transform



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



$$\begin{bmatrix} x_{\text{pixel}} \\ y_{\text{pixel}} \\ z_{\text{canonical}} \\ 1 \end{bmatrix} = (\mathbf{M}_{\text{vp}}\mathbf{M}_{\text{orth}}) \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Pseudo Code

construct $\mathbf{M}_{\text{vp}}$
construct $\mathbf{M}_{\text{orth}}$
$\mathbf{M} = \mathbf{M}_{\text{vp}}\mathbf{M}_{\text{orth}}$
for each line segment $(\mathbf{a}_i, \mathbf{b}_i)$ **do**
    $\mathbf{p} = \mathbf{M}\mathbf{a}_i$
    $\mathbf{q} = \mathbf{M}\mathbf{b}_i$
    drawline(x_p, y_p, x_q, y_q)

Orthographic transformation chain

- Start with coordinates in object's local coordinates
- Transform into world coords (modeling transform, M_m)
- Transform into eye coords (camera xf., $M_{cam} = F_c^{-1}$)
- Orthographic projection, M_{orth}
- Viewport transform, M_{vp}

$$\mathbf{p}_s = \mathbf{M}_{\text{vp}}\mathbf{M}_{\text{orth}}\mathbf{M}_{\text{cam}}\mathbf{M}_{\text{m}}\mathbf{p}_o$$

$$\begin{bmatrix} x_s \\ y_s \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} & \mathbf{e} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \mathbf{M}_{\text{m}} \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix}$$

Clipping planes

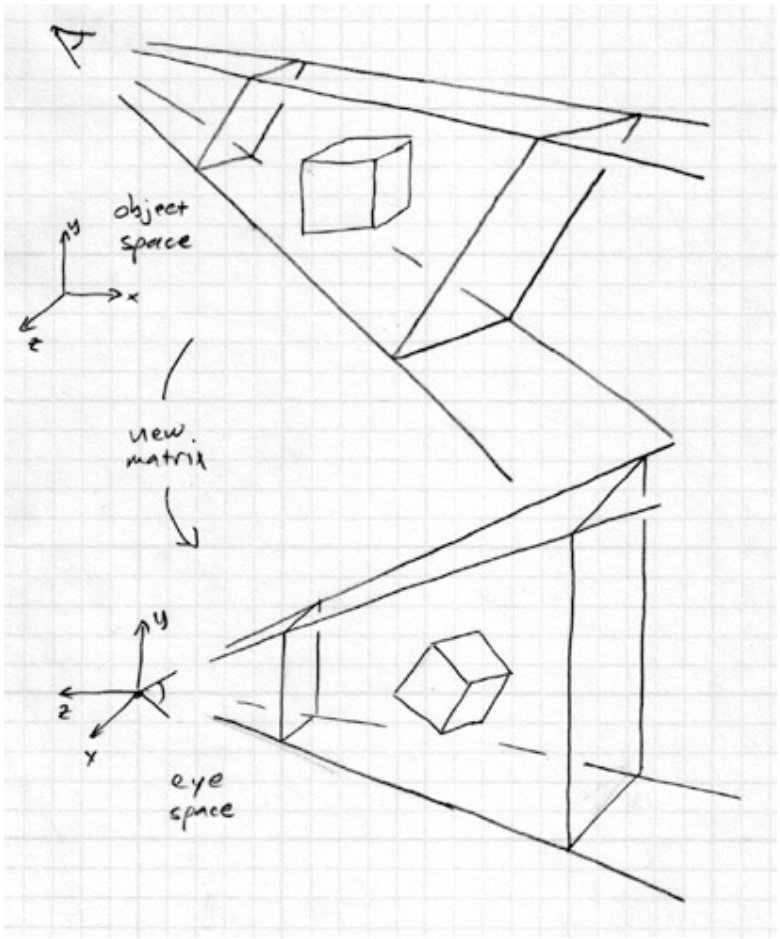
- In object-order systems we always use at least two *clipping planes* that further constrain the view volume
  - near plane: parallel to view plane; things between it and the viewpoint will not be rendered
  - far plane: also parallel; things behind it will not be rendered
- These planes are:
  - partly to remove unnecessary stuff (e.g. behind the camera)
  - but really to constrain the range of depths

