Announcements

• **MP4** released - 4.1 due on Nov 2, 11, 18 @11am

• Project proposals - graded. If you didn't get 100 - find me and talk to me.

Final project upcoming deadlines:

• **Oct 31**, picture-title.

• **Nov 14**, a short video of your progress.
This was geometry: LaValle, Ch 3
Now, read LaValle, Ch 7
Object-Order Rendering

(pixel-by-pixel)

Image-order rendering stages:

1. Ray generation
2. Ray intersection
3. Shading: assign RGB values

(triangle-by-triangle)

Object-order rendering stages:

1. Rasterization
2. Depth order
3. Shading: assign RGB values
Image-Order Rendering: Shading

Ambient light component:

\[ L = k_d \cdot I \cdot \max(0, n \cdot l) + k_s \cdot I \cdot \max(0, n \cdot h) + k_a \cdot I_a \]

diffuse

specular

ambient
Image-Order Rendering: Shading

\[ L = k_d \cdot I \cdot \max(0, n \cdot l) + k_s \cdot I \cdot \max(0, n \cdot h) + k_a \cdot I_a \]

diffuse

specular

ambient
Image-Order Rendering: Shading

Multiple light sources:

\[
L = \sum_{i=1}^{N} \left( k_d \cdot I \cdot \max(0, n \cdot l) + k_s \cdot I \cdot \max(0, n \cdot h) \right) + k_a \cdot I_a
\]

diffuse

specular

ambient
Image-Order Rending: Shading

Alternative approach: global illumination
- Handling multiple reflections
- Very expensive
diffraction, refraction, transparency, translucency

http://graphics.stanford.edu/~henrik/images/global.html
Image-Order Rending: Shading

BRDF: Bidirectional Reflectance Distribution Function

\( f(\theta_i, \epsilon_i, \theta_r, \epsilon_r) = \frac{\text{radiance}}{\text{irradiance}} \)
Image-Order Rendering: Shading in VR

- higher resolution!
- more FPS!
- two eyes!
- faster moving viewing point(s)
- wider FOV!
**Object-Order Rending**

**Image-order rendering stages:**

1. Ray generation
2. Ray intersection
3. Shading: assign RGB values

**Object-order rendering stages:**

1. Rasterization
2. Depth order
3. Shading: assign RGB values

**Rasterization:**

Assign pixels to triangle

**Problem:**

Aliasing, distortion of primitives

**Q:** does this happen in IO rendering?
Painter's Algorithm

- Display polygons in back-to-front order
- Sort polygons by z-value
  - Which vertex?
  - $O(n \log n)$ (in VR? 😅)
- Problems…
Key Observation: Each pixel displays color of only one triangle, ignores everything behind it
• Don’t need to sort triangles, just find for each pixel the closest triangle
• Z-buffer: one fixed or floating point value per pixel
• Algorithm:
For each rasterized fragment \((x,y)\)
  If \(z > z\text{buffer}(x,y)\) then
    \(\text{framebuffer}(x,y) = \text{fragment color}\)
    \(z\text{buffer}(x,y) = z\)

\[\begin{array}{cccc}
5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 \\
\end{array} + \begin{array}{cccc}
5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 \\
\end{array} = \begin{array}{cccc}
5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 \\
\end{array}
\]

\[\begin{array}{cccc}
5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 \\
\end{array} + \begin{array}{cccc}
7 & 6 & 7 & 7 \\
5 & 6 & 7 & 7 \\
4 & 5 & 6 & 7 \\
3 & 4 & 5 & 6 \\
2 & 3 & 4 & 5 \\
\end{array} = \begin{array}{cccc}
5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 \\
4 & 5 & 5 & 7 \\
3 & 4 & 5 & 6 \\
2 & 3 & 4 & 5 \\
\end{array}\]
Clipping/Culling

Clipping:

- Remove triangles behind the eye

Q: Where is clipping plane in VR?

Culling:

- View Frustum: Remove triangles outside of viewing frustum.

- Occlusion: Remove hidden triangles.

- Backface: Remove triangles on "back" of objects.
Interpolation and Barycentric Coordinates

Propagate color, RGB, other attributes efficiently:

Barycentric coordinates: 
\[(d_1, d_2, d_3)\]

\[P = d_1P_1 + d_2P_2 + d_3P_3\]

\[d_1, d_2, d_3 \in [0,1]\]

\[d_1 + d_2 + d_3 = 1\]

\[\forall x1: d_1 = d_2 = 0\]
\[\forall x2: d_1 = 0\]

Depth calculation:
\[z = d_1z_1 + d_2z_2 + d_3z_3\]
RGB Mapping

\[ R = d_1 R_1 + d_2 R_2 + d_3 R_3 \]
\[ G = d_1 G_1 + d_2 G_2 + d_3 G_3 \]
\[ B = d_1 B_1 + d_2 B_2 + d_3 B_3 \]

Examples: