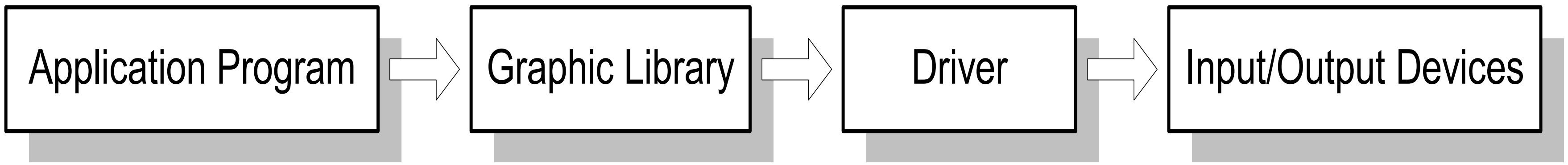
Contents

- Graphics Library
- Display 2D/3D objects
- Coordinate Systems
- Window and Viewport
- Transformation

Device Driver vs. Graphics Library

- Device Driver
  - A set of codes that controls a physical device directly with device-specific commands
  - Device dependent

- Graphics Library (cf. math library, e.g. draw a line)
  - A set of subroutines for graphics input/output functionality
  - Each subroutine == a series of a device driver commands
  - High level language binding (C, C++, FORTRAN, JAVA, etc)



Standard Graphics Library (1)

- CORE (1977)
  - ACM SIGGRAPH (revised '79)
  - Poor support to raster concept
- GKS (Graphical Kernel System, 1984)
  - ISO
  - GKS-3D : Extension to 3D
- PHIGS (Programmer's Hierarchical Graphics System, 1984)
  - ISO
  - Poor capabilities for dynamic graphics and user interaction
  - PHIGS+: Raster concepts, rendering capabilities, parametric curves and surfaces
  - PEX(1987) = PHIGS + X-Window (PHIGS extensions to X)

Standard Graphics Library (2)

- OpenGL (1996)
  - *de facto* standard in the industry
  - Originated from IRIS GL developed by Silicon Graphics, Inc.
  - Governed by ARB (Architecture Review Board)
    - DEC, Evans & Sutherland, HP, IBM, Intel, Intergraph, Microsoft, SGI, Sun Microsystems
  - Available on all UNIX workstations as well as the Windows 95 and NT PC's
  - C, C++, Ada and FORTRAN language bindings

Output Primitives

- Graphics elements displayed by a graphics library
- Common output primitives
  - Line / Polyline
  - Polygon
  - Marker / Polymarker
  - Text

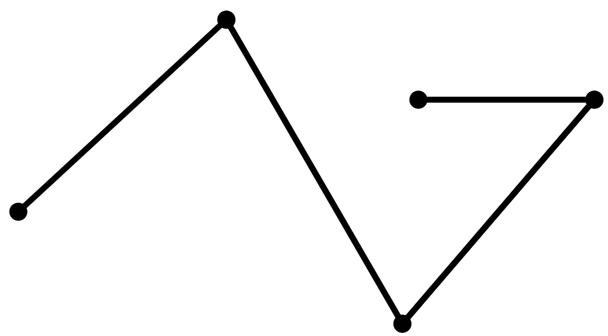
Line / Polyline

- A straight line segment defined by two end points
- Attributes : type, thickness, color, etc.
- Polyline is default in GKS, PHIGS, and OpenGL

Polyline Capability

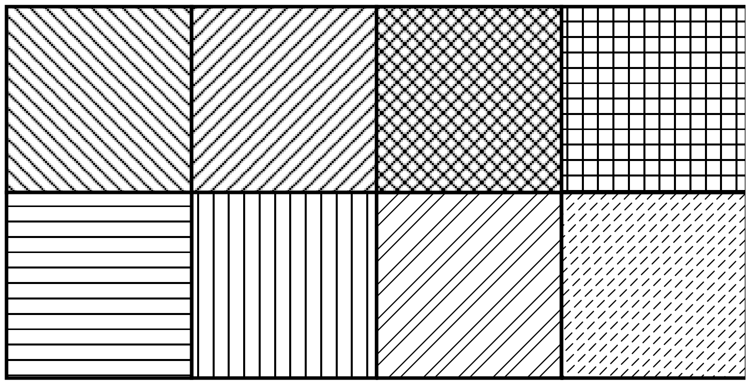
- A set of connected line segments defined by a point sequence
- Default line drawing in GKS, PHIGS and OpenGL

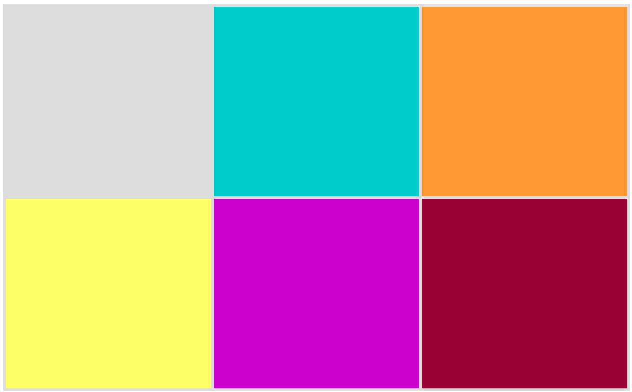
$$[p] = \begin{bmatrix} X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ X_n & Y_n & Z_n \end{bmatrix}$$



Polygon

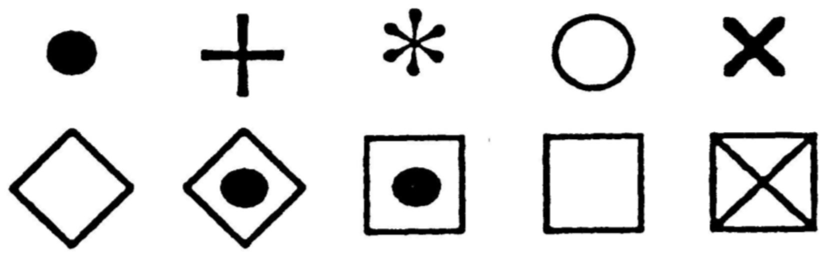
- A set of straight line segments closing a region on a plane (inside and outside information)
- The first and the last points for the polyline function are coincident
- Attribute: fill type





Marker / Polymarker

- Symbol placed at specific location
- Attribute: type



- Polymarker is a default in GKS and PHIGS
- OpenGL: not support explicitly, but any marker can be defined in a bitmap

Text

- A character string placed at a given location
- Annotation text (screen text or 2-D text) and 3-D Text
- Attribute: size, font (HW and SW font), ratio of height to width, slant angle, direction of text line, color, thickness, etc.

A B C **ABC** ***ABC*** **ABC** ABC

ABC

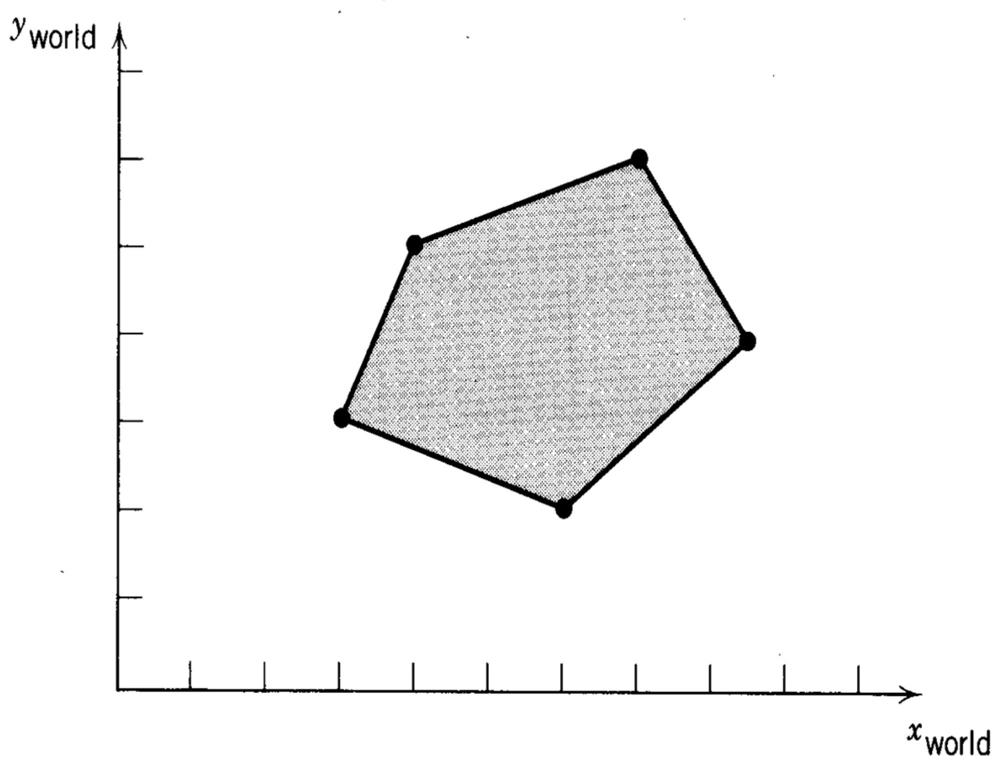
Display 2D Object

- Geometric Model $\rightarrow$ Graphical Image
  - (1) Convert the model into a screen coordinate system
  - (2) Decide how much the model is to appear and where it should appear on the screen (2D Clipping)

- 2D Coordinate Systems
  - World coordinate (WC) system
  - Device coordinate (DC) system
  - Normalized (Virtual) coordinate (NC) system

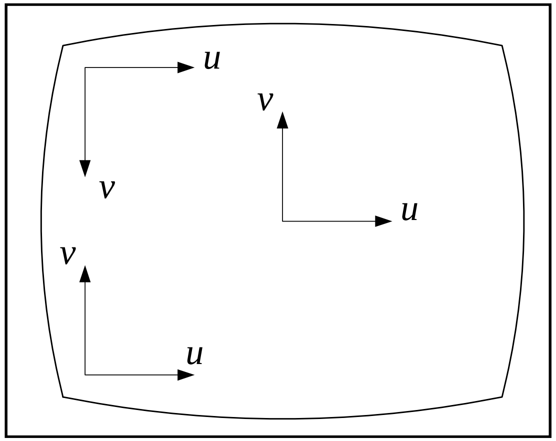
World Coordinate System

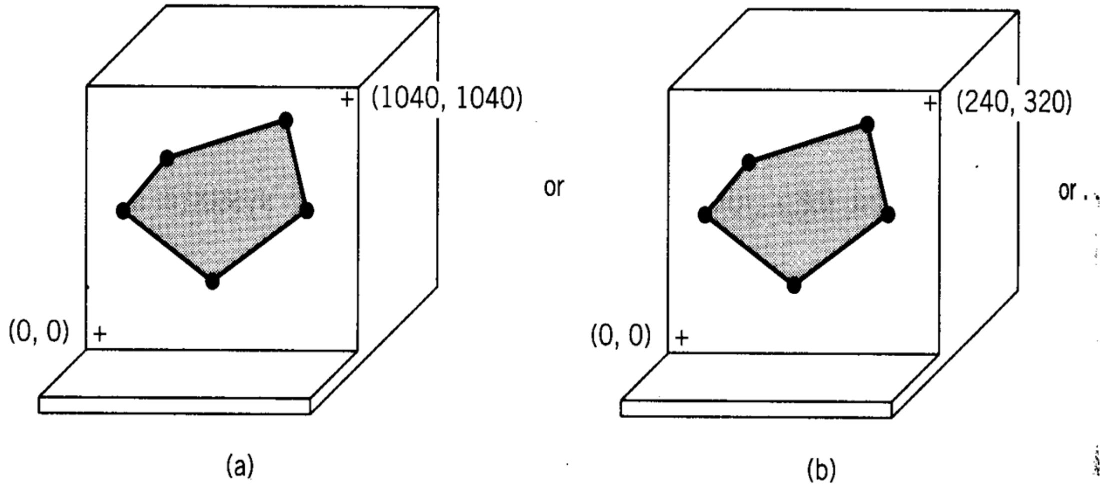
- Right-handed Cartesian system used by application program
- Actual coordinate of an object



Device Coordinate System

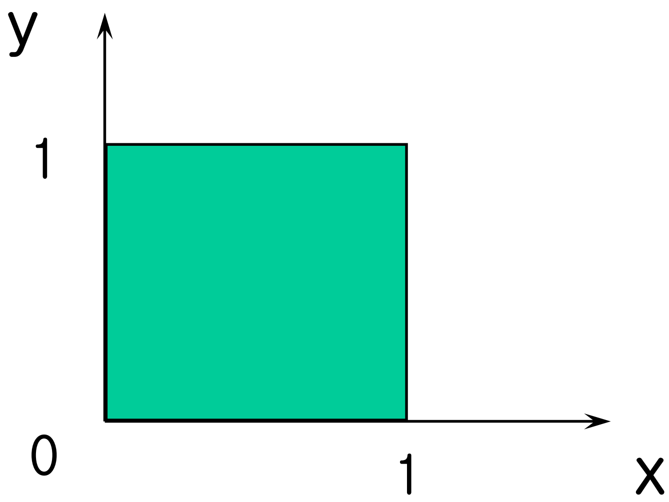
- Corresponds to the actual device used
- Device dependent ➔Lack of portability



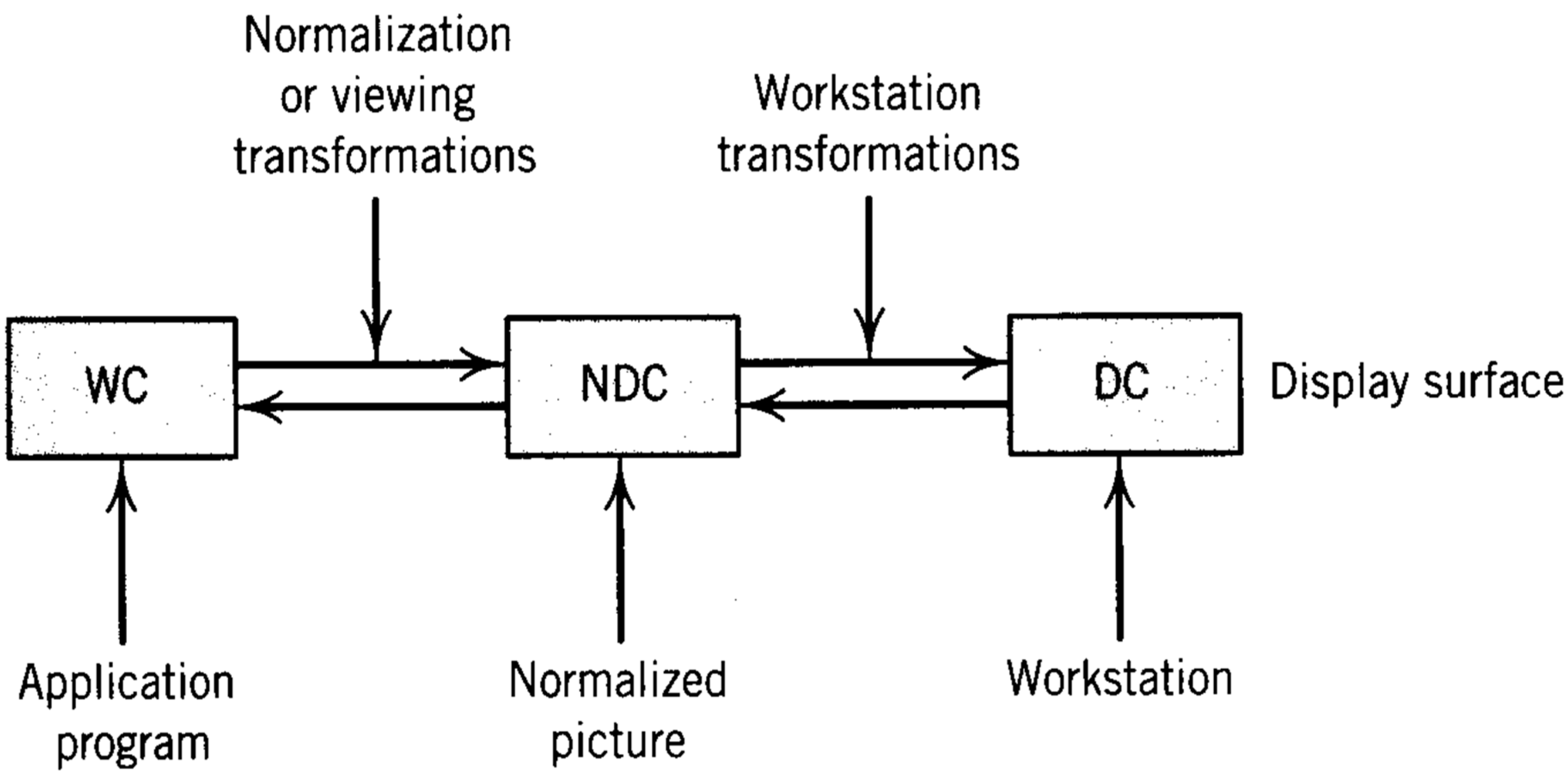


Normalized (Virtual) Device Coordinate

- Device independent
- Unit square represent display surface (screen)

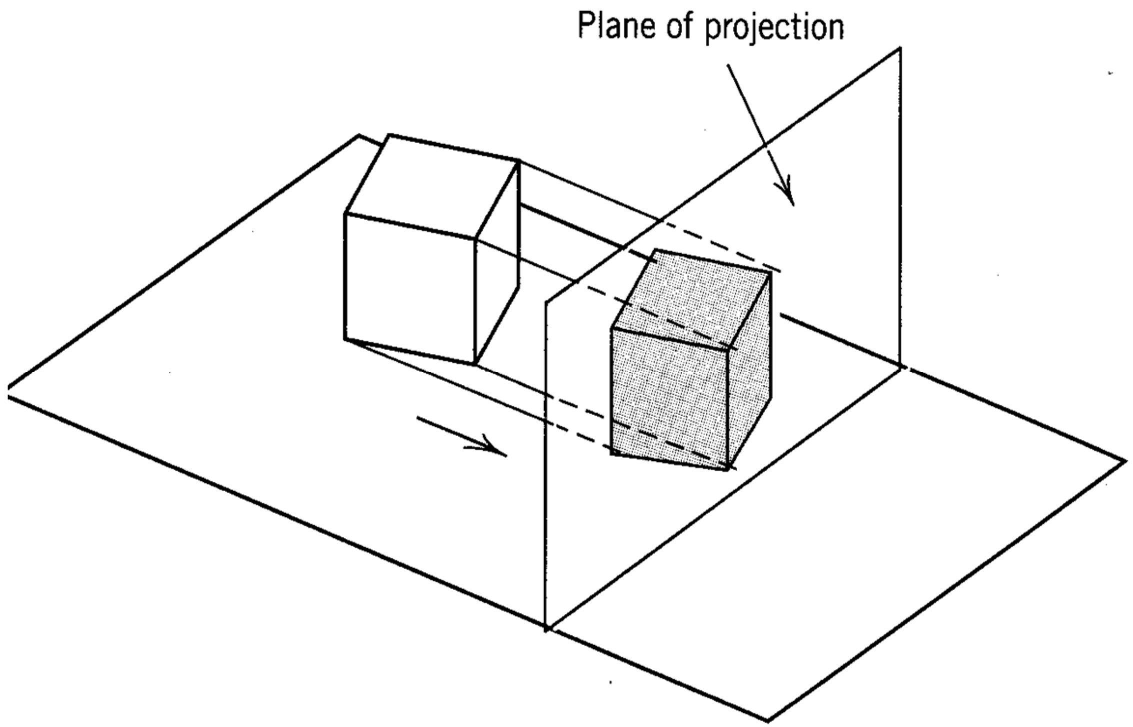


Coordinate Transformations (GKS)



Display 3D Object

- 3D Geometric Model $\rightarrow$ 2D Graphical Image
  - –Convert the model into a screen coordinate system
  - – Decide how much the model is to appear and where it should appear on the screen (Clipping)

