

CIS 4930/6930-002

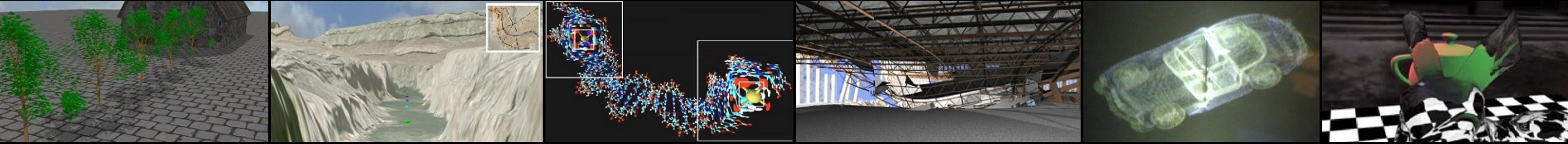
DATA VISUALIZATION



INTRODUCTION TO PROCESSING

Paul Rosen
Assistant Professor
University of South Florida

(slide acknowledgments: Hitesh Raju)



ANNOUNCEMENTS

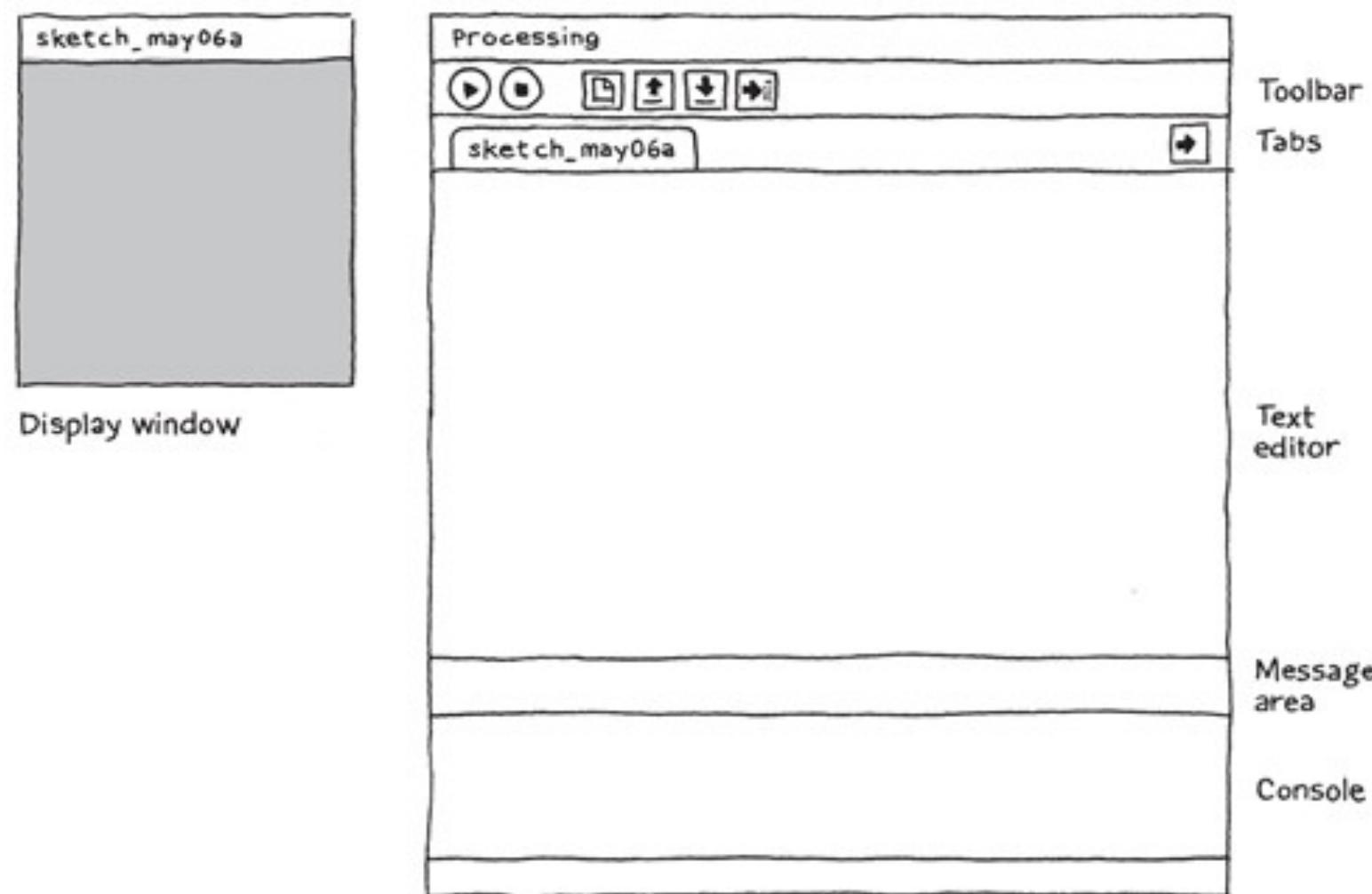
Project #1 Due Next Class (1/14)

Project #2 posted



WHAT IS IT?

Processing Development Environment (PDE)



WHAT IS IT?

Processing API

Reference. The Processing Language was designed to facilitate the creation of sophisticated visual structures.

Structure	Shape	Color
() (parentheses)	createShape()	Setting
, (comma)	loadShape()	background()
. (dot)	PShape	clear()
/* */ (multiline comment)		colorMode()
/** */ (doc comment)	2D Primitives	fill()
// (comment)	arc()	noFill()
; (semicolon)	ellipse()	noStroke()
= (assign)	line()	stroke()
[] (array access)	point()	
{ } (curly braces)	quad()	Creating & Reading
catch	rect()	alpha()
class	triangle()	blue()
draw()		brightness()
exit()	Curves	color()
extends	bezier()	green()
false	bezierDetail()	hue()
final	bezierPoint()	lerpColor()
implements	bezierTangent()	red()
import	curve()	saturation()
loop()	curveDetail()	
new	curvePoint()	Image
noLoop()	curveTangent()	createImage()
null	curveTightness()	PI mage
popStyle()		
private	3D Primitives	
public	box()	

WHAT IS IT?

open-source, online community

<https://processing.org/>

<http://forum.processing.org/>

<https://github.com/processing>



WHY NOT OTHER LANGUAGES?

difficulty to sketch with other languages

complicated setup

portability challenging

not easy to learn



WHY PROCESSING?

- +/- Java-based
 - complexity
 - + big standard library
 - + similar syntax & portability
- + lots of user-contributed libraries
 - + fast to startup

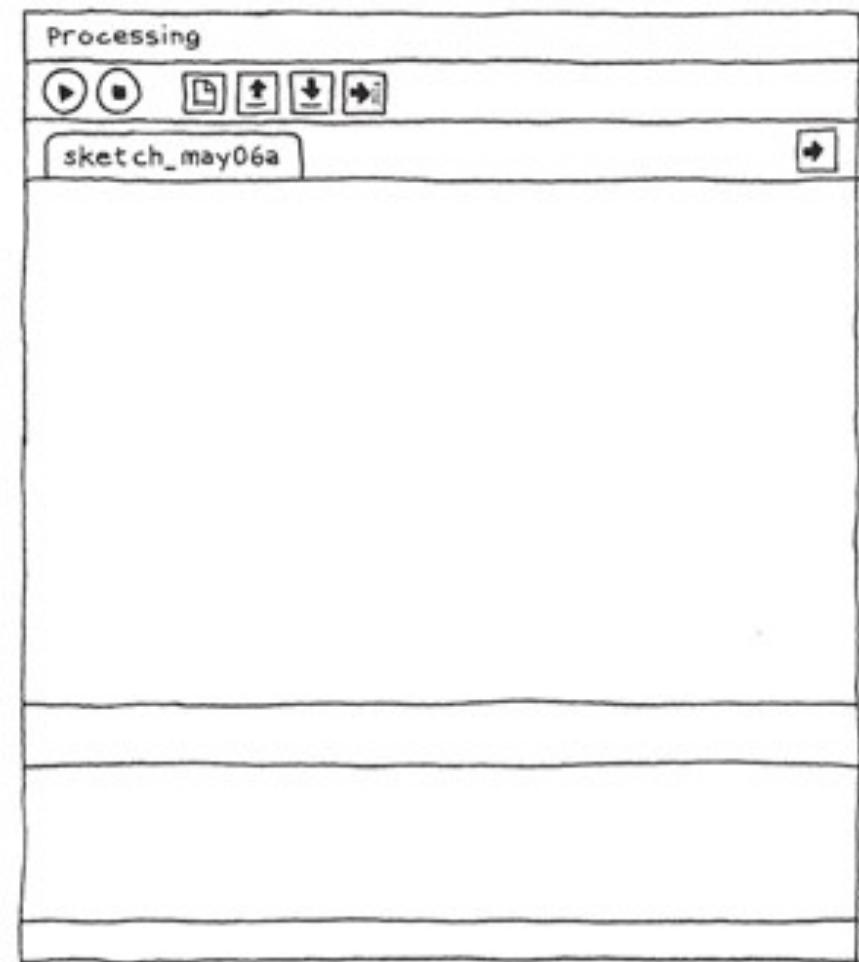


WHY PROCESSING?

program = sketch



Display window



Toolbar
Tabs

Text
editor

Message
area

Console

WHY PROCESSING?