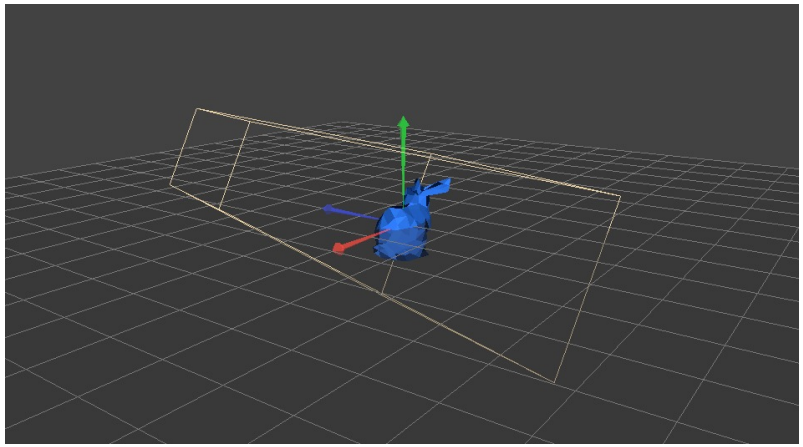
    (we'll see why later)

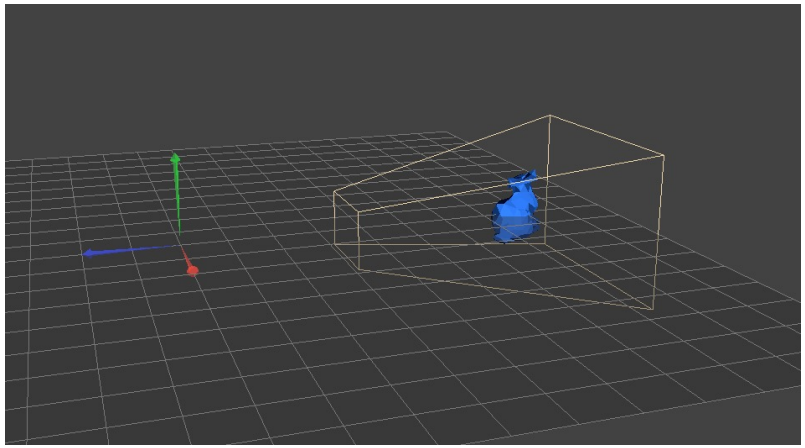
Camera and modeling matrices

- We worked out all the preceding transforms starting from eye coordinates
  - before we do any of this stuff we need to transform into that space
- Transform from world (canonical) to eye space is traditionally called the *viewing matrix*
  - it is the canonical-to-frame matrix for the camera frame
  - that is, F_c^{-1}
- Remember that geometry would originally have been in the object's local coordinates; transform into world coordinates is called the *modeling matrix*, M_m
- Note some systems (e.g. OpenGL) combine the two into a *modelview* matrix and just skip world coordinates

Viewing transformation



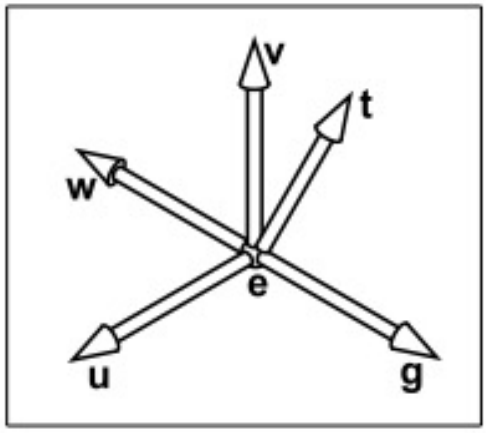




the camera matrix rewrites all coordinates in eye space

Camera Transformation

- Define camera coordinates
  - Eye position e
  - Gaze direction g
  - View-up vector t
- Construct an orthonormal coordinate system, uvw, for camera space

