ΗΥ416
ΓΡΑΦΙΚΑ ΥΠΟΛΟΓΙΣΤΩΝ

Προβολές

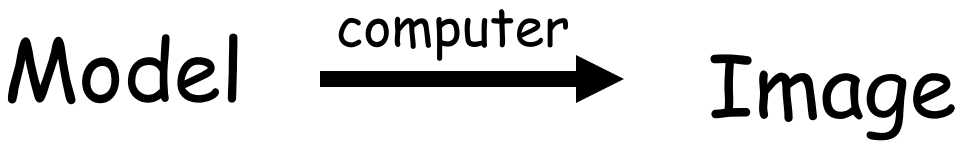
Π. ΤΣΟΜΠΑΝΟΠΟΥΛΟΥ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Projection

Projection

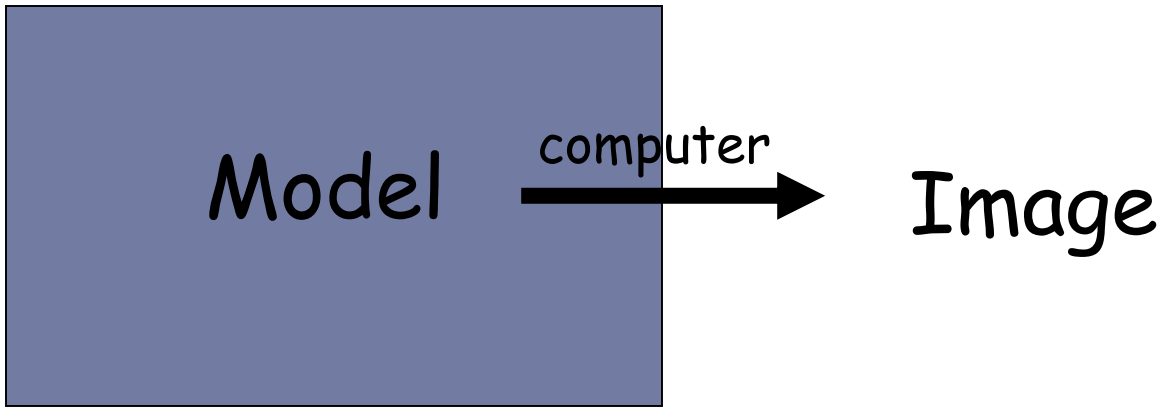
Quick Recap

- Computer Graphics is using a computer to generate an image from a representation.



Modeling

- What we have been studying so far is the mathematics behind the creation and manipulation of the 3D representation of the object.



What have we seen so far?

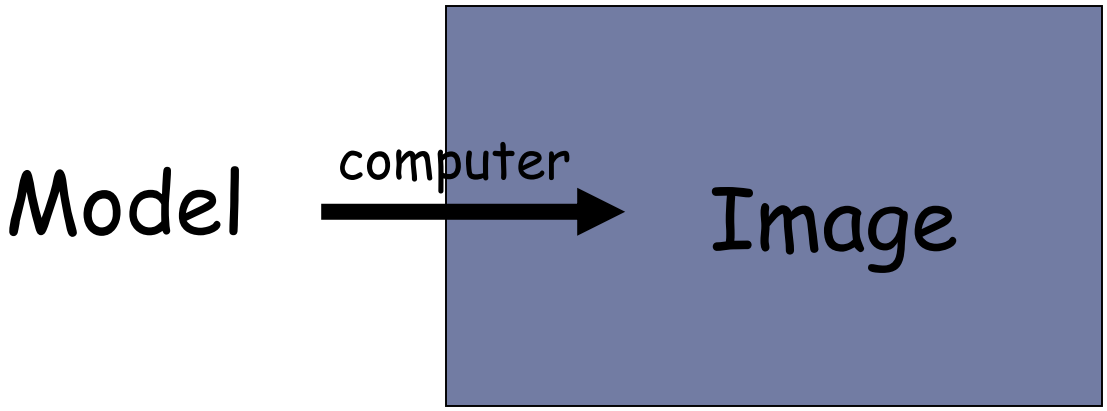
- Basic representations (point, vector)
- Basic operations on points and vectors (dot product, cross products, etc.)
- Transformation – manipulative operators on the basic representation (translate, rotate, deformations) – 4x4 matrices to "encode" all these.

Why do we need this?

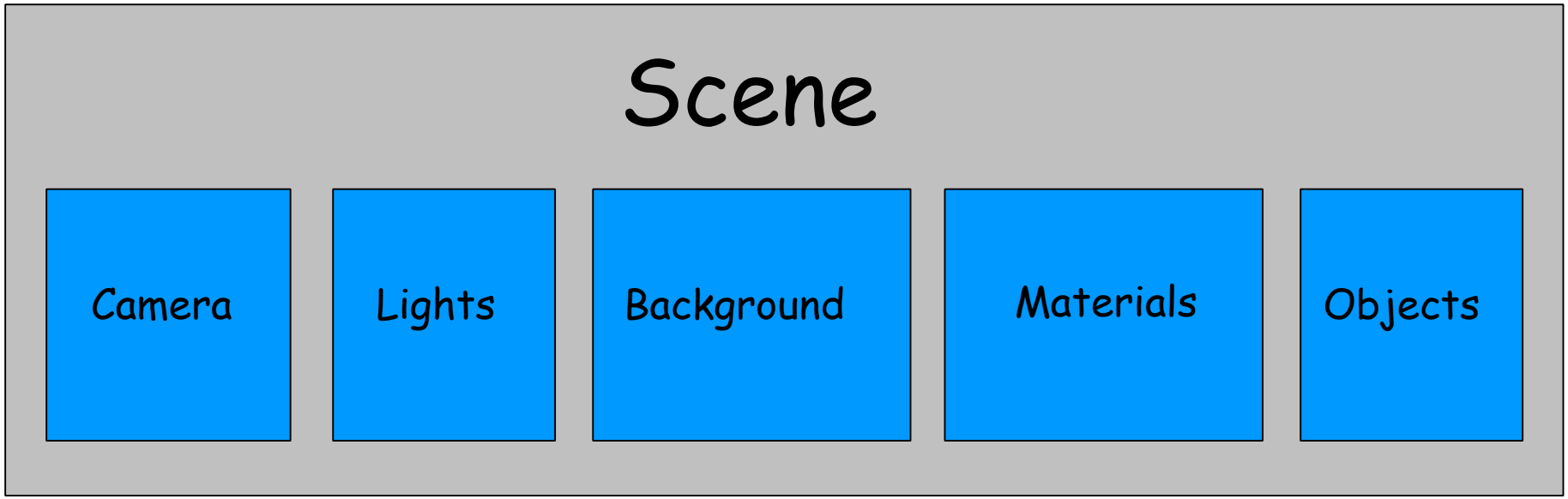
- In order to generate a picture from a model, we need to be able to not only specify a model (representation) but also manipulate the model in order to create more interesting images.

Overview

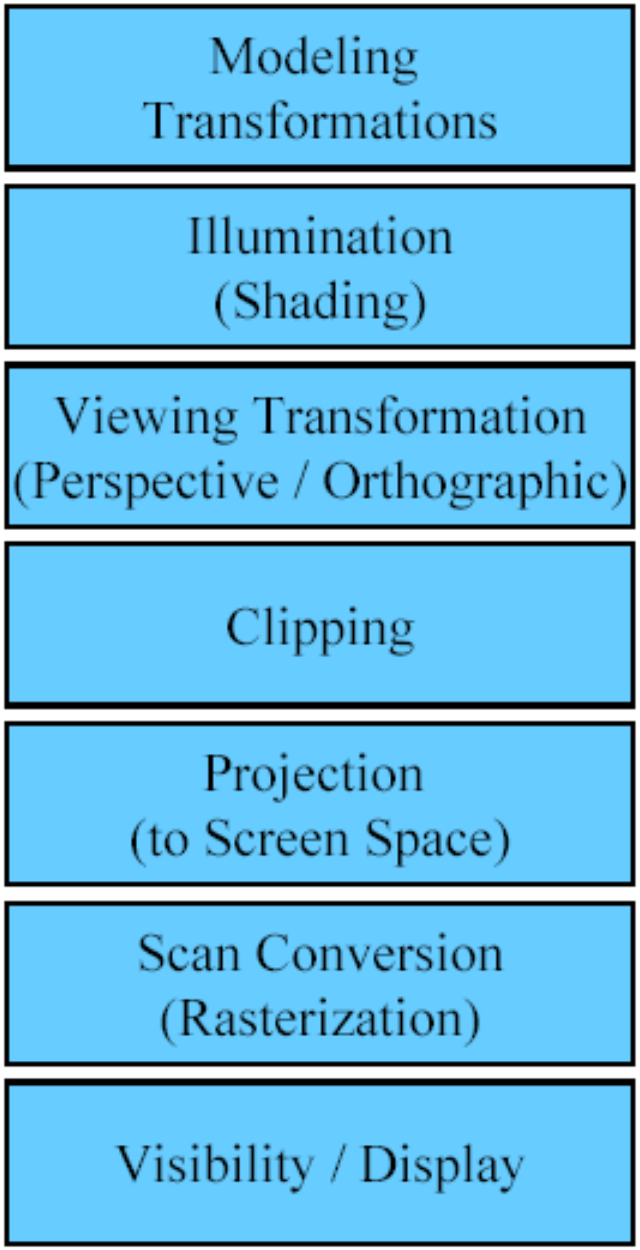
- The next set of slides will deal with the other half of the process.
- From a model, how do we generate an image?



Scene Description

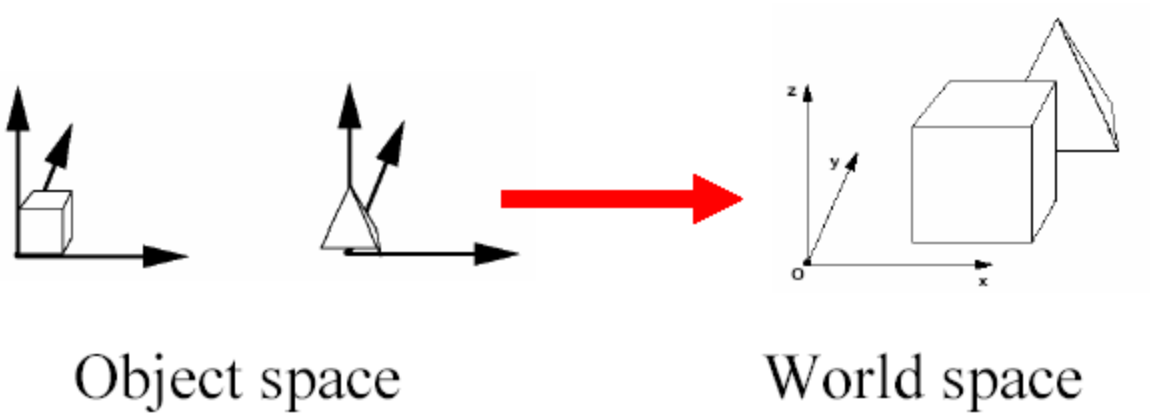


Graphics Pipeline



Graphics Pipeline

- Modeling transforms orient the models within a common coordinate frame (world space)



Modeling Transformations

Illumination (Shading)

Viewing Transformation (Perspective / Orthographic)

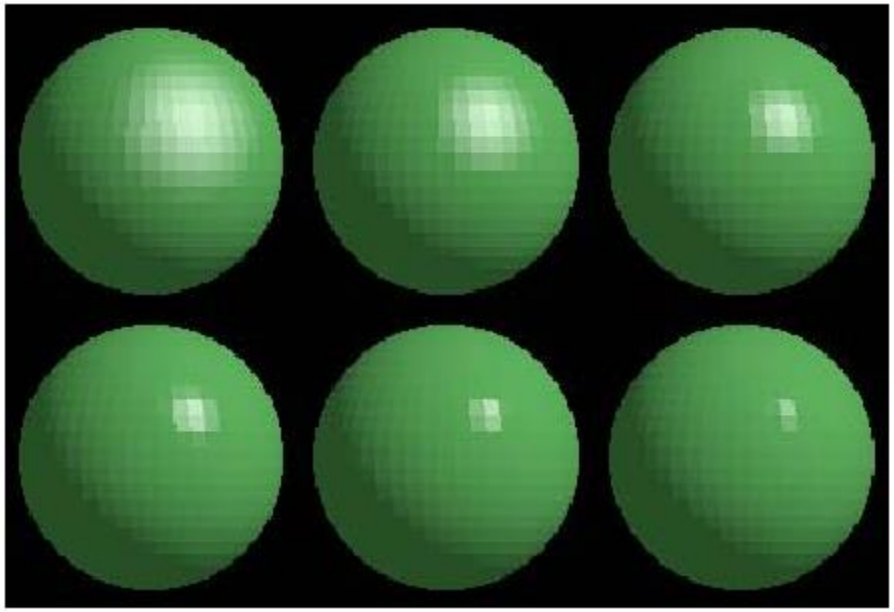
Clipping

Projection (to Screen Space)

Scan Conversion (Rasterization)

Visibility / Display

Graphics Pipeline



Modeling Transformations

Illumination (Shading)

Viewing Transformation (Perspective / Orthographic)

Clipping

Projection (to Screen Space)

Scan Conversion (Rasterization)

Visibility / Display

Graphics Pipeline

- Maps world space to eye space
- Viewing position is transformed to origin & direction is oriented along some axis (usually *z*)

Modeling Transformations

Illumination (Shading)

Viewing Transformation (Perspective / Orthographic)

Clipping

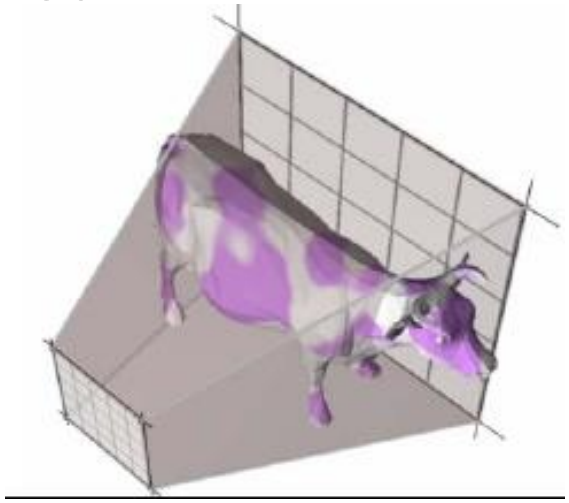
Projection (to Screen Space)

Scan Conversion (Rasterization)

Visibility / Display

Graphics Pipeline

- Transform to Normalized Device Coordinates (NDC)
- Portions of the object outside the view volume (view frustum) are removed



Modeling Transformations

Illumination (Shading)

Viewing Transformation (Perspective / Orthographic)

Clipping

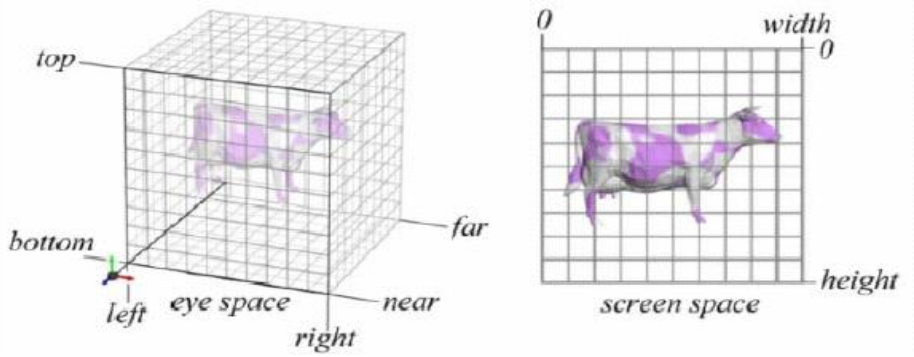
Projection (to Screen Space)

Scan Conversion (Rasterization)

Visibility / Display

Graphics Pipeline

- The objects are projected to the 2D image place (screen space)



Modeling Transformations

Illumination (Shading)

Viewing Transformation (Perspective / Orthographic)

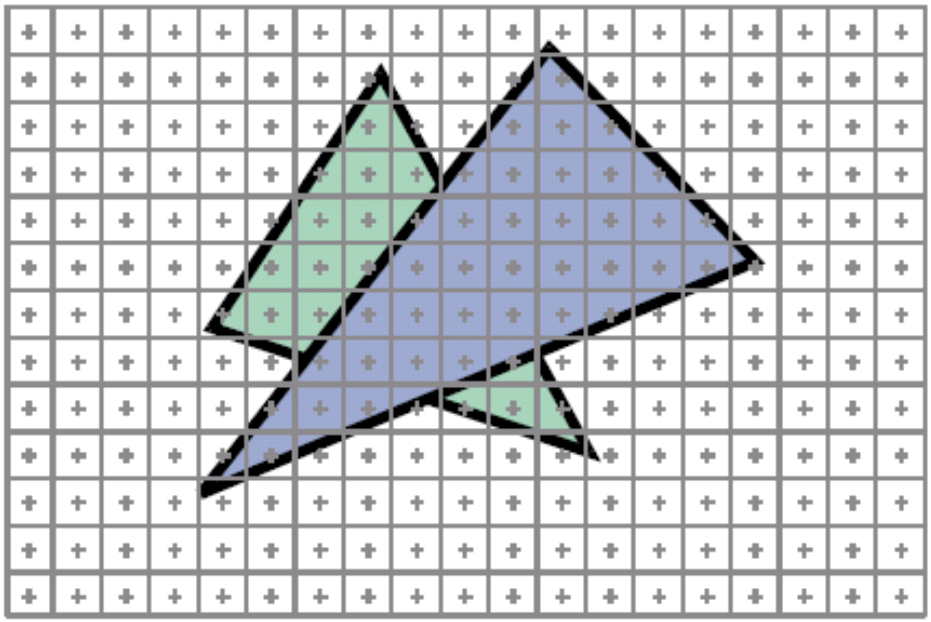
Clipping

Projection (to Screen Space)

