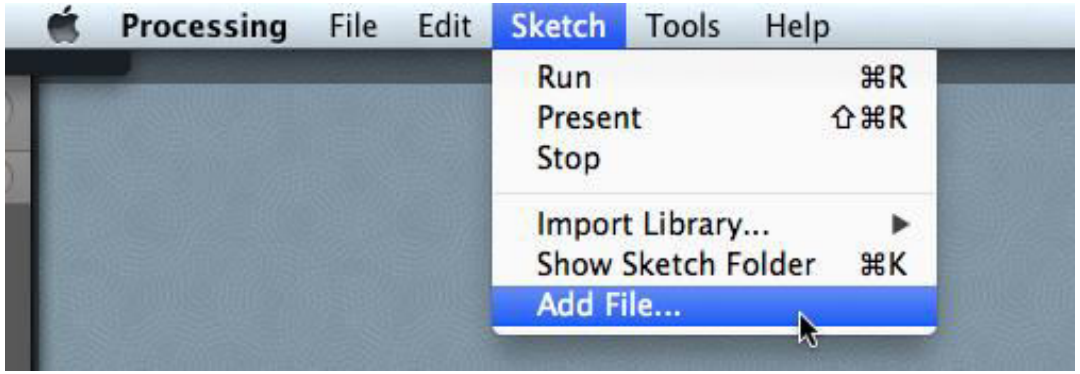
Images and Image Processing

- loading and displaying images
- image filters
- arrays of images
- image pixels

- Slides and demo code for “Images and Image Processing” created by Professor Dan Vogel

Loading Images

1. Add the image file to the sketch's **data folder**.

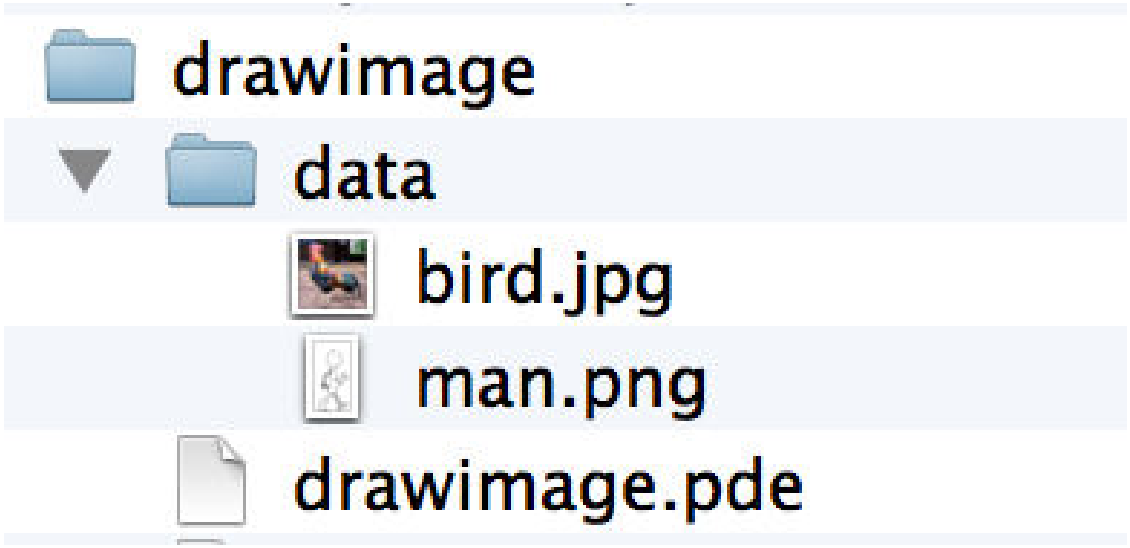


2. Create a `PImage` global variable to hold the image.

```
PImage img;
```

3. Load the image into the variable using `loadImage()`.

```
img = loadImage("bird.png");
```


drawimage
data
bird.jpg
man.png
drawimage.pde

```
The file "pheasant.png" is missing or inaccessible, make sure the
URL is valid or that the file has been added to your sketch and is
readable.
```

(this is a runtime error)

Drawing Images

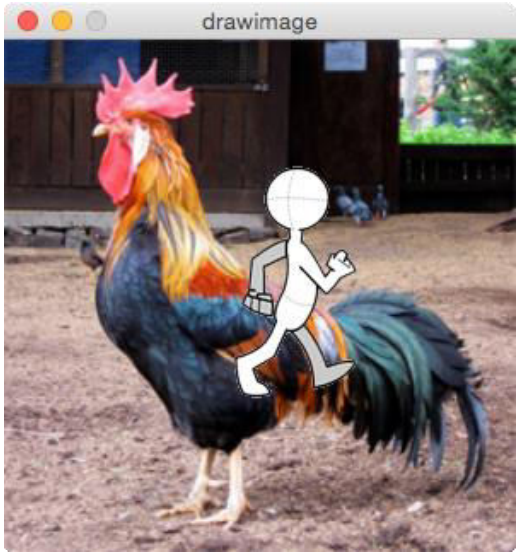
```
image(img, 14, 53);
```

variable storing the image

x and y location to draw image

```
imageMode(CENTER); // or CORNER, etc ..
```

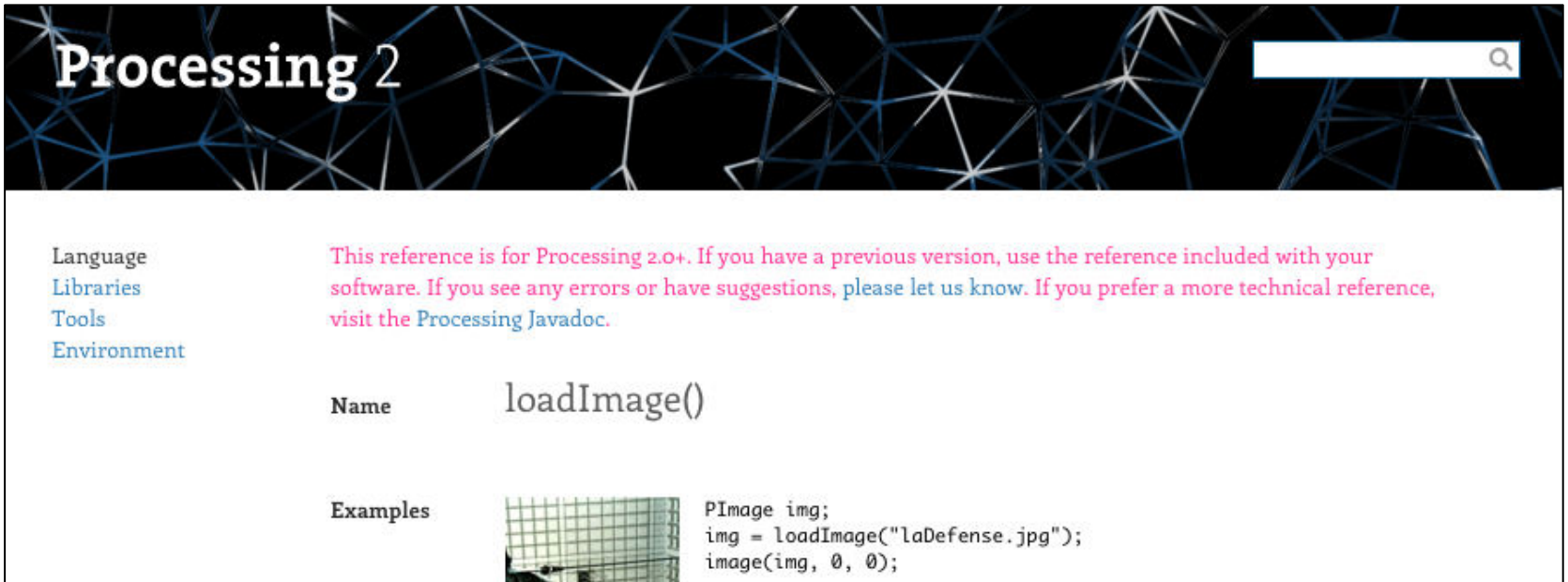
drawimages




```
PImage imgBird;
PImage imgMan;

void setup() {
  size(300, 300);
  imgBird = loadImage("bird.jpg");
  imgMan = loadImage("man.png");
}

void draw() {
 image(imgBird, 0, 0, width, height);
 image(imgMan, mouseX, mouseY);
}
```

loadImage: check the reference

- "In most cases, load all images in setup() to preload them at the start of the program. **Loading images inside draw() will reduce the speed of a program**. Images cannot be loaded outside setup() unless they're inside a function that's called after setup() has already run."
  - from loadImage reference
    right-click "loadImage" in your code and pick Find in Reference



Drawing Images and Scaling

```
image(img, 14, 53, 300, 400);
```

width and height to draw image

Image Size

- Processing images know their own size
  - PImage is an **object**
  - you access image width and height using object "dot syntax"

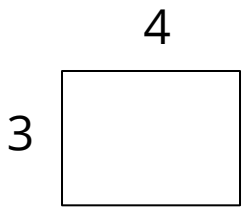
```
PImage img = loadImage("man.png");


println(img.height); // prints 158
println(img.width); // prints 93
```

say "img dot width"

