CG – T7 - Projection

L:CC, MI:ERSI

Miguel Tavares Coimbra

(course and slides designed by Verónica Costa Orvalho)

U. PORTO FC

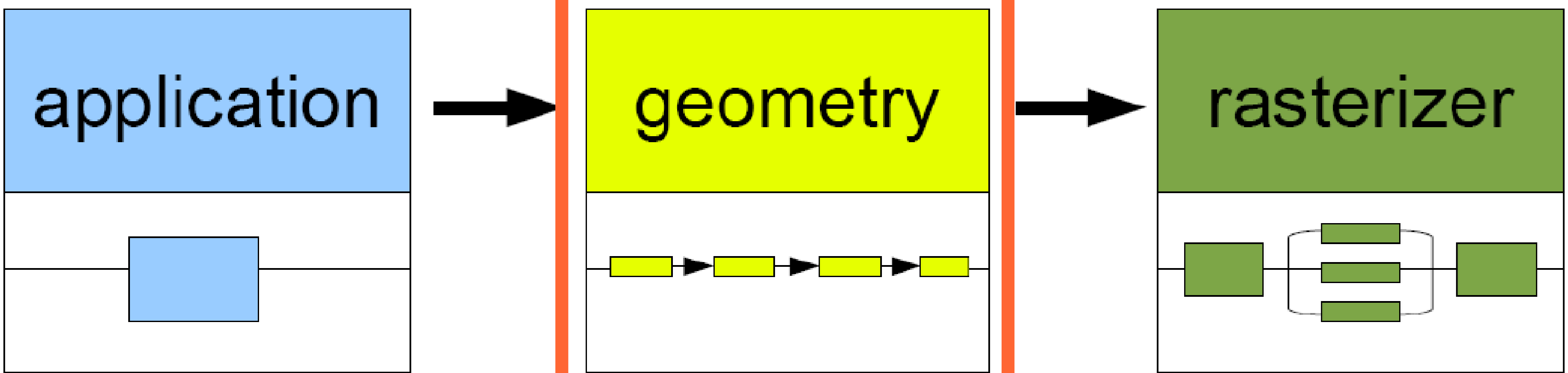
Basic steps for creating a 2D image out of a 3D world

- Create the 3D world
  - Vertexes and triangles in a 3D space
- Project it to a 2D ‘camera’
  - Use perspective to transform coordinates into a 2D space
- Paint each pixel of the 2D image
  - Rasterization, shading, texturing
  - Will break this into smaller things later on
- Enjoy the super cool image you have created

Today

How do we get 2D images out of a 3D world?

pipeline



. **collision** detection
. **animation** global acceleration
. **physics** simulation

process on CPU or GPU

. **transformation**
. **projection**

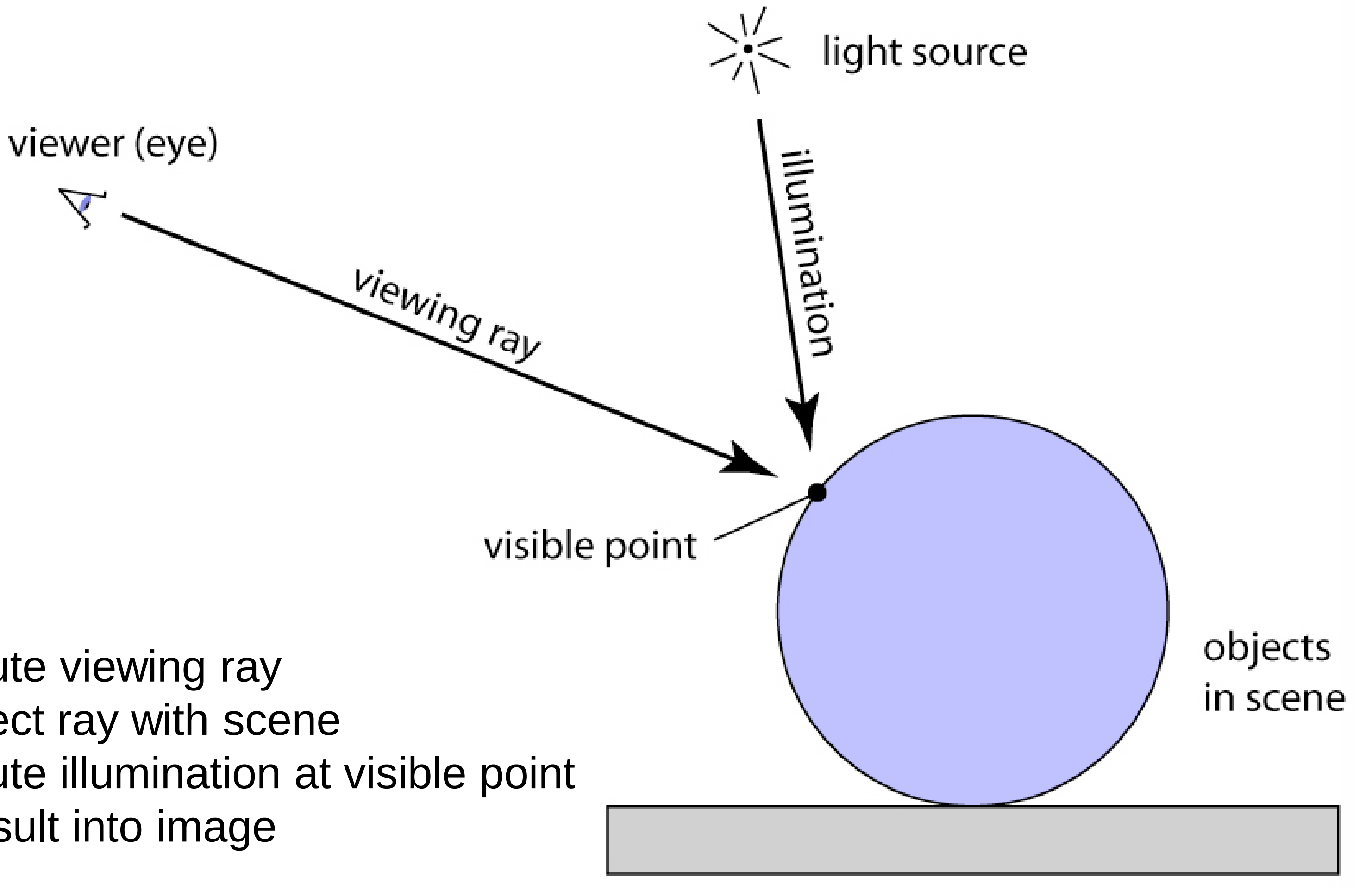
Computes:
. what is to be drawn
. how should be drawn
. where should be drawn

