Announcements

- No Class Monday; Lab on Tuesday as usual

Datatypes and Variables

Lawrence Snyder
University of Washington, Seattle

Today's Goals

- We have three basic ideas to cover –
  - Datatypes
  - Declarations
  - Variables
- They all interact … we'll just start on these ideas today

Ninja! Example for Discussion

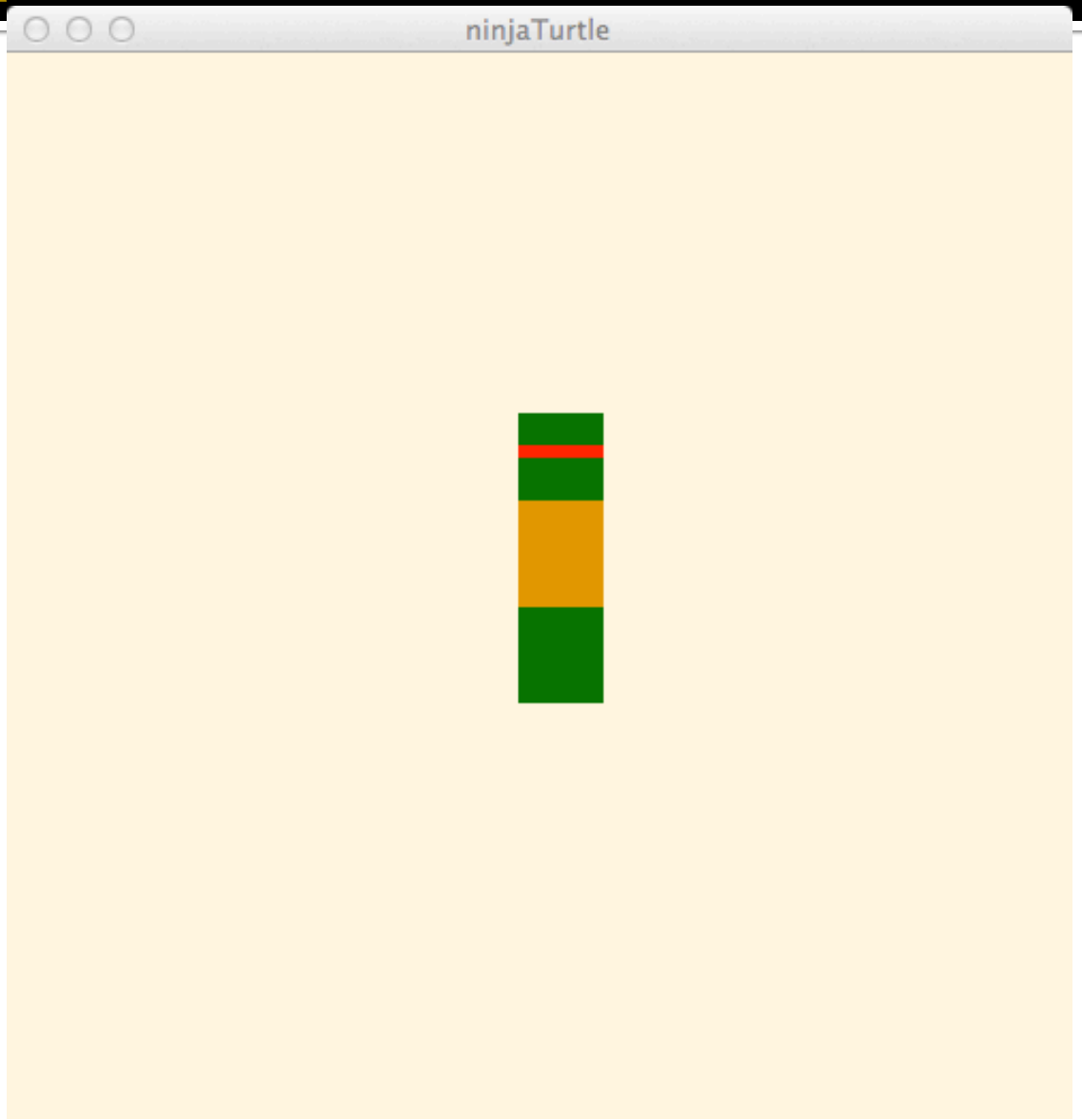
sketch_130114c §

```
void setup( ) {
  size(500,500);
  noStroke();
}

void draw() {
  background(255, 245, 220);
  raff( );
}

void raff( ) {
  fill(0,100,0);
  rect(240,260, 40, 45);
  fill(219,136,0);
  rect(240,210, 40, 50);
  fill(0,100,0);
  rect(240,190, 40, 20);
  fill(255,0,0);
  rect(240, 184, 40, 6);
  fill(0,100,0);
  rect(240, 169, 40, 15);

}
```



