Outline

- What is image blending?
- Basic arithmetic operations:
  - Subtraction/difference
  - Average/weighted average
  - Logic operations
- Image compositing:
  - Foreground mask
  - Alpha channel
  - Blending functions

Image Blending

Objective:
- Pixel-wise combination of 2 or more images using arithmetic and logic operators
- Applications:
  - Create new images and special effects

Subtraction/Difference

- Calculate per-pixel (absolute) difference
- Highlights differences between two images

\[ G(x,y) = F(x,y) - E(x,y) + c \]
\[ G(x,y) = |F(x,y) - E(x,y)| \]

Digital Subtraction Angiography (DSP)

Average/Weighted Average

- Calculate per-pixel weighted average
- Blend between the two input images

\[ G(x,y) = \frac{F(x,y) + E(x,y)}{2} \]
\[ G(x,y) = (1-t)F(x,y) + tE(x,y) \]
Logic Operations

- Perform logic operations on pixel-by-pixel basis
- Only need to implement AND, OR, & NOT
- Others can be implemented using these 3 above
- NOT operator performs the same as negative transformation

Extract ROI using AND/OR

- Both AND & OR operators can extract region of interest (ROI)
- Which one to use depends on how the mask is defined

Image Compositing

- Objective:
  - Combine multiple images as overlapping layers to produce a single output image
- Applications:
  - Carton animation
  - Blue screen matting

Foreground Mask

- Use a binary mask to indicate the location & shape of the foreground object
- Widely used in graphics-user interface
  - Irregular shape window

Compositing with Foreground Mask

- Algorithm:
  - \( R = (M \neq 0) ? R_f : R_b \)
  - \( G = (M \neq 0) ? G_f : G_b \)
  - \( B = (M \neq 0) ? B_f : B_b \)
- Problem:
  - No partial coverage, resulting aliasing around boundary
  - Cannot represent semi-transparent object

Alpha Channel

- Indicates the level of transparency or coverage
  - \( \alpha = 0 \)
    - No coverage
  - \( \alpha = 1 \)
    - Full coverage
    - Opaque
  - \( 0 < \alpha < 1 \)
    - Partial coverage
    - Semi-transparent
**Blending with Alpha Channel**
- Referred as “over” operator
- Proposed by Smith & Catnull
- Linearly interpolate between foreground & background colors
- Algorithm:
  - \( R = R_b \cdot (1 - \alpha_f) + R_f \cdot \alpha_f \)
  - \( G = G_b \cdot (1 - \alpha_f) + G_f \cdot \alpha_f \)
  - \( B = B_b \cdot (1 - \alpha_f) + B_f \cdot \alpha_f \)
  - \( \alpha = \alpha_b \cdot (1 - \alpha_f) + \alpha_f \cdot \alpha_f \)

**Multi-layer Composition**
- A problem when composite multiple layers:
  - The operation is not associative
  - \((I_0 \& I_1) \& I_2 \neq (I_0 \& (I_1 \& I_2))\)

**Revised Blending Function**
- Used in Java 2D
- Proposed by Wallace & Levoy
- Algorithm:
  - \( R = R_b \cdot \alpha_b \cdot (1 - \alpha_f) + R_f \cdot \alpha_f \)
  - \( G = G_b \cdot \alpha_b \cdot (1 - \alpha_f) + G_f \cdot \alpha_f \)
  - \( B = B_b \cdot \alpha_b \cdot (1 - \alpha_f) + B_f \cdot \alpha_f \)
  - \( \alpha = \alpha_b \cdot (1 - \alpha_f) + \alpha_f \)

**Premultiplied Pixel**
- Color value can be pre-multiplied for efficiency
  - \( R' = R \cdot \alpha \)
  - \( G' = G \cdot \alpha \)
  - \( B' = B \cdot \alpha \)
  - \((r, g, b, \alpha)\) means a pixel is \(\alpha\)% covered by color \((r/\alpha, g/\alpha, b/\alpha)\)
  - \((\frac{1}{2}, 0, 0, 1)\) -> half red, full coverage
  - \((\frac{1}{2}, 0, 0, \frac{1}{2})\) -> full red, half coverage
- Blending algorithm:
  - \( R = R_b \cdot \alpha_b \cdot (1 - \alpha_f) + R_f \cdot \alpha_f \)
  - \( G = G_b \cdot \alpha_b \cdot (1 - \alpha_f) + G_f \cdot \alpha_f \)
  - \( B = B_b \cdot \alpha_b \cdot (1 - \alpha_f) + B_f \cdot \alpha_f \)
  - \( \alpha = \alpha_b \cdot (1 - \alpha_f) + \alpha_f \)
- Be careful
  - Check whether pixel is premultiplied before do blending.