Hochiminh city University of Technology
Faculty of Computer Science and Engineering

COMPUTER GRAPHICS

CHAPTER 7 :

Viewing

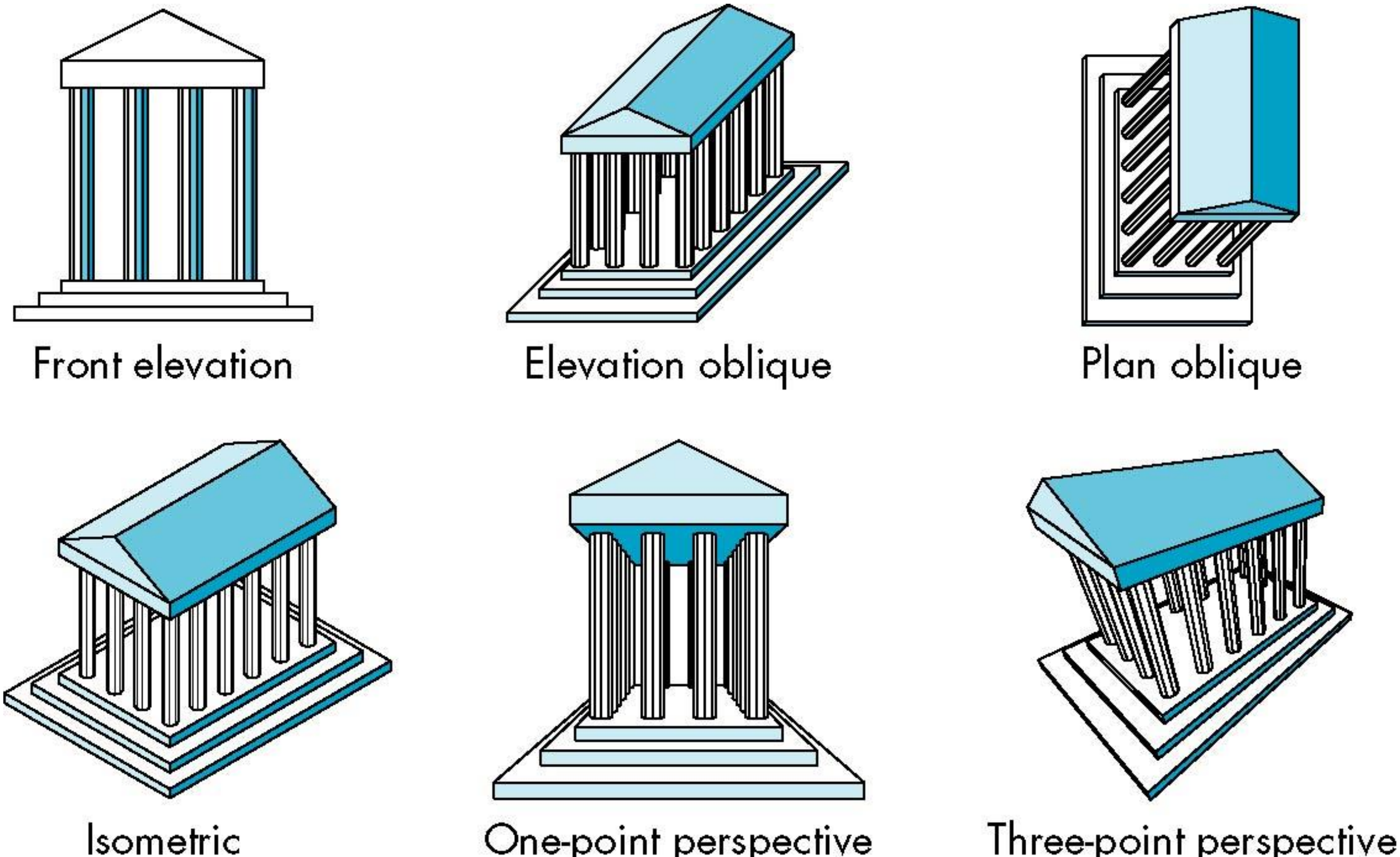
OUTLINE

- Classical Viewing
- Orthographic Projection
- Axonometric Projections
- Oblique Projection
- Perspective Projection
- Computer Viewing
- View Volume

Classical Viewing

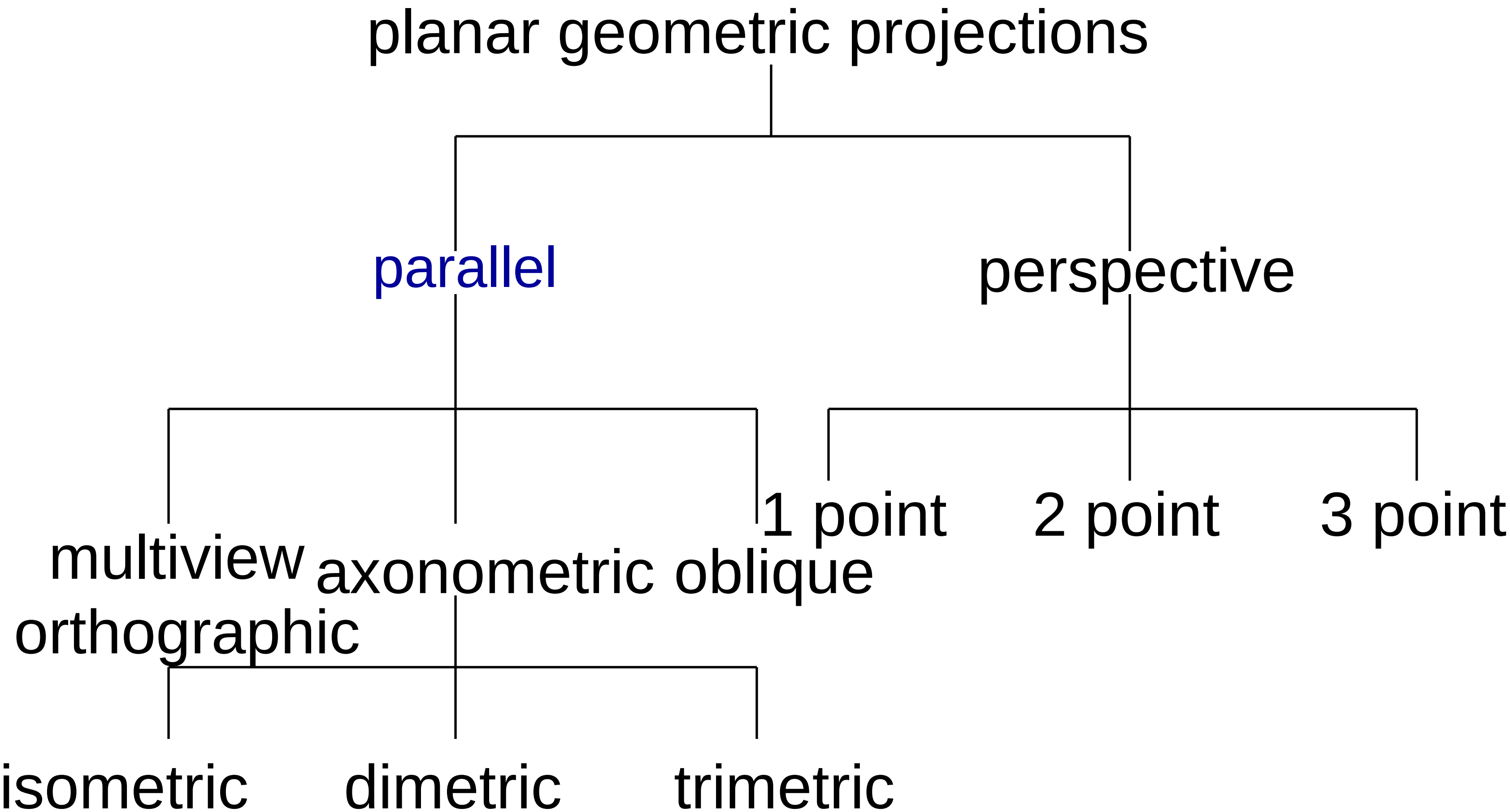
Classical Viewing

❑ Classical Projections



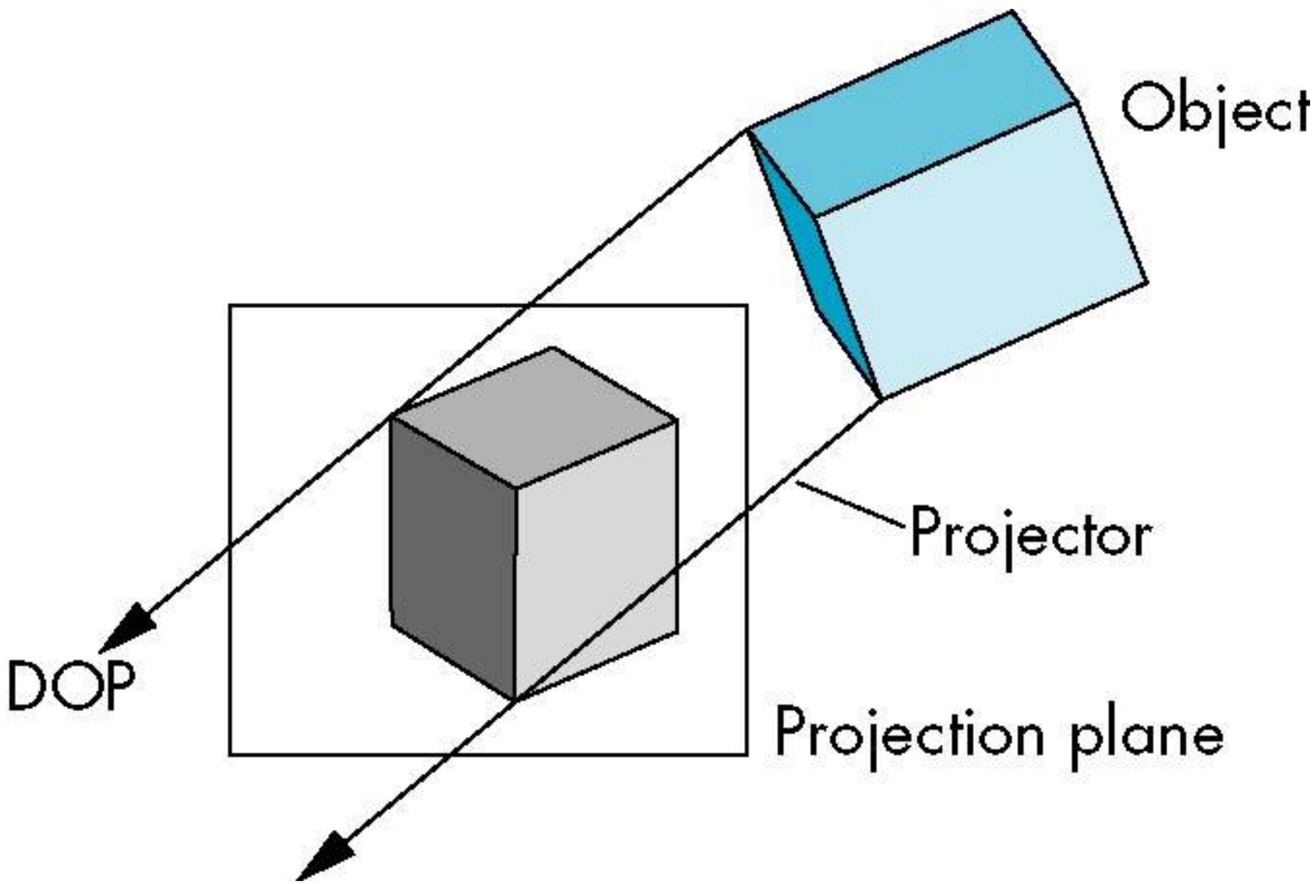
Classical Viewing

- Taxonomy of Planar Geometric Projections



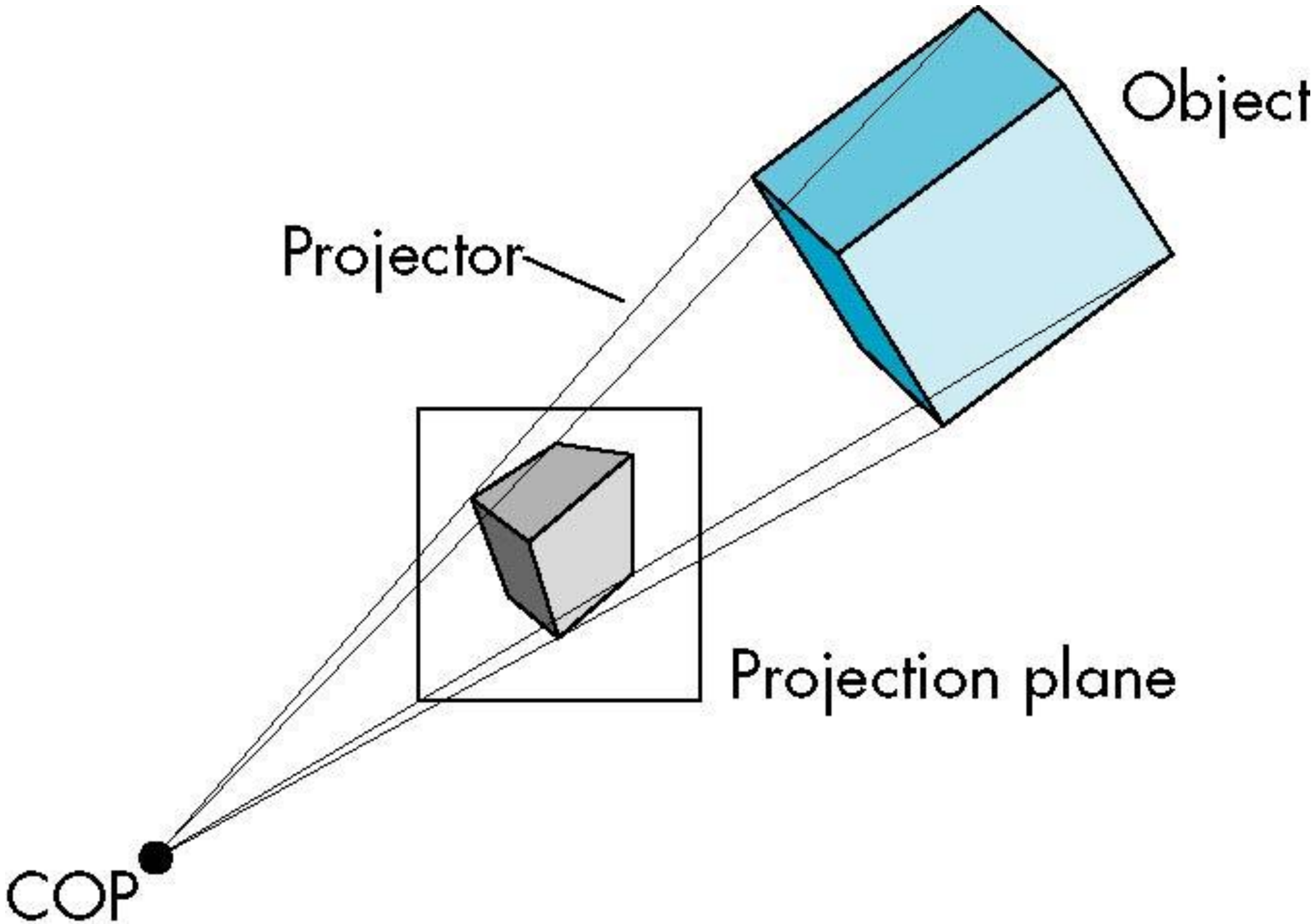
Classical Viewing

- Parallel Projection



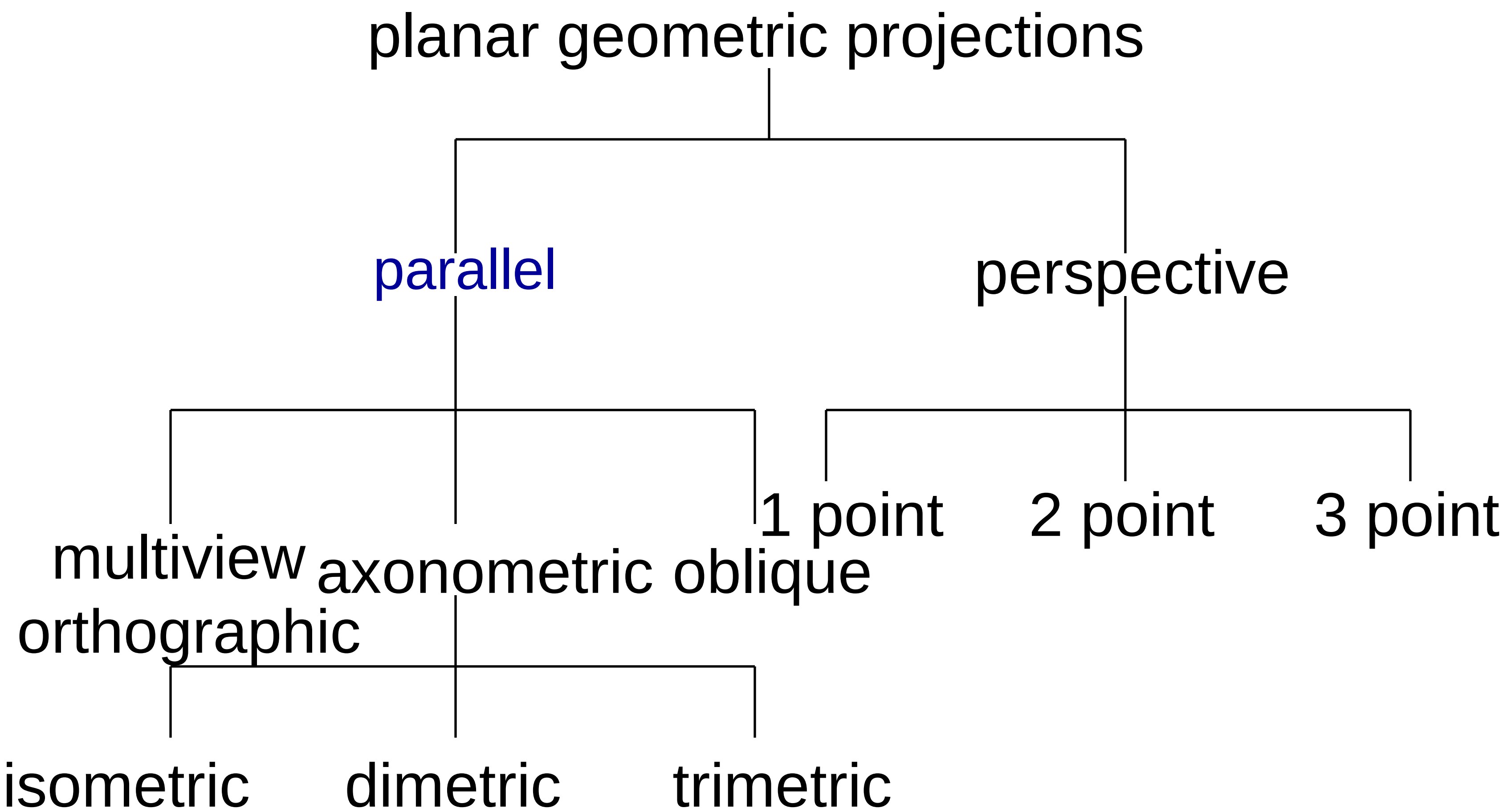
Classical Viewing

- Perspective Projection



Classical Viewing

- Taxonomy of Planar Geometric Projections



Orthographic Projection

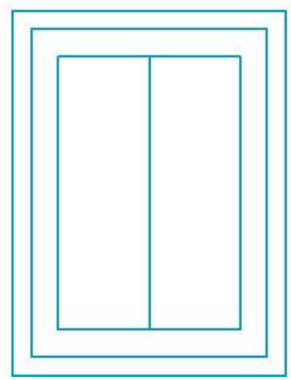
- Multiview Orthographic Projection
  - Projection plane parallel to principal face
  - Usually form front, top, side views

