

Lecture F3

Programmer Defined Functions

Computing and Art : Nature, Power, and Limits
CC 3.12: Spring 2008

Functionalia

Instructor

Robert Faludi, ***faludi@sci.brooklyn.cuny.edu***

Course Web Page

[http://www.sci.brooklyn.cuny.edu/~faludi/
cc3.12/](http://www.sci.brooklyn.cuny.edu/~faludi/cc3.12/)

- **HW F: DUE** Wednesday, April 30th, 11:59 pm

Functionalia

Today:

- Programmer Defined Functions

Programmer Defined Functions

If you have some piece code (multiple lines) that is useful and you want to use it many times - not just in a while loop, but other times in your program - then you can group the code into a **Function**.

A **function** can have **parameters**, which are ways that the piece of can vary in it's functionality.

Example: You have some code that draws a house. You can put that code into a **function** and use it many times in your program. **Parameters** to the house, could be it's place in the sketch and it's width and height.

You've been *using* functions all along!

All of the commands for drawing, setting the size of the sketch, setting the background color, the fill color etc... are functions that someone provided for you:

```
rect(200, 200, 20, 20);
```

This is the function to draw a rectangle. It's name is `rect`, and it has 4 parameters:

1st parameter: x-position of the rectangle

2nd parameter: y-position of the rectangle

3rd parameter: width of the rectangle

4th parameter: height of the rectangle

You've been *using* functions all along!

All of the commands for drawing, setting the size of the sketch, setting the background color, the fill color etc... are functions that someone provided for you:

```
rect(200, 200, 20, 20);
```

This is the function to draw a rectangle. It's name is `rect`, and it has 4 parameters:

1st parameter: x-position of the rectangle

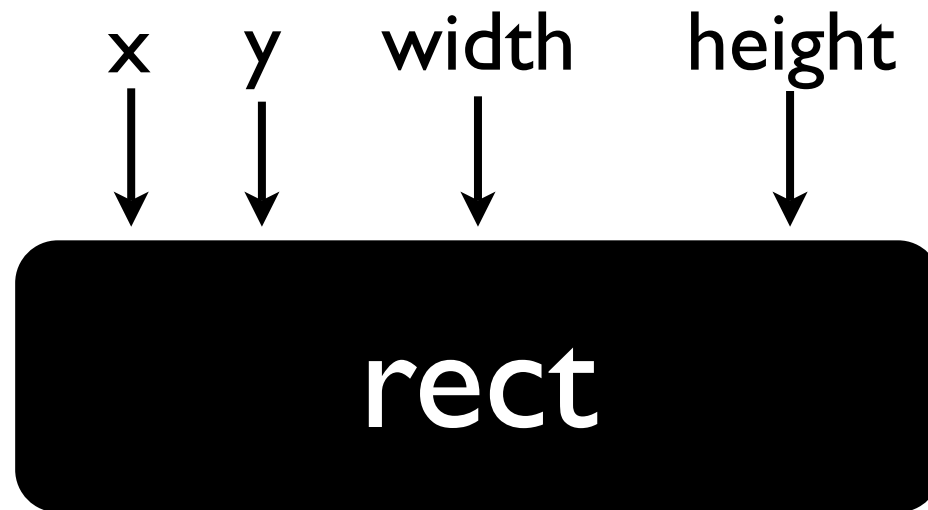
2nd parameter: y-position of the rectangle

3rd parameter: width of the rectangle

4th parameter: height of the rectangle

Functions are sometimes modeled as a black box

The “Black Box” model is that once you have a function, you don’t have to worry about the details inside of it.



You just use it, and you give it input. Sometimes functions can give you output, called a **return value**.

Programmer Defined Functions

Note: This tutorial goes step-by-step on adding your own function to a program.

NOT EVERY STEP WILL RUN CORRECTLY.

Steps that have the blue run, will run correctly.

