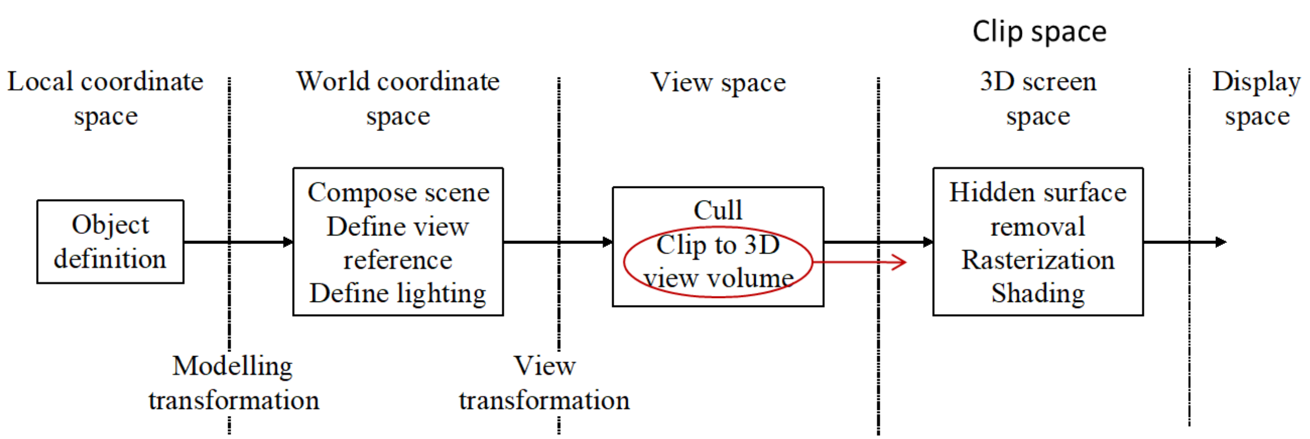
COM3503/4503/6503: 3D Computer Graphics

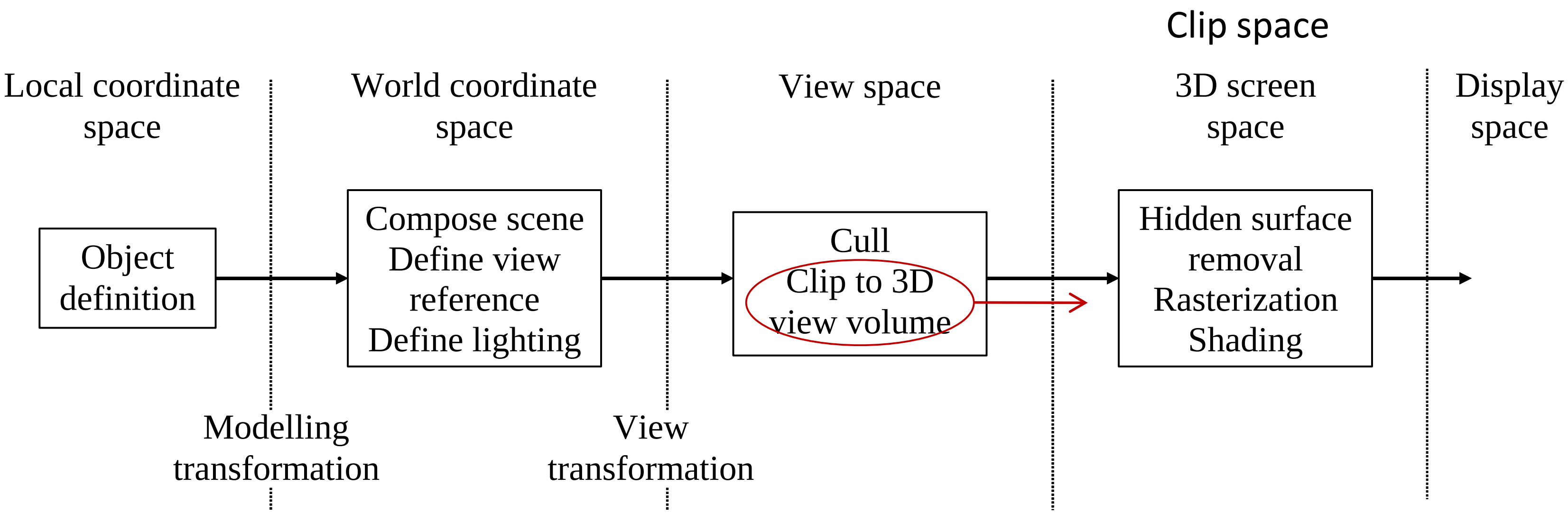
Lecture 6: The graphics pipeline

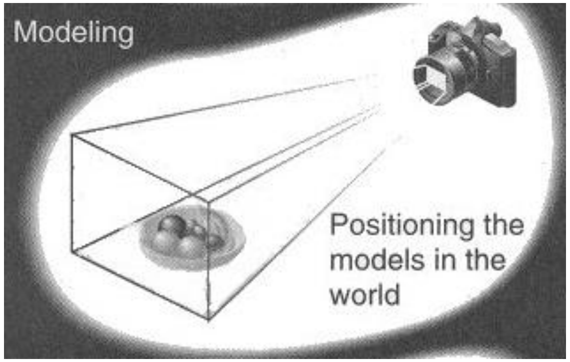
Dr. Steve Maddock
s.maddock@sheffield.ac.uk

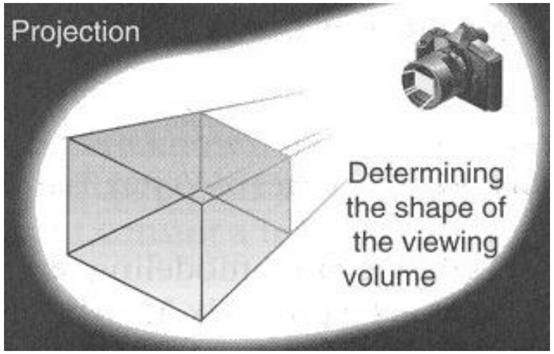


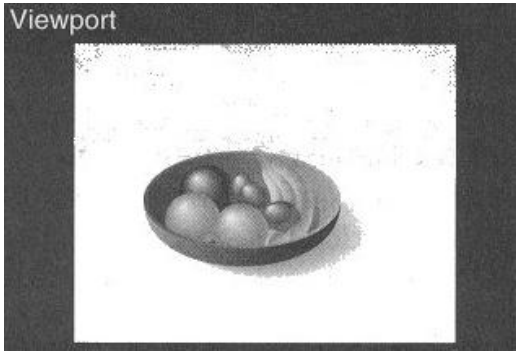
1. Coordinate spaces

- A graphics pipeline takes a description of a scene in 3D space and maps it into a 2D projection on the display space.



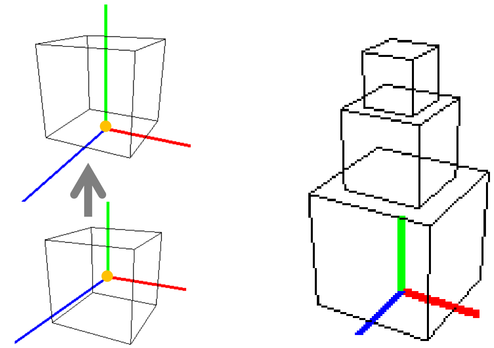




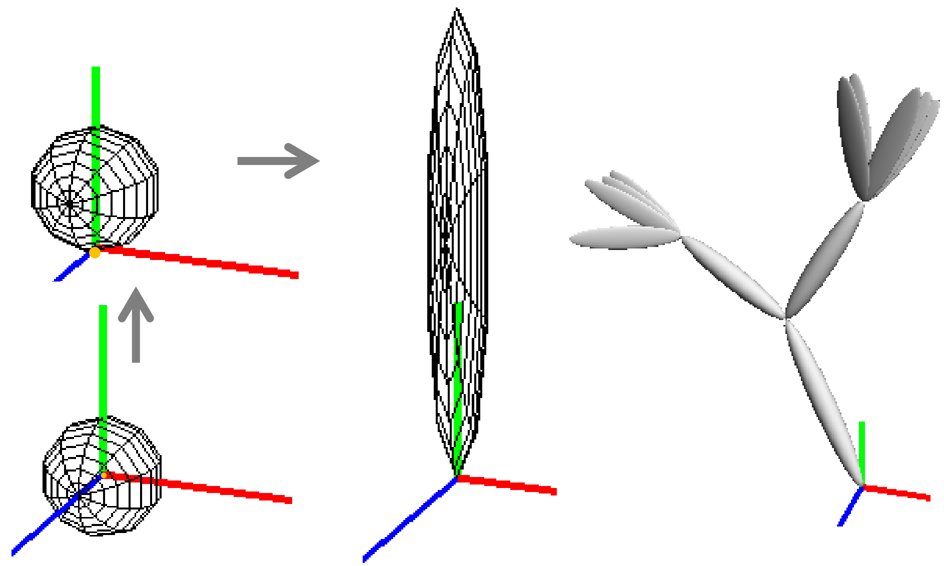


2. Local Coordinate System (LCS)

- Store vertices of a polygon mesh object with respect to some point located in, on, or near the object
- Promotes modelling flexibility
- Transform in LCS to produce basic object shape
  - Put coordinate frame at a sensible point to ease joining with other objects
  - Scaling can cause normal calculation complexities (e.g. for lighting), there is an argument that the object should be defined at the right scale
- Supports object instantiation



