


## Perspective Projection (cont.) Transformations of Normal Vectors

Wolfgang Heidrich

Wolfgang Heidrich



## Course News

**Assignment 1**

- Due next Monday

**No new homework this week**


**Homework 2**

- Exercise problems for perspective
- Discussed in labs this week

**Quiz 1**

- Wed, Jan 26. Duration: 40 minutes
- Topics: affine and perspective transformations

Wolfgang Heidrich



## Course News (cont.)


**Reading list**

- Previously published chapters numbers were from an old book version...

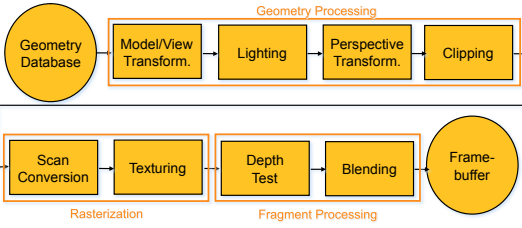
**Reading for Quiz (new book version):**

- Math prereq: Chapter 2.1-2.4, 4
- Intro: Chapter 1
- Affine transformations: Ch. 6 (was: Ch. 5, old book)
- Perspective: Ch 7 (was: Ch. 6, old book)
  - Also reading for this week...


Wolfgang Heidrich



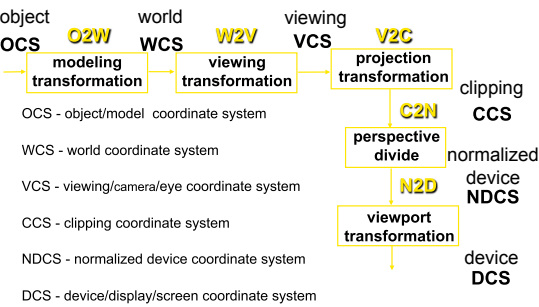
## The Rendering Pipeline



Wolfgang Heidrich




## 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

Wolfgang Heidrich



## Projective Transformations


**Convention:**

- Viewing frustum is mapped to a specific parallelepiped
  - *Normalized Device Coordinates (NDC)*
- Only objects inside the parallelepiped get rendered
- Which parallelepiped is used depends on the rendering system

**OpenGL:**

- Left and right image boundary are mapped to  $x=-1$  and  $x=+1$
- Top and bottom are mapped to  $y=-1$  and  $y=+1$
- Near and far plane are mapped to  $-1$  and  $1$

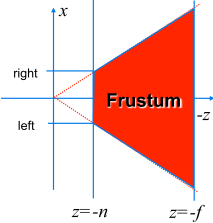
Wolfgang Heidrich



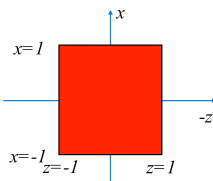
## Projective Transformations

### OpenGL Convention


Camera coordinates



Clipping Coordinates



Wolfgang Heidrich



## Perspective Derivation

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} E & 0 & A & 0 \\ 0 & F & B & 0 \\ 0 & 0 & C & D \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{aligned} x' &= Ex + Az \\ y' &= Fy + Bz \\ z' &= Cz + D \\ w' &= -z \end{aligned}$$


$$\begin{aligned} x = \text{left} &\rightarrow x'/w' = 1 \\ x = \text{right} &\rightarrow x'/w' = -1 \\ y = \text{top} &\rightarrow y'/w' = 1 \\ y = \text{bottom} &\rightarrow y'/w' = -1 \\ z = -\text{near} &\rightarrow z'/w' = 1 \\ z = -\text{far} &\rightarrow z'/w' = -1 \end{aligned}$$

$$y' = Fy + Bz, \quad \frac{y'}{w'} = \frac{Fy + Bz}{-z}, \quad 1 = \frac{Fy + Bz}{-z}, \quad 1 = \frac{Fy + Bz}{-z}$$

