Alpha compositing

In computer graphics, **alpha compositing** is the process of combining one image with a background to create the appearance of partial or full transparency. It is often useful to render picture elements (pixels) in separate passes or layers and then combine the resulting 2D images into a single, final image called the composite. Compositing is used extensively in film when combining computer-rendered image elements with live footage. **Alpha blending** is also used in 2D computer graphics to put rasterized foreground elements over a background.

This color spectrum image's alpha channel falls off to zero at its base, where it is blended with the background color.

In order to combine the picture elements of the images correctly, it is necessary to keep an associated *matte* for each element in addition to its color. This matte layer contains the coverage information—the shape of the geometry being drawn—making it possible to distinguish between parts of the image where something was drawn and parts that are empty.

Although the most basic operation of combining two images is to put one over the other, there are many operations, or blend modes, that are used.

Contents

Description

To store matte information, the concept of an **alpha channel** was introduced by Alvy Ray Smith in the late 1970s and fully developed in a 1984 paper by Thomas Porter and Tom Duff.[1] In a 2D picture element (pixel), which stores a color for each pixel, additional data is stored in the alpha channel with a value ranging from 0 to 1. A value of 0 means that the pixel is transparent and does not provide any coverage information; i.e. there is no occlusion at the image pixel window because the geometry did not overlap this pixel. A value of 1 means that the pixel is fully occluding because the geometry completely overlaps the pixel window.

Straight versus premultiplied

If an alpha channel is used in an image, there are two common representations that are available: straight (unassociated) alpha, and premultiplied (associated) alpha.

With straight alpha, the RGB components represent the color of the object or pixel, disregarding its opacity.

With premultiplied alpha, the RGB components represent the emission of the object or pixel, and the alpha represents the occlusion. A more obvious advantage of this is that, in certain situations, it can save a subsequent multiplication (e.g. if the image is used many times during later compositing). However, the most significant advantages of using premultiplied alpha are for correctness and simplicity rather than performance: premultiplied alpha allows correct filtering and blending. In addition, premultiplied alpha allows regions of regular alpha blending and regions with additive blending mode to be encoded within the same image, because channel values are usually stored in a fixed-point format which bounds the values to be between 0 and 1.[2]

Assuming that the pixel color is expressed using *straight* (non-premultiplied) RGBA tuples, a pixel value of (0, 0.7, 0, 0.5) implies a pixel that has 70% of the maximum green intensity and 50% opacity. If the color were fully green, its RGBA would be (0, 1, 0, 0.5).

However, if this pixel uses premultiplied alpha, all of the RGB values (0, 0.7, 0) are multiplied, or scaled for occlusion, by the alpha value 0.5, which is appended to yield (0, 0.35, 0, 0.5). In this case, the 0.35 value for the G channel actually indicates 70% green emission intensity (with 50% occlusion). A pure green emission would be encoded as (0, 0.5, 0, 0.5). Knowing whether a file uses straight or premultiplied alpha is essential to correctly process or composite it, as a different calculation is required. It is also entirely acceptable to have an RGBA triplet express emission with no occlusion, such as (0.4, 0.3, 0.2, 0.0). Fires and flames, glows, flares, and other such phenomena can only be represented using associated / premultiplied alpha.

The only important difference is in the dynamic range of the colour representation in finite precision numerical calculations (which is in all applications): premultiplied alpha has a unique representation for transparent pixels, avoiding the need to choose a "clear color" or resultant artifacts such as edge fringes (see the next paragraphs). In an associated / premultiplied alpha image, the RGB represents the emission amount, while the alpha is occlusion. Premultiplied alpha has some practical advantages over normal alpha blending because interpolation and filtering give correct results.[3]

Ordinary interpolation without premultiplied alpha leads to RGB information leaking out of fully transparent (A=0) regions, even though this RGB information is ideally invisible. When interpolating or filtering images with abrupt borders between transparent and opaque regions, this can result in borders of colors that were not visible in the original image. Errors also occur in areas of semitransparency because the RGB components are not correctly weighted, giving incorrectly high weighting to the color of the more transparent (lower alpha) pixels.

