What's In It For Me?

More On Processing …

Lawrence Snyder
University of Washington, Seattle

Programming, So Far …

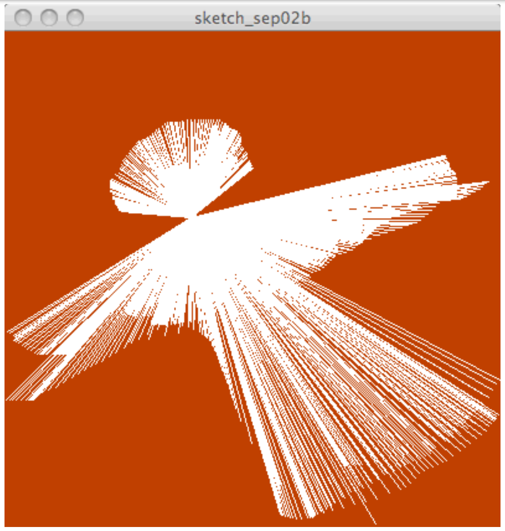
- From the Lightbot 2.0, we've learned …
  - Programming is giving instructions to an agent so he, she, it can perform the operations themselves
  - Instructions are given in sequence
  - Instructions are followed in sequence
  - Instructions come from a limited repertoire
  - Programs mix instructions and function calls
  - Programs are written first, and run later (without intervention of the programmer)
  - The last point means, programming = planning

Processing, So Far ...

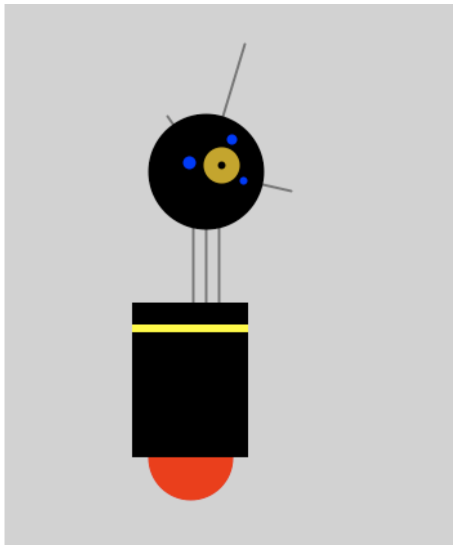
- We have seen that Processing is ...
  - A programming language that does graphics
  - That means, there are many functions for shapes
  - We give the positions of shapes in pixel units
  - To be successful, we must give
    - The window (a/k/a canvas) size, as in size(400,400);
    - The background color, as in background(0, 200, 0);
    - Correct punctuation, like matching parens, semicolons
  - The Processing IDE (interactive development environment) is easy to use

And, We've Modified Some Code

- An Angel



- A Robot



We've also located the Reference page, where all known facts about Processing are stored & accessible to help us create a program

So, What Else?

- There's plenty to learn, but we only learn what we need to know – it's a standard computing idea:

> Computing is ALL about getting the details right. That's true for most things, of course, but it is especially true when you're programming computers.

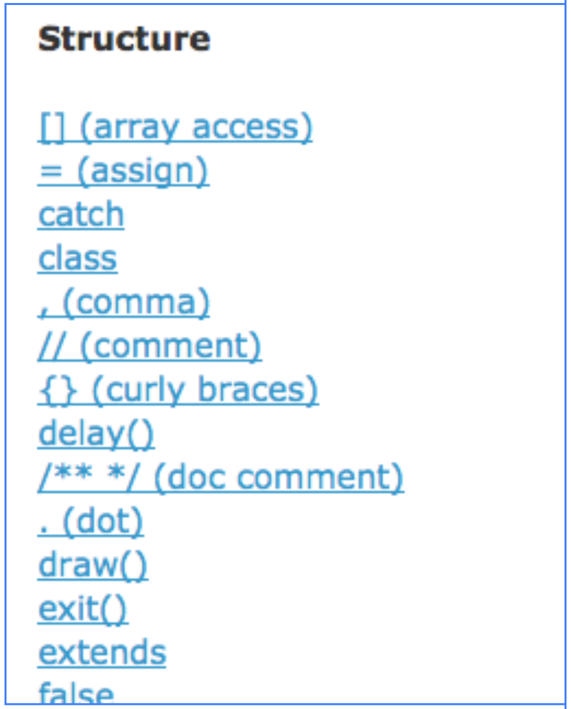
- Programmers don't remember all the detail – they either look it up, or they experiment to figure it out … you should, too.
- Instead we focus on the "idea," not specifics

Today …

- We introduce two important ideas … you're familiar with them both:
  - Variables
  - Functions

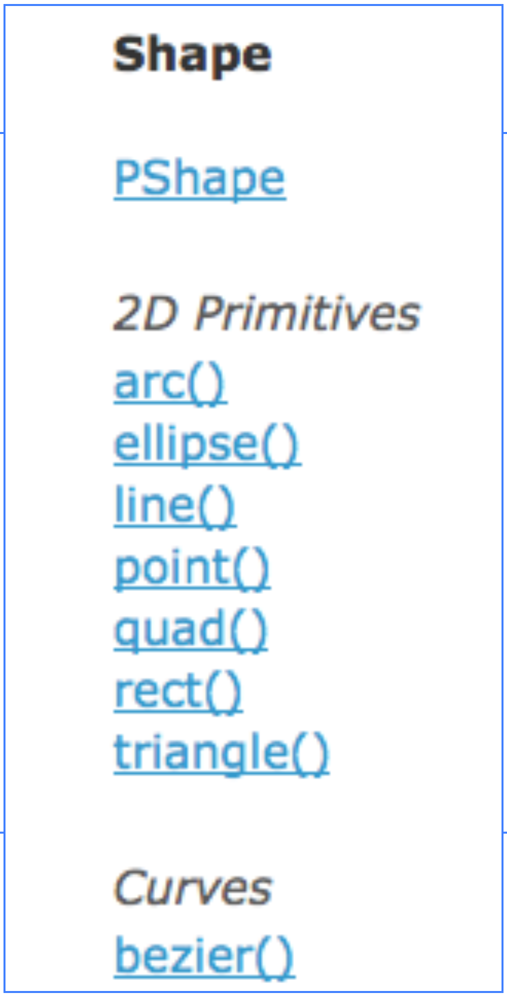
Checking Out The References …

- Need an explanation? The Ref Page's got it!



