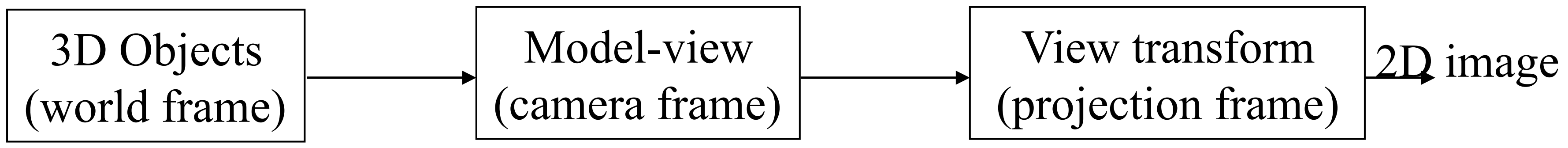
Viewing

Reading: Angel Ch.5

What is Viewing?

Viewing transform projects the 3D model to a 2D image plane

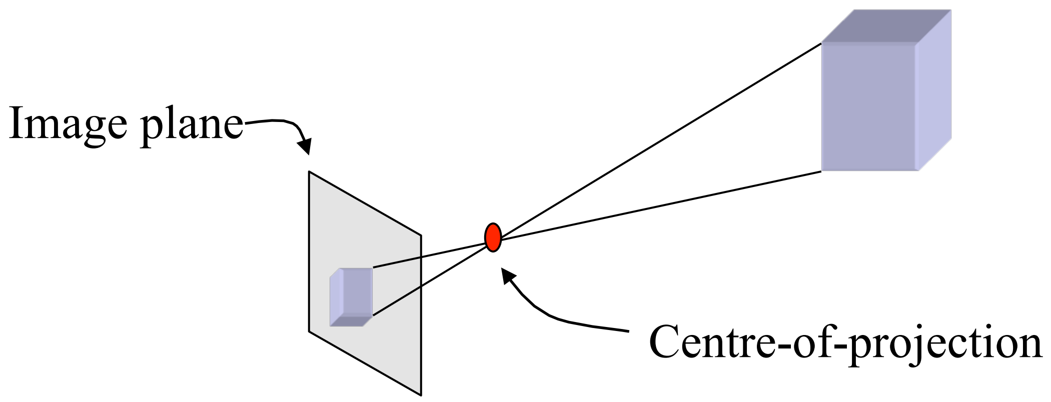


View transform models the camera projection
- pin-hole camera
- camera field of view

Understanding projections is **crucial** for writing graphics applications

Projection

Camera modelled using **geometric optics** ‘pin-hole camera’
- a ray of light travels in a straight line passing through the camera **centre-of-projection** and falling on the **image-plane**



Planar geometric projection - object is projected onto a plane

- Preserves straight lines
- Does not preserve angles

Camera Projection Models

2 Important camera projection models in computer graphics:

Perspective - Camera rays pass through a centre-of-projection at a distance d from the image plane

Parallel - all light rays are projected along parallel lines onto the image plane (centre-of-projection at infinity)

Both can be modelled by a 4x4 projection matrix in Homogeneous coordinates

Projection of a 3D point:

(1) Transform point from world to camera coordinates using 4x4 model-view matrix M in camera coordinates

$$x_c = M\, x_w$$

(2) Apply camera projection by 4x4 projection matrix C in camera coordinates

$$x_p = Cx_c = CM\, x_w$$

(3) Transform homogeneous (x,y,z,w) to real-coordinates (x/w,y/w,z/w)

Orthographic Projection

Special case of parallel projection

- image plane is orthogonal to direction of projection

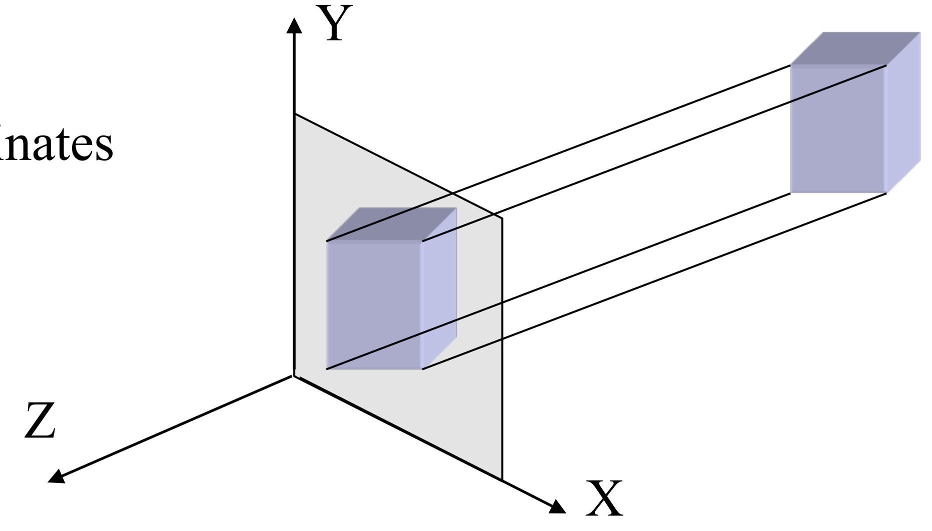
- distances orthogonal to projection plane are preserved
- simple to compute
- for a camera aligned with z-axis with projection plane at z_c=0