Premultiplication can reduce the available relative precision in the RGB values when using integer or fixed-point representation for the color components, which may cause a noticeable loss of quality if the color information is later brightened or if the alpha channel is removed. In practice, this is not usually noticeable because during typical composition operations, such as OVER, the influence of the low-precision colour information in low-alpha areas on the final output image (after composition) is correspondingly reduced. This loss of precision also makes premultiplied images easier to compress using certain compression schemes, as they do not record the color variations hidden inside transparent regions, and can allocate fewer bits to encode low-alpha areas. The same “limitations” of lower quantisation bit depths such as 8 bit per channel are also present in imagery without alpha, and this argument is problematic as a result.

With the existence of an alpha channel, it is possible to express compositing image operations using a *compositing algebra*. For example, given two image elements A and B, the most common compositing operation is to combine the images such that A appears in the foreground and B appears in the background. This can be expressed as A **over** B. In addition to **over**, Porter and Duff defined the compositing operators **in**, **held out by** (usually abbreviated **out**), **atop**, and **xor** (and the reverse operators **rover**, **rin**, **rout**, and **ratop**) from a consideration of choices in blending the colors of two pixels when their coverage is, conceptually, overlaid orthogonally:

The **over** operator is, in effect, the normal painting operation (see Painter's algorithm). The **in** operator is the alpha compositing equivalent of clipping.

As an example, the **over** operator can be accomplished by applying the following formula to each pixel value:

$$C_o = \frac{C_a \alpha_a + C_b \alpha_b (1 - \alpha_a)}{\alpha_a + \alpha_b (1 - \alpha_a)}$$

where C_o is the result of the operation, C_a is the color of the pixel in element A, C_b is the color of the pixel in element B, and α_a and α_b are the alpha of the pixels in elements A and B respectively. If it is assumed that all color values are premultiplied by their alpha values ($c_i = \alpha_i C_i$), we can rewrite the equation for output color as:

$$c_o = c_a + c_b (1 - \alpha_a)$$

and resulting alpha channel value is

$$\alpha_o = \frac{c_o}{C_o} = \alpha_a + \alpha_b (1 - \alpha_a)$$

Examples of different operations

Examples of red overlaid with green, with both colours fully opaque:

ADD operation | CLEAR operation | MULTIPLY operation | OVERLAY operation

Analytical derivation of the over operator

Porter and Duff gave a geometric interpretation of the alpha compositing formula by studying orthogonal coverages. Another derivation of the formula, based on a physical reflectance/transmittance model, can be found in a 1981 paper by Bruce A. Wallace.[4]

A third approach is found by starting out with two very simple assumptions. For simplicity, we shall here use the shorthand notation $a \odot b$ for representing the **over** operator.

The first assumption is that in the case where the background is opaque (i.e. $\alpha_b = 1$), the over operator represents the convex combination of a and b:

$$C_o = \alpha_a C_a + (1 - \alpha_a) C_b$$

The second assumption is that the operator must respect the associative rule:

$$(a \odot b) \odot c = a \odot (b \odot c)$$

Now, let us assume that a and b have variable transparencies, whereas c is opaque. We're interested in finding

$$o = a \odot b.$$

We know from the associative rule that the following must be true:

$$o \odot c = a \odot (b \odot c)$$

We know that c is opaque and thus follows that $b \odot c$ is opaque, so in the above equation, each $\odot$ operator can be written as a convex combination:

$$\begin{aligned} \alpha_o C_o + (1 - \alpha_o) C_c &= \alpha_a C_a + (1 - \alpha_a)(\alpha_b C_b + (1 - \alpha_b) C_c) \\ &= [\alpha_a C_a + (1 - \alpha_a)\alpha_b C_b] + (1 - \alpha_a)(1 - \alpha_b) C_c \end{aligned}$$

Hence we see that this represents an equation of the form $X_0 + Y_0 C_c = X_1 + Y_1 C_c$. By setting $X_0 = X_1$ and $Y_0 = Y_1$ we get

$$\alpha_o = 1 - (1 - \alpha_a)(1 - \alpha_b),$$
$$C_o = \frac{\alpha_a C_a + (1 - \alpha_a)\alpha_b C_b}{\alpha_o},$$

which means that we have analytically derived a formula for the output alpha and the output color of $a \odot b$.