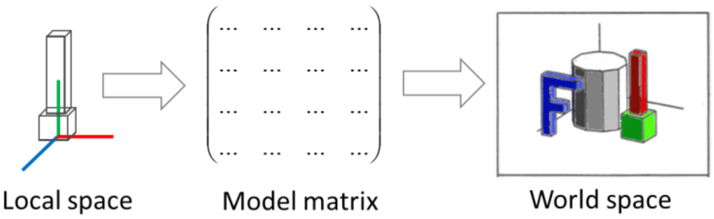
If cube side=1, shifting up produces a cube of height 1 with base centred on origin

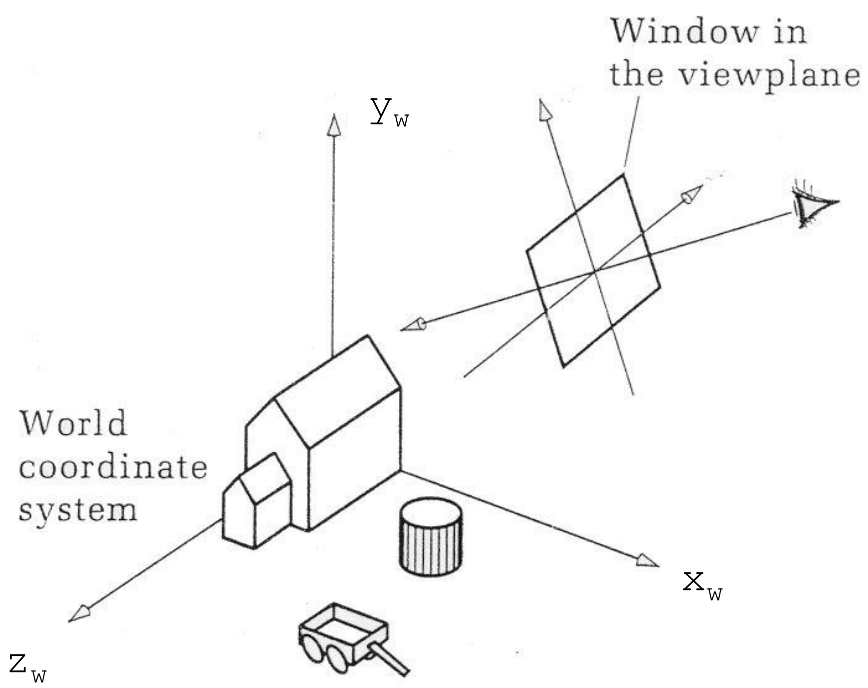


If radius=1, shifting up produces a sphere of height 2 with south pole at origin

3. World Coordinate System (WCS)

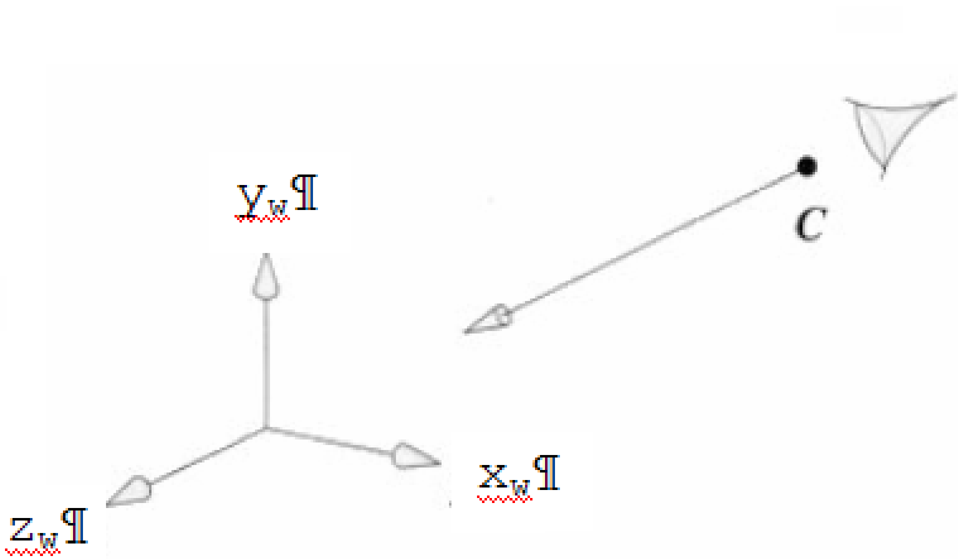
- To create a scene, transform all objects into one common, global coordinate system
  - Right handed system
- Define light source(s) and camera(s)
- Specify surface attributes, e.g. texture and colour of objects (see later lecture)

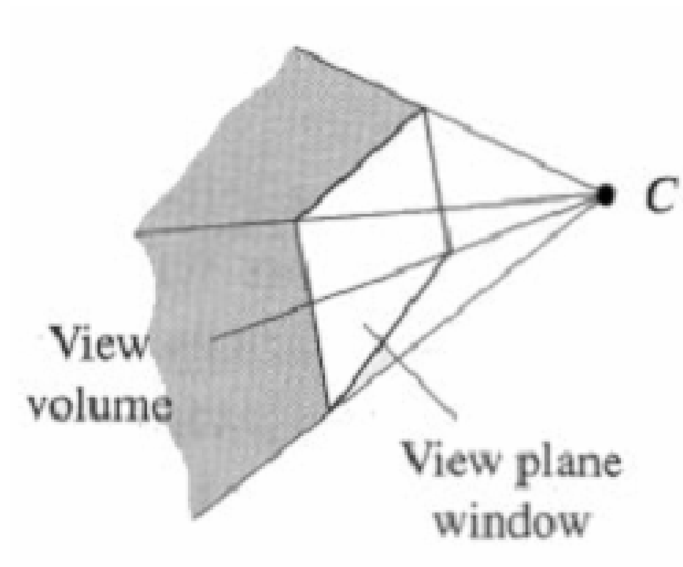




4. Defining view space

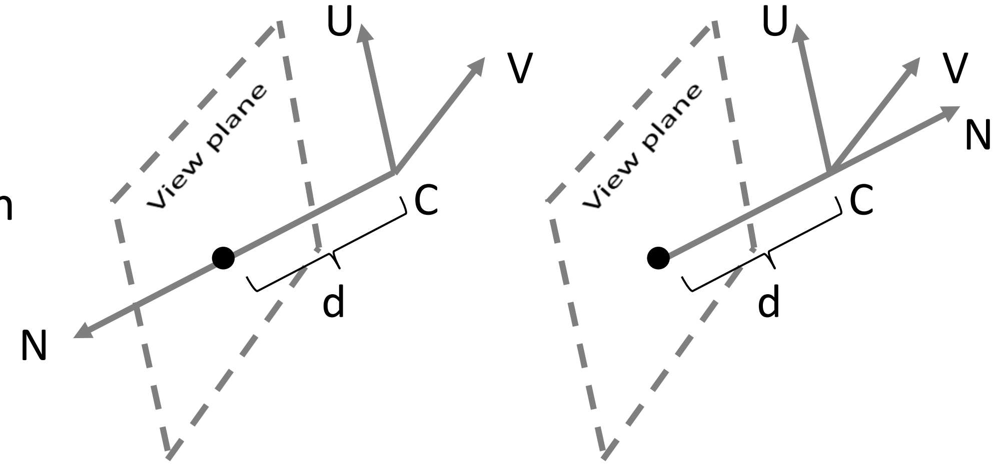
- (Alternative names: Camera space, eye space)
- Establish viewing parameters
  - viewpoint, viewing direction and a view volume
- The view frustum is the field of view





