Computer Graphics - Week 3

Bengt-Olaf Schneider
IBM T.J. Watson Research Center

Questions about Last Week ?

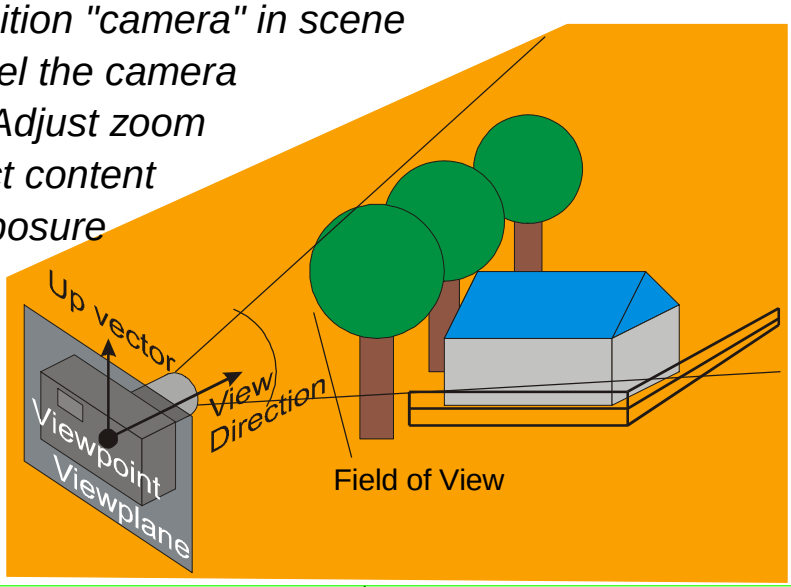
Overview of Week 3

- Viewing transformations
- Projections
- Camera models

Viewing Transformation

Viewing Compared to Taking Pictures

- **View Vector & Viewplane:** *Direct camera towards subject*
- **Viewpoint:** *Position "camera" in scene*
- **Up Vector:** *Level the camera*
- **Field of View:** *Adjust zoom*
- **Clipping:** *Select content*
- **Projection:** *Exposure*

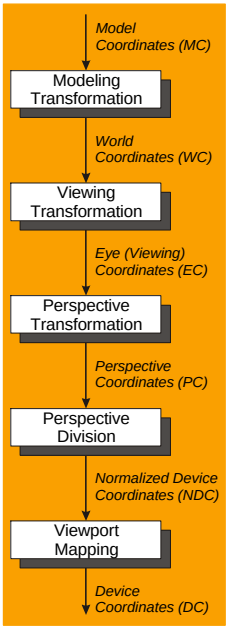


Viewing Transformation

- **Position and orient camera**
  - Set viewpoint, viewing direction and upvector
- **Use geometric transformation to position**
  - camera with respect to the scene or ...
  - scene with respect to the camera.
  - Both approaches are mathematically equivalent.

Viewing Pipeline and Coordinate Systems

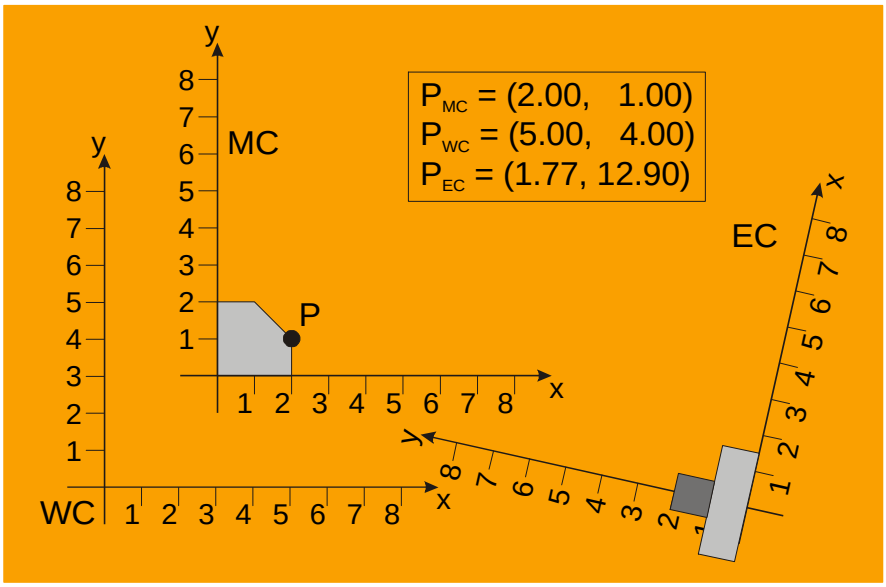
- **Modeling Transformation**
  - Position objects in world coordinates
- **Viewing Transformation**
  - Position objects in eye/camera coordinates
    - EC a.k.a. View Reference Coordinates
- **Perspective Transformation**
  - Convert view volume to a canonical view volume
  - Also used for clipping
    - PC a.k.a. clip coordinates
- **Perspective Division**
  - Perform mapping from 3D to 2D
- **Viewport Mapping**
  - Map normalized device coordinates into window/screen coordinates



Why a special eye coordinate system ?

- **In prinicipal it is possible to directly project on an arbitray viewplane. However, this is computationally very involved.**
- **Instead, the scene is transformed into special coordinate system, that makes projection easy.**
  - View down the negative z-axis
  - Projection plane parallel to z=0
- **This "extra" transformation is free !**
  - It can be combined with the modeling transformation
  - Therefore, every point must be transformed only once.
  - This transformation is known in OpenGL as the Model-View Matrix.

Model, World and Eye Coordinates



Computing the Viewing Transformation

- **Changing from WC to EC is simple change of coordinate systems which can be expressed using a 4x4 matrix:**

$$\mathbf{P}_{EC} = \mathbf{M}_{WE} \bullet \mathbf{P}_{WC}$$

$$\mathbf{M}_{WE} = \begin{pmatrix} \mathbf{Wx}_{EC} & \mathbf{Wy}_{EC} & \mathbf{Wz}_{EC} & \mathbf{Wo}_{EC} \end{pmatrix}$$

  - ***Wi*$_{EC}$** is unit vector on the WC ***i***-axis expressed in EC
  - ***Wo*$_{EC}$** is the WC origin expressed in EC
  - ***M*$_{ME}$** is the Model-View Matrix.

- **Same approach for changing from MC to WC**

Computing the Viewing Transformation: Example

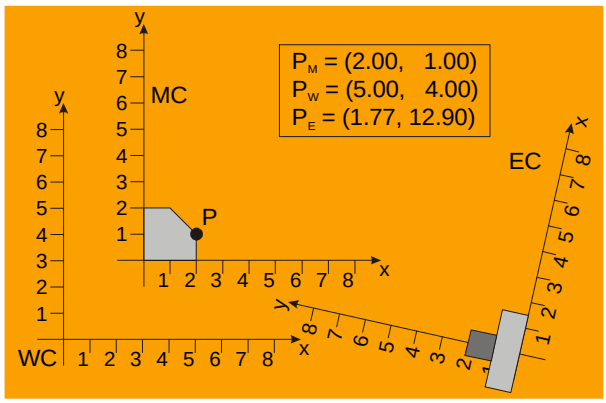
$$(1 \quad 0)^T_{WC} = (0.21 \quad -0.98)^T_{EC}$$
$$(0 \quad 1)^T_{WC} = (0.98 \quad 0.21)^T_{EC}$$
$$(0 \quad 0)^T_{WC} = (-3.25 \quad 17.00)^T_{EC}$$

$$\mathbf{M}_{WE} = \begin{pmatrix} 0.21 & 0.98 & -3.25 \\ -0.98 & 0.21 & 17.00 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{M}_{MW} = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{M}_{ME} = \mathbf{M}_{WE} \bullet \mathbf{M}_{MW} = \begin{pmatrix} 0.21 & 0.98 & 0.32 \\ -0.98 & 0.21 & 14.69 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{P}_E = \mathbf{M}_{ME} \bullet \mathbf{P}_M = \begin{pmatrix} 0.21 & 0.98 & 0.32 \\ -0.98 & 0.21 & 14.69 \\ 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} 2.0 \\ 1.0 \\ 1.0 \end{pmatrix} = \begin{pmatrix} 1.72 \\ 12.94 \\ 1 \end{pmatrix}$$



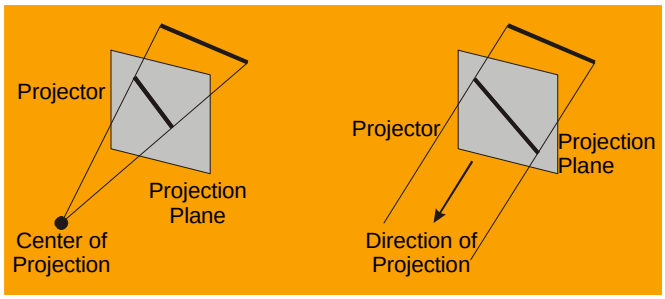
Projection

Projection: Overview

- **Now, the scene is transformed to the EC system**
- **Next, the scene must be projected onto the view plane**
- **This is done in several steps**
  - Selecting the projection type
  - Selecting center/direction of projection
  - Computing the view volume
  - Transforming the view volume into a *canonical* representation
  - Projection onto the viewplane

