Alpha Compositing

Alpha Compositing

- alpha compositing is the process of combining an image with a background to create the appearance of partial transparency.
- In order to correctly combine these image elements, it is necessary to keep an associated matte for each element.
- This matte contains the coverage information - the shape of the geometry being drawn
- This allows us to distinguish between parts of the image where the geometry was actually drawn and other parts of the image which are empty.

The Alpha Channel

- To store this matte information, the concept of an alpha channel was introduced by A. R. Smith (1970s)
- Porter and Duff then expanded this to give us the basic algebra of Compositing in the paper "Compositing Digital Images" in 1984.

The Alpha Channel

"A separate component is needed to retain the matte information, the extent of coverage of an element at a pixel.
In a full colour rendering of an element, the RGB components retain only the colour. In order to place the element over an arbitrary background, a mixing factor is required at every pixel to control the linear interpolation of foreground and background colours.
In general, there is no way to encode this component as part of the colour information.
For anti-aliasing purposes, this mixing factor needs to be of comparable resolution to the colour channels.
Let us call this an alpha channel, and let us treat an alpha of 0 to indicate no coverage, 1 to mean full coverage, with fractions corresponding to partial coverage."

Porter & Duff 84

Alpha Channel

- In a 2D image element which stores a colour for each pixel, an additional value is stored in the alpha channel containing a value ranging from 0 to 1.
- 0 means that the pixel does not have any coverage information; i.e. there was no colour contribution from any geometry because the geometry did not overlap this pixel.
- 1 means that the pixel is fully opaque because the geometry completely overlapped the pixel.
- It is important to distinguish between the following

black = (0,0,0,1)

clear = (0,0,0,0)

