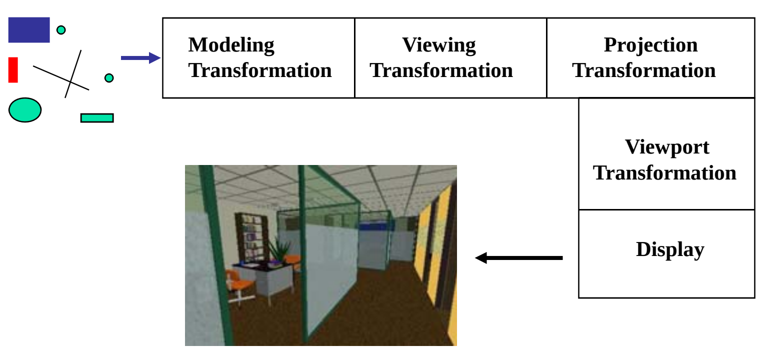
CS 362: Computer Graphics

3D Viewing

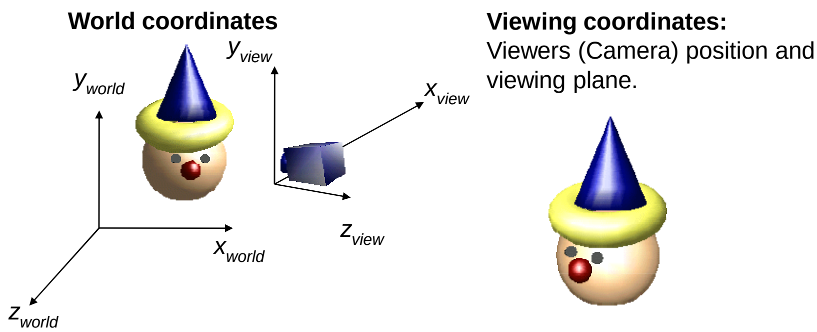
Dr. Samit Bhattacharya
Dept. of Comp. Sc. & Engg.
IIT Guwahati, Assam, India

3D Viewing Pipeline



3D Viewing

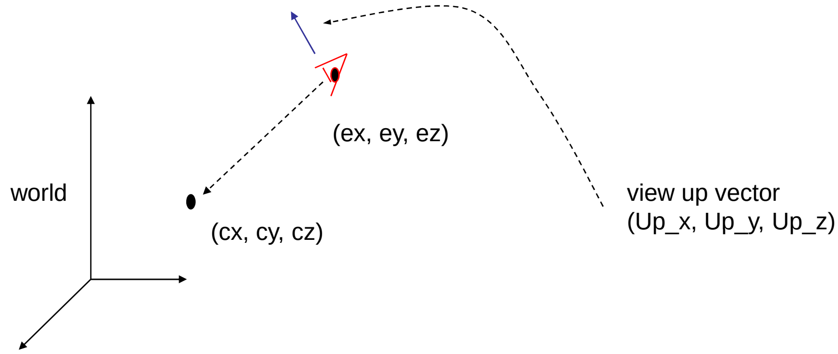
- Just like taking a photograph!
- World coordinates to viewing coordinates: viewing transformations



Camera Parameters

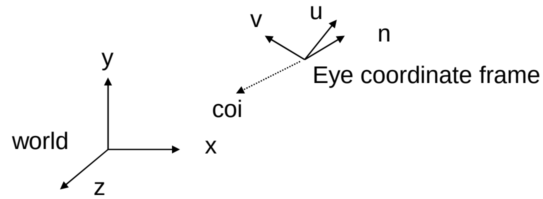
- Important camera parameters to specify
  - Camera (eye) position (ex,ey,ez) in world coordinate system
    - Also called *viewpoint/viewing position*
  - Center of interest (coi) (cx, cy, cz)
    - Also called *look-at point*
  - Orientation (which way is up?) View-up vector (Up_x, Up_y, Up_z)

Camera Parameters



Viewing Transformation

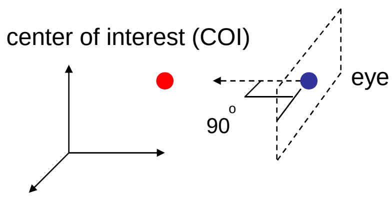
- Form a camera (eye/view) coordinate frame



- Transform objects from world to eye space
  - Construct a transformation matrix

