Review

- Exam 1 Review
- Assignment 4 Review
- Models of Motion
- Perspective Drawing

Image Processing

… computing with and about image data,

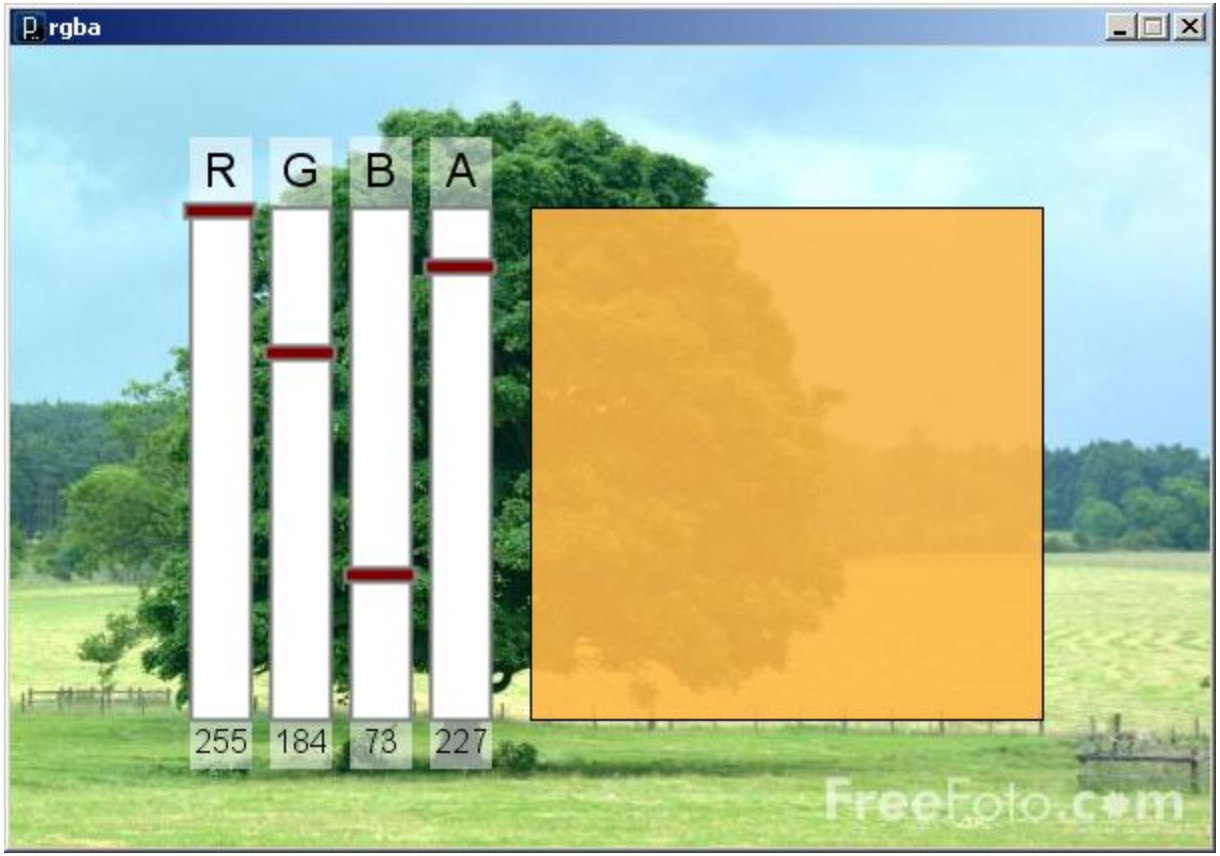
… where "image data" includes the colors and relative color locations that make up an image.

An image is a 2D arrangement of colors

Pixel : Picture Element

Color

- A triple of bytes [0, 255]
  - RGB or HSB
- Transparency (alpha)
  - How to blend a new pixel color with an existing pixel color



The color data type

- In Calico Processing, ‘color’ is a type of data

  … like number, string, boolean, …

- To create one of these ‘color’ things…

```
myTranspColor = color( red, green, blue, alpha )
myColor       = color( red, green, blue )
myTranspGray  = color( gray, alpha )
myGrayColor   = color( gray )
```

- The signatures of functions to make a ‘color’ match that of background(), fill() and stroke(), exactly!

