Rendering Pipeline

Courtesy of Adam Finkelstein, Princeton University
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
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

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

This is a pipelined sequence of operations to draw a 3D primitive into a 2D image.
OpenGL executes steps of 3D rendering pipeline for each polygon.

```c
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();
```
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

- Modeling Transformation
- Lighting
  - Illuminate according to lighting and reflectance
- Viewing Transformation
- Projection Transformation
- Clipping
- Scan Conversion
- Image

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

3D Geometric Primitives

- Modeling Transformation
  - Transform into 3D world coordinate system

- Lighting
  - Illuminate according to lighting and reflectance

- Viewing Transformation
  - Transform into 3D camera coordinate system

- Projection Transformation

- Clipping

- Scan Conversion

- Image
3D Rendering Pipeline (for direct illumination)

3D Geometric Primitives

- Modeling Transformation
- Transform into 3D world coordinate system
- Lighting
- Illuminate according to lighting and reflectance
- Viewing Transformation
- Transform into 3D camera coordinate system
- Projection Transformation
- Transform into 2D camera coordinate system
- Clipping
- Scan Conversion
- Image
3D Rendering Pipeline (for direct illumination)

3D Geometric Primitives

1. Modeling Transformation
   - Transform into 3D world coordinate system

2. Lighting
   - Illuminate according to lighting and reflectance

3. Viewing Transformation
   - Transform into 3D camera coordinate system

4. Projection Transformation
   - Transform into 2D camera coordinate system

5. Clipping
   - Clip primitives outside camera’s view

6. Scan Conversion

   Image
3D Rendering Pipeline (for direct illumination)

- **3D Geometric Primitives**
- **Modeling Transformation**
  - Transform into 3D world coordinate system
- **Lighting**
  - Illuminate according to lighting and reflectance
- **Viewing Transformation**
  - Transform into 3D camera coordinate system
- **Projection Transformation**
  - Transform into 2D camera coordinate system
- **Clipping**
  - Clip primitives outside camera’s view
- **Scan Conversion**
  - Draw pixels (includes texturing, hidden surface, ...)
- **Image**
Transformations

3D Geometric Primitives

- **Modeling Transformation**
  - Transform into 3D world coordinate system
- **Lighting**
  - Illuminate according to lighting and reflectance
- **Viewing Transformation**
  - Transform into 3D camera coordinate system
- **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.

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

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

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

- **Projection Transformation**
  - 3D Camera Coordinates
  - 2D Screen Coordinates

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

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

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

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

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

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

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

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

\[ p^c = T \cdot p^w \]

- Trick: find $T^{-1}$ taking objects in camera to world

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

\[
\begin{bmatrix}
  x' \\
  y' \\
  z' \\
  w'
\end{bmatrix}
= \begin{bmatrix}
  a & b & c & d \\
  e & f & g & h \\
  i & j & k & l \\
  m & n & o & p
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  w
\end{bmatrix}
\]
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
  \[ 3D \text{ Object Coordinates} \]
  \[ p(x',y',z') \]

- Viewing Transformation
  \[ 3D \text{ World Coordinates} \]
  \[ 3D \text{ Camera Coordinates} \]

- Projection Transformation
  \[ 2D \text{ Screen Coordinates} \]

- Window-to-Viewport Transformation
  \[ 2D \text{ Image Coordinates} \]
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
Taxonomy of Projections

Planar geometric projections

Parallel

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

Oblique
- Cabinet
  - Cavalier
- Other

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

Other
Parallel Projection

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

Angel Figure 5.4
Orthographic Projections

- DOP perpendicular to view plane
Oblique Projections

- DOP not perpendicular to view plane

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

Cabinet  
(DOP $\alpha = 63.4^\circ$)
Parallel Projection View Volume
Parallel Projection Matrix

• General parallel projection transformation:
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

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?
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?
Perspective Projection

- Compute 2D coordinates from 3D coordinates with similar triangles
Perspective Projection Matrix

- 4x4 matrix representation?

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

• 4x4 matrix representation?

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

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

\[
\begin{align*}
    x' &= x_c \\
    y' &= y_c \\
    z' &= z_c \\
    w' &= z_c / D
\end{align*}
\]
Perspective Projection Matrix

- 4x4 matrix representation?

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

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

Planar geometric projections

Parallel

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

Oblique
- Cabinet
- Cavalier
- Other

Perspective

One-point
- Two-point
- Three-point

Other
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
What’s next?

3D Geometric Primitives

Modeling Transformation
- Transform into 3D world coordinate system

Lighting
- Illuminate according to lighting and reflectance

Viewing Transformation
- Transform into 3D camera coordinate system

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