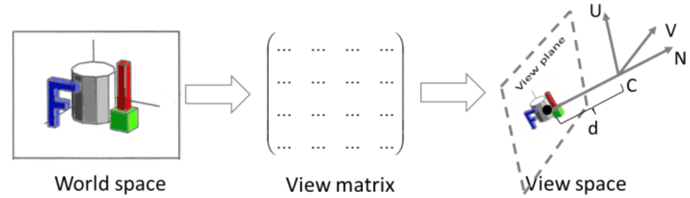
4.1 The camera

- The camera can be positioned anywhere in the WCS, pointed in any direction and rotated about the viewing direction
  - Left handed system
  - (Or right handed system)
- A view point **C**
- A view coordinate system:
  - z **N** – viewing direction
  - y **U** – ‘up’ vector
  - x V – cross product of N and U
- A view plane (parallel to V and U) onto which the scene is projected
  - **d** – distance from C



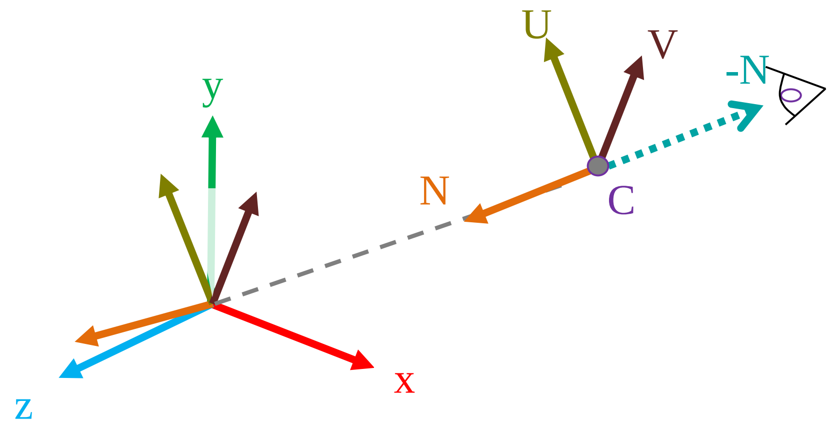
Jamin, Michigan State Univ, 2013

4.2 World to view coordinates

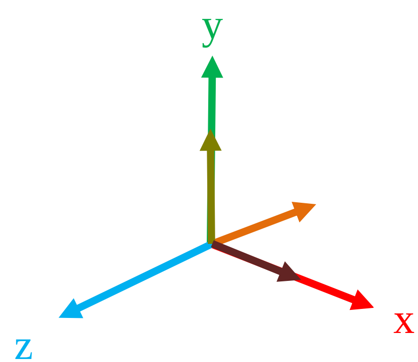


- Transform world coordinates to view coordinates

$$\begin{pmatrix} x_v \\ y_v \\ z_v \\ 1 \end{pmatrix} = T_{view} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix}$$

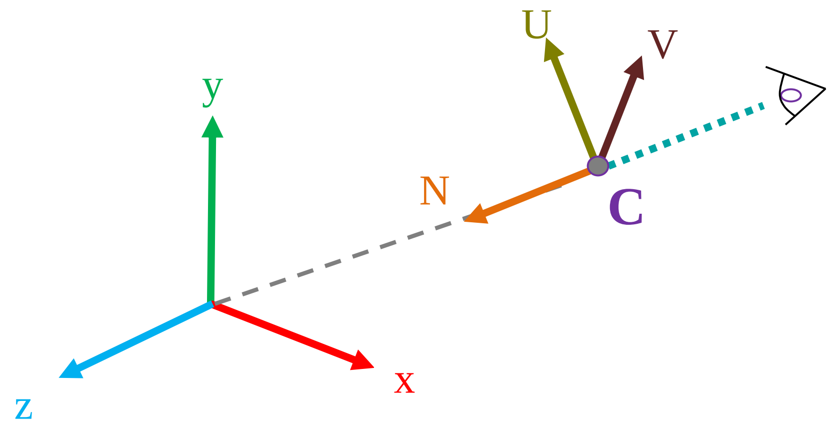


- Translate C to world origin
- Rotate UVN to align with world axes
  - N is on negative Z axis
- Inverse of these processes converts world coordinates to view coordinates
  - Need to change z coordinate too



4.3 User interface considerations

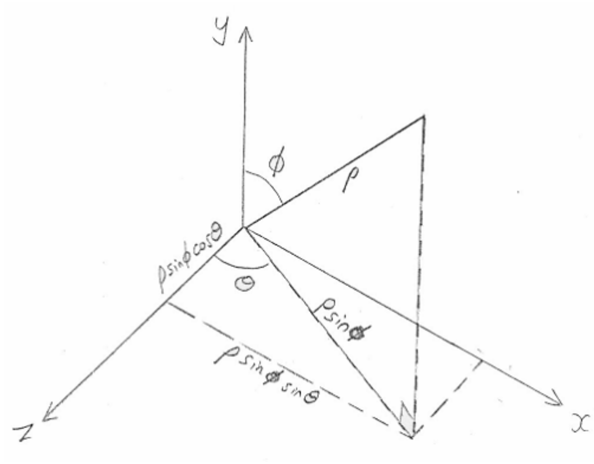
- **C** is specified as a 3D coordinate
- **N** can be calculated by giving a 3D coordinate to look at
- U and V are more problematic
- User specifies U', e.g. (0,1,0)
  - V = U' x N
  - U = V x N
  - Known as Gram-Schmidt process
- C,U,V,N can now be used to form the view matrix

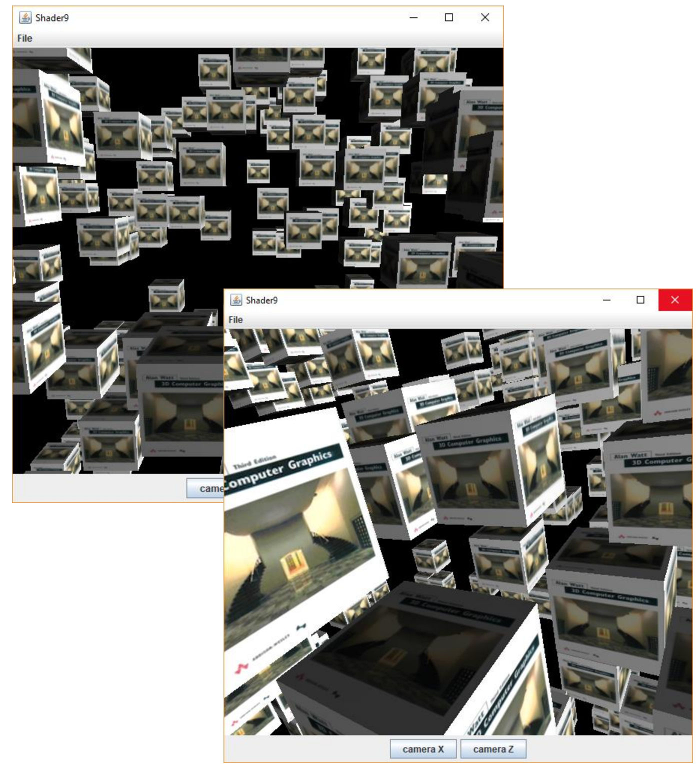


- (Note if N specified in opposite direction:
  - V = N x U'; U = N x V)

4.3.1 Example camera system

- Camera position at a fixed radius could be controlled by a mouse by using two angles (θ, Ø) in a spherical coordinate system centred on the world origin
- We can create a camera that stays in one position, but can point in any direction
  - Stand still and move your head around to look in different directions
  - Again, controlled by mouse and two angles
- Can also add ability to move in direction the camera is looking or sideways/up/down





4.4 Culling (= back-face elimination)

- Compare orientation of each polygon with viewpoint direction and remove polygons that are facing away from the camera
- visible = $N_p \cdot S > 0$