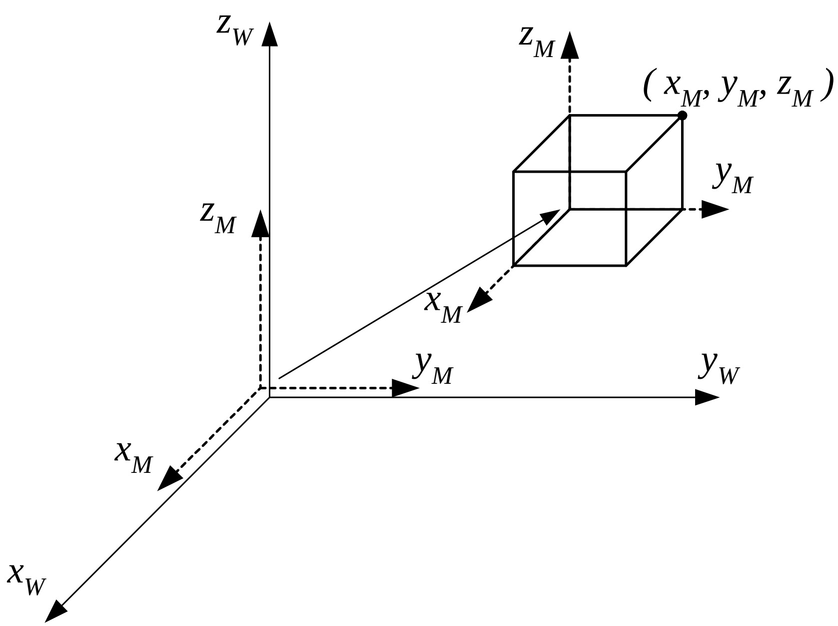


3D Viewing Operations

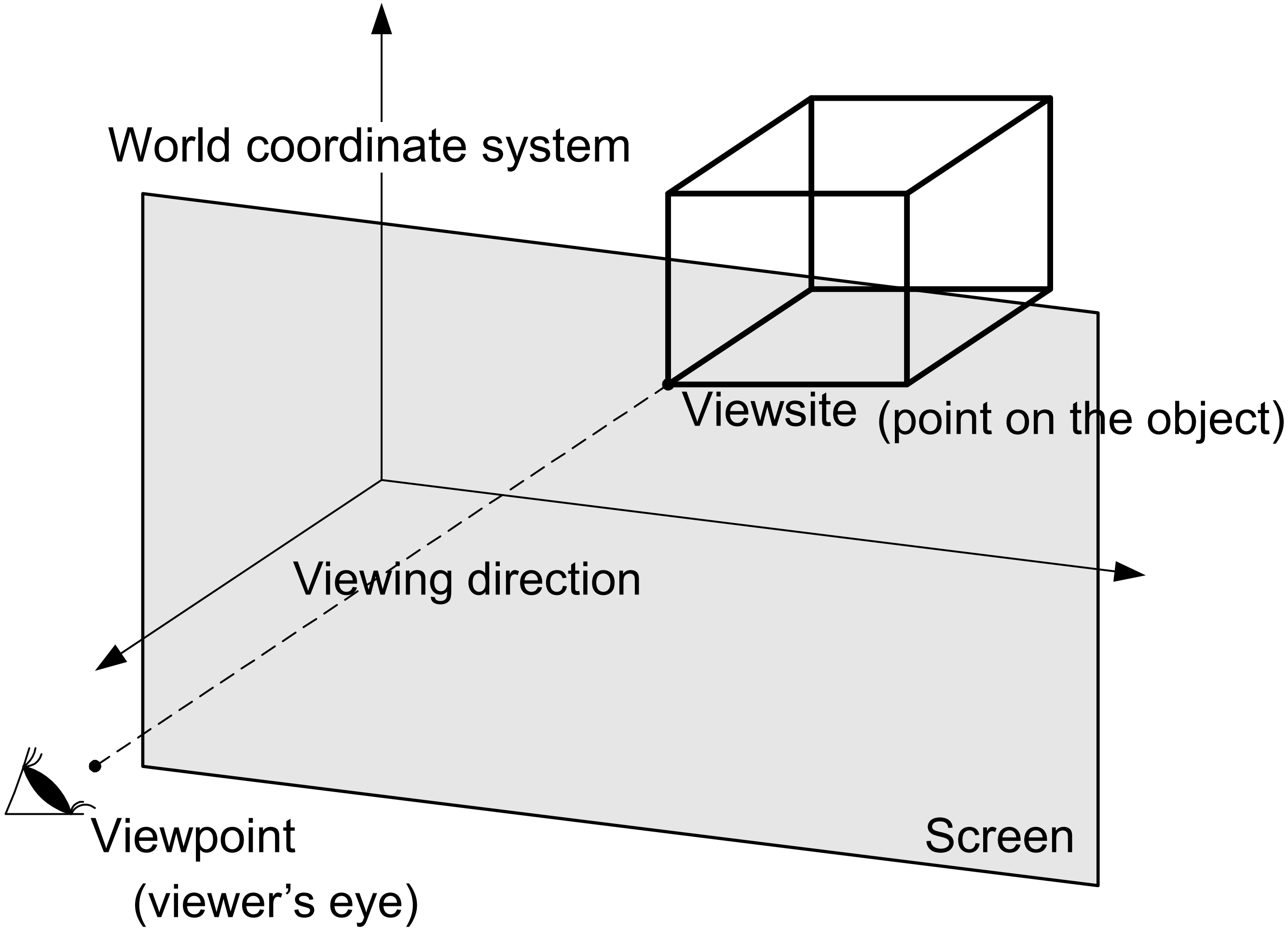
- Coordinate Systems
  - World coordinate system
  - Device coordinate system
  - Virtual (Normalized) device coordinate system
  - Model coordinate system
  - Viewing coordinate system
- Viewpoint and Viewsite
- Projections
- The Viewing Pipeline

Model Coordinate System

- A local coordinate system attached to the object
  - The values of point coordinates w.r.t model coordinate system do not changed for translation and rotation of the object

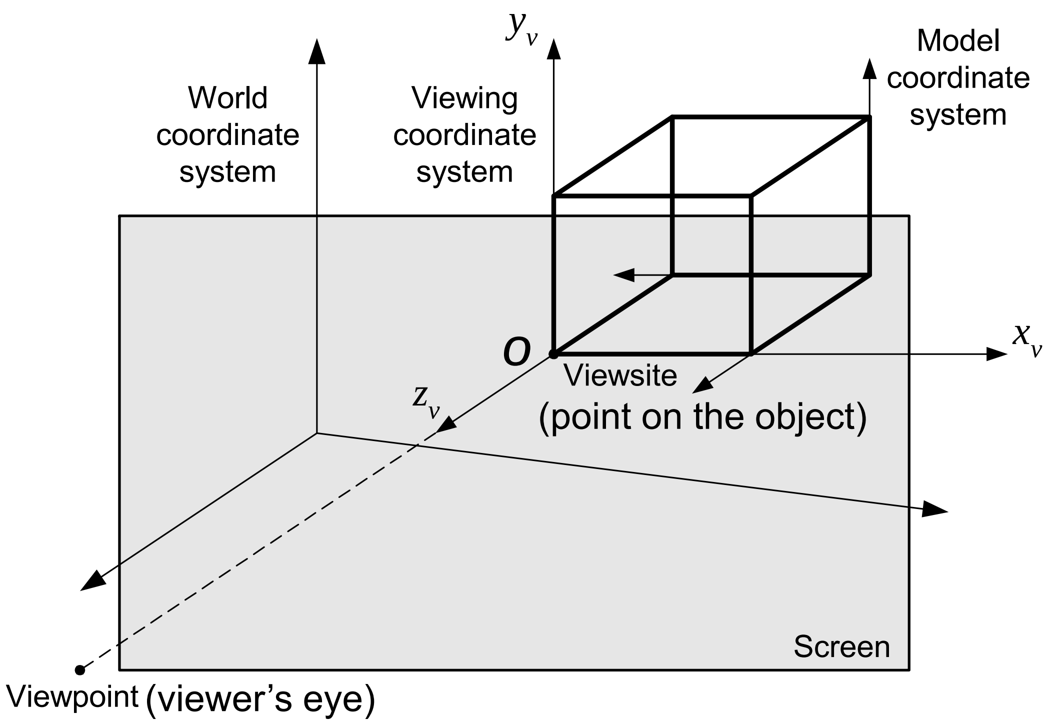


Viewpoint and Viewsite



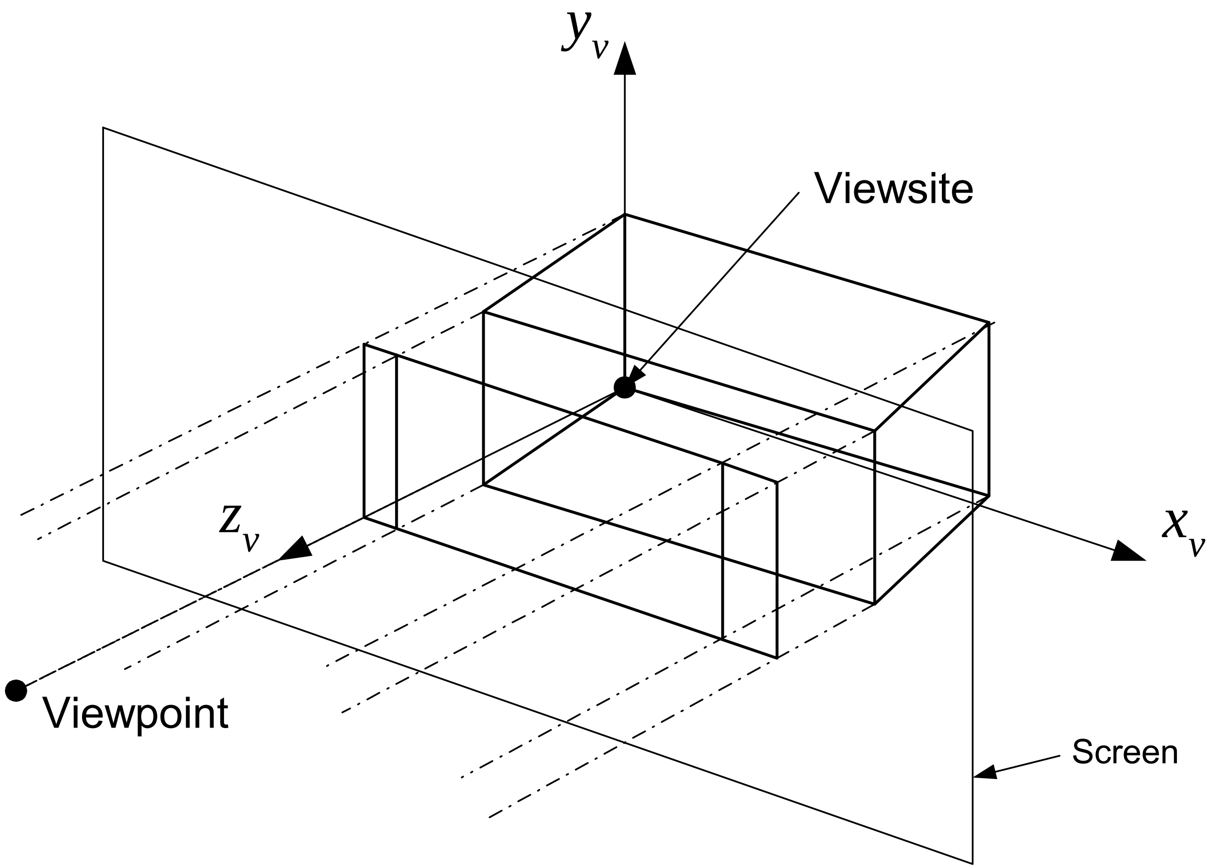
Viewing Coordinate System

For easy calculation of projection

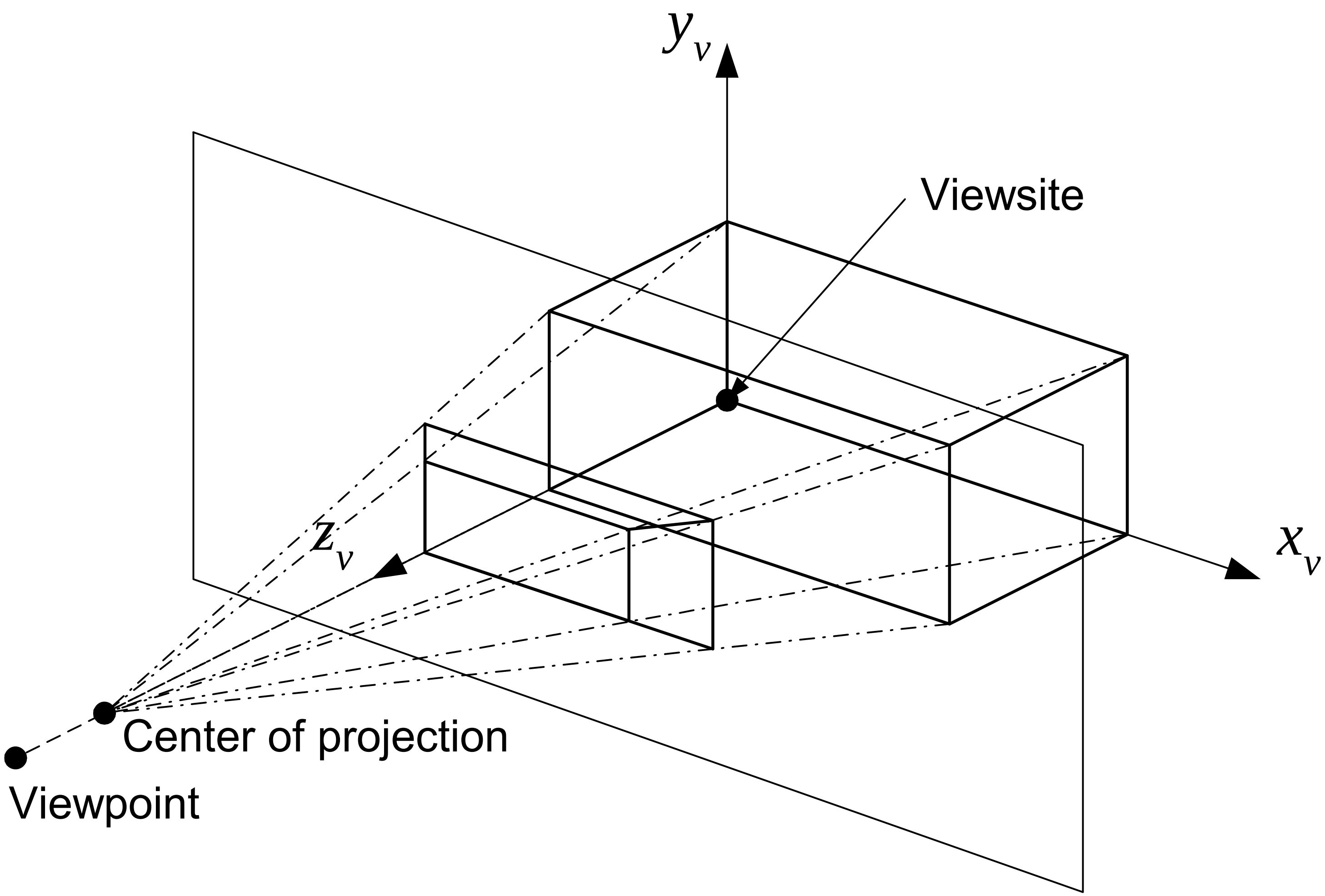


o : viewsite
z_v : viewsite --> viewpoint
y_v : vertical direction of the screen
x_v : cross product of y_v and z_v

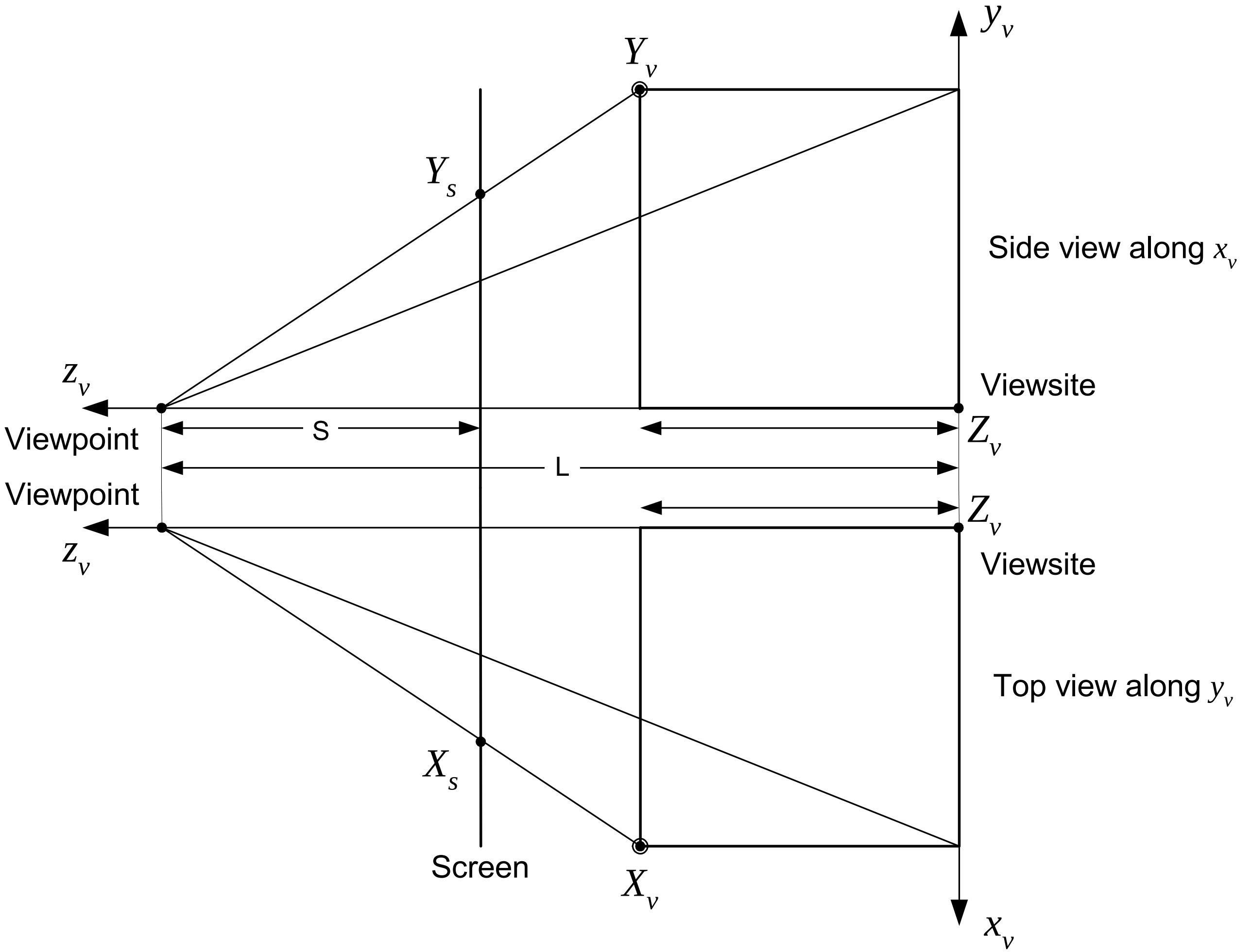
Parallel Projections (평행 투상)



1-Point Perspective Projection (투시 투상)