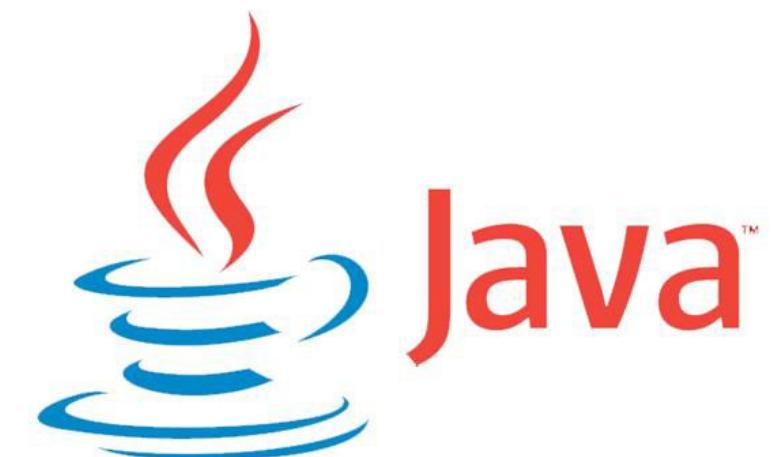
programming syntax

```
void setup() {  
    size(480, 120);  
}  
  
void draw() {  
    if (mousePressed) {  
        fill(0);  
    } else {  
        fill(255);  
    }  
    ellipse(mouseX, mouseY, 80, 80);  
}
```

WHY PROCESSING?

```
public class Hello
{
    public static void main (String args[])
    {
        System.out.println("Hello, world!");
    }
}
```

```
javac Hello.java
java Hello
```



WHY PROCESSING?

```
println("Hello, World!");
```



The screenshot shows the Processing 2.2.1 IDE interface. The title bar reads "sketch_140912a | Processing 2.2.1". The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar contains icons for play, stop, step, and run, along with a "Java" dropdown menu. The code editor window displays the line of code: `println("Hello, World!");`. Below the code editor is a preview window showing the text "Hello, World!". The bottom status bar indicates the line number "1".