Eye Coordinate Frame

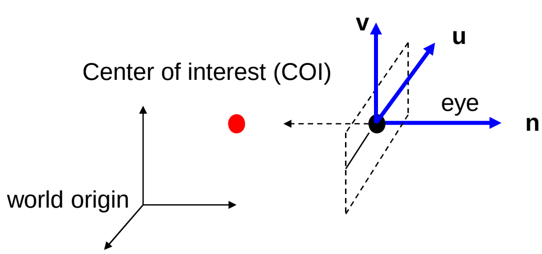
- Known: eye position, center of interest, view-up vector
- To find out: new origin and three basis vectors



Assumption: the direction of view is orthogonal to the view plane (the plane that objects will be projected onto)

Eye Coordinate Frame

- Origin: eye position (that was easy)
- Three basis vectors: one is the normal vector (**n**) of the viewing plane, the other two are the ones (**u** and **v**) that span the viewing plane



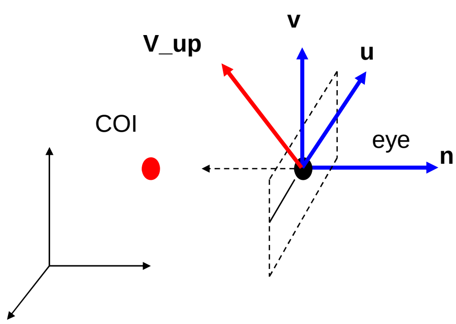
(u,v,n should be orthogonal to each other)

n is pointing away from the world because we use right hand coordinate system
N = eye – COI
n = N / | N |

Remember **u,v,n** should be all unit vectors

Eye Coordinate Frame

- What about u and v?



We can get u first -

u is a vector that is perpendicular to the plane spanned by n and view up vector (V_up)

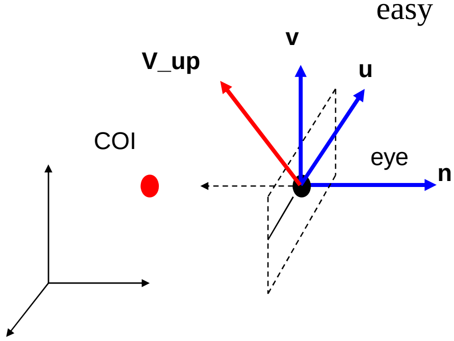
U = V_up x **n**

u = U / | U |

Eye Coordinate Frame

- How about v?

Knowing n and u, getting v is easy



v = n x u

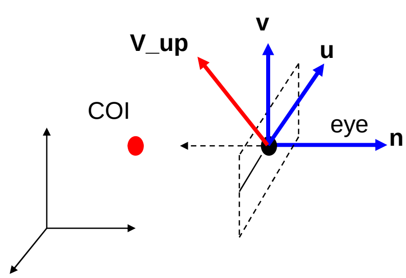
v is already normalized

Note

- How to set View Up vector
  - Choose a point P in the world co-ordinate
  - Form a vector from the eye-position to P
  - This is the view-up vector
    - Provided it is not parallel to **n**
- Viewing plane – parallel to **uv** plane
- Viewing direction – opposite **n**

Eye Coordinate Frame

- Put it all together



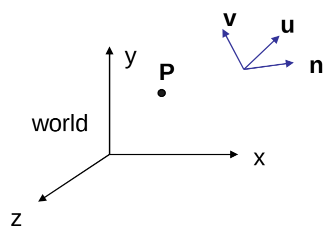
Eye space **origin: (ex , ey, ez)**

Basis vectors:

n = (eye – COI) / | eye – COI|
u = (V_up x **n**) / | V_up x **n** |
v = **n** x **u**

World to Eye Transformation

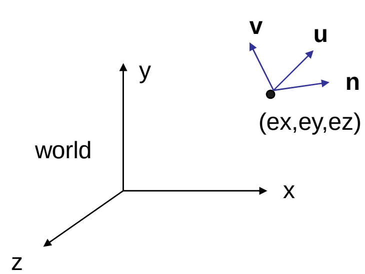
- Transform object description from WC to EC
  - Equivalent of transformation between co-ordinate systems
- Transformation matrix (M_{w2e}); $P' = M_{w2e} \times P$



- Come up with the transformation sequence to move eye coordinate frame to the world
- Apply this sequence to the point P in a reverse order

World to Eye Transformation

- Translate (-ex, -ey, -ez)
- Rotate the eye frame so that it is“aligned” with the world frame

