Transparency

- Blending colors
- Frame buffer limitations
- Compositing
Basic Alpha Blending

Transparent front color with opacity $\alpha$

Solid back color

$$rgb = \alpha \cdot rgb_f + (1 - \alpha) \cdot rgb_b$$
Basic Alpha Blending

Transparent front color with opacity $\alpha$

Solid back color

$$rgb = \alpha_1 \cdot rgb_1 + (1 - \alpha_1) \cdot [\alpha_2 \cdot rgb_2 + (1 - \alpha_2) \cdot rgb_3]$$

$$rgb = \alpha_1 \cdot rgb_1 + (1 - \alpha_1) \cdot \alpha_2 \cdot rgb_2 + (1 - \alpha_1) \cdot (1 - \alpha_2) \cdot rgb_3$$
Blending Alphas

Transparent front color with opacity $\alpha_f$  
Transparent back color with opacity $\alpha_b$

$$rgb = \alpha_f \cdot rgb_f + (1 - \alpha_f) \cdot \alpha_b \cdot rgb_b$$

$$\alpha = \alpha_f + (1 - \alpha_f) \cdot \alpha_b$$

Process in either front to back or back to front order
But can’t insert surface between them later - why?
Premultiply alpha

Transparent front color with opacity $\alpha_f$  
Transparent back color with opacity $\alpha_b$

\[
\text{rgb} = \alpha_f \cdot \text{rgb}_f + (1 - \alpha_f) \cdot \alpha_b \cdot \text{rgb}_b
\]

\[
\alpha = \alpha_f + (1 - \alpha_f) \cdot \alpha_b
\]

Notice - rgb always appears multiplied by it's $\alpha$
Can store rgb's as \textit{premultiplied} by $\alpha$. 
- What about z value in depth buffer?

- Manage z-values: glDepthMask(GL_FALSE)
Transparency in OpenGL

```c
setEnabled(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

Draw polygon that has alpha values set

setEnabled(GL_BLEND);
```
“Screendoor” transparency

- No blend - use alternating pixels from different surfaces

What about z value in depth buffer?
Stipple in OpenGL

Uses 32x32 pattern

```c
-GLubyte halftone[] = {
    0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55
    ...repeat 16 lines
}

glEnable(GL_POLYGON_STIPPLE);
glPolygonStipple(halftone);

draw polygon

glDisable(GL_POLYGON_STIPPLE);
```
Compositing

- Without pixel z values
- With pixel z
  - use depth at pixel corners to interpolate partial coverage
Compositing

Z buffer: keep z values with color buffer
1. Compare z values at corresponding pixels
2. Keep all or nothing

<table>
<thead>
<tr>
<th></th>
<th>6</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>9</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Compositing

Z buffer: keep z values with color buffer
1. Compare z values at corresponding pixels
2. Compute partial coverage:
   Interpolate corner z values
   Compare corner values for pixel and blend

![Diagram with Z buffer values]
Compositing

**Alpha Channel**
Value between 0 and 1
Combined partial coverage and transparency
Computed during rendering in front of a null background

2 1/2 D blend based on alpha of image in front

<table>
<thead>
<tr>
<th>RGB</th>
<th>alpha</th>
</tr>
</thead>
</table>

32 bit pixel values
Compositing - example