Structure

[] (array access)
= (assign)
catch
class
, (comma)
// (comment)
{} (curly braces)
delay()
/** */ (doc comment)
. (dot)
draw()
exit()
extends
false

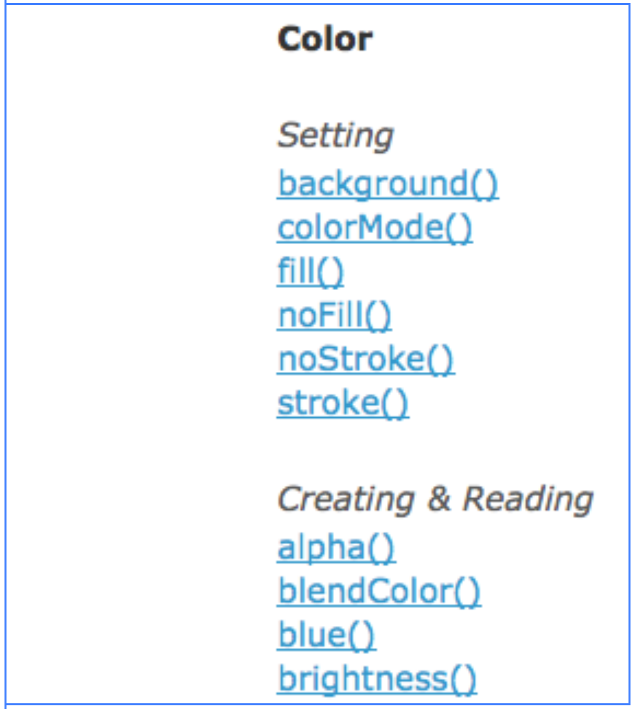


Shape

PShape

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

Curves
bezier()



Color

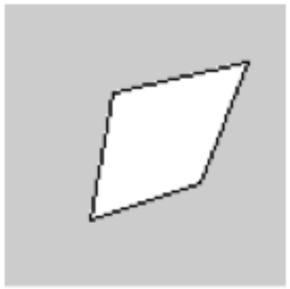
Setting
background()
colorMode()
fill()
noFill()
noStroke()
stroke()

Creating & Reading
alpha()
blendColor()
blue()
brightness()

Looking At The Quad

Name quad()

Examples



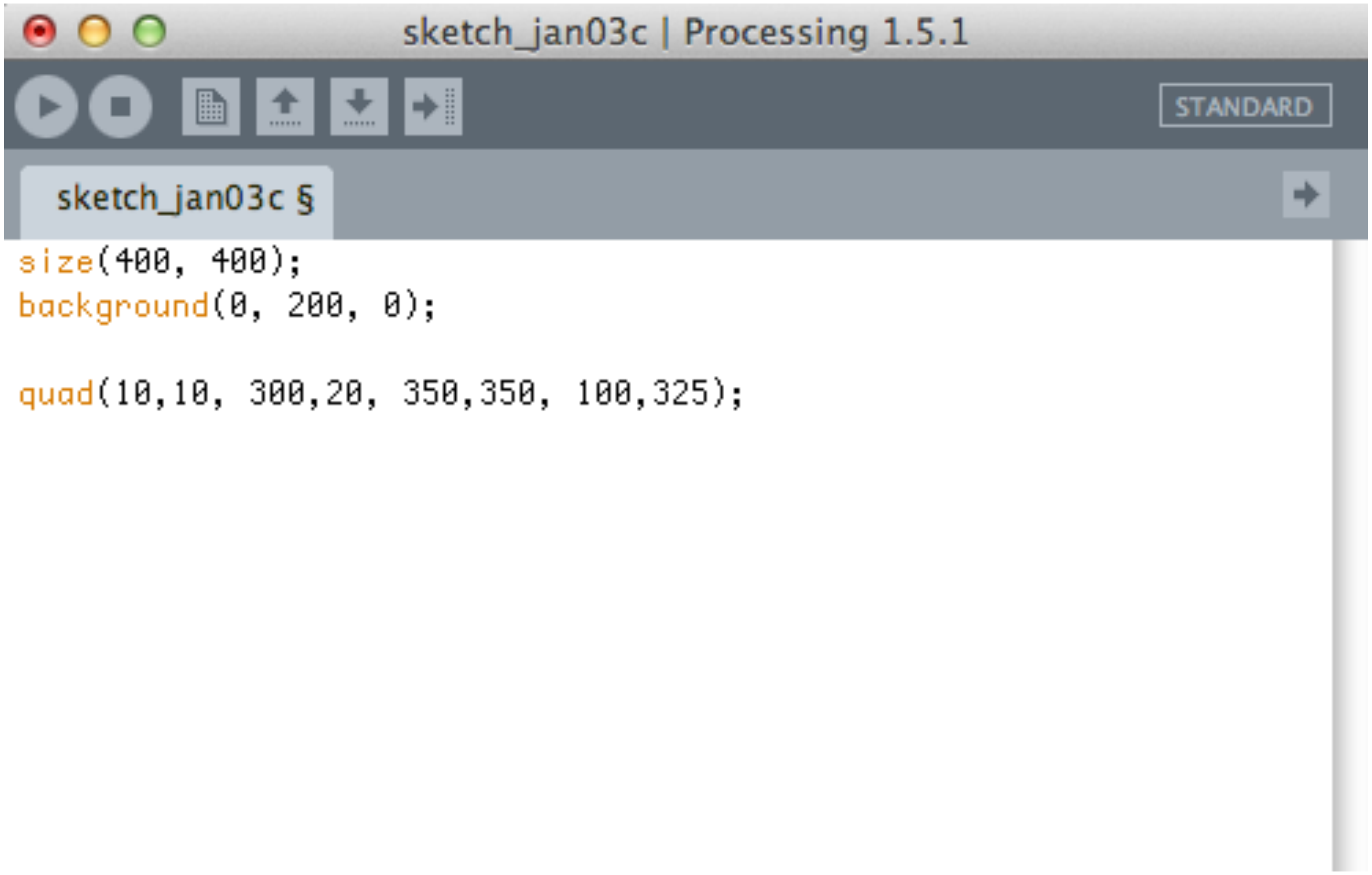
```
quad(38, 31, 86, 20, 69, 63, 30, 76);
```

Description A quad is a quadrilateral, a four sided polygon. It is similar to a rectangle, but the angles between its edges are not constrained to ninety degrees. The first pair of parameters (x1,y1) sets the first vertex and the subsequent pairs should proceed clockwise or counter-clockwise around the defined shape.