Scan Conversion (Rasterization)

Visibility / Display

Graphics Pipeline



Modeling Transformations

Illumination (Shading)

Viewing Transformation (Perspective / Orthographic)

Clipping

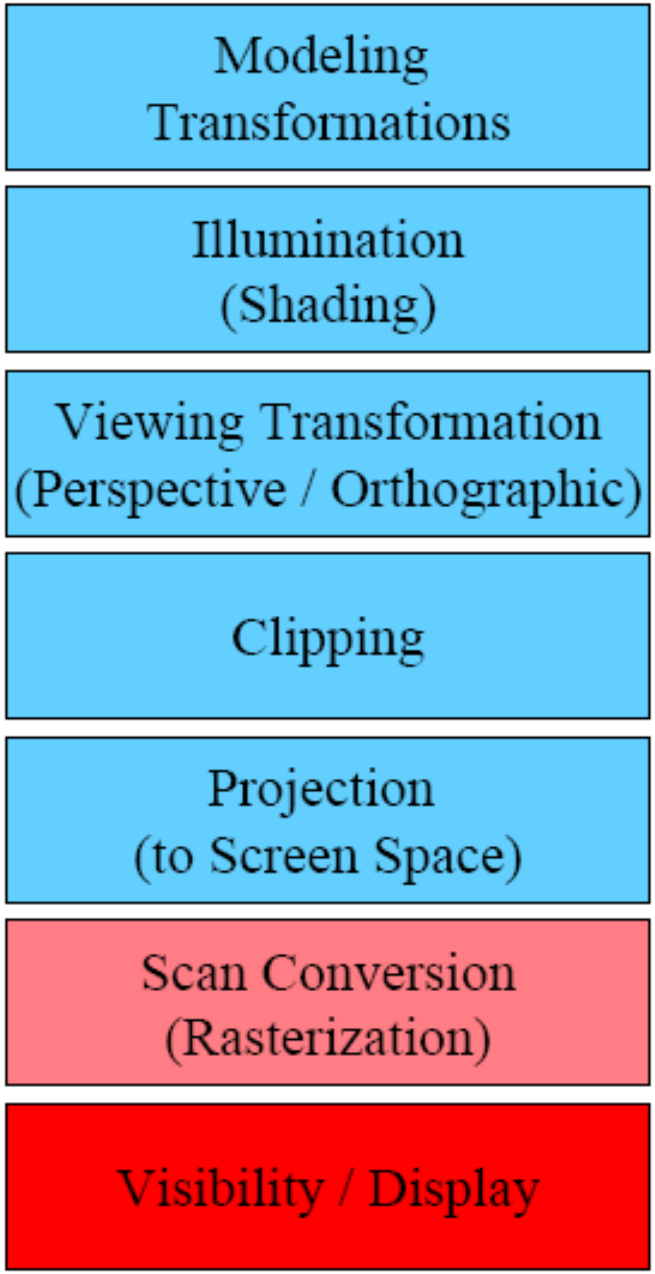
Projection (to Screen Space)

Scan Conversion (Rasterization)

Visibility / Display

Graphics Pipeline

- Z-buffer - Each pixel remembers the closest object (depth buffer)



Graphics Pipeline

- Almost every step in the graphics pipeline involves a change of coordinate system. Transformations are central to understanding 3D computer graphics.

Transformations Review

2D Coordinate transform

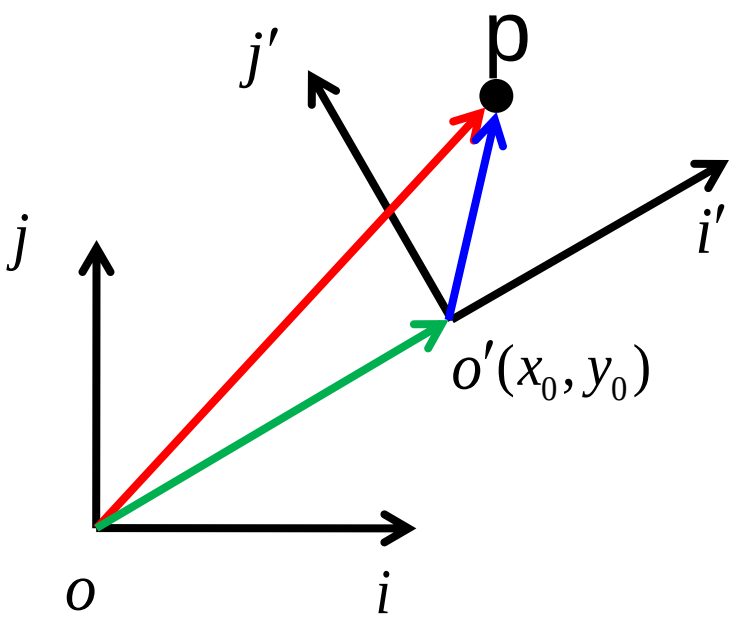
Composite transformation

3D transformation

Required readings: HB 5-9 to 5-17

Review: 2D Coordinate Trans.

- Transform object description from $i'j'$ to ij

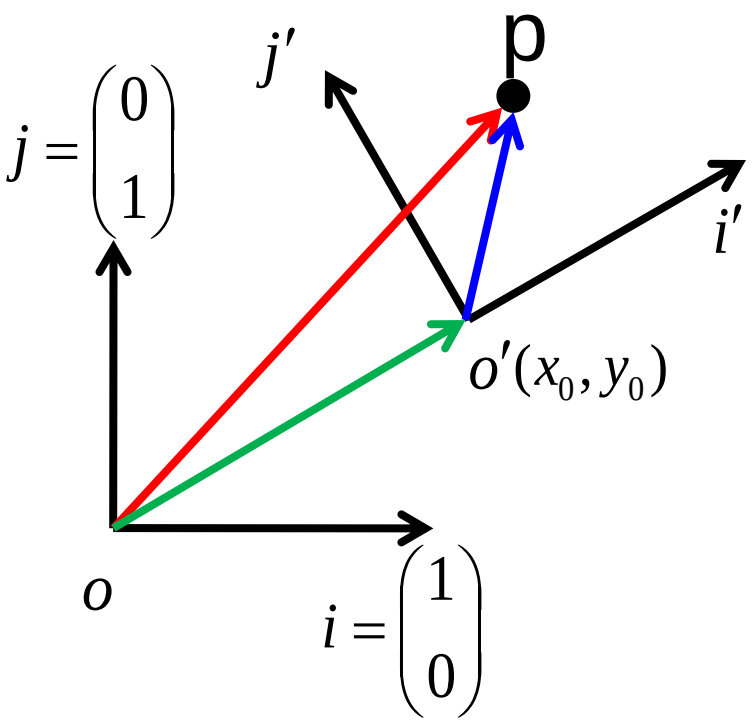


$$\begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix} = \begin{pmatrix} \vec{i}^T \\ \vec{j}^T \end{pmatrix} \begin{pmatrix} \vec{i}' & \vec{j}' \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \vec{i}^T\vec{i}' & \vec{i}^T\vec{j}' & x_0 \\ \vec{j}^T\vec{i}' & \vec{j}^T\vec{j}' & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

Review: 2D Coordinate Trans.

- Transform object description from $i'j'$ to ij

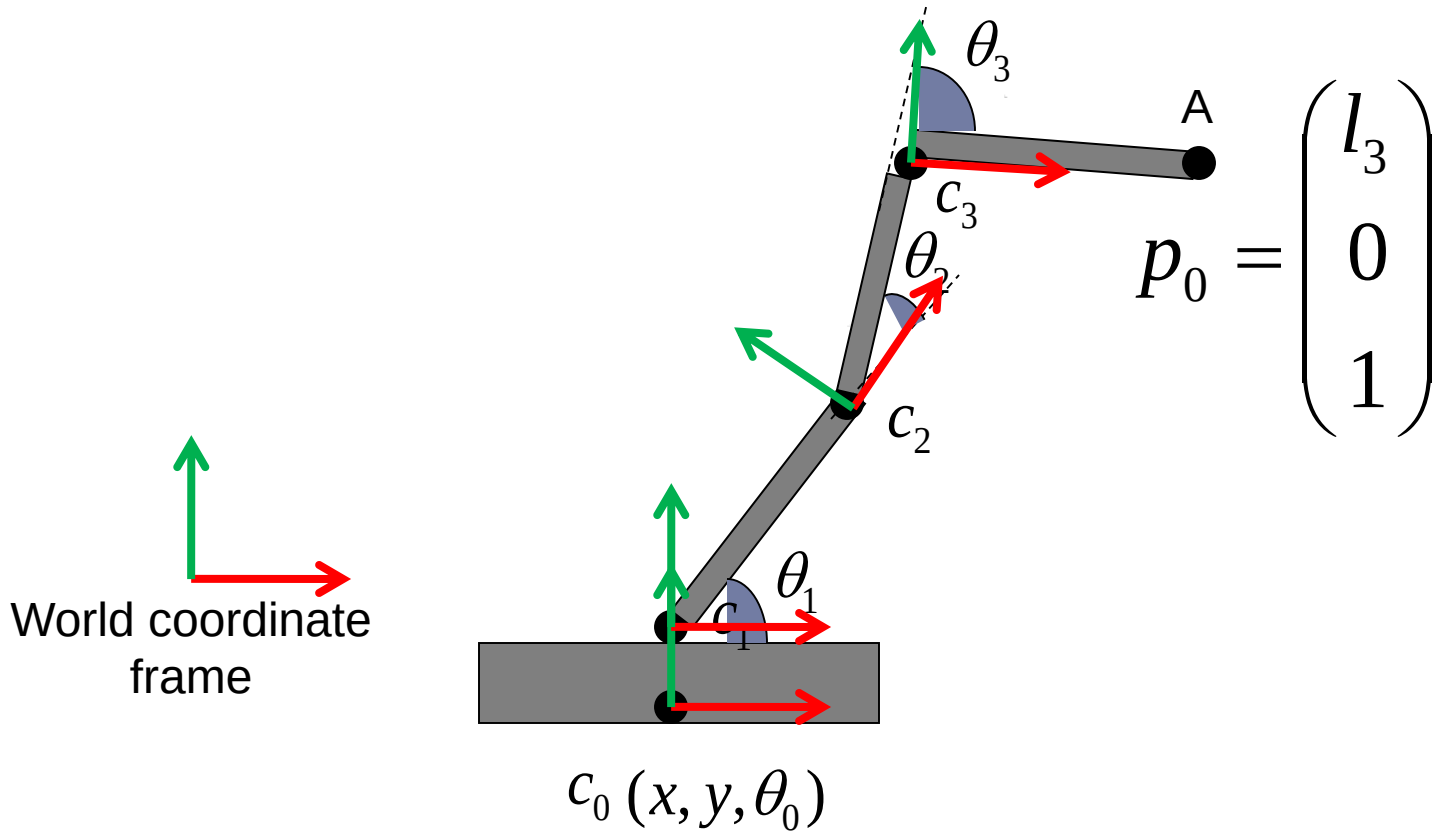


$$\begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix} = \begin{pmatrix} \vec{i}^T \\ \vec{j}^T \end{pmatrix} \begin{pmatrix} \vec{i}' & \vec{j}' \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} i'_x & j'_x & x_0 \\ i'_y & j'_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

Review: Composite 2DTrans.

- What's the world coordinate of *A* ?



$$p_0 = \begin{pmatrix} l_3 \\ 0 \\ 1 \end{pmatrix}$$

$$p = T(x,y)R(\theta_0)T(0,l_0)R(\theta_1)T(l_1,0)R(\theta_2)T(l_2,0)R(-\theta_3)\,p_0$$

Review: 3D Transformation

- Very similar to 2D transformation

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_x \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

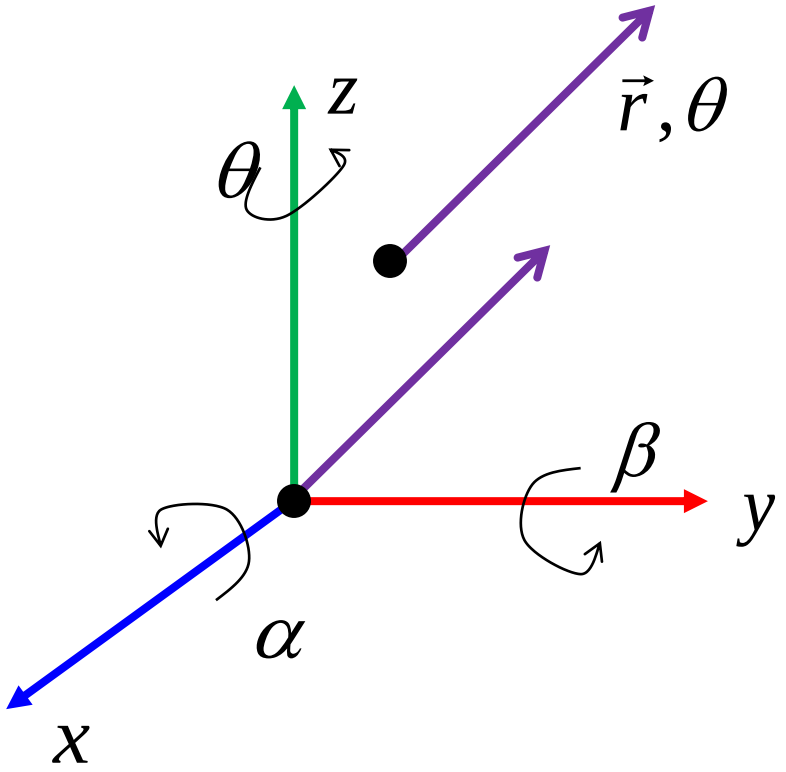
3D translation

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

3D Scaling

Review: 3D Rotation

- Transformation matrix for rotating about an arbitrary axis is more complex

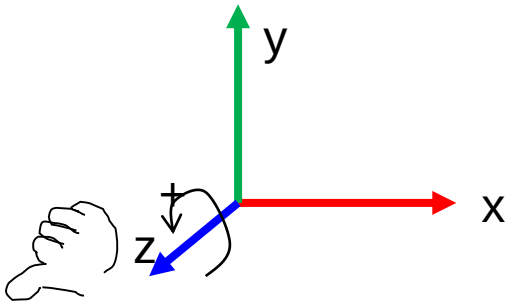


$$T^{-1}R_x(\alpha)R_y(\beta)R_z(\theta)R_y(-\beta)R_x(-\alpha)T$$

3D Transformation

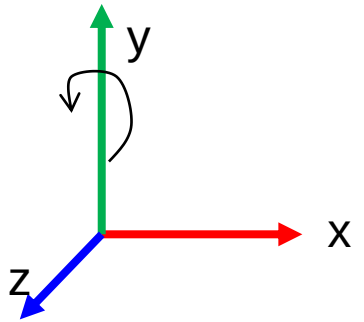
- Rotate about z

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



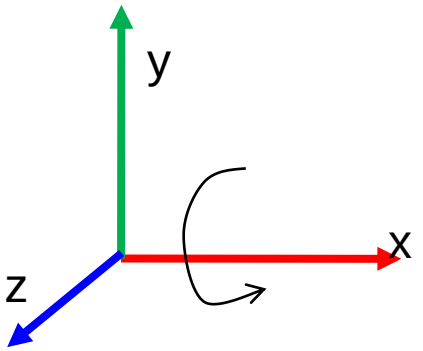
- Rotate about y

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



- Rotate about x

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma & 0 \\ 0 & \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



3D Viewing

Taking Pictures Using A Real Camera

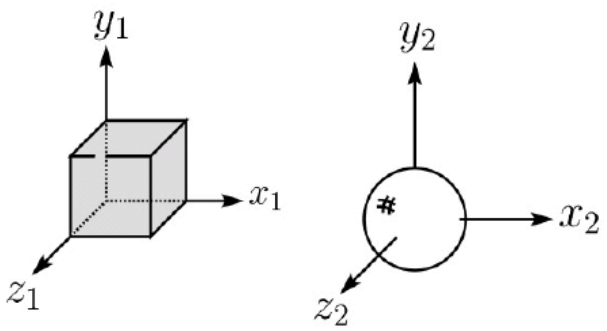
- Steps:
  - Identify interesting objects
  - Rotate and translate the camera to a desire camera viewpoint
  - Adjust camera settings such as focal length
  - Choose desired resolution and aspect ratio, etc.
  - Take a snapshot

Taking Pictures Using A Real Camera

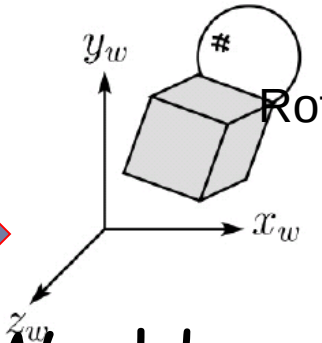
- Steps:
  - Identify interesting objects
  - Rotate and translate the camera to a desire camera viewpoint
  - Adjust camera settings such as focal length
  - Choose desired resolution and aspect ratio, etc.
  - Take a snapshot