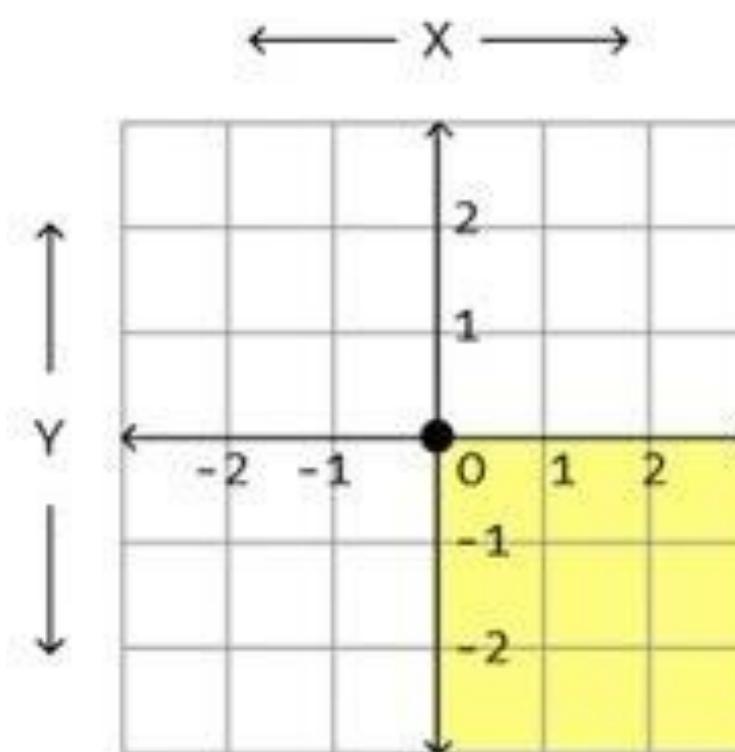


GRAPHICS

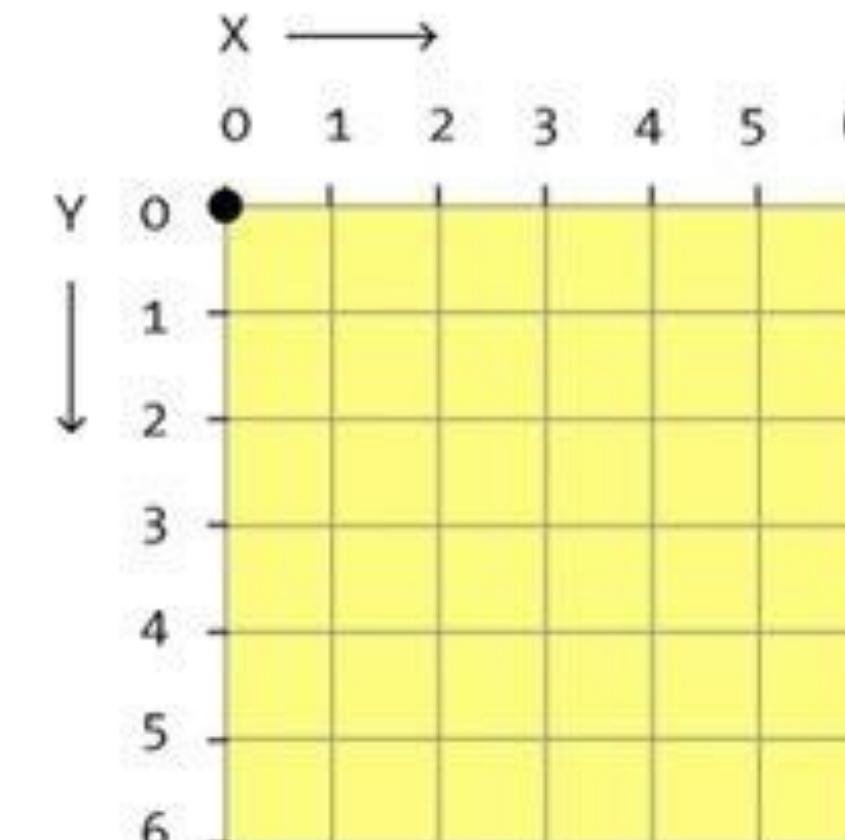


MONITORS

grid of pixels



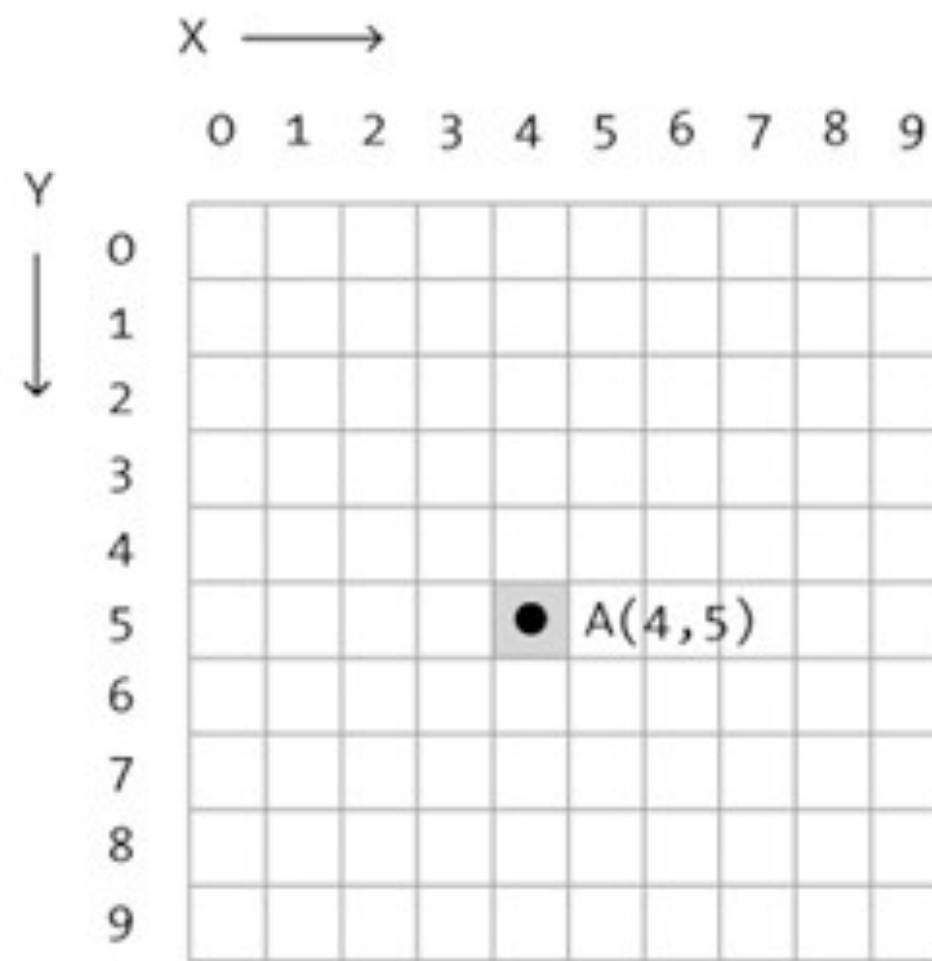
Eighth Grade



Computer

SHAPE

point (x, y);

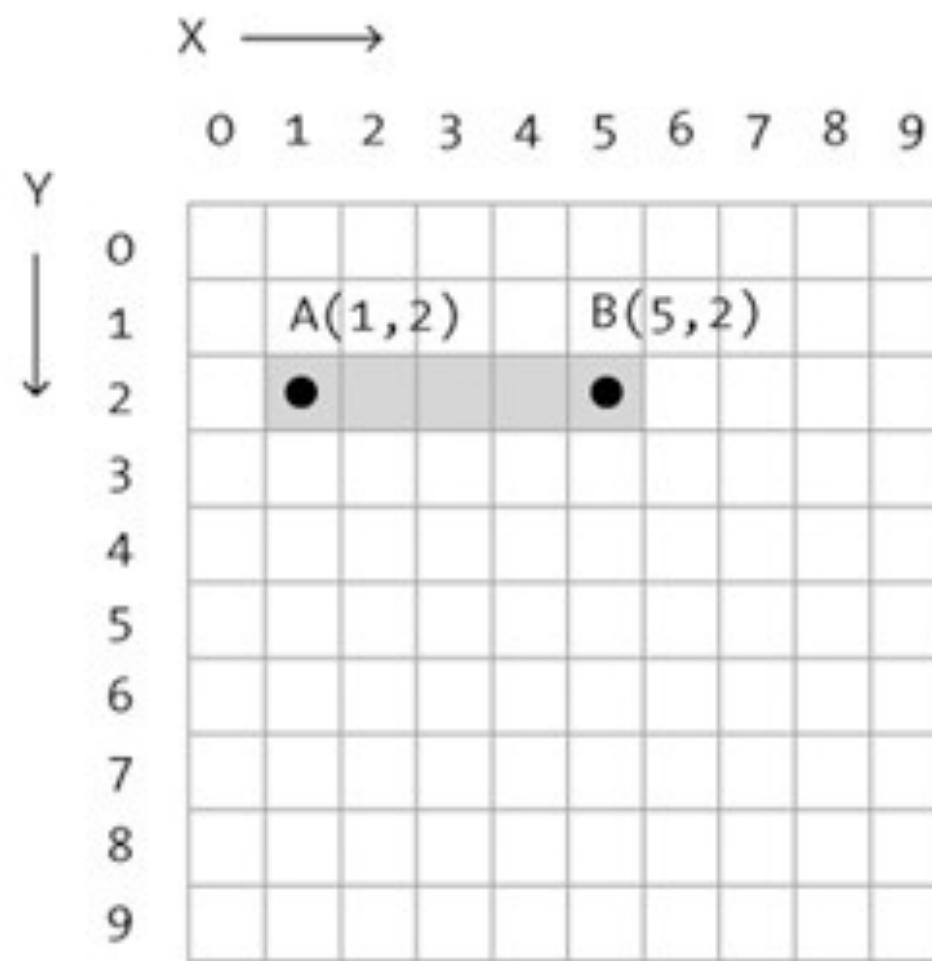


point(x,y);

Example:
A(4,5);

SHAPE

line(x1, y1, x2, y2);

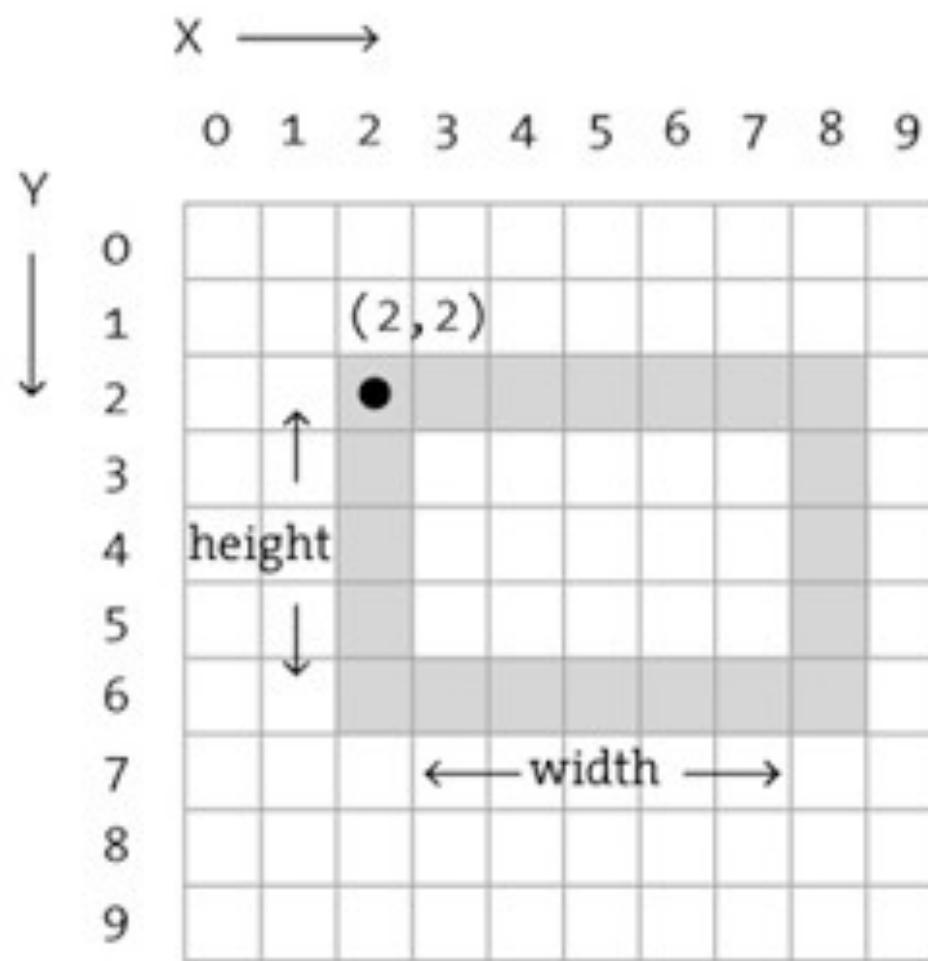


line(x1,y1,x2,y2);
 Point A Point B

Example:
line(1,2,5,2);