process on GPU

. **draws** images generated by **geometry stage**

process on GPU

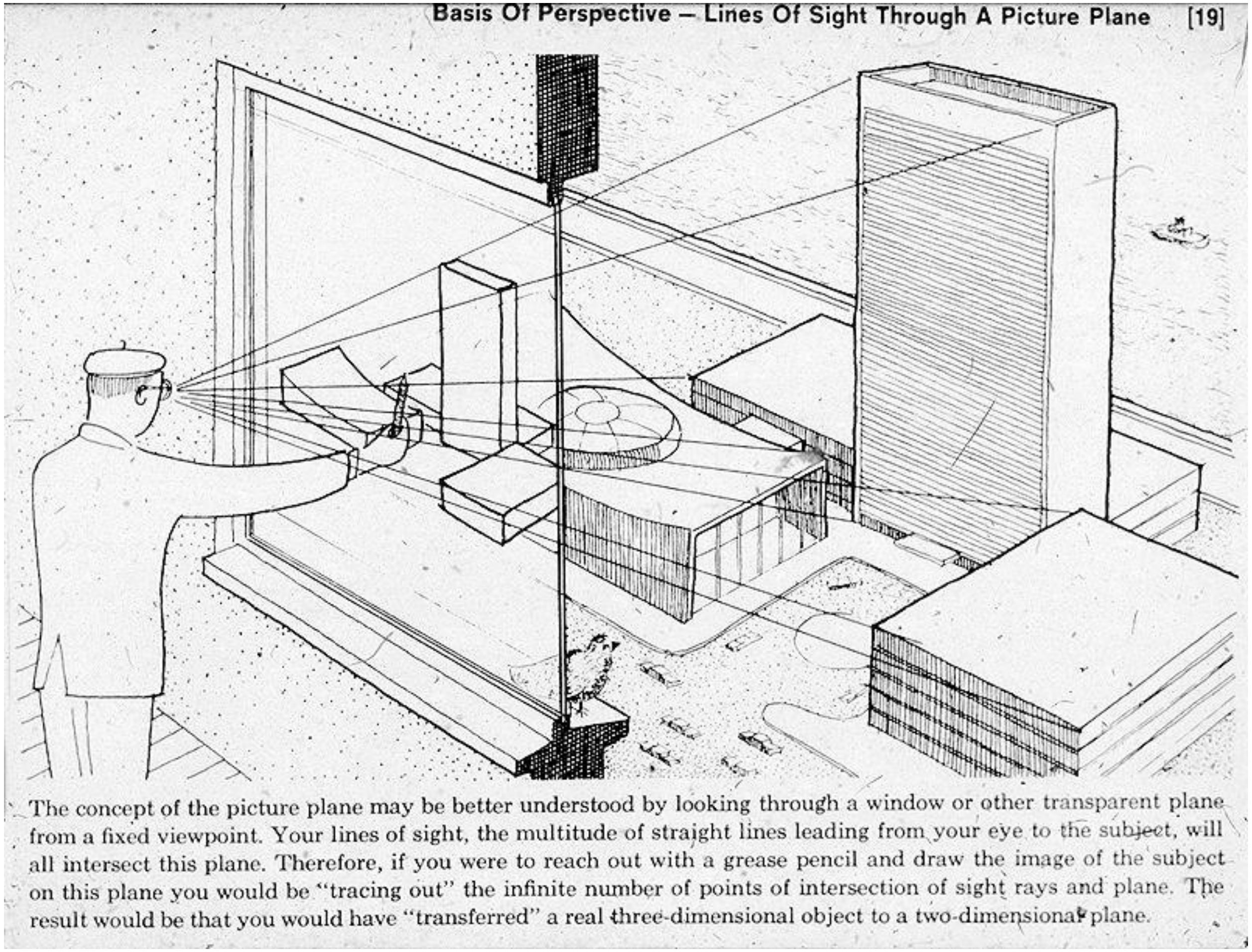
One possibility: Ray tracing



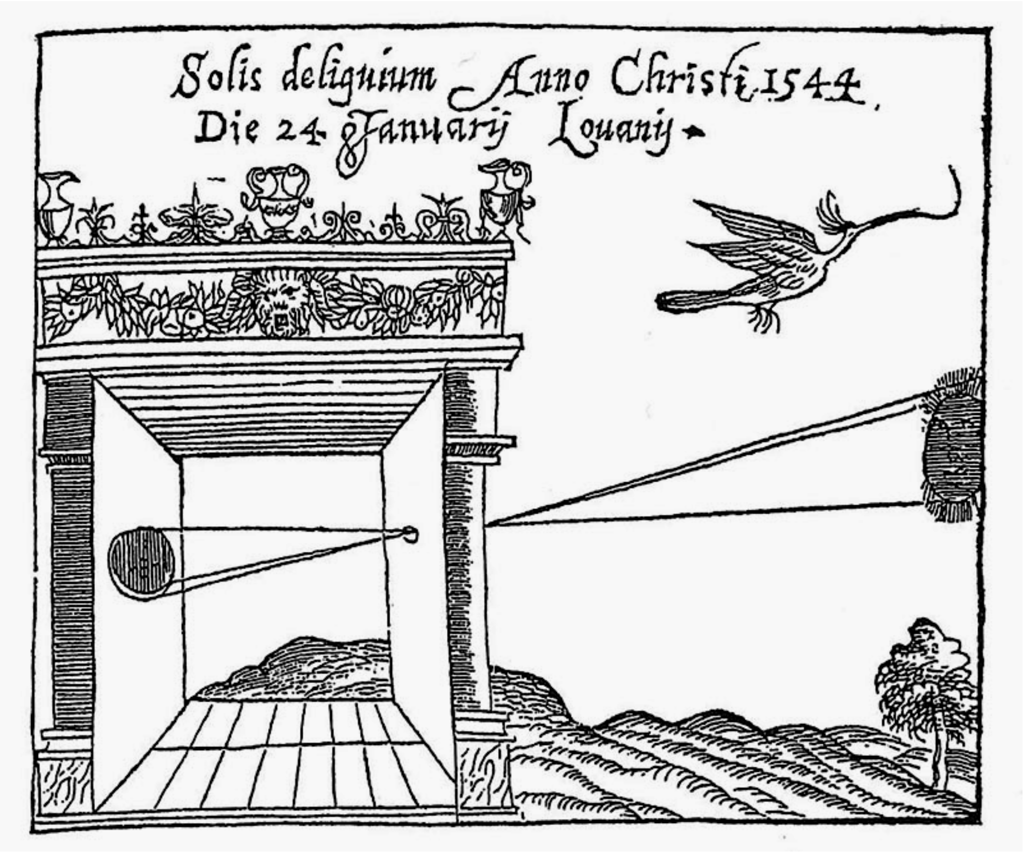
for each pixel {
compute viewing ray
intersect ray with scene
compute illumination at visible point
put result into image
}