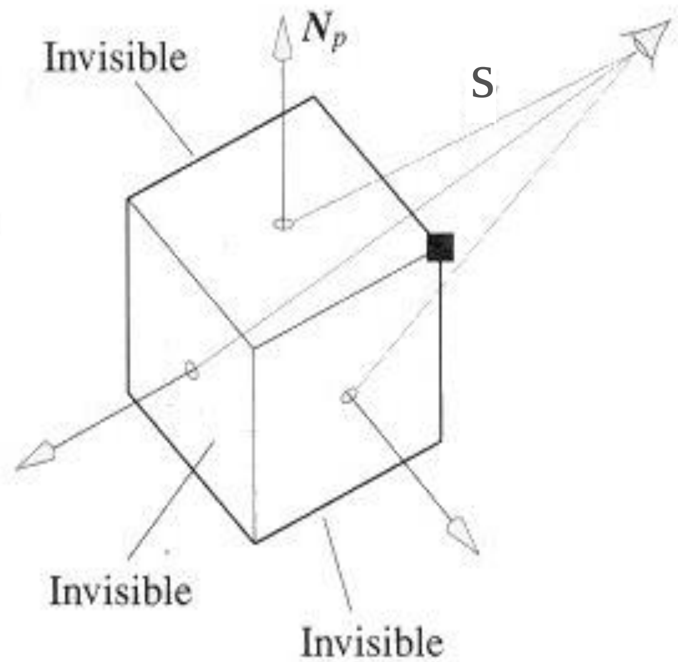
  $N_p \cdot S = |N_p||S| \cos\theta$

  where

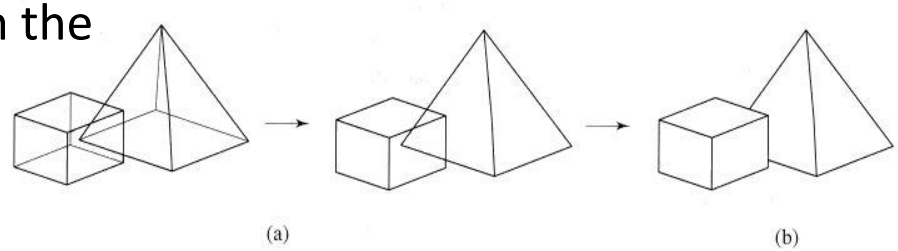
  N_p is the polygon normal

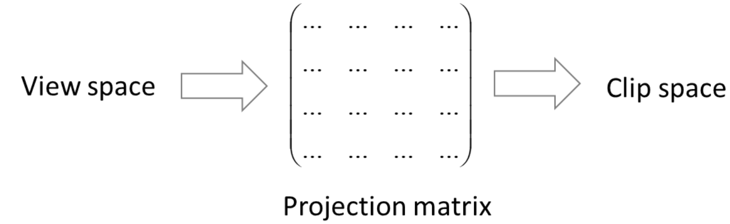
  S is the 'line of sight' vector (use any vertex on the polygon for simplicity)

  θ is the angle between N_p and S



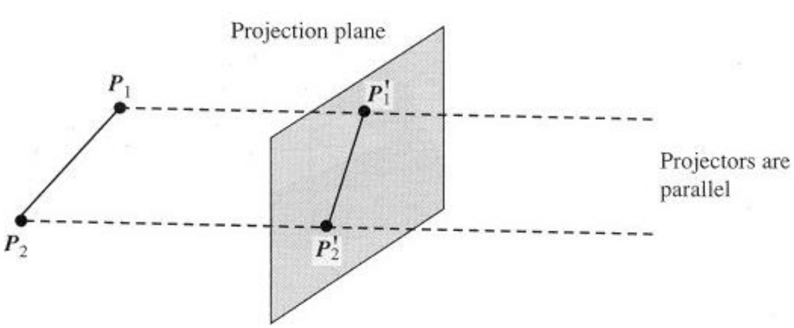
- (a) Culling only removes complete polygons that are facing away from the camera
- (b) HSR deals with the general problem of partial obscuration



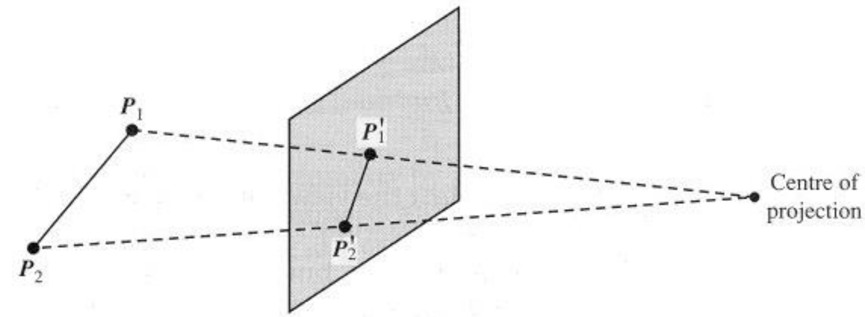


5. Projection

- This is the conversion from view space to clip space (and 3D screen space)
- Two basic projections:
  - Orthographic (or parallel) and Perspective



Parallel projection

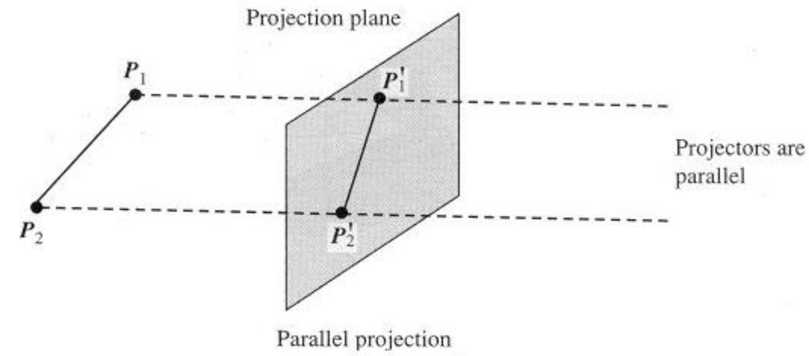


Perspective projection

Jamin, Michigan State Univ, 2013

5.1 Orthographic projection

- Used in CAD, where measurement is important



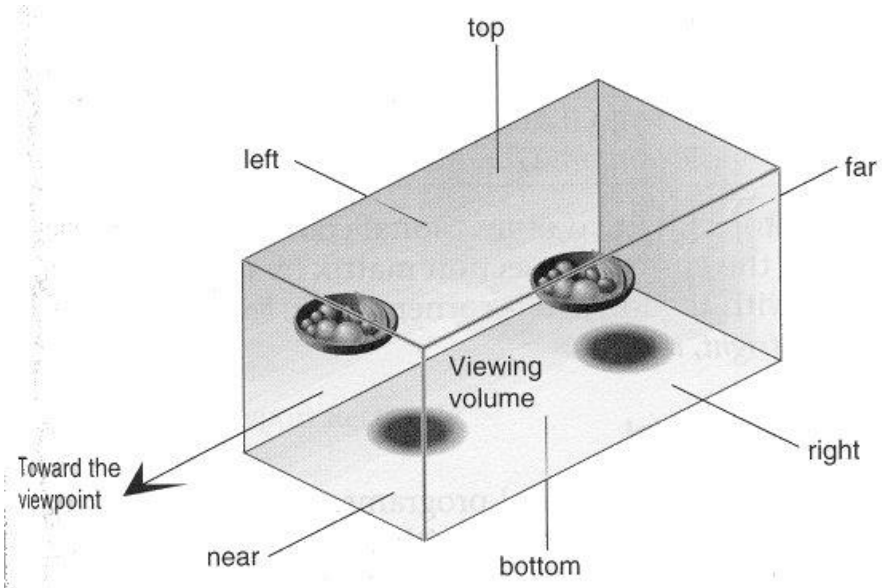
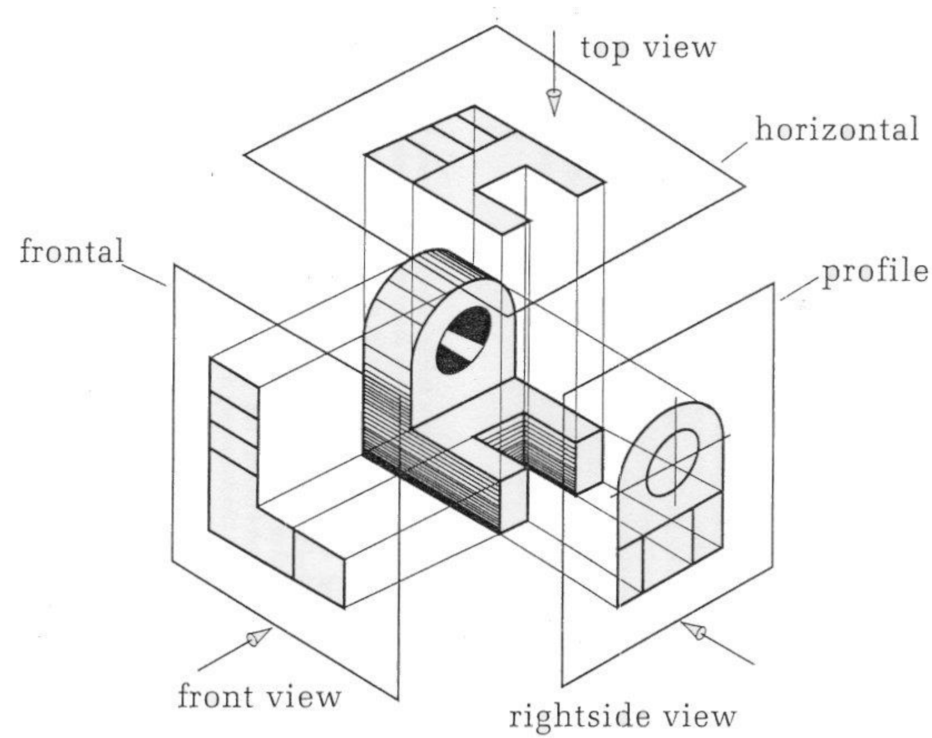
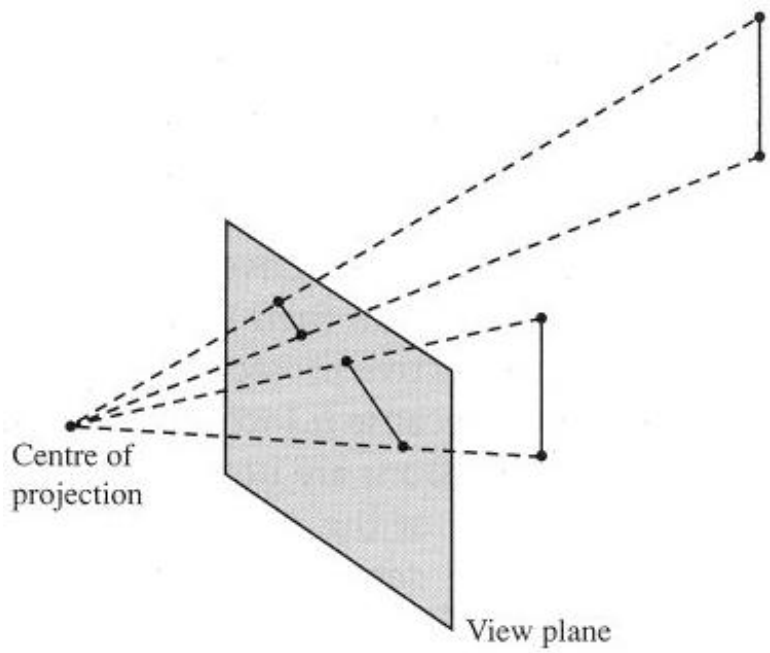