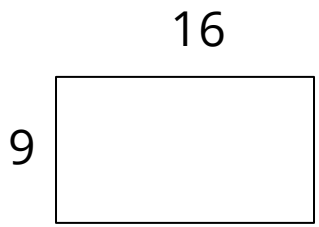
Calculate the Image Aspect Ratio

- aspect ratio is the ratio of width to height

  4:3 is 1.333333



  16:9 is 1.7777777



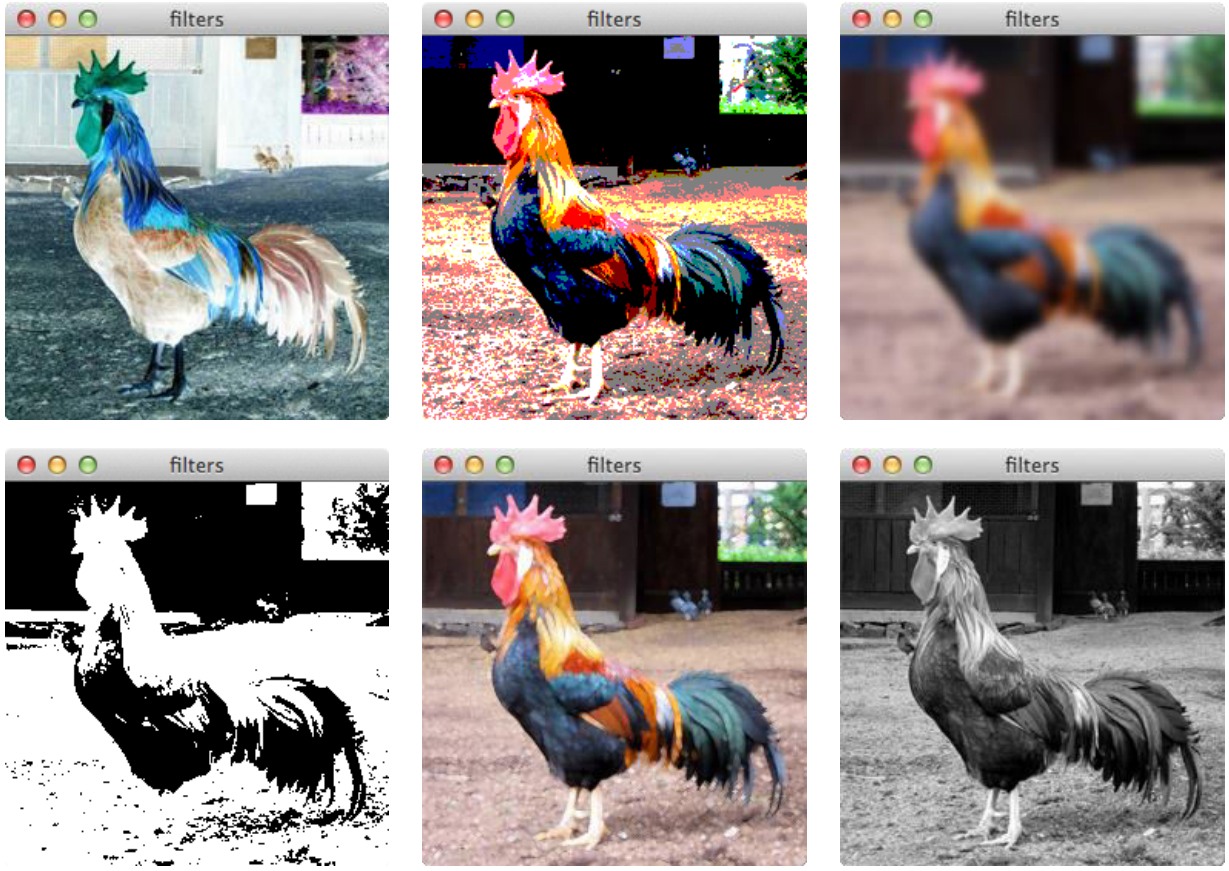
- You can calculate an image aspect ratio like this:

```
float ar = img.width / float(img.height);
```

- Use the aspect ratio to scale images without distorting them

filters

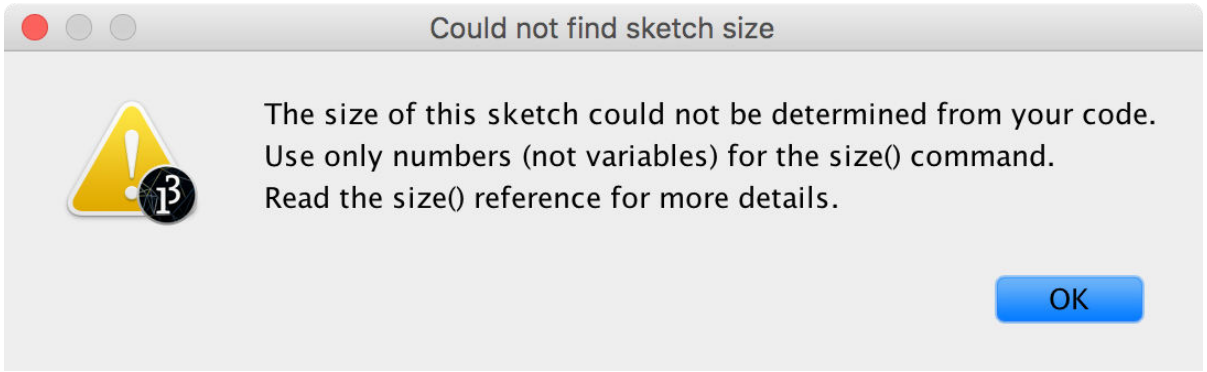
```
filter(INVERT);
filter(BLUR, 3);
...
```



Make canvas size match image size

- In setup():
- Can't use variables when calling size

```
size(img.width, img.height); // crashes
```



- Can "resize" canvas using this code:

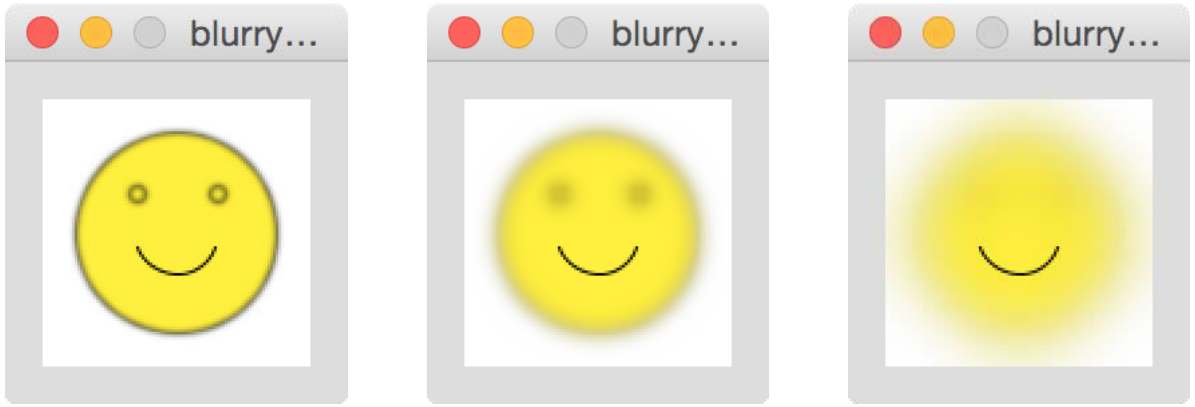
```
// set canvas size to image size
surface.setResizable(true);
surface.setSize(img.width, img.height);
```

ASCII Values

Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

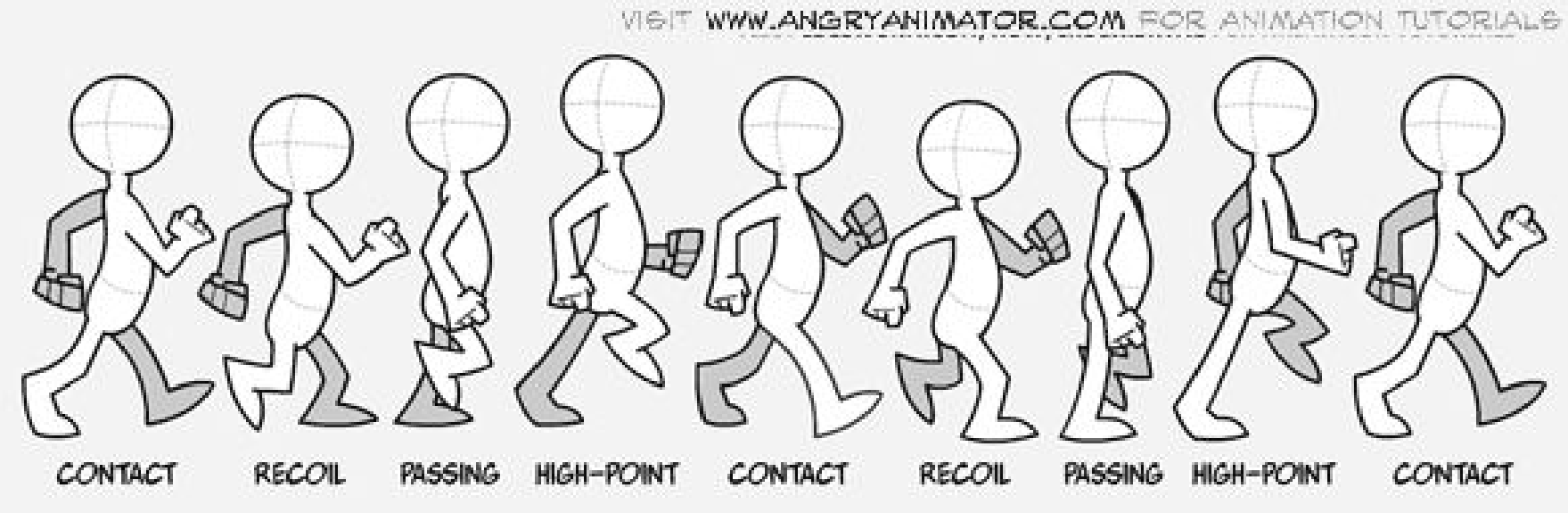
blurrydrawing

```
float iterations = map(mouseX, 0, width, 0, 10);
filter(BLUR, iterations);
```



Walk Cycle using Array of Images

- Keyframe Animation with images
  - (PH 315 – 319)
- Walk cycle
  - http://www.angryanimator.com/word/2010/11/26/tutorial-2-walk-cycle/



