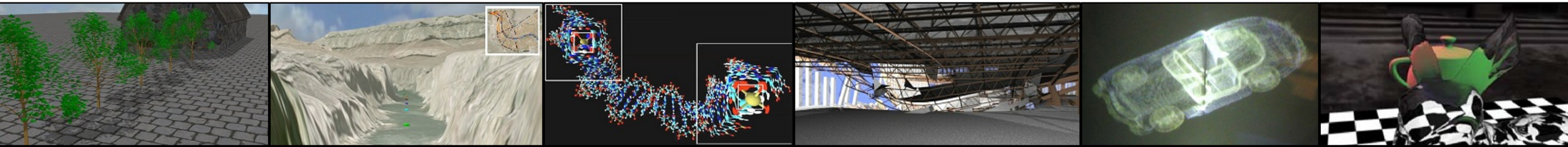
COT 4521: Introduction to Computational Geometry

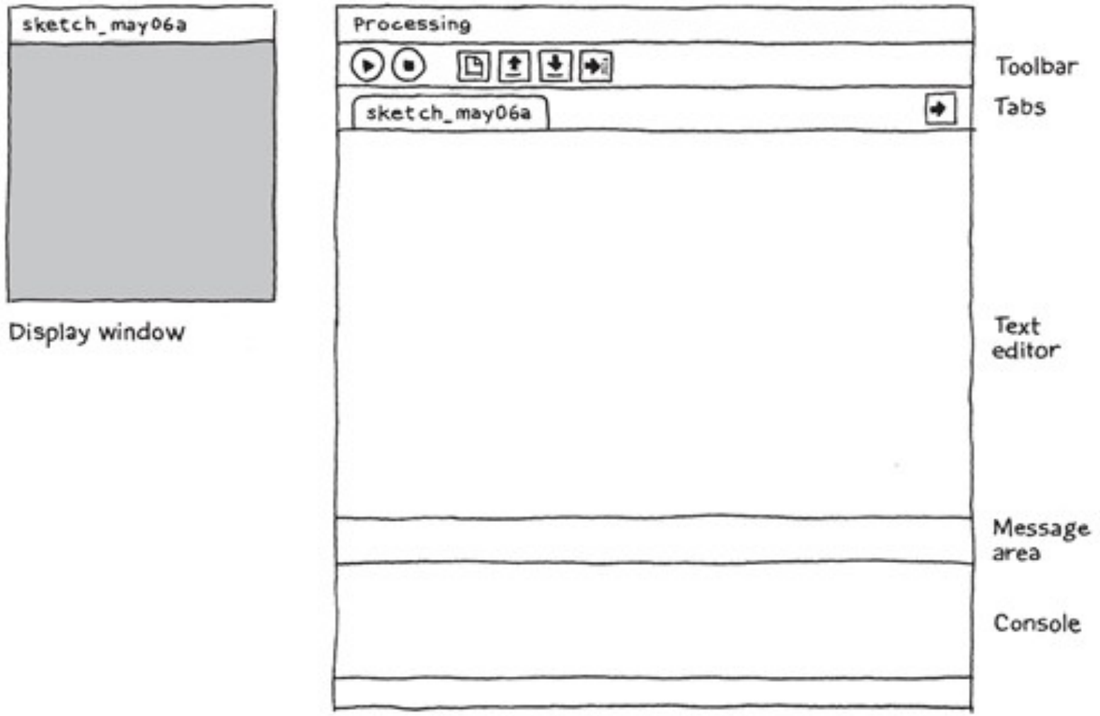
INTRODUCTION TO PROCESSING

Paul Rosen
Assistant Professor
University of South Florida



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

() (parentheses)
, (comma)
. (dot)
/* */ (multiline comment)
/** */ (doc comment)
// (comment)
; (semicolon)
= (assign)
[] (array access)
{} (curly braces)
catch
class
draw()
exit()
extends
false
final
implements
import
loop()
new
noLoop()
null
popStyle()
private
public

Shape

createShape()
loadShape()
PShape

2D Primitives
arc()
ellipse()
line()
point()
quad()
rect()
triangle()

Curves
bezier()
bezierDetail()
bezierPoint()
bezierTangent()
curve()
curveDetail()
curvePoint()
curveTangent()
curveTightness()

3D Primitives
box()

Color

Setting
background()
clear()
colorMode()
fill()
noFill()
noStroke()
stroke()

Creating & Reading
alpha()
blue()
brightness()
color()
green()
hue()
lerpColor()
red()
saturation()

