Computer Graphics 7: Viewing in 3-D

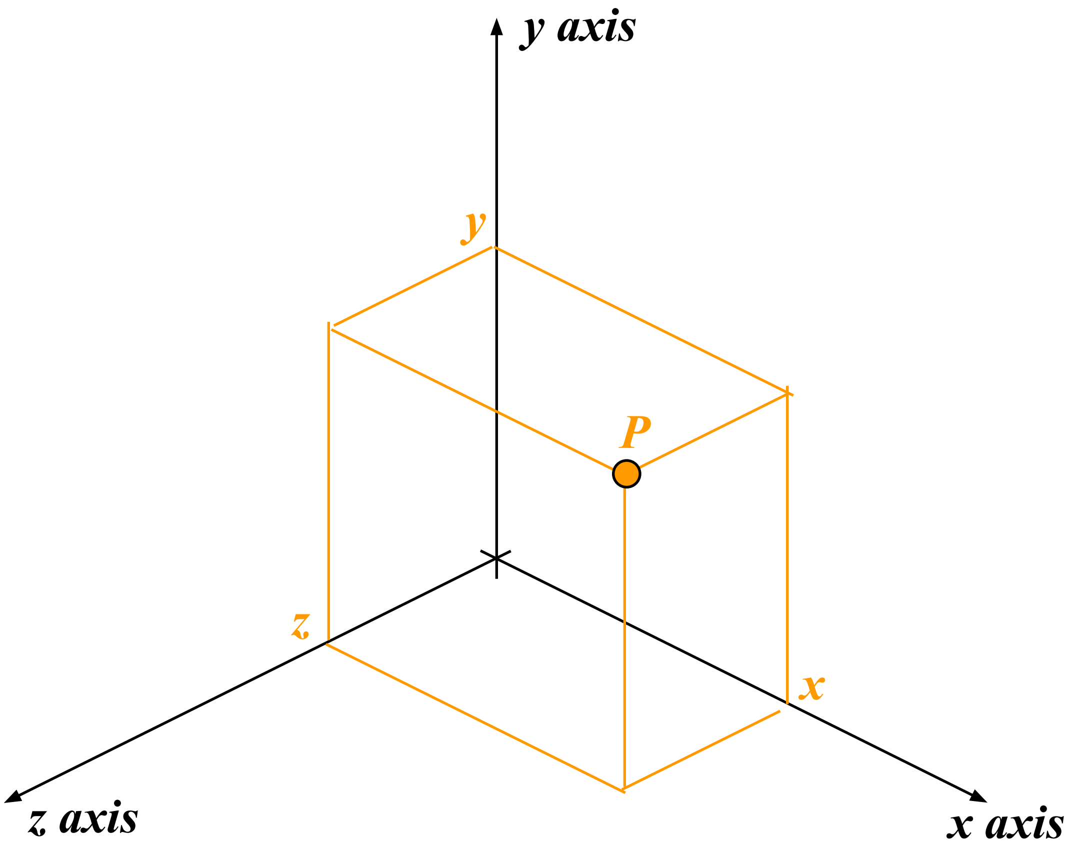
Contents

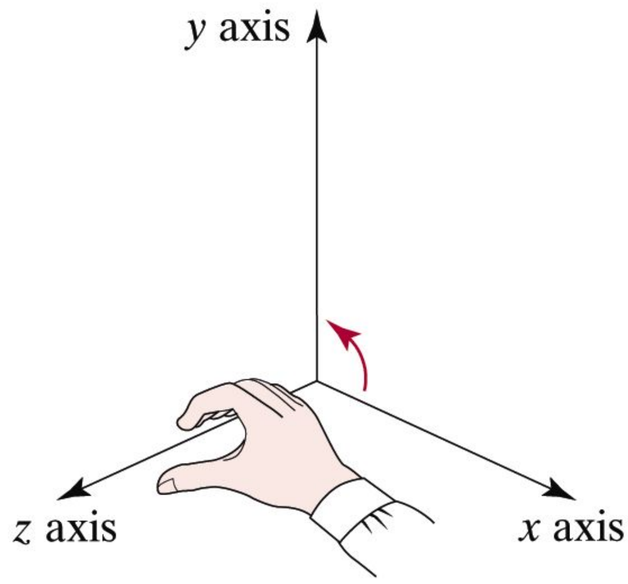
In today's lecture we are going to have a look at:

- Revisiting Transformations in 3-D
  - How do transformations in 3-D work?
  - 3-D homogeneous coordinates and matrix based transformations
- 3-D Viewing
- Basics of Projection
  - Geometrical Constructions
  - Types of Projection
- 3-D Object Modelling

3-D Coordinate Spaces

Remember what we mean by a 3-D coordinate space



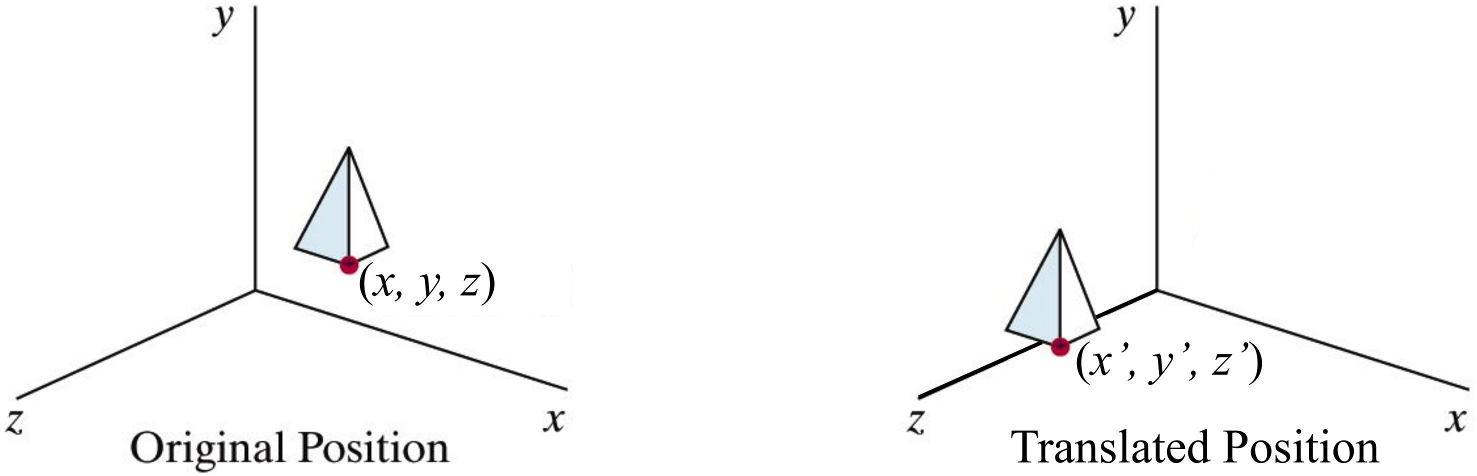


Right-Hand
Reference System

Translations In 3-D

To translate a point in three dimensions by dx, dy and dz simply calculate the new points as follows:

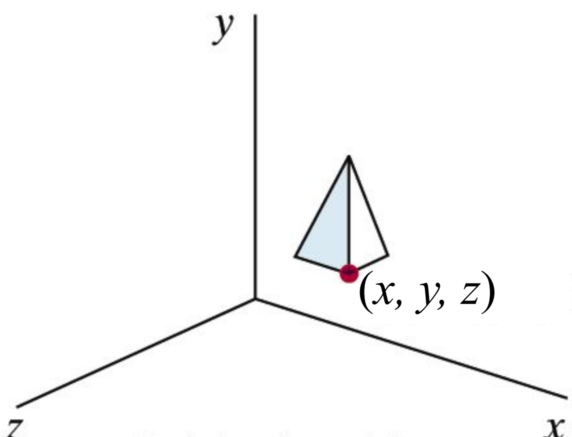
$$x' = x + dx \quad y' = y + dy \quad z' = z + dz$$



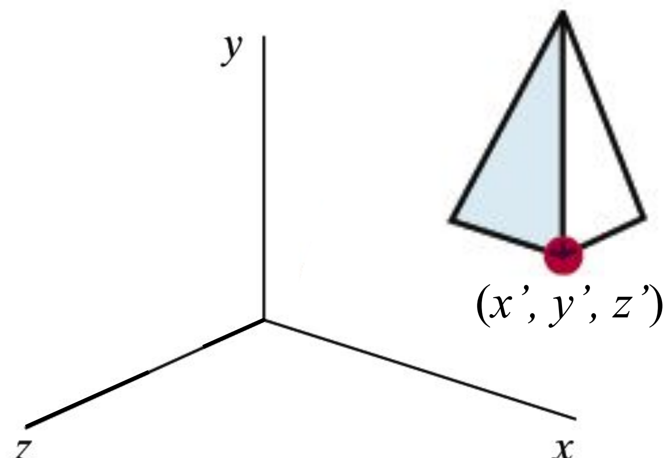
Scaling In 3-D

To sale a point in three dimensions by sx, sy and sz simply calculate the new points as follows:

$$x' = sx*x \qquad y' = sy*y \qquad z' = sz*z$$



Original Position



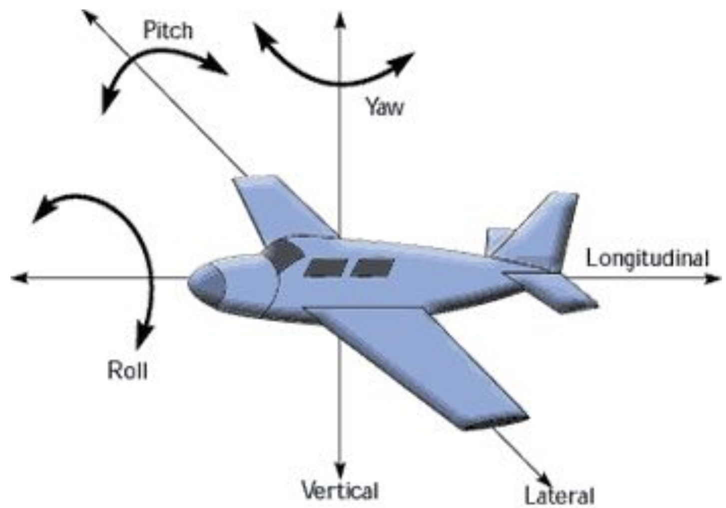
Scaled Position

Rotations In 3-D

When we performed rotations in two dimensions we only had the choice of rotating about the z axis

