today

Due: Change the World - not a school assignment
Intro + Draw
Introduce Ex 1: Draw it!
Sign up: Student Presentations

Reading Ch 1, 2, 3 (only up to 3-2)

next class

Due: Paper sketches for Ex 1
code: Draw it!
Start our Student Presentations

1 Basics and Draw

what is programming?

Needs human beings to provide instructions.
Write algorithms.

Need specificity - syntax; condition associated with logic

Pseudo-code: get use to doing, plan out the logic in your own language

Philosophy of Incremental development/modular system:
Put everything into one set of code all at once is kind of impossible.
Need lots of little pieces-write mini programs one at a time.
Fit the pieces together!

what is an algorithm?

instructions for achieving a task

the idea is like making muffins

1. Whisk dry ingredients
2. In another bowl, mix wet ingredients
3. Pour wet on top of dry and fold together
4. Scoop into muffin tins
5. Bake at 400 F
6. Remove from oven

what is syntax?

The grammatical rules and structural patterns governing the ordered use of appropriate words and symbols for issuing commands, writing code, etc., in a particular software application or programming language.

In Processing, write a tiff file to a folder called frames.

```
saveFrame("frames/####.tif");
```

what is pseudo-code?

No exact standard.

Idea: use the given structure of a programming language to write the logic in English.

In Processing, write a tiff file to a folder called frames.

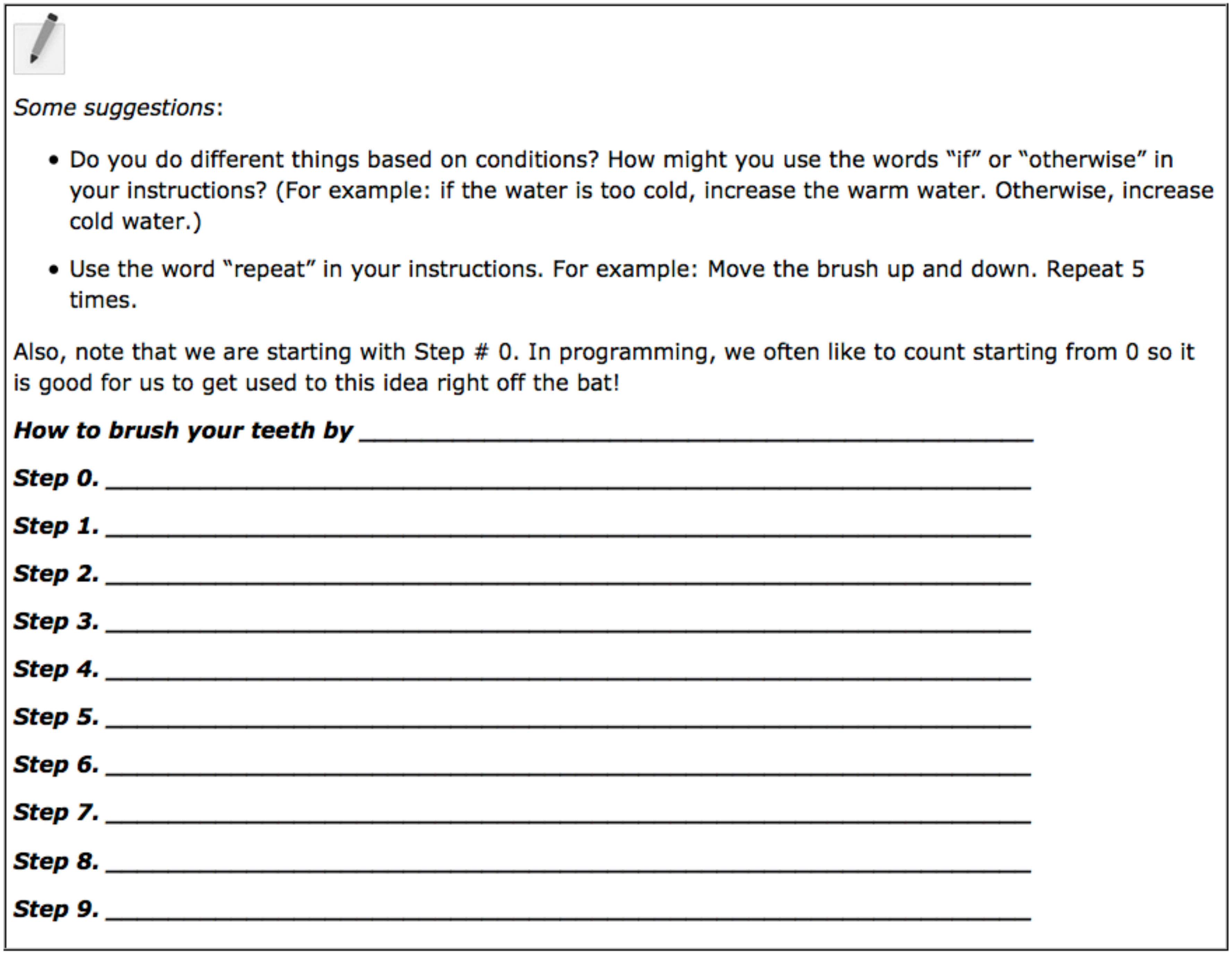
```
saveFrame(“frames/####.tif”);
```

```
saveFrame(to a folder called frames, with the name of the file of 4 digit
sequence.tif);
```

```
save the frame as a 4 digit sequence.tif to the frames folder.
```