Image

createImage()
PImage

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

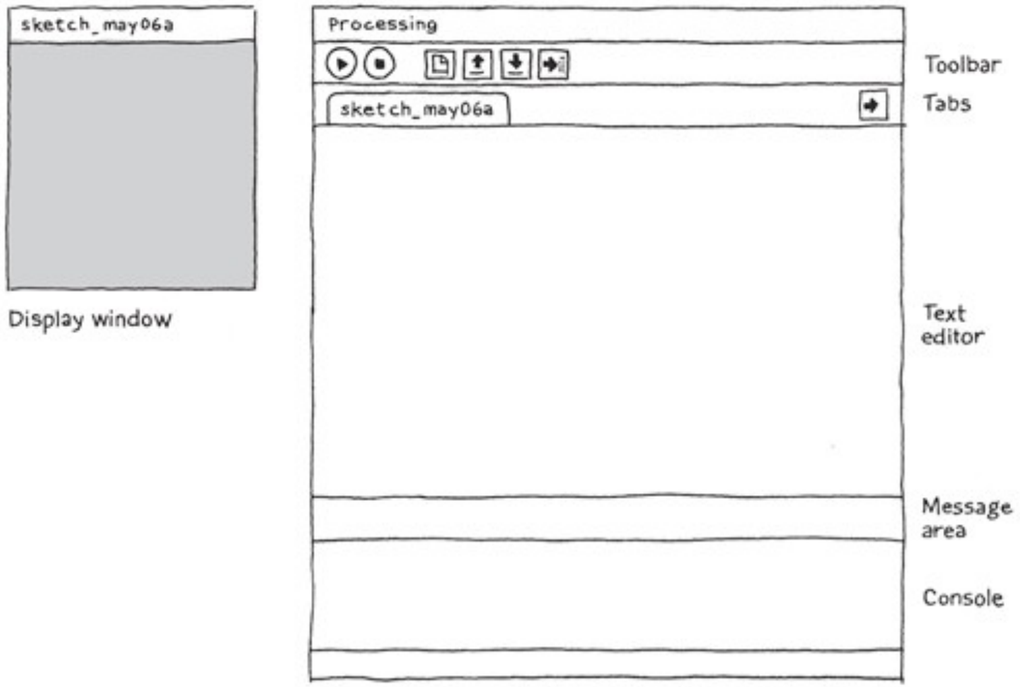
+ similar syntax & portability

+ lots of user-contributed libraries

+ fast to startup

WHY PROCESSING?

program = sketch



WHY PROCESSING?

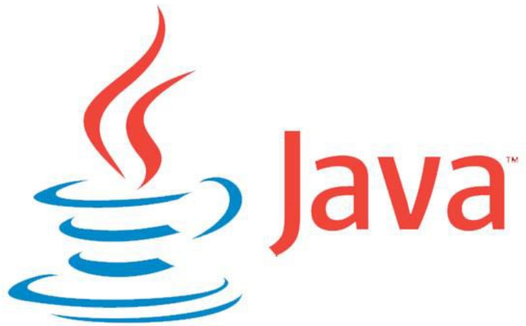
programming syntax

```
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
```



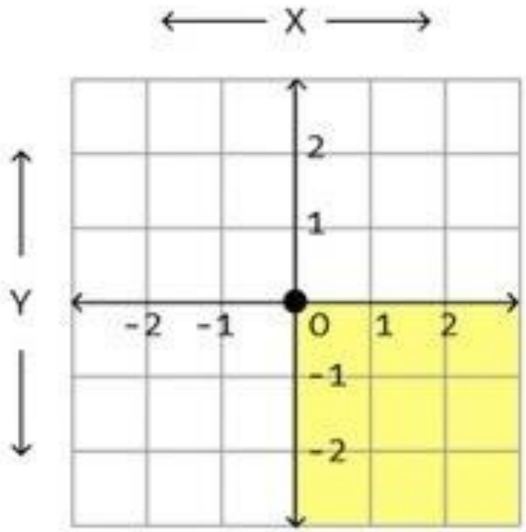
WHY PROCESSING?

```
println(“Hello, World!”);
```

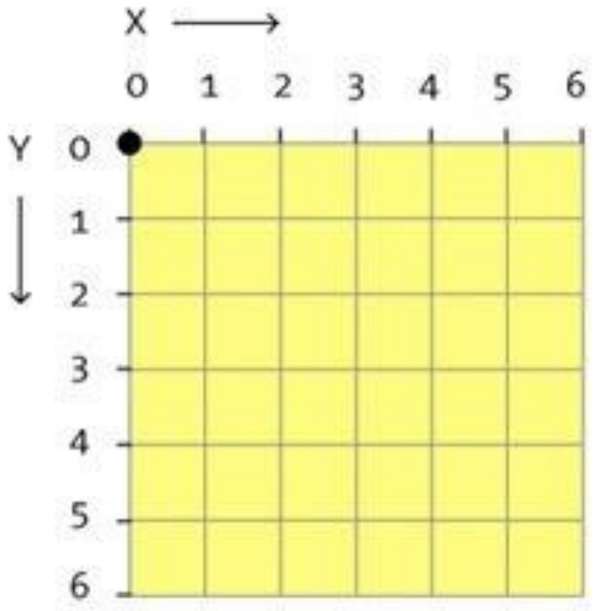
GRAPHICS

MONITORS

grid of pixels



Right Hand Coordinate System



Monitor Coordinate System
(Left Hand Coordinate System)

Right Hand (RHC)
vs
Left Hand (LHC)
Coordinate Systems

Everything we do in this class assumes RHC (not in processing)

