

Introduction to Processing

Class #4: 10/8/2007

Taught by Ms. Madsen
Assistants: 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);  
  
}
```

What are some common colors?

Black: (0, 0, 0)
White: (255, 255, 255)
Red: (255, 0, 0)
Green: (0, 255, 0)
Blue: (0, 0, 255)

What are some other useful colors?

Cyan: (0, 255, 255)
Pink: (255, 0, 255)
Yellow: (255, 255, 0)

Looking at variables and the "For" statement

Variables are named pieces of information. We're going to learn about some more common types of variables: specifically, Strings and integers (or "int"s.)

We can declare variables like this:

```
String yourStringName;  
int yourIntName;
```

Then we can give those variables information like this:

```
yourStringName = "some text goes here";  
yourIntName = 49483738; // this number can be anything you want  
// as long as it's not a fraction or a decimal
```

Here's some shorthand for the lines above:

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

Here's a variable in a "for" statement.... And here's what "for" statements act like:

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

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

We can use "for" statement to run certain sections of code multiple times; we can also use the fact that our variable x will increase each time.

Some examples:

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

```
void setup() {  
    size(300,300);  
}  
  
void draw() {  
    color myColor;  
    for (int x=0; x<5; x++){  
        myColor = color (0, 0, x*50);  
        fill(myColor);  
        rect(x*40, x*30, 15, 15);  
    }  
}
```

Animation using the "loop()" command

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

- 1) Make 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 something like 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.

Here's how we move a circle down the screen from the top:

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 (){
  background(backgroundColorName);
  // everything you put here is still here.
  loop();
}

void draw(){
  background(backgroundColorName);
  ellipse(x_start, distanceFromTop, width, height);
  // everything you put here is still here.
  distanceFromTop++;
}
```