>> the more syntax you can remember, the better!

pseudo code algorithms



Some suggestions:

- Do you do different things based on conditions? How might you use the words "if" or "otherwise" in your instructions? (For example: if the water is too cold, increase the warm water. Otherwise, increase cold water.)
- Use the word "repeat" in your instructions. For example: Move the brush up and down. Repeat 5 times.

Also, note that we are starting with Step # 0. In programming, we often like to count starting from 0 so it is good for us to get used to this idea right off the bat!

How to brush your teeth by ____

Step 0. ____

Step 1. ____

Step 2. ____

Step 3. ____

Step 4. ____

Step 5. ____

Step 6. ____

Step 7. ____

Step 8. ____

Step 9. ____

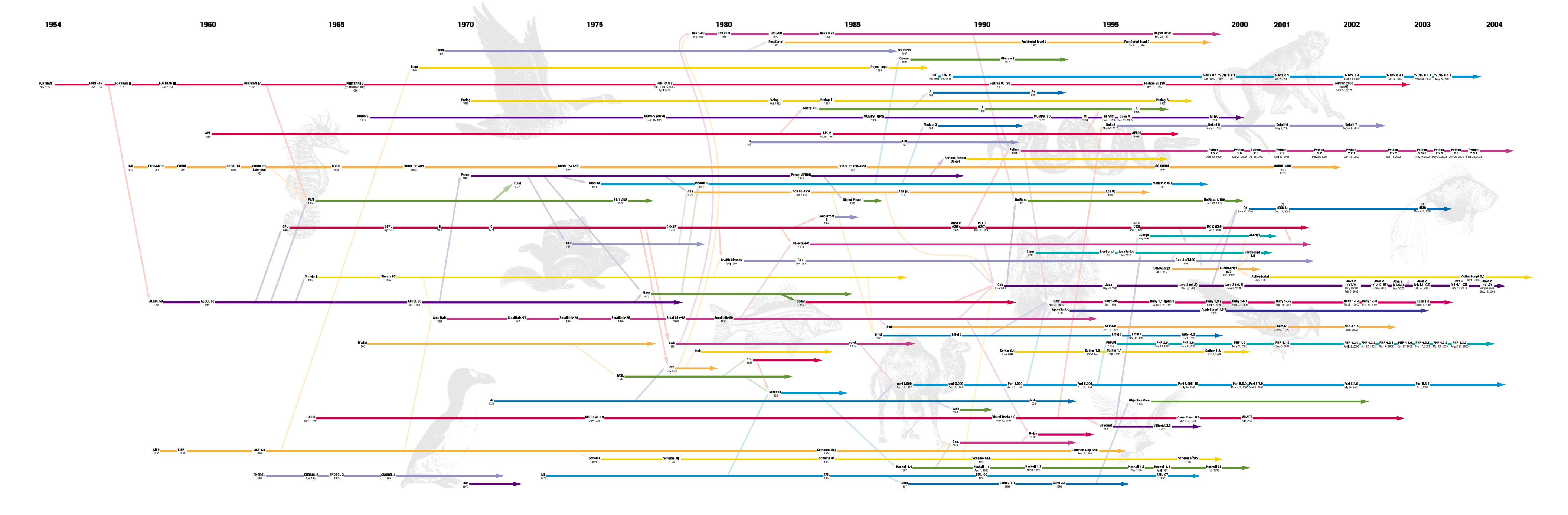
what is a program?

...collection of algorithms

what are some programming languages?

History of Programming Languages

O'REILLY



www.oreilly.com

For more than half of the fifty years computer programmers have been writing code, O'Reilly has provided developers with comprehensive, in-depth technical information. We've kept pace with rapidly changing technologies as new languages have emerged, developed, and matured. Whether you want to learn something new or need answers to tough technical questions, you'll find what you need in O'Reilly books and on the O'Reilly Network.