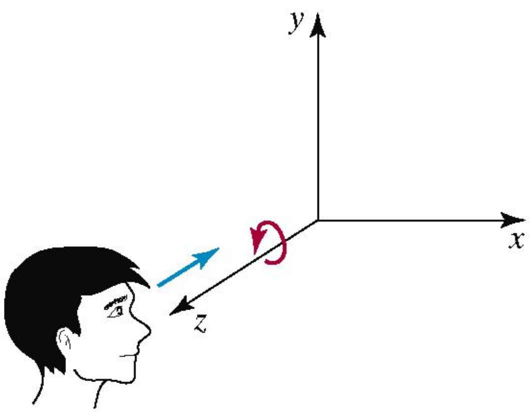
In the case of three dimensions we have more options

- Rotate about x – pitch
- Rotate about y – yaw
- Rotate about z - roll

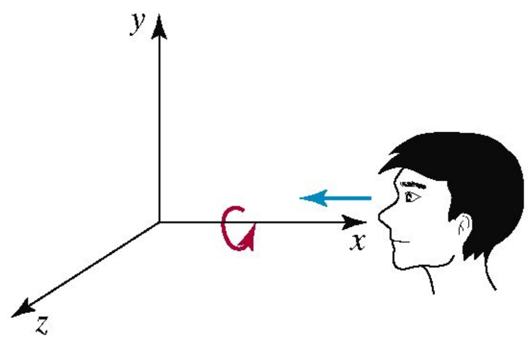


Rotations In 3-D (cont…)

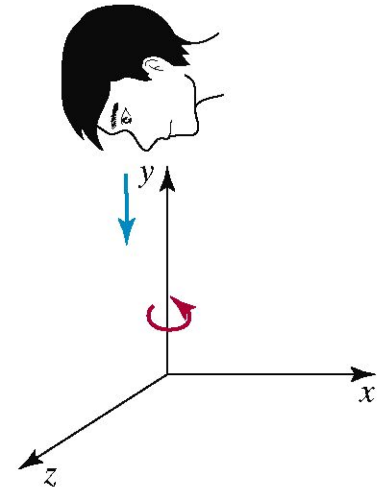
The equations for the three kinds of rotations in 3-D are as follows:



$$x' = x \cdot \cos\theta - y \cdot \sin\theta$$
$$y' = x \cdot \sin\theta + y \cdot \cos\theta$$
$$z' = z$$



$$x' = x$$
$$y' = y \cdot \cos\theta - z \cdot \sin\theta$$
$$z' = y \cdot \sin\theta + z \cdot \cos\theta$$



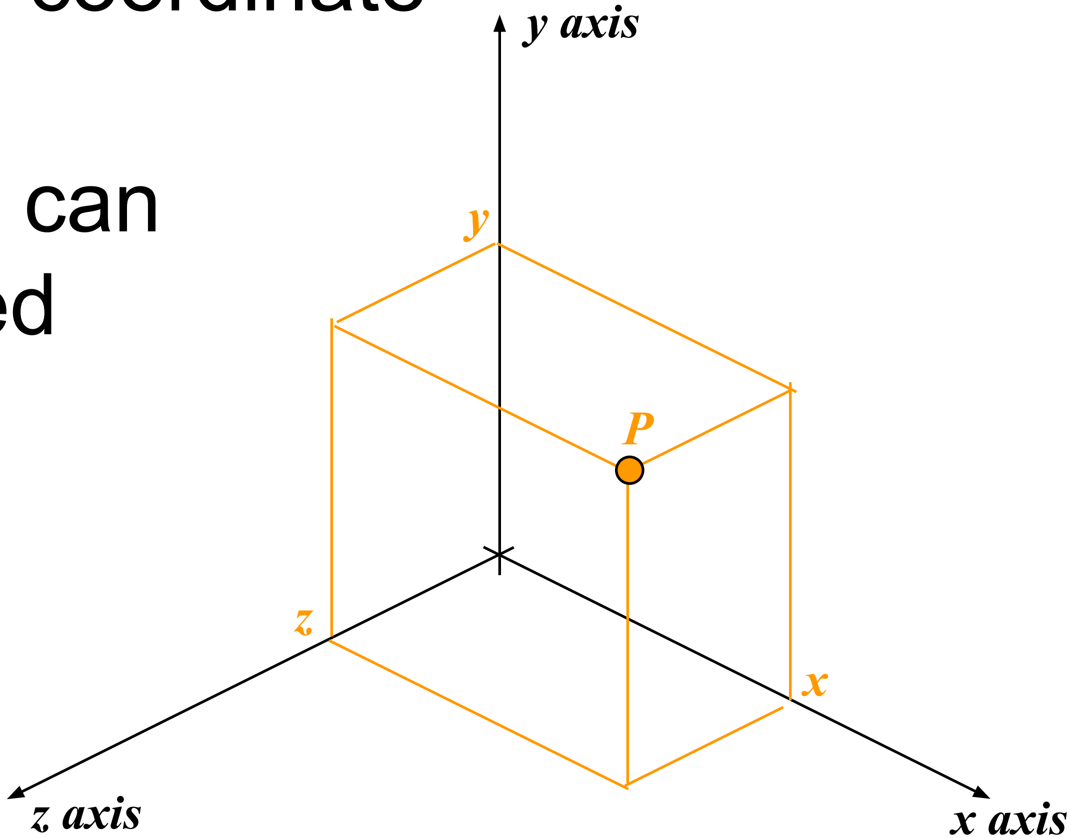
$$x' = z \cdot \sin\theta + x \cdot \cos\theta$$
$$y' = y$$
$$z' = z \cdot \cos\theta - x \cdot \sin\theta$$

Homogeneous Coordinates In 3-D

Similar to the 2-D situation we can use homogeneous coordinates for 3-D transformations - 4 coordinate column vector

All transformations can then be represented as matrices

$$P(x, y, z) = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



3D Transformation Matrices

Translation by dx, dy, dz

$$\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling by sx, sy, sz

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotate About X-Axis

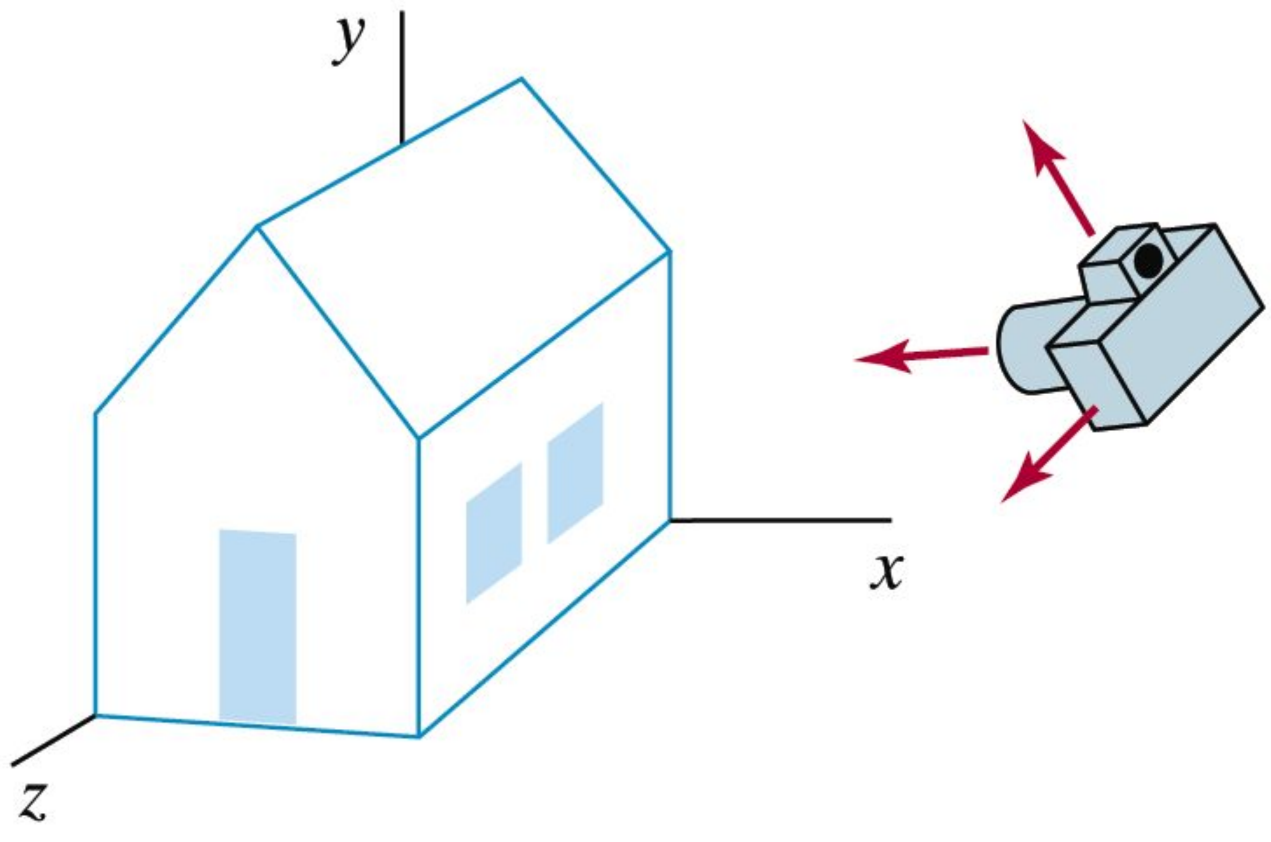
$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotate About Y-Axis

