Getting Started in Processing

CIS 110

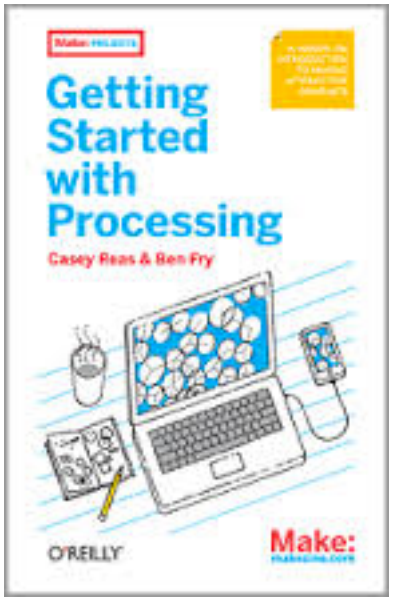
Penn Engineering

Processing

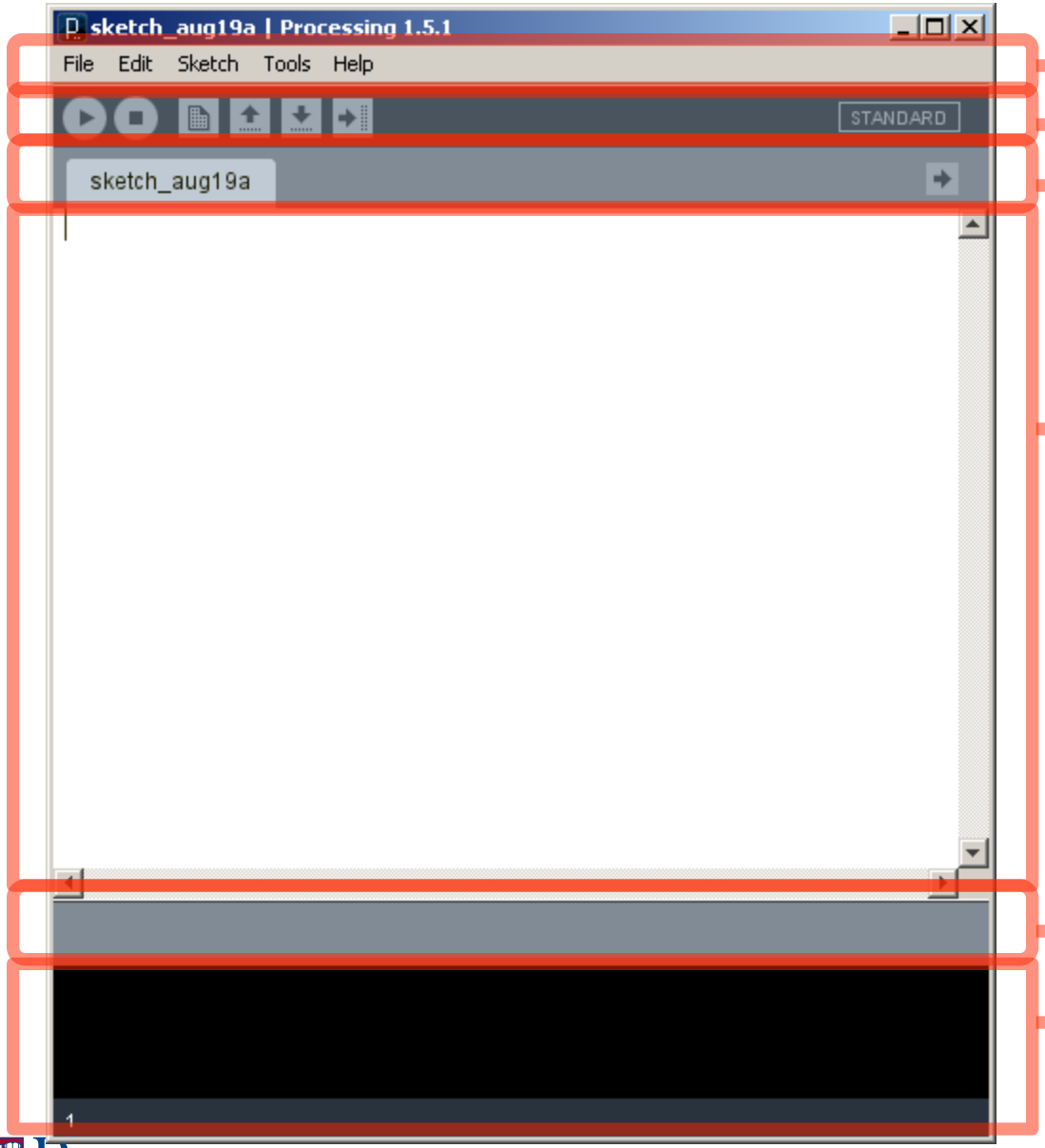
We are starting with **Processing**

- Already installed in the Moore labs
- Also available for your own computer @ www.processing.org
- Processing ≈ Java

Processing provides a gentle introduction to programming in Java through Computational Art