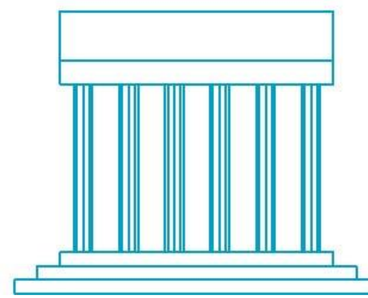
isometric (not multiview orthographic view)

front

in CAD and architecture, we often display three multiviews plus isometric

top





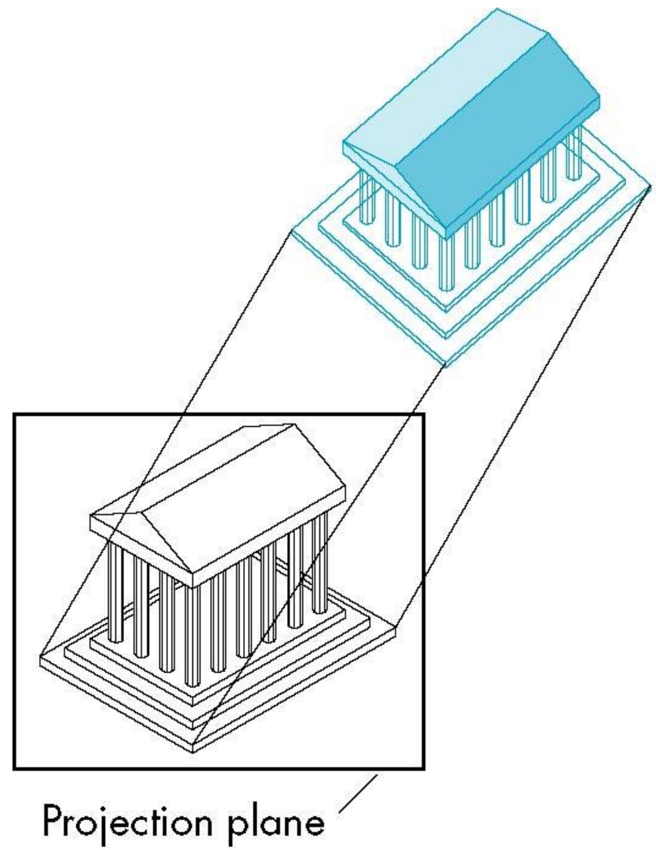
side

Orthographic Projection

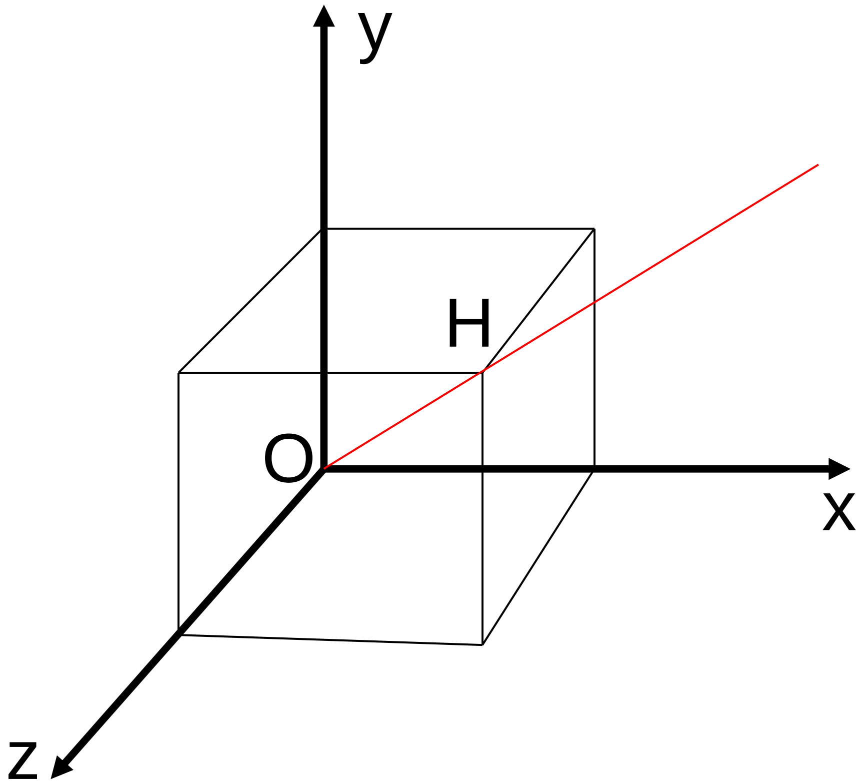
- Advantages
  - Preserves both distances and angles
    - Shapes preserved
    - Can be used for measurements
      - →Building plans
      - →Manuals
- Disadvantages
  - Cannot see what object really looks like because many surfaces hidden from view
    - Often we add the isometric

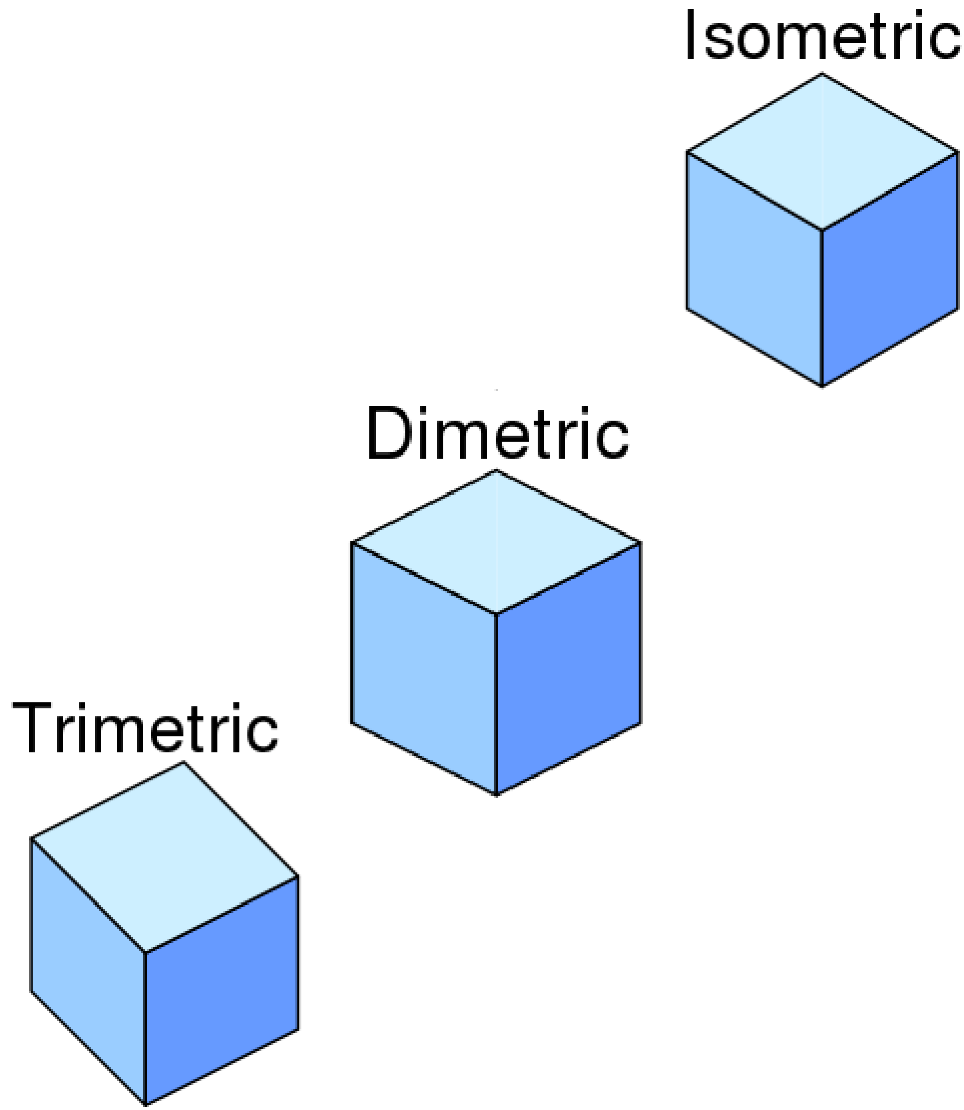
Axonometric Projections

- Allow projection plane to move relative to object
  - Classify by how many angles of a corner of a projected cube are the same
    - none: trimetric
    - two: dimetric
    - three: isometric(trục đo – đều)

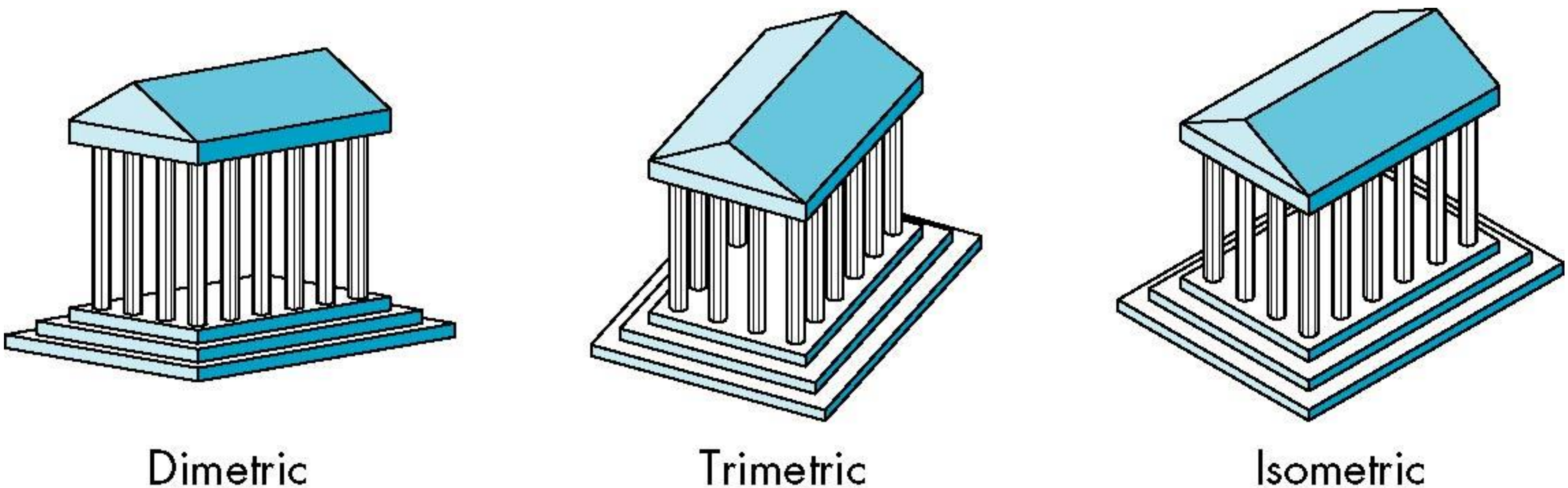


Axonometric Projections





Axonometric Projections

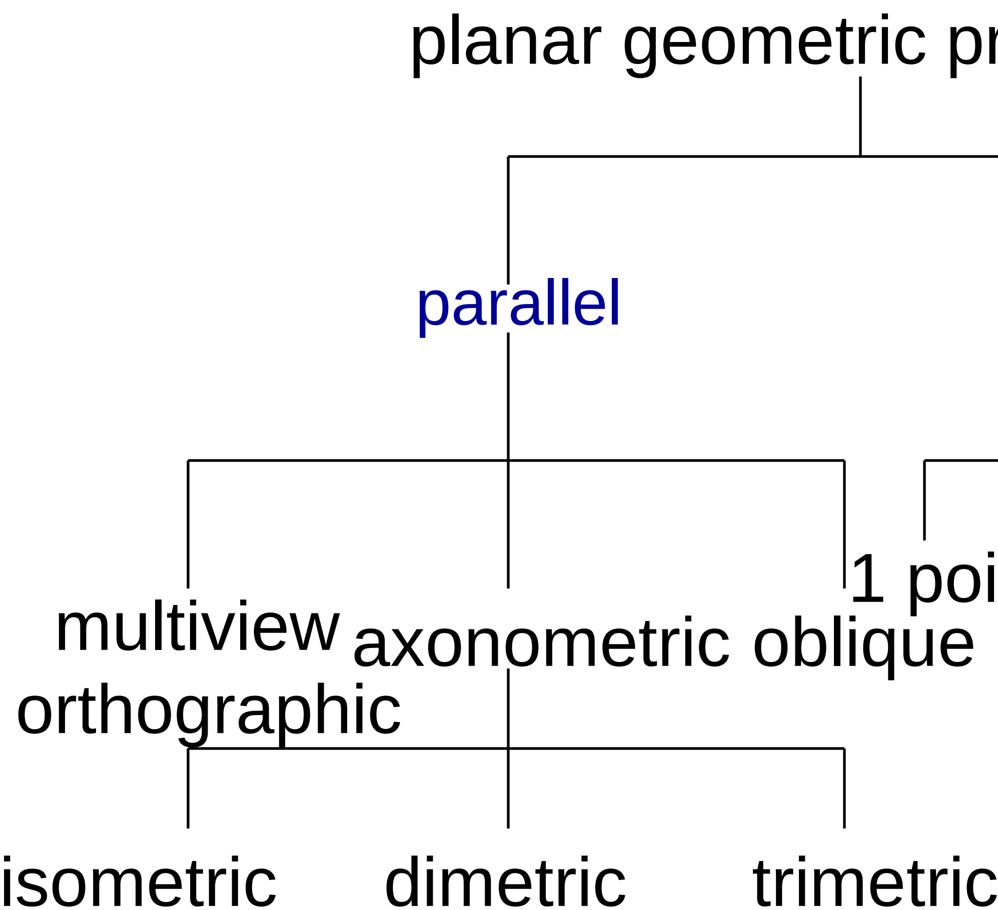


Axonometric Projections

- Lines are scaled (*foreshortened*) but can find scaling factors
- Lines preserved but angles are not
  - Projection of a circle in a plane not parallel to the projection plane is an ellipse
- Can see three principal faces of a box-like object
- Some optical illusions possible
  - Parallel lines appear to diverge
- Does not look real because far objects are scaled the same as near objects
- Used in CAD applications

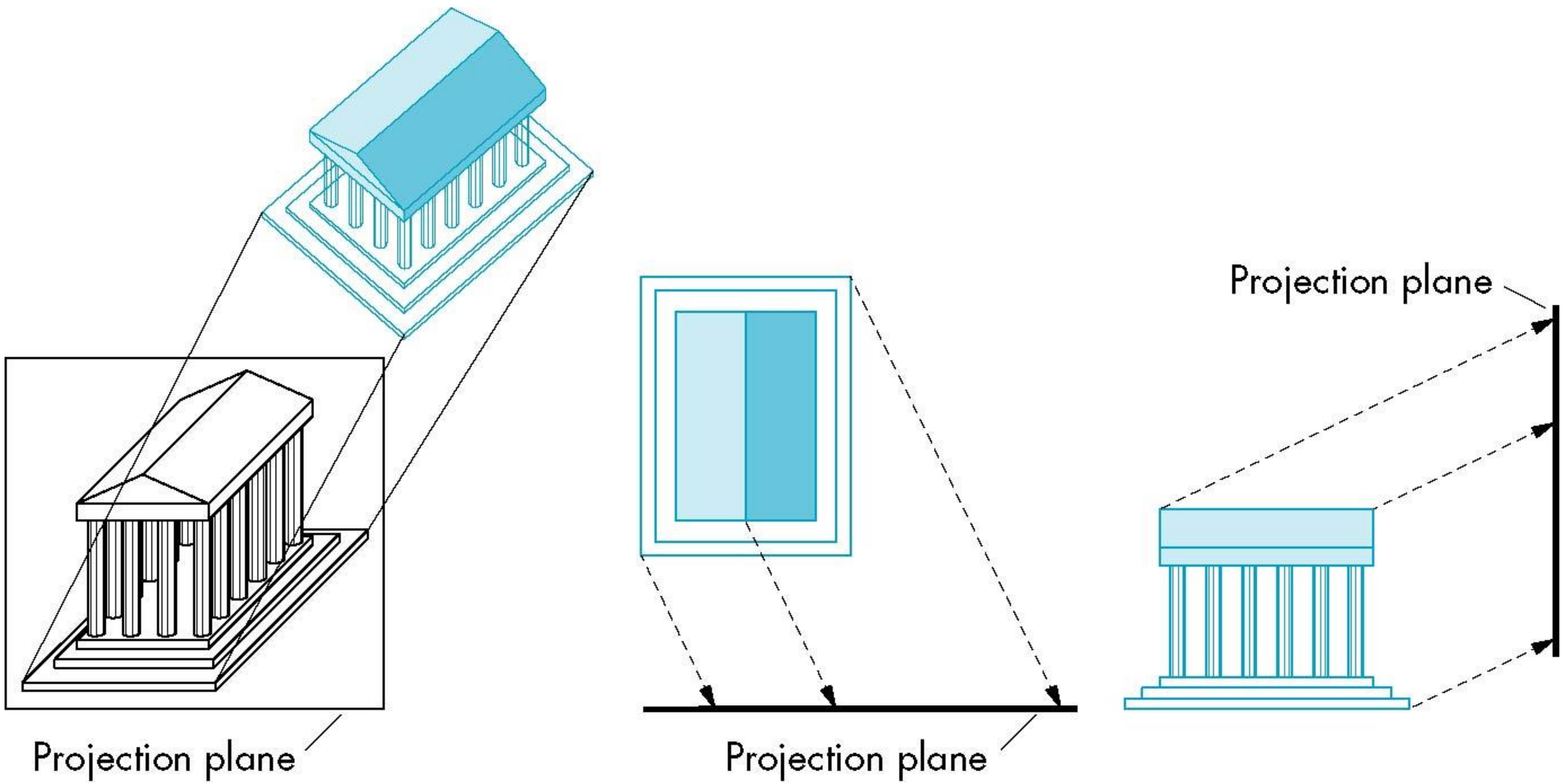
Classical Viewing

- Taxonomy of Planar Geometric Projections



Oblique Projection

- Arbitrary relationship between projectors and projection plane

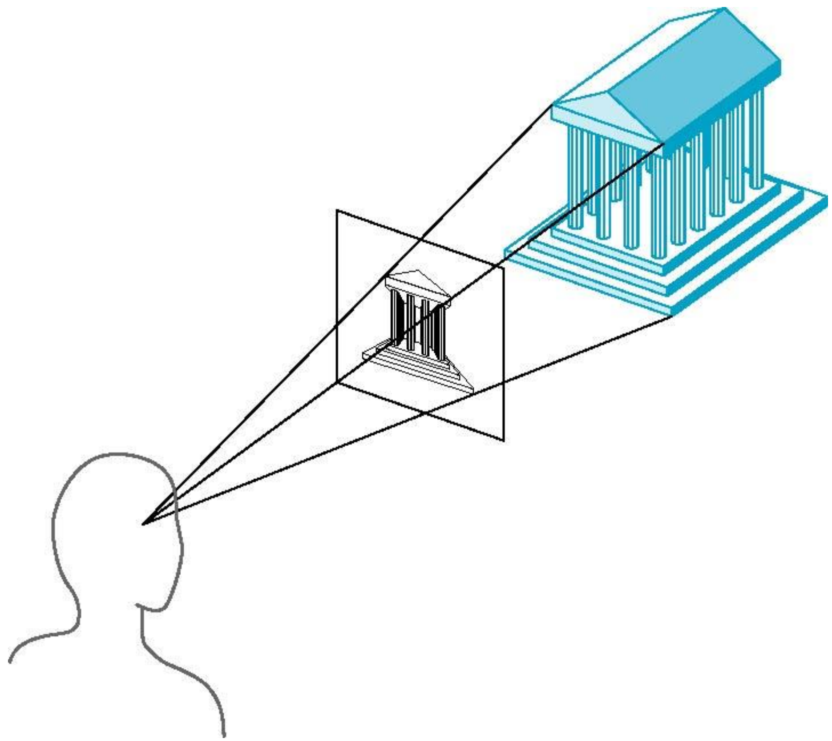


Oblique Projection

- Can pick the angles to emphasize a particular face
  - Architecture: plan oblique, elevation oblique
- Angles in faces parallel to projection plane are preserved while we can still see "around" side
- In physical world, cannot create with simple camera; possible with bellows camera or special lens (architectural)

