3D Polygon Rendering Pipeline

Michael Kazhdan
(601.457/657)

HB Ch. 12
FvDFH Ch. 6, 18.3
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

God of War
(Santa Monica Studio, 2018)
Ray Casting

• For each sample:
  ◦ Construct ray from the camera into the scene
  ◦ Find first surface intersected by ray through pixel
  ◦ Compute color of sample based on surface radiance
    ↓
  ◦ Send pixels into the scene and get color
3D Polygon Rendering

• For each primitive:
  ○ Send points to the camera and set the pixel color
3D Rendering Pipeline (for direct illumination)

3D Geometric Primitives

Modeling Transformation

Viewing Transformation

Lighting

Projection Transformation

Clipping

Scan Conversion

Image

3D Model

2D Image
3D Rendering Pipeline (for direct illumination)

3D Geometric Primitives

Modeling Transformation

Viewing Transformation

Lighting

Projection Transformation

Clipping

Scan Conversion

Image

Transform from current (local) coordinate system into 3D world coordinate system
3D Rendering Pipeline (for direct illumination)

1. **3D Geometric Primitives**
2. **Modeling Transformation**
3. **Viewing Transformation**
4. **Lighting**
5. **Projection Transformation**
6. **Clipping**
7. **Scan Conversion**
8. **Image**

*Transform into 3D world coordinate system*

*Transform into 3D camera coordinate system*
3D Rendering Pipeline (for direct illumination)

3D Geometric Primitives

- **Modeling Transformation**: Transform into 3D world coordinate system
- **Viewing Transformation**: Transform into 3D camera coordinate system
- **Lighting**: Illuminate according to lighting and reflectance
- **Projection Transformation**
- **Clipping**
- **Scan Conversion**
- **Image**
3D Geometric Primitives

Modeling Transformation

Viewing Transformation

Lighting

Projection Transformation

Clipping

Scan Conversion

Image

Transform into 3D world coordinate system

Transform into 3D camera coordinate system

Illuminate according to lighting and reflectance

Transform into 2D camera coordinate system
3D Rendering Pipeline (for direct illumination)

3D Geometric Primitives

- **Modeling Transformation**: Transform into 3D world coordinate system
- **Viewing Transformation**: Transform into 3D camera coordinate system
- **Lighting**: Illuminate according to lighting and reflectance
- **Projection Transformation**: Transform into 2D camera coordinate system
- **Clipping**: Clip (parts of) primitives outside camera’s view
- **Scan Conversion**: Image
3D Rendering Pipeline (for direct illumination)

3D Geometric Primitives

Modeling Transformation

Transform into 3D world coordinate system

Viewing Transformation

Transform into 3D camera coordinate system

Lighting

Illuminate according to lighting and reflectance

Projection Transformation

Transform into 2D camera coordinate system

Clipping

Clip (parts of) primitives outside camera’s view

Scan Conversion

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

3D Geometric Primitives

Modeling Transformation

Viewing Transformation

Lighting

Projection Transformation

Clipping

Scan Conversion

Image

**Transform** into 3D world coordinate system

**Transform** into 3D camera coordinate system

Illuminate according to lighting and reflectance

**Transform** into 2D camera coordinate system

Clip primitives outside camera’s view

Draw pixels (includes texturing, hidden surface, etc.)
Recall: Homogeneous Coordinates

- Add a 4\textsuperscript{th} coordinate to every 3D point
  - \((x, y, z, w)\) represents a point at location \(\left( \frac{x}{w}, \frac{y}{w}, \frac{z}{w} \right)\)
  - \((x, y, z, 0)\) represents a (directed) point at infinity
  - \((0, 0, 0, 0)\) is not allowed
Transformations map points from one coordinate system to another.

