IAT 265

Images and Text in Processing

SCHOOL OF INTERACTIVE
ARTS + TECHNOLOGY

Outline

- Programming concepts
  - Classes
  - Pimage
  - Pixel array: 2D array vs. 1D array
  - Pfont
- Flip images (vertically, horizontally, & diagonally) using nested *for*-loops
- Produce dynamic image & text effects

Review

- Object Oriented Programming
  - Class -- a type you define
  - Instance -- one variable (object) of a class
  - Fields -- variables within a class
  - Methods -- functions within a class
  - Constructors -- special methods for creating an instance of a class

Image class

- PImage is also a class – built-in class
  - Each actual image is an object

- Inside Processing library, it is defined as:

```
class PImage {
        float width;            //width of the image
        float height;           //height of the image
        float[] pixels;         //pixels of the image
        ...
}
```

Functions for Image loading & Displaying

- Function for image loading:

  PImage *loadImage* (String filename);

- Function for image display:

  void *image* (PImage img, int x, int y)

Loading Images

- Loading an image
  - Give your project a name and save it
  - Place the image file in:
    - **<project name>/data/**
  - Use this code to load it

```
PImage img = loadImage("image filename");
```

Displaying Images

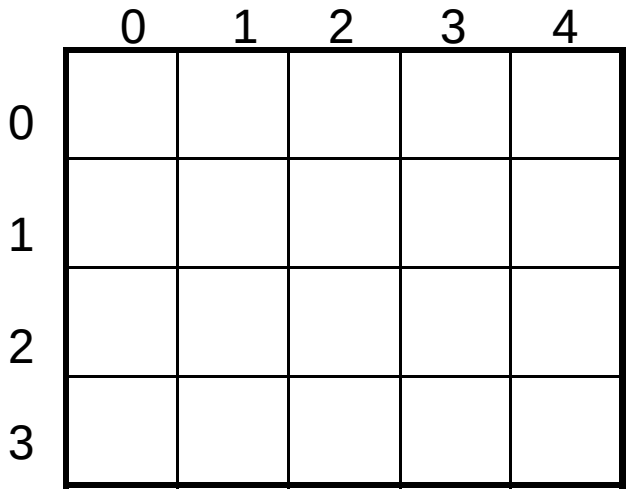
- **`image(PImage img, int x, int y);`** shows your image at location (x, y)
  - **`image(img, 0, 0)`** will display your image at the origin of the window
- Where is (x, y) referring to on the image?
  - It depends on imageMode

imageMode() sets the location of drawing

- imageMode(CORNER);
  - (x, y) is the **top left** corner of the image – the default location
- imageMode(CENTER);
  - (x, y) is the **center** of the image

Accessing Pixels

- How do we access pixels of an image, which is by its nature a 2D matrix?



2D Arrays

- Java allows us to make Array of Arrays – otherwise called 2D Array

```
int[][] bob = new int[3][4];
color[][] pixels2d = new color[200][200];
```

- However, Processing doesn't provide us with a 2D array of pixels to use

2D Arrays

- Interestingly, 2D Arrays are just covering up a 1D array

- The 2D array

```
color[][] pixels2d = new color[10][20];
color c2 = pixels2d[3][2];
```

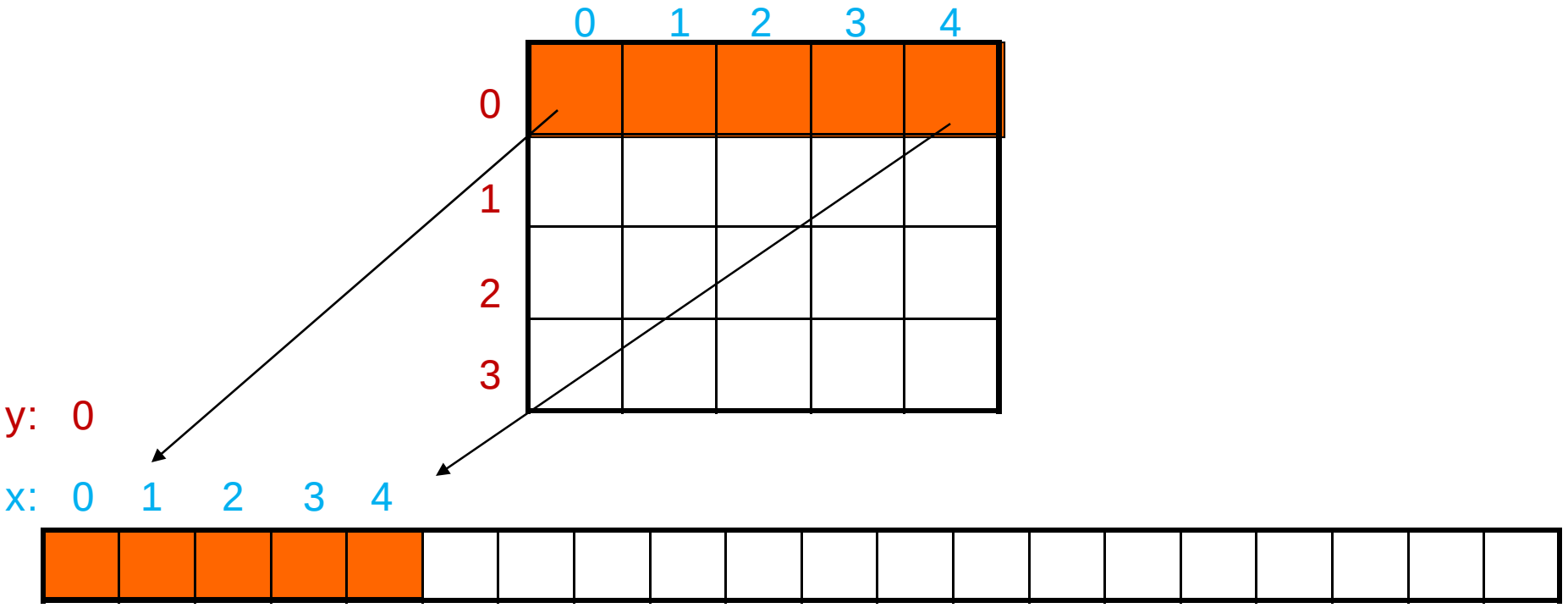
- is equivalent to:

```
color[] pixels1d = new color[10*20];
color c1 = pixels1d[3 + 2*10];
```