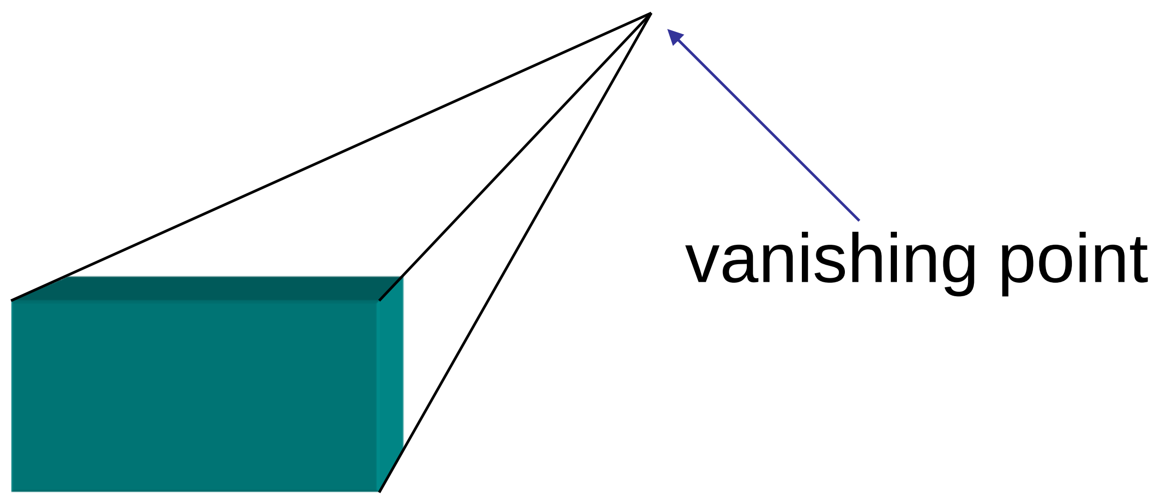
Perspective Projection

- Projectors coverge at center of projection



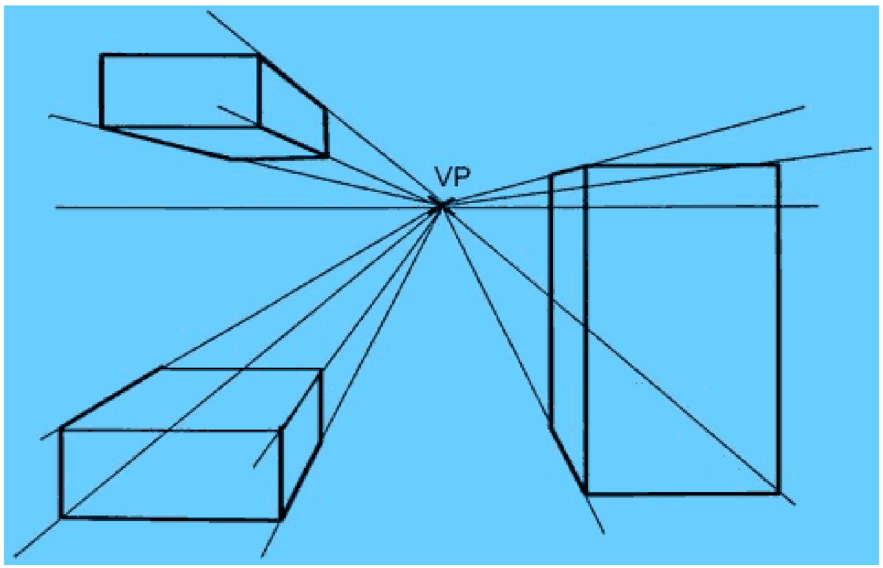
Perspective Projection

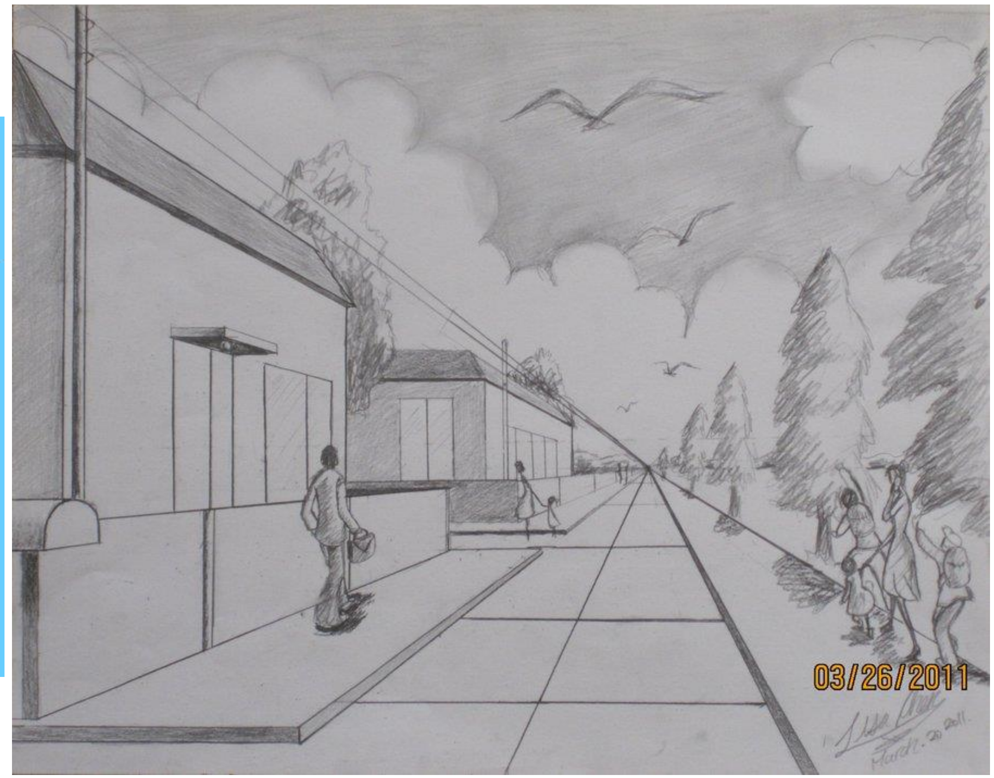
- Vanishing Points
  - Parallel lines (not parallel to the projection plan) on the object converge at a single point in the projection (the *vanishing point*)
  - Drawing simple perspectives by hand uses these vanishing point(s)



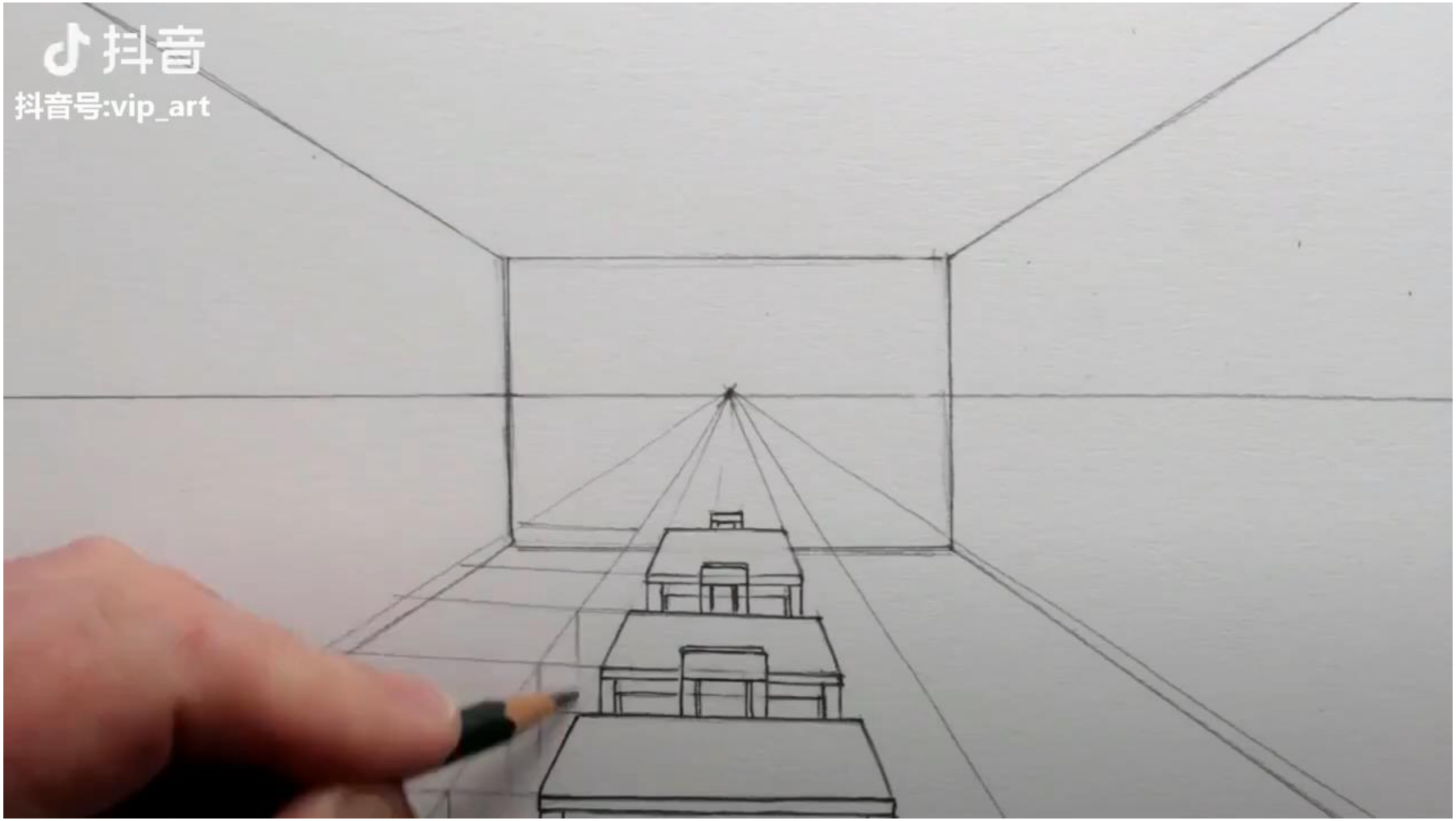
Perspective Projection

- One-Point Perspective
  - One principal face parallel to projection plane
  - One vanishing point for cube



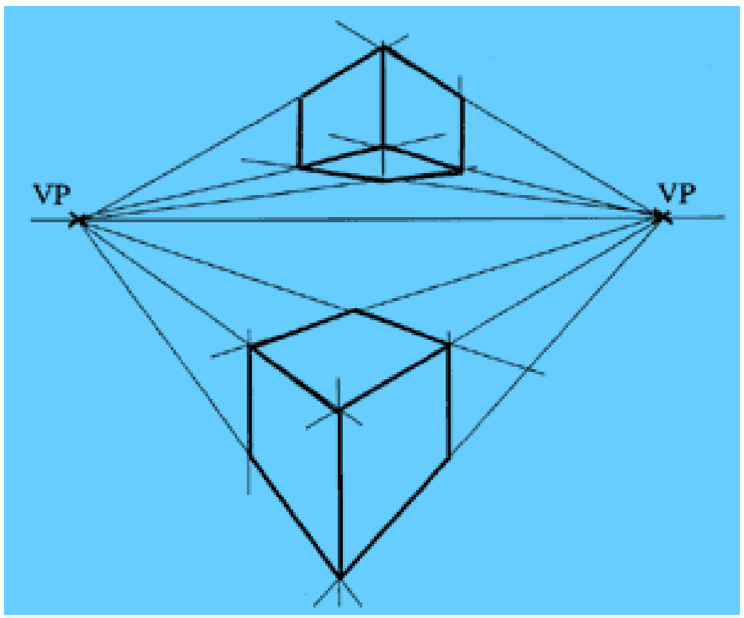


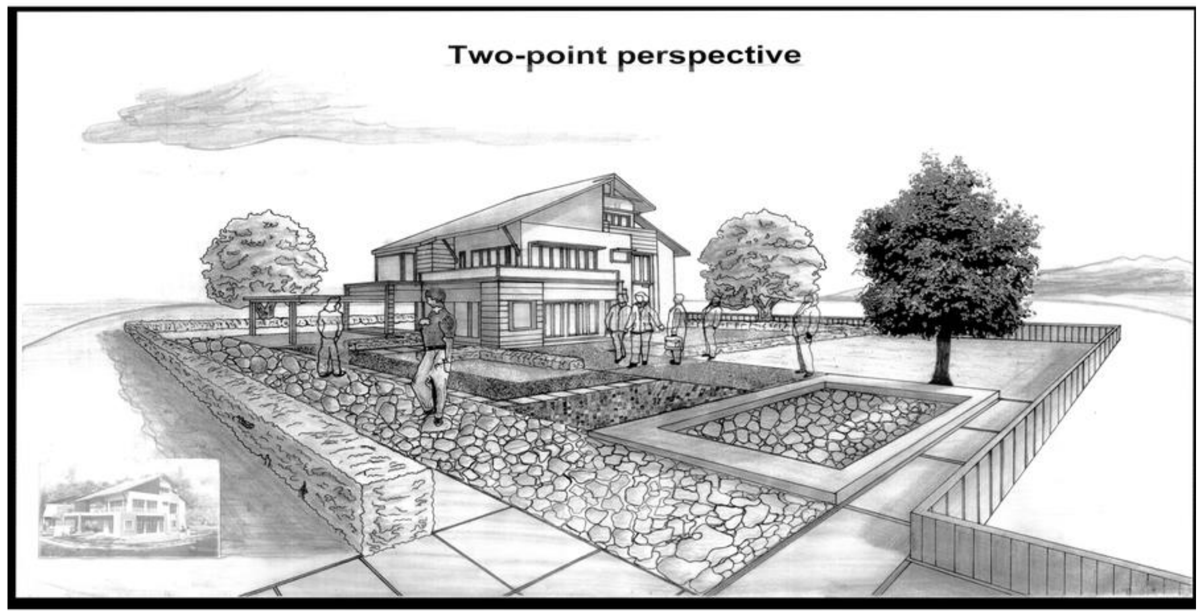
Perspective Projection



Perspective Projection

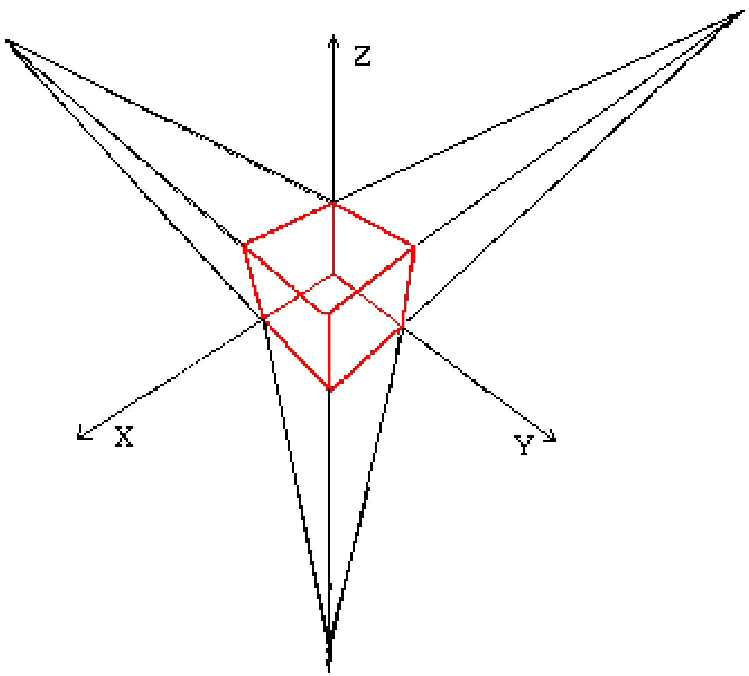
- Two-Point Perspective
  - On principal direction parallel to projection plane
  - Two vanishing points for cube

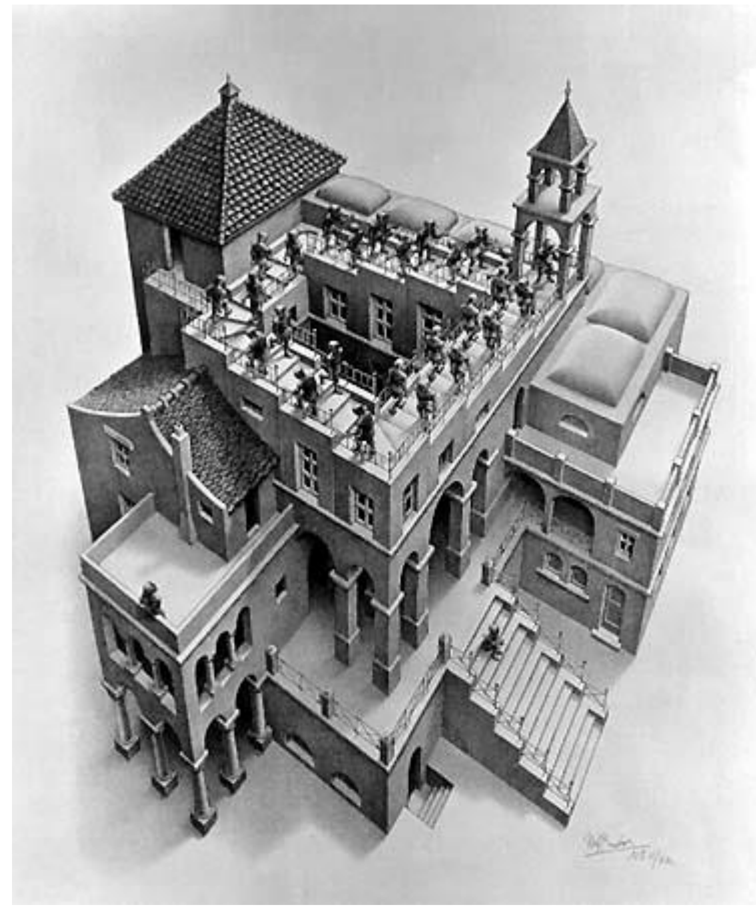




Perspective Projection

- Three-Point Perspective
  - No principal face parallel to projection plane
  - Three vanishing points for cube





