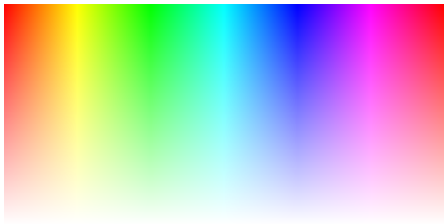
WIKIPEDIA

Alpha compositing

In computer graphics, **alpha compositing** or **alpha blending** is the process of combining one image with a background to create the appearance of partial or full transparency.[1] 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.

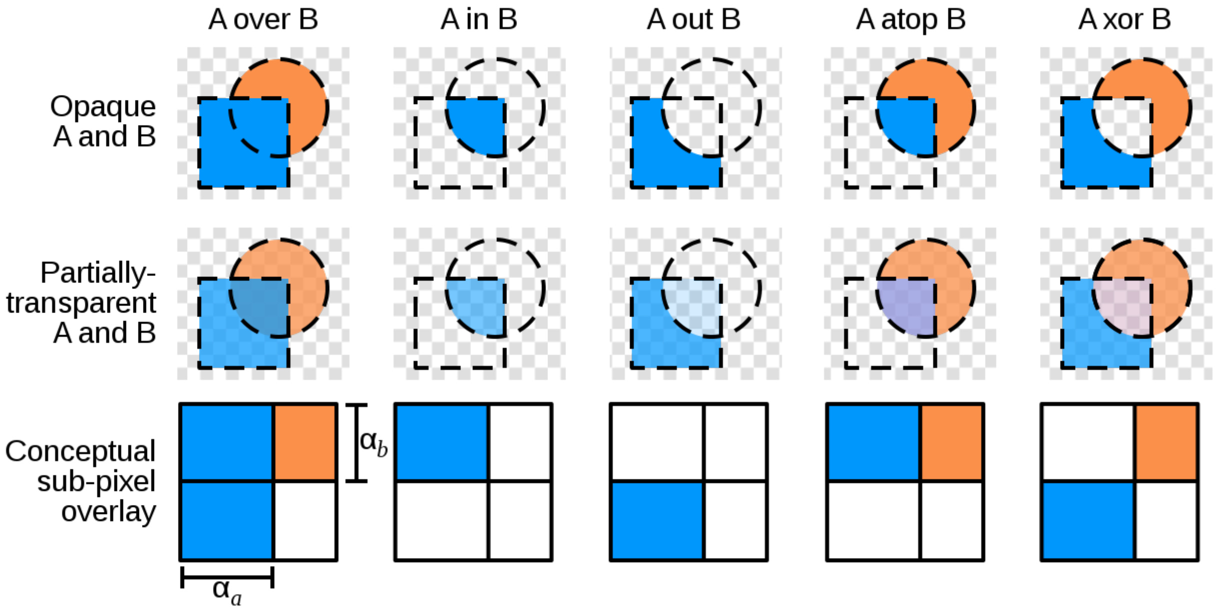
Although the most basic operation of combining two images is to put one over the other, there are many operations, or blend modes, that are used.

Contents

Description

In a 2D image a color combination is stored for each picture element (pixel), often a combination of red, green and blue (RGB). When alpha compositing is in use, each pixel has an additional numeric value stored in its **alpha channel**, with a value ranging from 0 to 1. A value of 0 means that the pixel is fully transparent and the color in the pixel beneath will show through. A value of 1 means that the pixel is fully opaque.

With the existence of an alpha channel, it is possible to express compositing image operations using a *compositing algebra*. For example, given two images A and B, the most common compositing operation is to combine the images so 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** (the phrase refers to holdout matting and is 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:



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

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

Here C_o, C_a and C_b stand for the color components of the pixels in the result, image A and image B respectively, applied to each color channel (red/green/blue) individually, whereas α_o, α_a and α_b are the alpha values of the respective pixels.

The **over** operator is, in effect, the normal painting operation (see Painter's algorithm). Bruce A. Wallace derived the **over** operator based on a physical reflectance/transmittance model, as opposed to Duff's geometrical approach.[2] The **in** and **out** operators are the alpha compositing equivalent of clipping. The two use only the alpha channel of the second image and ignore the color components.

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. The **over** operator then becomes:

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

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.[3]

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).