Syntax

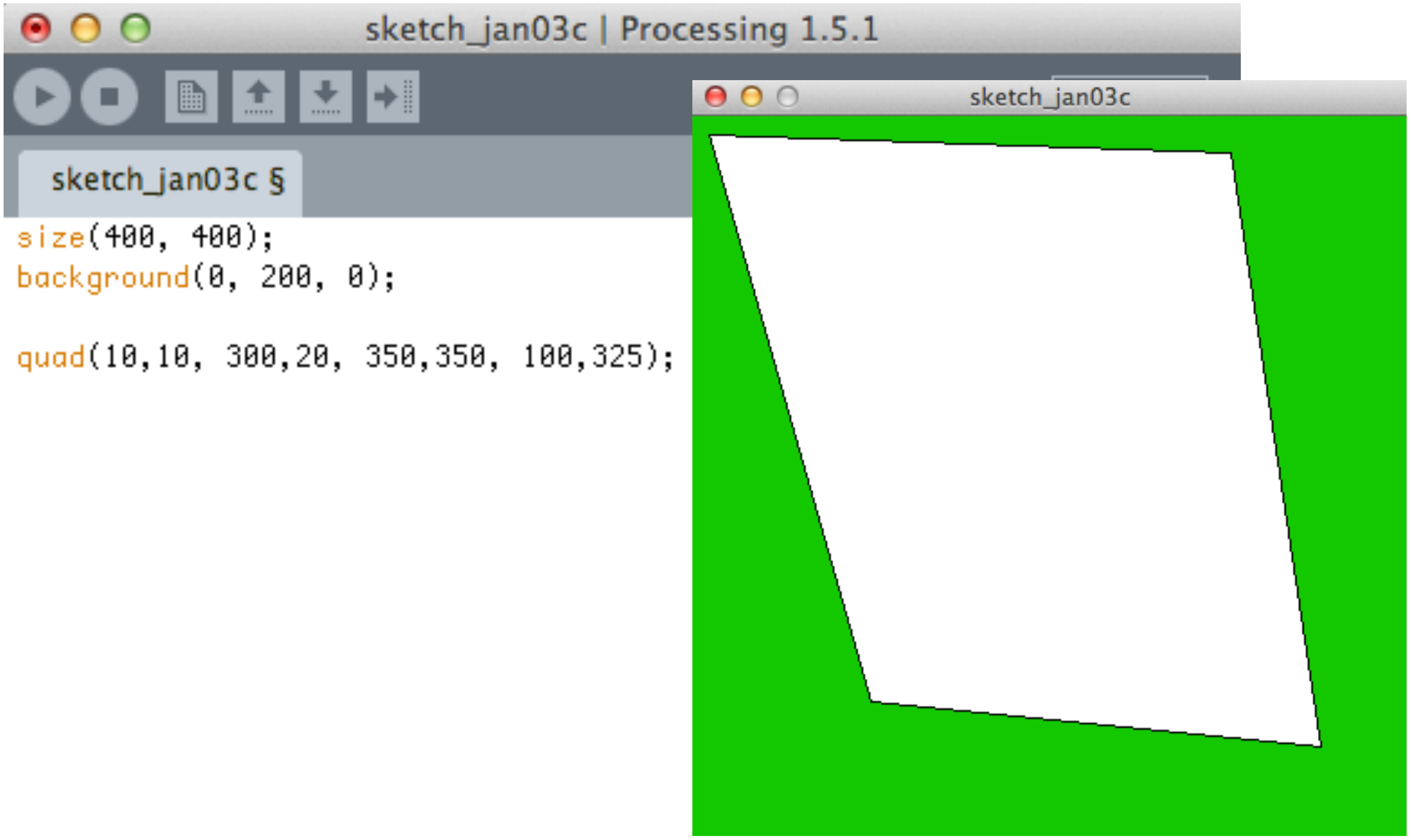
```
quad(x1, y1, x2, y2, x3, y3, x4, y4)
```

So, We Try It Out



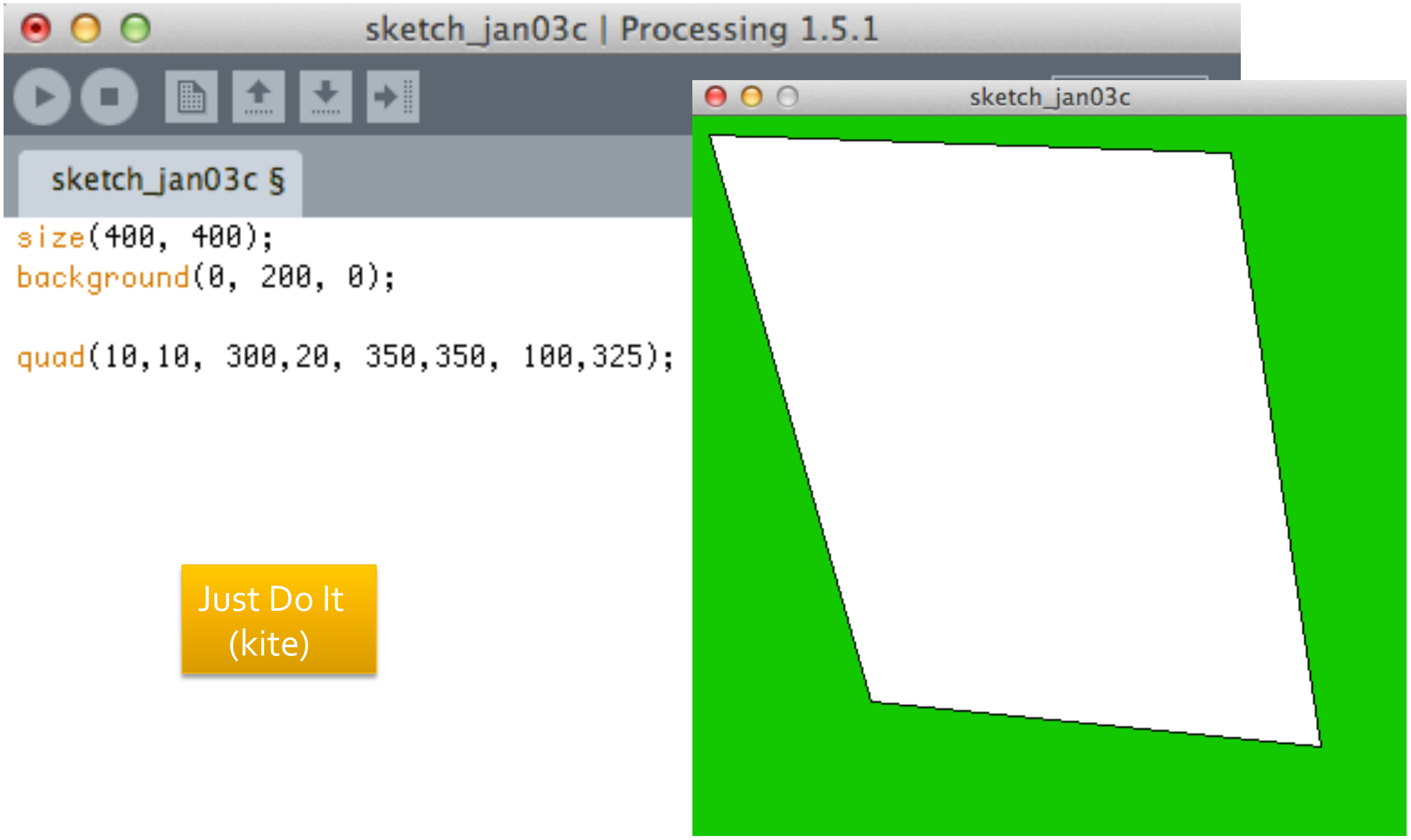
```
size(400, 400);
background(0, 200, 0);

quad(10,10, 300,20, 350,350, 100,325);
```

