Displaying 2D Spatial Variables with Alpha-Blending and Bump-Mapping

The DDS visualization technique displays single-valued functions $F_k(i,j)$ with two different computer graphics methods: *alpha-blending* and *bump-mapping.* Each function, or *variable*, is displayed in a separate graphical layer, and the final visualization is a multi-layer image. Figure 2.16 presents four variable layers from the SEM data set, two are displayed with DDS alpha-blending and two are displayed with DDS bump-mapping. The layers show amounts iron, magnesium, sulfur, and silicon in the sample. The amount of iron is displayed with red spots, magnesium is displayed with blue spots, sulfur is shown with the larger bumps and silicon is displayed with the smaller bumps.

Below I describe the implementation of each method, including my observations of the effectiveness of each at displaying data. In Chapter Five I further explore the efficacy of the system. There I pose several questions regarding how well the data can be seen and understood with DDS alpha-blending and bump-mapping, as well as present some exploratory images and thoughts concerning how to find the answers to the questions I put forth.

DDS Alpha-Blending

Sampling Variables with the Gaussian Sampling Array -- Alpha-Blending

The 2D spatial variable is sampled by computing the scalar product of the variable image and the Gaussian array for each point (i,j) in the variable image. The product image is a variable-modulated version of the Gaussian array. Data nearer the center of a Gaussian is scaled by a value that is near one and is represented more faithfully; data further out is sampled with a low value and contributes little to the final image. Examples of variable-modulated spot arrays are shown in Figure 2.17.

Color-Model

I describe DDS alpha-blending within the OpenGL framework [Kempf and Frazier, 1997]. OpenGL is not the only platform for implementing DDS, which was originally implemented on the PixelFlow graphics computer [Eyles, Molnar, Poulton, Greer, Lastra, England, and Westover, 1998]; however, OpenGL provides a standard language for describing the computer graphics principles used.