RUN

```
// Add your variables here
```

```
void setup() {
```

```
    size(600, 600);
```

```
}
```

```
void draw() {
```

```
    background(255);
```

```
    // draws a house
```

```
    rect(100,100,30,30);
```

```
    line(95, 105, 115, 85);
```

```
    line(115, 85, 135, 105);
```

```
}
```

1. Let us start with a simple sketch of a house.

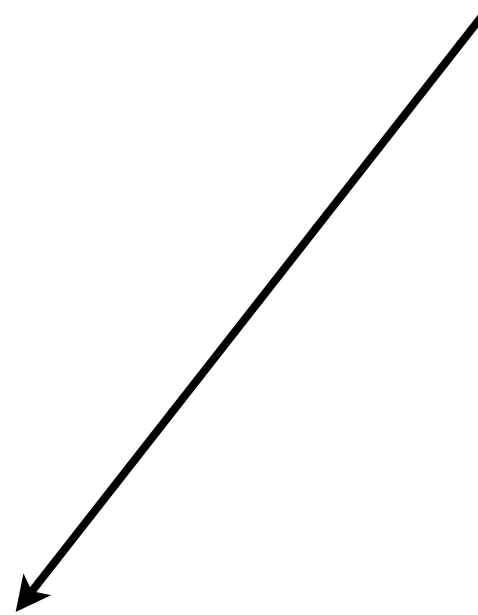
RUN

```
// Add your variables here
```

```
void setup() {  
    size(600, 600);  
}
```

```
void draw() {  
    background(255);
```

2. Copy the code that draws the house.



```
// draws a house  
rect(100,100,30,30);  
line(95, 105, 115, 85);  
line(115, 85, 135, 105);  
}
```

```
// Add your variables here
```

```
void setup() {  
  size(600, 600);  
}
```

```
void draw() {  
  background(255);  
  
}
```

```
// draws a house  
rect(100, 100, 300, 300);  
line(95, 105, 115, 85);  
line(115, 85, 135, 105);
```

3. Paste it **OUTSIDE** of the *draw* and *setup* functions.

Remove the code from the *draw()* function.

Leave the *background(255);* function.

```
// Add your variables here

void setup() {
    size(600, 600);
}

void draw() {
    background(255);
}

void house() {
    // draws a house
    rect(100,100,30,30);
    line(95, 105, 115, 85);
    line(115, 85, 135, 105);
}
```

4. Wrap the code with curly braces, and add:
`void house()`

Note: `house` is the name of the function. When creating your own functions, you can pick ANY name for your functions that makes sense to use. Watch out for RESERVED KEYWORDS.

What is similar to `void house()` in this program?

```
// Add your variables here

void setup() {
    size(600, 600);
}

void draw() {
    background(255);
    house();
}

void house() {
    // draws a house

    rect(100,100,30,30);

    line(95, 105, 115, 85);

    line(115, 85, 135, 105);
}
```

5. We can now use our house() function like all of the other functions we use to draw.

This is called “calling the function house”. This is the SAME way we are using all of the other functions that processing gives to us.

RUN

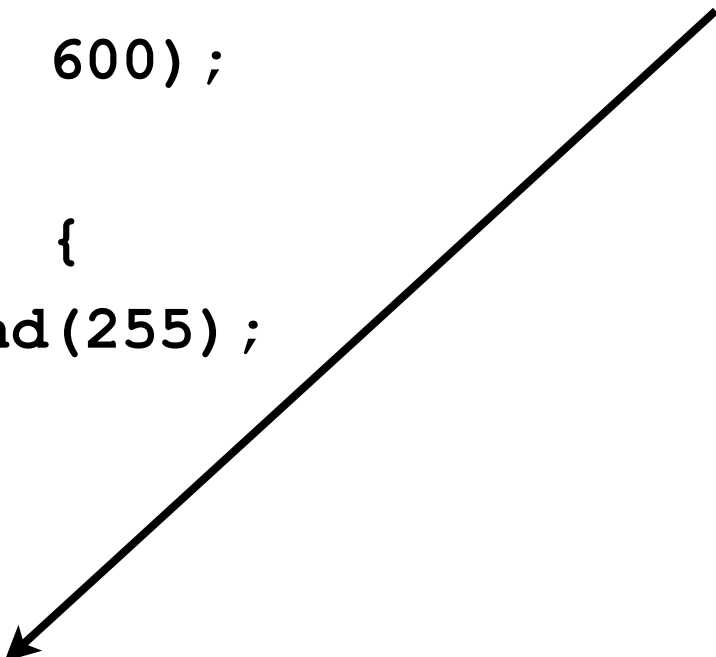
```
// Add your variables here
void setup() {
  size(600, 600);
}
void draw() {
  background(255);
  house();
  house();
  house();
}
void house() {
  // draws a house
  rect(100,100,30,30);
  line(95, 105, 115, 85);
  line(115, 85, 135, 105);
}
```