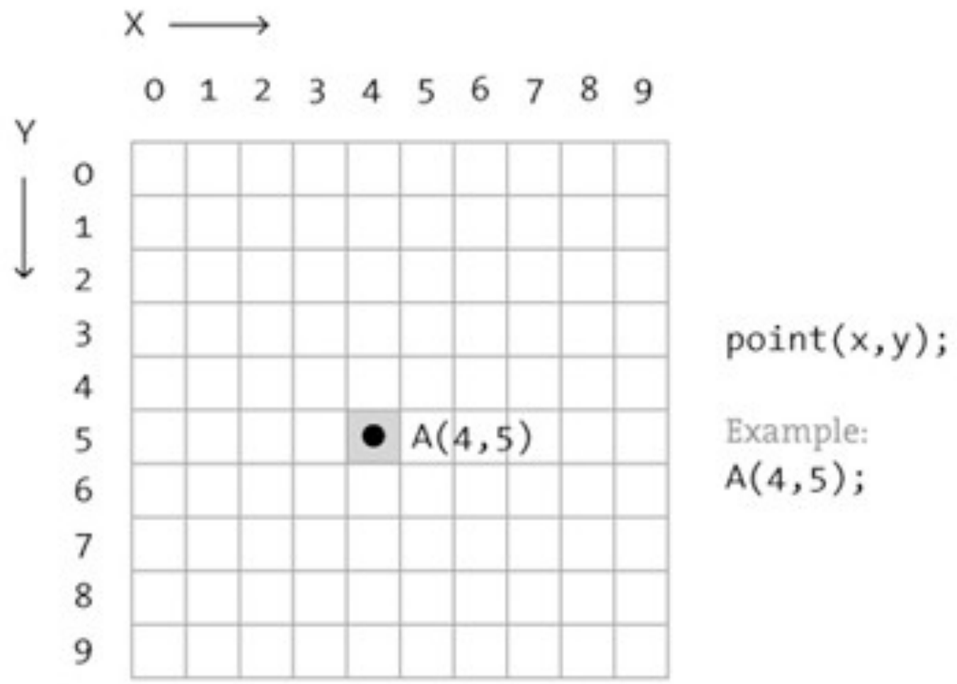
I've included some code in the skeleton to convert Processing into a RHC

LHC System (Processing default)

RHC System (provided modification)

