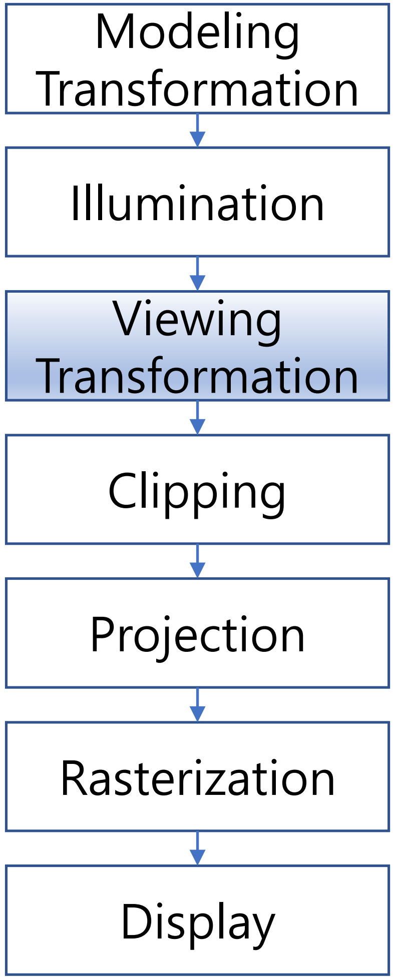
Lecture slides (CT4201/EC4215 – Computer Graphics)

Viewing Transformation

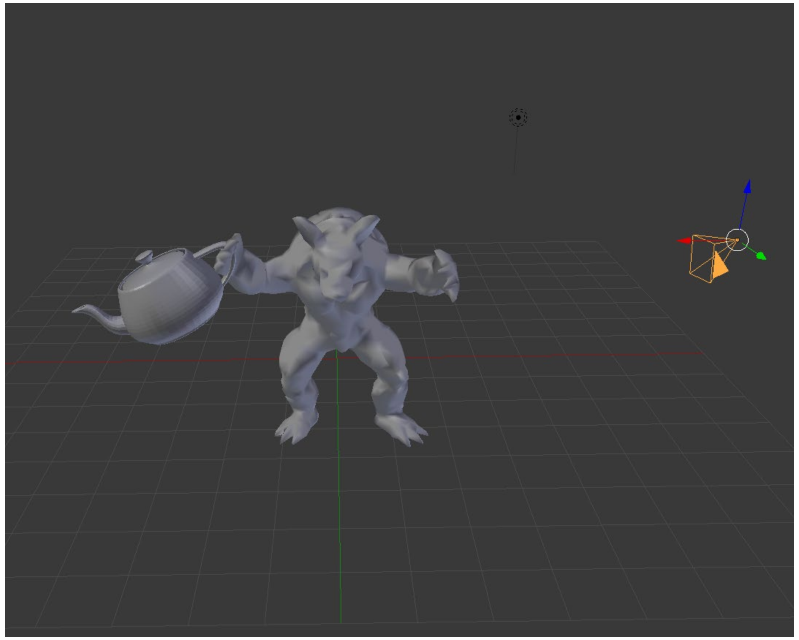
Lecturer: Bochang Moon

Computer Graphics
Laboratory

Viewing Transformation



- Transform all points from world space to *eye space*
  - Camera position transforms into the origin



Viewing Transformation

- Define camera position and its orientation

- Specify the following:
  - Location of the camera, $\boldsymbol{e} = (x_e, y_e, z_e)$
  - Direction where the camera is aiming at, vector $\boldsymbol{g} = (x_g, y_g, z_g)$
  - Upward direction of the camera, vector $\boldsymbol{t} = (x_t, y_t, z_t)$
    - Roughly orthogonal to $\boldsymbol{g}$ (not necessary)