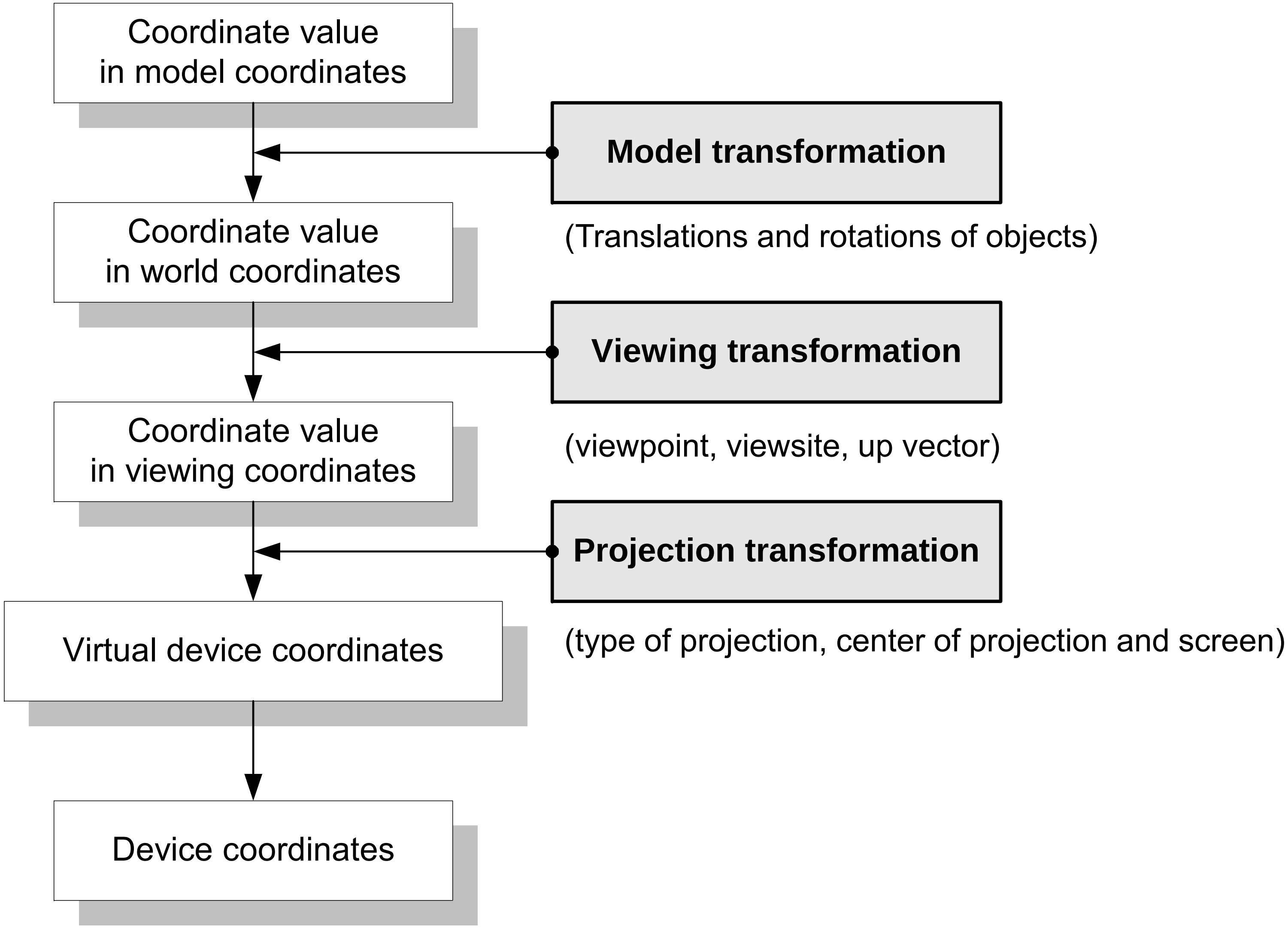
Calculation of Perspective Projection Point



$$\begin{cases} S : X_s = (L - Z_v) : X_v \\ S : Y_s = (L - Z_v) : Y_v \end{cases}$$

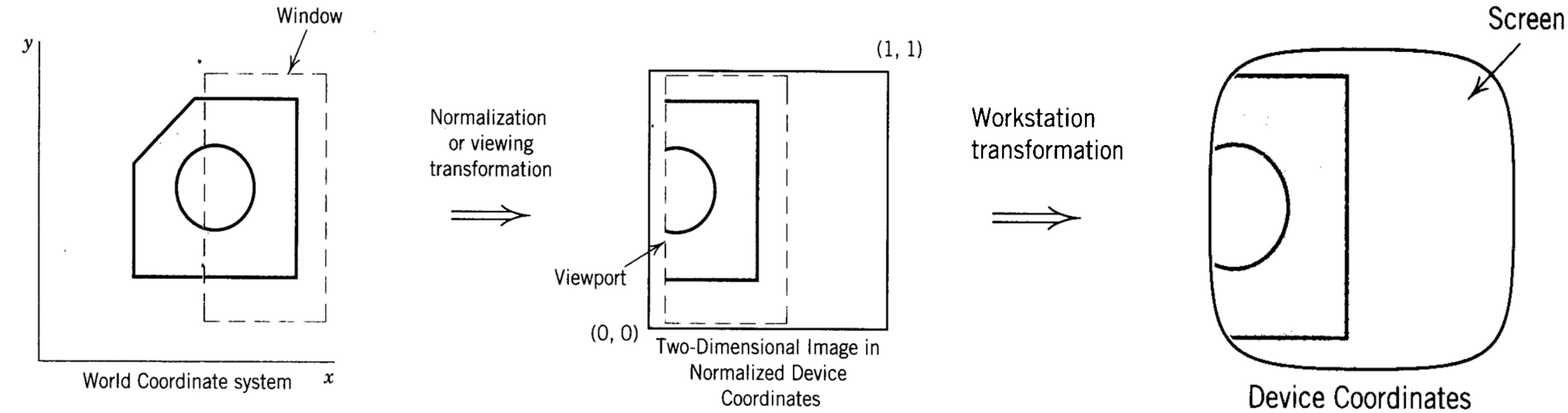
$$\begin{cases} X_s = \dfrac{S}{L - Z_v} X_v \\ Y_s = \dfrac{S}{L - Z_v} Y_v \end{cases}$$

Viewing Pipeline

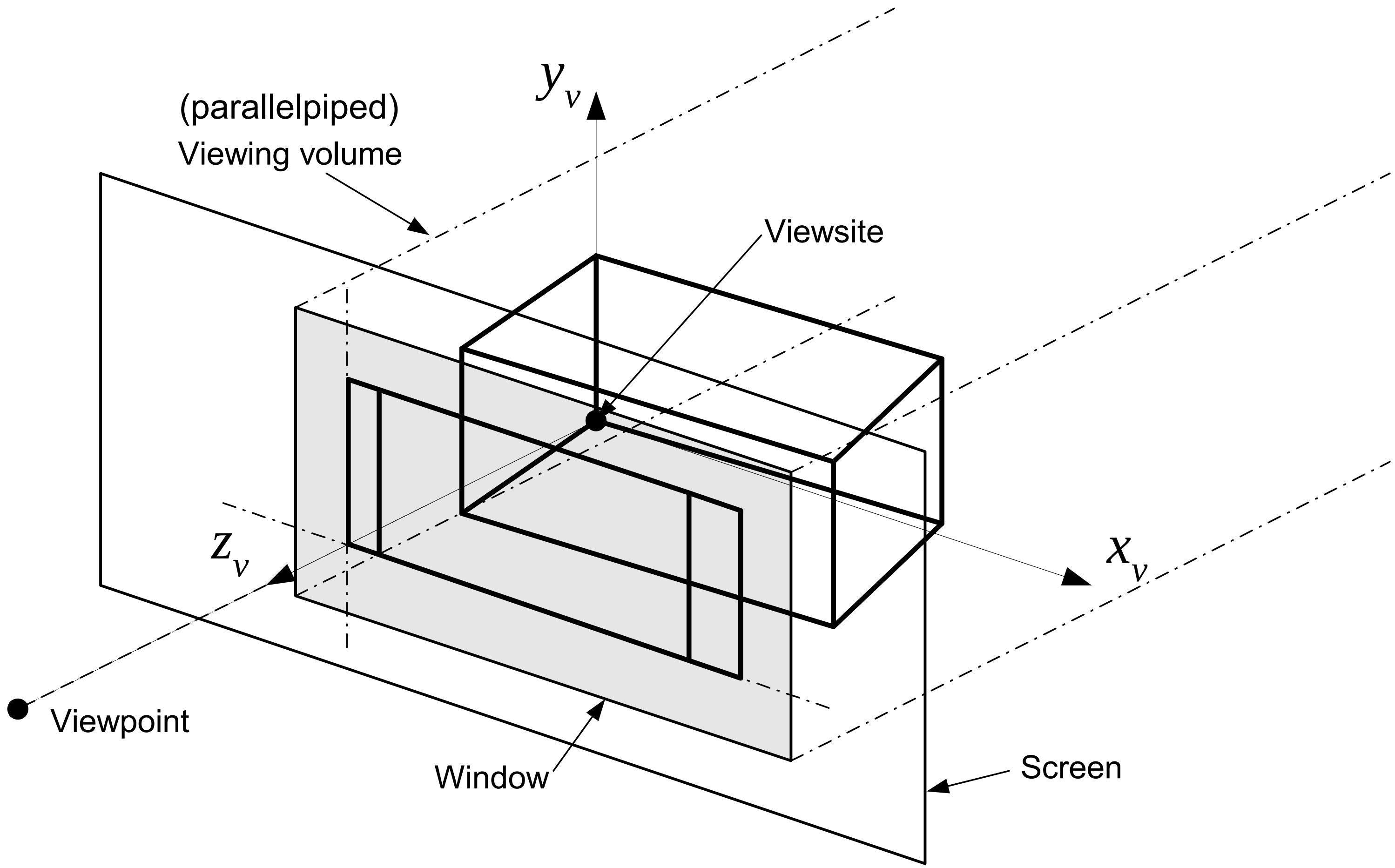


Window and Viewport Definitions

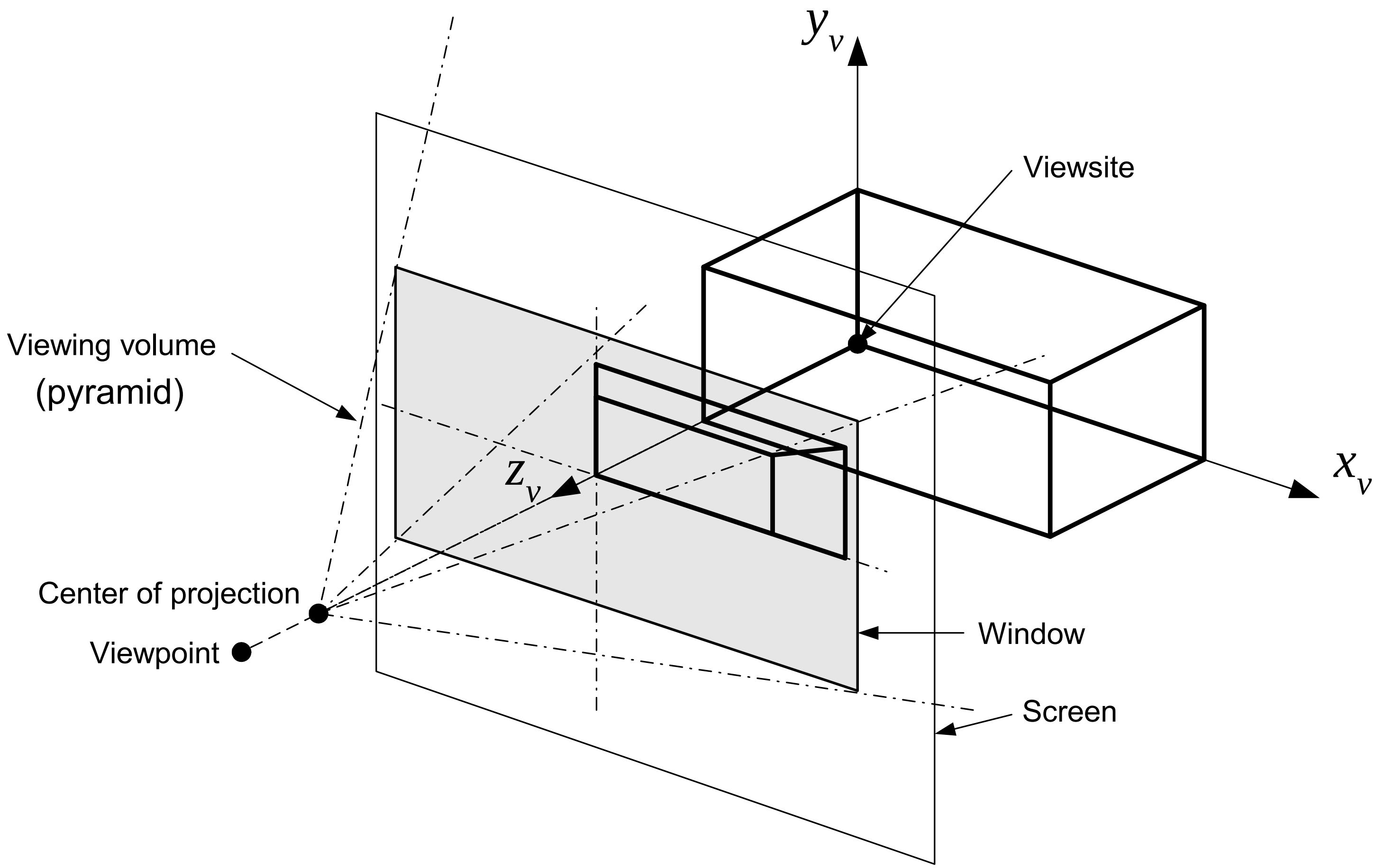
- Window : rectangular region of WC space
  - Region in space that will be projected onto the display monitor
- Viewport : rectangular region of NDC space
  - Area in the display monitor where we want the projected image to appear



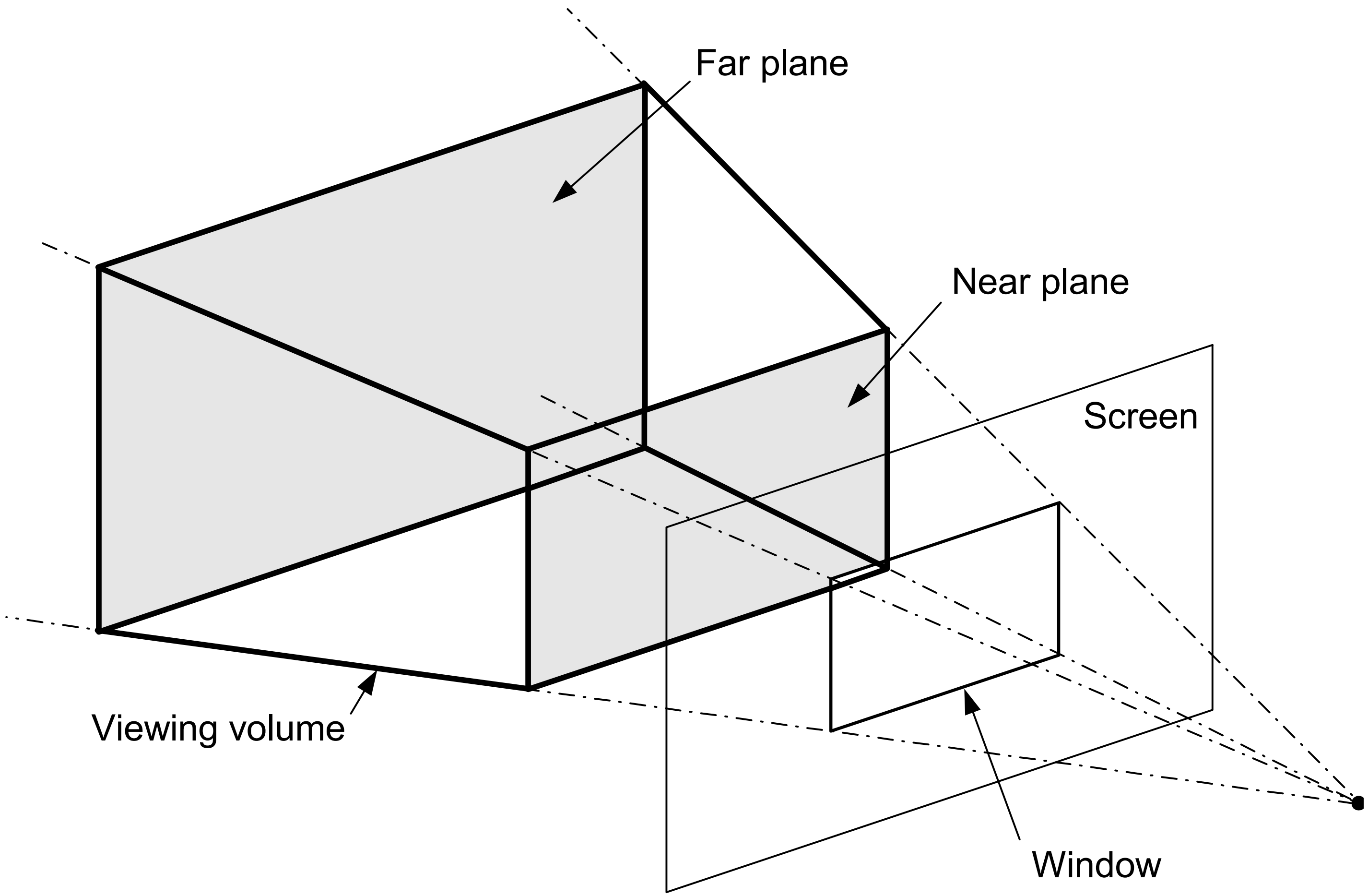
Viewing Volume for Parallel Projection



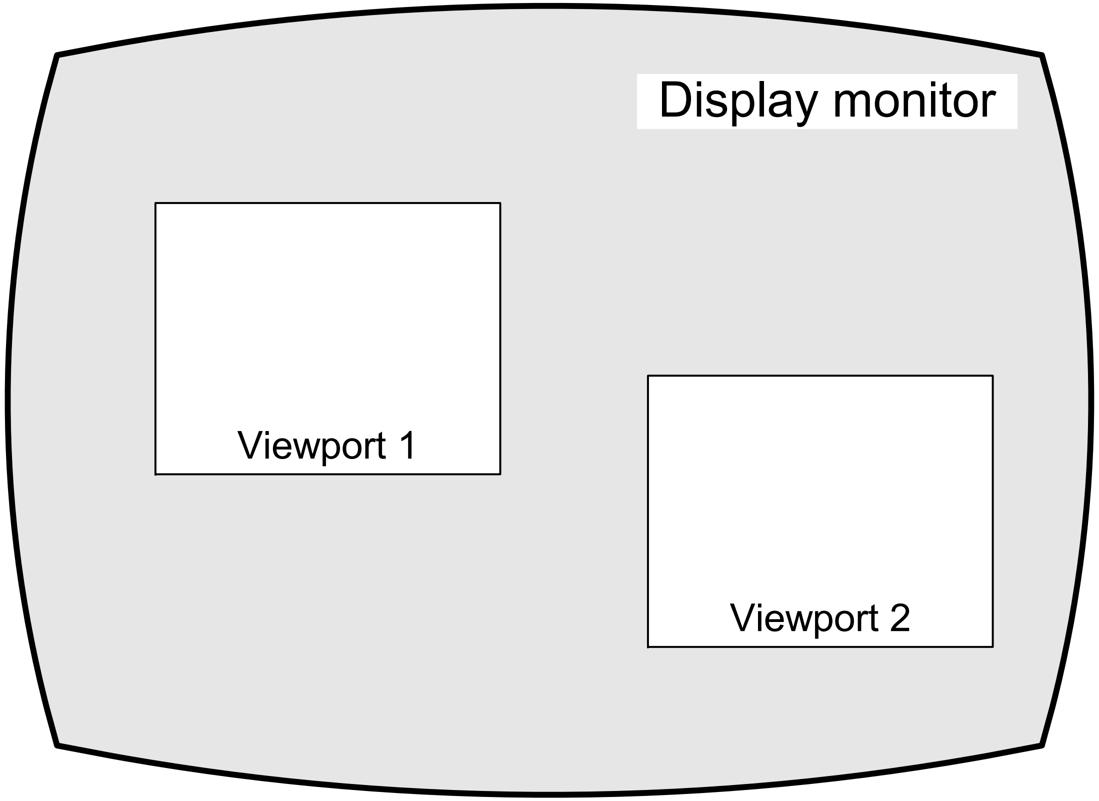
Viewing Volume for Perspective Projection



