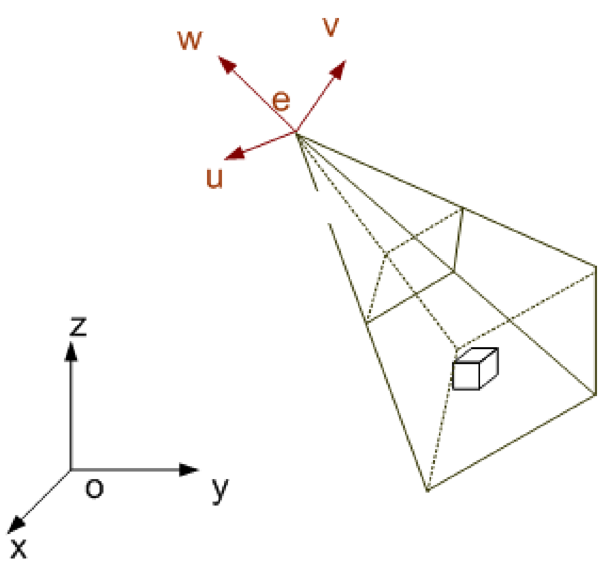
View Volumes

- Define 3D volume seen by camera

Perspective view volume

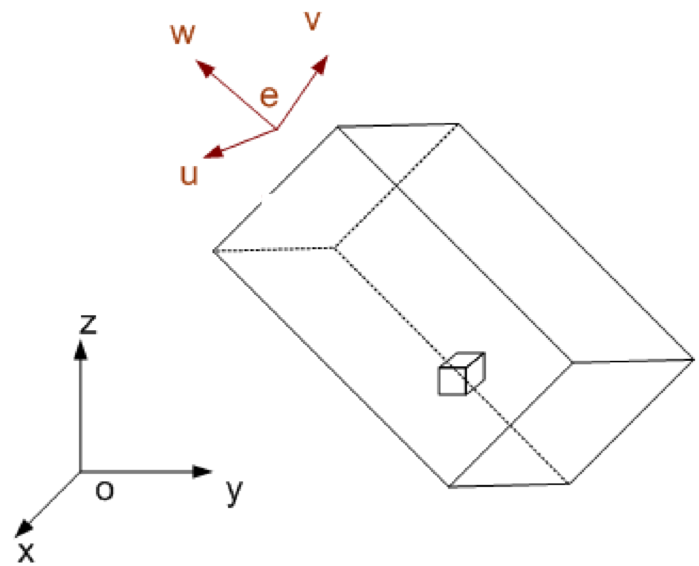
Camera coordinates



World coordinates

Orthographic view volume

Camera coordinates

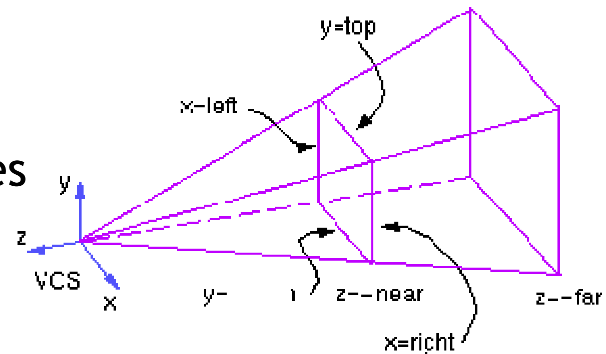


World coordinates

Perspective View Volume

General view volume

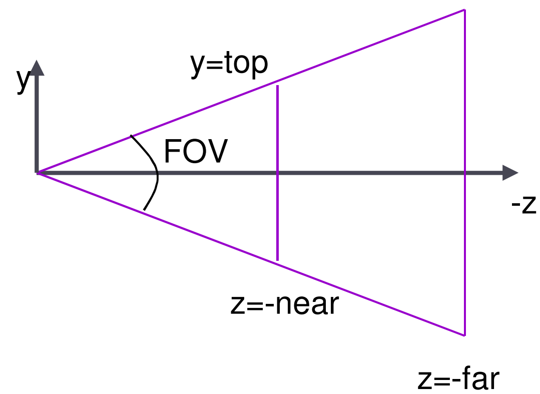
Camera
coordinates



- Defined by 6 parameters, in camera coordinates
  - Left, right, top, bottom boundaries
  - Near, far clipping planes
- Clipping planes to avoid numerical problems
  - Divide by zero
  - Low precision for distant objects
- Usually symmetric, i.e., left=-right, top=-bottom

Perspective View Volume

Symmetrical view volume



- Only 4 parameters
  - Vertical field of view (FOV)
  - Image aspect ratio (width/height)
  - Near, far clipping planes

$$\text{aspect ratio}=\frac{right-left}{top-bottom}=\frac{right}{top}$$

$$\tan(FOV/2)=\frac{top}{near}$$

Canonical View Volume

- Projection matrix is set such that
  - User defined view volume is transformed into canonical view volume, i.e., cube [-1,1]x[-1,1]x[-1,1]
  - Multiplying vertices of view volume by projection matrix and performing homogeneous divide yields canonical view volume
- Perspective and orthographic projection are treated exactly the same way
- Canonical view volume is last stage in which coordinates are in 3D
- Next step is projection to 2D frame buffer