Projection: Introduction

- **Projections map points from *n* dimensions to *m* dimensions, with *m* < *n*.**
  - Here: from 3D to 2D
- **Points are projected along straight lines, called projectors, onto the projection plane**
  - Perspective projection: projectors go through a single point
  - Parallel projection: projectors are parallel

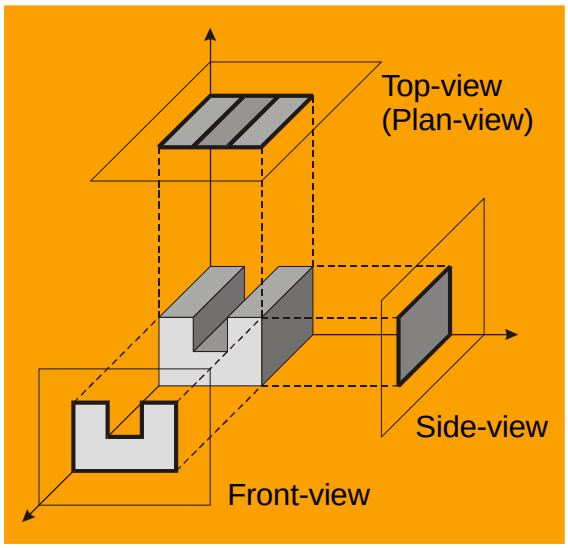


Parallel Projection: Properties

- **Objects appear at the same size irrespective of distance**
- **Lines map into lines**
- **Maintains distances but not angles**
  - Angles only maintained for lines parallel to the viewplane
  - However, parallel lines remain parallel

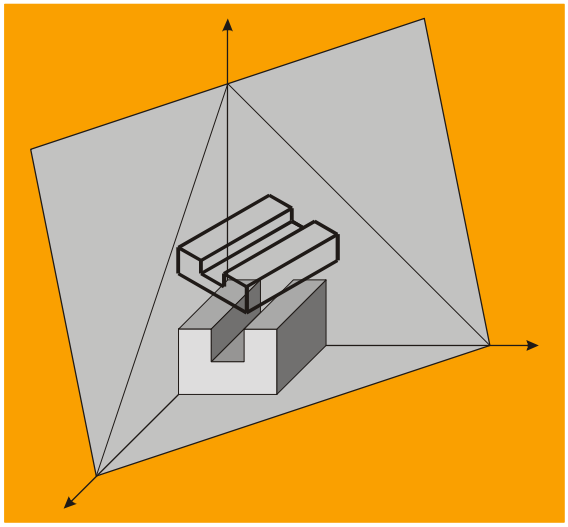
Parallel Projection: Orthographic

- **Projectors are perpendicular to the viewplane**
- **Frequently used orthographic projections have a viewplane perpendicular to a principal axis**
  - front elevation
  - top elevation (plan view)
  - side elevation



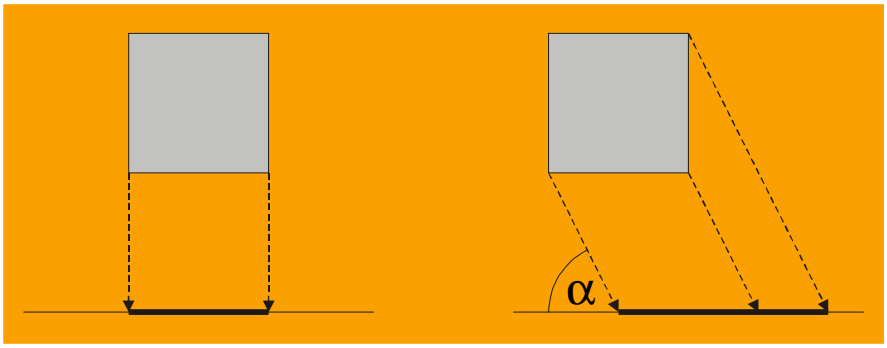
Parallel Projection: Orthographic

- **Axonometric orthographic projections have a viewplane that is not perpendicular to a principal axis**
  - Show more than one side
  - Uniform foreshortening, i.e. unlike perspective projection does not depend on depth of a point
  - Isometric projection is an axonometric projection where viewplane has same angle with all axes



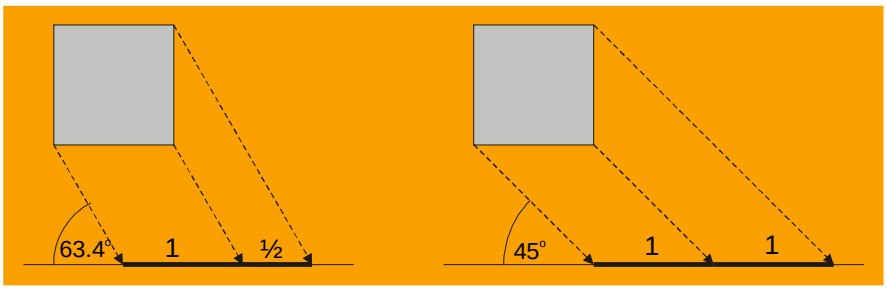
Parallel Projection: Oblique

- **Projectors are not perpendicular to the viewplane**
  - For viewplane perpendicular to an axis
  - Combination of plan-view projection and axonometric projection
  - Shows faces parallel to viewplane with proper length and angles
  - Uniform foreshortening of faces non-parallel to viewplane



Parallel Projection: Oblique

- **Angle between viewplane and projectors determines foreshortening**
  - 45 degrees: Cavalier projection
    - All axis appear at true length
    - Easy to measure lengths, but looks unrealistic
  - 63.4 degrees: Cabinet projection
    - Lines perpendicular to the viewplane are shortened to half length
    - Closer to perspective display

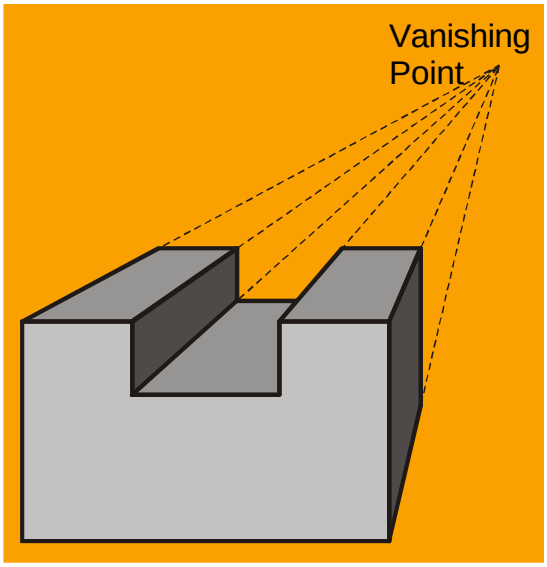


Perspective Projection: Properties

- **Distant objects appear smaller than close ones**
  - Perspective foreshortening
- **Lines map into lines**
- **Distances and angles are not maintained**
  - In particular parallel line do not remain parallel
  - Angles are maintained only for lines parallel to the view plane

Perspective Projections: Vanishing Points

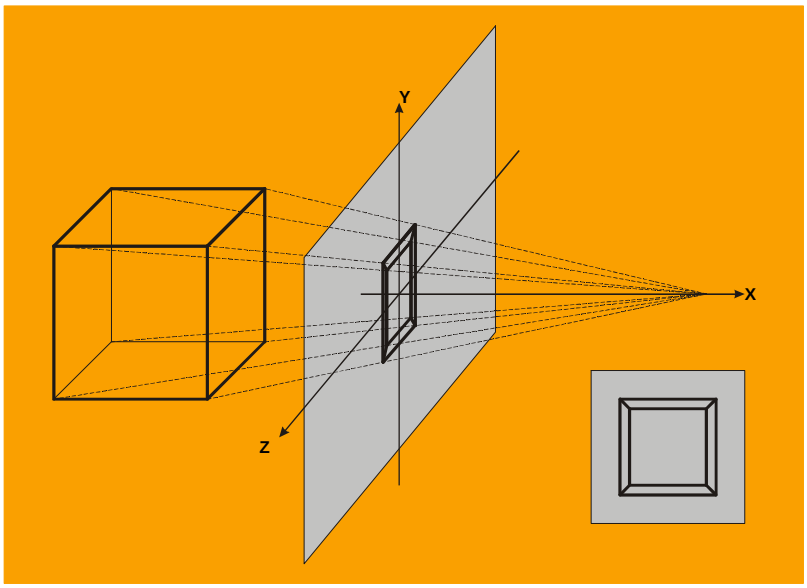
- **Parallel lines not parallel to the view plane intersect in a single point: the *vanishing point* for those lines**
  - Vanishing points can be used to construct perspective drawings