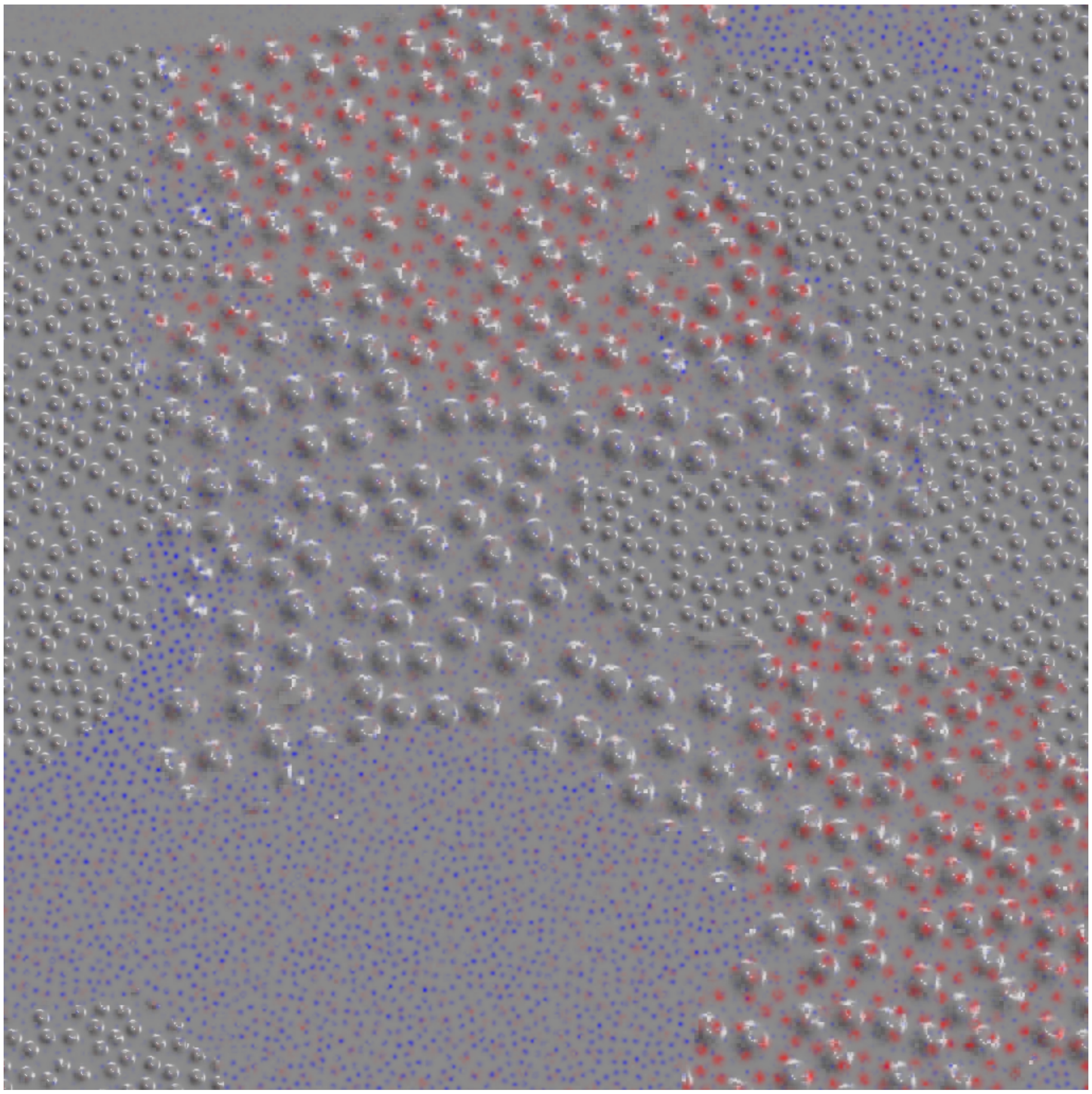


Figure 2.16: A DDS image of SEM data from a geological sample. The layers show amounts iron, magnesium, sulfur, and silicon in the sample. The amount of iron is displayed with red spots, magnesium is displayed with blue spots, sulfur is shown with the larger bumps and silicon is displayed with the smaller bumps.

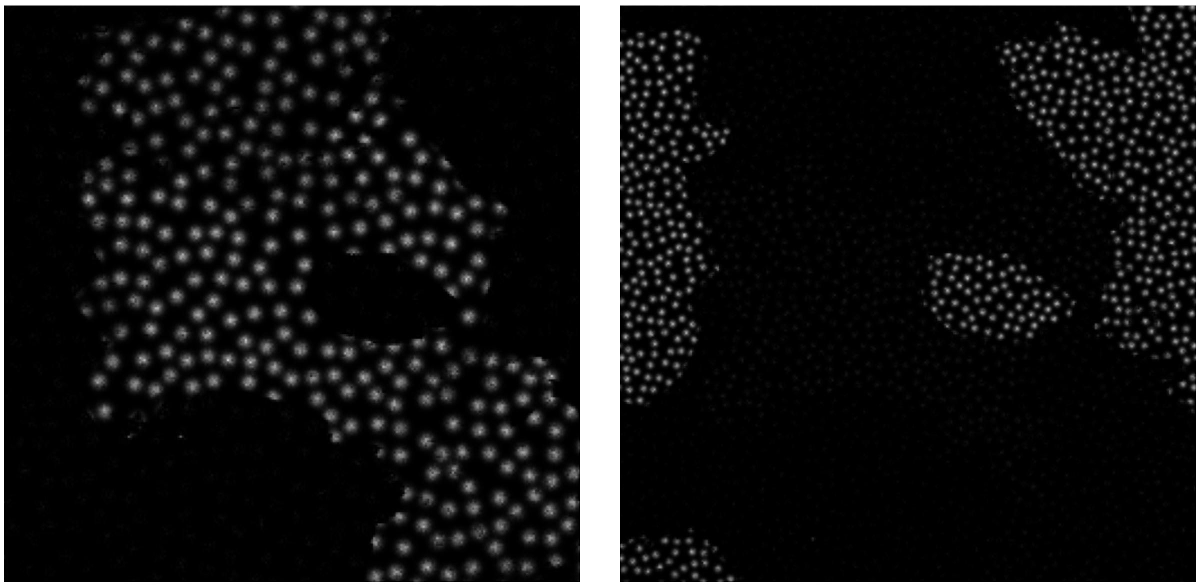


Figure 2.17: Variable-modulated spot arrays for the sulfur and silicon data shown in Figure 2.16.