Underneath, these two pieces of code do the same thing. Computer graphics, however, normally uses 1D array to store pixels of an image – pixels[]

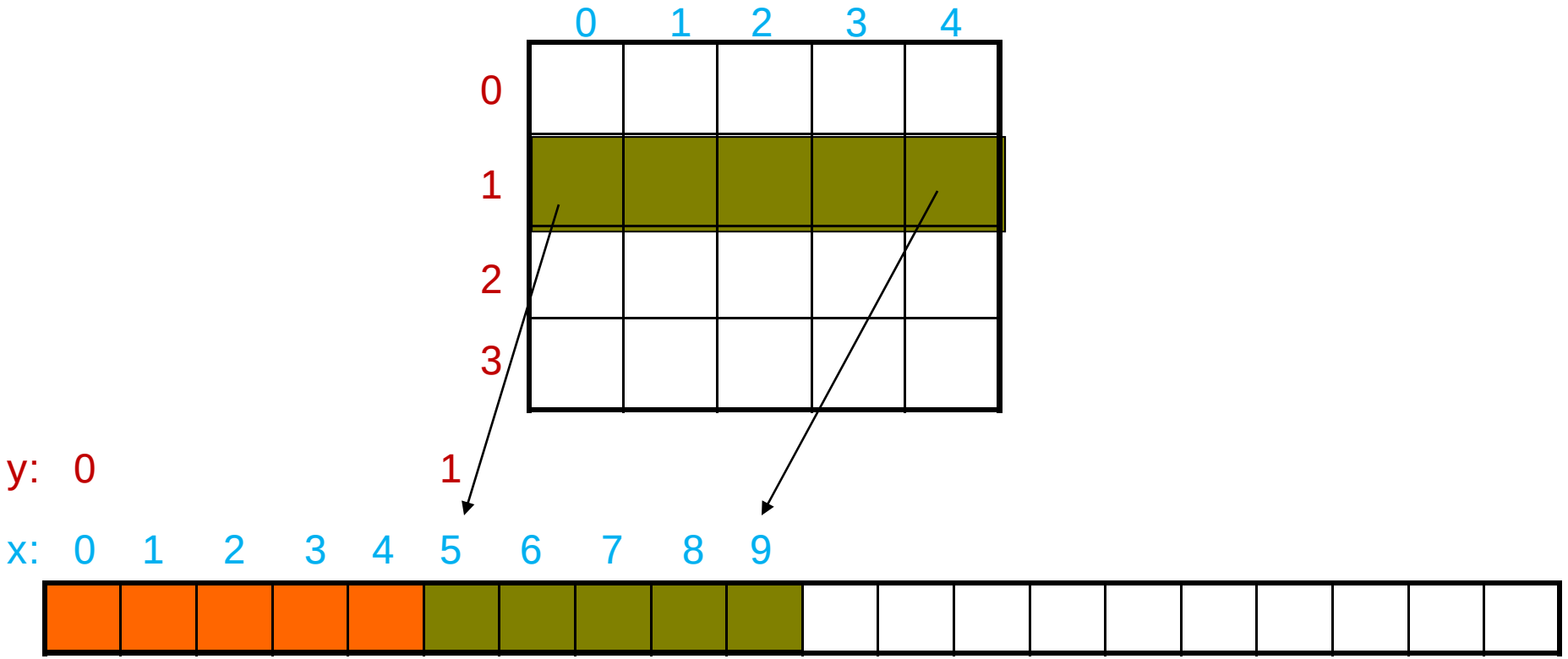
2D matrix converted into 1D array

- How do we map pixels of a 2D image into a 1D `pixels[]` array?



