University of British Columbia
CPSC 314 Computer Graphics
Sep-Dec 2009

Tamara Munzner
(guest lecturer)

Viewing/Projections

Week 5, Wed Oct 6

News

- assignment 1 posted

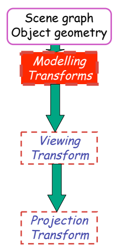
Viewing (Review?)

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

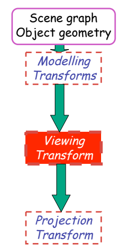
Rendering Pipeline

Rendering Pipeline



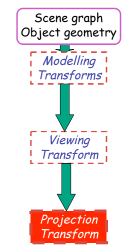
- result
  - all vertices of scene in shared 3D world coordinate system

Rendering Pipeline

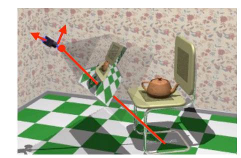


- result
  - scene vertices in 3D view (camera) coordinate system

Rendering Pipeline



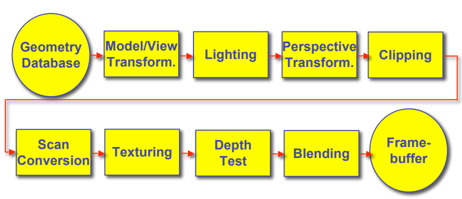
- result
  - 2D screen coordinates of clipped vertices



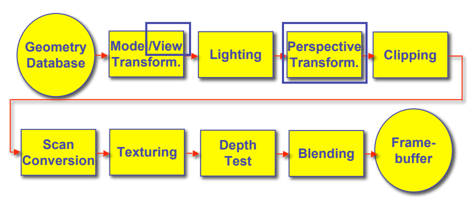
Viewing and Projection

- 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
    - flatten to image

Rendering Pipeline



Rendering Pipeline



OpenGL Transformation Storage

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

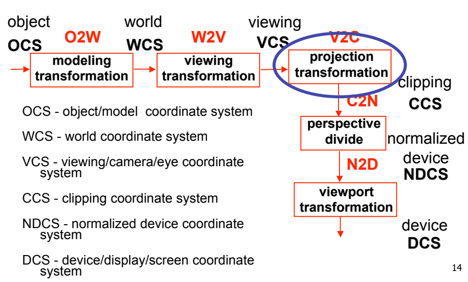
- specify which matrix is target of operations
  - common practice: return to default modelview mode after doing projection operations

```
glMatrixMode(GL_MODELVIEW);
glMatrixMode(GL_PROJECTION);
```


Coordinate Systems

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

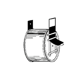
Projective Rendering Pipeline

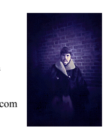


Projections I

Pinhole Camera

- ingredients
  - box, film, hole punch
- result
  - picture

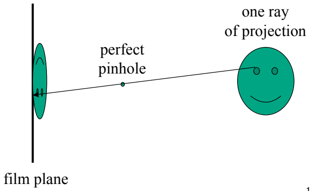

www.kodak.com


www.pinhole.org

www.debevec.org/Pinhole

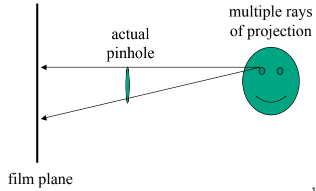

Pinhole Camera

- theoretical perfect pinhole
  - light shining through tiny hole into dark space yields upside-down picture



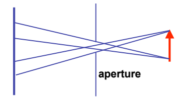
Pinhole Camera

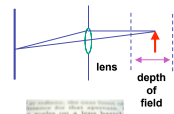
- non-zero sized hole
  - blur: rays hit multiple points on film plane

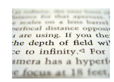


Real Cameras

- 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



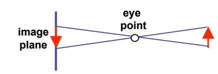




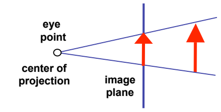
http://en.wikipedia.org/wiki/Image:DOF-ShallowDepthofField.jpg

