Image Manipulation: Pixel Traversal

Elements of Graphics
CS324e
Bits are binary (0 or 1)

Pixels are composed of bits

- Bits-per-pixel determine the range of color

Images are composed of pixels
Image Buffers

- Screen pixel data is stored in an array
- This array (or image buffer) allows us to access per-pixel information
Images in Processing

- Image buffers are stored in the PImage data type
- PImage allows for loading and displaying image data
- Some image manipulation:
  - Size
  - Position
  - Opacity
  - Tint
- To display: `image(PImage img, float x, float y, float width, float height)"
Loading and Displaying Images

PImage img;

void setup() {
    size(100, 100);
    img = loadImage("foo.png");
}

void draw() {
    image(img, 0, 0); //Note: we must load an image before displaying it!
}
Fitting Processing Window to Image Size

```java
void setup() {
    surface.setResizable(true);
    img = loadImage("foo.png");
    surface.setSize(img.width, img.height);
}
```
Changing Individual Pixels

- `loadPixels()` and `updatePixels()` should be called before and after pixel manipulation respectively

- `loadPixels()` allows us to **read** from the pixel data

- `updatePixels()` **writes** any changes back to the pixel data

- Calls not necessary for every Operating System, but **may not work without them**

- `PImage.pixels` array stores each pixel as a color

- Access the color of the pixel at index in `PImage img`:

```java
color c = img.pixels[index];
```
Consider…

❖ How can we access every pixel in an image?
❖ How can we access every pixel by its (x, y) value?
❖ Hint: remember this layout!

```
How the pixels look:

0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19
20 21 22 23 24
```

```
How the pixels are stored:

0 1 2 3 4 5 6 7 8 9 . . .
```
Accessing by (x, y) Coordinates

- We will perform a **stride** into the 1D array to know which **row** (e.g. y value) we are currently looking for.
- Once we get to the correct row, we can use the **column** (e.g. x value) to find the final placement of the index into the 1D array.
- To do this, we must know the **image width**
  - row * imageWidth = index of row in the 1D array
  - Then add in the column value
Traversing an Image Buffer

//access img’s pixels

img.loadPixels();

for (int x = 0; x < img.width; x++) {
    for (int y = 0; y < img.height; y++) {
        //access pixel at index and set c to its value
        int index = x + y*img.width;
        color c = img.pixels[index];
    }
}

//update any modifications to img’s pixels

img.updatePixels();
What data type is stored in the `pixels` array in a `PImage`?
Tint

- `tint()` modifies the color of the displayed images
- `noTint()` disables `tint()` modifications

```cpp
void draw() {
  tint(0, 153, 204);
  image(img, 0, 0);
  noTint();
  image(img, 50, 50);
}
```
Hands-on: Creating Tint

❖ Today’s activity:

1. Re-create Processing’s `tint` functionality using a method you create (i.e. do not use the existing tint function)

   1. This method can take RBG/color data like the Processing `tint` method does

2. You may want to make your `tint` method to be “per image” rather than “per screen” — to do this, your method should also have an argument for the `PImage` you will tint