Image Processing
... computing with and about data,
... where "data" includes the values and relative locations of the colors that make up an image.

An image is an array of colors

Pixel: Picture Element

Color
• A triple of bytes [0, 255]
  – RGB or HSB
• Transparency (alpha)
  – How to blend a new pixel color with an existing pixel color

Accessing the pixels of a sketch
• loadPixels()
  – Loads the color data out of the sketch window into a 1D array of colors named pixels[]
  – The pixels[] array can be modified
• updatePixels()
  – Copies the color data from the pixels[] array back to the sketch window

A 100-pixel wide image
• First pixel at index 0
• Right-most pixel in first row at index 99
• First pixel of second row at index 100

The pixels[] array is one-dimensional

See also colorNoise.pde
Accessing Pixels as a 2D Array
• Pixels can be accessed as a 2D array using the following formula:
  \[ \text{index} = r \times \text{width} + c \]
  \[ \text{index} = y \times \text{width} + x \]
• Using 0-based indices
  \[ \text{int idx} = \text{width} \times r + c; \]
  \[ \text{pixels}[\text{idx}] = \text{color}(b); \]

Rendering Images in a Sketch
• Image data can be loaded from a file using `loadImage()` method, and drawn on a sketch with the `image()` command
  ```
  \text{PImage img = loadImage("myImage.jpg"); image(img, 0, 0);}
  ```
• The PImage object also permits individual pixel color data to be modified.
  – like the sketch window

PImage

Methods

- `loadPixels()`: Loads the color data out of the PImage object into a 1D array of colors named `pixels[]`.
- `updatePixels()`: Copies the color data from the `pixels[]` array back to the PImage object.

Also

- `red(color)` extract the red component from a color
- `blue(color)` extract the green component from a color
- `green(color)` extract the blue component from a color

Basic Filters

• Color
  – Extracting Red/Green/Blue colors
    • `pixels[i] = color(red(c), 0, 0);`
    • `pixels[i] = color(0, 0, blue(c));`
  – Grayscale
    • `pixels[i] = color(0.3*red(c) + 0.59*green(c) + 0.11*blue(c));`
  – Negative
    • `pixels[i] = color(255-red(c), 255-green(c), 255-blue(c));`
Sepia

- Technique for archiving BW photos
  - float \( r = \) red(c)*0.393+green(c)*0.769+blue(c)*0.189;
  - float \( g = \) red(c)*0.349+green(c)*0.686+blue(c)*0.168;
  - float \( b = \) red(c)*0.272+green(c)*0.534+blue(c)*0.131;
  - pixels[i] = color(r, g, b);

get(…)

- Get a single pixel (very slow)
  Color c = img.get(x, y);

- Get a rectangular range of pixels
  PImage img2 = img.get(x, y, w, h);

tint(…)/noTint()

- tint() modifies the fill value for images
  tint( gray );
  tint( gray, alpha );
  tint( red, green, blue );
  tint( red, green, blue, alpha );

- Turn off applied tint() values with noTint()

PImage

Methods (Cont’d)

get(...) Reads the color of any pixel or grabs a rectangle of pixels

set(...) Writes a color to any pixel or writes an image into another

copy(...) Copies pixels from one part of an image to another

mask(...) Masks part of the image from displaying

save(...) Saves the image to a TIFF, TARGA, PNG, or JPEG file

resize(...) Changes the size of an image to a new width and height

blend(...) Copies a pixel or rectangle of pixels using different blending modes

filter(...) Processes the image using one of several algorithms

// warhol2

void setup() {
  // Load the image three times
  PImage warhol1 = loadImage("andy-warhol2.jpg");
  size(warhol1.width*3, warhol1.height);

  // Draw modified images
  tint(255, 0, 0);
  image(warhol1, 0, 0);
  tint(0, 255, 0);
  image(warhol1, 255, 0);
  tint(0, 0, 255);
  image(warhol1, 500, 0);
  }
```java
// fade
PImage[] img = new PImage[5];
int alpha = 255;
int i1 = 0, i2 = 1;
void setup() {
  size(600, 400);
  imageMode(CENTER);
  for (int i=0; i<img.length; i++) // Load images
    img[i] = loadImage("bmc"+i+.jpg);
}
void draw() {
  background(255);
  // Fade out current image
  tint(255, alpha);
  image(img[i1], 300, 200);
  // Fade in next image
  tint(255, 255-alpha);
  image(img[i2], 300, 200);
  // Swap images when fade complete
  alpha--;
  if (alpha < 0) {
    i1 = (i1 + 1) % img.length;
    i2 = (i2 + 1) % img.length;
    alpha = 255;
  }
}
```

```java
// pointillism
PImage img;
void setup() {
  img = loadImage("bmc3.jpg");
  image(img.width, img.height);
  imageMode(CENTER);
  image(img, width/2, height/2);
  noStroke();
  ellipseMode(CENTER);
  // Cover with random circles
  for (int i=0; i<20000; i++) addPoint();
}
void addPoint() {
  // Add a random filled circle to image
  int x = (int)random(width);
  int y = (int)random(height);
  int i = x * widthy;
  color c = pixels[i];
  fill(c);
  ellipse(x, y, 7, 7);
}
void draw() {
  addPoint();
}
```

Simple Image Visualization

- Sample pixel colors every n pixels
- Draw a grid of basic shapes (ellipse, rect, line, triangle, etc) using the sampled color as fill color or stroke color