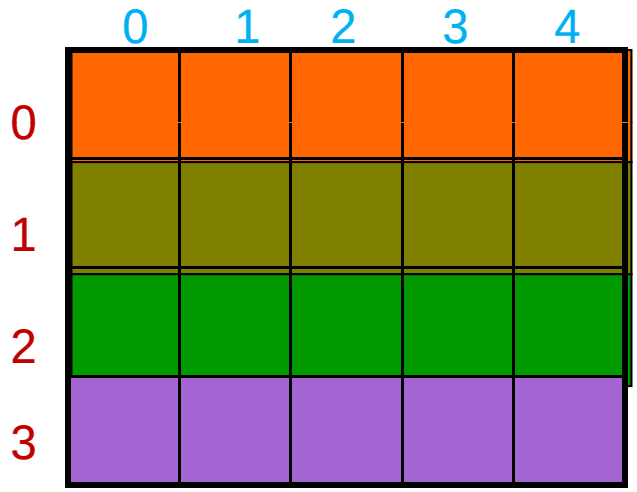
2D matrix converted into 1D array

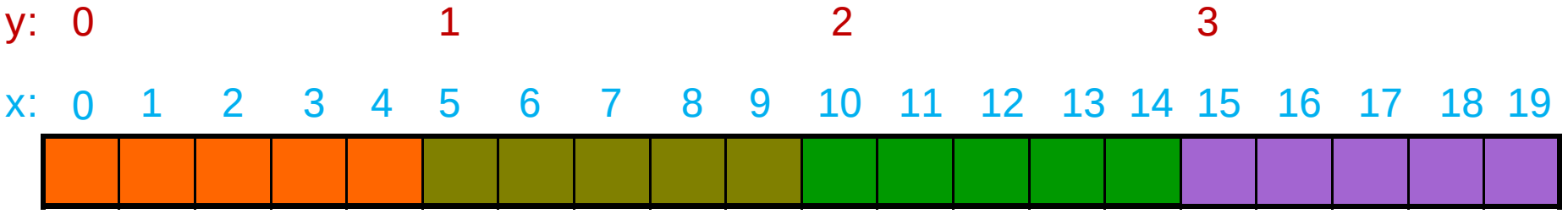
- Its 2nd row goes into 2nd segment of the pixels[] array



2D matrix converted into 1D array

- The same for 3rd and 4th rows...





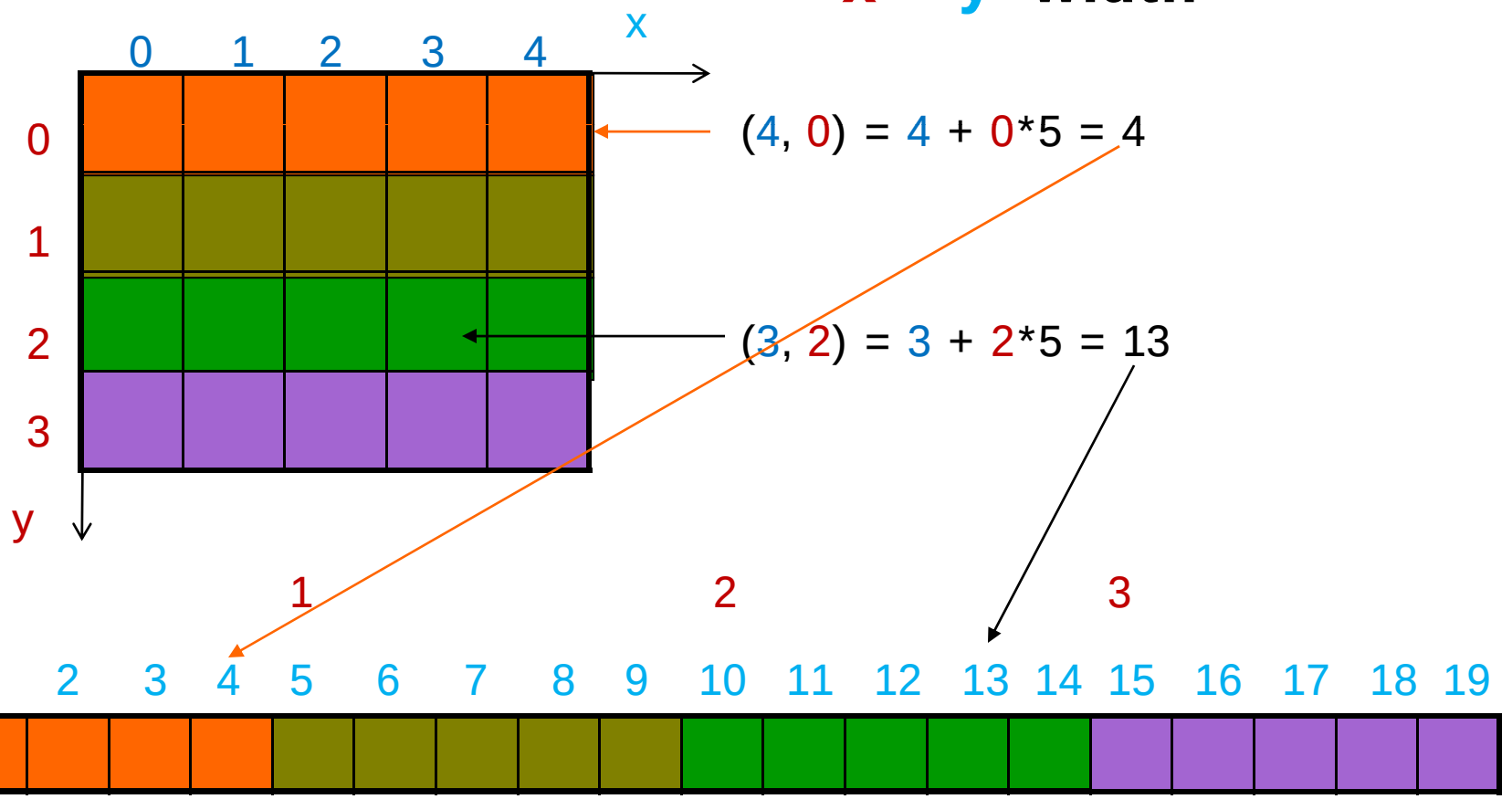
Accessing Pixels

- The **`PImage`** object allows you to access each of its pixels (color values) with the **`pixels[]`** array