Adapted from Steve Marschner, Cornell University

Another one: Projection

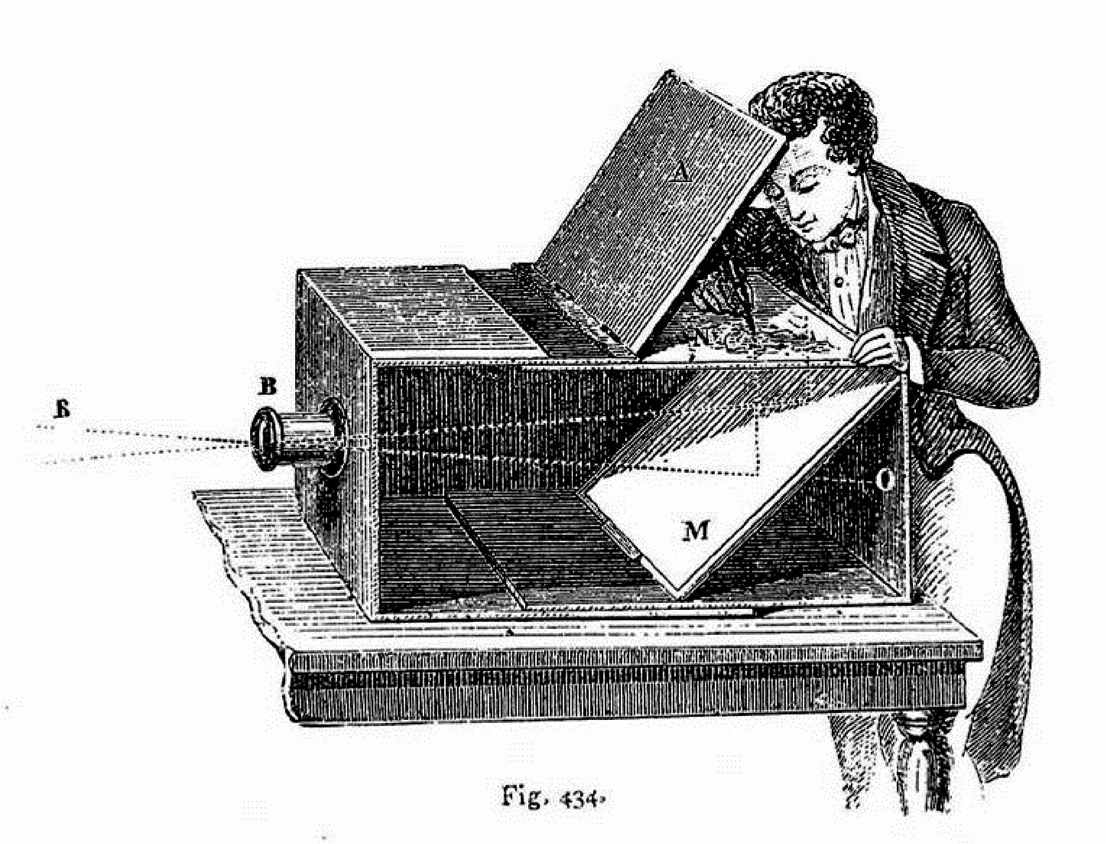


Projection in photography



Camera Obscura, Gemma Frisius, 1544

Lens based projection



Lens Based Camera Obscura, 1568

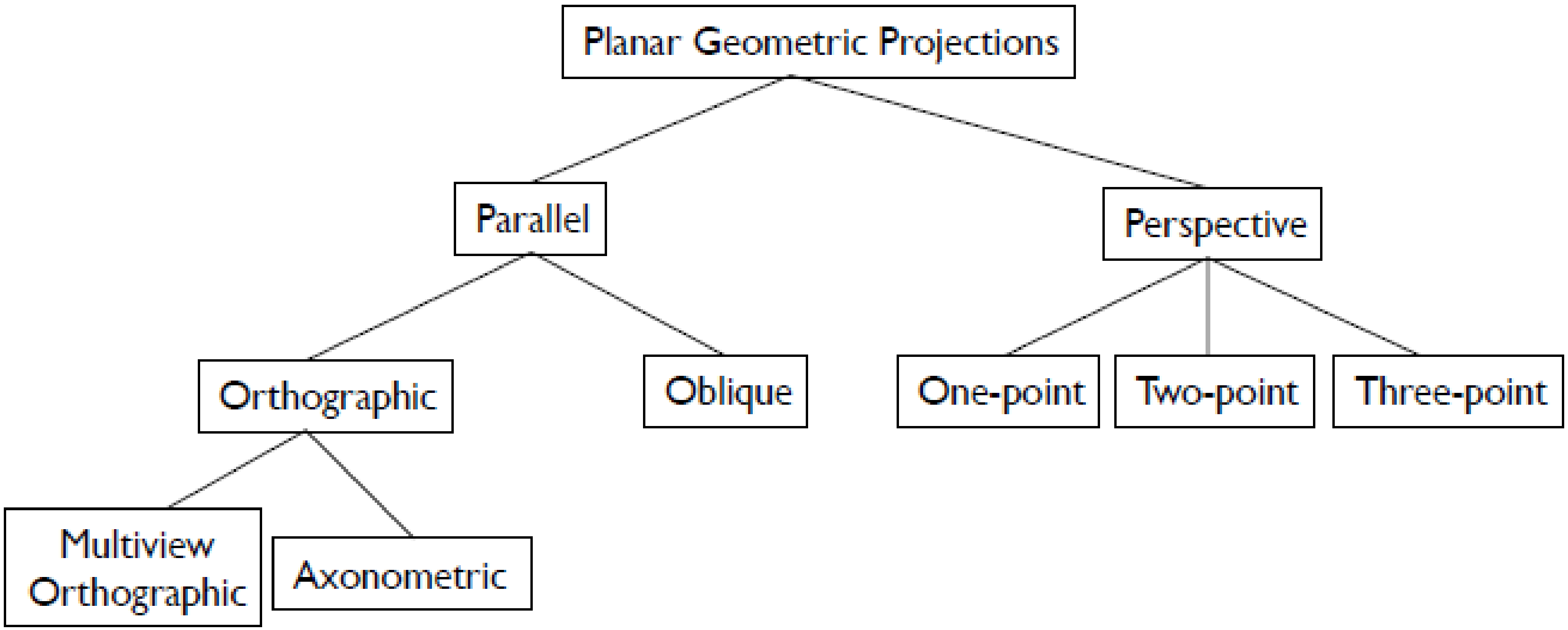
1544
1568

Ray tracing vs. Projection

- Viewing in ray tracing
  - start with image point
  - compute ray that projects to that point
  - do this using geometry
- Viewing by projection
  - start with 3D point
  - compute image point that it projects to
  - do this using transforms
- Inverse processes
  - ray gen. computes the preimage of projection

Are there different types of projections?

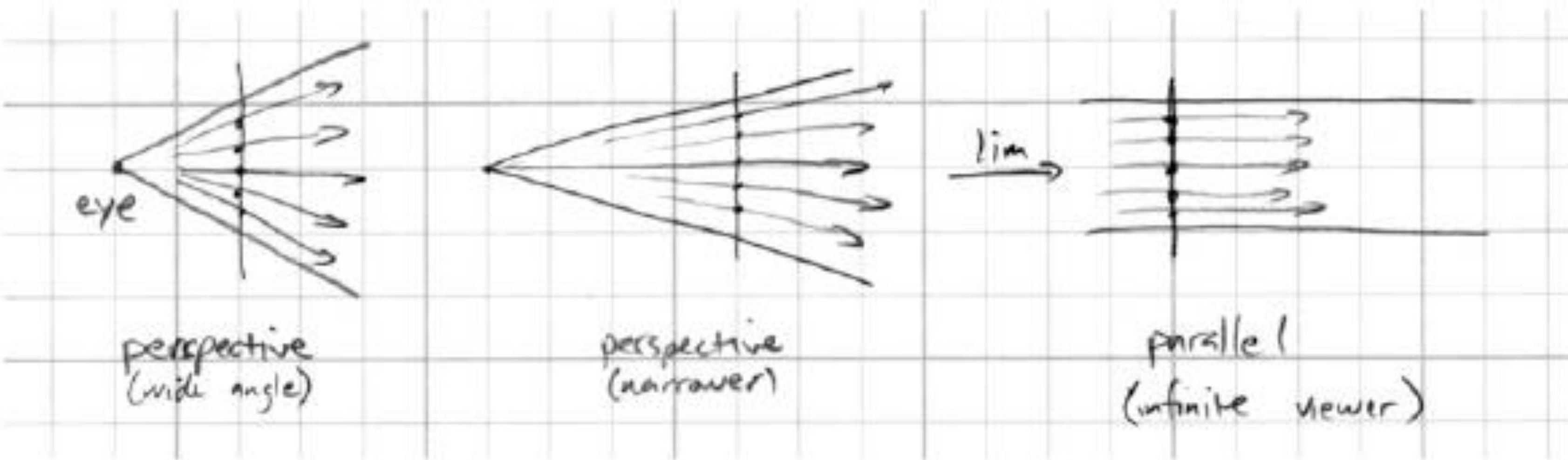
Classical projections



Adapted from Steve Marschner, Cornell University

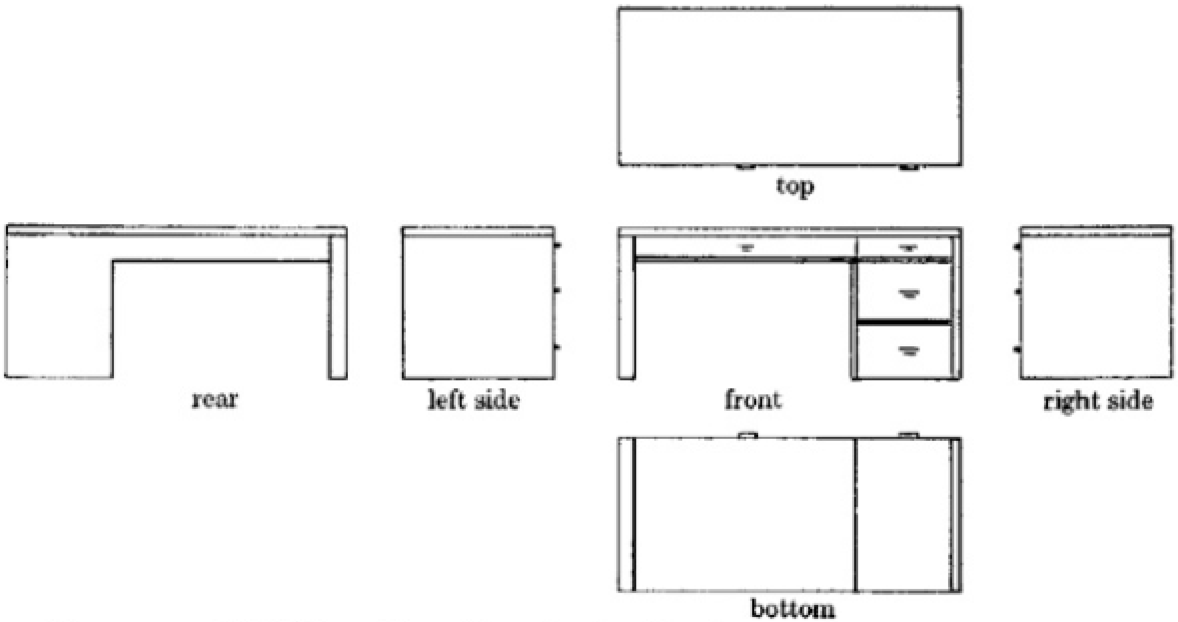
Parallel Projection

- Viewing rays are parallel rather than diverging
  - like a perspective camera that's far away