6. We can call house() over and over again.

BUT, how many houses do you see in the sketch?

```
// Add your variables here
void setup() {
  size(600, 600);
}
void draw() {
  background(255);
  house();
}

void house() {
  // draws a house
  rect(100, 100, 30, 30);
  line(95, 105, 115, 85);
  line(115, 85, 135, 105);
}
```



7. We put some variety in where the house is drawn by adding our own **parameters** to our function.

Parameters are similar to variables. We can name them anything we want. We use them like variables in our function, **but** we don't change the values of them.

They allow use to control the behavior **within** the function from the outside of the function.

```
// Add your variables here
void setup() {
  size(600, 600);
}
void draw() {
  background(255);
  house();
}

void house(int xpos) {
  // draws a house
  rect(100,100,30,30);
  line(95, 105, 115, 85);
  line(115, 85, 135, 105);
}
```

7 cont. Add a parameter between the parentheses in the definition of the function.

It is defined like a variable.

Then change the code within the function to USE the parameter.

Here we are adding a parameter to control where the house is being drawn in the x direction.

Notice replacing the x-coordinate of the rectangle function with the parameter.

```
// Add your variables here
void setup() {
    size(600, 600);
}

void draw() {
    background(255);
    house(100);
    house(300);
}

void house(int xpos) {
    // draws a house
    rect(xpos, 100, 30, 30);
    line(95, 105, 115, 85);
    line(115, 85, 135, 105);
}
```

8. We now can change how we call the house function with numerical values in the parameter.

Notice that the base of the house varies in the x-direction. What about the roof?

How might we have the roof also change with the xpos parameter?

RUN

```
// Add your variables here
void setup() {
  size(600, 600);
}

void draw() {
  background(255);
  house(100);
  house(300);
}

void house(int xpos) {
  // draws a house
  rect(xpos, 100, 30, 30);
  line(xpos - 5, 105, xpos + 15, 85);
  line(xpos + 15, 85, xpos + 35, 105);
}
```

9. Here is the house varying in the x direction with the base and the roof of the house.

Notice how `xpos` is used in **all** of the drawing functions.

RUN

```
// Add your variables here
void setup() {
  size(600, 600);
}
void draw() {
  background(255);
  house(100);
  house(300);
  house(350);
  house(440);
}
void house(int xpos) {
  // draws a house
  rect(xpos, 100, 30, 30);
  line(xpos - 5, 105, xpos + 15, 85);
  line(xpos + 15, 85, xpos + 35, 105);
}
```

10. We can add a whole row of houses.

You could also use a while loop to draw many houses by using the house function.

RUN

```
// Add your variables here
```

```
void setup() {  
    size(600, 600);  
}
```

```
void draw() {  
    background(255);  
    house(100);  
    house(300);  
    house(350);  
    house(440);  
}
```

```
// Draws a house
```

```
// xpos - the x position of the house
```

```
void house(int xpos) {  
    rect(xpos, 100, 30, 30);  
    line(xpos - 5, 105, xpos + 15, 85);  
    line(xpos + 15, 85, xpos + 35, 105);  
}
```

||. Let's add a comment above our function now, that describes what our parameter does.

This is useful for use to share our functions with others.