The color data type

- A ‘color’ can be used as an argument to set background(), fill(), and stroke()

```
myRed = color( 255, 0, 0 )
background( myRed )
fill( myRed )
stroke( myRed )
```

The color data type

- It is possible to extract information about a color.

```
myColor = color( 255, 128, 64, 255 )

re = red( myColor )          # Get the red component
gr = green( myColor )        # Get the green component
bl = blue( myColor )         # Get the blue component
al = alpha( myColor )        # Get the alpha component
sa = saturation( myColor ) # Get the saturation
hu = hue( myColor )          # Get the hue
br = brightness( myColor ) # Get the brightness
```

Accessing the pixels of a sketch

- loadPixels()
  - Loads the color data out of the sketch window into an internal data structure of colors
  - The colors in this internal structure can be modified programmatically

- updatePixels()
  - Copies the color data from the internal data structure back to the sketch window

loadPixels() and updatePixels() must bracket all image processing

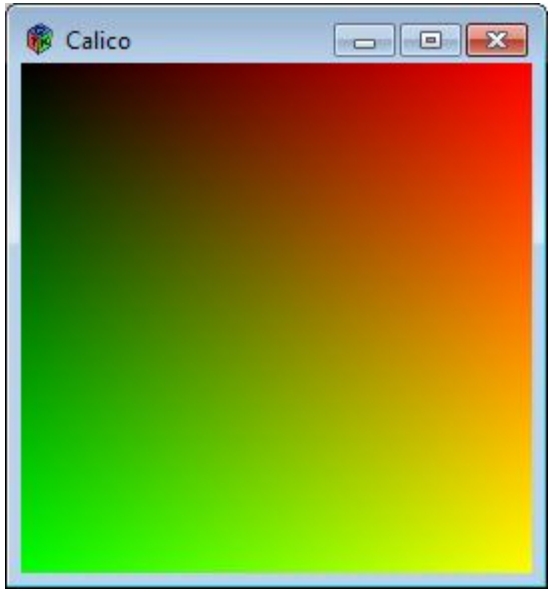
Accessing the pixels of a sketch

- getPixel(i, j)
  - Gets and returns the color at the pixel located at column i, row j.

- setPixel(i, j, color)
  - Sets the color of the pixel at column i, row j, to color, a color object.

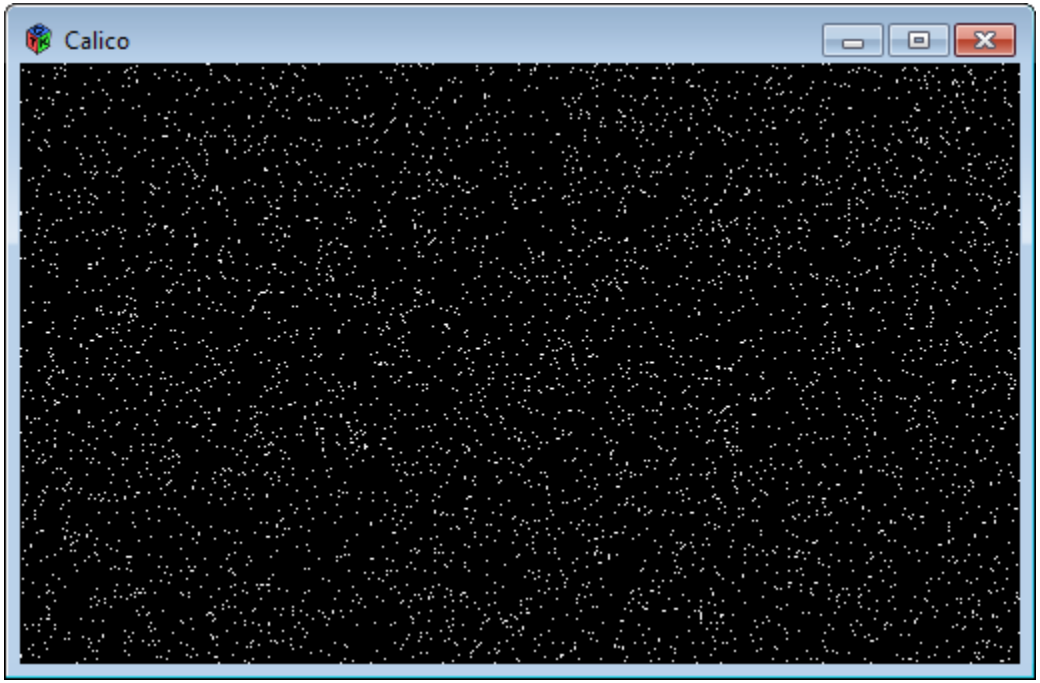
loadPixels() and updatePixels() must bracket all image processing

```
# shade.py
from Processing import *
window(255, 255)

# Load the pixels
loadPixels()

# Set pixel grayscale value to row index
for i in range(255):
    for j in range(255):
        clr = color(i, j, 0)
        setPixel(i, j, clr)

# Update pixels
updatePixels()
```



shade.py

```
# noise.py
from Processing import *
window(500, 300)

# Delay all screen updates
# until redraw is called
immediateMode(False)

# Generate the color white
white = color(255)

def noise():
    background(0)
    loadPixels()
    for i in range(5000):
        x = int( random(500) )
        y = int( random(300) )
        setPixel(x, y, white)
    updatePixels()

    # Render current sketch window
    redraw()

# Generate 1000 frames of random noise
for i in range(1000):
    noise()
```