Figure 3-15 Orthographic Viewing Volume



5.2 Perspective projection

- A perspective projection is the more common choice in computer graphics
- A perspective projection incorporates foreshortening
  - Relative dimensions are not preserved
  - Enables perception of depth



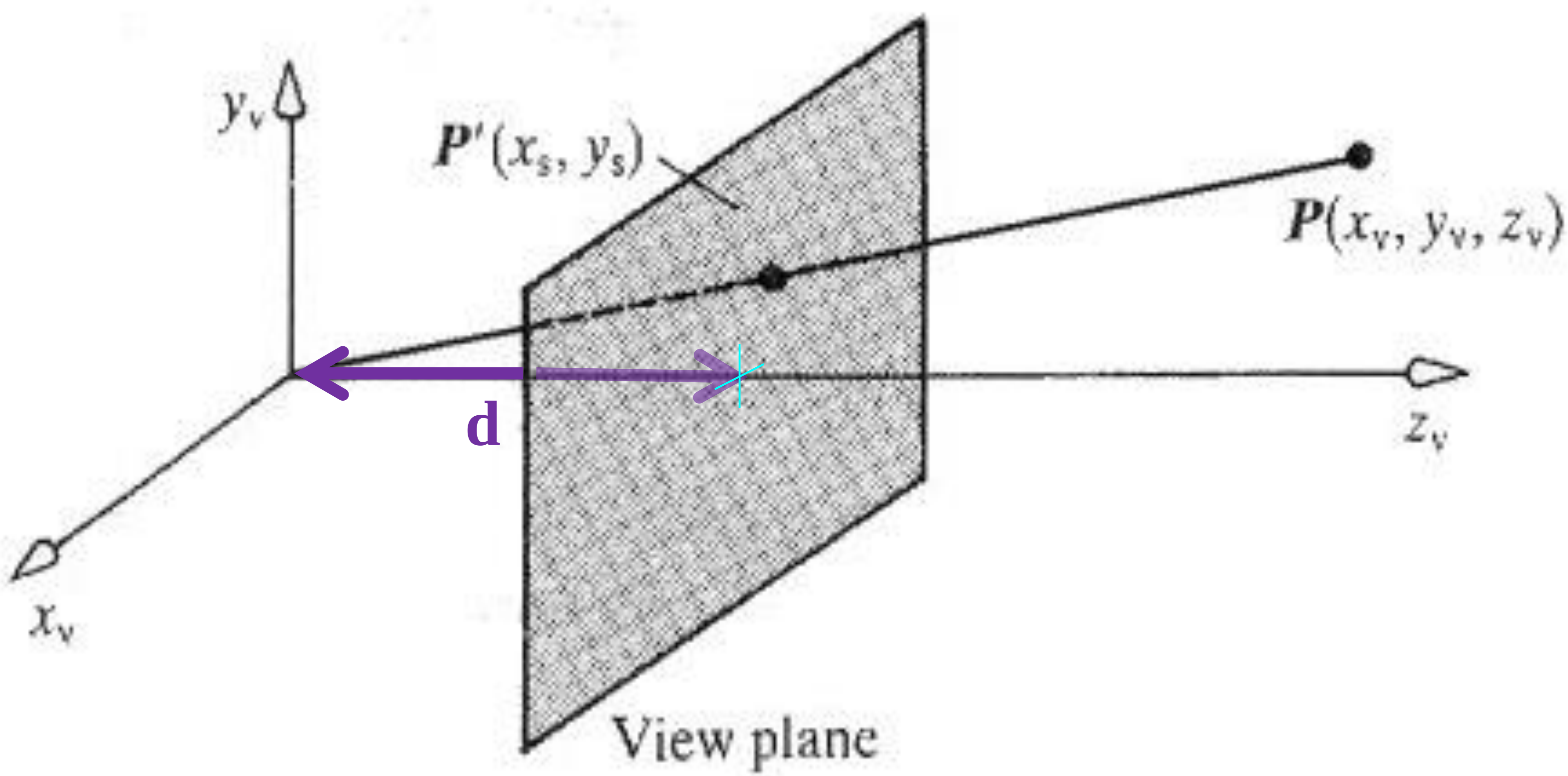
5.3 False perspective

- *Satire on False Perspective*, William Hogarth, 1754, engraving
- "Whoever makes a DESIGN without the Knowledge of PERSPECTIVE will be liable to such Absurdities as are shewn in this Frontiſpiece"
- Some 'errors':
  - The man in the foreground's fishing rod's line passes behind that of the man behind him.
  - The sign is moored to two buildings, one in front of the other, with beams that show no difference in depth
  - The sign is overlapped by two distant trees.
  - The man climbing the hill is lighting his pipe with the candle of the woman leaning out of the upper story window.
  - The crow perched on the tree is massive in comparison to it.
  - …