Penn Engineering

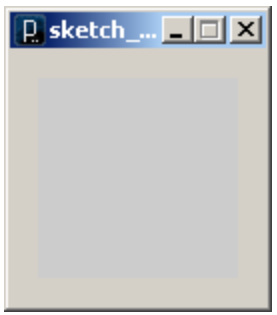


Menu bar

Tool bar

Tab strip

Text editor



Display Window

Message area

Console

Penn Engineering

Computational Art

Examples

Penn Engineering

Protobytes by Ira Greenberg

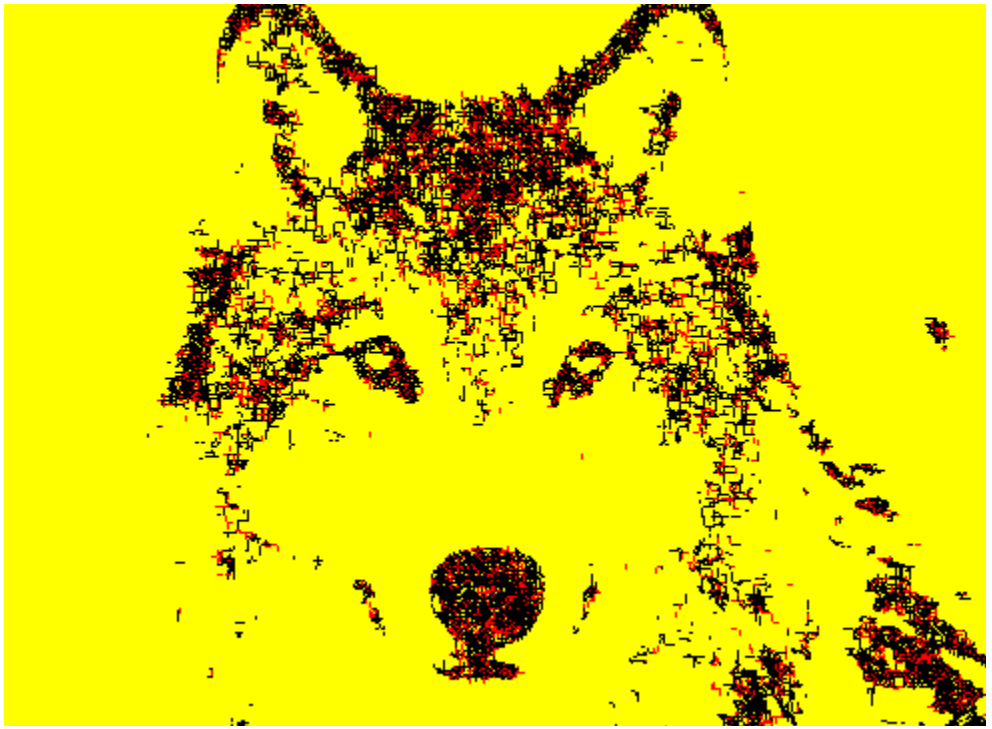
Penn Engineering

Shepard Fairey

Penn Engineering

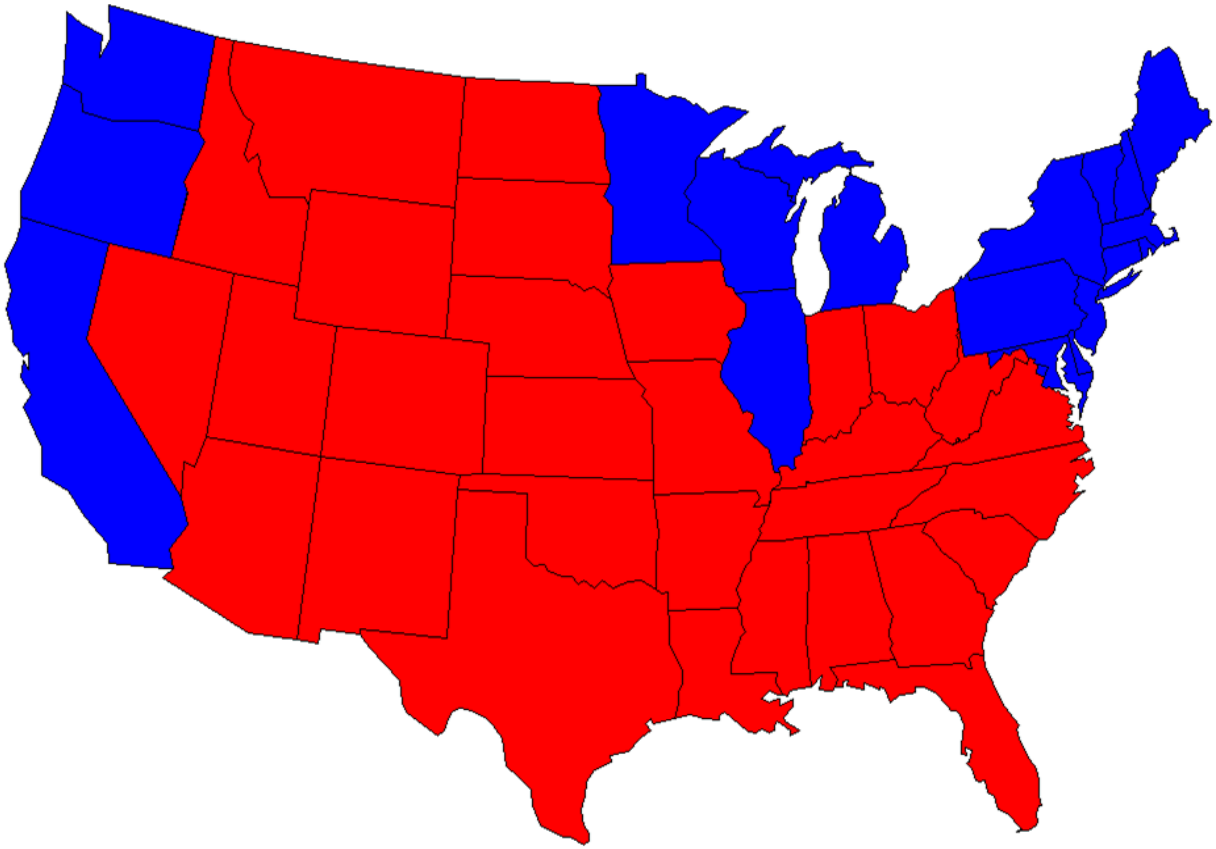

Baikal-1

Abstract Art



Penn Engineering

Red & Blue States



Penn Engineering

Summertime

Summertime,
And the livin' is easy
Fish are jumpin'
And the cotton is high

Your daddy's rich
And your mamma's good lookin'
So hush little baby
Don't you cry

One of these mornings
You're going to rise up singing
Then you'll spread your wings
And you'll take to the sky

But till that morning
There's a'nothing can harm you
With daddy and mamma standing by

Summertime,
And the livin' is easy
Fish are jumpin'
And the cotton is high

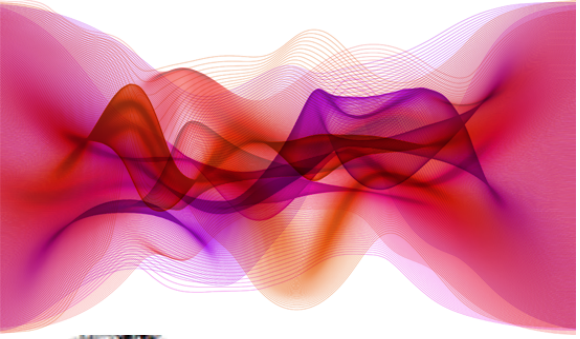
Your daddy's rich
And your mamma's good lookin'
So hush little baby
Don't you cry

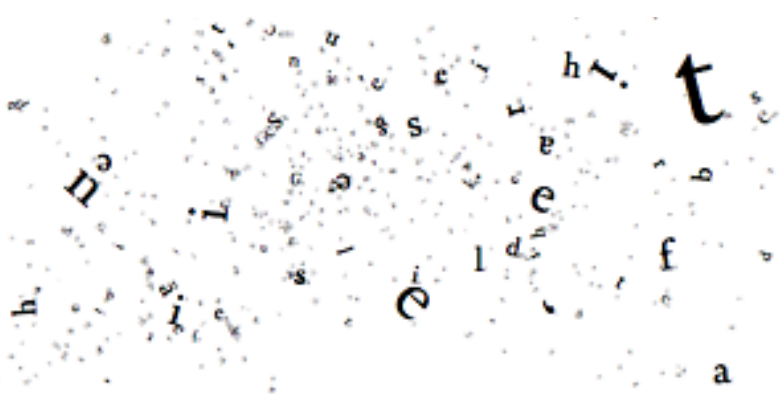
Lyrics by George Gershwin

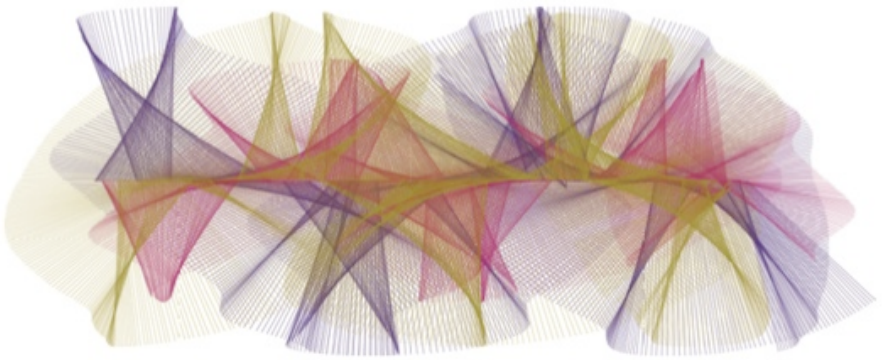
Word Cloud

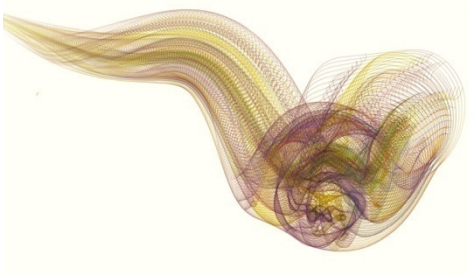
Created using: wordle.net

Penn Engineering

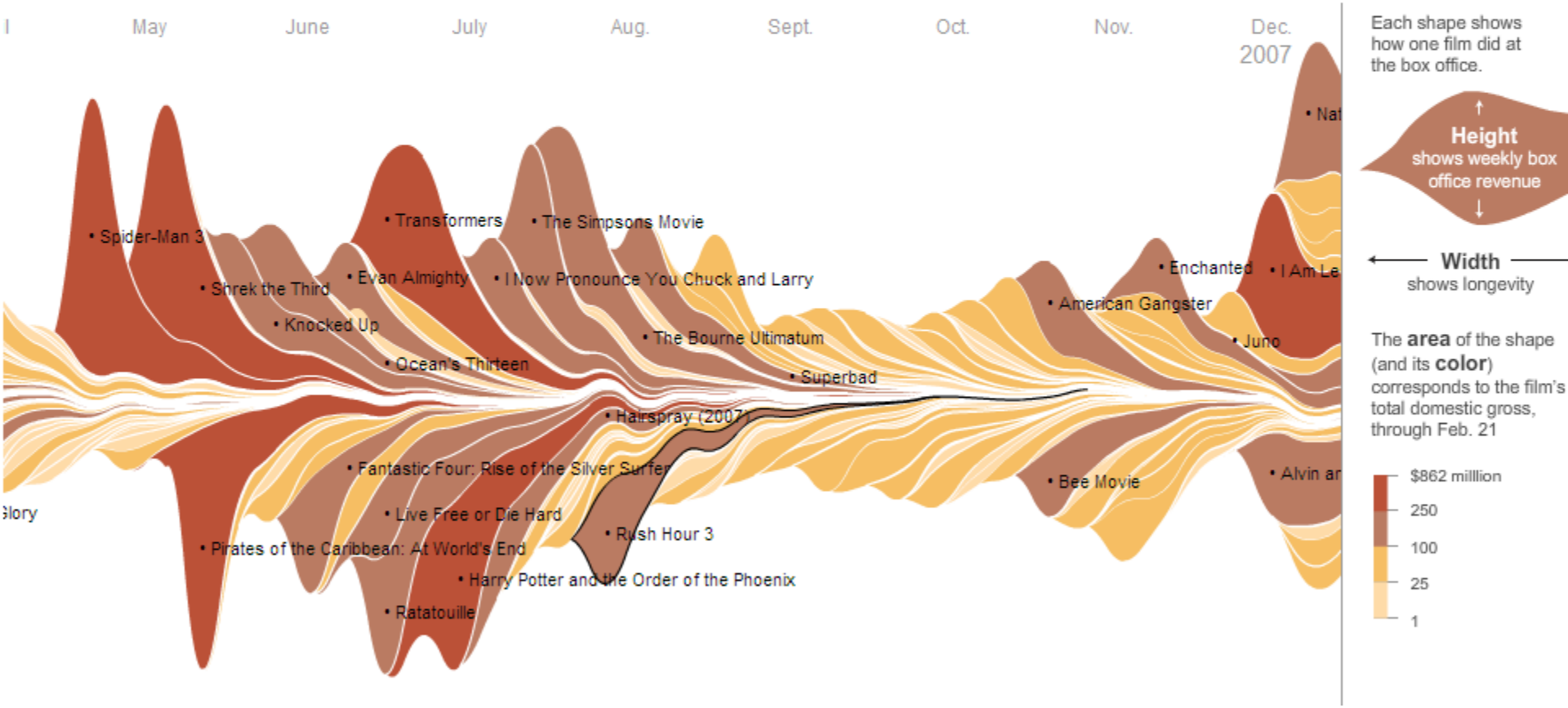








Box Office Earnings



From: The Ebb and Flow of Movies: Box Office Receipts 1986 — 2008
nytimes.com
February 23, 2008