Graphics Cameras

- real pinhole camera: image inverted

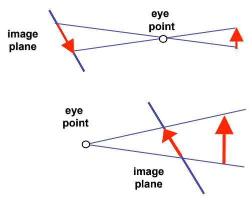


- computer graphics camera: convenient equivalent



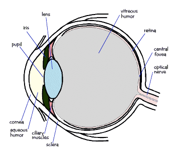
General Projection

- image plane need not be perpendicular to view plane



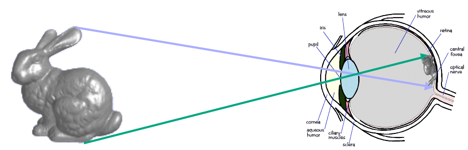
Perspective Projection

- our camera must model perspective



Perspective Projection

- our camera must model perspective



Projective Transformations

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

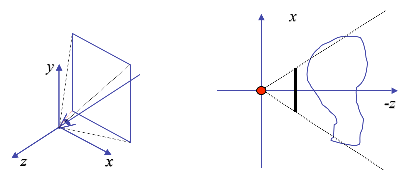
Projective Transformations

- properties
  - lines mapped to lines and triangles to triangles
  - parallel lines do NOT remain parallel
    - e.g. rails vanishing at infinity

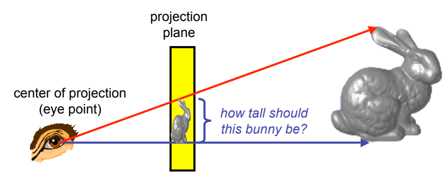
  - affine combinations are NOT preserved
    - e.g. center of a line does not map to center of projected line (perspective foreshortening)

Perspective Projection

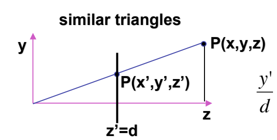
- project all geometry
  - through common center of projection (eye point)
  - onto an image plane



Perspective Projection



Basic Perspective Projection



$$\frac{y'}{d} = \frac{y}{z} \rightarrow y' = \frac{y \cdot d}{z}$$

$$\frac{x'}{d} = \frac{x}{z} \rightarrow x' = \frac{x \cdot d}{z} \qquad \text{but} \qquad z' = d$$

- nonuniform foreshortening
  - not affine

Perspective Projection

- desired result for a point $[x, y, z, 1]^T$ projected onto the view plane:

$$\frac{x'}{d} = \frac{x}{z}, \quad \frac{y'}{d} = \frac{y}{z}$$

$$x' = \frac{x \cdot d}{z} = \frac{x}{z/d}, \quad y' = \frac{y \cdot d}{z} = \frac{y}{z/d}, \quad z' = d$$

- what could a matrix look like to do this?

Simple Perspective Projection Matrix

$$\begin{bmatrix} \frac{x}{z/d} \\ \frac{y}{z/d} \\ d \end{bmatrix}$$

Simple Perspective Projection Matrix

$$\begin{bmatrix} \frac{x}{z/d} \\ \frac{y}{z/d} \\ d \end{bmatrix}$$ is homogenized version of $$\begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

where w = z/d

Simple Perspective Projection Matrix

$$\begin{bmatrix} \frac{x}{z/d} \\ \frac{y}{z/d} \\ d \end{bmatrix}$$ is homogenized version of $$\begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

where w = z/d

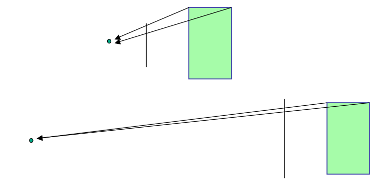
$$\begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

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



Orthographic Camera Projection

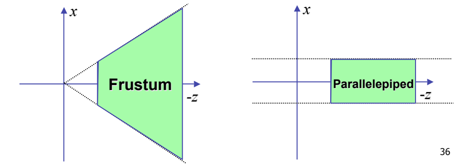
- camera's back plane parallel to lens
- infinite focal length
- no perspective convergence
- just throw away z values

$$\begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

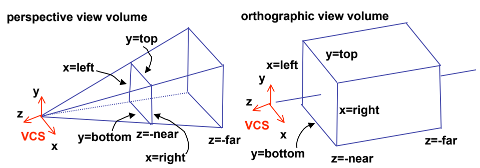
Perspective to Orthographic

- 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

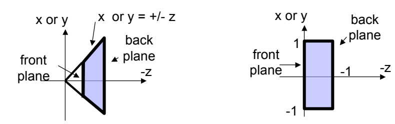
- Nate Robins tutorial (projection)
  - http://www.xmission.com/~nate/tutors.html

Canonical View Volumes

- standardized viewing volume representation

perspective

orthographic
orthogonal
parallel



Why Canonical View Volumes?

