Digital compositing

CS465 Lecture 3

Combining images

- Often useful combine elements of several images
- Trivial example: video crossfade
  - smooth transition from one scene to another

$$r_C = t r_A + (1-t) r_B$$
$$g_C = t g_A + (1-t) g_B$$
$$b_C = t b_A + (1-t) b_B$$

  - note: weights sum to 1.0
    - no unexpected brightening or darkening
    - no out-of-range results
  - this is *linear interpolation*

Combining images

- Often useful combine elements of several images
- Trivial example: video crossfade
  - smooth transition from one scene to another

$$r_C = t r_A + (1-t) r_B$$
$$g_C = t g_A + (1-t) g_B$$
$$b_C = t b_A + (1-t) b_B$$

  - note: weights sum to 1.0
    - no unexpected brightening or darkening
    - no out-of-range results
  - this is *linear interpolation*

Combining images

- Often useful combine elements of several images
- Trivial example: video crossfade
  - smooth transition from one scene to another

$$r_C = t r_A + (1-t) r_B$$
$$g_C = t g_A + (1-t) g_B$$
$$b_C = t b_A + (1-t) b_B$$

  - note: weights sum to 1.0
    - no unexpected brightening or darkening
    - no out-of-range results
  - this is *linear interpolation*

Combining images

- Often useful combine elements of several images
- Trivial example: video crossfade
  - smooth transition from one scene to another

$$r_C = t r_A + (1-t) r_B$$
$$g_C = t g_A + (1-t) g_B$$
$$b_C = t b_A + (1-t) b_B$$

  - note: weights sum to 1.0
    - no unexpected brightening or darkening
    - no out-of-range results
  - this is *linear interpolation*

Combining images

- Often useful combine elements of several images
- Trivial example: video crossfade
  - smooth transition from one scene to another

$$r_C = t r_A + (1-t) r_B$$
$$g_C = t g_A + (1-t) g_B$$
$$b_C = t b_A + (1-t) b_B$$

  - note: weights sum to 1.0
    - no unexpected brightening or darkening
    - no out-of-range results
  - this is *linear interpolation*

Foreground and background

- In many cases just adding is not enough
- Example: compositing in film production
  - shoot foreground and background separately
  - also include CG elements

Compositing example: film effects

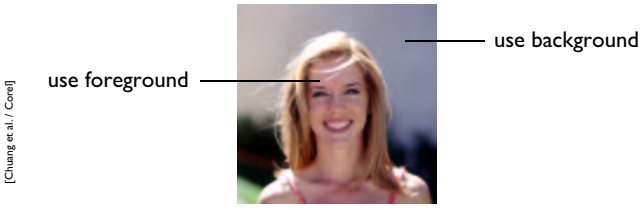
[*Titanic* ; DigitalDomain; vfxhq.com]

Foreground and background

- In many cases just adding is not enough
- Example: compositing in film production
  - shoot foreground and background separately
  - also include CG elements
  - this kind of thing has been done in analog for decades
  - how should we do it digitally?

Foreground and background

- How we compute new image varies with position

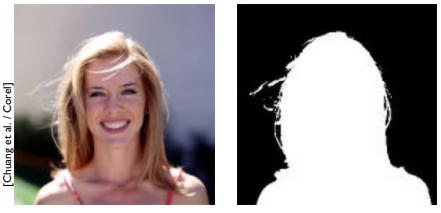


[Chuang et al. / Corel]

- Therefore, need to store some kind of tag to say what parts of the image are of interest

Binary image mask

- First idea: store one bit per pixel
  - answers question "is this pixel part of the foreground?"



[Chuang et al. / Corel]

  - does not work well near edges

Binary image mask

- First idea: store one bit per pixel
  - answers question "is this pixel part of the foreground?"

[Chuang et al. / Corel]

  - does not work well near edges

Binary image mask

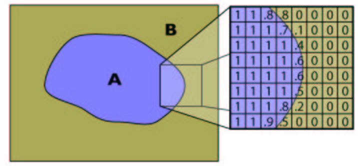
- First idea: store one bit per pixel
  - answers question "is this pixel part of the foreground?"

[Chuang et al. / Corel]

  - does not work well near edges

Partial pixel coverage

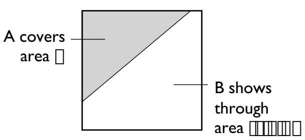
- The problem: pixels near boundary are not strictly foreground or background



  - how to represent this simply?
  - interpolate boundary pixels between the two colors

Alpha compositing

- Formalized in 1984 by Porter & Duff
  - used in essentially identical form since
- Store fraction of pixel covered, called α

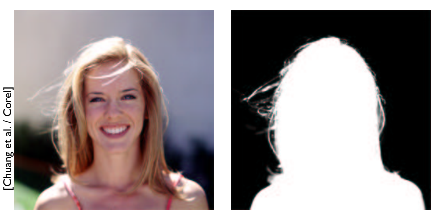


$$C = A \text{ over } B$$
$$r_C = \alpha_A r_A + (1 - \alpha_A) r_B$$
$$g_C = \alpha_A g_A + (1 - \alpha_A) g_B$$
$$b_C = \alpha_A b_A + (1 - \alpha_A) b_B$$

  - nice clean implementation: 8 more bits makes 32
  - 2 multiplies + 1 add per pixel for compositing

Alpha compositing—example

[Chuang et al. / Corel]



Alpha compositing—example

[Chuang et al. / Corel]

Alpha compositing—example

[Chuang et al. / Corel]

An optimization

- Compositing equation again

$$c_C = \alpha_A c_A + (1 - \alpha_A) c_B$$

- Note c_A appears only in the product $\alpha_A c_A$
  - so why not do the multiplication ahead of time?
- Leads to *premultiplied alpha*:
  - store pixel value (r', g', b', α) where $c' = \alpha c$
  - C = A **over** B becomes

$$c'_C = c'_A + (1 - \alpha_A) c'_B$$

  - this turns out to be more than an optimization…
  - hint: so far the background has been opaque!

