Some notes for Processing – Computational Art Workshop SBHS –
Zoë Wood zwood@calpoly.edu

Introduction to functions (for use in animating)
- functions are a group of commands, enclosed in curly braces with a name
- the “draw” function in Processing will allow us to animate our art

Consider this program:
```java
void setup() {
  size(200, 200);
}

void draw() {
  background(random(0, 255), random(0, 255), random(0, 255));
}
```
- it was two functions. One named “setup”, which is executed once and one named “draw” – run this code and tell me what you learn about the “draw” function.

Now add the command frameRate(4); to the setup function – ie:
```java
void setup() {
  size(200, 200);
  frameRate(4);
}

void draw() {
  background(random(0, 255), random(0, 255), random(0, 255));
}
```
We can use the fact that the “draw” function repeatedly executes (like a loop) to animate. Let's try it with variables. Consider:
```java
int i;

void setup() {
  size(200, 200);
  frameRate(4);
  i= 0;
}

void draw() {
  ellipse(i, i, 20, 20);
  i = i+10;
}
It should make something like this:

![Image](sketch_131025a)

If however, we want it to appear that there is only one ball – we can add a “background” command to the start of the draw loop like this:

```java
int i;

void setup() {
  size(200, 200);
  frameRate(4);
  i= 0;
}

void draw() {
  background(200);
  ellipse(i, i, 20, 20);
  i = i+10;
}
```
Functions can also help us organize our code. For example this code creates a function to draw a small pyramid:

```cpp
//This code just daws one pyramid
void setup() {
    size(300, 300);
    background(0);
    smooth();
    stroke(255);
}

void drawP() {
    fill(55, 27, 232);
    rect(20, 10, 20, 10);
    fill(43, 123, 255);
    rect(10, 20, 20, 10);
    rect(30, 20, 20, 10);
}

void draw() {
    drawP();
}
```
And now we can use transforms to animate our drawing:

// This code draws a pyramid that is slowly falling
int tx, ty;

void setup() {
  size(300, 300);
  background(0);
  smooth();
  stroke(255);
  frameRate(8);
  tx = 0;
  ty = 0;
}

void drawP() {
  fill(55, 27, 232);
  rect(20, 10, 20, 10);
  fill(43, 123, 255);
  rect(10, 20, 20, 10);
  rect(30, 20, 20, 10);
}

void draw() {
  background(0);

  pushMatrix();
  translate(tx, ty);
  drawP();
  popMatrix();

  tx = tx+4;
  ty = ty+2;
}
Or we can write code to get our drawing to follow the mouse:

```cpp
// This code draws a pyramid that follows the mouse
int tx, ty;

void setup() {
  size(300, 300);
  background(0);
  smooth();
  stroke(255);
  frameRate(8);
  tx = 0;
  ty = 0;
}

void mouseDragged() {
  tx = mouseX;
  ty = mouseY;
}

void drawP() {
  fill(55, 27, 232);
  rect(0, 0, 20, 10);
  fill(43, 123, 255);
  rect(-10, 10, 20, 10);
  rect(10, 10, 20, 10);
}

void draw() {
  background(0);

  pushMatrix();
  translate(tx, ty);
  drawP();
  drawP();
  popMatrix();

}