See also colorNoise.py

Optimizing Pixel Animations

- immediateMode determines when Calico Processing updates the sketch

```
immediateMode(True)
```

… sketch updates as soon as possible

```
immediateMode(False)
```

… sketch update is delayed until `redraw()`

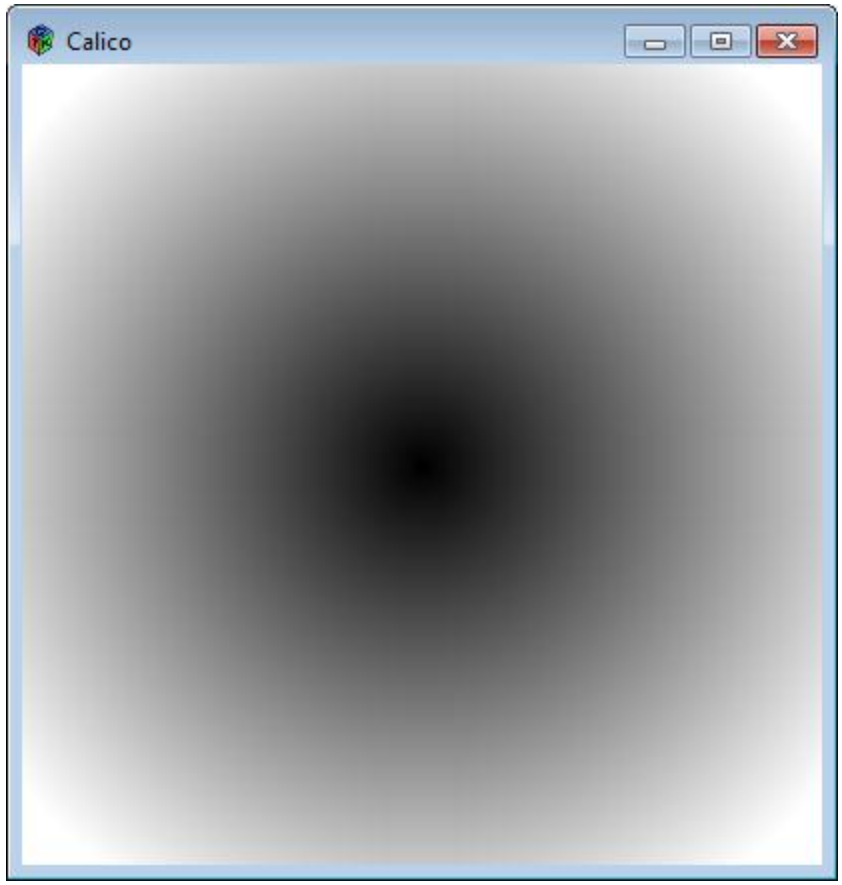
Optimizing Pixel Animations

- `redraw()`
  - tells Calico Processing to update the sketch window immediately

- `delay( milliseconds )`
  - tells Calico Processing to wait for *milliseconds* before proceeding

With immediateMode(False), redraw() and delay(), you can optimize pixel animation behavior in Calico Processing

```
# cone.py
from Processing import *
window(400, 400)

# Get pixel colors from window
loadPixels()

for i in range( 400 ):
    for j in range( 400 ):
        # Compute distance to center.
        b = dist( i, j, 200, 200 )
        b = constrain(b, 0, 255)

        # Set pixel color
        c = color( b )
        setPixel(i, j, c)

# Repaint
updatePixels()
```



```
# ripple.py
from Processing import *
import math

window(400, 400)

# Get pixel colors from window
loadPixels()

for i in range( 400 ):
    for j in range( 400 ):
        # Compute distance to center of sket
        d = dist( i, j, 200, 200 )

        # Compute the sine of the distance a
        b = math.sin(d/5.0)
        b = ((200.0-d)/200.0)*map(b, -1.0, 1.0, 0, 255);

        # Constrain distance to byte range
        b = constrain(b, 0, 255)

        # Set pixel color
        c = color( b )
        setPixel(i, j, c)

# Copy to sketch
updatePixels()
```