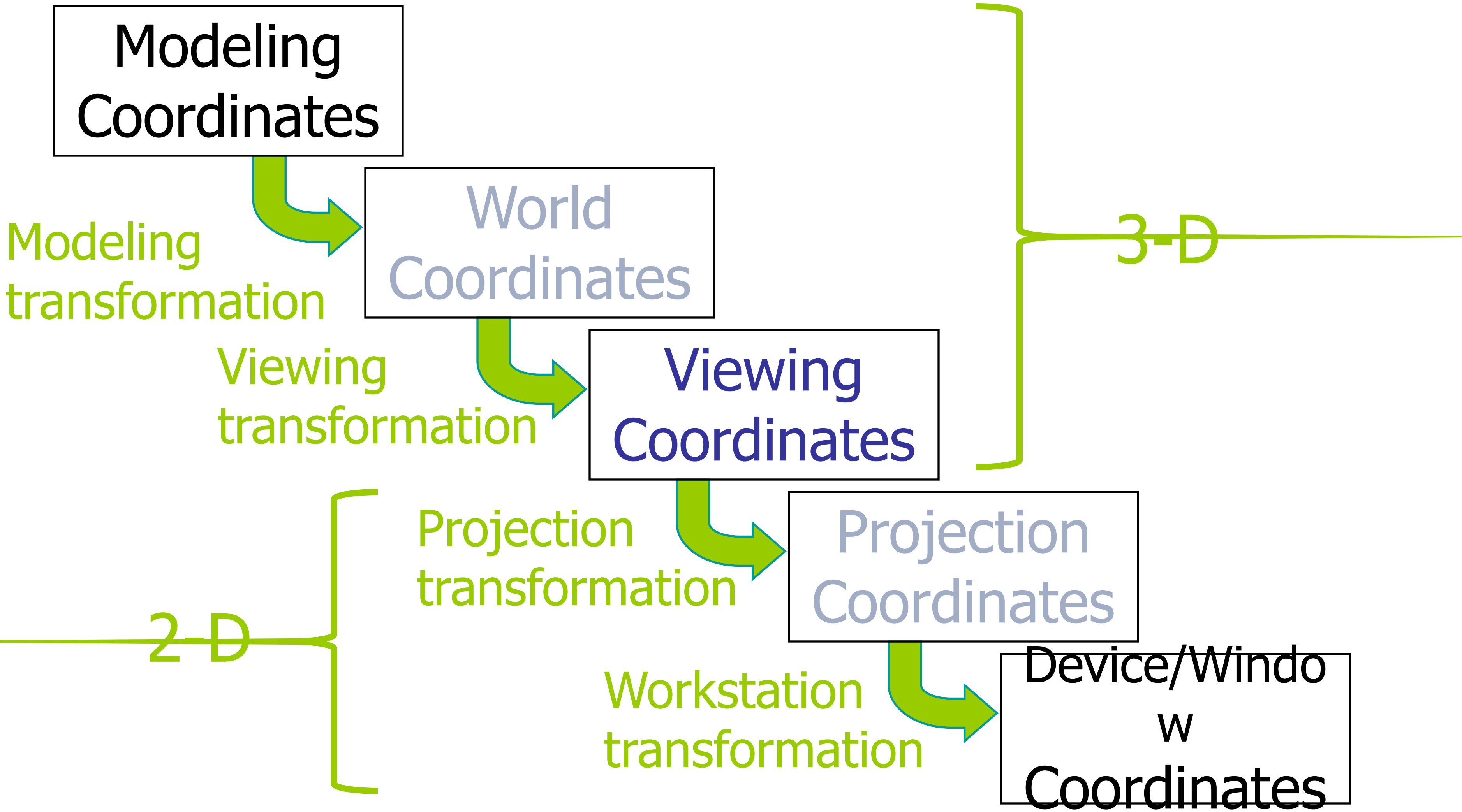
$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotate About Z-Axis

3-D Viewing Basics



3-D Viewing Process

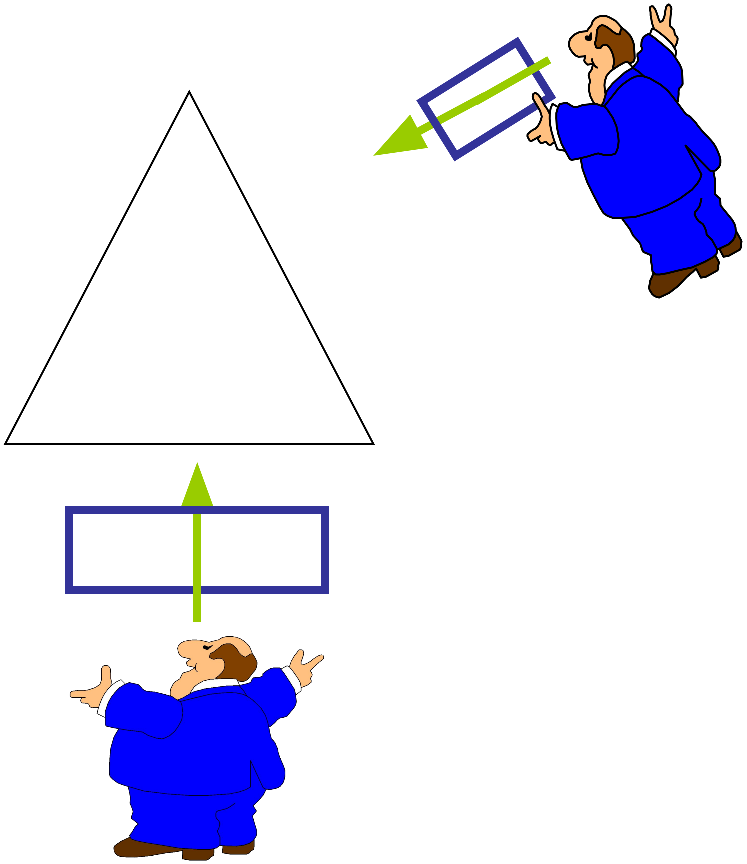


Viewing Coordinate System

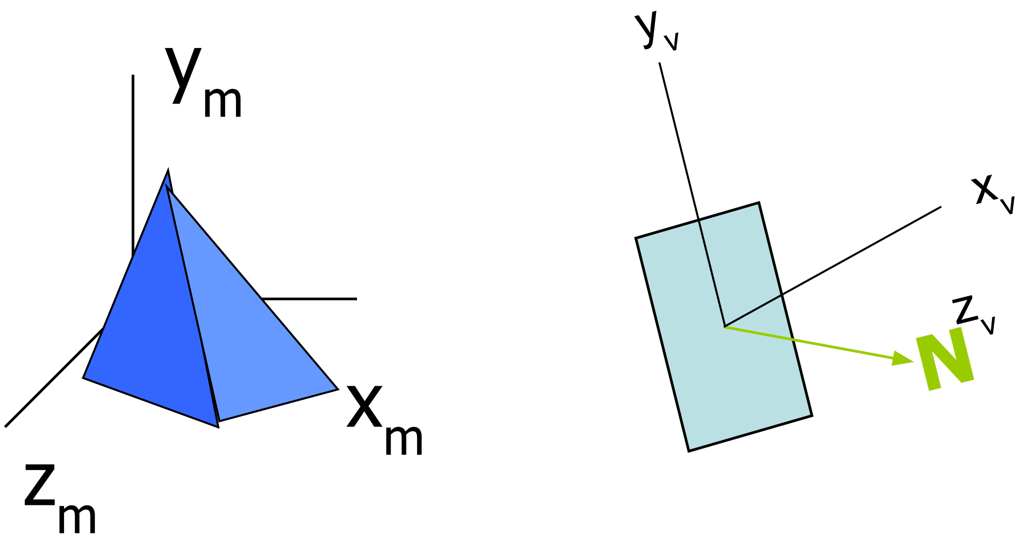
Identify viewer position relative to scene

Viewer "looks through" a window

Must specify position and view direction

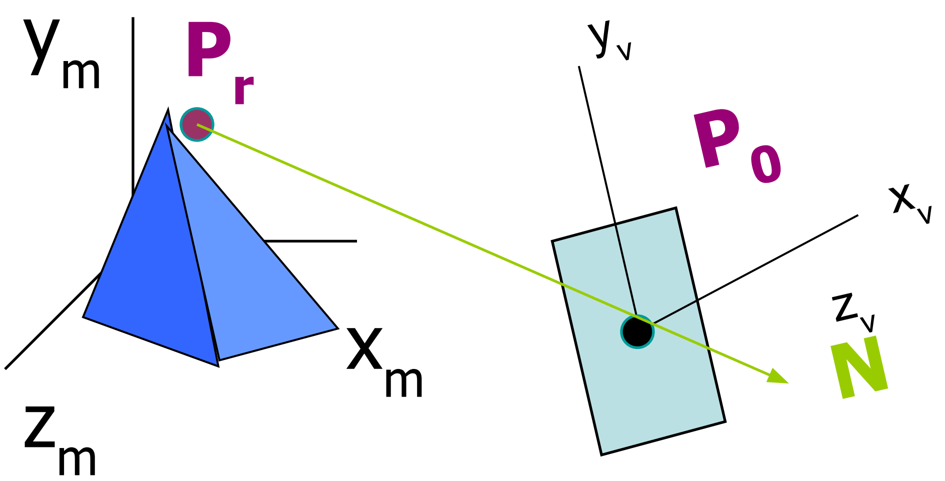


View Plane



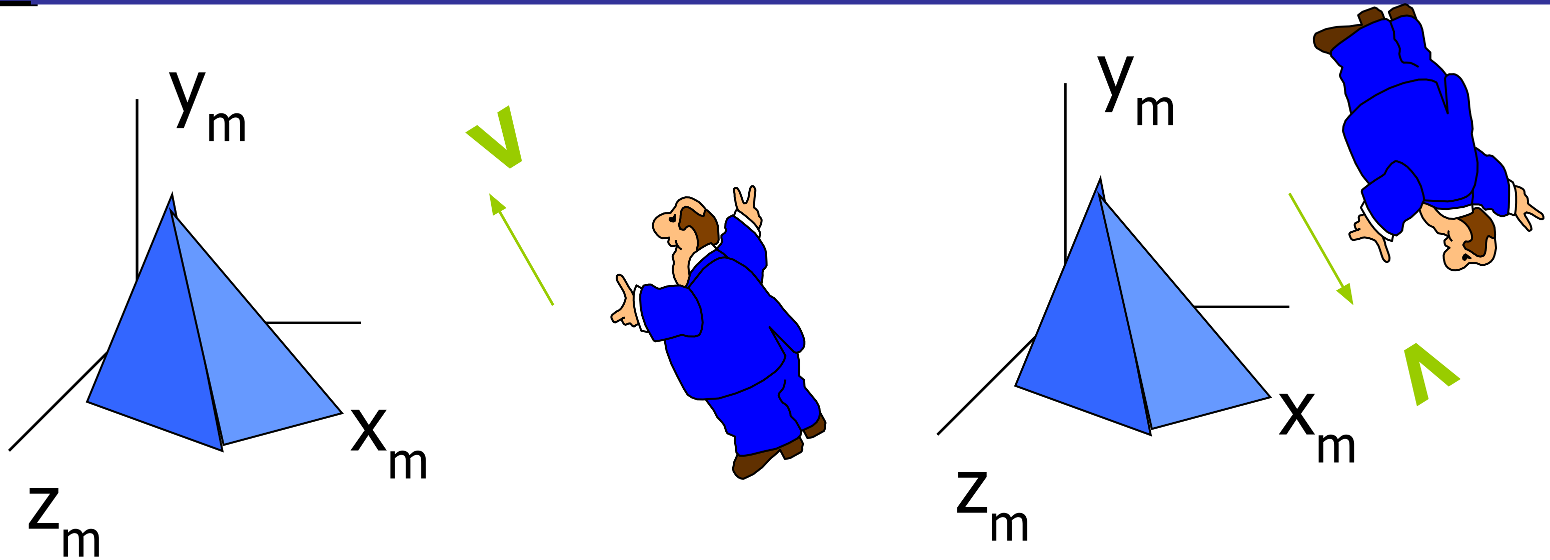
View plane defined by normal vector (**N**)

View Reference Points



- P_r: a point in the scene we are looking at
- P_0: a distant point from which we're looking
- Note P_r, P_0, and N are expressed in $x_m y_m z_m$

