Graphics: The First Revolution

- The birth of Raster Graphics three decades ago
- Consumer-driven demand for TVs
- VLSI-driven fall in memory prices
- Affordable graphics systems with CRTs and framebuffers

Raster Graphics Architecture

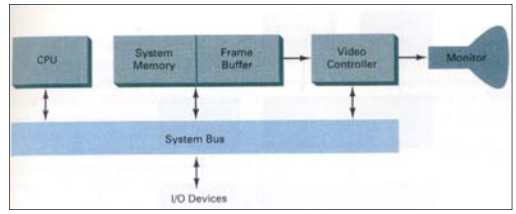


Image from *Computer Graphics* by Hearn and Baker

Graphics: The Second Revolution

- Consumer-driven demand for Games
- The birth of Graphics Processing Units (GPUs)
- Dramatic increase in GPU performance
- High-resolution Displays (LCDs, HDTVs)

Display-Processor-based Graphics

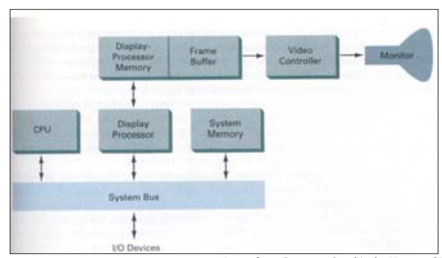


Image from *Computer Graphics* by Hearn and Baker

The Graphics Pipeline

Geometry Processing

Model
(description)
Retrieve
(database)
Transform
(eye space)
Clip
(viewing volume)
Light
(for each spectral component)

→

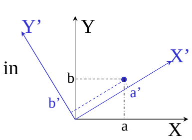
Image Processing

Scan
(primitives to pixels)
Visibility
(at each pixel)
Texture
(texture map)
Composite
(masks, other images)
Frame Buffer

Display
(LCD/DMD/HMD)

Transformation

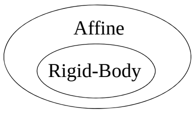
- Consider transforming the coordinate system from (X, Y) to (X', Y') with points stationary.
- For every point (a, b), we need to compute its representation (a', b') in new coordinate system.



- Let $X' = \alpha X + \beta Y$, $Y' = \gamma X + \delta Y$
- Thus $\begin{pmatrix}X'\\Y'\end{pmatrix} = \begin{pmatrix}\alpha & \beta\\ \gamma & \delta\end{pmatrix}\begin{pmatrix}X\\Y\end{pmatrix}$ or $\mathbf{P'} = \mathbf{M}\,\mathbf{P}$

2D Transformations

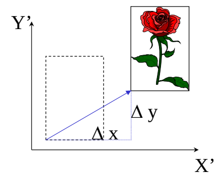
- Need transformations in graphics for:
  - Constructing scenes (hierarchical, articulated joints, animation)
  - Viewing (perspective, object to world, world to screen, …)
  - Navigating (pan, zoom, tilt, walk, …)
  - Manipulating (picking, dragging, placing, …)
- Types of common graphics transformations:
  - Rigid body transformations (preserve lengths and angles)
  - Affine transformations (preserve parallel lines)



Translation

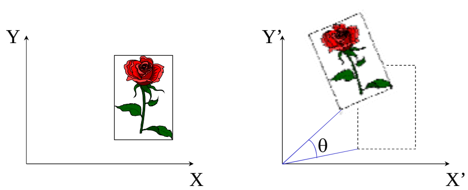
- Displaces points by a fixed distance
- $x' = x + \Delta x$; $y' = y + \Delta y$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$



Rotation

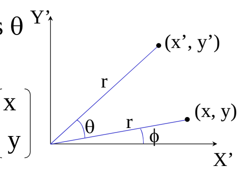
- With respect to fixed point (origin)
- Amount of rotation: θ



Rotation

- $x = r \cos \phi$; $y = r \sin \phi$
- $x' = r \cos (\theta + \phi)$; $y' = r \sin (\theta + \phi)$
- $x' = x \cos \theta - y \sin \theta$
  $y' = x \sin \theta + y \cos \theta$

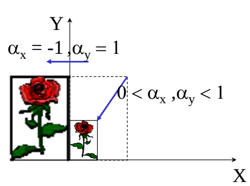
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$



Scaling

- Relative to the origin
- Scale factor α
  - $\alpha > 1$: Magnify
  - $0 < \alpha < 1$: Diminish
  - $\alpha < 0$: Reflect

