

Chapter 8

Digital Compositing

8.1 Digital Compositing

In 1984, Porter and Duff wrote a paper which has become a classic. They were interested in what happens when digital images are overlaid in various ways, a process called image *compositing*. Digital compositing has become increasingly important in film and television post-production work. For example, the rendering cost of full 3D animation is too high to permit interactive adjustment of lighting. Instead, 3D components are rendered to images and the images are layered together, with various adjustments being made within the compositor in several passes. Even without graphics special effects, many movies will have used compositing. Blue-screen techniques depend on it. A compositor thus sits at the core of most image manipulation pipelines.

Compositing pre-dates computers. Early cinematographers would use in-camera compositing, also called double exposure, to bring together in one scene elements which were separately filmed. This soon developed into separate filming of the elements, which were then optically combined by printing both films onto fresh film stock. Similarly, animators would draw different elements of the scene, background and perhaps several foreground elements, on transparent sheets of celluloid. These were then stacked together and a frame of film exposed. (Celluloid is unstable and acetate sheets are used today, though they are still called “cels”.) Digital compositing is the descendent of these various film-related crafts.

We will now discuss digital compositing, based on Porter and Duff’s paper. It uses a novel and simple idea but uses it well. For those interested in the original paper and others closely related to it, here is a short list.

Papers

Papers with Bath authors are available via:

<http://www.cs.bath.ac.uk/pubdb/>

1. Bruce Wallace,
 “Merging and Transformation of Raster Images for Cartoon Animation”,
 ACM Computer Graphics, 15 (3), Aug 1981, pp. 253-261,
 (Proceedings of ACM SIGGRAPH 81)
 This paper introduced the idea of layering digital images together to make the Flintstones television cartoon series.
2. Thomas Porter and Tom Duff,
 “Compositing Digital Images”,
 ACM Computer Graphics, 18 (3), July, 1984, pp. 253-259,
 (Proceedings of ACM SIGGRAPH 84)
 The key paper where all the main ideas came together.
3. Robert J Oddy and Philip J Willis,
 “A Physically Based Colour Model”,
 Computer Graphics Forum, 10 (2), 1991, pp. 121-127.
 A more advanced approach simulating transparency and lighting effects.
4. Philip Willis
 “Projective Alpha Colour”,
 Computer Graphics Forum, 25 (3) (Conference Issue), 2006, pp. 557-566.
 Shows that projective geometry and alpha colours are the same thing, that this explains all known effects and that many additional effects are possible.

8.2 The Alpha Channel Model

In computer graphics, it is very common to represent colours with RGB values. Typically each of these colour “channels” requires 8 bits on a PC (12-16 bits or floating point for professional applications). Porter and Duff described the use of a fourth channel, which they called alpha, to represent the opacity of the colour. An alpha value of 255 represents completely opaque while a value of zero represents completely clarity. They used RGBA to indicate the four channel model. (You will also see the correct letter α used, instead of A.) For convenience, it is often simpler to think of each channel being in the range $[0.0, 1.0]$, rather than $[0, 255]$, then we can ignore the number of bits.

Suppose you want to overlay a foreground element, such as a person, on a background such as street scene. You start with an image of the person, which image will have been taken against its own background. If you can arrange its background pixels to have zero alpha and the person’s pixels to have value 1.0, then we can easily blend the two images together in proportion to this alpha. Where the person appears, the output image has person. Elsewhere the street scene will be visible. Of course the colouration and lighting generally might need adjusting, to get the two components to look like they go together, but we can use other tools to do that. Basically we have re-invented cut-and-paste; but there is more to come.

We have ignored the alpha of the background image. It too can have an alpha channel and it may be that some of its pixels are not fully opaque. Maybe there is a window through which we are supposed to see a room interior, a third image which will be composited later. It is important to retain the correct alpha values even after we have combined images, otherwise we cannot

continue to composite. So we need a better model of compositing than simple cut-and-paste, one which composites partially opaque images and retains the correct alpha values.