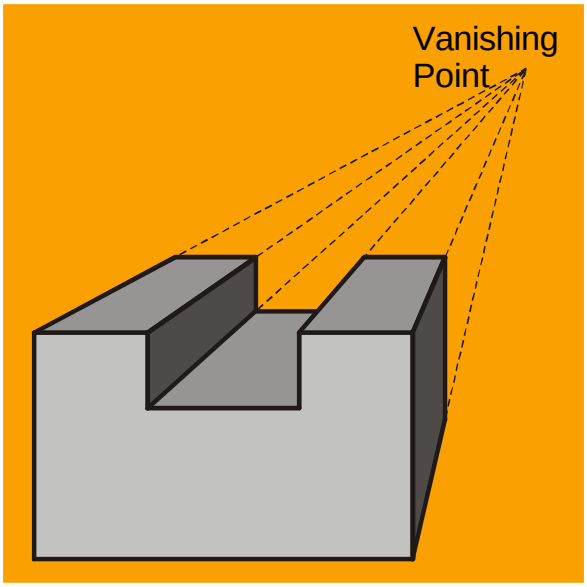
Perspective Projection: Vanishing Points

- **Lines parallel to one of the axes converge in a *principal vanishing point***
- **Number of principal vanishing points equals the number of axes intersected by the view plane.**
  - In 3D there are up to 3 prinicipal vanishing points.
  - Most frequently used are 1 and 2 vanishing points
  - Three vanishing points add little realism over two vanishing points

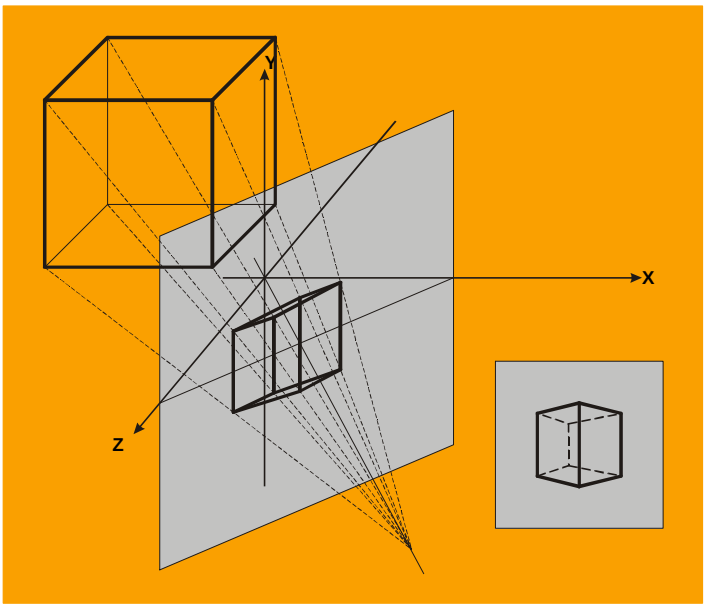
Perspective Projection: One-point Perspective



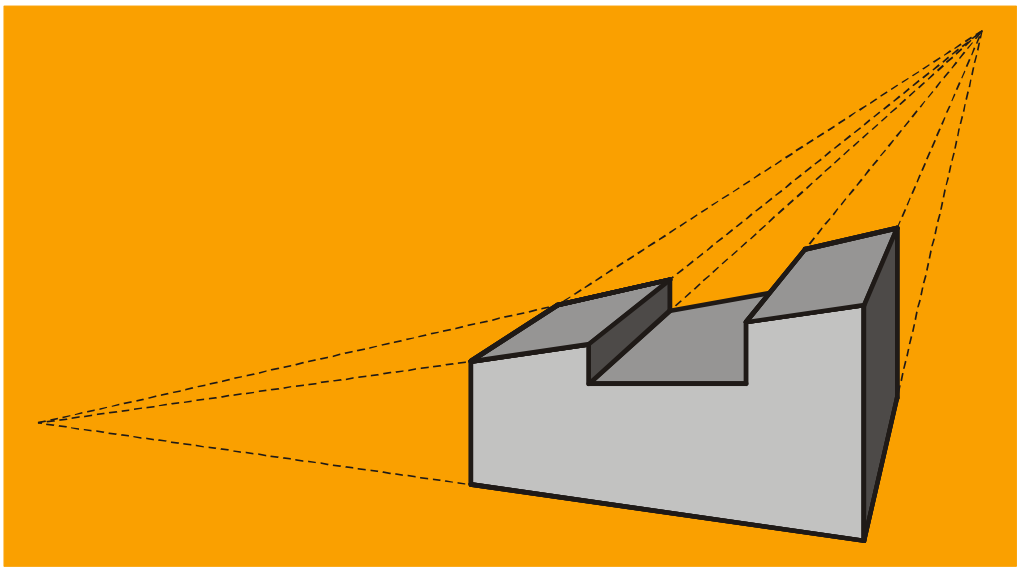
Perspective Projection: 1 Vanishing Point



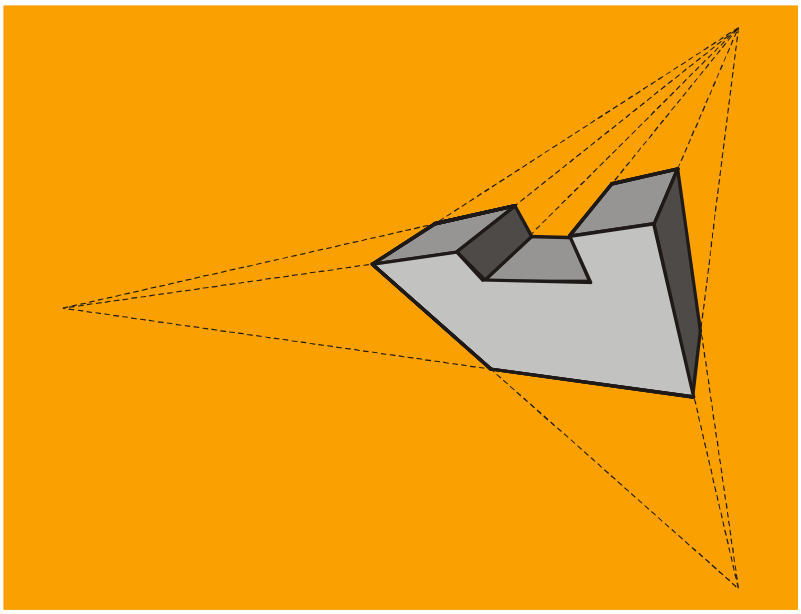
Perspective Projection: Two-Point Perspective



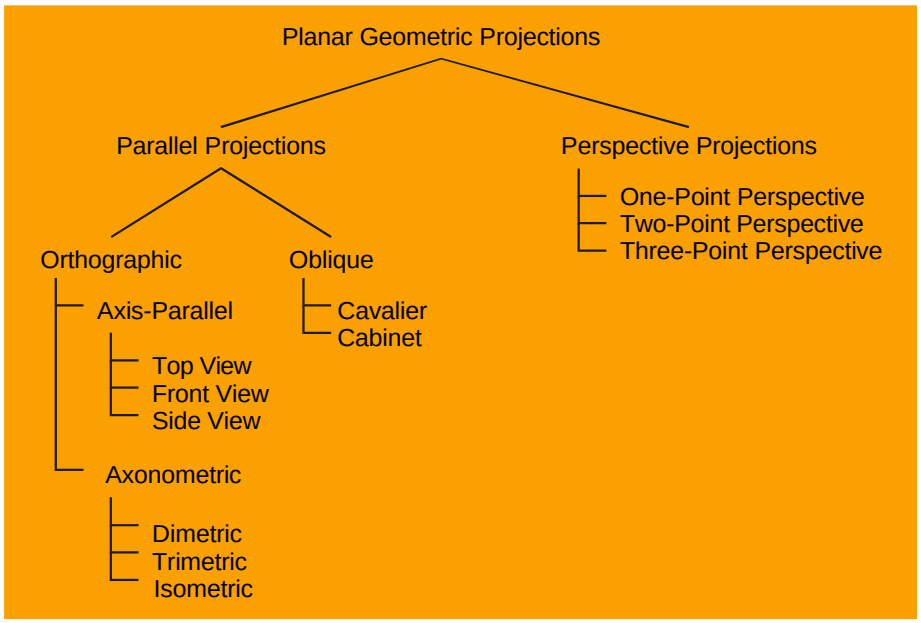
Perspective Projection: 2 Vanishing Points



Perspective Projection: 3 Vanishing Points



Projection Types: Summary

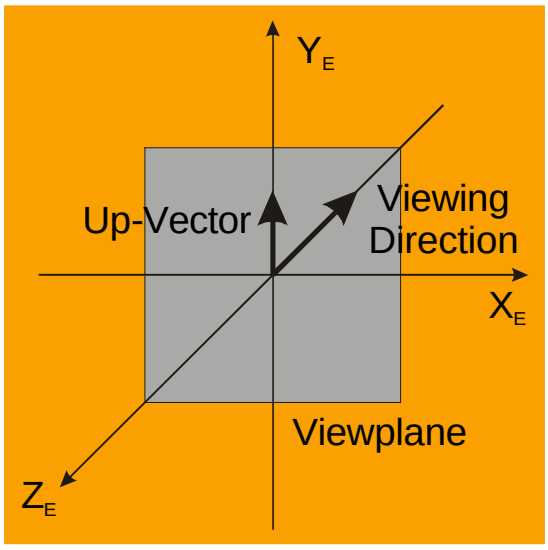


Projections: Mathematical Treatment

- **How can we describe all these types of projections ?**
- **Overview:**
  - Define a suitable coordinate system
  - Compute the mapping of points from 3-space on the viewplane
  - Determine matrix form