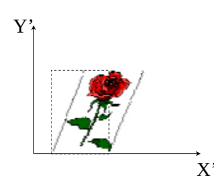
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \alpha_x & 0 \\ 0 & \alpha_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$



Shearing

- Distortion along one axis is a function of the value along the other

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & h_x \\ h_y & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$



Compositing Transformations

- Concatenate transformations right-to-left:
  - Shear (H) then Rotate (R) then Scale (S):
    P' = (S R H) P
- Scaling, shearing, and rotation can be concatenated but not translation
  - Why is concatenation so important?
    - Allows pre-multiplication of all matrices into one
    - Run-time multiplication of one matrix with all points

Translation is a Problem!

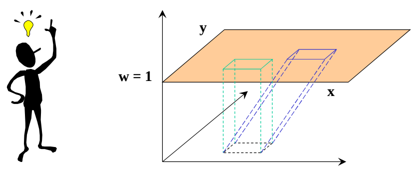
2D translation can not be represented by a 2D matrix multiply

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} & T & \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Eg: Can not move the origin by a 2×2 multiply

Homogeneous Coordinates

Idea:



- Place the xy plane away from origin in 3D
- 2D translation then becomes 3D shear
- This is the Homogeneous Coordinate System

2D Translation is a 3D Shear

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Other Homogeneous Transforms

Rotation $\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} \alpha_x & 0 & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Scaling

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & h_x & 0 \\ h_y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Shearing

Interpretation of the Transformation Matrix

$$\begin{pmatrix} a & b & e \\ c & d & f \\ g & h & i \end{pmatrix}$$

- a, b, c, d: Rotation, Scaling, Shearing
- e, f: Translation
- i: Overall scaling
- g, h: Projection to arbitrary planes ($w \neq 1$)

Efficiencies in Transformation

- Save on transforming common vertices
  - Polylines, Polygons, Triangle Fans and Strips
- Store only the 2 × 3 matrix
  - Special matrix multiply: 4 multiplies and 4 adds
  - Save space for 3 floats

$$\begin{pmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{pmatrix}$$

Homogeneous Coordinates

- Point (x, y) in 2D becomes line through (0, 0, 0) and (x, y, w)
- Homogenizing
  - Convert arbitrary (x, y, w) to (x/w, y/w, 1)
  - Analogous to normalizing a vector
    - Normalized vectors lie on a sphere with radius = 1
    - Homogenized points lie on the plane w = 1

Homogeneous Coordinates

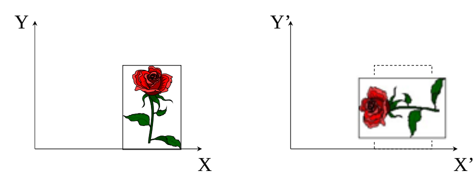
- Interpretation of w = 0 points:

$$\begin{aligned}(x, y, 0) &= \lim_{\delta \to 0} (x, y, \delta) \\ &= \lim_{\delta \to 0} (x/\delta, y/\delta, 1) \\ &= \text{infinity in the direction } (x, y)\end{aligned}$$

- Convenient way to represent points (viewpoints, light sources) at infinity

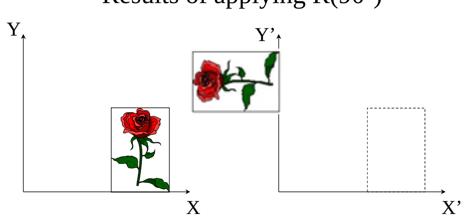
Aggregating Transformations

Task: Rotate by 90 degrees



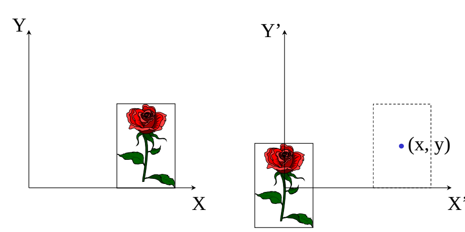
Aggregating Transformations

Results of applying R(90^o)



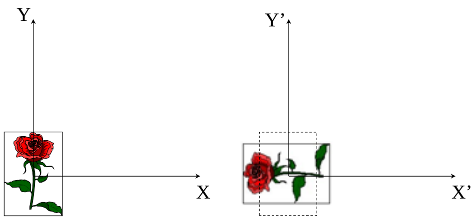
Problem: Rotation is defined about the origin