SHAPE

```
point(x, y);
```



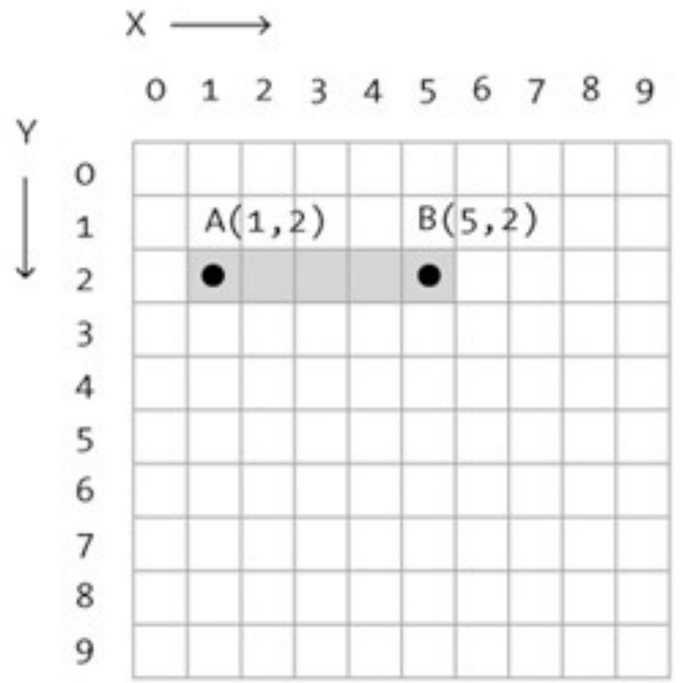

```
point(x,y);
```

Example:

```
A(4,5);
```

SHAPE

```
line(x1, y1, x2, y2);
```




```
line(x1,y1,x2,y2);
```

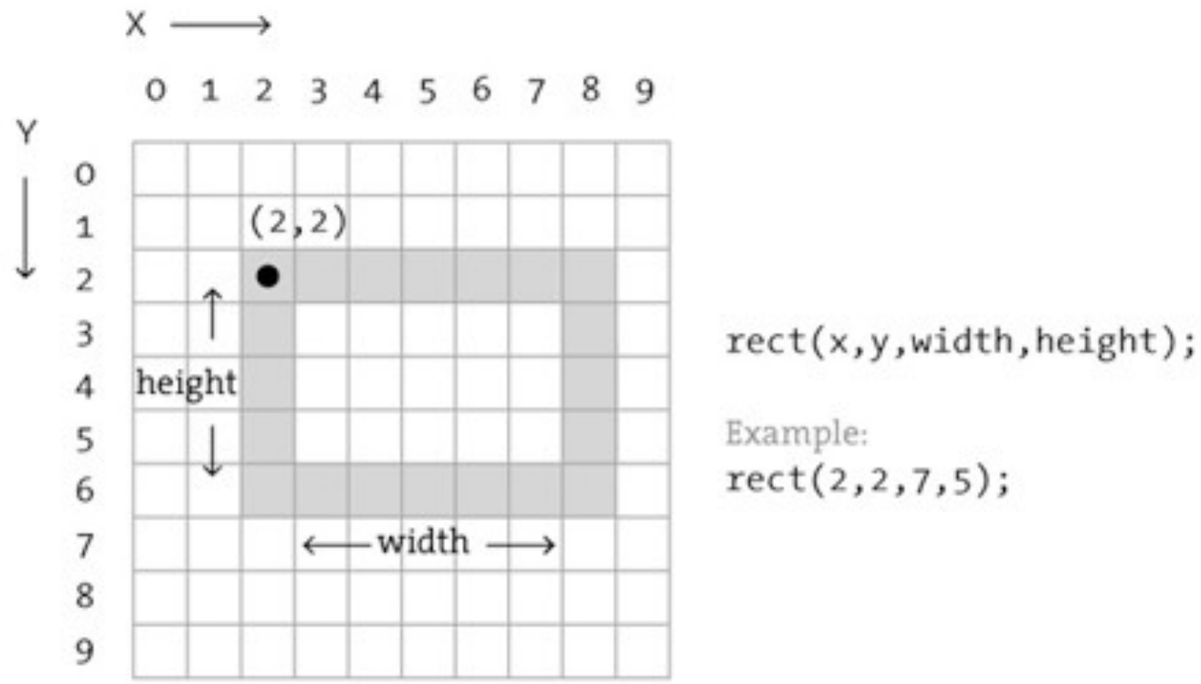
Point A Point B

Example:

```
line(1,2,5,2);
```

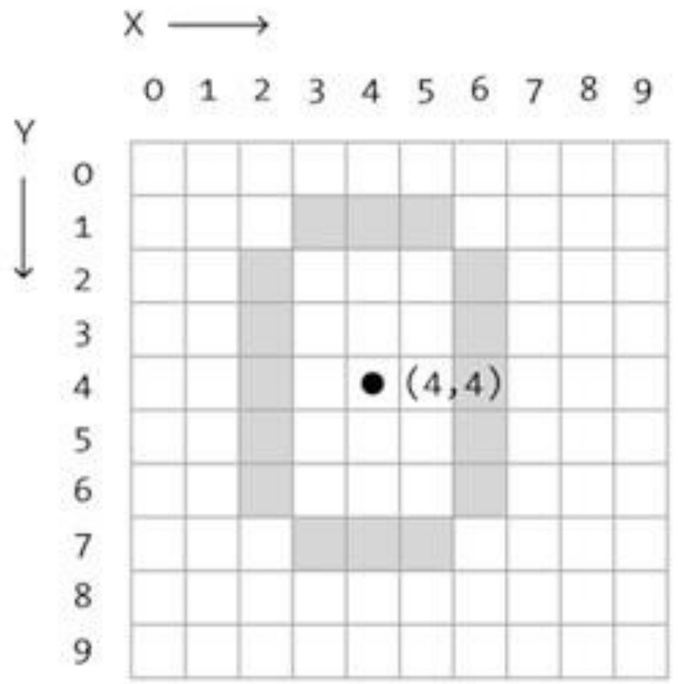
SHAPE

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



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

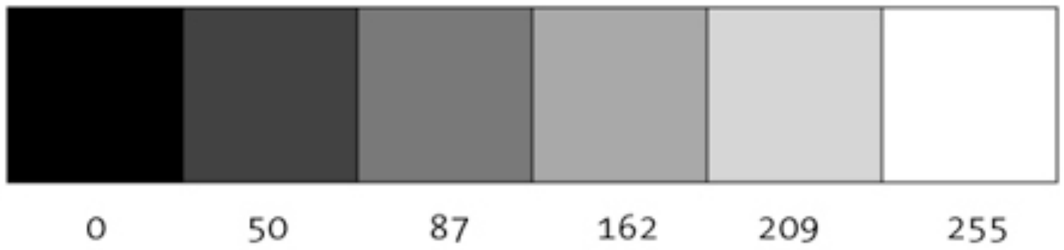
More complex shapes (convex polygons) available with:

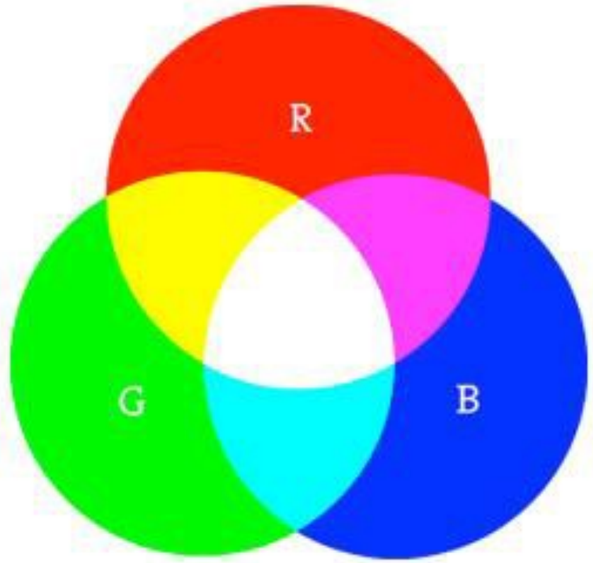
beginShape() / vertex() / endShape()