Penn Engineering

Drawing in Processing

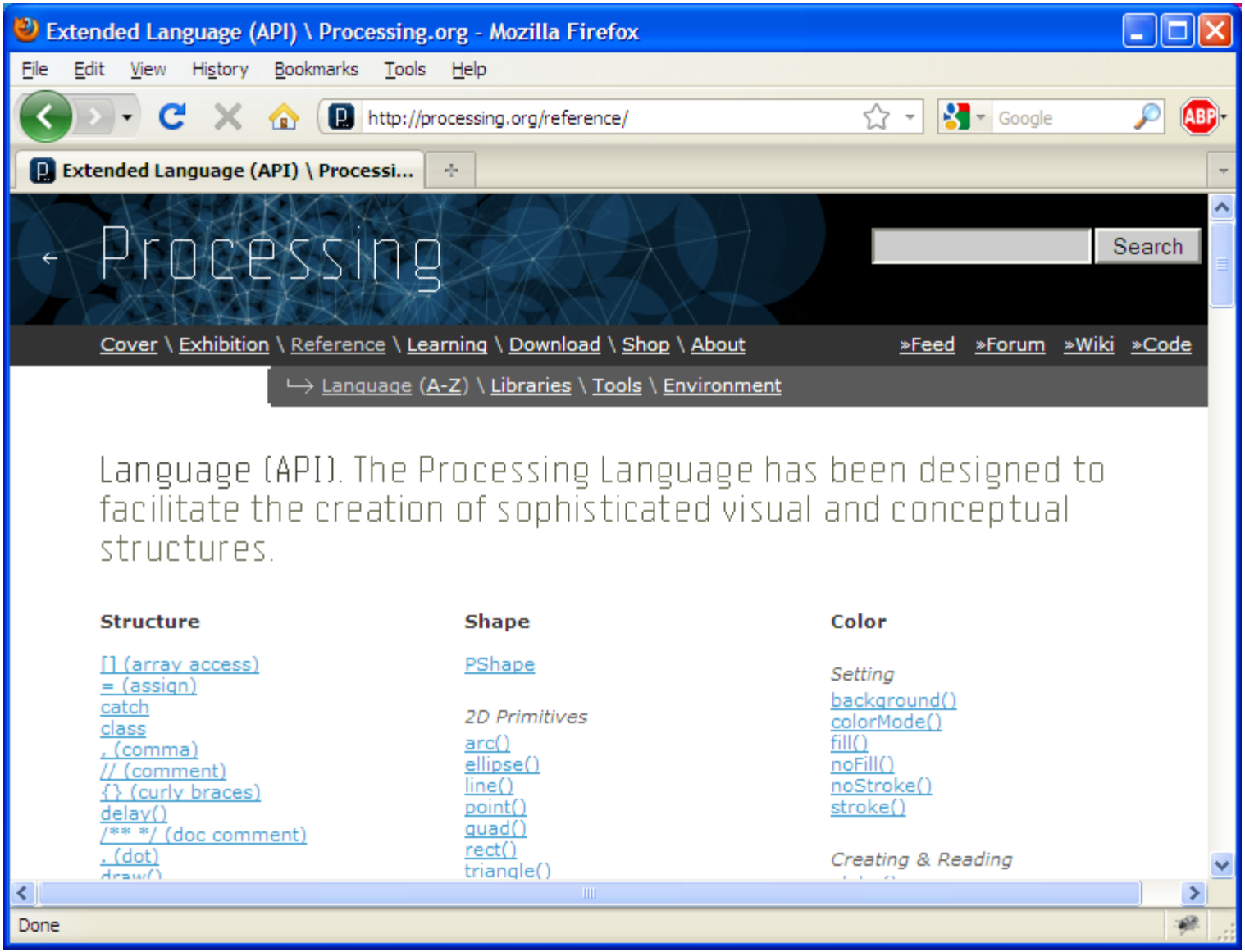
CIS 110

Penn Engineering

Primitive 2D Shapes

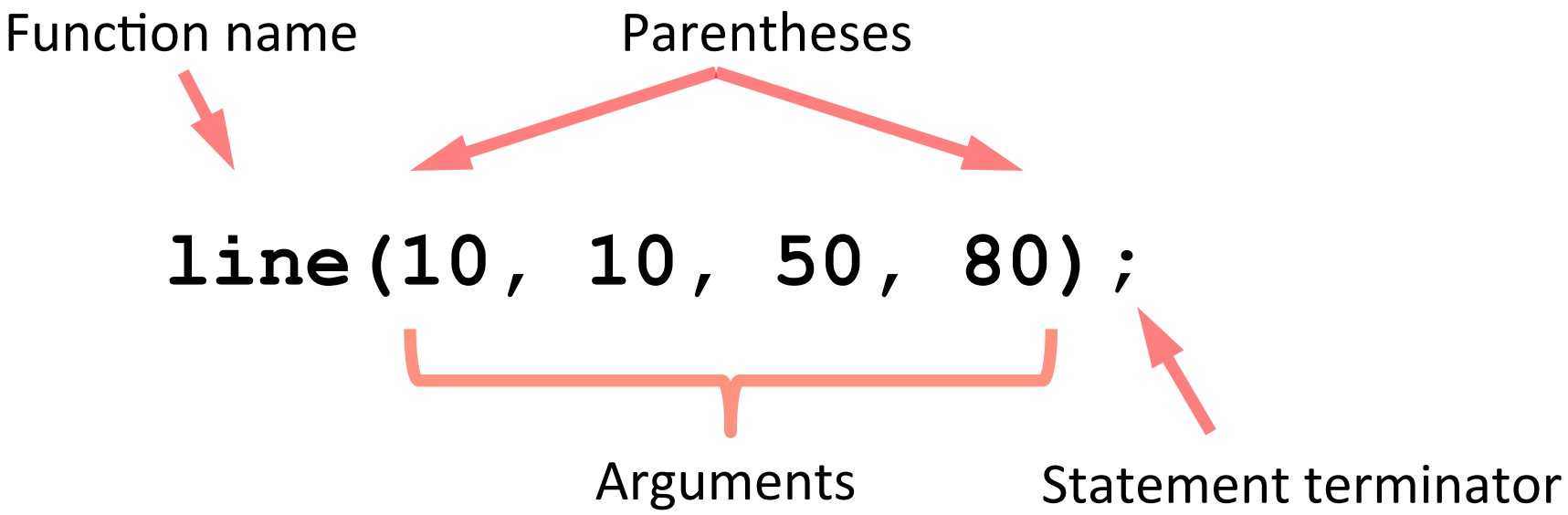
- point
- line
- triangle
- rect (rectangle)
- quad (quadrilateral, four-sided polygon)
- ellipse
- arc (section of an ellipse)
- curve (Catmull-Rom spline)
- bezier (Bezier curve)

