Outline
- Representing images
- Sampling & quantization
- Image compositing
  - Foreground mask
  - Alpha channel
  - Blending functions

What is an Image?
- A 2D representation of any intensity over a finite spatial range
  - Imagine the film
  - Math representation: \( f(x, y) \)
    - \( x \) & \( y \) are spatial coordinates
    - \( x_{\text{min}} \leq x \leq x_{\text{max}} \)
    - \( y_{\text{min}} \leq y \leq y_{\text{max}} \)
    - \( f(x, y) \) is the intensity at location \((x, y)\)

Digital Image
- The spatial parameters are sampled at discrete locations:
  - \( F[p, q] = f(p \times \Delta x, q \times \Delta y) \)
- The intensities are quantized to be represented by a finite number of bits:
  - \( F[p, q] = k \)
  - \( k = \text{int}(f / \Delta I) \)

Sampling & Quantization
- The result of sampling & quantization is a matrix of numbers
  - Each element of the matrix is called image element or pixel
  - In computer, the matrix is normally kept in a 1D array in row-first order

Representing Digital Images
- The result of sampling & quantization is a matrix of numbers
  - Each element of the matrix is called image element or pixel
  - In computer, the matrix is normally kept in a 1D array in row-first order

**Representing Colors**
- The 3 channels in color images are kept using:
  - 3 unsigned characters (3 bytes)
  - Or a single integer (4 bytes)
- The range represented by a byte is limited to 0~255
- High-dynamic range images use 16-bit integer or float to store each channel

---

**Conversion between Integer and 3 Bytes**
- R, G, & B to integer:
  \[ n = r << 16 | g << 8 | b; \]
- Integer to R, G, & B:
  \[ r = (n & 0x00FF0000) >> 16; \]
  \[ g = (n & 0x0000FF00) >> 8; \]
  \[ b = n & 0x000000FF; \]

---

**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

---

**Foreground/Background Compositing**
- 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
    - Transparent
  - \( \alpha = 1 \)
    - Full coverage
    - Opaque
  - \( 0 < \alpha < 1 \)
    - Partial coverage
    - Semi-transparent
**Blending Function I**

- Referred as “over” operator
- Proposed by Smith & Catnull
- Algorithm:
  - $R = R_b \cdot (1 - \alpha_f) + R_f \cdot \alpha_f$
  - $G = G_b \cdot (1 - \alpha_f) + G_f \cdot \alpha_f$
  - $B = B_b \cdot (1 - \alpha_f) + B_f \cdot \alpha_f$
  - $\alpha_f = \alpha_b \cdot (1 - \alpha_f) + \alpha_f \cdot \alpha_f$

**Multi-layer Composition**

- A problem when composite multiple layers:
  - The operation is not associative
  - $I_0 \& I_1 \& I_2 \& I_3 \neq (I_0 \& I_1) \& (I_2 \& I_3)$

**Blending Function II**

- Used in Java 2D
- Proposed by Wallace & Levoy
- Later by Porter & Duff
- Algorithm:
  - $R = R_b \cdot \alpha_b \cdot (1 - \alpha_f) + R_f \cdot \alpha_f$
  - $G = G_b \cdot \alpha_b \cdot (1 - \alpha_f) + G_f \cdot \alpha_f$
  - $B = B_b \cdot \alpha_b \cdot (1 - \alpha_f) + B_f \cdot \alpha_f$
  - $\alpha_f = \alpha_b \cdot (1 - \alpha_f) + \alpha_f$

**Premultiplied Pixel**

- Color value can be pre-multiplied for efficiency.
  - $R' = R \cdot a$
  - $G' = G \cdot a$
  - $B' = B \cdot a$
  - $(r, g, b, a)$ represents a pixel is $a\%$ covered by the color $(r/a, g/a, b/a)$
  - $(1, 0, 0, 1) \rightarrow$ full red, full coverage.
  - $(1/2, 0, 0, 1) \rightarrow$ half red, full coverage.
  - $(1/2, 0, 0, 1/2) \rightarrow$ full red, half coverage.

**Blending with Premultiplied Pixel**

- Algorithm:
  - $R = R_b' \cdot (1 - a_f) + R_f'$
  - $G = G_b' \cdot (1 - a_f) + G_f'$
  - $B = B_b' \cdot (1 - a_f) + B_f'$
  - $a = a_b \cdot (1 - a_f) + a_f$
  - Be careful:
    - Check whether pixel is premultiplied before do blending.