An even more compact representation is given by noticing that $(1 - \alpha_a)\alpha_b = \alpha_o - \alpha_a$:

$$C_o = \frac{\alpha_a}{\alpha_o} C_a + \left(1 - \frac{\alpha_a}{\alpha_o}\right) C_b$$

The $\odot$ operator fulfills all the requirements of a non-commutative monoid, where the identity element e is chosen such that $e \odot a = a \odot e = a$ (i.e. the identity element can be any tuple $\langle C, \alpha \rangle$ with $\alpha = 0$).

Alpha blending

Alpha blending is the process of combining a translucent foreground color with a background color, thereby producing a new color blended between the two. The degree of the foreground color's translucency may range from completely transparent to completely opaque. If the foreground color is completely transparent, the blended color will be the background color. Conversely, if it is completely opaque, the blended color will be the foreground color. The translucency can range between these extremes, in which case the blended color is computed as a weighted average of the foreground and background colors.

Alpha blending is a convex combination of two colors allowing for transparency effects in computer graphics. The value of `alpha` in the color code ranges from 0.0 to 1.0, where 0.0 represents a fully transparent color, and 1.0 represents a fully opaque color. This alpha value also corresponds to the ratio of "SRC over DST" in Porter and Duff equations.

The value of the resulting color is given by:

$$\begin{cases} \mathrm{out}_A = \mathrm{src}_A + \mathrm{dst}_A(1 - \mathrm{src}_A) \\ \mathrm{out}_{\mathrm{RGB}} = \left(\mathrm{src}_{\mathrm{RGB}}\mathrm{src}_A + \mathrm{dst}_{\mathrm{RGB}}\mathrm{dst}_A\,(1 - \mathrm{src}_A)\right) \div \mathrm{out}_A \\ \mathrm{out}_A = 0 \Rightarrow \mathrm{out}_{\mathrm{RGB}} = 0 \end{cases}$$

If the destination background is opaque, then $\mathrm{dst}_A = 1$, and if you enter it to the upper equation:

$$\begin{cases} \mathrm{out}_A = 1 \\ \mathrm{out}_{\mathrm{RGB}} = \mathrm{src}_{\mathrm{RGB}}\mathrm{src}_A + \mathrm{dst}_{\mathrm{RGB}}(1 - \mathrm{src}_A) \end{cases}$$

The alpha component may be used to blend to red, green and blue components equally, as in 32-bit RGBA, or, alternatively, there may be three alpha values specified corresponding to each of the primary colors for spectral color filtering.

If premultiplied alpha is used, the above equations are simplified to:

$$\begin{cases} \mathrm{out}_A = \mathrm{src}_A + \mathrm{dst}_A(1 - \mathrm{src}_A) \\ \mathrm{out}_{\mathrm{RGB}} = \mathrm{src}_{\mathrm{RGB}} + \mathrm{dst}_{\mathrm{RGB}}\,(1 - \mathrm{src}_A) \end{cases}$$

Other transparency methods

Although used for similar purposes, transparent colors and image masks do not permit the smooth blending of the superimposed image pixels with those of the background (only whole image pixels or whole background pixels allowed).

A similar effect can be achieved with a 1-bit alpha channel, as found in the 16-bit RGBA Highcolor mode of the Truevision TGA image file format and related TARGA and AT-Vista/NU-Vista display adapters' Highcolor graphic mode. This mode devotes 5 bits for every primary RGB color (15-bit RGB) plus a remaining bit as the "alpha channel".

Composing alpha blending with gamma correction

The RGB values stored in computer images are actually not the real light intensities, but they have been transformed by a gamma correction, in order to optimize the usage of bits when encoding an image.

Alpha blending, not taking into account the gamma correction

Alpha blending, taking into account the gamma correction.

The gamma correction can be roughly summarised as below:

- let $\mathbf{displayed}_{\mathrm{RGB}}$ be the RGB intensity that is displayed on the screen (in normalised intensities, i.e. between 0 and 1)
- let $\mathbf{stored}_{\mathrm{RGB}}$ be the RGB intensity that is stored as bits in the computer memory (in normalised intensities also)
- let γ be the "decoding" gamma of 2.2 of the $\mathbf{stored}_{\mathrm{RGB}}$ image (γ has a typical value of 2.2)

Then we have the following relation:

$$\mathbf{displayed}_{\mathrm{RGB}} = \mathbf{stored}_{\mathrm{RGB}}^{\gamma}$$

Thus, when dealing with computer stored RGB values, alpha blending will look much better (and be more "correct", in a physical sense of additive light) if the gamma correction is unapplied before averaging the images and re-applied afterwards.[5][6] It is also possible to perform this technique with premultiplied pixels; OpenGL has extensions meant for this operation.[7]

