Lecture 5: 3D Rendering Pipeline (II)

Parallel Projection

- Project from 3D space to viewer's 2D space

Perspective Projection

- The farther the object is, the smaller it appears
- Some photo editing software allows you to perform "Perspective Correction"
Perspective Projection

\[ \frac{x}{d} = \frac{x}{z} \Rightarrow x_s = \frac{x}{d} \]

\[ \frac{y}{d} = \frac{y}{z} \Rightarrow y_s = \frac{y}{d} \]

\[ z_s = d \Rightarrow z_s = \frac{z}{d} \]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & \frac{1}{w} \\
0 & 0 & 0 & w
\end{bmatrix}
\]

Viewing Frustum

Think about looking through a window in a dark room

Define Viewing Frustum (Projection Transformation)

Parameters:
- Fov: Field of View
- Aspect ratio = Width/Height
- Near z
- Far z
Viewport Transformation

- The actual 2D projection to the viewer
- Copy to your back buffer (frame buffer)
- Can be programmed, scaled, ...

Other Geometry Optimizations

- Backface Culling
- Clipping
- Hidden surface removal (Occlusion)

Backface Culling

- Determine "facing direction"
- Triangle order matters
- How to compute a normal vector for 2 given vectors?
  - Using Cross product of 2 given vectors

Compute the Surface Normal for a Triangle

\[ \vec{v}_1 = \hat{i} + 3\hat{j} + 0\hat{k} \]
\[ \vec{v}_2 = 4\hat{i} + 0\hat{j} + 0\hat{k} \]
Compute the Surface Normal for a Triangle

\[ \vec{v}_1 = 4\hat{i} + 0\hat{j} + 0\hat{k} \]
\[ \vec{v}_2 = 3\hat{i} + 3\hat{j} + 0\hat{k} \]

Backface Culling Method
- Check if the normal is facing the camera
- How to determine that?
  - Use Dot Product

\[ \vec{A} \cdot \vec{B} = |A||B|\cos \theta \]
\[ \vec{A} \cdot \vec{B} > 0 \quad \Rightarrow \quad -\frac{\pi}{2} < \theta < \frac{\pi}{2} \]
When to Perform Backface Culling?

- Transform your camera to the world space first!

How?

- 3D Clipping
  - Test 6 planes if a triangle is inside, outside, or partially inside the view frustum
  - Clipping creates new triangles (triangulation)
    - Interpolate new vertices info

How to Clip Against a Plane?

- Test each vertex of a triangle
  - Outside
  - Inside
  - Partially inside
- Incurred computation overhead
- Save unnecessary computation (and bandwidth) later
- Need to know how to determine a plane
- Need to know how to determine a vertex is inside or outside a plane

What it Takes to Determine a Plane?

- You need two things to specify a plane
  - A point on the plane \((p0, p1, p2)\)
  - A vector (normal) perpendicular to the plane \((a, b, c)\)
  - Plane \( a * (x - p0) + b * (y - p1) + c * (z - p2) = 0 \)
Distance Calculation from a Plane

- Given a point R, calculate the distance
  - Distance > 0 inside the plane
  - Distance = 0 on the plane
  - Distance < 0 outside the plane

\[
\mathbf{d} = |\mathbf{R} - \mathbf{P}| \cdot \cos \theta = |\mathbf{R} - \mathbf{P}| \cdot \frac{\mathbf{v} \cdot (\mathbf{R} - \mathbf{P})}{|\mathbf{v}|^2} = \frac{\mathbf{v} \cdot (\mathbf{R} - \mathbf{P})}{|\mathbf{v}|}
\]

Illumination Models

- It won’t look 3D without lighting
- Part of geometry processing
- Illumination Types
  - Ambient
  - Diffuse
  - Specular
  - Emissive
Local vs. Global Illumination

- **Local Illumination**
  - Direct illumination: Light shines on all objects without blocking or reflection
  - Used in most games

- **Global Illumination**
  - Indirect illumination: Light bounces from one object to other objects
  - Add more realism (non real-time rendering)
  - Computationally much more expensive
  - Ray tracing, radiosity

Common Light Sources

- **Directional Light** (Infinitely far away)
- **Point Light** (Emit in all directions)
- **Spot Light** (Emit within a cone)

Illumination: Ambient Lighting

- Not created by any light source
- A constant lighting from all directions
- Contributed by scattered light in a surrounding

\[ C_{\text{ambient}} = M_{\text{ambient}} \otimes L_{\text{ambient}} \]

Illumination: Diffuse Lighting

- Light sources are given
- Assume light bounces in all directions

Reflected Light will reach the eyes no matter where the camera is!
Reflected Light Intensity Calculation

- Reflectivity ∝ the entry angle
- Use Lambert's cosine Law

\[ F(\theta) = \cos(\theta) \]

Illumination: Diffuse Lighting

\[ C_{\text{diff}} = \max (\vec{L} \cdot \vec{n}, 0) \cdot (M_{\text{diff}} \otimes I_{\text{diff}}) \]

Illumination: Specular Lighting

- Create shining surface (surface perfectly reflects)
- Viewpoint dependent

\[ C_{\text{spec}} = (\max (\vec{n} \cdot \vec{H}, 0))^5 \cdot M_{\text{spec}} \otimes I_{\text{spec}} \]

Jim Blinn’s Specular Model

- An improved version from Phong Specular model which uses Reflective vector R
- “S” controls the bright region around surface
Brightness Effect (S)

\[ C_{sp} = (\max(n \cdot \vec{H}, 0))^2 \cdot M_{sp} \otimes L_{sp} \]

where \( n \cdot \vec{H} = |\vec{H}| \cos \theta \)

More shining

\[ \begin{align*}
    \cos^2 \theta > \cos \omega & \approx 0.65 \\
    \cos^3 \theta & \approx 0.62 \\
    \cos^4 \theta & \approx 0.00
\end{align*} \]

Specular Lighting Effect

- A larger S shows more concentration of the reflection

Illumination: Emissive Lighting

\[ C_{all} = C_e + M_e \otimes L_e + \max(-\vec{L} \cdot \vec{n}, 0) \cdot (M_d \otimes L_d) + (\max(n \cdot \vec{H}, 0))^2 \cdot M_s \otimes L_s \]

- Color is emitted by the material only
Common Light Sources

- **Directional Light** (Infinite far away)
- **Point Light** (Emit in all directions)
- **Spot Light** (Emit within a cone)

Light Source Properties

- **Position**
- **Range**
  - Specifying the visibility
- **Attenuation**
  - The farther the light source, the dimmer the color
    
    $$\text{Atten} = a_0 + a_1 \cdot d + a_2 \cdot d^2$$

- Spotlight Effect
  - Similar idea to specular lighting
  - Falloff factor determines the fading effect of a spotlight
  - "f" exponentially decreases the cos$$\alpha$$ value