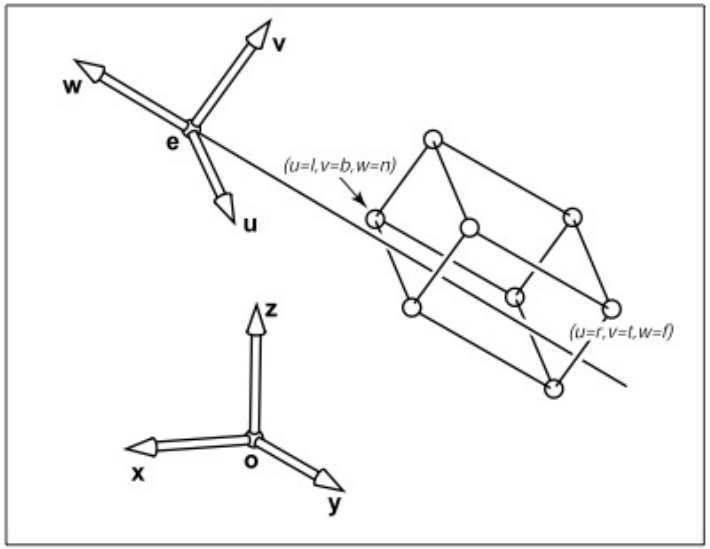


$$\mathbf{w} = -\frac{\mathbf{g}}{\|\mathbf{g}\|},$$

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

$$\mathbf{v} = \mathbf{w} \times \mathbf{u}.$$

Camera Transformation II



construct $\mathbf{M}_{\text{vp}}$
construct $\mathbf{M}_{\text{orth}}$
construct $\mathbf{M}_{\text{cam}}$
$\mathbf{M} = \mathbf{M}_{\text{vp}}\mathbf{M}_{\text{orth}}\mathbf{M}_{\text{cam}}$
for each line segment $(\mathbf{a}_i, \mathbf{b}_i)$ **do**
  $\mathbf{p} = \mathbf{M}\mathbf{a}_i$
  $\mathbf{q} = \mathbf{M}\mathbf{b}_i$
  drawline(x_p, y_p, x_q, y_q)

$$\mathbf{M}_{\text{cam}} = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} & \mathbf{e} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Projective Transformation

- What is projective transformation

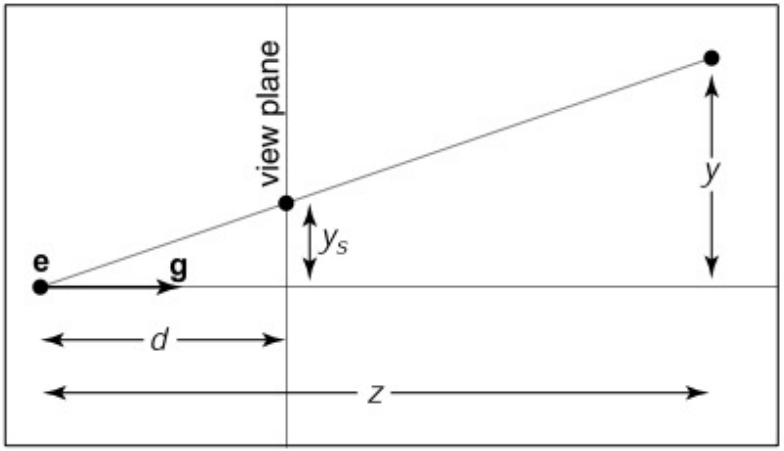
$$y_s = \frac{d}{z} y,$$

- Homogeneous coordinate (w !=1)

