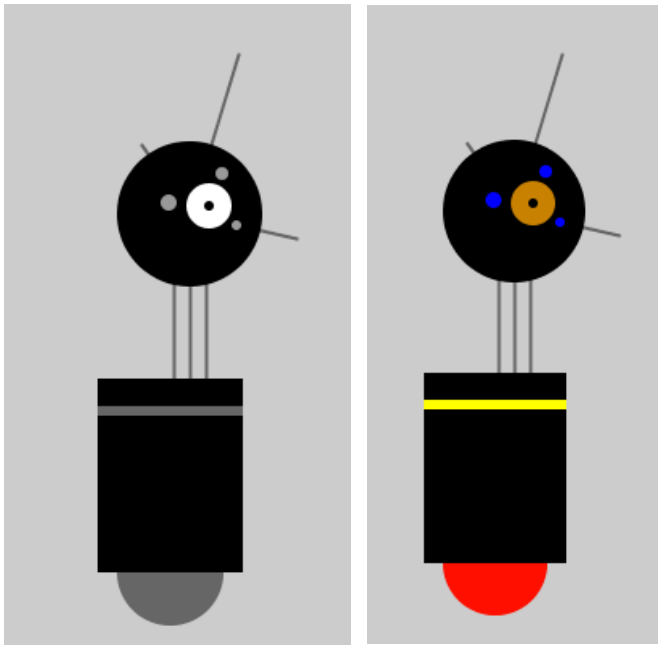
Lab Exercise 3: Practice with Processing

Goal:

The objective is to practice using the Processing language, and to learn the difference between a *static* program and a *dynamic* program. Hint: Static Processing code runs, and then stops. Dynamic Processing code keeps running. That makes it a lot more fun.

Step 1 Color The Bot

Grab the RF robot program (as Reas and Fry gave it in their book) from the CSE120 Class Calendar for this lab. Copy/Paste the code into a Processing window, and run it. The image you will see looks like figure (a). The first task is to make the robot more interesting by coloring it, like we did in class. One example is image (b).



(a) (b)

As shown in class, to color a robot part, you must (a) find the component in the program, (b) find a color using the color selector [Tools > Color Selector], (c) transfer the three RGB color numbers to the `fill( )` function just ahead of the component, and (d) try the program out to see that it is good. If it is not, this is the time to fix it!

You are required to color the gray parts of the robot to complete this step. You are welcome to add more color, but there is also more to do in this lab.

Step 2, Make The Bot Program Dynamic

The program you have written so far is static. That is, the Processing code runs, draws your beautiful robot, and quits. The Processing "engine," which is what we are calling the software that runs your program, goes to sleep.

We want it to be active, that is, to keep running, so we add a few more commands. Then in the final step (below) we will make the robot move. But, we need to make the program dynamic first. Notice the two programs below. The one on the left is the static robot, the one on the right is the dynamic version. Make the highlighted modifications.
Caution: Notice that both parentheses ( ) and braces {} are used in these changes.