Adapted from Steve Marschner, Cornell University

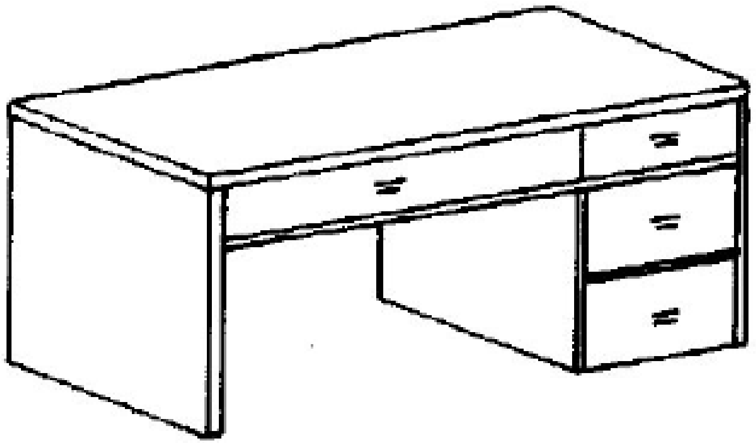
Multiview orthographic

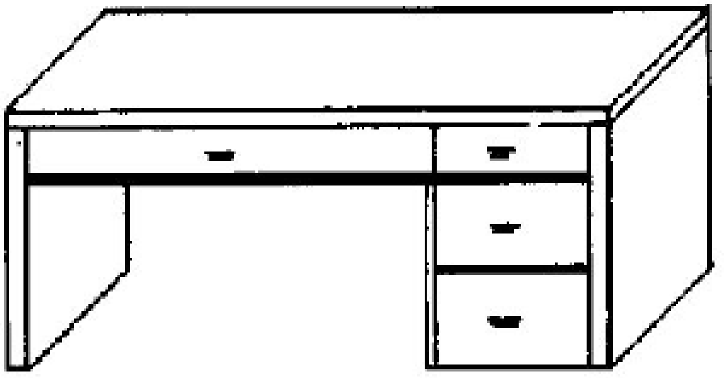


– projection plane parallel to a coordinate plane
– projection direction perpendicular to projection plane

Adapted from Steve Marschner, Cornell University

Off-axis parallel



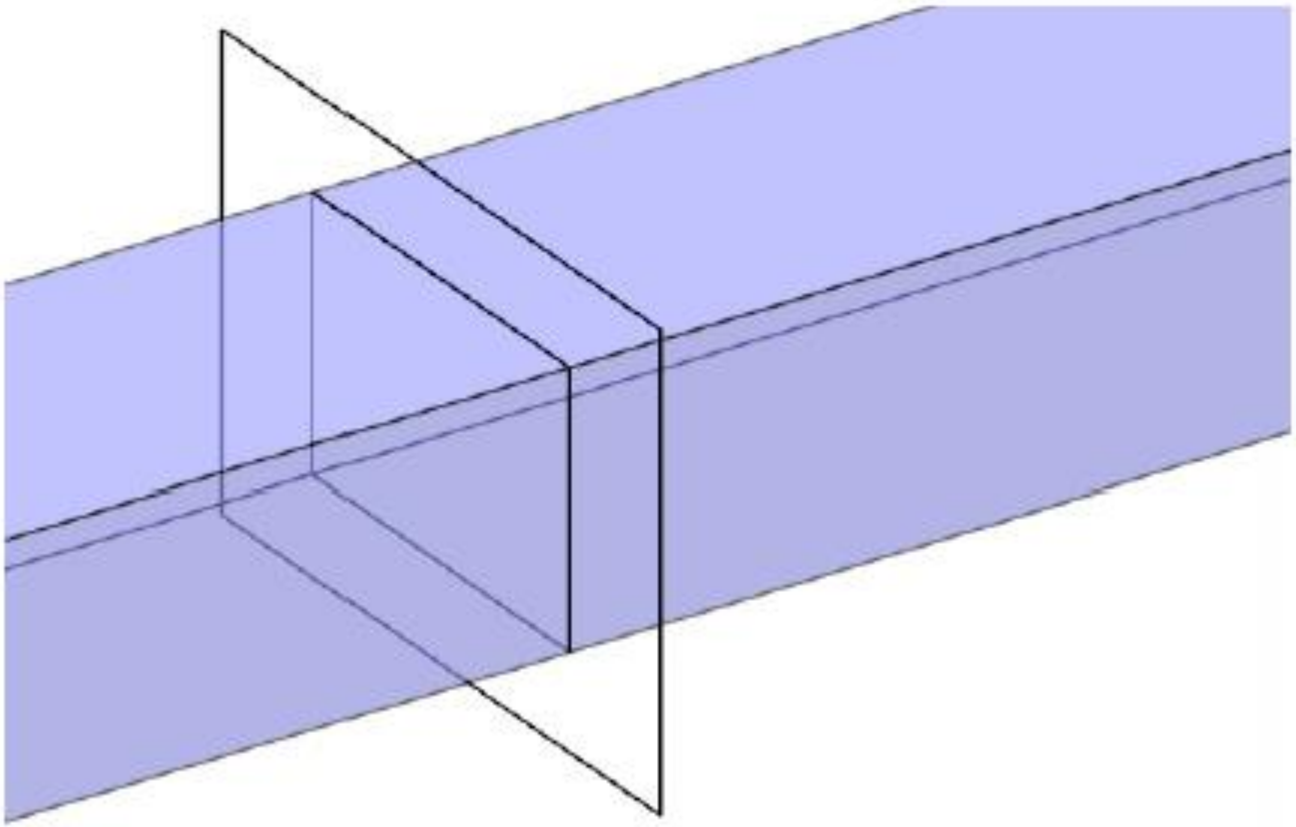


axonometric: projection plane perpendicular to projection direction but not parallel to coordinate planes

oblique: projection plane parallel to a coordinate plane but not perpendicular to projection direction.

Adapted from Steve Marschner, Cornell University

View volume: Orthographic



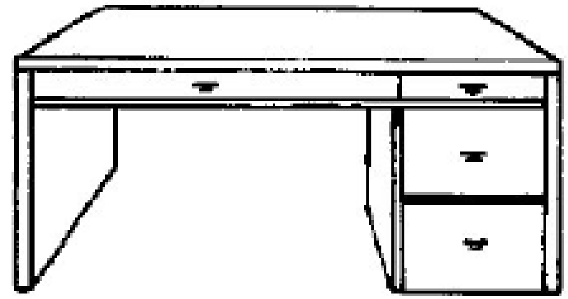
Adapted from Steve Marschner, Cornell University

Perspective

one-point: projection plane parallel to a coordinate plane (to two coordinate axes)

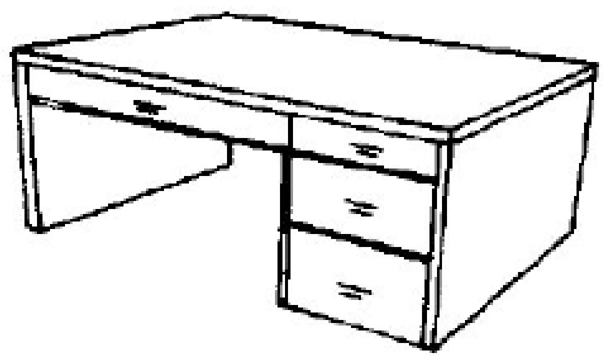
two-point: projection plane parallel to one coordinate axis