So, We Try It Out



```
size(400, 400);
background(0, 200, 0);

quad(10,10, 300,20, 350,350, 100,325);
```

So, We Try It Out

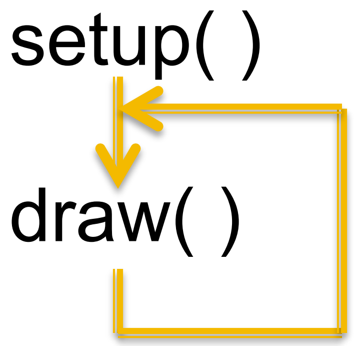


```
size(400, 400);
background(0, 200, 0);

quad(10,10, 300,20, 350,350, 100,325);
```

Just Do It
(kite)

Recall setup( ) and draw( )

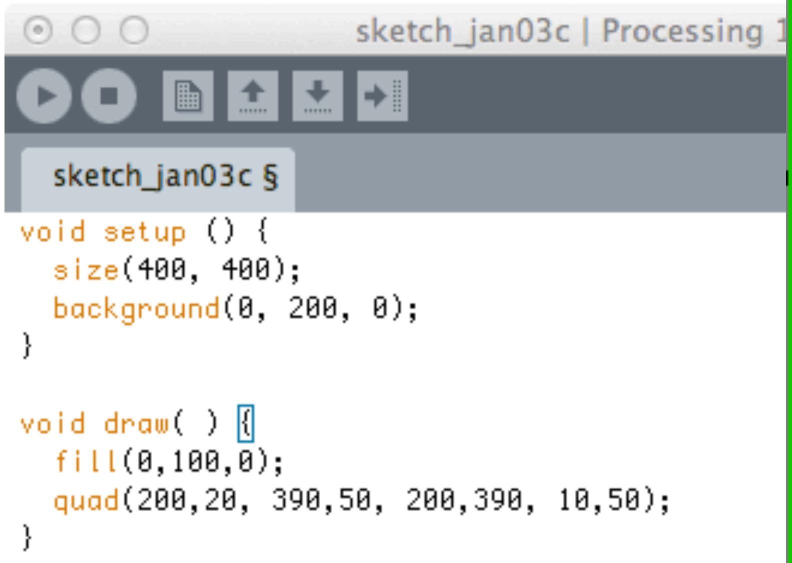
- The functions setup( ) and draw( ) allow the Processing computations to be dynamic
- Recall that they work as follows:



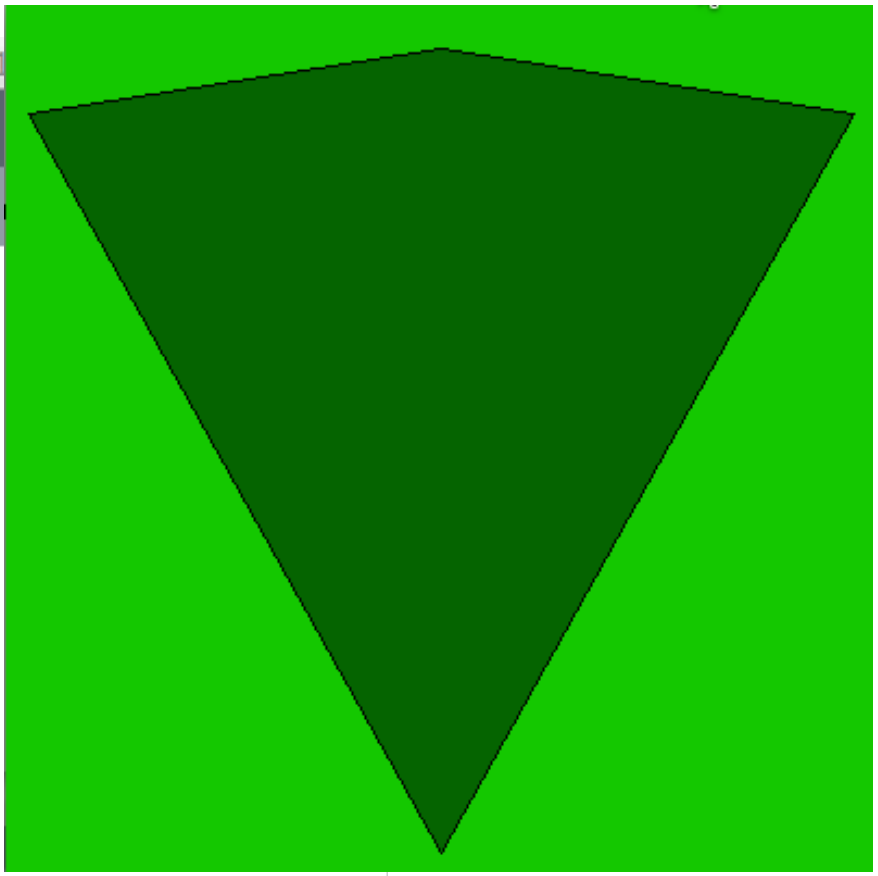
- Make the Kite dynamic

Dynamic Kite

… but it doesn't move!



```
void setup () {
  size(400, 400);
  background(0, 200, 0);
}

void draw( ) {
  fill(0,100,0);
  quad(200,20, 390,50, 200,390, 10,50);
}
```



Introduce A Variable

- Variables are a key computer idea
- They look like "unknowns" in math, but they are (really) very different … more on that later
- Variables must be declared, as in

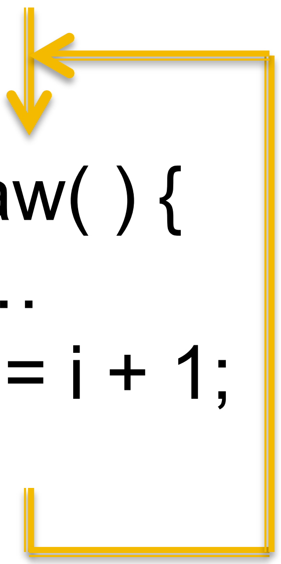
```
int i=0;   //declare i, an integer, and set value
```

- Once declared, a variable can be used as if it's a number, like in pixel positions

Using Variables w/ setup( ) draw( )

- Here's a strategy to use a variable i …

```
int i = 0;
setup( ) {
   . . .
}


draw( ) {
   . . .
   i = i + 1;
}
```



Using Variables w/ setup( ) & draw( )