- Graphics does the same thing for rendering an image for 3D geometric objects

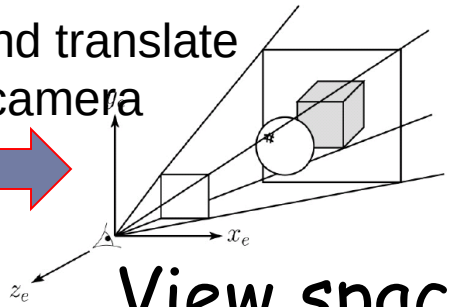
3D Geometry Pipeline



Object space



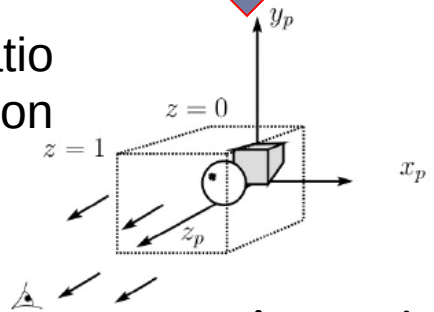
Rotate and translate the camera



World space

View space

Focal length



Aspect ratio & resolution

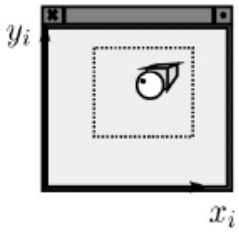
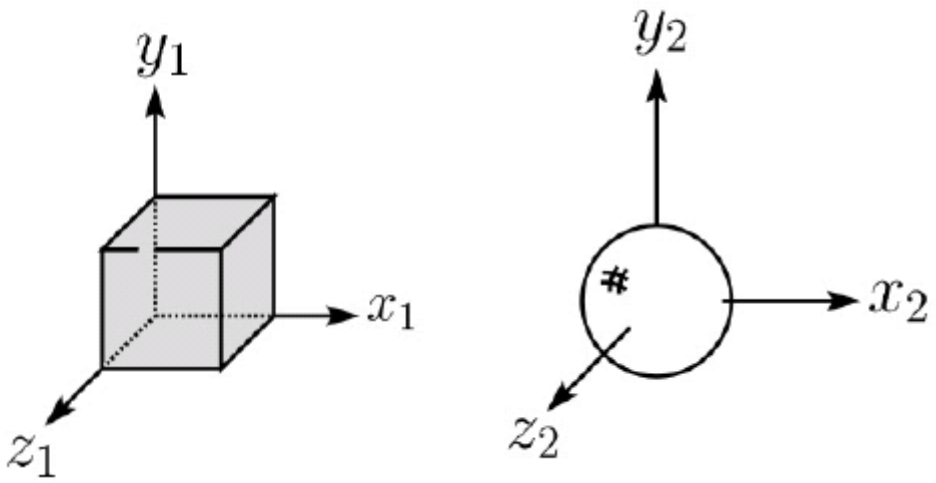


Image space

Normalized project space

3D Geometry Pipeline

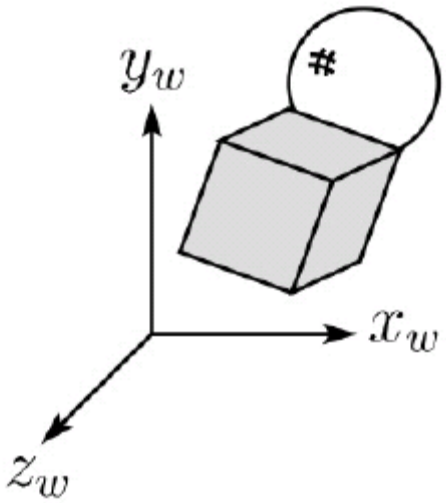
- Before being turned into pixels by graphics hardware, a piece of geometry goes through a number of transformations...



Model space
(Object space)

3D Geometry Pipeline

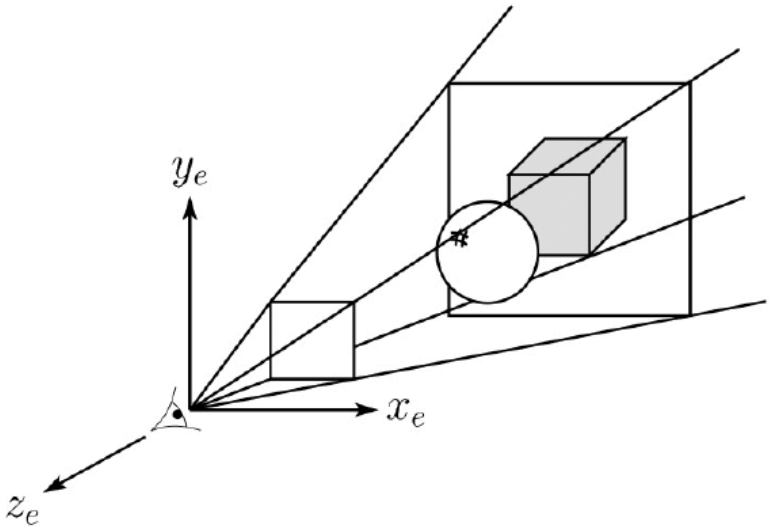
- Before being turned into pixels by graphics hardware, a piece of geometry goes through a number of transformations…



World space
(Object space)

3D Geometry Pipeline

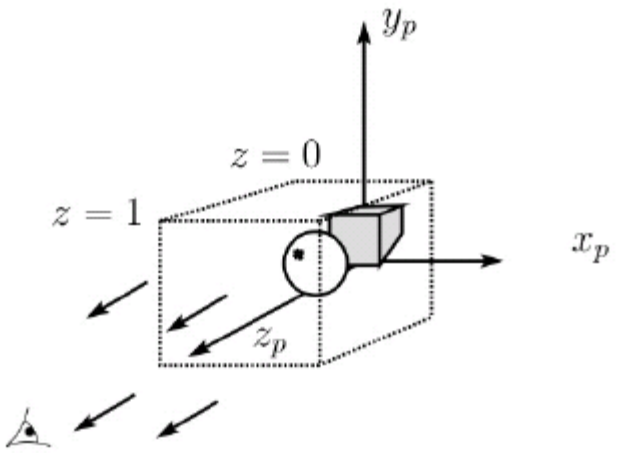
- Before being turned into pixels by graphics hardware, a piece of geometry goes through a number of transformations…



Eye space
(View space)

3D Geometry Pipeline

- Before being turned into pixels by graphics hardware, a piece of geometry goes through a number of transformations…



Normalized projection space

3D Geometry Pipeline

- Before being turned into pixels by graphics hardware, a piece of geometry goes through a number of transformations…

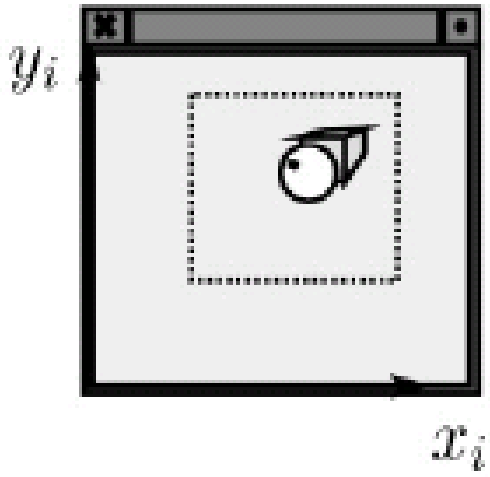
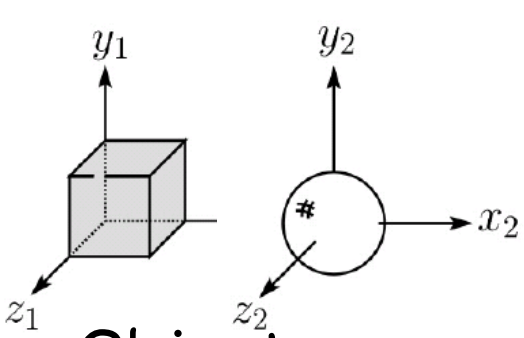
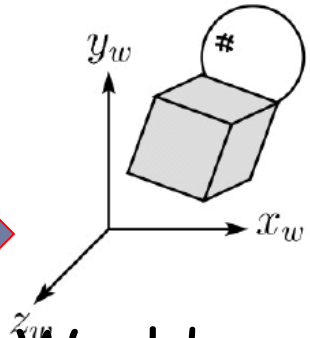


Image space, window space, raster space, screen space, device space

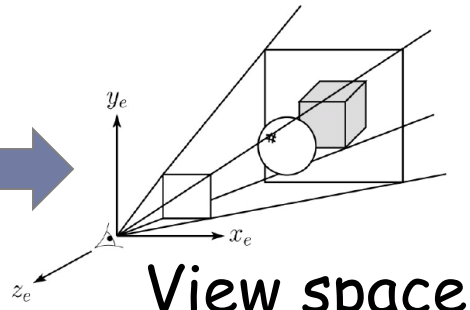
3D Geometry Pipeline



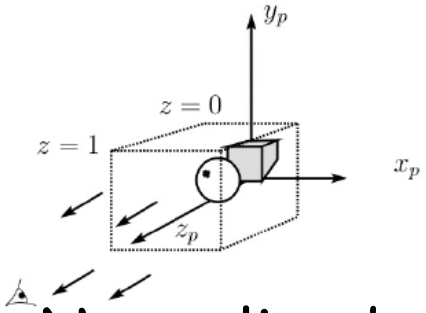
Object space



World space



View space



Normalized project space

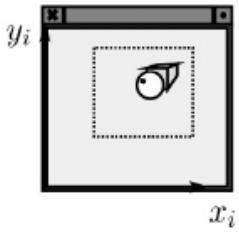
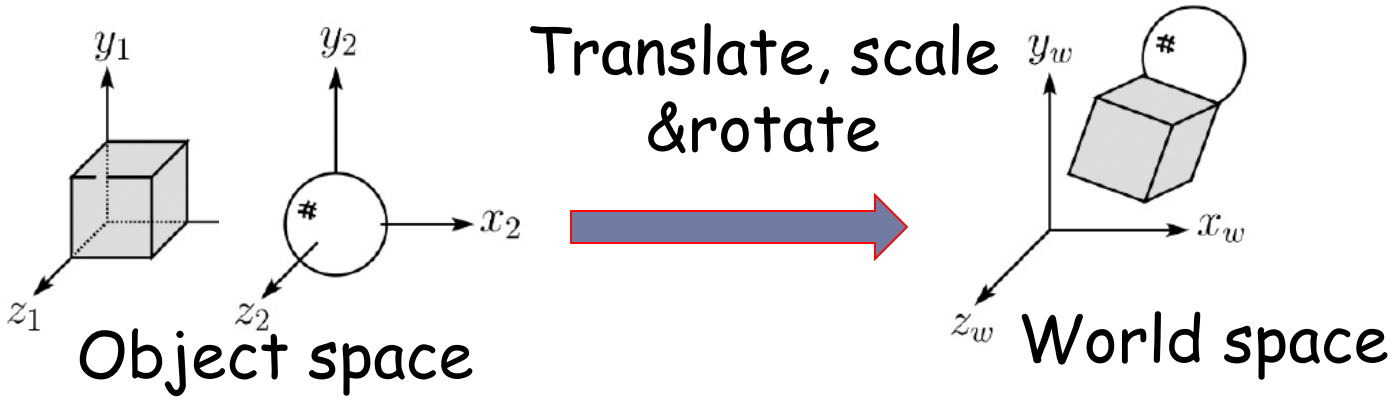


Image space

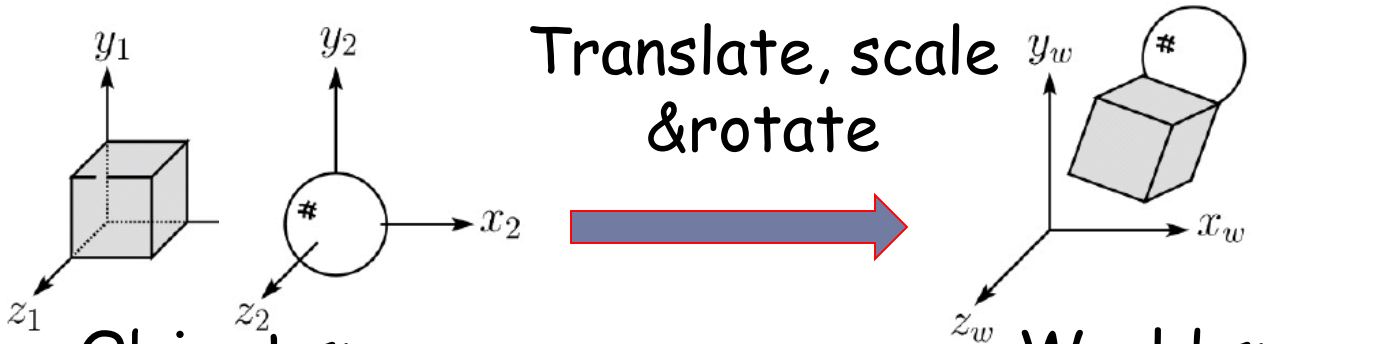
3D Geometry Pipeline



$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_x \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

glTranslate*(tx,ty,tz)

3D Geometry Pipeline



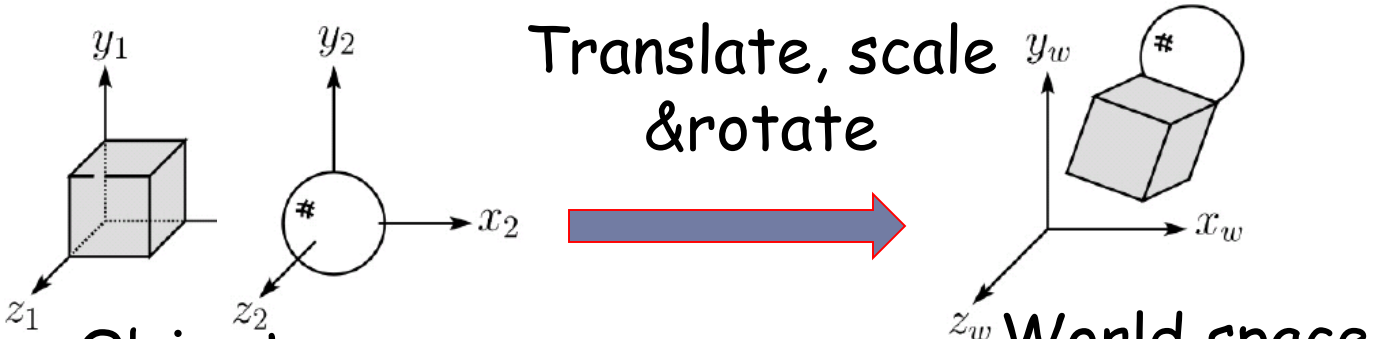
Object space

World space

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

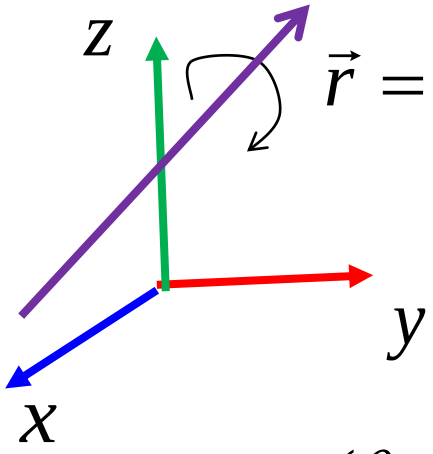
glScale*(sx,sy,sz)

