Combining images

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

  \[
  \begin{align*}
  r_C &= t_A + (1-t)r_B \\
  g_C &= t_g A + (1-t)g_B \\
  b_C &= t_B A + (1-t)b_B
  \end{align*}
  \]

  – 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

  \[
  \begin{align*}
  r_C &= t_A + (1-t)r_B \\
  g_C &= t_g A + (1-t)g_B \\
  b_C &= t_B A + (1-t)b_B
  \end{align*}
  \]

  – 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**

**Foreground and background**

- How we compute new image varies with position
  - use background

- 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?”

- 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?”
  - 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 $\alpha$
  - nice clean implementation: 8 more bits makes 32
  - 2 multiplies + 1 add per pixel for compositing

Alpha compositing—example

\[
\begin{align*}
C &= A \text{ over } B \\
r_C &= \alpha r_A + (1 - \alpha) r_B \\
g_C &= \alpha g_A + (1 - \alpha) g_B \\
b_C &= \alpha b_A + (1 - \alpha) b_B
\end{align*}
\]
An optimization

- Compositing equation again
  
  \[ c_C = a_A c_A + (1 - a_A) c_B \]

- Note \( c_A \) appears only in the product \( a_A c_A \)
  - so why not do the multiplication ahead of time?

- Leads to premultiplied alpha:
  - store pixel value \((r', g', b', \alpha)\) where \( c' = \alpha c \)
  - \( C = A \) over \( B \) becomes
    \[ c'_C = c'_A + (1 - a_A)c'_B \]
  - 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 \( \alpha \)?
  - desirable property: associativity

\[
A \over (B \over C) = (A \over B) \over C
\]

- to make this work we need to be careful about how \( \alpha \) is computed

Compositing composites

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

\[
c_D = a_A c_A + (1 - a_A) [a_B c_B + (1 - a_B) c_C]
\]

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

\[
c'_D = c'_A + (1 - a_A)[c'_B + (1 - a_B)c'_C]
\]

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?

\[
c_D = a_A c_A + (1 - a_A) [a_B c_B + (1 - a_B) c_C]
\]

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

\[
\alpha_C = a_A + (1 - a_A) \alpha_B
\]

looks just like blending colors, and it leads to associativity.
**Independent coverage assumption**

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

\[
\begin{array}{c|c}
\text{description} & \text{area} \\
A \cap B & (1-\sigma_A)(1-\sigma_B) \\
A \cap \bar{B} & \sigma_A(1-\sigma_B) \\
\bar{A} \cap B & (1-\sigma_A)\sigma_B \\
\bar{A} \cap \bar{B} & \sigma_A\sigma_B \\
\end{array}
\]

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

**Alpha compositing—failures**

- positive correlation: too much foreground
- negative correlation: too little foreground

**Other compositing operations**

- Generalized form of compositing equation:
  \[
  \alpha C = A \text{ op } B \\
  c = F_Aa + F_Bb \\
  \]

\[
\begin{array}{c|c|c}
\text{A or 0} & \text{A or B or 0} \\
\text{0} & \text{B or 0} \\
\end{array}
\]

\[1 \times 2 \times 3 \times 2 = 12\text{ reasonable choices}\]