- Here's a strategy to a variable i …

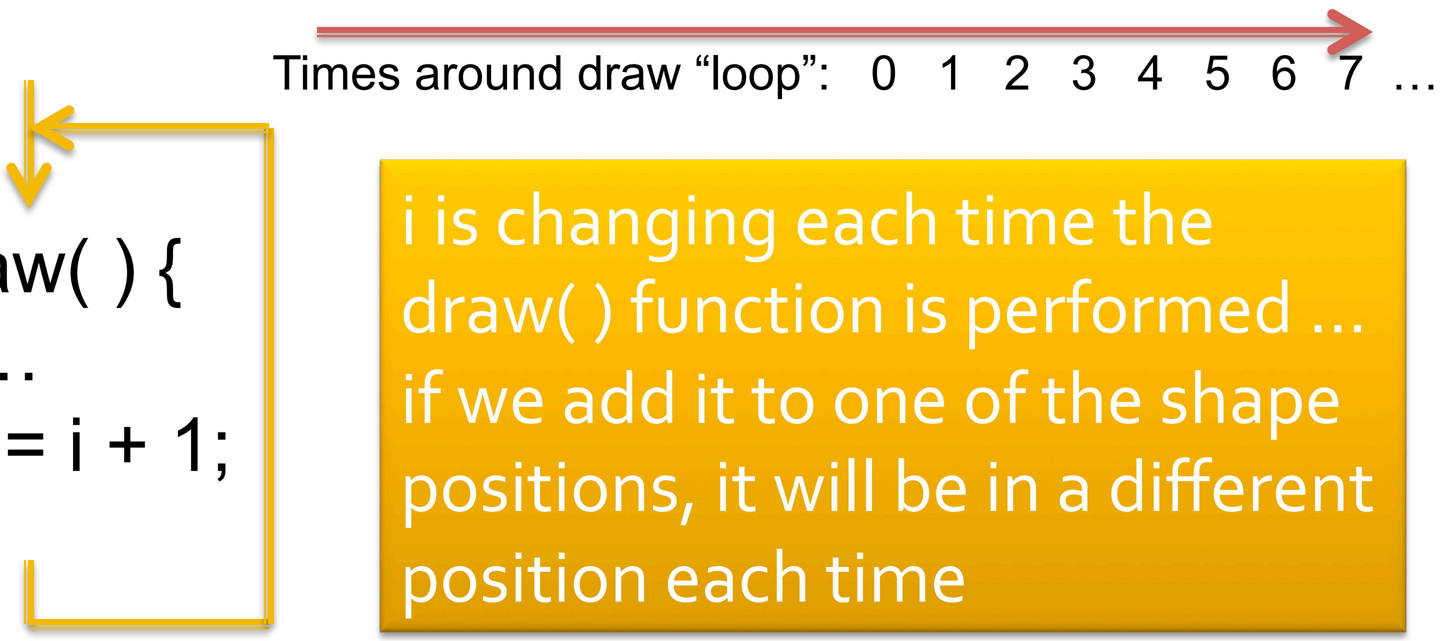
```
int i = 0;
setup( ) {
   . . .
}


draw( ) {
   . . .
   i = i + 1;
}
```

```
                      Value of i : 0   1   2   3   4   5   6   7  …
Times around draw "loop":   0   1   2   3   4   5   6   7  …
```

Using Variables w/ setup( ) & draw( )

- Here's a strategy to a variable i …

```
int i = 0;
setup( ) {
   …
}

draw( ) {
   …
   i = i + 1;
}
```


Value of i : 0 1 2 3 4 5 6 7 …
Times around draw "loop": 0 1 2 3 4 5 6 7 …

i is changing each time the draw( ) function is performed … if we add it to one of the shape positions, it will be in a different position each time

Using Variables w/ setup( ) & draw( )

- Here's a strategy to a variable i ...

```
int i = 0;
setup( ) {
   ...
}

draw( ) {
   ...
   i = i + 1;
}
```

Value of i : 0 1 2 3 4 5 6 7 ...

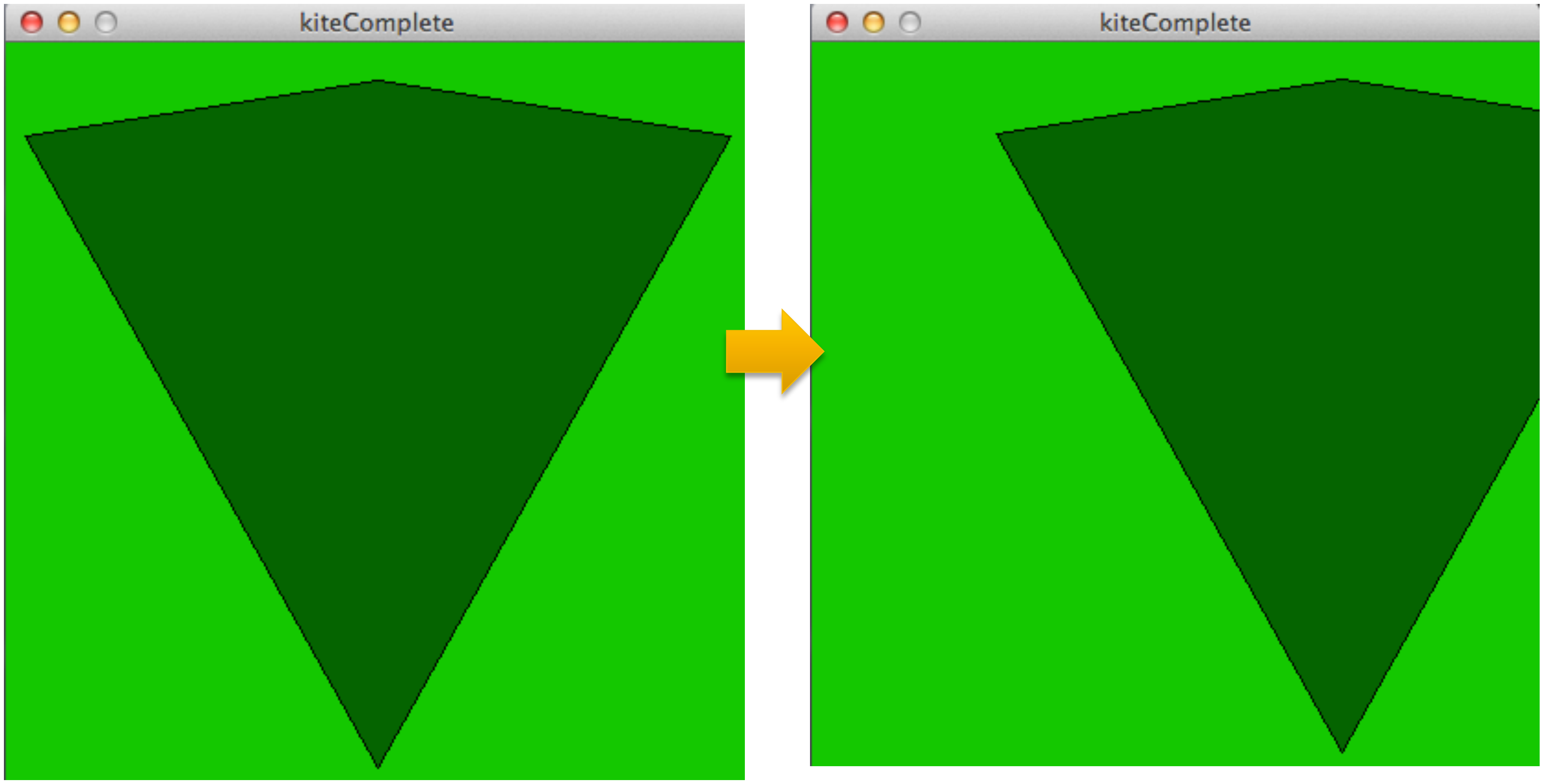
Times around draw "loop": 0 1 2 3 4 5 6 7 ...