Variables

- **variables** are names used in a program for quantities that vary … get it? Variables vary!
- So, one thing we can do is give them values:
  - `x = 12;`

  x is the variable, and it's being assigned the value 12
- Now, whenever I use the variable x, as in
  - `y = x + 1;`

it is as if I had used its value (12) directly: y=12+1

- It's pretty obvious … but there's more to it

Caution: variables are NOT unknowns

Datatypes

- The data that variables name has certain properties … we group information with similar properties into "types" --
  - integers, or whole numbers
  - floating point, usually called decimal numbers
  - colors, a triple of numbers for R, G and B
  - Etc.

Give Datatypes in Declarations

- Processing has a series of **datatypes**
- The most important datatypes for us are

  int, float, boolean and color

… we add more later

  - Find details in the references

Tell Processing About Your Values

- Processing (and all languages) need to know the types of data you are working with
- We tell them the type by **declaring** a variable's datatype
- When declaring variables we list them after the type, as in
  - `int x, y, z;`
  - `float half_step = 0.5, whole = 1.0;`
  - `color yellow = color(200,200,0);`

Declaration & Variable Rules

- Variables are case sensitive

```
int leftSide, left_side, leftside; // declare 3 vars
```

- Variables can be initialized

```
float temperature = 98.6;     // declare & initialize
```

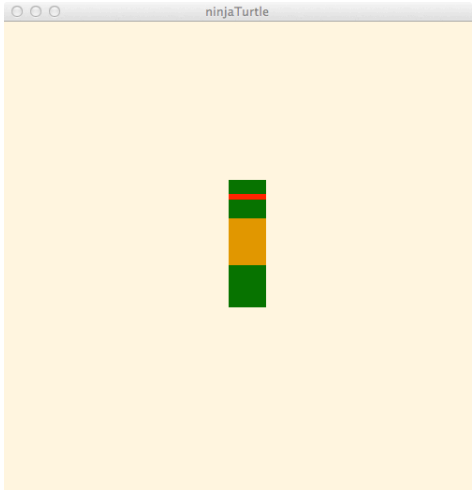
- Variables names are meaningless to computers, but meaningful to people … don't lie

```
color myWhite = color(0,0,0); //White … ha, ha!
```

- Variables are declared at top of a program

Add A Variable

- Raphael gets a var
- Adding the variable value (0) to each horizontal position results in no change