This timeline includes fifty of the more than 2500 documented programming languages. It is based on an original diagram created by Éric Lévénez (*www.levenez.com*), augmented with suggestions from O'Reilly authors, friends, and conference attendees.

For information and discussion on this poster, go *to www.oreilly.com/go/languageposter*.

what are some programming languages?

HIGH LEVEL (more English like)

Lingo (Director)
Processing (simple environment)
openFrameworks, Cinder
actionscript
ruby
javascript
python
Max/MSP/Jitter (DataFlow environments - box and flow chart based - maybe ore intuitive)

Programming language -
Java (it does everything, and are extendable through libraries)
C/C++/Object C (apple)
Pascal

Assembly language: usually uses 4-bit combination (2^4)=16 combinations

Machine language: 0/1
1. talk to screen
2. screen has address
3. specify pixel location
4. each is 0/1
5. specify which one to turn on

LOW LEVEL (execute very fast)

compiler: translate high level codes to low level language

open source

software is free!

source code is openly available

Processing is created by Casey Reas and Ben Fry under John Maeda’s MIT Media Lab

download: processing.org

It’s a non-profit foundation.

15th anniversary! Hooray and thank you!

processing

Launch Processing!

command+r to run a sketch

```
println("hello world”);
```

Save sketch - note the folder structure!

Transport by folder, not file!

== for folks coming from Processing 2.

shift+command+r to presentation mode

command+t to tidy up!

shift+command+t to tweet

fullScreen(); replaces size(); ==> new in P3

fullScreen(SPAN); ==> span across multiple screens!

pixelDensity(2); ==> finer smoother lines for retina displays :)

also see pixelHeight and pixelWidth, etc in Reference.

option to export to application for self-running (executable),Android (apps) or JavaScript (web)

warning!

Do not rename folders and file - it gets very messy!

error messages

check out sketch_0_2_error_messages folder inside of Class Files on Google drive.

size();

should always have hard numbers, not variables.

good ==> size (640, 320);

not good ==> size (float, float);

size(); should always be the first line code.

code elements

`//` comments...use them :)