- You can get the width and height of the image using the **`width`** and **`height`** **fields** of `PImage`

Accessing Pixels

- Calculate array Index:
  - **x + y*width**



Accessing Pixels

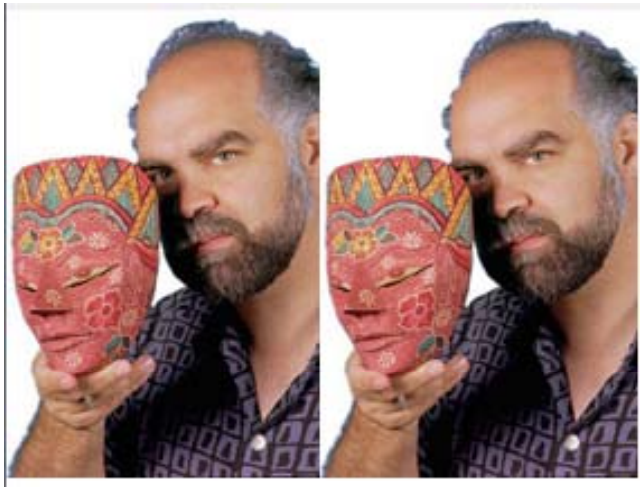
- Now we know the array index, how to get the color from a pixel?

```
PImage img = loadImage("face.jpg");
image(img, 0, 0);
//get color at (15, 10)
color c1 = img.pixels[15 + 10*img.width];
// set our lines' color
stroke(c1);
line(50, 150, 80, 180);
line(80, 180, 110, 140);
```

- Let's look at some applications of manipulate image using pixels[]

Use Loops and Pixel Array to Manipulate Images

- Copying Pixels
  - To copy a pixel array, use two nested for loops:
    - one moves across the rows, and the other moves across the columns to copy over every pixel



Recap: Nested Loops

- Nesting of loops refers to putting one loop inside the braces of another

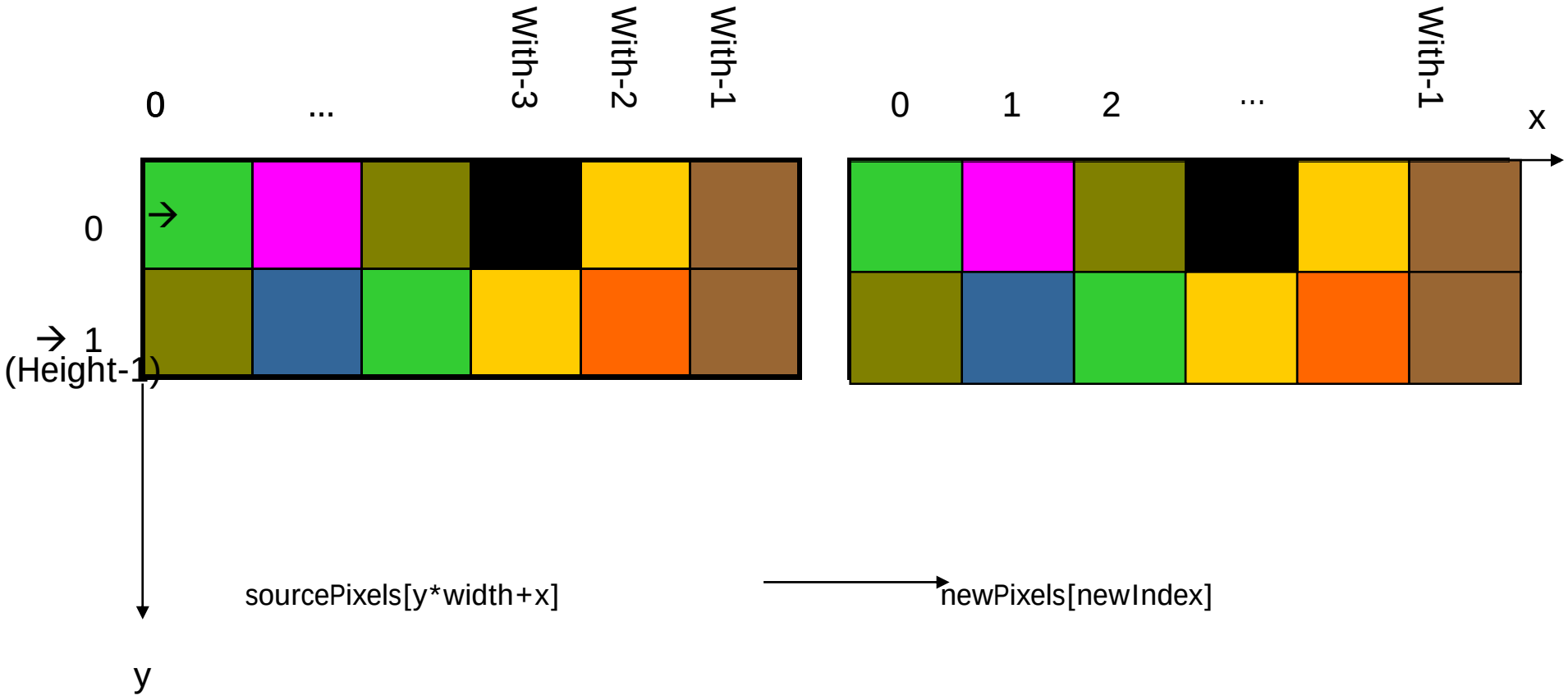
```
for(int i = 10; i <= 100; i += 10) {
  for(int j = 10; j <= 100; j += 10) {
    point(i, j);
  }
}
```

- For each **i**, the inside **j** loop will run through;
- then **i** will increment;
- then the inside **j** loop will run through again...

