Lessson\_1\_2\_Draw and Environmental Variables

Okay, so we can do this thing now, we can draw a circle in a window, great. For real world applications that knowledge won’t get you very far, but there are a few things that processing offers that ramp things up pretty quickly. The **draw()** function and Processing’s list of **environmental variables**.

draw()

So far, we figured how to draw things one time in a window we create in Processing. What if you wanted to continuously draw something in a window, and maybe move that ellipse around the screen? That is where the draw() function comes into play. How we use the draw function is very similar to how we use the setup() function.

void draw(){

//do something here

}

Void – declares that this is a function, and it will return nothing

Draw – is the name of the function

() – is where we could put arguments into our function, but the draw function never takes an argument for other reasons.

{} – to tell the computer where this function starts and ends.

Draw() and setup() are both what we call threads, which is actually quite a complicated topic, so we’ll just leave it at this. A thread is something that can continuously run simultaneously with other threads.

What is different about draw() is that it will calculate everything inside the function repeatedly 60 times per second at default. Setup() only happens once and then it’s over.

Setup() always has to happen before draw(), so it also has to always be placed before draw(). Now we have something like this.

void setup(){

size(500,500);

}

void draw(){

background(0);

ellipse(250,250,20,20);

}

If you run this code in Processing, you will get the same sort of thing we got in the last lesson, but what you can’t see is that it is drawing the background() function and the ellipse() function 60 times per second!

Let’s take a moment to divert to another concept really quickly before getting back to how cool this can be.

Environment Variables

Processing has yet another awesome feature! In the background, Processing is also recording a bunch of other data and performing some simple calculations. This data that Processing takes care of for you are collectively called environment variables. One thing that is very useful for testing is being able to use the mouse coordinates to control your sketch, as you want to tune in constants on this big, bad mutha-fucka of a function you will eventually be creating, being able to control them dynamically to understand what exactly is happening can be very important.

Processing gives us the mouse coordinates by default! You can call is this way.

mouseX – gives you the x coordinate of the current position of your mouse

mouseY – gives you the y-coordinate of the current position of your mouse

And together they represent the location of your mouse in two dimensional space!

Let’s go back to this idea of the draw function. If we want to have this ellipse that we are drawing on screen 60 times per second follow the position of the mouse, we can do this.

void setup(){

size(500,500);

}

void draw(){

background(0);

fill(255);

ellipse( mouseX, mouseY, 20, 20);

}

Broken down step by step, this is what this text means to processing.

* Run the setup function
  + Create a window 500 pixels wide, 500 pixels tall
* Run the draw function 60 times per second
  + Set the background to black (0)
  + Set the fill color to white (255)
  + Draw an ellipse on screen with the fill color, at the location of where the mouse is at a size of 20 x 20 pixels.
  + repeat

Now the thing follows your mouse! Holy balls, magic.

Basically, this is the whole lesson, but I would like to encourage you to play with this new thing a little bit. Try putting this background function back in the setup() function and deleting it form the draw function. What happens? Why?