  $x_p = x_c$

  $y_p = y_c$

  $z_p = 0$
- in Homogeneous coordinates

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



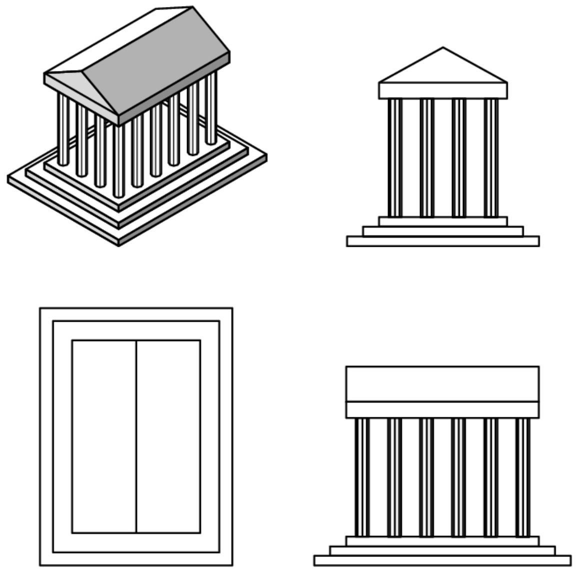
$$x_p = Cx_c$$

- linear projection matrix (no division)
- can be implemented using same hardware as perspective projection

Examples: Orthographic Views

Parallel projection with projection planes parallel to principal axis of object

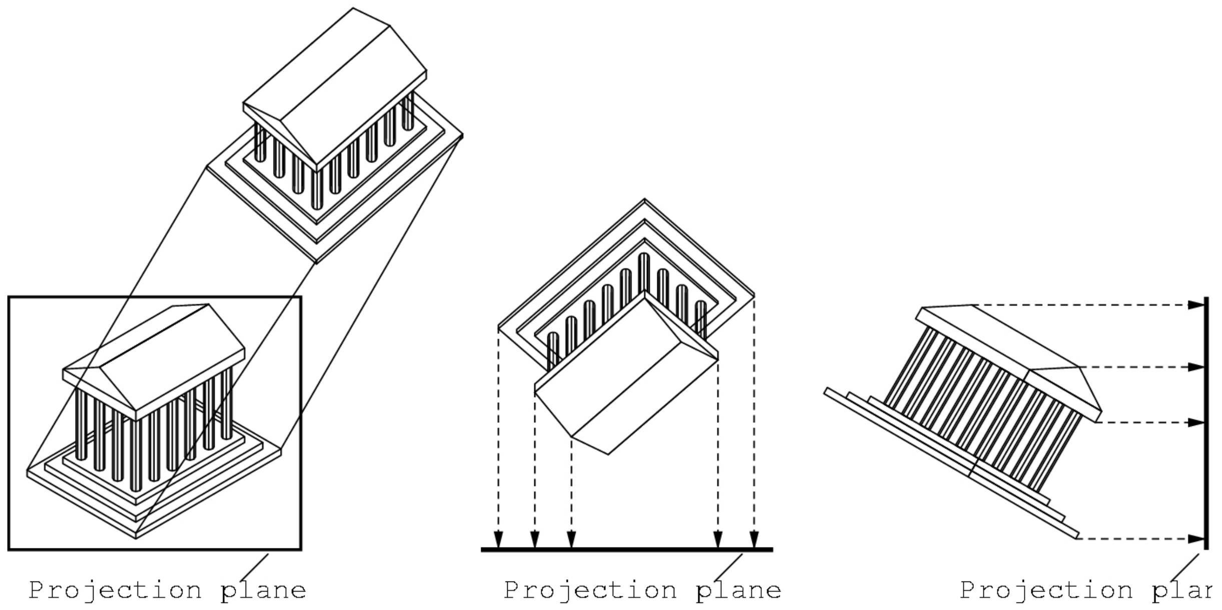
- length of lines is preserved
- angle between lines preserved



Example: Axonometric Views

Projection orthogonal to the projection plane from any general view

- line length in image is less than or equal to true 3D length
- parallel lines preserved
- angles not preserved