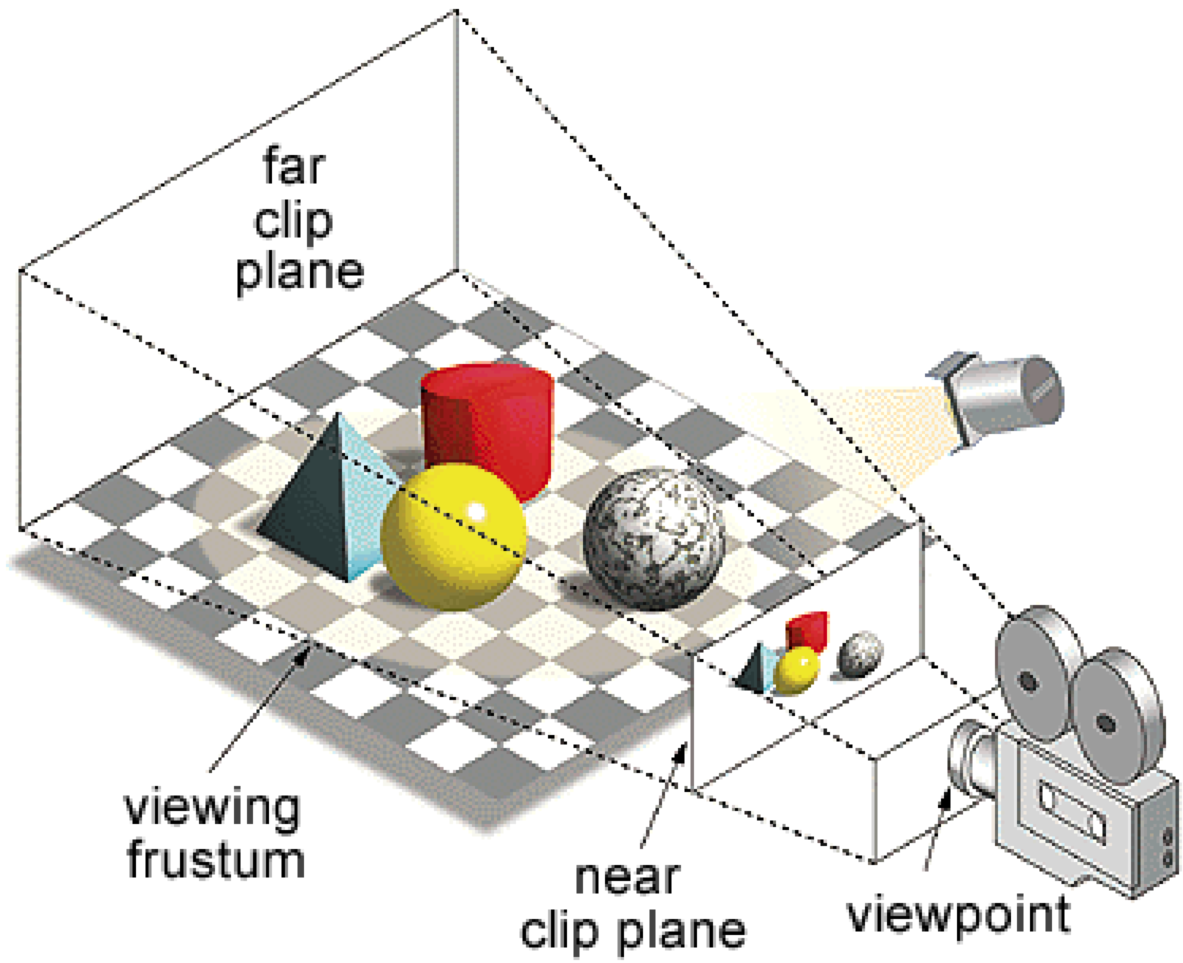
Perspective Projection

- Objects further from viewer are projected smaller than the same sized objects closer to the viewer (*diminution*)
  - Looks realistic
- Equal distances along a line are not projected into equal distances (*nonuniform foreshortening*)
- Angles preserved only in planes parallel to the projection plane
- More difficult to construct by hand than parallel projections (but not more difficult by computer)

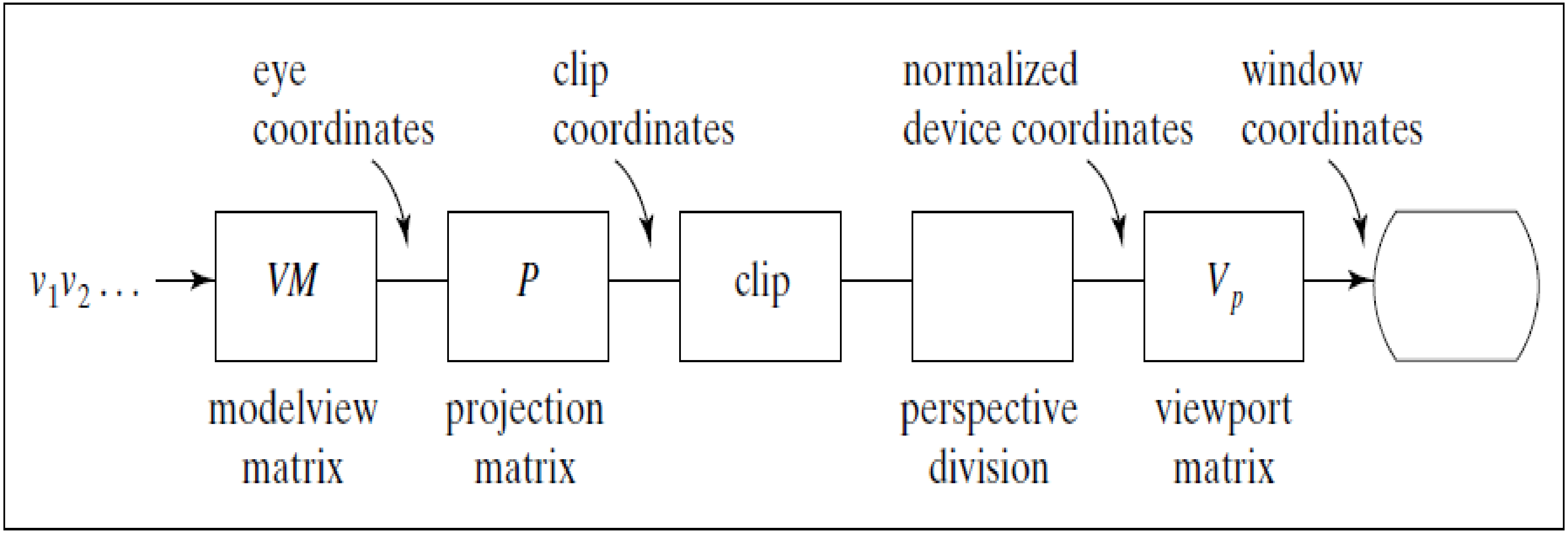
Computer Viewing

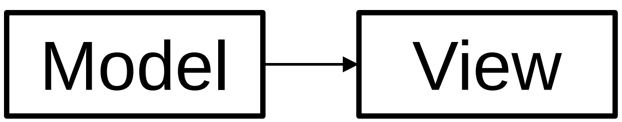
- There are two aspects of the viewing process, all of which are implemented in the pipeline,
  - Positioning the camera
    - Setting the model-view matrix
  - Selecting the view volume
    - Setting the projection matrix

Computer Viewing



Computer Viewing



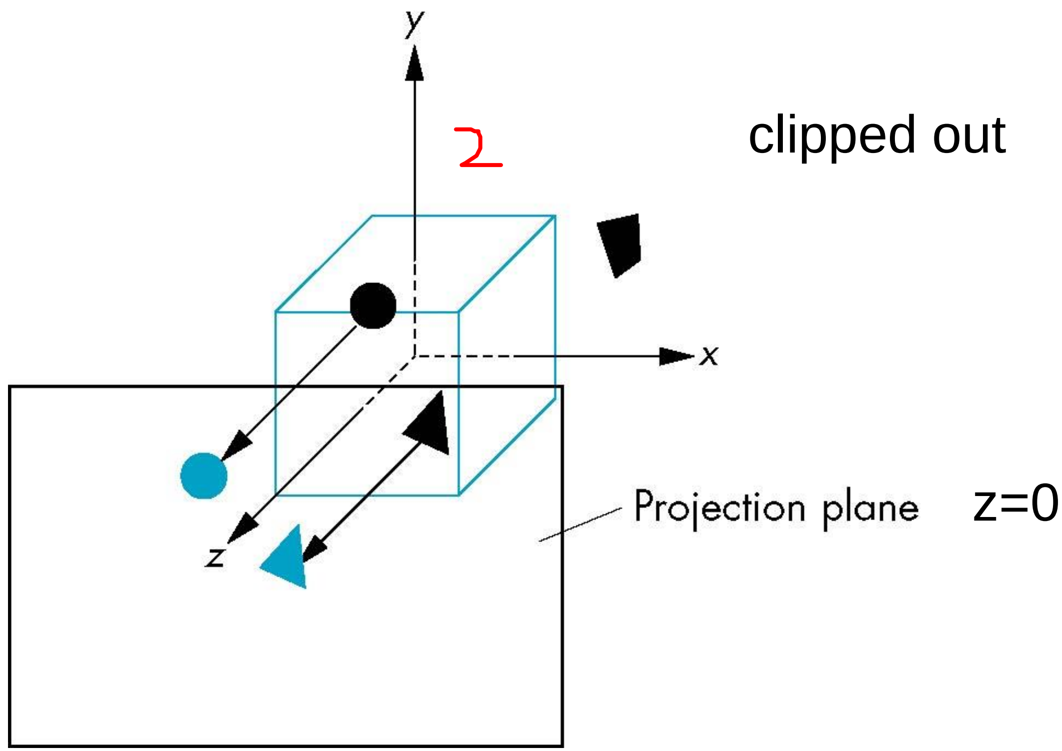


Computer Viewing

- In OpenGL, initially the object and camera frames are the same
  - Default model-view matrix is an identity
- The camera is located at origin and points in the negative z direction
- OpenGL also specifies a default view volume that is a cube with sides of length 2 centered at the origin
  - Default projection matrix is an identity

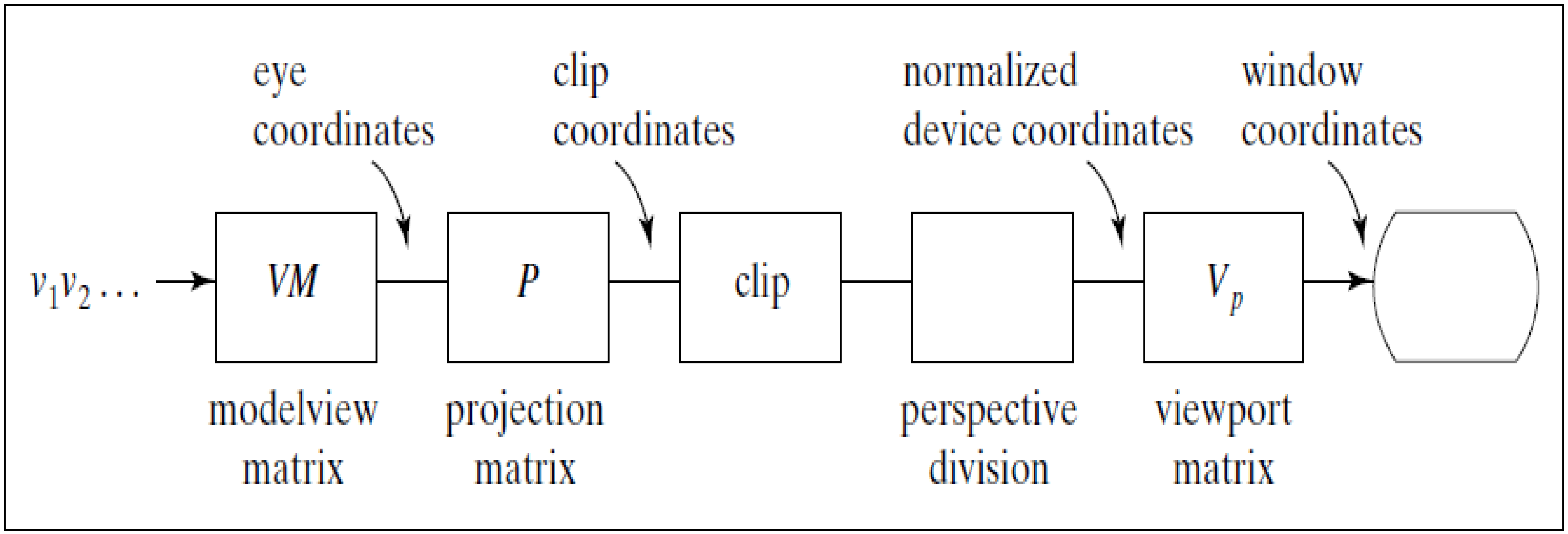
Computer Viewing

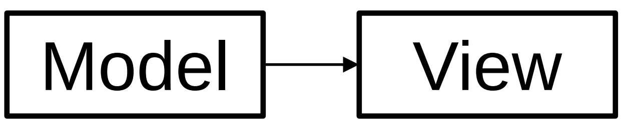
- Default projection is orthogonal



Computer Viewing

	Orthographic	Oblique	Perspective
Position, direction (V)	**gluLookAt**		
View Volume (P)	**glOrtho**		**glFrustum or gluPerspective**