Recap:
Counting Backward loop

```
/*
 * CountDown: Demonstrates how to make a for loop
 * count backward by using the decrement (--) operator
 */
println("Countdown:");
for (int t=10; t > 0; t--){
  print(t + " ");
}
print("\nBLASTOFF!");
```

Demo: Copying Pixels



Implementation: Copying Pixels

```
// Create a method to copy an pixel array
int[] copyPixels(int srcPixels[], int w, int h) {
   int newPixels[] = new int[w * h];
   int newIndex = 0;

   //Loop forward through all the rows
   for (int y = 0; y < h; y++)
        //for each row loop forward through all the columns
        for (int x = 0; x < w; x++){
             newPixels[newIndex] = srcPixels[x + y * w];
             newIndex++;
        }

   return newPixels;
}
```

Create an Image from pixels

- To create an image from new pixels:
  - Create an Image object by calling the built-in function:

```
PImage createImage(int width, int
height, format)
```

    - `format`: RGB, ARGB, ALPHA (grayscale alpha channel)
  - Set its pixels[] array to the new pixels

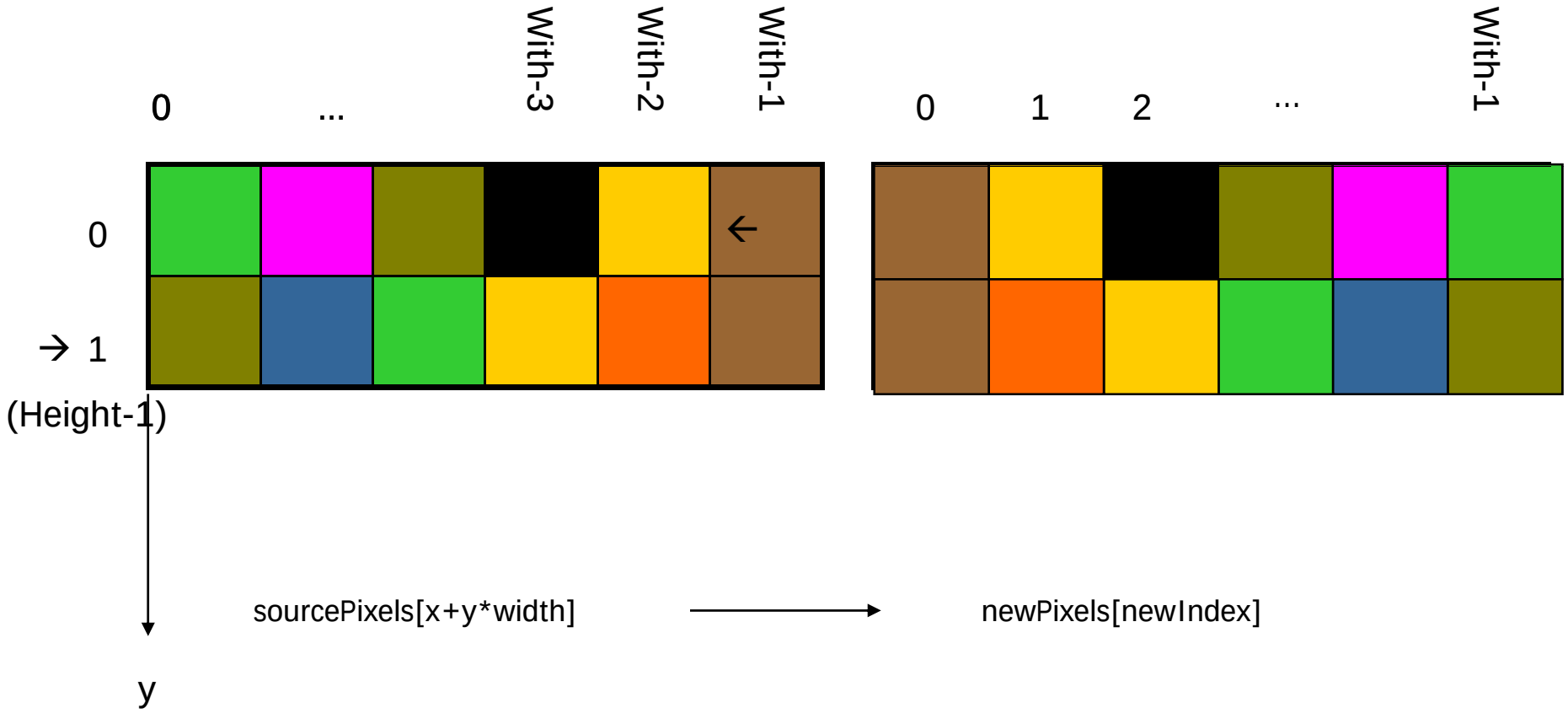
Example: Create an Image from Copied Pixels

```
void setup(){

  PImage img = loadImage("face.jpg");
  size(img.width*2, img.height);
  image(img, 0, 0);

 //Create & display copied image
  PImage cpImg = createImage(img.width, img.height, RGB);
  cpImg.pixels = copyPixels(img.pixels, img.width,
   img.height);

  image(cpImg, img.width, 0);
}
```

Mirroring an Image

- Mirroring an Image vertically (i.e. along a vertical axis)