3D Geometry Pipeline



Object space

World space

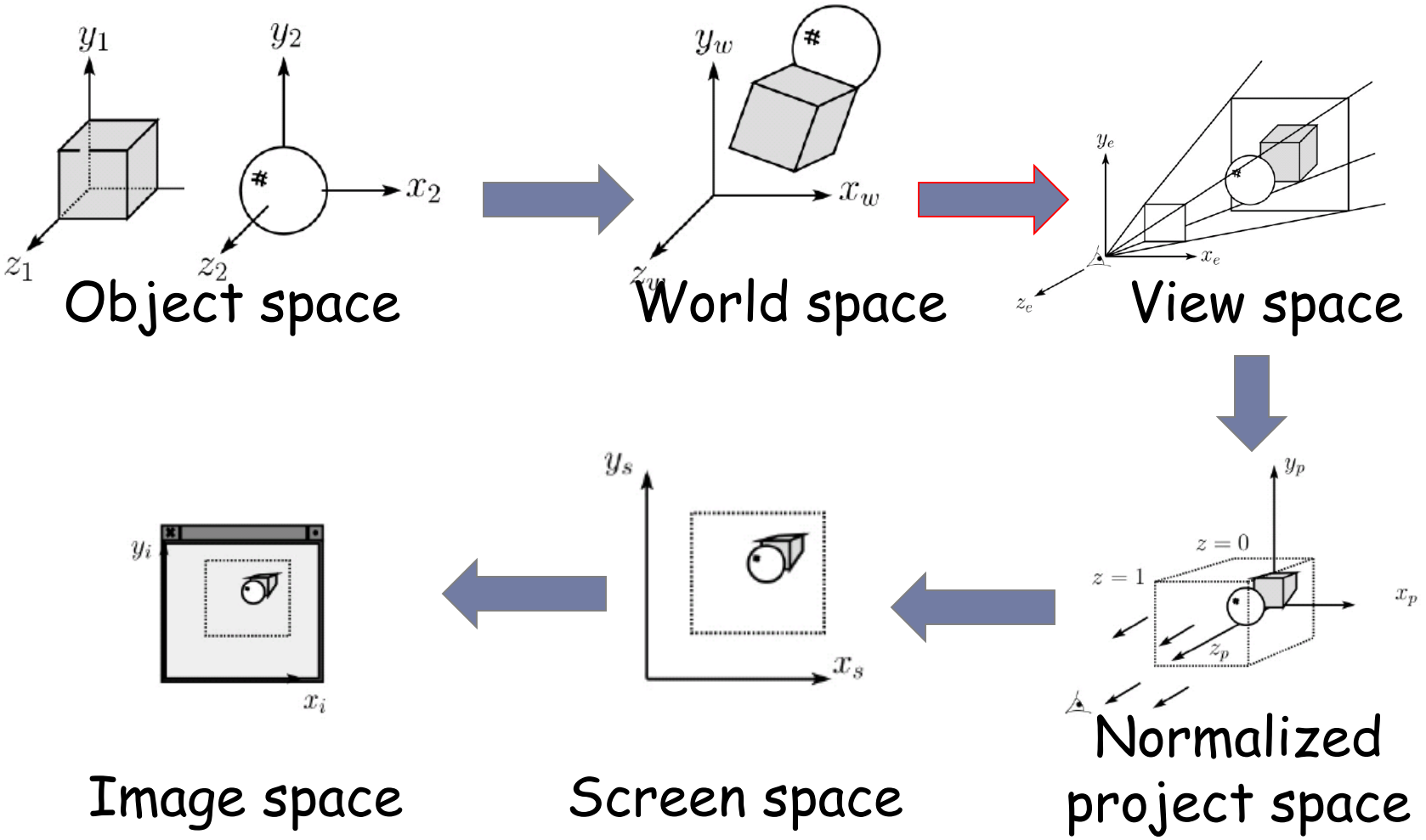


$\vec{r} = (v_x, v_y, v_z)$ Rotate about r by the angle θ

$$T^{-1}R_z(\alpha)R_y(\beta)R_z(\theta)R_y(-\beta)R_z(-\alpha)T$$

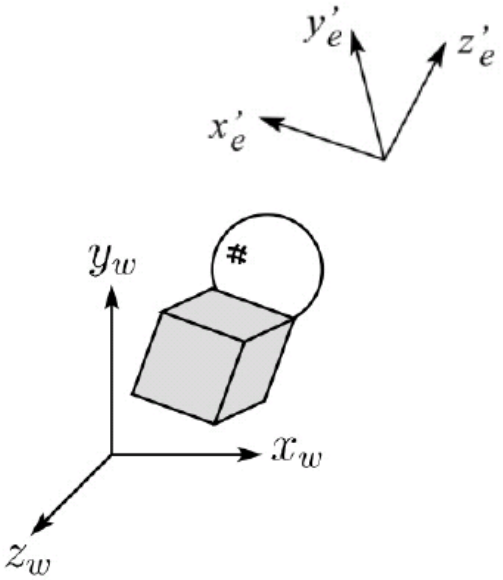
glRotate*(θ, v_x, v_y, v_z)

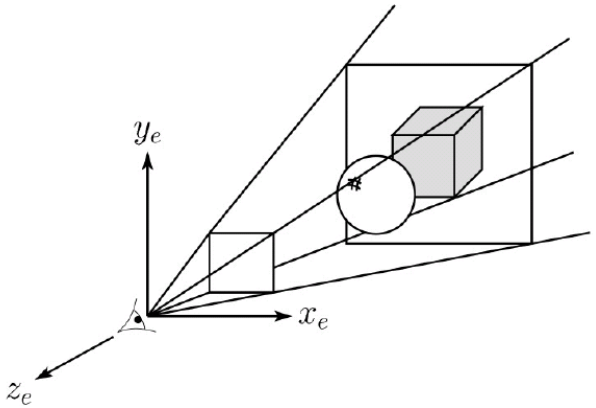
3D Geometry Pipeline



3D Geometry Pipeline

- Now look at how we would compute the world->eye transformation



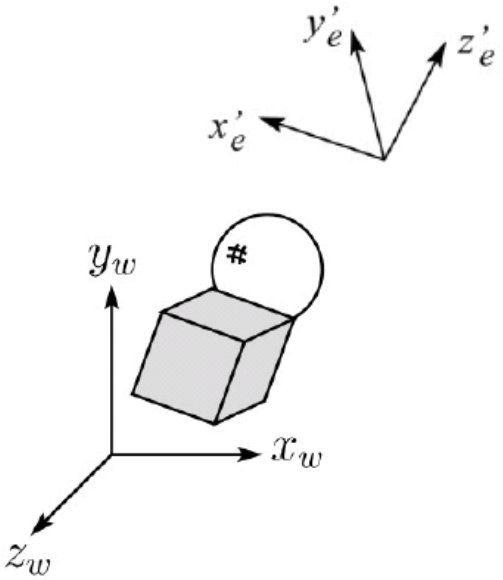


World space

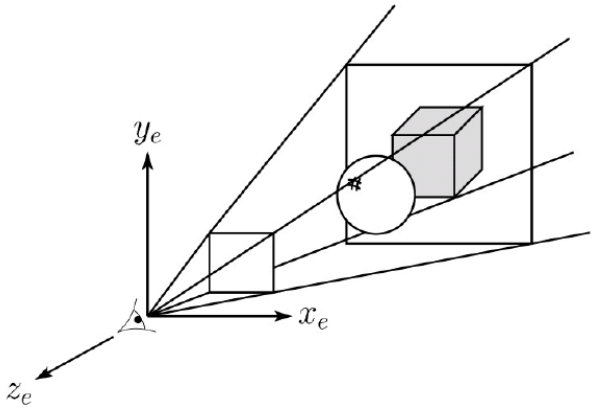
View space

3D Geometry Pipeline

- Now look at how we would compute the world->eye transformation



Rotate&translate

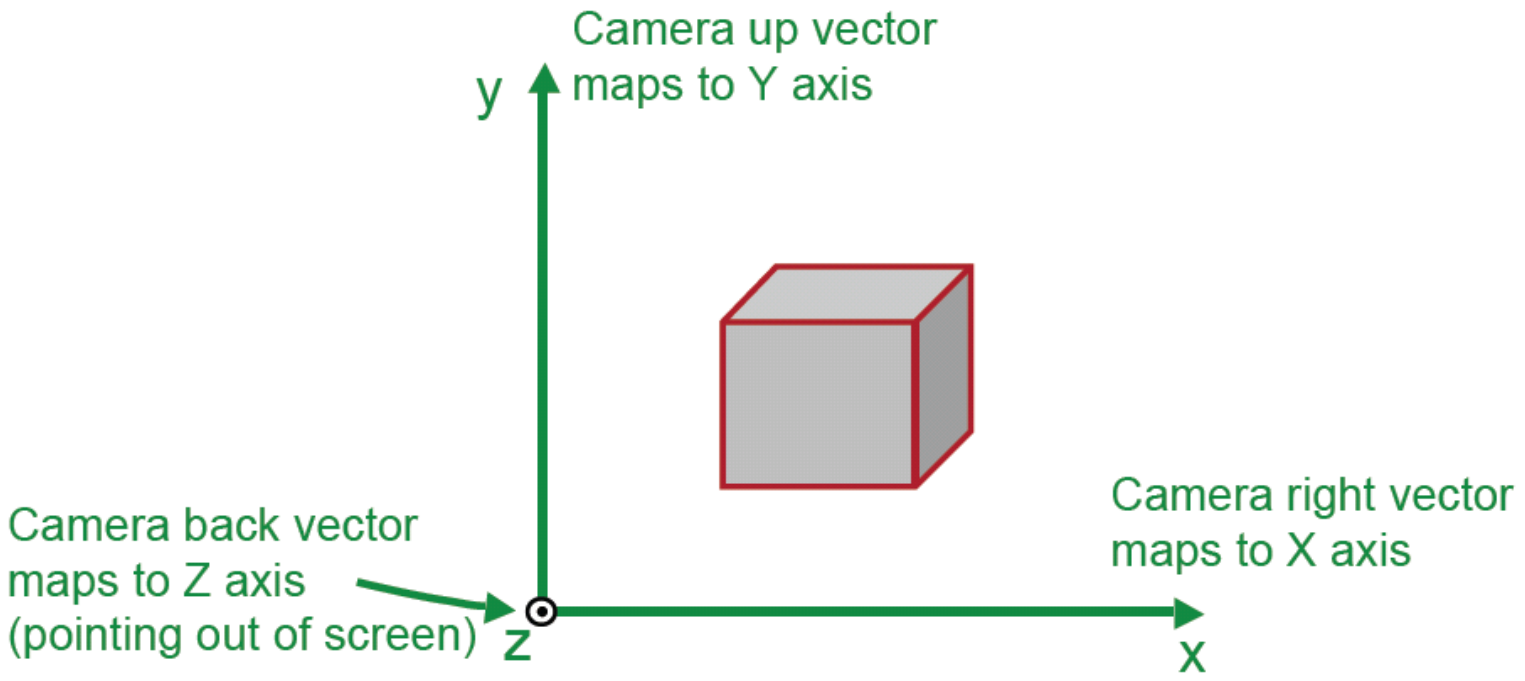


World space

View space

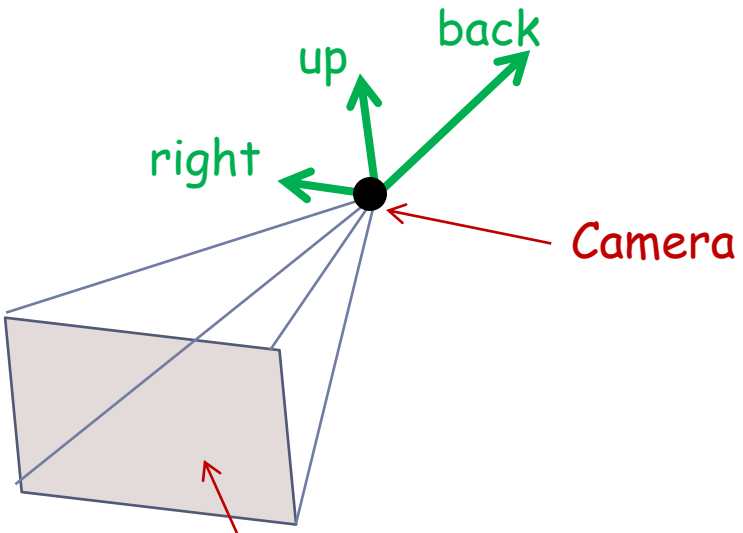
Camera Coordinate

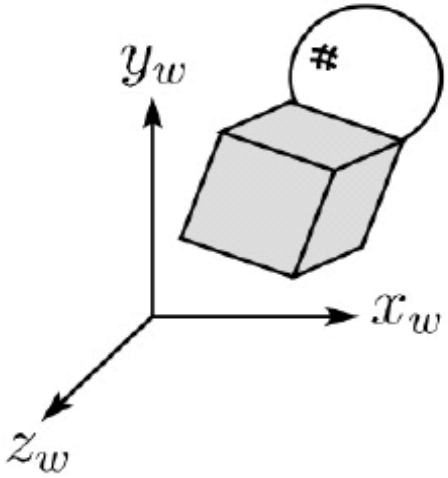
- Canonical coordinate system
  - usually right handed (looking down –z axis)
  - convenient for project and clipping



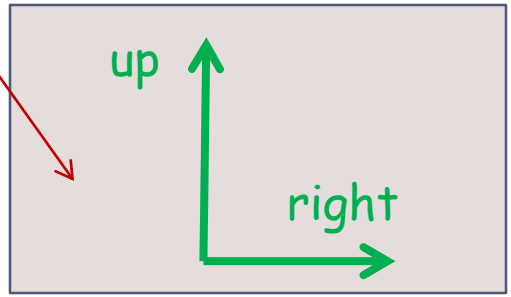
Camera Coordinate

- Mapping from world to eye coordinates
  - eye position maps to origin
  - right vector maps to x axis
  - up vector maps to y axis
  - back vector maps to z axis





View plane



Viewing Transformation

- We have the camera in world coordinates
- We want the transformation T which takes object from world to camera

$$p^c = T_{w,c} p^w$$

Viewing Transformation

- We have the camera in world coordinates
- We want the transformation T which takes object from world to camera

$$p^c = T_{w,c} p^w$$

- Trick: find T^{-1} taking object from camera to world

$$p^w = (T_{w,c})^{-1} p^c$$

Viewing Transformation

- We have the camera in world coordinates
- We want the transformation T which takes object from world to camera

$$p^c = T_{w,c} p^w$$

- Trick: find T^{-1} taking object from camera to world

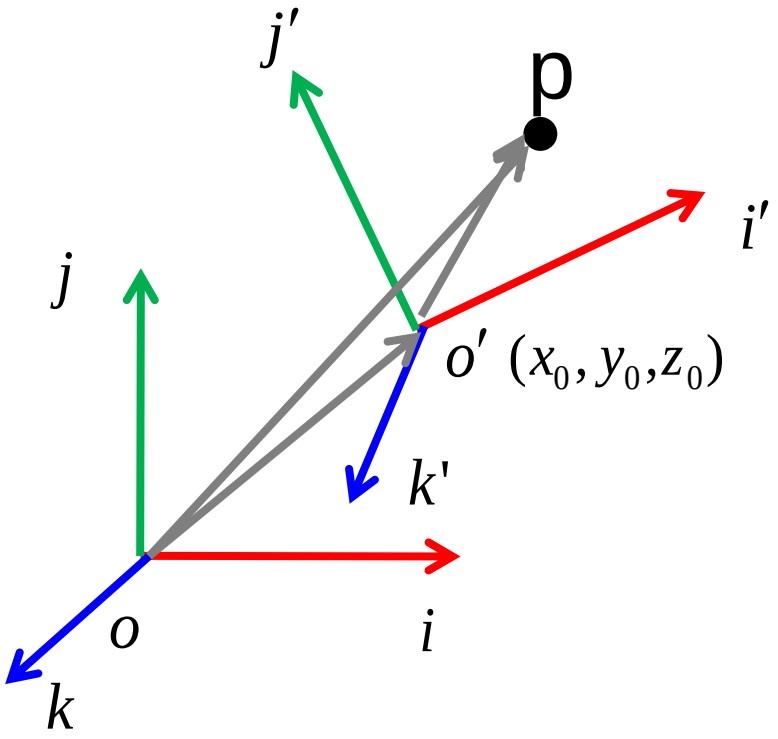
$$p^w = (T_{w,c})^{-1} p^c$$

$$T_{w,c}{}^{-1} = T_{c,w} = \begin{pmatrix} a & e & i & m \\ b & f & j & n \\ c & g & k & o \\ d & h & l & p \end{pmatrix} \quad ?$$

Review: 3D Coordinate Trans.

- Transform object description from $i'j'k'$ to ijk

  (i,j,k are orthonormal basis vectors so i', j', k')

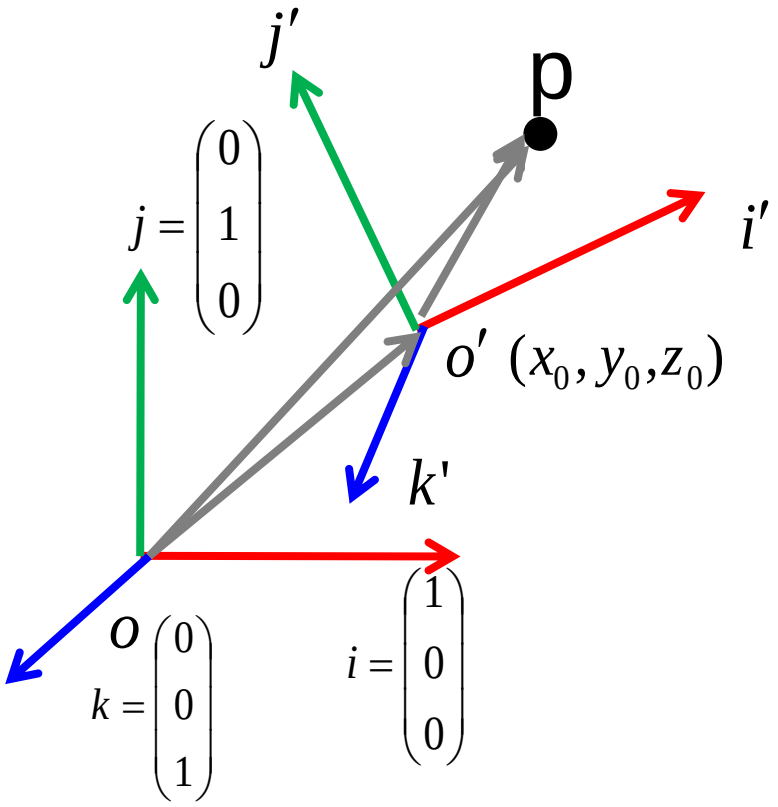


$$\overrightarrow{op} = \overrightarrow{oo'} + \overrightarrow{o'p}$$

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \vec{i}^T\vec{i}' & \vec{i}^T\vec{j}' & \vec{i}^T\vec{k}' & x_0 \\ \vec{j}^T\vec{i}' & \vec{j}^T\vec{j}' & \vec{j}^T\vec{k}' & y_0 \\ \vec{k}^T\vec{i}' & \vec{k}^T\vec{j}' & \vec{k}^T\vec{k}' & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

Review: 3D Coordinate Trans.