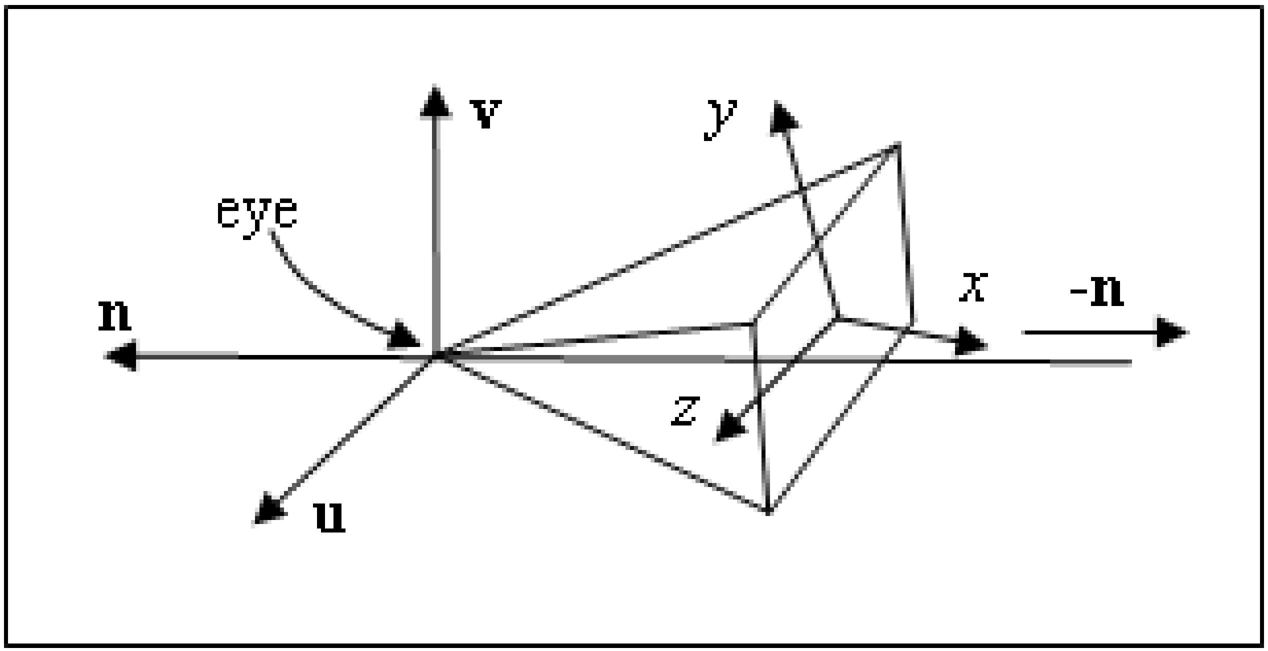




Computer Viewing

- Set up position & direction of camera

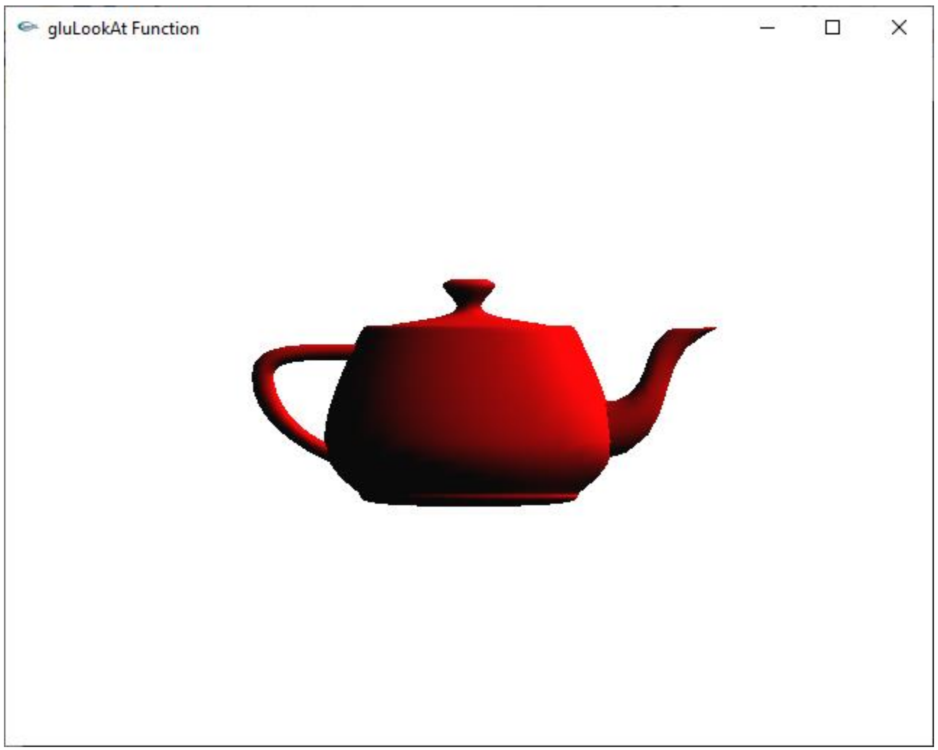
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(eye.x,eye.y,eye.z,look.x,look.y,look.z,
          up.x,u p.y,up.z);
```

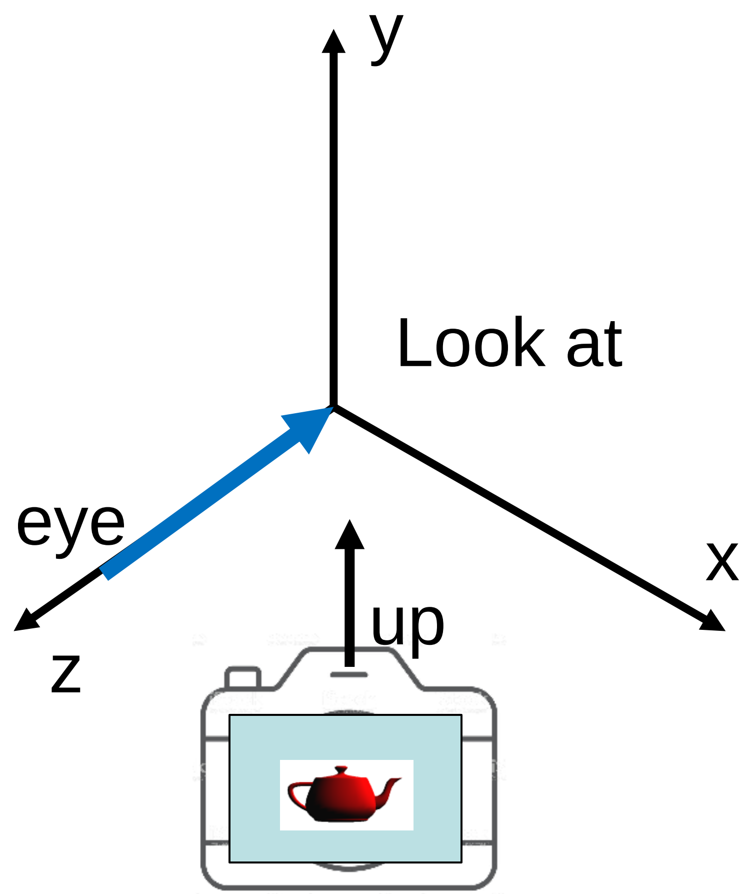


Computer Viewing

- Set up position & direction of camera

  gluLookAt(0, 0, 10, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

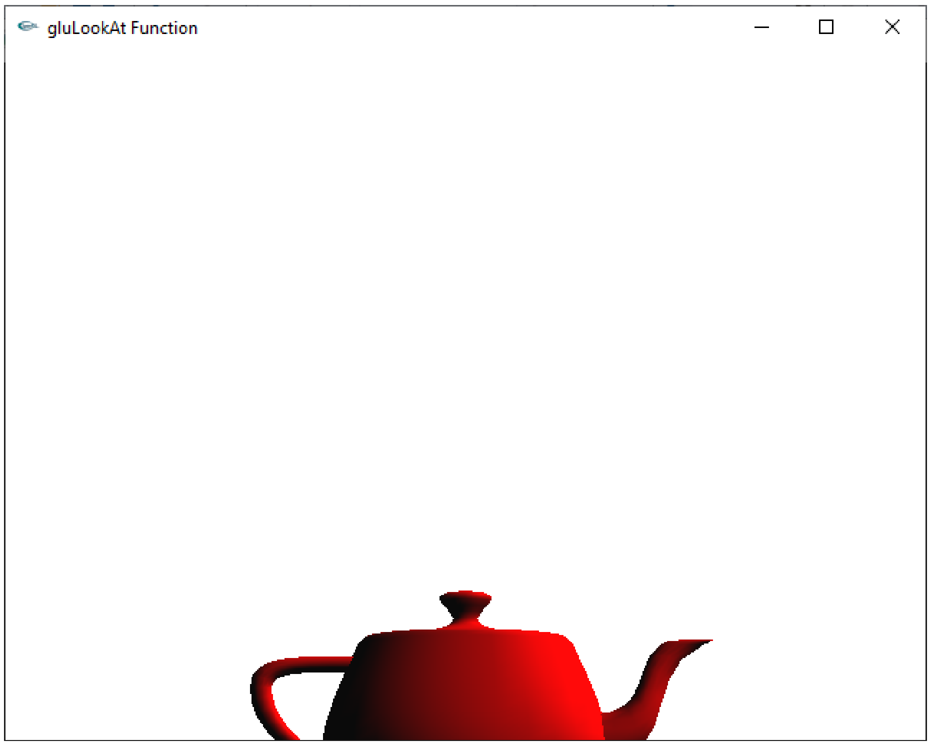


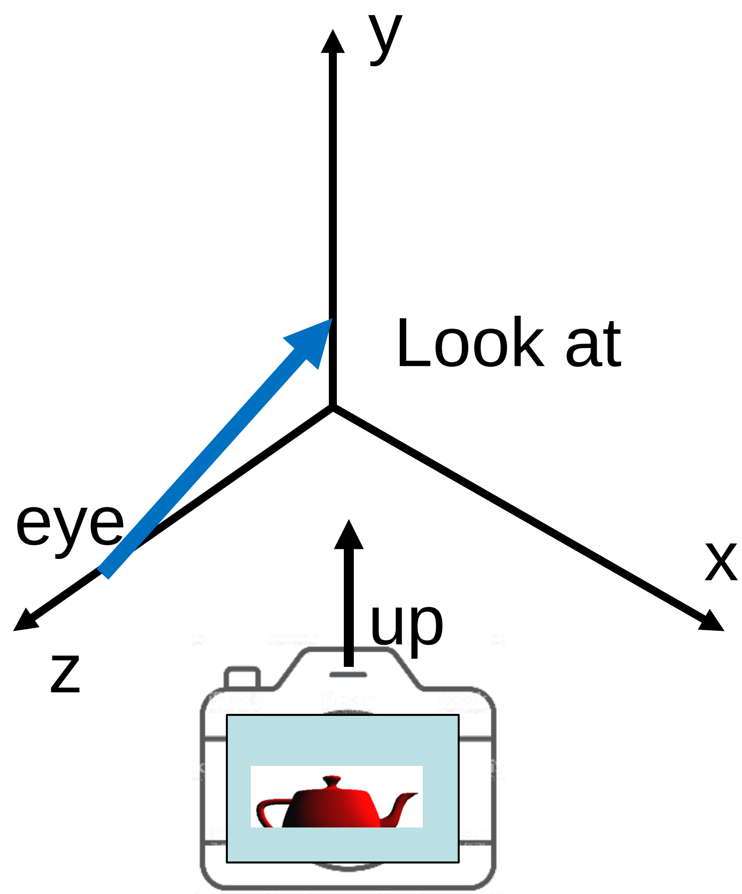


Computer Viewing

- Set up position & direction of camera

  gluLookAt(0, 0, 10, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0);

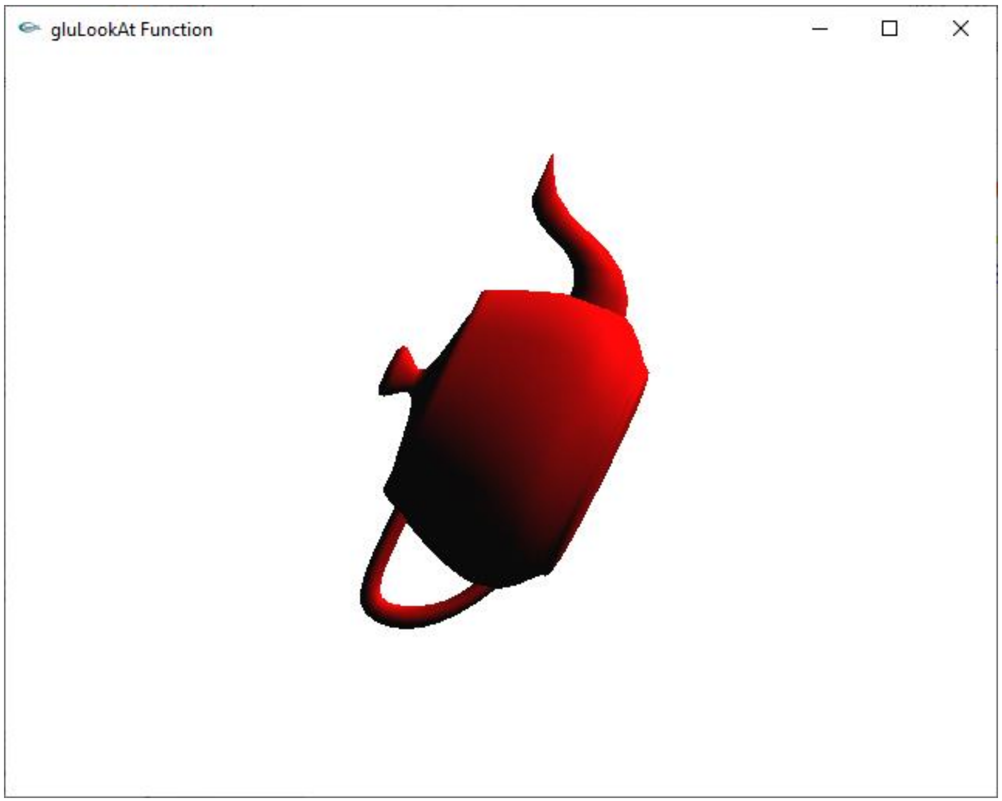


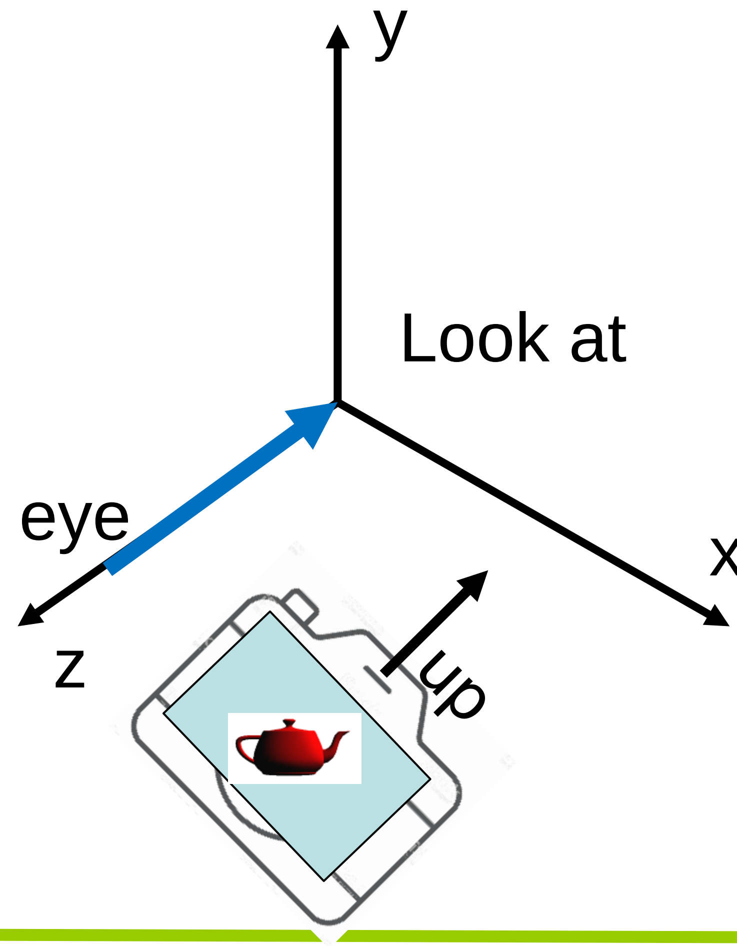


Computer Viewing

- Set up position & direction of camera

  gluLookAt(0, 0, 10, 0.0, 0.0, 0.0, 2.0, 1.0, 0.0);

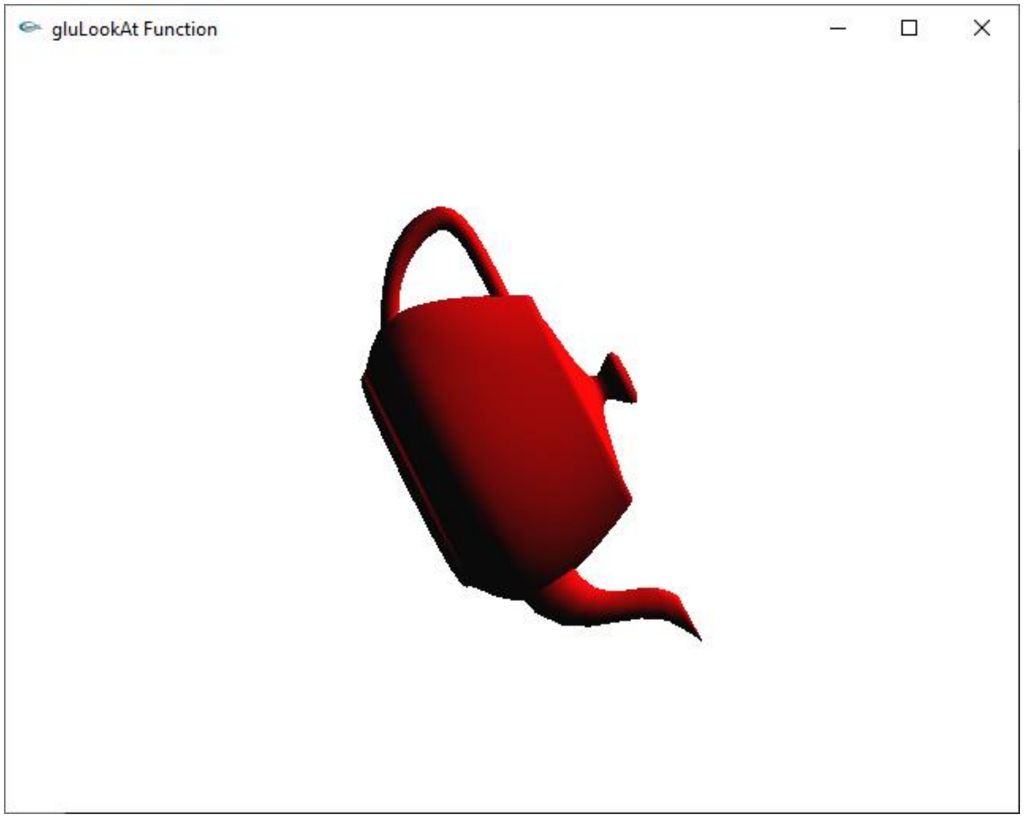


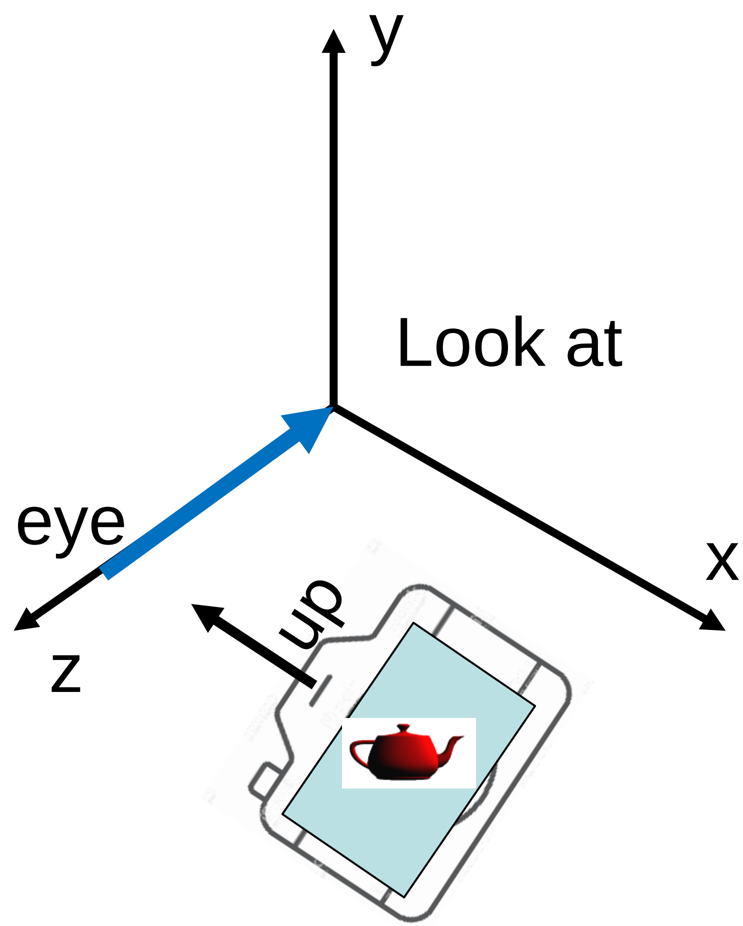


Computer Viewing

- Set up position & direction of camera

  gluLookAt(0, 0, 10, 0.0, 0.0, 0.0, -2.0, 1.0, 0.0);

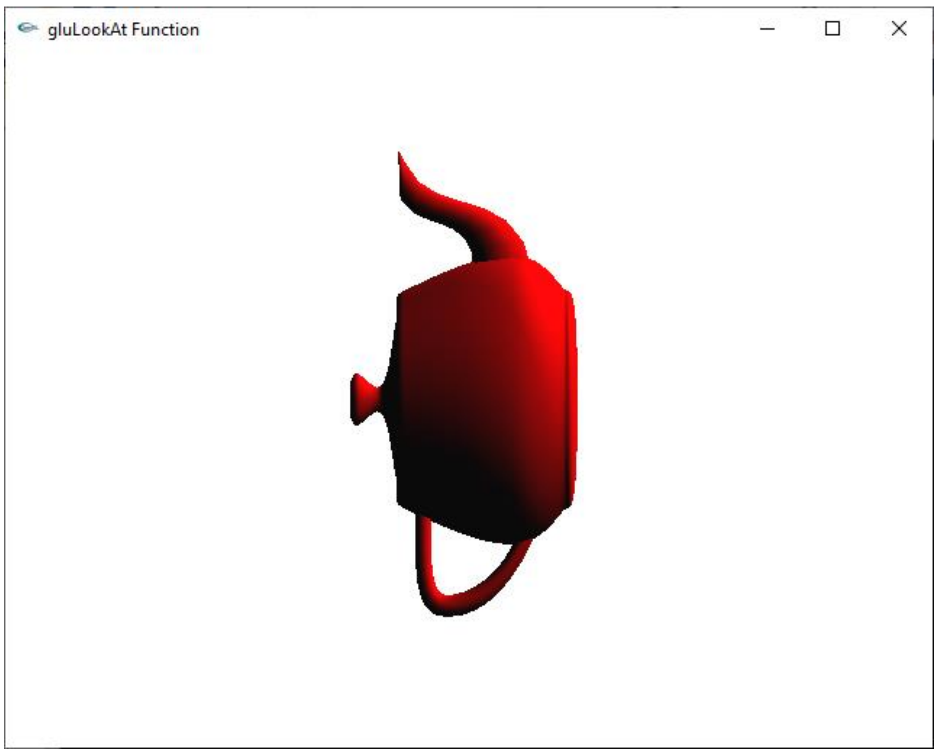


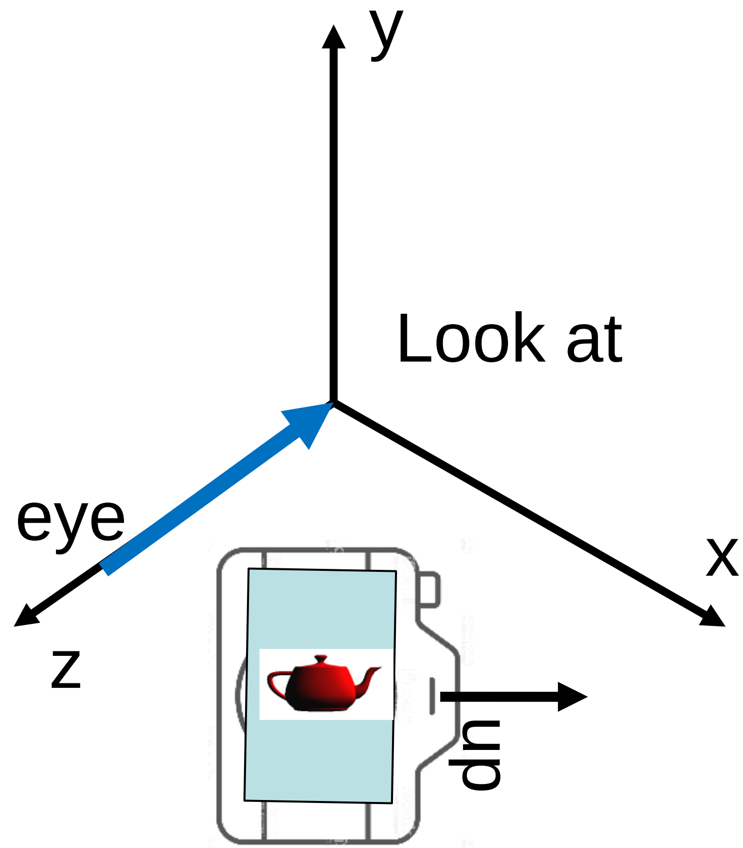


Computer Viewing

- Set up position & direction of camera

  gluLookAt(0, 0, 10, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0);

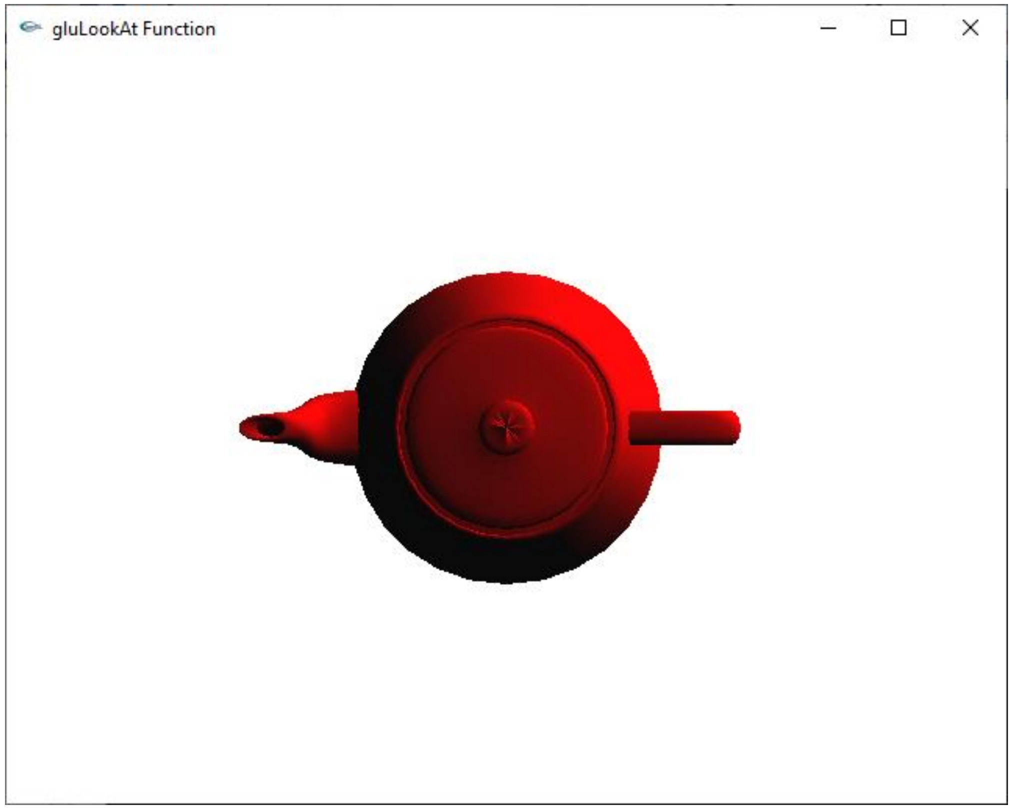




Computer Viewing

- Set up position & direction of camera

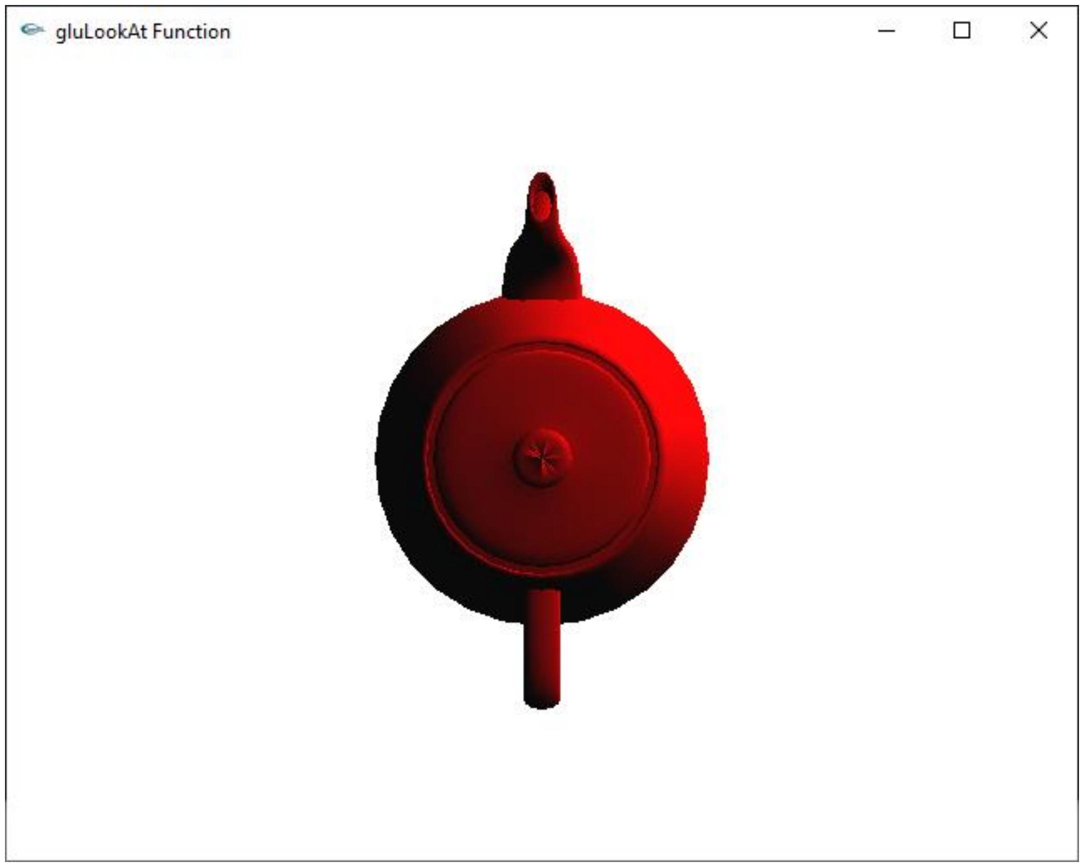
  gluLookAt(0, 10, 0, 0.0, 0.0, 0.0, 0, 0, 1);



Computer Viewing

- Set up position & direction of camera

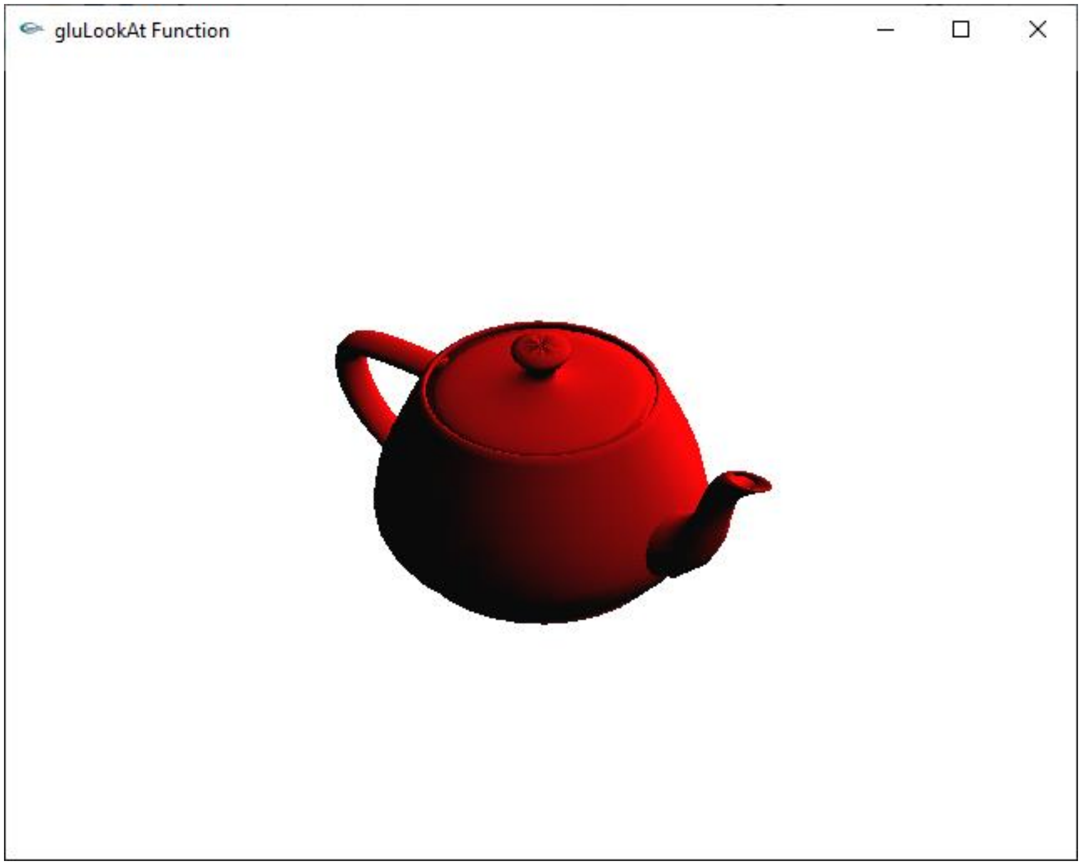
  gluLookAt(0, 10, 0, 0.0, 0.0, 0.0, 1, 0, 0);



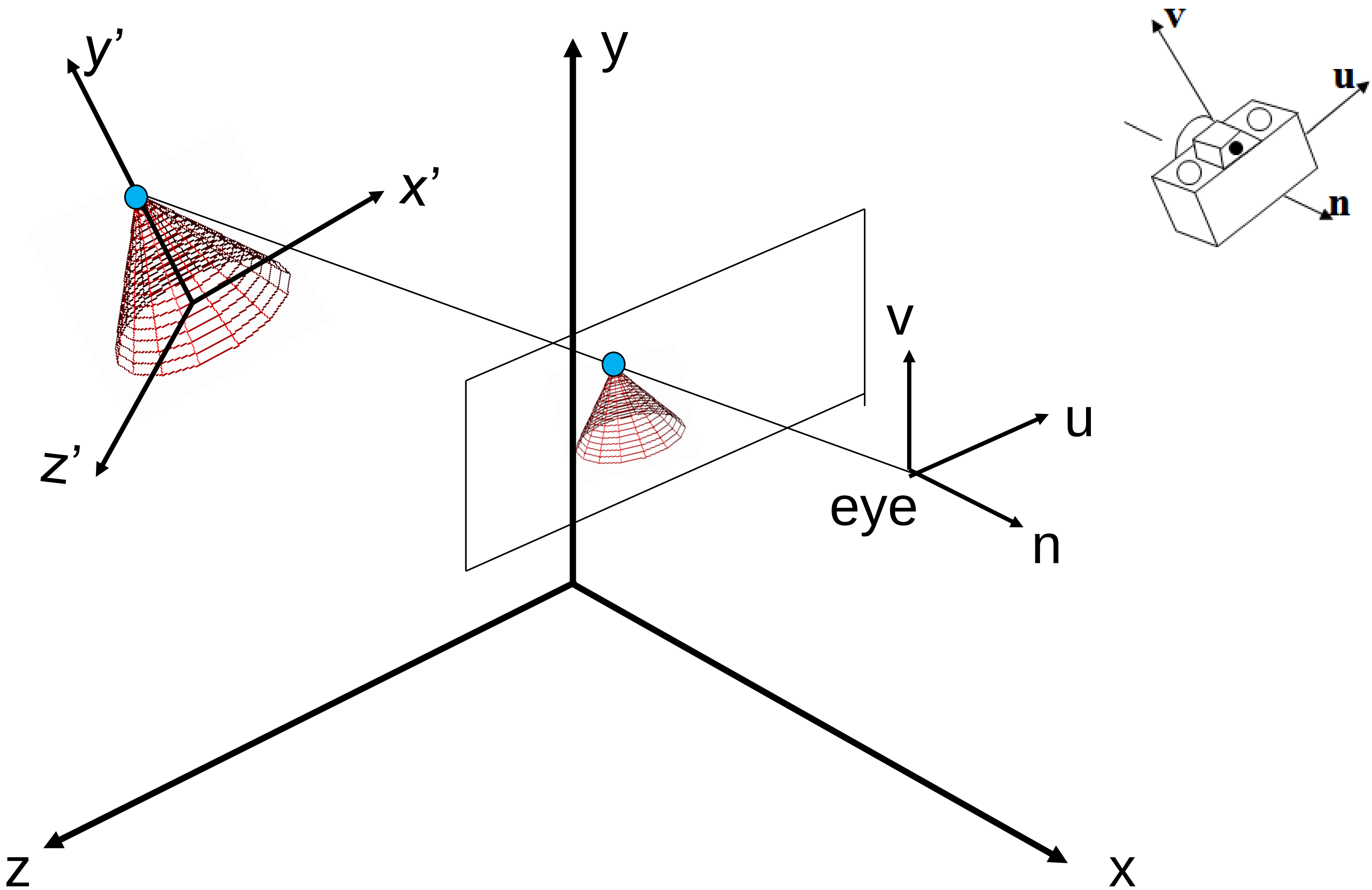
Computer Viewing

- Set up position & direction of camera

  gluLookAt(6, 7, 8, 0.0, 0.0, 0.0, 0, 1, 0);



Computer Viewing



Computer Viewing

- Orthogonal projection
  - The default projection in the eye (camera) frame is orthogonal
  - For points within the default view volume