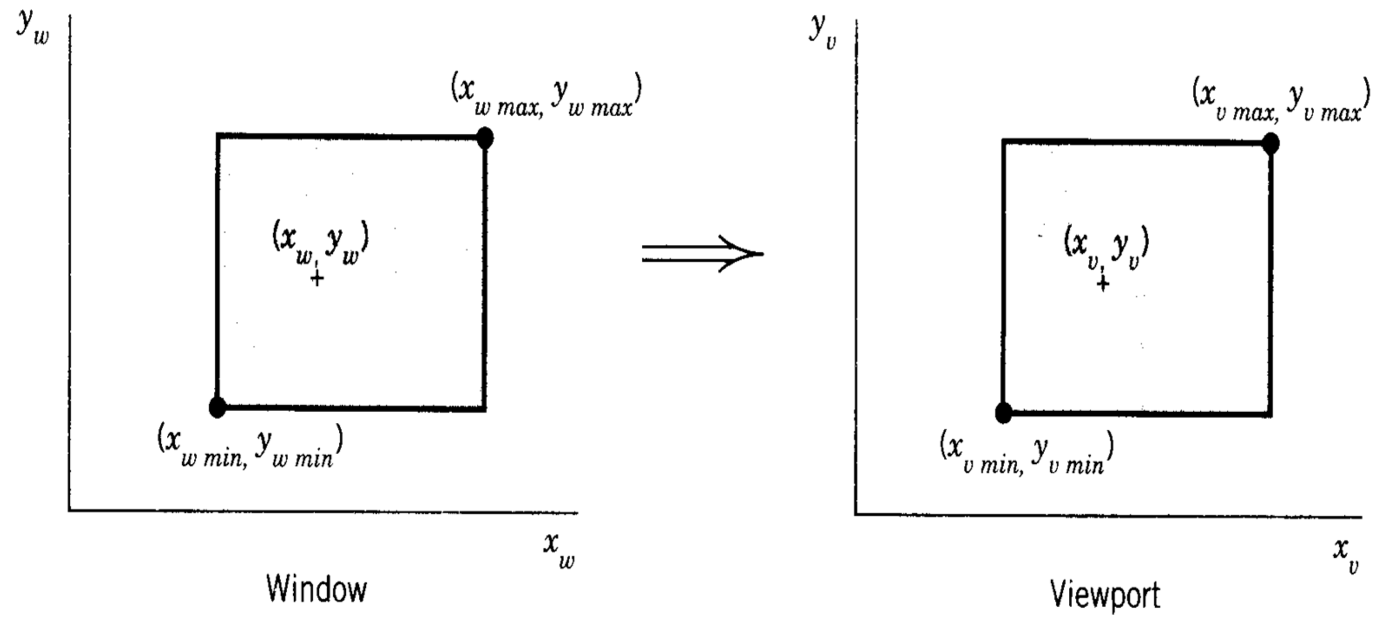
Near and Far Plane



Multiple Viewports



Window-to-Viewport Mapping



$$\frac{x_v - x_{v\,\min}}{x_{v\,\max} - x_{v\,\min}} = \frac{x_w - x_{w\,\min}}{x_{w\,\max} - x_{w\,\min}} \qquad \frac{y_v - y_{v\,\min}}{y_{v\,\max} - y_{v\,\min}} = \frac{y_w - y_{w\,\min}}{y_{w\,\max} - y_{w\,\min}}$$

$$x_v = \left(x_w - x_{w\,\min}\right)\left(\frac{x_{v\,\max} - x_{v\,\min}}{x_{w\,\max} - x_{w\,\min}}\right) + x_{v\,\min} \qquad y_v = \left(y_w - y_{w\,\min}\right)\left(\frac{y_{v\,\max} - y_{v\,\min}}{y_{w\,\max} - y_{w\,\min}}\right) + y_{v\,\min}$$

Window-to-Viewport Mapping

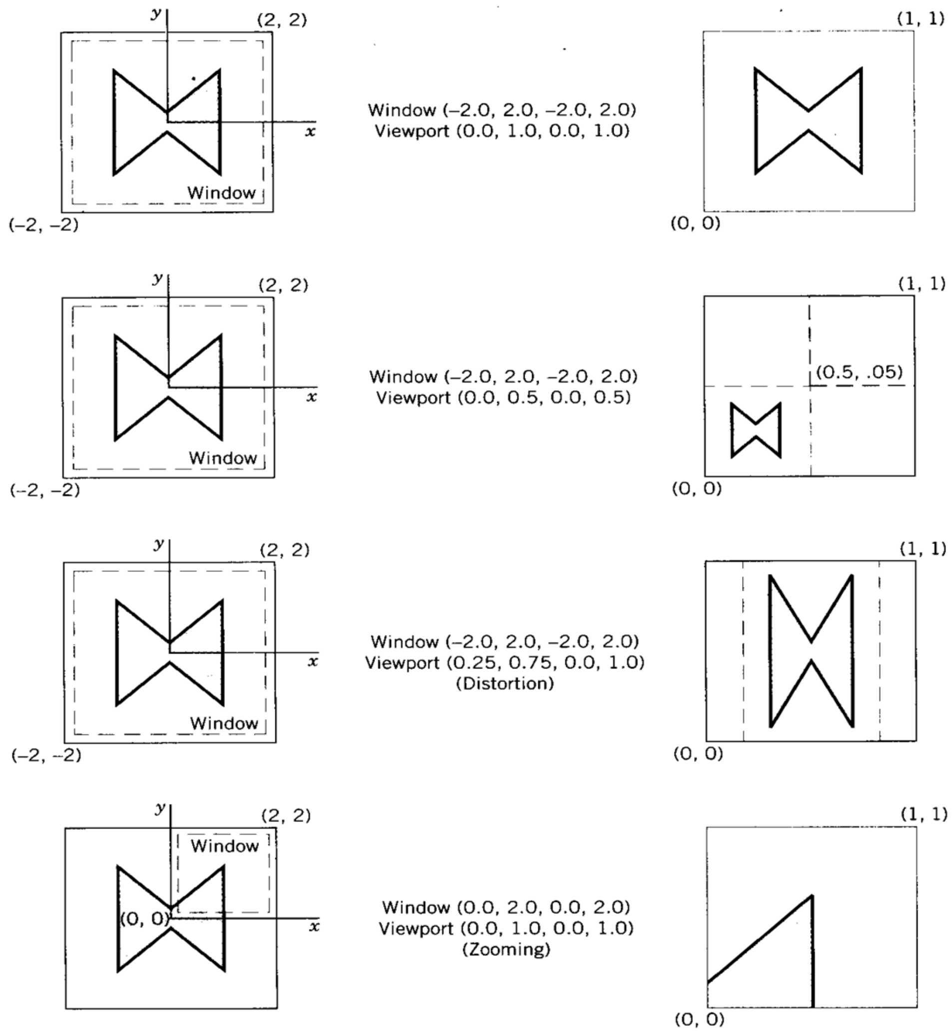
- Scaling factors : S_x, S_y
  - If S_x != S_y, distortion occurs

$$S_x = \left(\frac{x_{v\,\max} - x_{v\,\min}}{x_{w\,\max} - x_{w\,\min}} \right)$$

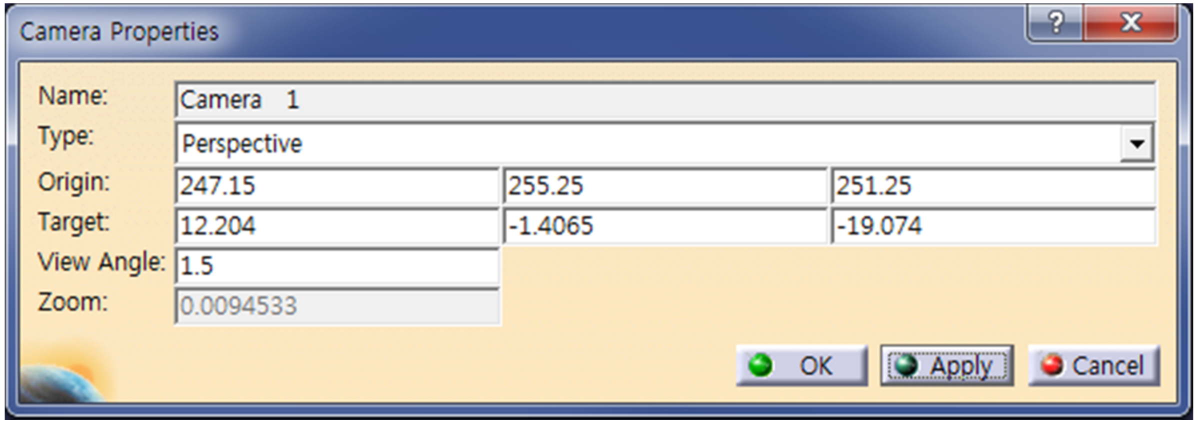
$$S_y = \left(\frac{y_{v\,\max} - y_{v\,\min}}{y_{w\,\max} - y_{w\,\min}} \right)$$

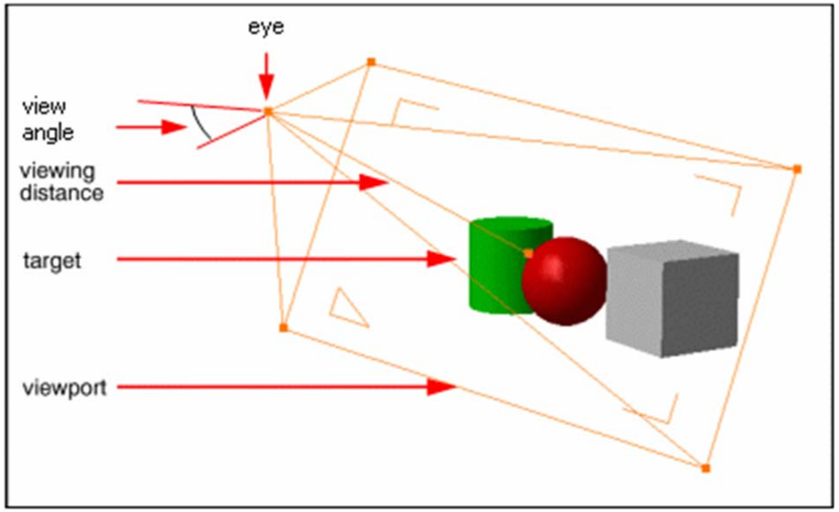
- Aspect ratio (AR)

$$AR = \left(\frac{x_{\max} - x_{\min}}{y_{\max} - y_{\min}} \right)$$



CATIA V5: View→Render Style

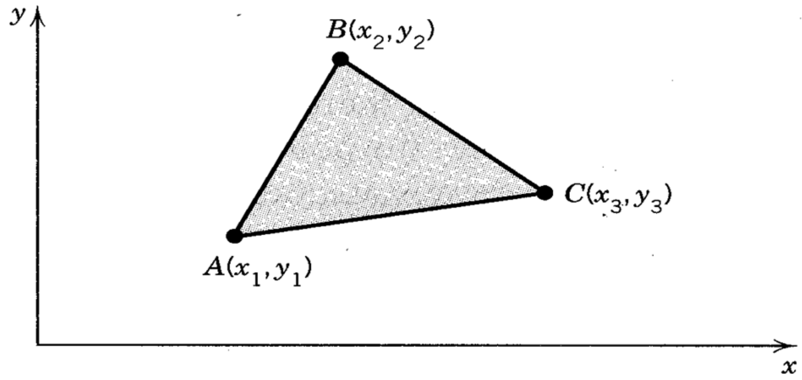




Type: lets you set a Parallel or Perspective view projection
Origin: coordinates of your eye position
Target: coordinates of the center of rotation of the camera (the point located at the center of the viewport).
You can set the center of rotation by clicking the middle mouse button on the desired point: the coordinated are memorized with the camera
View angle: sets the angle of the pyramid-like shape with which you look at the geometry (available in perspective views only).
A large angle has the effect of zooming out to make the geometry look small; a small angle has the reverse effect
Zoom: zoom factor (available in parallel views only).

Representation of 2D Geometry

- Ordinary Cartesian Coordinates



  - (Note) Some geometric transformation are obtained by matrix multiplications and others by vector additions

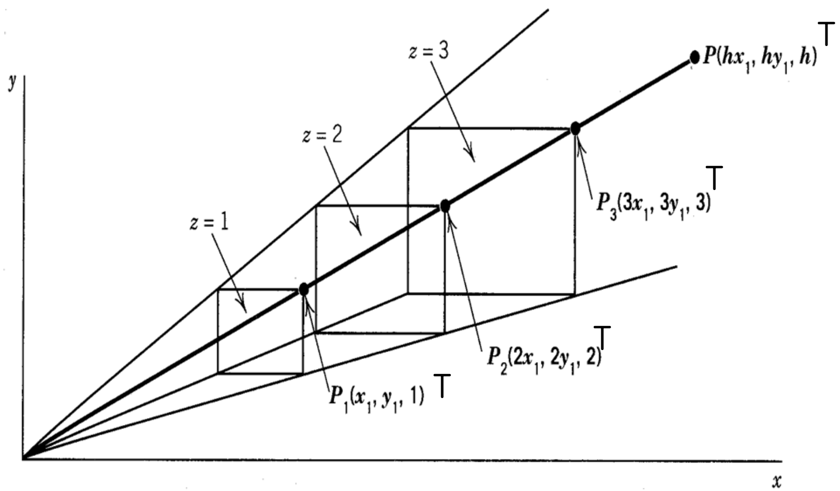
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} dx \\ dy \end{bmatrix}$$

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

Homogeneous Coordinates (1)

- Unified approach to the description of geometric transformations
- homogeneous space: ray tracing

$$P(x, y, z)^T = P(hx_1, hy_1, h)^T$$



General form: $P(hx, hy, h)^T$

$$P(m, n, h)^T \rightarrow P\left(\frac{m}{h}, \frac{n}{h}, 1\right)^T$$

triangle: $[P] = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}^T$

Homogeneous Coordinates (2)

- All geometric transformations are obtained by matrix multiplications

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} dx \\ dy \end{bmatrix} \rightarrow \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

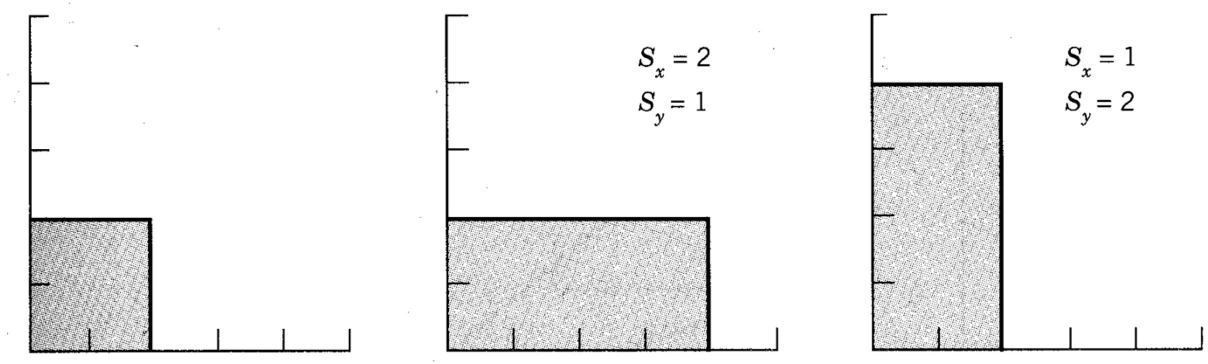
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2D Transformations

- Calculation of new coordinates
  - Rigid body transformation
    - Scaling
    - Translation
    - Rotation
- Object transformation
- Coordinate system transformation

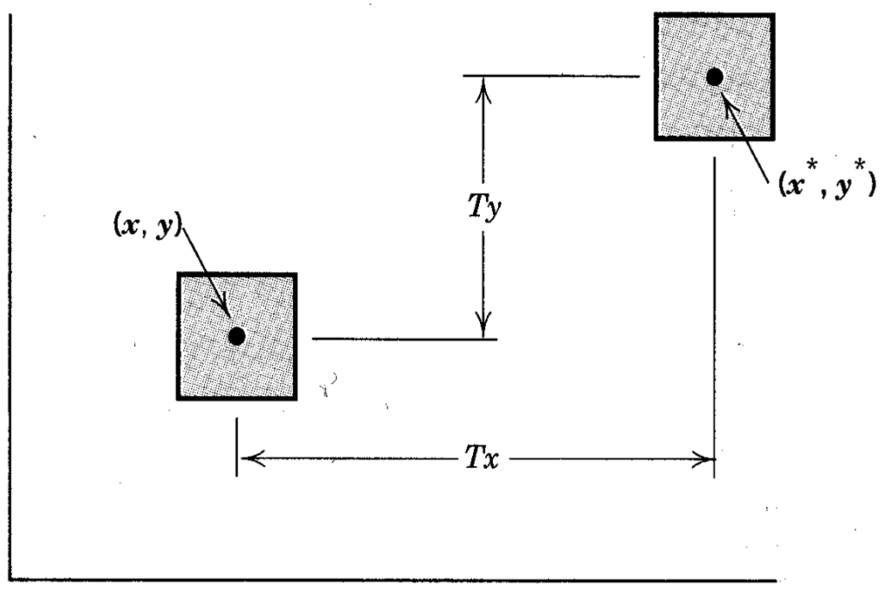
Scaling

- Scaling about the origin
- S_x, S_y : positive
  - if negative, the transformation is reflection



$$[P'] = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x x \\ S_y y \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = Scale\left(S_x, S_y\right)[P]$$

Translation

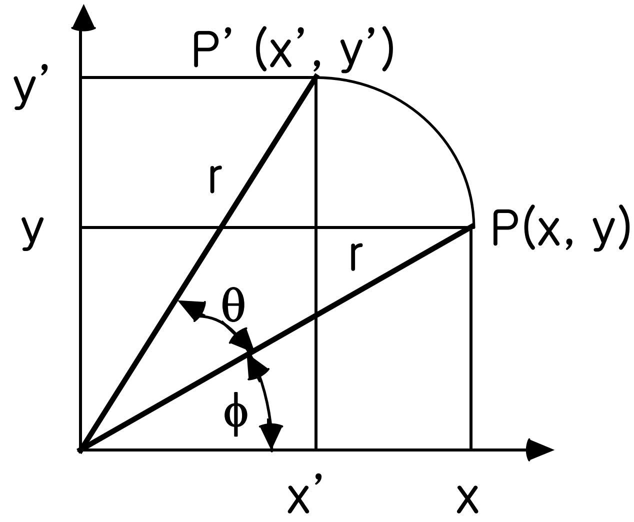


$$[P'] = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x + T_x \\ y + T_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = Trans(T_x, T_y)[P]$$

Rotation

$$\begin{cases} x = r\cos\phi \\ y = r\sin\phi \end{cases}$$

$$\begin{cases} x' = \\ \quad = \\ y' = \\ \quad = \end{cases}$$



$$[P'] = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = Rot(\theta)[P]$$

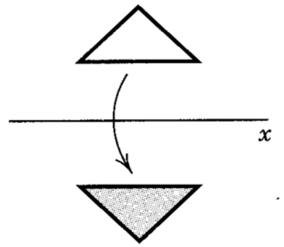
Reflection

$$Ref\left(a,b\right)=\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- A special case of scaling: negative scaling

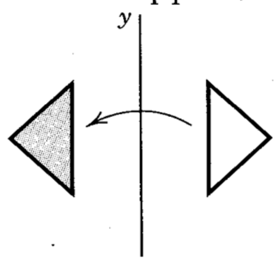
About the x axis (x values are kept and y values are flipped:)

$$[T_{\mathrm{RFL}}]_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



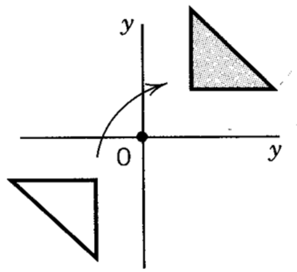
About the y axis (y values are kept and x values are flipped:)

$$[T_{\mathrm{RFL}}]_y = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



About the origin (both x and y values are flipped:)