Color

luminance (black & white; grayscale)

```
background(255); // clear background
```





COLOR

RGB (default)

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

Color

RGBA

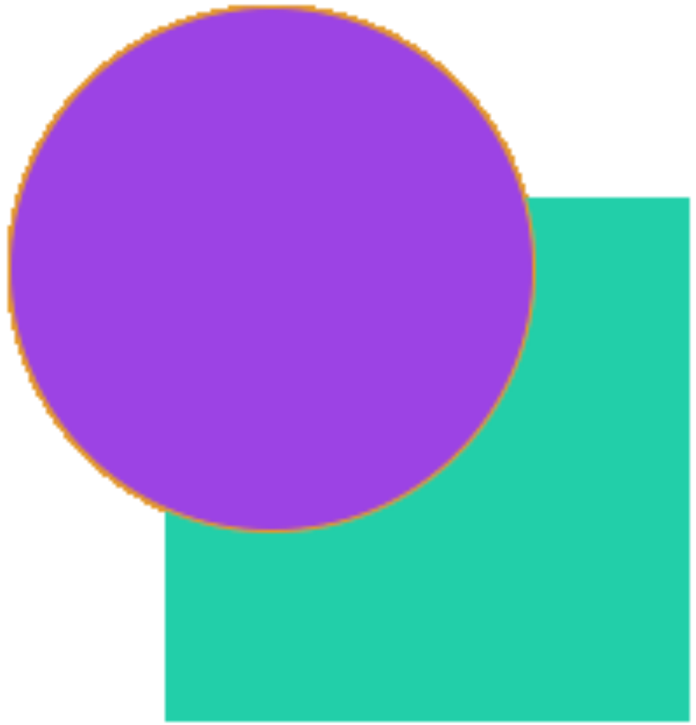
a = alpha / transparency / opacity

0 = transparent; 255 = opaque (solid)

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

PROPERTIES

```
noStroke();
fill(c1);
rect(...);

fill(c2);
stroke(c3);
ellipse(...);
```



PROPERTIES

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


rect(...);
```

PROPERTIES

Graphics is a STATE MACHINE.

When you set a state (no matter where in the code),
that state will remain until changed

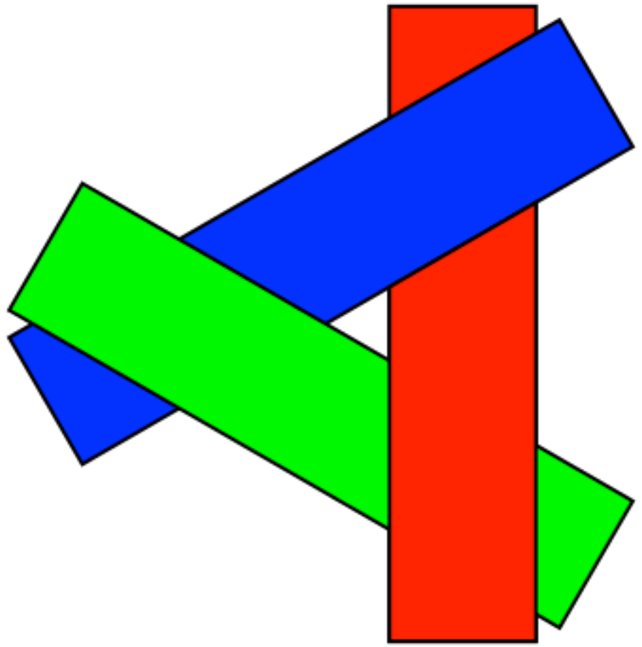
This is particularly noticeable with color

ORDER

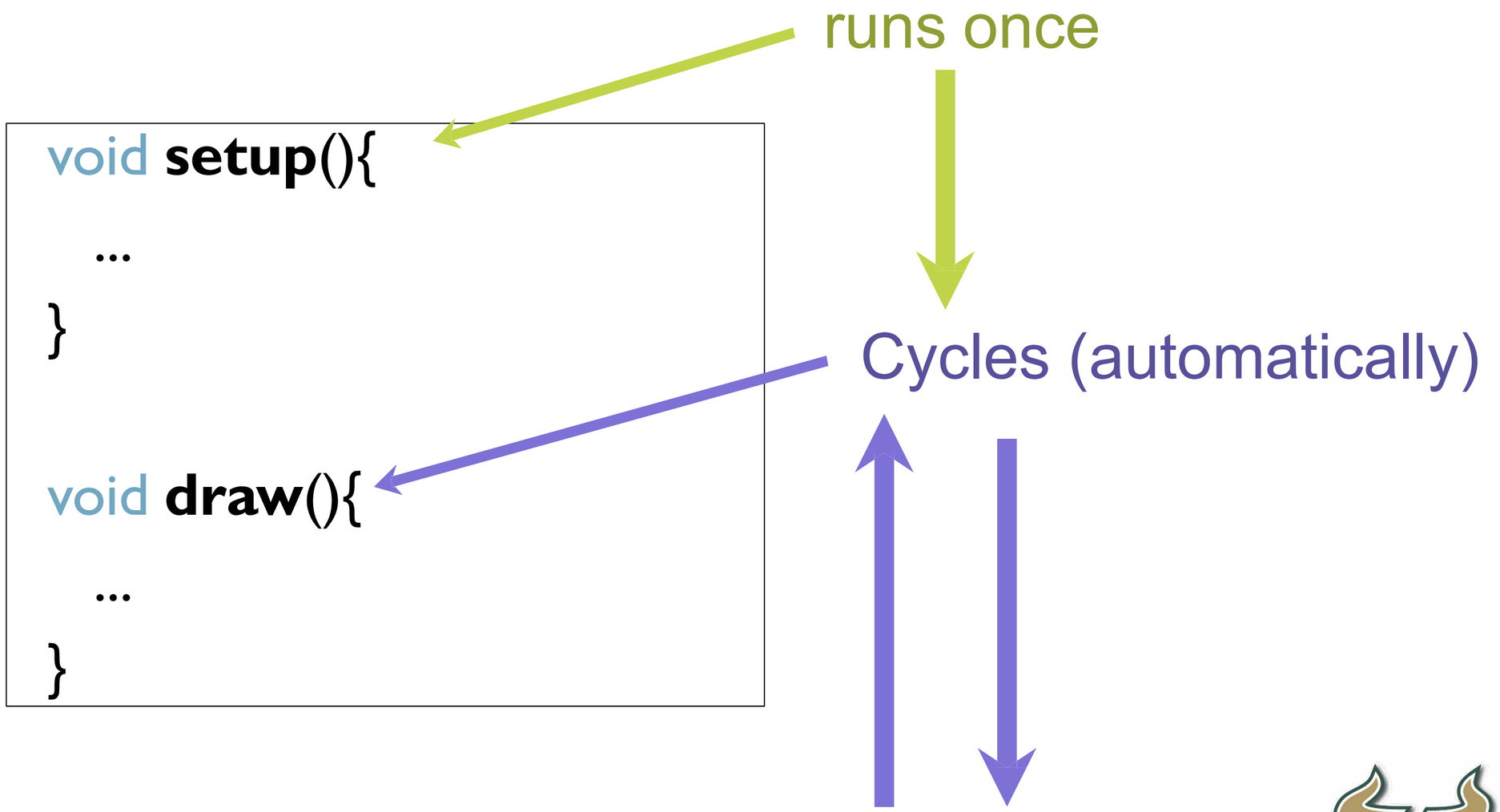
shapes are painted one at a time

overlap can occur

some shapes are not supported



ANIMATION



TEXT