Pre multiplied alpha

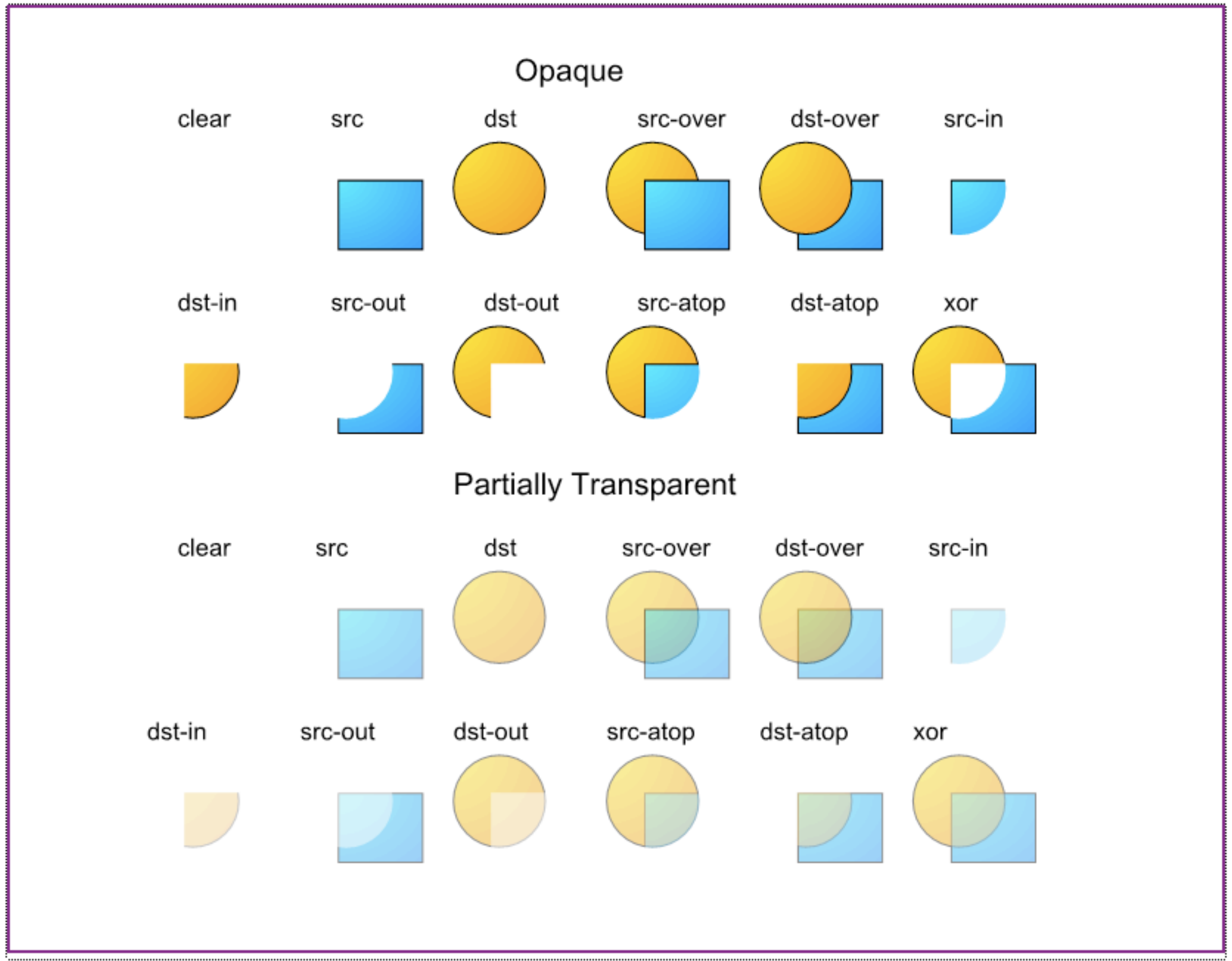
*"What is the meaning of the quadruple (r,g,b,a) at a pixel?
How do we express that a pixel is half covered by a full red object?
One obvious suggestion is to assign (1,0,0,.5} to that pixel: the .5 indicates the coverage and the (1,0,0) is the colour.
There are a few reasons to dismiss this proposal, the most severe being that all compositing operations will involve multiplying the 1 in the red channel by the .5 in the alpha channel to compute the red contribution of this object at this pixel.
The desire to avoid this multiplication points up a better solution, storing the pre-multiplied value in the colour component, so that (.5,0,0,.5) will indicate a full red object half covering a pixel".*

Porter & Duff 84

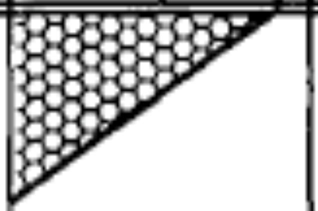
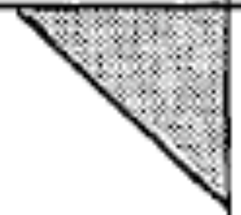
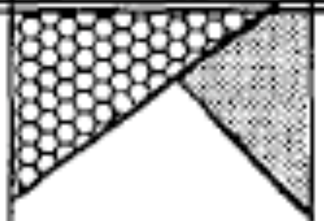
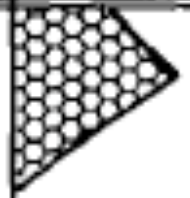
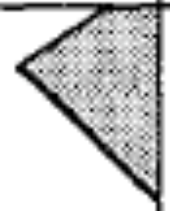
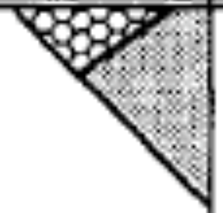
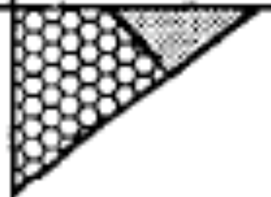
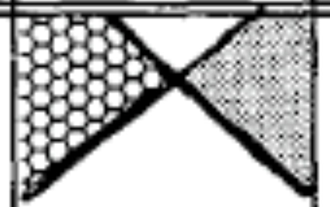
Pre multiplied alpha

- If an alpha channel is used in an image, it is common to also multiply the colour by the alpha value, in order to save on additional multiplication during the Compositing process.
- This is usually referred to as pre-multiplied alpha.
- Thus, assuming that the pixel colour is expressed using RGB triples,
- a pixel value of (0.0, 0.5, 0.0, 0.5) implies a pixel which is fully green and has 50% coverage.