http://en.wikipedia.org/wiki/Satire_on_False_Perspective

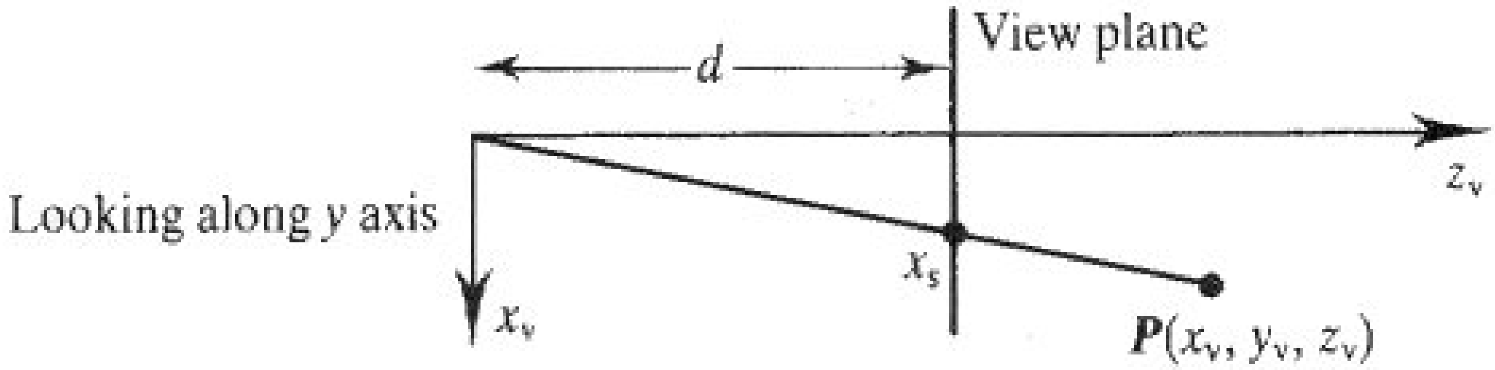
5.4 Deriving a perspective projection

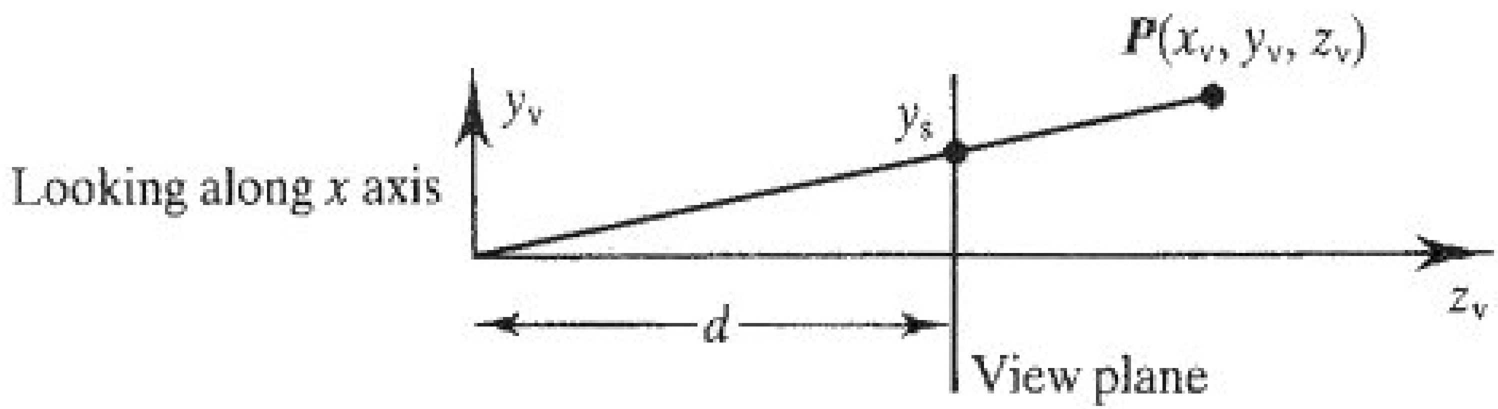
- Define a focal distance, d
  - Distance from camera to view plane



5.4 Deriving a perspective projection

- Similar triangles gives: $$\frac{x_s}{d} = \frac{x_v}{z_v} \qquad \frac{y_s}{d} = \frac{y_v}{z_v}$$





5.4 Deriving a perspective projection

$$\frac{x_s}{d} = \frac{x_v}{z_v} \qquad \frac{y_s}{d} = \frac{y_v}{z_v}$$

- Rearranging, setting w=zv/d, and using homogeneous coordinates:

$$\begin{pmatrix} X \\ Y \\ Z \\ w \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix}}_{\boxed{\mathrm{T}_{\mathrm{pers}}}} \begin{pmatrix} x_v \\ y_v \\ z_v \\ 1 \end{pmatrix}$$

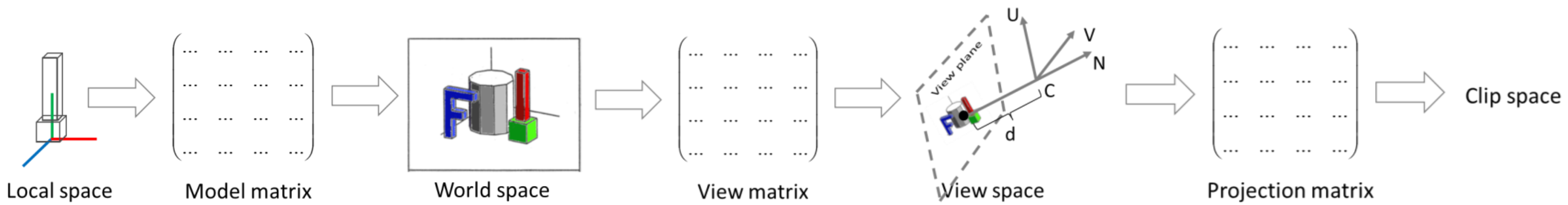
- The perspective divide is clear; screen coordinates are given by:

$$x_s = X/w$$
$$y_s = Y/w$$
$$z_s = Z/w$$

A more complex perspective matrix used in practice is shown in the Appendix.

5.5 Combining the matrices

- Every vertex v_i is transformed by ModelViewProjection matrix
- This is done in the vertex shader