RUN

12. Now, how can we vary the houses with a y variable?

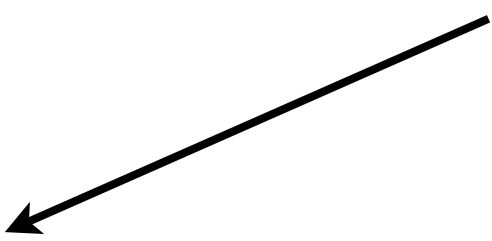
```
// Add your variables here
void setup() {
  size(600, 600);
}
void draw() {
  background(255);
  house(100);
}

// Draws a house
// xpos - the x position of the house
void house(int xpos) {
  rect(xpos, 100, 30, 30);
  line(xpos - 5, 105, xpos + 15, 85);
  line(xpos + 15, 85, xpos + 35, 105);
}
```

13. First, add a y parameter.

```
// Add your variables here
void setup() {
  size(600, 600);
}
void draw() {
  background(255);
  house(100, 100);
}
```

We'll call it ypos, and we'll have to add an additional parameter when we call the function.



```
// Draws a house
// xpos - the x position of the house
void house(int xpos, int ypos) {
  rect(xpos, 100, 30, 30);
  line(xpos - 5, 105, xpos + 15, 85);
  line(xpos + 15, 85, xpos + 35, 105);
}
```

```
// Add your variables here
```

```
void setup() {  
    size(600, 600);  
}
```

```
void draw() {  
    background(255);  
    house(100, 100);  
}
```

```
// Draws a house
```

```
// xpos - the x position of the house
```

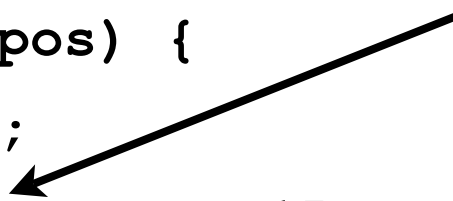
```
void house(int xpos, int ypos) {  
    rect(xpos, ypos, 30, 30);  
    line(xpos - 5, ypos + 5, xpos + 15, ypos - 15);  
    line(xpos + 15, ypos - 15, xpos + 35, ypos + 5);  
}
```

14. Now we will use `ypos` within the function to draw the house.

Notice again that `ypos` is used in every drawing function.

Everything that is drawn in the function is drawn using the parameters with some offset.

For example, this line starts at the point $(xpos - 5, ypos + 5)$.



RUN

```
// Add your variables here
```

```
void setup() {  
    size(600, 600);  
}
```

```
void draw() {  
    background(255);  
    house(100, 100);  
    house(130, 220);  
    house(400, 50);  
}
```

```
// Draws a house
```

```
// xpos - the x position of the house
```

```
void house(int xpos, int ypos) {  
    rect(xpos, ypos, 30, 30);  
    line(xpos - 5, ypos + 5, xpos + 15, ypos - 15);  
    line(xpos + 15, ypos - 15, xpos + 35, ypos + 5);  
}
```

15. We can now draw houses all over the place. By varying both of the parameters in the **house** function.

RUN

```
// Add your variables here
void setup() {
    size(600, 600);
}

void draw() {
    background(255);
    house(100, 100);
}

// Draws a house
// xpos - the x position of the house
// ypos - the y position of the house
void house(int xpos, int ypos) {
    rect(xpos, ypos, 30, 30);
    line(xpos - 5, ypos + 5, xpos + 15, ypos - 15);
    line(xpos + 15, ypos - 15, xpos + 35, ypos + 5);
}
```

16. We should now update our comments for the function by documenting what the second parameter does.

RUN

```
// Add your variables here
void setup() {
  size(600, 600);
}
void draw() {
  background(255);
  house(mouseX, mouseY);
}
// Draws a house
// xpos - the x position of the house
// ypos - the y position of the house
void house(int xpos, int ypos) {
  rect(xpos, ypos, 30, 30);
  line(xpos - 5, ypos + 5, xpos + 15, ypos - 15);
  line(xpos + 15, ypos - 15, xpos + 35, ypos + 5);
}
```

17. We can use the function now like any other drawing function we did in the past.

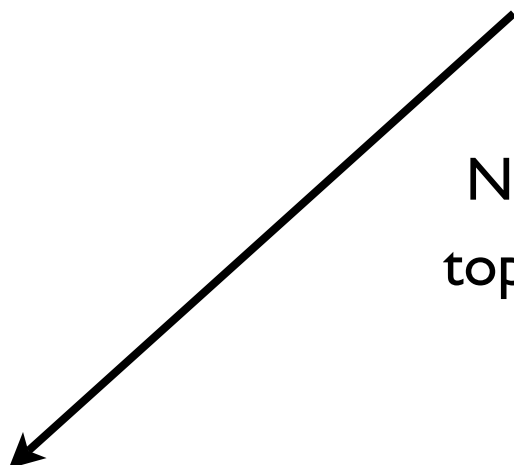
Here we will have the house drawn with the mouse variables. The house follows the mouse.