Credit: http://www.angryanimator.com/

walkcycle

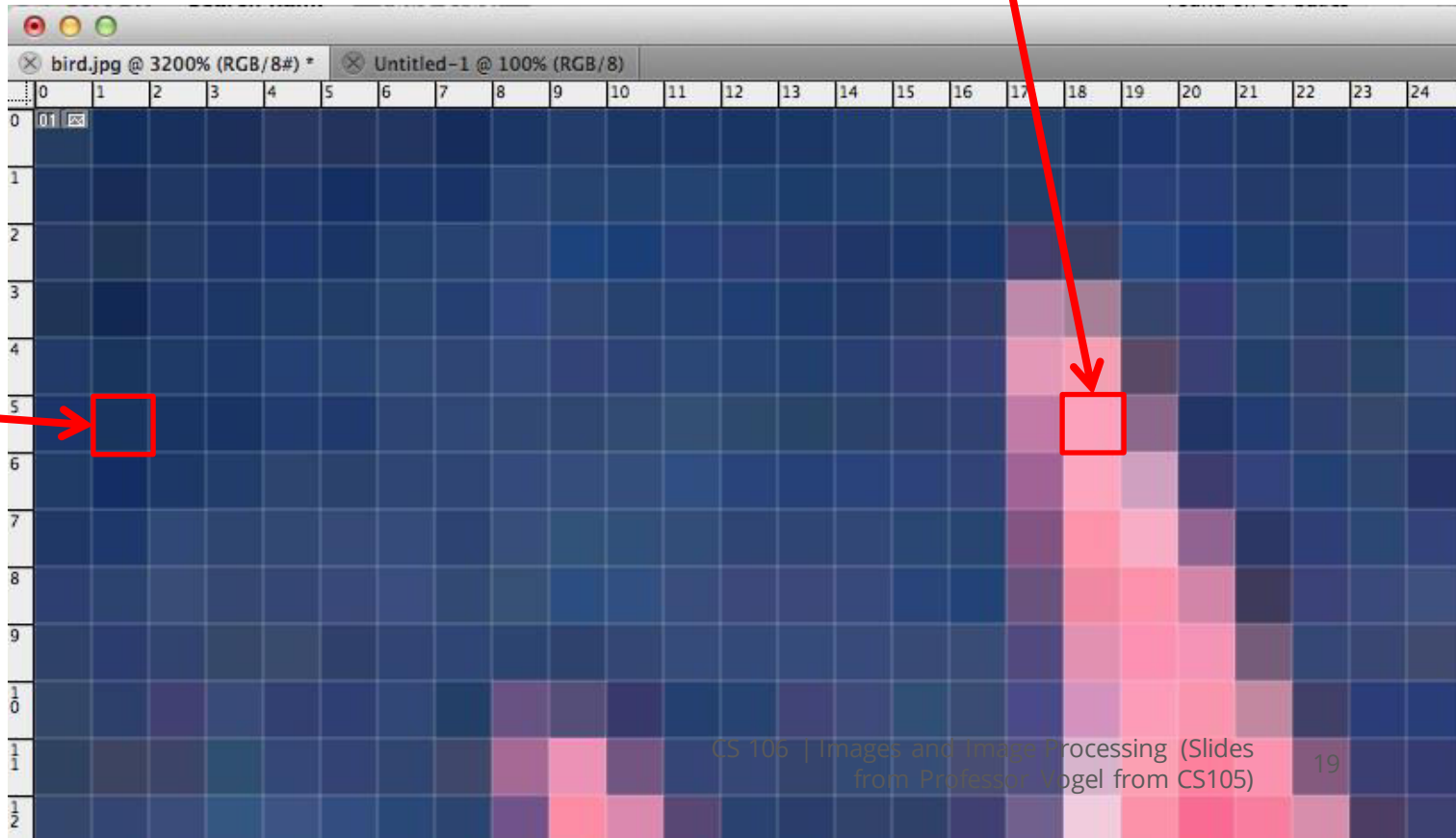

```
// store the walkcycle image sequence
PImage[] walkcycle = new PImage[8];
...

// load all the walk cycle images
walkcycle[0] = loadImage("1.png");
walkcycle[1] = loadImage("2.png");
...

// draw the image in one element of the array
image(walkcycle[frameToShow], x, 0);

x += 12;

// move to the next index in the array
frameToShow++;
```

Image Pixels

- an image **can be indexed** as a two-dimensional array of RGB color values

position 18, 5
(255, 161, 189)

position 1, 5
(31, 58, 103)



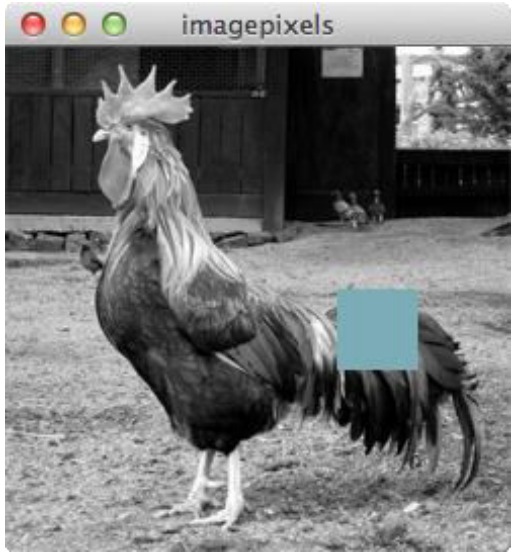
“Getting” Pixel Colour

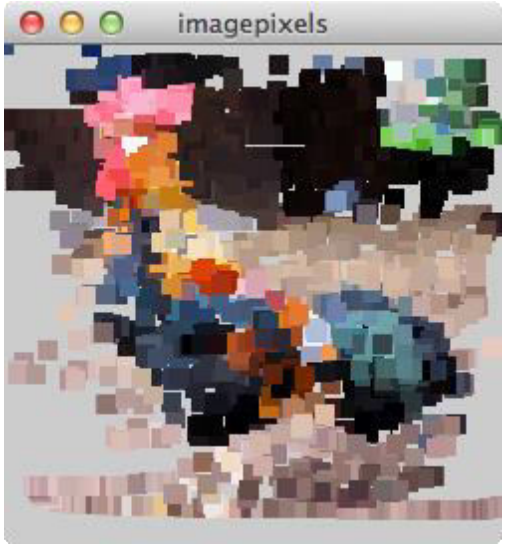
- use dot syntax to “get” colour at pixel location
  - returns a `color` type

```
// get the colour of the pixel
color pixelColour = img.get(18, 5);

// use the colour in a fill
fill(pixelColour);
```

imagepixels

```
// get the colour of the pixel at mouse position
color pixelColour = img.get(mouseX, mouseY);

// draw rectangle using pixel colour
fill(pixelColour);
rect(mouseX, mouseY, 40, 40);
```





Extracting Red, Green, Blue

- extract red, green, and blue values from a `color`

```
// get the colour of the pixel
color pixelColour = img.get(18, 5);


// extract red, green, and blue parts
float r = red(pixelColour);
float g = green(pixelColour);
float b = blue(pixelColour);
```

imagepixelsrgb

```
// get the colour of the pixel at mouse position
color pixelColour = img.get(mouseX, mouseY);

// extract red, green, and blue parts
float r = red(pixelColour);
float g = green(pixelColour);
float b = blue(pixelColour);

// draw r, g, b rectangles
fill(r, 0, 0);
rect(mouseX - 40, mouseY, 40, 40);
fill(0, g, 0);
rect(mouseX, mouseY, 40, 40);
fill(0, 0, b);
rect(mouseX + 40, mouseY, 40, 40);
```

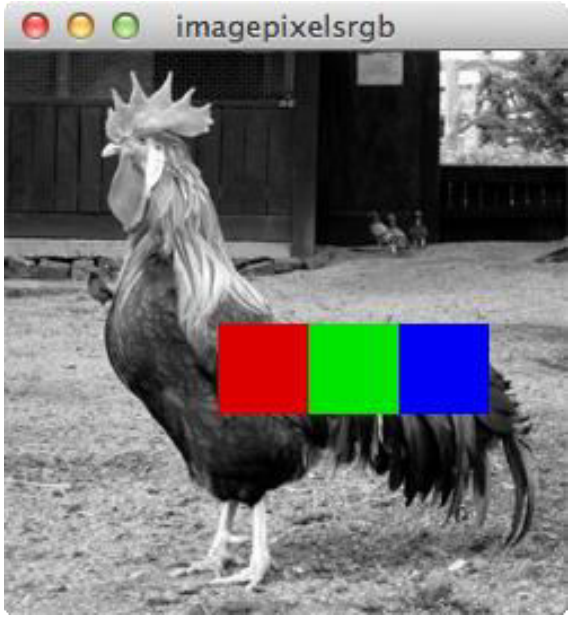
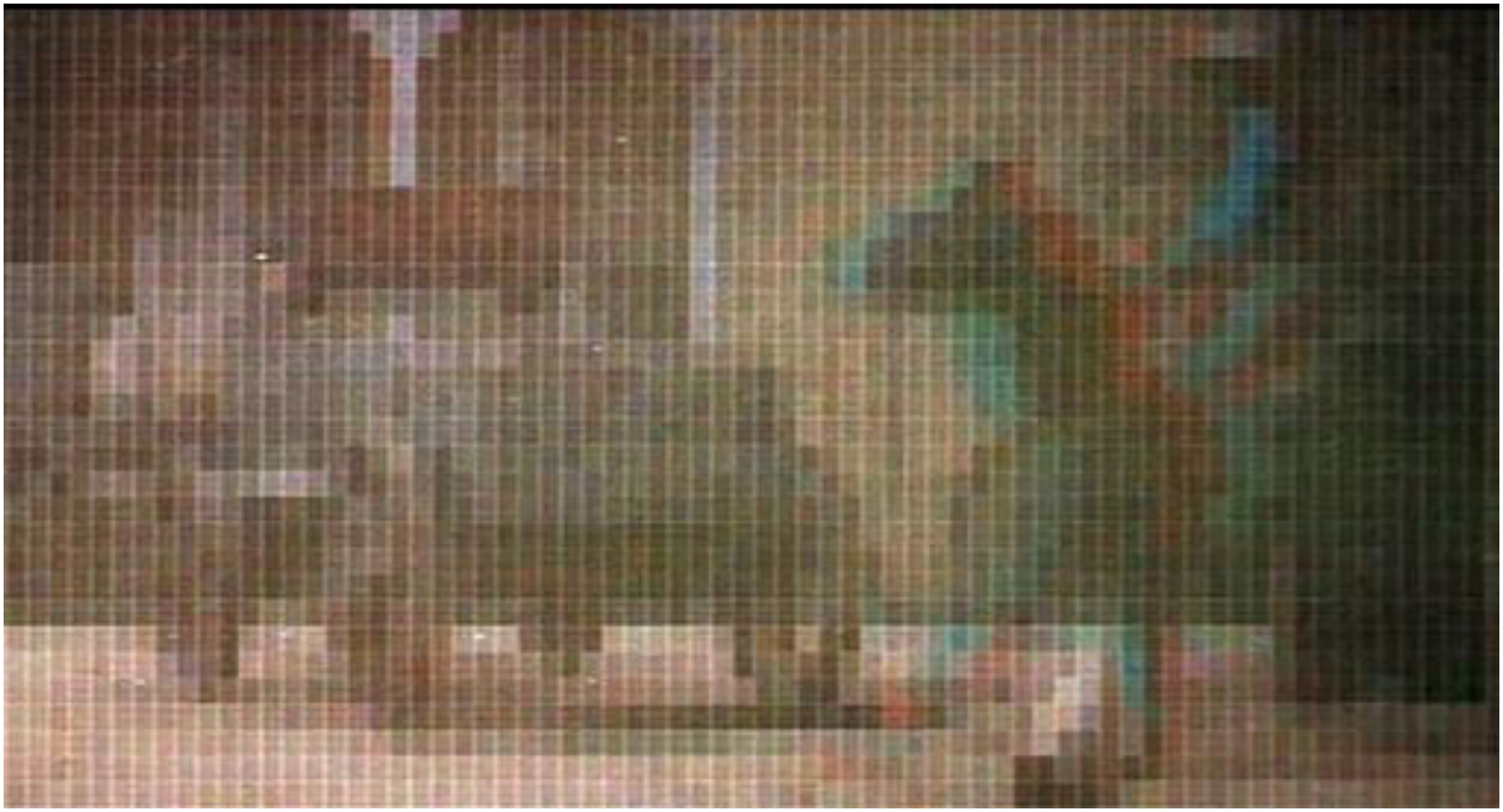