SHAPE

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



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

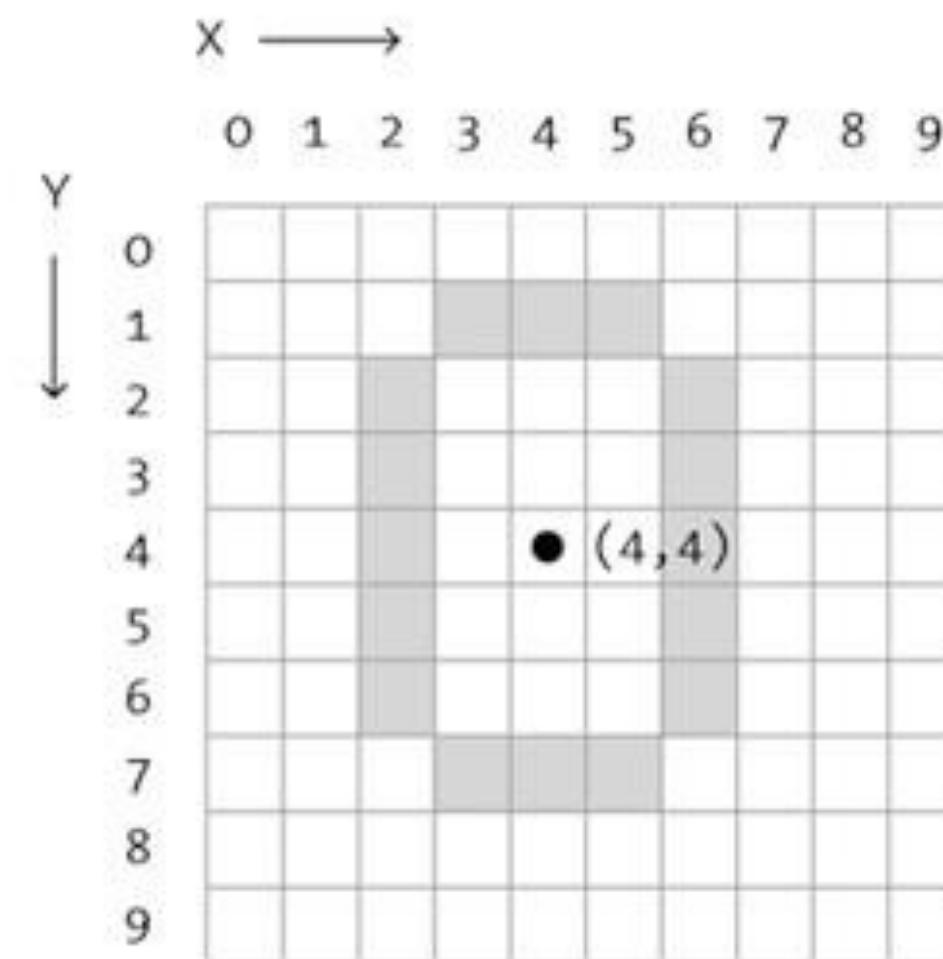
Example:

`rect(2,2,7,5);`

SHAPE

```
ellipseMode (CENTER) ;
```

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



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

Example:

```
ellipseMode(CENTER);  
ellipse(4,4,5,7);
```

SHAPE

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

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

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

SHAPE

More complex shapes available with:

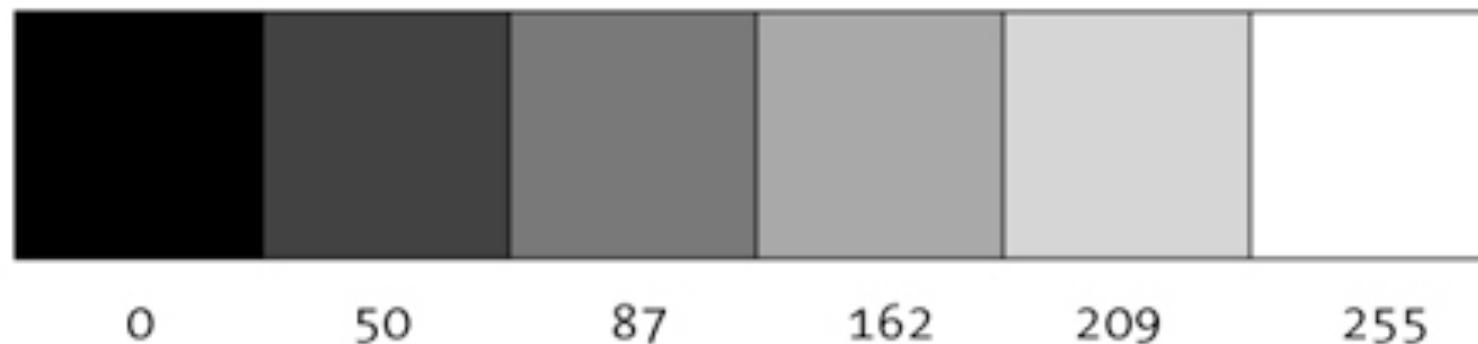
[beginShape\(\)](#) / [vertex\(\)](#) / [endShape\(\)](#)



COLOR

luminance

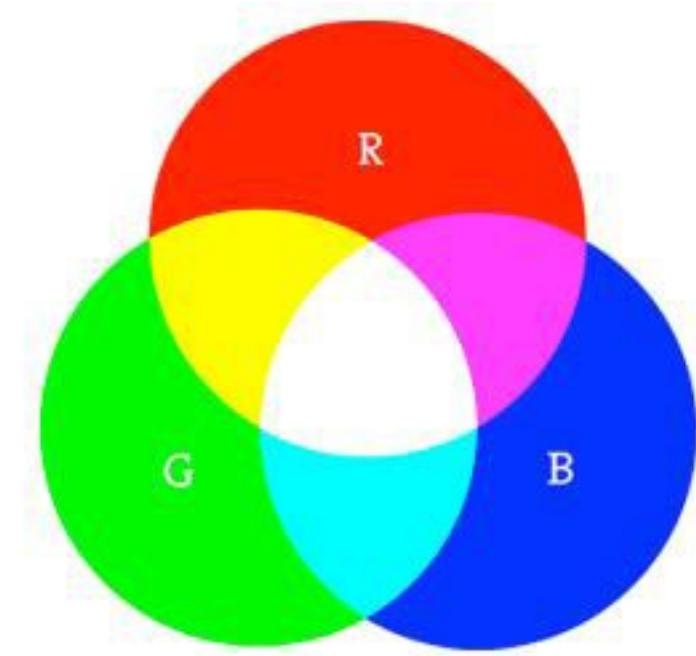
background(255);



COLOR

RGB (default)

```
color c1 = color(r, g, b);  
color c2 = #RRGGBB;
```



COLOR

RGBA

a = alpha / transparency / opacity

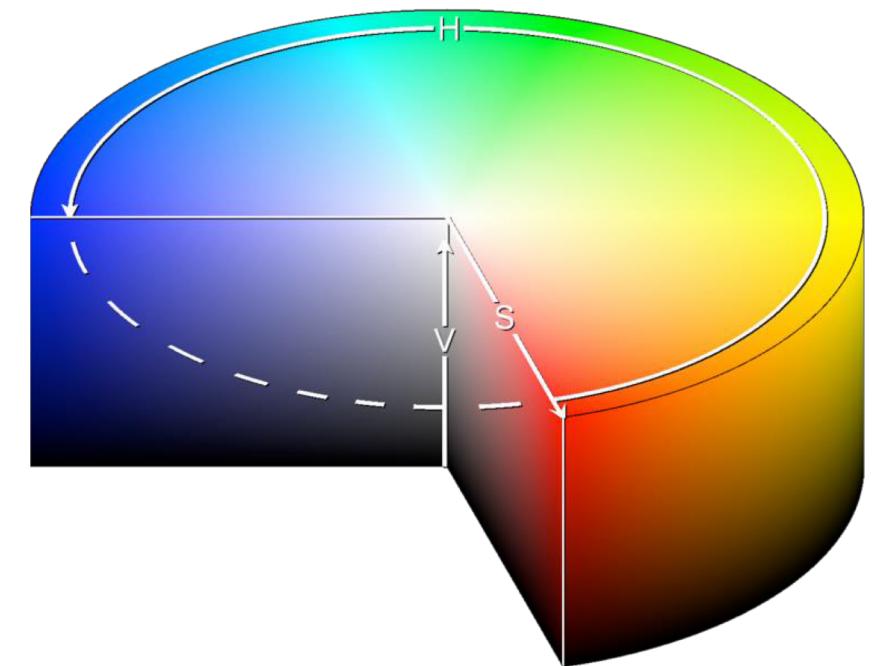
0 = transparent; 255 = opaque (solid)

color c1 = color(r, g, b, a);

