3D Polygon Rendering Pipeline

Thomas Funkhouser
Princeton University
COS 426, Fall 2000

3D Polygon Rendering

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

- Many applications use rendering of 3D polygons with direct illumination

**Quake II**
(id Software)

3D Polygon Rendering

- Many applications use rendering of 3D polygons with direct illumination

**Quake II**
(id Software)
Ray Casting Revisited

- For each sample …
  - Construct ray from eye position through view plane
  - Find first surface intersected by ray through pixel
  - Compute color of sample based on surface radiance

3D Polygon Rendering

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

This is a pipelined sequence of operations to draw a 3D primitive into a 2D image.

Example: OpenGL

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

OpenGL executes steps of 3D rendering pipeline for each polygon.
3D Rendering Pipeline (for direct illumination)

3D Geometric Primitives

- Modeling Transformation
  - Transform into 3D world coordinate system
- Lighting
- Viewing Transformation
- Projection Transformation
- Clipping
- Scan Conversion
- Image
3D Rendering Pipeline (for direct illumination)

3D Geometric Primitives

Transform into 3D world coordinate system

Illuminate according to lighting and reflectance

Transform into 3D camera coordinate system

Scan Conversion

Image
3D Rendering Pipeline (for direct illumination)

3D Geometric Primitives

Modeling Transformation

Lighting

Viewing Transformation

Projection Transformation

Clipping

Scan Conversion

Image

Transform into 3D world coordinate system

Illuminate according to lighting and reflectance

Transform into 3D camera coordinate system

Transform into 2D camera coordinate system

Clip primitives outside camera’s view

3D Rendering Pipeline (for direct illumination)

3D Geometric Primitives

Modeling Transformation

Lighting

Viewing Transformation

Projection Transformation

Clipping

Scan Conversion

Image

Transform into 3D world coordinate system

Illuminate according to lighting and reflectance

Transform into 3D camera coordinate system

Transform into 2D camera coordinate system

Clip primitives outside camera’s view

Draw pixels (includes texturing, hidden surface, etc.)
Transformations

Transformations map points from one coordinate system to another.

- **3D Geometric Primitives**
  - Modeling Transformation
  - Lighting
  - Viewing Transformation
  - Projection Transformation
  - Clipping
  - Scan Conversion
  - Image

- **Transform** into 3D world coordinate system
  - Illuminate according to lighting and reflectance

- **Transform** into 3D camera coordinate system
  - Clip primitives outside camera’s view

- **Transform** into 2D camera coordinate system
  - Draw pixels (includes texturing, hidden surface, etc.)
**Viewing Transformations**

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

- 3D Object Coordinates
- Modeling Transformation
- 3D World Coordinates
- Viewing Transformation
- 3D Camera Coordinates
- Projection Transformation
- 2D Screen Coordinates
- Window-to-Viewport Transformation
- 2D Image Coordinates