Aggregating Transformations



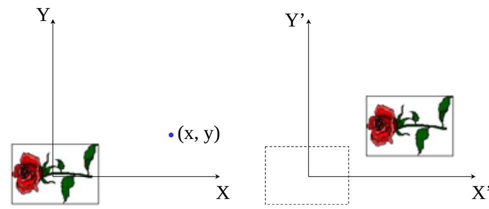
Step 1: Translate first to the origin : T (-x, -y)

Aggregating Transformations



Step 2: Rotate by 90°: R (90°)

Aggregating Transformations



Step 3:Translate back to (x, y): T(x, y)

Aggregating Transformations

Translate (-x, -y)

Rotate by 90°

Translate (x, y)

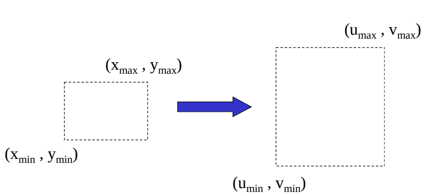
Aggregated Transform:

$$T(x, y)\ R(90^o)\ T(-x, -y)$$

Commutative 2D Transformations

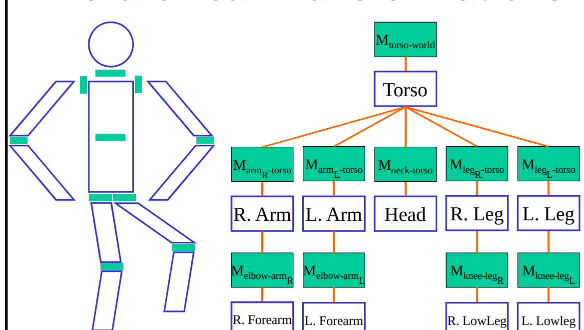
- Order is important
  - Matrix multiplication in general is not commutative: $A\,B \neq B\,A$
- What 2D transformations are commutative?
  - Rotation-rotation, translation-translation, scaling-scaling, $shear_x$-$shear_x$, $shear_y$-$shear_y$,
  - Scaling-rotation

2D World to 2D Screen Transformation

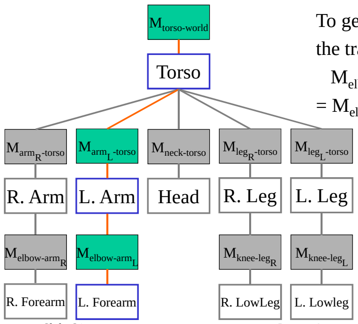


$T(u_{min}, v_{min})\ S((u_{max} - u_{min})/(x_{max} - x_{min}), (v_{max} - v_{min})/(y_{max} - y_{min})\ T(-x_{min}, -y_{min})$

Hierarchical Transformations



Hierarchical Transformations



To get the position of left arm, the transformation matrix is:

$M_{elbow\text{-}arm_L}\ M_{arm_L\text{-}torso}\ M_{torso\text{-}world}$

$= M_{elbow\text{-}\cancel{arm_L}}\ M_{\cancel{arm_L}\text{-}\cancel{torso}}\ M_{\cancel{torso}\text{-}world}$

$= M_{elbow\text{-}world}$

3D Transformations

- Coordinate systems are 3D Cartesian
- 2D transformations generalize easily (most of the time) to 3D transformations
- Represented by 4×4 homogeneous matrices

3D Transformations

$$T(\Delta x, \Delta y, \Delta z) = \begin{pmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Translation

$$S(\alpha_x, \alpha_y, \alpha_z) = \begin{pmatrix} \alpha_x & 0 & 0 & 0 \\ 0 & \alpha_y & 0 & 0 \\ 0 & 0 & \alpha_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Scaling

Shearing along x-axis: $H_x(h_y, h_z) = \begin{pmatrix} 1 & h_y & h_z & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

3D Rotations

- Axis of rotation: Line along which points remain unchanged after 3D rotation
- 3D rotations are defined by:
  - amount of rotation θ
  - axis of rotation $\bar{\mathbf{d}} = (r_x, r_y, r_z)$

3D Principal Axes Rotations

Rotation about z-axis: $R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

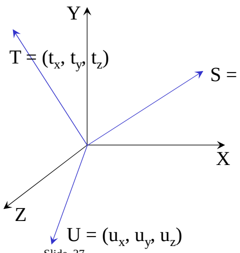
3D Principal Axes Rotations

Rotation about x-axis: $R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Rotation about y-axis: $R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Arbitrary 3D Rotations

