3D Polygon Rendering Pipeline

Michael Kazhdan

(600.357 / 600.457)

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

Crysis 3
(Electronic Arts, 2013)
3D Polygon Rendering

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

GTA 5
(Rockstar Games, 2013)
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

More efficient algorithms utilize spatial coherence!
3D Polygon Rendering

• Logical inverse of ray casting
  
  Instead of sending rays from the camera into the scene, send rays from the scene into the camera.
3D Polygon Rendering

• Ray casting:
  Pick pixel and figure out what color it should be based on what object its ray hits

• Polygon rendering:
  Pick polygon and figure out what pixels it should affect
3D Rendering Pipeline (for direct illumination)

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

3D Model

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

3D Geometric Primitives

<table>
<thead>
<tr>
<th>Modeling Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera Transformation</td>
</tr>
<tr>
<td>Lighting</td>
</tr>
<tr>
<td>Projection Transformation</td>
</tr>
<tr>
<td>Clipping</td>
</tr>
<tr>
<td>Scan Conversion</td>
</tr>
<tr>
<td>Image</td>
</tr>
</tbody>
</table>

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

3D Geometric Primitives

- Modeling Transformation
- Camera Transformation
- Lighting
- Projection Transformation
- Clipping
- Scan Conversion
- 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
- Camera Transformation
  - Transform into 3D camera coordinate system
- Lighting
  - Illuminate according to lighting and reflectance
- Projection Transformation
- Clipping
- Scan Conversion
- Image
3D Rendering Pipeline (for direct illumination)

3D Geometric Primitives

- Modeling Transformation
  - Transform into 3D world coordinate system

- Camera Transformation
  - Transform into 3D camera coordinate system

- Lighting
  - Illuminate according to lighting and reflectance

- Projection Transformation
  - Transform into 2D camera coordinate system

- Clipping

- Scan Conversion

- Image
3D Rendering Pipeline (for direct illumination)

3D Geometric Primitives

- **Modeling Transformation**
  - Transform into 3D world coordinate system

- **Camera 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
- **Camera 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

Transform into 3D world coordinate system

Camera Transformation

Transform into 3D camera coordinate system

Lighting

Illuminate according to lighting and reflectance

Projection Transformation

Transform into 2D camera coordinate system

Clipping

Clip primitives outside camera’s view

Scan Conversion

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

Image
Transformations map points from one coordinate system to another. Transformations are used to map points from one coordinate system to another. The process involves several steps:

1. **Modeling Transformation**
   - Maps from 3D Object Coordinates to 3D World Coordinates.

2. **Camera Transformation**
   - Maps from 3D World Coordinates to 3D Camera Coordinates.

3. **Projection Transformation**
   - Maps from 3D Camera Coordinates to 2D Screen Coordinates.

4. **Window-to-Viewport Transformation**
   - Maps from 2D Screen Coordinates to 2D Image Coordinates.

The resulting coordinates are denoted as $(x', y')$. This diagram illustrates the transformation process from 3D Object Coordinates to 2D Image Coordinates.
Viewing Transformations

\[(x, y, z)\]

3D Object Coordinates

Modeling Transformation

3D World Coordinates

Camera Transformation

3D Camera Coordinates

Projection Transformation

2D Screen Coordinates

Window-to-Viewport Transformation

2D Image Coordinates

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

- Mapping from world to camera coordinates (in homogeneous) coordinates:
  - Eye position maps to origin:
    $$(E_x, E_y, E_z, 1) \rightarrow (0,0,0,1)$$
  - Right vector maps to $x$-axis:
    $$(R_x, R_y, R_z, 0) \rightarrow (1,0,0,0)$$
  - Up vector maps to $y$-axis:
    $$(U_x, U_y, U_z, 0) \rightarrow (0,1,0,0)$$
  - Back vector maps to $z$-axis:
    $$(B_x, B_y, B_z, 0) \rightarrow (0,0,1,0)$$
Camera Coordinates

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

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

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

- Trick: map from camera coordinates to world
  - \((E_x, E_y, E_z, 1) \leftarrow (0,0,0,1)\)
  - \((R_x, R_y, R_z, 0) \leftarrow (1,0,0,0)\)
  - \((U_x, U_y, U_z, 0) \leftarrow (0,1,0,0)\)
  - \((B_x, B_y, B_z, 0) \leftarrow (0,0,1,0)\)

\[
p^w = T^{-1}p^c
\]

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

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

\[(x, y, z)\]

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

Axonometric
  - Isometric

Oblique
  - Cabinet
  - Cavalier

One-point
  - Two-point
  - Three-point

Perspective
  - 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

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
  - Far away objects don’t get smaller
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

Other

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.