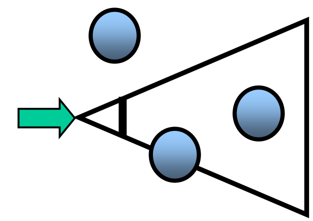


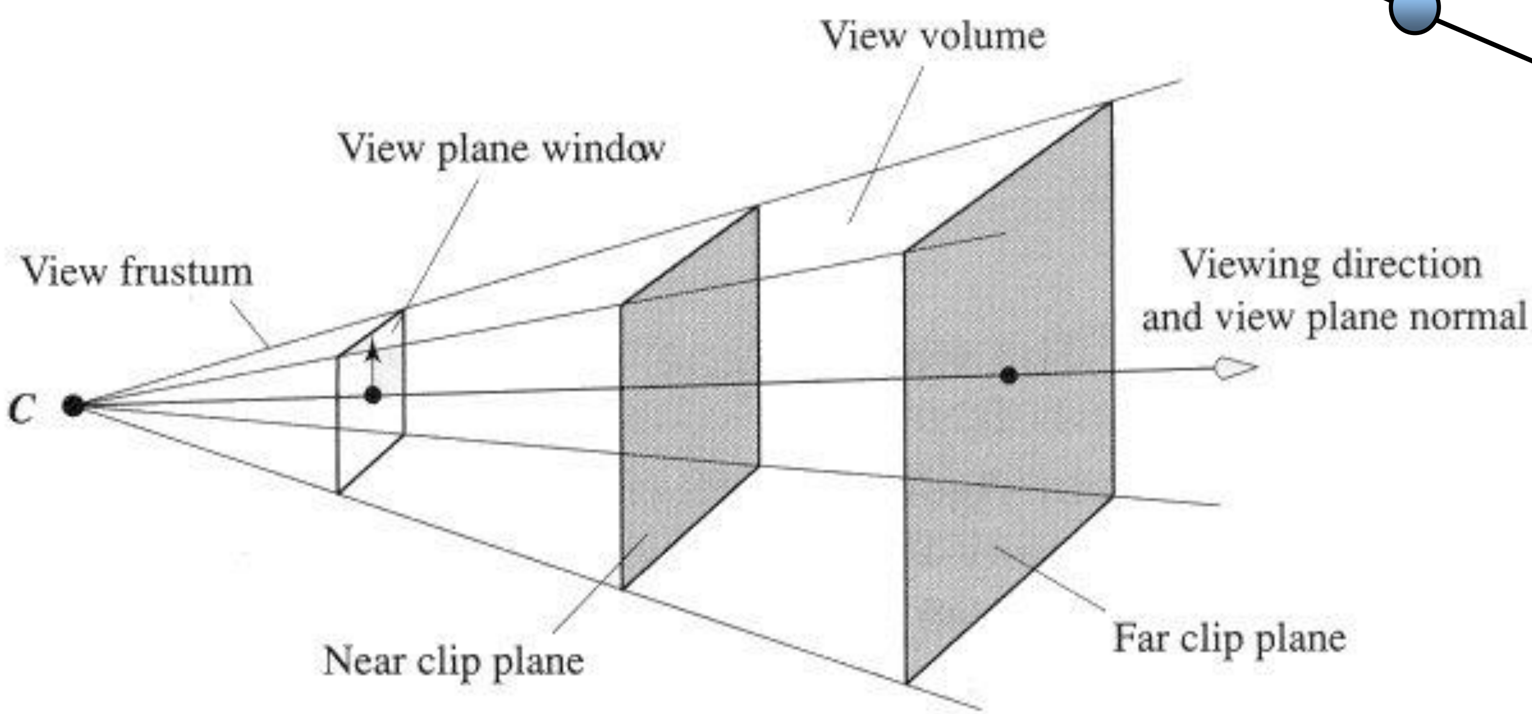
MVPmatrix = projection * view * model

transformed_vertex_position = MVPmatrix * vertex_position

6. The view volume

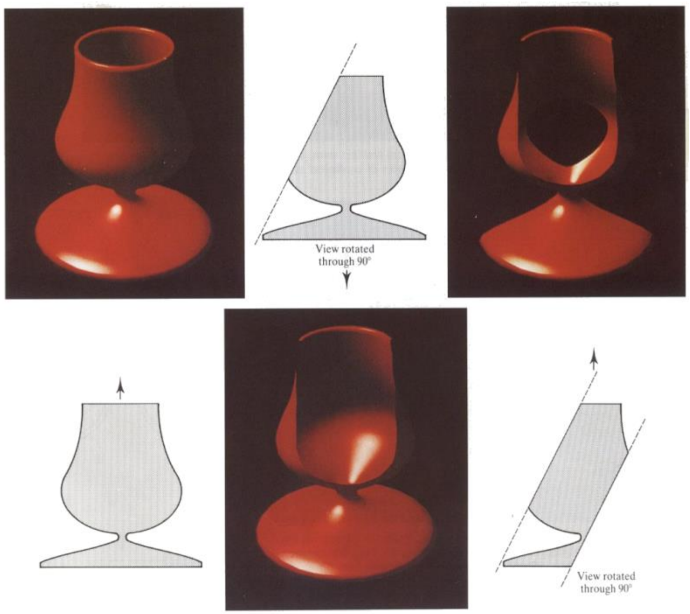
- Defined by a view plane window, has a finite width and height, and near and far clipping planes





6.1 Clipping planes

- Near plane clips things 'behind' the camera
- Far plane clips distant things
  - In games, use 'fog' to blend out



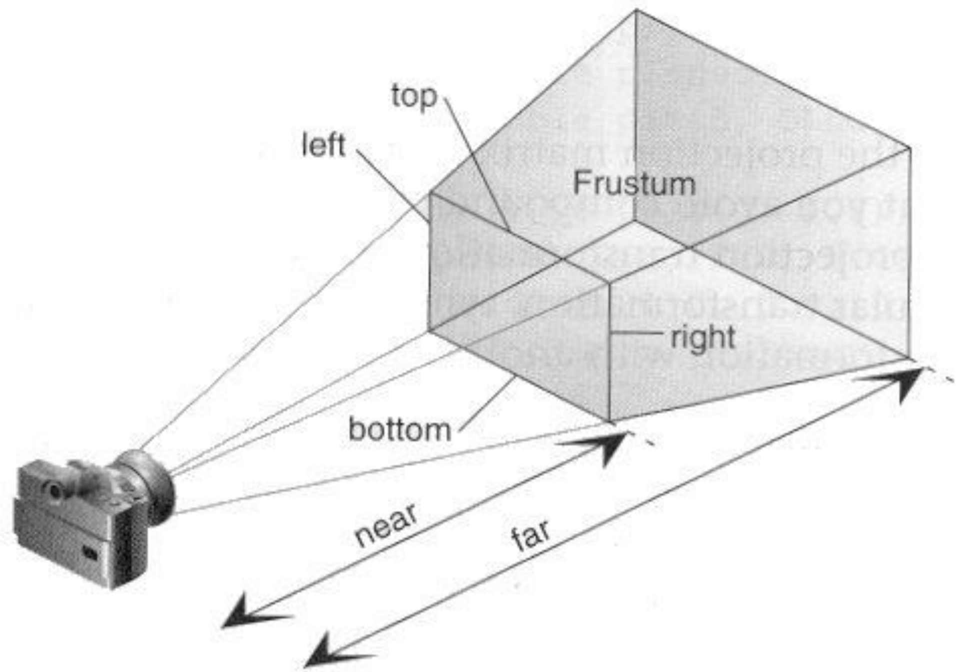
2 A view volume interacting with a shaded object. Bringing near and far planes into coincidence with an object; *(centre)* near plane coincides; *(right)* both planes coincide with the object.

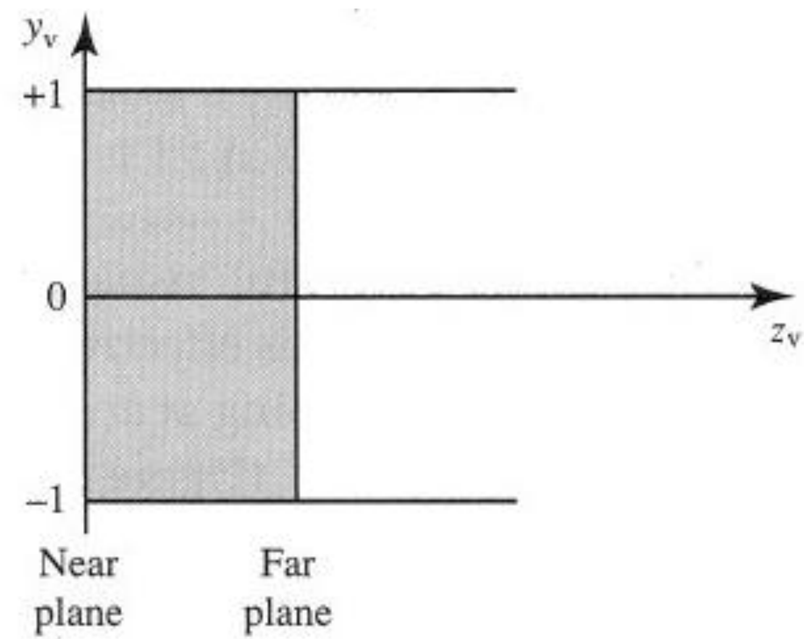
Finite view volume

Finite view volume + fog

7. 3D Screen space

- We need depth information to decide what's in front
- View space (xs,ys,zs) is transformed into a box-shaped space (xs,ys) (-1 to +1), with z_s as the depth (0 to 1), which supports easy clipping

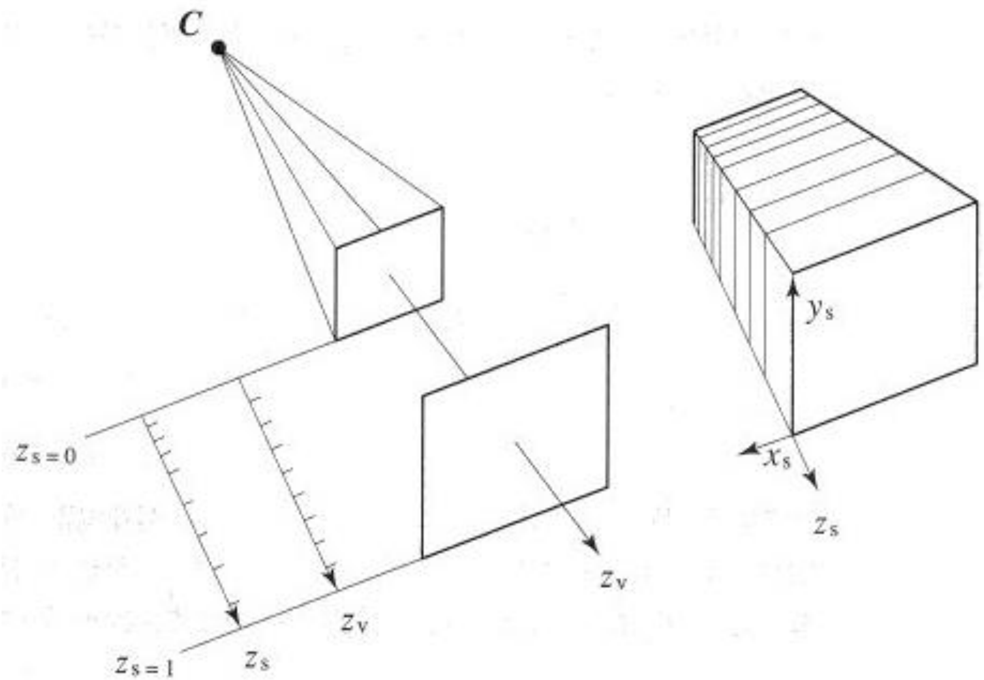


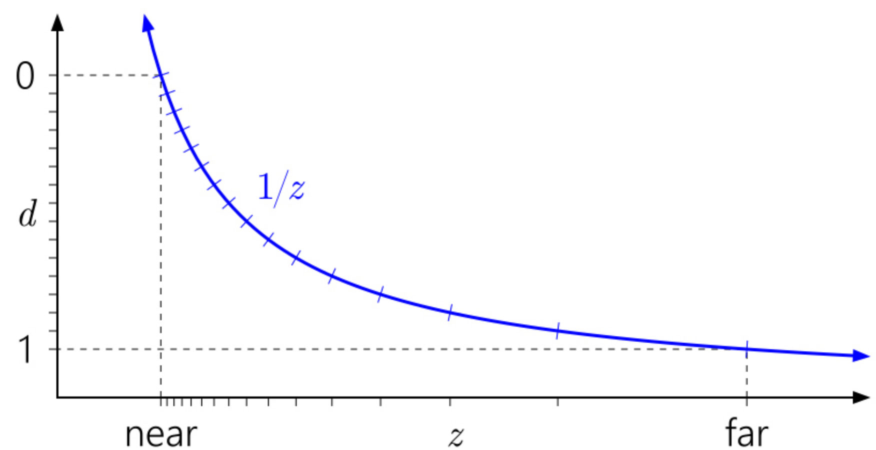


OpenGL red book

7.1 The relationship between z_v and z_s

- Interpolating along a line in view space (eye space) is not the same as interpolating this line in 3D screen space
- As z_v approaches the far clipping plane, z_s approaches 1 more rapidly
- Thus objects get distorted towards the back of the view frustum