Demo: Mirroring an image vertically



Implementation: Mirroring an Image Vertically

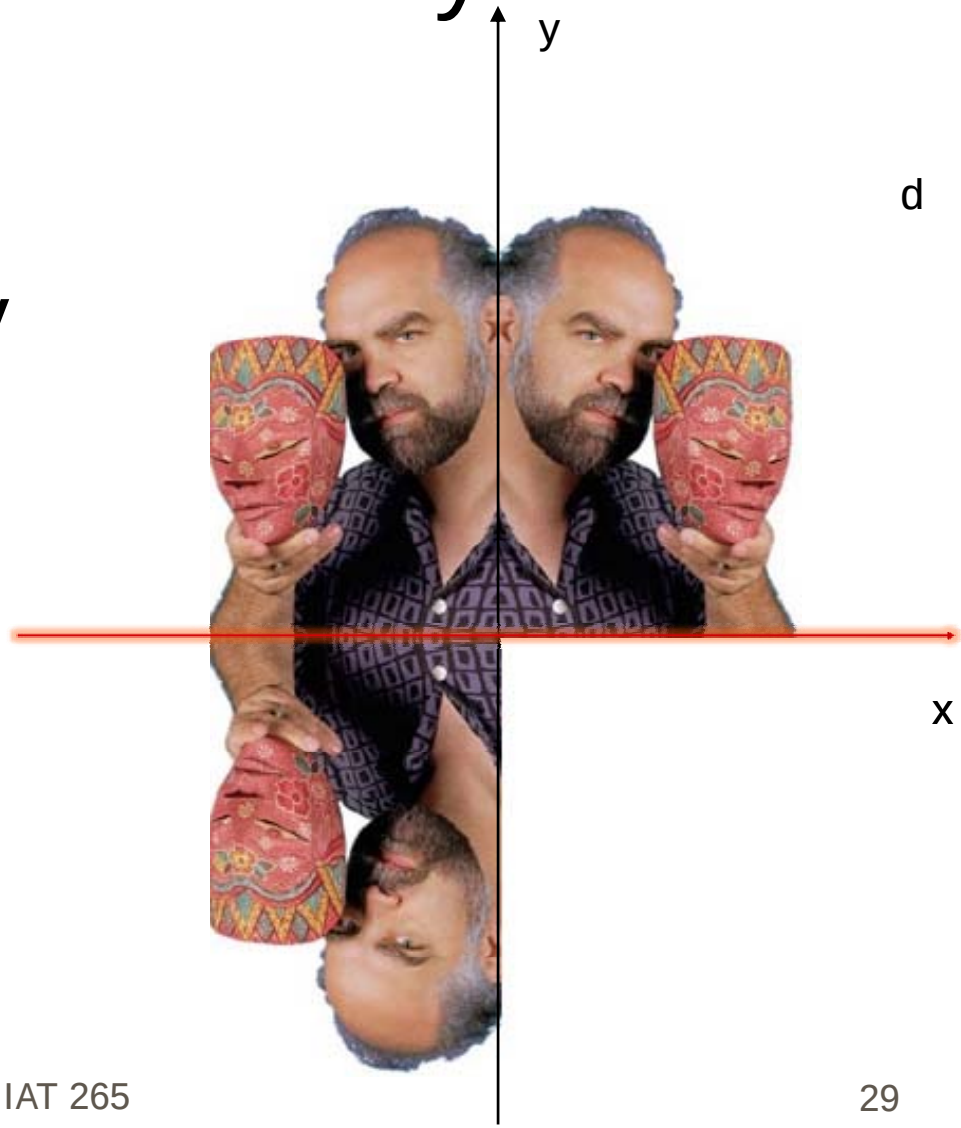
```
// Create method to mirror pixels vertically, left-to-right
int[] mirrorVertical(int srcPixels[], int w, int h) {
   int newPixels[] = new int[w * h];
   int newIndex = 0;

   //Loop forward across the rows
   for (int y = 0; y < h; y++)
       //for each row loop backward through the columns
       for (int x = w-1; x >= 0 ; x--){
              newPixels[newIndex] = srcPixels[x + y * w];
              newIndex++;
       }

   return newPixels;
}
```

