CSE120: Computer Science: Principles

Lab Exercise 6: The Mice And The Owl

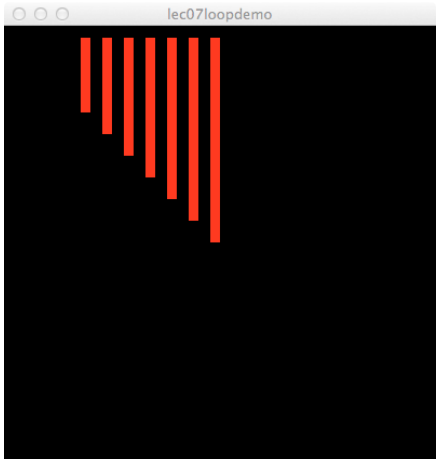
Goal

The objective is to get more practice controlling action on the screen. The programming in this exercise is the kind you've seen before. The goal is to work out the logic of the computation.

for-loop practice

Create a practice program using this code from lecture 07:

```
void setup( ) {
    size(800, 400);
    background(0);
    fill(255,0,0);
}
void draw ( ) {
    for (int i = 0; i < 5; i = i + 1) {
        rect(10+20*i,10,10, 10);
    }
}
```



Spend a few minutes playing with the `for`-loop: the limit value (5), and the parameters of the `rect( )` function. Make the image at right.

Assignment

The plan is to draw an owl and some mice, and then have the owl eat the mice.
NOTICE: *This assignment gives all of the steps, but it doesn't say how to program everything … you should expect to fill in some details that have been left out.*

Base Program

Open a new Processing program, and copy/paste the mouse.txt code linked to this lab on the Calendar page. Notice that this is actually five functions: `setup( )`, `draw( )`, `pick( )`, `owl( )`, and `mouse( )`. Run the program and notice the mouse.

for-loop programming

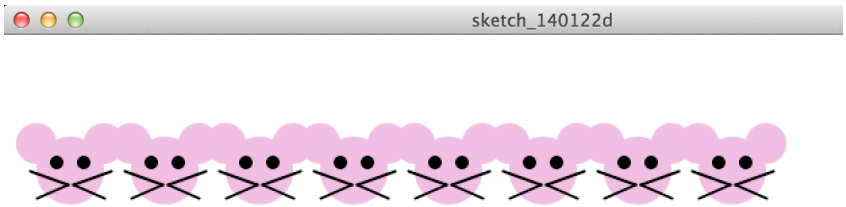
Create a function called `mouse8( )` that is a row of eight mice using a `for` loop. The function will draw a row of 8 mice with one statement, which is a loop of the form

```
for (int i = <starting value here>; i < <limit value here>; i = i +1) {
    <a call to the mouse function goes here>
}
```

the start of the control portion of the loop. You need to fill in the starting value of `i`, the limit value of `i`, and the call in the body of the loop; that is, the code that is repeated. The value of `i` will be tested to be less than (<) the limit value, meaning that this is the first value that would be too big. So, for example, if the starting value is 0 and you want to make eight mice, then the limit must be 8, because you think about the mice as being numbered, 0, 1, 2, 3, 4, 5, 6 and 7. Consider what parameter(s) you will need – for example to position the row – and then write `mouse8()`, so that mice are placed every 70 pixels.

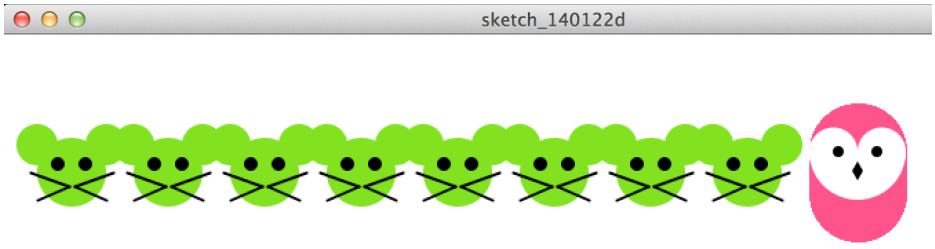
The mice should have a random color, but not flicker. So, you need to declare (at the top of the program) a variable of type ____ . Suppose it's called `tint`, then pick( ) its value in `setup( )` so it doesn't change. Also, because this is a more complex program, you should generate the mice in the `setup( )`, too.

At this point your program looks like this (except possibly for the color).



The owl appears

In `draw( )`, call the `owl( )` function (using a random color that doesn't flicker like the mice) so it appears at the right end of the row after the last mouse. It might take a little fiddling (or calculation) to get the owl in the right place. We give this position a name; call it `danger` and declare it as an `int` at the top of the program. Use the variable `danger` in the call to `owl( )`. This is what it looks like:



if-statement

Now we will use an if/else statement to make a choice. Each time that `draw( )` is called it will either erase the owl, or redraw it. For the purpose of controlling the two cases, we use the `mousePressed` predicate, as in

```
if (mousePressed) ...
```

You will fill in the two choices for the if/else. In one choice (true) you will erase the owl by drawing a white rectangle on top of it. (To work this out, you may want the `stroke(0)` set for positioning, but once you get it positioned, you will remove that.) Your "true" code, that is, the statements in the braces that you run when you erase the owl, will require you to set the `fill( )` and then draw the `rect( )`. You should be able to run the program, click the mouse and have the owl disappear.

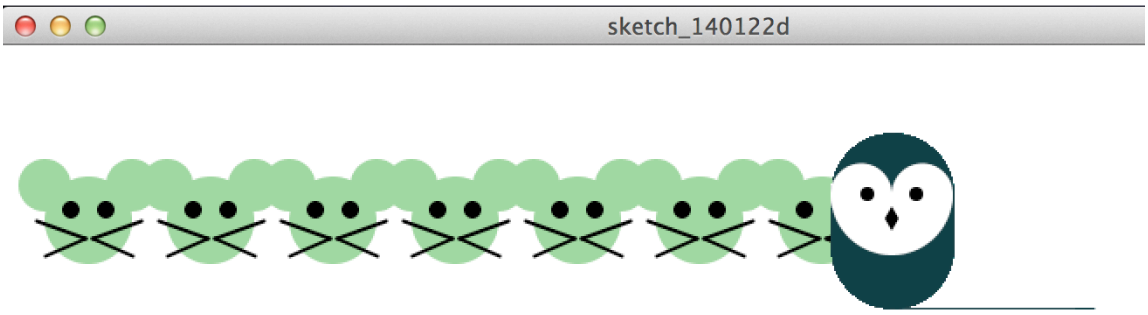
The other choice (false), implemented with an `else` statement, redraws the owl at the `danger` position. The owl will disappear and then reappear Try out your `if` construction.

Supper

It's now time to make the owl consume the mice. Expand your `if` so that

a) after erasing the owl, it subtracts 2 from `danger`.
b) redraws the owl at the new `danger` position, moving it left as you know.

The result is that when you hold down the mouse, the owl should move across the screen and "consume" the mice. The picture looks like this:



The owl leaves a line as it moves across. This can be fixed, but I leave it now to prove how the owl is really moving.

Wrap Up

The program draws a row of mice using a `for`-loop, saving us considerable typing. Using `if`- statements, we were able to develop some simple logic based on the `mousePressed` flag's Boolean value. Your logic chose between true (mouse pressed) in which it erased the owl and redrew it just to the left, and false (mouse not pressed) in which it just drew the owl.

Be sure the portion of the program that YOU wrote is commented, and submit it to the class dropbox.