Questions

1. Show that the "over" operative is associative. In particular, if F, M, B are premultiplied $rgb\alpha$ images where F is foreground, M is middle, B is background, then

$$F \text{ over } (M \text{ over } B) \;=\; (F \text{ over } M) \text{ over } B.$$

   Warning: the proof is just turning the crank. The point here is for you to make sure you know what needs to be proved i.e. you understand the definitions.

2. Suppose we have five pixels ($x = 1, \ldots, 5$) that are contained in two images F and B. Let the F_α values for these five pixels be $(0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1)$ and let the B_α values for these five pixels be $(1, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0)$. What are the α values of F over B ?

3. "Blue screen matting" uses a blue opaque screen for the background, i.e. $B_{rgb\alpha} = (0, 0, 1, 1)$. Suppose we were to restrict our foreground scenes so that they had zero blue component, i.e. $F_{rgb} \approx (F_r, F_g, 0)$. Show that these conditions provides sufficient information for us to estimate $F_{rgb\alpha}$, if we are given F over B.

4. When we use the term "pre-multiplied", we mean that the RGB values have been multiplied by α as in:

$$(F \text{ over } B)_{rgb} = F_\alpha F_{RGB} + (1 - F_\alpha) B_\alpha B_{RGB}$$

   Suppose we used the $I_{RGB\alpha}$ representation for our layers instead of the pre-multiplied representation i.e. $I_{rgb\alpha}$.
   What would be the formula for $(F \text{ over } B)_{RGB}$?

5. Estimating alpha for two layers in a single given image $(F \text{ over } B)_{rgb}$ impossible since, at each pixel, you have more unknown values than known values. How many unknowns? How many knowns? Why? (You may assume the background is opaque, i.e. $B_\alpha = 1$.

6. In OpenGL, one can draw opaque objects in any order and depth buffering automatically handles visibility. However, when there are transparent objects present, depth buffering doesn't make sense because multiple objects can be visible along any line of sight.

   Suppose a scene consists of some opaque objects and some partially transparent ones. How could could we draw the objects so that visibility is handled correctly? One idea is to sort all the objects from far to near as in the painter's algorithm and then draw from back to front with depth buffering disabled. Can you think of an alternative solution?

7. Suppose you wanted to blend four images so that each one, when viewed on its own, would be 25 per cent as bright as the original. That is, you want to average the four images. How would you do this using alpha blending in OpenGL ? How would you set the source and destination factors.

Answers

1. In the following derivation, any expression that does *not* have an α subscript really should have a $rgb\alpha$ subscript. For example, (F over M) refers to (F over $M)_{rgb\alpha}$. I leave out these extra subscripts to keep it simple. See me if you don't understand.

$$\begin{aligned}
& (F \text{ over } M) \text{ over } B \\
= \; & (F + (1 - F_\alpha)M) \text{ over } B \\
= \; & (F + (1 - F_\alpha)M) \; + (1 - (F + (1 - F_\alpha)M)_\alpha) \; B \\
= \; & F + (1 - F_\alpha)M \; + (1 - (F_\alpha + (1 - F_\alpha)M_\alpha)) \; B \\
= \; & F + (1 - F_\alpha)M \; + (1 - F_\alpha)B - (1 - F_\alpha)M_\alpha \; B \\
= \; & F + (1 - F_\alpha)M \; + (1 - F_\alpha)(1 - M_\alpha) \; B
\end{aligned}$$

   Similarly,

$$\begin{aligned}
& F \text{ over } (M \text{ over } B) \\
& F \; + \; (M \; + (1 - M_\alpha) \; B) \\
& F \; + \; (1 - F_\alpha)(M \; + (1 - M_\alpha) \; B) \\
= \; & F + (1 - F_\alpha)M \; + (1 - F_\alpha)(1 - M_\alpha) \; B
\end{aligned}$$

2. Using the formula $(F \text{ over } B)_\alpha = F_\alpha + (1 - F_\alpha)B_\alpha$, we get

$$(F \text{ over } B)_\alpha = (1, \frac{13}{16}, \frac{3}{4}, \frac{13}{16}, 1).$$

3. We have

$$(F \text{ over } B)_b = F_b + (1 - F_\alpha)B_b = (1 - F_\alpha)$$

   since $B_b = 1$ and $F_b = 0$. The left side is measured, so we immediately get F_α. This is good enough since

$$(F \text{ over } B)_{rg} = F_{rg} + (1 - F_\alpha)B_{rg} = F_{rg}$$

   since $B_{rg} = 0$, so we can get F_r and F_g as well.

4. Use $(F \text{over} B)_\alpha$ gives the alpha of the resulting image. To find the non-pre-multiplied RGB color, take the premultiplied color and divide it by $(F \text{over} B)_\alpha$. Dividing by the alpha of the combined image gives the RGB color that a single layer would need to have in order to have that rgb premultiplied color.

$$\frac{F_\alpha F_{RGB} + (1 - F_\alpha)B_\alpha B_{RGB}}{F_\alpha + (1 - F_\alpha)B_\alpha}$$

   Notice that the new quantity doesn't correspond to anything meaningful, since there is no actual surface that has this RGB values.

5. We have 8 unknowns in general at each pixel: $F_{rgb\alpha}$ and $B_{rgb\alpha}$. The background is assumed to have $\alpha = 1$, so this drops us down to 7 unknowns. We have only three knowns, namely $(F \text{over} B)_{rgb}$. We cannot solve for 7 uknowns given only 3 knowns.

6. We could partition objects into two sets: opaque and non-opaque. We could draw the opaque ones with depth buffering enabled, which would allow us to draw them in any order. We could then disable the depth buffering, sort the non-opaque as we would with the painter's algorithm and then draw them from far to near. Before drawing each non-opaque object at a pixel, the fragment shader would need to check the depth buffer and make sure the non-opaque object (source) being drawn doesn't have a greater depth than the destination z value (in the depth buffer), since in that case it should be occluded by the opaque object and hence not visible.

7. Make each of the images have an alpha channel of 0.25 for all pixels. (Note the image RGB values would *not* be premultiplied.) Call `glBlend(GL_SRC_ALPHA, GL_ONE)` . This sets the source factor to `GL_SRC_ALPHA` and it sets the destination factor to `GL_ONE`. Then just draw the four images.