- Transform object description from $i'j'k'$ to ijk

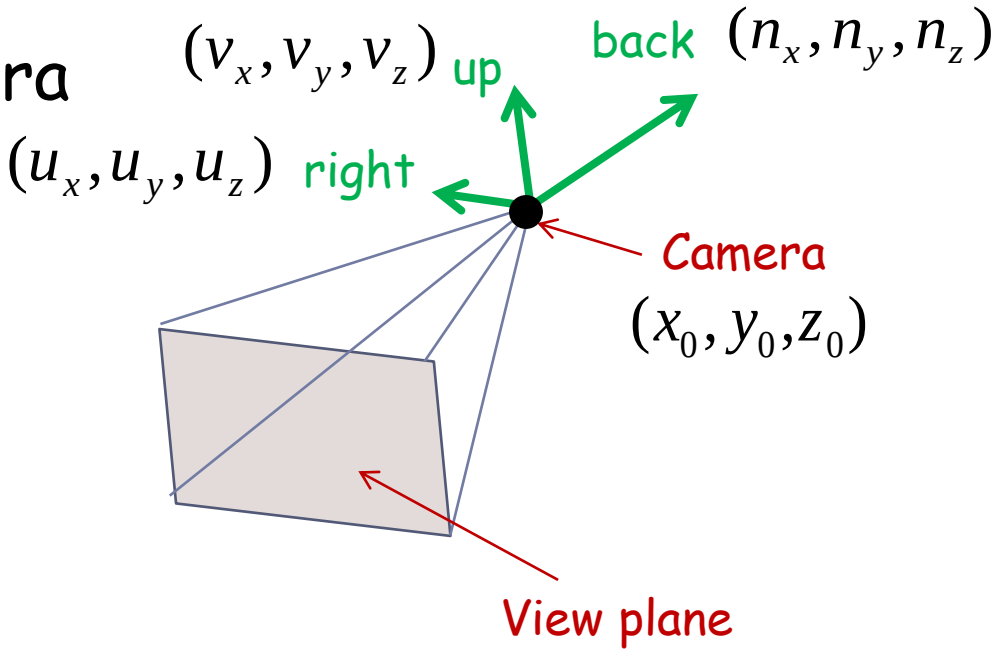


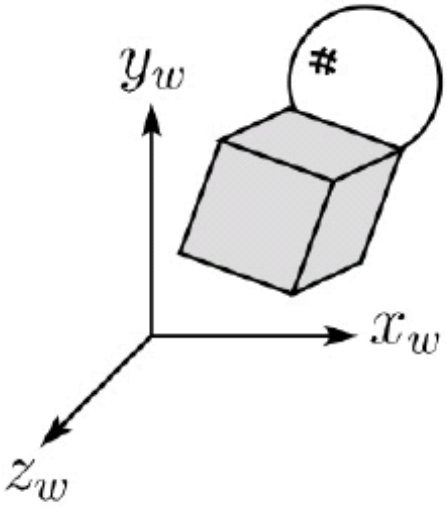
$$\overrightarrow{op} = \overrightarrow{oo'} + \overrightarrow{o'p}$$

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} i'_x & j'_x & k'_x & x_0 \\ i'_y & j'_y & k'_y & y_0 \\ i'_z & j'_z & k_z' & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

Review: 3D Coordinate Trans.

- Transform object description from camera to world

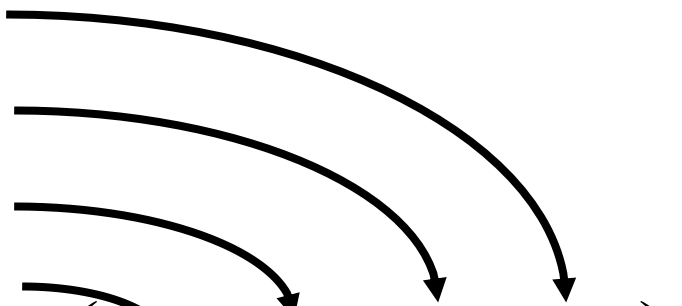




$$T_{w,c}^{-1} = T_{c,w} = \begin{pmatrix} u_x & v_x & n_x & x_0 \\ u_y & v_y & n_y & y_0 \\ u_z & v_z & n_z & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Viewing Transformation

- Trick: find T^{-1} taking object from camera to world
  - eye position maps to origin
  - back vector maps to z axis
  - up vector maps to y axis
  - right vector maps to x axis



$$T_{w,c}^{-1} = T_{c,w} = \begin{pmatrix} u_x & v_x & n_x & x_0 \\ u_y & v_y & n_y & y_0 \\ u_z & v_z & n_z & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Viewing Transformation

- Trick: find T^{-1} taking object from camera to world

$$T_{w,c} = \begin{pmatrix} u_x & v_x & n_x & x_0 \\ u_y & v_y & n_y & y_0 \\ u_z & v_z & n_z & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} = \begin{bmatrix} \vec{u} & \vec{v} & \vec{n} & \vec{p}_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}$$

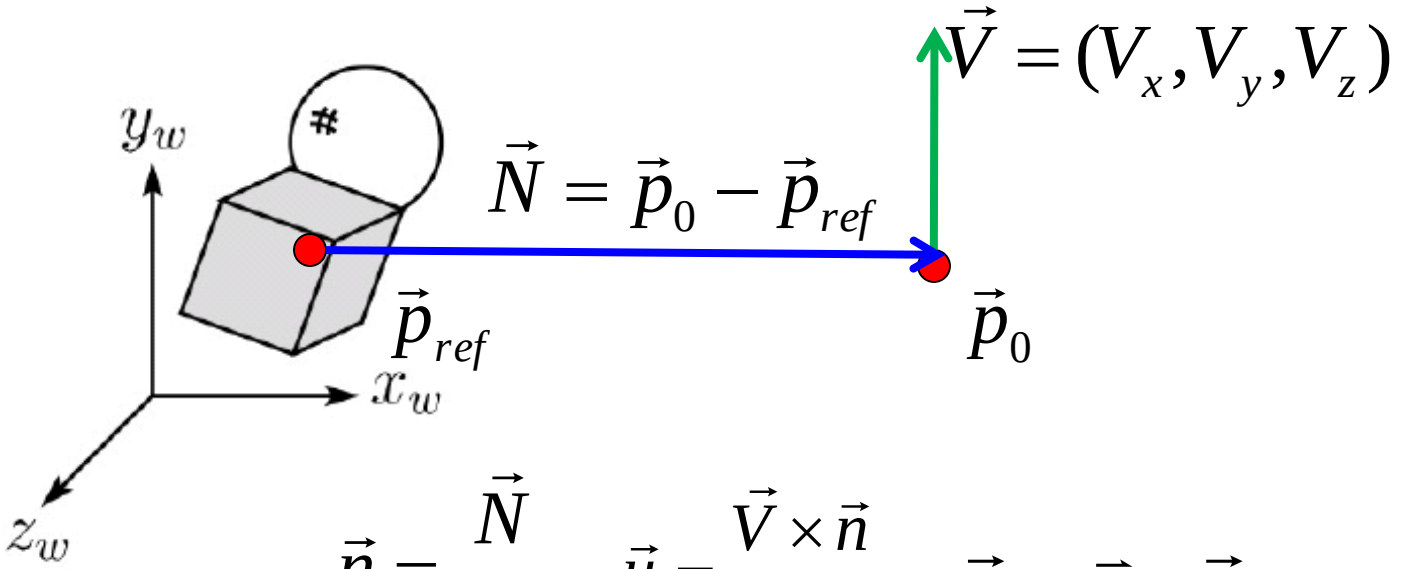
$$T_{w,c} = \begin{bmatrix} \vec{u}^T & -\vec{u} \bullet \vec{p}_0 \\ \vec{v}^T & -\vec{v} \bullet \vec{p}_0 \\ \vec{n}^T & -\vec{n} \bullet \vec{p}_0 \\ \vec{0} & 1 \end{bmatrix}$$

H&B equation (7.4)

Viewing Trans: gluLookAt

- Mapping from world to eye coordinates

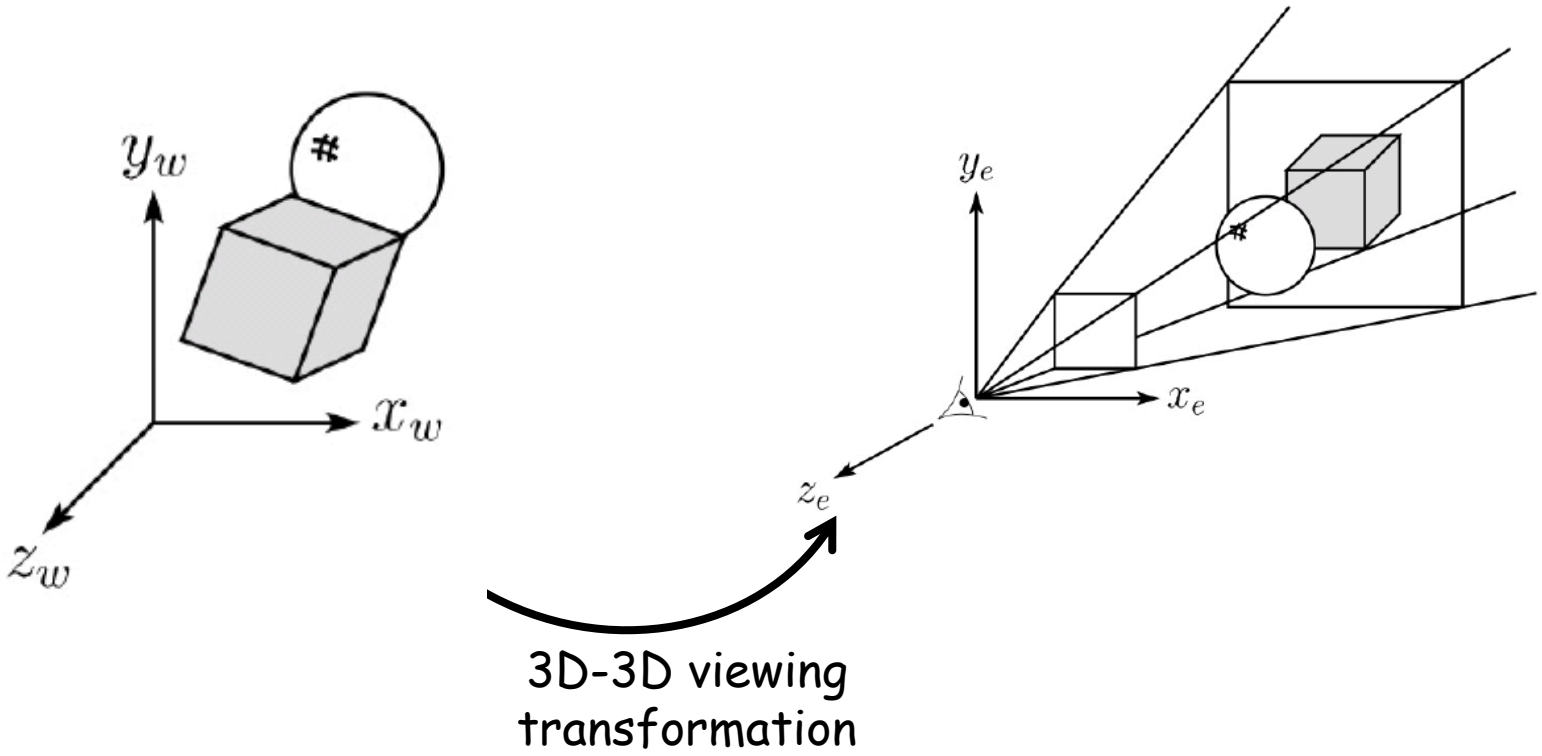
gluLookAt (x0,y0,z0,xref,yref,zref,Vx, Vy,Vz)



$$\vec{n} = \frac{\vec{N}}{|\vec{N}|} \qquad \vec{u} = \frac{\vec{V} \times \vec{n}}{|\vec{V}|} \qquad \vec{v} = \vec{n} \times \vec{u}$$

H&B equation (7.1)

