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

Compute \( C_A \over C_B \) assuming NON-pre-multiplied alpha.

2. Pre-multiplied Alpha

   a. If \( CA = (0.125, 0.125, 0.5, 0.5) \) uses pre-multiplied alpha, what is the equivalent color using non-pre-multiplied alpha?

   b. In WebGL, which 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.
3. Alpha Blending in WebGL

Alpha blending in WebGL most commonly uses the function call:

\texttt{gl.blendFunc(GLenum sfactor, GLenum dfactor)}

where the final color is generated by

\[Cs_{over\_d} = sfactor \cdot Cs + dfactor \cdot Cd\]

Possible values for \texttt{sfactor} and \texttt{dfactor} are:

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>RGB \begin{tabular}{c}BLEND FACTORS\end{tabular}</th>
<th>ALPHA \begin{tabular}{c}BLEND FACTOR\end{tabular}</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{GL.ZERO}</td>
<td>\begin{tabular}{c}(0, 0, 0)\end{tabular}</td>
<td>0</td>
</tr>
<tr>
<td>\texttt{GL.ONE}</td>
<td>\begin{tabular}{c}(1, 1, 1)\end{tabular}</td>
<td>1</td>
</tr>
<tr>
<td>\texttt{GL.SRC_COLOR}</td>
<td>\begin{tabular}{c}(R_s, G_s, B_s)\end{tabular}</td>
<td>\begin{tabular}{c}A_s\end{tabular}</td>
</tr>
<tr>
<td>\texttt{GL.ONE_MINUS_SRC_COLOR}</td>
<td>\begin{tabular}{c}(1, 1, 1) - (R_s, G_s, B_s)\end{tabular}</td>
<td>\begin{tabular}{c}1 - A_s\end{tabular}</td>
</tr>
<tr>
<td>\texttt{GL.DST_COLOR}</td>
<td>\begin{tabular}{c}(R_d, G_d, B_d)\end{tabular}</td>
<td>\begin{tabular}{c}A_d\end{tabular}</td>
</tr>
<tr>
<td>\texttt{GL.ONE_MINUS_DST_COLOR}</td>
<td>\begin{tabular}{c}(1, 1, 1) - (R_d, G_d, B_d)\end{tabular}</td>
<td>\begin{tabular}{c}1 - A_d\end{tabular}</td>
</tr>
<tr>
<td>\texttt{GL.SRC_ALPHA}</td>
<td>\begin{tabular}{c}(A_s, A_s, A_s)\end{tabular}</td>
<td>\begin{tabular}{c}A_s\end{tabular}</td>
</tr>
<tr>
<td>\texttt{GL.ONE_MINUS_SRC_ALPHA}</td>
<td>\begin{tabular}{c}(1, 1, 1) - (A_s, A_s, A_s)\end{tabular}</td>
<td>\begin{tabular}{c}1 - A_s\end{tabular}</td>
</tr>
<tr>
<td>\texttt{GL.DST_ALPHA}</td>
<td>\begin{tabular}{c}(A_d, A_d, A_d)\end{tabular}</td>
<td>\begin{tabular}{c}A_d\end{tabular}</td>
</tr>
<tr>
<td>\texttt{GL.ONE_MINUS_DST_ALPHA}</td>
<td>\begin{tabular}{c}(1, 1, 1) - (A_d, A_d, A_d)\end{tabular}</td>
<td>\begin{tabular}{c}1 - A_d\end{tabular}</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 0.5
Fragment 3: (0.125, 0.125, 0.125, 0.25) with a depth value of 0.9
(Hint: WebGL depth buffer assumes smaller being closer to the viewer)