$$x_p = x$$
$$y_p = y$$
$$z_p = 0$$

Computer Viewing

Orthogonal projection

default orthographic projection

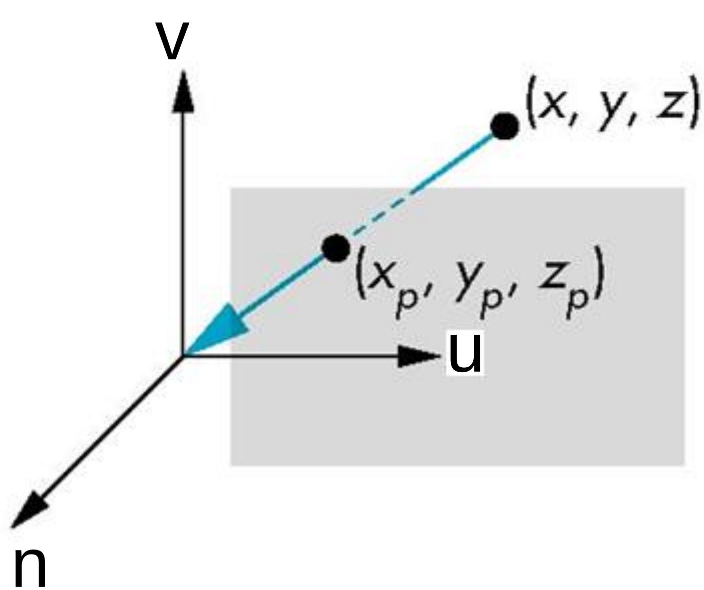
$x_p = x$
$y_p = y$
$z_p = 0$
$w_p = 1$

$$\mathbf{p}_p = \mathbf{Mp}$$

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

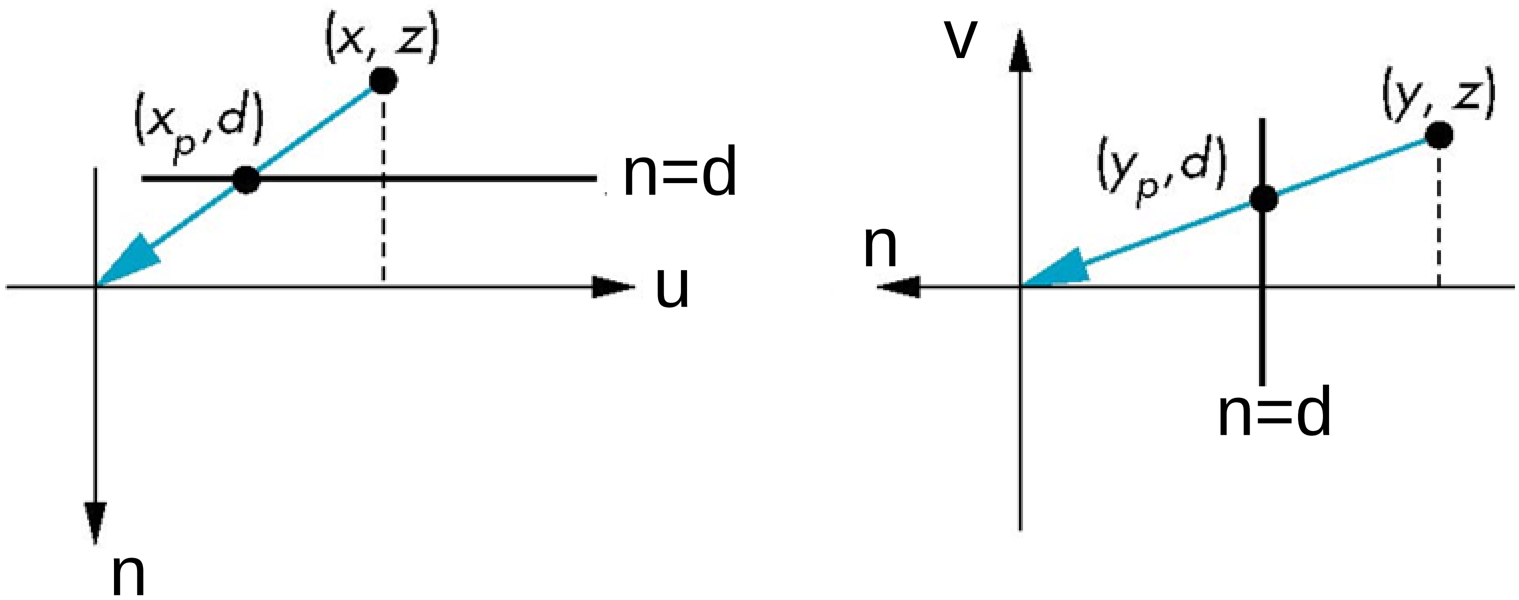
Computer Viewing

- Perspective Projection
  - Center of projection at the origin
  - Projection plane $n = d$, $d < 0$



Computer Viewing

- Perspective Projection
  - Consider top and side views



$$x_p = \frac{x}{z/d} \qquad y_p = \frac{y}{z/d} \qquad z_p = d$$

Computer Viewing

- Perspective Projection

consider $\mathbf{q} = \mathbf{Mp}$ where $\quad \mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \Rightarrow \mathbf{q} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

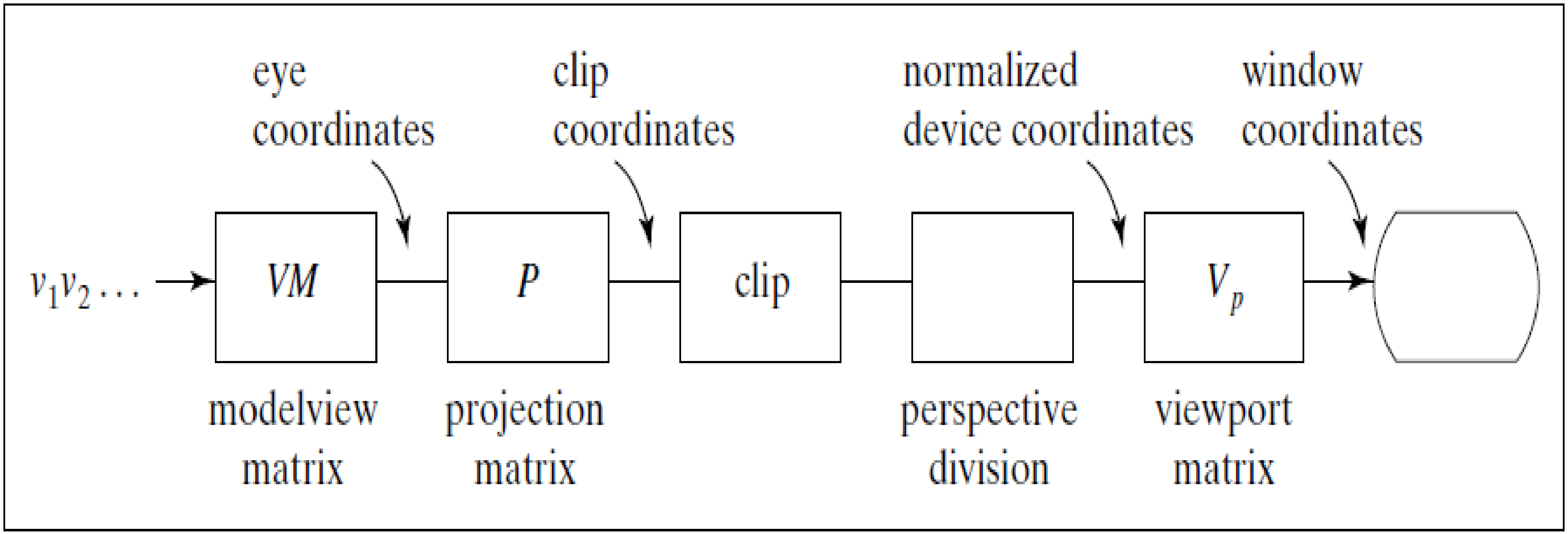
Computer Viewing

- Perspective Projection
  - However $w \neq 1$, so we must divide by w to return from homogeneous coordinates
  - This *perspective division* yields

$$x_p = \frac{x}{z/d} \qquad y_p = \frac{y}{z/d} \qquad z_p = d$$

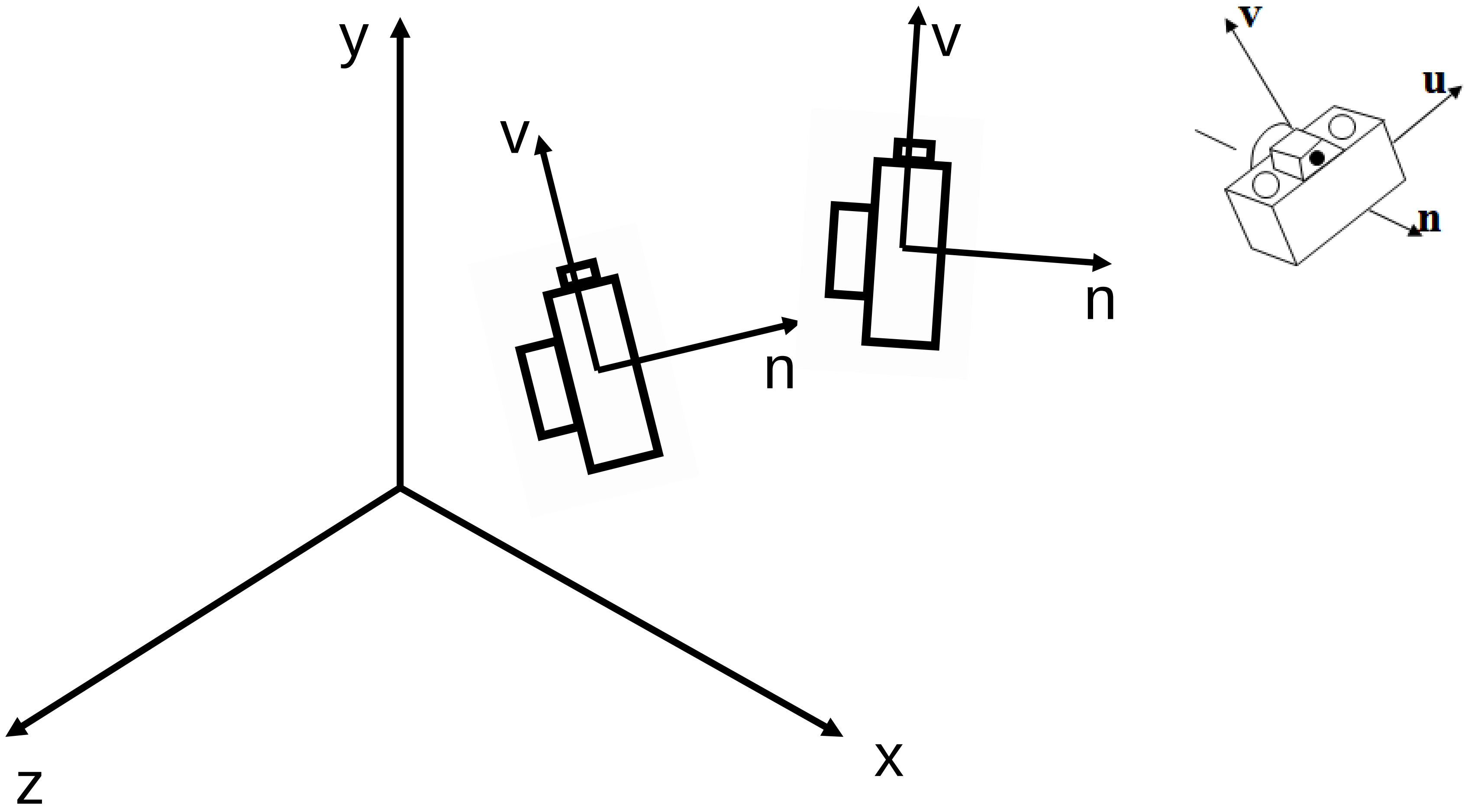
the desired perspective equations

Computer Viewing



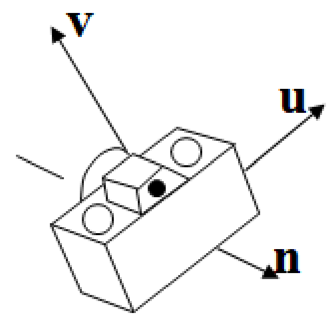
Computer Viewing

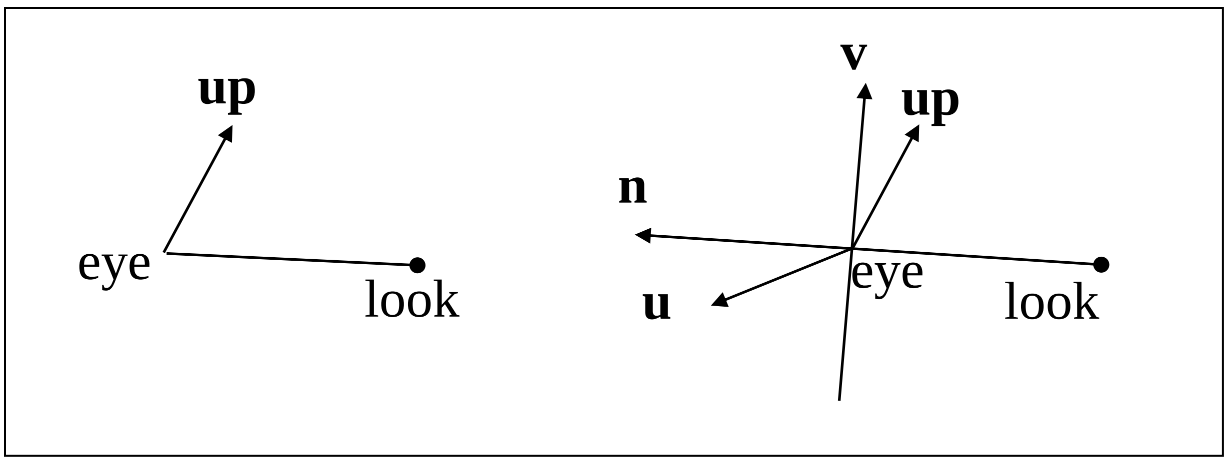
```
gluLookAt(eye.x,eye.y,eye.z,look.x,look.y,look.z,
          up.x,u p.y,up.z);
```



Computer Viewing

- Matrix transfrom from wordld frame to camera frame
  - eye, look at, up → u, v, n
  - **n** = eye – look.
  - **u** = **up**×**n**,
  - **v** = **n**×**u**
  - **u, v, n : unit vector**





Computer Viewing

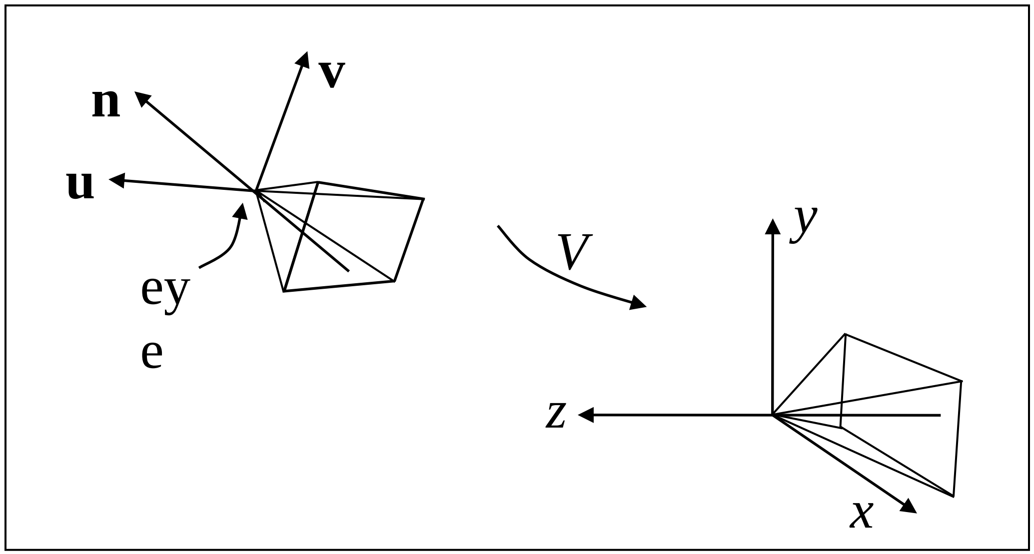
- Matrix transfrom from world frame to camera frame

  CTM = V.M

$$V = \begin{pmatrix} u_x & u_y & u_z & d_x \\ v_x & v_y & v_z & d_y \\ n_x & n_y & n_z & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

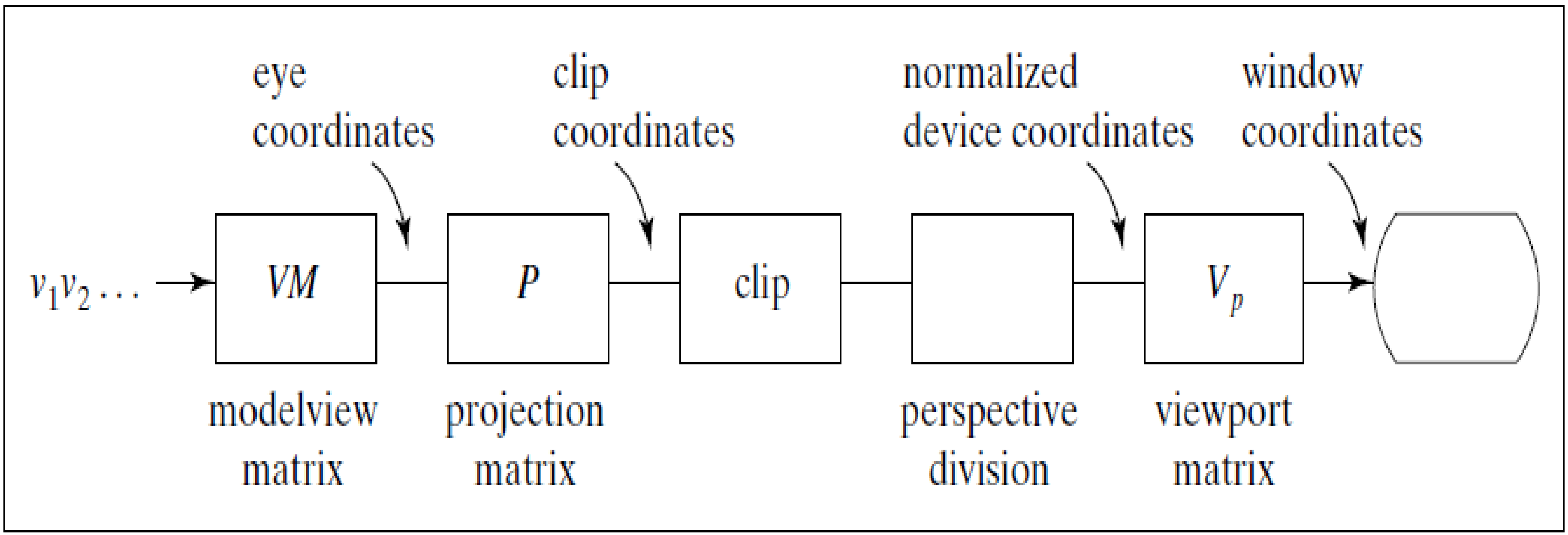