$(x, y, z) \rightarrow (x', y')$

- **Modeling Transformation**
  - 3D Object Coordinates

- **Viewing Transformation**
  - 3D World Coordinates

- **Projection Transformation**
  - 3D Camera Coordinates

- **Window-to-Viewport Transformation**
  - 2D Screen Coordinates

- 2D Image Coordinates

**3D World Coordinates**

**3D Camera Coordinates**

**3D Object Coordinates**

**3D Screen Coordinates**
Transformations

\((x, y, z)\)

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

\((x', y')\)
Viewing Transformation

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

![Diagram showing 3D World Coordinates and 3D Camera Coordinates with arrows indicating Camera Up, Camera Right, and Camera Back.]
Viewing Transformation

- The transformation, $T_{W\rightarrow C}$, taking us from world coordinates to camera coordinates should map:
  - The right vector to the $x$-axis: $(R_x, R_y, R_z, 0) \rightarrow (1,0,0,0)$
  - The up vector to the $y$-axis: $(U_x, U_y, U_z, 0) \rightarrow (0,1,0,0)$
  - The back vector to the $z$-axis: $(B_x, B_y, B_z, 0) \rightarrow (0,0,1,0)$
  - The eye position to the origin: $(E_x, E_y, E_z, 1) \rightarrow (0,0,0,1)$

How should we define this transformation/matrix?
Viewing Transformation

- Consider the inverse transformation, $T_{C \rightarrow W}$, taking us from camera coordinates to world coordinates:

  - $(R_x, R_y, R_z, 0) \leftarrow (1,0,0,0)$
  - $(U_x, U_y, U_z, 0) \leftarrow (0,1,0,0)$
  - $(B_x, B_y, B_z, 0) \leftarrow (0,0,1,0)$
  - $(E_x, E_y, E_z, 1) \leftarrow (0,0,0,1)$

- This is described by the matrix:

  $\begin{pmatrix}
  x^w \\
  y^w \\
  z^w \\
  1 \\
  \end{pmatrix} =
  \begin{pmatrix}
  R_x & U_x & B_x & E_x \\
  R_y & U_y & B_y & E_y \\
  R_z & U_z & B_z & E_z \\
  0 & 0 & 0 & 1 \\
  \end{pmatrix}
  \begin{pmatrix}
  x^c \\
  y^c \\
  z^c \\
  1 \\
  \end{pmatrix}$
Finding the Viewing Transformation

- The camera-to-world matrix:

\[
\begin{pmatrix}
  x^w \\
y^w \\
z^w \\
1
\end{pmatrix} =
\begin{pmatrix}
  R_x & U_x & B_x & E_x \\
  R_y & U_y & B_y & E_y \\
  R_z & U_z & B_z & E_z \\
  0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
  x^c \\
y^c \\
z^c \\
1
\end{pmatrix}
\]

- The world-to-camera matrix is its inverse:

\[
\begin{pmatrix}
  x^c \\
y^c \\
z^c \\
1
\end{pmatrix} =
\begin{pmatrix}
  R_x & U_x & B_x & E_x \\
  R_y & U_y & B_y & E_y \\
  R_z & U_z & B_z & E_z \\
  0 & 0 & 0 & 1
\end{pmatrix}^{-1}
\begin{pmatrix}
  x^w \\
y^w \\
z^w \\
1
\end{pmatrix}
\]

\[T_{W\rightarrow c} = T_{C\rightarrow W}^{-1}\]
Transformations

\[(x, y, z)\]

- **Modeling Transformation**
  - 3D Object Coordinates

- **Viewing Transformation**
  - 3D World Coordinates

- **Projection Transformation**
  - 3D Camera Coordinates

- **Window-to-Viewport Transformation**
  - 2D Screen Coordinates

\[(x', y')\]
Projection

• General definition:
  ◦ A linear transformation of 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

Isometric

Side elevation

Axonometric

Cabinet

Cavalier

One-point

Perspective

Two-point

Three-point

Other

Other

FvDFH Figure 6.13
Projection

- Two general classes of projections, both of which shoot rays from the scene, through the view plane:
  - Parallel Projection:
    » Rays converge at a point at infinity and are **parallel**
  - Perspective “Projection”:
    » Rays converge at a finite point, giving rise to **perspective distortion**
Taxonomy of Projections

Planar geometric projections

Parallel

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

Axonometric
- Isometric
- Other

Oblique
- Cabinet
- Cavalier
- Other

One-point
- Two-point
- Three-point

Perspective
- Other

FvDFH Figure 6.13
Parallel Projection

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

✓ Parallel lines remain parallel
✓ Relative proportions of objects preserved
✗ Angles are not preserved
✗ Less realistic looking
Taxonomy of Projections

