3D Viewing Pipeline

Instructor – Stephen J. Guy
Overview

- Review
  - Linear Transformations & Homogeneous coordinates
  - Scene Graph
- 3D Polygon rendering
  - Graphics Pipeline
  - Viewing Transformation
Overview

- Review
  - Linear Transformations & Homogeneous coordinates
  - Scene Graph
- 3D Polygon rendering
  - Graphics Pipeline
  - Viewing Transformation
Reviewing Transforms

- Linear Transformations
  - Scale
  - Rotate
  - Shear
  - Mirror

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  a & b \\
  c & d
\end{bmatrix}\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

\[
x' = ax + by \\
y' = cx + dy
\]
Review – Homogeneous Coordinates

- Use 3 coordinates for a 2D point (4 coord. for 3D point)
  - \((x,y,w)\) represents a point at \((x/w, y/w)\)
  - \((x,y,0)\) represents a point at infinity
  - \((0,0,0)\) … not allowed!

\[(2,1,1)\] or \[(4,2,2,\) or \[(6,3,3)\] …
Review – Translating and More

- Basic 2D Transformations as 3x3

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} = \begin{bmatrix}
  1 & 0 & tx \\
  0 & 1 & ty \\
  0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Translate

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} = \begin{bmatrix}
  \cos \Theta & -\sin \Theta & 0 \\
  \sin \Theta & \cos \Theta & 0 \\
  0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Rotate

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} = \begin{bmatrix}
  sx & 0 & 0 \\
  0 & sy & 0 \\
  0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Scale

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} = \begin{bmatrix}
  1 & shx & 0 \\
  shy & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Shear
Review – 3D Transforms

- Homogenous coordinates \((x, y, z, w)\)
- 4x4 transformation matrix

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} = \begin{bmatrix}
a & b & c & d \\
e & f & g & h \\
i & j & k & l \\
m & n & o & p
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]
Review – Basic 3D Transforms

Identity

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]

Scale

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} = \begin{bmatrix}
sx & 0 & 0 & 0 \\
0 & sy & 0 & 0 \\
0 & 0 & sz & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]

Translation

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 & tx \\
0 & 1 & 0 & ty \\
0 & 0 & 1 & tz \\
0 & 0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]

Mirror over X axis

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} = \begin{bmatrix}
-1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]
Review – 3D Rotations

Rotate around Z axis:

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} =
\begin{bmatrix}
\cos \Theta & -\sin \Theta & 0 & 0 \\
\sin \Theta & \cos \Theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]

Rotate around Y axis:

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} =
\begin{bmatrix}
\cos \Theta & 0 & \sin \Theta & 0 \\
0 & 1 & 0 & 0 \\
-\sin \Theta & 0 & \cos \Theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]

Rotate around X axis:

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \Theta & -\sin \Theta & 0 \\
0 & \sin \Theta & \cos \Theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]
Review – Transformation Hierarchies

- Successively applied matrixes
- Stored in a DAG == Scene Graph

Examples
- Robot Arm
- Articulated Characters
- Complex Scenes
Transform Example - Robot Arms

Base \([M_1]\)

Upper Arm \([M_2]\)

Lower Arm \([M_3]\)

Robot Arm

Angel Figures 8.8 & 8.9
Example – Humanoid Character

Root

- Chest
  - Neck
  - LCollar
    - Head
    - LShld
    - LElbow
    - LWrist
  - LCollar
  - LHip
  - LKnee
  - LANkle
- RHip
  - RKnee
  - RAnkle

Rose et al. '96
Example – Complex Scenes

What does the Scene graph look like?

Draw same 3D data with different transformations
Example – Complex Scene Graph
Ray Casting with Hierarchies

• Transform rays, not primitives
  ○ For each node ...
    » Transform ray by inverse of matrix
    » Intersect with primitives
    » Transform hit by matrix

Robot Arm

Angel Figures 8.8 & 8.9
Overview

- Review
  - Linear Transformations & Homogeneous coordinates
  - Scene Graph
- 3D Polygon rendering
  - Graphics Pipeline
  - Viewing Transformation
3D Polygon Rendering

- Many applications make use of 3D polygons with direct illumination
3D Polygon Rendering - Examples

Quake II
(Id Software)
3D Polygon Rendering - Examples

Architectural Walkthrough
Ray Casting Revisiting

- For each sample
  - Construct ray through pixel
  - Find first surface intersected by ray through pixel
  - Compute color of sample based on surface radiance

More Efficient Algorithms
Utilize Spatial Coherence!
3D Polygon Rendering

- What steps are necessary to utilize spatial coherence while drawing polygons on a 2D image?
3D Rendering Pipeline (direct illumination only)

