

Chapter 4: Modifying Pictures using Loops, part 1

Lecture 6

We perceive light different from how it actually is

- Color is continuous

- Visible light is in the wavelengths between 370 and 730 nanometers
 - That's 0.00000037 and 0.00000073 meters

- But we *perceive* light with color sensors that peak around 425 nm (blue), 550 nm (green), and 560 nm (red).

- Our brain figures out which color is which by figuring out how much of each kind of sensor is responding
- Dogs and other simpler animals have only two kinds of sensors
 - They *do* see color. Just *less* color.

Luminance vs. Color

- We perceive **borders** of things, motion, depth via *luminance*
 - Luminance is *not* the amount of light, but our *perception* of the amount of light.
 - We see blue as “darker” than red, even if same amount of light.
- Much of our luminance perception is based on comparison to backgrounds, not raw values.



Luminance is actually *color blind*. Completely different part of the brain does luminance vs. color.

Digitizing pictures as bunches of little dots

- We digitize pictures into lots of little dots
- Enough dots and it looks like a continuous whole to our eye
 - Our eye has limited resolution
 - Our background/depth *acuity* is particularly low
- Each picture element is referred to as a *pixel*



A Picture is a *matrix* of pixels

- It's not a continuous line of elements – that's an *array*
- A picture has two dimensions: Width and Height
- We need a two-dimensional array: a *matrix*

An array

	0	1	2	3
	15	12	13	10

A matrix

	0	1	2	3
0	15	12	13	10
1	9	7	2	1
2	6	3	9	10





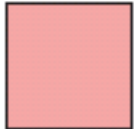
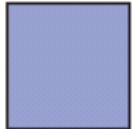


Referencing a matrix

	0	1	2	3
0	15	12	13	10
1	9	7	2	1
2	6	3	9	10

- We talk about positions in a matrix as (x,y), or (horizontal, vertical)
- Element (1,0) in the matrix at left is the value 12
- Element (0,2) is 6

Encoding RGB

- Each component color (red, green, and blue) is encoded as a single byte
- Colors go from (0,0,0) to (255,255,255)
 - If all three components are the same, the color is in greyscale
 - (200,200,200) at (3,1)
 - (0,0,0) (at position (3,0) in example) is black
 - (255,255,255) is white

	0	1	2	3
0	 255, 30, 30	 30, 30, 255	 30, 255, 30	 0, 0, 0
1	 255, 150, 150	 150, 150, 255	 150, 255, 150	 200, 200, 200

16M colors - is it enough?

- We're representing color in 24bits (three colors * 8 bits)
 - That's 16,777,216 (2^{24}) possible colors
 - Our eye can discern millions of colors, so it's probably pretty close
 - But the real limitation is the physical devices: We don't get 16 million colors out of a monitor

COMPARISON

Bits Per Pixel	Number of Colors Available	Common Name(s)
8	256	VGA
16	65536	XGA, High Color
24	16777216	SVGA, True Color
32	16777216 + Transparency	



Size of images

	320 x 240 image	640 x 480 image	1024 x 768 monitor
<i>24 bit color</i>	230,400 bytes	921,600 bytes	2,359,296 bytes
<i>32 bit color</i>	307,200 bytes	1,228,800 bytes	3,145,728 bytes

Manipulating pixels in Jython

getPixel (picture, x, y) gets a single pixel.

getPixels (picture) gets *all* of them in an array.

(To try this, first create a picture object)

```
>>> pixel=getPixel (picture, 1, 1)
```

```
>>> print pixel
```

```
Pixel, color=color r=168 g=131 b=105
```

```
>>> pixels=getPixels (picture)
```

```
>>> print pixels[0]
```

```
Pixel, color=color r=168 g=131 b=105c
```

What can we do with a pixel?

- `getRed()`, `getGreen()`, and `getBlue()` are functions that take a pixel as input and return a value between 0 and 255
- `setRed()`, `setGreen()`, and `setBlue()` are functions that take a pixel as input *and* a value between 0 and 255

We can also get, set, and make Colors

- `getColor()` takes a pixel as input and returns a Color object with the color at that pixel
- `setColor()` takes a pixel as input *and* a Color, then sets the pixel to that color
- `makeColor()` takes red, green, and blue values (in that order) between 0 and 255, and returns a Color object
- `pickAColor()` lets you use a color chooser and returns the chosen color
- We also have functions that can `makeLighter()` and `makeDarker()` an input color

Homework

- READ for next time:
 - Sections 4.3 - 4.7 (p86-104)
 - Note: it's just a lot of techniques using loops; be sure to try the example programs in the book.
 - You will NOT need to turn in the example programs from the book: but there will be another in-class exercise on THURSDAY that IS turned-in, and it assumes you tried the programs from the book.