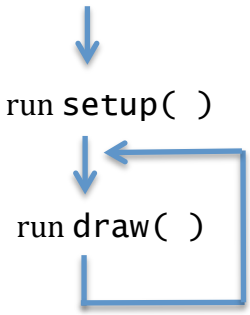
```
size(720, 480);
smooth();
strokeWeight(2);
ellipseMode(RADIUS);

//Neck
stroke(102);                // Set stroke to gray
line(266, 257, 266, 162); // Left
line(276, 257, 276, 162); // Middle
line(286, 257, 286, 162); // Right

// Antennae
line(276, 155, 246, 112); // Small
line(276, 155, 306, 56);  // Tall
line(276, 155, 342, 170); // Medium

// Body
noStroke();                 // Disable stroke
fill(102);                  // Set to gray
ellipse(264, 377, 33, 33);// Antigravity Orb
fill(0);                    // Set to black
rect(219, 257, 90, 120);  // Main body
fill(102);                  // Set medium gray
rect(219, 274, 90, 6);    // Gray stripe

// Head
fill(0);                    // Set to black
ellipse(276, 155, 45, 45);// Head
fill(255);                  // Set to white
ellipse(288, 150, 14, 14);// Large eye
fill(0);                    // Set to black
ellipse(288, 150, 3, 3);  // Pupil
fill(153);                  // Set to gray
ellipse(263, 148, 5, 5);  // Small eye 1
ellipse(296, 130, 4, 4);  // Small eye 2
ellipse(305, 162, 3, 3);  // Small eye 3
```

```
void setup () {
  size(720, 480);
  smooth();
  strokeWeight(2);
  ellipseMode(RADIUS);
}

void draw () {

  //Neck
  stroke(102);                // Set stroke to gray
  line(116, 207, 116, 112); // Left
  line(126, 207, 126, 112); // Middle
  line(136, 207, 136, 112); // Right

  // Antennae
  line(126, 105, 96, 62);   // Small
  line(126, 105, 156, 6);   // Tall
  line(126, 105, 192, 120); // Medium

  // Body
  noStroke();                 // Disable stroke
  fill(102);                  // Set to gray
  ellipse(114, 327, 33, 33);// Antigravity Orb
  fill(0);                    // Set to black
  rect(69, 207, 90, 120);   // Main body
  fill(102);                  // Set back to gray
  rect(69, 224, 90, 6);     // Gray stripe

  // Head
  fill(0);                    // Set to black
  ellipse(126, 105, 45, 45);// Head
  fill(255);                  // Set to white
  ellipse(138, 100, 14, 14);// Large eye
  fill(0);                    // Set to black
  ellipse(138, 100, 3, 3);  // Pupil
  fill(102);                  // Set to gray
  ellipse(113, 90, 5, 5);   // Small eye 1
  ellipse(146, 80, 4, 4);   // Small eye 2
  ellipse(155, 112, 3, 3);  // Small eye 3
}
```

The new code groups the instructions for the robot into two *functions*, so they can run separately, the setup( ) function and the draw( ) functions.

Tip: In programming, functions are written with a pair of parentheses after the name, *f*( ).

The setup( ) function runs once, when the Processing engine starts. It defines things like the size( ) of the canvas and other things that must be known at the start. Our setup( ) has four instructions. After the setup( ) runs, then the draw( ) function runs over and over again. This redraws the image. If we have made no changes to the image, it looks like it did before. It looks static, but is really dynamic and unchanging. (We'll make changes in a moment.) The following diagram shows what is happening when the Processing engine runs a dynamic program.



The way to read this diagram is that the Processing engine starts running setup( ), finishes it, starts running draw( ), finishes that, and then "goes around, and runs draw( ) again, again and again … ."

Step 3, Move the Robot

This is more fun! Add two more instructions to your dynamic program:

```
background(205);           //Draw background
translate(mouseX,mouseY); //Draw robot where mouse
```

The instructions go inside of draw( ) – that is, after the line that starts the `draw( )` function – and before the `//Neck` comment (see blue arrow in program above).
Caution: Notice that "mouse" is written in lower case letters, and the X and Y are capitals; this is required. Run your program and move your mouse. You will see the robot move around the screen under mouse control. The cursor (arrow) is at the upper left of the figure because the figure is drawn with reference to that point, i.e. that is an imaginary point (0,0).

Challenge

Remove the `background(205);` line from your program, and run it again. Can you explain what is happening, and why it is different with/without the background( ) function? Hint: Check the diagram above.

Wrap Up

You have colored the RF robot, learned the difference between static and dynamic programs, written a dynamic program, and made the robot follow your cursor.

Turn-In

Restore the background( ) function.
Prepare a (Word) document with the following content: (a) Your explanation of the Challenge question above, and (b) your active colored robot program text. Turn them in in the class dropbox.