A pipelined sequence of operations to draw a 3D primitive onto a 2D image.
OpenGL Example

```
glBegin(GL_POLYGON);
 glVertex3f(0.0, 0.0, 0.0);
 glVertex3f(1.0, 0.0, 0.0);
 glVertex3f(1.0, 1.0, 1.0);
 glVertex3f(0.0, 1.0, 1.0);
 glEnd();
```

OpenGL runs the 3D rendering pipeline for each polygon
3D Rendering Pipeline (direct illumination only)

3D Geometric Primitives

- Modeling Transformation

- Lighting

- Viewing Transformation

- Projection Transformation

- Clipping

- Scan Conversion

- Image

- Transform in 3D word coordinate system
3D Rendering Pipeline (direct illumination only)

- Transform in 3D word coordinate system
- Illuminate according to lighting and reflectance
3D Rendering Pipeline (direct illumination only)

- Transform in 3D word coordinate system
- Illuminate according to lighting and reflectance
- Transform into 3D camera coordinate system
3D Rendering Pipeline (direct illumination only)

3D Geometric Primitives

- Modeling Transformation
- Lighting
- Viewing Transformation
- **Projection Transformation**
- Clipping
- Scan Conversion

- Transform in 3D word coordinate system
- Illuminate according to lighting and reflectance
- Transform into 3D camera coordinate system
- Transform into 2D screen coordinate system
3D Rendering Pipeline  (direct illumination only)

- **Modeling Transformation**
- **Lighting**
- **Viewing Transformation**
- **Projection Transformation**
- **Clipping**
- **Scan Conversion**

- Transform in 3D word coordinate system
- Illuminate according to lighting and reflectance
- Transform into 3D camera coordinate system
- Transform into 2D screen coordinate system
- Clip primitives outside camera’s view

3D Geometric Primitives
3D Rendering Pipeline (direct illumination only)

3D Geometric Primitives

- Modeling Transformation
- Lighting
- Viewing Transformation
- Projection Transformation
- Clipping
- Scan Conversion

- Transform in 3D word coordinate system
- Illuminate according to lighting and reflectance
- Transform into 3D camera coordinate system
- Transform into 2D screen coordinate system
- Clip primitives outside camera’s view
- Draw Pixels (also texturing, hidden surface, …)
3D Rendering Pipeline (direct illumination only)

3D Geometric Primitives

- Modeling Transformation
- Lighting
- Viewing Transformation
- Projection Transformation
- Clipping
- Scan Conversion

- Transform in 3D word coordinate system
- Illuminate according to lighting and reflectance
- Transform into 3D camera coordinate system
- Transform into 2D screen coordinate system
- Clip primitives outside camera’s view
- Draw Pixels (also texturing, hidden surface, ...)

Image
3D Rendering Pipeline (direct illumination only)

3D Geometric Primitives

- **Modeling Transformation**
- **Lighting**
- **Viewing Transformation**
- **Projection Transformation**
- **Clipping**
- **Scan Conversion**

- **Transform** in 3D word coordinate system
- **Illuminate** according to lighting and reflectance
- **Transform** into 3D camera coordinate system
- **Transform** into 2D screen coordinate system
- Clip primitives outside camera’s view
- **Draw Pixels** (also texturing, hidden surface, ...)

10/6/2010
Transformations map points from one coordinate system to another.
Viewing Transformations

\[ p(x,y,z) \]

3D Object Coordinates

### Modeling Transformation

\[ 3D \text{ World Coordinates} \]

### Viewing Transformation

\[ 3D \text{ Camera Coordinates} \]

### Projection Transformation

\[ 2D \text{ Screen Coordinates} \]

### Window-to-Viewport Transformation

\[ 2D \text{ Image Coordinates} \]

\[ p'(x',y') \]
Camera Coordinates

- Canonical coordinate system
  - Convention is right-handed (looking down z-axis)
  - Convenient for projection, clipping, etc
Viewing Transformation

- Mapping from world to camera coordinates
  - Eye position maps to origin
  - Right vector maps to X-axis
  - Up vector maps to Y-axis
  - Back vector maps to Z-axis
Finding the Viewing Transformation

- We have the camera (in world coordinates)
- We want $T$ taking objects from world to camera
  \[ P^c = T \ast P^w \]
- Trick: Find $T^{-1}$ taking objects from camera to world
  \[ P^w = T \ast P^c \]

\[
\begin{bmatrix}
  x' \\
  y' \\
  z' \\
  w'
\end{bmatrix} =
\begin{bmatrix}
  a & b & c & d \\
  e & f & g & h \\
  i & j & k & l \\
  m & n & o & p
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  w
\end{bmatrix}
\]
Finding the Viewing Transformation