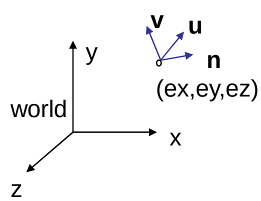


Translation: $\begin{vmatrix} 1 & 0 & 0 & -ex \\ 1 & 0 & 0 & -ey \\ 1 & 0 & 0 & -ez \\ 0 & 0 & 0 & 1 \end{vmatrix}$

Rotation: $\begin{vmatrix} ux & uy & uz & 0 \\ vx & vy & vz & 0 \\ nx & ny & nz & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$

World to Eye Transformation

$$M_{w2e} = \begin{vmatrix} ux & uy & uz & 0 \\ vx & vy & vz & 0 \\ nx & ny & nz & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & -ex \\ 1 & 0 & 0 & -ey \\ 1 & 0 & 0 & -ez \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

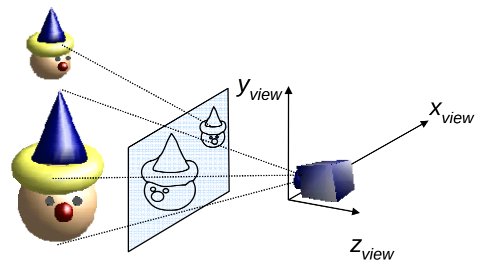


Generating 3D Viewing Effects

- By varying the viewing parameters
  - Composite display consisting of multiple views from a fixed camera position
    - Fixed eye position, change **n**
  - Simulate animation panning effect
    - **n** fixed, change eye position
  - To view an object from different positions
    - Move the eye position around the object

Projection Transformations

- Projection: 3D to 2D



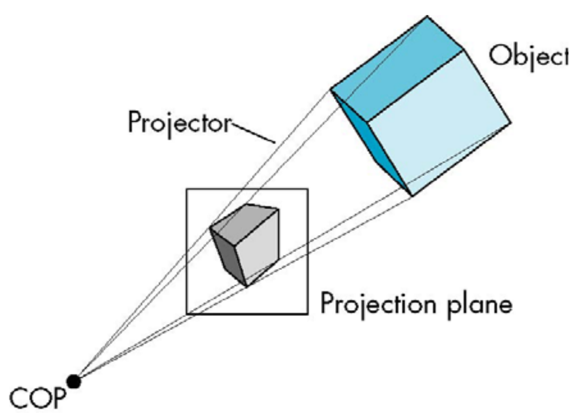
Projection

- Map objects from 3D space to 2D screen
  - Defined by straight lines called *projectors*
- Graphics deals with
  - Planar projections (projection surface is a plane)
  - Geometric projections – the projectors are straight lines

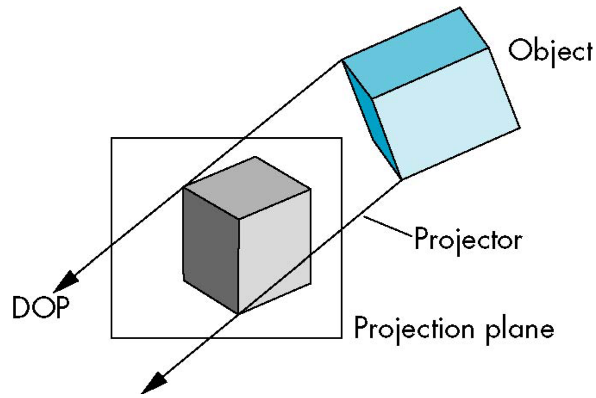
Planar Geometric Projections

- Projectors are lines that either
  - Converge at a center of projection (perspective projection)
  - Are parallel (parallel projection) – center of projection at infinity

Perspective Projection



Parallel Projection



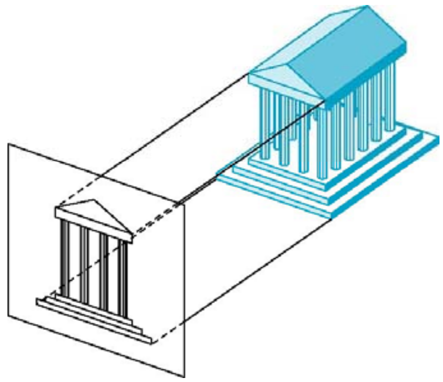
DOP = Direction Of Projection

Projection Taxonomy

- planar geometric projections
  - parallel
    - orthographic
      - multiview
      - axonometric
        - isometric
        - dimetric
        - trimetric
    - oblique
      - cavalier
      - cabinet
  - perspective
    - 1 point
    - 2 point
    - 3 point