Image Processing

- Manipulating pixels of an image (or sequence of images)
- Since image pixels are in an array, this is another example of the "array operation idiom"

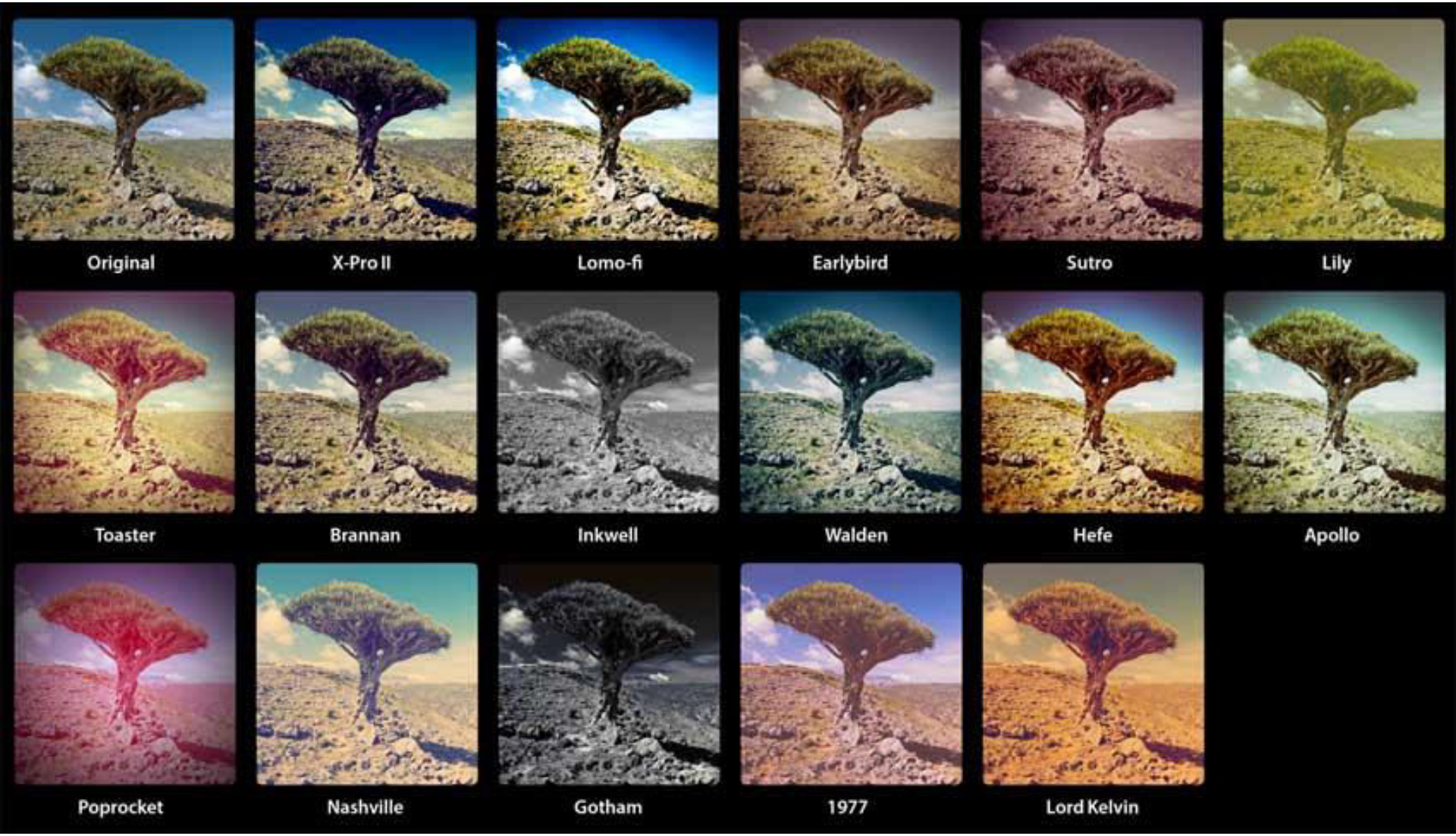


Westworld (1973)
http://youtu.be/MgQtIwEfuGo

Eighties Image Processing
http://youtu.be/8t9ZfCYzbm8?t=5m

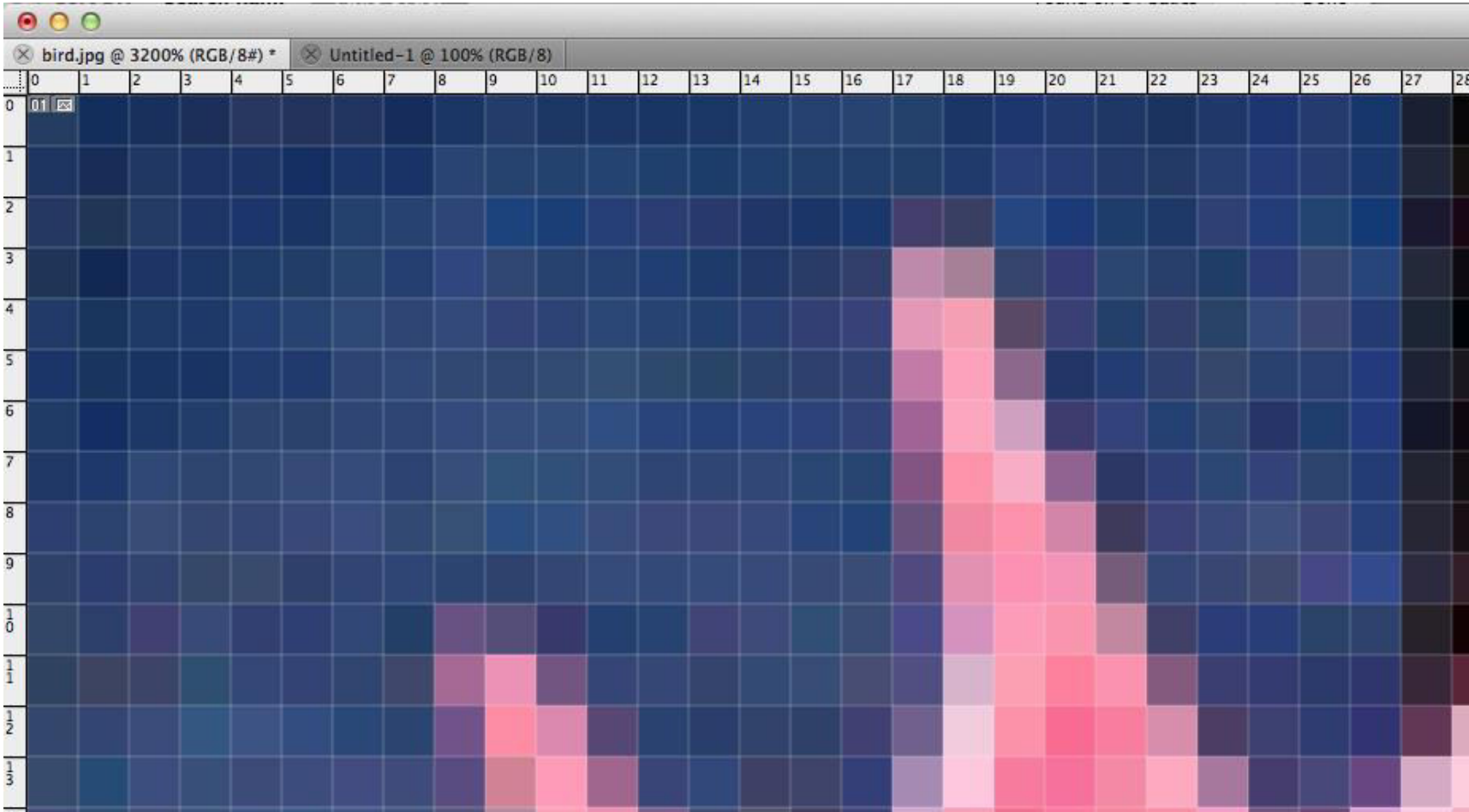
Credit: www.aimersoft.com

http://www.aimersoft.com/


Original
X-Pro II
Lomo-fi
Earlybird
Sutro
Lily
Toaster
Brannan
Inkwell
Walden
Hefe
Apollo
Poprocket
Nashville
Gotham
1977
Lord Kelvin

What is an image?