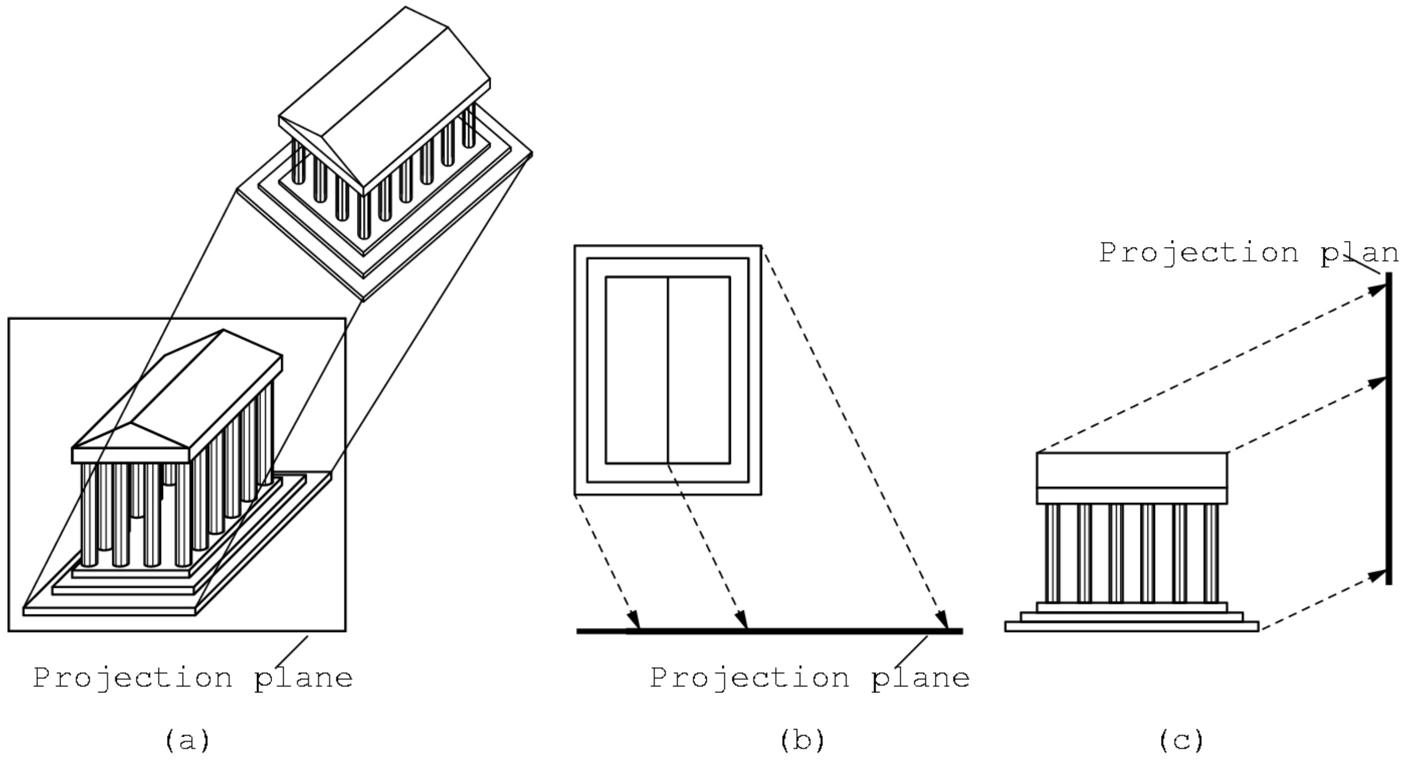


(a) (b) (c)