Rendering Images in a Sketch

- Image data can be loaded from a file using `loadImage()`, and drawn on a sketch with the `image()` command

```
img = loadImage("myImage.jpg")
image(img, 0, 0)
```

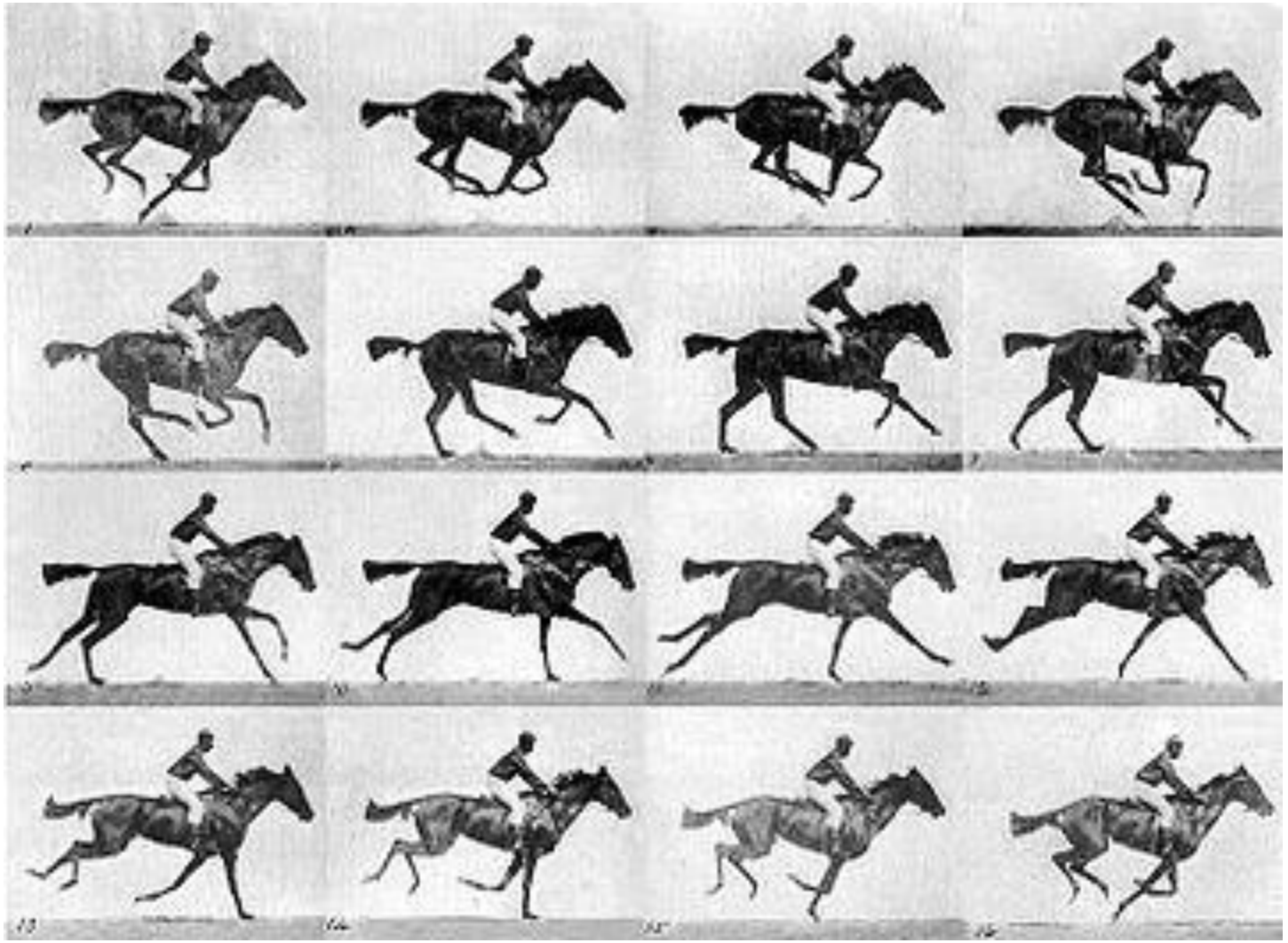
- The `loadImage()` command returns a PImage object (stored in `img` above)

- The PImage object also permits individual pixel color data to be modified.
  - like the sketch window

Animating with Images

- Animations can be created using
  - Lists of PImage objects, and
  - Transformations