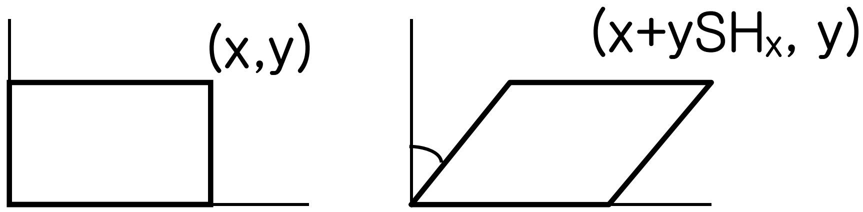
$$[T_{\mathrm{RFL}}]_o = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



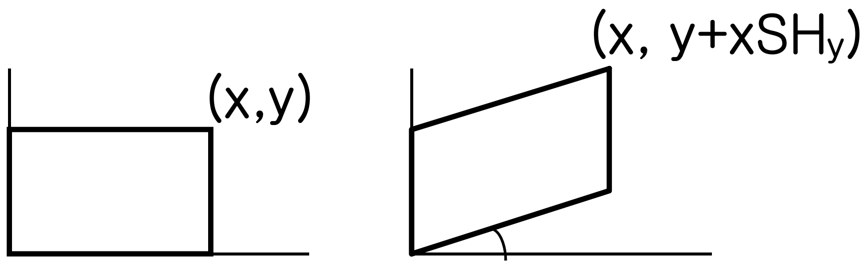
Shearing

$$Shear(SH_x, SH_y) = \begin{bmatrix} 1 & SH_x & 0 \\ SH_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

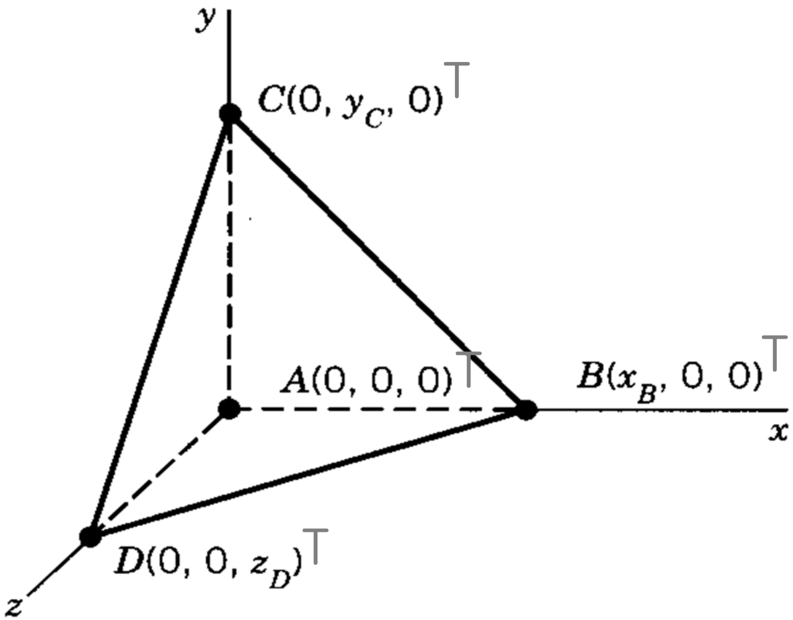
$$Shear(SH_x, 0) = \begin{bmatrix} 1 & SH_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} x + ySH_x \\ y \\ 1 \end{bmatrix}$$



$$Shear(0, SH_y) = \begin{bmatrix} 1 & 0 & 0 \\ SH_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y + xSH_y \\ 1 \end{bmatrix}$$



Representation of 3D Geometry



$$P = \begin{bmatrix} 0 & x_B & 0 & 0 \\ 0 & 0 & y_C & 0 \\ 0 & 0 & 0 & z_D \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

3D Transformation Matrix (1)

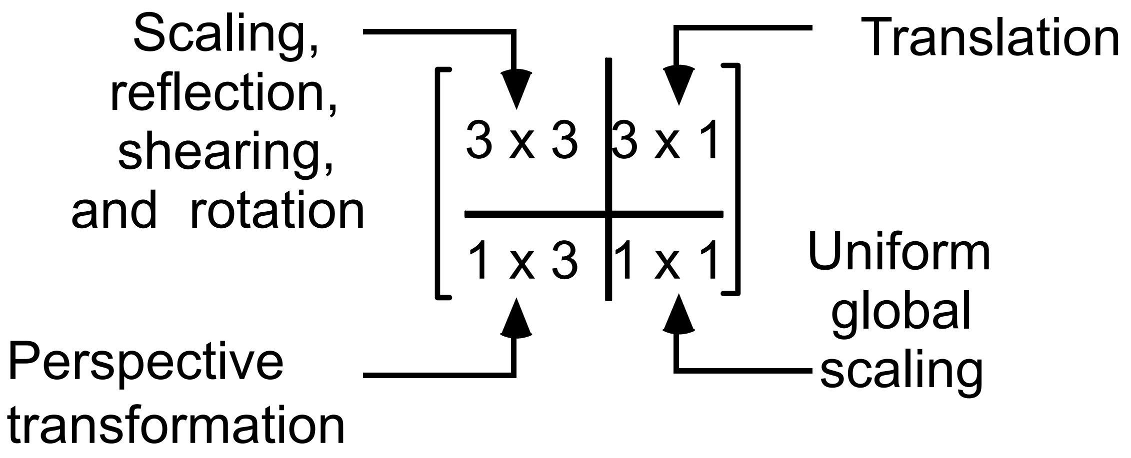
- Translation and Rotation
  - To calculate the world coordinates from the model coordinates
  - The model has been translated and rotated from the initial position

- Mapping
  - To map the world coordinate system to the viewing coordinate system
  - Calculate the viewing coordinates of the same points from the world coordinates

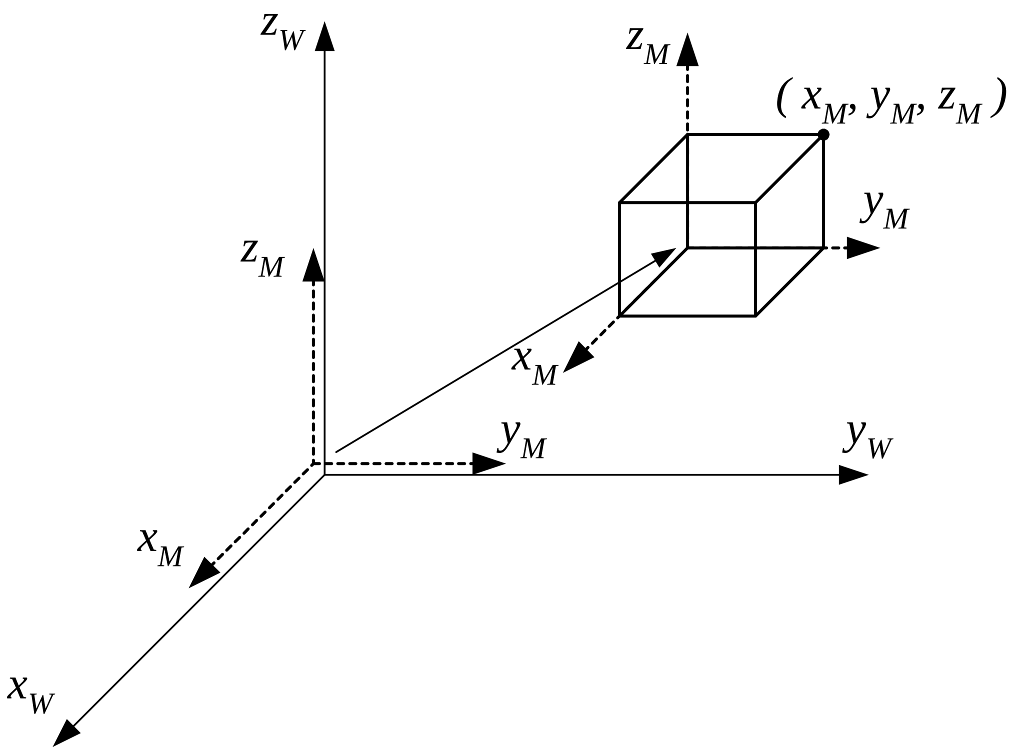
3D Transformation Matrix (2)

- 4x4 Transformation Matrix

$$P = \left[\begin{array}{ccc|c} A & B & C & J \\ D & E & F & K \\ G & H & I & L \\ \hline 0 & 0 & 0 & S \end{array}\right]$$



Translation

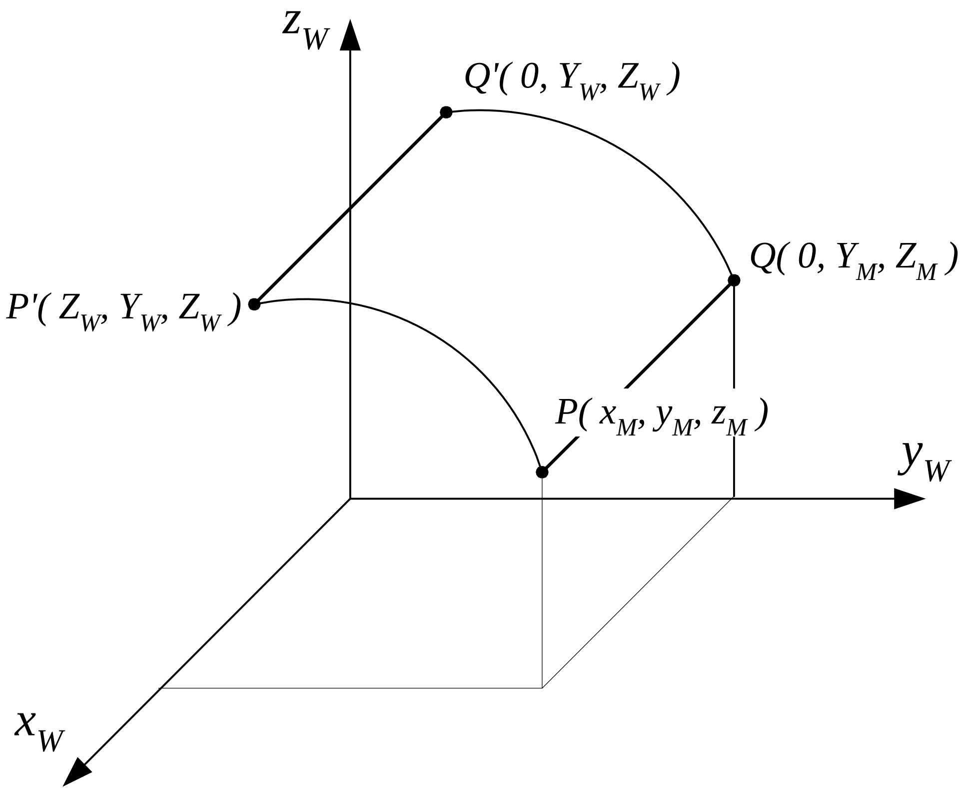


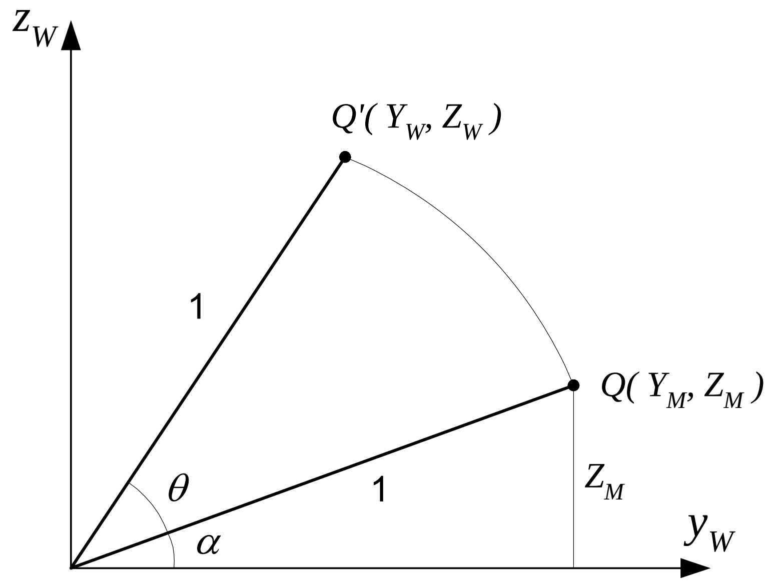
$$\begin{cases} X_w = X_M + a \\ Y_w = Y_M + b \\ Z_w = Z_M + c \end{cases}$$

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_M \\ Y_M \\ Z_M \\ 1 \end{bmatrix}$$

$$Trans(a,b,c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

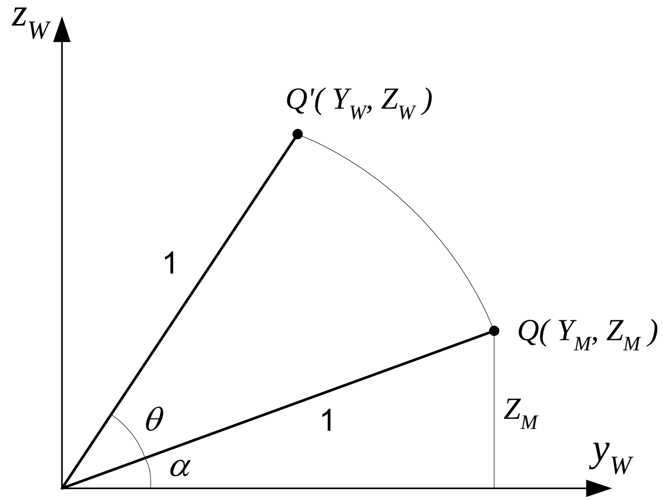
Rotation about X Axis (1)





Projection onto the *yz* plane

Rotation about X Axis (2)



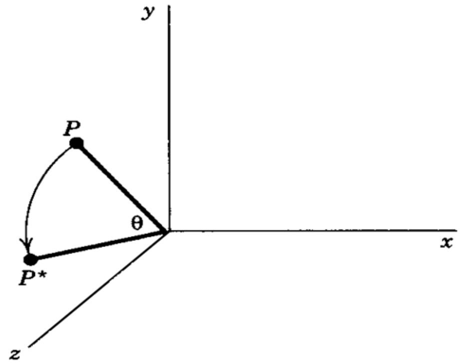
$$\begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_M \\ Y_M \\ Z_M \\ 1 \end{bmatrix}$$

$$Rot(x,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

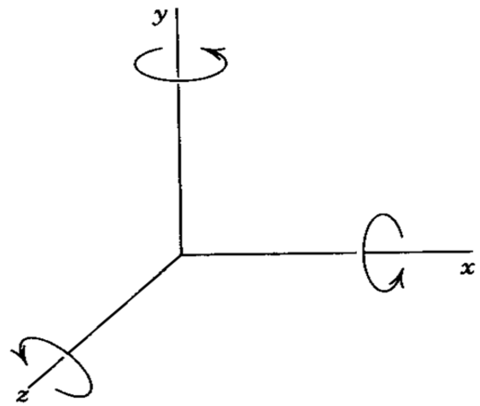
$$\begin{aligned} X_W &= X_M \\ Y_W &= \ell\cos(\theta+\alpha) \\ &= \ell(\cos\theta\cos\alpha - \sin\theta\sin\alpha) \\ &= \ell\cos\alpha\cos\theta - \ell\sin\alpha\sin\theta \\ &= Y_M\cos\theta - Z_M\sin\theta \\ Z_W &= \ell\sin(\theta+\alpha) \\ &= \ell(\sin\theta\cos\alpha + \cos\theta\sin\alpha) \\ &= \ell\cos\alpha\sin\theta + \ell\sin\alpha\cos\theta \\ &= Y_M\sin\theta + Z_M\cos\theta \end{aligned}$$

Rotation Matrix

$$Rot(x,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



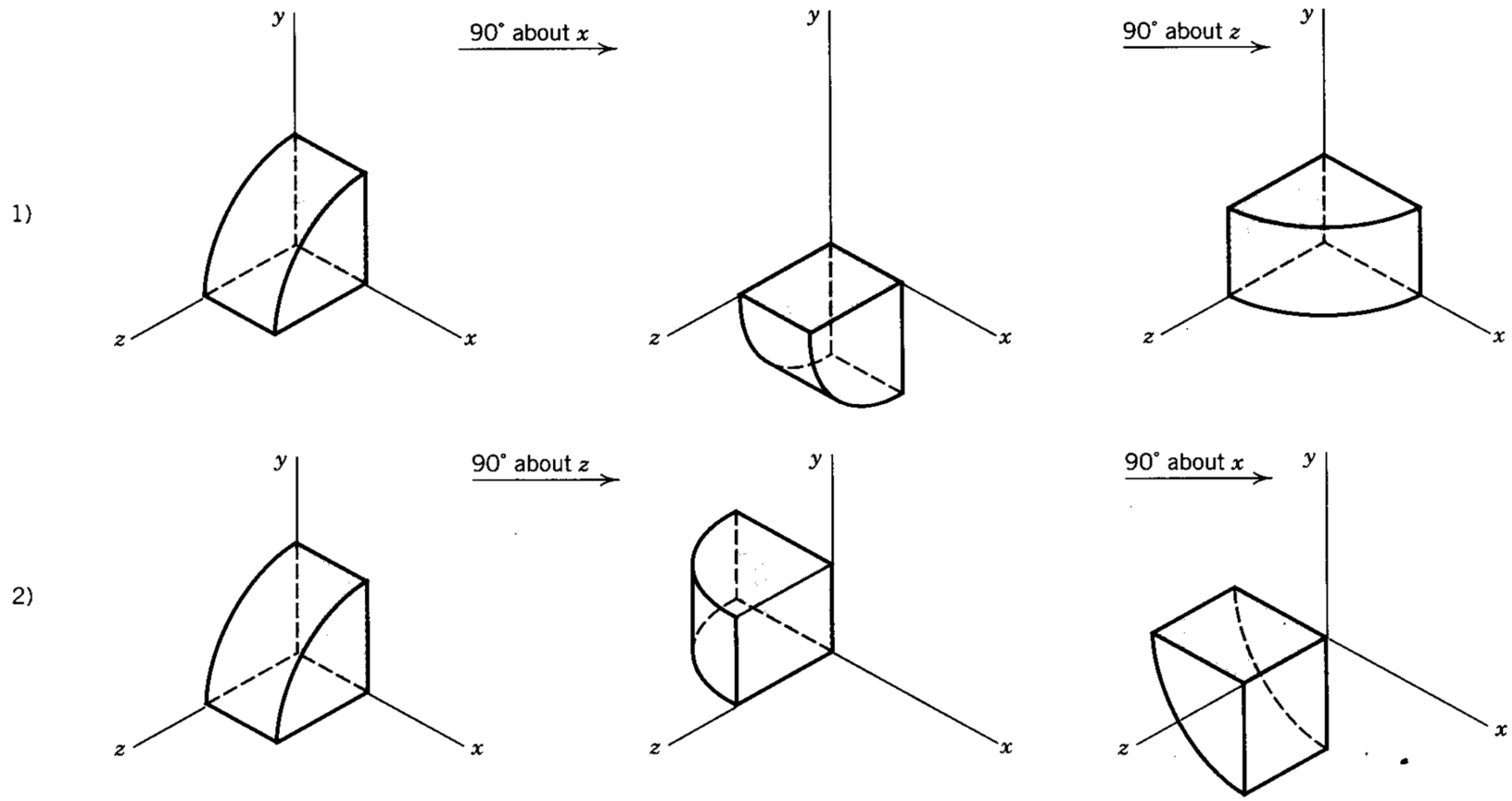
$$Rot(y,\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$Rot(z,\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Order of Rotation

