Perspective Projection

Wolfgang Heidrich

Course News

Assignment 1
- Due January 31

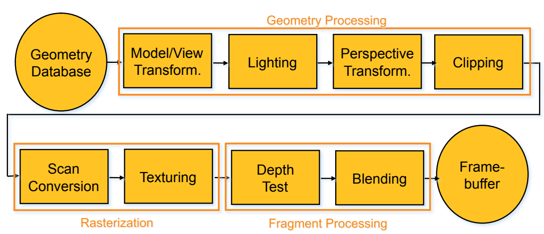
Homework 2
- Exercise problems for perspective
- Discussed in labs next week

Quiz 1
- One week from today (Wed, Jan 26)

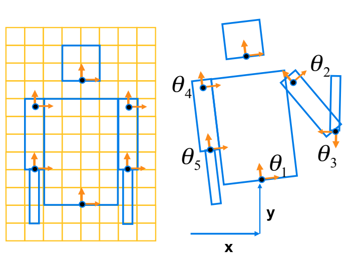
Course News (cont.)

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

The Rendering Pipeline



Recap: Transformation Hierarchies




```
glTranslate3f(x,y,0);
glRotatef(θ1,0,0,1);
DrawBody();
glPushMatrix();
  glTranslate3f(0,7,0);
  DrawHead();
glPopMatrix();
glPushMatrix();
  glTranslate(2.5,5.5,0);
  glRotatef(θ2,0,0,1);
  DrawUArm();
  glTranslate(0,-3.5,0);
  glRotatef(θ3,0,0,1);
  DrawLArm();
glPopMatrix();
... (draw other arm)
```


Hierarchical Modeling

Advantages
- Define object once, instantiate multiple copies
- Transformation parameters often good control knobs
- Maintain structural constraints if well-designed

Limitations
- Expressivity: not always the best controls
- Can't do closed kinematic chains
  - *Keep hand on hip*

Display Lists

Concept:

- If multiple copies of an object are required, it can be compiled into a display list:

```
glNewList( listId, GL_COMPILE );
  glBegin( …);
  … // geometry goes here
glEndList();
// render two copies of geometry offset by 1 in z-direction:
glCallList( listId );
glTranslatef( 0.0, 0.0, 1.0 );
glCallList( listId) ;
```


Display Lists

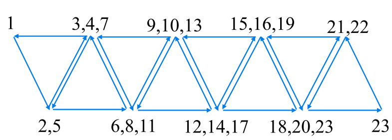
Advantages:

- More efficient than individual function calls for every vertex/attribute
- Can be cached on the graphics board (bandwidth!)
- Display lists exist across multiple frames
  - *Represent static objects in an interactive application*

Shared Vertices

Triangle Meshes

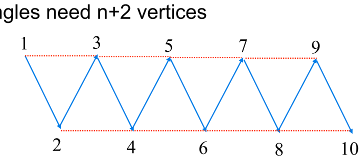
- Multiple triangles share vertices
- If individual triangles are sent to graphics board, every vertex is sent and transformed multiple times!
  - *Computational expense*
  - *Bandwidth*



Triangle Strips

Idea:

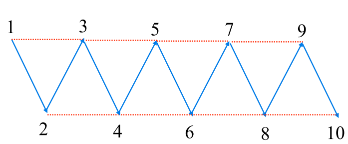
- Encode neighboring triangles that share vertices
- Use an encoding that requires only a constant-sized part of the whole geometry to determine a single triangle
- N triangles need n+2 vertices



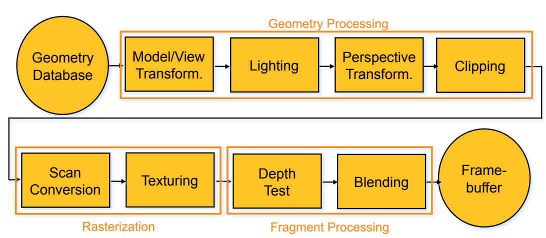
Triangle Strips

Orientation:

- Strip starts with a counter-clockwise triangle
- Then alternates between clockwise and counter-clockwise



The Rendering Pipeline



Projective Rendering Pipeline

object OCS → O2W modeling transformation → world WCS → W2V viewing transformation → viewing VCS → V2C projection transformation → clipping CCS → C2N perspective divide → normalized device NDCS → N2D viewport transformation → device DCS

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

Rendering Pipeline

Scene graph Object geometry → Modelling Transforms → Viewing Transform → Projection Transform

Wolfgang Heidrich

Rendering Pipeline

Scene graph Object geometry → **Modelling Transforms** → Viewing Transform → Projection Transform

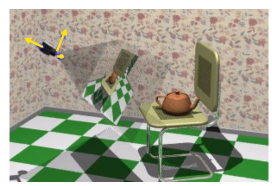
- result
  - all vertices of scene in shared 3D world coordinate system

Wolfgang Heidrich

Rendering Pipeline

Scene graph Object geometry → Modelling Transforms → **Viewing Transform** → Projection Transform