three-point: projection plane not parallel to a coordinate axis



one-point

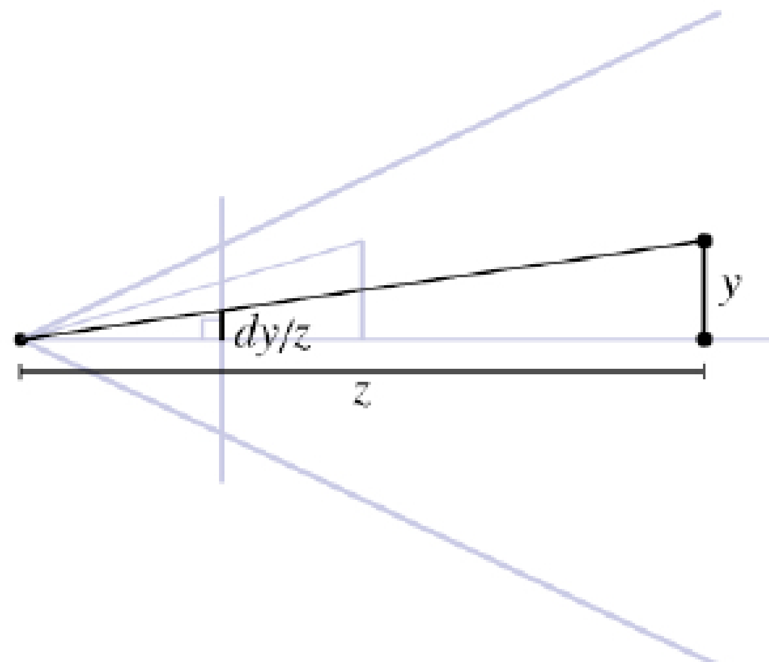
two-point



three-point

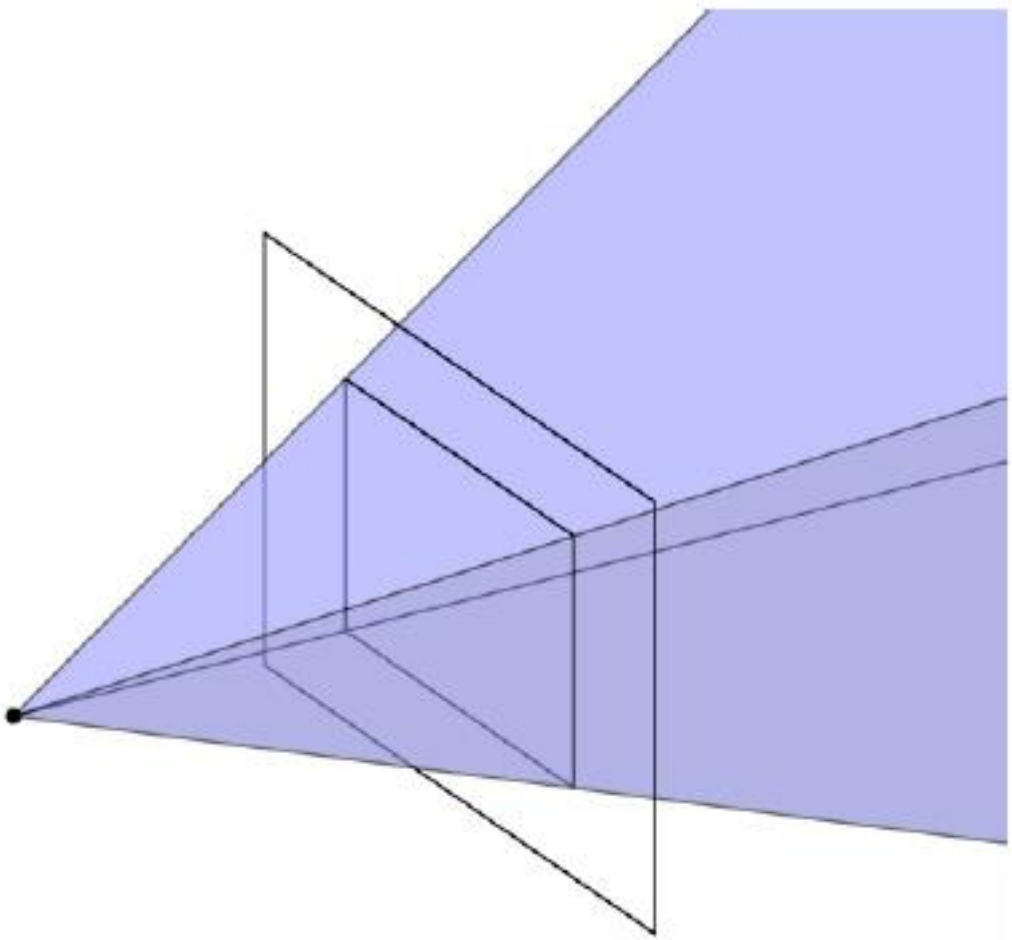
Perspective projection (normal)

- Perspective is projection by lines through a point; "normal" = plane perpendicular to view direction
  - magnification determined by:
    - image height
    - object depth
    - image plane distance
  - f.o.v. $\alpha = 2 \operatorname{atan}(h/(2d))$
  - $y' = d\, y / z$
  - "normal" case corresponds to common types of cameras



Adapted from Steve Marschner, Cornell University

View volume: Perspective



Adapted from Steve Marschner, Cornell University

Field of view

– Angle between the rays corresponding to opposite edges of a perspective image
– Determines ‘strength’ of perspective effects

Adapted from Steve Marschner, Cornell University

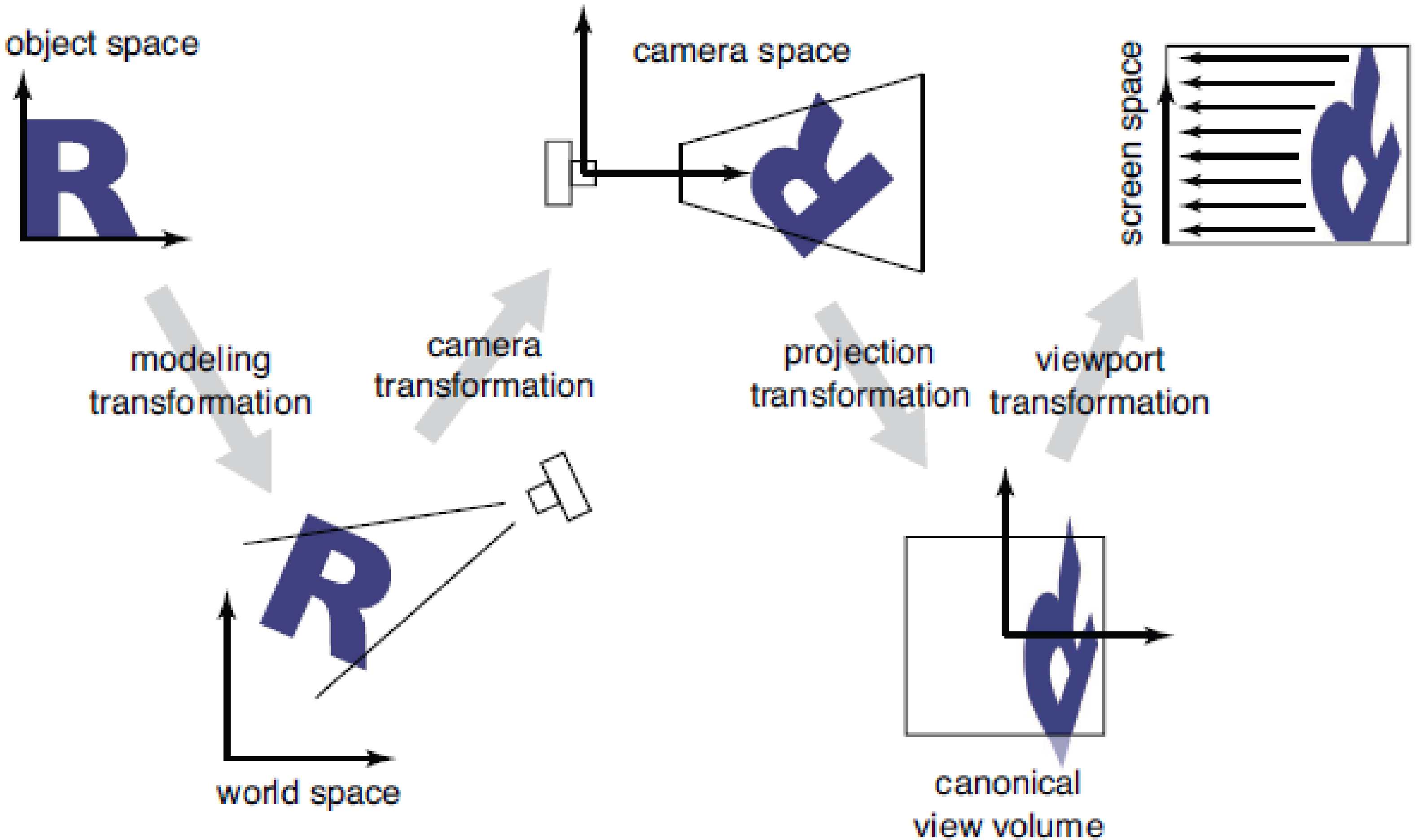
Adapted from Steve Marschner, Cornell University