3D Geometry Pipeline



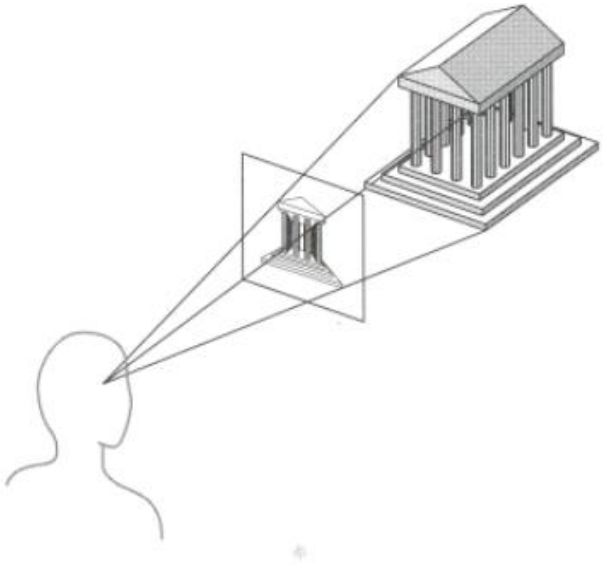
Προβολές

Projection

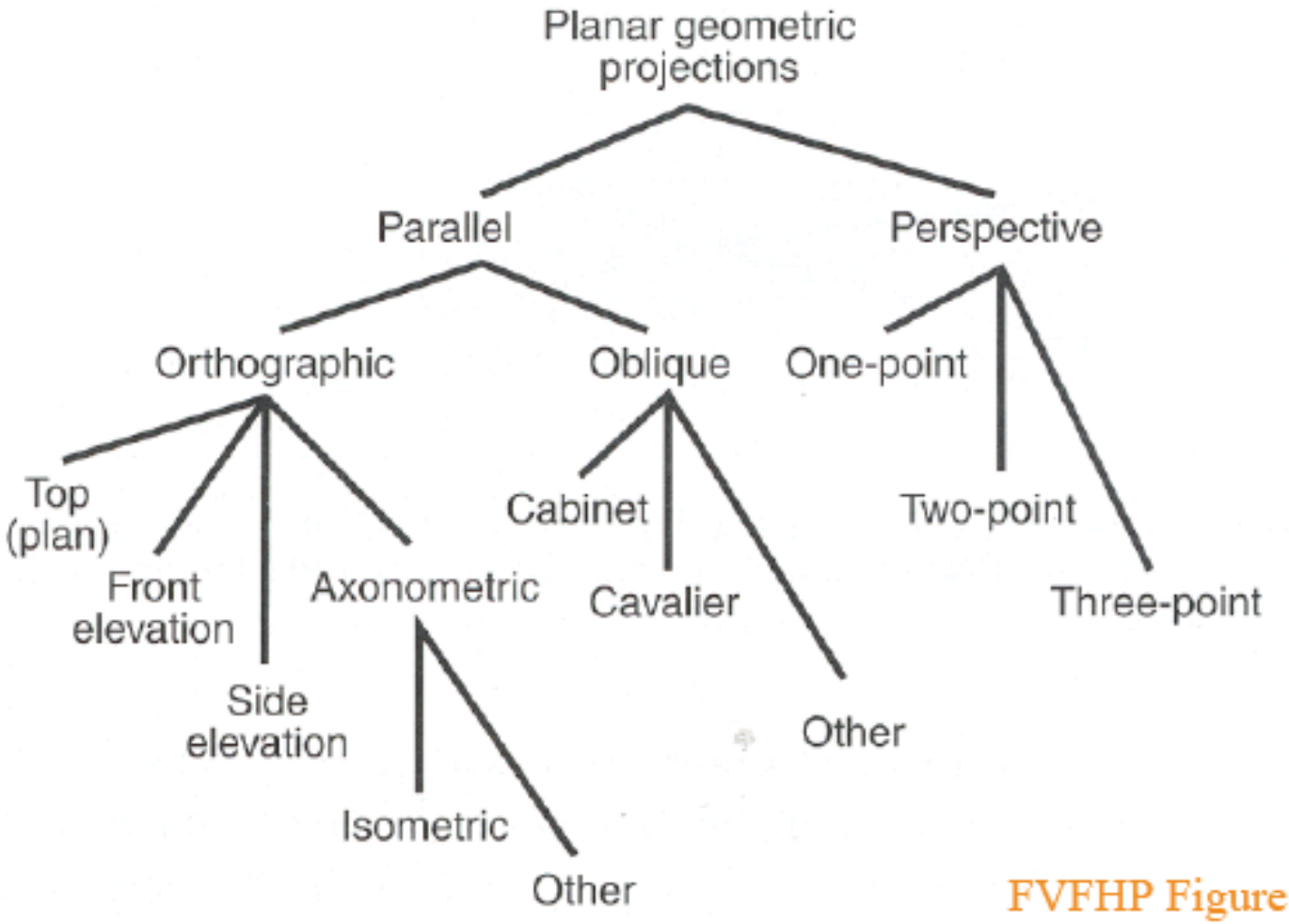
- General definition

  transform points in n-space to m-space (m<n)
- In computer graphics

  map 3D coordinates to 2D screen coordinates



Taxonomy of Projections

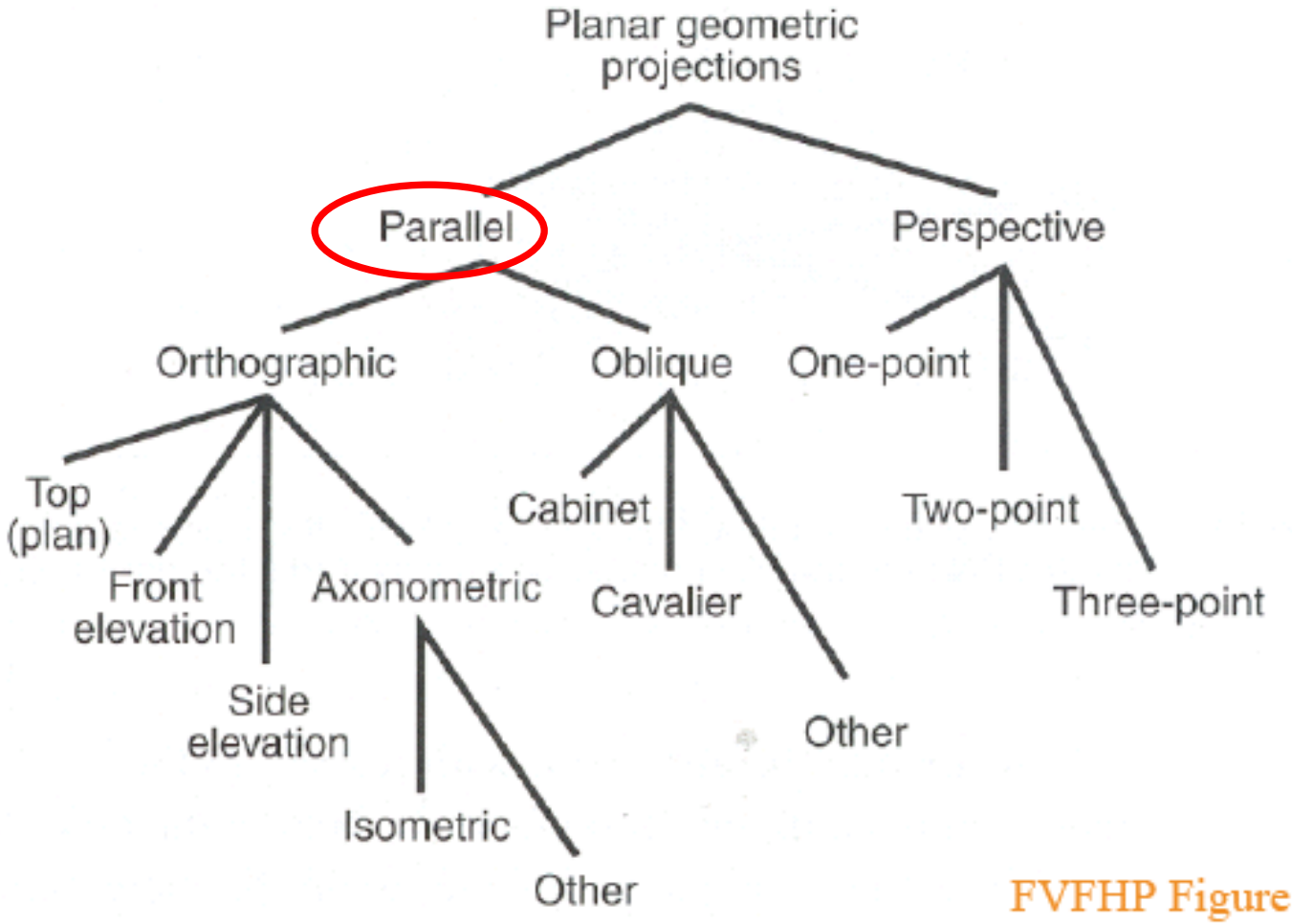


FVFHP Figure 6.10

Όροι

- Perspective Projection (Προοπτική προβολή)
- Parallel Projection (Παράλληλη προβολή)
  - Orthographic projections (Ορθογραφικές/ορθογώνιες προβολές)
  - Oblique projections (Πλάγιες)

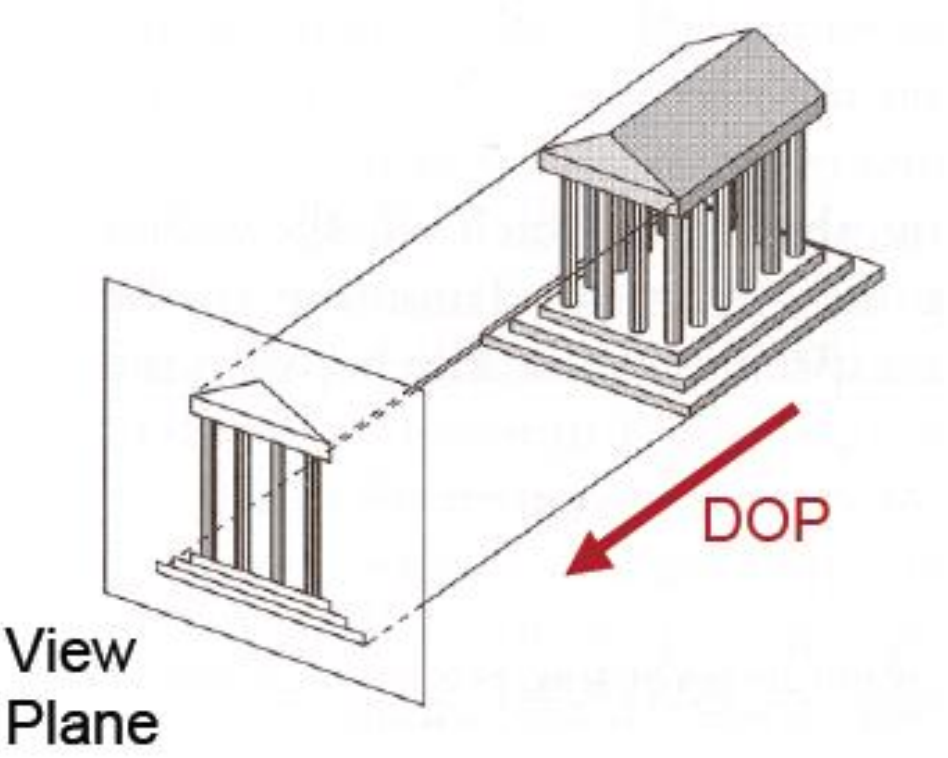
Taxonomy of Projections



FVFHP Figure 6.10

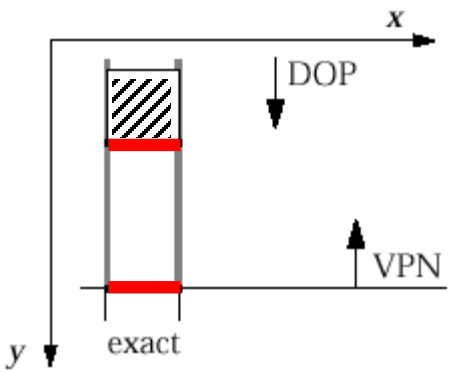
Parallel Projection

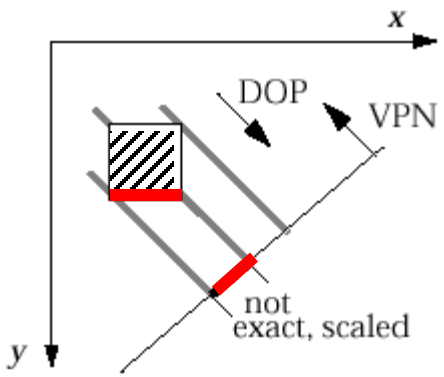
- Center of projection is at infinity
  Direction of projection (DOP) same for all points

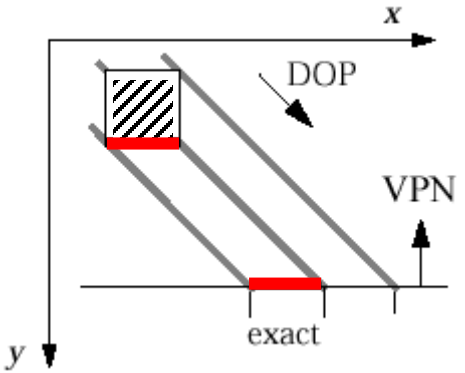


Overview of Parallel Projections

Assume object face of interest lies in principal plane, i.e., parallel to xy, yz, or zx planes. (DOP – Direction of Projection, VPN = View Plane Normal)







1) Multiview Orthographic

-VPN || a principal coordinate axis

- DOP || VPN

-shows single face, exact measurements

2) Axonometric

-**NOT** (VPN || a principal) coordinate axis

-DOP || VPN

-adjacent faces, none exact, uniformly foreshortened (function of angle between face normal and DOP)

3) Oblique

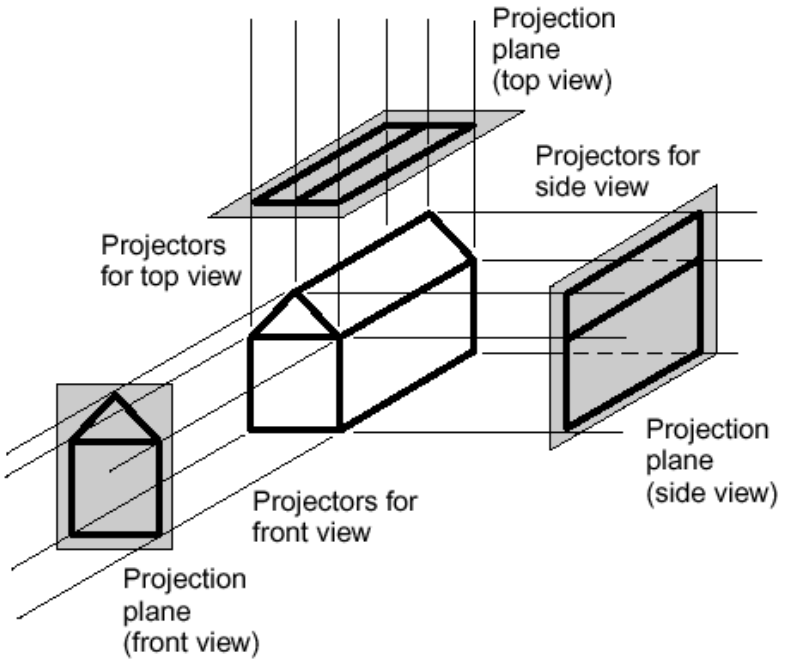
-VPN || a principal coordinate axis

–**NOT** (DOP || VPN)

-adjacent faces, one exact, others uniformly foreshortened

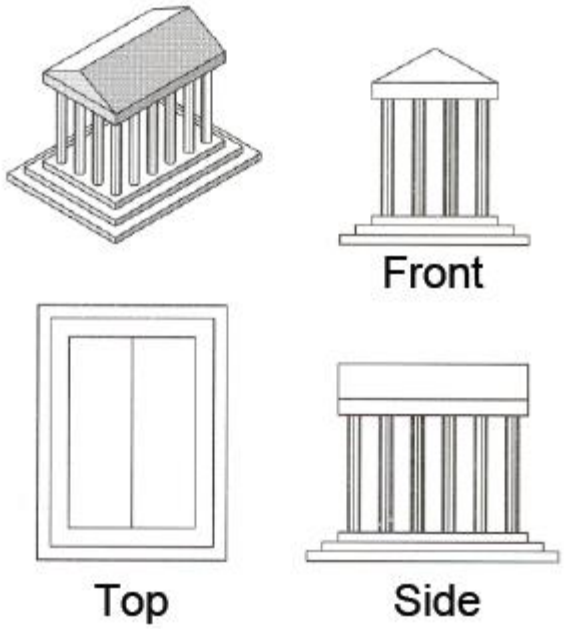
MultiView Orthographic (Parallel)

- Used for:
  - engineering drawings of machines, machine parts
  - working architectural drawings
- Pros:
  - accurate measurement possible
  - all views are at same scale
- Cons:
  - does not provide "realistic" view or sense of 3D form
- Usually need multiple views to get a three-dimensional feeling for object



Orthographic Projection

- Direction of projection (DOP) perpendicular to view plane

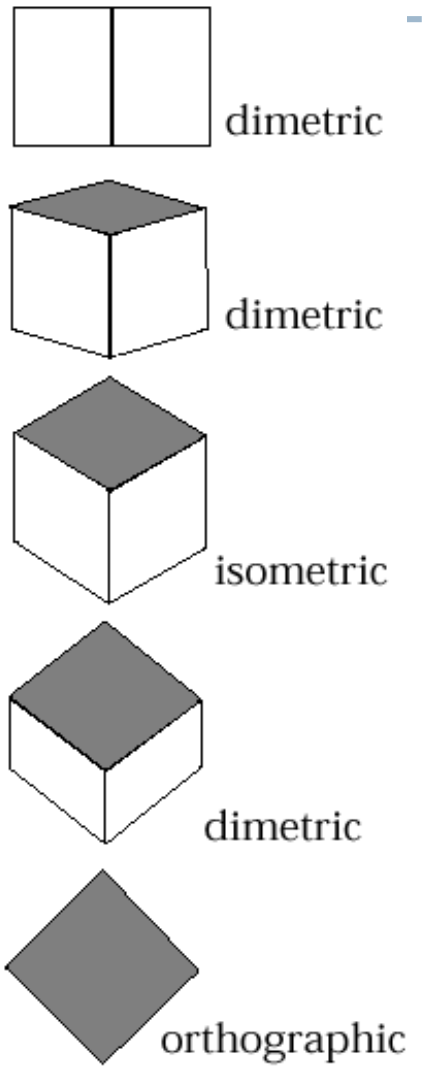


$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Axonometric (Parallel)

- Same method as multiview orthographic projections, except projection plane not parallel to any of **coordinate planes**; parallel lines equally foreshortened

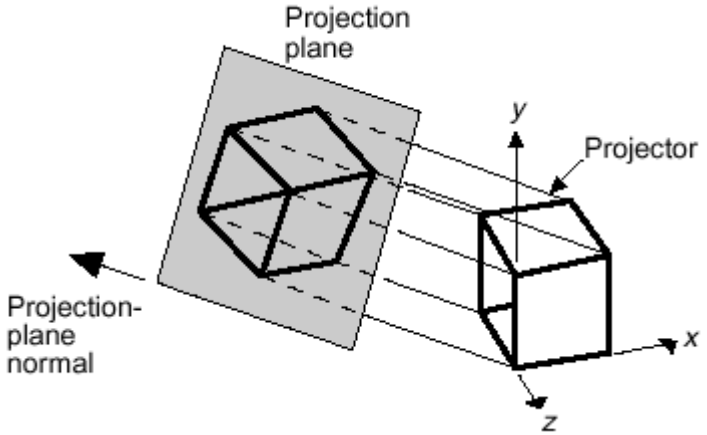
- ***Isometric***: Angles between all three principal axes equal (120°). Same scale ratio applies along each axis
- ***Dimetric***: Angles between two of the principal axes equal; need two scale ratios
- ***Trimetric***: Angles different between three principal axes; need three scale ratios



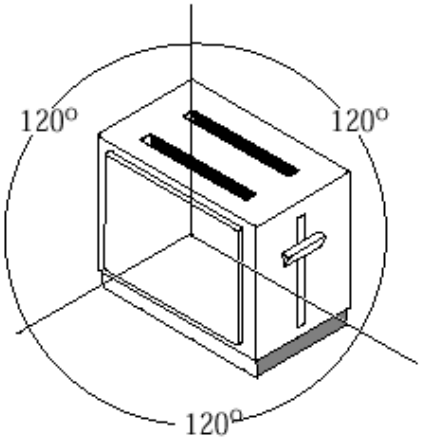
Carlbom Fig.3-8

Isometric Projection

- Used for:
  - catalogue illustrations
  - patent office records
  - furniture design
  - structural design
  - 3d Modeling in real time (Maya, AutoCad, etc.)
- Pros:
  - don't need multiple views
  - illustrates 3D nature of object
  - measurements can be made to scale along principal axes
- Cons:
  - lack of foreshortening creates distorted appearance
  - more useful for rectangular than curved shapes



Construction of an isometric projection: projection plane cuts each principal axis by 45°