Colors are represented in OpenGL with the RGBA (red, green, blue, alpha) color model. In the RGBA color model colors have four components: Red, Green, Blue, and alpha. Although OpenGL uses RGBA, for the purposes of this chapter the Hue, Lightness, and Saturation, (HLS) color model provides a more intuitive way to understand how DDS uses color. RGBA is easily translated to the HLS color model, and the two models can be considered interchangeable.

Hue is what we typically think of as color, for example, red as opposed to blue or yellow. In the HLS color model the values for hue can be thought of as a color circle which begins with red and passes through green, blue and all other hues such as blue-green, cyan, orange, and purple. Lightness represents the perceived reflectance of a surface [Ware, 2000]. A lightness value of 0 appears as black, and a value of 1 appears as bright white and washes out the hue. Saturation is defined as how brilliant or faded the color appears. The values range from no color saturation, i.e. a neutral gray, to full saturation. Saturation can be thought of as the difference between light pink and hot pink, or between pastels and jewel tones.

The HLS color model is defined on a double-hexicon (see [Foley, van Dam, Feiner, and Hughes, 1990] for full details). The vertical axis of the hexicon is lightness with 0 for black at the tip of the lower hexicon and 1 for white at the tip of the upper hexicon. Saturation is the horizontal distance from the vertical axis. Hue is the angle around the vertical axis with red at 0°, yellow at 60°, green at 120°, cyan at 180°, blue at 240°, and magenta at 300°.

Many data visualization techniques use hue to convey quantitative information about a single variable; some examples are discussed in Chapter Four. One common example is a weather map where levels of rainfall are displayed as different colors and typically range from light green, which indicates low levels of rain accumulation, through red, yellow, and purple. Figure 2.18 shows an example of Doppler radar data from the National Oceanic and Atmospheric Administration [NOAA, 2003]. The color scale to the left of the image shows the mapping of data value to color in dBZ (decibels of Z), which is a measurement of the different reflectivity values measured during each elevation scan. A dBZ value of 65 represents rainfall in the amount of 16+ inches per hour; a value of 60 represents 8 inches/hour; 55 is 4 inches/hour, 40 is 0.5 inches/hour, and 20 represents trace amounts of rainfall.

A problem with using different hues to indicate different amounts of a single variable is that the viewer must learn what the colors mean, and he or she must constantly refer to the legend – it is very difficult to devise a color map where the changes in hue are perceived correctly by intuition alone. The image on the following page is a prime example. The yellow regions are the brightest and most salient, but arguably less important than the orange and red regions, which represent larger amounts of rainfall. The blackbody color map displays color transitions that are interpreted correctly from intuition, however, the blackbody color map changes not just in hue but monotonically increases in lightness as the data values increase, and one could argue that the increasing intensity is a necessary and dominant cue over hue. Using color in visualizing data is a challenge – in everyday usage red does not mean more than blue and less than yellow – they are more commonly used to describe or to differentiate among separate objects (blue sky, red apple, and yellow school bus). Most color maps force the viewer to use hue in an uncommon, perhaps unnatural, way. An alternative to hue-varying color maps are color maps that vary in saturation or lightness with changes in the data, suggested in [Rogowitz and Treinish, 1998].

