Functions

Look at the command for drawing a line:

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

In the above command, `line` is the name of the command. We also call these a function (or a method). Following the name of the function, in the parentheses, are four arguments (or parameters).

Next, look at the command:

```plaintext
float x = random(10);
```

Above, `x` will be assigned a random value from the range `[0.0 .. 10)`. `random()` is also a function. However, unlike `line()`, we use `random()` where a value can be used (i.e. in an expression). In Processing, some functions do a task (like, `line()`) while others return (or stand for) a value. Therefore, where and how they can be used varies. Functions that just do a task are written as commands (like, `line()`), while those that do a computation and return a result in a value (like, `random()`) are used in expressions.

Which of the following functions you have seen return values:\(^1\):

1. `line()` ________
2. `random()` ________
3. `int()` ________
4. `square()` ________
5. `draw()` ________
6. `round()` ________
7. `println()` ________
8. `radians()` ________
9. `frameRate()` ________
10. `width` ________

As was the case for variables, there are a whole lot of pre-defined functions, and you can also define your own functions. Let us learn how to define and use a function:

In the last Chapter, we learned to draw a truck using variables so that given the coordinates of its anchor point `(x, y)` and its width and height, we were able to draw a truck anywhere on the canvas. Now, imagine if we could simply draw a truck using a command:

```plaintext
drawTruck(x, y, w, h);
```

This is indeed possible by defining your own functions. `drawTruck()`, like the other drawing functions, performs a task. It will draw a truck anchored at `(x, y)` of width, `w` and height `h`. To define the function `drawTruck()` we use the following code:

```plaintext
void drawTruck(float x, float y, float w, float h) {
```

\(^1\) The `line()`, `square()`, `draw()`, `println()`, and `framerate()` do not have any return values; `int()` and `round()` return an `int` value; `radians()` returns a `float` value; `width` is not a function (it is a variable).
// commands to draw the truck at (x, y) of width, w and height, h go here

Here is a complete definition:

```cpp
void drawTruck(float x, float y, float w, float h) {
    // Draws a truck at (x, y) of width, w and height, h go here
    float bodyW = w * 2/3, bodyH = h;  // width, height of body
    float hoodW = W = bodyW, hoodH = bodyH/2; // width, height of hood
    float wheelD = bodyW*0.3;    // wheel width (dia)

    // Draw Body
    rect(x, y-h, bodyW, bodyH);

    // Draw Hood
    rect(x+bodyW, y-hoodH, hoodW, hoodH);

    // Draw Window
    fill(0);
    triangle(x+bodyW, y-bodyH, x+bodyW, y-hoodH, x+bodyW+hoodW/2, y-hoodH);

    // Draw Wheels
    fill(0);
    circle(x+bodyW/2, y, wheelD);
    circle(x+bodyW+hoodW/2, y, wheelD);
}
```

Notice the structure of this definition. Looks familiar? It is almost like the definition of `setup()` and `draw()`. Including that weird word: `void`. What is the deal with `void`?

In fact, you can now add the definition above to a Processing program and try it defining the following `draw()` function:

```cpp
void draw() {
    fill(255, 46, 46);  // truck color
    drawTruck(50, 200, 90, 60);
}
```

Go ahead and try it. Did you see the truck? Good! Next, add the following to your program:

```cpp
void mousePressed() {
    fill(random(255), random(255), random(255));  // random truck color
    drawTruck(mouseX, mouseY, random(60, 100), random(30, 70));
}
```
Run the sketch. Now, whenever you click the mouse button a random colored, random sized truck will appear! So that is what functions are all about!

The syntax for defining functions that can be used as commands is show below:

```c
void <name of function> (<parameter definitions>) {
   <commands to do the task>
}
```

Now, read the definition of the `drawTruck()` function carefully. Notice that it takes four parameters. Each of the four parameters are defined with their types.

**Do this:** define a function called `deathlyHallows()` that draws the sign of the Deathly Hallows, from the Harry Potter series at a given location, of a given size.

### Functions That Return a Value

Above, we mentioned functions that do a computation and return a value. For example,

```c
float x = random(10);
```

`random(n)` does a computations to generate a random value in the range [0..n) and returns that as a result. Thus, we can use `random(10)` as above to represent that value. You can also define your own functions to do computations and return a resulting value. Let us take a simple example: compute the area of a circle, given its radius.

Let us call this function `areaOfCircle()`, thus given a radius, `r`, we could use the function as shown below:

```c
float r = 4.1;
float area = areaOfCircle(r);
```

Once defined, we can use the function repeatedly to compute the radius of any circle. Here is the definition of `areaOfCircle()`:

```c
float area OfCircle(float radius) {
   return PI * radius * radius;
}
```

Again, it kind of looks like the function definitions above. And it is (but no `void`!!). Functions that do a computation and return a result are defined as shown below:

```c
<return-value-type> <function name> (<parameters>) {
   <computations that need to be done go here>
   return <resulting value>
}
```

The two differences from defining functions that do a task (like `drawTruck()`) are:
• You must specify the return value type. That is, what is the type of value that the function returns. In areaOfCircle(), the return type is float. See?
• The value that the function returns when used, is communicated back by the return statement. The syntax of the return statement is:

    return <expression of return-value-type>;

Try: Write a function to compute the area of a square, given the length of its side.

What is the deal with void?

OK, here we go. The syntax for defining a function is just this:

<return-value-type> <function name> (<parameters>) {
    <computations that need to be done go here>
    return <resuling value>
}

If a function performs a task (like drawing something), it has nothing to return as a value. void is used as a return value type to indicate exactly that. There is nothing to return. If a function returns a void, it must be used as a command. It can still have a return statement of the form:

    return;

But it is not required. If present, it indicates that the task is done. In defining drawTruck() above we did not use the return statement. That is because, once all commands in a function are carried out, it returns. No return statement is needed. Later, you will see situations where a return statement may be needed in a function that returns nothing (i.e. void).

Tips on Defining Functions

• When something is going to be used repeatedly, it is a good idea to define a function.
• Pick a good name for the function.
• Deliberate over what parameters are needed, their types, return type, etc.
• Write down the algorithm to do the task/computation in terms of the parameters.
• Write the code and test it our several times for different values of parameters.