`/* */` multiline comments

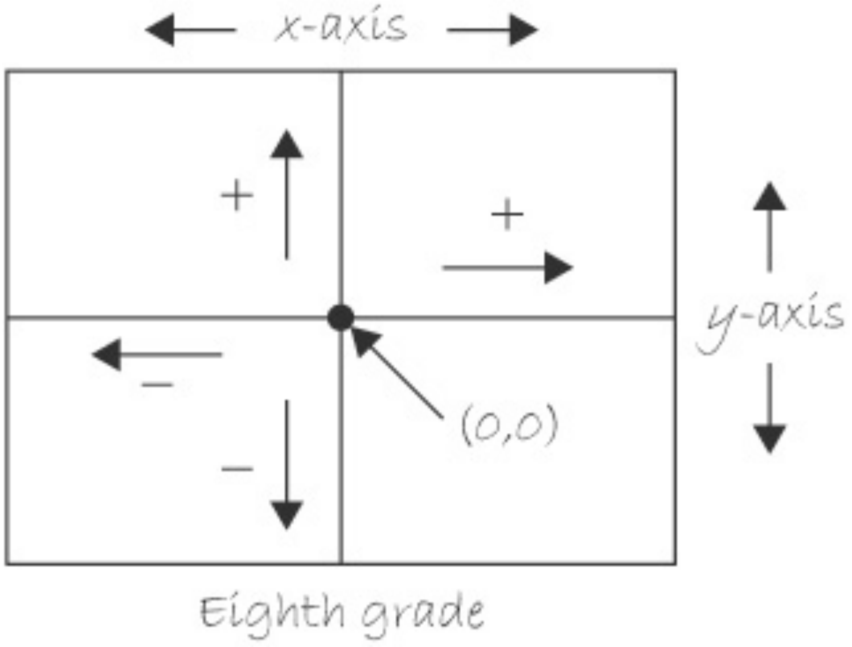
`;` statement terminator

`,` comma, separates parameters or arguments of functions

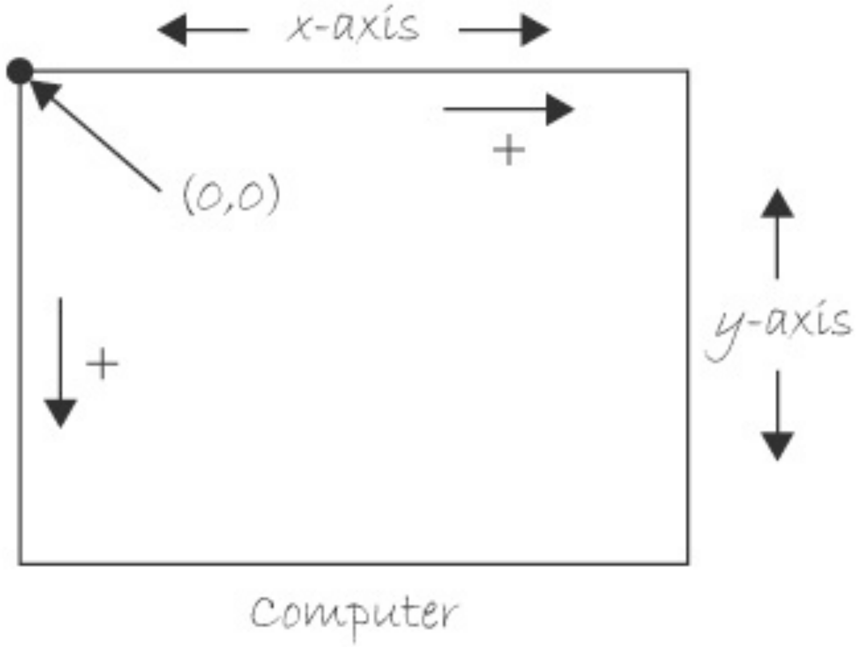
`print()` print to console

`println()` print to console as block

Cartesian Coordinates vs. Computer Coordinates

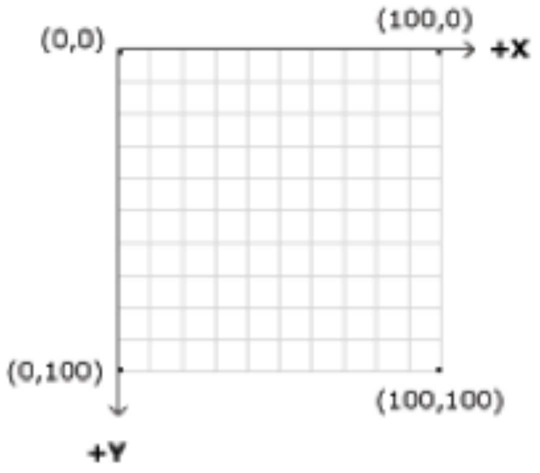


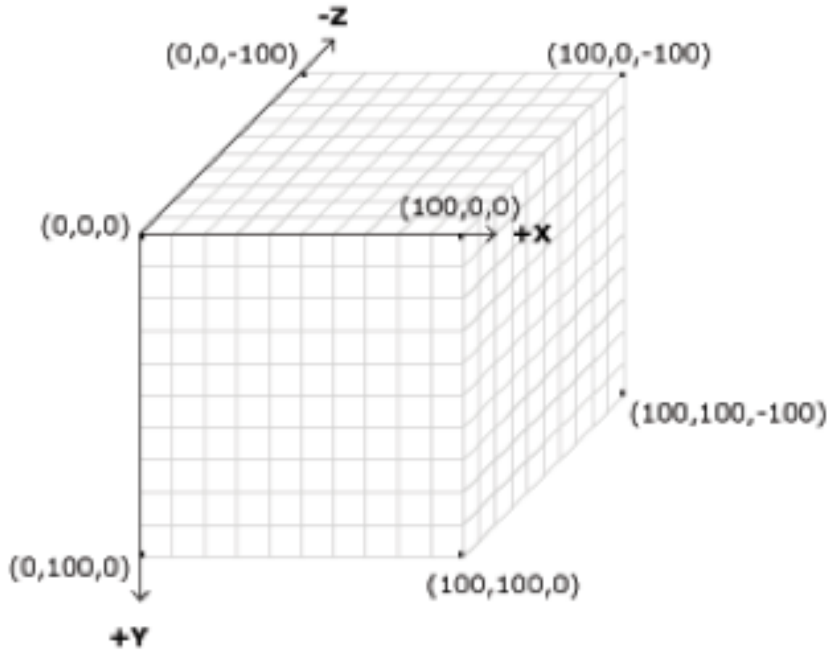
Eighth grade



Computer

coordinates and 3d space





draw

draw a line:

write in pseudo code first: two points define a line, and use a coordinate system to define these points.
Processing syntax: `line(100,50,600,250);`

So, the format is: functionName (_,_,_,_);
functionName followed by parenthesis, followed by arguments (values separated by commas), ending with a parenthesis, and ending with a semicolon.