Example: Oblique Views

Projection at an arbitrary angle to the projection plane

- angles in planes parallel to the projection plane are preserved



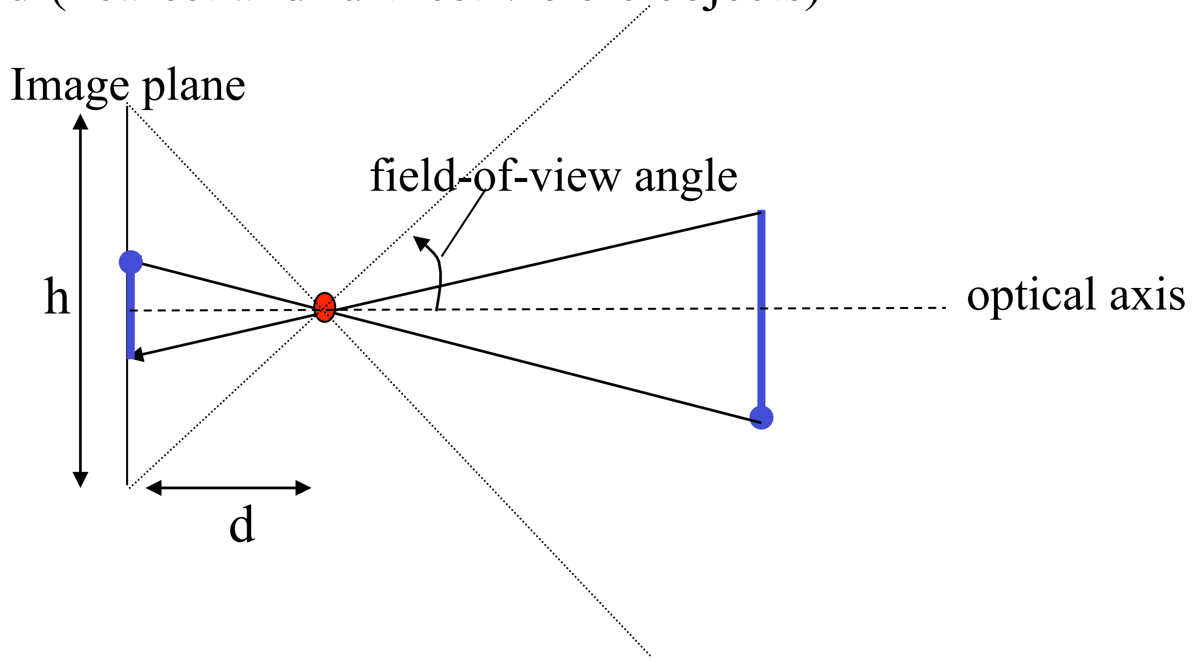
Perspective Projection

Models real camera image projection

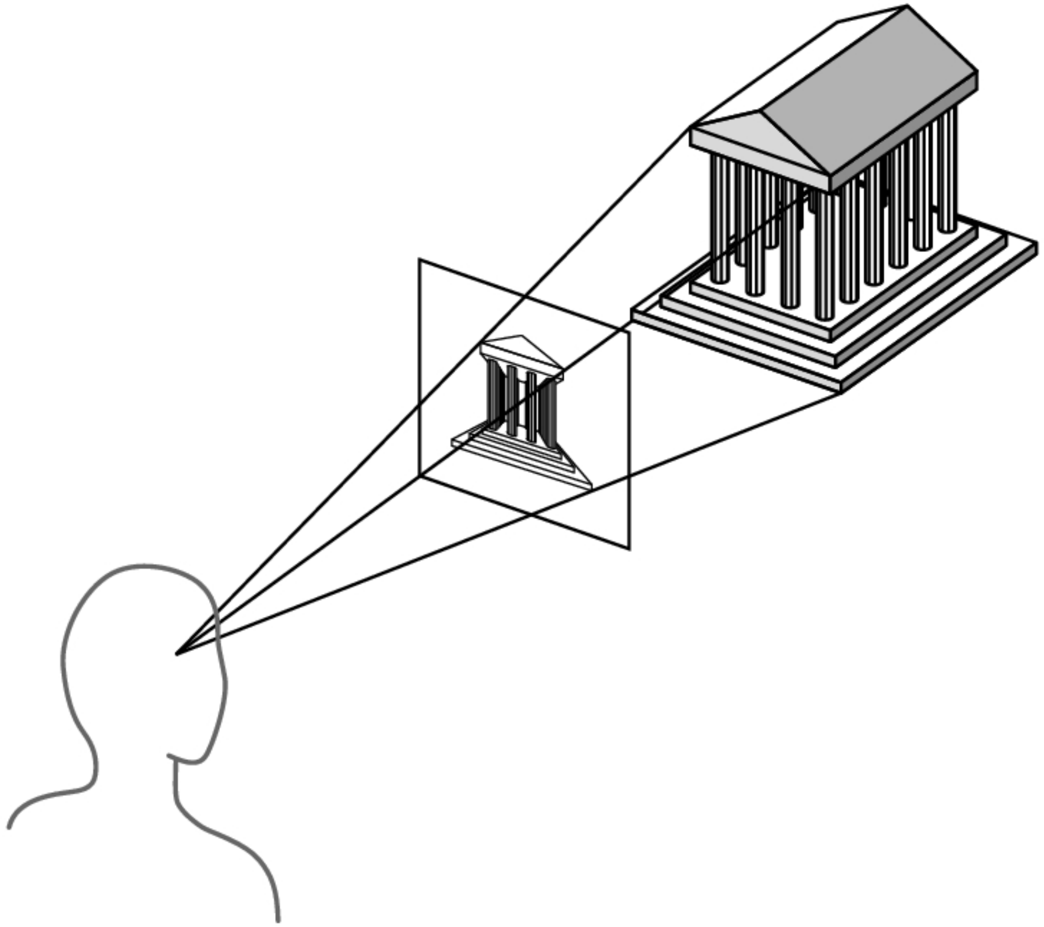
- pin-hole camera
- used in graphics to achieve real- looking images
- object size on image decreases with distance
- parallel lines converge

Projection is determined by:

- distance d of **centre-of-projection** to image plane
- height of image plane or **field-of-view** angle
- **depth-of-field** (nearest and furthest visible objects)



Example: Perspective views



Simple Perspective Projection

Assume image projection plane is orthogonal to the view direction (optical axis)

Simplify the camera model by:

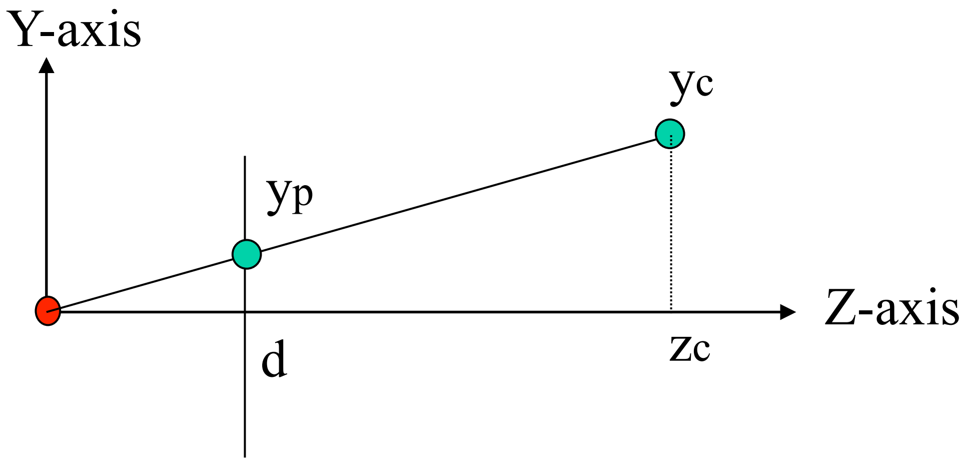
- placing the center-of-projection at the origin of the camera frame
- alligning the optical axis or view direction with the -z-axis
- place the image projection plane frame infront of the camera

Projection gives:

$$\frac{y_c}{z_c} = \frac{y_p}{d}$$

$$y_p = \frac{y_c d}{z_c}$$

& similarly for x_p
with $z_p = d$



Non-linear equations due to division by z_c

Homogeneous Transform for Perspective Projection

What is the 4x4 Homogeneous transform matrix for perspective projection?

In Homogeneous coordinates a point $x_c = [x_c, y_c, z_c, 1]$

This represents a line in space for $0 < w$ from the center-of-projection (origin)

For a camera at the origin aligned with the z-axis
the simple perspective projection matrix is given by:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

$$x_p = Cx_c$$

$$x_p = [x_c, y_c, z_c, \frac{z_c}{d}]$$

In real-coordinates:

$$x_p = \frac{x_c d}{z_c}, y_p = \frac{y_c d}{z_c}, z_p = d$$

General Perspective Projection

We have derived the perspective projection matrix C for the simple case of a camera at the origin aligned with the z-axis.