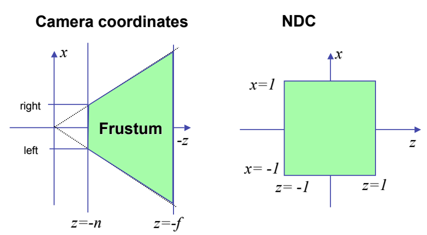
- permits standardization
  - clipping
    - easier to determine if an arbitrary point is enclosed in volume with canonical view volume vs. clipping to six arbitrary planes
  - rendering
    - projection and rasterization algorithms can be reused

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

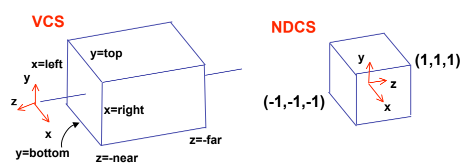
Normalized Device Coordinates

left/right x =+/- 1, top/bottom y =+/- 1, near/far z =+/- 1



Understanding Z

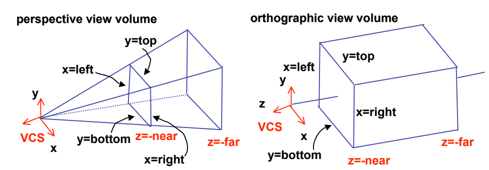
- z axis flip changes coord system handedness
  - RHS before projection (eye/view coords)
  - LHS after projection (clip, norm device coords)



Understanding Z

near, far always positive in OpenGL calls

glOrtho(left,right,bot,top,near,far);
glFrustum(left,right,bot,top,near,far);
glPerspective(fovy,aspect,near,far);

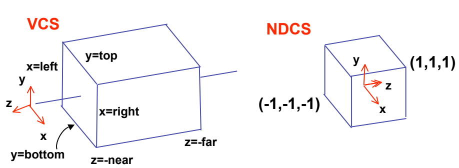


Understanding Z

- 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

Orthographic Derivation

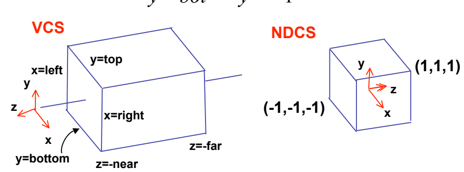
- scale, translate, reflect for new coord sys



Orthographic Derivation

- scale, translate, reflect for new coord sys

$$y' = a \cdot y + b \qquad \begin{matrix} y = top \to y' = 1 \\ y = bot \to y' = -1 \end{matrix}$$



Orthographic Derivation

- scale, translate, reflect for new coord sys

$$y' = a \cdot y + b \qquad \begin{matrix} y = top \to y' = 1 \\ y = bot \to y' = -1 \end{matrix} \qquad \begin{matrix} 1 = a \cdot top + b \\ -1 = a \cdot bot + b \end{matrix}$$

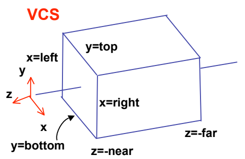
$$b = 1 - a \cdot top, b = -1 - a \cdot bot$$
$$1 - a \cdot top = -1 - a \cdot bot$$
$$1 - (-1) = -a \cdot bot - (-a \cdot top)$$
$$2 = a(-bot + top)$$
$$a = \frac{2}{top - bot}$$

$$1 = \frac{2}{top - bot} top + b$$
$$b = 1 - \frac{2 \cdot top}{top - bot}$$
$$b = \frac{(top - bot) - 2 \cdot top}{top - bot}$$
$$b = \frac{-top - bot}{top - bot}$$

Orthographic Derivation

- scale, translate, reflect for new coord sys

$$y' = a \cdot y + b \qquad y = top \rightarrow y' = 1 \qquad y = bot \rightarrow y' = -1$$



$$a = \frac{2}{top - bot}$$

$$b = -\frac{top + bot}{top - bot}$$

same idea for right/left, far/near

Orthographic Derivation

- scale, translate, reflect for new coord sys

$$P' = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bot} & 0 & -\frac{top+bot}{top-bot} \\ 0 & 0 & \frac{-2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix} P$$

Orthographic Derivation

- scale, translate, reflect for new coord sys

$$P' = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bot} & 0 & -\frac{top+bot}{top-bot} \\ 0 & 0 & \frac{-2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix} P$$