```
// in setup()
PFont myFont;
myFont = createFont("Georgia", 32);


// in draw()
textFont(myFont);
textAlign(CENTER, CENTER);
text("Hello, World!", width/2, height/2);
```

THERE IS SOME EXTRA CODE PROVIDED FOR RHC

PROGRAMMING

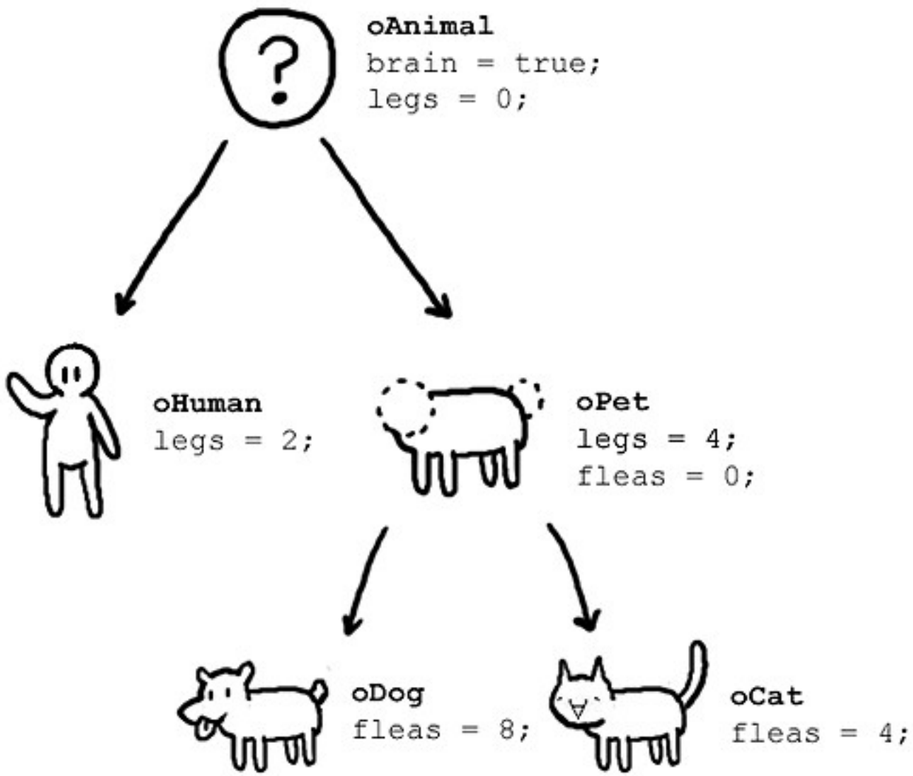
STRUCTURE

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()

PVECTOR

Container for points and vectors

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

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

http://processing.github.io/processing-javadocs/core/processing/core/PVector.html

STUDY AND USE THIS CLASS!

Object Storage

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

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

`Table`, `XML`, `JSON`

(and anything else Java!)

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

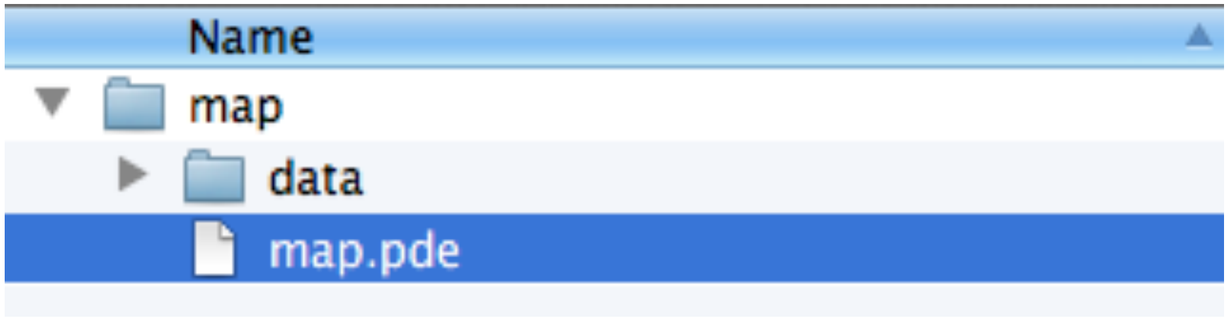
```
void keyTyped(){