COLOR MODES

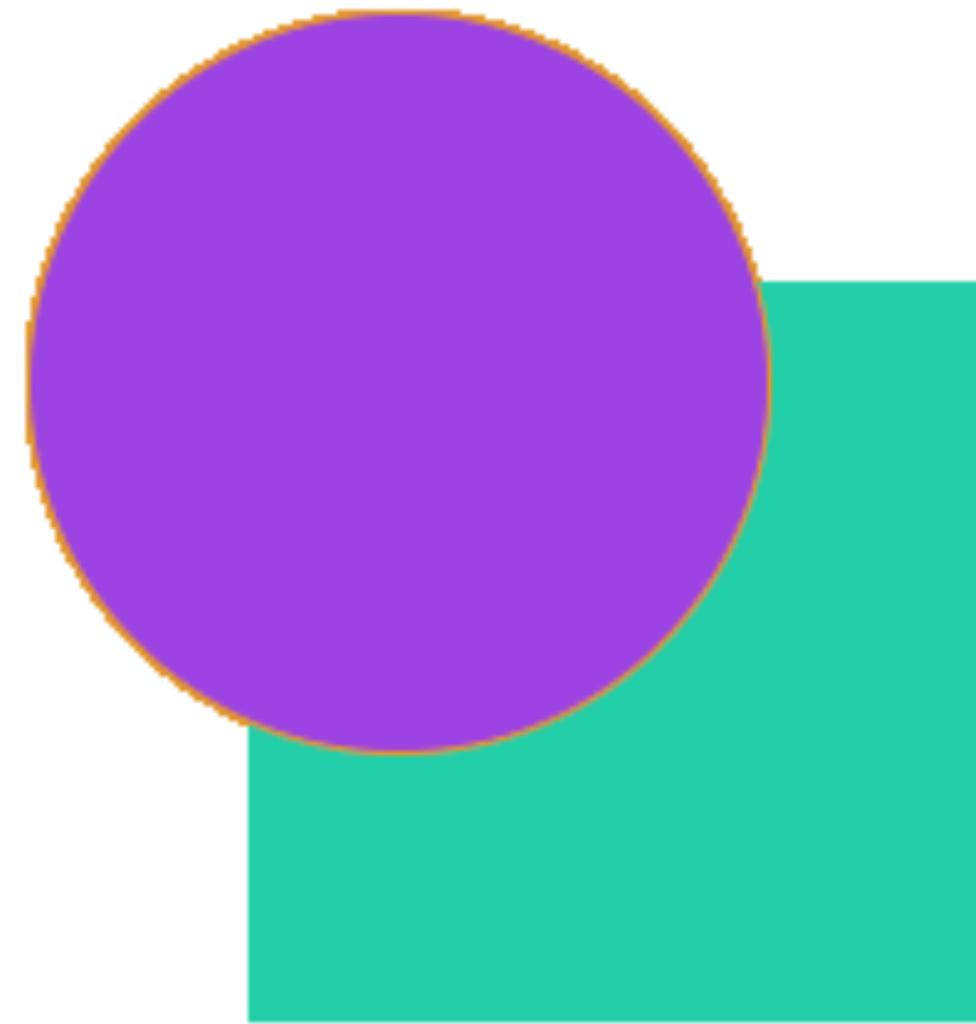
custom range: `colorMode (RGB, 100);`

HSB: `colorMode (HSB);`



PROPERTIES

```
noStroke();  
fill(c1);  
rect(...);  
  
fill(c2);  
stroke(c3);  
ellipse(...);
```



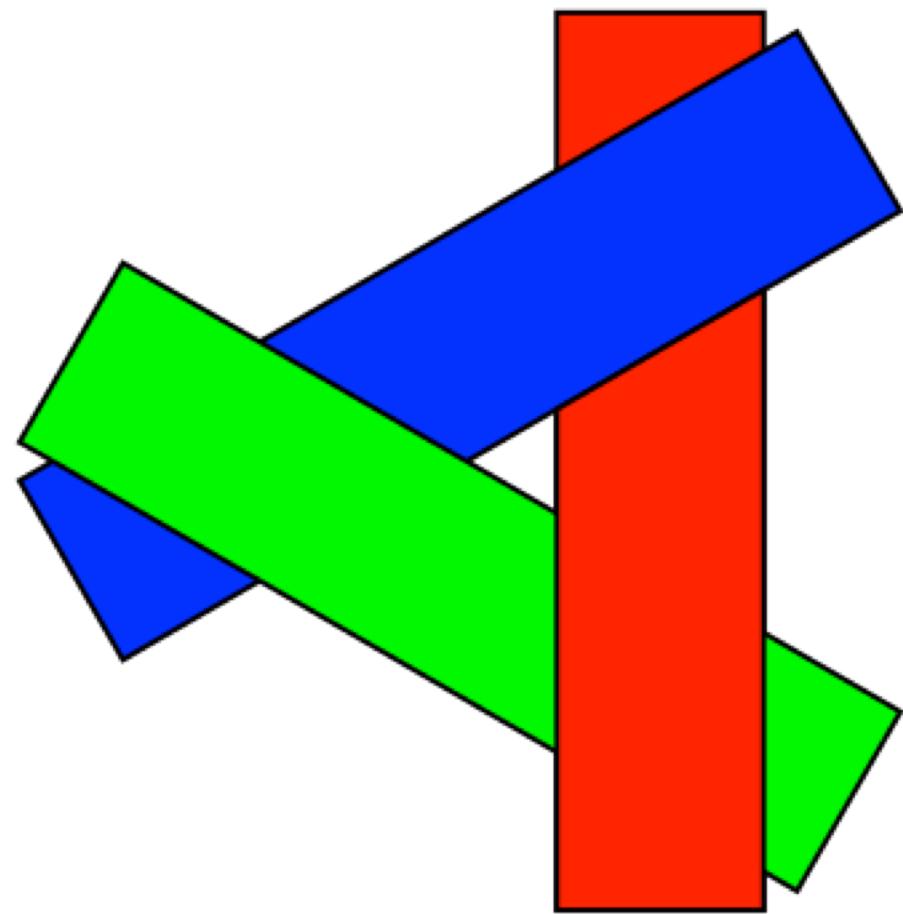
PROPERTIES

```
noFill();  
noStroke();  
ellipse(...);  
  
rect(...);
```

ORDER

shapes are painted one at a time
overlap can occur

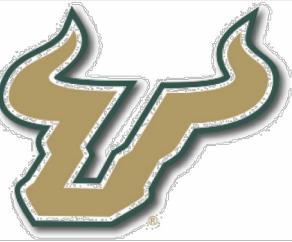
some shapes are not supported



TEXT

```
// in setup()  
PFont myFont;  
myFont = createFont("Georgia", 32);  
  
// in draw()  
textFont(myFont);  
textAlign(CENTER, CENTER);  
text("Hello, World!", width/2, height/2);
```

PROGRAMMING



STRUCTURE

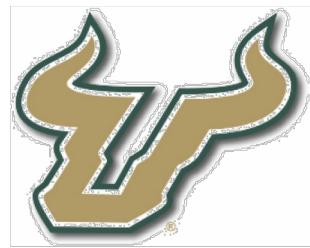
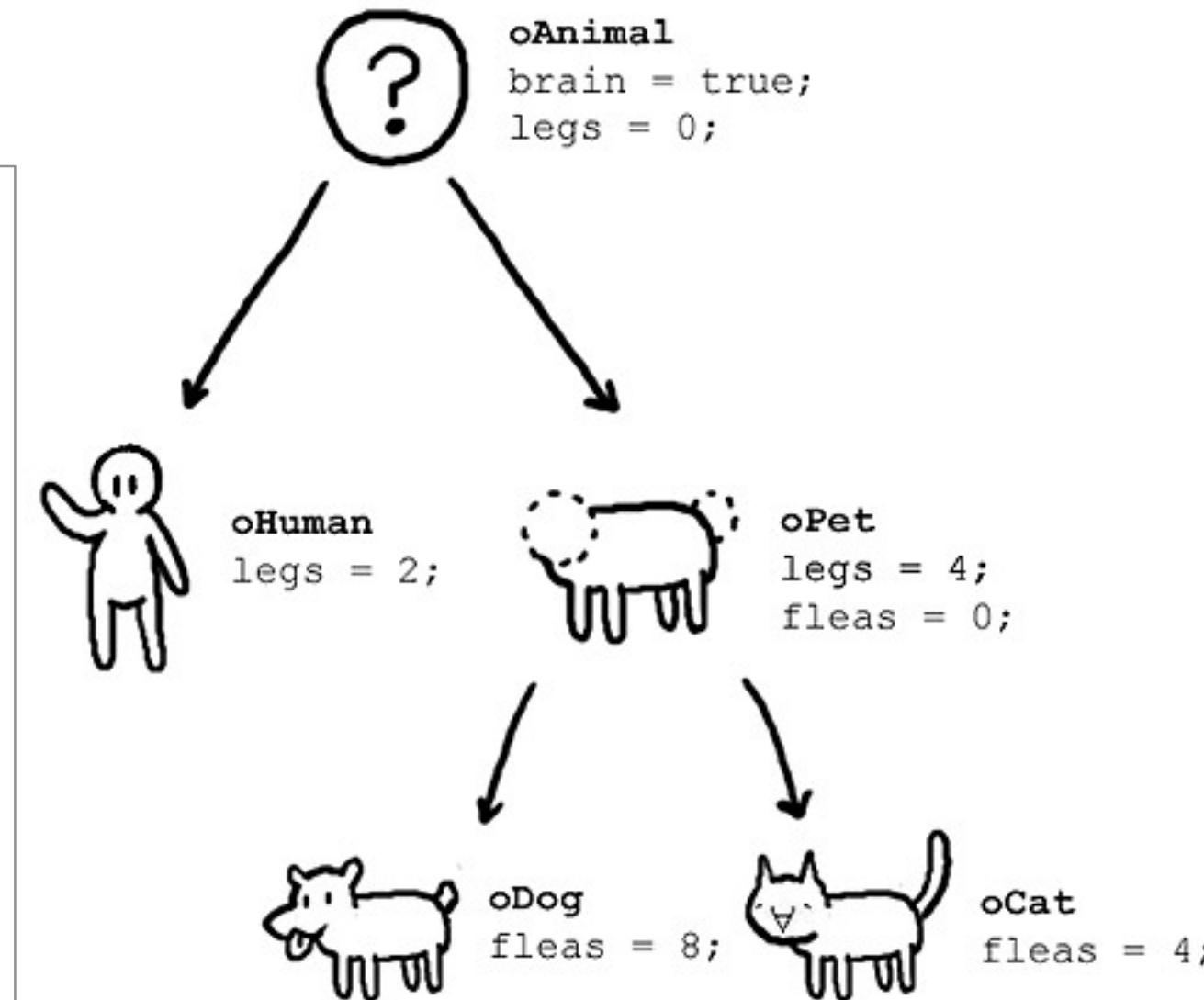
comments, variables, arrays, loops