How can we perform perspective projection for an arbitrary camera position & orientation?

- transform the world frame into the camera frame such that the center-of-projection is at the origin & view direction on the -z axis

Camera frame has 6 degrees-of-freedom (dof) specified by

(1) position of centre-of-projection	$c = [cx, cy, cz, 1]$	(3dof)
(2) image projection plane unit normal	$n = [nx, ny, nz, 0]$	(2dof)
image projection plane unit up vector	$v = [vx, vy, vz, 0]$	(1dof)

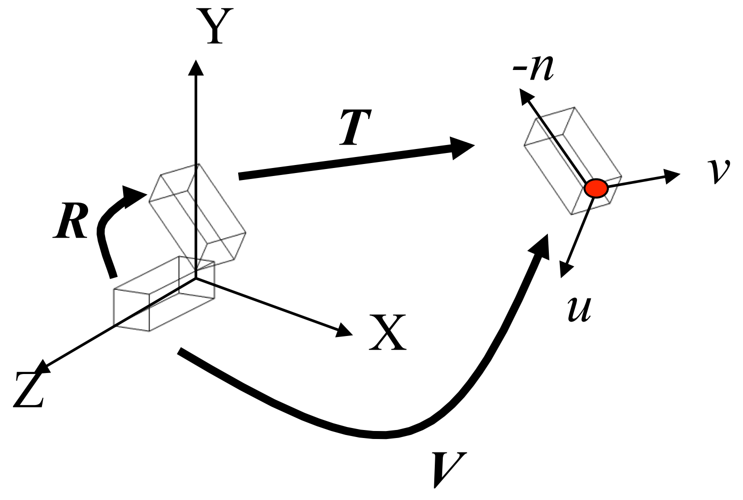
Now we want to construct the transform V which moves the camera to the new position and orientation

The transform is composed of a rotation about the origin to the correct orientation followed by a translation:

$$V = TR$$

Camera Positioning

Camera Transform *V=TR*



The camera translation is

$$T = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The orientation of the camera is defined by unit vectors *n* and *v* which are orthogonal to each other: $n.v = 0$

$$u = n \times v$$

This gives a matrix M which orients a vector in the *(u,v,n)* basis with respect to the world frame *(X,Y,Z)* basis:

$$M = \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Camera Positioning

The rotation matrix R orients a vector in (X,Y,Z) basis with respect to the (u,v,n) basis:

$$R = M^{-1} = M^{T} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying by the translation T we obtain the camera transform:

$$V = TR = \begin{bmatrix} u_x & u_y & u_z & -c_x \\ v_x & v_y & v_z & -c_y \\ n_x & n_y & n_z & -c_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Therefore, we can transform points in world frame to camera frame for an arbitrary camera position and orientation by:

$$x_c = Vx_w$$

where x_c and x_w are in Homogeneous co-ordinates

Projection Pipeline

We can transform any point x_w in the world frame to the perspective projection in any camera view using a concatenation of Homogeneous transforms:

$$\begin{tikzcd} {} \arrow[r, "x_w"] & \boxed{\begin{array}{c}\text{Model-view}\\ V\end{array}} \arrow[r, "x_c"] & \boxed{\begin{array}{c}\text{Projection}\\ \text{C}\end{array}} \arrow[r, "x_p"] & \boxed{\begin{array}{c}\text{Perspective}\\ \text{Division}\end{array}} \arrow[r, "\begin{array}{c}\text{Image}\\ \text{Frame}\end{array}"] & {} \end{tikzcd}$$

(1) Transform point from world to camera coordinates using 4x4 model-view matrix V in camera coordinates

$$x_c = Vx_w$$

(2) Apply camera projection by 4x4 projection matrix C in camera coordinates

$$x_p = Cx_c = CV\,x_w$$

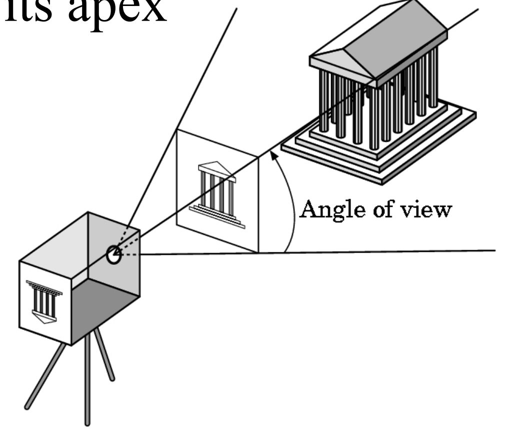
(3) Transform Homogeneous *(x,y,z,w)* to real-coordinates *(x/w,y/w,z/w)* giving:

$$x_p = \frac{x_c d}{z_c}, y_p = \frac{y_c d}{z_c}, z_p = d$$

View-volume

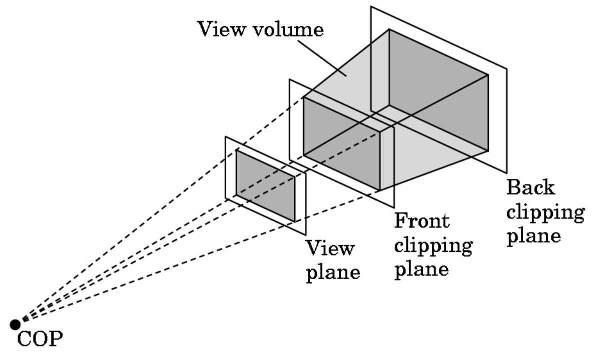
Projection allows us to transform points from the world frame to a projective camera frame but does not account for camera properties.