8.2.1 Alpha compositing

When one partially opaque pixel is placed over another, there are various ways in which they can be made to interact. In order to describe some of these, we will adopt an opacity model and a visual representation similar to that of Oddy and Willis. They describe a partially opaque material as microscopic opaque coloured particles distributed in a medium. For the basic alpha model, we can assume that the medium is colourless. The density of particles determines the opacity alpha. It is as though we have sprayed paint particles at a fixed density alpha onto a clear material: some microscopic areas will still be clear, some will have an opaque, coloured paint particle there. The alpha value can also be thought of as the probability that a microscopic point sample will be opaque. If a second pixel is placed behind the first, the latter's coloured particles will only be directly visible where they happen to fall in line with the first material's clear areas. In some places, particles from both layers will coincide and the front colour will hide the rear. As the physical distribution of the particles is not important, we may stylistically represent the outcome as in Figure ?? . This shows two overlapping coloured but not wholly opaque layers, *A* and *B*, represented with the two axes showing their alphas. We can visualise the density of particles in the front layer simply by dividing the unit square into a clear and a coloured section, as though all the particles of colour A have been swept to one side. For the rear layer we sweep all the particles of colour B downward. There are now four regions within the unit square, which we can imagine is the resultant pixel.

This shows that Region A with area $\alpha_A(1 - \alpha_B)$ has only opaque colour A present; Region B with area $\alpha_B(1 - \alpha_A)$ has only opaque colour B present. Region AB (shown here with its own colour to help identify it) with area $\alpha_A\alpha_B$, will also be opaque colour A; and Region 0, with area $(1 - \alpha_A)(1 - \alpha_B)$, is clear.

All we now have to do is decide which regions contribute to the outcome. For a simple “over” operation, they all contribute as just described, leading to the following two formulae for the resultant colour and the resultant alpha.

$$C_R = \alpha_A C_A + (1 - \alpha_A)\alpha_B C_B.$$

$$\alpha_R = \alpha_A + (1 - \alpha_A)\alpha_B.$$

You have seen the results of these formulae every time you have seen Hollywood special effects or animation. If you can understand these two formulae, you can understand every compositing formula.

Let's go through them, term by term. In the first formula, the term $\alpha_A C_A$ says that the front layer contributes an amount α_A of colour C_A . For a large alpha, this term will dominate the equation because the second term is weighted by $(1 - \alpha_A)$. In other words, the more opaque the front layer, the less you will see of the rear layer. The rear layer offers a contribution of $\alpha_B C_B$, just as the front layer did. However, its colour can only be seen through the clear part of the front layer; the proportion of that is $(1 - \alpha_A)$.

In the formula, both colours are multiplied by their own alpha values, so their opacities are retained by this calculation. In fact all the Porter and Duff compositing formulae have this characteristic, so it is more efficient to store colours in so-called “pre-multiplied” form $(\alpha r, \alpha g, \alpha b, \alpha)$, rather than (r, g, b, α) .

Now let's look at the second formula, the one for alpha. If you replace the colours from the first formula with 1.0, you get the alpha formula. This is because the alpha formula tells us what proportion of the result is opaque. Where any colour is contributing, it must be opaque; and the rest must be clear. You can also calculate this as $(1 - (1 - \alpha_A)(1 - \alpha_B))$ i.e. the whole area minus the remaining clear part.