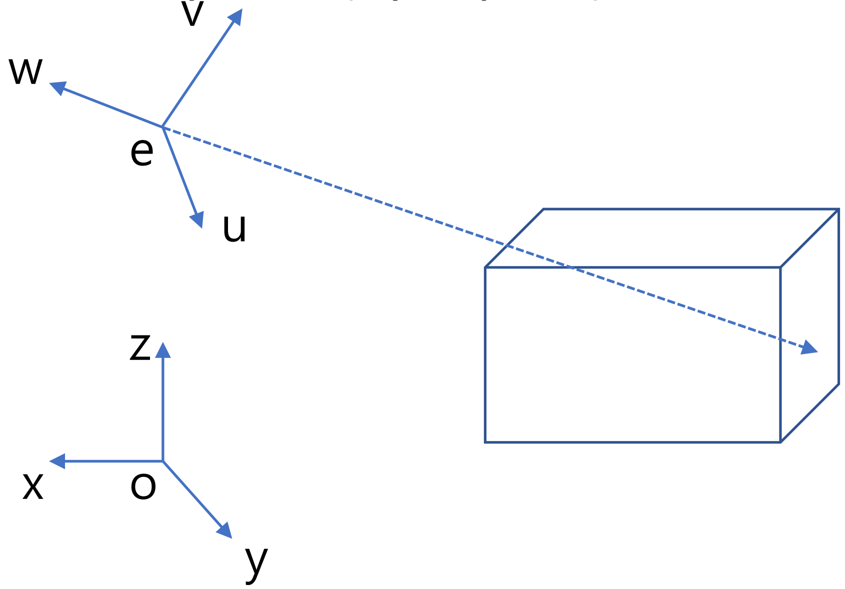
- A user specifies these variables.

- These variables are defined in *world space.*

Viewing Transformation

- Our task: transform all points defined in world space into new points in eye space

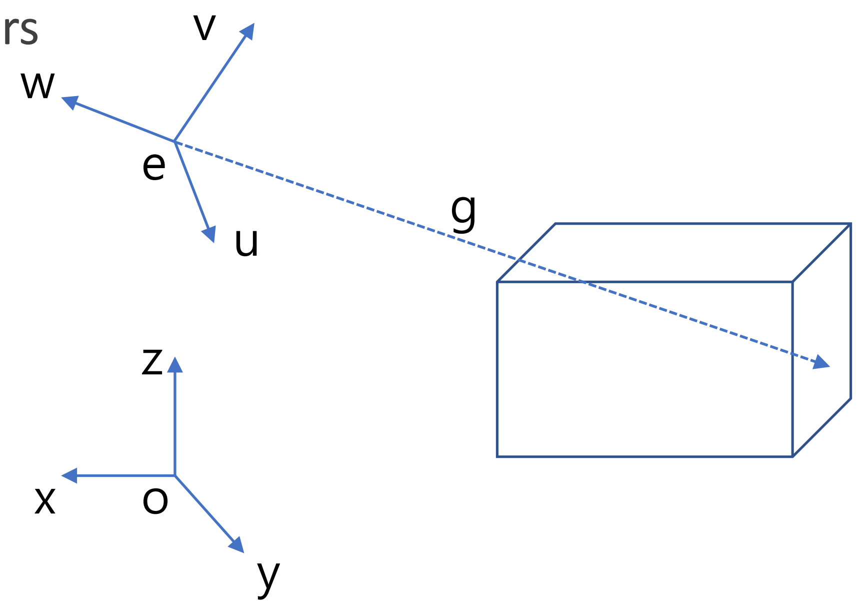
- Need to build a coordinate system (eye space) from the specifications of the camera



Viewing Transformation

- Our task: transform all points defined in world space into new points in eye space

- Need to build a coordinate system (eye space) from the specifications of the camera
  - Construct basis vectors from two input vectors
  - $\boldsymbol{w} = -\frac{\boldsymbol{g}}{\|\boldsymbol{g}\|}$
  - $\boldsymbol{u} = \frac{\boldsymbol{t}\times\boldsymbol{w}}{\|\boldsymbol{t}\times\boldsymbol{w}\|}$
  - $\boldsymbol{v} = \boldsymbol{w} \times \boldsymbol{u}$



Viewing Transformation in OpenGL

- void gluLookAt(GLdouble eyeX, GLdouble eyeY, GLdouble eyeZ,
- GLdouble centerX, GLdouble centerY, GLdouble centerZ,
- GLdouble upX, GLdouble upY, GLdouble upZ);
- Parameters
  - eyeX, eyeY, eyeZ
    - Specifies the position of the camera
  - centerX, centerY, centerZ
    - Specifies the position of the reference point that your camera is looking at
  - upX, upY, upZ
    - Specifies the direction of the up vector
- Issue: centerX, centerY, centerZ is not the gaze vector **g**. How can we compute this?

Viewing Transformation in OpenGL

- This can be considered as the following matrix transformations:
  - Step 1: translate the camera position **e** to the origin in world space
  - Step 2: rotate **u, v, w** to be aligned to **x, y, z**

- $\boldsymbol{M}_{\boldsymbol{view}} = \begin{bmatrix} \boldsymbol{u} & \boldsymbol{v} & \boldsymbol{w} & \boldsymbol{e} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}$

- $= \begin{bmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$ (why?)