camera tilted up: converging vertical lines

lens shifted up: parallel vertical lines

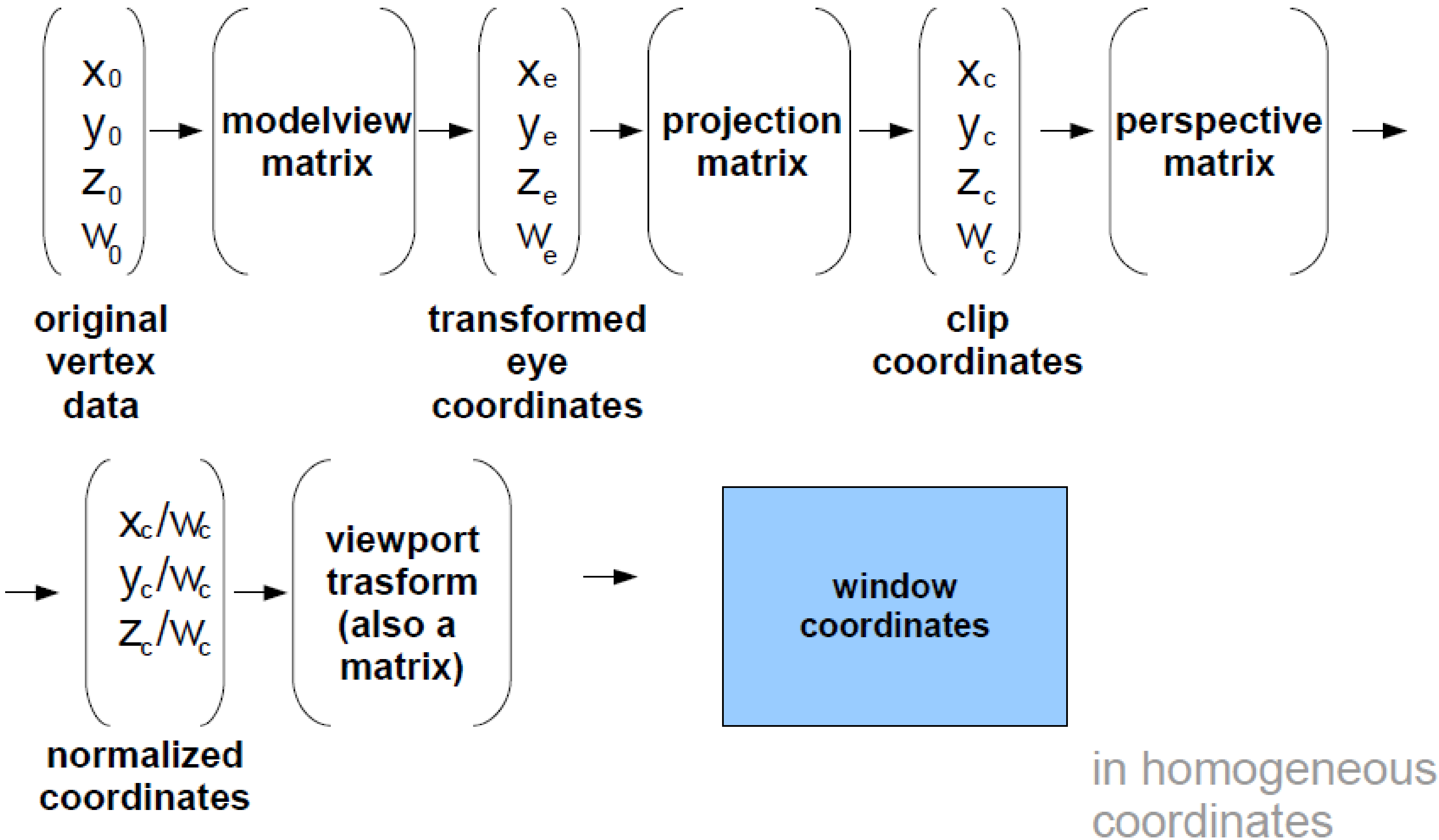
3D Viewing

Pipeline of transformations



Adapted from Steve Marschner, Cornell University

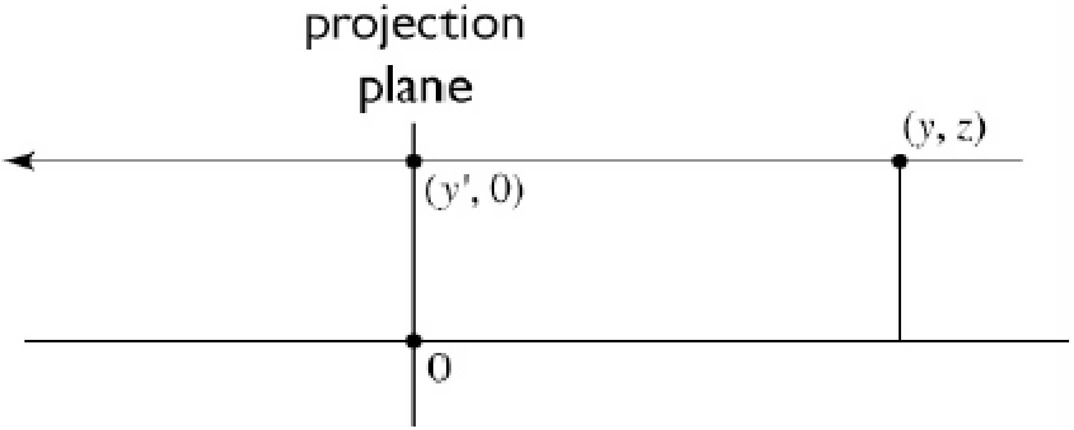
OpenGL transformations pipeline



Mathematics of projection

- Always works in eye coordinates
  - Assume eye point is at 0 and plane perpendicular to z
- Orthographic case
  - Simple projection: Just discard z
- Perspective case: scale diminishes with z
  - And increases with d

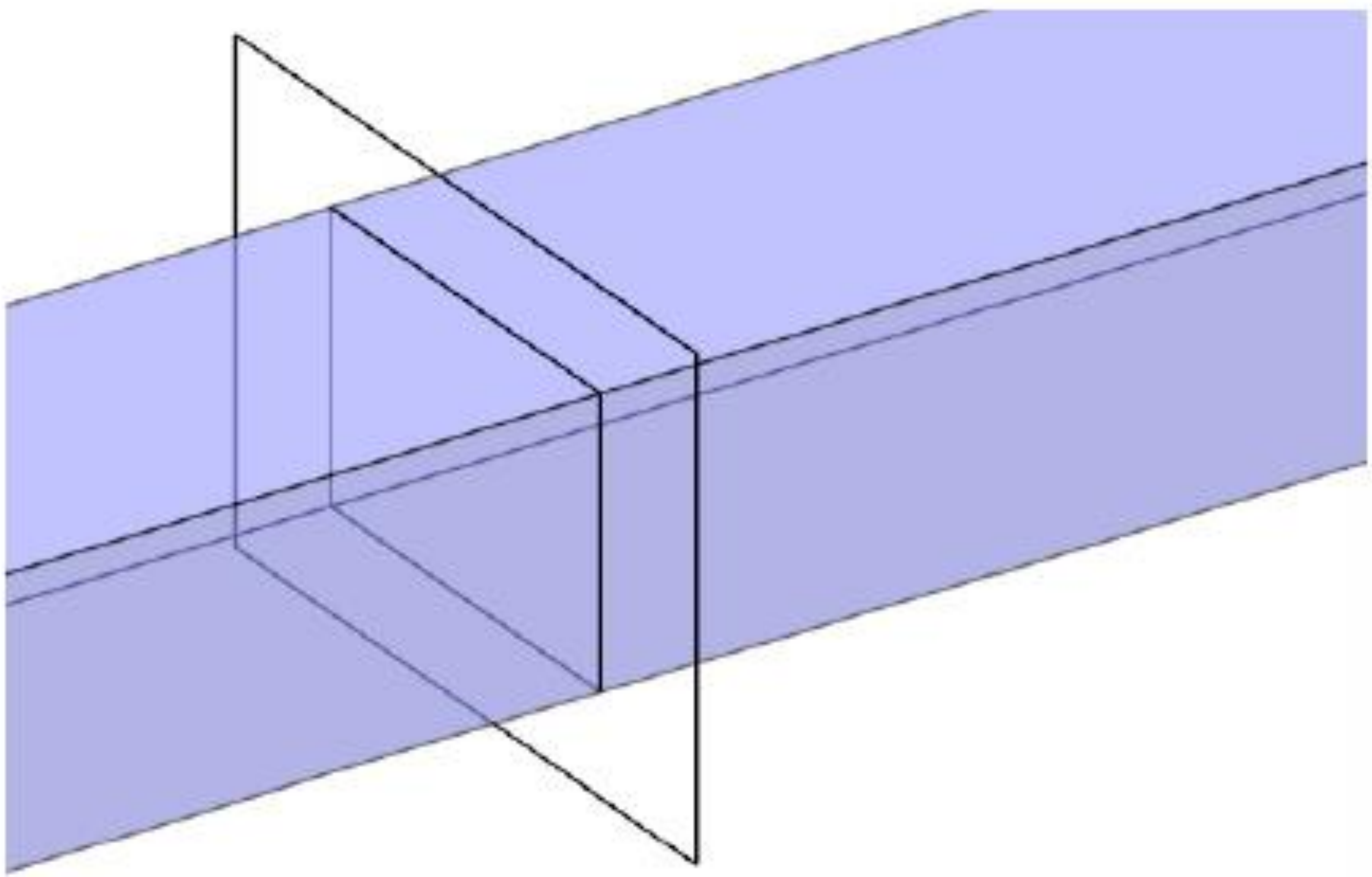
Orthographic projection



to implement orthographic, just toss out z:

