Functions in Processing

CSE 120 Spring 2017

Instructor: Justin Hsia

Teaching Assistants: Anupam Gupta, Braydon Hall, Eugene Oh, Savanna Yee

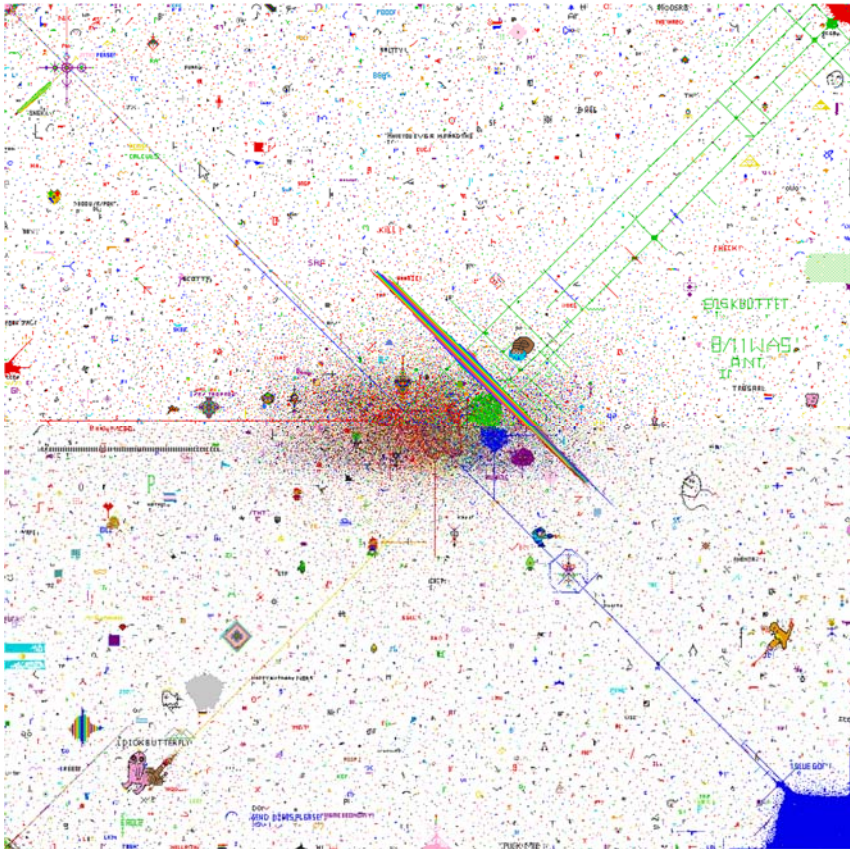
When Pixels Collide

For April Fool's Day, Reddit launched a little experiment. It gave its users, who are all anonymous, a blank canvas called Place.

The rules were simple. Each user could choose one pixel from 16 colors to place anywhere on the canvas. They could place as many pixels of as many colors as they wanted, but they had to wait a few minutes between placing each one.

Over the following 72 hours, what emerged was nothing short of miraculous.

- http://sudoscript.com/reddit-place/



Administrivia

- Assignments:
  - Custom Logo due today (4/7)
  - Lego Family due Sunday (4/9)

- Make sure to take advantage of office hours and Piazza!

Drawing a Square with Functions

- [See Demo on Panopto]

Donatello as a Function

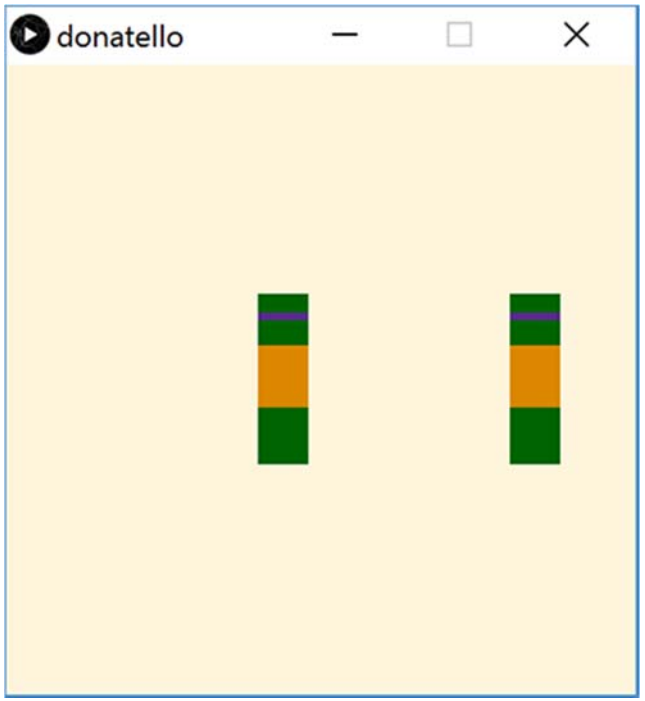
```
13 // draw Donatello
14 void donatello() {
15   fill(0,100,0);            // dark green
16   rect(x_pos,182,40,15);    // top of head
17
18   fill(88,44,141);          // purple
19   rect(x_pos,197,40,6);     // bandana mask
20
21   fill(0,100,0);            // dark green
22   rect(x_pos,203,40,20);    // bottom of head
23
24   fill(219,136,0);          // dark yellow
25   rect(x_pos,223,40,50);    // shell
26
27   fill(0,100,0);            // dark green
28   rect(x_pos,273,40,45);    // lower body
29 }
```