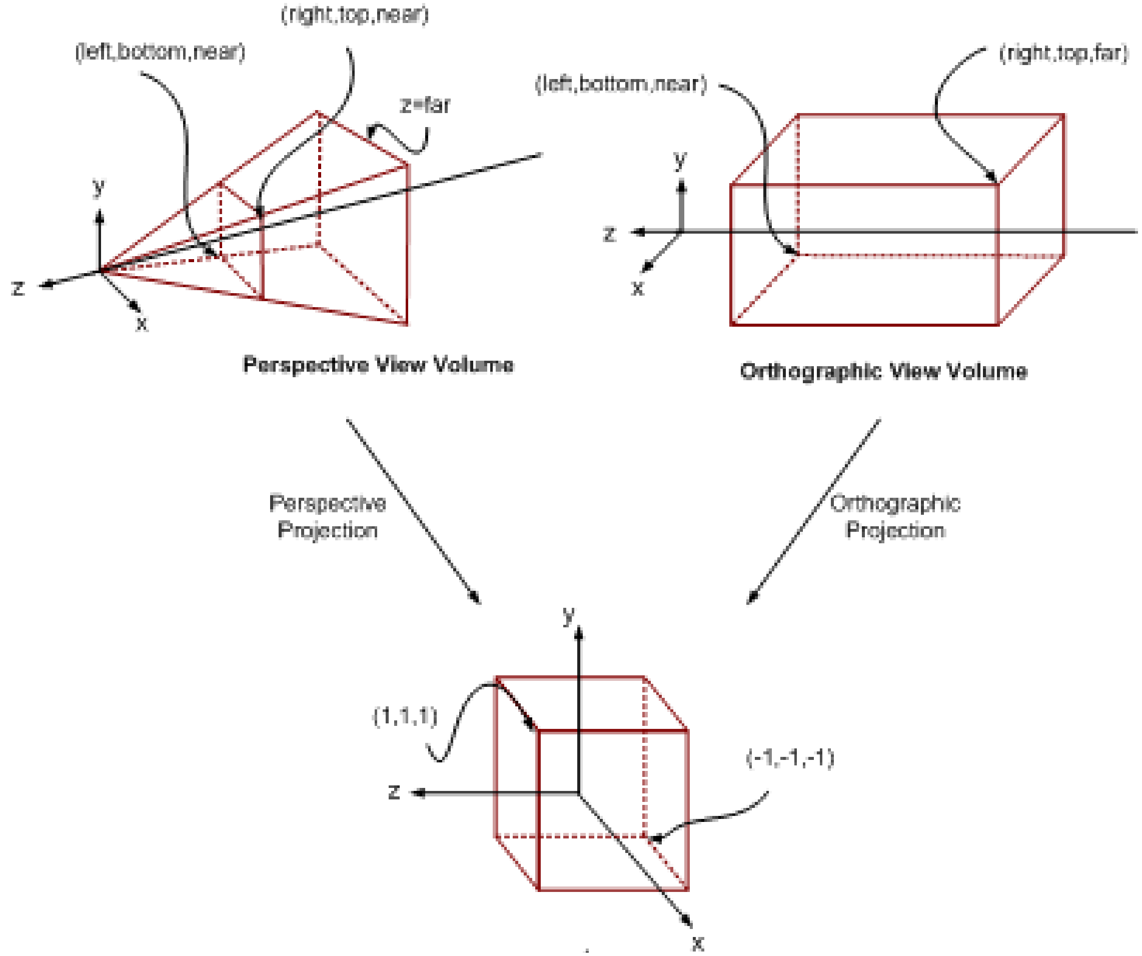
Projection Matrix

Camera coordinates

Projection matrix

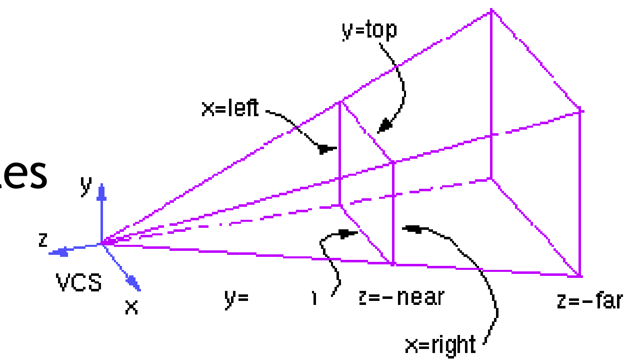
Canonical view volume

Clipping



Perspective Projection Matrix

- General view frustum with 6 parameters



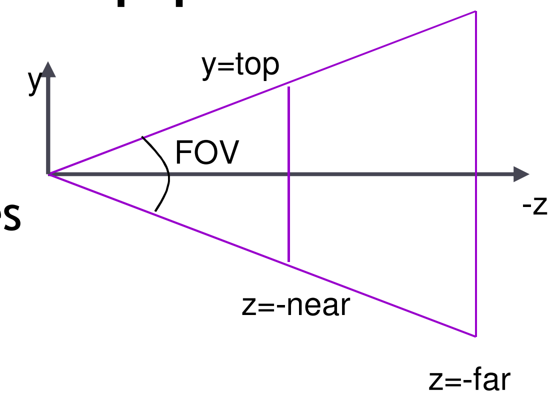
$$\mathbf{P}_{persp}(left, right, top, bottom, near, far) =$$