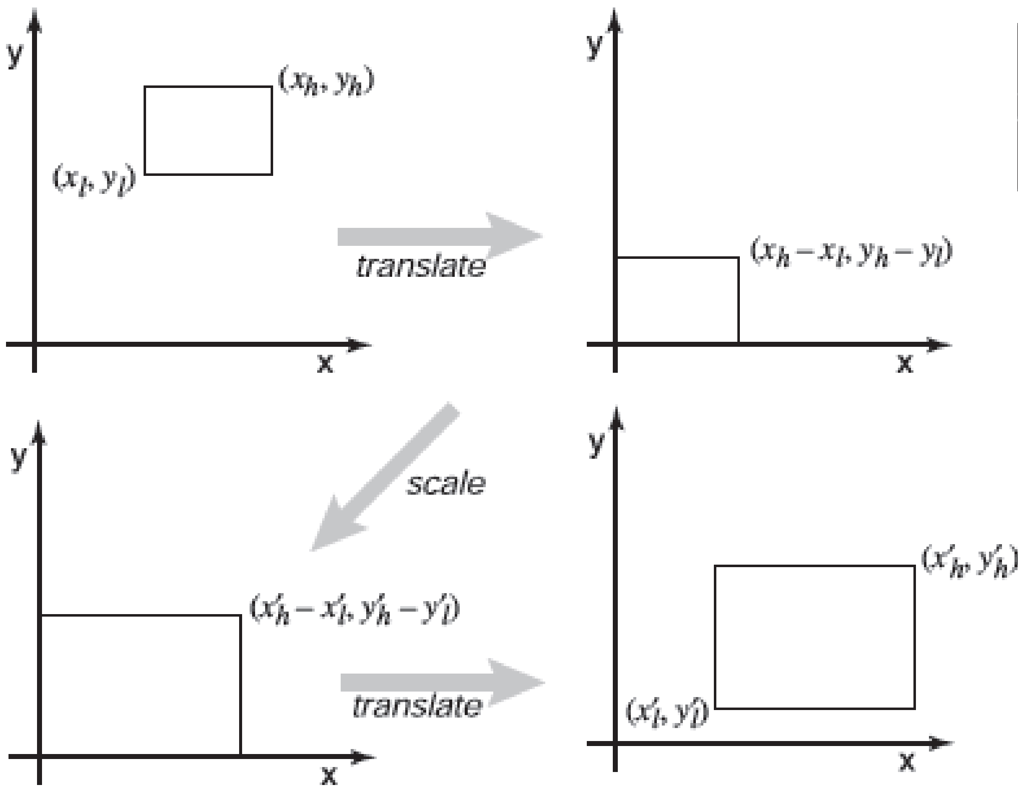
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

What about the view volume?



Adapted from Steve Marschner, Cornell University

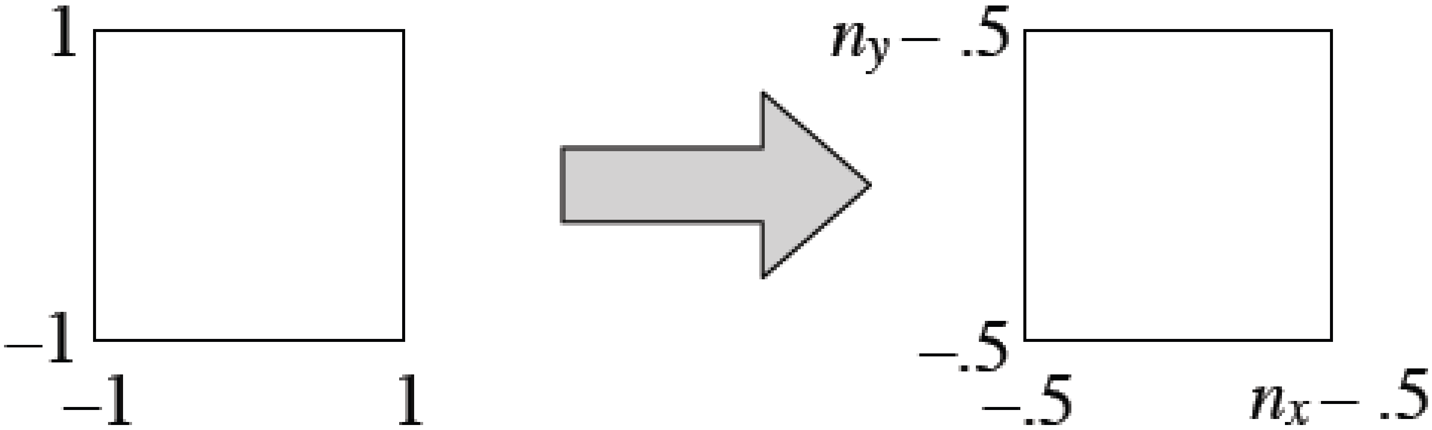
Windowing transforms



$$\begin{bmatrix} 1 & 0 & x'_l \\ 0 & 1 & y'_l \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{x'_h - x'_l}{x_h - x_l} & 0 & 0 \\ 0 & \frac{y'_h - y'_l}{y_h - y_l} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_l \\ 0 & 1 & -y_l \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \frac{x'_h - x'_l}{x_h - x_l} & 0 & \frac{x'_l x_h - x'_h x_l}{x_h - x_l} \\ 0 & \frac{y'_h - y'_l}{y_h - y_l} & \frac{y'_l y_h - y'_h y_l}{y_h - y_l} \\ 0 & 0 & 1 \end{bmatrix}$$

Viewport transformation



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

Adapted from Steve Marschner, Cornell University

Viewport transformation

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

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

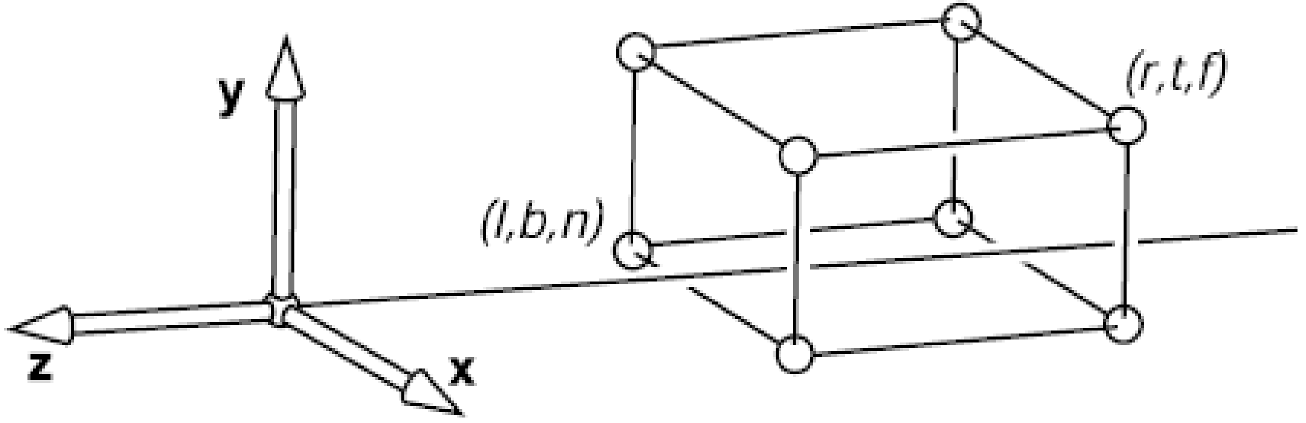
Adapted from Steve Marschner, Cornell University

What about the z direction?