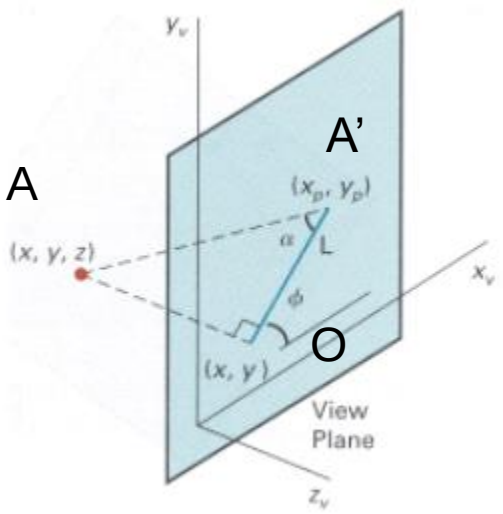


Example

Carlbom Fig.2.2

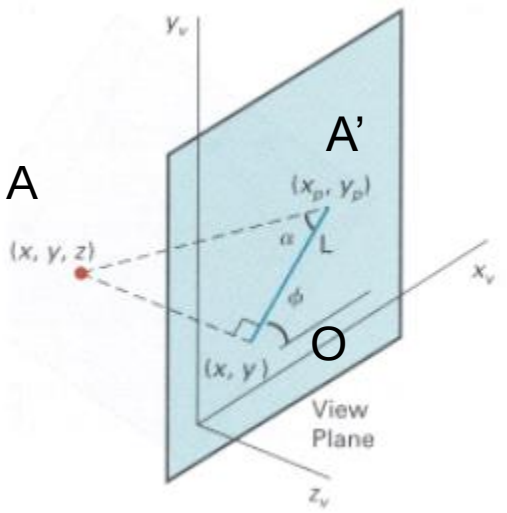
Oblique Projection

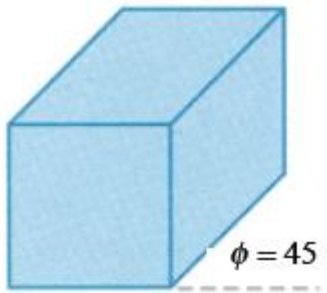
- DOP not perpendicular to view plane

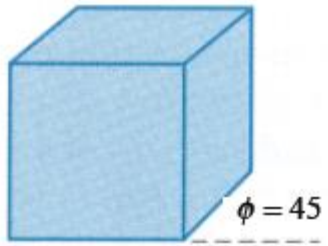


Oblique Projection

- DOP not perpendicular to view plane







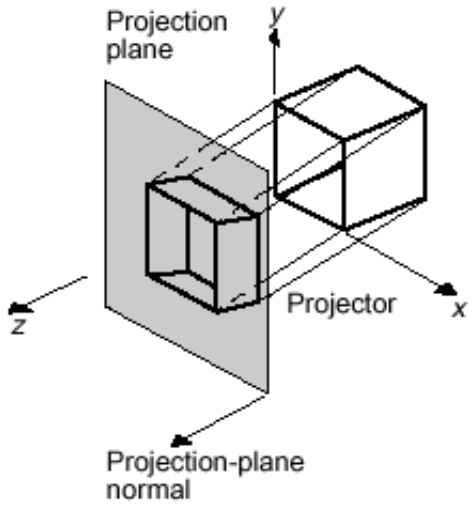
Cavalier (DOP $\alpha = 45^{o}$)

Cabinet (DOP $\alpha = 63.4^{o}$)

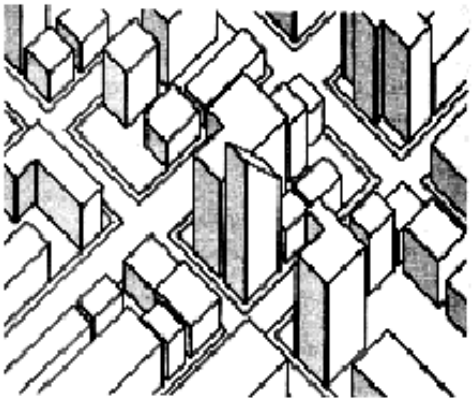
Oblique Projection (Parallel)

- Projectors at oblique angle to projection plane; view cameras have accordion housing, can adjust the projection plane
- Pros:
  - can present exact shape of one face of an object (can take accurate measurements): better for elliptical shapes than axonometric projections, better for "mechanical" viewing
  - lack of perspective foreshortening makes comparison of sizes easier
  - displays some of object's 3D appearance
- Cons:
  - objects can look distorted if careful choice not made about position of projection plane (e.g., circles become ellipses)
  - lack of foreshortening (not realistic looking)

Examples of Oblique Projections

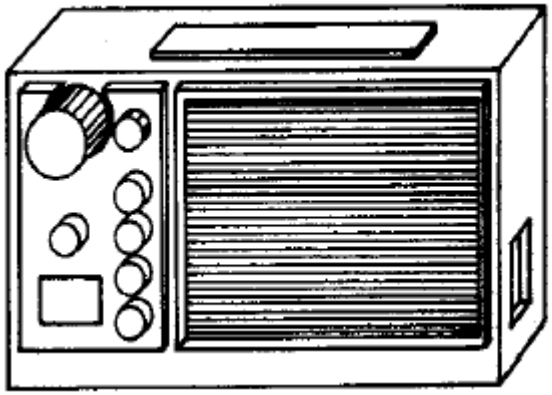


Construction of
oblique parallel projection



(Carlbom Fig. 2-6)

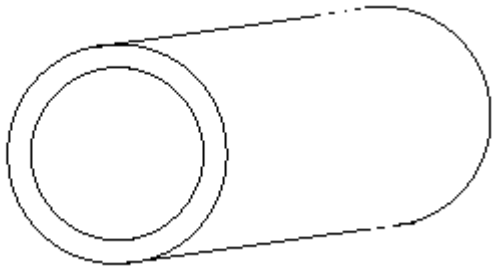
Plan oblique projection of city



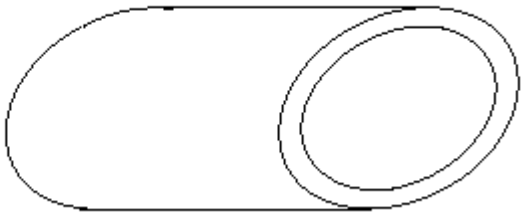
Front oblique projection of radio (Carlbom Fig. 2-4)

Rules for Constructing Oblique Views

- Rules for placing projection plane for oblique views: projection plane should be chosen according to one or several of following:
  - parallel to most irregular of principal faces, or to one which contains circular or curved surfaces
  - parallel to longest principal face of object
  - parallel to face of interest



Projection plane parallel to circular face

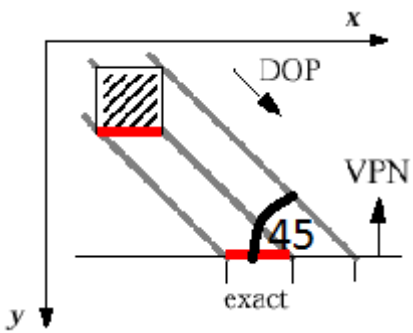


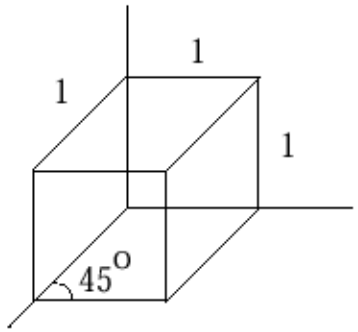
Projection plane not parallel to circular face

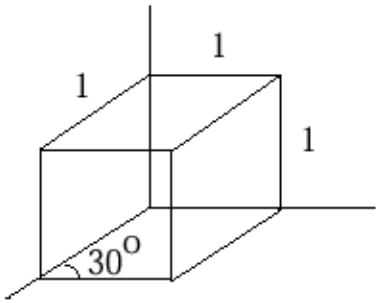
(shearing)

Main Types of Oblique Projections

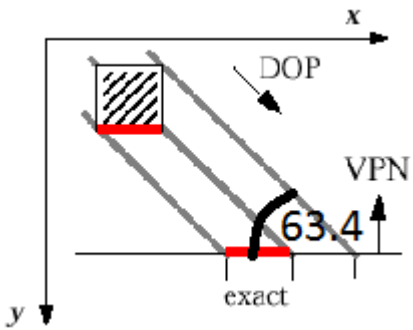
- *Cavalier:* Angle between projectors and projection plane is 45°. Perpendicular faces projected at full scale

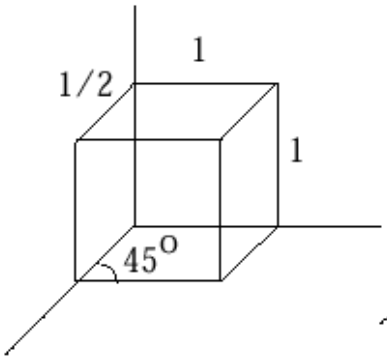


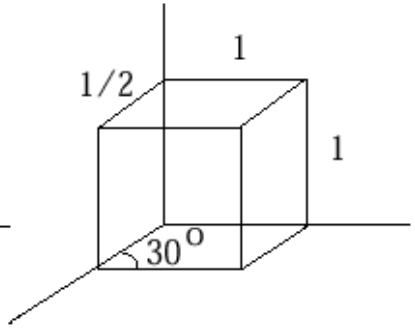




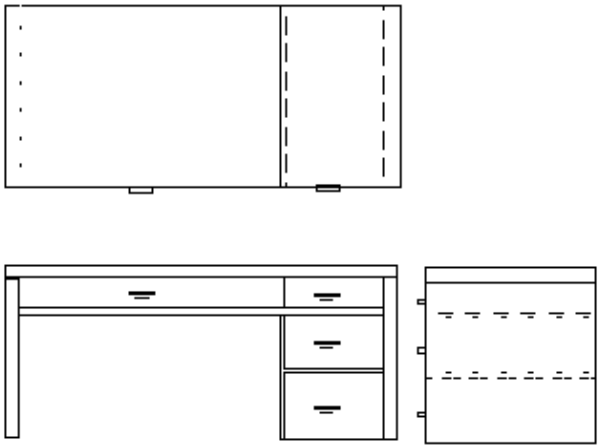
- *Cabinet:* Angle between projectors and projection plane: arctan(2) = 63.4°. Perpendicular faces projected at 50% scale



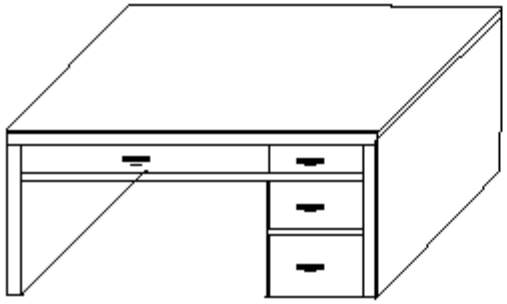




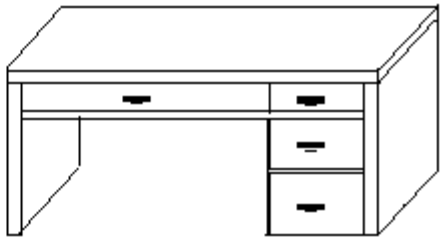
A Desk in Parallel



multiview orthographic



cavalier

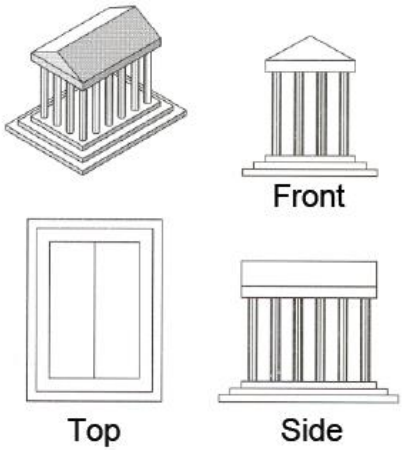


cabinet

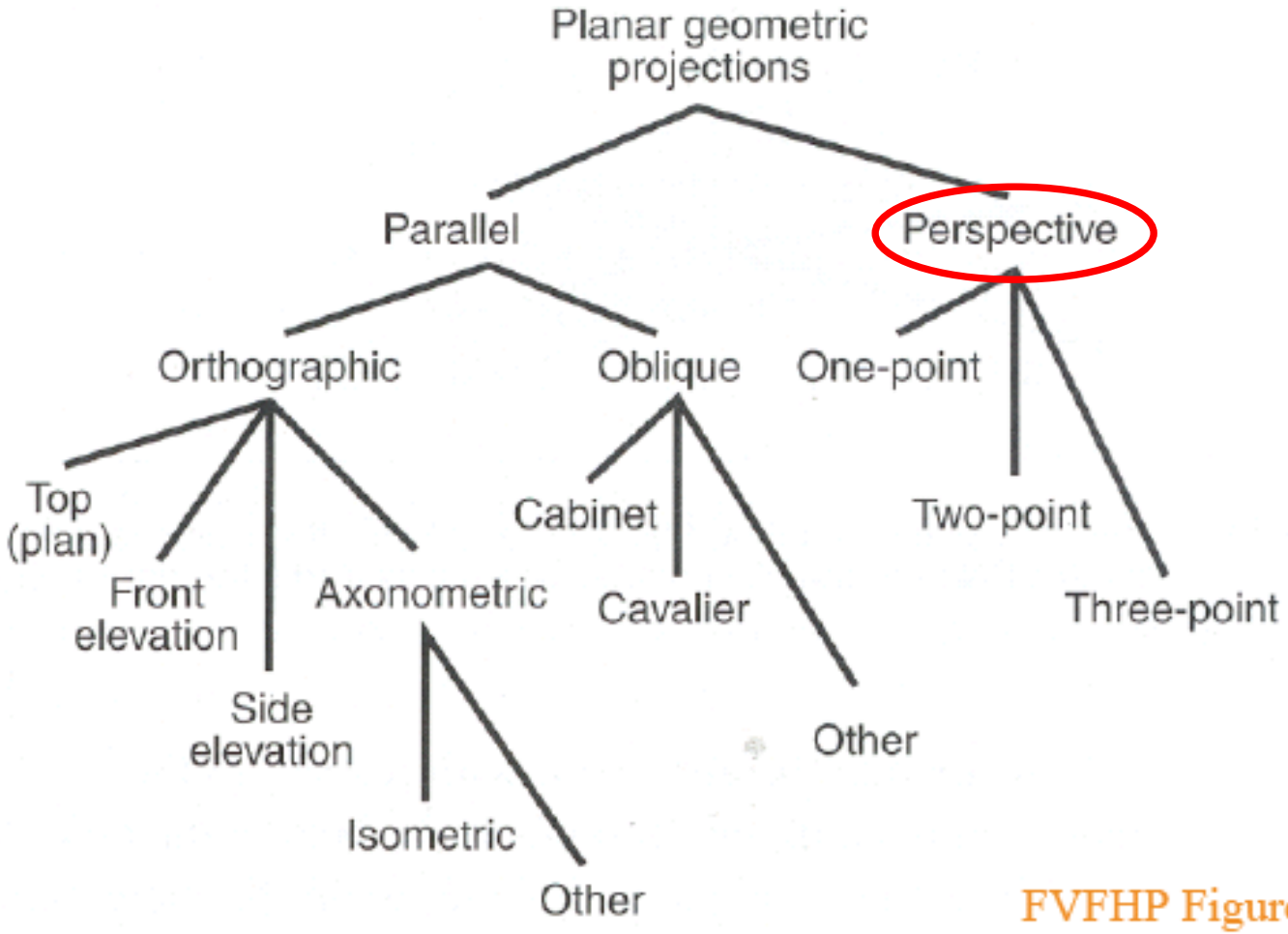
Carlbom Fig. 3-2

Properties of Parallel Projection

- Not realistic looking
- Good for exact measurement
- Are actually affine transformat
  - parallel lines remain parallel
  - ratios are preserved
  - angles are often not preserved
- Most often used in CAD, architectural drawings, etc. where taking exact measurement is important



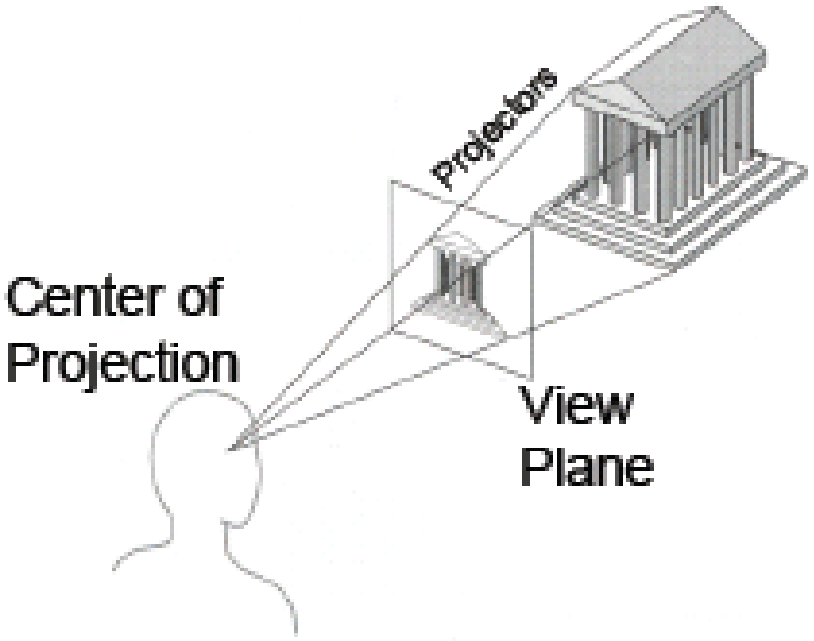
Taxonomy of Projections



FVFHP Figure 6.10

Perspective Projection

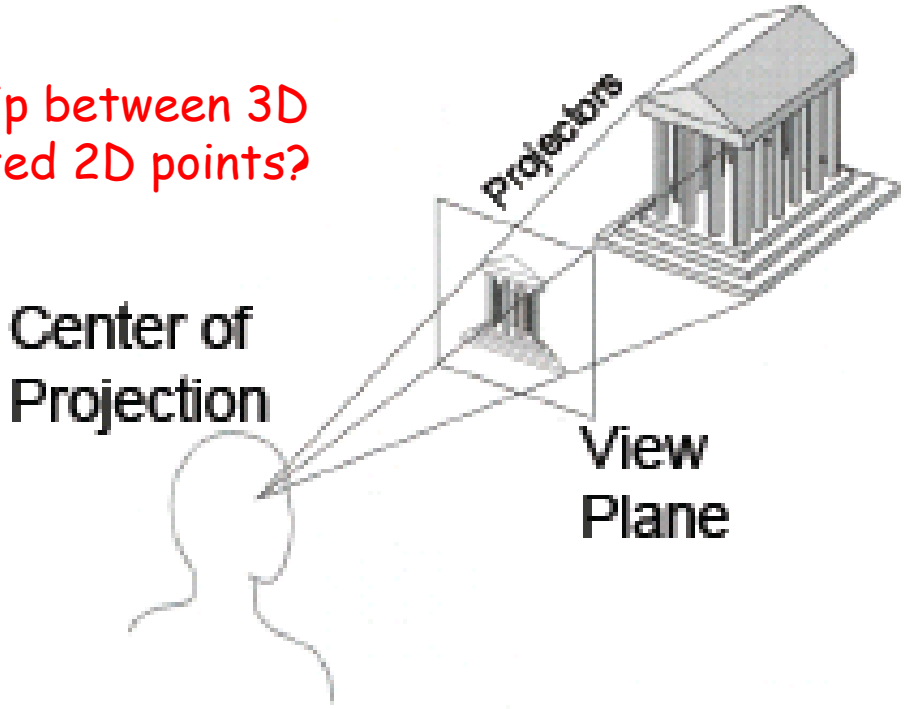
- Maps points onto "view plane" along projectors emanating from "center of projection" (COP)
(κέντρο προβολής / σημείο αναφοράς της προβολής)



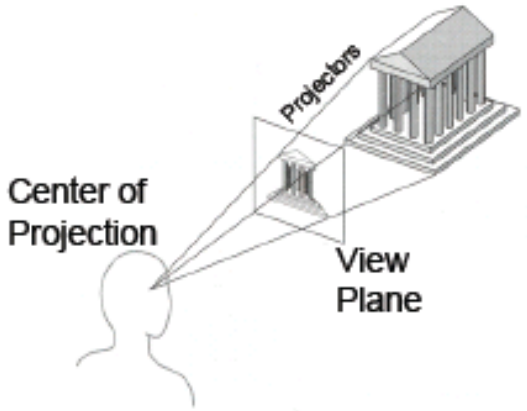
Perspective Projection

- Maps points onto "view plane" along projectors emanating from "center of projection" (COP)

What's relationship between 3D points and projected 2D points?