Planar geometric projections

Parallel

Orthographic

Top (plan)  Front elevation

Axonometric  Side elevation

Isometric

Oblique

Cabinet  Cavalier

One-point

Two-point  Three-point

Perspective

FvDFH Figure 6.13
Orthographic Projections

- DoP perpendicular to view plane

Angel Figure 5.5
Orthographic Projections

- DoP perpendicular to view plane

- Lines perpendicular to the view plane vanish
- Faces parallel to the view plane are un-distorted.
Taxonomy of Projections

Planar geometric projections

Parallel

Orthographic

- Top (plan)
- Front elevation
- Axonometric
  - Side elevation
- Isometric
  - Other

Oblique

- Cabinet
  - Cavalier
- Other

Perspective

One-point

Two-point

Three-point
Oblique Projections

- DoP **not** perpendicular to view plane

\[
(x^s, y^s) = L(\cos \phi, \sin \phi)
\]

\[
(x^c, y^c, 1) = L(\cos \phi, \sin \phi, 1)
\]

- \(\phi\) is the angle of the projection of the view plane’s normal
- \(L\) is the scale factor applied to the view plane’s normal

Cavalier
(DoP \(\alpha = 45^\circ\))

\[
\begin{align*}
\phi &= 45^\circ \\
L &= 1
\end{align*}
\]

Cabinet
(DoP \(\alpha = 63.4^\circ\))

\[
\begin{align*}
\phi &= 45^\circ \\
L &= \frac{1}{2}
\end{align*}
\]

H&B Figure 12.21
Parallel Projection Matrix

• General parallel projection transformation:

\[
\begin{bmatrix}
 x^s \\
y^s \\
0 \\
1
\end{bmatrix} =
\begin{bmatrix}
 1 & 0 & L \cos \phi & 0 \\
 0 & 1 & L \sin \phi & 0 \\
 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x^c \\
y^c \\
z^c \\
1
\end{bmatrix}
\]

Cavalier
(DoP \( \alpha = 45^\circ \))

\[
\phi = 45^\circ \\
L = 1
\]

Cabinet
(DoP \( \alpha = 63.4^\circ \))

\[
\phi = 45^\circ \\
L = 1/2
\]

H&B Figure 12.21
Parallel Projection Matrix

- General parallel projection transformation:

\[
\begin{bmatrix}
  x^s \\
y^s \\
0 \\
1
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 & L \cos \phi & 0 \\
  0 & 1 & L \sin \phi & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x^c \\
y^c \\
z^c \\
1
\end{bmatrix}
\]

Note:
This matrix represents an affine transformation.

- Cavalier (DoP \(\alpha = 45^\circ\))

- Cabinet (DoP \(\alpha = 63.4^\circ\))

H&B Figure 12.21
Parallel Projection View Volume

Parallelepiped View Volume

Back Plane

Front Plane

window

$z_v$

H&B Figure 12.30
Taxonomy of Projections

Planar geometric projections

Parallel

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

Oblique
  - Cabinet
  - Cavalier

One-point
  - Two-point
  - Three-point

Perspective

Isometric
  - Other
Perspective “Projection”

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

• How many vanishing points?

Number of vanishing points determined by number of axes parallel to the view plane

Angelo Figure 5.10
Perspective Projection

• Parallel lines do not remain parallel!
Perspective Projection View Volume

H&B Figure 12.30
Perspective Projection

• What are the coordinates of the point resulting from projection of \((x^c, y^c, z^c)\) onto the view plane a unit distance along the \(z\)-axis?
Perspective Projection

- For any point \((x^c, y^c, z^c)\) and any scalar \(\alpha\), the points \((x^c, y^c, z^c)\) and \((\alpha \cdot x^c, \alpha \cdot y^c, \alpha \cdot z^c)\) map to the same location.
Perspective Projection

• For any point \((x^c, y^c, z^c)\) and any scalar \(\alpha\), the points \((x^c, y^c, z^c)\) and \((\alpha \cdot x^c, \alpha \cdot y^c, \alpha \cdot z^c)\) map to the same location.