- Two *clipping planes* 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

Orthographic projection

- First generalization: different view rectangle
  - retain the minus-z view direction



  - specify view by left, right, top, bottom (as in RT)
  - also near, far

Adapted from Steve Marschner, Cornell University

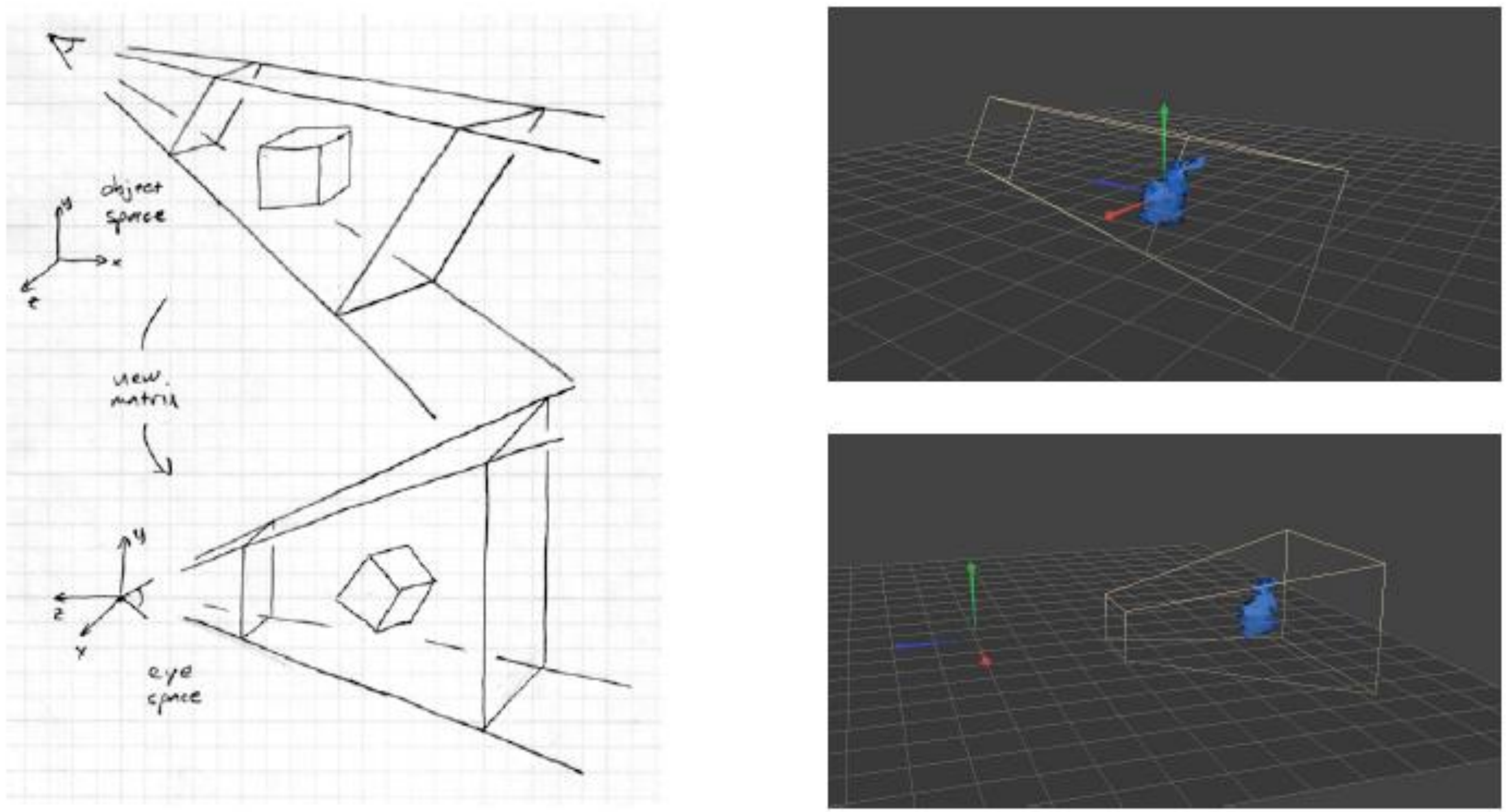
Orthographic projection matrix

- We can implement this by mapping the view volume to the canonical view volume.
- This is just a 3D windowing transformation!

$$\begin{bmatrix} \frac{x'_h - x'_l}{x_h - x_l} & 0 & 0 & \frac{x'_l x_h - x'_h x_l}{x_h - x_l} \\ 0 & \frac{y'_h - y'_l}{y_h - y_l} & 0 & \frac{y'_l y_h - y'_h y_l}{y_h - y_l} \\ 0 & 0 & \frac{z'_h - z'_l}{z_h - z_l} & \frac{z'_l z_h - z'_h z_l}{z_h - z_l} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

Viewing transformation



the camera matrix rewrites all coordinates in eye space

Advanced topic: Refer to the text book if interested!

Adapted from Steve Marschner, Cornell University

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., $M_{\text{cam}} = F_c^{-1}$)
- Orthographic projection, M_{orth}
- Viewport transform, M_{vp}

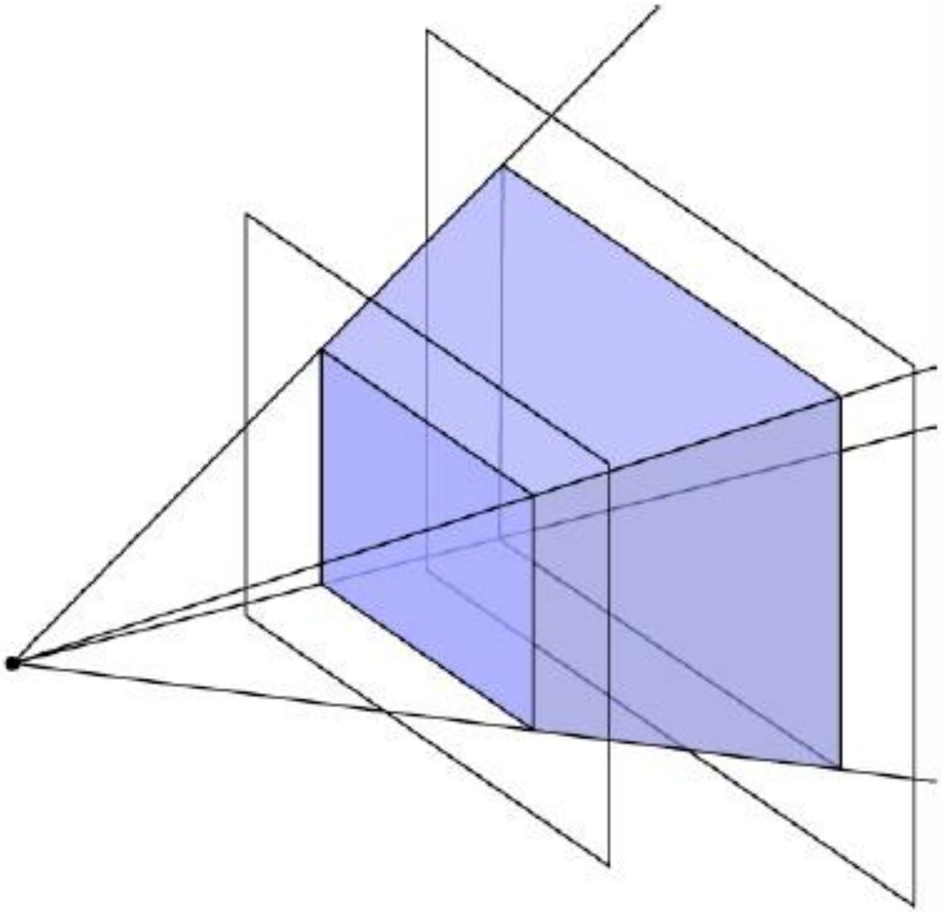
$$\mathbf{p}_s = \mathbf{M}_{\text{vp}}\mathbf{M}_{\text{orth}}\mathbf{M}_{\text{cam}}\mathbf{M}_{\text{m}}\mathbf{p}_o$$

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

Adapted from Steve Marschner, Cornell University

Perspective projection

View volume: Perspective (clipped)



Adapted from Steve Marschner, Cornell University

Perspective projection matrix

- Product of perspective matrix with orth. projection matrix

$$\mathbf{M}_{\mathrm{per}} = \mathbf{M}_{\mathrm{orth}}\mathbf{P}$$

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

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

Advanced topic: Refer to the text book if interested!

Adapted from Steve Marschner, Cornell University

Perspective transformation chain

- Transform into world coords (modeling transform, M_m)
- Transform into eye coords (camera xf., $M_{\text{cam}} = F_c^{-1}$)
- Perspective matrix, P
- Orthographic projection, M_{orth}
- Viewport transform, M_{vp}

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

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

Summary

- Different types of projection
  - Orthographic
  - Perspective
- Integrate nicely into the transformation chain
- Other elements:
  - Viewing transform
  - Viewport transform