Look-Up Vector

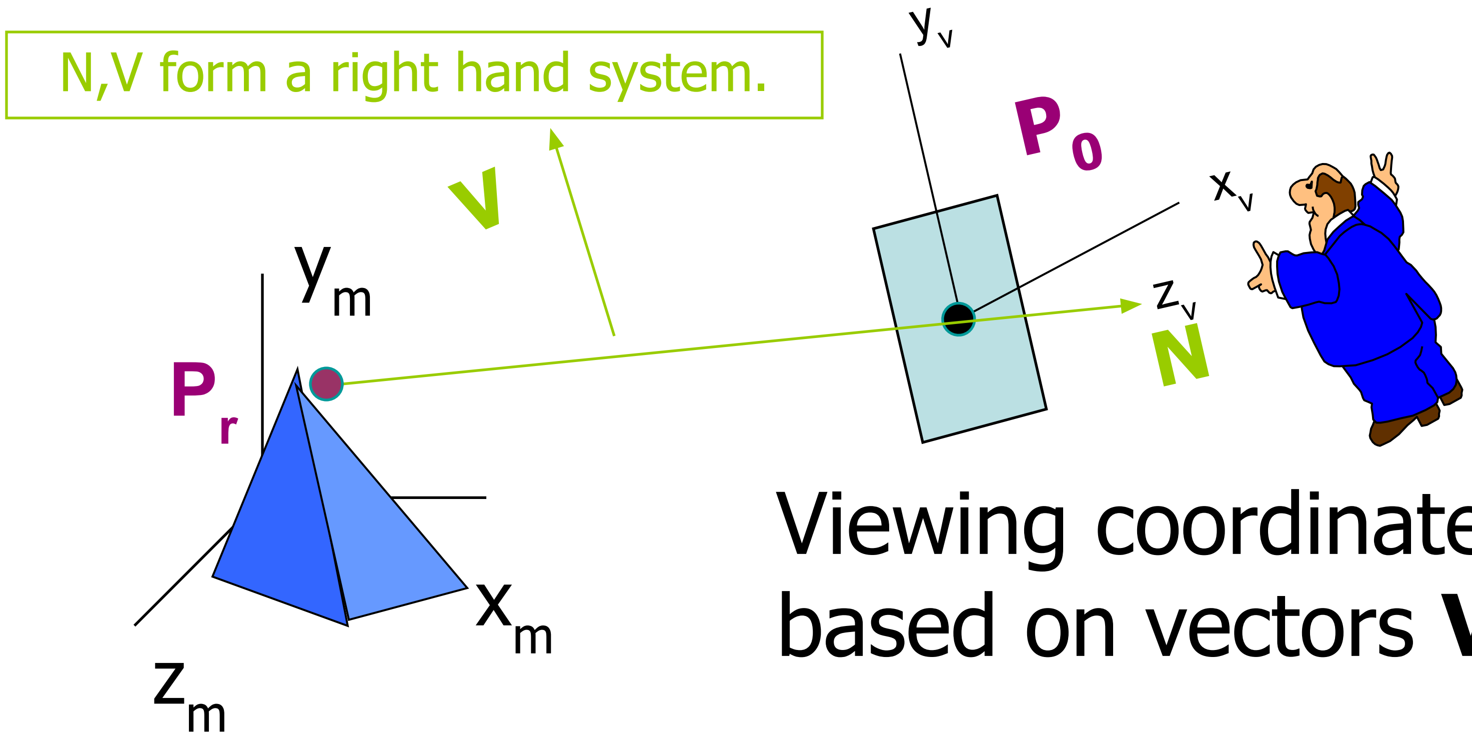


View-plane normal vector and reference point are not enough

We also need to specify orientation of view(er)

View-up vector (**V**) must be normal to **N**

Viewing Coordinates

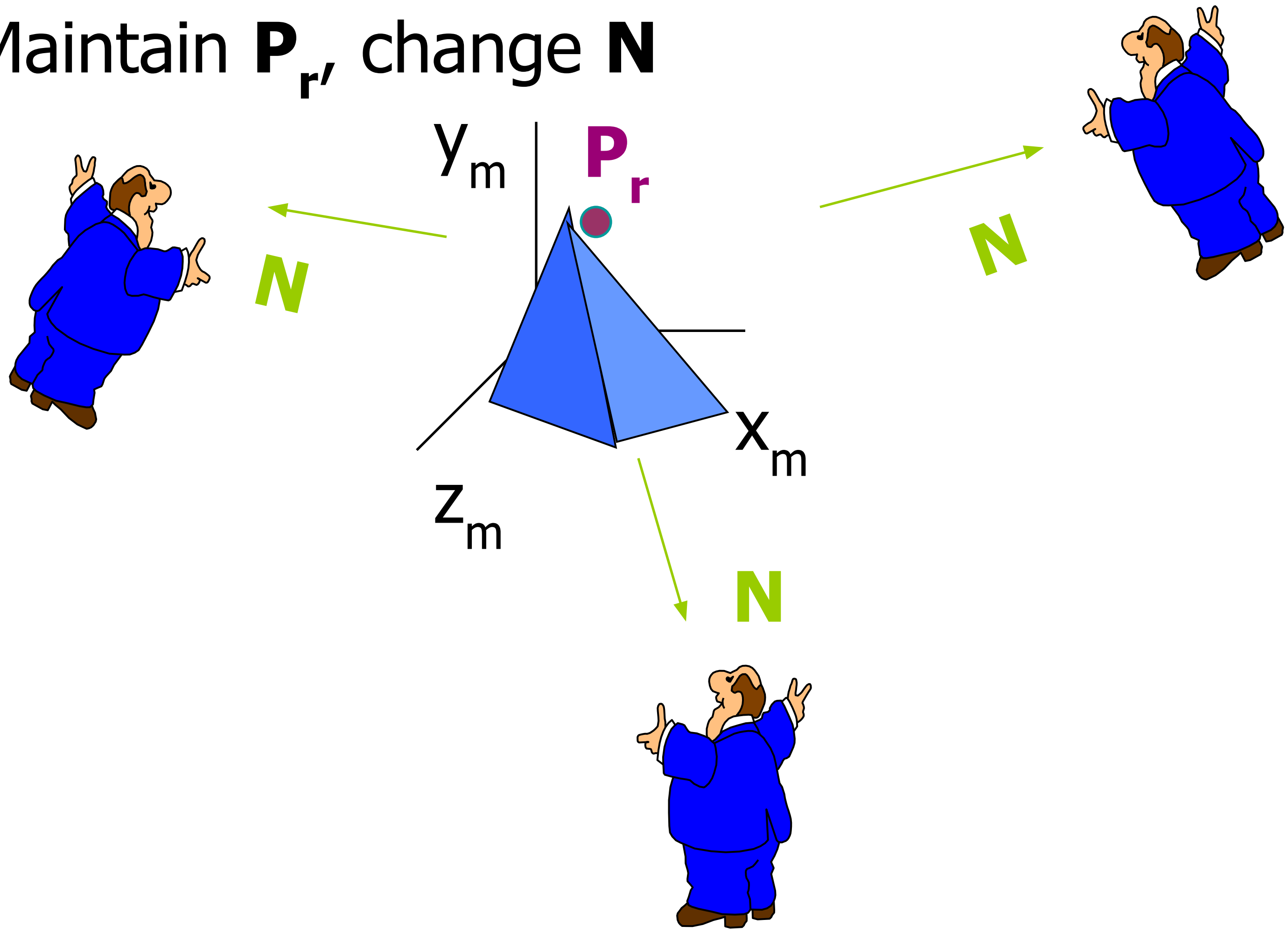


Viewing coordinate system based on vectors **V,N**

Forms the (x_v, y_v, z_v) coordinate axes

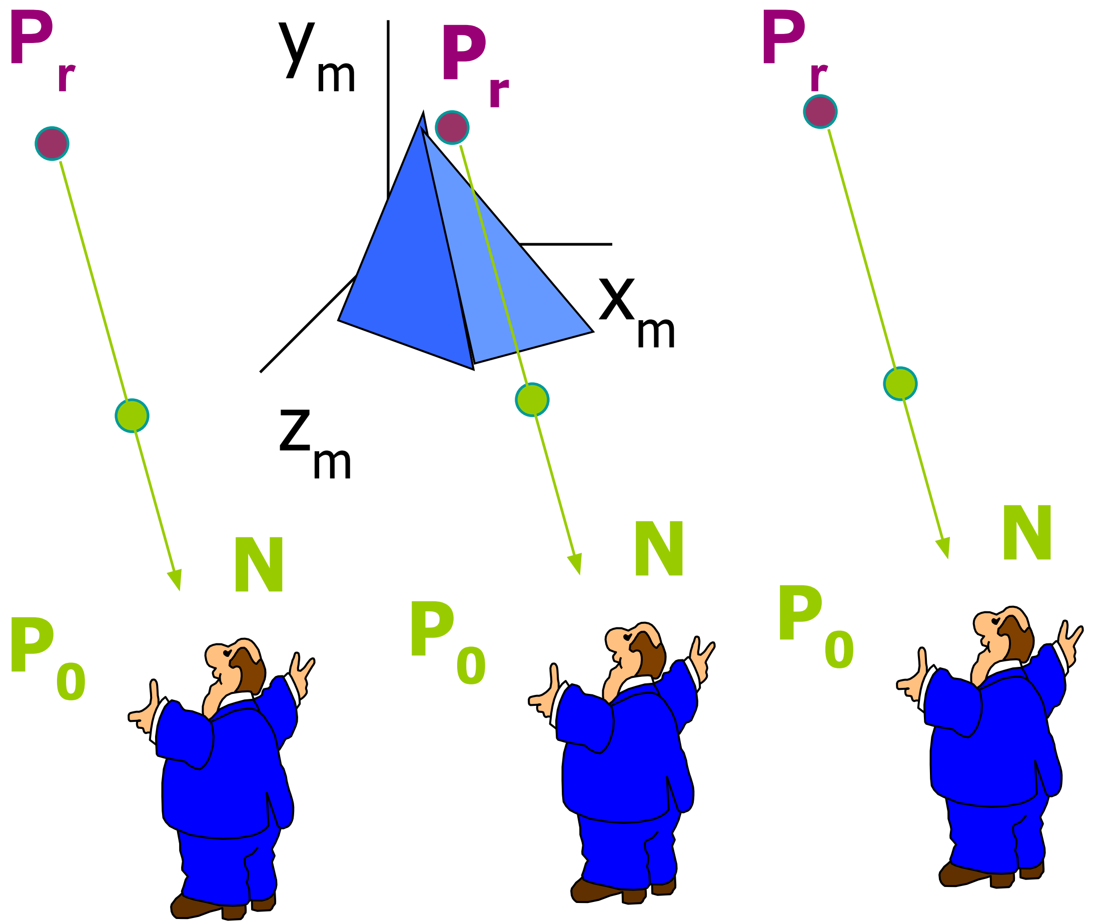
Changing Views (1)