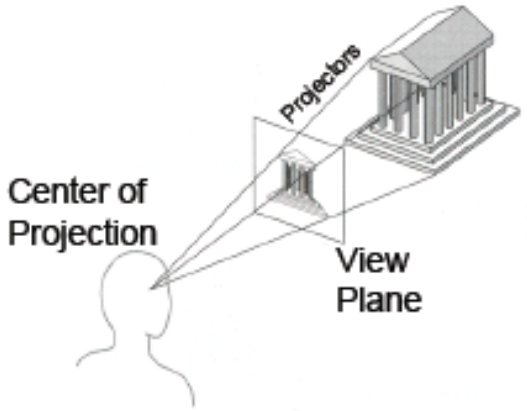


Camera->Screen



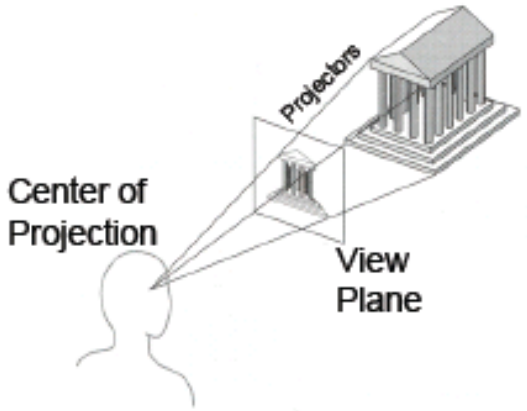
- Remember: Object->camera->screen

Camera->Screen



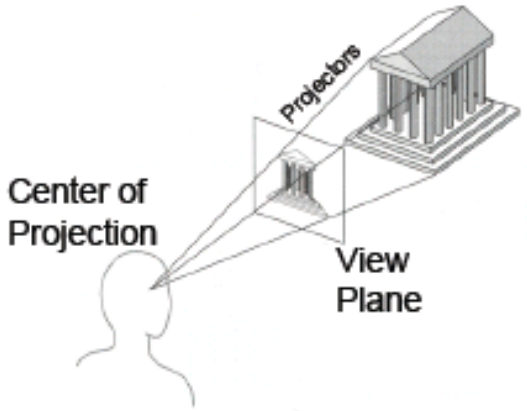
- Remember: Object->camera->screen
- Screen is z=-d plane for some constant d

Camera->Screen



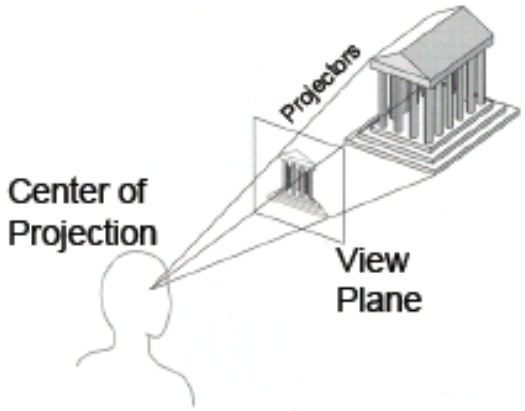
- Remember: Object->camera->screen
- Screen is z=-d plane for some constant d
- Coordinates of origin of screen is (0,0,-d)

Camera->Screen



- Remember: Object->camera->screen
- Screen is z=-d plane for some constant d
- Coordinates of origin of screen is (0,0,-d)
- Its x and y axes is parallel to the x and y axes of eye coordinate system

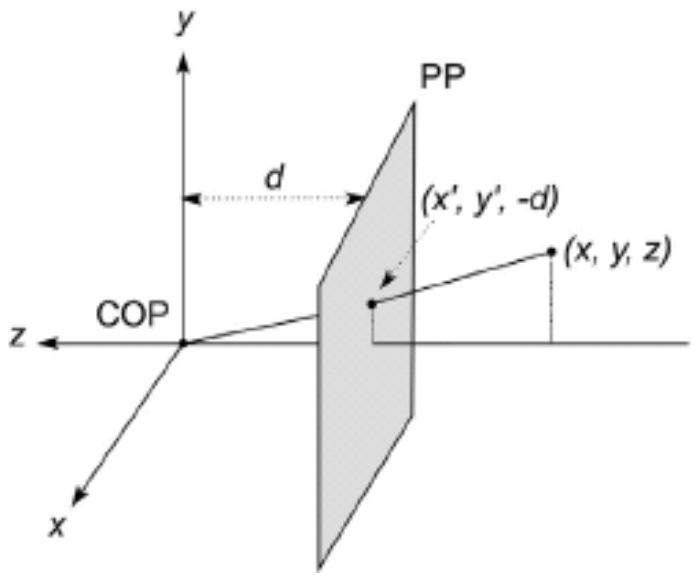
Camera->Screen



- Remember: Object->camera->screen
- Screen is z=-d plane for some constant d
- Coordinates of origin of screen is (0,0,-d)
- Its x and y axes is parallel to the x and y axes of eye coordinate system
- All these coordinates are in camera space now

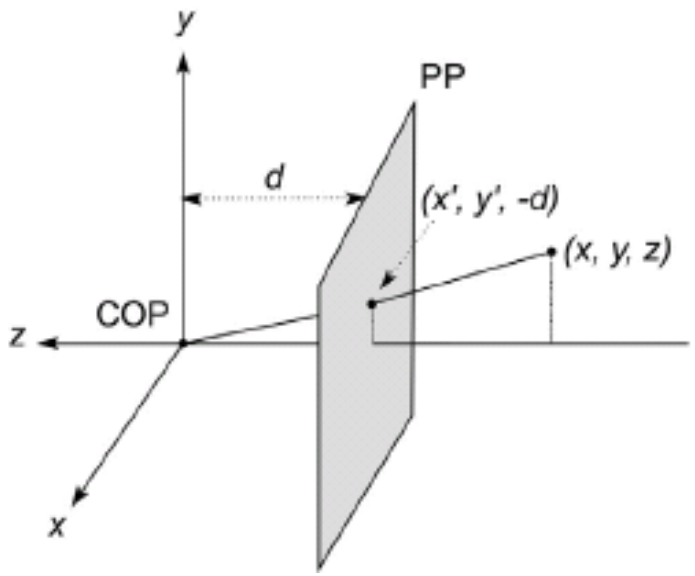
Camera->Screen

- Consider the projection of a point on the camera plane



Camera->Screen

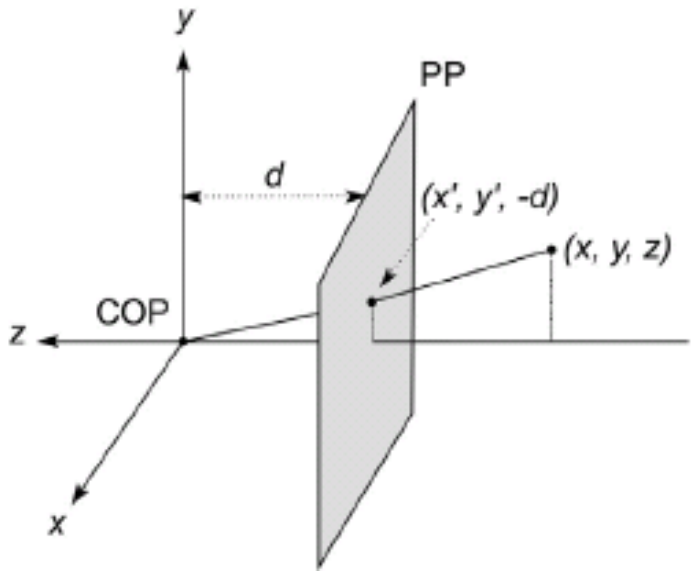
- Consider the projection of a point on the camera plane



By similar triangles, we can compute how much the x and y coordinates are scaled

Camera->Screen

- Consider the projection of a point on the camera plane



$$\frac{x}{z} = \frac{x'}{-d} \Rightarrow x' = \frac{-dx}{z}$$

$$\frac{y}{z} = \frac{y'}{-d} \Rightarrow y' = \frac{-dy}{z}$$

By similar triangles, we can compute how much the x and y coordinates are scaled

Homogeneous Point Revisited

- Remember how we said 2D/3D geometric transformations work with the last coordinate always set to one
- What happens if the coordinate is not one
- We divide all coordinates by *w*:

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \rightarrow \begin{bmatrix} x/w \\ y/w \\ z/w \\ 1 \end{bmatrix}$$

If w=1, nothing happens

Sometimes, we call this division step the "perspective divide"

The Perspective Matrix

- Now we can rewrite perspective projection equation as a matrix equation

$$\frac{x}{z} = \frac{x'}{-d} \Rightarrow x' = \frac{-dx}{z}$$

$$\frac{y}{z} = \frac{y'}{-d} \Rightarrow y' = \frac{-dy}{z}$$

The Perspective Matrix

- Now we can rewrite perspective projection equation as a matrix equation

$$\frac{x}{z} = \frac{x'}{-d} \Rightarrow x' = \frac{-dx}{z}$$

$$\frac{y}{z} = \frac{y'}{-d} \Rightarrow y' = \frac{-dy}{z}$$

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

The Perspective Matrix

- Now we can rewrite perspective projection equation as a matrix equation

$$\frac{x}{z} = \frac{x'}{-d} \Rightarrow x' = \frac{-dx}{z}$$

$$\frac{y}{z} = \frac{y'}{-d} \Rightarrow y' = \frac{-dy}{z}$$

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

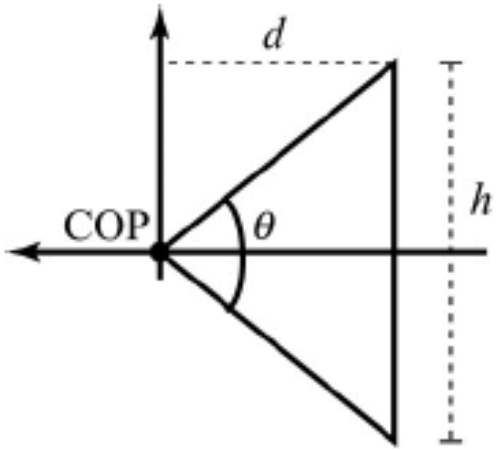
After the division by *w*, we have

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} -dx/z \\ -dy/z \\ 1 \end{pmatrix}$$

Note that this is not a linear transformation!!

Viewing Angle

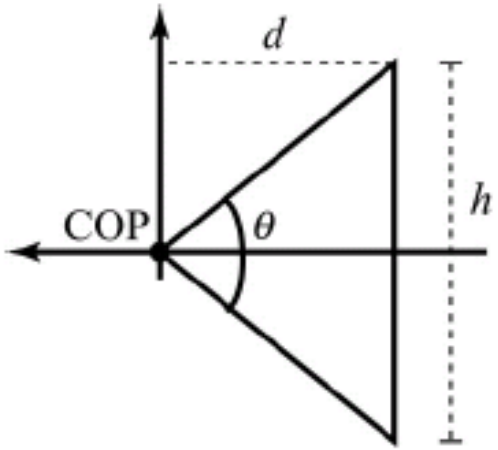
- An alternative to specifying the distance between COP and PP is to specify viewing angle:



Given the height of the image *h* and θ , what is *d*?

Viewing Angle

- An alternative to specifying the distance between COP and PP is to specify viewing angle:

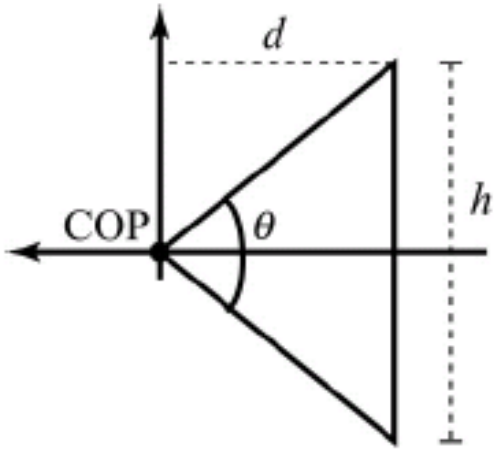


Given the height of the image *h* and θ , what is *d*?

$$\tan\frac{\theta}{2} = \frac{h}{2d}$$

Viewing Angle

- An alternative to specifying the distance between COP and PP is to specify viewing angle:

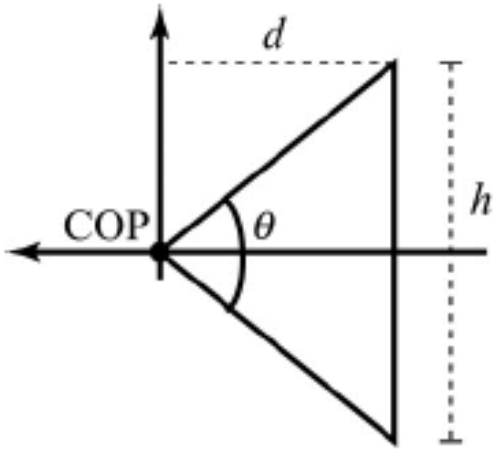


Given the height of the image *h* and θ , what is *d*?

$$\tan\frac{\theta}{2} = \frac{h}{2d} \quad \Rightarrow d = \frac{h \cdot \cot\frac{\theta}{2}}{2}$$

Viewing Angle

- An alternative to specifying the distance between COP and PP is to specify viewing angle:



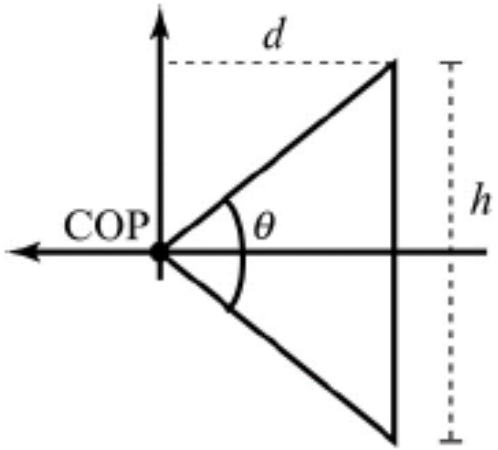
Given the height of the image *h* and θ , what is *d*?

$$\tan\frac{\theta}{2}=\frac{h}{2d} \quad \Rightarrow d=\frac{h\cdot\cot\frac{\theta}{2}}{2} \quad \Rightarrow d=\frac{w\cdot\cot\frac{\theta}{2}}{2\cdot aspect}$$

where aspect= height/width of the view plane

Viewing Angle

- An alternative to specifying the distance between COP and PP is to specify viewing angle:



Given the height of the image *h* and θ , what is *d*?

$$\tan\frac{\theta}{2}=\frac{h}{2d} \quad \Rightarrow d=\frac{h\cdot\cot\frac{\theta}{2}}{2} \quad \Rightarrow d=\frac{w\cdot\cot\frac{\theta}{2}}{2\cdot aspect}$$

What happens to *d* as θ increases (keep *d* constant)?