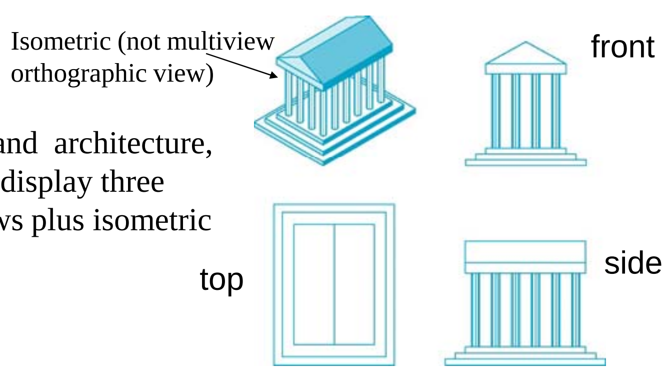
Orthographic Projection

- Projectors are orthogonal to projection surface (DOP perpendicular to projection plane)



Multiview Orthographic Projection

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



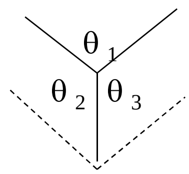
In CAD and architecture, we often display three multiviews plus isometric

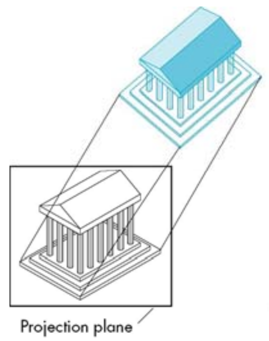
Axonometric Projections

- View plane not parallel to principle faces, i.e. allow projection plane to move relative to object
  - Used in CAD applications

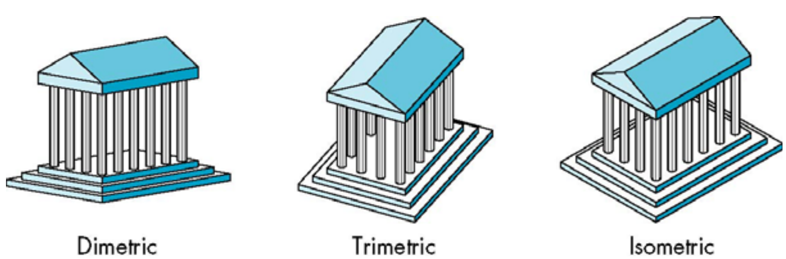
Classify by how many angles with the three *principle* axes are equal

None: trimetric
Two: dimetric
Three: isometric



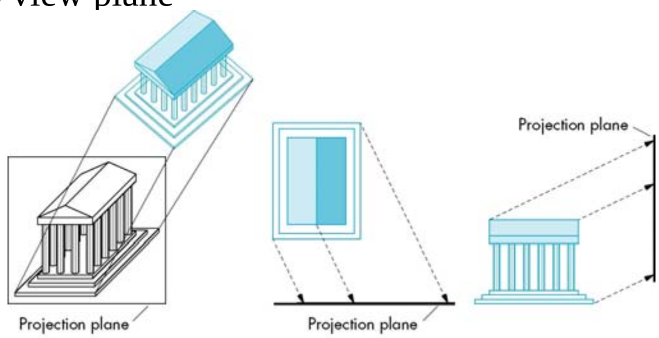


Types of Axonometric Projections



Oblique Projection

- Arbitrary relationship between projectors and projection plane, i.e. projectors not perpendicular to view plane

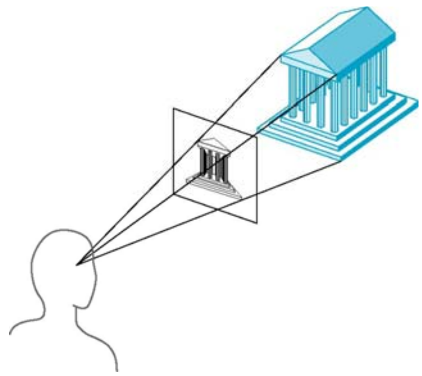


Oblique Projection

- Cavalier
  - DOP is such that no foreshortening of lines perpendicular to the projection plane
- Cabinet
  - DOP is such that the lines perpendicular to the projection plane are foreshortened by half their lengths