  if(key == 'b')
    fill(0);

  else if(key == 'w')
    fill(255);

  else
    fill(126);
}
```

```
void keyPressed()

void keyReleased()

void keyTyped()
```

```
keyPressed
key
keyCode
```

FOLDER STRUCTURE

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



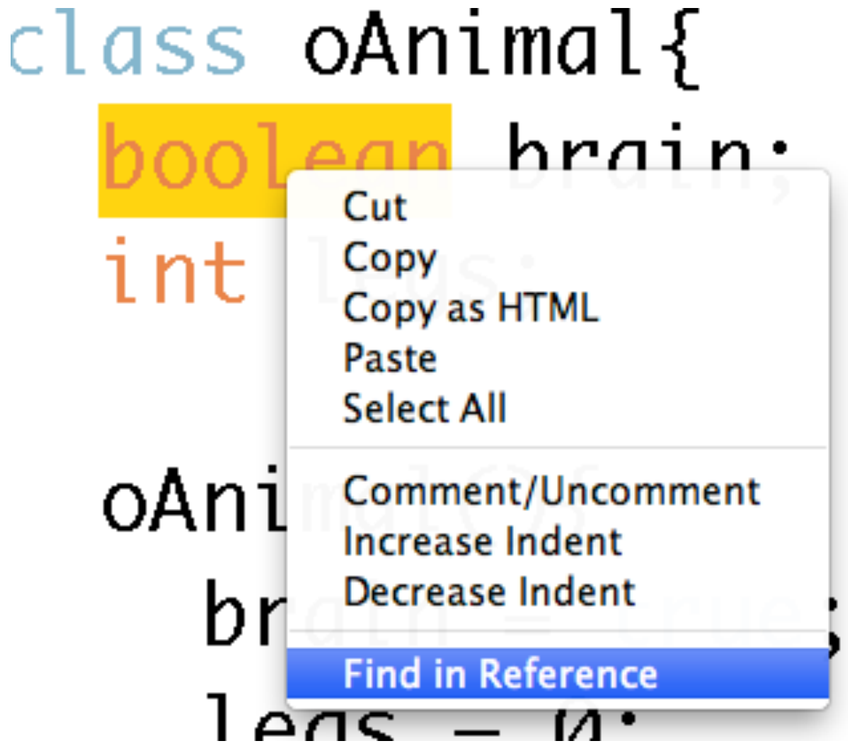
