Q: There seems to be a problem with compositing. Using the CompositeLab in /NextDeveloper/Examples on a 2-bit screen, set the following parameters and use the SOVER operation:

Source gray=1 opacity=0.3 (white mostly transparent source)
Dest gray=0.8 opacity=1 (light gray opaque destination)

You will see that the result is the same color as the destination. Thus the 30% coverage from the white source is having no effect at all! Now change the source opacity to 0; this causes no change in the result. What's going on here?

A: The behavior is correct. Assume the case where the source is all white and is 33% opaque. Say the destination is 66% white and opaque. (This assures that we are using exact pixel values with no dithering.) The SOVER formula is:

sourceColor * sourceOpacity + destColor * (1 - sourceOpacity)

which for our case reduces to

1 * 1/3 + 2/3 * 2/3 = 7/9

which is rounded to 6/9, given that compositing only works on a per-pixel basis. Changing the opacity all the way down to 0 simply changes the resulting pixel to 6/9, thus no color change occurs.

Now using the parameters in the question

Source gray=1 opacity=0.3 (white mostly transparent source)

Dest gray=0.8 opacity=1 (light gray opaque destination)

We see that there are some source pixels with opacity of 0, and a few with opacity of 1/3. Source color is 1 in all cases. There are some dest pixels with gray of 1 and others with 2/3; opacity is 1 in both cases.

Thus every resulting pixel is computed from one of four formulas:

sourceColor * sourceOpacity + destColor * (1 - sourceOpacity)							
1	*	0	+	1	*	(1 - 0)	= 1
1	*	1/3	+	1	*	(1 -1/3)	= 1
1	*	0	+	2/3	*	(1 -0)	= 2/3
1	*	1/3	+	2/3	*	(1 -1/3)	= 7/9

Thus the result is equal to the dest color in all cases.

If you wish to composite in a more accurate fashion, you can use 8-bit deep grayscale windows. However, this will use up a lot more memory and is probably not worth it.

QA712

Valid for 2.0, 3.0