- An indexed list of colour values



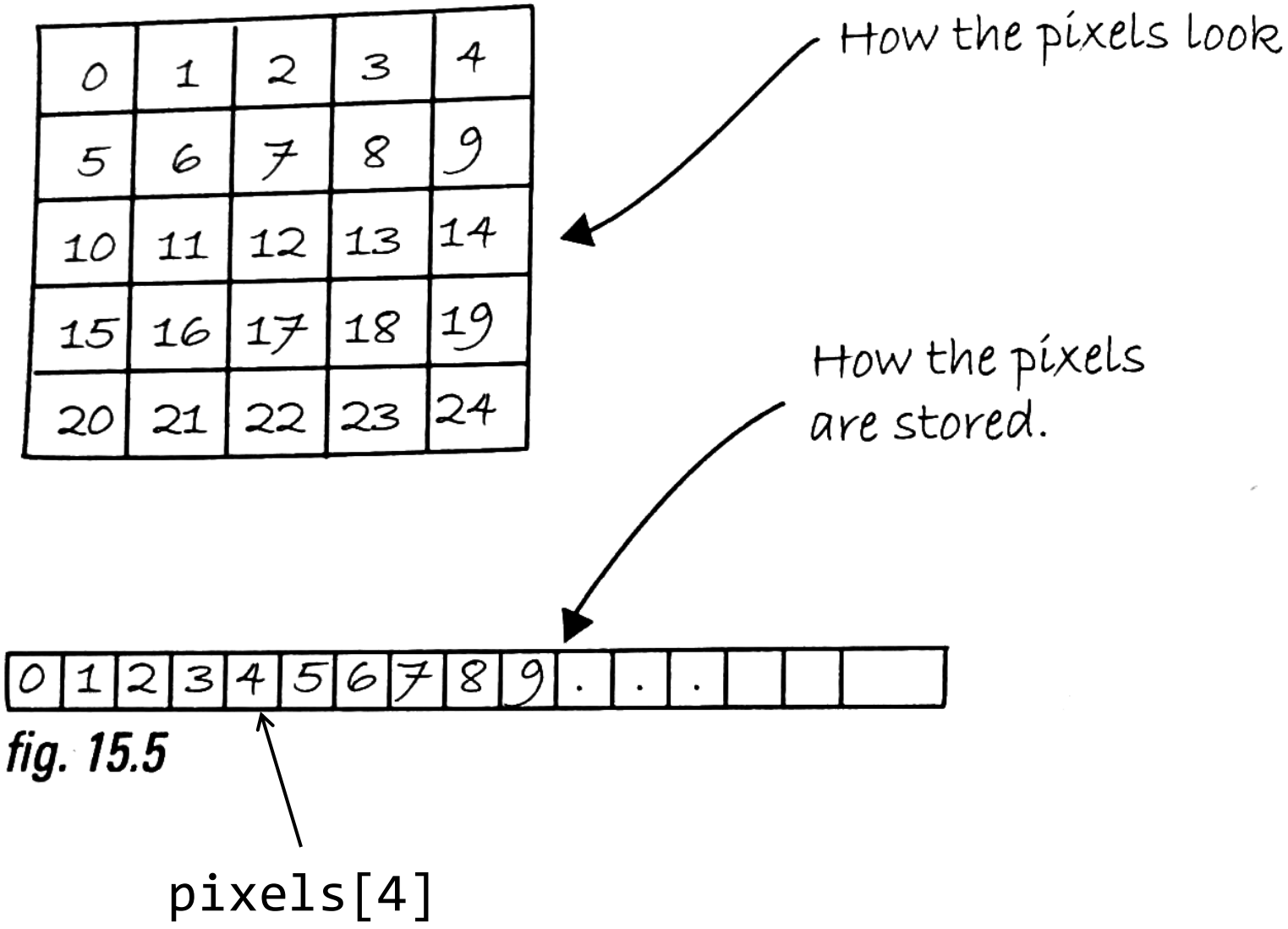


fig. 15.5

Credit: LP p.262

```
int i = x + y * w; // w is image width
```

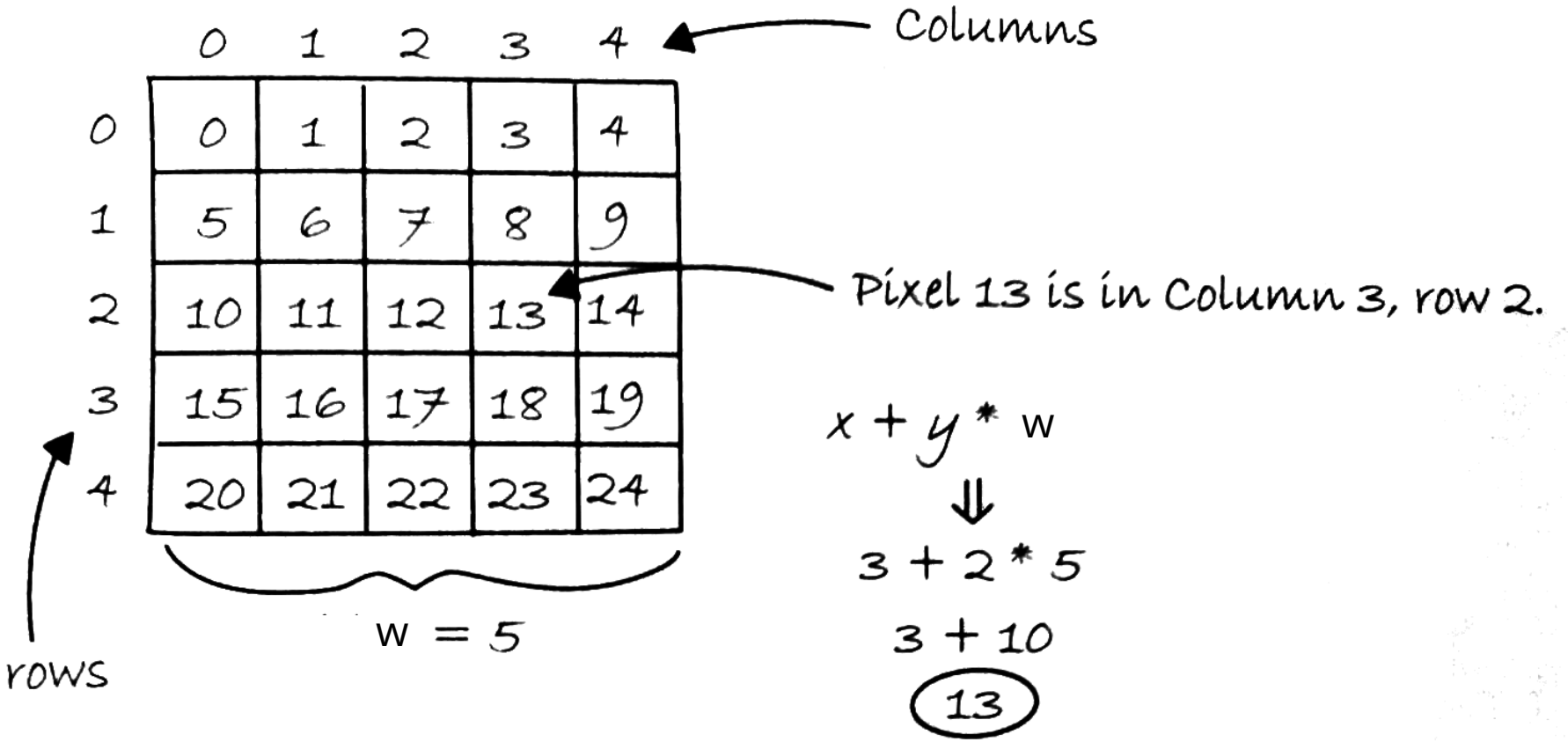
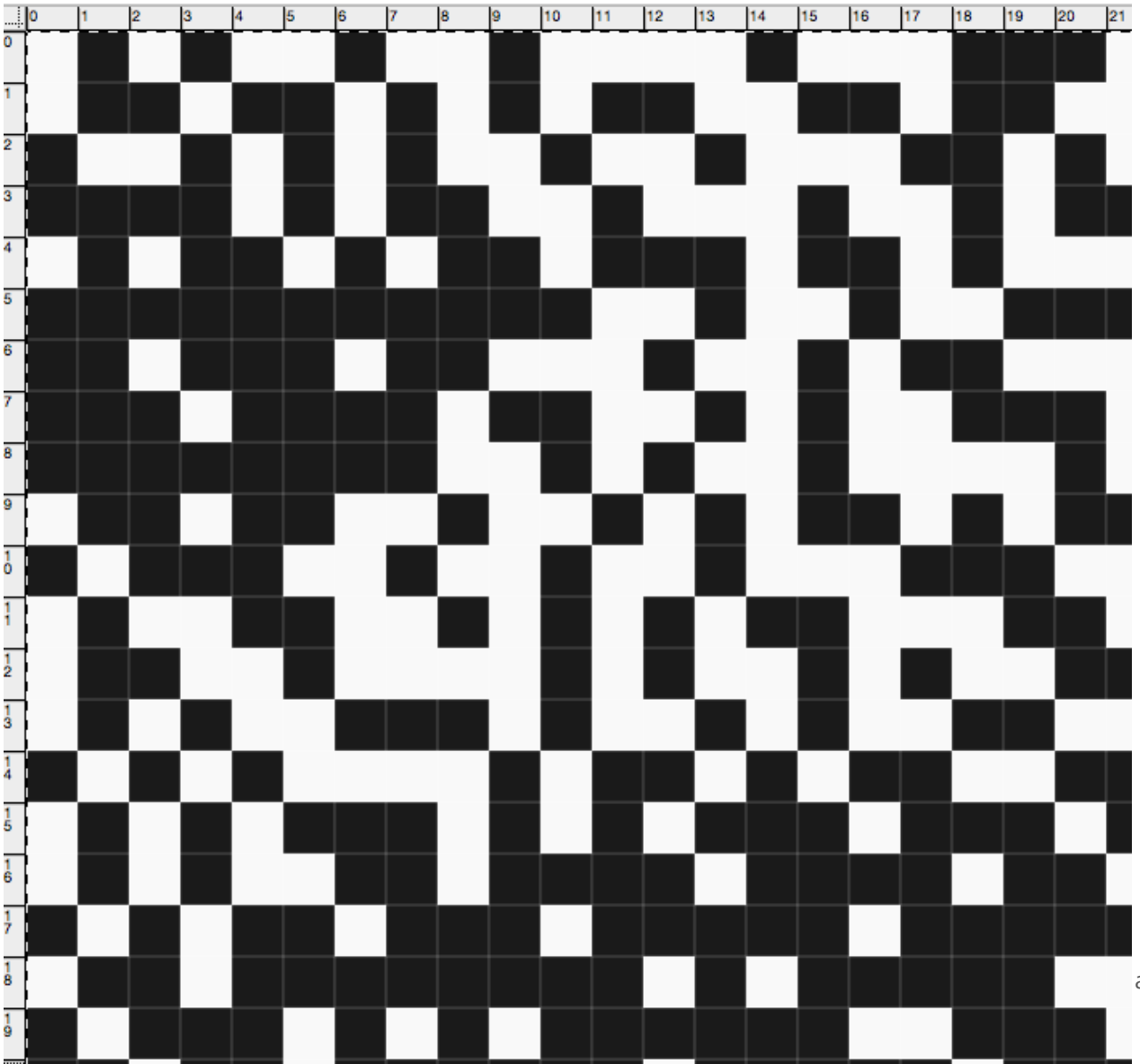