i is changing each time the draw( ) function is performed ... if we add it to one of the shape positions, it will be in a different position each time

Just Do It

Kite Moves Right

- Each redrawing positions kite one more pixel R

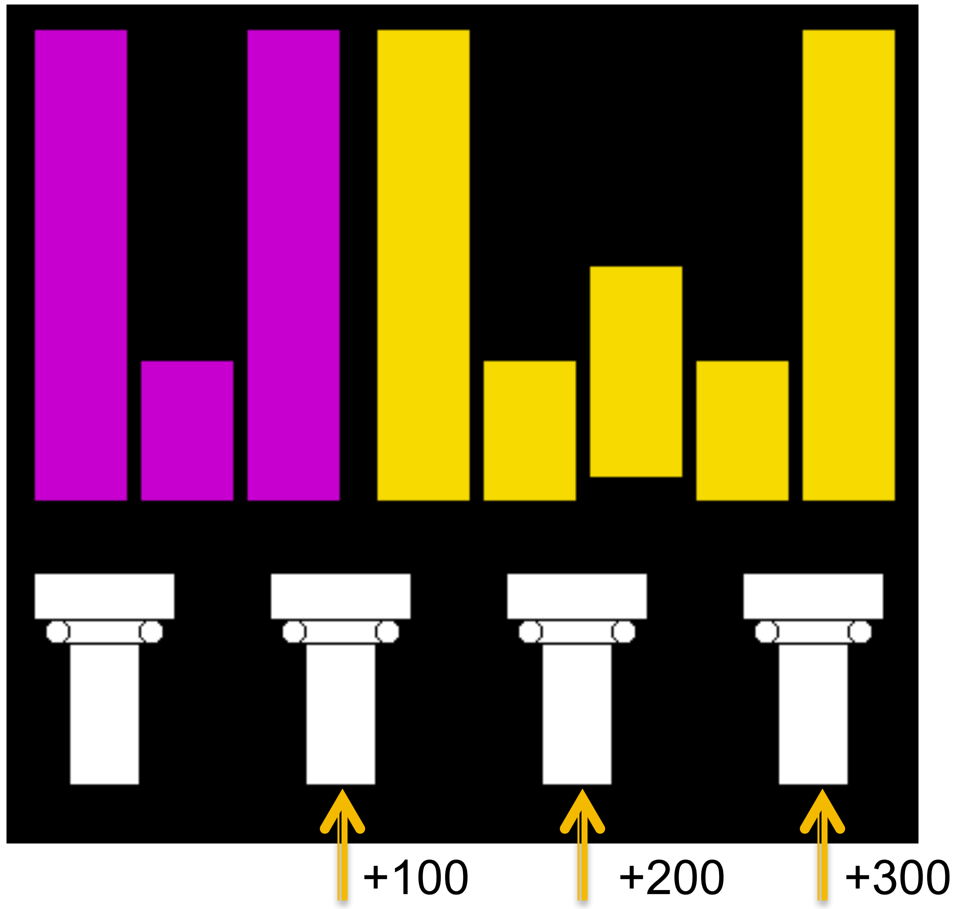


Recall The "Logo Lab"

- Yesterday in lab you drew UW logos in Processing

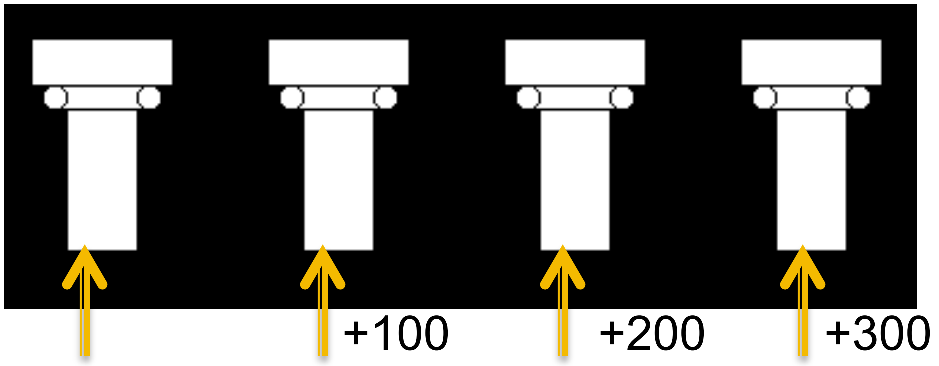
These columns were created as follows:

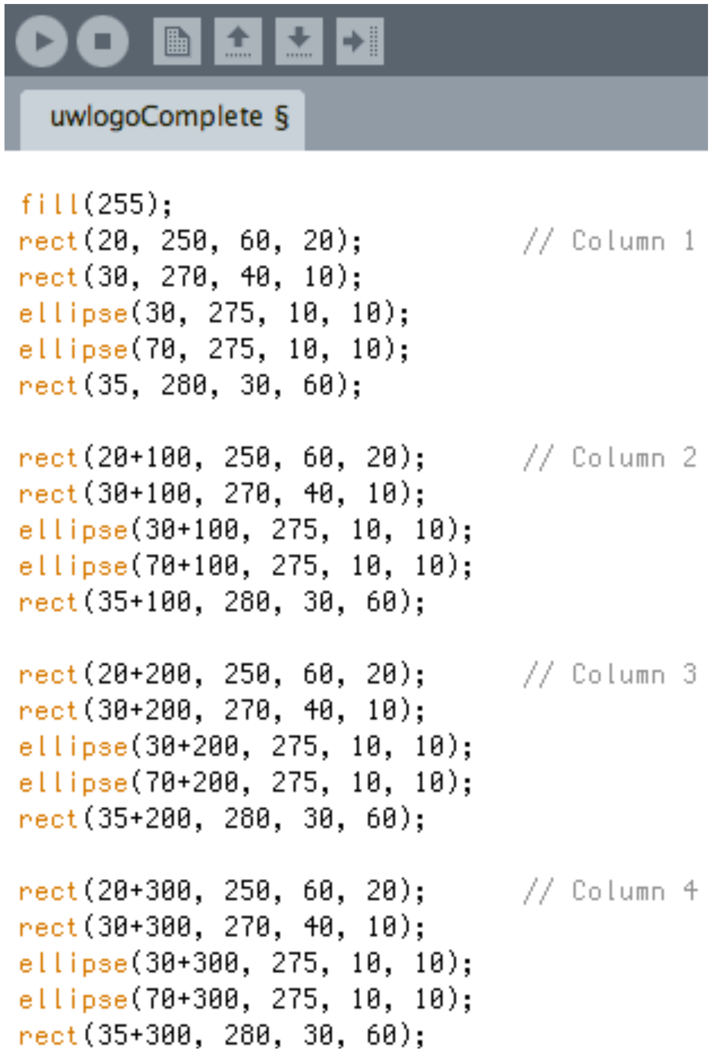
1) Draw the first column
2) Copy and paste 3 more instances into the program
3) Change the x-coordinate of each copy by an amount (100, 200, 300) to draw them in different (and proper) positions