Projection: Eye Coordinate System

- **Viewplane: z=0**
  - Viewing direction down negative z-axis
- **Up vector: y-axis**
- **Project onto z=0 plane**



Orthographic Projection

- Projection by "dropping" z-coordinate

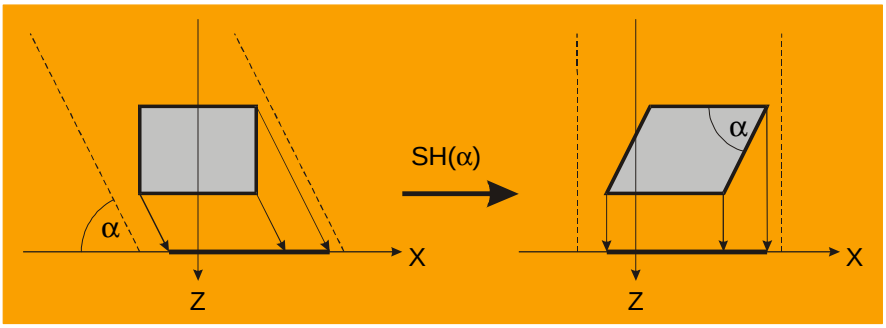
$$\mathbf{P}_S = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \mathbf{P}_E$$

$$\mathbf{P}_S = \mathbf{M}_{OZ} \bullet \mathbf{P}_E$$

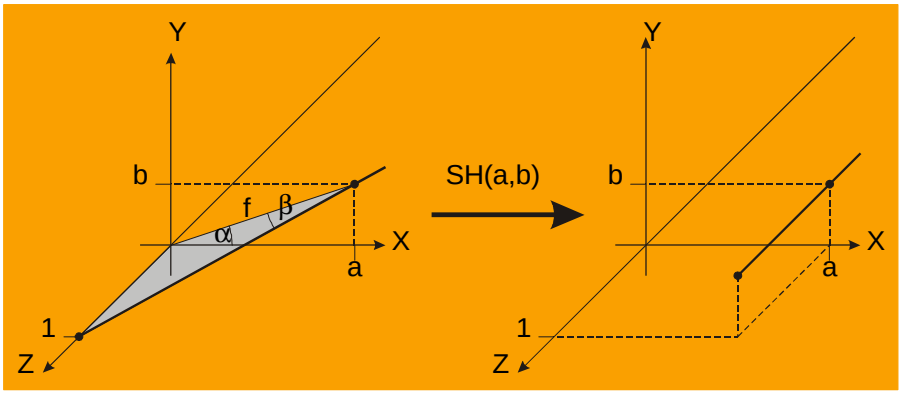
- Projection along other axes can be achieved similarly.
- This matrix will be the final part of all other projections.

Oblique Projections: 2D Explanation

- Two-step process
  - Oblique transformation
  - Shear along the x-axis
  - Orthographic projection along the z-axis



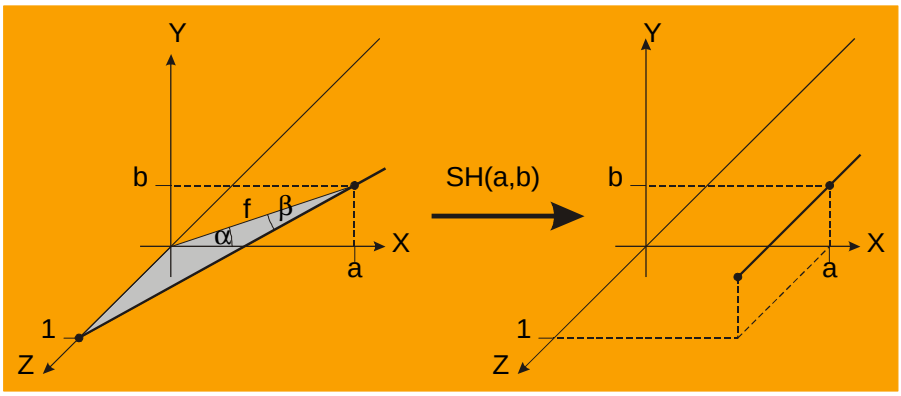
Oblique Projection: 3D Analysis



$$\mathbf{P}_s = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \mathbf{P}_E = \begin{pmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \mathbf{P}_E$$

$$\mathbf{P}_s = \mathbf{SH}(a,b) \bullet \mathbf{M}_{OZ} \bullet \mathbf{P}_E = \mathbf{M}_{ObZ} \bullet \mathbf{P}_E$$

Oblique Projection: 3D Analysis (cont'd)



$f = \cot b$ (Foreshortening factor)

$$\mathbf{M}_{ObZ} = \begin{pmatrix} 1 & 0 & -a & 0 \\ 0 & 1 & -b & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & f\cos a & 0 \\ 0 & 1 & f\sin a & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Two-Step Process

- **The two-step process is also used with other projections, in particular with perspective projection.**
  - First projection *transformation,* i.e. distortion of objects
  - Then, orthographic *projection*
- **This a typical divide-and-conquer approach**
  - Partition problem into simpler subproblems

Perspective Projection

- **We will look at a simple and common case**
  - Viewplane at $z = 0$
  - Center of projection at $z = d$
  - This can always be ensured with the appropriate combination of viewing and projection transformations !
- **Analysis of a general projection**
  - Center of projection off the z-axis
  - Line through center of projection and center of window not perpendicular to viewplane
  - Parallel and perspective projections
  - See Foley et al. or backup charts at the end of this chart set.

Perspective Transformation

$$\frac{x'}{x} = \frac{d}{d-z} \;;\; \frac{y'}{y} = \frac{d}{d-z}$$

$$\Leftrightarrow$$

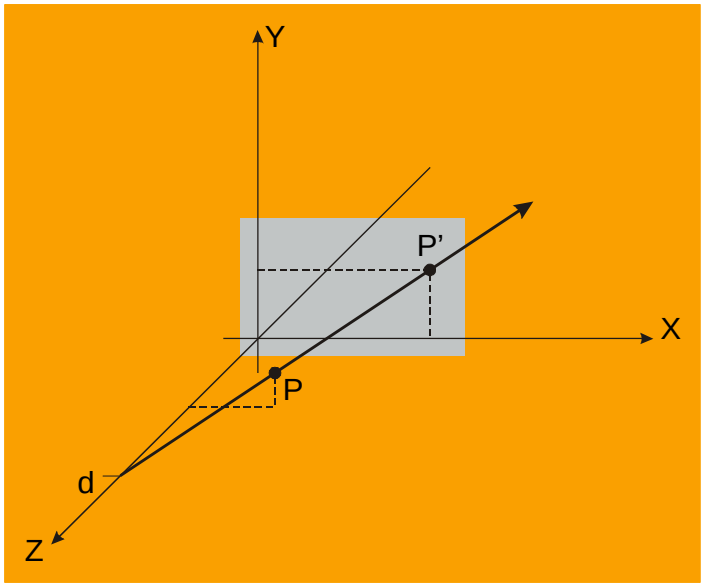
$$x' = \frac{x}{1 - z/d} \;;\; y' = \frac{y}{1 - z/d}$$

By analogy we define

$$z' = \frac{z}{1 - z/d}$$

$$\mathbf{P}' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 1 \end{pmatrix} \bullet \mathbf{P}$$

$$\mathbf{P}' = \mathbf{M}_{\mathbf{PER}} \bullet \mathbf{P}$$



Perspective Projection

- **Perform orthographic projection after perspective transformation**

$$\mathbf{P}' = \mathbf{M}_{\mathbf{Oz}} \bullet \mathbf{M}_{\mathbf{PER}} \bullet \mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 1 \end{pmatrix} \bullet \mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/d & 1 \end{pmatrix} \bullet \mathbf{P}$$

Perspective Projection: Vanishing Points

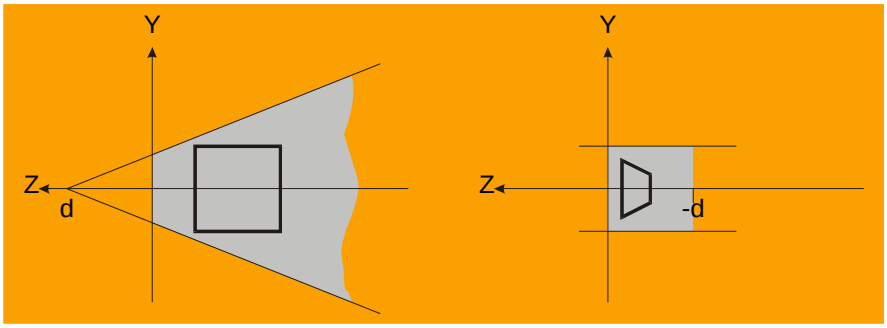
- Mapping of points at infinity

$$\mathbf{VP} = \mathbf{M}_{\mathbf{PER}} \bullet \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 1 \end{pmatrix} \bullet \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1/d \end{pmatrix} \Rightarrow \quad \mathbf{vp} = \begin{pmatrix} 0 \\ 0 \\ -d \end{pmatrix}$$

- Note that the vanishing point for lines parallel to the direction of projection (z-axis) is reflection of the center of projection at the view plane.