Penn Engineering

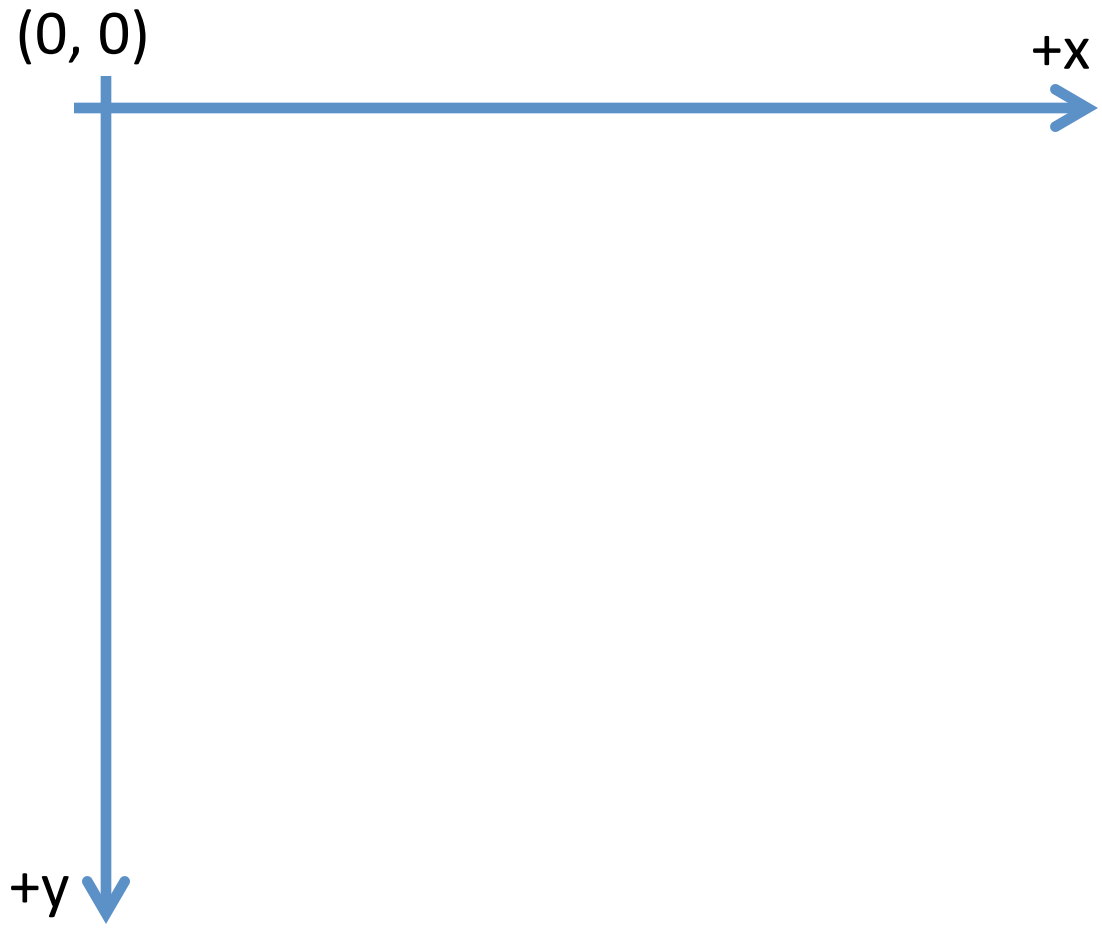


http://processing.org/reference/

Anatomy of a Function Call

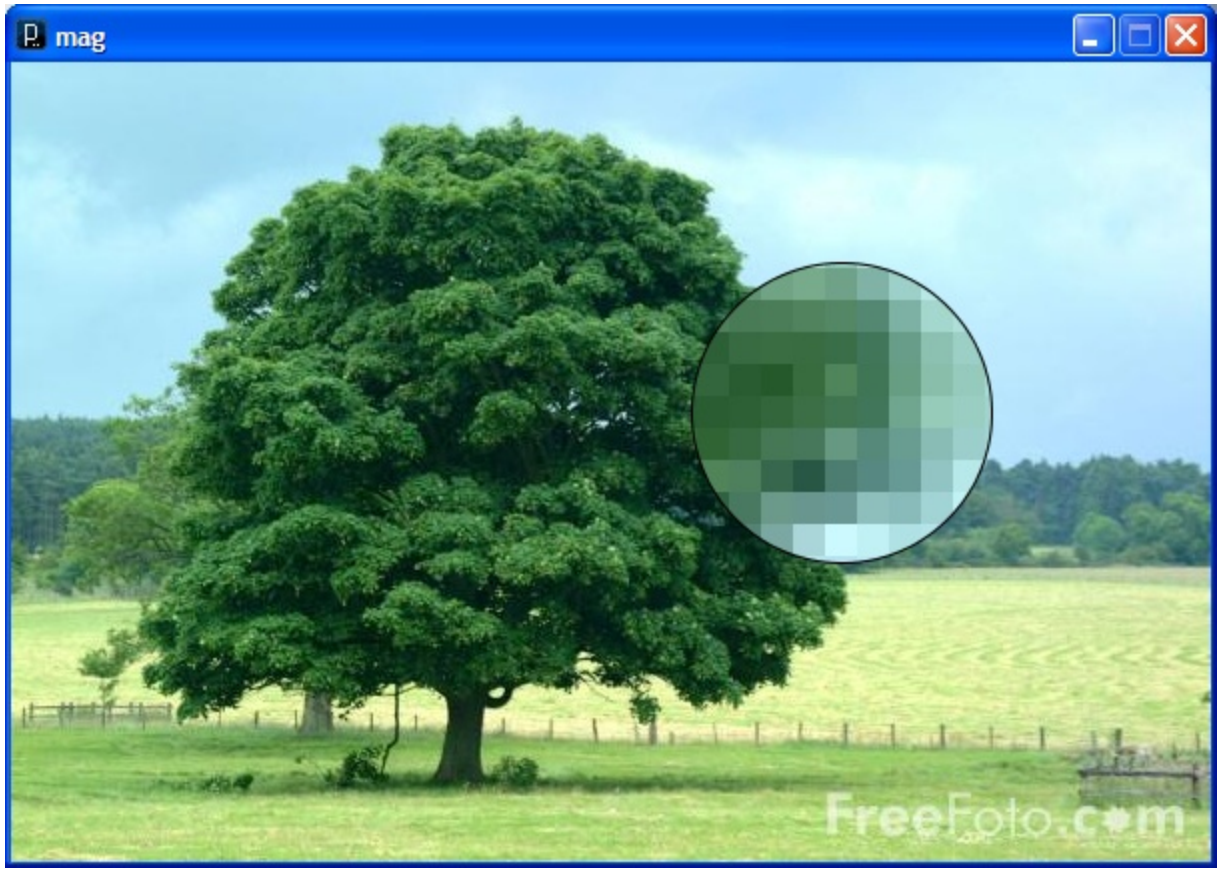


Coordinate System



Penn Engineering

Pixels



Processing Canvas

`size(`*`width, height`***`);`**

Set the size of the canvas.

`background(`*`[0..255]`***`);`**

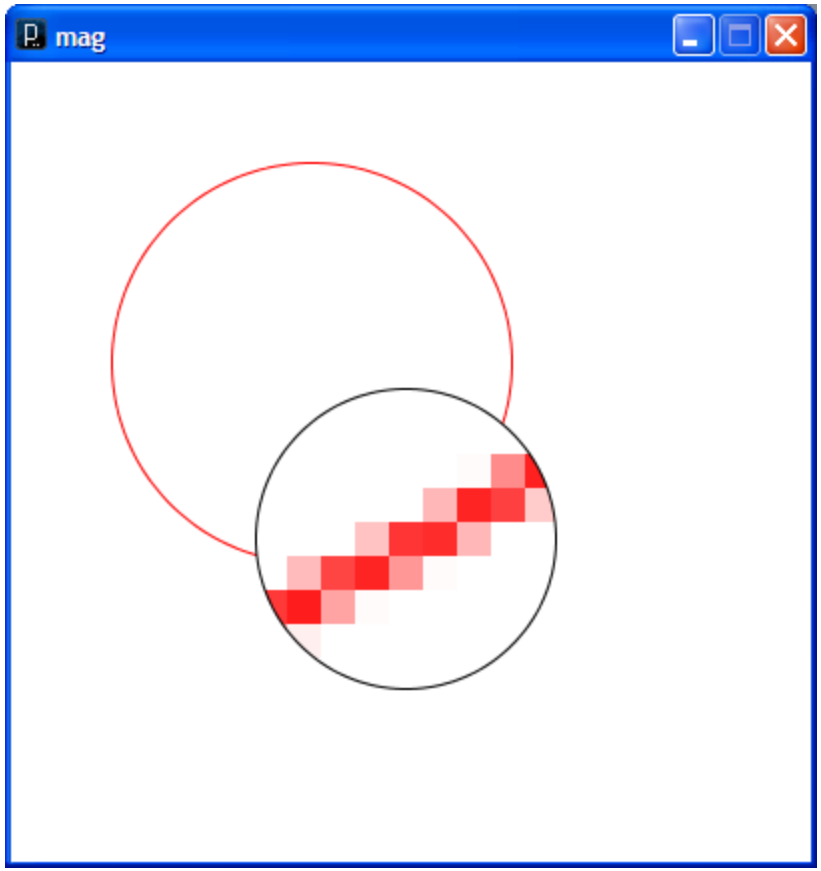
Set the background grayscale color.

