Functions #2 – More of More Than You Want to Know… But Still Not Everything You Need to Know…

If you have not looked at Shiffman and the previous set of notes, you should do that first.

Last time we considered a new way to look at functions -- in terms of:

- *who "owns" them*
- *who "uses" them*

By way of a review. . .
The entity that "owns" a function is the entity that wrote it. In the first week, all of the functions you used were owned by or written by Processing programmers: `fill(0)`, `ellipse( 10, 10, 5, 2 )`*... were all written by Processing programmers some time in the past.*

When someone "writes" a function (the better term to use is "defines a function") , they put in the code needed to tell Processing how to do whatever task the function is supposed to do. Processing defined all of the functions you have used thus far in the class.

The entity that "uses" the function is the entity that types the name of the function and the required arguments somewhere in a program. You were the user of the functions. You used the functions. A better term is call. You called the functions that were owned by Processing. Your programs have been a list of function calls.

Using this nomenclature, we can categorize the functions we will use in the next three class meetings.

The previous set of notes considered functions that we write or define the functions and Processing uses or calls. The ones considered were:

- `setup( )`
- `draw( )`

IF they are in our code, Processing will execute them. If not – no big deal.

The agreement we have with Processing is the following:

1. *Processing will take care of any variables we declare first.*
2. *Processing will call (and execute the code inside) the* `setup( )` *function if we have one defined (written) in or program.*
3. *Processing will call (and execute the code inside) the* `draw( )` *function if we have one defined (written) in or program.*

In table form we have this:

Which Function	*API*	*setup( )* *draw( )*	*Event Functions*	***?????***
Owns it	*Processing*	*us*	*Us*	***?????***
Uses it	*us*	*Processing*	*Processing*	***?????***

The blue stuff is for today and the ??? follows very soon.

Hopefully you are asking, "what is an event function[1]?"

When a user interacts with the computer, the interaction causes or generates what is called an event.

- *Every key press and key release generates its own event. Every mouse movement, mouse button press, and mouse button release generates an event.*
- *Moving the mouse from one window to another generates at least three events: mouse moved, mouse left a window, mouse entered a window.*

Each event has a "listener" that literally waits for the event to happen. Sort of like a little brother or sister waiting to tell on you… When the event happens, the listener tells the operating system what happened and any associated information that is needed – like which mouse button was pressed or which key was pressed.

Programs can request that they be notified when certain events occur. Processing asks for several event notifications. For the mouse it asks for the following:

- **mouseDragged**
- **mouseMoved**
- **mousePressed**
- **mouseReleased**

For the keyboard, it asks to be notified when the following events occur:

- **keyPressed**
- **keyReleased**
- **keyTyped**

For each of these, we can define (write) a function with the same name and Processing will all it when the even occurs.

- **mouseDragged( )**
- **mouseMoved( )**

[1] This is another "Jim term" and is not found in any real documentation.

- mousePressed( )
- mouseReleased( )
- keyPressed( )
- keyReleased( )
- keyTyped( )

Even though we write the definition of these functions in our code, they are explained in the API. You have to read the API very carefully to learn the difference between similar functions such as keyPressed( ) *and* keyTyped( ).

You will also find several variables with the same names:

- mousePressed
- keyPressed

Do not use these right now. We will look at them a bit later.

You can put anything you want in the body of these functions. The idea is to have your code respond to the user's input. Since you are the artist and programmer, the response is up to you.

Very Important!
We never call setup(), draw(), or any of these event functions
–
NEVER!!

We will write some code in class to explore this category of function and that code will be posted on the class code web page.

Also, if you want to respond to different keys being pressed, you need more syntax that we currently have. You can wait a bit or look up the following in the API and/or the book:

- if
- else

We will cover the if *and* else *in a few days and class notes and code will be posted then.*

If you have programmed before and are familiar with this syntax or you want to read ahead, feel free to explore and use what you know/learn in your code.

If you want to use the arrow keys for input, you must look up the variable keyCode in the API first.