Perspective Transformation: Space Distortion

- Remember: Perspective projection is a combination of perspective transformation and orthographic projection
  - Distant objects appear smaller
  - Lines along the viewing direction map into parallels to the z-axis.
  - View frustum is transformed into a box
- Space is increasingly "squashed" as z grows.



Perspective Transformation: Coordinate Mapping (1)

- The perspective transformation remaps z !

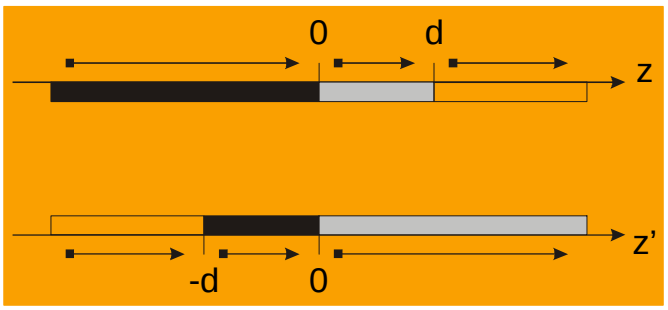
$$\mathbf{P}' = \mathbf{M}_{PER} \bullet \mathbf{P} \quad \Leftrightarrow \quad \begin{pmatrix} X' \\ Y' \\ Z' \\ W' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 1 \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} X'/W' \\ Y'/W' \\ Z'/W' \end{pmatrix}$$

$$Z' = z \;,\; W' = 1 - z/d \;,\; z' = \frac{z}{1 - z/d} = \frac{1}{1/z - 1/d}$$

$$z:[-\infty,0] \rightarrow z':[-d,0] \qquad z:[0,d] \rightarrow z':[0,+\infty] \qquad z:[d,+\infty] \rightarrow z':[+\infty,-d]$$

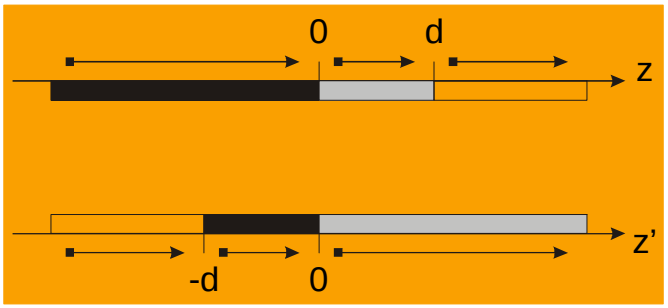
Perspective Transformation: Coordinate Mapping (2)

- The perspective transformation remaps z !
  - Points between eye and projection plane ($0 < z < d$) map to behind the eye ($z'>0$).
  - Points beyond the projection plane ($z<0$) map to $-d < z' < 0$.
  - Points behind the eye ($z > d$) map to $z' < -d$.
  - Points behind the eye wrap to in front of the eye !!



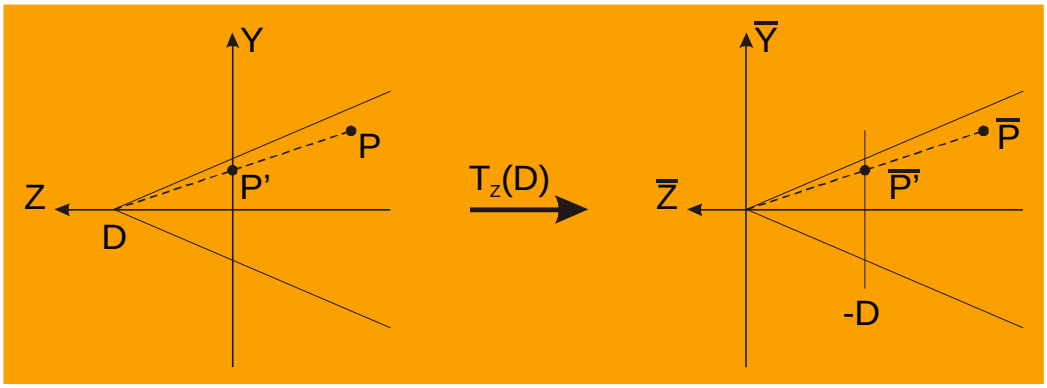
Perspective Transformation: Coordinate Mapping (3)

- **The perspective transformation remaps z !**
  - Objects extending from behind the eye to in front of the eye
  - This requires special attention unless any objects (or parts of objects) behind the eye are eliminated, i.e. clipped.
  - Therefore, we introduce clip planes to reject points behind the eye.



Perspective Transformation: Viewpoint at the Origin (1)

- **Transform coordinate system such, that *z* = *d* maps to *z* = 0.**
  - Translate coordinate system by +*d* along the z-axis.



Perspective Transformation: Viewpoint at the Origin (2)

- **Transform coordinate system such, that $z = d$ maps to $z = 0$.**
  - Translate coordinate system by $+d$ along the z-axis.

$$\tilde{\mathbf{P}}' = \tilde{\mathbf{M}}_{PER} \bullet \tilde{\mathbf{P}} = \mathbf{M}_{PER} \bullet \mathbf{T}_Z(d) \bullet \mathbf{P}$$

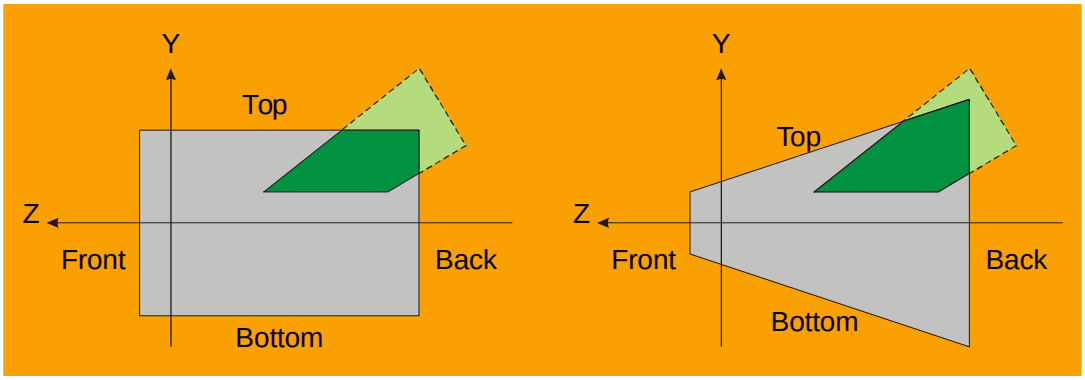
$$\tilde{\mathbf{M}}_{PER} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 1 \end{pmatrix} \bullet \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & -1/d & 0 \end{pmatrix}$$

Clipping

- **Clipping is the process of eliminating parts of the scene outside the view volume.**
- **Clipping serves multiple purposes**
  - Elimination of "fake" objects, that are wrapped in front of the eye by the perspective transformation.
  - Reduction of geometry that is processed during rasterization.
  - Prevention of numerical problems for objects outside the view volume.
  - Decrease of visual clutter by removal of close and distant objects.
- **The view volume is delimited by clipping planes.**
  - Reasonable approximation to human view volume, that is described by overlapping view cones.
  - Matches the view volume of a photographic camera.

Clipping: View Volume

- **Clipping planes define the extent of the view volume.**
  - Perspective view volume a.k.a. view frustum
  - Front and back clipping planes a.k.a. hither and yon.
  - Front and back clipping planes are optional.



Clipping Algorithm

- **Each object must be intersected with each clip plane.**
- **Many objects are delimited by edges or polygons.**

- **Edge-plane intersection:**
  - Straight-forward process
  - However, intersection of lines with arbitrary planes is compute-intensive.