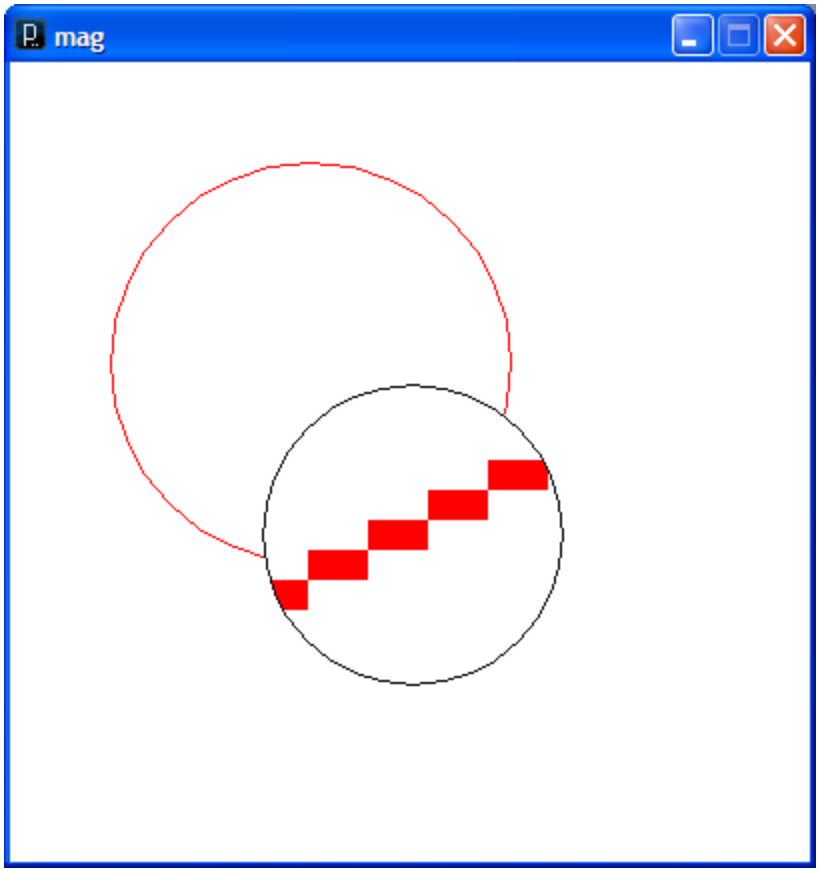
Drawing Primitives

```
point(x, y);

line(x1, y1, x2, y2);

triangle(x1, y1, x2, y2, x3, y3);

quad(x1, y1, x2, y2, x3, y3, x4, y4);

rect(x, y, width, height);

ellipse(x, y, width, height);
```

Penn Engineering

smooth() vs. noSmooth()



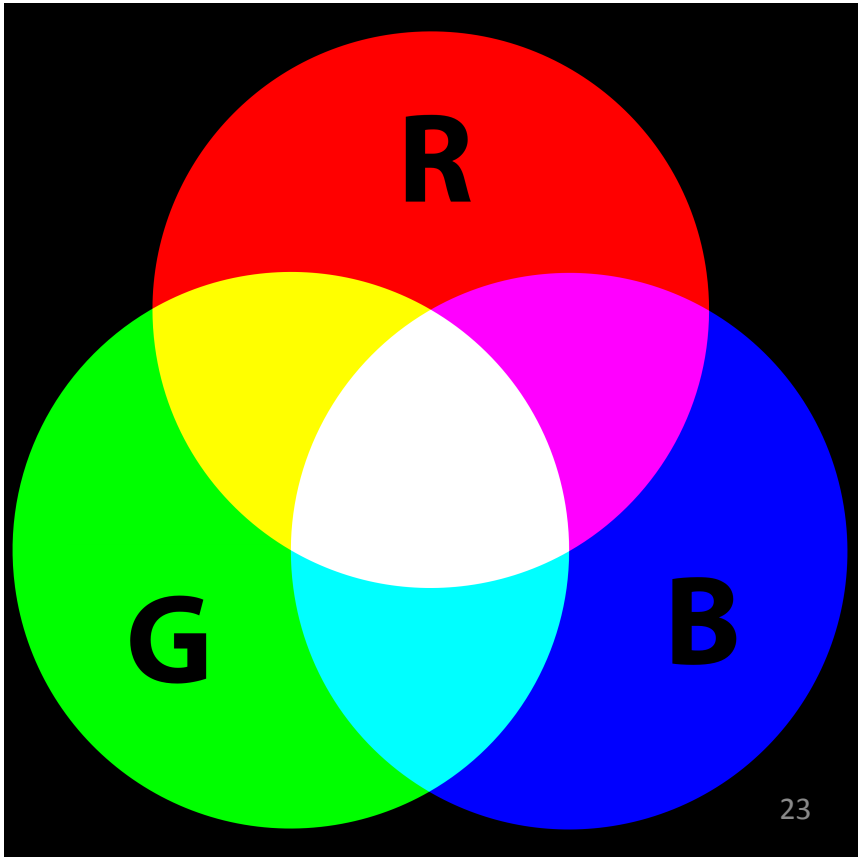


Penn Engineering

Colors

Composed of four elements:

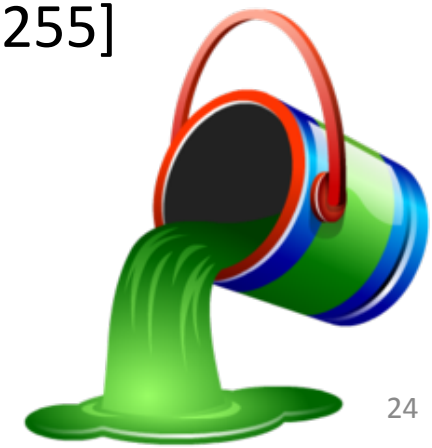
1. Red
2. Green
3. Blue
4. Alpha (Opacity)



Background

```
background(gray);
background(gray, alpha);
background(red, green, blue);
background(red, green, blue, alpha);
```

Each color component takes on a value [0 ... 255]

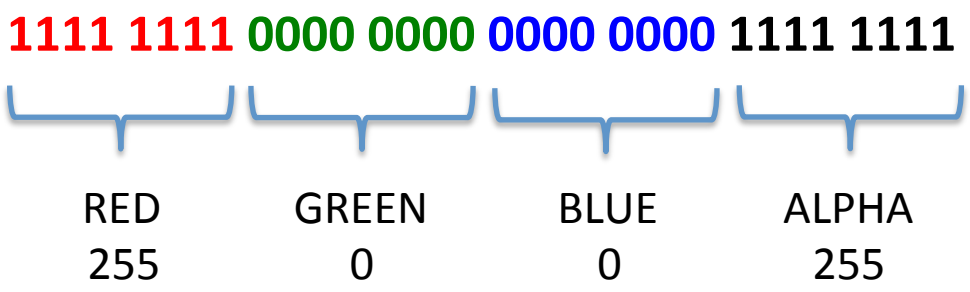


Why 0 … 255?

Why 0 ... 255?

Decimal	Binary
0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
5	00000101
6	00000110
7	00000111
8	00001000
9	00001001
10	00001010
11	00001011
12	00001100
13	00001101
14	00001110
15	00001111
16	00010000
17	00010001
18	00010010
. . .	. . .
255	11111111

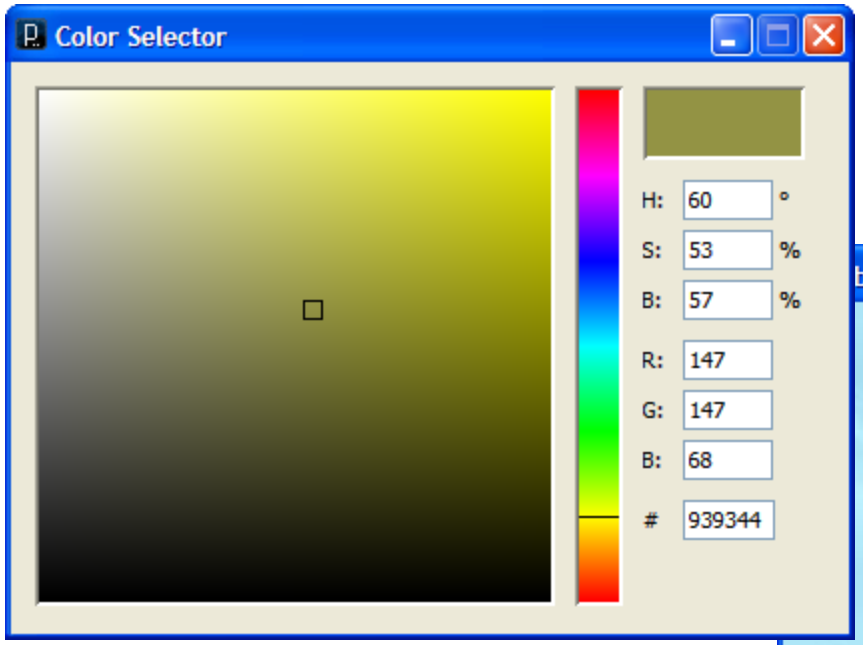
Each color is represented by 32 bits:

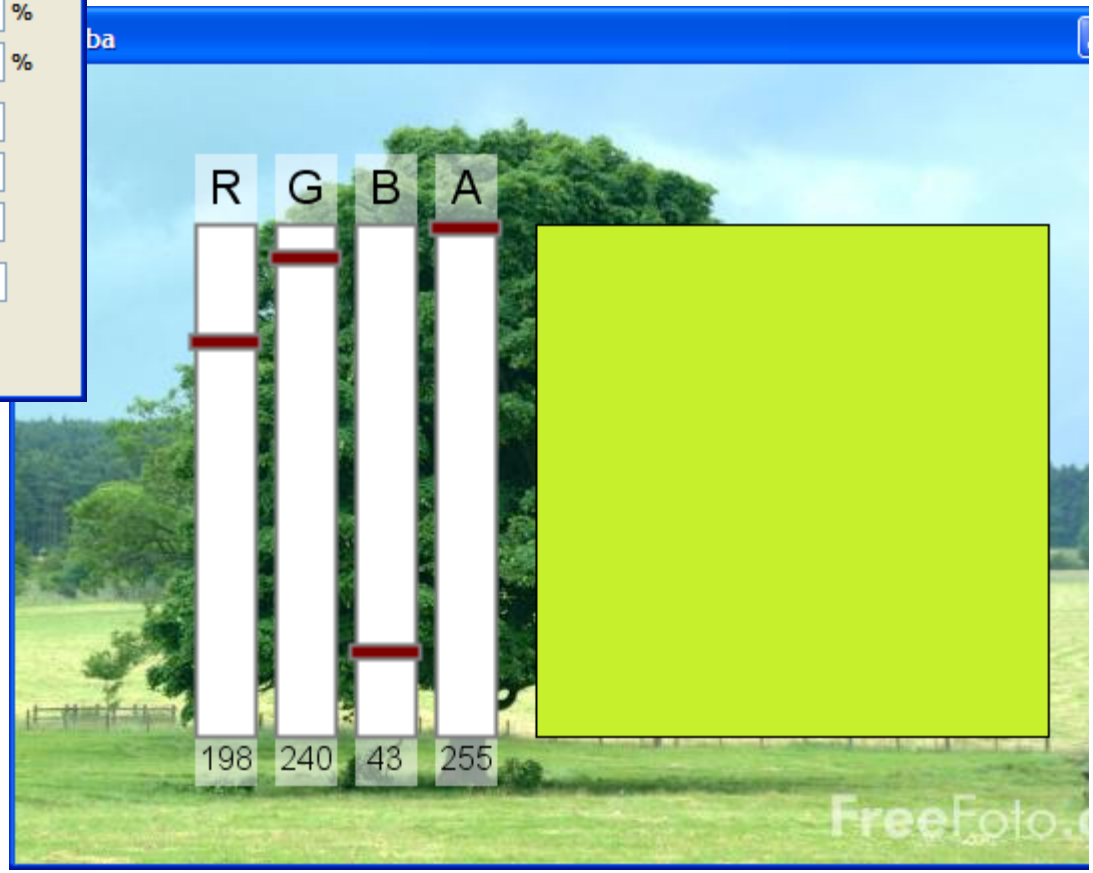


Notice there are 8 bits per color component.

The maximum value (all 1's) that can be represented in 8 bits is 255 in decimal.

Therefore, the range for each color component is 0 (off) ... 255 (full).

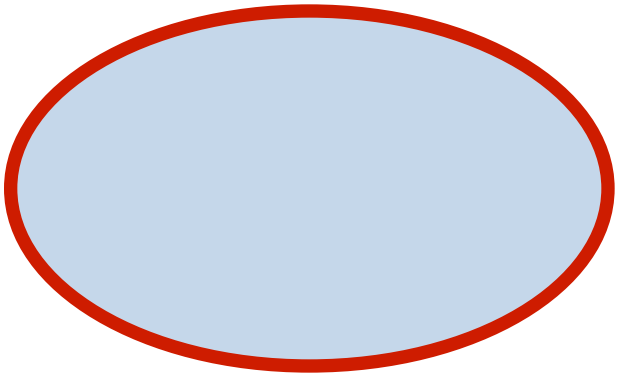




Penn Engineering

Shape Formatting

1. Fill color
2. Line thickness
3. Line color



These are properties of your paintbrush, not of the object you are painting.

Fill Color

```
fill(gray);
fill(gray, alpha);
fill(red, green, blue);
fill(red, green, blue, alpha);

noFill();
```

Penn Engineering

Stroke (Line) Color

```
stroke(gray);
stroke(gray, alpha);
stroke(red, green, blue);
stroke(red, green, blue, alpha);

noStroke();
```

Penn Engineering

strokeCap()

```
smooth();
strokeWeight(12.0);
strokeCap(ROUND);
line(20, 30, 80, 30);
strokeCap(SQUARE);
line(20, 50, 80, 50);
strokeCap(PROJECT);
line(20, 70, 80, 70);
```

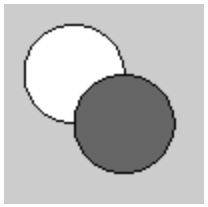
strokeWeight()

```
smooth();
strokeWeight(1);   // Default
line(20, 20, 80, 20);
strokeWeight(4);   // Thicker
line(20, 40, 80, 40);
strokeWeight(10);  // Beastly
line(20, 70, 80, 70);
```

Penn Engineering

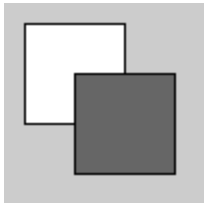
http://processing.org/reference/strokeCap_.html
http://processing.org/reference/strokeWeight_.html

ellipseMode



```
ellipseMode(CENTER);
ellipse(35, 35, 50, 50);
ellipseMode(CORNER);
fill(102);
ellipse(35, 35, 50, 50);
```

rectMode



```
rectMode(CENTER);
rect(35, 35, 50, 50);
rectMode(CORNER);
fill(102);
rect(35, 35, 50, 50);
```

Penn Engineering

http://processing.org/reference/ellipseMode_.html
http://processing.org/reference/rectMode_.html

`random(high);`

`random(low, high);`

Generate a random number in the range *low* (or 0) to *high*

`mouseX`

`mouseY`

Built-in predefined variables that hold the current mouse X and Y locations

`print("something");`

`println("something");`

Print "something" to the console

```
void setup()
{
  // Called once when program starts
}

void draw()
{
  /* Called repeatedly
     while program runs */
}
```

randomEllipse

```
void setup()
{
  size(300, 300);
  smooth();
}

void draw()
{
  fill(random(255), random(255), random(255));
  ellipse(mouseX, mouseY, 30, 30);
}
```

Controlling draw()

`frameRate(`*`fps`*`);`

Sets number of frames displayed per second.

i.e. the number of times draw() is called per second. Default = 60.

`noLoop();`

Stops continuously calling draw().

`loop();`

Resumes calling draw().

```
void mousePressed() {
   // Called when the mouse is pressed
}

void mouseReleased() {
   // Called when the mouse is released
}

void mouseClicked() {
   // Called when the mouse is pressed and released
   // at the same mouse position
}

void mouseMoved() {
   // Called while the mouse is being moved
   // with the mouse button released
}

void mouseDragged() {
   // Called while the mouse is being moved
   // with the mouse button pressed
}
```

```
void keyPressed() {
   // Called each time a key is pressed
}

void keyReleased() {
   // Called each time a key is released
}

void keyTyped() {
   // Called when a key is pressed
   // Called repeatedly if the key is held down
}
```

More Graphics

arc(…)

curve (…)

bézier(…)

shape(…)