https://developer.nvidia.com/content/depth-precision-visualized

7.2 3D Screen space

- *Later Lecture*: Rendering processes:
  - Rasterisation – decide which pixels are covered by polygons.
  - Hidden surface removal (HSR) – decide what is in front.
  - Shading – decide what colour things are.

Vertex shader
- uses ModelViewProjection matrix to transform vertex positions into 3D screen space

Primitive assembly and clipping

Rasterisation
- interpolates vertex values for a polygon to produce a set of fragments for the inside of the polygon

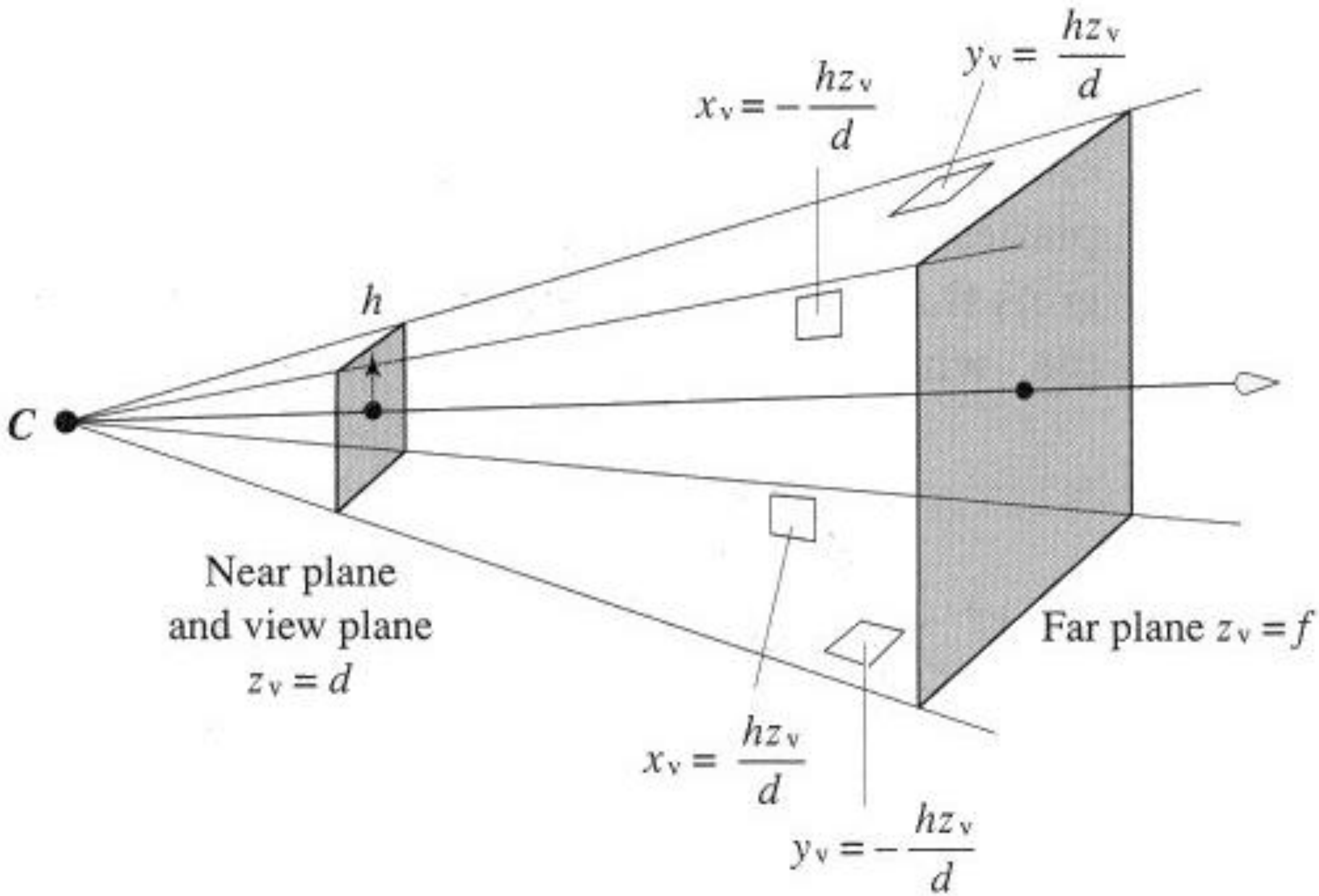
Fragment shader
- calculates final values for the fragment which are written to relevant buffers, e.g. pixel colour and depth value

8. Summary

- Using a local coordinate system to define objects supports the idea of object instantiation
- Individual objects are brought together in the (global) world space
- The conversion from world to view space involves translation and rotation
- The conversion from view space to screen space is called *projection*
- A ModelViewProjection matrix is used to transform vertices from local space to 3D screen space

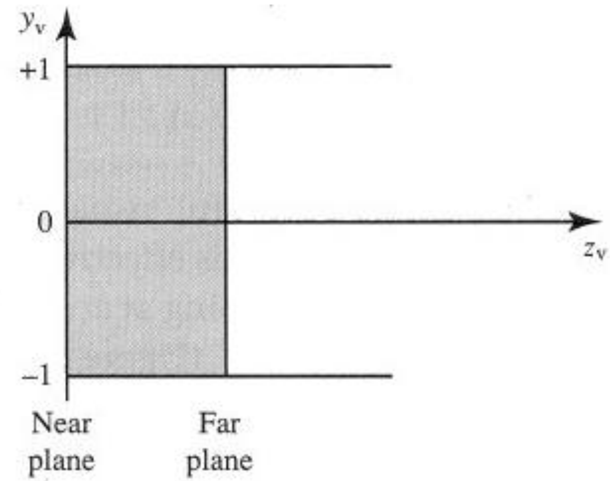
A1. The view volume

- Simplify diagram by setting width = height = 2h



A.2 3D screen space

- (Newman and Sproull, 73)
- Useful properties:
  - Image plane width and height map to [–1..1]
  - Points on the image plane should map to $z_s = 0$
  - Points on the far clip plane should map to $z_s = 1$
  - Intersections of lines and planes in view space should map to their intersections in screen space
  - Straight lines should transform to straight lines
  - Planes should transform to planes
- This is the case if $z_s = A + B/z_v$
- Using the properties above as constraints, together with the view volume, leads to:



$$x_s = d\,\frac{x_v}{hz_v}$$

$$y_s = d\,\frac{y_v}{hz_v}$$

$$z_s = \frac{f\left(1 - d/z_v\right)}{\left(f - d\right)}$$

A.3 Matrix form

- Using homogeneous coordinates we can write:

$$\begin{pmatrix} X \\ Y \\ Z \\ w \end{pmatrix} = \begin{pmatrix} d/h & 0 & 0 & 0 \\ 0 & d/h & 0 & 0 \\ 0 & 0 & f/(f-d) & -df/(f-d) \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_v \\ y_v \\ z_v \\ 1 \end{pmatrix} = T_{pers} \begin{pmatrix} x_v \\ y_v \\ z_v \\ 1 \end{pmatrix}$$

A.4 Decomposing the transformation

- To see what is happening, we can break the view to screen space into two parts:

$$T_{pers} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & f/(f-d) & -df/(f-d) \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} d/h & 0 & 0 & 0 \\ 0 & d/h & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= T_{pers2} T_{pers1}$$

- T_{pers1} scales by d/h in x and y, so that the side clipping planes are of the form x=z and y=z
- T_{pers2} maps the regular pyramid into a box, with the far plane at z=1