The Program Code

- A column is a “concept” …

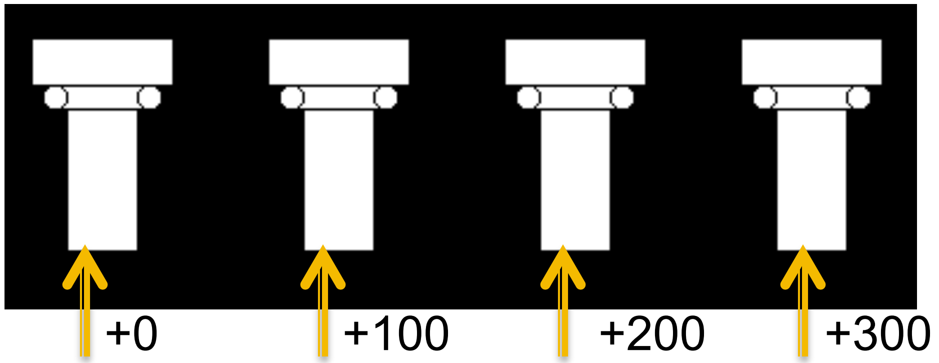




```
fill(255);
rect(20, 250, 60, 20);           // Column 1
rect(30, 270, 40, 10);
ellipse(30, 275, 10, 10);
ellipse(70, 275, 10, 10);
rect(35, 280, 30, 60);

rect(20+100, 250, 60, 20);       // Column 2
rect(30+100, 270, 40, 10);
ellipse(30+100, 275, 10, 10);
ellipse(70+100, 275, 10, 10);
rect(35+100, 280, 30, 60);

rect(20+200, 250, 60, 20);       // Column 3
rect(30+200, 270, 40, 10);
ellipse(30+200, 275, 10, 10);
ellipse(70+200, 275, 10, 10);
rect(35+200, 280, 30, 60);

rect(20+300, 250, 60, 20);       // Column 4
rect(30+300, 270, 40, 10);
ellipse(30+300, 275, 10, 10);
ellipse(70+300, 275, 10, 10);
rect(35+300, 280, 30, 60);
```

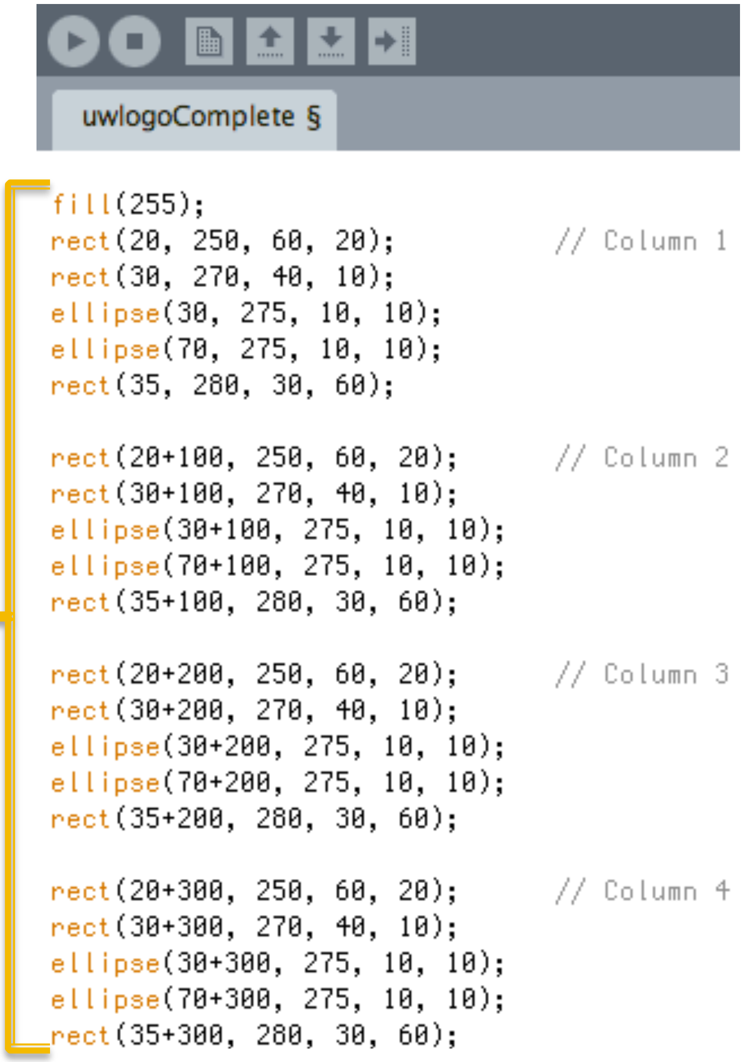
The Program Code

- A column is a "concept" …



- Abstract! Think "function!"

