Computer Graphics

Transparency and Shadows in OpenGL

Based on slides by Dianna Xu, Bryn Mawr College
Opacity and Transparency

- Opaque surfaces permit no light to pass through
- Transparent surfaces permit all light to pass
- Translucent surfaces pass some light

translucency = 1 – \( \alpha \)
Translucency Model

• Surface $s_1$ is translucent and only allows a fraction $t$ (transmittance) of the light reflected from surface $s_2$ (behind it) to pass through:

$$i = (1 - t)r_1 + t r_2$$

$t =$ transmittance of $s_1$
$r_1 =$ reflected light from $s_1$
$r_2 =$ reflected light from $s_2$

$t = 0 \Rightarrow i = r_1 \Rightarrow s_1$ is opaque
$t = 1 \Rightarrow i = r_2 \Rightarrow s_1$ is transparent
Blending Equation

- Source: the color of the polygon going to cover the pixel
- Destination: the color of original pixel in the frame buffer
- Blending factor is the same for all color components

\[ c = B_d * C_s + B_s * C_d \]
OpenGL Blending and Compositing

- Must enable blending and pick source and destination factors

  ```
  glEnable(GL_BLEND)
  glBlendFunc(source_factor, destination_factor)
  ```

- Only certain factors supported
  - `GL_ZERO, GL_ONE`
  - `GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA`
  - `GL_DST_ALPHA, GL_ONE_MINUS_DST_ALPHA`
  - See Redbook for complete list
Example

• Suppose that we start with the opaque background color \((R_0, G_0, B_0, 1)\)
  – This color becomes the initial destination color
• We now want to blend in a translucent polygon with color \((R_1, G_1, B_1, \alpha_1)\)
• Select \texttt{GL\_SRC\_ALPHA} and \texttt{GL\_ONE\_MINUS\_SRC\_ALPHA} as the source and destination blending factors
  \[
  \alpha_1 R_1 + (1- \alpha_1) R_0, \quad \alpha_1 G_1 + (1- \alpha_1) G_0, \quad \alpha_1 B_1 + (1- \alpha_1) B_0
  \]
• Note this formula is also correct if polygon is either opaque or transparent
Writing Model

- Use $\alpha$ component of RGBA (or RGB$\alpha$) color to store opacity
- During rendering we can expand our writing model to use RGBA values

![Diagram of writing model with source blending factor and destination blending factor connected to blend, which in turn is connected to the Color Buffer.](image-url)
Order Dependency

• Is this image correct?
  – Probably not
  – Polygons are rendered in the order they pass down the pipeline
  – Blending functions are order dependent
Translucency and HSR

• Suppose that we have a group of polygons some of which are opaque and some translucent
• How do we use hidden-surface removal?
• Opaque polygons block all polygons behind them and affect the depth buffer
• Translucent polygons should not affect depth buffer
  – Render with `glDepthMask(GL_FALSE)` which makes depth buffer read-only
• Sort polygons first to remove order dependency
Changing Transmittance by Viewing Angle
Applied to Lots of Small Polygons

MIRALABS, UNIVERSITY OF MONTREAL
Specular Highlights Reduce Transmittance
Fog

- Blend in distance-dependent color as each polygon is processed.
- Add a fog factor which is used much like alpha, but is dependent on depth, and blends between fog color and polygon color.

\[ c = f \cdot C_s + (1 - f) \cdot C_f \]

- \( f \) is the *fog factor*, and is calculated by a function on depth
  - Exponential
  - Gaussian
  - Linear (depth cueing)
OpenGL Fog

GLfloat fogc[4] = {...};
glEnable(GL_FOG);
glFogfv(GL_FOG_COLOR, fogc);
glFogf(GL_FOG_MODE, GL_EXP);
glFogf(GL_FOG_DENSITY, 0.5); // \( f = e^{-0.5z^2} \)
Fog Functions

![Graph showing attenuation vs distance with lines representing $e^{-z^2}$ and $1 - 0.5z$]
Fog or Haze
Foggy Chessmen
Transmittance Function: Varies over Interior of Ellipsoidal Shapes
Using a Denser Transmittance Function (or 1)
Line Aliasing

- Ideal raster line is one pixel wide
- All line segments, other than vertical and horizontal segments, partially cover pixels
- Simple algorithms color only whole pixels
- Lead to the “jaggies” or aliasing
- Similar issue for polygons
Antialiasing

- Can try to color a pixel by adding a fraction of its color to the frame buffer
  - Fraction depends on percentage of pixel covered by fragment
  - Fraction depends on whether there is overlap
Area Averaging

- Use average area $\alpha_1 + \alpha_2 - \alpha_1 \alpha_2$ as blending factor
OpenGL Antialiasing

```c
glEnable(GL_POINT_SMOOTH);
glEnable(GL_LINE_SMOOTH);
glEnable(GL_POLYGON_SMOOTH);
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```
Shadows

• Shadows are a consequence of light and substance.
• Shadows help depth perception and spatial relationships.
• Shadows can be hard- or soft-edged.
• Various shadow algorithms:
  – Finding silhouette edges and creating shadow polygons
  – Ray tracing
  – Radiosity
Shadow Volumes for Non-Ray-Traced Renderings

Light source

Silhouette or Contour edges: Normal flips from forward to backward facing polygons

Edges of semi-infinite shadow polygons
Clipping Shadow Volumes to View Pyramid to Form Shadow Polygons

- Point light source
- Silhouette edge polygon
- Semi-infinite shadow volume produced by polygon
- Intersection of semi-infinite shadow volume with view volume
- View volume
Clipped Shadow Volume Polygons Added to Scene
E.g., these polygons (exploded view):
Shadows Add Depth and Spatial Relationships
Shadows Help Show Contacts
Shadows Show Us Where Lights Might Be
Shadows Accent Otherwise Like Color Features
Sunbeams as Polyhedral Channels and Light Density
Light and Shadow over the Edge of Believability
Soft Shadows  (Before Radiosity)

© 1982 NAKAME, E. & NISHITA, T. & OKAMURA, I.