Functions:
rect
ellipse
line
point

Check out: sketch_1_draw.pde

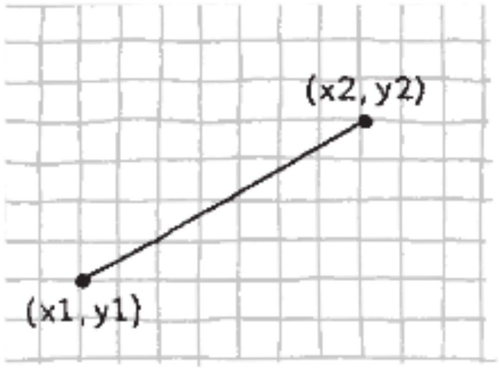
Look up syntax: processing.org, then click on Reference. Command+f.

in class exercise:
Use graph paper to sketch out.
Then try to make a pseudo code.
Then try to code it in Processing.
Don’t go wild on me!

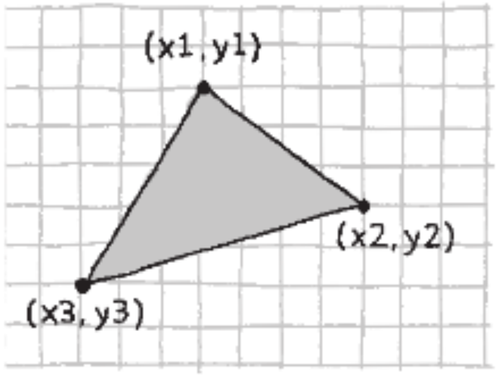
primitive shapes

point, line, triangle, quad, rect, ellipse, arc, bezier

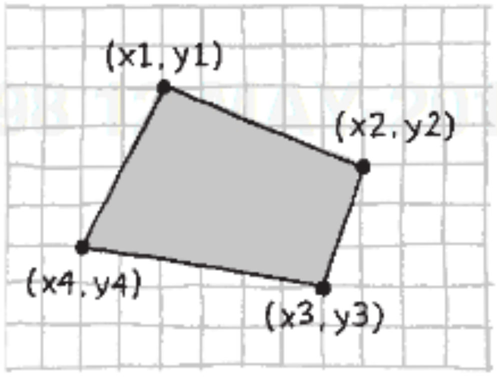
```
point(x,y);
```



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

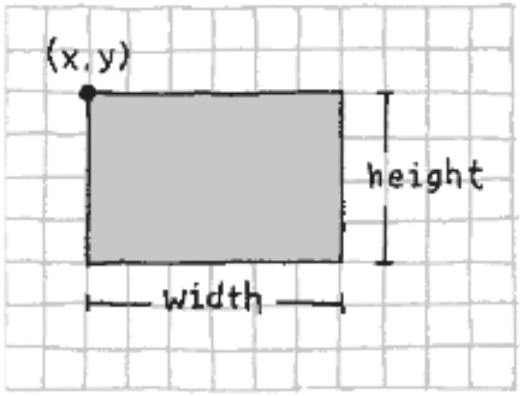


```
triangle(x1, y1, x2, y2, x3, y3)
```

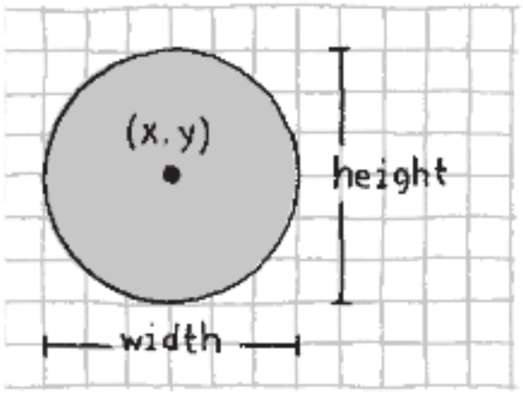


```
quad(x1, y1, x2, y2, x3, y3, x4, y4)
```