Compositing composites

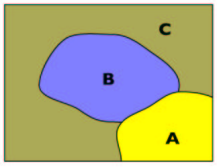
- so far have only considered single fg. over single bg.
- in real applications we have *n* layers
  - *Titanic* example
  - compositing foregrounds to create new foregrounds
    - what to do with α?
- desirable property: associativity

$$A \textbf{ over } (B \textbf{ over } C) = (A \textbf{ over } B) \textbf{ over } C$$

  - to make this work we need to be careful about how α is computed

Compositing composites

- Now some pixels have fractional coverage in more than one layer

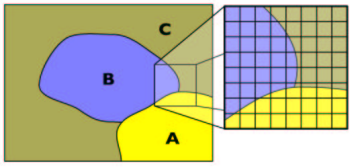


  - in D = A **over** (B **over** C) what will be the result?

$$c_D = \alpha_A c_A + (1 - \alpha_A)[\alpha_B c_B + (1 - \alpha_B) c_C]$$
$$= c'_A + (1 - \alpha_A)[c'_B + (1 - \alpha_B) c'_C]$$

Compositing composites

- Now some pixels have fractional coverage in more than one layer

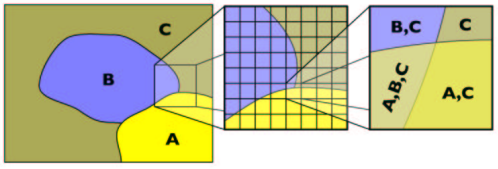


  - in D = A **over** (B **over** C) what will be the result?

$$c_D = \alpha_A c_A + (1 - \alpha_A)[\alpha_B c_B + (1 - \alpha_B) c_C]$$
$$= c'_A + (1 - \alpha_A)[c'_B + (1 - \alpha_B) c'_C]$$

Compositing composites

- Now some pixels have fractional coverage in more than one layer

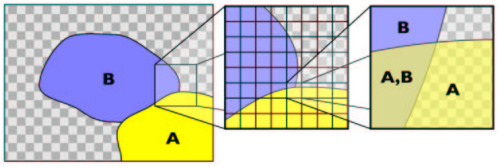


  - in D = A **over** (B **over** C) what will be the result?

$$c_D = \alpha_A c_A + (1 - \alpha_A)[\alpha_B c_B + (1 - \alpha_B) c_C]$$
$$= c'_A + (1 - \alpha_A)[c'_B + (1 - \alpha_B) c'_C]$$

Compositing composites

- What about just C = A **over** B (with B transparent)?



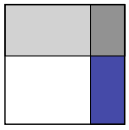
  - in premultiplied alpha, the result

$$\alpha_C = \alpha_A + (1 - \alpha_A)\alpha_B$$

looks just like blending colors, and it leads to associativity.

Independent coverage assumption

- Why is it reasonable to blend *a* like a color?
- Simplifying assumption: covered areas are independent
  - that is, uncorrelated in the statistical sense

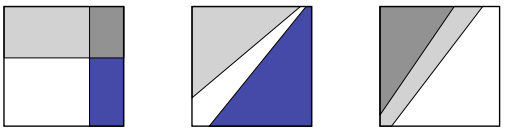


description	area
$\bar{A} \cap \bar{B}$	$(1-\alpha_A)(1-\alpha_B)$
$A \cap \bar{B}$	$\alpha_A(1-\alpha_B)$
$\bar{A} \cap B$	$(1-\alpha_A)\alpha_B$
$A \cap B$	$\alpha_A\alpha_B$

[Porter & Duff 84]

Independent coverage assumption

- Holds in most but not all cases

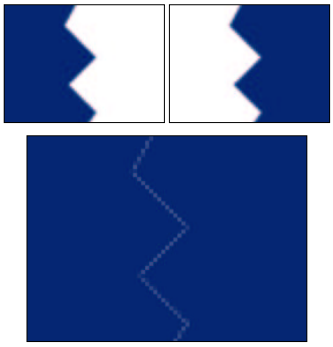


- This will cause artifacts
  - but we'll carry on anyway because it is simple and usually works…

Alpha compositing—failures

[Chuang et al. / Corel] [Cornell PCG]

positive correlation:
too much foreground

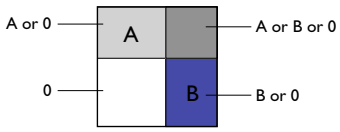


negative correlation:
too little foreground

Other compositing operations

- Generalized form of compositing equation:

$$\alpha C = A \ \mathbf{op} \ B$$
$$c = F_A a + F_B b$$



1 x 2 x 3 x 2 = 12 reasonable choices

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)		$1-\alpha_B$	$1-\alpha_A$

[Porter & Duff 84]