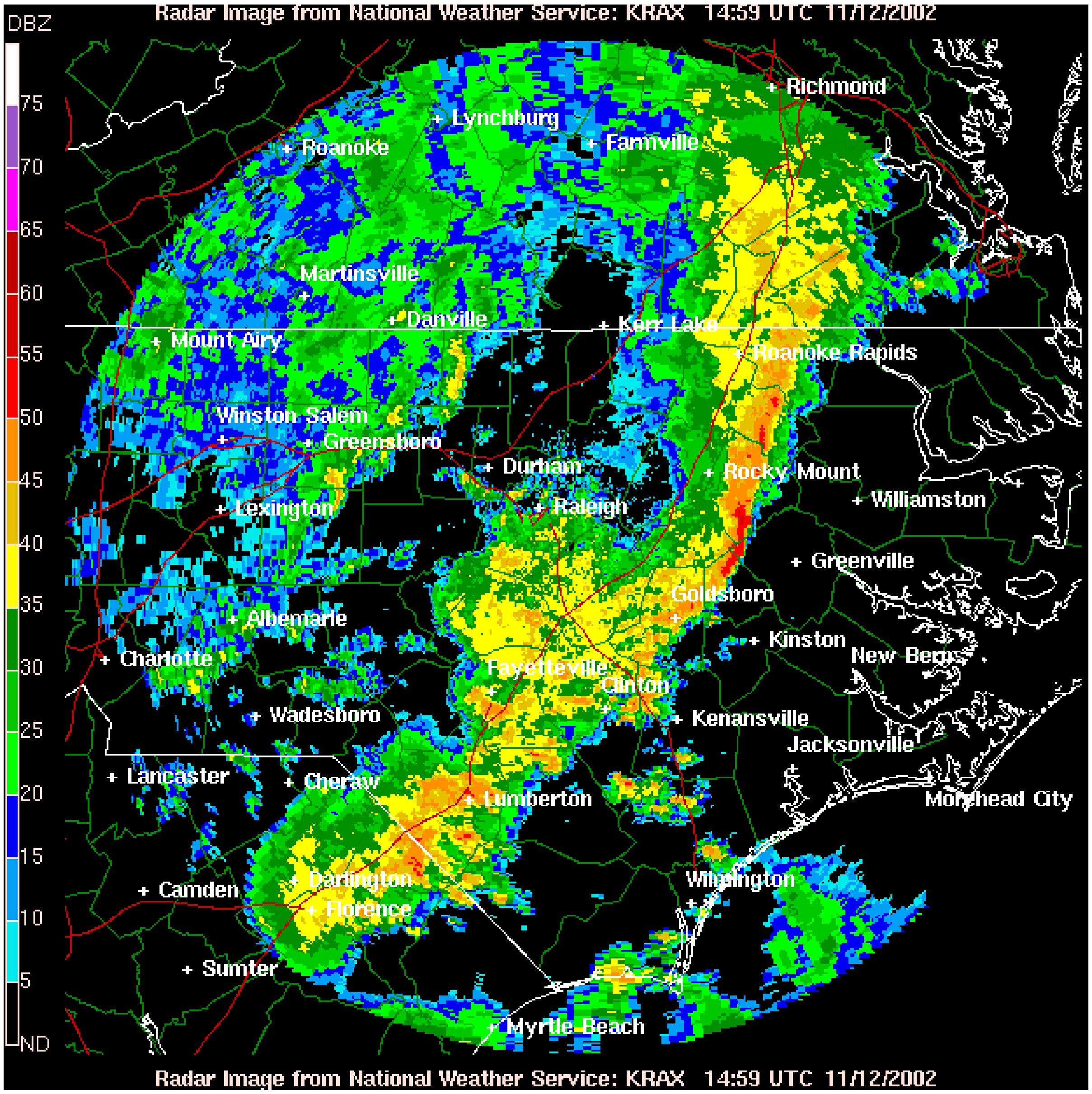
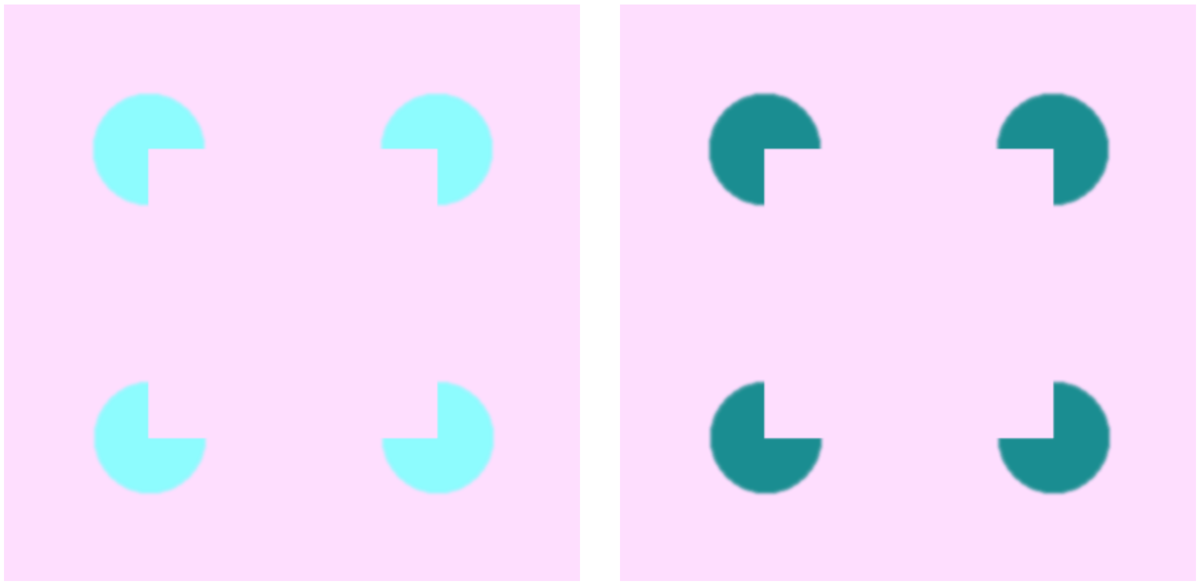