$(d_x, d_y, d_z) = (-eye \bullet \mathbf{u}, -eye \bullet \mathbf{v}, -eye \bullet \mathbf{n})$

$$V\begin{pmatrix} eye_x \\ eye_y \\ eye_z \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad V\begin{pmatrix} u_x \\ u_y \\ u_z \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$



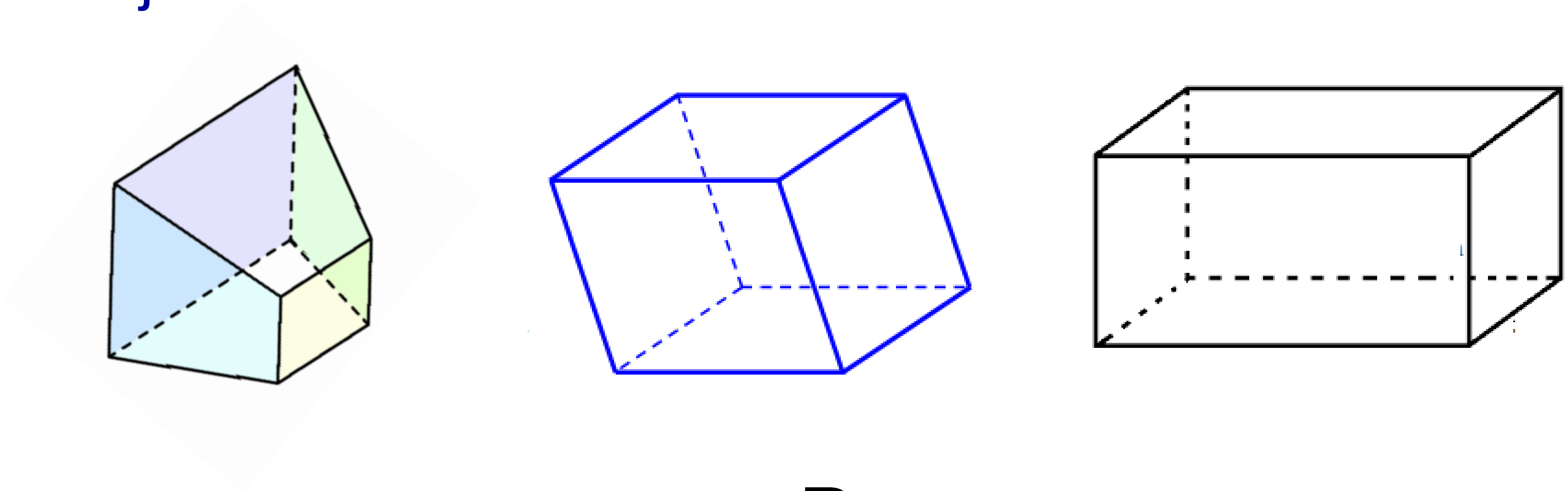
View Volume

	Orthographic	Oblique	Perspective
Position, direction (V)		**gluLookAt**	
View Volume (P)	**glOrtho**		**glFrustum or gluPerspective**



View Volume

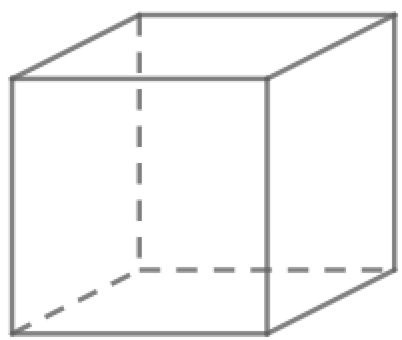
- Projection Matrix P



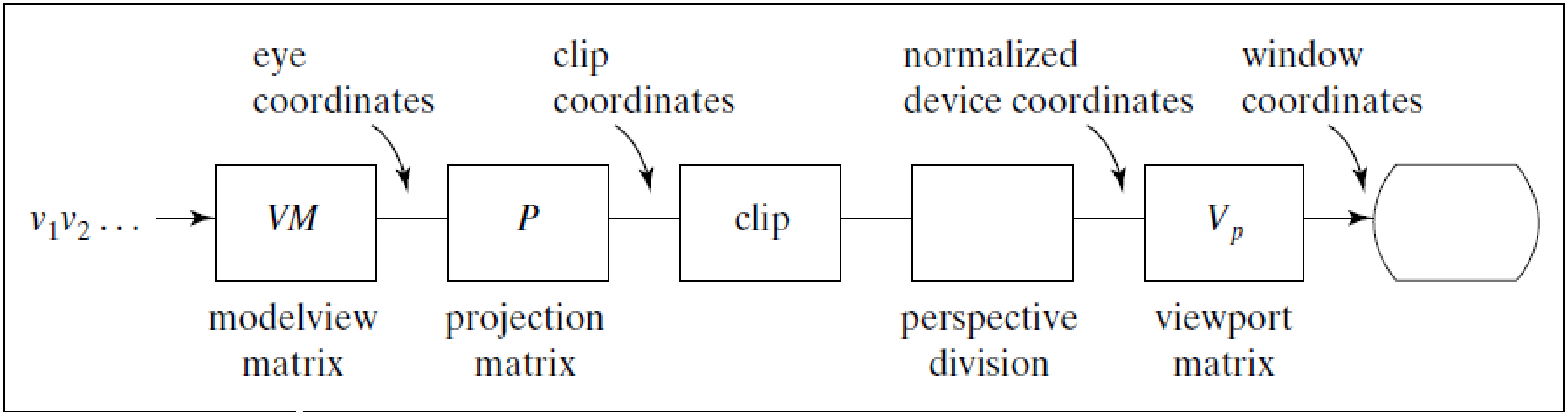
× P

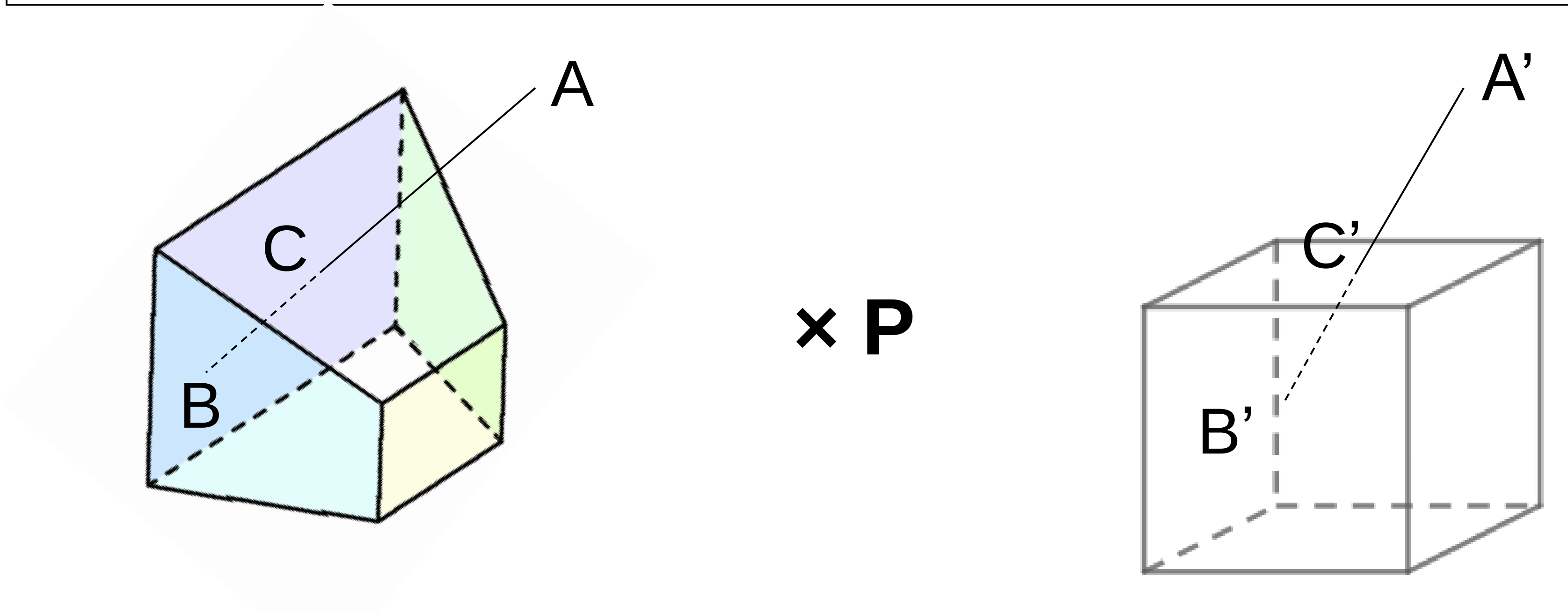
View Volume: CCV

Ortho. Projection

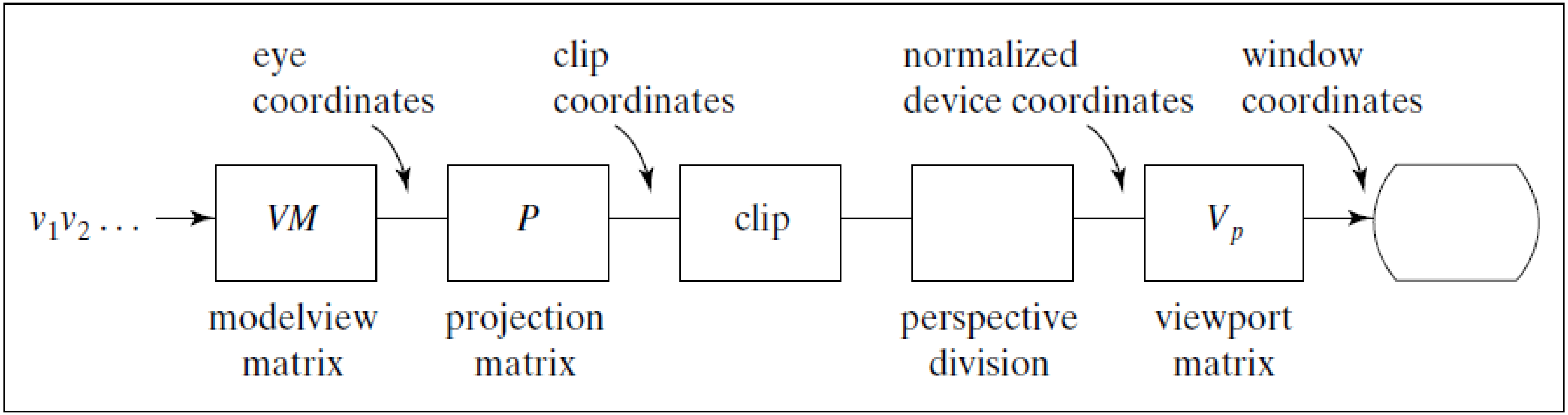


View Volume


eye
coordinates
clip
coordinates
normalized
device coordinates
window
coordinates
$v_1 v_2 \ldots$
VM
P
clip
V_p
modelview
matrix
projection
matrix
perspective
division
viewport
matrix


A
C
B
× P
A'
C'
B'

View Volume

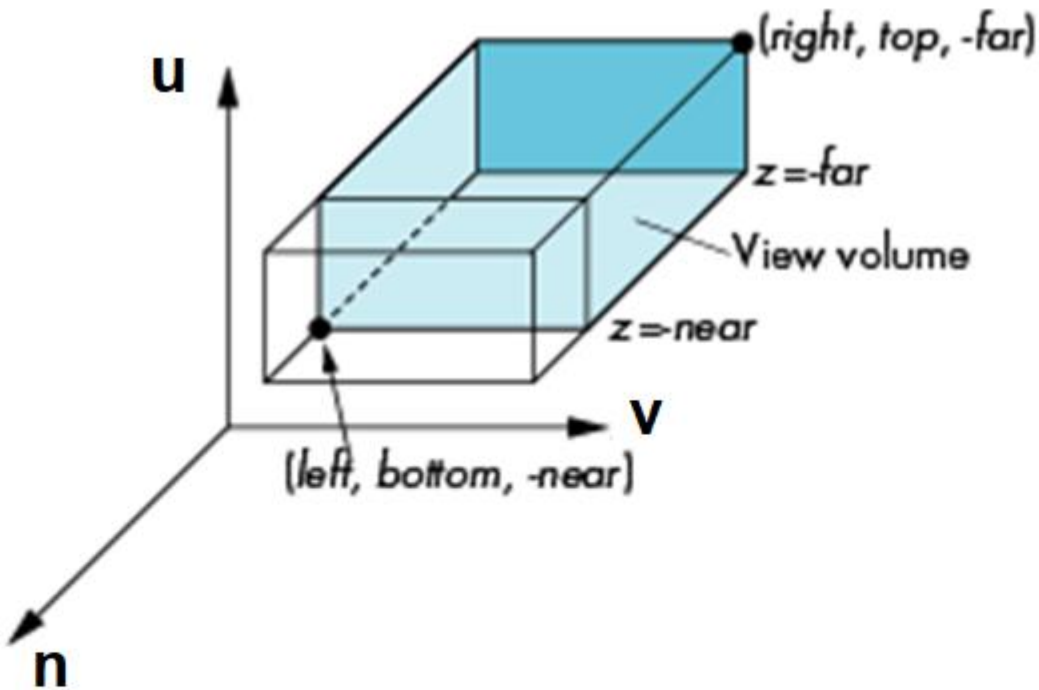


- Perspective division: divide by w component

  (x, y, z, w) → (x/w, y/w, z/w, 1)

View volume

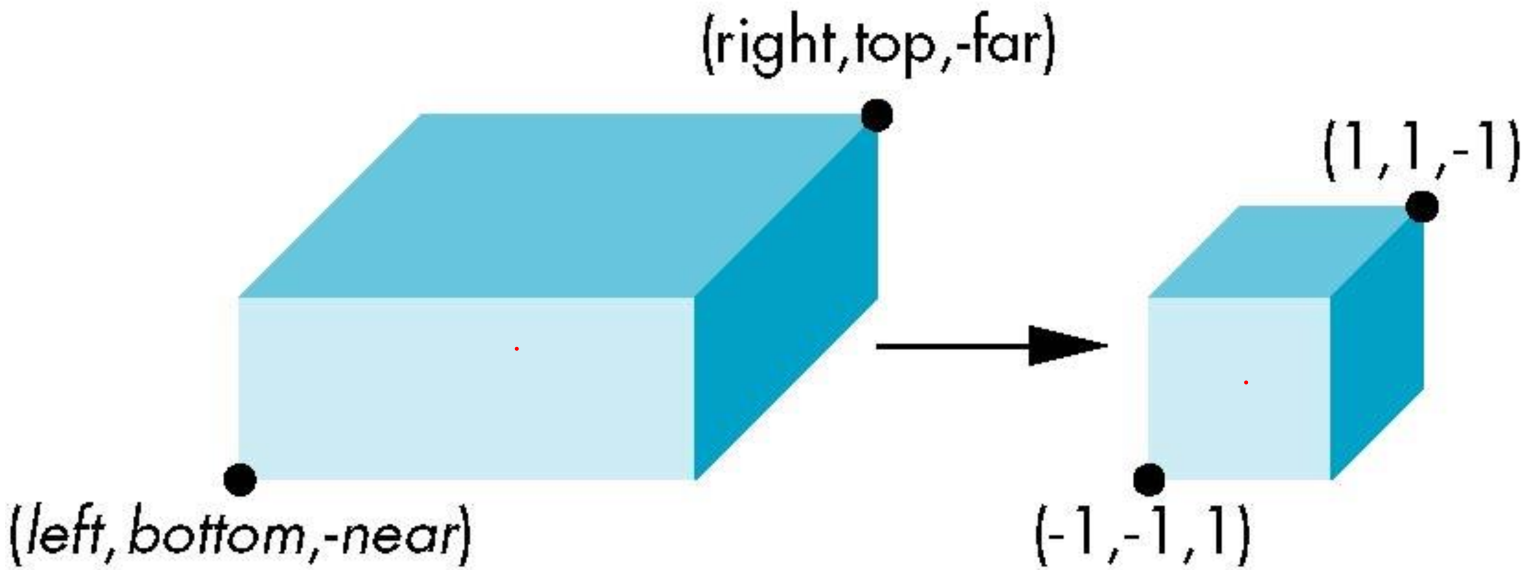
- Orthogonal projection
  - **`glOrtho(left,right,bottom,top,near,far)`**
  - **`near`** and **`far`** measured from camera



Orthogonal Projection

`glOrtho(left,right,bottom,top,near,far)`

normalization ⇒ find transformation to convert specified clipping volume to default



Flip: -far → 1; -near → 1

0 < near < far

Orthogonal Projection

- Two steps
  - Move center to origin

    T(-(left+right)/2, -(bottom+top)/2,(near+far)/2))
  - Scale to have sides of length 2

    S(2/(right-left),2/(top-bottom),2/(near-far))

$$\mathbf{P} = \mathbf{ST} = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & \frac{2}{near-far} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthogonal Projection

- Set up projection matrix in the pipeline
  - Method 1

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-1.2, 1.2, -1.2, 1.2, 0.1, 100);
```