$$1 = F \frac{y}{-z} + B \frac{z}{-z}, \quad 1 = F \frac{y}{-z} - B, \quad 1 = F \frac{\text{top}}{-(-\text{near})} - B,$$

$$1 = F \frac{\text{top}}{\text{near}} - B$$

Wolfgang Heidrich




## Perspective Derivation

*similarly for other 5 planes*  
**6 planes, 6 unknowns**

$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Wolfgang Heidrich




## Perspective Example

**view volume**  
left = -1, right = 1  
bot = -1, top = 1  
near = 1, far = 4

$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

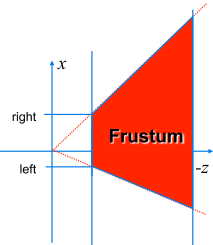
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -5/3 & -8/3 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Wolfgang Heidrich




## Projective Transformations

### Asymmetric Viewing Frusta



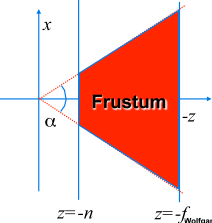
Wolfgang Heidrich



## Projective Transformations

### Alternative specification of symmetric frusta

- Field-of-view (fov)  $\alpha$
- Fov/2
- Field-of-view in y-direction (fovy) + aspect ratio



Wolfgang Heidrich

**Perspective Matrices in OpenGL**

**Perspective Matrices:**

- glFrustum( left, right, bottom, top, near, far )
  - Specifies perspective transform (near, far are always positive)

**Convenience Function:**

- gluPerspective( fovy, aspect, near, far )
  - Another way to do perspective

Wolfgang Heidrich

**Projective Transformations**

**Properties:**

- All transformations that can be expressed as homogeneous 4x4 matrices (in 3D)
- 16 matrix entries, but multiples of the same matrix all describe the same transformation
  - 15 degrees of freedom
  - The mapping of 5 points uniquely determines the transformation

Wolfgang Heidrich

**Projective Transformations**

**Properties**

- Lines are 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)
  - The center of a line segment does **not**, in general map to the center of the transformed line segment
    - Same for other points in triangles

Wolfgang Heidrich

**Orthographic Camera Projection**

- Camera's back plane parallel to lens
- Infinite focal length
- No perspective convergence
- Just throw away z values

OpenGL:

- glOrtho
- gluOrtho2D

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

Wolfgang Heidrich

**Projection Taxonomy**

http://ceprofs.tamu.edu/kramer/ENGR%20111/5.1/20

Wolfgang Heidrich

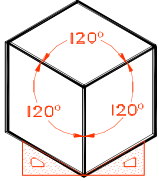
**Perspective Projections classified by vanishing points**

Wolfgang Heidrich

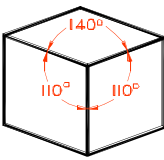
## Axonometric Projections

- projectors perpendicular to image plane

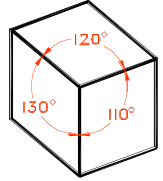
<p>3 Equal axes 3 Equal angles</p>	<p>2 Equal axes 2 Equal angles</p>	<p>0 Equal axes 0 Equal angles</p>
--	--	--



A. ISOMETRIC



B. DIMETRIC



C. TRIMETRIC

http://ceprofs.tamu.edu/tkramer/ENGR%20111/5.1/20 Wolfgang Heidrich

## Projective Rendering Pipeline

object OCS  $O2W$  world WCS  $W2V$  viewing VCS  $V2C$

modeling transformation → viewing transformation → projection transformation

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

clipping CCS

perspective divide  $C2N$  normalized device NDCS

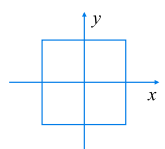
viewport transformation  $N2D$  device DCS

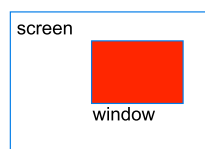
Wolfgang Heidrich

## Window-To-Viewport Transformation

**Generate pixel coordinates**

- Map  $x, y$  from range  $-1 \dots 1$  (normalized device coordinates) to pixel coordinates on the screen
- Map  $z$  from  $-1 \dots 1$  to  $0 \dots 1$  (used later for visibility)
- Involves 2D scaling and translation





Wolfgang Heidrich

## Homogeneous Planes & Normals

Wolfgang Heidrich

## Normals & Affine Transformations

**Question:**

- If we transform some geometry with an affine transformation, how does that affect the normal vector?