Figure 2.18: Weather data from the National Oceanic and Atmospheric Administration, National Weather Service, [NOAA, 2003] over North Carolina for November 12, 2002. Hue represents data values, with red representing the most rainfall at 4 inches per hour and cyan representing trace rainfall amounts.

Another problem Rogowitz and Treinish [1998] describe for the hue-based rainbow color map in that yellow regions are perceived as brighter and therefore as more important than other colors, even though the underlying data may not be of practical interest. They suggest different hues are best used for nominal data, and Ware agrees: "*Color is often extremely effective as a nominal code. When we wish to make it easy for someone to classify visual objects into separate categories, giving the objects distinctive colors is often the best solution.*" [p. 133, 2000]. The trend in data visualization is following everyday life, where most often hue differentiates among objects, and this is how hue is used in the DDS visualization technique. DDS uses hue to differentiate between variable layers; hue does not convey quantitative information, but qualitative – which variable is being displayed.

Choosing colors to distinguish between DDS layers is an interesting realm of investigation in its own right. There is not a good definition of equal when it comes to colors – one definition is equal perceived brightness, which is problematic in that it is viewer-dependent. When intensity or perceived brightness is used to convey data, it is reasonable to think that it is important to start with colors that all appear the same for a given reference point, however there is evidence that perceptually isoluminant colors are not the best choice. They are not easily distinguishable and lose coherence of motion; figure and ground are harder to separate when both are displayed with isoluminant colors; perceptual illusions are not seen as well with isoluminance [Hoffman 1998, Livingstone and Hubel 1988]. Figure 2.19 shows an example of the illusory square, which is less effective when displayed with perceptually-isoluminant colors.

In DDS hue is used to categorize or label different variables. Amount of transparency is used to display value information about those variables. When a layer is blended with the gray background it appears as if the data is displayed with levels of saturation. An alpha value of zero, which translates to a completely transparent region, is seen as a neutral gray and an alpha value of one, which translates to a completely opaque region, is seen as a fully saturated color. This is true only for the first layer applied, however, as the second and additional layers are blended with both the gray background and colors from previous layers.



(a) **(b)**

Figure 2.19: Which illusory square is easier to see? One displayed with perceptually isoluminant colors (a)? Or one displayed with perceptually non-isoluminant colors (b)? The background pink is the same for both images: HLS values: 220, 240, 180. The circle inducers are cyan in both images, but brighter in (a) than in (b): HSL values: 120, 240, 180 for (a) and 120, 240, 60 for (b).

Displaying Alpha-Blended Layers

Layers are added one at a time to the underlying gray background surface through an iterative process. The color for the first layer is blended with the gray background using standard computer graphics blending techniques, described in the next paragraph. Each additional layer is blended with the resulting image derived from the layers that have already been applied. The accumulation image, $Accum_k(i,j)$, is the surface the new layer is added to; it may be either the original gray background image before the first layer is applied, or an image with multiple layers. The blending function is the same regardless of how many layers are already incorporated in the accumulation image.

The alpha value for each layer, $\alpha_k(i,j)$, is the product of the function $F_k(i,j)$ multiplied by the Gaussian sampling array $G_k(i,j)$, for each grid point, i,j. The alpha value is then used to blend the label color for the layer, C_k, onto the visualization image. Each layer is blended in succession onto the visualization image; the result is stored in an accumulation image. The blending process for layer k is shown in equation 2.3.