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

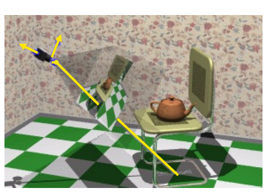


Wolfgang Heidrich

Rendering Pipeline

Scene graph Object geometry → Modelling Transforms → Viewing Transform → **Projection Transform**

- result
  - 2D screen coordinates of clipped vertices

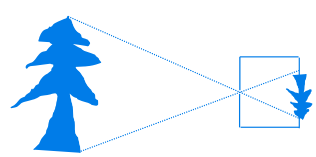


Wolfgang Heidrich

Perspective Transformation

Pinhole Camera:

- Light shining through a tiny hole into a dark room yields upside-down image on wall



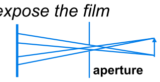
Wolfgang Heidrich

Perspective Transformation

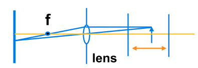
Pinhole Camera

Real Cameras

- pinhole camera has small aperture (lens opening)
  - *hard to get enough light to expose the film*



- lens permits larger apertures
- lens permits changing distance to film plane without actually moving the film plane

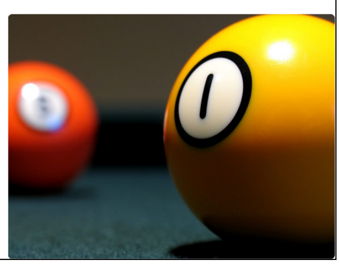


price to pay: limited depth of field

Real Cameras - Depth of Field

Limited depth of field

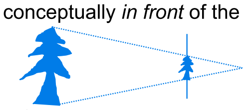
- Can be used to direct attention
- Artistic purposes



Perspective Transformation

In computer graphics:

- Image plane is conceptually *in front* of the center of projection

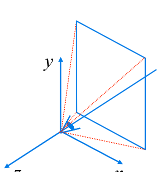


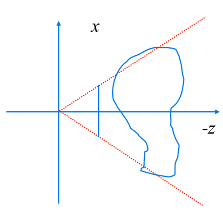
- Perspective transformations belong to a class of operations that are called *projective transformations*
- Linear and affine transformations also belong to this class
- *All* projective transformations can be expressed as *4x4* matrix operations

Perspective Projection

Synopsis:

- Project all geometry through a common <u>*center of projection*</u> (<u>*eye point*</u>) onto an <u>*image plane*</u>

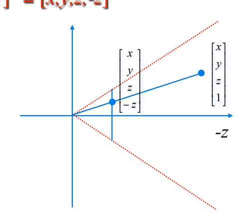




Perspective Projection

Example:

- Assume image plane at $z=-1$
- A point $[x,y,z,1]^T$ projects to $[-x/z,-y/z,-z/z,1]^T \equiv [x,y,z,-z]^T$



Perspective Projection

Analysis:

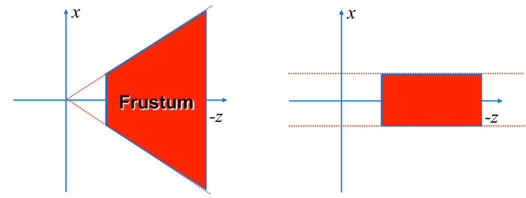
- This is a special case of a general family of transformations called *projective transformations*
- These can be expressed as 4x4 homogeneous matrices!
  - *E.g. in the example:*

$$T\left(\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ -z \end{bmatrix} \equiv \begin{bmatrix} -x/z \\ -y/z \\ -1 \\ 1 \end{bmatrix}$$

Projective Transformations

Transformation of space:

- Center of projection moves to infinity
- Viewing frustum is transformed into a parallelpiped



Projective Transformations

Convention:

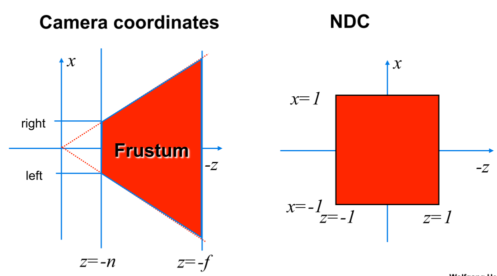
- Viewing frustum is mapped to a specific parallelpiped
  - *Normalized Device Coordinates (NDC)*
- Only objects inside the parallelpiped get rendered
- Which parallelpied is used depends on the rendering system

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

Projective Transformations

OpenGL Convention



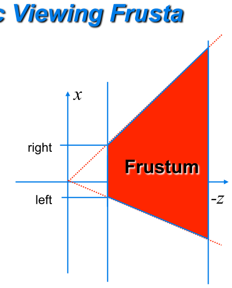
Projective Transformations

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*

Projective Transformations

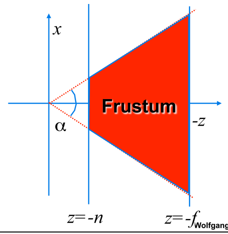
Asymmetric Viewing Frusta



Projective Transformations

Alternative specification of symmetric frusta

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



Demos

Tuebingen applets from Frank Hanisch

- http://www.gris.uni-tuebingen.de/edu/projects/grdev/doc/html/etc/AppletIndex_en.html#Transform

Coming Up:

Wednesday:

- More on perspective projection

Friday/Next Week

- Lighting/shading