Coordinate Systems
- result of a transformation
- names
  - convenience
    - mouse: leg, head, tail
  - standard conventions in graphics pipeline
    - object/modelling
    - world
    - camera/viewing/eye
    - screen/window
    - raster/device

Projective Rendering Pipeline
- object
- world
- viewing
  - modeling transformation
  - viewing transformation
  - projection transformation
  - clipping
  - normalized device coordinate system
    - source: various coordinate systems
  - device coordinate system

Projections I
- Pinhole Camera
  - ingredients
    - box, film, hole punch
  - result
    - picture

Rendering Pipeline
- assignment 1 posted
- viewing and projection
- two pieces
  - viewing transform
    - where is the camera, what is it pointing at?
  - perspective transform
    - 3D to 2D flattened to image

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);
  ```

Viewing and Projection
- need to get from 3D world to 2D image
- projection: geometric abstraction
  - what eyes or cameras do
- two pieces
  - viewing transform
    - where is the camera, what is it pointing at?
  - perspective transform
    - 3D to 2D flattened to image
Pinhole Camera
- theoretical perfect pinhole
- light shining through tiny hole into dark space yields upside-down picture

Real Cameras
- pinhole camera has small aperture (lens opening)
- minimize blur
- problem: hard to get enough light to expose the film
- solution: lens
- permits larger apertures
- permits changing distance to film plane without actually moving it
- cost: limited depth of field where image is in focus

GRAPHICS CAMERAS
- computer graphics cameras: convenient equivalent

PERSPECTIVE PROJECTION
- desired result for a point \([x, y, z, 1]^T\) projected onto the view plane:
- through common center of projection (eye point)
- onto an image plane

Perspective Projection Matrix

**Simple Perspective Projection**
\[
\begin{pmatrix}
 x \\
 y \\
 z/d \\
 d
\end{pmatrix}
\]

**Projected Value**
\[
\begin{pmatrix}
 x' \\
 y' \\
 z'/d \\
 d
\end{pmatrix}
\]

**Is Homogenized Version Of**
\[
\begin{pmatrix}
 x \\
 y \\
 z \\
 z/d
\end{pmatrix}
\]

**Pinhole Camera**
- non-zero sized hole
- blur: rays hit multiple points on film plane

General Projection
- image plane need not be perpendicular to view plane

Perspective Transformations
- properties
- lines mapped to lines and triangles to triangles
- parallel lines do NOT remain parallel
- e.g. rails vanishing at infinity
- affine combinations are NOT preserved
- e.g. center of a line does not map to center of projected line (perspective foreshortening)

Perspective Projection
- our camera must model perspective
- project all geometry

Projective Transformations
- planar geometric projections
- planar: onto a plane
- geometric: using straight lines
- projections: 3D -> 2D
- aka projective mappings

Simple Perspective Projection Matrix

**Example**
\[
\begin{pmatrix}
 x \\
 y \\
 z/d \\
 d
\end{pmatrix}
\]

**Is Homogenized Version Of**
\[
\begin{pmatrix}
 x \\
 y \\
 z \\
 z/d
\end{pmatrix}
\]
**Perspective Projection**
- expressible with 4x4 homogeneous matrix
- use previously untouched bottom row
- perspective projection is irreversible
- many 3D points can be mapped to same (x, y, d) on the projection plane
- no way to retrieve the unique z values

**Moving COP to Infinity**
- as COP moves away, lines approach parallel
- when COP at infinity, orthographic view

**Perspective to Orthographic**
- transformation of space
- center of projection moves to infinity
- view volume transformed
- from frustum (truncated pyramid) to parallelepiped (box)

**View Volumes**
- specifies field-of-view, used for clipping
- restricts domain of z stored for visibility test

**Demo: Perspective and Ortho Volumes**
- Nate Robins tutorial (projection)

**Normalized Device Coordinates**
- convention
  - viewing frustum mapped to specific parallelepiped
  - normalized device coordinates (NDC)
  - same as clipping coords
  - only objects inside the parallelepiped get rendered
  - which parallelepiped?
    - depends on rendering system

**Orthographic Camera Projection**
- camera’s back plane parallel to lens
- infinite focal length
- no perspective convergence
- just throw away z values

**Understanding Z**
- why near and far plane?
  - near plane:
    - avoid singularity (division by zero, or very small numbers)
  - far plane:
    - store depth in fixed-point representation (integer), thus have to have fixed range of values (0…1)
    - avoid/reduce numerical precision artifacts for distant objects

**Canonical View Volumes**
- standardized viewing volume representation
- perspective
- orthogonal
- parallel

**Why Canonical View Volumes?**
- permits standardization
- clipping
- easier to determine if an arbitrary point is enclosed in volume with canonical view volume vs. clipping to six arbitrary planes
- rendering
  - projection and rasterization algorithms can be reused
Orthographic Derivation
- scale, translate, reflect for new coord sys
\[ y' = a \cdot y + b \]
y = top \rightarrow y' = 1
y = bot \rightarrow y' = -1
\[
a = \frac{2}{top - bot}
\]
\[
b = \frac{-2}{top + bot}
\]
same idea for right/left, far/near

Orthographic OpenGL
- scale, translate, reflect for new coord sys
\[
\begin{bmatrix}
2 & 0 & 0 & \text{right-left}

0 & 2 & 0 & \text{top-bot}

0 & 0 & -2 & \text{far-near}

0 & 0 & 0 & 1
\end{bmatrix}
\]

Perspective OpenGL
- scale, translate, reflect for new coord sys
\[
\begin{bmatrix}
2 & 0 & 0 & \text{right-left}

0 & 2 & 0 & \text{top-bot}

0 & 0 & -2 & \text{far-near}

0 & 0 & 0 & 1
\end{bmatrix}
\]

Asymmetric Frusta
- our formulation allows asymmetry
- why bother?

Simpler Formulation
- left, right, bottom, top, near, far
- nonintuitive
- often overkill
- look through window center
- symmetric frustum
- constraints

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

Demo: Frustum vs. FOV
- Brown applets: viewing techniques
- parallel/orthographic camera transformations

Perspective Rendering Pipeline
- warp perspective view volume to orthogonal view volume
- render all scenes with orthographic projection!
- aka perspective warp

Projection Normalization
- warp perspective view volume to orthogonal view volume
- render all scenes with orthographic projection!
- aka perspective warp
Perspective Normalization
- perspective viewing frustum transformed to cube
- orthographic rendering of cube produces same image as perspective rendering of original

Predistortion

Demos
- Tuebingen applets from Frank Hanisch
  - http://www.gris.uni-tuebingen.de/projects/grdev/doc/html/etc/AppletIndex.html#Transformationen

Projective Rendering Pipeline
OCS - object/model coordinate system
WCS - world coordinate system
VCS - viewing/camera/eye coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device/display/screen coordinate system

Separate Warp From Homogenization
- warp requires only standard matrix multiply
  - distort such that orthographic projection of distorted objects is desired perspective projection
    - w is changed
    - clip after warp, before divide
    - division by w: homogenization