Maintain $\mathbf{P_r}$, change **N**



Changing Views (2)

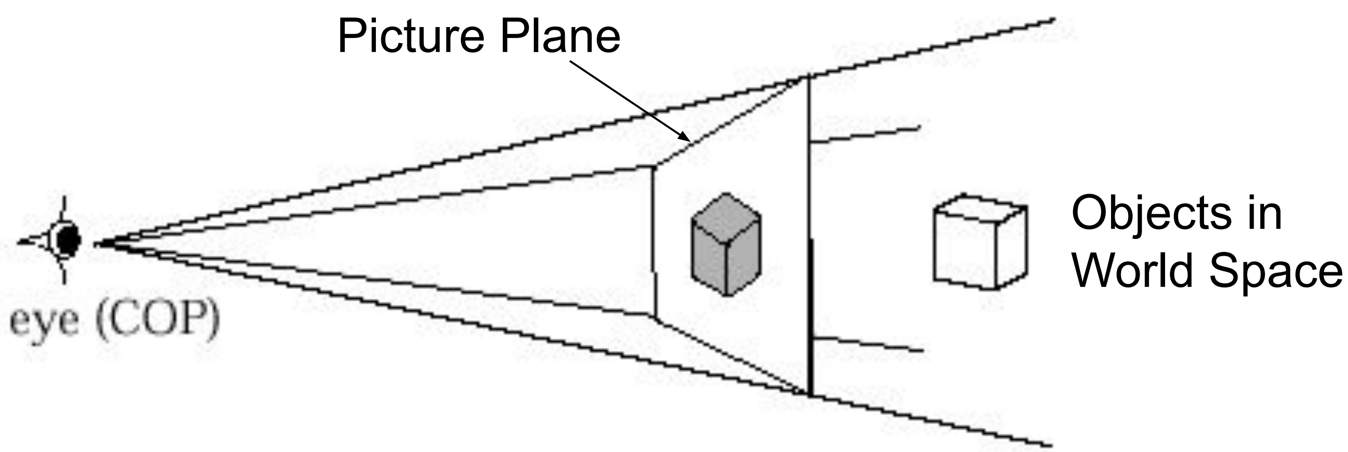
Maintain **N**, change $\mathbf{P_r}$ and $\mathbf{P_0}$



What Are Projections?

Our 3-D scenes are all specified in 3-D world coordinates

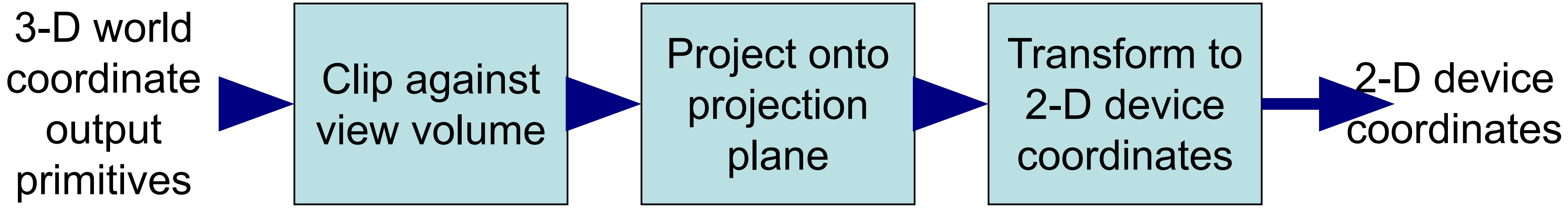
To display these we need to generate a 2-D image - *project* objects onto a *picture plane*



So how do we figure out these projections?

Converting From 3-D To 2-D

Projection is just one part of the process of converting from 3-D world coordinates to a 2-D image

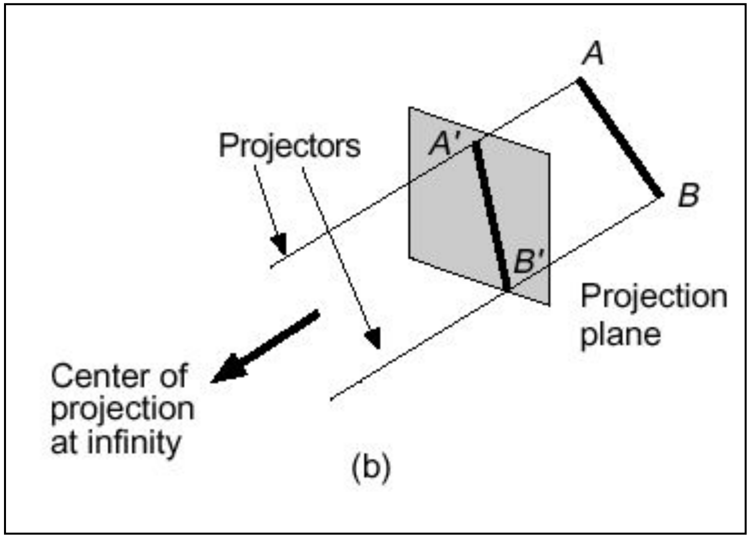


Types Of Projections

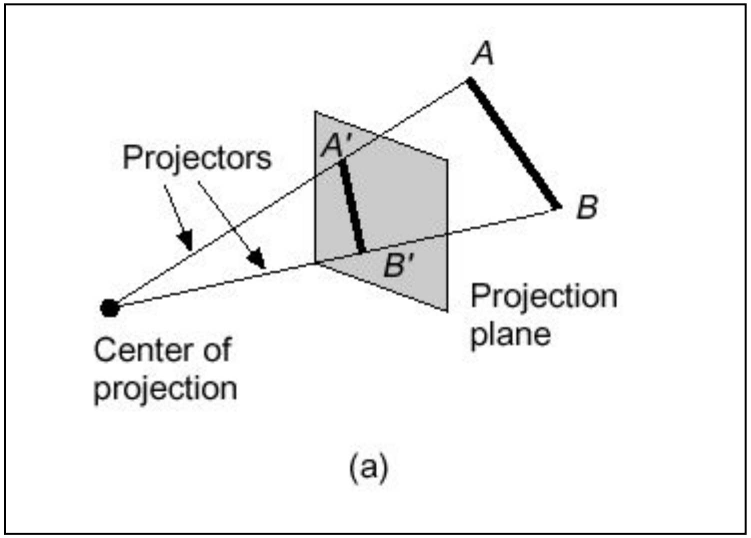
There are two broad classes of projection:

– Parallel: Typically used for architectural and engineering drawings

– Perspective: Realistic looking and used in computer graphics



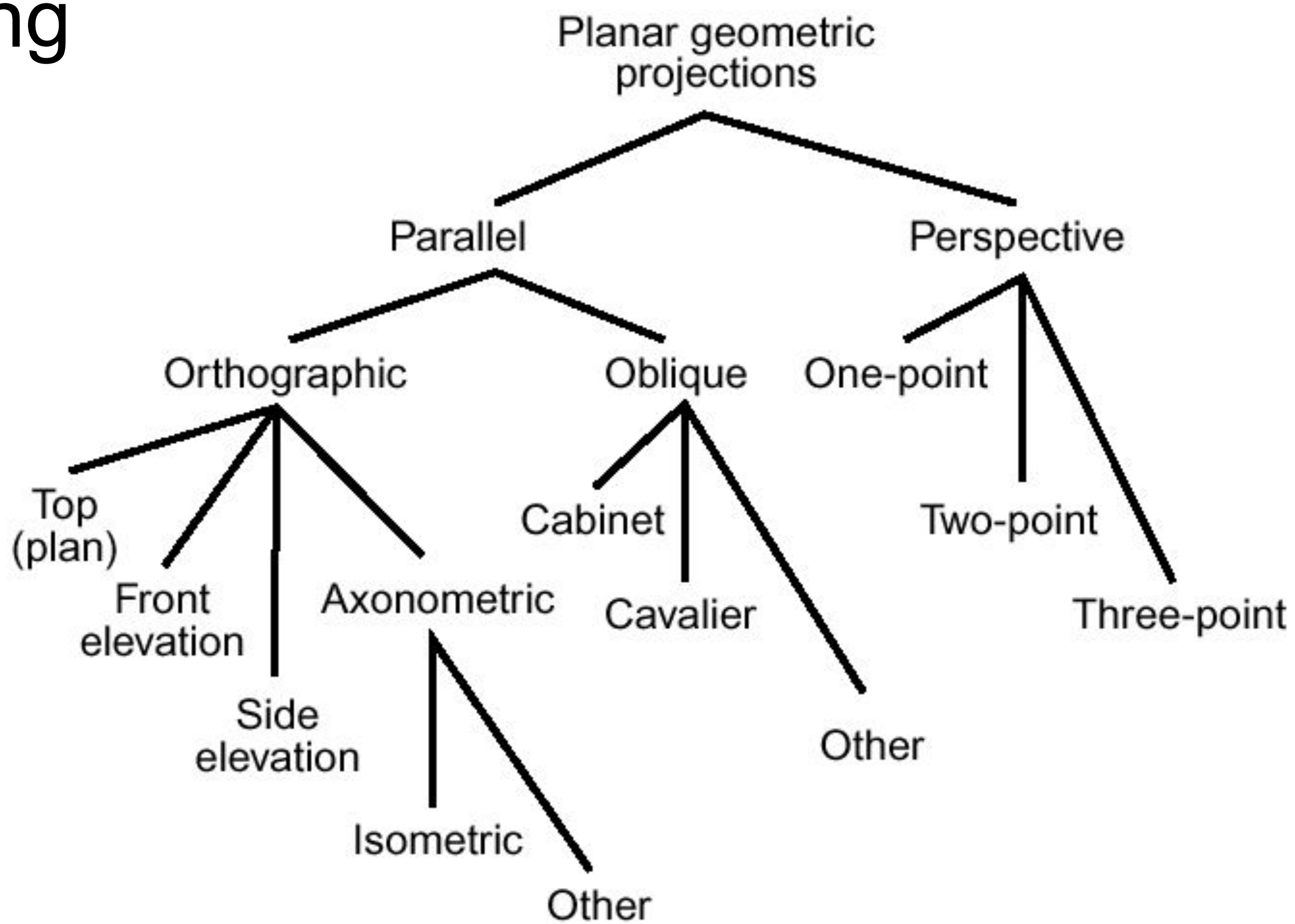
Parallel Projection



Perspective Projection

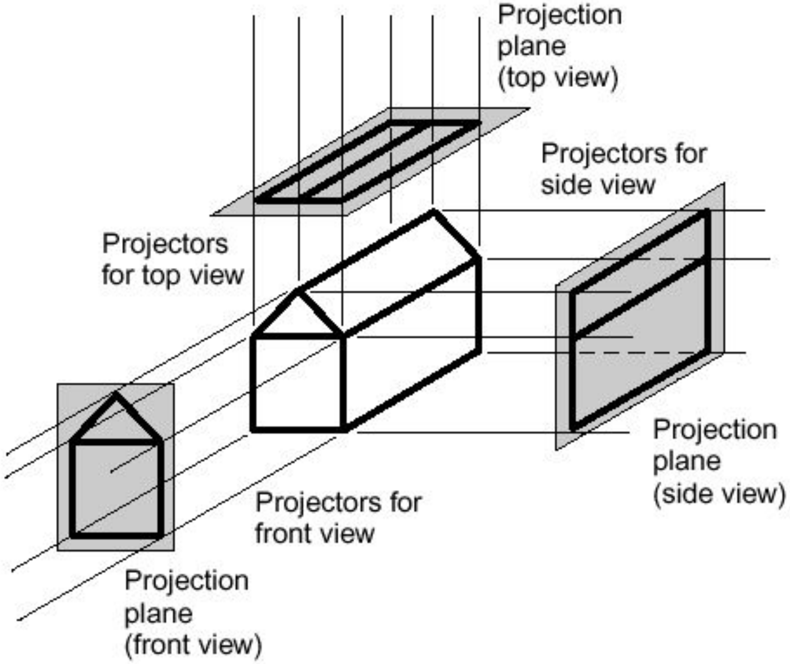
Types Of Projections (cont…)