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

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

Projective Transformation



$$x' = \frac{a_1 x + b_1 y + c_1 z + d_1}{ex + fy + gz + h},$$

$$y' = \frac{a_2 x + b_2 y + c_2 z + d_2}{ex + fy + gz + h},$$

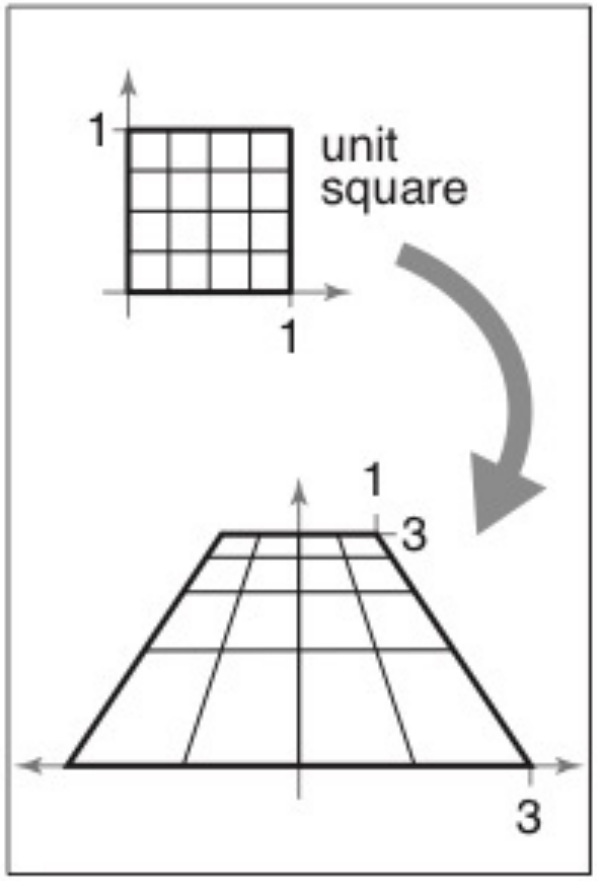
$$z' = \frac{a_3 x + b_3 y + c_3 z + d_3}{ex + fy + gz + h}.$$

Projective Transformation Example

$$\mathbf{M} = \begin{bmatrix} 2 & 0 & -1 \\ 0 & 3 & 0 \\ 0 & \frac{2}{3} & \frac{1}{3} \end{bmatrix} \qquad 3\mathbf{M} = \begin{bmatrix} 6 & 0 & -3 \\ 0 & 9 & 0 \\ 0 & 2 & 1 \end{bmatrix}$$

Transform [1,0] to [3, 0], [0,0] to [-3, 0], etc.

$$\begin{bmatrix} 2 & 0 & -1 \\ 0 & 3 & 0 \\ 0 & \frac{2}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \frac{1}{3} \end{bmatrix}$$



Perspective Projection

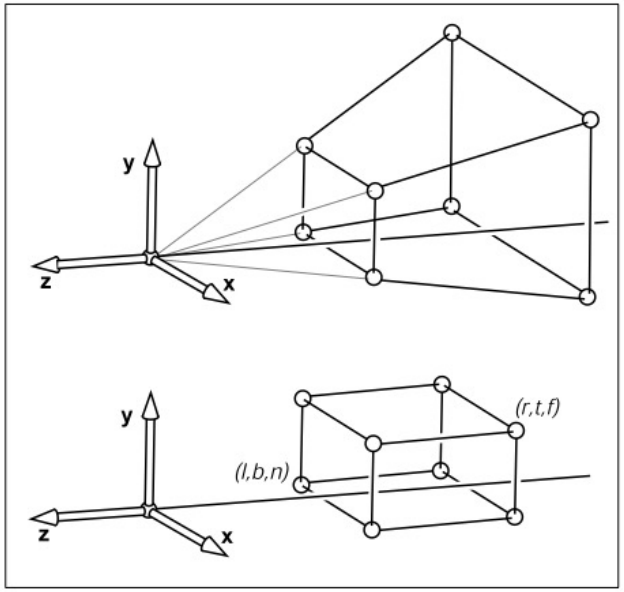
- Z coordinate has unavoidable distortion.