- **field-of-view** angle defines the **view-volume**
  only objects inside the view-volume are seen by the camera
- **view-volume** is a semi-infinite pyramid with its apex
  at the centre-of-projection
  Objects not within the view-volume are
  clipped from the scene



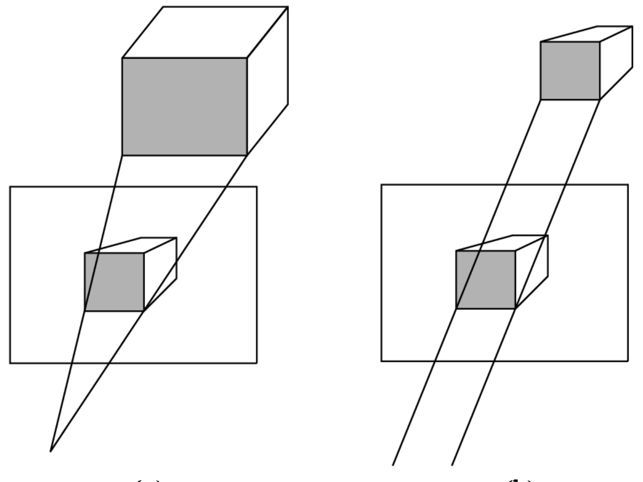
most graphics API's define a finite view volume by a front and back plane

- **view frustum** is a truncated pyramid
  bounded by the view-volume and front/back planes
  Objects not within the frustum are clipped



Projection Normalisation

Projection normalisation converts all projections into orthogonal projections by distorting the objects such that the orthogonal projection of the distorted object is the same as the desired projection of the original object



Object distortion can be described by a Homogeneous transformation matrix
Therefore, we can implement projective normalisation by concatenating the distortion matrix with the projection

Projective normalisation is used to transform the camera view-volume to a standard **canonical view-volume** for clipping.

Orthographic Projection Normalisation

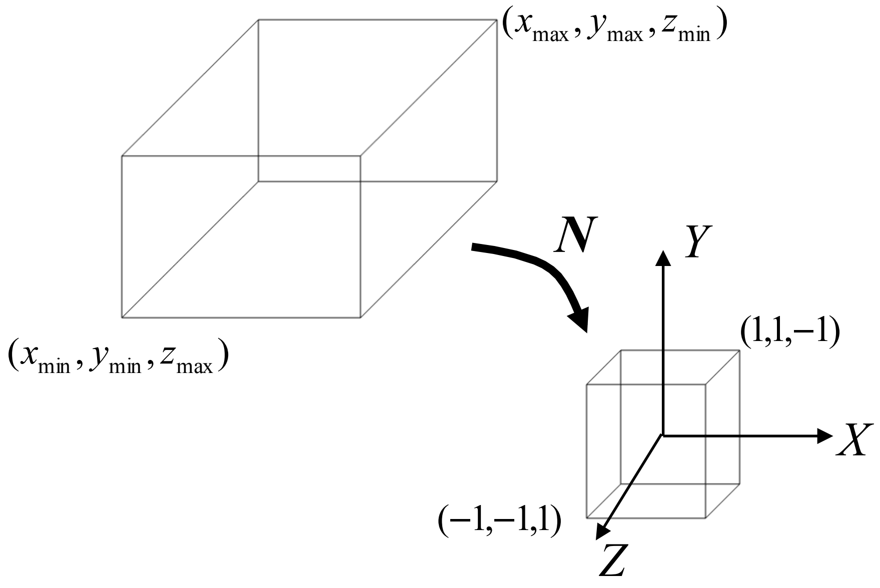
Two stages:

(1) Transform the view-volume for an arbitrary orthographic camera to a standard view volume

(2) Apply an orthographic projection on the transformed objects

For an orthographic camera with view direction aligned with the z-axis the view-volume is bounded by the parallelepiped enclosing such that:

$$x_{\min} \le x \le x_{\max}$$
$$y_{\min} \le y \le y_{\max}$$
$$z_{\min} \le z \le z_{\max}$$



$(x_{\max}, y_{\max}, z_{\min})$
$(x_{\min}, y_{\min}, z_{\max})$
N
Y
$(1,1,-1)$
X
$(-1,-1,1)$
Z

The simplest clipping volume to deal with is a cube center at the origin such that:

$$-1 \le x \le 1$$
$$-1 \le y \le 1$$
$$-1 \le z \le 1$$

- **canonical view volume** in OpenGL

How do we transform the arbitrary view volume to the canonical?

Orthographic Projection Normalisation II

Transform from arbitrary to canonical view volume by: N=ST

(i) Translation of the centre of the arbitrary view volume to the origin by translation

$$T = \begin{bmatrix} 1 & 0 & 0 & -(x_{\max} + x_{\min})/2 \\ 0 & 1 & 0 & -(y_{\max} + y_{\min})/2 \\ 0 & 0 & 1 & -(z_{\max} + z_{\min})/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(ii) Scaling the arbitrary view volume to the canonical

$$S = \begin{bmatrix} 2/(x_{\max} - x_{\min}) & 0 & 0 & 0 \\ 0 & 2/(y_{\max} - y_{\min}) & 0 & 0 \\ 0 & 0 & 2/(z_{\max} - z_{\min}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

