Compositing for volume rendering

The initial pixel color = Black opaque

Back-to-Front compositing:
use 'under' operator

\[ C = \text{background} \ 'under' \ C_3 \]
\[ C = C \ 'under' \ C_2 \]
\[ C = C \ 'under' \ C_1 \]

\[ C_{out} = C_{in} \ast (1-\alpha(x)) + C(x) \ast \alpha(x) \]

Or you can use 'Front-to-Back'
Compositing formula

Front-to-Back compositing:
use 'over' operator

\[ C = \text{background} \ 'over' \ C_1 \]
\[ C = C \ 'over' \ C_2 \]
\[ C = C \ 'over' \ C_3 \]

\[ C_{out} = C_{in} + C(x) \ast \alpha(x) \ast (1-\alpha_{in}) \]
\[ \alpha_{out} = \alpha_{in} + \alpha(x) \ast (1-\alpha_{in}) \]

Compositing for volume rendering

What is compositing anyway?

- A method for combining two or more images in a way that approximates the intervisibility of the scenes

\[ \text{2.5D rendering} \quad \text{entities have to be disjoint in depth} \]

Why compositing?

- Special effects (shake hands with important people...)
- Share the load of rendering
- Render translucent objects (translucent polygons, volume rendering, etc)

How to composite?

- A separate component other than RGB is needed to represent the coverage of an element at a pixel

- This component is called alpha channel
  - \( \alpha = 0 \) -> zero coverage
  - \( \alpha = 1 \) -> full coverage

Digital Image Compositing
How to composite? (2)

- 1 bit matte

Foreground CF
- R
- G
- B
1-bit mask

Background CB
- R
- G
- B

The value of alpha can be in [0,1] to indicate the extent of the coverage (or how opaque the object is).

A pixel’s ‘color’ is represented by a quadruple \((r,g,b,\alpha)\)

\(\alpha = 1\) = opaque black
\(\alpha = 0\) = transparent

Alpha Channel (2)

- How to represent a pixel that is half covered by a full red object?
  -> \((1,0,0,0.5)\) ?

  the red contribution is \(1 \times 0.5\)

- If we want to composite a foreground color \(C_f\) \((1,0,0)\) over a background color \(C_b\)
  then we do \(C = (1,0,0) \times 0.5 + (1-0.5)\times C_b\)
  i.e. \(C = C_f \times \alpha + (1-\alpha) \times C_b\)

Pre-multiplied alpha

Given \(C = C_f \times \alpha + (1-\alpha) \times C_b\)

Every time we want to perform composite, we need to multiply the color by its alpha

- why not just pre-multiplied the color components by alpha and stored that way?

\((R,G,B,\alpha) \rightarrow (R\alpha, G\alpha, B\alpha, \alpha)\)

This way, we have \(C = C_f + (1-\alpha) C_b\)

\((r,g,b,\alpha)\) premultiplied quadruple \(\rightarrow (r/\alpha, g/\alpha, b/\alpha, \alpha)\)
real color

Compositing Algebra

- Foreground over background is only one of the compositing (the simplest) methods.

- What are the formula for all possible kind of compositing (A over B, A under B, A in B...)?

- The issues is to understand and formulate the interplay between the alpha values of two input picture

Compositing Algebra (2)

What is alpha any way?

1. Represents the opaqueness of semitransparent objects. With alpha = \(\alpha\), the object will let \((1-\alpha)\) of background color go through

Screen door

Smaller alpha (more transparent)
Larger alpha (more opaque)
Compositing Algebra (3)

2. Represents the amount of pixel area covered by the object. (1-a) of the pixel is not covered, and a of the pixel is covered. (this method is better for understanding this paper)

\[ \alpha_A, 1-\alpha_A \quad \alpha_B, 1-\alpha_B \]

Compositing Algebra (4)

Assumption: If B has alpha value \( \alpha_B \), then the area A is also divided as \( \alpha_A \) and \( 1- \alpha_A \) and vice versa

Possible Compositing of A,B

All the possible compositing of A and B can be enumerated based on the value in the four regions (0, A, B, AB)

A over B:
- \( A \) and \( B \) (AB)
- A and B (A)
- B and A (B)
- A and B (0)

B over A:
- \( 0, A,B,A \)
- \( 0, A,B,B \)

Compositing Arithmetic

Basic Idea:
To composite A an B: Each input picture source (A or B) will survive in its own matte (A or B), and the fraction (FA) of its own matte not covered in the output picture

\[\alpha = \alpha_A F_A + \alpha_B F_B\]

Example: A survives in \( \alpha_A \), and \( 1-\alpha_A \)
So we have
\[Co = cA F_A + cB \]

Compositing Arithmetic (2)

\[Co = cA F_A + cB F_B\]

since \( \alpha_A = \alpha_A F_A + \alpha_B F_B \)

\[cA = \alpha_A \]
\[cB = \alpha_B \]

Compositing Arithmetic (3)

\[Co = cA F_A + cB F_B\] (note that Co is also alpha premultiplied color)

Example 1: Now let's look at 'Over'
We know \( FA = 1 \), \( FB = 1 - \alpha_A \)
So we have \( Co = cA + cB (1-\alpha_A) \)

Example 2: 'Under'
\( FA = 1 - \alpha_B \), \( FB = 1 \)
So Co = \( cA (1-\alpha_B) + cB \)
Or you can use ‘Front-to-Back’ Compositing formula

Front-to-Back compositing:
use ‘over’ operator

\[ C = \text{clear} \ 'over' \ C_1 \]
\[ C = C 'over' C_2 \]
\[ C = C 'over' C_3 \]

\[ C_{out} = C_{in} + C(x)\alpha(x)(1-\alpha_{in}) \]

\[ \alpha_{out} = \alpha_{in} + \alpha(x)(1-\alpha_{in}) \]

Compositing for volume rendering

The initial pixel color = Black opaque

Back-to-Front compositing:
use ‘under’ operator

\[ C = \text{background} 'under' C_3 \]
\[ C = C 'under' C_2 \]
\[ C = C 'under' C_1 \]

\[ C_{out} = C_{in} \times (1-\alpha_{out}) + C(x)\alpha_{out} \]

\[ C_{out} = C_{in} \times (1-\alpha_{out}) + C(x)\alpha_{out} \]