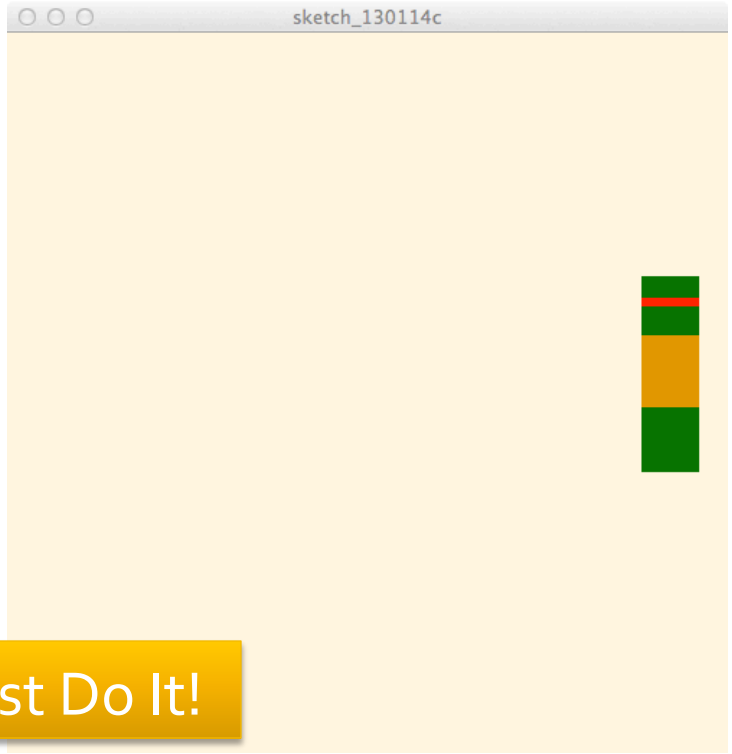



```
int ra = 0;

void setup( ) {
  size(500,500);
  noStroke();
}
void draw() {
  background(255, 245, 220);
  raff( );
}
void raff( ) {
  fill(0,100,0);
  rect(240+ra,260, 40, 45);
  fill(219,136,0);
  rect(240+ra,210, 40, 50);
  fill(0,100,0);
  rect(240+ra,190, 40, 20);
  fill(255,0,0);
  rect(240+ra, 184, 40, 6);
  fill(0,100,0);
  rect(240+ra, 169, 40, 15);
}
```

Change Value!

- When ra has the value of 200, Raff's position is changed



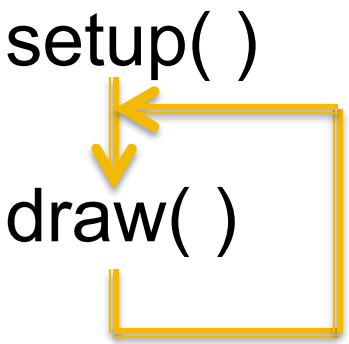
Just Do It!

```
int ra = 200;

void setup( ) {
  size(500,500);
  noStroke();
}
void draw() {
  background(255, 245, 220);
  raff( );
}
void raff( ) {
  fill(0,100,0);
  rect(240+ra,260, 40, 45);
  fill(219,136,0);
  rect(240+ra,210, 40, 50);
  fill(0,100,0);
  rect(240+ra,190, 40, 20);
  fill(255,0,0);
  rect(240+ra, 184, 40, 6);
  fill(0,100,0);
  rect(240+ra, 169, 40, 15);
}
```

Recall setup( ) and draw( )