Create an Image from Mirrored Pixels

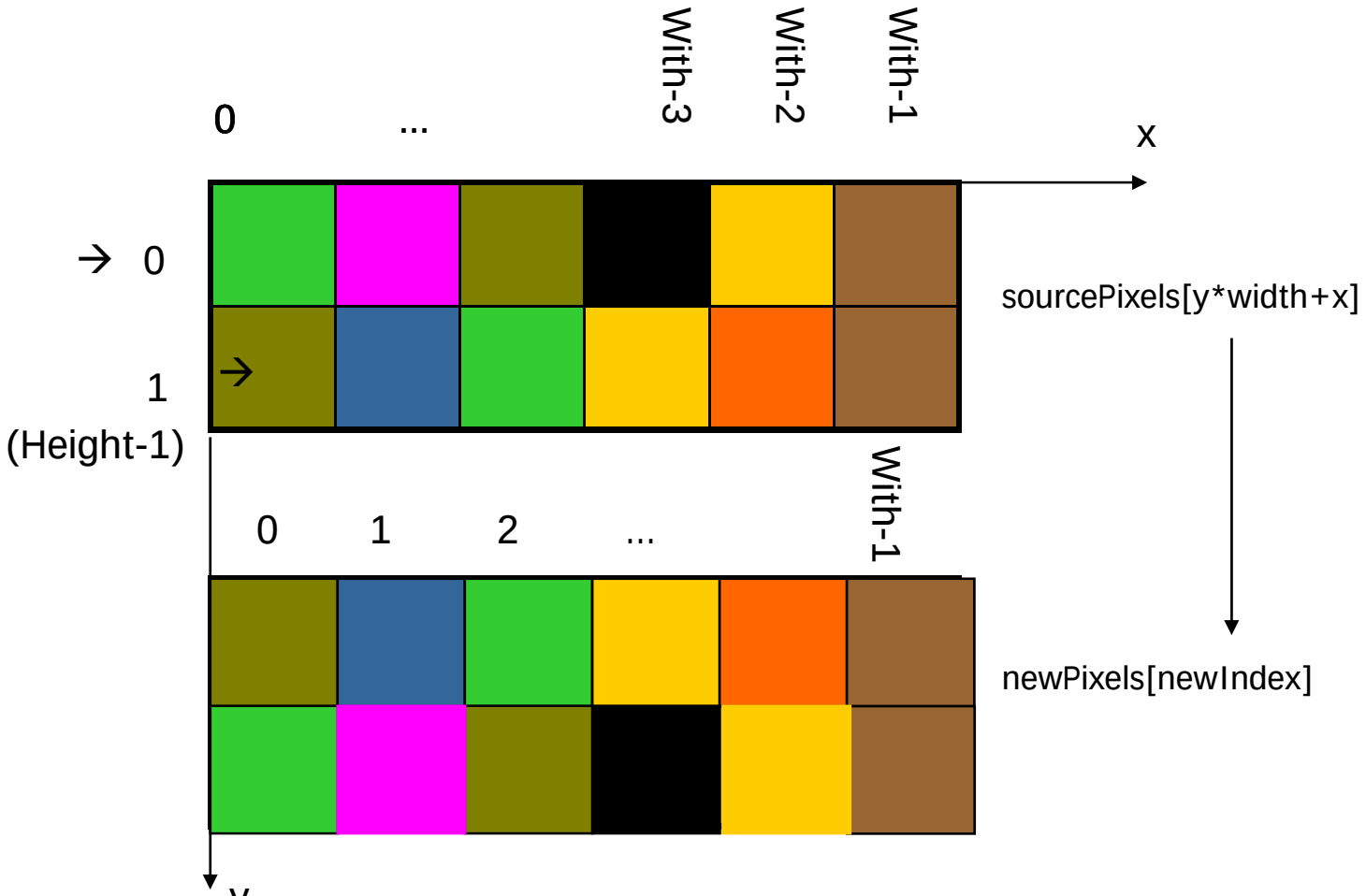
```
void setup(){

  PImage img = loadImage("face.jpg");
  size(img.width*2, img.height);
  image(img, 0, 0);

  //Create & display vertically mirrored image
  PImage cpImg = createImage(img.width, img.height, RGB);
  cpImg.pixels = mirrorVertical(img.pixels, img.width,
   img.height);

  image(cpImg, img.width, 0);
}
```

Mirror Horizontally

- Mirroring an Image horizontally (i.e. along a horizontal axis, from top to bottom)



Demo: Mirror Horizontally



Implementation: Mirroring an Image Horizontally

```
// Method to mirror pixels horizontally, top-to-bottom
int[] mirrorHorizontal(int srcPixels[], int w, int h) {
   int newPixels[] = new int[w * h];
   int newIndex = 0;

   //Loop backward through all the rows
   for (int y = h-1; y >=0; y--)
       //for each row loop forward through the columns
       for (int x = 0; x < w ; x++){
              newPixels[newIndex] = srcPixels[x + y * w];
              newIndex++;
       }

   return newPixels;
}
```

