Alpha Blending

The Over Operator

Compositing of two colors with alpha values can be accomplished using the over operator:

\[ C_{A \over B} = \alpha_A C_A + (1 - \alpha_A) \alpha_B C_B \]

\[ \alpha_{A \over B} = \alpha_A + (1 - \alpha_A) \alpha_B \]

1. Blending

Suppose \( C_A = (0.5, 0.5, 1.0, 0.25) \)
\( C_B = (0.25, 0.25, 0.25, 0.75) \)

Compute \( C_A \over C_B \) assuming post-multiplied alpha.
\[
\frac{1}{4} \left( \frac{1}{2}, \frac{1}{2}, 1 \right) + \frac{3}{4} \left( \frac{1}{4}, \frac{1}{4}, 1 \right) = \left( \frac{8}{64}, \frac{8}{64}, \frac{16}{64} + \frac{9}{64}, \frac{9}{64}, \frac{9}{64} \right) = \left( \frac{17}{64}, \frac{17}{64}, \frac{25}{64} \right)
\]

2. Pre-multiplied Alpha

a. If \( C_A = (0.125, 0.125, 0.5, 0.5) \) uses pre-multiplied alpha, what is the equivalent color using non-pre-multiplied alpha?
\[
\frac{1}{1/2} \left( 0.125, 0.125, 0.5, 0.5 \right) = \left( 0.25, 0.25, 1.0 \right)
\]

b. In WebGL, which \texttt{gl.blendFunc} parameters correspond to using pre-multiplied alpha in your blending? You can consult the table on the back for a list of blending parameters.
\texttt{gl.blendFunc(gl.ONE,gl.ONE_MINUS_SRC_ALPHA)}
3. **Alpha Blending in WebGL**

Alpha blending in WebGL most commonly uses the function call:

```
gl.blendFunc(GLenum sfactor, GLenum dfactor)
```

where the final color is generated by as

\[
Cs \over d = sfactor * Cs + dfactor * Cd
\]

Possible values for `sfactor` and `dfactor` are:

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>RGB BLEND FACTORS</th>
<th>ALPHA BLEND FACTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>gl.ZERO</td>
<td>(0, 0, 0)</td>
<td>0</td>
</tr>
<tr>
<td>gl.ONE</td>
<td>(1, 1, 1)</td>
<td>1</td>
</tr>
<tr>
<td>gl.SRC_COLOR</td>
<td>(R_s, G_s, B_s)</td>
<td>A_s</td>
</tr>
<tr>
<td>gl.ONE_MINUS_SRC_COLOR</td>
<td>(1, 1, 1) - (R_s, G_s, B_s)</td>
<td>1 - A_s</td>
</tr>
<tr>
<td>gl.DST_COLOR</td>
<td>(R_d, G_d, B_d)</td>
<td>A_d</td>
</tr>
<tr>
<td>gl.ONE_MINUS_DST_COLOR</td>
<td>(1, 1, 1) - (R_d, G_d, B_d)</td>
<td>1 - A_d</td>
</tr>
<tr>
<td>gl.SRC_ALPHA</td>
<td>(A_s, A_s, A_s)</td>
<td>A_s</td>
</tr>
<tr>
<td>gl.ONE_MINUS_SRC_ALPHA</td>
<td>(1, 1, 1) - (A_s, A_s, A_s)</td>
<td>1 - A_s</td>
</tr>
<tr>
<td>gl.DST_ALPHA</td>
<td>(A_d, A_d, A_d)</td>
<td>A_d</td>
</tr>
<tr>
<td>gl.ONE_MINUS_DST_ALPHA</td>
<td>(1, 1, 1) - (A_d, A_d, A_d)</td>
<td>1 - A_d</td>
</tr>
</tbody>
</table>

Suppose alpha blending is enabled using the over operator and the following fragments are processed in the given order for the same pixel location. What is the final color?

Assume:
1. Framebuffer is set to (0,0,0,1) at all pixel locations.
2. Depth testing is on
3. We are using pre-multiplied alpha

Fragment 1: (0.5, 0.5, 0.5, 0.25) with a depth value of 0.25
Fragment 2: (0.25, 0.25, 0.25, 0.25) with a depth value of 0.25
Fragment 3: (0.125, 0.125, 0.125, 0.25) with a depth value of 0.5
(Hint: WebGL depth buffer assumes smaller being closer to the viewer)

Only the first fragment passes the depth test, so we have:

\[(0.5,0.5,0.5,0.25) + (1 - 0.25)(0,0,0,1) = (0.5,0.5,0.5,1.0)\]