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

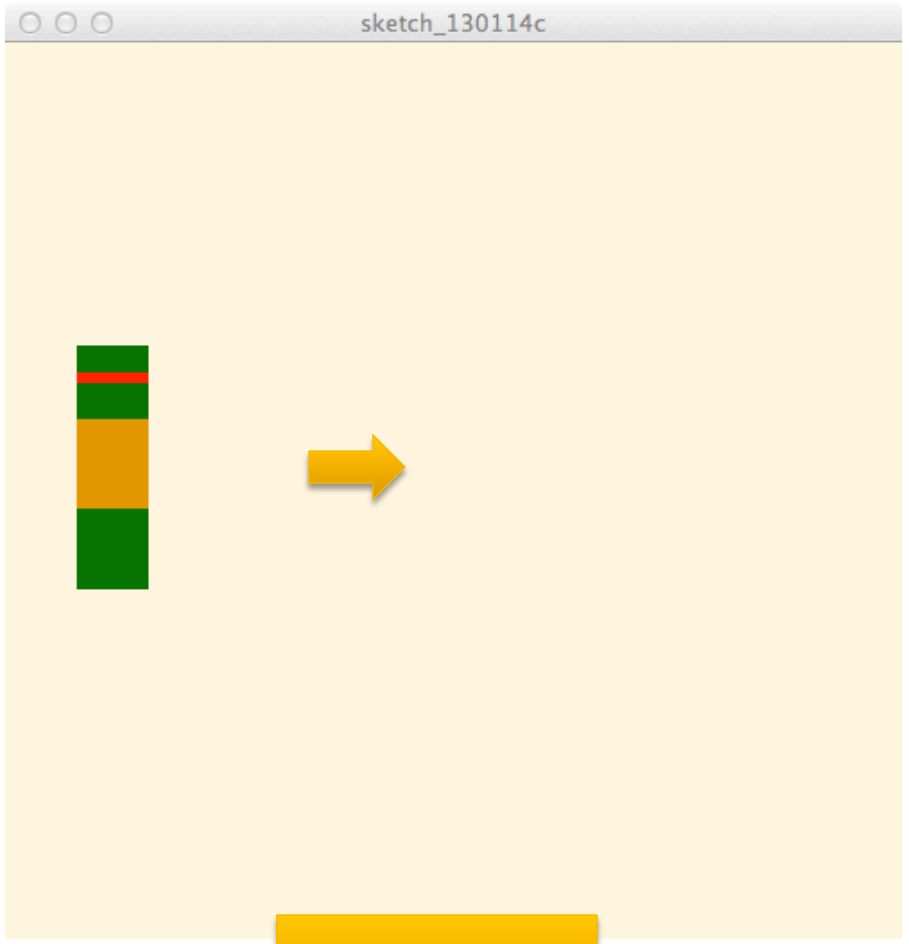


- Make Raphael run!

Start Raphael Left, Move Right

```
int ra = -200;

void setup( ) {
  size(500,500);
  noStroke();
}
void draw() {
  background(255, 245, 220);
  raff( );
  ra = ra + 1; //Add 1 to ra
}
void raff( ) {
  fill(0,100,0);
  rect(240+ra,260, 40, 45);
  fill(219,136,0);
  rect(240+ra,210, 40, 50);
  fill(0,100,0);
  rect(240+ra,190, 40, 20);
  fill(255,0,0);
  rect(240+ra, 184, 40, 6);
  fill(0,100,0);
  rect(240+ra, 169, 40, 15);
}
```



Just Do It!

Make Him Appear

- Start Raff off-screen to right, by initializing him to … ?
- Then make him move left by … ?
- And speed his movement up by … ?

Just Do It!

Raff The Left Running Ninja

- Note 400 is enough to hide him off screen
- Subtracting moves him left
- Changing `ra` by 2 speeds him up

```
int ra = 400;

void setup( ) {
  size(500,500);
  noStroke();
}
void draw() {
  background(255, 245, 220);
  raff( );
  ra = ra - 2; //Add 1 to ra
}
void raff( ) {
  fill(0,100,0);
  rect(240+ra,260, 40, 45);
  fill(219,136,0);
  rect(240+ra,210, 40, 50);
  fill(0,100,0);
  rect(240+ra,190, 40, 20);
  fill(255,0,0);
  rect(240+ra, 184, 40, 6);
  fill(0,100,0);
  rect(240+ra, 169, 40, 15);
}
```

More Ninjas

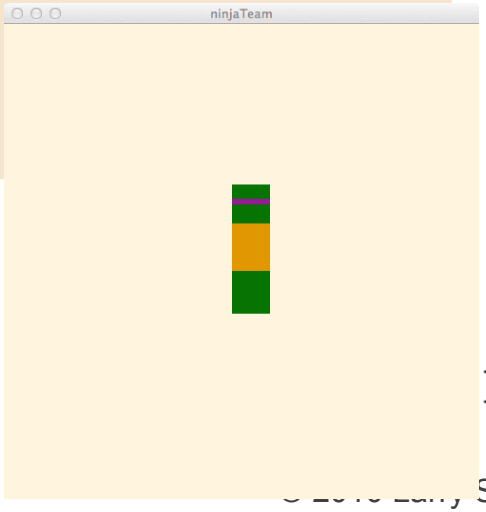
- Using Copy And Paste, create functions for the other three Ninjas
  - Each function has a Ninja name: mike, leo, don
  - Each ninja has a off-set variable: mi, le, dn
  - Each function is revised to off-set by its new variable
  - Each ninja has its off-set variable declared and initialized

Just Do It!