OBJECT-ORIENTED

with classes

```
class oAnimal{  
    boolean brain;  
    int legs;  
  
    oAnimal() {  
        brain = true;  
        legs = 0;  
    }  
}
```



BUILT-IN TRIGONOMETRY FUNCTIONS

[acos\(\)](#) / [asin\(\)](#) / [atan\(\)](#) / [atan2\(\)](#)

[degrees\(\)](#) / [radians\(\)](#)

[cos\(\)](#) / [sin\(\)](#) / [tan\(\)](#)



BUILT-IN CALCULATION FUNCTIONS

CALCULATION

max() / min()

abs() / ceil() / constrain() / floor() / round()

lerp() / map() / norm()

dist() / mag()

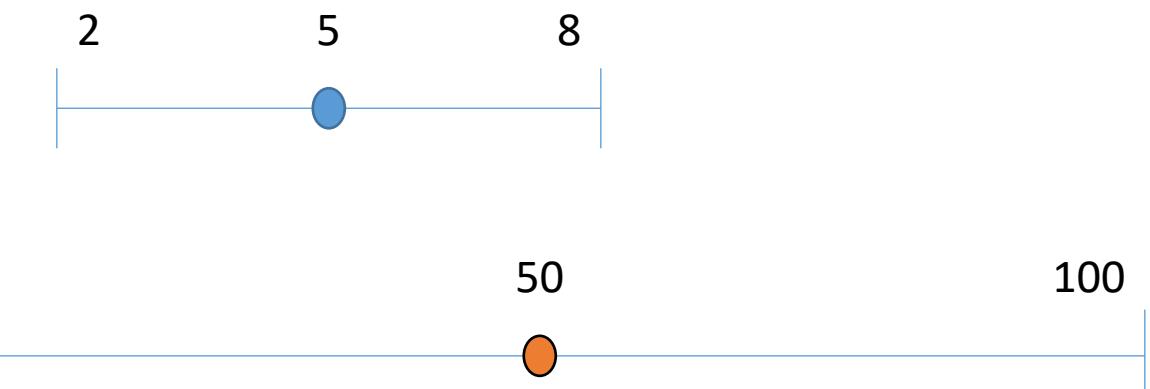
exp() / log() / pow() / sq() / sqrt()



MAP FUNCTION

`map(5, 2, 8, 0, 100)`

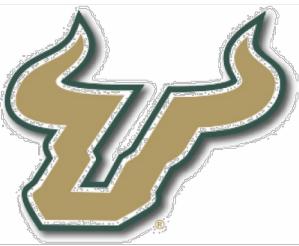
input value input range output range



CONSTRAIN FUNCTION

`constrain(10, 2, 8)`

input value input range



PVECTOR

Container for points and vectors

Variety of functionalities—vector arithmetic,
dot/cross products, normalization, and interpolation

<https://processing.org/reference/PVector.html>

[http://processing.github.io/processing-
javadocs/core/processing/core/PVector.html](http://processing.github.io/processing-javadocs/core/processing/core/PVector.html)



OBJECT STORAGE

ArrayList (**also** FloatList, IntList,
StringList)

HashMap (**dict:** also FloatDict, IntDict,
StringDict)

(and anything else Java!)



DATA LOADING

Built in parsers for:

csv, tsv – [loadTable\(\)](#) / [Table](#) / [TableRow](#)

xml – [loadXML\(\)](#) / [XML](#)

json – [loadJSONObject\(\)](#) / [loadJSONArray\(\)](#) /
[JSONObject](#) / [JSONArray](#)



INTERACTION: MOUSE

```
void mouseClicked() {  
  
    if(mouseButton == LEFT)  
        fill(0);  
  
    else if(mouseButton == RIGHT)  
        fill(255);  
  
    else  
        fill(126);  
}
```

void **mousePressed()**

void **mouseReleased()**

void **mouseClicked()**

void **mouseDragged()**

void **mouseMoved()**

void **mouseWheel()**

mouseX

mouseY

pmouseX

pmouseY

INTERACTION: KEYBOARD

```
void keyTyped() {  
  
    if(key == 'b')  
        fill(0);  
  
    else if(key == 'w')  
        fill(255);  
  
    else  
        fill(126);  
}
```

void keyPressed()

void keyReleased()

void keyTyped()

keyPressed

key

keyCode

ANIMATION

runs once



```
void setup(){
```

...
}

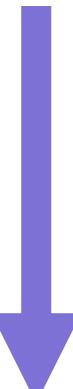
```
void draw(){
```

...
}

```
void mousePressed(){
```

...
}

cycles



when triggered

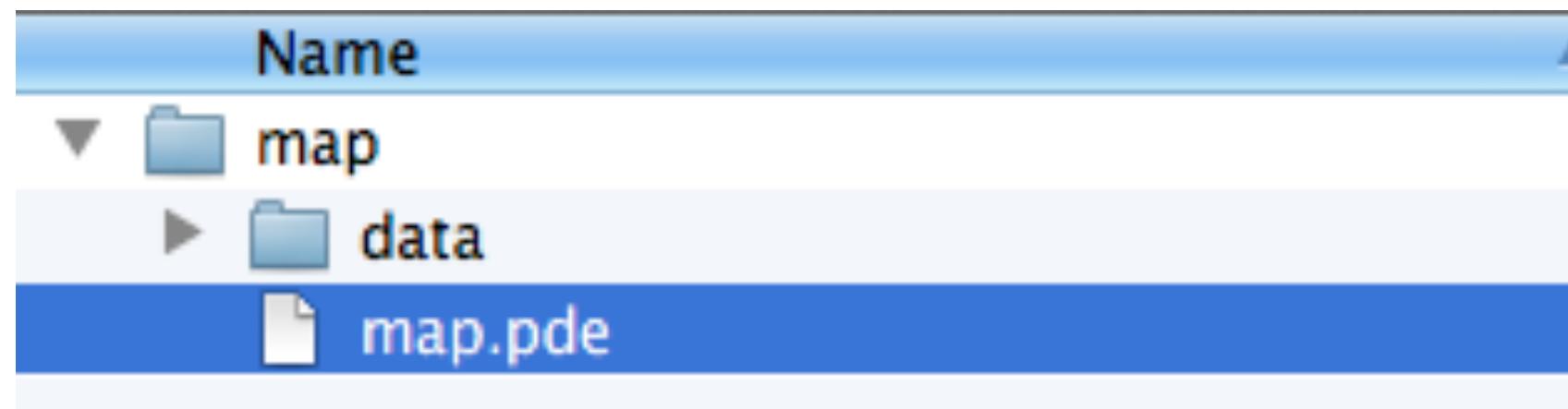


Calls with dotted
lines are automatic



FOLDER STRUCTURE

folder [NAME] & [NAME].pde must match



optional data folder (for images, input)