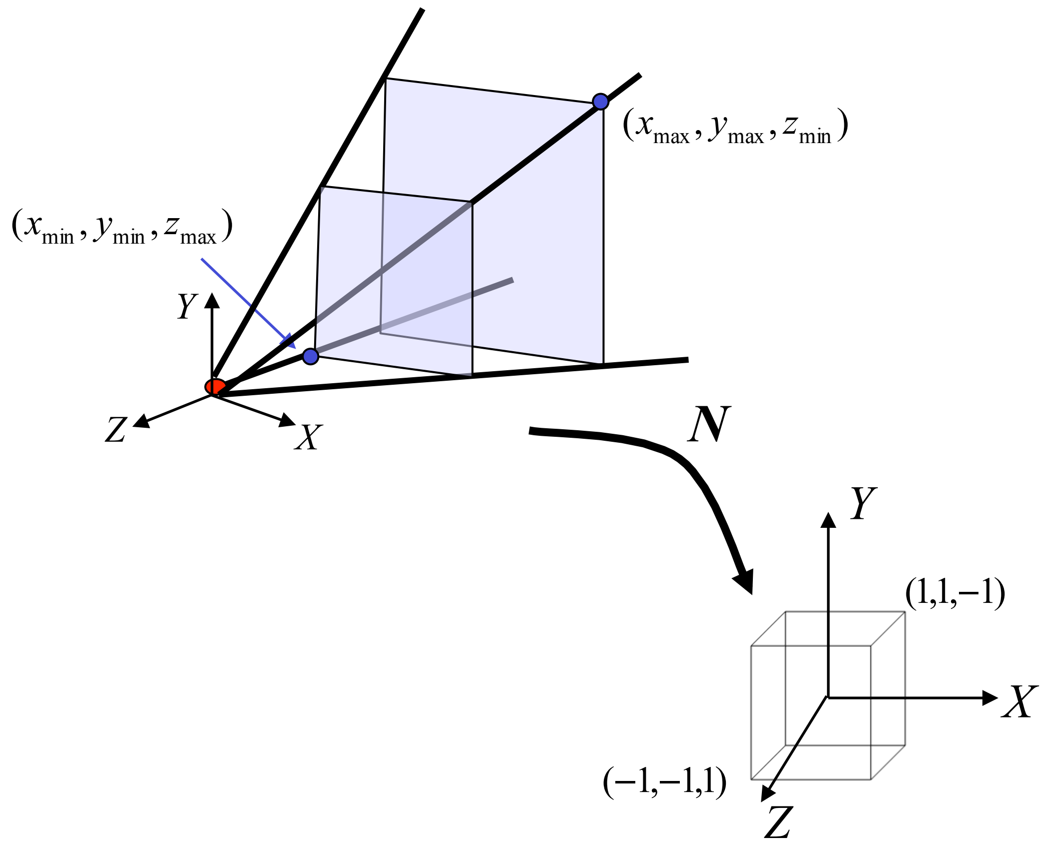
The resulting normalisation transform for orthographic projection is:

$$N = ST = \begin{bmatrix} \frac{2}{(x_{\max} - x_{\min})} & 0 & 0 & -\frac{(x_{\max} + x_{\min})}{(x_{\max} - x_{\min})} \\ 0 & \frac{2}{(y_{\max} - y_{\min})} & 0 & -\frac{(y_{\max} + y_{\min})}{(y_{\max} - y_{\min})} \\ 0 & 0 & \frac{2}{(z_{\max} - z_{\min})} & -\frac{(z_{\max} + z_{\min})}{(z_{\max} - z_{\min})} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This is a non-singular transformation which can be used to normalise points to the canonical volume for clipping/hidden-surface removal/shading.

Perspective Normalisation

We want to find a transformation which normalises the view frustum for a perspective projection to the canonical view frustum



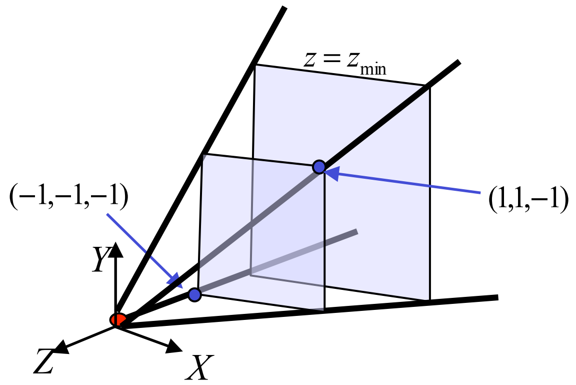
$(x_{\max}, y_{\max}, z_{\min})$
$(x_{\min}, y_{\min}, z_{\max})$
Y
Z
X
N
Y
$(1,1,-1)$
X
$(-1,-1,1)$
Z

Simple Perspective Normalisation

A simple perspective projection with the projection plane at -d on the z-axis can be represented as:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

$$x_p = Cx_c$$



Setting d=-1 gives the semi-infinite view-volume bounded by the planes: $x_c = \pm z_c, y_c = \pm z_c$

We define a finite view frustum by: $z_{\min} < z_c < z_{\max}$

Consider the transformation N which transforms point (x,y,z,1) to (x’,y’,z’,w):

$$N = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \alpha & \beta \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$$x' = x, y' = y$$
$$z' = \alpha z + \beta$$
$$w' = -z$$

In real co-ordinates:

$$x' = -x/z, y' = -y/z$$
$$z' = -(\alpha + \beta/z)$$

Simple Perspective Normalisation II

If we apply an orthographic projection to N we obtain:

$$C_{orth}N = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \alpha & \beta \\ 0 & 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

which results in the projection: $x' = x, y' = y, z' = 0, w' = -z$

in real-coordinates this is: $x' = -x/z, y' = -y/z, z' = 0$

which is a perspective projection of the point for an image plane at *z=-1*

Thus we have achieved a perspective projection by combining the matrix *N* with an orthographic projection.