- **Therefore, we define a *canonical view volume.***
  - Intersection calculations are simplified
  - Transformation to the canonical view volume is "free". It can be combined with the viewing transformation

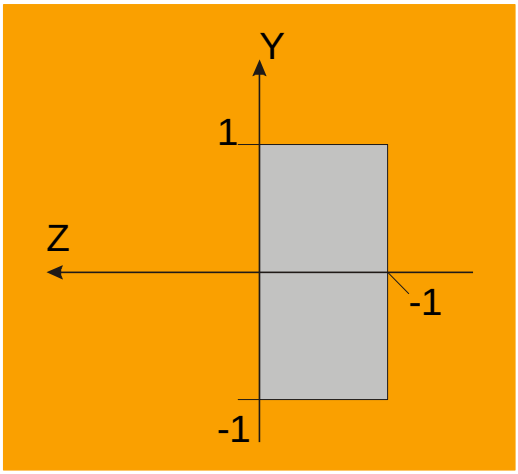
Computing a Canonical View Volume

- **Definition of canonical view volume**
- **Transformation of the view volume after viewing transformation into the canonical view volume**
- **Process is known as *normalizing* the view volume**
  - Coordinates are called Normalized Device Coordinates (NDC)
- **Similar procedure for parallel and perspective projection**
  - We will treat them separately

Normalization: Parallel Projection

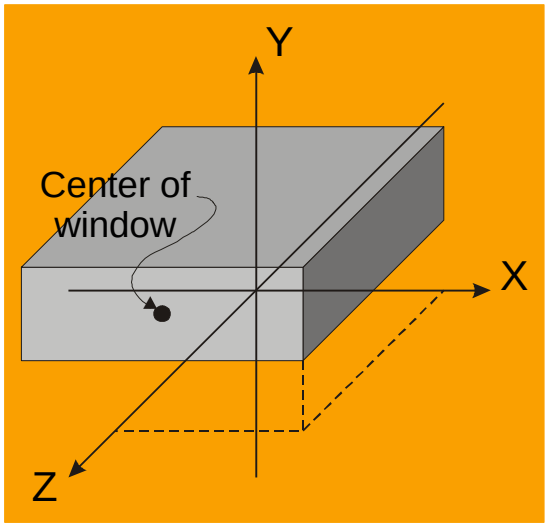
Normalization: Parallel Projection (1)

- **Axis-aligned box**
  - Right, Left $x = +/-1$
  - Top, Bottom $y = +/-1$
  - Front, Back $z = 0, -1$
  - Much simpler intersection calculations.
  - Cost of computing canonical view volume is amortized over many objects.



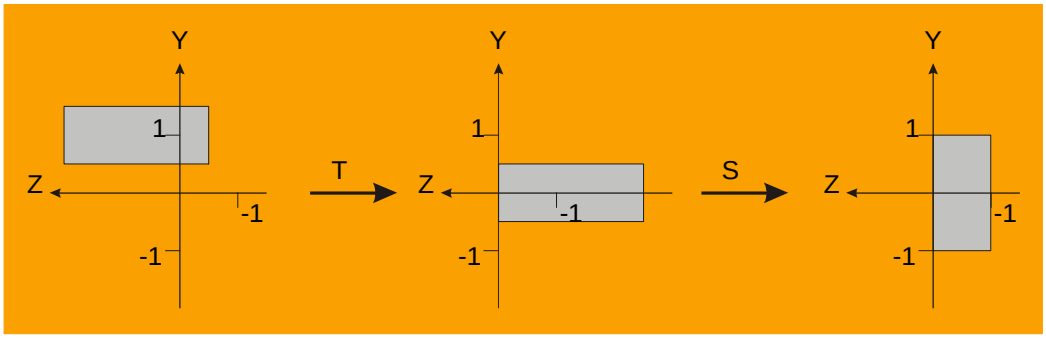
Normalization: Parallel Projection (2)

- **View volume after viewing transformation:**
  - Viewing direction along negative z-axis
  - Clip planes asymmetric around z-axis and not aligned with ***z*** = 0.
- **X and Y clipping planes define a *window*.**
  - Center of the window may be displaced from the origin.
  - Width and height determine the window aspect ratio.
  - The window will eventually get mapped to the viewport.



Normalization: Parallel Projection (3)

- **Steps to create the canonical view volume from the view volume after the viewing transformation:**
  - Translate the view volume such that the front clipping plane lies in the plane $z = 0$.
  - Translate and scale such that the other clip planes become the canonical clip planes.



Normalization: Parallel Projection (4)

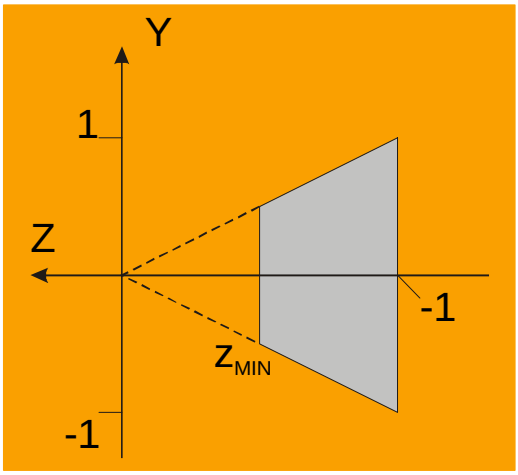
- **Center of window is at $C = (C_X, C_Y, C_Z)^T$**
- **Window has the dimensions W x H x D**
- **Then we get for the normalization transformation:**

$$\mathbf{N}_{PAR} = \mathbf{S} \bullet \mathbf{T} = \begin{pmatrix} 2/W & 0 & 0 & 0 \\ 0 & 2/H & 0 & 0 \\ 0 & 0 & 1/D & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} 1 & 0 & 0 & -C_X \\ 0 & 1 & 0 & -C_Y \\ 0 & 0 & 1 & -C_Z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2/W & 0 & 0 & -2C_X/W \\ 0 & 2/H & 0 & -2C_Y/H \\ 0 & 0 & 1/D & -C_Z/D \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Normalization: Perspective Projection

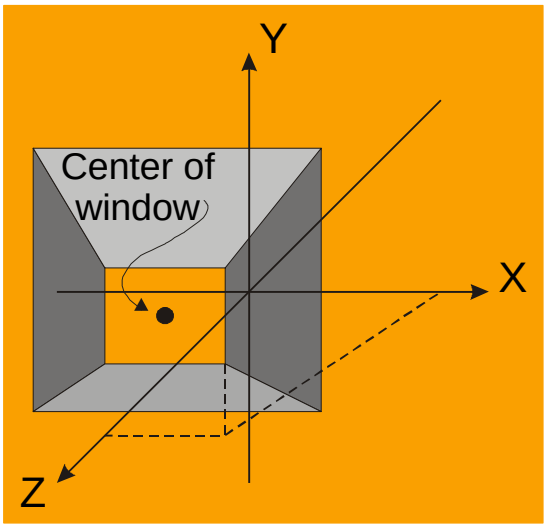
Normalization: Perspective Projection (1)

- **Truncated pyramid with tip at the origin**
  - Right, Left $x = +/-z$
  - Top, Bottom $y = +/-z$
  - Front, Back $z = z_{MIN}, -1$



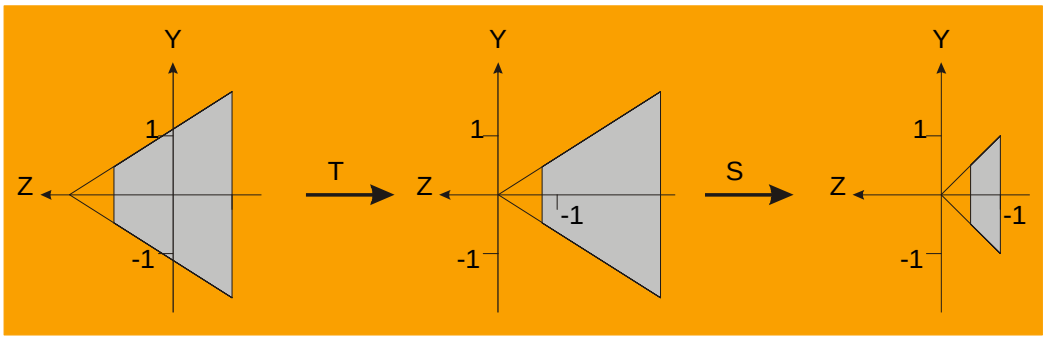
Normalization: Perspective Projection (2)

- **View volume after viewing transformation:**
  - Viewing direction along negative z-axis
  - Clip planes asymmetric around z-axis and not aligned with ***z*** = 0.
- **X and Y clipping planes define a *window*.**
  - Center of the window may be displaced from the origin.
  - Width and height determine the window aspect ratio.
  - The window will eventually get mapped to the viewport.



Normalization: Perspective Projection (3)

- **Steps to create the canonical view volume from the view volume after the viewing transformation:**
  - Translate the view volume such that the center of projection lies at the origin
  - Translate and scale such that the other clip planes become the canonical clip planes.



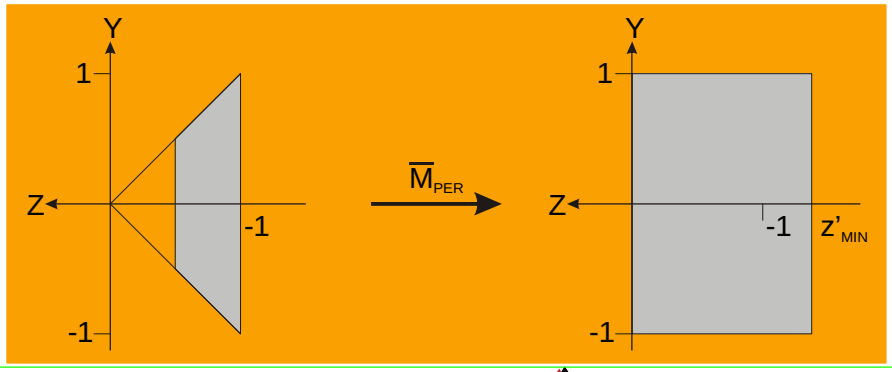
Normalization: Perspective Projection (4)

- Center of projection is at origin
- Window has the dimensions *W* x *H* x *D*
- Then we get for the normalization transformation:

$$\mathbf{N}_{\mathbf{PER}} = \mathbf{S} \bullet \mathbf{T} = \begin{pmatrix} 2/W & 0 & 0 & 0 \\ 0 & 2/H & 0 & 0 \\ 0 & 0 & 1/D & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2/W & 0 & 0 & 0 \\ 0 & 2/H & 0 & 0 \\ 0 & 0 & 1/D & -d/D \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Perspective Division (1)

- After normalization, perspective transformation turns the canonical viewing frustum into a axis-aligned box.
- We would like to convert that box into the canonical view volume for the parallel projection.
- First: Compute z'_{MIN} ...



