Reading for This and Next 2 Lectures
- FCG Chapter 7 Viewing
- FCG Section 6.3.1 Windowing Transforms
- RB rest of Chap Viewing
- RB rest of App Homogeneous Coords

Review: Display Lists
- precompile/cache block of OpenGL code for reuse
  - usually more efficient than immediate mode
  - exact optimizations depend on driver
- good for multiple instances of same object
  - but cannot change contents, not parameterizable
- good for static objects redrawn often
- display lists persist across multiple frames
- interactive graphics: objects redrawn every frame from new viewpoint from moving camera
- can be nested hierarchically
- snowman example: 3x performance improvement, 36K polys

Review: Computing Normals
- normal
  - direction specifying orientation of polygon
  - w=0 means direction with homogeneous coords
  - vs. w=1 for points/vectors of object vertices
  - used for lighting
  - must be normalized to unit length
  - can compute if not supplied with object

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

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

Rendering Pipeline
- result
  - all vertices of scene in shared 3D world coordinate system

Rendering Pipeline
- result
  - scene vertices in 3D view (camera) coordinate system

Rendering Pipeline
- result
  - 2D screen coordinates of clipped vertices

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
    - mouse: leg, head, tail
  - standard conventions in graphics pipeline
    - object/modeling
    - world
    - camera/viewing/eye
    - screen/window
    - raster/device
### Projective Rendering Pipeline
- **object** O2W world W2V viewing V2C
- **OCS** - object/model coordinate system
- **WCS** - world coordinate system
- **VCS** - viewing/camera/eye coordinate system
- **DCS** - display coordinate system
- **NDCS** - normalized device coordinate system
- **viewport** clipping

### Viewing Transformation
- **object** OCS
- **world** WCS
- **viewing** VCS
- **V2C** - viewport coordinate system

### Basic Viewing
- **starting spot** - OpenGL
  - camera at world origin
  - 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
- **where is camera in P1 template code?**
  - 5 units back, looking down -z axis

### Convenient Camera Motion
- **rotate/translate/scale versus**
  - eye, gaze/lookat direction, up vector
- **demo** - Robins transformation, projection

### 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

### Deriving W2V Transformation
- **rotate** view vector (lookat – eye) to w axis
  - **w** - normalized opposite of view/gaze vector $g$
- **view** coordinate system

### 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

### Deriving W2V Transformation
- **rotate** from WCS xyz into uvw coordinate system with matrix $T$
  - has columns $u,v,w$
  - $u = \frac{1}{x \times w}$
  - $v = w \times u$
  - $w = -\frac{g}{g}$
- **viewport** clipping

### W2V vs. V2W
- **$M_{W2V} = TR$**
  - we derived position of camera in world
  - invert for world with respect to camera
  - $M_{W2V} = (M_{W2V})^{-1} R^{T'} T^{-1}$
  - inverse is transpose of orthonormal matrices

### 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 $z = -3$
  - Note OpenGL models viewing in left-hand coordinates
  - 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 Example
- $a = (1,1)_w$
- $b = (1,1)_{C1} = (5,3)_w$
- $c = (1,1)_{C2} = (1,3)_{C1} = (5,5)_w$