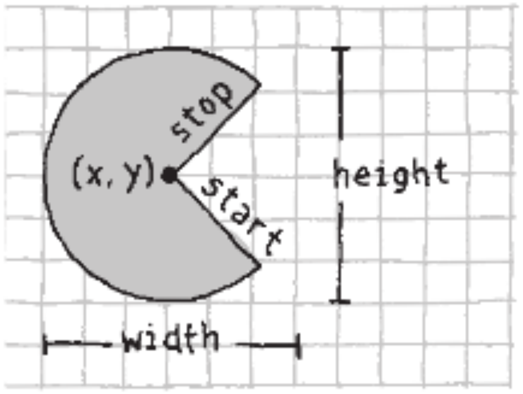
primitive shapes



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



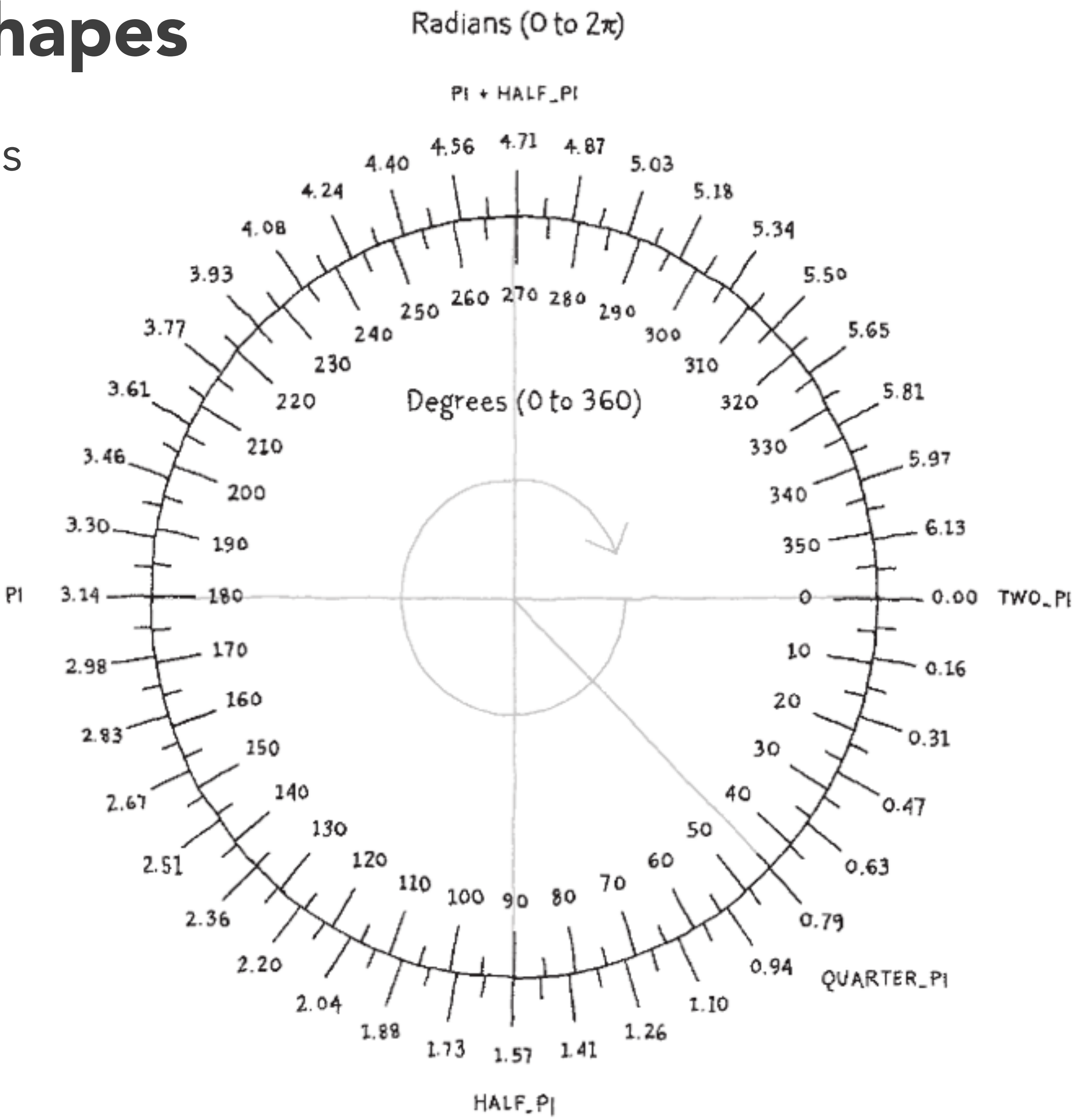
```
ellipse(x, y, width, height)
```



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

primitive shapes

arcs & radians



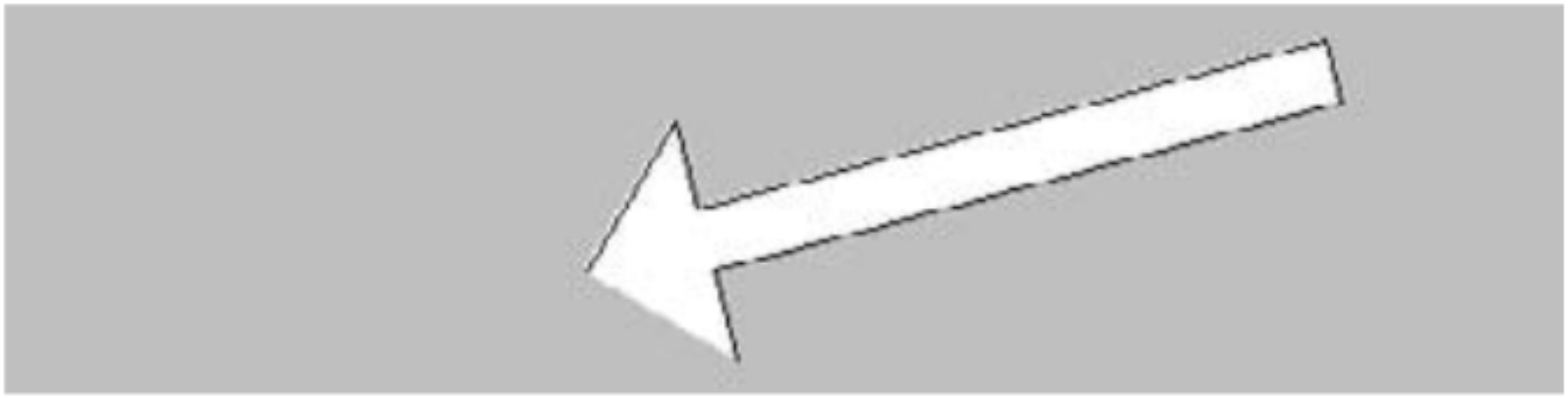
arcs & radians

If you prefer to use degree measurements, you can convert to radians with the *radians()* function. This function takes an angle in degrees and changes it to the corresponding radian value.

```
size(480, 120);
arc(90, 60, 80, 80, 0, radians(90));
arc(190, 60, 80, 80, 0, radians(270));
arc(290, 60, 80, 80, radians(180), radians(450));
arc(390, 60, 80, 80, radians(45), radians(225));
```

complex shapes

The *beginShape()* function signals the start of a new shape. The *vertex()* function is used to define each pair of x- and y-coordinates for the shape. Finally, *endShape()* is called to signal that the shape is finished.



```
size(480, 120);
beginShape();
vertex(180, 82);
vertex(207, 36);
vertex(214, 63);
vertex(407, 11);
vertex(412, 30);
vertex(219, 82);
vertex(226, 109);
endShape();
```

colors

Additive color mixing system: HSB, RGB, hexadecimal

Range - 0-255 (black to white)

Think Illustrator vector shape and lines:
stroke and fill are key words that set colors in Processing.

fill(255, 255, 255); is the same as fill (255);
fill(grayScale,transparency);
fill(R,G,B, alpha)
alpha=0 is completely transparent
alpha=255 is completely opaque

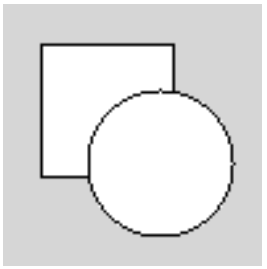
guess, what is background(135);

order of operation