Ninja's By Copy/Paste… What???

```
int ra = 0;
int mi = 0;
int le = 0;
int dn = 0;

void setup( ) {
  size(500,500);
  noStroke();
}
void draw() {
  background(255, 245, 220);
  raff( );
  mike( );
  leo( );
  don( );
}
```




```
void raff( ) {
  fill(0,100,0);
  rect(240+ra,260, 40, 45);
  fill(219,136,0);
  rect(240+ra,210, 40, 50);
  fill(0,100,0);
  rect(240+ra,190, 40, 20);
  fill(255,0,0);
  rect(240+ra, 184, 40, 6);
  fill(0,100,0);
  rect(240+ra, 169, 40, 15);
}
```

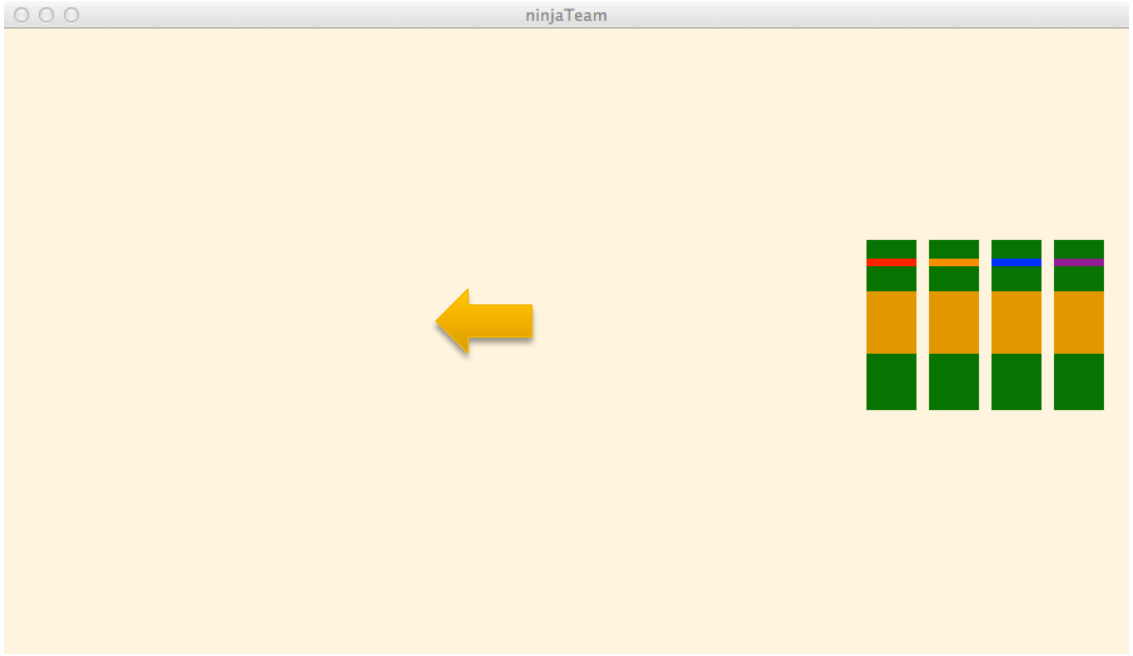
```
void mike( ) {
  fill(0,100,0);
  rect(240+mi,260, 40, 45);
  fill(219,136,0);
  rect(240+mi,210, 40, 50);
  fill(0,100,0);
  rect(240+mi,190, 40, 20);
  fill(250,122,0);
  rect(240+mi, 184, 40, 6);
  fill(0,100,0);
  rect(240+mi, 169, 40, 15);
}
```

```
void leo( ) {
  fill(0,100,0);
  rect(240+le,260, 40, 45);
  fill(219,136,0);
  rect(240+le,210, 40, 50);
  fill(0,100,0);
  rect(240+le,190, 40, 20);
  fill(0,0,255);
  rect(240+le, 184, 40, 6);
  fill(0,100,0);
  rect(240+le, 169, 40, 15);
}
```

```
void don( ) {
  fill(0,100,0);
  rect(240+dn,260, 40, 45);
  fill(219,136,0);
  rect(240+dn,210, 40, 50);
  fill(0,100,0);
  rect(240+dn,190, 40, 20);
  fill(128,0,128);
  rect(240+dn, 184, 40, 6);
  fill(0,100,0);
  rect(240+dn, 169, 40, 15);
}
```

Now We're Ready To Play!