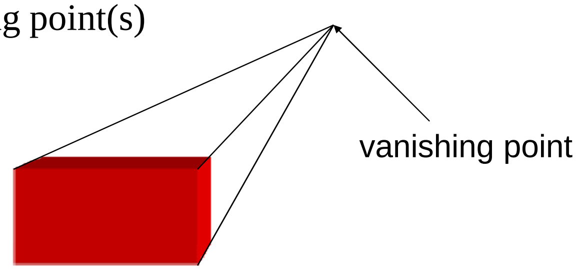
Perspective Projection

- Projectors converge at center of projection



Vanishing Points

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



One-Point Perspective

- One principal face parallel to projection plane

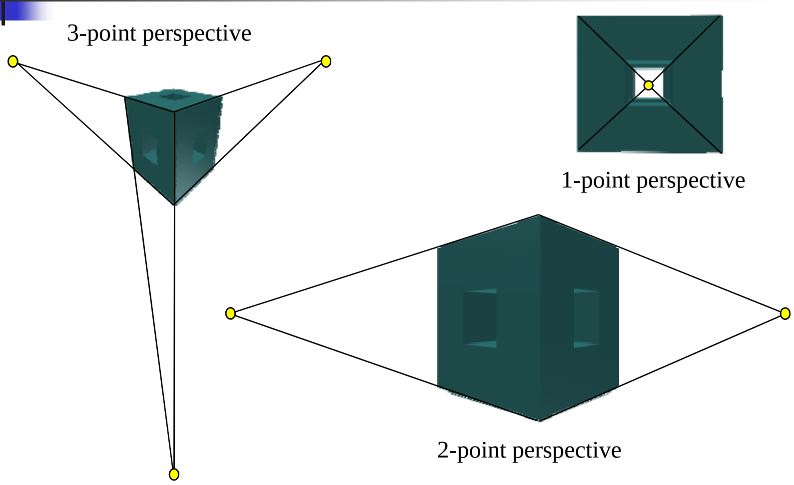
Two-Point Perspective

- One principal direction parallel to projection plane (projection plane intersects two of the *principle* axes)

Three-Point Perspective

- No principal face parallel to projection plane (projection plane intersects all the three *principle* axes)

Perspective Projections

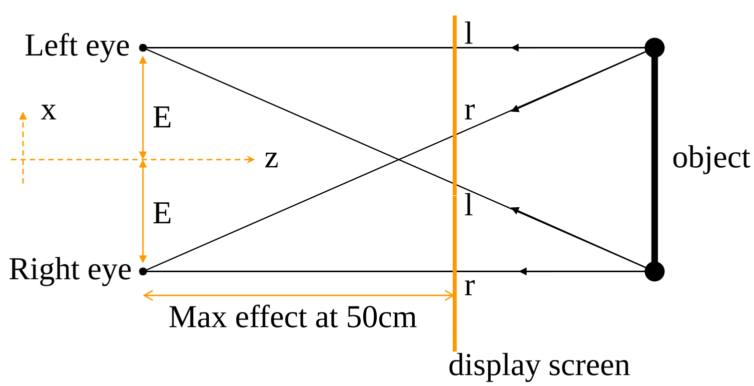


Characteristics

- 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 (*non-uniform 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)

Stereo Projection

- Combination of two perspective projections



Projection Transformation

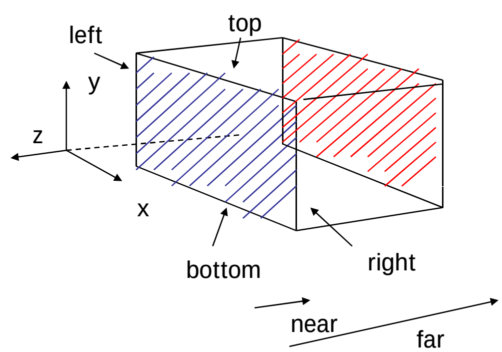
- Once WC→EC transformation is done, the 3D objects are projected on the 2D view plane
- Important Considerations
  - Clipping window – an area on the view/projection plane, which will contain the projected points
  - View volume – a region in the 3D space that contains the objects to be projected
  - Note that we don't project the entire 3D scene

Projection Transformation

- Important things to control (to define view volume)
  - Perspective or Orthographic
  - For perspective
    - Field of view and aspect ratio (width/height of the clipping window) OR
    - X (X_max, X_min) and Y (Y_max, Y_min) extent of the clipping window
  - Near and far clipping planes