fig. 15.7

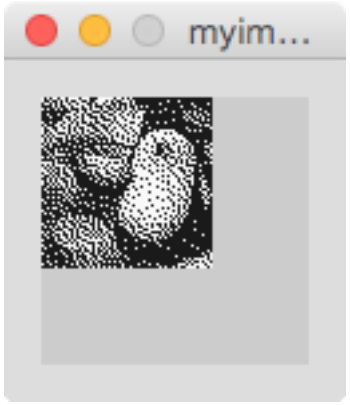
Credit: LP p.263

Black and White (Binary) Images



and Image Processing (Slides Professor Vogel from CS105)

myimage_bw




```
void setup() {
  for (int y = 0; y < h; y++) {
    for (int x = 0; x < w; x++) {
      int i = x + y * w; // indexing
      if (myImage[i] == true)
        stroke(255);
      else
        stroke(0);
      point(x, y);
    }
  }
}
int w = 64;
int h = 64;

boolean[] myImage =
{true,false,true,false,true,true,false,true,true,fal
se,true,true,true,true,false,true,…
```

Grayscale Images

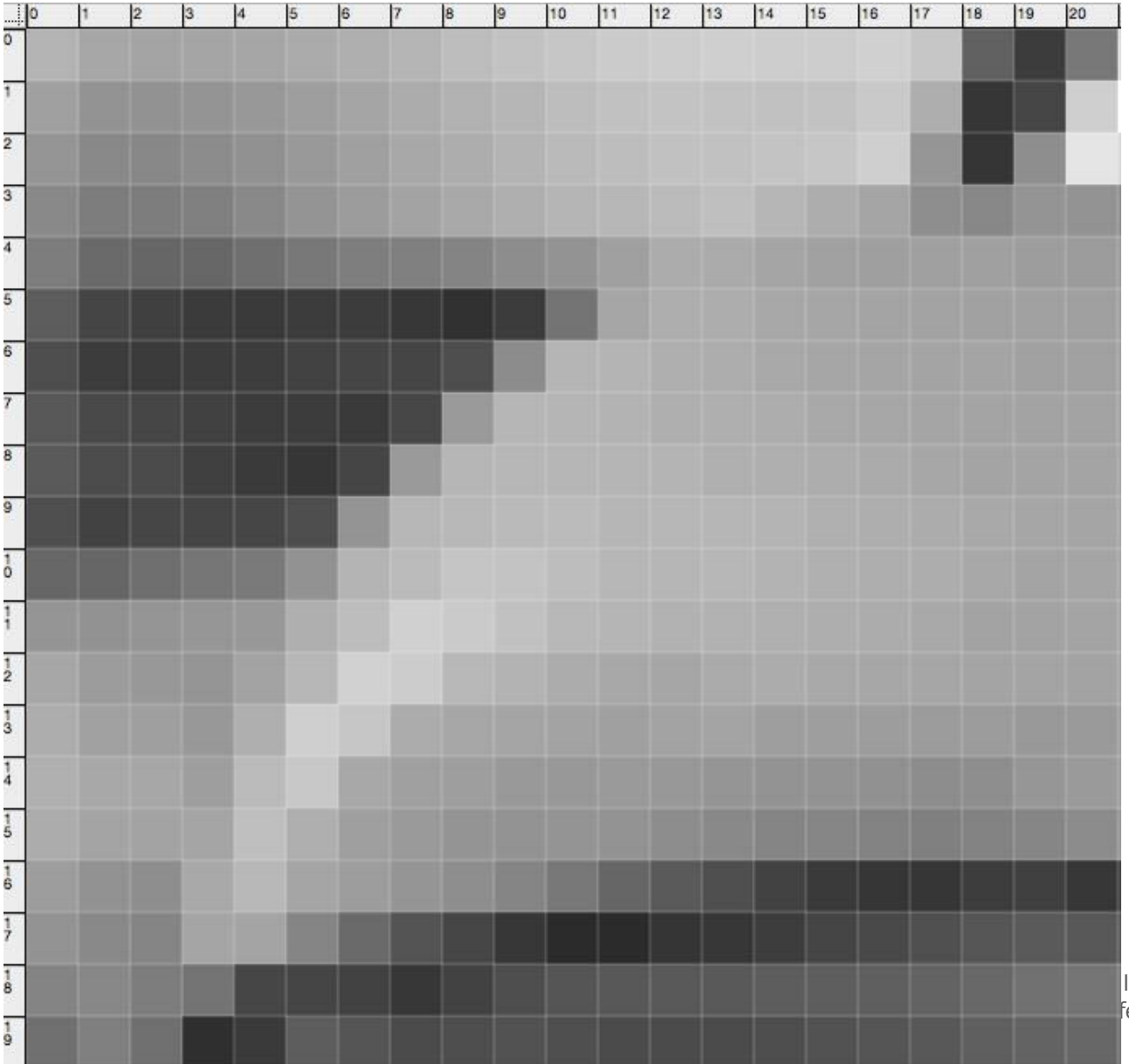
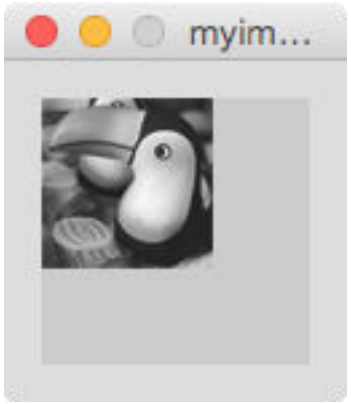


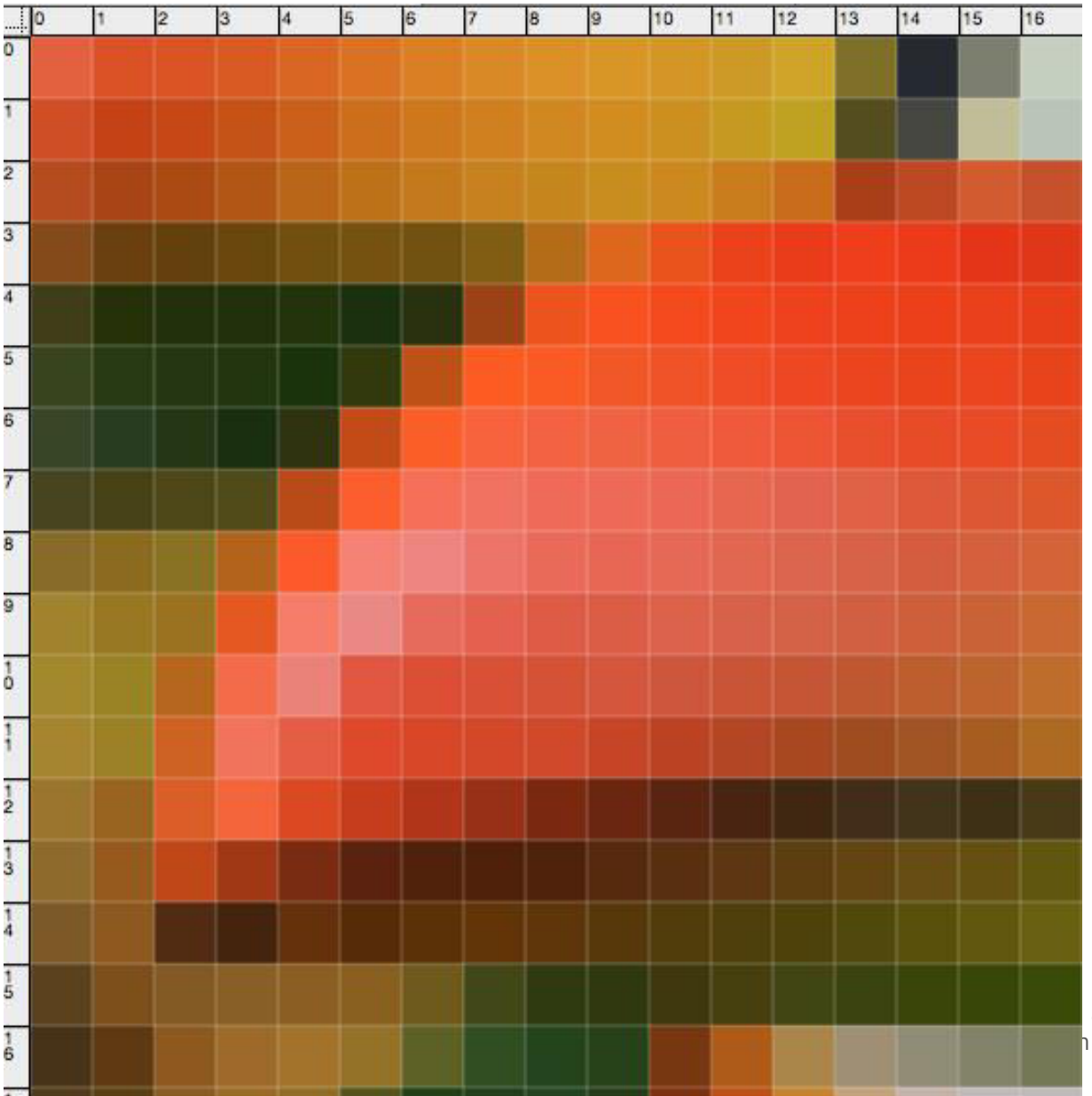
Image Processing (Slides
fessor Vogel from CS105)