- Start Ninja's to the right, and move them left, but at different rates …
  - raff moves 1
  - mike moves 2
  - leo moves 3
  - don moves 4



Just Do It!

New Variables Mean New Stunts

- Return to Raff, and add five new variables of type float … and add to vertical dim.

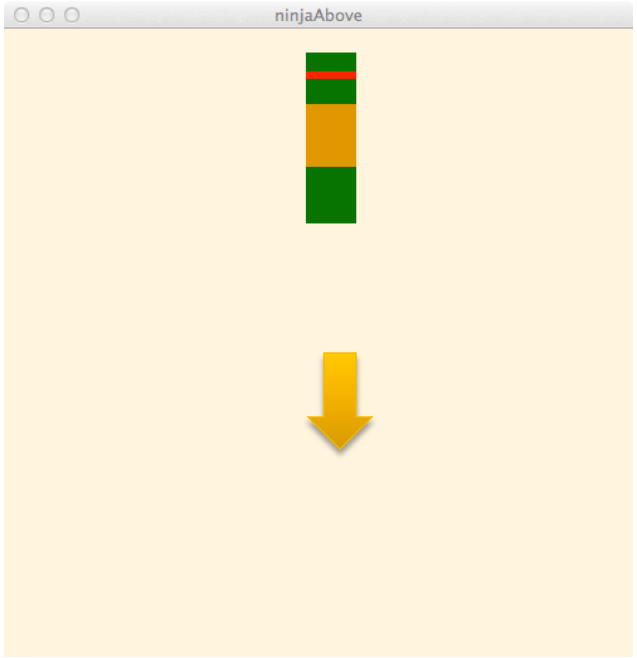
```
float ra = 0.0;
float rb = 0.0;
float rc = 0.0;
float rd = 0.0;
float re = 0.0;

void setup( ) {
  size(500,500);
  noStroke();
}
void draw() {
  background(255, 245, 220);
  raff( );
}
void raff( ) {
  fill(0,100,0);
  rect(240,260+ra, 40, 45);
  fill(219,136,0);
  rect(240,210+rb, 40, 50);
  fill(0,100,0);
  rect(240,190+rc, 40, 20);
  fill(255,0,0);
  rect(240,184+rd, 40, 6);
  fill(0,100,0);
  rect(240,169+re, 40, 15);
}
```

Add Some Action!

- We want Raff to drop down …
  - Translate his position by -150
  - Add 1 to each new variable
  - … but, he doesn't stop

Just Do It!




```
float ra = -150.0;
float rb = -150.0;
float rc = -150.0;
float rd = -150.0;
float re = -150.0;

void setup( ) {
  size(500,500);
  noStroke();
}

void draw() {
  background(255, 245, 220);
  raff( );
  ra = ra + 1;
  rb = rb + 1;
  rc = rc + 1;
  rd = rd + 1;
  re = re + 1;
}
```

. . .

Analyze What Happens

- As the value of ra, say, changes, Raff's position changes …

```
fill(0,100,0);
rect(240,260+ra, 40, 45);
     ...
ra = ra + 1; //Add 1 to ra
```

- Consider changes [position blue; ra red]
  - 110 = 260+(-150)
    - 149 = -150 + 1
  - 111 = 260+(-149)
    - 148 = -149 + 1
  - 112 = 260+(-148)
    - 147 = -148 + 1

…

Continuing The Analysis

- The off-set ra gets less and less negative, eventually getting to
  - 259 = 260+-(1)
    - 0 = -1 + 1
  - 260 = 260+0
    - 1 = 0 + 1

      ...
- We want to stop when ra gets to 0
- So, don't do `ra = ra+1`, write `ra = min(0, ra)`
- What happens??? min(a,b) gives the smaller of a, b

Check Out The min( ) Function

min(a,b) gives the smaller of a, b

- 110 = 260+(-150)
  - 149 = min(0,-150 + 1) The same!
- 111 = 260+(-149)
  - 148 = min(0,-149 + 1) The same!
- 112 = 260+(-148)
  - 147 = min(0,-148 + 1) The same!