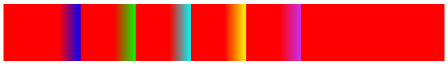
However, if this pixel uses premultiplied alpha, all of the RGB values (0, 0.7, 0) are multiplied, or scaled for occlusion, by the alpha value 0.5, which is appended to yield (0, 0.35, 0, 0.5). In this case, the 0.35 value for the G channel actually indicates 70% green emission intensity (with 50% occlusion). 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 color 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.[4]

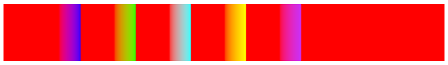
Ordinary interpolation without premultiplied alpha leads to RGB information leaking out of fully transparent (A=0) regions, even though this RGB information is ideally invisible. When interpolating or filtering images with abrupt borders between transparent and opaque regions, this can result in borders 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 color 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.

Gamma correction



Alpha blending, not taking into account gamma correction



Alpha blending, taking into account gamma correction.

The RGB values of typical digital images do not directly correspond to the physical light intensities, but are rather compressed by a gamma correction function:

$$C_{\text{encoded}} = C_{\text{linear}}^{\gamma}$$

This transformation better utilizes the limited number of bits in the encoded image by choosing γ that better matches the non-linear human perception of luminance.

Accordingly, computer programs that deal with such images must decode the RGB values into a linear space (by undoing the gamma-compression), blend the linear light intensities, and re-apply the gamma compression to the result:[5][6]

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

When combined with premultiplied alpha, pre-multiplication is done in linear space, prior to gamma compression.[7] This results in the following formula:

$$C_o = \left(C_a^{1/\gamma} + C_b^{1/\gamma} (1 - \alpha_a) \right)^{\gamma}$$

Note that only the color components undergo gamma-correction; the alpha channel is always linear.

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 high color mode of the Truevision TGA image file format and related TARGA and AT-Vista/NU-Vista display adapters' high color graphic mode. This mode devotes 5 bits for every primary RGB color (15-bit RGB) plus a remaining bit as the "alpha channel".

Screendoor transparency can be used to simulate partial occlusion where only 1-bit alpha is available.

For some applications, a single alpha channel is not sufficient: a stained-glass window, for instance, requires a separate transparency channel for each RGB channel to model the red, green and blue transparency separately. More alpha channels can be added for accurate spectral color filtration applications.

History

The concept of an alpha channel was introduced by Alvy Ray Smith and Ed Catmull in the late 1970s at the New York Institute of Technology Computer Graphics Lab, and fully developed in a 1984 paper by Thomas Porter and Tom Duff.[8]

The use of the term *alpha* is explained by Smith as follows: "We called it that because of the classic linear interpolation formula $\alpha A + (1 - \alpha)B$ that uses the Greek letter α (alpha) to control the amount of interpolation between, in this case, two images A and B".[9] That is, when compositing image A atop image B, the value of α in the formula is taken directly from A's alpha channel.

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. "Definition of alpha blending" (https://www.pcmag.com/encyclopedia/term/alpha-blending). *PCMAG*. Retrieved 2021-08-07.
2. 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. S2CID 1147910 (https://api.semanticscholar.org/CorpusID:1147910).
3. "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.
4. "ALPHA COMPOSITING – Animationmet" (https://web.archive.org/web/20190925175147/http://animationmet.com/alpha-compositing/). *animationmet.com*. Archived from the original (http://animationmet.com/alpha-compositing/) on 2019-09-25. Retrieved 2019-09-25.
5. Minute Physics (March 20, 2015). "Computer Color is Broken" (https://www.youtube.com/watch?v=LKnqECcg6Gw). *YouTube*. Archived (https://ghostarchive.org/varchive/youtube/20211122/LKnqECcg6Gw) from the original on 2021-11-22.
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*.
8. 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. S2CID 18663039 (https://api.semanticscholar.org/CorpusID:18663039). 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.

9. Alvy Ray Smith (1995-08-15). "Alpha and the History of Digital Compositing" (https://web.archive.org/web/20211025162251/http://alvyray.com/Memos/CG/Microsoft/7_alpha.pdf) (PDF). *alvyray.com*. p. 6. Archived from the original (http://alvyray.com/Memos/CG/Microsoft/7_alpha.pdf) (PDF) on 2021-10-25.

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! (https://web.archive.org/web/20180726161540/http://www.svgopen.org/2005/papers/abstractsvgopen/)
- Alpha Matting and Premultiplication (http://dvd-hq.info/alpha_matting.php)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Alpha_compositing&oldid=1066349114#Alpha_blending"

This page was last edited on 17 January 2022, at 23:51 (UTC).