• Since we want the position of the point on the line that intersect the image plane at a unit distance along the \(z\)-axis:

\[
(x^c, y^c, z^c) \rightarrow \left(\frac{x^c}{z^c}, \frac{y^c}{z^c}, 1\right)
\]

\((x^c, y^c, z^c)\)
Perspective Projection Matrix

$$(x^c, y^c, z^c) \rightarrow \left(\frac{x^c}{z^c}, \frac{y^c}{z^c}, 1\right)$$

We can’t represent this with a $3 \times 3$ matrix!

With homogenous coordinates, we can write this as:

$$(x^c, y^c, z^c, 1) \rightarrow \left(\frac{x^c}{z^c}, \frac{y^c}{z^c}, 1, 1\right) \equiv (x^c, y^c, z^c, z^c)$$

In matrix form, this gives:

$$\begin{bmatrix} x^s \\ y^s \\ 1 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x^c \\ y^c \\ z^c \\ 1 \end{bmatrix}$$
Perspective Projection Matrix

\[(x^c, y^c, z^c) \rightarrow \left( \frac{x^c}{z^c}, \frac{y^c}{z^c}, 1 \right)\]

We can’t represent this with a 3 \times 3 matrix!

With homogenous coordinates, we can write this as:

\[(x^c, y^c, z^c, 1) \rightarrow \left( \frac{x^c}{z^c}, \frac{y^c}{z^c}, 1, 1 \right) \equiv (x^c, y^c, z^c, z^c)\]

Note:
This matrix represents a projective transformation

In matrix form, this gives:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x^c \\
y^c \\
z^c \\
1 \\
1
\end{bmatrix}
\equiv
\begin{bmatrix}
x^s \\
y^s \\
1 \\
1
\end{bmatrix}
\]
Taxonomy of Projections

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

Front elevation

Elevation oblique

Plan oblique

Isometric

One-point perspective

Three-point perspective

Angel Figure 5.3
Perspective vs. Parallel

• Perspective projection
  ✓ Size varies inversely with distance - looks realistic
  ✓ Angles are preserved on faces parallel to the view plane
  ✗ Distance are not preserved
  ✗ Only parallel lines that are parallel to the view plane remain parallel

• Parallel projection
  ✓ Good for exact measurements
  ✓ Parallel lines remain parallel
  ✓ Angles and distance are preserved on faces parallel to the view plane
  ✗ Less realistic looking
Transformations

\[(x, y, z)\]

\[\xrightarrow{\text{Modeling Transformation}}\]

3D Object Coordinates

\[\xrightarrow{\text{Viewing Transformation}}\]

3D World Coordinates

\[\xrightarrow{\text{Projection Transformation}}\]

3D Camera Coordinates

\[\xrightarrow{\text{Window-to-Viewport Transformation}}\]

2D Screen Coordinates

\[\xrightarrow{\text{}}\]

2D Image Coordinates

\[
V = \text{viewport transform}
\]

\[
V = \begin{bmatrix}
1 & 0 & v_x^1 \\
0 & 1 & v_x^2 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
\frac{v_x^2 - v_x^1}{w_x^2 - w_x^1} & 0 & 0 \\
0 & \frac{v_y^2 - v_y^1}{w_y^2 - w_y^1} & 0 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
1 & 0 & -w_x^1 \\
0 & 1 & -w_y^1 \\
0 & 0 & 1
\end{bmatrix}
\]
3D Rendering Pipeline (for direct illumination)

\[(x, y, z)\]

- 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

\[(x', y')\]
Transformations

3D Geometric Primitives

Modeling Transformation

Viewing Transformation

Lighting

Projection Transformation

Clipping

Scan Conversion

Image

3D Model

2D Screen

\[ I = I_E + \sum_L \left[ K_A \cdot I_L^A + (K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R} \rangle^n) \cdot I_L \right] \]
Transformations

3D Geometric Primitives

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

Vertex processing

- Originally, vertex processing was fixed
- On modern cards this can be programmed in the vertex shader

3D Model

2D Screen