myimage_gray



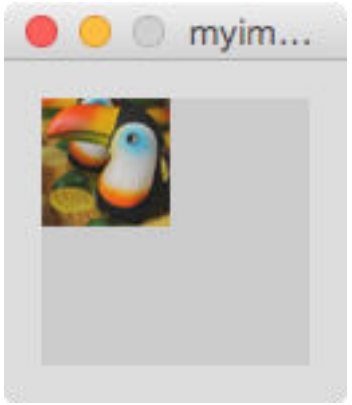

```
void setup() {
  for (int y = 0; y < h; y++) {
    for (int x = 0; x < w; x++) {
      int i = x + y * w; // indexing
      stroke(myImage[i]);
      point(x, y);
    }
  }
}

int w = 64;
int h = 64;

// grayscale image data
int[] myImage = { 162, 148, 144, 145, 146, 152, 157,
163, 173, 179, 184, 190, 192, 194, 192, …
```

RGB Image



and Image Processing (Slides
Professor Vogel from CS105)

myimage_rgb




```
void setup() {
  for (int y = 0; y < h; y++) {
    for (int x = 0; x < w; x++) {
      int i = x + y * w; // indexing
      stroke(myImageR[i], myImageG[i], myImageB[i]);
      point(x, y);
    }
  }
}

int w = 48;
int h = 48;

int[] myImageR = {229,221,220,220,219,220,221, …
int[] myImageG = {97,81,84,91,102,114,127,138, …
int[] myImageB = {56,26,19,16,14,14,14,14,14, …
```

Image Processing Algorithm

for each x,y location in the image:

  get the pixel colour

  do something to the pixel colour (could do nothing)

  set the pixel colour to the "new" colour

Why not use `.set(x, y)`?

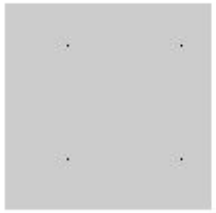
- Setting the color of a single pixel with set(x, y) is easy, but not as fast as putting the data directly into pixels[]. The equivalent statement to set(x, y, #000000) using pixels[] is pixels[y*width+x] = #000000. See the reference for pixels[] for more information.

  from https://processing.org/reference/set_.html

This reference is for Processing 3.0+. If you have a previous version, use the reference included with your software in the Help menu. If you see any errors or have suggestions, please let us know. If you prefer a more technical reference, visit the Processing Core Javadoc and Libraries Javadoc.

Name set()

Examples



```
color black = color(0);
set(30, 20, black);
set(85, 20, black);
set(85, 75, black);
set(30, 75, black);
```

```
for (int i = 30; i < width-15; i++) {
  for (int j = 20; j < height-25; j++) {
    color c = color(204-j, 153-i, 0);
```

imagepixels2

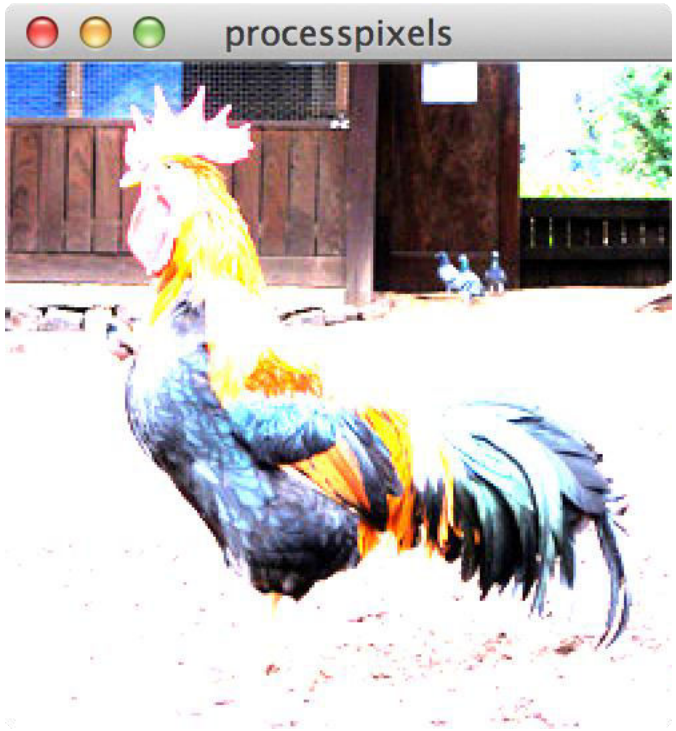
```
// get the colour of the pixel
img.loadPixels();
int i = mouseX + mouseY * width;
color pixelColour = img.pixels[i];
```

processimagepixels

```
img.loadPixels();
for (int y = 0; y < img.height; y++) {
  for (int x = 0; x < img.width; x++) {

    int i = x + y * img.width;
    color pixelColour = img.pixels[i];

    // extract red, green, and blue parts
    float r = red(pixelColour);
    float g = green(pixelColour);
    float b = blue(pixelColour);

    // image processing here
    img.pixels[i] = color(r, g, b);
  }
}
img.updatePixels();
```

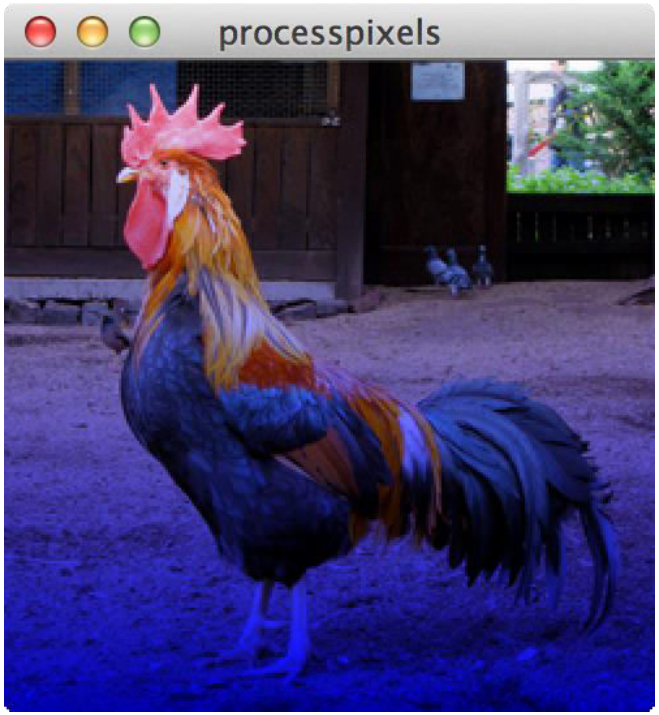
(contrast)

```
img.pixels[i] = color(r * 3, g * 3, b * 3);
```



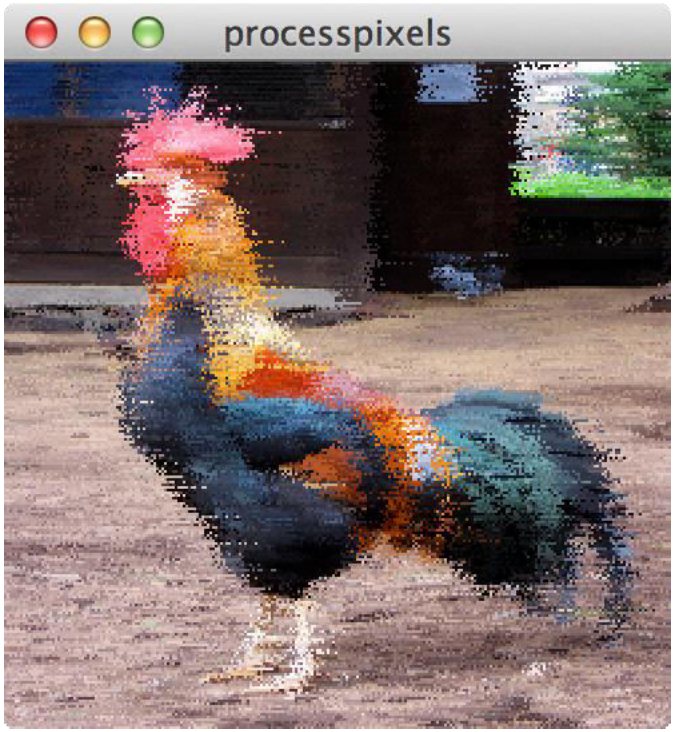
(gradient)

```
float f = map(y, 0, img.height, 1, 0);
img.pixels[i] = color(r * f, g * f, b);
```