Image Sequence



http://commons.wikimedia.org/wiki/File:Muybridge_race_horse_animated_184px.gif

```
# sequence.py
from Processing import *

# Load all images
images = []
for i in range(1, 16):
    fileName = "images/horse" + str(i) + ".gif"
    img = loadImage( fileName )
    images.append( img )

# Open a window
window( 184, 135 )

# Draw all images to window in sequence
while not isMousePressed():
    # Repeatedly display all images in list
    for i in range(15):
        img = images[i]
        image( img, 0, 0)
        delay(60)

# Stop the program and close the window
exit()
```

```
# sequence2.py
from Processing import *

# Load all images
images = []
for i in range(1, 16):
    fileName = "images/horse" + str(i) + ".gif"
    img = loadImage( fileName )
    images.append( img )

# Open a window
window( 800, 135 )

x = 0

# Draw all images to window in sequence
while not isMousePressed():

    # Repeatedly display all images in list
    for i in range(15):
        background(230);          # Clear background
        img = images[i]           # Draw image
        image( img, x, 0)
        x = (x + 20) % width()    # Update position
        delay(60)

exit()  # Exit the window and stop the program
```

Pimage Object

Methods

width()	- Returns the width of the image
height()	- Returns the height of the image
loadPixels()	- Loads color data into buffer
updatePixels()	- Copies color data to image
setPixel(i, j, c)	- Sets pixel at i, j to color c
getPixel(i, j)	- Gets pixel color at i, j
get(i, j, w, h)	- Gets part of an image at i, j
	- of width of w and height h

PImage objects are like in-memory sketch windows

```
# warhol.py
from Processing import *

# Load image three times
warhol_r = loadImage("images/andy-warhol2.jpg")
warhol_g = loadImage("images/andy-warhol2.jpg")
warhol_b = loadImage("images/andy-warhol2.jpg")

# Load all pixels to allow color manipulation
warhol_r.loadPixels()
warhol_g.loadPixels()
warhol_b.loadPixels()

# Create window three times image width
w = warhol_r.width()
h = warhol_r.height()
window(3*w, h)

…
```

```
…

# Load pixels in image to allow pixel manipulation
for i in range(w):
    for j in range(h):
        # Use only red color values
        c = warhol_r.getPixel(i, j)
        c = color( red(c), 0, 0 )
        warhol_r.setPixel(i, j, c)

        # Use only green color values
        c = warhol_g.getPixel(i, j)
        c = color( 0, green(c), 0 )
        warhol_g.setPixel(i, j, c)

        # Use only blue color values
        c = warhol_b.getPixel(i, j)
        c = color( 0, 0, blue(c) )
        warhol_b.setPixel(i, j, c)

…
```

…

```
# Update pixels in images
warhol_r.updatePixels()
warhol_g.updatePixels()
warhol_b.updatePixels()

# Draw modified images
image(warhol_r, 0, 0)
image(warhol_g, w, 0)
image(warhol_b, 2*w, 0)
```


Calico

```
# pointlillism.py
from Processing import *

# Load picture, create a window of same size,
# and draw the picture
img = loadImage("images/bmc3.jpg")
window( img.width(), img.height() )
image(img, 0, 0)

# Load pixels so all colors are available to read
loadPixels()

# Sample 20000 colors from random locations
# and paint a small circle of same color at same location
noStroke()

for i in xrange(20000):
    x = random(width())
    y = random(height())
    c = getPixel(x, y)      # Get a color at a random location
    fill(c)
    ellipse(x, y, 7, 7)     # Draw ellipse of same color
```


Calico
Calico

color(…)	- Creates a color
red(…)	- Get the red component
green(…)	- Get the green component
blue(…)	- Get the blue component
alpha(…)	- Get the alpha component
saturation(…)	- Get the saturation
hue(…)	- Get the hue
brightness(…)	- Get the brightness
immediateMode(…)	- Specify when to draw sketch
redraw()	- Redraw sketch on demand
delay(…)	- Delay program execution
PImage object	- Object encapsulating an image

loadImage(path)	- Load an PImage from a file
image(img, x, y)	- Draws a PImage on a sketch
width()	- Returns the width of the image
height()	- Returns the height of the image
loadPixels()	- Loads color data into buffer
updatePixels()	- Copies color data to image
setPixel(i, j, c)	- Sets pixel at i, j to color c
getPixel(i, j)	- Gets pixel color at i, j
get(i, j, w, h)	- Gets part of an image

All apply to both the sketch window and a PImage. (I wonder why?)