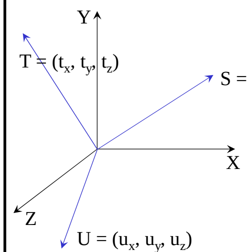
Point is stationary, coordinate system is rotated



$$R(XYZ \rightarrow STU) = \begin{pmatrix} s_x & s_y & s_z & 0 \\ t_x & t_y & t_z & 0 \\ u_x & u_y & u_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Arbitrary 3D Rotations

Point is rotated, coordinate system is stationary



$$R = \begin{pmatrix} s_x & t_x & u_x & 0 \\ s_y & t_y & u_y & 0 \\ s_z & t_z & u_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation Matrices

- Inverse of a rotation is rotation about the same axis in the reverse direction: $-\theta$: $R(-\theta) = R(\theta)^{-1}$
- Also, the row (and column) vectors of a rotation matrix have the property:
  - $r_i \cdot r_j = 0$ if $i \neq j$
  - $r_i \cdot r_j = 1$ if $i = j$
- Hence $R(\theta)R(\theta)^T = I \Rightarrow R(\theta) = (R(\theta)^{-1})^T$
- Thus, if the only transformation is rotation, normals can be transformed by $R(\theta)$.

Transformation of a Triangle

- To transform a triangle we transform each of its vertices.
- We can then recompute the triangle normal using the transformed vertices.
- Alternatively, we can transform the normal itself.

Transformation of the Normal

- Let us define the equation of a plane as $\bar{N}^T \bullet \mathbf{P} = \mathbf{0}$

$$\begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = 0$$

If we transform the point **P** by matrix **M**, we would like to find the matrix **Q** by which we should transform the normal **N**

Transformation of the Normal

$\bar{N}^T \bullet \mathbf{P} = \mathbf{0}$ **(1)**

$(\mathbf{Q}\ \bar{N})^T \bullet (\mathbf{MP}) = \mathbf{0}$ **(2)**

$\bar{N}^T \mathbf{Q}^T \mathbf{M}\ \mathbf{P} = \mathbf{0}$ **(3)**

We can satisfy **(3)** using **(1)** if
$(\mathbf{Q}^T \mathbf{M}) = \mathbf{I}$ (or even c**I,** where c is a constant)

Thus, $\mathbf{Q} = (\mathbf{M}^{-1})^T$

Viewing in 3D

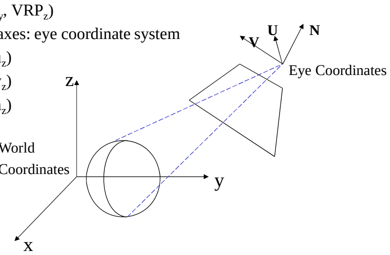
- World (3D) $\rightarrow$ Screen (2D)
- Orienting Eye coordinate system in World coordinate system
  - View Orientation Matrix
- Specifying viewing volume and projection parameters for $\Re^n \rightarrow \Re^d$ (d < n)
  - View Mapping Matrix

Slide 43

World to Eye Coordinates

The eye coordinate system is specified by:

- View reference point (VRP)
  - (VRP_x, VRP_y, VRP_z)
- Direction of the axes: eye coordinate system
  - **U** = (u_x, u_y, u_z)
  - **V** = (v_x, v_y, v_z)
  - **N** = (n_x, n_y, n_z)



Slide 44

World to Eye Coordinates

- There are two steps in the transformation (in order)
  - Translation
  - Rotation

Slide 45

World to Eye Coordinates

- Translate World Origin to VRP

$$\begin{pmatrix} a \\ b \\ c \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & -VRP_x \\ 0 & 1 & 0 & -VRP_y \\ 0 & 0 & 1 & -VRP_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Slide 46

World to Eye Coordinates

- Rotate World X, Y, Z to the Eye coordinate system *u*, *v*, *n*, also known as the View Reference Coordinate system

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ 1 \end{pmatrix}$$

Slide 47

Specifying 3D View
(Camera Analogy)

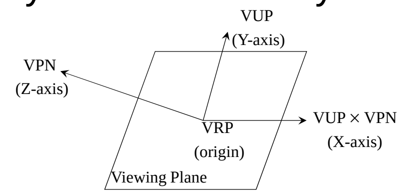
- Center of camera (x, y, z) : 3 parameters
- Direction of pointing (θ,φ) : 2 parameters
- Camera tilt (ω) : 1 parameter
- Area of film (w, h) : 2 parameters
- Focus (f) : 1 parameter