RUN

```
// Add your variables here
int x;
void setup() {
  size(600, 600);
}
void draw() {
  background(255);
  x = 0;
  while(x < 600) {
    house(x, 100);
    x = x + 55;
  }
}
// Draws a house
// xpos - the x position of the house
// ypos - the y position of the house
void house(int xpos, int ypos) {
  rect(xpos, ypos, 30, 30);
  line(xpos - 5, ypos + 5, xpos + 15, ypos - 15);
  line(xpos + 15, ypos - 15, xpos + 35, ypos + 5);
}
```

18. We can use our function in a while loop to draw a row of houses.

Notice that we added a variable at the top of the program like we did in lab F1.



RUN

```
// Add your variables here
```

```
int x;
```

```
void setup() {  
    size(600, 600);  
    x = 0;  
}
```

```
void draw() {  
    background(255);  
    x = x + 1;  
    if(x > 600) {  
        x = 0;  
    }  
    house(x, 100);  
}
```

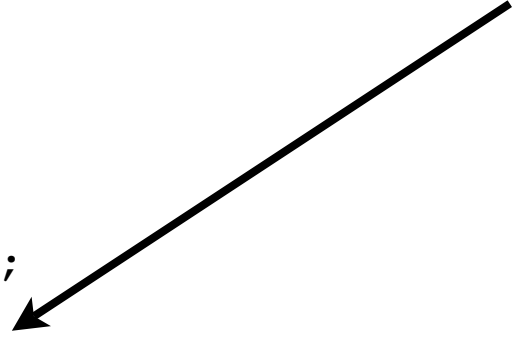
```
// Draws a house
```

```
// xpos - the x position of the house
```

```
// ypos - the y position of the house
```

```
void house(int xpos, int ypos) {  
    rect(xpos, ypos, 30, 30);  
    line(xpos - 5, ypos + 5, xpos + 15, ypos - 15);  
    line(xpos + 15, ypos - 15, xpos + 35, ypos + 5);  
}
```

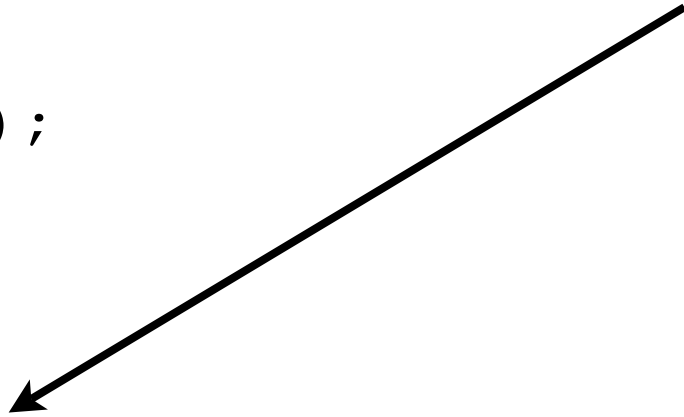
19. We can also use a variable to move the house across the screen. Just like we did with the Teddy Bear in Lab E1.



RUN

```
// Add your variables here
int x;
void setup() {
  size(600, 600);
  x = 0;
}
void draw() {
  background(255);
  x = x + 1;
  if(x > 600) {
    x = 0;
  }
  house(x, 100);
  house(200, x);
}
// Draws a house
// xpos - the x position of the house
// ypos - the y position of the house
void house(int xpos, int ypos) {
  rect(xpos, ypos, 30, 30);
  line(xpos - 5, ypos + 5, xpos + 15, ypos - 15);
  line(xpos + 15, ypos - 15, xpos + 35, ypos + 5);
}
```

20. Notice I can add another call to the house function, but by using the variable in the other parameter of the house function I can get another house to move vertically.



RUN

HW F

HW F Part II continues to work with programmer defined functions

- **HW F: DUE** Wednesday, April 30th, 11:59 pm