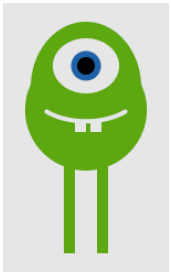
Homework Exercise 7: Mike Jumps

The goal of this assignment is to practice controlling screen activity using Boolean variables and if-statements.



Getting Started

We activate Mike from Monster University fame. Load into a new Processing window the basic function mike( ) that is included with this assignment, and run it. You should see Mike. Check the code to see how the different parts are drawn, and to show that you've done that, add comments to the lines with rect( ) and ellipse( ) calls saying what they are for.

Eye Movements

We want Mike's eyes to track the mouse left and right. We do this by declaring a float variable, which I will call focus and initialize to 0, but *you must use a different name*.

My code to make him look left is

```
if (mouseX < 145) {
  focus = max(focus - 0.5, -10);
}
```

and works like this: If the mouse's x-coordinate is positioned in the first 145 pixels, it's left of Mike's body. So, I decrease my focus variable by half a pixel, and I limit the change to -10. More than that is too much! I put the code just before drawing the eyes, though there is some flexibility in where this can go. Finally, I changed the code for the blue and black parts of his eye to add focus to the x-position. Adding in a negative focus decreases the x-coordinate, making his eye look left when the cursor is left.

Part 1: Add code to make Mike's eyes track. This means you need to implement your version of my code. And you need to implement the code to track the cursor when it's to the right of his body, which will also use the focus variable and is very similar to mine.

Smarter Eye Movements

The problem with Mike's Eye is that it starts out looking left because the cursor begins at 0,0 by default. So, let's don't even worry about moving his eye until we get a click from the user saying that we want him to move.

To do this, we begin by declaring a Boolean variable. I will call mine started and initialize it to false, but you will have to choose a different name. As usual, this declaration goes at the top of the program.

Next, declare a function called mousePressed( ), which the Processing Engine will run when the user clicks the mouse. (It will be highlighted blue if you did this right.) What should it do? Set started to true, of course. That's all.

Finally, it's necessary to make changes to the focus computation to be responsive to the started click. To do that, I simply modify the code I showed before to be

```
if (started && mouseX < 145) {
  focus = max(focus - 0.5, -10);
}
```

Notice what the if-test asks. It says, if started is true and (that's the && operator) the mouse is to the left, then change the focus variable. Otherwise, do nothing. So, until started is set to true by a click AND the mouse is left, the eye won't look left. Clever, no?

Part 2. Implement the logic just described, including the declaration, the new function, and the mods to the previous code for both eyes.

Monsters Surprise

Mike needs more action that just looking around, so we will make him crouch and jump. In this section we set up the crouch.

The motion will be controlled by a float variable that I'm calling posish and initializing it to 0, but you should call it something else. Change the call so it's

mike(posish);

Just to see how things are working, add a line following the call
posish = posish + 0.5;
Watch it run. Notice why the behavior is as it is … amt hasn't been added to the legs. (Don't fix it now.)

Next we're going to stop his crouching using an if-statement. Following the assignment to posish, add an if-statement which checks to see if posish is more than 70, and if so, it sets posish back to zero. It does nothing otherwise. Check that out!

Part 3. Verify that mike( ) has been updated with these changes in your code.

Mike Jumps

Now change the if-statement so that if posish is more than 70, it sets posish to minus 150. Weird! How does he *do* that???

So, now we need his legs to travel with him. To see what we need to do, add amt to the y-coordinate of each leg. We've ruined the crouch. So, we fix this, because we want him, as he crouches, to overdraw his legs. So, we don't want his legs to start below their beginning position. So, we put in a min( ) function for the y-coordinates for the legs. (We saw min( ) in the second Simpsons program.) The two values we need to choose between are the leg's current position (370) and anything less than that, which we'll get after the jump of -150.

Part 4. Add the min( ) call needed to keep the legs with his body and still allow him to crouch. He should now crouch and jump.

Monsters Wait For The Right Moment

Mike shouldn't start jumping until after he has looked around. He'll start looking after a mouse click. Let's have him start jumping after a second mouse click.

This will require the declaration of a second Boolean variable, which I'll call jump and will initialize to false, though you need to choose another name. In the mousePressed( ) function from before, I add a new first line, which is an if-statement testing if started is already set. If it is, then set jump to true. If not, do nothing. (The next statement will set started.)

Finally, the statement that we added to increment posish by 0.5, should only be done if jump is true. If it is true, increment posish; if not, do nothing.

Part 5. Implement the two-click start to the jumping.

Challenge: Mike Accepts The Gravity of His Situation

Before the crouch, Mike should descend faster than when he's crouching.

Part 6. Add a single if-statement so Mike descends twice as fast as he does when he goes into his crouch.

Wrap-up. We've used Boolean variables and if-statements to control the action on the screen. The logic used here was simple, but it covers most of the cases one needs to make the screen work the way we intend.

Turn-In. Turn in your documented program incorporating all six parts to the class dropbox.