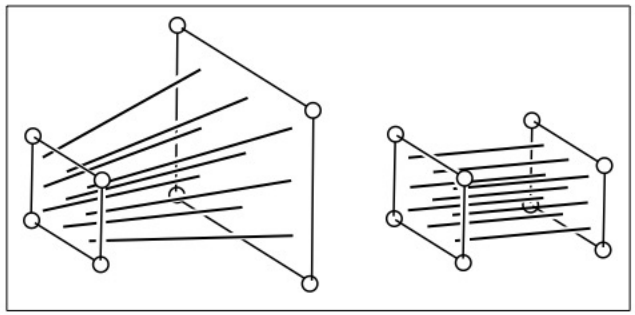
$$\begin{bmatrix} y_s \\ 1 \end{bmatrix} \sim \begin{bmatrix} d & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} y \\ z \\ 1 \end{bmatrix}.$$

[dy, z] = [dy/z, 1]

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

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





Perspective Projection II

- Z-order preserving (why?)
- Bring screen coordinate plus z back to original space
- Multiplying a scalar value for homogenous vector/matrix will not change meaning.
- Perspective transformation Matrix

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

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

$$\mathbf{M}_{\text{per}} = \mathbf{M}_{\text{orth}}\mathbf{P}.$$

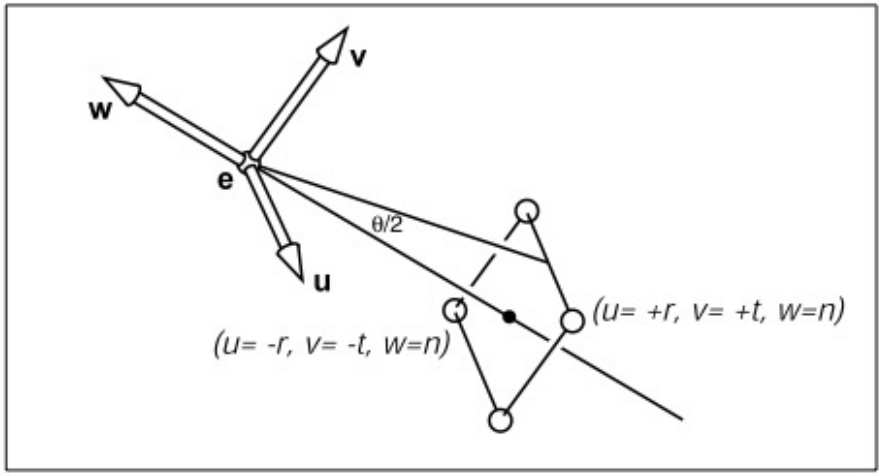
Perspective Projection Summary

compute $\mathbf{M}_{\text{vp}}$
compute $\mathbf{M}_{\text{per}}$
compute $\mathbf{M}_{\text{cam}}$
$\mathbf{M} = \mathbf{M}_{\text{vp}}\mathbf{M}_{\text{per}}\mathbf{M}_{\text{cam}}$
for each line segment $(\mathbf{a}_i, \mathbf{b}_i)$ **do**
  $\mathbf{p} = \mathbf{M}\mathbf{a}_i$
  $\mathbf{q} = \mathbf{M}\mathbf{b}_i$
  drawline$(x_p/w_p, y_p/w_p, x_q/w_q, y_q/w$

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

Properties of Perspective Projection

- Line to Line, Plane to Plane.
- Field of view
  - Instead of using (l, r, t, b) and n.
  - Look through the center of the window, l=-r, t=-b.



$$\frac{n_x}{n_y} = \frac{r}{t}.$$

$$\tan\frac{\theta}{2} = \frac{t}{|n|}.$$