Perspective Division (2)

$$\tilde{\mathbf{M}}_{PER} \bullet (CP\text{-}top,\ CP\text{-}bot,\ CP\text{-}front,\ CP\text{-}back) \quad \text{for } d = 1$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix} \bullet \begin{pmatrix} x & x & x & x \\ -z & z & y & y \\ z & z & z_{MIN} & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} x & x & x & x \\ -z & z & y & y \\ z+1 & z+1 & z_{MIN}+1 & 0 \\ -z & -z & -z_{MIN} & 1 \end{pmatrix}$$

Top: y = 1
Bottom: y = -1
Front: z = -(z_N+1)/z_N
Back: z = 0

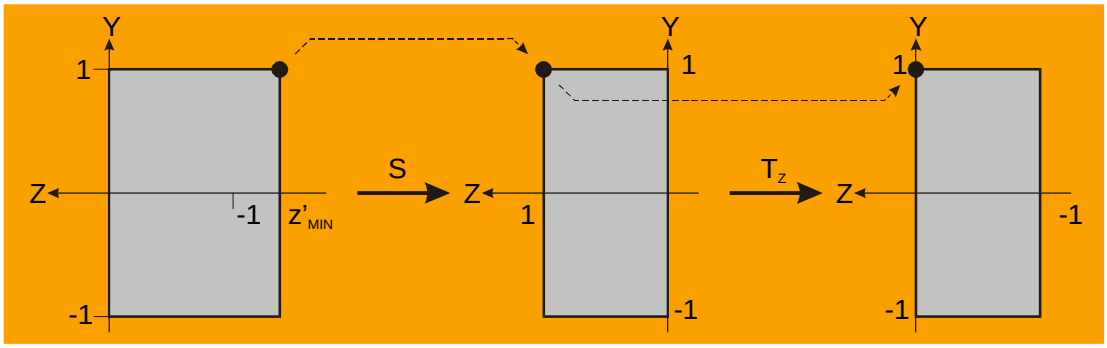
Perspective Division (3)

- **No adjustment of necessary for X and Y dimensions**
- **Z dimension:**
  - Near clip plane farther from viewer than far clip plan
  - Z orientation reversed
  - Depth range not 0 ... -1
    - -(z_{MIN}+1)/z_{MIN} = -1 - 1/z_{MIN} < -1 for 0 > z_{MIN} > -1
- **Steps:**
  - Reverse orientation by reflection on z=0 and scale to depth = 1
  - Transform so that near clip plane maps to z=0

Perspective Division (4)

$$\mathbf{M} = \mathbf{T}_z \bullet \mathbf{S} \bullet \tilde{\mathbf{M}}_{PER}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{z_{MIN}}{z_{MIN}+1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{z_{MIN}+1} & \frac{-z_{MIN}}{z_{MIN}+1} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$



Viewport Mapping (1)

- **Now both, parallel and perspective transformation have yielded the same normalized view volume.**
- **Final step: Map the normalized device coordinates to screen coordinates**

- **Terminology** (to be consistent with the textbook**)**
  - Window: Rectangle in World-coordinates describing (together with the projection) which objects are visible
    - Do not confuse with on-screen windows, managed by the window manager !!!
  - Viewport: On-screen rectangle describing where the image will appear on the screen

Viewport Mapping (2)

- **Window in world coordinates:**
  - Width W_w and Height H_w
  - Aspect ratio $A_w = W_w / H_w$
- **Viewport in screen coordinates:**
  - Width W_v and Height H_v
  - Aspect ratio $A_v = W_v / H_v$
- **Window aspect ratio should be maintained**
- **If the aspect ratios are not equal, several choices**
  - Fit the window into the viewport, leaving part of the viewport empty
  - Fill the the viewport, clipping off parts of the window

Viewport Mapping (3)

- **Transformation to the canonical view volume has destroyed the window aspect ratio (now: unit square)**
- **Window aspect ratio can be restored by non-uniform scaling of the unit square.**
- **Steps:**
  - Restore window aspect ratio (normalized device coordinates)
  - Compare with viewport aspect ratio
  - Determine appropriate scale factor (device coordinates)
  - Translate adjusted window to viewport

Viewport Mapping (4)

- **Assuming that $A_w > 1$ and that $W_V/W_w < H_V/H_w$**
  - Center of viewport at (v_X, v_Y)

$$\mathbf{M}_{NV} = \mathbf{T} \bullet \mathbf{S}_V \bullet \mathbf{S}_N = \begin{pmatrix} 1 & 0 & v_X \\ 0 & 1 & v_Y \\ 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} W_V / W_W & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1/A_W & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} W_V / W_W & 0 & v_X \\ 0 & 1/A_V & v_Y \\ 0 & 0 & 1 \end{pmatrix}$$

- **Sometimes screen coordinates are left-handed coordinate systems, e.g. origin at top-right corner and z-axis pointing out of the screen.**
  - Then the y-axis must be reflected during the viewport mapping.

That's basically it !

- **To summarize ...**

Modeling and Viewing

- **Modeling**
  - Transforms objects from local model coordinates to world coordinates
  - Typically done with linear transformations
- **Viewing**
  - Places and orients the camera in world coordinates
  - Look down negative z-axis of EC
  - Up vector along y-axis of EC

Clipping

- **Clip planes define a view volume**
- **Clipping intersects objects with view frustum**
- **Clipping occurs in normalized device coordinates**
- **Canonical view volume is delimited by planes parallel or at 45^{o} angle to coordinate axes**
- **Canonical view volume facilitates clipping**

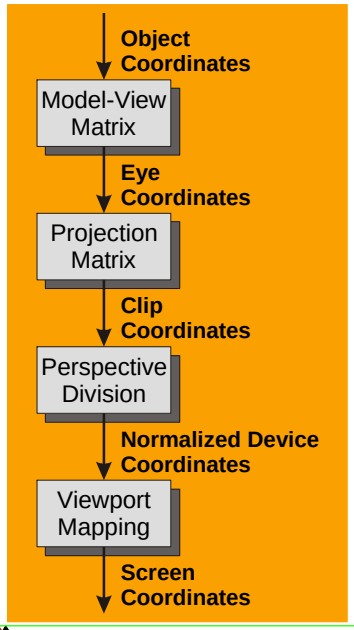
Projection and Viewport Mapping

- **Perspective Transformation**
  - Distort objects such that view volume becomes a rectilinear box
  - Remap the transformed the perspective canonical view volume into the parallel canonical view volume.
- **Parallel Projection**
  - "Drop" the z-coordinate, i.e. project onto z=0.
- **Viewport Mapping**
  - Map the projected scene from normalized device coordinates to screen coordinatesMap the projected scene from normalized device coordinates to screen coordinates

OpenGL

OpenGL: Rendering Pipeline

- **Very similar to our discussion of the various transformations**
- **Single step for modeling + viewing transformation**
  - World coordinates are not explicitely specified
- **Projection Matrix creates canonical view volume**



OpenGL: Viewing Transformation

- **All modeling transformations**
  - glRotate(), glTranslate(), glScale(), glLoadMatrix(), ...
- **gluLookAt()**
  - Specifies in current coordinate system: viewpoint, look-at point and up-vector
  - Viewpoint and Look-at point specify viewing direction
  - Can be re-implemented using standard modeling transformations

OpenGL: Projections

- **glOrtho()**
  - Specifies orthographic view volume, delimited by 6 clip planes
- **glFrustum()**
  - Specifies perspective view volume as the position and size of the near clip window (and implicitely the viewpoint).
- **gluPerspective()**
  - Wrapper around glFrustum() to more conveniently define a perspective view volume based on ...
  - field of view and aspect ratio of the view volume
  - distance of the near and far clip planes

Summary

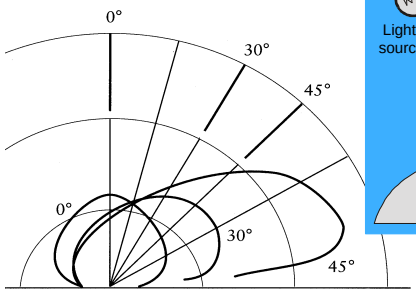
- **Mathematical description of**
  - Viewing Transformations
  - Clipping Volume
  - Projections (parallel and perspective)
  - Viewport mapping

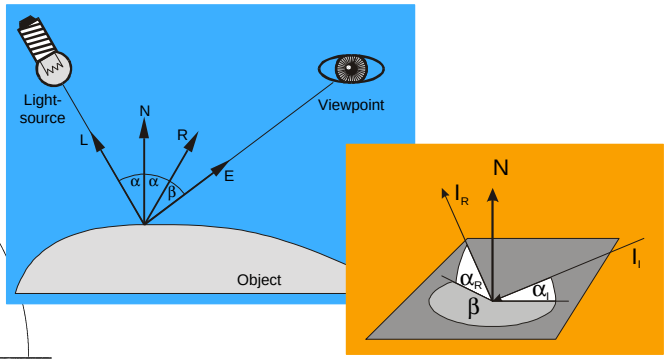
Homework

- OpenGL rendering primitives
- Foley et al. Chapter 16

Next Week ...