Arcs

```
arc(x, y, width, height, start, stop);
```

An arc is a section of an ellipse

`x, y, width, height`

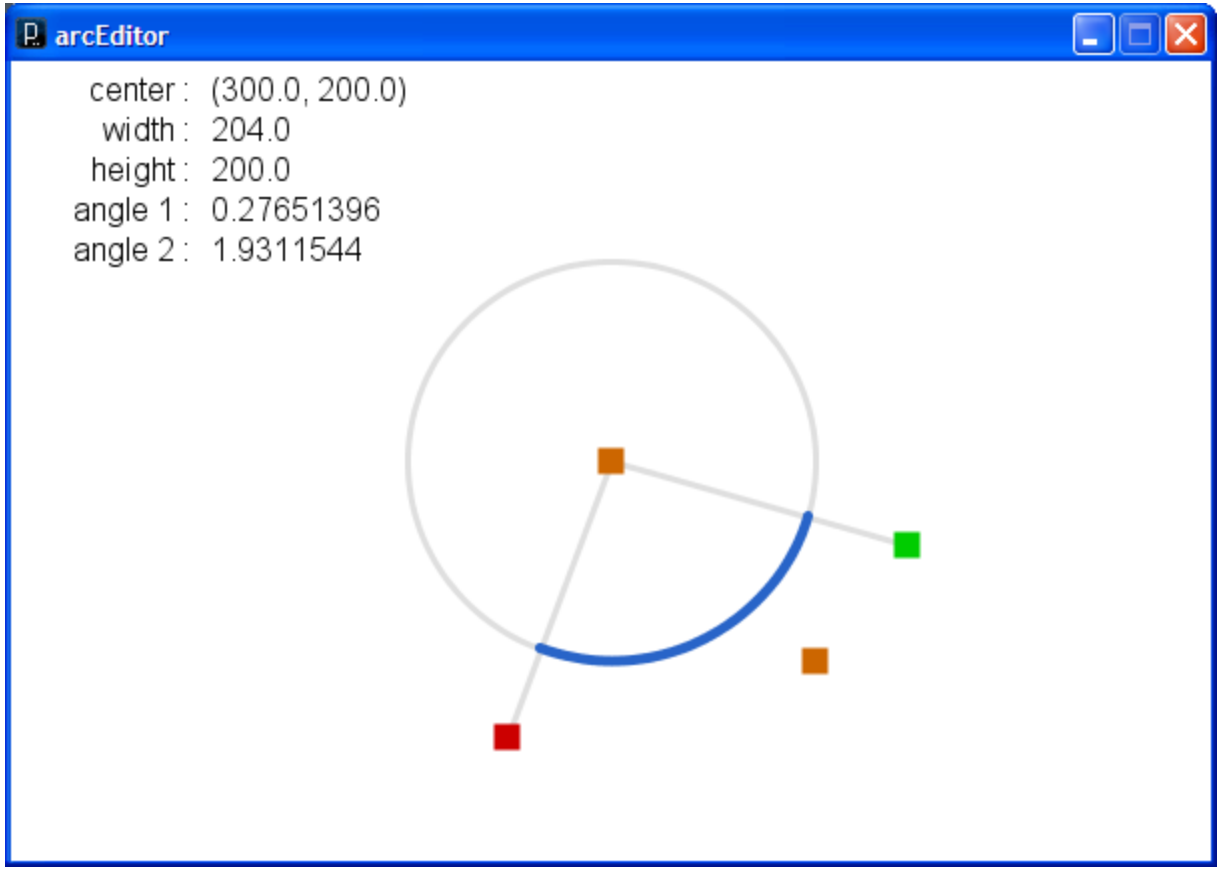
location and size of the ellipse

`start, stop`

arc bounding angles (in radians)

Arcs

```
arc(x, y, width, height, start, stop);
```



Spline Curves

```
curve(x1, y1, x2, y2, x3, y3, x4, y4);
```

Spline: A smooth line drawn through a series of points

A curve is a Catmull-Rom (cubic Hermite) spline defined by four points

`x2, y2` *and* `x3, y3`

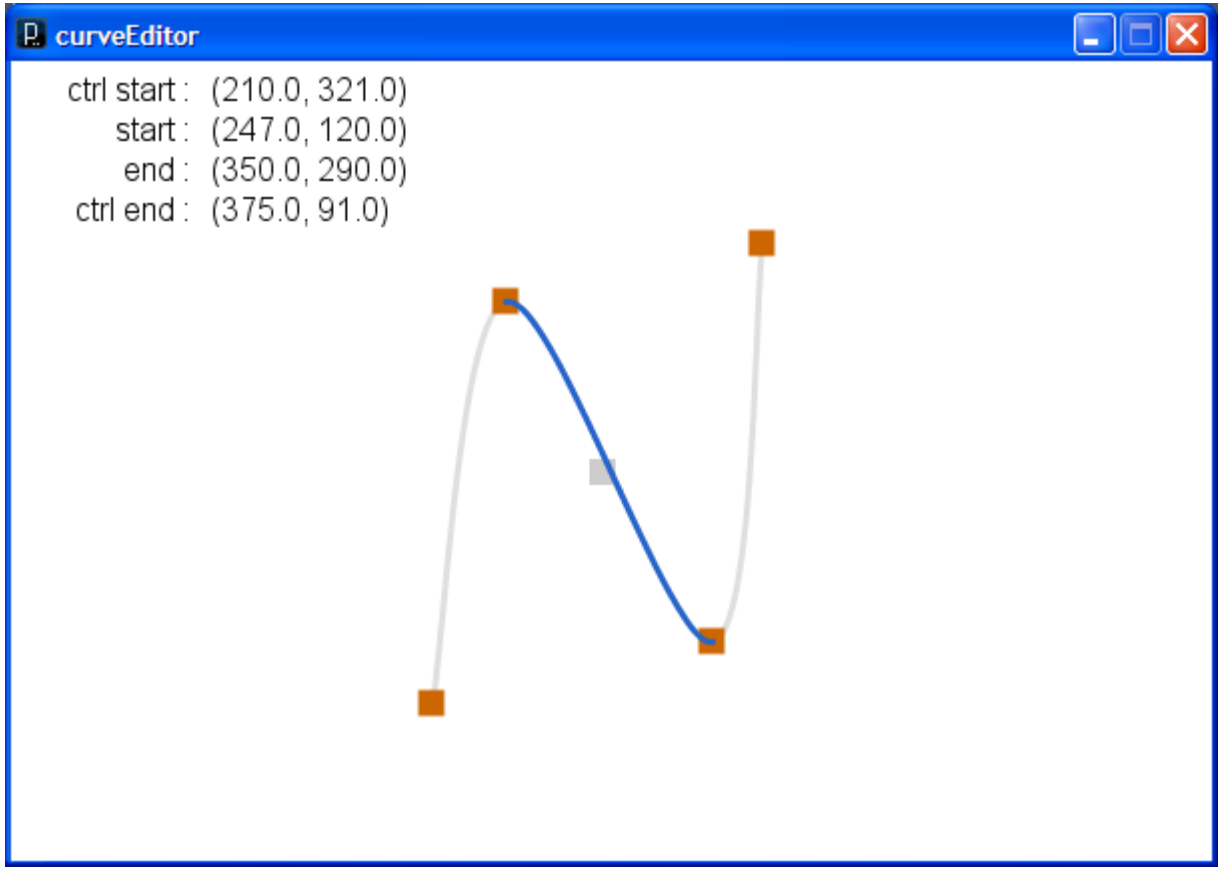
beginning/end points of visual part of curve