- **Trick:** map from camera coordinates to world
  - Origin maps to eye position
  - Z axis maps to Back vector
  - Y axis maps to Up vector
  - X axis maps to Right vector

\[
\begin{bmatrix}
    x' \\
    y' \\
    z' \\
    w'
\end{bmatrix} =
\begin{bmatrix}
    R_x & U_x & B_x & E_x \\
    R_y & U_y & B_y & E_y \\
    R_z & U_z & B_z & E_z \\
    R_w & U_w & B_w & E_w
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    z \\
    w
\end{bmatrix}
\]

- This matrix is \( T^{-1} \) so we invert it to get \( T \) ... easy!
Viewing Transformations

$p(x,y,z)$

Modeling Transformation

$3D$ Object Coordinates

$3D$ World Coordinates

Viewing Transformation

$3D$ Camera Coordinates

Projection Transformation

$2D$ Screen Coordinates

Window-to-Viewport Transformation

$2D$ Image Coordinates

$p'(x',y')$
Projection

- General definition:
  - Transform points in \( n \)-space to \( m \)-space (\( m < n \))
- In computer graphics
  - Map 3D camera coordinates to 2D screen coordinates
Taxonomy of Projections

Planar geometric projections

Parallel
- Orthographic
  - Top (plan)
  - Front elevation
  - Side elevation
- Axonometric
- Isometric

Oblique
- Cabinet
- Cavalier

Perspective
- One-point
- Two-point
- Three-point

Other
Taxonomy of Projections

Planar geometric projections

Parallel

Orthographic
- Top (plan)
- Front elevation
- Side elevation

Axonometric
- Isometric

Oblique
- Cabinet
- Cavalier

Perspective
- One-point
- Two-point
- Three-point

Other

FVFHP Figure 6.10
Parallel Projection

- Center of projection is at infinity
  - Direction of Projection (DOP) is same for all points

[Diagram of Parallel Projection]
Orthographic Projection

- DOP Parallel to View Plane

![Orthographic views of a building]

Front
Top
Side

Angel Figure 5.5
Oblique Projection

- DOP not perpendicular to view plane

Cavalier
(DOP $\alpha = 45^\circ$)

Cabinet
(DOP $\alpha = 63.4^\circ$)

H&B Figure 12.24
Oblique Projection

- Examples

Oblique Cabinet

Cavalier Fortification Diagrams
Orthographic Perspective in Games

- Common in pre-hardware card Graphics
Parallel Projection View Volume
Orthographic Project Matrix

- Straight on projection... throw away Z
- What does the matrix look like?
Parallel Projection Matrix

- General parallel projection transformation:

\[
\begin{bmatrix}
    x_s \\ y_s \\ z_s \\ w_s
\end{bmatrix} =
\begin{bmatrix}
    1 & 0 & L_1 \cos \phi & 0 \\
    0 & 1 & L_1 \sin \phi & 0 \\
    0 & 0 & 0 & 0 \\
    0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    x_c \\ y_c \\ z_c \\ 1
\end{bmatrix}
\]
Taxonomy of Projections

Planar geometric projections

Parallel

Orthographic
- Top (plan)
- Front elevation
- Side elevation
- Isometric

Oblique
- Cabinet
- Cavalier
- Other

One-point

Two-point

Three-point

Perspective

FVFHP Figure 6.10
Perspective Projection

- Map points onto “view plane” along “projectors” emanating from “center of projection” (COP)
Perspective View Volume

H&B Figure 12.30
Camera to Screen

- Remember: Object -> Camera -> Screen
- Just like raytracer
  - “screen” is the z=d plane for some constant d
- Origin of screen coordinate is (0,0,d)
- Its x and y axes are parallel to the x and y axes of the eye coordinate system
- All these coordinates are in camera space
Overhead view of our screen

Yeah, similar triangles!

\[ \frac{x}{z} = \frac{x'}{d} \Rightarrow x' = \frac{dx}{z} \]

\[ \frac{y}{z} = \frac{y'}{d} \Rightarrow y' = \frac{dy}{z} \]
The Perspective Matrix

- This “division by z” can be accomplished by a 4x4 matrix too!

\[ P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \]

- What happens to the point \((x, y, z, 1)\)

\[(x, y, z, z/d)\]

- What’s this in non-homogenous coordinates?

\[(dx/z, dy/z, d)\]
What’s Next?

3D Geometric Primitives

- Transform in 3D word coordinate system
- Illuminate according to lighting and reflectance
- Transform into 3D camera coordinate system
- Transform into 2D screen coordinate system
- Clip primitives outside camera’s view
- Draw Pixels (also texturing, hidden surface, …)
What’s Next?

3D Geometric Primitives

- Transform in 3D word coordinate system
- Illuminate according to lighting and reflectance
- Transform into 3D camera coordinate system
- Transform into 2D screen coordinate system
- Clip primitives outside camera’s view
- Draw Pixels (also texturing, hidden surface, ...)

Modeling Transformation

Lighting

Viewing Transformation

Projection Transformation

Clipping

Scan Conversion

Image