Compositing Algebra



- Porter and Duff proposed a number of basic operations, which are performed on a per pixel, per RGB channel basis
- Most of these operations pre-suppose that the RGB channels have already been pre-multiplied by the alpha value

operation	quadruple	diagram	F_A	F_B
clear	(0,0,0,0)		0	0
A	(0,A,0,A)		1	0
B	(0,0,B,B)		0	1
A **over** B	(0,A,B,A)		1	$1-\alpha_A$
B **over** A	(0,A,B,B)		$1-\alpha_B$	1
A **in** B	(0,0,0,A)		α_B	0
B **in** A	(0,0,0,B)		0	α_A
A **out** B	(0,A,0,0)		$1-\alpha_B$	0
B **out** A	(0,0,B,0)		0	$1-\alpha_A$
A **atop** B	(0,0,B,A)		α_B	$1-\alpha_A$
B **atop** A	(0,A,0,B)		$1-\alpha_B$	α_A
A **xor** B	(0,A,B,0)		$1-\alpha_B$	$1-\alpha_A$

A over B

- The basic over operator is similar to the painters algorithm where the A element is placed over the B element.
- For each Pixel we do the following

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

Other Functions

Function	Use	Maths
Atop	Add effects to foreground	$C_o = C_a \times \alpha_b + (C_b \times (1 - \alpha_a))$
ADD	Add mattes together	$C_o = C_a + C_b$
DIV		$C_o = C_a \div C_b$
Mult	Mask Elements	$C_o = C_a \times C_b$
Inside	Mask Elements	$C_o = C_a \times \alpha_b$
Sub	Subtract	$C_o = C_a - C_b$
outside	Mask Elements	$C_o = C_a \times (1 - \alpha_b)$
xor		$C_o = C_a \times (1 - \alpha_b) + C_b \times (1 - \alpha_a)$

Unary operators

- To assist in dissolving and colour balancing Porter and Duff also suggested the following operations

$$\text{darken}(A, \phi) \equiv (\phi r_A, \phi g_A, \phi b_A, \alpha_A)$$
$$\text{dissolve}(A, \delta) \equiv (\delta r_A, \delta g_A, \delta b_A, \delta \alpha_A)$$

Normally $0 \leq \phi, \delta \leq 1$ although none of the theory requires it

As ϕ varies from 1 to 0, the element will change from normal to complete blackness.

If $\phi > 1$ the element will be brightened.

As δ goes from 1 to 0 the element will gradually fade from view

Luminescent objects

- Luminescent objects, which add colour information without obscuring the background, can be handled with the introduction of a opaqueness factor $\omega, 0 \leq \omega \leq 1$

$$\text{opaque}(A, \omega) \equiv (r_A, g_A, b_A, \omega\alpha_A)$$

As ω varies from 1 to 0, the element will change from normal coverage over the background to no obscuratior

The Plus Operator

- The expression A plus B holds no notion of precedence in any area covered by both pictures; the components are simply added.
- This allows us to dissolve from one picture to another by specifying

$$\mathrm{dissolve}(A, \alpha) \text{ plus } \mathrm{dissolve}(B, 1-\alpha)$$

- In terms of the binary operators above, plus allows both pictures to survive in the subpixel area AB.

operation	diagram	F_A	F_B
(0,A,B,AB) *A* **plus** *B*		1	1

References

- Thomas Porter and Tom Duff, Compositing Digital Images, Computer Graphics, 18(3), July 1984,
- http://en.wikipedia.org/wiki/Alpha_transparency
- Apple Shake Ref Guide