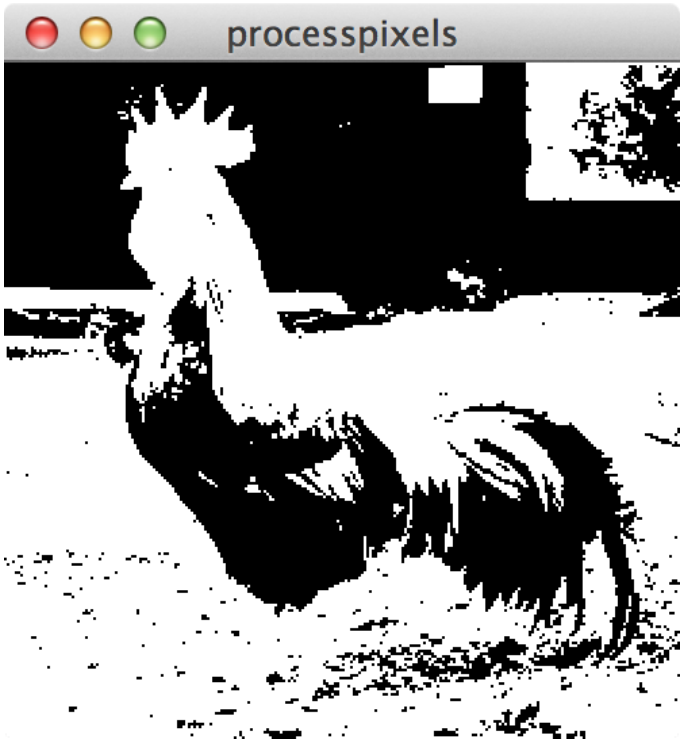
(simple motion blur)

```
int x2 = x + int(random(-5, 5));
x2 = constrain(x2, 0, img.width - 1);
int j = x2 + y * img.width;
img.pixels[i] = img.pixels[j];
```



(thresholding)

```
float f = brightness(pixelColour);
if (f < 128) {
  img.pixels[i] = color(0);
} else {
  img.pixels[i] = color(255);
}
```

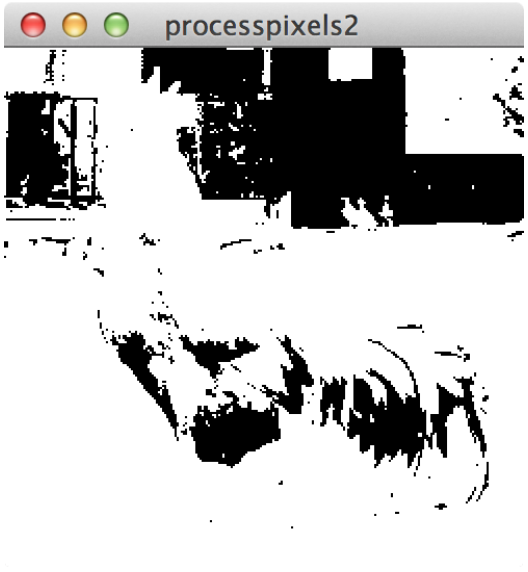


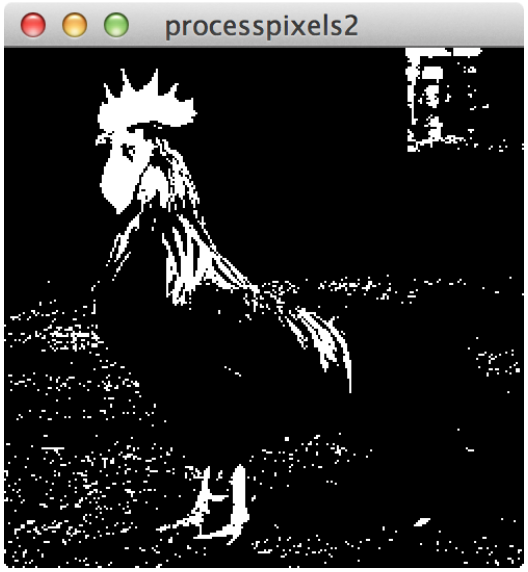
processcanvaspixels

```
loadPixels();
for (int y = 0; y < height; y++) {
  for (int x = 0; x < width; x++) {

    int i = x + y * width;
    color pixelColour = pixels[i];

    // extract red, green, and blue parts
    float r = red(pixelColour);
    float g = green(pixelColour);
    float b = blue(pixelColour);

    // image processing here
    pixels[i] = color(r, g, b);
  }
}
updatePixels();
```

(interactive thresholding)

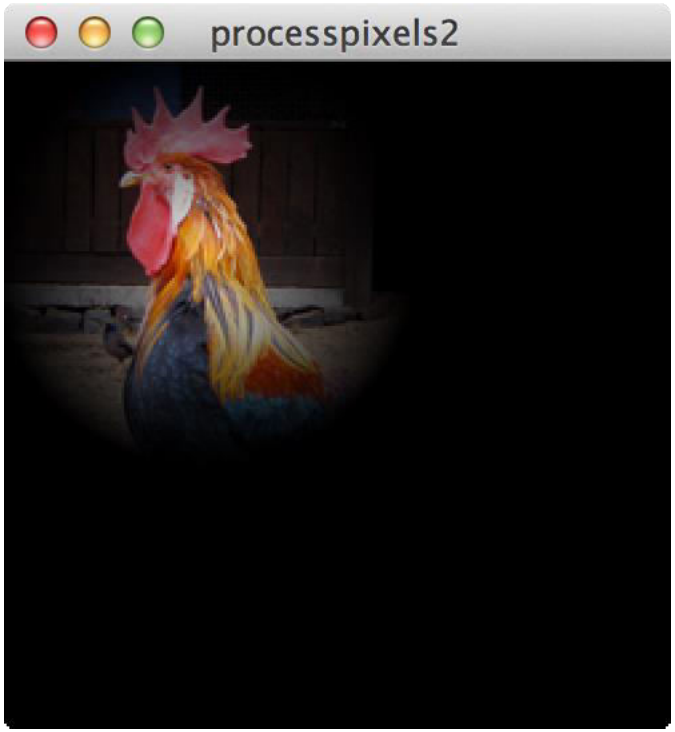
```
float f = brightness(pixelColour);
float threshold = map(mouseY, 0, height, 0, 255);
if (f < threshold) {
  pixels[i] = color(0);
} else {
  pixels[i] = color(255);
}
```





(interactive flashlight effect)

```
float d = dist(mouseX, mouseY, x, y);
float f = map(d, 0, 80, 1.0, 0);
pixels[i] = color(r * f, g * f, b * f);
```



processmiley

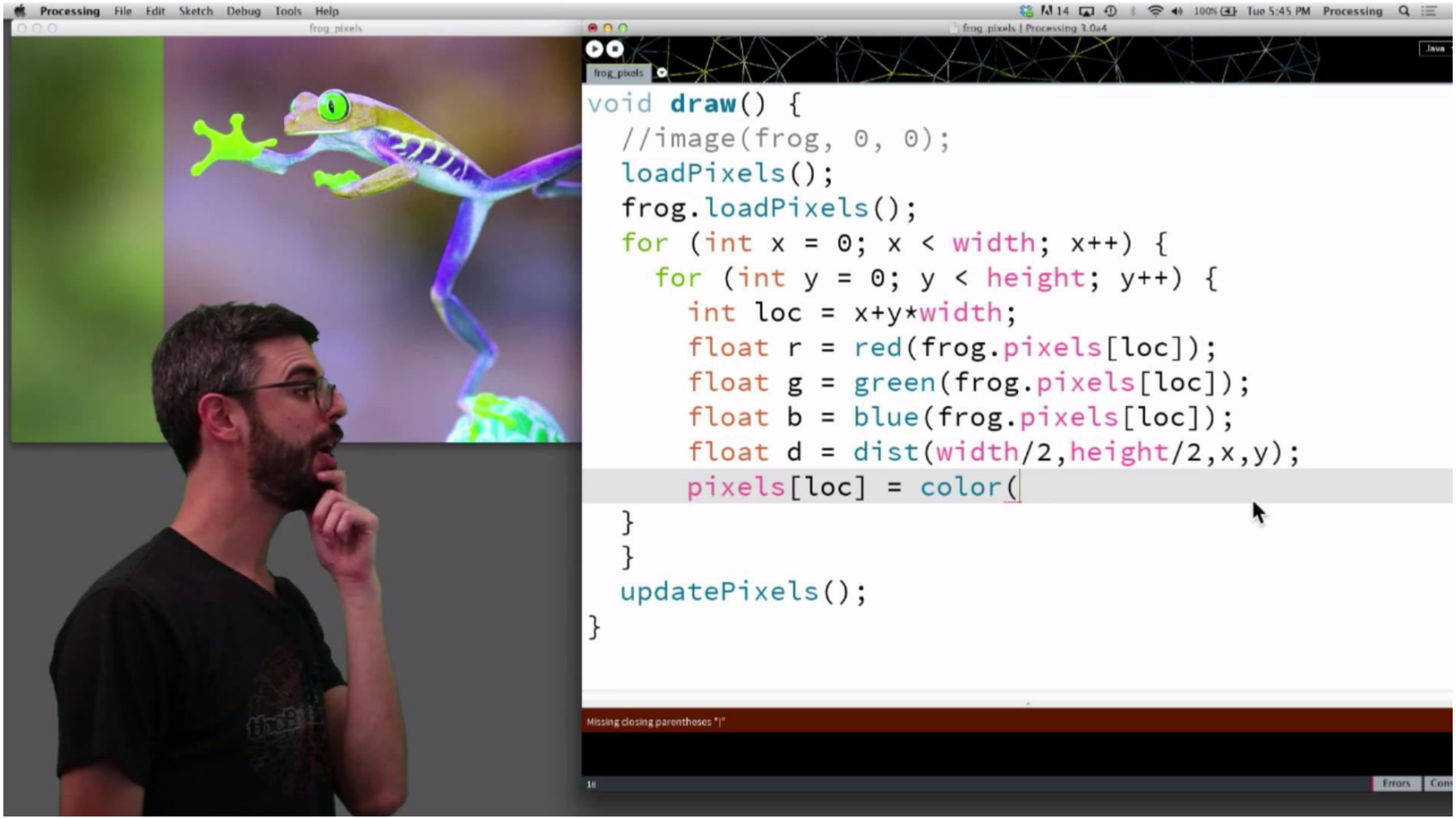


Image Processing with Pixels

http://vimeo.com/108975594