Review: Transforming Normals

- cannot transform normals using same matrix as points
- nonuniform scaling would cause to be not perpendicular to desired plane!

\[ P \rightarrow P' = MP \]
\[ N \rightarrow N' = QN \]

Given \( M \), what should \( Q \) be?

\[ Q = (M^{-1})^T \]

Inv transpose of the modelling transformation

Using Transformations

- three ways
  - modelling transforms
  - place objects within scene (shared world)
  - affine transformations
  - viewing transforms
  - place camera
  - rigid body transformations: rotate, translate
  - projection transforms
  - change type of camera
  - projective transformation

OpenGL Transformation Storage

- modeling and viewing stored together
- possible because no intervening operations
- perspective stored in separate matrix

specify which matrix is target of operations
- common practice: return to default modelview mode after doing projection operations
  
  `glMatrixMode(GL_MODELVIEW);`  
  `glMatrixMode(GL_PROJECTION);`

Coordinate Systems

- result of a transformation
- names
  - convenience
    - armadillo: leg, head, tail
  - standard conventions in graphics pipeline
    - object/modeling
    - world
    - camera/viewing/eye
    - screen/window
    - raster/device
Convenient Camera Motion
- rotate/translate/scale not intuitive
- arbitrary viewing position
- eye point, gaze/lookat direction, up vector

Deriving W2V Transformation
- translate eye to origin
- rotate view vector (lookat – eye) to w axis
- rotate around w to bring up into vw-plane

From World to View Coordinates: W2V
- translate eye to origin
- rotate view vector (lookat – eye) to w axis
- rotate around w to bring up into vw-plane

OpenGL Viewing Transformation
- gluLookAt(ex,ey,ez,lx,ly,lz,ux,uy,uz)
  - postmultiplies current matrix, so to be safe:
  - glMatrixMode(GL_MODELVIEW);
  - glLoadIdentity();
  - gluLookAt(ex,ey,ez,lx,ly,lz,ux,uy,uz)
  - // now ok to do model transformations
  - demo: Nate Robins tutorial projection

Moving the Camera or the World?
- two equivalent operations
- move camera one way vs. move world other way
- example
  - initial OpenGL camera: at origin, looking along -z axis
  - create a unit square parallel to camera at z = -10
  - translate in z by 3 possible in two ways
  - camera moves to z = -3
  - camera stays put, but world moves to -7
  - resulting image same either way
  - possible difference: are lights specified in world or view coordinates?

World vs. Camera Coordinates
- a = (1,1)\textsubscript{w}
- b = (1,1)\textsubscript{C1} = (5,3)\textsubscript{w}
- c = (1,1)\textsubscript{C2} = (1,3)\textsubscript{C1} = (5,5)\textsubscript{w}

Projective Rendering Pipeline
- \text{object}\textsubscript{O2W}
- \text{world}\textsubscript{W2V}
- \text{viewing}\textsubscript{V2C}

Viewing Transformation
- translate eye to origin
- rotate view vector (lookat – eye) to w axis
- rotate around w to bring up into vw-plane

Basic Viewing
- starting spot - OpenGL
  - camera at world origin
    - probably inside an object
    - y axis is up
    - looking down negative z axis
  - why? RHS with x horizontal, y vertical, z out of screen
  - translate backward so scene is visible
  - move distance d = focal length
  - can use rotate/translate/scale to move camera
    - demo: Nate Robins tutorial transformations

Convenient Camera Motion
- rotate/translate/scale not intuitive
- arbitrary viewing position
- eye point, gaze/lookat direction, up vector

OpenGL Viewing Transformation
- gluLookAt(ex,ey,ez,lx,ly,lz,ux,uy,uz)