`x1, y1` *and* `x4, y4`

control points that define curve curvature

Spline Curves

```
curve(x1, y1, x2, y2, x3, y3, x4, y4);
```



Penn Engineering

Bézier Curves

```
bezier(x1, y1, cx1, cy1, cx2, cy2, x2, y2);
```

A smooth curve defined by two anchor points and two control points

`x2, y2` *and* `x2, y2`

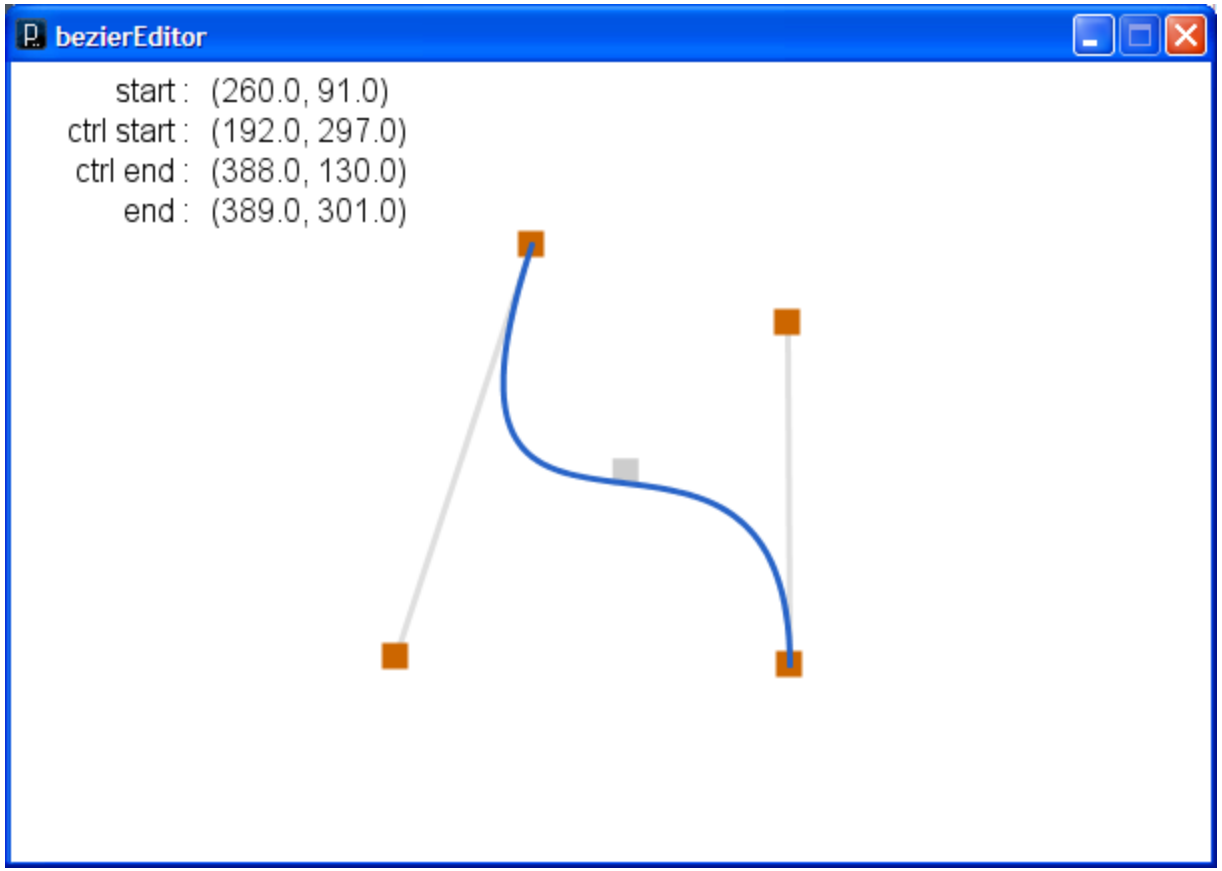
anchor points of bézier curve

`cx1, cy1` *and* `cx2, cy2`

control points that define curvature

Bézier Curves

```
bezier(x1, y1, cx1, cy1, cx2, cy2, x2, y2);
```



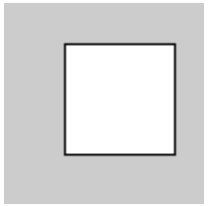
Penn Engineering

Custom Shapes

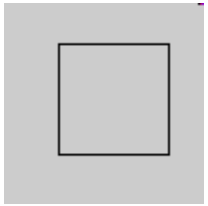
- Composed of a series of vertexes (points)
- Vertexes may or may not be connected with lines
- Lines may join at vertexes in a variety of manners
- Lines may be straight, curves, or bézier splines
- Shape may be closed or open

Custom Shapes

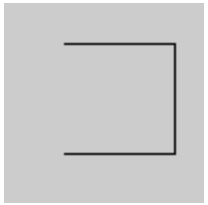
```
beginShape([option]);

vertex(x, y);

curveVertex(x, y);

bezierVertex(cx1, cy1, cx2, cy2, x, y);

endShape([CLOSE]);
```



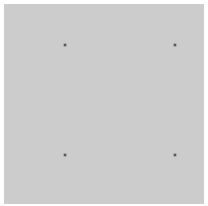
```
beginShape();
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape(CLOSE);
```



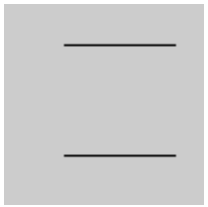
```
noFill();
beginShape();
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape(CLOSE);
```



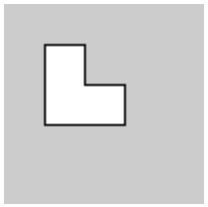
```
noFill();
beginShape();
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape();
```



```
beginShape(POINTS);
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape();
```

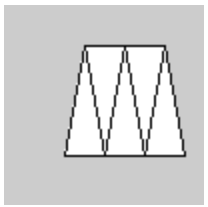


```
beginShape(LINES);
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape();
```

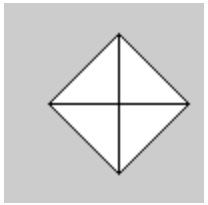


```
beginShape();
vertex(20, 20);
vertex(40, 20);
vertex(40, 40);
vertex(60, 40);
vertex(60, 60);
vertex(20, 60);
endShape(CLOSE);
```

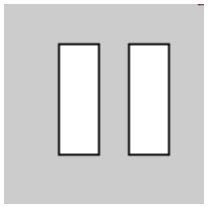
```
beginShape(TRIANGLES);
vertex(30, 75);
vertex(40, 20);
vertex(50, 75);
vertex(60, 20);
vertex(70, 75);
vertex(80, 20);
endShape();
```



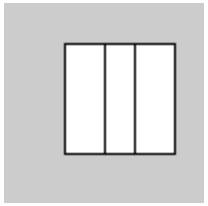
```
beginShape(TRIANGLE_STRIP);
vertex(30, 75);
vertex(40, 20);
vertex(50, 75);
vertex(60, 20);
vertex(70, 75);
vertex(80, 20);
vertex(90, 75);
endShape();
```



```
beginShape(TRIANGLE_FAN);
vertex(57.5, 50);
vertex(57.5, 15);
vertex(92, 50);
vertex(57.5, 85);
vertex(22, 50);
vertex(57.5, 15);
endShape();
```



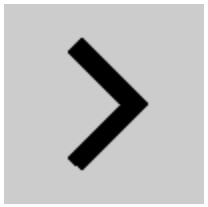
```
beginShape(QUADS);
vertex(30, 20);
vertex(30, 75);
vertex(50, 75);
vertex(50, 20);
vertex(65, 20);
vertex(65, 75);
vertex(85, 75);
vertex(85, 20);
endShape();
```



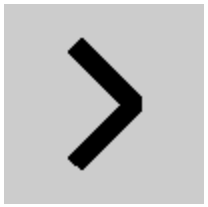
```
beginShape(QUAD_STRIP);
vertex(30, 20);
vertex(30, 75);
vertex(50, 20);
vertex(50, 75);
vertex(65, 20);
vertex(65, 75);
vertex(85, 20);
vertex(85, 75);
endShape();
```

Penn Engineering

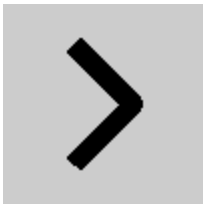
strokeJoin()



```
noFill();
smooth();
strokeWeight(10.0);
strokeJoin(MITER);
beginShape();
vertex(35, 20);
vertex(65, 50);
vertex(35, 80);
endShape();
```



```
noFill();
smooth();
strokeWeight(10.0);
strokeJoin(BEVEL);
beginShape();
vertex(35, 20);
vertex(65, 50);
vertex(35, 80);
endShape();
```



```
noFill();
smooth();
strokeWeight(10.0);
strokeJoin(ROUND);
beginShape();
vertex(35, 20);
vertex(65, 50);
vertex(35, 80);
endShape();
```

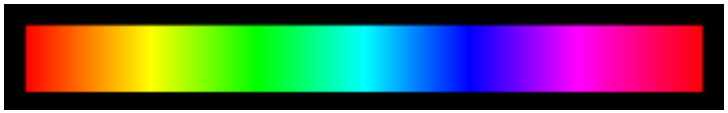
More Color

`colorMode(`*`RGB or HSB`*`);`

RGB: (red, green, blue)

HSB:

- hue
  - “pure color”
- saturation
  - “intensity”
- brightness
  - “lightness”



Images

`loadImage(`*`filename`*`);`

- Loads an image from a file in the *data* folder in sketch folder.
- Must be assigned to a variable of type PImage.

`image(`*`img, X, Y, [X2, Y2]`*`);`

- Draws the image *img* on the canvas at X, Y
- Optionally fits image into box X,Y and X2,Y2

`imageMode(CORNER);`

- X2 and Y2 define width and height.

`imageMode(CORNERS);`

- X2 and Y2 define opposite corner.

Image Example

Mario
- Mario.pde
-  data
  - MarioJumping.jpg

```
PImage img;

void setup() {
  size(500, 400);
  img = loadImage("MarioJumping.jpg");
}

void draw() {
  background(255);
  image(img, width/4, height/4);
}
```

Example Sketches...

- LadyBug1
- Monster1
- Ndebele
- Penguin1
- SouthParkCharacter1
- Sushi
- Mario
- GiorgioMorandi

text()

- Strings can be drawn on a sketch using text()
- Can set text position, font, size, alignment, …

```
// Set attributes
textSize( sizeInPixels );
textAlign( {LEFT | CENTER | RIGHT}
    [,{TOP, BOTTOM, CENTER, BASELINE}] );
fill( color );

// Render text
text( string, X, Y );
text( string, X, Y, width, height );
```

```
// text
void setup() {
  size(500, 500);
  noLoop();
}

void draw() {
  // bounding box
  stroke(0);
  fill(255);
  rect(50, 50, 400, 400);

  // text options
  fill(0);    // black text
  text("Default", 50, 50, 400, 400);
  textAlign(CENTER);
  text("CENTER", 50, 50, 400, 400);
  textAlign(RIGHT);
  text("RIGHT", 50, 50, 400, 400);
  textAlign(CENTER, CENTER);
  text("CENTER-CENTER", 50, 50, 400, 400);
  textAlign(RIGHT, BOTTOM);
  text("RIGHT-BOTTOM", 50, 50, 400, 400);
  textAlign(LEFT, BOTTOM);
  text("LEFT-BOTTOM", 50, 50, 400, 400);
}
```

Comments

- ***Line comment***
  - Begins with the symbols **//**
  - Compiler ignores remainder of the line
  - Used for the coder or for a programmer who modifies the code

```
// draw monster's head
```

- ***Block comment***
  - Begins with **/*** and ends with ***/**
  - Compiler ignores anything in between
  - Can span several lines
  - Provides documentation for the users of the program

```
/* File: Date
   Author: Joe Smith
   Date: 9/1/09
*/
```

Now go draw something!