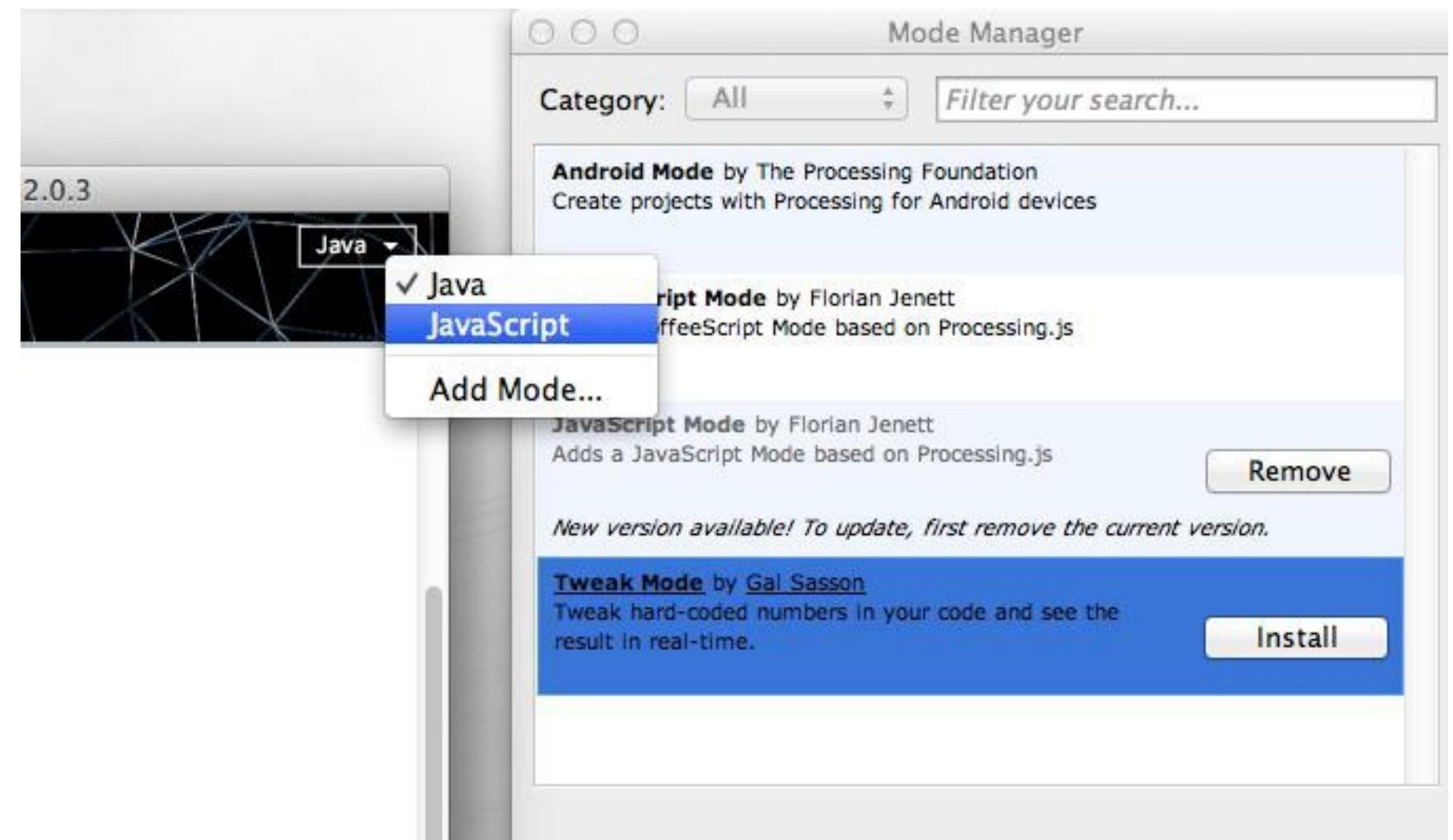
MODES

Java (default)

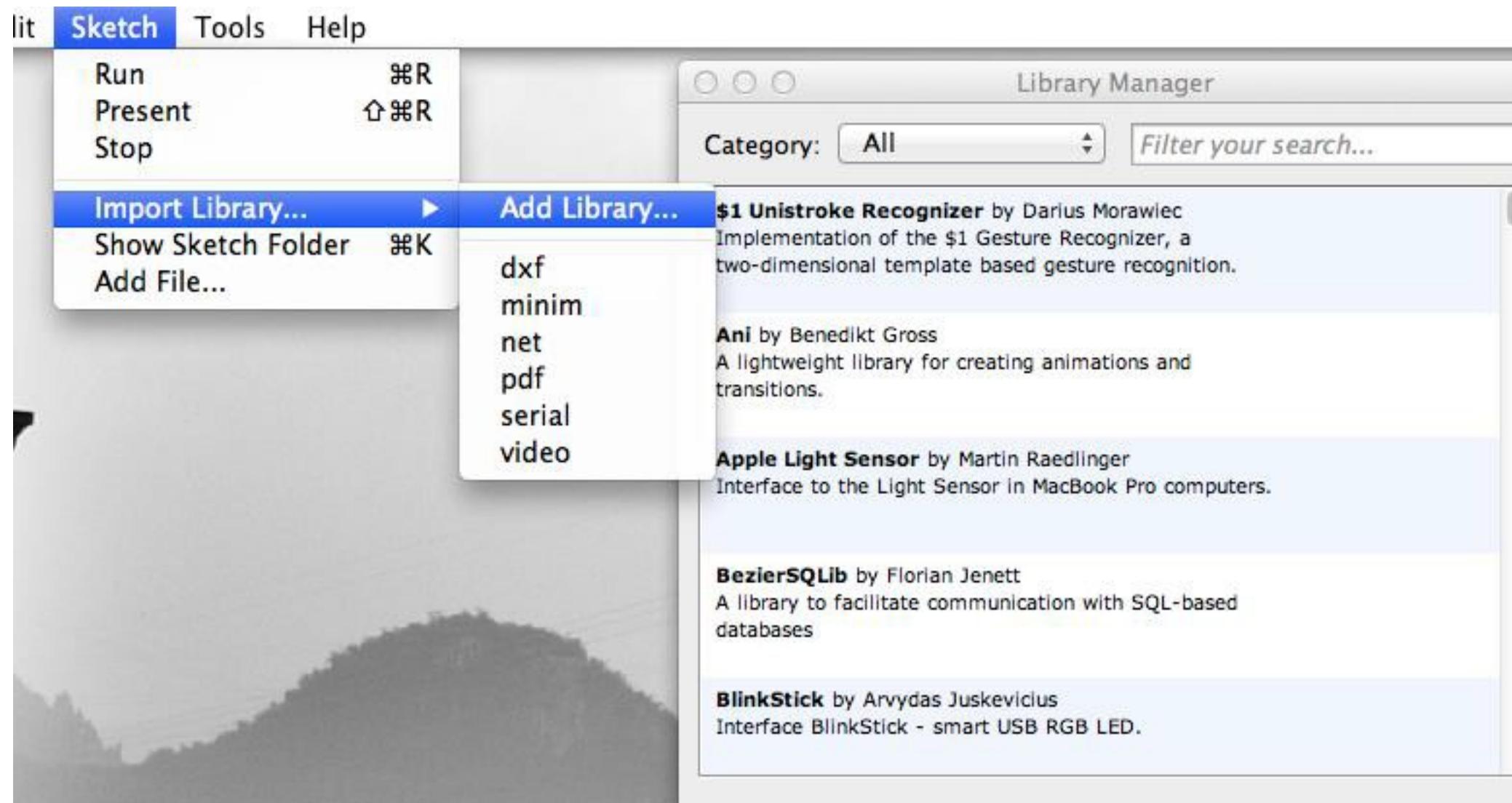
JavaScript

Android

etc.