For example, if one wants to superimpose an image named $\boldsymbol{overlay}_{\mathrm{rgb}}$ with an alpha channel $\boldsymbol{overlay}_{\alpha}$ onto a background image $\boldsymbol{background}_{\mathrm{rgb}}$, then the resulting image $\boldsymbol{out}_{\mathrm{rgb}}$ can be calculated like this:

$$\mathbf{out}_{\mathrm{rgb}} = (\mathbf{overlay}_{\mathrm{rgb}}^{\gamma} \times \mathbf{overlay}_{\alpha} + \mathbf{background}_{\mathrm{rgb}}^{\gamma} \times (1 - \mathbf{overlay}_{\alpha}))^{1/\gamma}$$

Note: $\mathbf{out}_{\mathrm{rgb}}$ is the image as it will be stored in the computer memory; and it will be displayed as $\mathbf{out}_{\mathrm{rgb}}^{\gamma}$ on the computer display.

See also

- Alpha to coverage
- Bit blit
- Blend modes
- Digital compositing
- Image masks
- Magic Pink

- Portable Network Graphics
- RGBA color space
- Texture splatting
- Transparency (graphic)
- Transparent color in palettes
- Truevision TGA

References

1. Porter, Thomas; Duff, Tom (July 1984). "Compositing Digital Images" (http://graphics.pixar.com/library/Compositing/paper.pdf) (PDF). *SIGGRAPH Computer Graphics*. New York City, New York: ACM Press. **18** (3): 253–259. doi:10.1145/800031.808606 (https://doi.org/10.1145%2F800031.808606). ISBN 9780897911382. Archived (https://web.archive.org/web/20110429041428/http://graphics.pixar.com/library/Compositing/paper.pdf) (PDF) from the original on 2011-04-29. Retrieved 2019-03-11.
2. "TomF's Tech Blog - It's only pretending to be a wiki" (https://tomforsyth1000.github.io/blog.wiki.html#%5B%5BPremultiplied+alpha%5D%5D). *tomforsyth1000.github.io*. Archived (https://web.archive.org/web/20171212111056/http://tomforsyth1000.github.io/blog.wiki.html#%5B%5BPremultiplied+alpha%5D%5D) from the original on 12 December 2017. Retrieved 8 May 2018.
3. "ALPHA COMPOSITING – Animationmet" (http://animationmet.com/alpha-compositing/). *animationmet.com*. Retrieved 2019-09-25.
4. Wallace, Bruce A. (1981). "Merging and transformation of raster images for cartoon animation" (https://archive.org/details/siggraph81confer15sigg/page/253). *SIGGRAPH Computer Graphics*. New York City, New York: ACM Press. **15** (3): 253–262 (https://archive.org/details/siggraph81confer15sigg/page/253). CiteSeerX 10.1.1.141.7875 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.141.7875). doi:10.1145/800224.806813 (https://doi.org/10.1145%2F800224.806813). ISBN 0-89791-045-1.
5. Minute Physics (March 20, 2015). "Computer Color is Broken" (https://www.youtube.com/watch?v=LKnqECcg6Gw). *YouTube*.
6. Novak, John (September 21, 2016). "What every coder should know about gamma" (https://blog.johnnovak.net/2016/09/21/what-every-coder-should-know-about-gamma/).
7. "Gamma Correction vs. Premultiplied Pixels – Søren Sandmann Pedersen" (http://ssp.impulsetrain.com/gamma-premult.html). *ssp.impulsetrain.com*.

External links

- Compositing Digital Images - Thomas Porter and Tom Duff (Original Paper) (http://keithp.com/~keithp/porterduff/p253-porter.pdf)
- Image Compositing Fundamentals (http://www.cs.princeton.edu/courses/archive/fall00/cs426/papers/smith95a.pdf)
- Understand Compositing and Color extensions in SVG 1.2 in 30 minutes! (http://www.svgopen.org/2005/papers/abstractsvgopen/)
- Alpha Matting and Premultiplication (http://dvd-hq.info/alpha_matting.php)

Foundation, Inc., a non-profit organization.