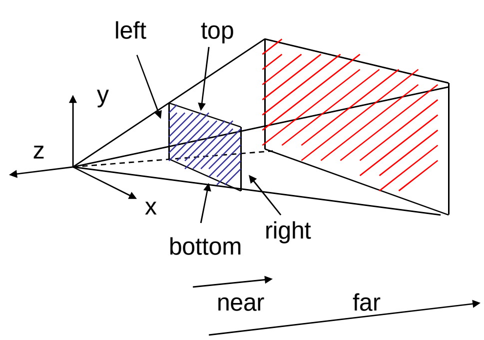
View Volume

- Determines how much of the 3D scene is projected
  - Objects outside are clipped
- Shape depends on the type of projection
  - A box/rectangular parallelepiped for parallel projection

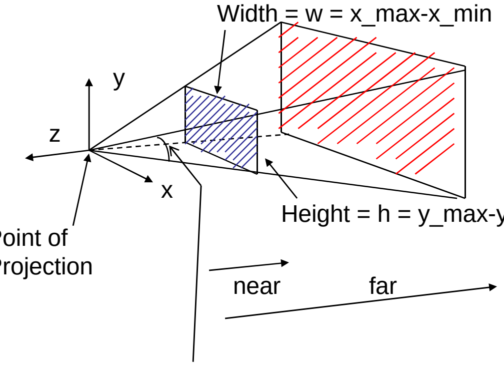


View Volume

- Determines how much of the 3D scene is projected
  - Objects outside are clipped
- Shape depends on the type of projection
  - A view frustrum for perspective projection



View Frustrum Specification

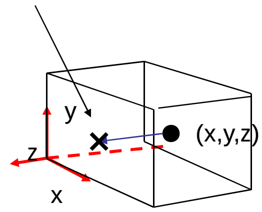


Parallel Projection

- After transforming the object to the eye space, parallel projection is relatively easy – just drop the Z

  Xp = x
  Yp = y
  Zp = -d

- We actually want to keep Z – why?



Parallel Projection

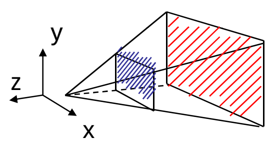
- Put in a matrix form (assuming the view plane at a distance d from origin, along the –z direction)

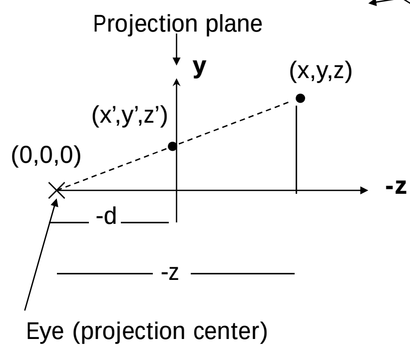
$$\begin{bmatrix} x'' \\ y'' \\ z'' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -d \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Note that this is in homogeneous coordinate
  - x' = the actual projected point = x''/w etc…

Perspective Projection

- Side view





Based on similar triangle:

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

$$\Rightarrow \quad y' = y \times \frac{d}{z}$$

Perspective Projection

- Same for x, so we have

  $x' = x \times d/z$

  $y' = y \times d/z$

  $z' = -d$

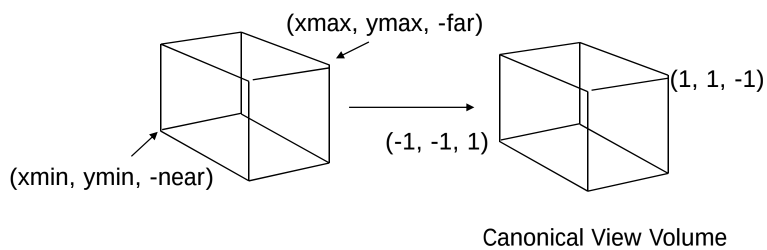
- Put in a matrix form (in homogeneous coordinate system)
  - Actual points are: x'=x''/w etc…

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

Canonical View Volume

- Objects outside view volume are clipped
- Clipping can be done in two ways
  - Direct clipping: clip against whatever view volume is given
    - May involve calculation of intersection points of view volume boundary planes and lines. For arbitrary bounding planes, such computations may take significant time
  - Much easier (and sometimes faster) to clip against *canonical* view volumes