- Affects the final position of an object



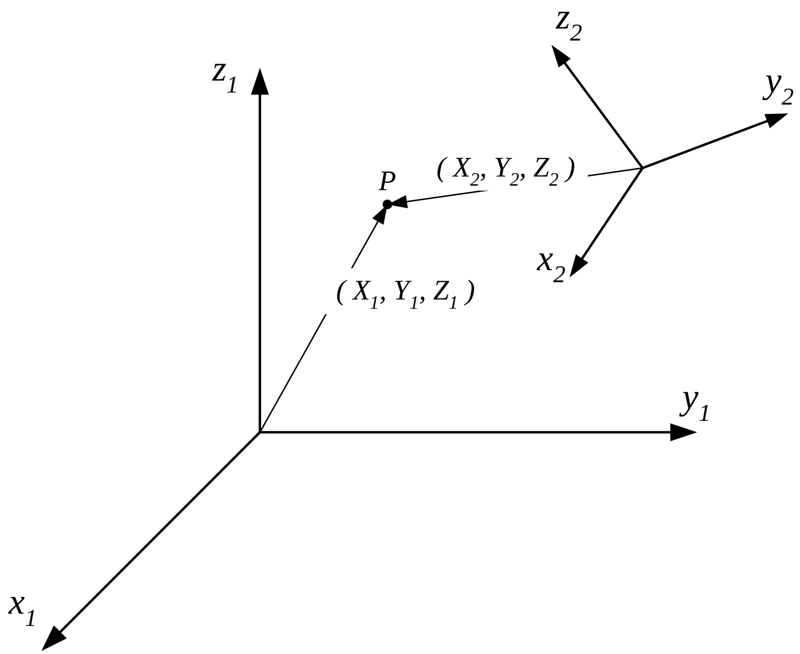
Examples

- An object in space is translated by 5 units in the y-direction of the world coordinate system and then rotated by 90 degrees about the x-axis of the world coordinate system. If a point on the object has the coordinates (0,0,1) with respect to its model coordinate system, what will be the world coordinates of the same point after the translation and rotation? <u>Ans. (0,-1,5)</u>

- An object in space is rotated by 90 degrees about an axis that is parallel to the x-axis of the world coordinate system and passes through a point having world coordinates (0,3,2). If a point on an object has model coordinates (0,0,1), what will be the world coordinates of the same point after the rotation? <u>Ans. (0,4,-1)</u>

Mapping between Two Coordinate Systems (1)

- Known: (X_1, Y_1, Z_1), $x_2y_2z_2$ coordinate system w.r.t. $x_1y_1z_1$ *system*
- Find: (X_2, Y_2, Z_2)

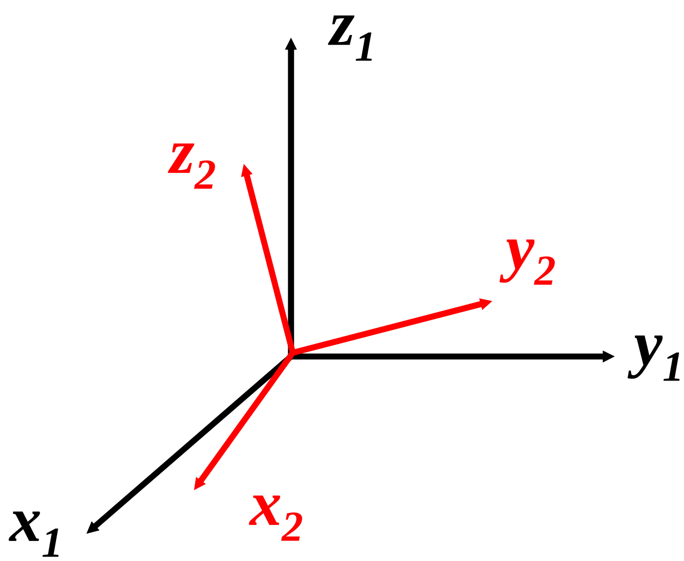


$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix} = T_{1-2} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix}$$

$$T_{1-2} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Mapping between Two Coordinate Systems (2)

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix}$$



- p_x, p_y, p_z : x_2, y_2, z_2 components of the origin of the $x_1y_1z_1$ system $(X_1 = 0,\ Y_1 = 0,\ Z_1 = 0)$
- n_x, n_y, n_z : x_2, y_2, z_2 components of the x_1 axis $(X_1 = 1,\ Y_1 = 0,\ Z_1 = 0)$
- o_x, o_y, o_z : x_2, y_2, z_2 components of the y_1 axis $(X_1 = 0,\ Y_1 = 1,\ Z_1 = 0)$
- a_x, a_y, a_z : x_2, y_2, z_2 components of the z_1 axis $(X_1 = 0,\ Y_1 = 0,\ Z_1 = 1)$

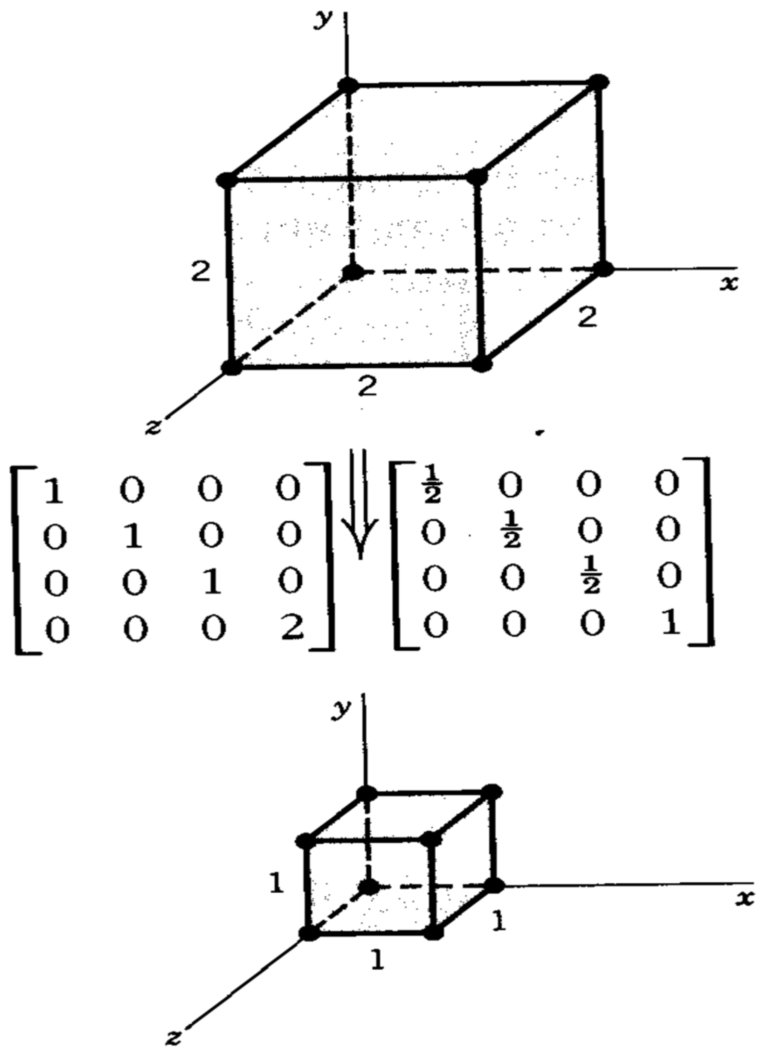
Examples

- Corresponding to the viewpoint (-10,0,1), the viewsite (0,0,1) and the up vector (0,0,1), the viewing coordinate system is drawn. Note that all the coordinate and component values are given in world coordinates. From the relative position between the viewing coordinate system and the world coordinate system, (1) calculate the mapping transformation $T_{w\text{-}v}$ and (2) calculate the coordinates of a point in viewing coordinates if it has world coordinates (5,0,1). Ans. (0,0,-5)

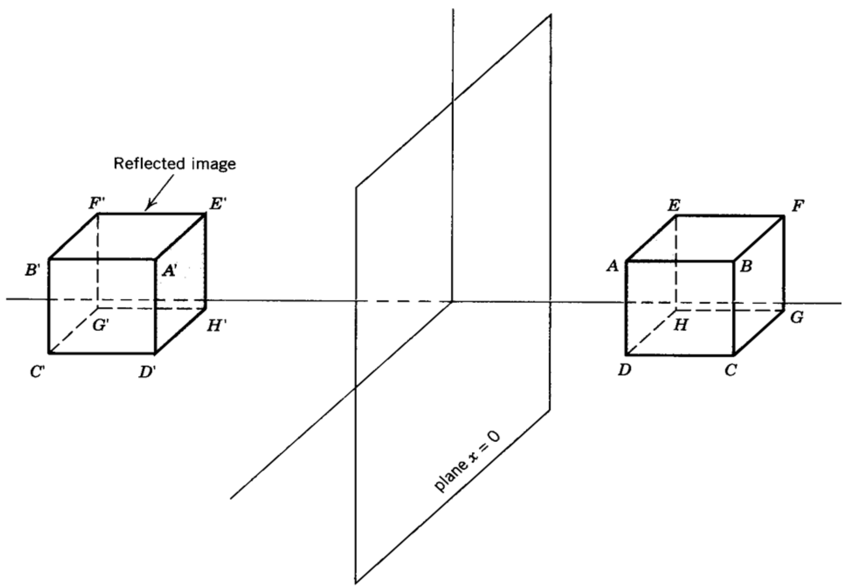
- The viewpoint and the viewsite are set at (5,5,5) and (0,0,0), respectively, to draw an isometric view, and the up vector is chosen to be (0,0,1). Derive the mapping transformation matrix $T_{w\text{-}v}$, and the viewing coordinates of a point represented by (0,0,5) in world coordinates. Ans. (0,5√6/3)

Scaling

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



Mirror Reflection



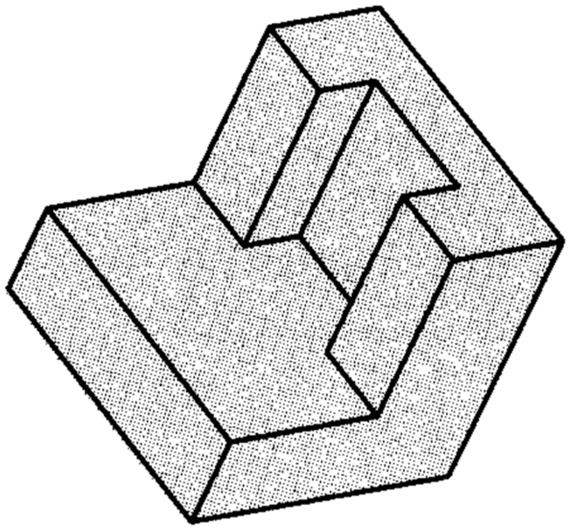
Plane	$x = 0$	$y = 0$	$z = 0$	Point (0,0,0)
$[T]_{RFL}$	$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

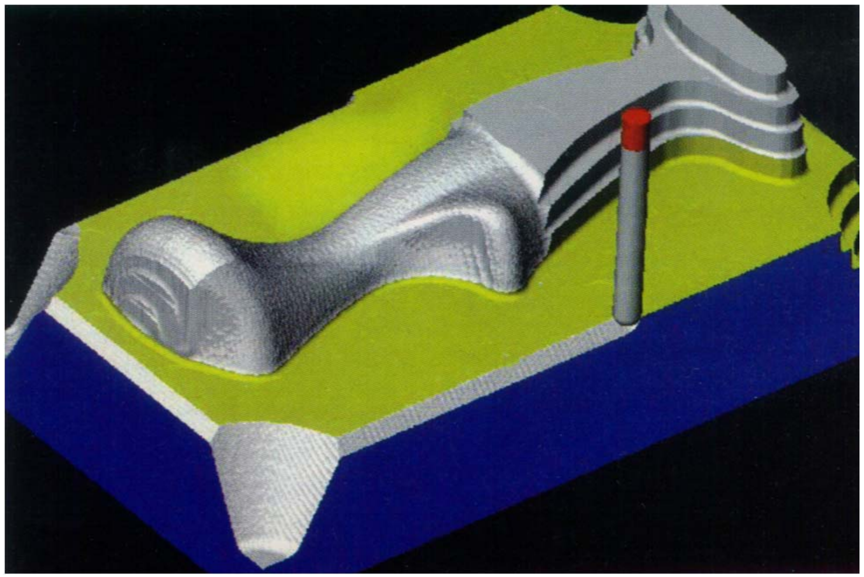
Contents

- Hidden line removal
  - Object space
  - Image space
- Rendering
  - Shading
  - Ray tracing
- Color model
  - RGB Model
  - CMY Model
  - HSV Model

Visual Realism

- Hidden Line/Surface Removal
- Rendering





Hidden Line/Surface Removal

- Hidden Line/Surface Removal
  - The task of deciding which edges or faces (or portions thereof) must be removed
  - Requires substantial computer time and memory
- Dependent on Type of Display
  - Vector display: removal of hidden lines
  - Raster display: removal of lines and faces showing shaded surfaces

Approaches for Hidden Line/Surface Removal

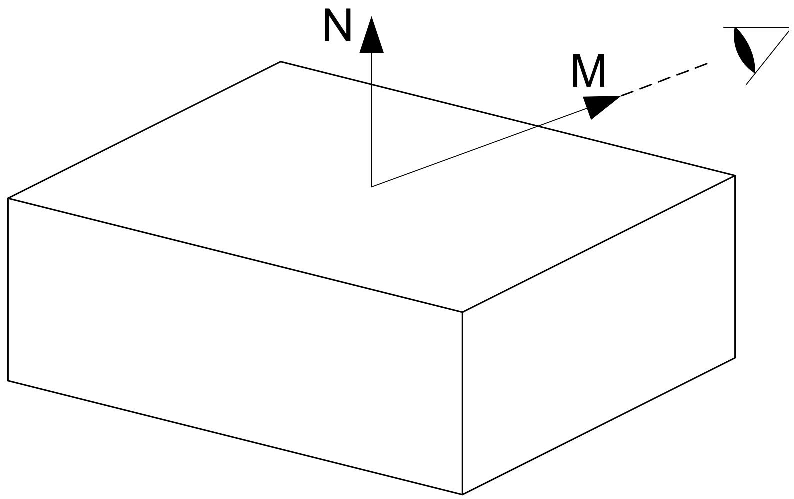
- Object space approach
  - Determines which parts of object are visible
  - Using spatial and geometrical relationships
  - Operates with object database precision
  - Software-based approach
- Image space approach
  - Determines what is visible at each image pixel
  - Concentrating on the final image
  - Operates with image resolution precision
  - Very adaptable for use in raster display
  - Hardware-based approach : z-buffer

Back-Face Removal Algorithm (1)

N : outward normal vector of a face

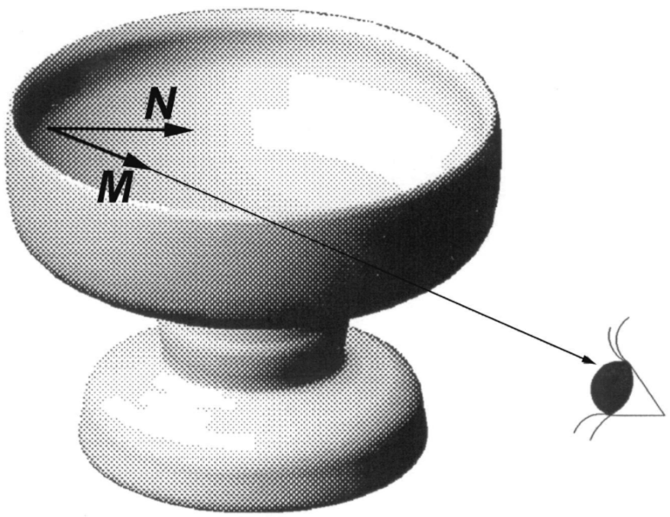
M : a view vector from a point on the face to the viewer

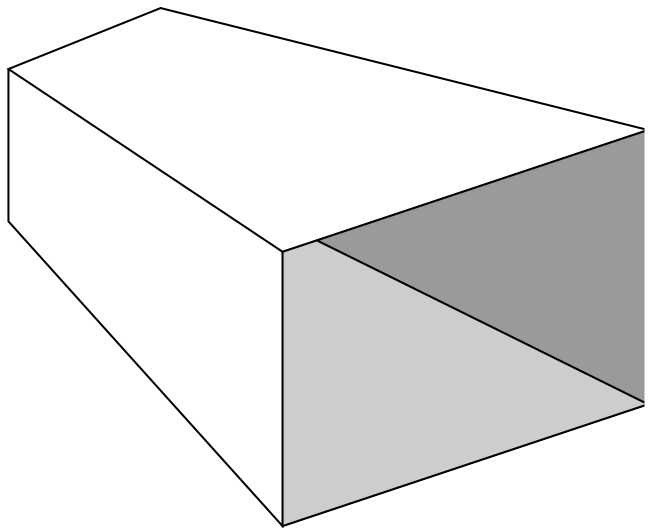
$$\text{if } \begin{cases} M \bullet N > 0, & \text{the face is visible} \\ M \bullet N = 0, & \text{the face is displayed as a line (silhouette line)} \\ M \bullet N < 0, & \text{the face is invisible} \end{cases}$$



Back-Face Removal Algorithm (2)

- Limitations
  - Can not be applied to a CONCAVE object

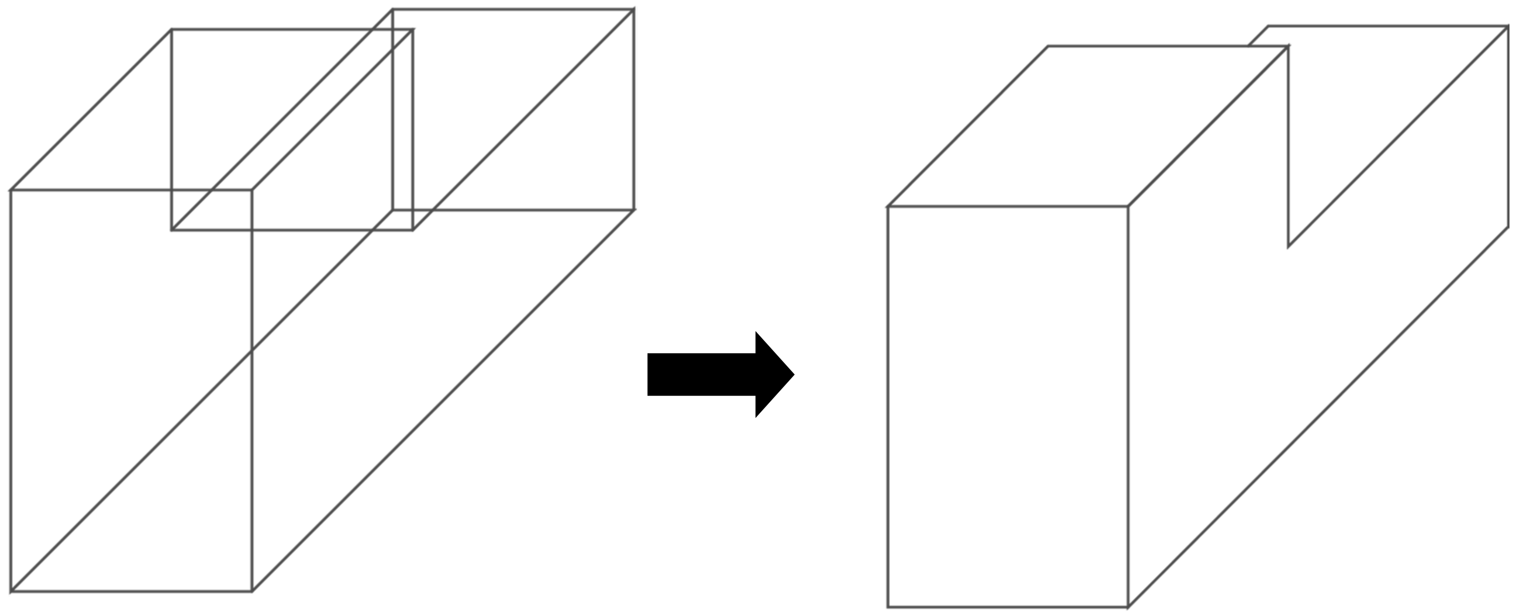




Depth-Sorting or Painter's Algorithm

- (Step 1) The faces of the objects are sorted based on their distances from the viewer
- (Step 2) Paint from the farthest face to nearer
    - from the face of minimum Z_v to the face of maximum Z_v in the Viewing Coordinate System
- NOTE:
    - If the range of Z_v values of all the points on a surface overlaps the Z_v range of the other surface, splitting each of the surfaces into two or several pieces until the Z_v ranges do not overlap