Angel Figure 5.5
Taxonomy of Projections

Planar geometric projections

Parallel

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

Oblique

Perspective

One-point

Cabinet

Cavalier

Two-point

Three-point

Other
Oblique Projections

• DoP not perpendicular to view plane

\( \phi \) 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 \( \alpha = 63.4^\circ \))

\[ \begin{align*}
\phi &= 45^\circ \\
L &= 1/2
\end{align*} \]
Parallel Projection Matrix

- General parallel projection transformation:

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

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

\[
\phi = 45^\circ \\
\frac{L}{L} = 1
\]

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

\[
\phi = 45^\circ \\
\frac{L}{L} = \frac{1}{2}
\]

H&B Figure 12.21
Parallel Projection View Volume

H&B Figure 12.30
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

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?

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

Angel Figure 5.10
Perspective Projection

- As with translation, perspective “projection” is not really a projection because it is not a linear map from 3D to 2D.
  - Parallel lines do not remain parallel!
Perspective Projection View Volume

View Plane

Frustum View Volume

Back Plane

Front Plane

window

Projection Reference Point

H&B Figure 12.30
Perspective Projection

• What are the coordinates of the point resulting from projection of \((x_0, y_0, z_0)\) onto the view plane at a distance of \(D\) along the \(z\)-axis?
Perspective Projection

- Use the fact that for any point \((x_0, y_0, z_0)\) and any scalar \(\alpha\), the points \((x_0, y_0, z_0)\) and \((\alpha x_0, \alpha y_0, \alpha z_0)\) map to the same location:
Perspective Projection

• Use the fact that for any point \((x_0, y_0, z_0)\) and any scalar \(\alpha\), the points \((x_0, y_0, z_0)\) and \((\alpha x_0, \alpha y_0, \alpha z_0)\) map to the same location.

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

\[
(x_0, y_0, z_0) \rightarrow \left( \frac{x_0D}{z_0}, \frac{y_0D}{z_0}, D \right)
\]
Perspective Projection Matrix

- 4x4 matrix representation?

\[
\begin{align*}
    x^s &= \frac{x^c D}{z^c} \\
    y^s &= \frac{y^c D}{z^c} \\
    z^s &= D \\
    w^s &= 1
\end{align*}
\]

\[
\begin{pmatrix}
    (x^c D, y^c D, D, 1) \\
    \left(\frac{x^c}{z^c}, \frac{y^c}{z^c}, 1\right)
\end{pmatrix}
\]

We want to divide by the \( z \) coordinate. How do we do that with a 4 \( \times \) 4 matrix?

Recall that in homogenous coordinates:

\[
(x, y, z, w) = \left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}, 1\right)
\]

\[
\begin{bmatrix}
    x^s \\
    y^s \\
    z^s \\
    w^s
\end{bmatrix} =
\begin{bmatrix}
    ? & ? & ? & ?
\end{bmatrix}
\begin{bmatrix}
    x^c \\
    y^c \\
    z^c \\
    1
\end{bmatrix}
\]
Perspective Projection Matrix

- 4x4 matrix representation?

\[
\begin{align*}
    x^s &= \frac{x^c D}{z^c} \\
    y^s &= \frac{y^c D}{z^c} \\
    z^s &= D \\
    w^s &= 1
\end{align*}
\]

We want to divide by the \( z \) coordinate. How do we do that with a \( 4 \times 4 \) matrix?

Recall that in homogenous coordinates:

\[
(x, y, z, w) = \left( \frac{x}{w}, \frac{y}{w}, \frac{z}{w}, 1 \right)
\]

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

Planar geometric projections

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

Perspective
- Oblique
  - Cabinet
    - Cavalier
    - Other
  - Other
- One-point
- Two-point
- Three-point
Classical Projections

- Front elevation
- Elevation oblique
- Plan oblique
- Isometric
- One-point perspective
- Three-point perspective
Perspective vs. Parallel

• Perspective projection
  + Size varies inversely with distance - looks realistic
    – Distance and angles 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 are preserved on faces parallel to the view plane
    – Less realistic looking