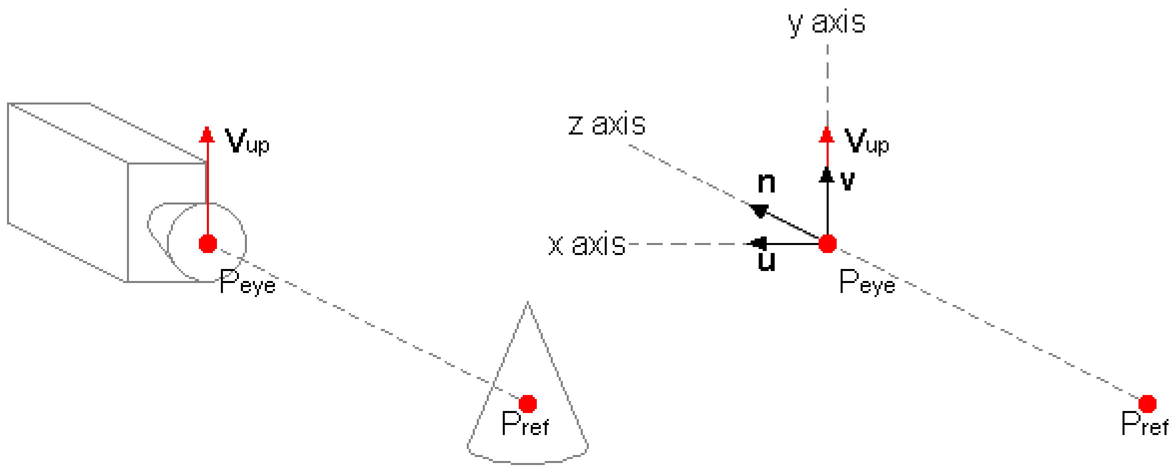
CSC 418/2504 – Computer Graphics, Fall 2003

Tutorial 3. Viewing Transformations

---

1. Viewing and Modeling Transformation – *modelview matrix*

Derivation



- to express points in world coordinates (WCS) in terms of camera (VCS)
- defined by:
  - an eye point P_{eye}
  - a reference point P_{ref}
  - an up vector $\mathbf{V}_{up}$
- unit vectors of VCS (call them **i**, **j**, **k** if you prefer)

$$\mathbf{n} = \frac{P_{eye} - P_{ref}}{| P_{eye} - P_{ref} |}$$

$$\mathbf{u} = \frac{\mathbf{V}_{up} \times \mathbf{n}}{| \mathbf{V}_{up} \times \mathbf{n} |}$$

$$\mathbf{v} = \mathbf{n} \times \mathbf{u}$$

- intuition

- suppose P_{eye} is fixed
- to pan the camera (like shaking your head left and right)
  - move P_{ref} "horizontally"
  - corresponds to rotating the VCS along the y axis
- to tilt the camera (like nodding your head up and down)
  - move P_{ref} "vertically"
  - corresponds to rotating the VCS along the x axis
- to rock the camera (like tilting your head left and right)
  - let P_{eye} - P_{ref} be the normal vector of a plane A
  - change $\mathbf{V}_{up}$ so that its projection onto A "rotates" left and right
  - corresponds to rotating the VCS along the z axis
- $\mathbf{V}_{up}$ represents a general "upwardness" for the camera

* view matrix
  - first express camera in terms of world:

$$M_{cam} = \begin{bmatrix} 1 & 0 & 0 & P_{eye,x} \\ 0 & 1 & 0 & P_{eye,y} \\ 0 & 0 & 1 & P_{eye,z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - then invert M_{cam} to express world in terms of camera:

$$M_{cam}^{-1} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -P_{eye,x} \\ 0 & 1 & 0 & -P_{eye,y} \\ 0 & 0 & 1 & -P_{eye,z} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

* so $P_{VCS} = M_{cam}^{-1} P_{WCS}$

OpenGL

* performs these calculations internally with a call to gluLookAt()
* code:

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(P_{eye,x}, P_{eye,y}, P_{eye,z}, P_{ref,x}, P_{ref,y}, P_{ref,z}, V_{up,x}, V_{up,y}, V_{up,z} );
```

Example

- scene with camera at (4,4,4), pointed at (0,1,4), with up vector (0,1,0)
- $$\mathbf{n} = \frac{(4,4,4) - (0,1,4)}{|\,(4,4,4) - (0,1,4)\,|} = \frac{(4,3,0)}{|\,(4,3,0)\,|} = (4/5,3/5,0)$$
- $$\mathbf{u} = \frac{(0,1,0) \times (4/5,3/5,0)}{|\,(0,1,0) \times (4/5,3/5,0)\,|} = \frac{(0,0,-4/5)}{|\,(0,0,-4/5)\,|} = (0,0,-1)$$
- $$\mathbf{v} = (4/5,3/5,0) \times (0,0,-1) = (-3/5,4/5,0)$$
- $$M_{cam}^{-1} = \begin{bmatrix} 0 & 0 & -1 & 0 \\ -3/5 & 4/5 & 0 & 0 \\ 4/5 & 3/5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -4 \\ 0 & 1 & 0 & -4 \\ 0 & 0 & 1 & -4 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 & 4 \\ -3/5 & 4/5 & 0 & -4/5 \\ 4/5 & 3/5 & 0 & -28/5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

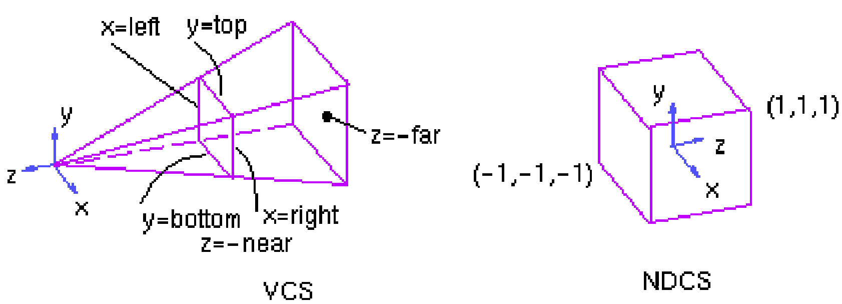
- check
- $$M_{cam}^{-1} \begin{bmatrix} 4 \\ 4 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad M_{cam}^{-1} \begin{bmatrix} 4 \\ 4 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

2. Projection Transformation – projection matrix

Derivation

- maps points in VCS to NDCS
- see Hill, lecture notes for derivation
  - $$\begin{bmatrix} x \\ y \\ z \\ -z/d \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

OpenGL



- code for perspective projection:

- `glMatrixMode(GL_PROJECTION);`
- `glLoadIdentity();`
-
- `gluPerspective(fovy, aspect, near, far);`
- **`or`**
- `glFrustum(left, right, bottom, top, near, far);`

- gluPerspective()
  - fovy
    - field of view in the y-direction, centered about y = 0
    - in degrees
  - aspect
    - aspect ratio that determines the field of view in the x direction
    - ratio of x (width) to y (height)
  - near
    - in camera coordinates
    - close to 0 (but not 0)
  - far
    - in camera coordinates
- glFrustum()
  - left, right, bottom, top, near, far
    - specifies the clipping planes of the view volume explicitly
    - in camera coordinates