Field-of-View Formulation
- FOV in one direction + aspect ratio (w/h)
- determines FOV in other direction
- also set near, far (reasonably intuitive)

Perspective OpenGL
- glMatrixMode(GL_PROJECTION);
- glLoadIdentity();
- glFrustum(left, right, bot, top, near, far);
- or
- glPerspective(fovy, aspect, near, far);

Asymmetric Frusta
- our formulation allows asymmetry
- why bother?

Asymmetric Frusta
- our formulation allows asymmetry
- why bother? binocular stereo
- view vector not perpendicular to view plane

Demo: Frustum vs. FOV
- Nate Robins tutorial (take 2): projection
- frustum vs perspective

Projective Rendering Pipeline
- object
- world
- viewing
- projection
- glMatrixMode(GL_PROJECTION);
- glLoadIdentity();
- glFrustum(left, right, bot, top, near, far);
- or
- glPerspective(fovy, aspect, near, far);
Perspective Warp
- warp perspective view volume to orthogonal view volume
- render all scenes with orthographic projection
- aka perspective normalization

$xz=\alpha z=d$
$xz=0 z=d$

DCS - device/display/screen coordinate
VCS - viewing/camera/eye coordinate
OCS - object/model coordinate system
WCS - world coordinate system
VCS - viewing/camera/eye coordinate system
OCS - object/model coordinate system
WCS - world coordinate system

Separate Warp From Homogenization
- complete: shear, scale, projection-normalization
- earlier:

Perspective Divide Example
- specific example
- assume image plane at $z = -1$
- a point $[x,y,z,1]^T$ projects to $[-x/z,-y/z,-z/z,1]^T \equiv [x,y,z,-z]^T$ in VCS

Perspective Derivation
- similar for other 5 planes
- 6 planes, 6 unknowns

Projective Rendering Pipeline
- object O2W world W2V viewing V2C
- perspective normalization
- perspective divide
- orthographic rendering of warped objects in cube produces same image as perspective rendering of original frustum
NDC to Device Transformation
- map from NDC to pixel coordinates on display
- NDC range is x = -1...1, y = -1...1, z = ... 500
- typical display range: x = 0...500, y = 0...300
- maximum is size of actual screen
- z range max and default is (0, 1), use later for visibility
  gglViewport(0,0,w,h);
gglDepthRange(0,1); // depth = 1 by default

NDC to Device Transformation
- yet more (possibly confusing) conventions
  - OpenGL origin: lower left
  - most window systems origin: upper left
  - then must reflect in y
  - when interpreting mouse position, have to flip your y coordinates

Device vs. Screen Coordinates
- viewport/window location wrt actual display not available within OpenGL
  - usually don’t care
    - use relative information when handling mouse events, not absolute coordinates
    - could get actual display height/width, window offsets from OS
  - loose use of terms: device, display, window, screen...

Projective Rendering Pipeline
- viewing (4-space, W=1)
- object modeling transformation
- world viewing transformation
- projection transformation
- clipping
- normalized device
  - device (3-space, parallelepiped)
  - projection matrix
  - divide by w
  - scale & translate

Coordinate Systems
- object - OCS
- world - OW
- viewing - WC
- viewing - VCS
- projection - VC
- clipping - C2N
- normalized device - NDC
- device - DCS

Perspective Example
- tracks in VCS: left x=-1, right x=1
- view volume
- near = 1, far = 4

OpenGL Example
- object
- world
- viewing
- clipping
- projection
- transformations that are applied to object first are specified last

Reading for Next Time
- RB Chap Color
- FCG Sections 3.2-3.3
- FCG Chap 20 Color
- FCG Chap 21.2.2 Visual Perception (Color)