$$\begin{bmatrix} \frac{2near}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2near}{top-bottom} & \frac{top+bottom}{top-bottom} & 0 \\ 0 & 0 & \frac{-(far+near)}{far-near} & \frac{-2far\cdot near}{far-near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Perspective Projection Matrix

- Symmetrical view frustum with field of view, aspect ratio, near and far clip planes

Camera coordinates



$$\mathbf{P}_{persp}(FOV, aspect, near, far) = \begin{bmatrix} \frac{1}{aspect \cdot \tan(FOV/2)} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(FOV/2)} & 0 & 0 \\ 0 & 0 & \frac{near + far}{near - far} & \frac{2 \cdot near \cdot far}{near - far} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

The Complete Transform

- Mapping a 3D point in object coordinates to pixel coordinates:

$$\mathbf{p}' = \mathbf{D}\mathbf{P}\mathbf{C}^{-1}\mathbf{M}\underbrace{\mathbf{p}}_{\text{Object space}}$$

  - **M**: Object-to-world matrix
  - **C**: camera matrix
  - **P**: projection matrix
  - **D**: viewport matrix

The Complete Transform

- Mapping a 3D point in object coordinates to pixel coordinates:

$$\mathbf{p}' = \mathbf{D}\mathbf{P}\mathbf{C}^{-1}\mathbf{M}\mathbf{p}$$

Object space

World space

  - **M**: Object-to-world matrix
  - **C**: camera matrix
  - **P**: projection matrix
  - **D**: viewport matrix

The Complete Transform

- Mapping a 3D point in object coordinates to pixel coordinates:

$$\mathbf{p}' = \mathbf{D}\mathbf{P}\underbrace{\mathbf{C}^{-1}\underbrace{\mathbf{M}\underbrace{\mathbf{p}}_{\text{Object space}}}_{\text{World space}}}_{\text{Camera space}}$$

  - **M**: Object-to-world matrix
  - **C**: camera matrix
  - **P**: projection matrix
  - **D**: viewport matrix

The Complete Transform

- Mapping a 3D point in object coordinates to pixel coordinates:

$$\mathbf{p}' = \mathbf{D}\mathbf{P}\mathbf{C}^{-1}\mathbf{M}\mathbf{p}$$

Object space

World space

Camera space

Canonical view volume

  - **M**: Object-to-world matrix
  - **C**: camera matrix
  - **P**: projection matrix
  - **D**: viewport matrix

The Complete Transform

- Mapping a 3D point in object coordinates to pixel coordinates: $\mathbf{p}' = \mathbf{D}\mathbf{P}\mathbf{C}^{-1}\mathbf{M}\mathbf{p}$

Object space

World space

Camera space

Canonical view volume

Image space

  - **M**: Object-to-world matrix
  - **C**: camera matrix
  - **P**: projection matrix
  - **D**: viewport matrix

The Complete Transform

- Mapping a 3D point in object coordinates to pixel coordinates:

$$\mathbf{p}' = \mathbf{D}\mathbf{P}\mathbf{C}^{-1}\mathbf{M}\mathbf{p}$$

$$\mathbf{p}' = \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix}$$

Pixel coordinates: x'/w' y'/w'

  - **M**: Object-to-world matrix
  - **C**: camera matrix
  - **P**: projection matrix
  - **D**: viewport matrix

The Complete Transform in OpenGL

- Mapping a 3D point in object coordinates to pixel coordinates:

OpenGL GL_MODELVIEW matrix

$$\mathbf{p}' = \mathbf{D}\underbrace{\mathbf{P}}_{\text{OpenGL GL\_PROJECTION matrix}}\overbrace{\mathbf{C}^{-1}\mathbf{M}}^{\text{OpenGL GL\_MODELVIEW matrix}}\mathbf{p}$$

OpenGL GL_PROJECTION matrix

  - **M**: Object-to-world matrix
  - **C**: camera matrix
  - **P**: projection matrix
  - **D**: viewport matrix

The Complete Transform in OpenGL

- GL_MODELVIEW, $\mathbf{C^{-1}M}$
  - Defined by programmer
- GL_PROJECTION, **P**
  - Utility routines to set it by specifying view volume: glFrustum(), glPerspective(), glOrtho()
  - Do not use utility functions in homework project 2
  - You will implement a software renderer in project 3, which will not use OpenGL
- Viewport, **D**
  - Specify implicitly via glViewport()
  - No direct access with equivalent to GL_MODELVIEW or GL_PROJECTION