Introduction to Images

ISE 108: Introduction to Programming
Stony Brook University

Images and PImage

- The PImage type is used to represent an image in a Processing program
  - e.g., PImage myPic;
- PImage has two fields (height and width) that tell us the size of an image
  - These fields are constants
  - We can't change them to resize an image!

Why is it called "PImage" and not just "Image"?

- Processing is based on Java
- Java already has a class called "Image"
- "PImage" is Processing's redefined version of this class

Loading Images from Disk

- Use the loadImage() function to open an image file from disk
  - loadImage() returns a PImage object
  - loadImage() takes a String representing the name of the file
    - e.g., PImage myPic = loadImage("dog.jpg");
- Image files ***MUST*** be stored in a directory named "data" inside your sketch folder

Displaying an Image

- To display an image, use the image() function
- Usage: image (source, x, y)
  - source = PImage representing the file/image
  - x and y: coordinates of the top left corner of the image
- Add two more (integer) arguments to specify how wide and tall the image should be drawn
  - The image will be scaled/stretched as needed

Image Example


```
size(400, 400);
PImage pic =
 loadImage("jesterdog.jpg");
image(pic, 0, 0);
```

Image Manipulation

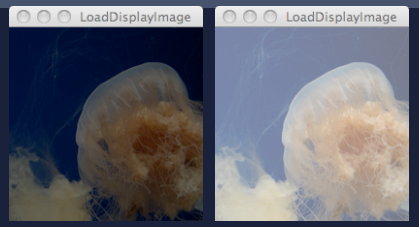
- Processing lets us change an image before we display it
- Processing defines several basic filters that can modify an image
- We can also create our own image filters

Built-in Image Filters

- tint() changes the way an image looks
  - tint(a) changes the image's brightness
    - 0 = black, 255 = normal brightness
  - tint(a, b) affects alpha transparency as well
  - tint(a, b, c) adjusts the red/green/blue
  - tint(a, b, c, d) adjusts r/g/b and alpha

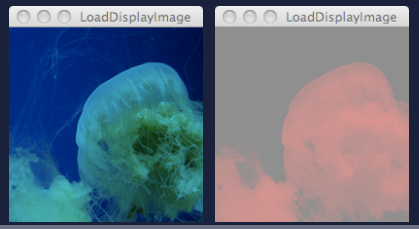
Experiments with tint()

tint(100);



tint(255, 127);

tint(0,200,255);



tint(255,0,0,100);

PImage Properties

- The PImage class has two fields (width and height) that tell us how large an image is
- e.g., int numRows = myPic.height
- Note that these fields are constants; we can't change them to resize an image directly!

Pixel-Perfect Filters

- Every PImage has an internal array of pixels
  - We can directly edit pixel values in this array
- The internal array is named pixels
- Before we can use this array, we need to fill it by calling loadPixels():
  - myPic.loadPixels();
- To save our changes, use updatePixels()
  - This doesn't modify the original file

Locating a Specific Pixel

- We describe pixels by their row and column, but the array is one-dimensional
- If we know the pixel's location, and the width of the image, we can use the formula

  array index = COLUMN + (ROW * WIDTH)

Pixel Editing

- The "pixels" array contains variables of type "color"
  - We can change a pixel by assigning that pixel a new color
    - e.g., pixels[5] = color(120, 34, 170);
  - We can "automate" this process with a loop

Color Components

- Processing provides helper functions to get one component of a given pixel's color
  - red() returns the value of the red component
  - blue() and green() do the same thing
  - WARNING: these methods return a float!
- e.g., float rValue = red(myPic.pixels[17]);

A Simple Image Filter

```
// This filter tints an image by removing
// the blue from all of its pixels

PImage myPic = loadImage("jesterdog.jpg");

myPic.loadPixels(); // fill the pixels[] array

for (int i = 0; i < myPic.pixels.length; i = i + 1)
{
   // Save original red and green values
   float r = red(myPic.pixels[i]);
   float g = green(myPic.pixels[i]);
   myPic.pixels[i] = color(r, g, 0);
}

myPic.updatePixels(); // "save" the changes
```