\[ p'(x', y') \]
```

**Viewing Transformation**

- Mapping from world to camera coordinates
  - Origin moves to eye position
  - Up vector maps to Y axis
  - Right vector maps to X axis
Camera Coordinates

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

Viewing Transformation

- Transformation matrix maps camera basis vectors to canonical vectors in camera coordinate system
**Viewing Transformations**

- $p(x,y,z)$
  - 3D Object Coordinates
  - Modeling Transformation
  - 3D World Coordinates
  - Viewing Transformation
  - 3D Camera Coordinates
  - Projection Transformation
  - 2D Screen Coordinates
  - Window-to-Viewport Transformation
  - 2D Image Coordinates
  - $p'(x',y')$

\[ \{ \text{Viewing Transformations} \} \]

**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
  - Isometric
  - Other

Oblique
- Cabinet
  - One-point
- Cavalier
  - Two-point
  - Three-point

Perspective
- Other

FVFHP Figure 6.10
Parallel Projection

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

![Parallel Projection Diagram](image)

Orthographic Projections

- DOP perpendicular to view plane

![Orthographic Projections Diagram](image)
Oblique Projections

- DOP not perpendicular to view plane

Cavalier (DOP at 45°)

Cabinet (DOP at 63.4°)

Parallel Projection View Volume

Parallelepiped View Volume

Back Plane

Front Plane

window

H&B Figure 12.24

H&B Figure 12.30
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

Oblique

One-point

Top (plan)

Front elevation

Axonometric

Cabinet

Two-point

Side elevation

Isometric

Cavalier

Three-point

Other

Other

FVFHP Figure 6.10
Perspective Projection

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

Perspective Projection

- How many vanishing points?

3-Point Perspective  2-Point Perspective  1-Point Perspective
Perspective Projection View Volume

H&B Figure 12.30

Perspective Projection

• Compute 2D coordinates from 3D coordinates with similar triangles

What are the coordinates of the point resulting from projection of \((x,y,z)\) onto the view plane?

\((x,y,z)\)

\(-z\)

\(-z\)

\((0,0,0)\)

\(D\)

View Plane

View Plane

Frustum View Volume

Front Plane

Back Plane

Window

Projection Reference Point

(z)
**Perspective Projection**

- Compute 2D coordinates from 3D coordinates with similar triangles

\[(x, y, z)\]
\[\text{View Plane}\]
\[(xD/z, yD/z)\]
\[(0, 0, 0)\]

**Perspective Projection Matrix**

- 4x4 matrix representation?

\[x_s = x_c D / z_c\]
\[y_s = y_c D / z_c\]
\[z_s = D\]
\[w_s = 1\]

\[
\begin{bmatrix}
x_s \\
y_s \\
z_s \\
w_s
\end{bmatrix}
= \begin{bmatrix}
? & ? & ? & ?
\end{bmatrix}
\begin{bmatrix}
x_c \\
y_c \\
z_c \\
1
\end{bmatrix}
\]
### Perspective Projection Matrix

- **4x4 matrix representation?**

\[
\begin{align*}
x_s &= x_c D / z_c \\
y_s &= y_c D / z_c \\
z_s &= D \\
w_s &= 1
\end{align*}
\]

\[
\begin{bmatrix}
x_s \\
y_s \\
z_s \\
w_s
\end{bmatrix} =
\begin{bmatrix}
x_c \\
y_c \\
z_c \\
1
\end{bmatrix}
\begin{bmatrix}
? & ? & ? & ?
\end{bmatrix}
\]

### Perspective Projection Matrix

- **4x4 matrix representation?**

\[
\begin{align*}
x_s &= x_c D / z_c \\
y_s &= y_c D / z_c \\
z_s &= D \\
w_s &= 1
\end{align*}
\]

\[
\begin{bmatrix}
x_s \\
y_s \\
z_s \\
w_s
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 / D & 0
\end{bmatrix}
\begin{bmatrix}
x_c \\
y_c \\
z_c \\
1
\end{bmatrix}
\]
### Taxonomy of Projections

- **Planar geometric projections**
  - **Parallel**
    - **Orthographic**
      - **Top (plan)**
      - **Front elevation**
      - **Side elevation**
    - **Axonometric**
      - **Isometric**
        - **Other**
  - **Oblique**
    - **Cabinet**
    - **Cavalier**
    - **Axonometric**
      - **Isometric**
        - **Other**
  - **Perspective**
    - **One-point**
    - **Two-point**
    - **Three-point**

*FVFHP Figure 6.10*

### Perspective vs. Parallel

- **Perspective projection**
  + Size varies inversely with distance - looks realistic
  - Distance and angles are not (in general) preserved
  - Parallel lines do not (in general) remain parallel

- **Parallel projection**
  + Good for exact measurements
  + Parallel lines remain parallel
  - Angles are not (in general) preserved
  - Less realistic looking
Classical Projections

- Front elevation
- Elevation oblique
- Plan oblique
- Isometric
- One-point perspective
- Three-point perspective

Angel Figure 5.3

Summary

- Camera transformation
  - Map 3D world coordinates to 3D camera coordinates
  - Matrix has camera vectors as rows

- Projection transformation
  - Map 3D camera coordinates to 2D screen coordinates
  - Two types of projections:
    - Parallel
    - Perspective
Next Time

- **Transform into 3D world coordinate system**
- **Illuminate according to lighting and reflectance**
- **Transform into 3D camera coordinate system**
- **Transform into 2D camera coordinate system**
- **Clip primitives outside camera’s view**
- **Draw pixels (includes texturing, hidden surface, etc.)**