

Introduction to Processing

Class #5: 10/17/2007

Taught by Ms. Madsen, Ms. Huhn & Ms. Yen

Course website: >> web.mit.edu/mish/www/processing

Processing website: >> www.processing.org

Email the teachers: mish@mit.edu, ahuhn@mit.edu, cyyen@mit.edu,

To run the following code, just replace everything in **bold** with names or numbers.

```
void setup() {  
  
    size(width, height);  
    color yourColorName;  
    yourColorName = color (redValue, greenValue, blueValue);  
  
    background (yourColorName);  
  
}  
  
void draw() {  
  
    rect( x_start, y_start, width, height);  
    ellipse(x_start, y_start, width, height);  
    for (int x=0; x<5; x++){  
        rect(0, x*20, width, height);  
    }  
}
```

Looking at variables and the "For" statement

Variables are named pieces of information. We're going to learn about Strings and integers (or "int"s.) We can declare variables like this:

```
String yourStringName = "some text goes here";  
int yourIntName = 49483738;
```

Here's what "for" statements look like:

```
for (int x = 0; x<3; x++) {  
    some code here;  
}
```

And here's what "for" statements act like:

```
int x;  
x=0;  
some code here;  
x=1;  
some code here;  
x=2;  
some code here;
```

Here's an example:

```
void setup() {  
    size(300,300);  
}  
void draw() {  
    for (int x=0; x<5; x++){  
        rect(0, x*20, x*5, 10);  
    }  
}
```

Animation using the "loop()" command

Animation in Processing is pretty easy, especially compared to a lot of other programming languages. The basic idea is this:

- 1) Declare a variable *not* in a method, so that you can access and change it in different methods.
- 2) Use that variable in the *draw()* method, with a shape or a color.
- 3) At the end of *draw()*, add some code that will change your variable.
- 4) Put the "loop()" command into the *setup()* method: this will run your *draw()* method over and over, changing the variable over and over.

Example! Here's how to move a circle down the screen:

1. First, we'll make a variable, *distanceFromTop*, that says how far the circle will be from the top. (This variable will start at 0 – the circle starts at zero distance from the top.)
2. We will also move our background color statement outside of "setup()" and put it in our new format:
color **backgroundColorName** = color(**redValue**, **greenValue**, **blueValue**).
3. Then we'll make a circle in "draw()" – and we'll use *distanceFromTop* as our second argument. We will also use the "background" command right before our circle, so that every time the loop runs, the background will be set again (and we won't see our old circle.)
4. Then at the end of "draw()", we'll say `distanceFromTop++` - which means "*distanceFromTop* = whatever-it-was + 1."
5. Finally, we'll put the command `loop()` in our "setup()" method.

So this is what our code will end up looking like at the end:

```
int distanceFromTop = 0;
color backgroundColorName = color(redValue, greenValue, blueValue);

void setup (){
  size(width, height);
  background(backgroundColorName);
  loop();
}

void draw(){
  background(backgroundColorName);
  ellipse(x_start, distanceFromTop, width, height);
  distanceFromTop++;
}
```

Using the “if” statement to control your code

What if we want to run certain code only *if* something is true? Let’s use the “if” statement. We check a statement and then run the code inside the curly brackets only if that statement is true. Here are some possible statements we could check:

5>4: **true** 4>5: **false** true: **true** false: **false**
distanceFromTop > 50: **may or may not be true – but we can check!**

Here are some statements we *can’t* check using “if”, because they will never be true OR false: 4, “hello there”, setup(), etc.

As an example, let’s move the circle back to the top whenever the distance from the top is bigger than the height of the window (which will always, in Processing, just be called “height”.) In other words: if the distanceFromTop > height, let’s set distanceFromTop = 0.

```
if (distanceFromTop > height){  
    distanceFromTop = 0;  
}
```

Just put that in your draw() function somewhere, and whenever the function loops, it will check if the distance from the top is bigger than the height – in which case it will set that distance equal to zero for the next draw() loop.

Using the mousePressed method

We can use the mousePressed method, which is just like setup() and draw() – it’s its own method, not inside anything else. Whatever is inside it runs whenever the mouse is clicked in your Processing window. We get two useful variables – mouseX (distance of click from left side) and mouseY (distance of click from top.)

What if we want to reset the circle to be whatever height we click at? We should change distanceFromTop to mouseY whenever the mouse is clicked. So add this to your code, somewhere outside of the other methods:

```
void mousePressed(){  
    distanceFromTop = mouseY;  
}
```