For anyone who did engineering or technical drawing

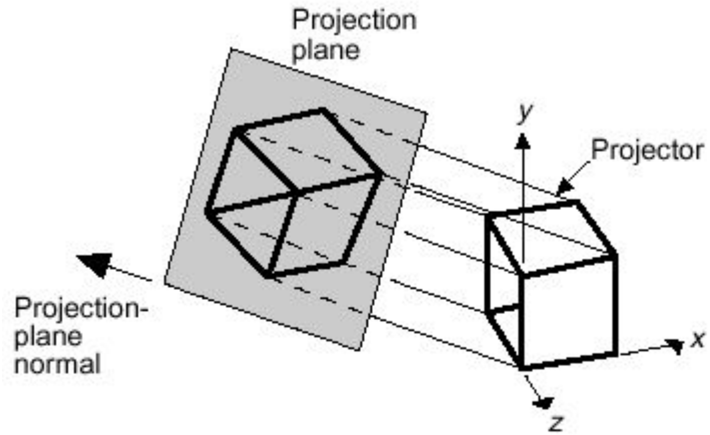


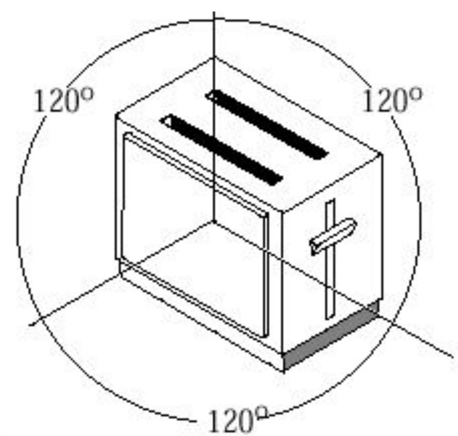
Parallel Projections

Some examples of parallel projections



Orthographic Projection





Isometric Projection

Isometric Projections

Isometric projections have been used in computer games from the very early days of the industry up to today