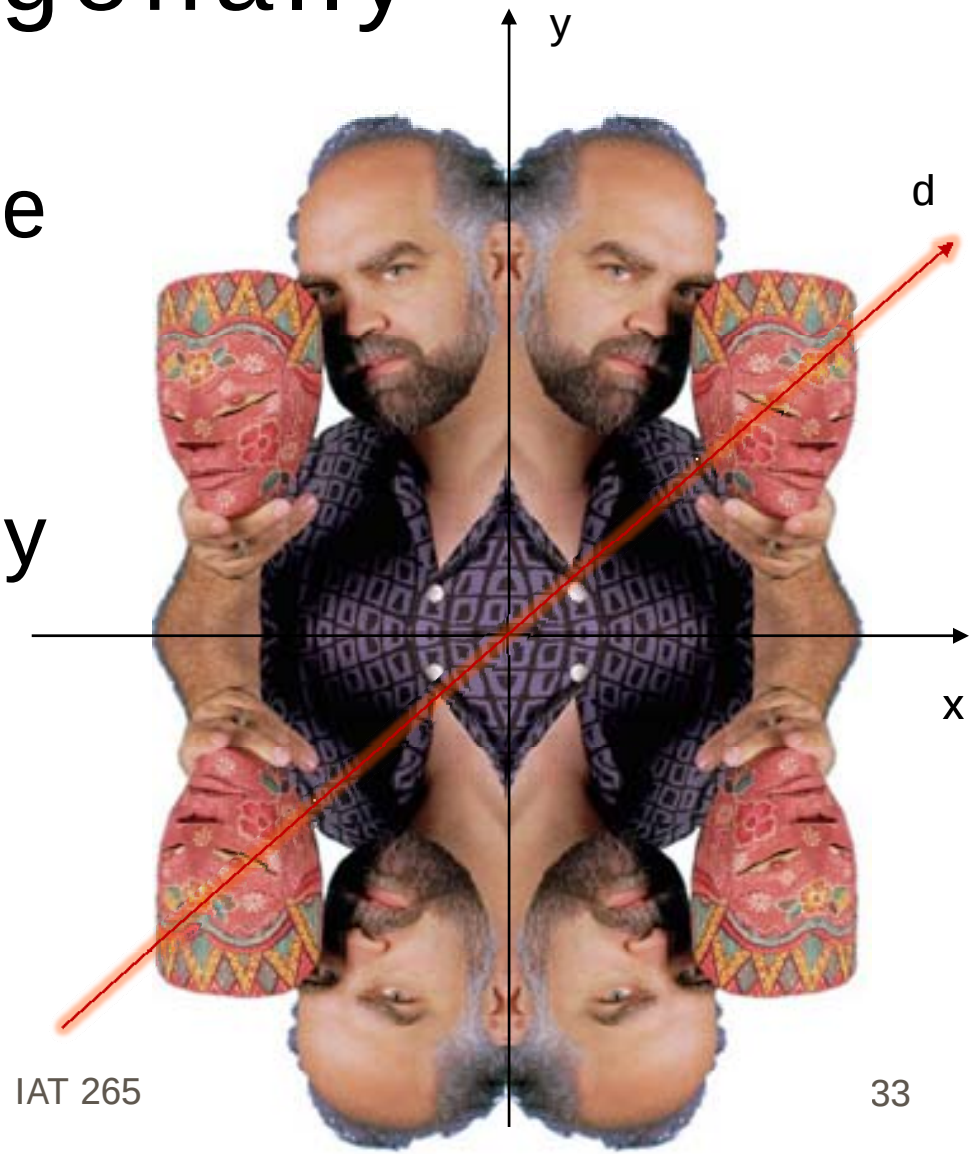
Create an Image from Mirrored Pixels

```
void setup(){

  PImage img = loadImage("face.jpg");
  size(img.width*2, img.height*2);
  image(img, 0, 0);
   //Create & display vertically mirrored image
   …
  //Create & display horizontally mirrored image
  PImage cpImg1 = createImage(img.width, img.height, RGB);
  cpImg1.pixels = mirrorHorizontal(img.pixels, img.width,
   img.height);
  image(cpImg1, 0, img.height);
}
```

Homework: Mirror Diagonally

- Mirroring an Image diagonally

  (i.e. along an axis in-between x and y axes)



PFont

- **`PFont`** is the Processing class for manipulating fonts
  - Like **`PImage`** for images

- Use PFont with functions:
  - **`PFont loadFont(String fileName)`** – loads a font
  - **`textFont(PFont font, int size)`** – sets the current font
  - **`text(String str, int x, int y)`** – draws a string in the current font at the specified location

Simple example

```
// the fonts must be located in the data directory
PFont arial = loadFont("Arial-BoldMT-48.vlw");
PFont viva = loadFont("Vivaldii-48.vlw");
textFont(arial, 44);
text("Eric", 10, 30);
textFont(viva, 44);
text("Eric", 10, 60);
```

Use fill() to change the color of text

```
// the fonts must be located in the data directory
PFont arial = loadFont("Arial-BoldMT-48.vlw");
PFont viva = loadFont("Vivaldii-48.vlw");
 fill( 0, 255, 0 );
textFont(arial, 44);
text("Eric", 10, 30);
textFont(viva, 44);
 fill( 255, 0, 0);
text("Eric", 10, 60);
```

textMode sets the alignment

- **`textAlign( LEFT );`**
  - (x, y) is the bottom left corner of the text – the default location for the baseline
- **`textAlign( RIGHT );`**
  - (x, y) is the bottom right corner of the text
- **`textAlign( CENTER );`**
  - (x, y) is the bottom center of the text

Producing dynamic image & text effects

- All the transform methods apply to image and text drawing as well
  - That means we can translate, rotate, and scale image & text
- We can also change text content dynamically!!

Example of Text Effect

```
PImage img ;
PFont  ar ;
float angle =0;
String txt;

void setup()
{
  //Make sure to do loadImage and loadFont
  //in setup() NOT draw(). Important!!
  img = loadImage( "face.jpg" );
  ar = loadFont( "Arial-BoldMT-48.vlw" );
  size( img.width*4, img.height*3 );
}

void draw( )
{
   background(255);
   imageMode(CORNER);
   for( int i = 0 ; i < width ; i += img.width )
      for(int j = 0; j < height; j += img.height)
          image( img, i, j );

  if(mousePressed) {
    fill( 0, 255, 255);
    txt = "Me!";        //dynamic content
  }
  else {
    fill( 255, 0, 0);
    txt = "You?";       //dynamic content
  }

  //rotate text and image
  pushMatrix();
  translate (mouseX, mouseY);
  angle += 0.02;
  rotate( angle);
  imageMode(CENTER);
  image(img, 0, 0);

  textAlign( CENTER );
  textFont(ar, 60);
  text(txt, 0, 0 );
  popMatrix();
}
```

Summary

- Programming concepts
  - Classes
  - Pimage
  - Pixel array: 2D array vs. 1D array
  - Pfont
- Flip images (vertically, horizontally, & diagonally) using nested *for*-loops
- Produce dynamic image & text effects