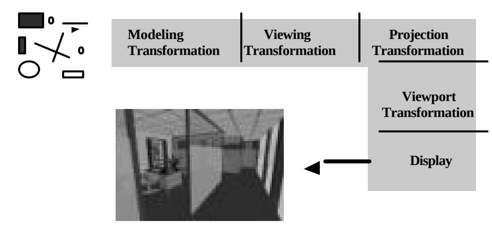
CS 4731: Computer Graphics
Lecture 12: More 3D Viewing

Emmanuel Agu

3D viewing under the hood



3D viewing under the hood

- Topics of Interest:
  - Viewing transformation
  - Projection transformation

Viewing Transformation

- Transform the object from world to eye space
  - Construct eye coordinate frame
  - Construct matrix to perform coordinate transformation
  - Flexible Camera Control

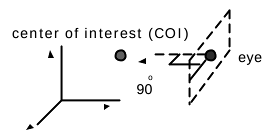
Viewing Transformation

- Recall OpenGL way to set camera:
  - gluLookAt (Ex, Ey, Ez, cx, cy, cz, Up_x, Up_y, Up_z)
  - The view up vector is usually (0,1,0)
  - Remember to set the OpenGL matrix mode to GL_MODELVIEW first
- Modelview matrix:
  - combination of modeling matrix *M* and Camera transforms *V*
- gluLookAt fills *V* part of modelview matrix
- What does gluLookAt do with parameters (*eye, COI, up vector*) you provide?

Eye Coordinate Frame

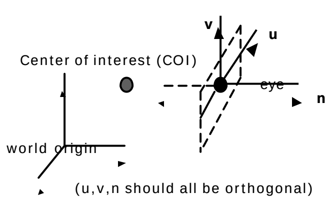
- Known: eye position, center of interest, view-up vector
- To find out: new origin and three basis vectors

Assumption: direction of view is orthogonal to view plane (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,
  - other two (**u** and **v**) span the viewing plane



n is pointing away from the world because we use left hand coordinate system

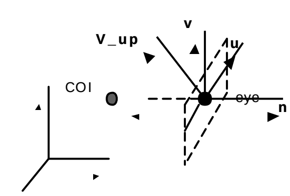
N = eye – COI
n = N / | N |

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

(u,v,n should all be orthogonal)

Eye Coordinate Frame

- How about u and v?



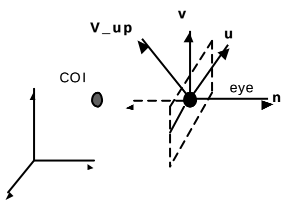
- We can get u first -
  - u is a vector that is perp to the plane spanned by N and view up vector (V_up)

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

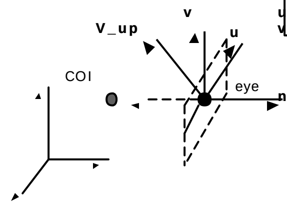
Eye Coordinate Frame

- Put it all together

Eye space **origin:**
(Eye.x , Eye.y, Eye.z)

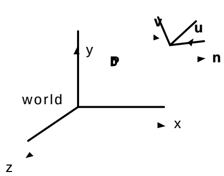
Basis vectors:

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



World to Eye Transformation

- Transformation matrix (M_{w2e}) ?
  P' = M_{w2e} x P

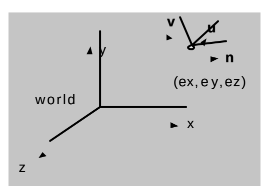


1. Come up with the transformation sequence to move eye coordinate frame to the world

2. And then apply this sequence to the point P in a reverse order

World to Eye Transformation

- Rotate the eye frame to “align” it with the world frame
- Translate (-ex, -ey, -ez)



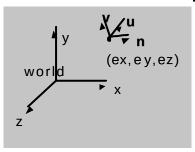
Rotation: $\begin{vmatrix} ux & uy & uz & 0 \\ vx & vy & vz & 0 \\ nx & ny & nz & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$

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

World to Eye Transformation

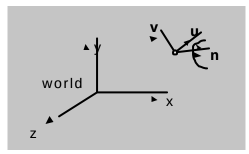
- Transformation order: apply the transformation to the object in a reverse order - translation first, and then rotate

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



World to Eye Transformation

- Head tilt: Rotate your head by δ
- Just rotate the object about the eye space z axis - δ
- $M_{w2e} = \begin{vmatrix} \cos(-\delta) & -\sin(-\delta) & 0 & 0 \\ \sin(-\delta) & \cos(-\delta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} ux & uy & ux & 0 \\ vx & vy & vz & 0 \\ nx & ny & nz & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & -ex \\ 0 & 1 & 0 & -ey \\ 0 & 0 & 1 & -ez \\ 0 & 0 & 0 & 1 \end{vmatrix}$



Why $-\delta$?

When you rotate your head by δ, it is like rotate the object by $-\delta$