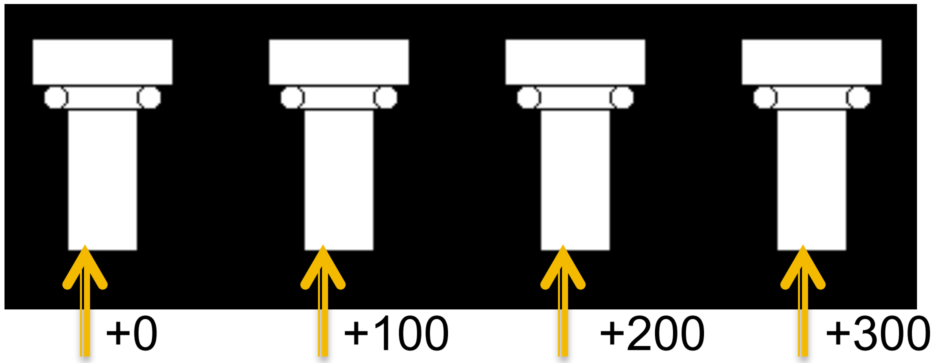
```
fill(255);
draw_column(   0 ); // Col 1
draw_column( 100 ); // Col 2
draw_column( 200 ); // Col 3
draw_column( 300 ); // Col 4
```



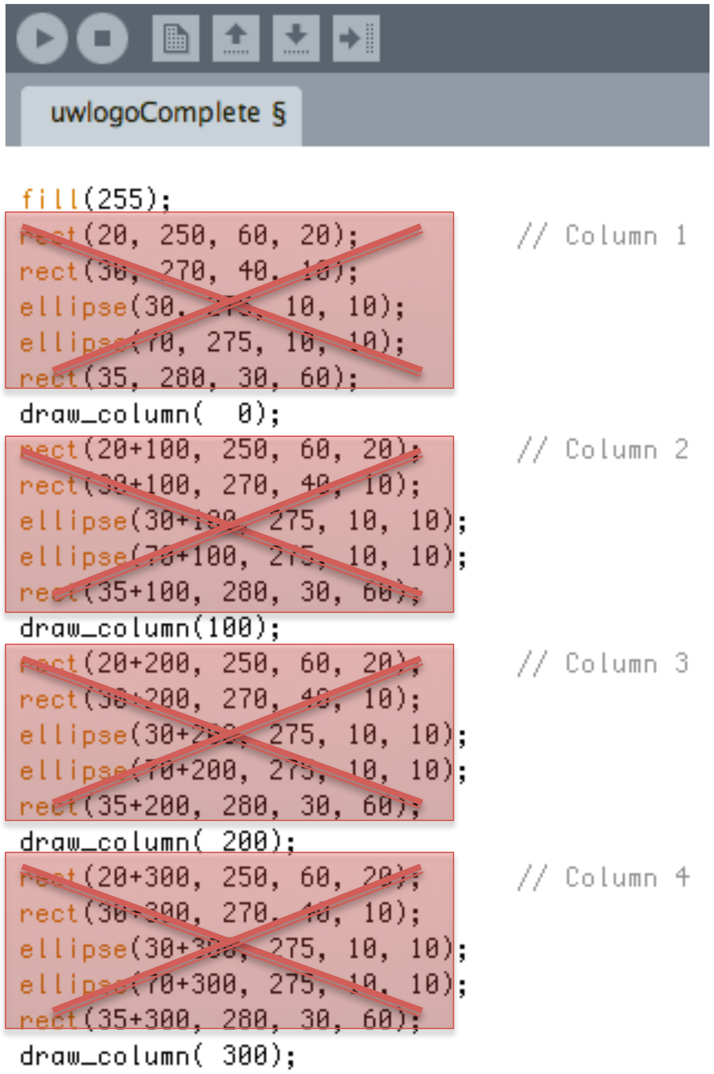
```
fill(255);
rect(20, 250, 60, 20);              // Column 1
rect(30, 270, 40, 10);
ellipse(30, 275, 10, 10);
ellipse(70, 275, 10, 10);
rect(35, 280, 30, 60);

rect(20+100, 250, 60, 20);          // Column 2
rect(30+100, 270, 40, 10);
ellipse(30+100, 275, 10, 10);
ellipse(70+100, 275, 10, 10);
rect(35+100, 280, 30, 60);

rect(20+200, 250, 60, 20);          // Column 3
rect(30+200, 270, 40, 10);
ellipse(30+200, 275, 10, 10);
ellipse(70+200, 275, 10, 10);
rect(35+200, 280, 30, 60);

rect(20+300, 250, 60, 20);          // Column 4
rect(30+300, 270, 40, 10);
ellipse(30+300, 275, 10, 10);
ellipse(70+300, 275, 10, 10);
rect(35+300, 280, 30, 60);
```

The draw_column( ) Function

- Code that draws a column becomes the function … "package" it (below) and change the "hundreds" to a **variable**, x

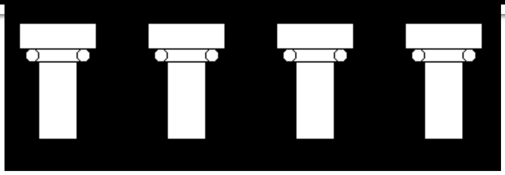


```
void draw_column (int x ) {
     rect(20+x, 250, 60, 20);          // Col
     rect(30+x, 270, 40, 10);
     ellipse(30+x, 275, 10, 10);
     ellipse(70+x, 275, 10, 10);
     rect(35+x, 280, 30, 60);
}
```



```
fill(255);
rect(20, 250, 60, 20);                    // Column 1
rect(30, 270, 40, 10);
ellipse(30, 275, 10, 10);
ellipse(70, 275, 10, 10);
rect(35, 280, 30, 60);
draw_column(  0);
rect(20+100, 250, 60, 20);                // Column 2
rect(30+100, 270, 40, 10);
ellipse(30+100, 275, 10, 10);
ellipse(70+100, 275, 10, 10);
rect(35+100, 280, 30, 60);
draw_column(100);
rect(20+200, 250, 60, 20);                // Column 3
rect(30+200, 270, 40, 10);
ellipse(30+200, 275, 10, 10);
ellipse(70+200, 275, 10, 10);
rect(35+200, 280, 30, 60);
draw_column( 200);
rect(20+300, 250, 60, 20);                // Column 4
rect(30+300, 270, 40, 10);
ellipse(30+300, 275, 10, 10);
ellipse(70+300, 275, 10, 10);
rect(35+300, 280, 30, 60);
draw_column( 300);
```

The draw_column( ) Function



- Like Lightbot,
  - The function *declaration* defines the function
  - The function *call* runs the function
  - Both parts are needed

```
void draw_column (int x ) {
    rect(20+x, 250, 60, 20);        // Column
    rect(30+x, 270, 40, 10);
    ellipse(30+x, 275, 10, 10);
    ellipse(70+x, 275, 10, 10);
    rect(35+x, 280, 30, 60);
}
```

Function Declaration

```
fill(255);
draw_column(   0);
draw_column( 100);
draw_column( 200);
draw_column( 300);
```

4 Function Calls

- More on functions later …

Summary

- The "idea" in this lecture is a *variable*
- It's a name, like x or i or radius, that has a value
- Variables must be declared – int and float are two kinds of variables
- The best, meaning the most powerful, part is variables can be changed … *they vary*!
- We change them by giving them a new value
- i = i + 1   says, give i its old value plus 1
- We use variables just like we'd use numbers, say in pixel positions or function parameters