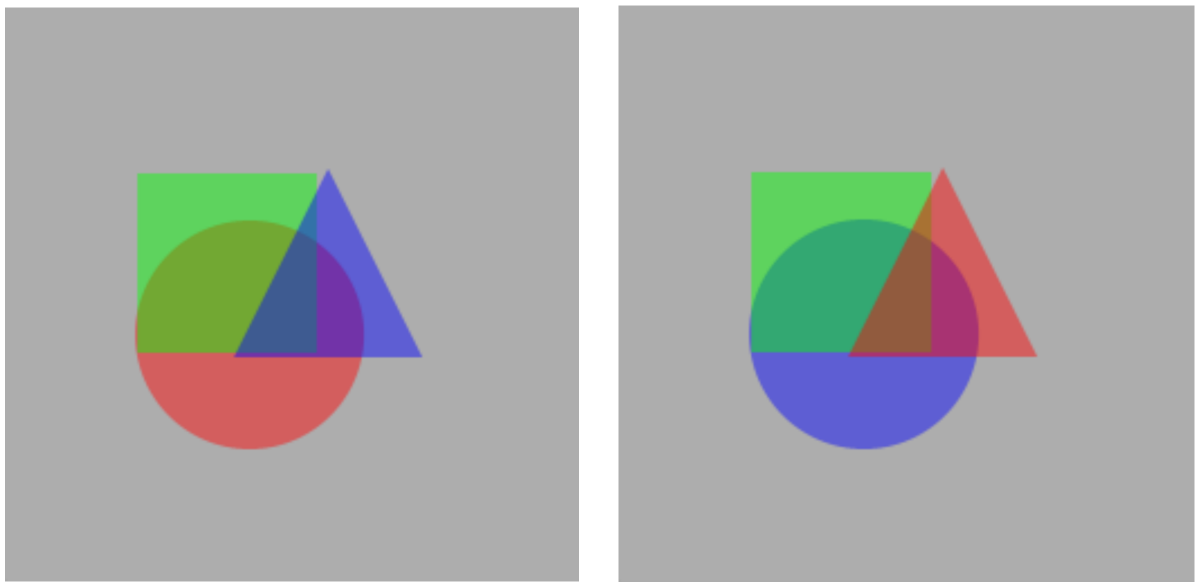

(a) (b)

Figure 2.20: The two images illustrate how blending is done in DDS. In both images there are three layers: a circle on the bottom layer, a square on the middle layer, and a triangle on the top layer. In (a) the circle is red, the square is green, and the triangle is blue. In (b) the circle is blue, the square green, and the triangle red. Although each layer has a value of 50% alpha the color of the overlapping areas is different in the two images – this difference is necessary to show the order of the layers, without it it would be impossible to know which shape was on top.

$$Accum_k(i,j) = (1 - \alpha_k(i,j)) * Accum_{k-1}(i,j) + \alpha_k(i,j) * C_k \qquad 2.3$$

When alpha is equal to 1.0 the final color is based solely on the new layer; when alpha is equal to 0 the final color is based solely on the current image color. An alpha value of 0.5 produces an equal blend between the two colors. The blend function produces different results based on the order layers are applied. For example two layers, one red and one blue, both with 50% transparency, will produce different final RGB values depending on which is blended with the gray background first. This difference is necessary, as without it there would be no way to determine which layer is on top. Figure 2.20 (a-b) illustrates this; in both images the circle lies under the square, which in turn lies under the triangle.

If all the spots within a layer appear to be at the same depth in terms of layer order, then they maintain a more coherent figure for perceptual grouping. I believe that the more perceptual cues that support perceptual grouping can be tied to a layer, the better the visual discrimination of that layer. The spots within a layer all have the same color, size, apparent depth (layer order), and, for animated layers, the same direction and speed of motion – all these cues support perceptual grouping.   The perception that all the spots within a layer have the same depth, may be as important or vital a cue for discriminating layers.

Because the final visualization image is different depending on which layers are blended with the gray background first, it is important to be able to interactively manipulate the layer depth – to try different layer orders to get the best final image.

A drawback of DDS alpha-blending is there is no natural way to display negative values with transparency. The alpha-blended display for variables that have both positive and negative values must scale negative values so that they are positive.