Example: Hard vs Soft Shadows

Hard Shadow

Soft Shadow
Definitions

- Point light: create hard shadows (unrealistic)
- Area light: create soft shadows (more realistic)
Shadow Map Problems

- Low shadow map resolution results in jagged shadows

from viewpoint

from light
Percentage Closer Filtering

- Blend multiple shadow map samples to reduce jaggies
Shadow Map Result
Arbitrary geometry

- Shadow mapping and shadow volumes can render shadows onto arbitrary geometry
  - Recent focus on shadow volumes, because currently most popular, and works on most hardware
- Works in real time...
- Shadow mapping is used in Pixar’s rendering software
Shadow volumes

- Most popular method for real time
- Shadow volume concept
Shadow volumes

- Create volumes of space in shadow from each polygon in light
- Each triangle creates 3 projecting quads
Shadow Volume Example

Image courtesy of NVIDIA Inc.
Fog
Fog example

- Fog is atmospheric effect
  - Better realism, helps determine distances
Fog

- Fog was part of OpenGL fixed function pipeline

- Programming fixed function fog
  - **Parameters:** Choose fog color, fog model
  - **Enable:** Turn it on

- Fixed function fog deprecated!!

- Shaders can implement even better fog

- **Shaders implementation:** fog applied in fragment shader just before display
Rendering Fog

- Mix some color of fog: \( \mathbf{c}_f \) + color of surface: \( \mathbf{c}_s \)

\[
\mathbf{c}_p = f\mathbf{c}_f + (1-f)\mathbf{c}_s \quad f \in [0,1]
\]

- If \( f = 0.25 \), output color = 25% fog + 75% surface color
  - \( f \) computed as function of distance \( z \)
  - 3 ways: linear, exponential, exponential-squared
  - Linear:

\[
f = \frac{z_{end} - z_p}{z_{end} - z_{start}}
\]
Fog Shader Fragment Shader Example

float dist = abs(Position.z);
Float fogFactor = (Fog.maxDist - dist) / Fog.maxDist - Fog.minDist;
fogFactor = clamp(fogFactor, 0.0, 1.0);

vec3 shadeColor = ambient + diffuse + specular
vec3 color = mix(Fog.color, shadeColor, fogFactor);
FragColor = vec4(color, 1.0);

\[ f = \frac{z_{end} - z_p}{z_{end} - z_{start}} \]
\[ c_p = fc_f + (1 - f)c_s \]
Fog

- Exponential \( f = e^{-d_f z_p} \)
- Squared exponential \( f = e^{-(d_f z_p)^2} \)
- Exponential derived from Beer’s law

**Beer’s law**: intensity of outgoing light diminishes exponentially with distance
Fog Optimizations

- $f$ values for different depths ($z_p$) can be pre-computed and stored in a table on GPU
- Distances used in $f$ calculations are planar
- Can also use Euclidean distance from viewer or radial distance to create *radial fog*
Shadow Map Result
Fog example

- Fog is atmospheric effect
  - Better realism, helps determine distances
References

- Real Time Rendering by Akenine-Moller, Haines and Hoffman