  …

- 259 = 260+-(1)
  - 0 = min(0,-1 + 1) No difference, the same!
- 260 = 260+0
  - 0 = min(0,0 + 1) Stays at 0 … forever!

Just Do It!

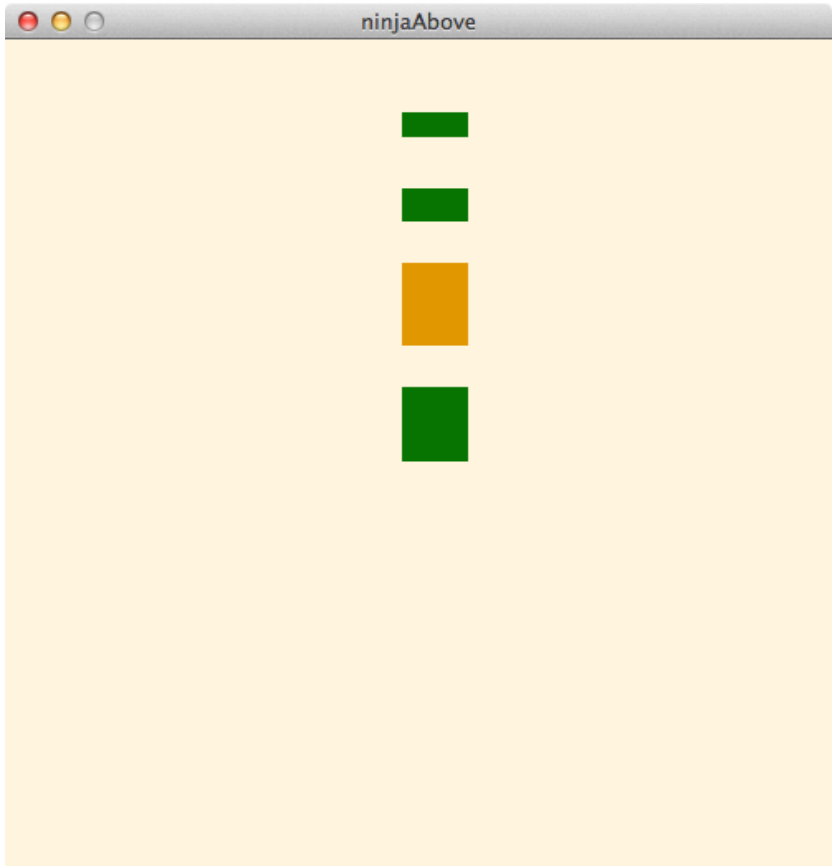
Raff Drops And Stops

- The code simply applies the `min( )` function

```
void draw() {
  background(255, 245, 220);
  raff( );
  ra = min(0,ra + 1);
  rb = min(0,rb + 1);
  rc = min(0,rc + 1);
  rd = min(0,rd + 1);
  re = min(0,re + 1);
}
void raff( ) {
  fill(0,100,0);
  rect(240,260+ra, 40, 45);
  fill(219,136,0);
  rect(240,210+rb, 40, 50);
  fill(0,100,0);
  rect(240,190+rc, 40, 20);
  fill(255,0,0);
  rect(240, 184+rd, 40, 6);
  fill(0,100,0);
  rect(240, 169+re, 40, 15);
}
```

Best Stunt Of All: Reform

- Change the amount Raff's parts fall so he appears to reassemble!



Requires float rd

```
void draw() {
  background(255, 245, 220);
  raff( );
  ra = min(0,ra + 4);
  rb = min(0,rb + 3);
  rc = min(0,rc + 2);
  rd = min(0,rd + 0.75);
  re = min(0,re + 1);
}
```

Just Do It!

Summary

- Today, we learned about
  - variables … names for quantities that vary in the program
  - datatypes … forms of data like integers, floating point numbers (decimal numbers), colors, booleans, etc.
  - declarations … statements that define what datatype variables are, as in int ra = 0;
  - And we learned the min( ) function