Hidden-Line Removal Algorithm (1)

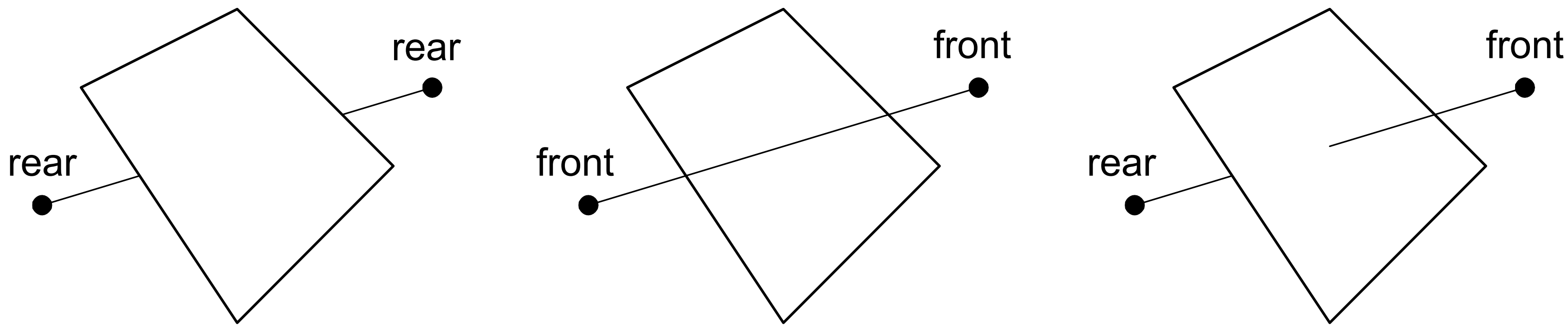


- **Step 1.** The faces pointing toward the viewer are collected by applying back-face algorithm and stored in an array FACE-TABLE with the min/max Z_v values

Hidden-Line Removal Algorithm (2)

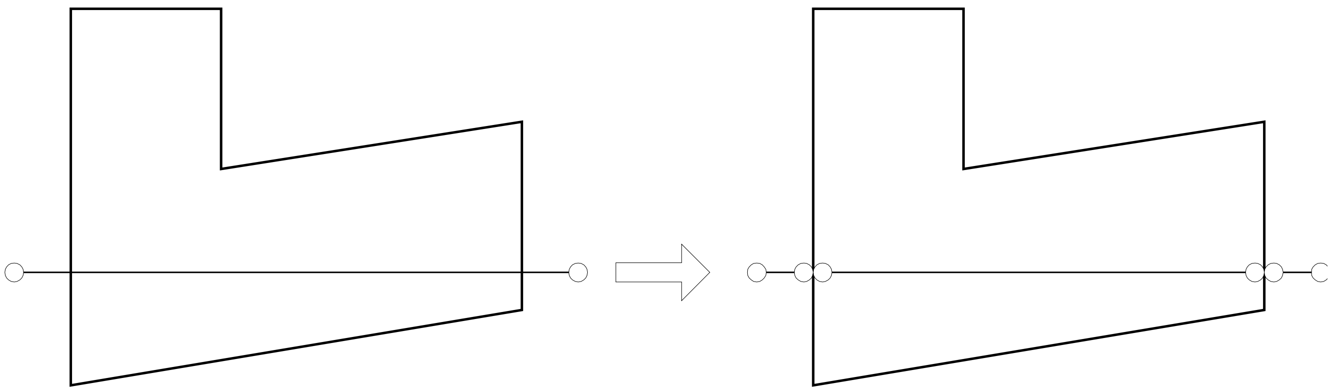
- **Step 2.** Collect the edges of the faces in FACE-TABLE and test whether they are obscured by each face in FACE-TABLE in sequence

- **Step 3.** Classify the edges as three situations



Hidden-Line Removal Algorithm (3)

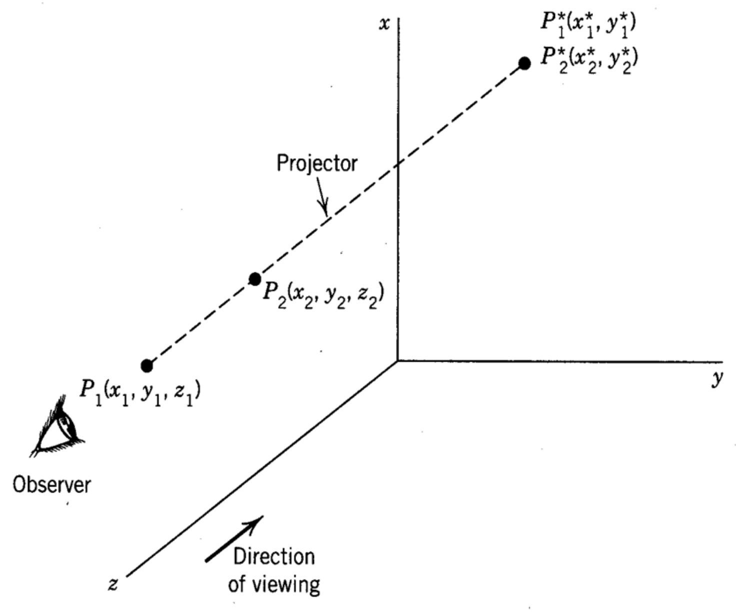
- **Step 4.** If the projected entities overlap, the edge is split at the intersection points with the face. Then repeat Step 3.

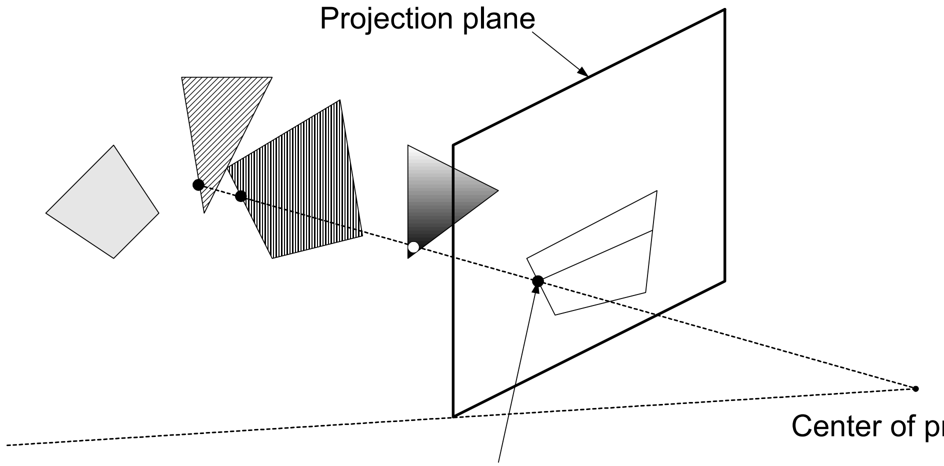


- **Step 5.** The edges which pass through the test with all the faces in the FACE-TABLE are collected to form the visible edges

Z-Buffer Method

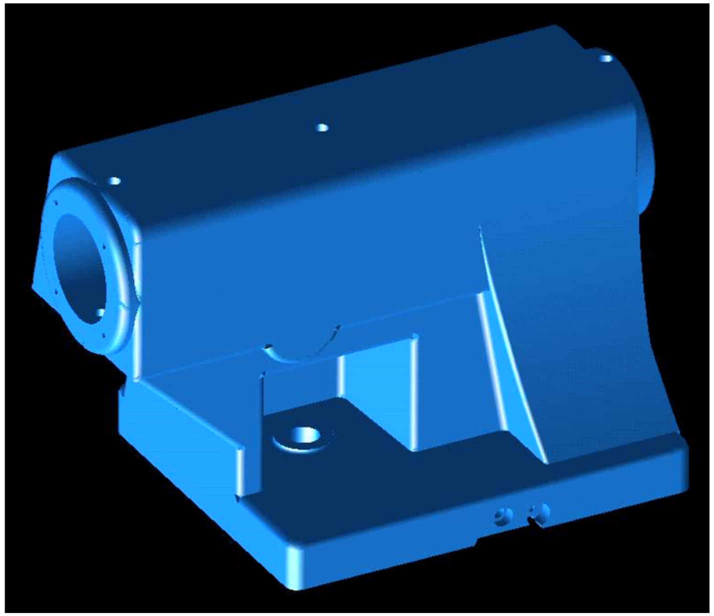
- Frame buffer stores the z coordinate or depth of every pixel in the image
- Step 1. Initialize z-buffer with the smallest z value
- Step 2. If (new_z > existing_z) then, place new_z into the z-buffer at (x,y)





Rendering

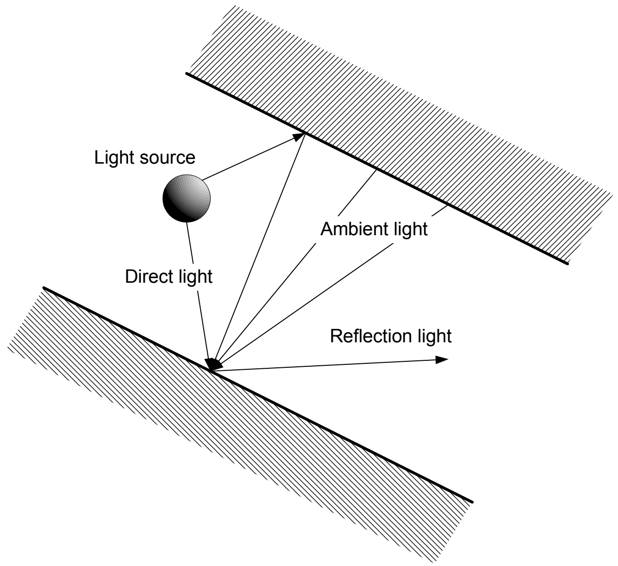
- The process to recreate the effects of light on surfaces of objects for simulation of a real scene
- Two major rendering techniques
  - Shading
  - Ray Tracing



Shading (1)

- Shading Parameters
  - Light sources
  - Characteristics of the surface
  - Relative position of the light sources w.r.t. the surface

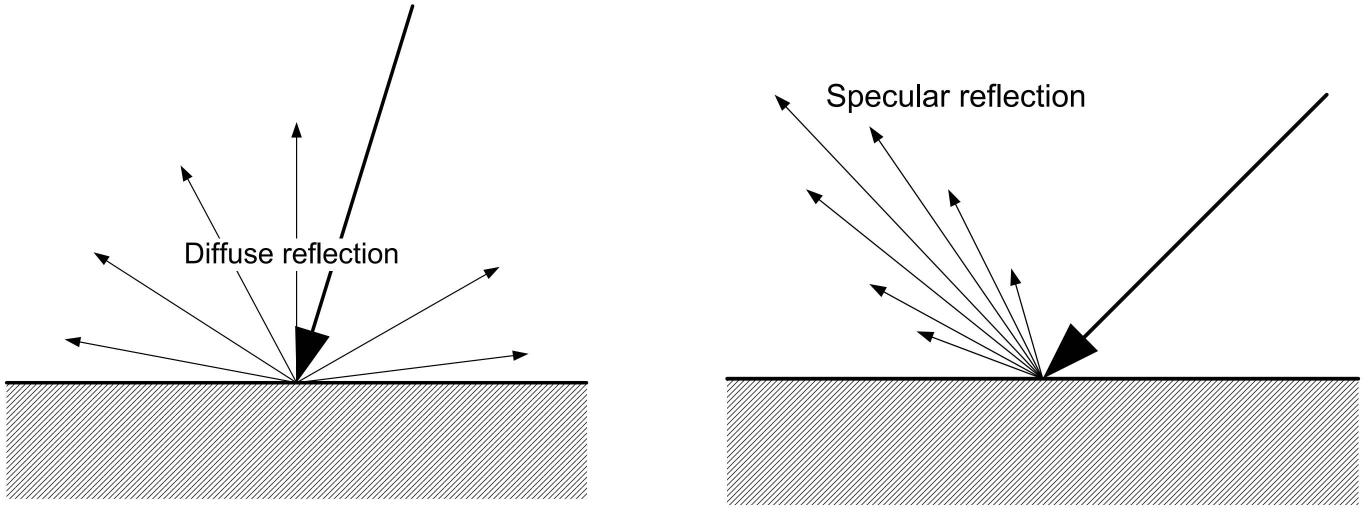
- Light Source
  - Position, intensity, direction, etc.
  - Classification of Light Source
    - Point light : distributes the light from a single point in specific directions
    - Ambient light : distributes uniformly in all directions, without regard to location



Shading (2)

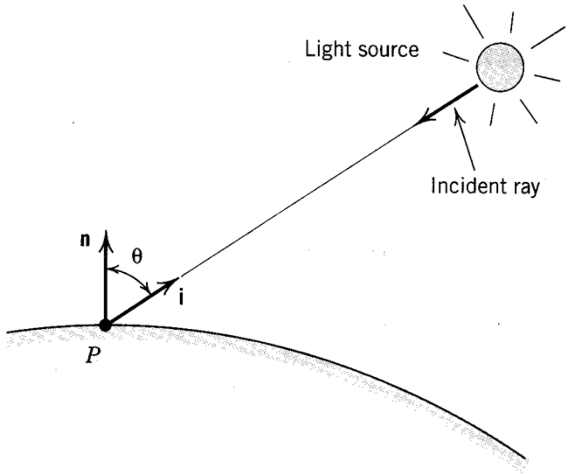
- Surface Characteristics (Ways of Reflection)
  - Diffuse reflection: light is scattered equally in all directions - commonly used
  - Specular reflection: reflects light in only one direction
  - *Real surfaces reflect light by a combination of these two effects*

$$I = k_d I_a + \frac{I_s}{D + D_0}\left[k_d\left(n \cdot i\right) + k_s(\boldsymbol{V} \cdot \boldsymbol{R})^n\right]$$



Intensity of Diffuse Reflection

- Lambert's Cosine Law
  - Intensity Value



$$I = I_s k_d \cos\theta = I_s k_d \left(n \cdot i\right) \rightarrow I = \frac{I_s k_d}{D + D_0}\left(n \cdot i\right)$$

I_s : intensity of the point source

k_d : coefficient indicating the surface reflectivity $\left(0 < k_d < 1\right)$

D : distance between the point of interest and the viewpoint

D_0 : to avoid division by zero

  - Several Light Sources and Ambient Light

$$I = k_d I_a + \sum_j I_j k_d \left(n \cdot i_j\right)$$

I_a : contribution from ambient light

Intensity of Specular Reflection

- Phong Model

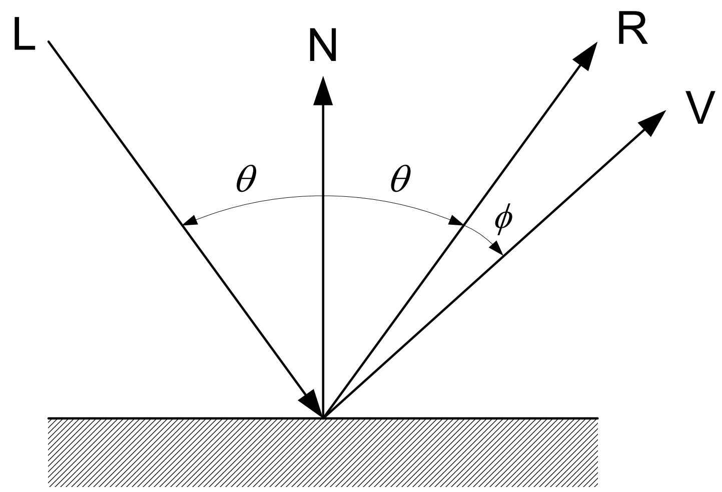
$$I = \frac{I_s}{D + D_0} k_s (\boldsymbol{V} \cdot \boldsymbol{R})^n$$

I_s : intensity of the point source

k_s : specular reflectance, coefficient indicating the surface reflectivity $(0 < k_s < 1)$

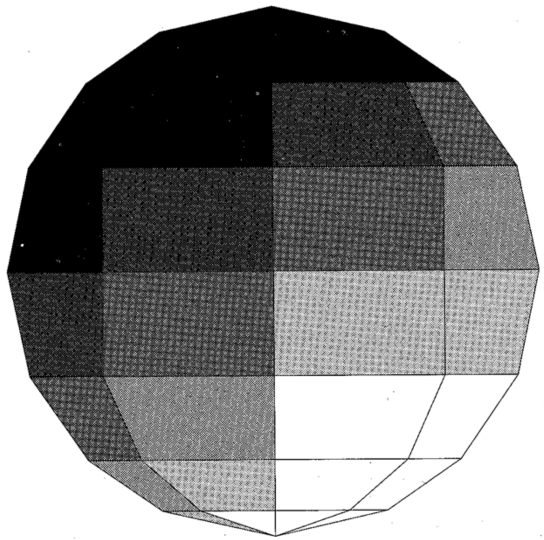
D : distance between the point of interest and the viewpoint

V : line of sight vector



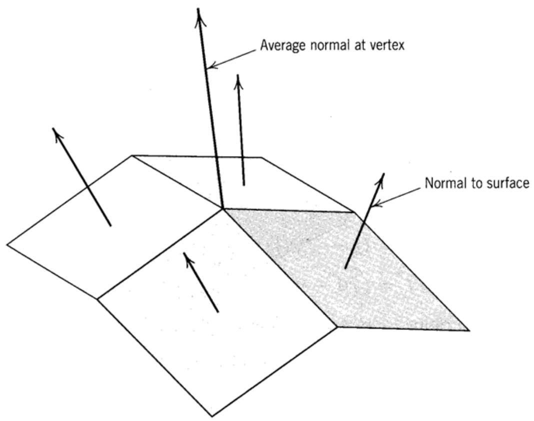
Faceted Shading

- For Polygonal Models
  - –Assign the same light intensity to all points on a facet
  - –Creating banded appearance
- Quickest shading algorithm
- Commonly used as a preliminary design check