optional data folder (for images, input)

DOCUMENTATION

available online

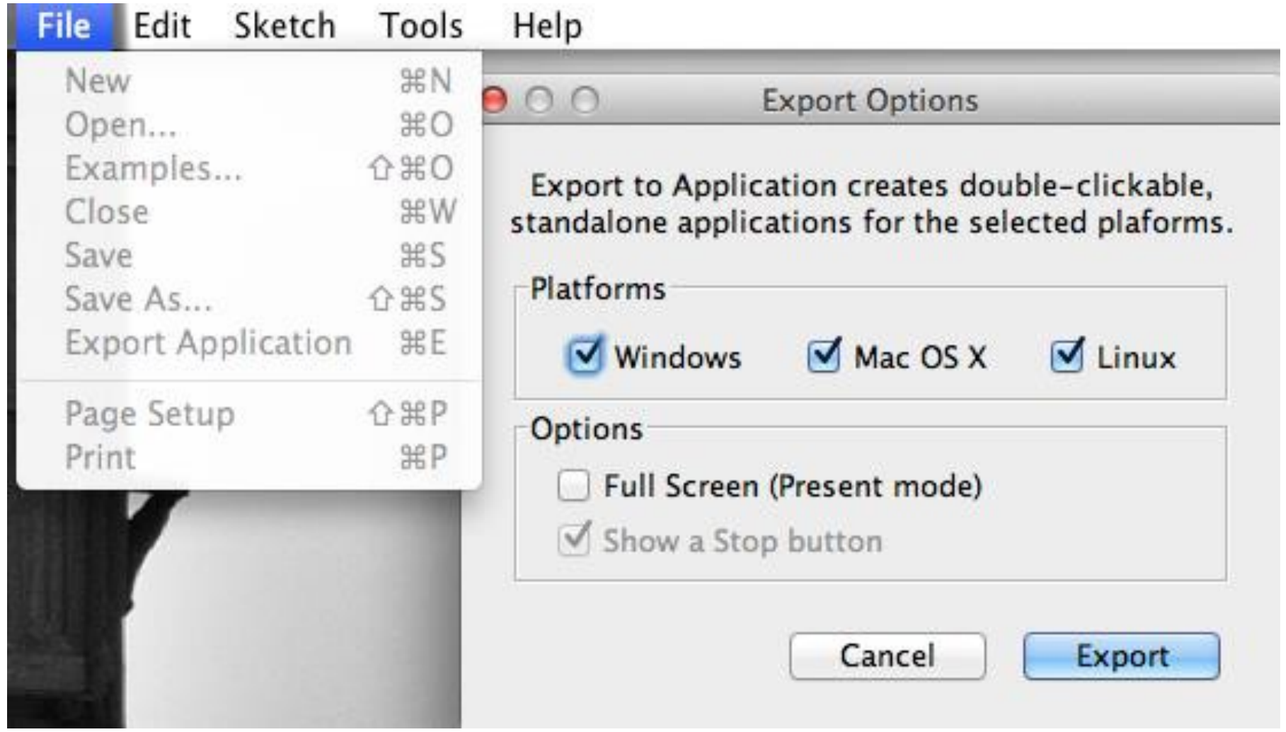
also in the PDE

http://processing.org/reference/



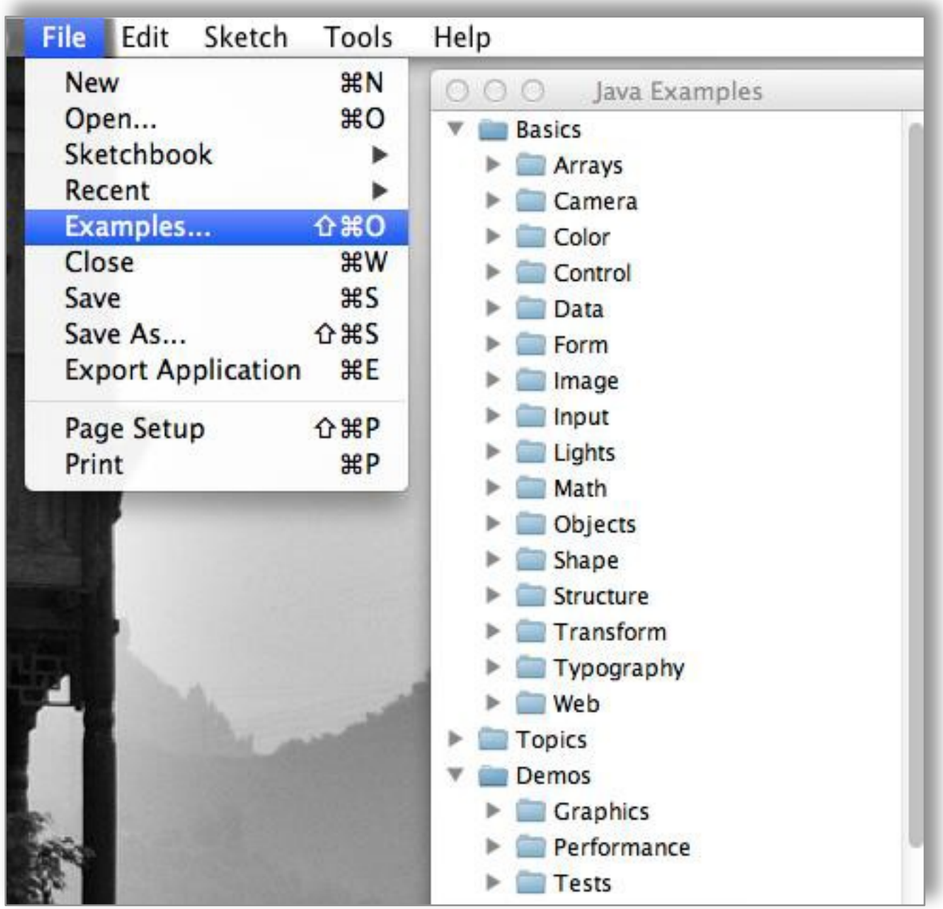
EXPORTING

creating applications is simple



EXAMPLES

variety of samples



Demo