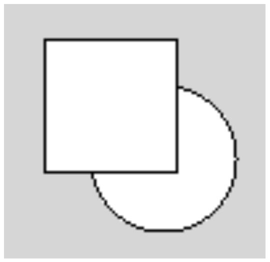
the program is drawn in the order it is written

default: what appears first in the code is the lowest layer

```
rect(15,15,50,50);
ellipse(60,60,55,55);
```



```
ellipse(60,60,55,55);
rect(15,15,50,50);
```



white space

spacing does not matter

this:

```
size


    (      300,
                                         300      )
;
```

is the same as:

```
size(300,300);
```

readability of code

group things together that belong.
put paragraph break to separate them.

```
size(600,400);
background(135);

stroke(255, 0, 0);
fill(100);
rect(100, 100, 200, 20);
```

commenting

Your future self will thank the present self, if you build a habit to comment.

```
// Here is an explanation.

/* Here I have a lot a lot a lot a lot a lot a lot
a lot a lot a lot a lot a lot a lot a lot a lot to say.
*/
```

the flow of program

most sketches use **`setup`** and **`draw`** functions

each is a block of code, indicated with curly brackets

```
void setup() {
  //code here
}

void draw() {
  //code here
}
```

setup is a block of code

setup (stuff that happens once, only once)
It’s like File > New in Photoshop, where we want to setup the size of our canvas and the background color.

```
void setup() {
  size(300,300);
  background(255);
}
```

draw is a block of code

stuff that’s going to happen over and over again

```
void draw() {
  background(0);
  fill(255,0,0);
  rect (100,100,200,200);
}
```

pre-defined functions

Both of these words, setup and draw, are functions that have been pre-defined by Processing as what to do **once** and what to do **over and over again**.