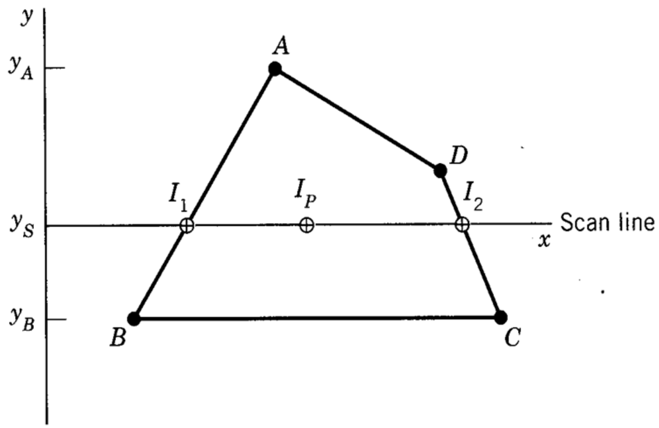


Gouraud (Smooth) Shading

- (1) Light intensities are calculated at each vertex
- (2) They are blended across the overall surface by linear interpolation



$$n_{avg} = \sum_i n_i \Big/ \left| \sum_i n_i \right|$$



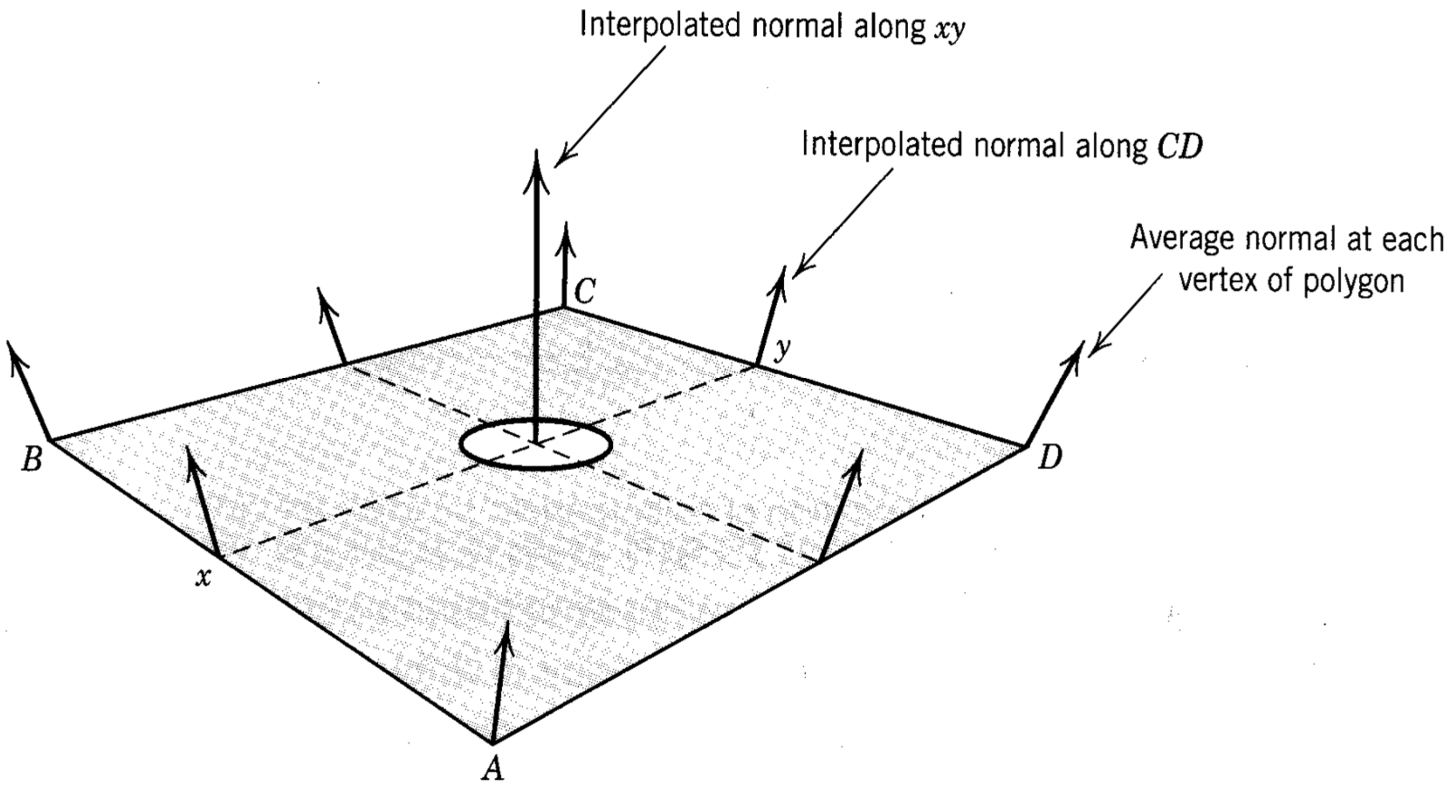
$$I_1 = \frac{(I_B - I_A)(y_A - y_S)}{(y_A - y_B)} + I_A$$

$$I_2 = \frac{(I_C - I_D)(y_D - y_S)}{(y_D - y_C)} + I_D$$

$$I_P = \frac{(I_2 - I_1)(x_P - x_1)}{(x_2 - x_1)} + I_1$$

Phong Shading

- Normal Interpolation Shading
  - –Interpolates normal vectors instead of shade intensities

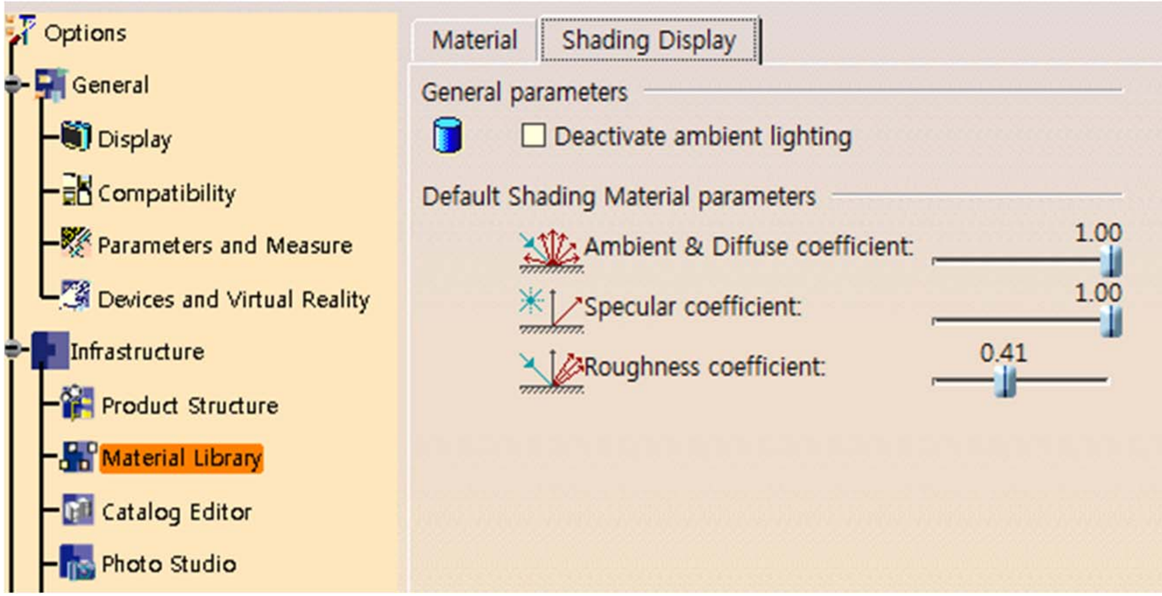


Comparison of Gouraud and Phong

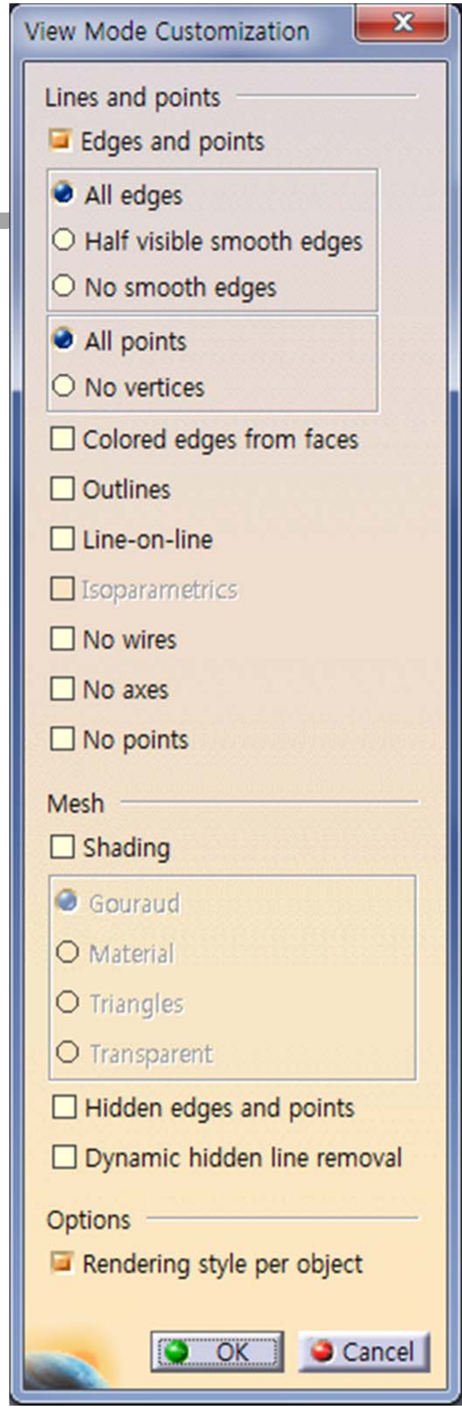
- Gouraud Shading
  - Highlights can be seen only at the vertices and can be distorted
  - Mach bands (rapid brightness change) can occasionally be produced

- Phong Shading
  - Much slower than Gouraud shading but create very realistic pictures
  - Constitute the core of most rendering systems
  - Silhouettes of curved surface are always shown in a polygonal form (unrealistic)
  - Variety of methods can add texture, shadows, transparency, and the like, to the shaded surfaces

CATIA V5

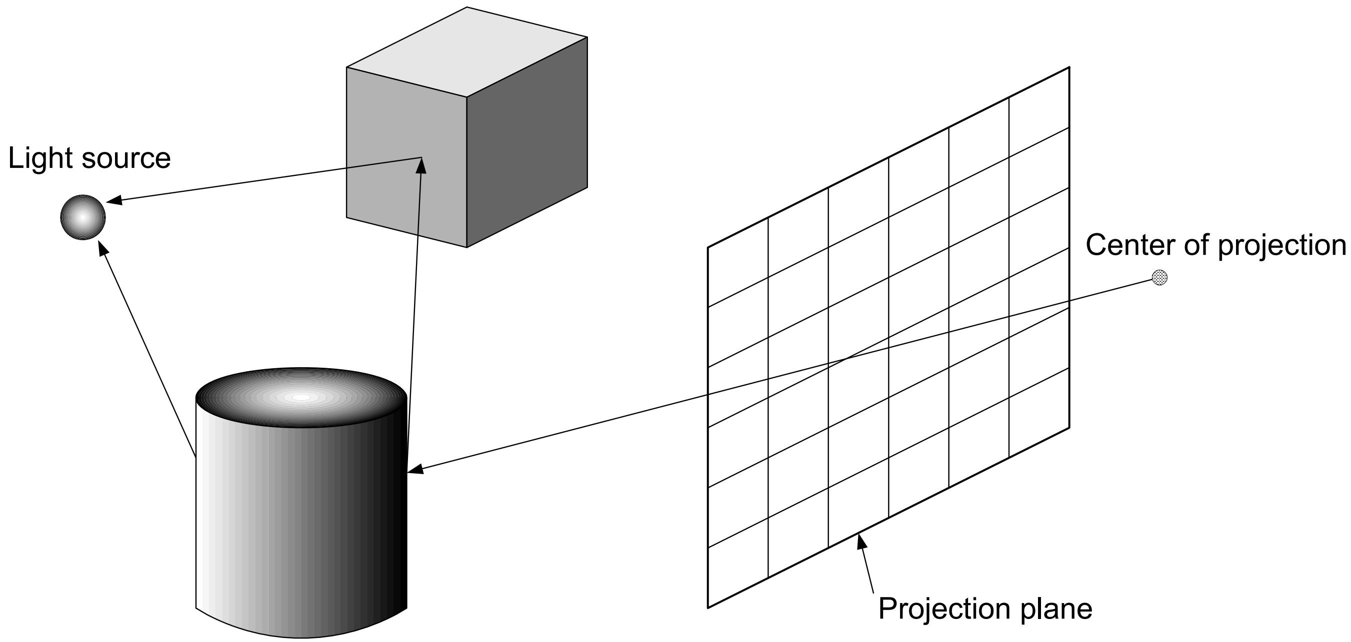

Options
General
Display
Compatibility
Parameters and Measure
Devices and Virtual Reality
Infrastructure
Product Structure
Material Library
Catalog Editor
Photo Studio
Material
Shading Display
General parameters
Deactivate ambient lighting
Default Shading Material parameters
Ambient & Diffuse coefficient:
1.00
Specular coefficient:
1.00
Roughness coefficient:
0.41


View mode


View Mode Customization
Lines and points
Edges and points
All edges
Half visible smooth edges
No smooth edges
All points
No vertices
Colored edges from faces
Outlines
Line-on-line
Isoparametrics
No wires
No axes
No points
Mesh
Shading
Gouraud
Material
Triangles
Transparent
Hidden edges and points
Dynamic hidden line removal
Options
Rendering style per object
OK
Cancel

Ray Tracing

- Support multi-object rendering unlike Shading
- Trace back the rays passing through the screen pixels
  - Pass out of the viewing volume: color of the light source
  - Strike a diffuse surface: background color
  - Hit the light source: color of the reflection on the surface

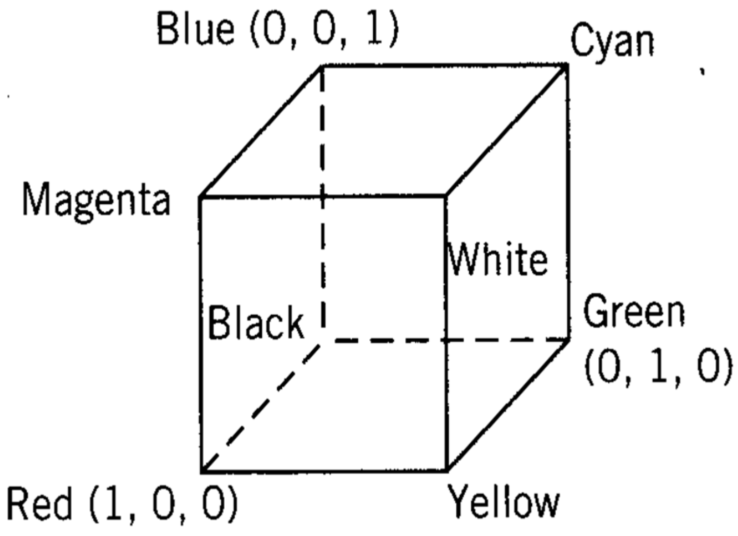


Color Models

- RGB Model
- CMY Model
- HSV Model

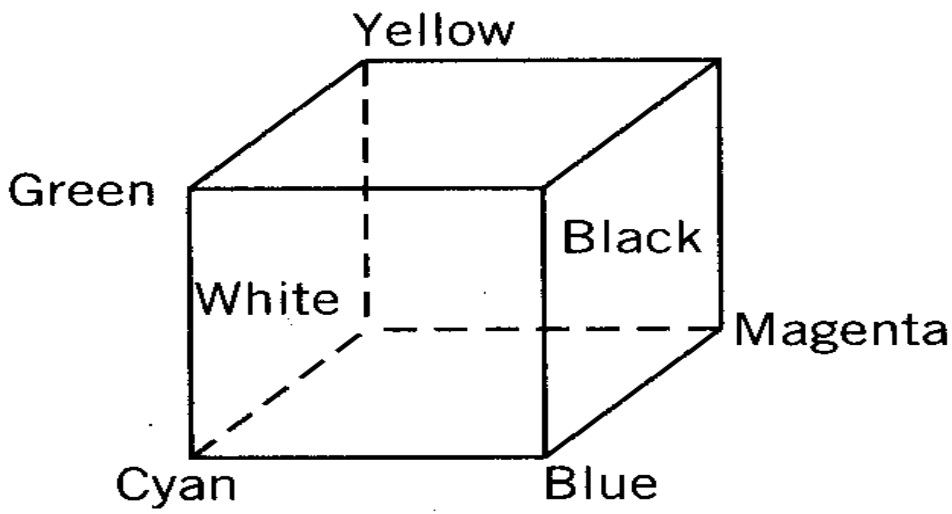
RGB Model

- RGB primaries - red, green, and blue
- Additive model
- Black - origin (0,0,0) : White - at the point (1, 1, 1)
- All other colors are represented by points inside the cube



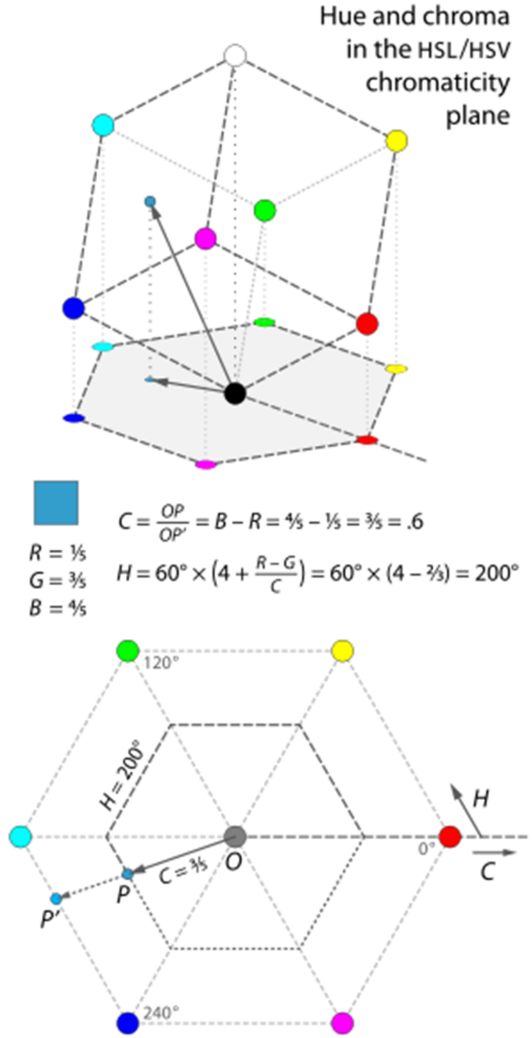
CMY(K) Model

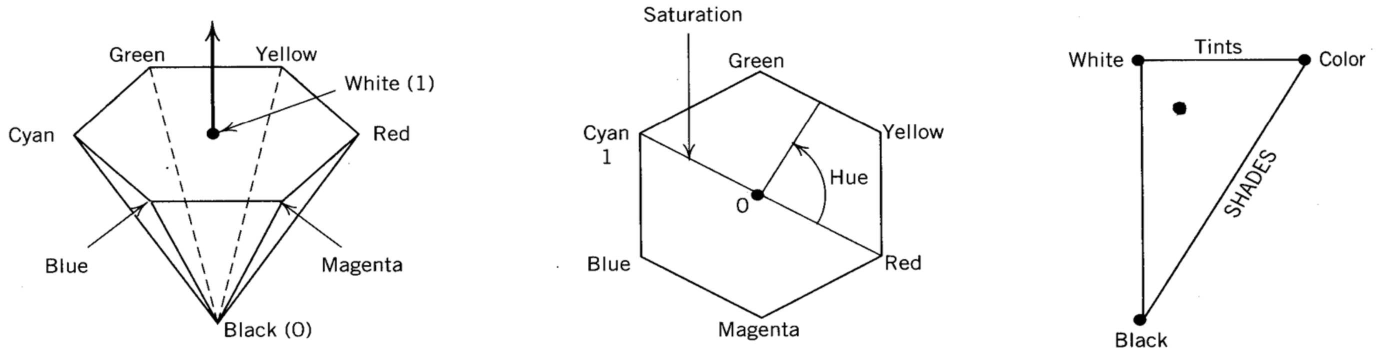
- Cyan, Magenta, and Yellow, (Key: Black)
- Subtractive model
- White - Origin (0,0,0) : Black - at (1,1,1)



HSV Model

- cylindrical-coordinate representations of points in an RGB color model
  - Hue - color spectrum : angle (0^o to 360^o)
  - Saturation - purity : (0 to 1)
  - Value of a color – lightness, brightness (0 to 1)

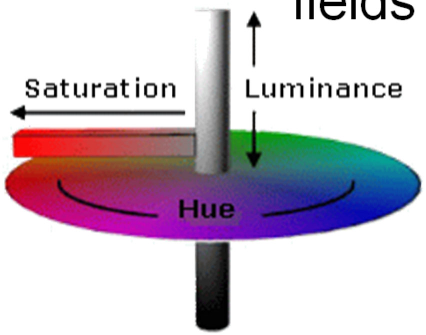




(a) (b) (c)

CATIA V5

– The HSL (Hue, Saturation and Luminance) and RGB (Red, Green and Blue) values vary according to where the cross is located. You can also enter HSL and RGB values in the fields provided to suit your exact color specifications.



View →Toolbars →Graphic properties