Orthographic Derivation

- scale, translate, reflect for new coord sys

$$P' = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bot} & 0 & -\frac{top+bot}{top-bot} \\ 0 & 0 & \frac{-2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix} P$$

Orthographic Derivation

- scale, translate, reflect for new coord sys

$$P' = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bot} & 0 & -\frac{top+bot}{top-bot} \\ 0 & 0 & \frac{-2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix} P$$

Orthographic OpenGL

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(left,right,bot,top,near,far);
```

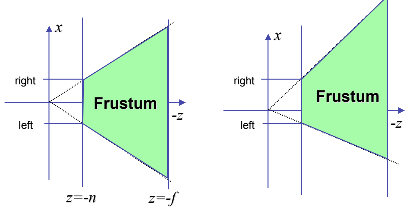

Demo

- Brown applets: viewing techniques
  - parallel/orthographic camera transformations
- http://www.cs.brown.edu/exploratories/freeSoftware/catalogs/viewing_techniques.html

Projections II

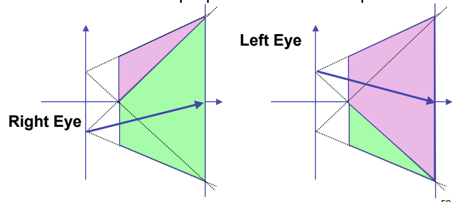
Asymmetric Frusta

- our formulation allows asymmetry
- why bother?



Asymmetric Frusta

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

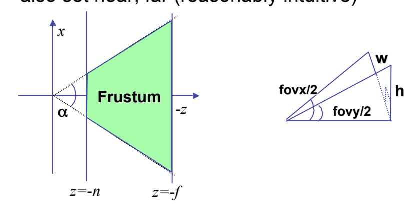


Simpler Formulation

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

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

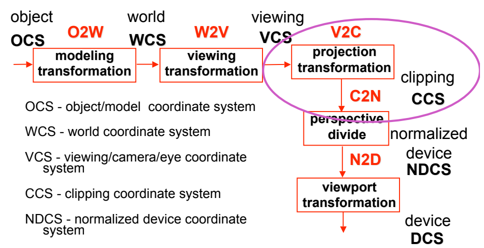
```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();

glFrustum(left,right,bot,top,near,far);
 or
glPerspective(fovy,aspect,near,far);
```


Demo: Frustum vs. FOV

- Nate Robins tutorial (take 2):
  - http://www.xmission.com/~nate/tutors.html

Projective Rendering Pipeline



OCS - object/model coordinate system

WCS - world coordinate system

VCS - viewing/camera/eye coordinate system

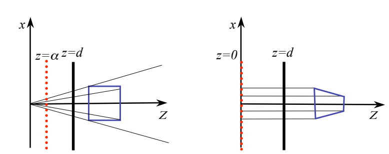
CCS - clipping coordinate system

NDCS - normalized device coordinate system

DCS - device/display/screen coordinate system

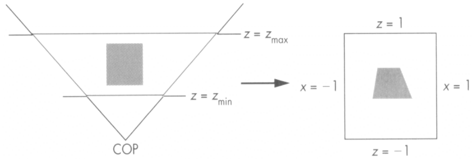
Projection Normalization

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

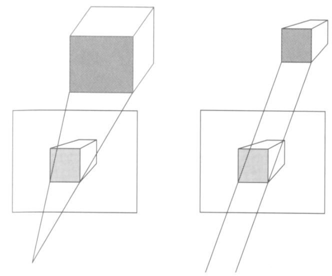


Perspective Normalization

- perspective viewing frustum transformed to cube
- orthographic rendering of cube produces same image as perspective rendering of original



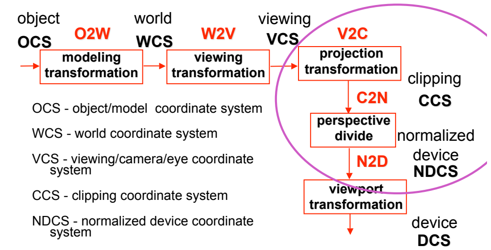
Predistortion



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

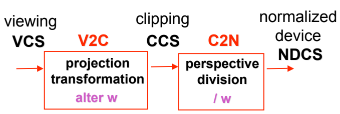
VCS - viewing/camera/eye coordinate system

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 persp projection
    - w is changed
  - clip after warp, before divide
  - division by w: homogenization