Homework 4: The Raes-Fry Robot

Goal: You will familiarize yourself Processing by working with the Raes-Fry Robot discussed in lecture. You'll make it *fly*. Besides getting experience with Processing, you will learn how a design becomes active. It's an idea we will use all term.

Part 1: Color the Robot

In class, we colored parts of the R-F Robot. Change the color of **at least** four of the robot's parts. You can color it as much as you'd like! (*Use the copy of the program that comes with this assignment,* Robot.txt*, since it has small additions needed later. Open the file, copy it, and paste it into a new Processing window.*)

Recall that we have several functions available to specify colors:

```
        background ( r, g, b );
        stroke ( r, g, b );
        fill ( r, g, b );
```

All of Processing's functions are described in the references. Navigate help > reference.

Working with the RGB colors is easy in Processing. There is a Color Selector that allows you to find the RGB values of exactly the hue you want. Navigate Tools > Color Selector.

Part 2: What to Notice

Now that you have a beautifully colored robot, it's time to make it move.

In the Robot.txt file that you have been using, you will note that it contains a few extra lines that the original R & F Robot design in the book didn't have. These define two functions used to animate the robot, called setup( ) and draw( ).

At the start of the file the lines

```
        void setup ( ) {
           normal Processing operations go here
        }
```

are visible. Locate these instructions. The instructions of this function run one time at the start of the computation, and are not run again. They define things like the window size.

After the setup( ) function the lines

```
        void draw ( ) {
           normal Processing operations go here
        }
```

are visible. Locate these instructions. These instructions run any time something changes, such as the mouse is repositioned by the user.

Translate

The function translate( ) adjusts the position of items on the screen. In geometry "translate" means "move to a new position without other changes." So, the function is

```
translate ( x, y );
```

The function's parameters tell how much to move in the x direction and y direction. So,

```
translate ( 10, 20 );
```

draws the following shapes ten pixels further right (x), and twenty pixels further down (y). We will use translate to move the robot.

mouseX and mouseY

When your mouse pointer is on the Processing window, the system knows where it is. It tells you the position by telling how many pixels the mouse pointer is from the **upper left corner, which is, 0,0**. The first coordinate position is x, making 100,0 a position that is one hundred pixels to the right along the top edge of the window. The second coordinate position is y, making 0,200 a position two hundred pixels down the left edge of the window. If the window is 300,400 pixels in size, then the lower right corner pixel is 299,399.

The Processing system tells you where the mouse is by using two special variables, mouseX and mouseY. (Recall that Processing is case sensitive, so the lowercase-uppercase matters.)

Making The Robot Move

Our plan is to animate your beautifully colored robot by first running the setup( ) function. That comes automatically because that's how Processing works. Then in the start of the draw( ) function we add two instructions:

```
background(205, 205, 205);
translate( mouseX, mouseY);
```

The first defines the background to be a light gray color. The second tells the Processing system to draw any figures translated by the mouse position. That is, the system adds the amount of the mouse's x and y coordinates to the x and y coordinates of any figure it draws. So, by drawing the robot next, we get it to be positioned under control of the mouse. And as the mouse moves around it moves. Sweet!

Modify your colored robot to move around as just explained.

Wrap Up.

You've programmed Processing for the first time, and made the R-F Robot move. You've learned that the setup( ) function runs at the start of the computation, and the draw( ) function runs whenever anything changes, like the mouse position. And, you've learned that the pixels of the screen are counted with a coordinate system that starts with 0,0 in the upper left corner of the window.

Rename a copy of the .pde file with your name, and submit it into the class drop box.