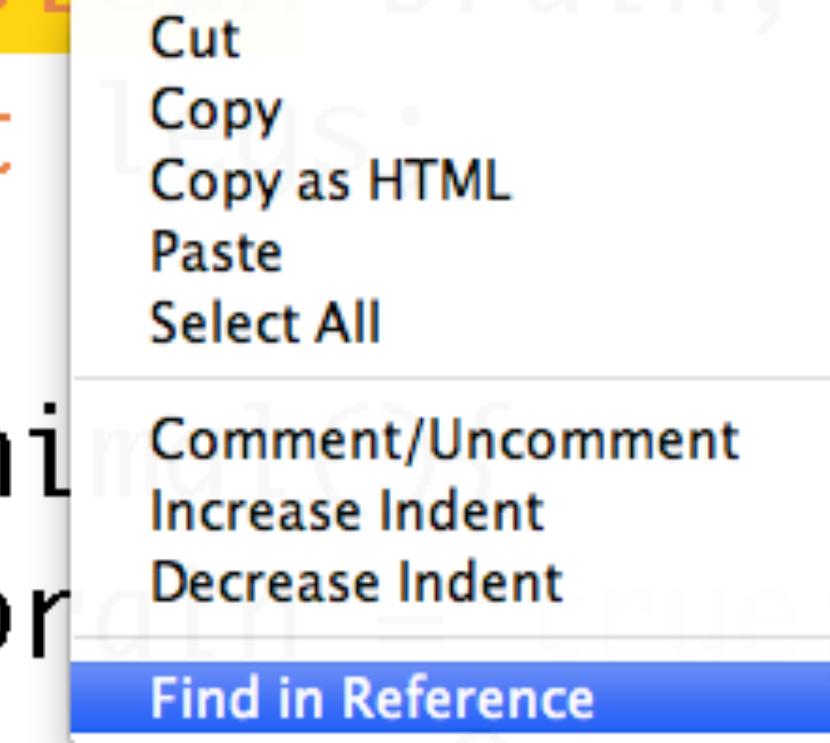
LIBRARIES



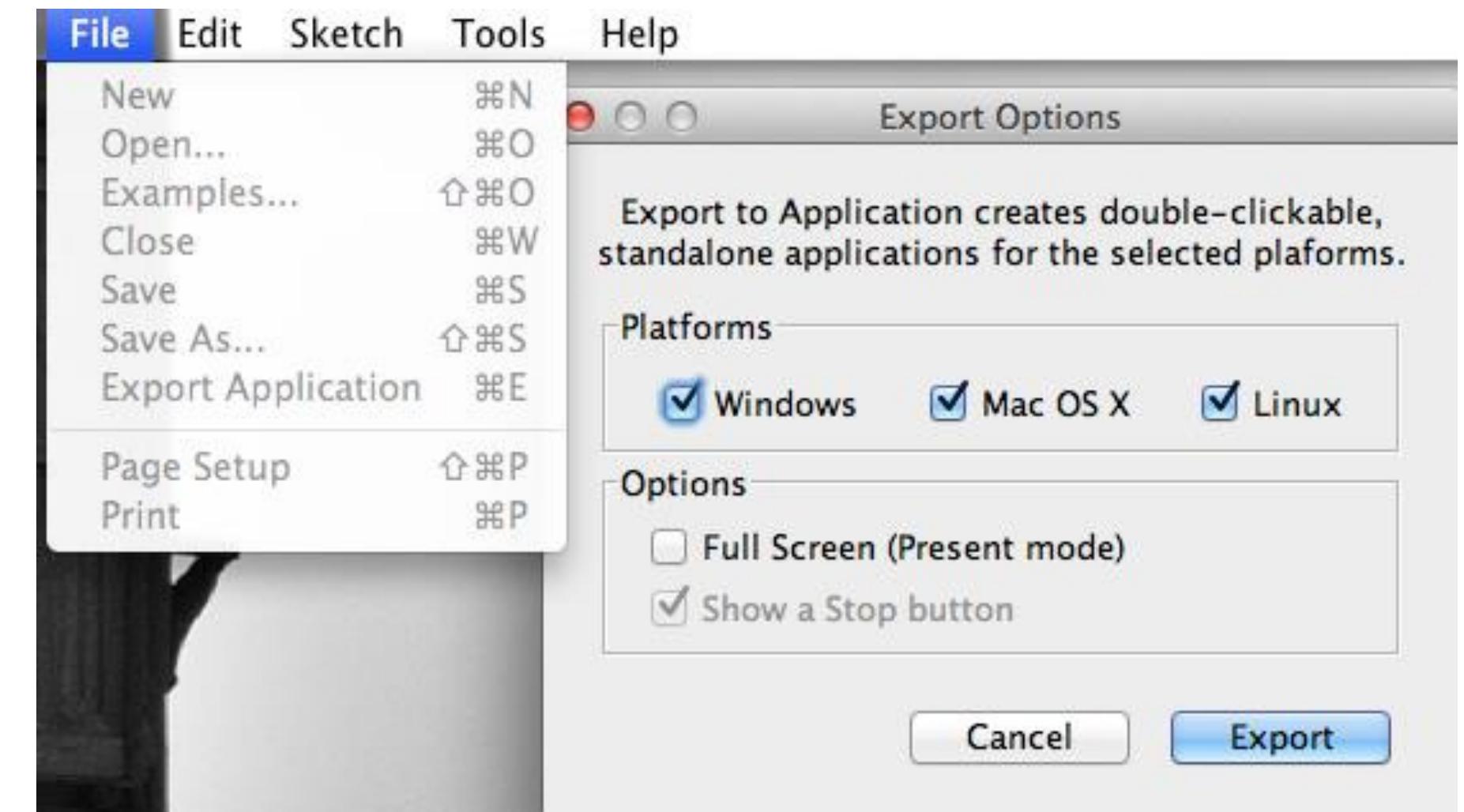
DOCUMENTATION
available online
also in the PDE

<http://processing.org/reference/>

```
class oAnimal{  
    boolean brain.  
    int  
    oAnimal  
    brain;  
    brain;  
    brain - 100 - 100;
```

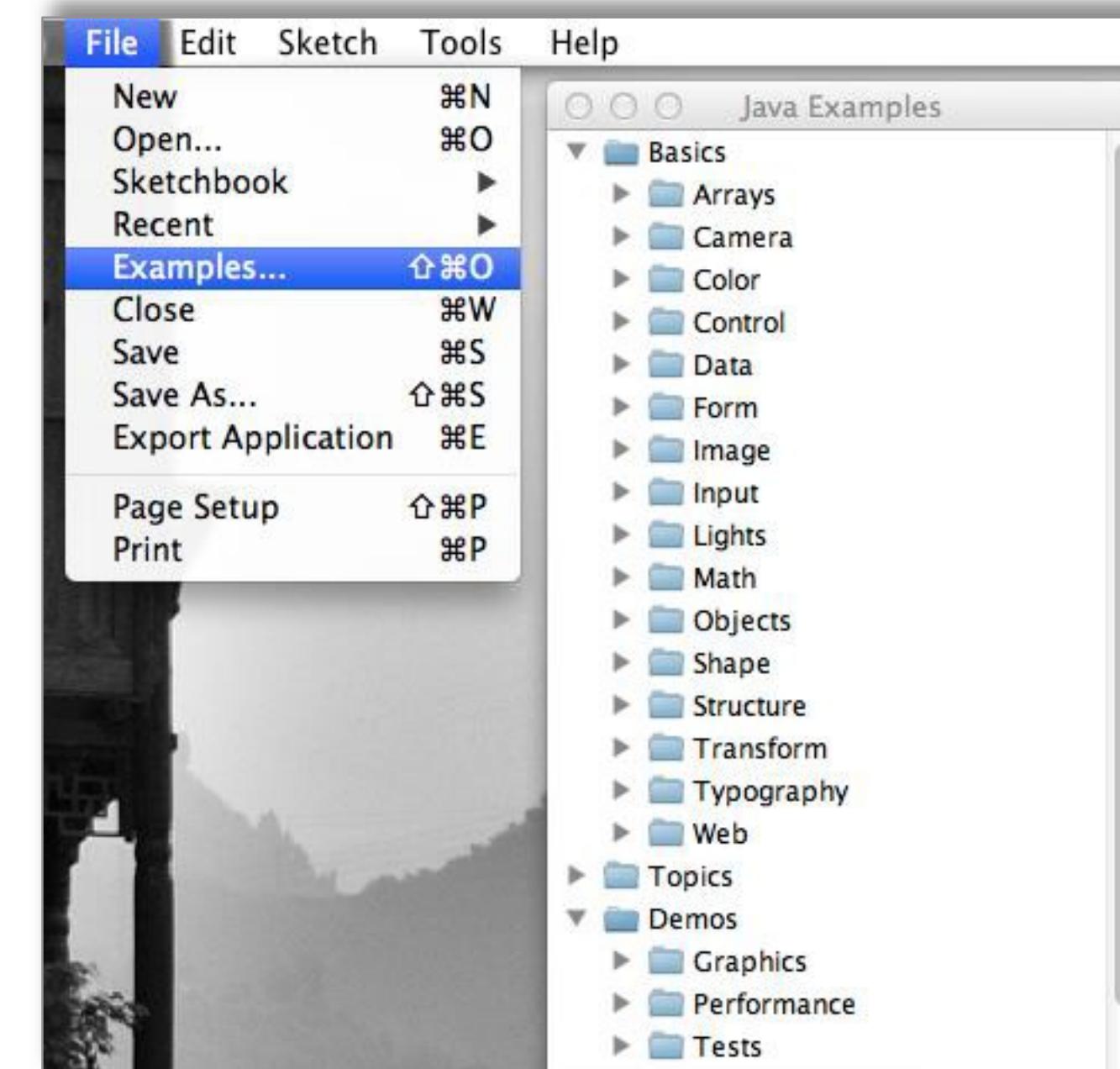


EXPORTING
creating applications is
simple



EXAMPLES

variety of samples



DEMO