**Consider**

- Rotation
- Translation
- Scaling
- Shear

Wolfgang Heidrich

## Normals & Affine Transformations

**Want:**

- Representation for normals that allows us to easily describe how they change under affine transformation

**Why?**

- Normal vectors will be of special interest when we talk about lighting (next week)

Wolfgang Heidrich

**Homogeneous Planes And Normals**

**Planes in Cartesian Coordinates:**

$$\{(x, y, z)^T \mid n_x x + n_y y + n_z z + d = 0\}$$

- $n_x, n_y, n_z$  and  $d$  are the parameters of the plane (normal and distance from origin)

**Planes in Homogeneous Coordinates:**

$$\{(x, y, z, w)^T \mid n_x x + n_y y + n_z z + dw = 0\}$$

Wolfgang Heidrich

**Homogeneous Planes And Normals**

**Planes in homogeneous coordinates are represented as row vectors**

- $E = [n_x, n_y, n_z, d]$
- Condition that a point  $[x, y, z, w]^T$  is located in E

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \in E = [n_x, n_y, n_z, d] \Leftrightarrow [n_x, n_y, n_z, d] \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = 0$$

Wolfgang Heidrich

**Homogeneous Planes And Normals**

**Transformations of planes**

$$[n_x, n_y, n_z, d] \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = 0 \Leftrightarrow T([n_x, n_y, n_z, d]) \cdot (\mathbf{A} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}) = 0$$

Wolfgang Heidrich

**Homogeneous Planes And Normals**

**Transformations of planes**

$$[n_x, n_y, n_z, d] \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = 0 \Leftrightarrow ([n_x, n_y, n_z, d] \cdot \mathbf{A}^{-1}) \cdot (\mathbf{A} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}) = 0$$

- Works for  $T([n_x, n_y, n_z, d]) = [n_x, n_y, n_z, d] \mathbf{A}^{-1}$
- Thus: planes have to be transformed by the *inverse* of the affine transformation (multiplied from left as a row vector)!

Wolfgang Heidrich

**Homogeneous Planes And Normals**

**Homogeneous Normals**

- The plane definition also contains its normal
- Normal written as a vector  $[n_x, n_y, n_z, 0]^T$

$$\begin{pmatrix} n_x \\ n_y \\ n_z \\ 0 \end{pmatrix} \cdot \begin{bmatrix} v_x \\ v_y \\ v_z \\ 0 \end{bmatrix} = 0 \Leftrightarrow ((\mathbf{A}^{-T} \cdot \begin{bmatrix} n_x \\ n_y \\ n_z \\ 0 \end{bmatrix}) \cdot (\mathbf{A} \cdot \begin{bmatrix} v_x \\ v_y \\ v_z \\ 0 \end{bmatrix})) = 0$$

- Thus: the normal to any surface has to be transformed by the inverse transpose of the affine transformation (multiplied from the right as a column vector)!

Wolfgang Heidrich

**Transforming Homogeneous Normals**

**Inverse Transpose of**

- Rotation by  $\alpha$ 
  - Rotation by  $\alpha$
- Scale by  $s$ 
  - Scale by  $1/s$
- Translation by  $t$ 
  - Identity matrix!
- Shear by  $a$  along x axis
  - Shear by  $-a$  along y axis

Wolfgang Heidrich



## Coming Up:

### **Wednesday:**

- Quiz...!

### **Friday**

- Lighting/shading

Wolfgang Heidrich