N is distorting the world space

How can we use *N* to normalise the projective view frustum to the canonical view frustum?

Simple Perspective Normalisation III

Matrix N is non-singular and transforms the original view-volume into a new view volume.
Choose α and β such that the new volume is the canonical clipping frustum

The sides of the projective view-volume after normalisation N are transformed to be the same as the canonical view-volume:

$$x_c = \pm z_c, y_c = \pm z_c$$

$$x' = Nx_c \Rightarrow x' = \pm 1, y' = \pm 1$$

The front and back are transformed to the planes:

$$z_c = z_{\min} \Rightarrow z' = -\left(\alpha + \frac{\beta}{z_{\min}}\right)$$

$$z_c = z_{\max} \Rightarrow z' = -\left(\alpha + \frac{\beta}{z_{\max}}\right)$$

Therefore, setting α and β we obtain the canonical view frustum:

$$\alpha = \frac{(z_{\max} + z_{\min})}{(z_{\max} - z_{\min})}, \beta = -\frac{2z_{\max}z_{\min}}{(z_{\max} - z_{\min})} \Rightarrow z' = \pm 1$$

Thus, we have transformed from a simple projective to canonical view-frustum

- This is a non-linear mapping which preserves the depth ordering

General Perspective Normalisation

In the simple perspective normalisation we assumed that the projection plane was at *zc=-1*

We can generalise this to an arbitrary perspective projection

(1) Convert the view-frustum to the simple symmetric projection frustum

(2) Transform the simple frustum to the canonical view frustum

Conversion of an arbitrary asymmetric view frustum to the simple frustum:

(i) Shear the frustum to obtain a symmetric frustum

$$\left(\frac{x_{\max}+x_{\min}}{2}, \frac{y_{\max}+y_{\min}}{2}, z_{\min}\right) \Rightarrow (0,0,z_{\min})$$

$$H(\cot\theta,\cot\phi) = H\left(\frac{x_{\max}+x_{\min}}{2z_{\min}}, \frac{y_{\max}+y_{\min}}{2z_{\min}}\right)$$

$$\Rightarrow x = \pm\left(\frac{x_{\max}-x_{\min}}{2z_{\min}}\right), y = \pm\left(\frac{y_{\max}-y_{\min}}{2z_{\min}}\right), z = z_{\max}, z = z_{\min}$$

$$H = \begin{bmatrix} 1 & 0 & \frac{(x_{\min}+x_{\max})}{2z_{\min}} & 0 \\ 0 & 1 & \frac{(y_{\min}+y_{\max})}{2z_{\min}} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(ii) Scale the sides of the frustum

$$S\left(\frac{2z_{\min}}{x_{\max}-x_{\min}}, \frac{2z_{\min}}{y_{\max}-y_{\min}}, 1\right)$$

$$\Rightarrow x = \pm z, y = \pm z$$

$$S = \begin{bmatrix} \frac{2z_{\min}}{(x_{\max}-x_{\min})} & 0 & 0 & 0 \\ 0 & \frac{2z_{\min}}{(y_{\max}-y_{\min})} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

General Perspective Normalisation II

We can then concatenate the transform with the transformation of the simple view frustum to transform an arbitrary projective frustum to the canonical view frustum

$$P = NSH = \begin{bmatrix} \frac{2z_{\min}}{(x_{\max} - x_{\min})} & 0 & \frac{(x_{\max} + x_{\min})}{(x_{\max} - x_{\min})} & 0 \\ 0 & \frac{2z_{\min}}{(y_{\max} - y_{\min})} & \frac{(y_{\max} + y_{\min})}{(y_{\max} - y_{\min})} & 0 \\ 0 & 0 & -\frac{(z_{\max} + z_{\min})}{(z_{\max} - z_{\min})} & -\frac{2z_{\max} z_{\min}}{(z_{\max} - z_{\min})} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

This can then be orthographically projected to obtain the original perspective projection

Hence, we can model any perspective projection as a transform to the canonic view frustum followed by an orthographic projection

The canonic view frustrum is important for implementing visibility, shadowing

Summary

- Two projection models: Orthographic (parallel), Perspective (pin-hole)

Orthographic $$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$x_p = Cx_c$$

Simple Perspective camera at origin view along z-axis $$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

$$x_p = Cx_c$$

$$x_p = [x_c, y_c, z_c, \frac{z_c}{d}]$$

- Any transform implemented by concatenation of transforms
  (i) Transform model to camera coordinates ‘ModelView’
  (ii) Apply perspective projection

- Camera FOV transformed into canonic FOV (unit cube) for clipping

$$P = NSH = \begin{bmatrix} \frac{2z_{\min}}{(x_{\max} - x_{\min})} & 0 & \frac{(x_{\max} + x_{\min})}{(x_{\max} - x_{\min})} & 0 \\ 0 & \frac{2z_{\min}}{(y_{\max} - y_{\min})} & \frac{(y_{\max} + y_{\min})}{(y_{\max} - y_{\min})} & 0 \\ 0 & 0 & -\frac{(z_{\max} + z_{\min})}{(z_{\max} - z_{\min})} & -\frac{2z_{\max}z_{\min}}{(z_{\max} - z_{\min})} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$