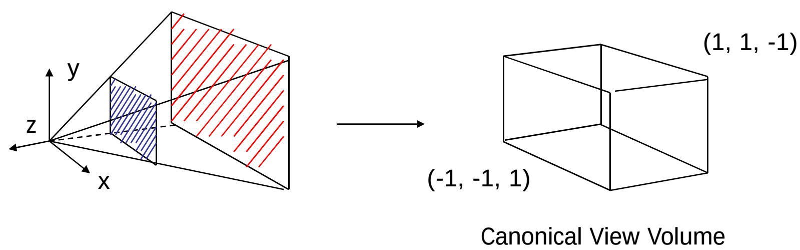
Parallel CVV



CVV Contd...

- For perspective projection, the canonical view is still a frustrum - perspective canonical view volume (PeC)
  - With unit slope planes instead of arbitrary slopes
- Usually PeC is transformed to parallel canonical view volume (PrC)
- PeC → PrC can be achieved using a combination of transformations on PeC
  - Shear
  - Followed by scaling (scaling factor a function of distance)

Perspective CVV

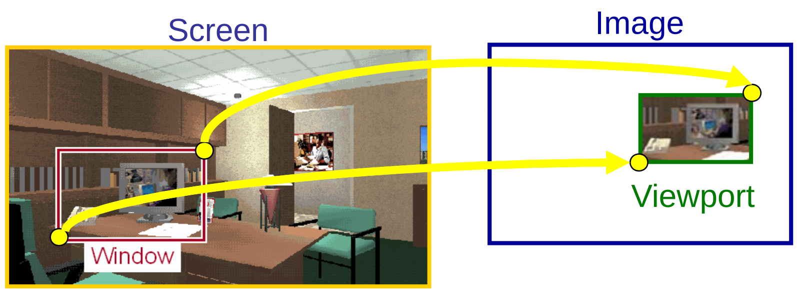


Note

- The projection matrices shown before are not final
- They undergo some changes
  - In order to preserve depth (z) information – required for hidden surface removal
  - Due to normalization (CVV) transformations
- The near plane of the CVV (symmetric cube) considered as the view/projection plane
  - Normalized clipping window

Viewport Transformation

- Transform from projection coordinates (normalized clipping window) to device coordinates

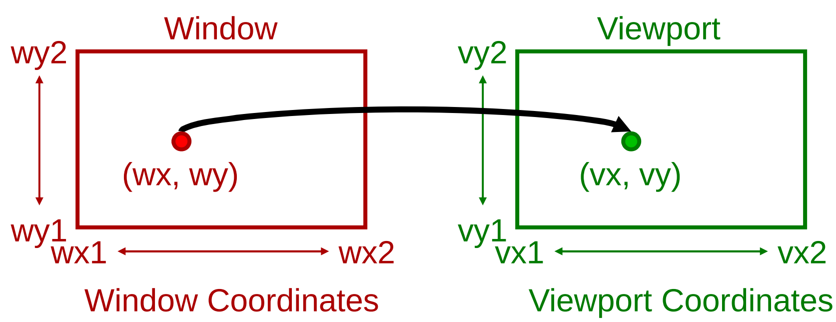


Window vs Viewport

- Window
  - World-coordinate area selected for display
  - What is to be viewed
- Viewport
  - Area on the display device to which a window is mapped
  - Where it is to be displayed

Viewport Transformation

- Window-to-Viewport Mapping



Viewport Transformations

- To maintain relative position, we must have,

$$\frac{wx - wx1}{wx2 - wx1} = \frac{vx - vx1}{vx2 - vx1}; \qquad \frac{wy - wy1}{wy2 - wy1} = \frac{vy - vy1}{vy2 - vy1}$$

Which, after simplification can be written as,

vx = sx.wx + tx

where

sx = (vx2-vx1)/(wx2-wx1); tx = sx.(-wx1) + vx1

Similarly for vy

Viewport Transformations

- The transformation can be represented in a matrix form as,

$$M_{WV} = \begin{bmatrix} sx & o & tx \\ 0 & sy & ty \\ 0 & 0 & 1 \end{bmatrix}$$

- Note that if sx ≠ sy, the transformed object will be scaled (up/down)

Note

- The normalized clipping window is *not* really 2D
  - It preserves depth information
- Hence viewport transformation is *not* between 2D window to 2D viewport
  - Actually, between window to viewport in 3D (since we have depth info)
  - Viewport is the mapping surface of 3D device space