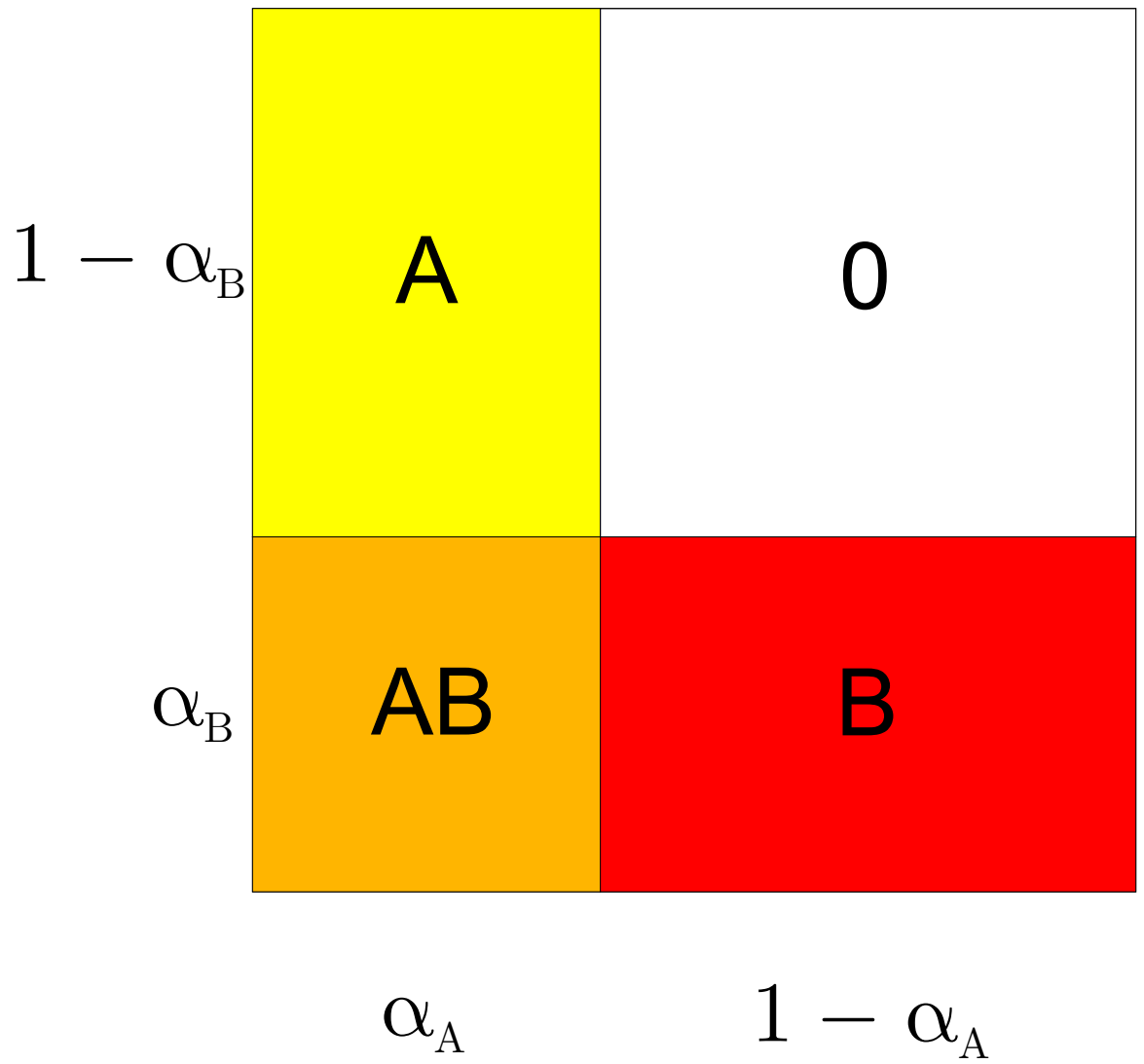


Figure 8.1: Combining colours

Porter and Duff described 12 basic ways of combining images, such as “over”, “xor”, “atop” etc. These are constructed by missing out various combinations of regions and introduce nothing new conceptually. These other operations are less useful in practice; it is the “over” operation which dominates and given calculation so we have concentrated on that.