Donatello Function *Parameterized*

- Can now call `donatello()` function with different `x_pos`

```
14 // draw Donatello
15 void donatello(int x_pos) {
16   fill(0,100,0);            // dark green
17   rect(x_pos,182,40,15);    // top of head
18
```

```
8  void draw() {
9    background(255,245,220);
10   donatello(200);
11   donatello(400);
12 }
```



Return Type

return type

```
14 // draw Donatello
15 void donatello(int x_pos) {
16   fill(0,100,0);              // dark green
17   rect(x_pos,182,40,15);      // top of head
18
```

- What the function sends back to whoever called it
  - Can be any of the datatypes: `int`, `float`, `color`, etc.
  - If not returning anything, then we use `void`

Function Name

donTurtle
don_turtle
don-turtle
5onatello

function name

```
14 // draw Donatello
15 void donatello(int x_pos) {
16   fill(0,100,0);              // dark green
17   rect(x_pos,182,40,15);      // top of head
18
```

- Does not matter to computer, but does to humans
  - Should describe what the function does
- *Must* start with a letter, but can contain numbers and underscores
  - Why not hyphen? – → hyphen? → subtraction?
- No two functions (or variables) can have the same name

Parameters

parameters

```
14 // draw Donatello
15 void donatello(int x_pos) {
16   fill(0,100,0);              // dark green
17   rect(x_pos,182,40,15);      // top of head
18
```

- Required part of every function definition
  - Must be surrounded by parentheses
  - If no parameters, parentheses are left empty
- Datatype and name for every parameter must be specified
  - Separate parameters with commas

Function Body

parameters are usable within the function body

```
12 // draw Donatello
13 void donatello(int x_pos) {
14   fill(0,100,0);              // dark green
15   rect(x_pos,182,40,15);      // top of head
16
17   fill(88,44,141);            // purple
18   rect(x_pos,197,40,6);       // bandana mask
19
20   fill(0,100,0);              // dark green
21   rect(x_pos,203,40,20);      // bottom of head
22
23   fill(219,136,0);            // dark yellow
24   rect(x_pos,223,40,50);      // shell
25
26   fill(0,100,0);              // dark green
27   rect(x_pos,273,40,45);      // lower body
28 }
```

body

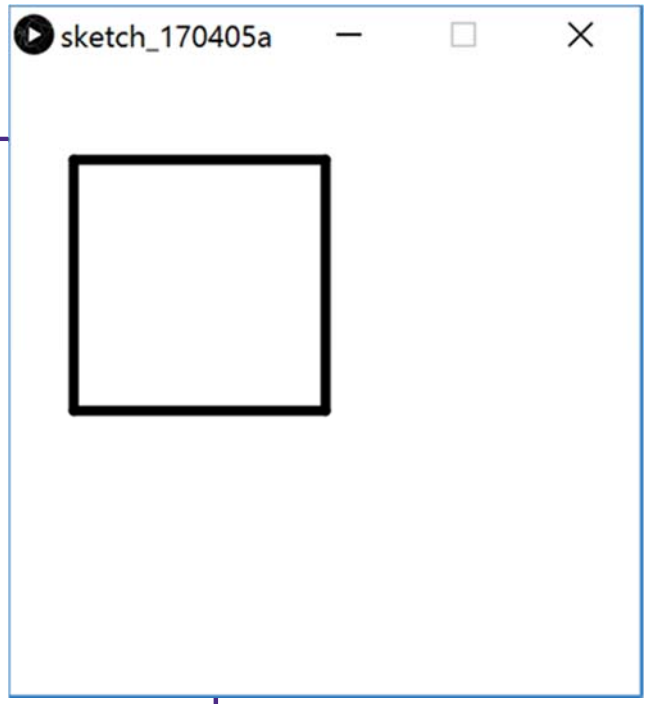
Lightbot Functions

- Lightbot functions had a different syntax, but similar parts:

function name parameters body

```
F.turn_around() Right, Right.
```

Parameters vs. Arguments




```
1  void setup() {
2    size(500,500);
3    background(255);
4    strokeWeight(8);
5  }
6
7  void draw() {
8    drawSquare(50,75,200,color(0));
9    noLoop();
10 }
11
12 void drawSquare(int x, int y, int len, color c) {
13   stroke(c);
14   line(x,     y,     x+len,y);
15   line(x+len,y,     x+len,y+len);
16   line(x+len,y+len,x,     y+len);
17   line(x,     y+len,x,     y);
18 }
```

arguments

parameters

Parameters vs. Arguments

- When you define a function, you specify the parameters
  - Use parameters for values that you want to be different on different calls to this function
- When you call a function, you pass arguments
  - The order of the arguments must match the order of the parameters
- We define a function once, but can call it as many times as we want!

Variable Scope

- When an argument is passed to a function, what does the function actually get?
  - Internal variables (*i.e.* parameters) get a *copy* of the argument value

- Internal variables only exist within the function they are declared
  - The variables "cease to exist" when the function finishes
  - "Scope" of a variable is the part(s) of code where that variable name binding is valid (*i.e.* where it exists)

Question

- If you're dreaming and someone in your dream hands you a turnip, do you wake up with a turnip in your bed?

  A. Yes
  B. No
  C. I will report back on Monday

- Variable scope demo in Processing: [see Panopto]

Parameter Example

```
20 // draw mouse at position (x,y) in color c
21 void mouse() {
22   noStroke();
23   fill(color(255,0,255));     // magenta color
24   ellipse(50, 50, 50, 50);  // head
25   ellipse(25, 30, 30, 30);  // right ear (left on screen)
26   ellipse(75, 30, 30, 30);  // left ear (right on screen)
27
28   fill(0);                  // black color
29   ellipse(40, 44, 10, 10);  // right eye (left on screen)
30   ellipse(60, 44, 10, 10);  // left eye (right on screen)
31
32   stroke(0);                // black color
33   line(20, 50, 48, 60);     // upper-right whisker
34   line(80, 50, 52, 60);     // upper-left whisker
35   line(25, 70, 48, 60);     // lower-right whisker
36   line(75, 70, 52, 60);     // lower-left whisker
37 }
```

Parameter Example

```
13 void draw() {
14   mouse(0,   0, color(255, 0, 0));
15   mouse(100, 0, color(0, 255, 0));
16   mouse(200, 0, color(0, 0, 255));
17 }
18
19 // draw mouse at position (x,y) in color c
20 void mouse(int x, int y, color c) {
21   noStroke();
22   fill(c);                          // argument color
23   ellipse(50+x, 50+y, 50, 50);      // head
24   ellipse(25+x, 30+y, 30, 30);      // right ear (left on screen)
25   ellipse(75+x, 30+y, 30, 30);      // left ear (right on screen)
26
27   fill(0);                          // always black
28   ellipse(40+x, 44+y, 10, 10);      // right eye (left on screen)
29   ellipse(60+x, 44+y, 10, 10);      // left eye (right on screen)
30
31   stroke(0);                        // always black
32   line(20+x, 50+y, 48+x, 60+y); // upper-right whisker
33   line(80+x, 50+y, 52+x, 60+y); // upper-left whisker
34   line(25+x, 70+y, 48+x, 60+y); // lower-right whisker
35   line(75+x, 70+y, 52+x, 60+y); // lower-left whisker
36 }
```

Solving Problems

- Understand the problem
  - What is the problem description?
  - What is specified and what is *un*specified?
  - What has been given to you (*e.g.* starter code)?
- Break the task down into less complex subtasks

- Example: Make a function that draws a row of five mice with their ears touching/overlapping. The mice should all be the same color except for the middle one, which should be red.

Looking Forward

- Lego Family
  - Design an abstracted family
  - Create functions for drawing each family member, including variables for position/movement
  - Have family members start at corners, then move into place
- Events
  - Introduce user interactions! Due Tuesday (4/11)
- Animal Functions
  - Start in lab on Tuesday, due Wednesday (4/12)
  - Design your own animal (like the mouse shown here)