Slide 48

Specifying 3D View

- Center of camera (x, y, z) : View Reference Point (VRP)
- Direction of pointing (θ,φ) : View Plane Normal (VPN)
- Camera tilt (ω) : View Up (VUP)
- Area of film (w, h) : Aspect Ratio (w/h), Field of view (fov)
- Focus (f) : Will not consider for now

Eye Coordinate System



- View Reference Point (VRP)
- View Plane Normal (VPN)
- View Up (VUP)

World to Eye Coordinates

- Translate World Origin to VRP
- Rotate World X, Y, Z to the Eye coordinate system, also known as the View Reference Coordinate system, VRC = (VUP × VPN, VUP, VPN), respectively:

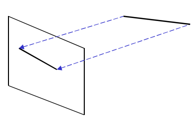
$$\begin{pmatrix} (\ VUP \times VPN\ ) & 0 \\ (\quad VUP \quad) & 0 \\ (\quad VPN \quad) & 0 \\ 0 \quad 0 \quad 0 & 1 \end{pmatrix}$$

Projection

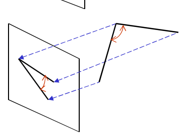
- Object representation is in 3D
  - Eg. Triangle mesh
- Screen is 2D
- Projection: World (3D) → Screen (2D)
  - Necessary to display objects on screen
- Two kinds of projection
  - Parallel projection
  - Perspective projection

Parallel Projection

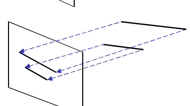
- Lengths may be scaled



- Angles may not be preserved

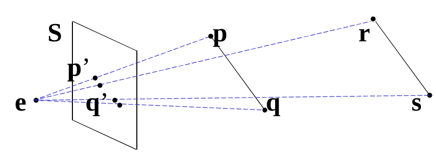


- Preserves parallelism of lines
  - If lines are parallel in 3D then their 2D projections are also parallel
  - *Note:* If lines are parallel in 2D then they need not necessarily be parallel in 3D



Perspective Projection

- Lengths may be scaled
- Angles may not be preserved
- Parallelism of lines may not be preserved
- *Perspective foreshortening:* nearer objects appear larger



Projection Summary

Both projections have the following parameters

- projection plane (**S**)
- projectors (lines)
  - Parallel projection : projectors are parallel
  - Perspective projection : projectors intersect at the center of projection **(e)**

Slide 55

Parallel Projection Example

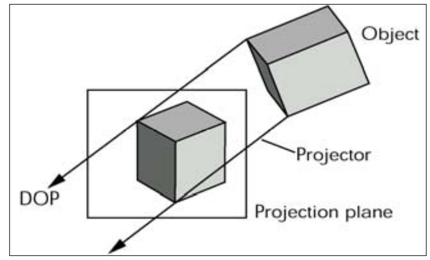


Image from: Interactive Computer Graphics by Ed Angel

Slide 56

Parallel Projection

Project on the plane, $z = 0$

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

Slide 57

Perspective Projection Example

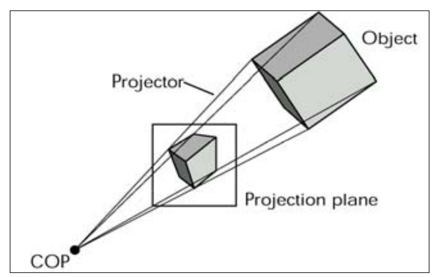
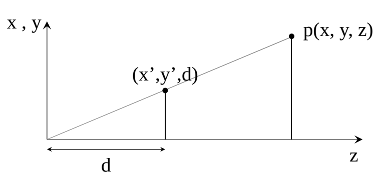


Image from: Interactive Computer Graphics by Ed Angel

Slide 58

Perspective Projection



- Let **S** be the plane $z = d$ and let **e** = (0,0,0)
- Using similar triangles:
  - $x'/d = x / z \Rightarrow x' = x / (z/d)$
  - $y'/d = y / z \Rightarrow y' = y / (z/d)$

Slide 59

Perspective Projection

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

- Homogenize (divide by $w'' = z / d$) to get:
  - $x' = x / (z / d) = x''/ w''$
  - $y' = y / (z / d) = y''/ w''$
  - $z' = z / (z / d) = z''/ w'' = d$

Slide 60