8.2.2 Alpha and sampling defects

Our old enemy aliasing can creep in to any sampled image. Taking our earlier example, if we cut out the person from the background we are basically deciding which pixels to keep (alpha=1) and which to ignore (alpha=0). When we overlay the person on the street scene, the result will not be correctly sampled. The person’s edge pixels will have been correctly sampled against the original background, but will be wrongly sampled for the new one. We can reduce the visual effect of this by not making such a sharp cut. We arrange that pixels close to the edge have fractional alpha values, decreasing as we go outside the person. This gives a soft edge where the new background will blend in.

8.2.3 Alpha and blue screen

The blue screen technique consists of shooting an actor against a uniformly-lit blue background. This makes it easier to identify the foreground pixels.

It is not satisfactory just to search for blue pixels and force them to have zero alpha however. For one thing it is not possible to get a uniformly lit background, so the shade of blue varies. If we search for a range of blues, we may wrongly pick up colours which are in the foreground, which happen to have a lot of blue; a bright white area for example. For another, the background will never be a pure blue, with no red or green. Also, precisely because the background is illuminated, you get blue spill on the actor. This can show around their silhouette as blue colour on their shoulder etc. If the actor has fine hair, the blue will show through the hair and even illuminate it blue. A poor extraction will leave the actor missing some of their hair and with bits of their silhouette reduced here and there. Finally blue is sometimes a bad choice. Putting Superman against a blue screen would be a challenge! Green screen is the common alternative but red is also possible, if rare. When to choose which depends very much on the subject and on the lighting. We will assume blue screen is satisfactory.

How do we separate the foreground object from the blue background? Instead of searching for a shade of blue, we can search for colours in which blue dominates the red and green components and use that to set the opacity (alpha). For example,

$$\alpha = \text{blue} - \max(\text{red}, \text{green})$$

will give a high alpha where blue dominates and will have a low alpha – usually zero or negative (which we clamp to zero) – within the area of the person. This gives us a matte which we can use to isolate the person from the background. A *matte* is an image used to mask another one. We will normally pull a separate matte image, rather than force the alphas of the original image; we will often want the complementary matte as well, to mask the new background image.

This technique is not foolproof and, in practice various other techniques are used to assist. A common one is the use of a *garbage matte*, where the user simply sketches around obviously unwanted areas and a mask is created to blank out that material. Another issue is that film has a grain and digitally-generated images do not. Combining them gives a poor result unless synthetic film grain is introduced to the synthetic image to make it look more like real film.

8.3 Projective alpha colour

Given that alpha compositing has been around since the late 70s, with the main publications in the early 80s, it is perhaps surprising that there is anything new to say about it. In fact 2006 saw the University of Bath take a major step forward in understanding the process and indeed in extending it. We showed that the colours of two layers can be combined in a very general way: Porter and Duff only overlay colours, they do not mix them. New colour operations include illumination, filtering, fluorescence, colour mixing etc. Willis (cited at the start of this Chapter; the work is also patented) achieves this by unifying projective geometry with alpha compositing. It turns out that they are mathematically the same thing, in the one case applied to geometry and in the other case applied to colour. As a result, the paper also gives meaning to negative colours, negative alphas and subtractive colours (as used for printing), as well as colour and alpha values greater than 1.0. Please read the paper.

Most PC graphics cards now work with RGBA colours. Some games make use of alpha to simulate mist etc. Alpha blending is an additive operation, performing a weighted average of the two layers. When cross-fading from one scene to another, the in-between frames are often visually unsatisfactory, having a foggy appearance due to the averaging process. In computer games with white mist, averaging gives a plausible effect because mist basically replaces part of the object colour with white light, which is an additive process.

Alpha is sometimes described as providing transparency. In fact, the alpha model *cannot* simulate true transparency. Coloured transparent layers behave like filters: they are selective in the colours that they transmit. Filtration is a multiplicative operation, something which the projective alpha colour model achieves but the basic alpha model does not. Since the projective version uses the same transformations as projective geometry, graphics cards can achieve these new effects already, by using the same hardware for both geometry and colour, though developers are not yet widely aware of this.