Q*Bert

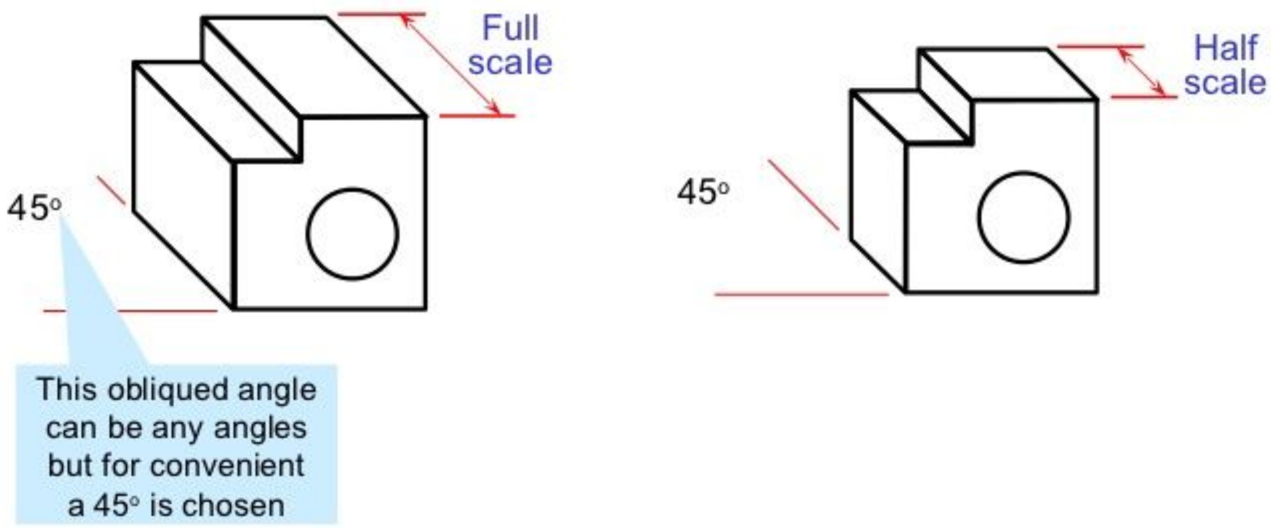
Sim City

Virtual Magic Kingdom

Oblique Projections

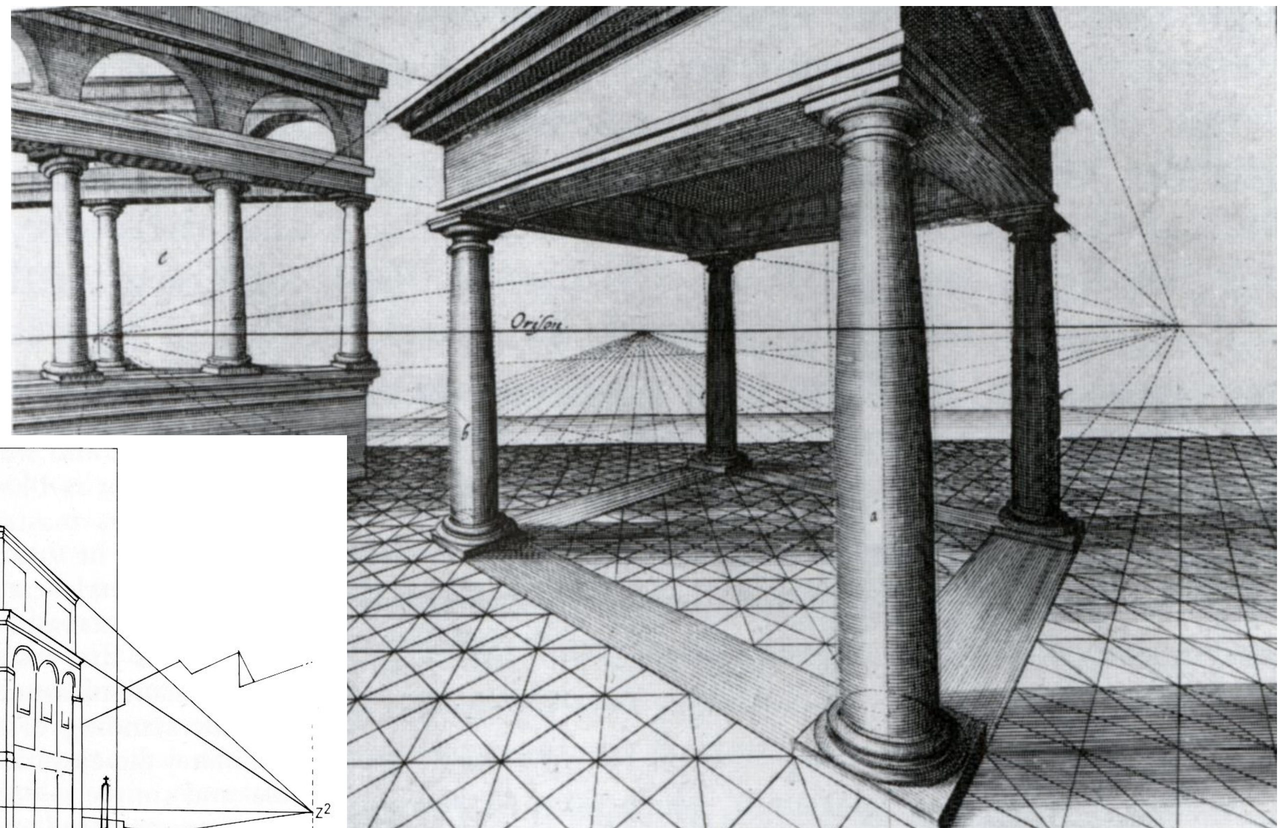
Type of an oblique projection

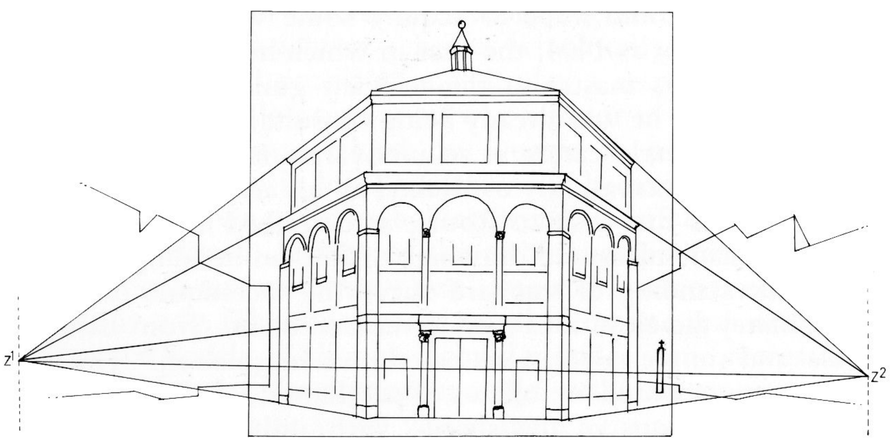
1. Cavalier **2. Cabinet**



Perspective Projections

Perspective projections are much more realistic than parallel projections

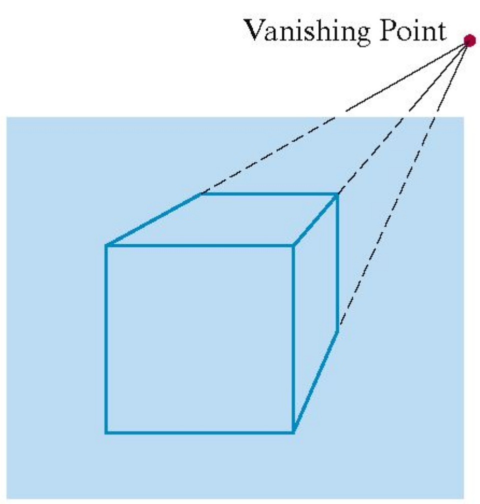




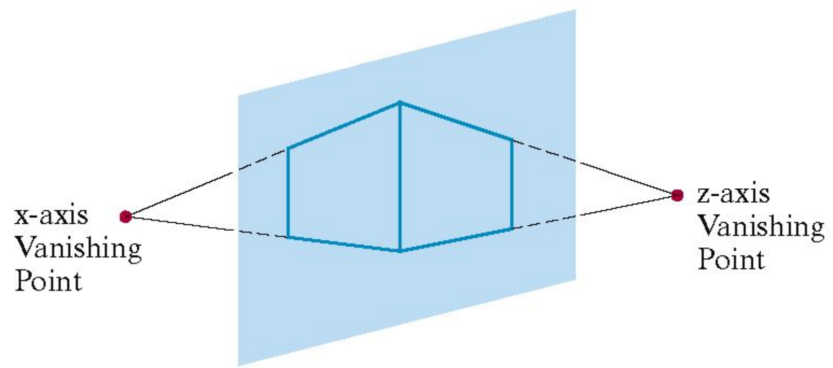
Perspective Projections

There are a number of different kinds of perspective views

The most common are one-point and two point perspectives

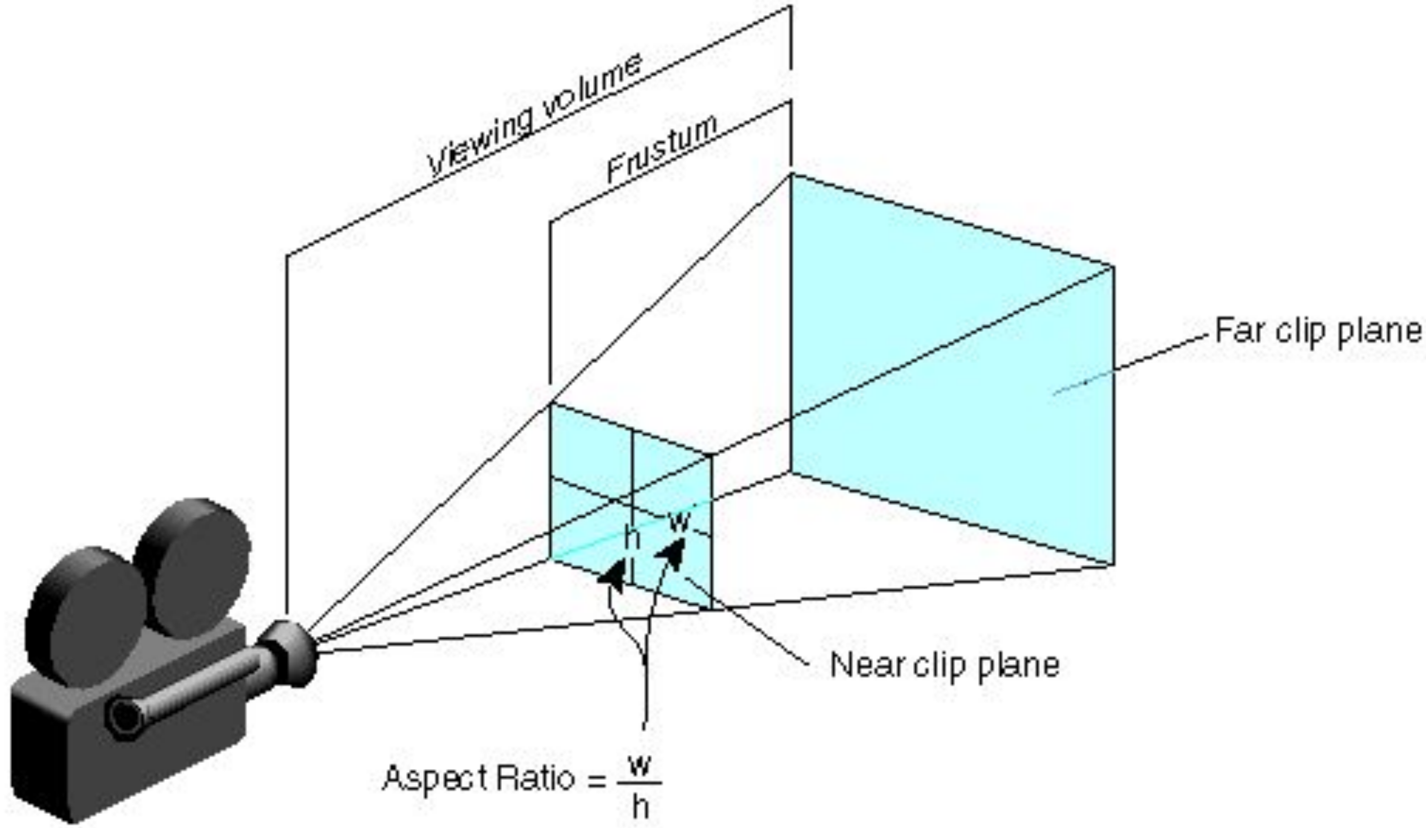


One Point Perspective Projection



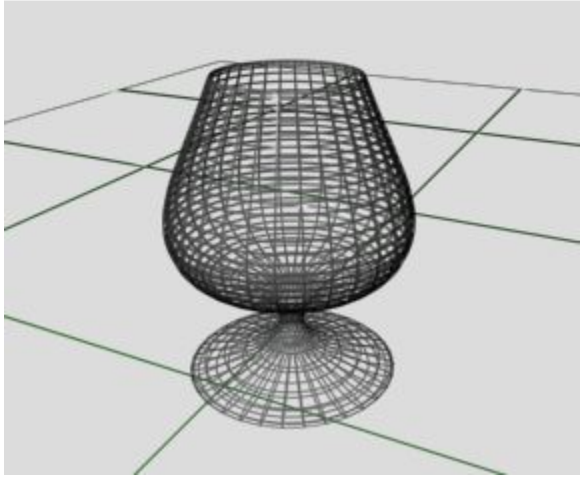
Two-Point Perspective Projection

Elements Of A Perspective Projection



Virtual
Camera

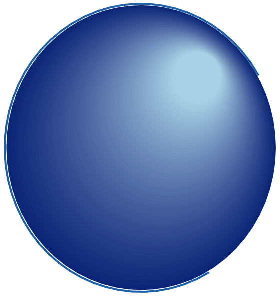
3-D Object Modeling



There are basically two ways to model objects in 3-D

Space-partitioning (Solid modeling)

- Describe interior as union of solids
- Parametric modeling
  - what parameters would we use?

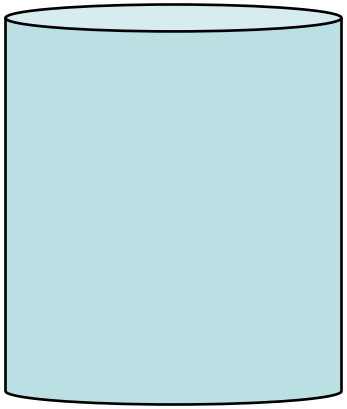


Boundary representations (B-reps)

- Describe the set of surfaces
  - That define the exterior surface of an object
  - That separate object interior from exterior
  - What kinds of surfaces?

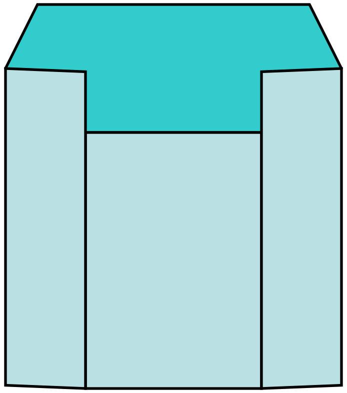
Most common is Boundary Representation

Advantage: surface equations are linear



Representing objects:

- Polyhedrons - no problem
- General shapes - approximation
  - Tessellate (subdivide) to polygon mesh