- Lighting Models





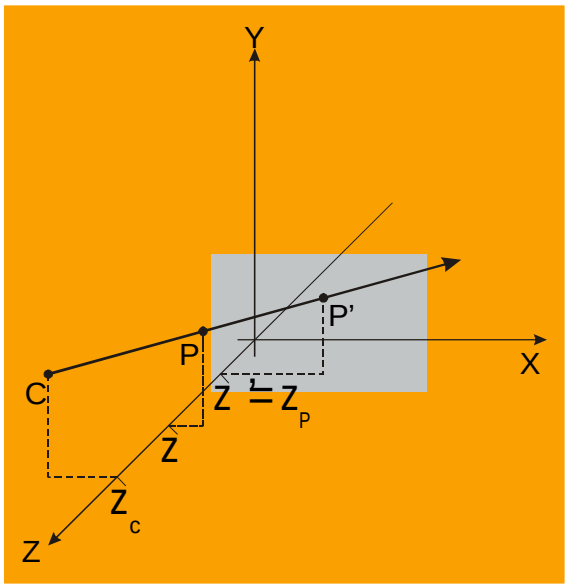
$$I = k_A \cdot I_A + k_D \cdot \sum_{i=0}^{l} I_i \cdot (\mathbf{N} \cdot \mathbf{L}_i) + k_S \cdot \sum_{i=0}^{l} I_i (\mathbf{R} \cdot \mathbf{E})^n$$

Backup Charts

- General Projection

Projection: General Case (1)

- Viewplane at $z=z_P$
- Center of projection C off the z-axis



Projection: General Case (2)

$$P' = C + l(P - C) \quad ; \quad z' = z_P$$

$$\Rightarrow$$

$$z' = z_P = z_C + l(z - z_C)$$

$$\Rightarrow$$

$$l = \frac{z_P - z_C}{z - z_C} \quad ; \quad \Lambda = 1/\ l = \frac{z - z_C}{z_P - z_C}$$

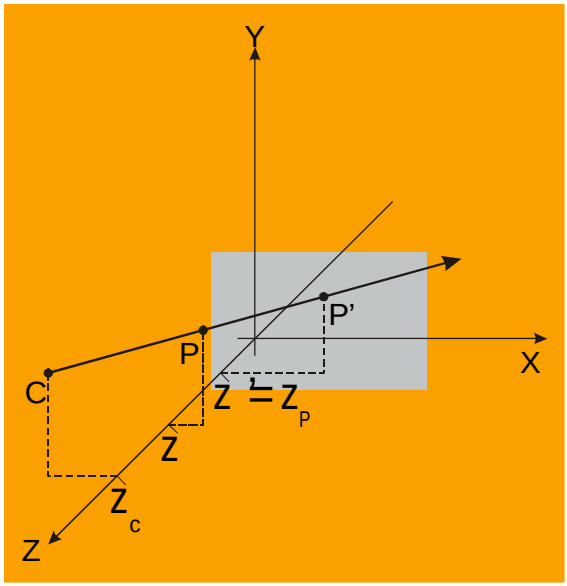
$$\Rightarrow$$

$$x' = x_C + l(x - x_C)$$

$$= x_C \frac{\Lambda}{\Lambda} + \frac{x - x_C}{\Lambda} = \frac{x + x_C(\Lambda - 1)}{\Lambda}$$

$$= \frac{x + x_C \dfrac{z - z_P}{z_P - z_C}}{\Lambda} = \frac{x + z\dfrac{x_C}{z_P - z_C} - \dfrac{x_C \cdot z_P}{z_P - z_C}}{\Lambda}$$

$$y' = \frac{y + z\dfrac{y_C}{z_P - z_C} - \dfrac{y_C \cdot z_P}{z_P - z_C}}{\Lambda}$$



Projection: General Case (3)

$$x' = \frac{1}{\Lambda} \cdot \left(x + z\frac{x_C}{z_P - z_C} - \frac{x_C \cdot z_P}{z_P - z_C} \right)$$

$$y' = \frac{1}{\Lambda} \cdot \left(y + z\frac{y_C}{z_P - z_C} - \frac{y_C \cdot z_P}{z_P - z_C} \right)$$

$$z' = z_P = z_P \frac{\Lambda}{\Lambda} = \frac{1}{\Lambda} \cdot z_P \frac{z - z_C}{z_P - z_C}$$

$$= \frac{1}{\Lambda} \cdot \left(z\frac{z_P}{z_P - z_C} - \frac{z_P \cdot z_C}{z_P - z_C} \right)$$

$$w' = \Lambda = \frac{z}{z_P - z_C} - \frac{z_C}{z_P - z_C}$$

$$\mathbf{P}' = \begin{pmatrix} 1 & 0 & \dfrac{x_C}{z_P - z_C} & -x_C\dfrac{z_P}{z_P - z_C} \\ 0 & 1 & \dfrac{y_C}{z_P - z_C} & -y_C\dfrac{z_P}{z_P - z_C} \\ 0 & 0 & \dfrac{z_P}{z_P - z_C} & -z_C\dfrac{z_P}{z_P - z_C} \\ 0 & 0 & \dfrac{1}{z_P - z_C} & -\dfrac{z_C}{z_P - z_C} \end{pmatrix} \bullet \mathbf{P}$$

$$= \begin{pmatrix} 1 & 0 & x_C/d & -x_C \cdot z_P/d \\ 0 & 1 & y_C/d & -y_C \cdot z_P/d \\ 0 & 0 & x_P/d & -z_C \cdot z_P/d \\ 0 & 0 & 1/d & -z_C/d \end{pmatrix} \bullet \mathbf{P}$$

$$\mathbf{P}' = \mathbf{P}_{per} \bullet \mathbf{P}$$

Projection: General Case (4)

$$\mathbf{P}' = \mathbf{P}_{\text{per}} \bullet \mathbf{P} = \begin{pmatrix} 1 & 0 & x_C / d & -x_C \cdot z_P / d \\ 0 & 1 & y_C / d & -y_C \cdot z_P / d \\ 0 & 0 & x_P / d & -z_C \cdot z_P / d \\ 0 & 0 & 1/d & -z_C / d \end{pmatrix} \bullet \mathbf{P}$$

Orthographic Projection: $z_P = 0 \;,\; C = \lim_{z \to \infty} \begin{pmatrix} 0 & 0 & z \end{pmatrix}$

Cavalier Projection: $z_P = 0 \;,\; C = \lim_{z \to \infty} \begin{pmatrix} z & 0 & z \end{pmatrix}$

Perspective Projection: $z_P = 0 \;,\; C = \begin{pmatrix} 0 & 0 & d \end{pmatrix}$

Perspective Transformation: Clip Planes

- How are the clip planes transformed by the perspective transformation ?

Back CP
Front CP
Bottom CP
Top CP

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 1 \end{pmatrix} \bullet \begin{pmatrix} x & x & x & x \\ y & y & -a \cdot (z-d) & a \cdot (z-d) \\ z_F & z_B & z & z \\ 1 & 1 & 1 & 1 \end{pmatrix} =$$

$$\begin{pmatrix} x & x & x & x \\ y & y & -a \cdot (z-d) & a \cdot (z-d) \\ z_F & z_B & z & z \\ 1 - \frac{z_F}{d} & 1 - \frac{z_B}{d} & 1 - \frac{z}{d} & 1 - \frac{z}{d} \end{pmatrix}$$

Perspective Transformation: Clip Planes

- Plane at $y = y'_B$
- Plane at $y = y'_T$
- Plane at $z = z'_B$
- Plane at $z = z'_F$

$$\begin{pmatrix} \frac{x \cdot d}{d - z_F} & \frac{x \cdot d}{d - z_B} & \frac{x \cdot d}{d - z} & \frac{x \cdot d}{d - z} \\ \frac{y \cdot d}{d - z_F} & \frac{y \cdot d}{d - z_B} & \frac{-a \cdot d \cdot (z - d)}{d - z} & \frac{a \cdot d \cdot (z - d)}{d - z} \\ \frac{z_F \cdot d}{d - z_F} & \frac{z_B \cdot d}{d - z_B} & \frac{z \cdot d}{d - z} & \frac{z \cdot d}{d - z} \end{pmatrix} = \begin{pmatrix} x' & x' & x' & x' \\ y' & y' & y'_T & y'_B \\ z'_F & z'_B & z' & z' \end{pmatrix}$$