Orthogonal Projection

- Set up projection matrix in the pipeline
  - Method 2

$$\begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & \frac{2}{near-far} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
float   m[16];
……………………..//Calculate m
glMatrixMode(GL_PROJECTION);
glLoadMatrixf(m);
```

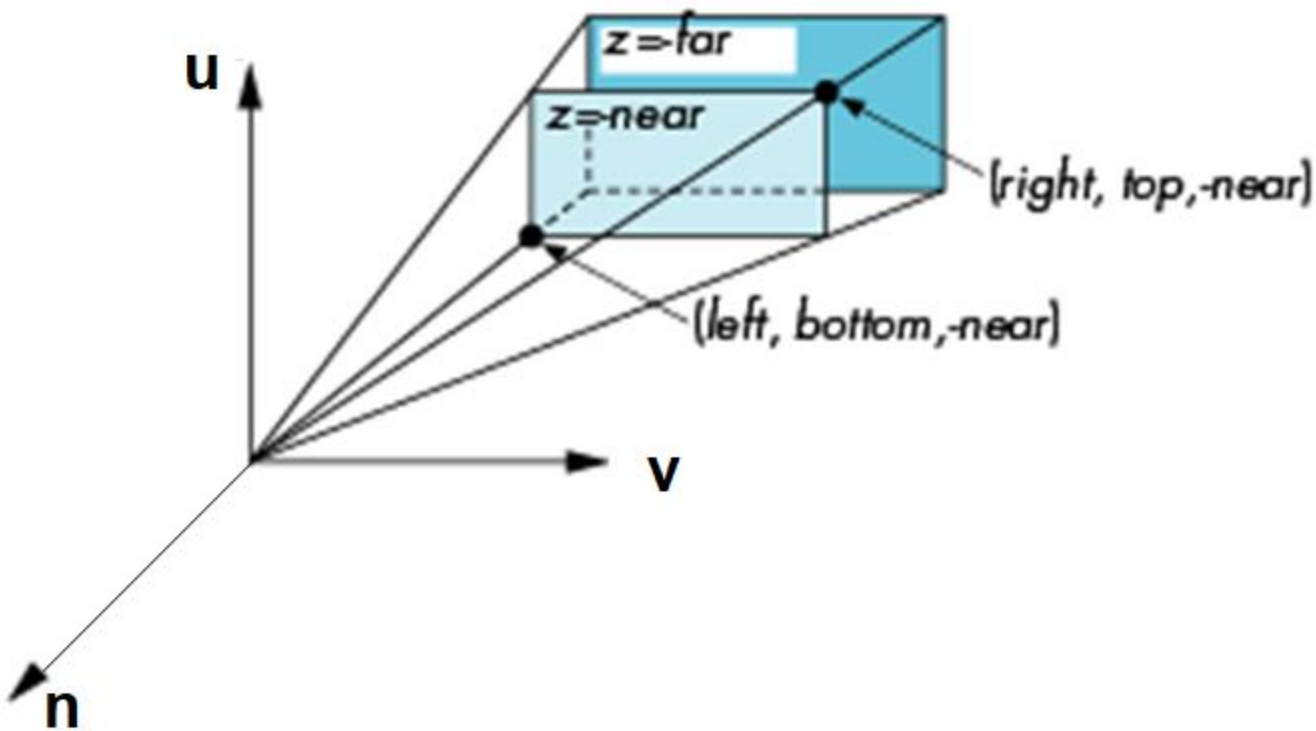
Orthogonal Projection

- Set up projection matrix in the pipeline
  - Method 3

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
……………. //Calculate S matrix
glMultMatrixf(s);
……………. //Calculate T matrix
glMultMatrixf(t);
```

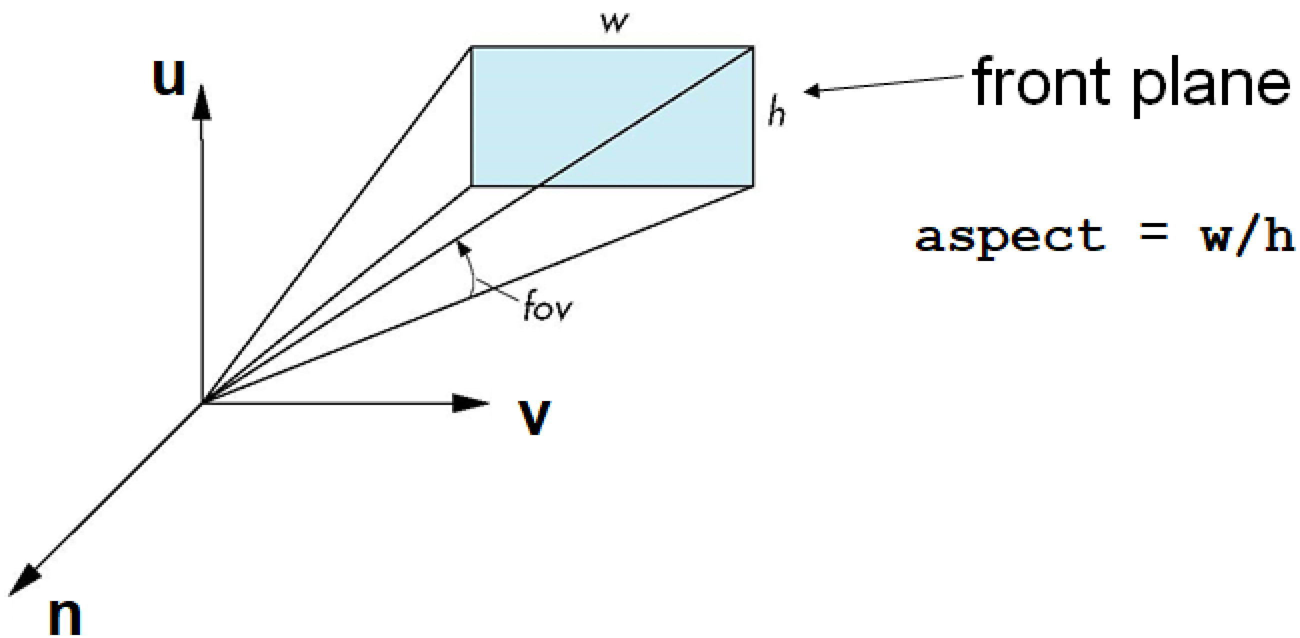
Perspective Projection

- Perspective Projection
  - `glFrustum(left,right,bottom,top,near,far)`



Perspective Projection

- Perspective Projection
  - With **`glFrustum`** it is often difficult to get the desired view
  - **`gluPerpective(fovy, aspect, near, far)`** often provides a better interface



aspect = w/h

Perspective Projection

$$P = \begin{bmatrix} \frac{2 \times near}{right - left} & 0 & \frac{right + left}{right - left} & 0 \\ 0 & \frac{2 \times near}{top - bottom} & \frac{top + bottom}{top - bottom} & 0 \\ 0 & 0 & -\frac{far + near}{far - near} & -\frac{2 \times far \times near}{far - near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

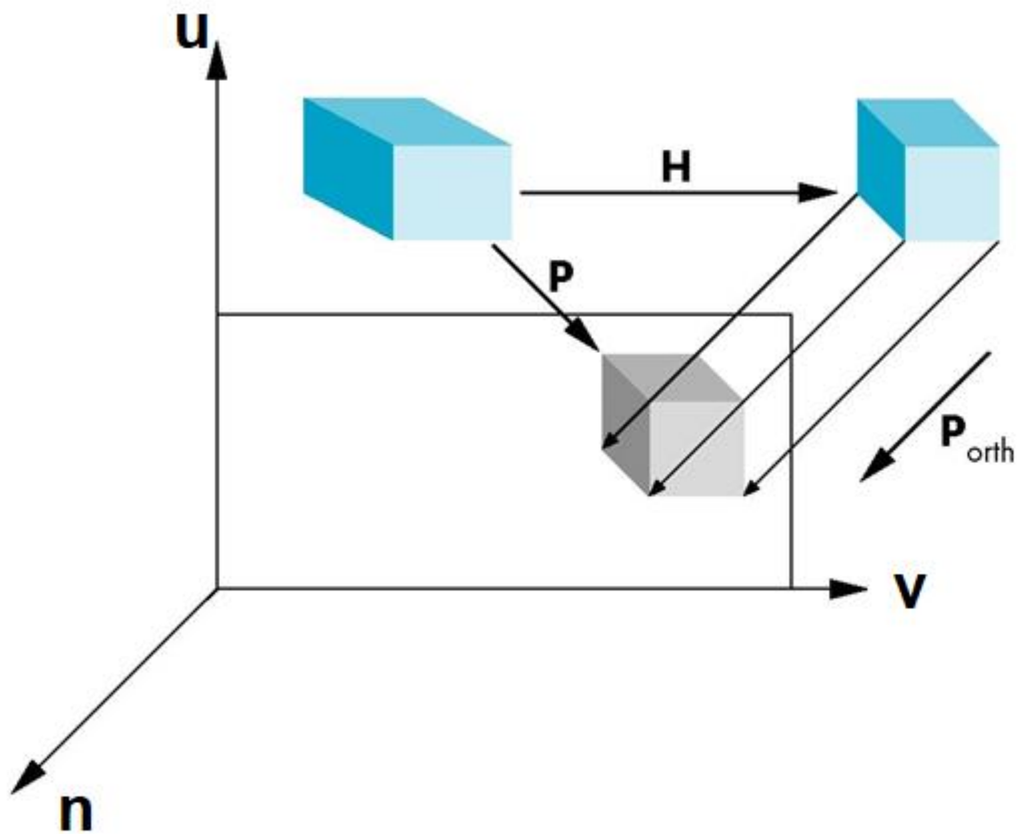
$0 < near < far$

Oblique Projections

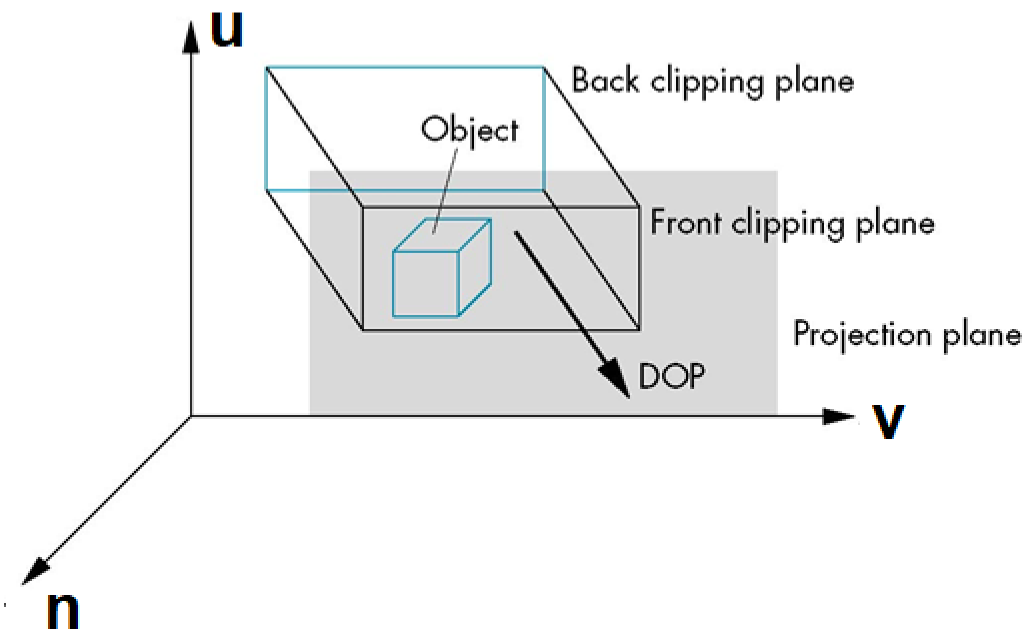
- The OpenGL projection functions cannot produce general parallel projections such as

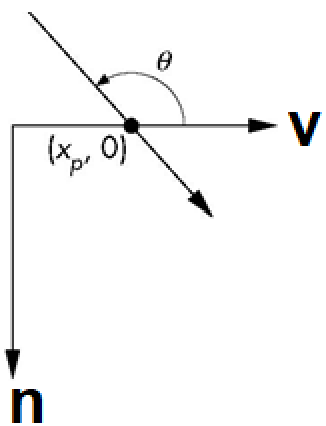
- However if we look at the example of the cube it appears that the cube has been sheared
- Oblique Projection = Shear + Orthogonal Projection

Oblique Projections

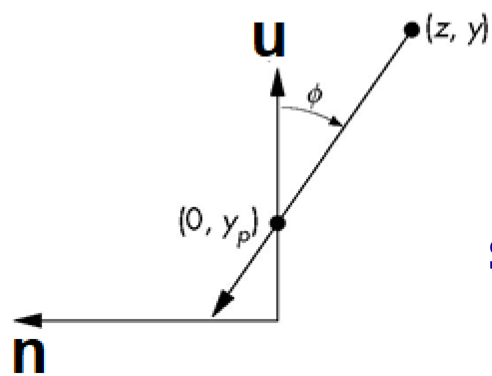


Oblique Projections





top view



side view

Oblique Projections

- Shear matrix

  xy shear (*z* values unchanged)

$$\mathbf{H}(\theta,\phi) = \begin{bmatrix} 1 & 0 & \cot\theta & 0 \\ 0 & 1 & \cot\phi & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  Projection matrix

$$\mathbf{P} = \mathbf{M}_{\text{orth}}\, \mathbf{H}(\theta,\phi)$$

Further Reading

- "**Interactive Computer Graphics: A Topdown Approach Using OpenGL**", *Edward Angel*
  - Chapter 5: Viewing
- "Đồ họa máy tính trong không gian ba chiều", Trần Giang Sơn
  - Phép nhìn trong không gian ba chiều