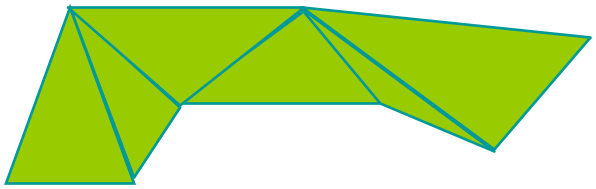


Polygon Meshes can be used to describe a general surface

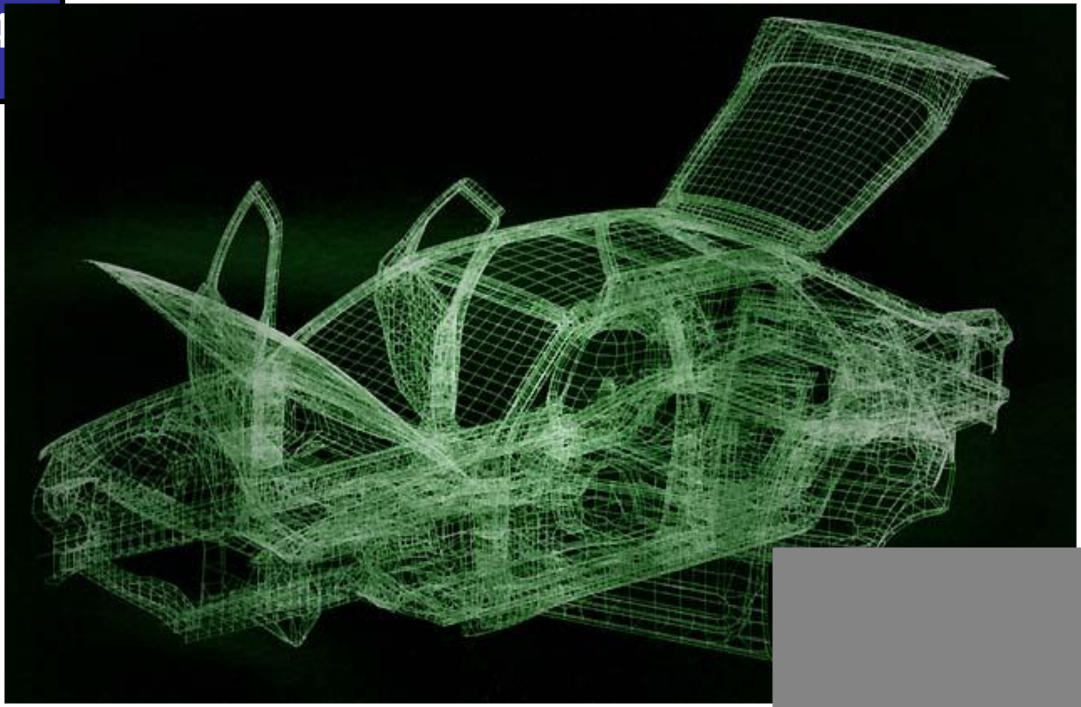
Replaces it with an approximation

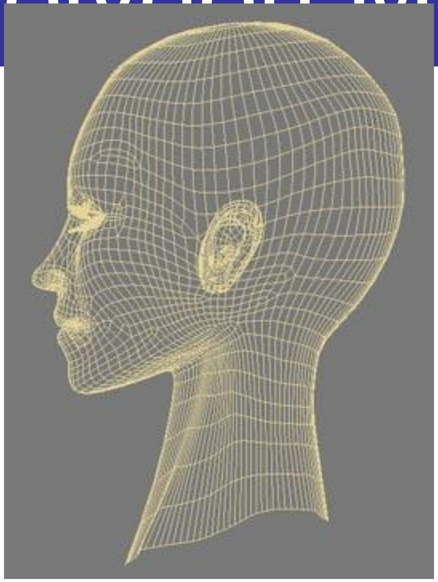
Common sets of polygons

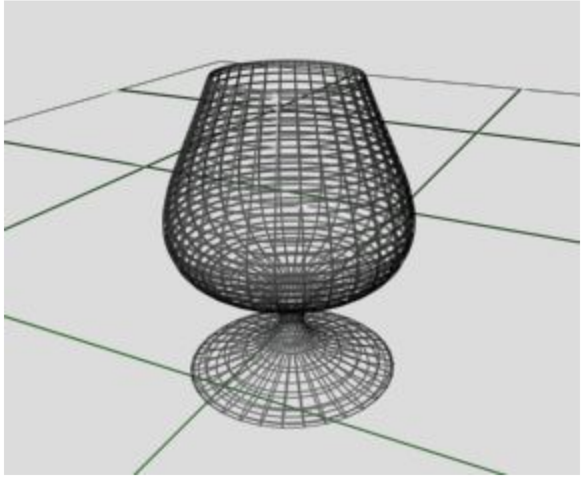
- Triangles
  - Advantage: 3 vertices determine plane
- Quadrilaterals

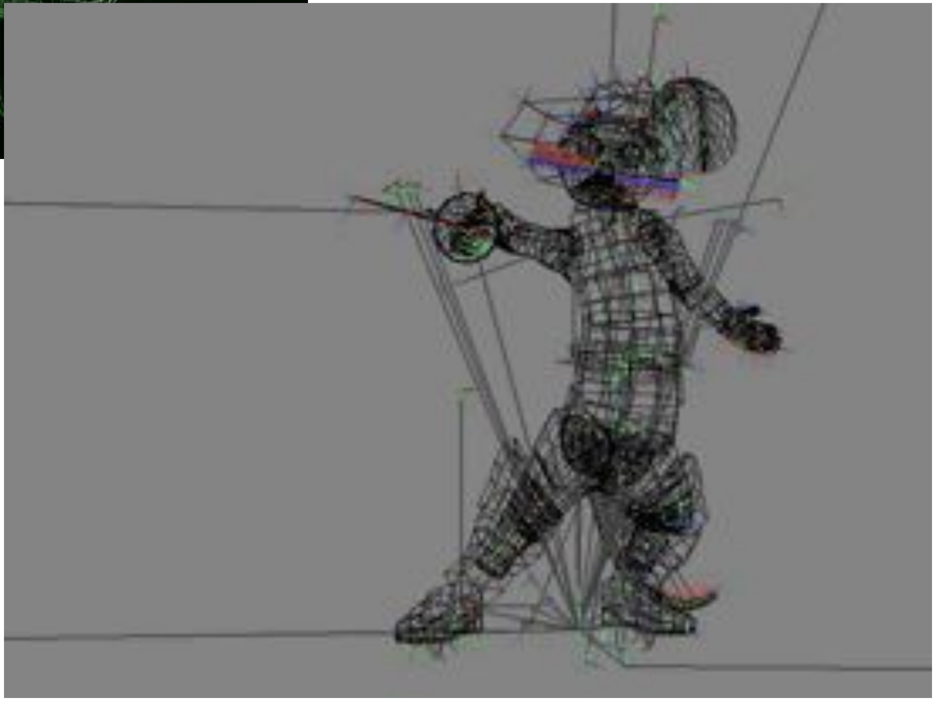


Polygon Meshes









Polygon Description

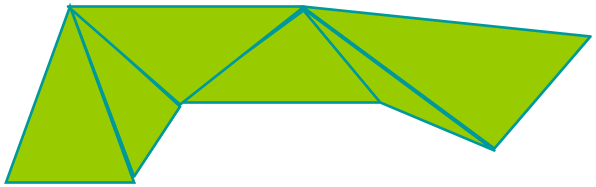
Geometric data
- Description of position and shape

Attribute data
- Description of surface
  - Color
  - Transparency
  - Surface reflectivity
  - Texture

Polygon Description

How do you represent a polygon surface approximation?

- List of vertices (in 3-D) could work
  - Sufficient description
  - But, polygons are joined
    - Shared vertices and edges



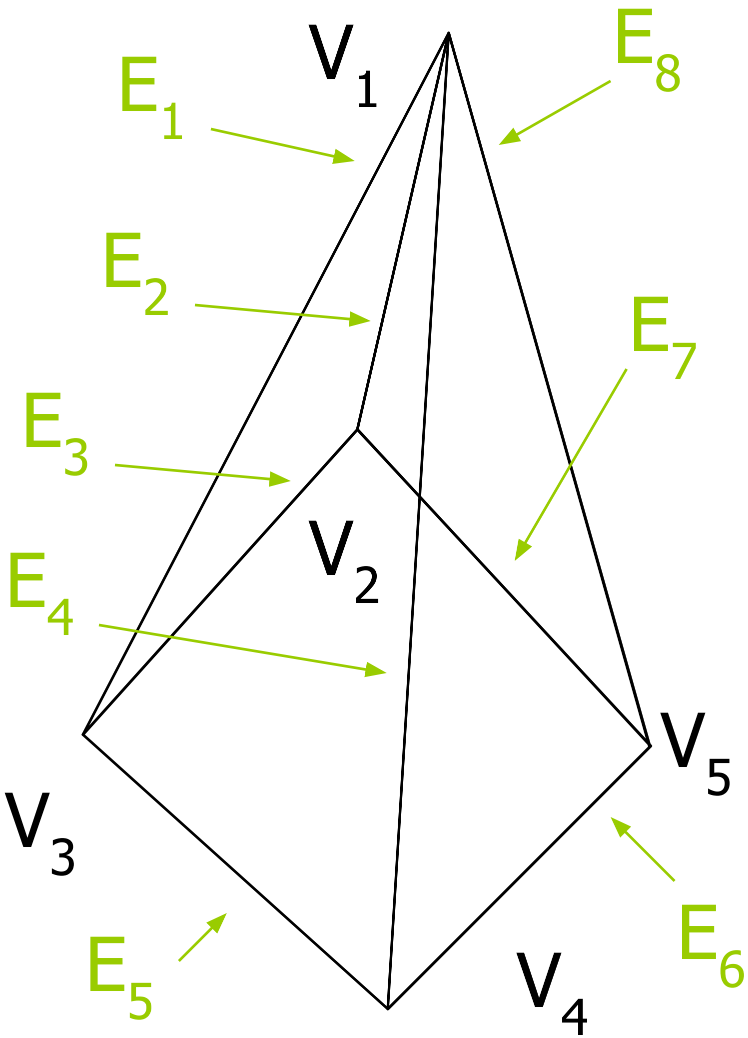
- We need a more general description that
  - Reduces redundancy
  - Represents component polygons

Polygon Tables

- Vertex table
  - For all polygons in composite object
- Edge table
  - Each edge listed once
    - Even if part of more than one polygon
- Polygon-surface table

Pointers or other links between tables for easy access

Polygon Table Example



$E_1: V_1, V_3$
$E_2: V_1, V_2$
$E_3: V_2, V_3$
$E_4: V_1, V_4$
$E_5: V_3, V_4$
$E_6: V_4, V_5$
$E_7: V_2, V_5$
$E_8: V_1, V_5$

$S_1: E_1, E_2, E_3$
$S_2: E_2, E_7, E_8$
$S_3: E_1, E_4, E_5$
$S_4: E_4, E_6, E_8$
$S_5: E_3, E_5, E_6, E_7$

Polygon surfaces can be represented by Plane Equations

Often need information on

- Spatial orientation of surfaces
  - Visible surface identification
  - Surface rendering (e.g. shading)

Processing a 3-D object involves

- Equations of polygon planes
- Coordinate transformations (3-D)

The Plane Equation

$$Ax + By + Cz + D = 0$$

Must be satisfied for any point (x,y,z) in the plane.

We want to solve for coefficients (A, B, C, D).

How many points define a plane?

How many unknowns are there?

Vector Formulation

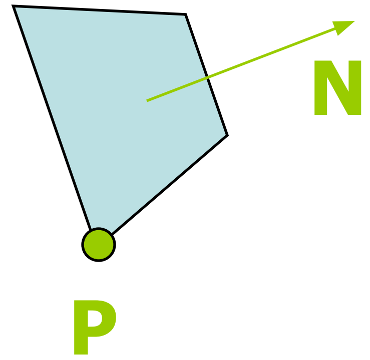
Solving for D in the plane equation:

$$Ax + By + Cz + D = 0$$

$$Ax + By + Cz = -D$$

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = -D$$

$$\mathbf{N} \cdot \mathbf{P} = -D$$



Testing Points

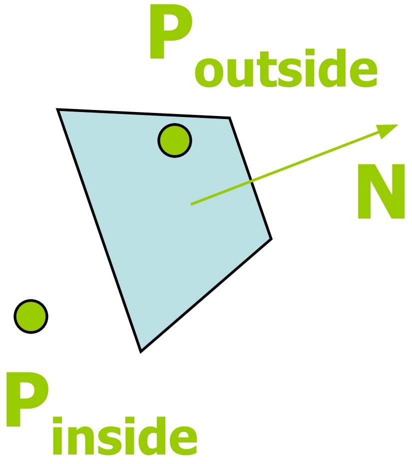
If point is "inside" the plane, then:

$$Ax + By + Cz + D < 0$$

If point is "outside" the plane, then:

$$Ax + By + Cz + D > 0$$

This can be used to determine what surfaces are hidden



Summary

In today's lecture we looked at:

- Transformations in 3-D
  - Very similar to those in 2-D
- Projections
  - 3-D scenes must be projected onto a 2-D image plane
  - Lots of ways to do this
    - Parallel projections
    - Perspective projections
  - The virtual camera