We are further defining these two functions within what we specify inside the curly brackets.

```
void draw() {
  background(0);
  fill(255,0,0);
  rect (100,100,200,200);
}
```

case sensitive

this:

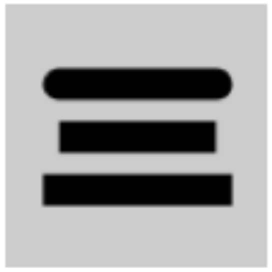
```
size(300,300);
```

is not the same as:

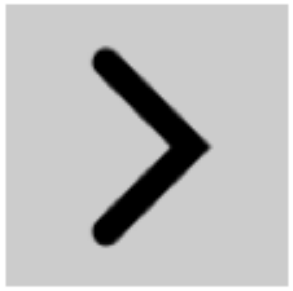
```
Size(300,300);
```

drawing attributes

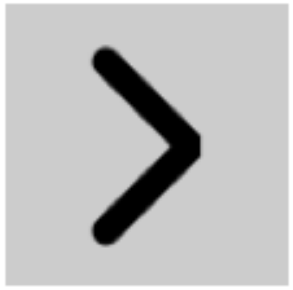
```
smooth(); // adds anti-aliasing

noSmooth(); // turns of anti-aliasing

ellipseMode();

ellipseMode(CORNER); // center is default

rectMode();

rectMode(CORNER); // corner is default

strokeWeight(x); // x is amount in px
```



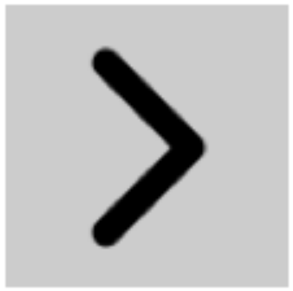
```
strokeWeight(12.0);
strokeCap(ROUND);
line(20, 30, 80, 30);
strokeCap(SQUARE);
line(20, 50, 80, 50);
strokeCap(PROJECT);
line(20, 70, 80, 70);
```



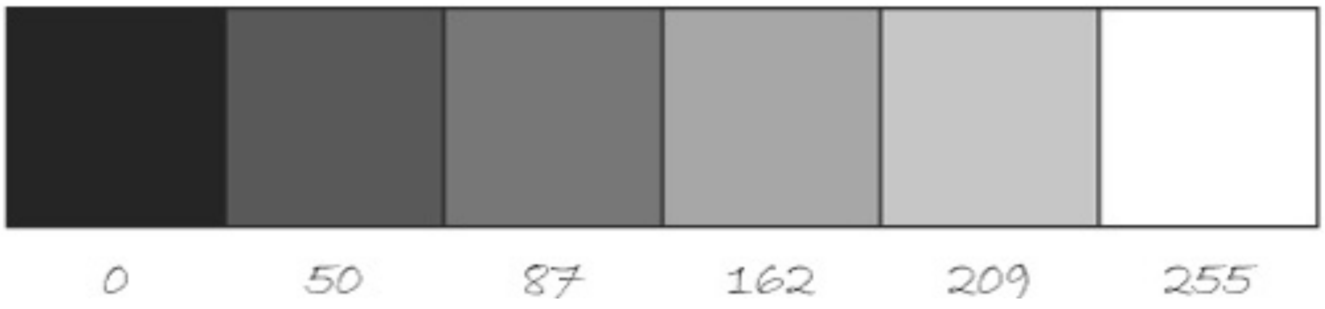
```
noFill();
strokeWeight(10.0);
strokeJoin(MITER);
beginShape();
vertex(35, 20);
vertex(65, 50);
vertex(35, 80);
endShape();
```



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



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



specifying color syntax

```
fill(rgb)
fill(rgb, alpha)
fill(gray)
fill(gray, alpha)
fill(v1, v2, v3)
fill(v1, v2, v3, alpha)

Parameters
rgb  int: color variable or hex value
alpha    float: opacity of the fill
gray float: number specifying value between white and black
v1   float: red or hue value (depending on current color mode)
v2   float: green or saturation value (depending on current color mode)
v3   float: blue or brightness value (depending on current color mode)
```

Tools > Color Selector or http://colorschemedesigner.com or https://kuler.adobe.com

color attributes

```
stroke(); // stroke color

noStroke(); // turns off stroke

fill(); // fill color

noFill(); // turns off fill

background(); // background color

colorMode(); // accepts RGB or HSB
```

once an attribute is set, it remains active until set again

truly

Program NOT working is normal!
It’s the rite of passage.

Ex 1 Draw it!

check it out!

paper sketches (due next class, at the start of class)