CSE120: Computer Science: Principles

Homework 11: Looking At The Code

Goal: The basis of this assignment is a small "game" like Lightbot 2.0 … so we'll call it Ninjabot. You get all the code for the game, you will play it, study it, and answer questions about it. This is making the point that "code is not gibberish" … you can understand it. It also shows how you can modify existing programs to make something better for yourself.

The Plan

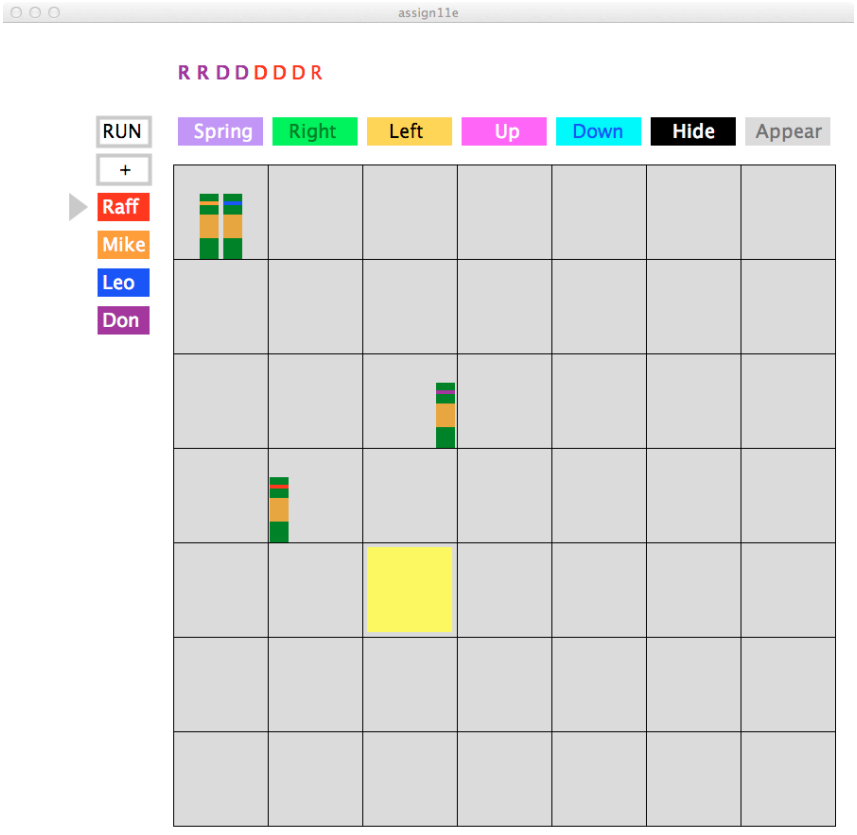
Grab a copy of the code from the Calendar. Also **load the font "LucidaSans-20.vlw".** In the next sections, we will

1) Explain how the game works
2) Explain at a high level how the code works
3) Illustrate a sample question and how to answer it
4) Present a series of questions about the program for you to answer

You will be preparing a text document (e.g. Word) of your answers, not a program. But you will be doing a little programming. Also grab the line-numbered version, too.

The Game

Here is the game board. It's similar to like Lightbot. You pick a Ninja, which is highlighted by a triangle pointer, and then you give commands, such as RIGHT, which means for the Ninja to move 1 square right. The commands are listed at the top, and



when the program Runs, the code follows the instructions just like in Lightbot. Notice that the instructions are color coded for each Ninja. Verify that you see how the Ninja's moved from the upper left corner (starting place) to their present positions in the figure. The goal is to move the Ninja's to the yellow square, which is positioned at random. Once the program has run, it is possible to continue giving the turtles instructions by clicking on the Plus button. Play it. (Note, this is not a polished, production-grade game; its only a homework exercise. It's OK, but it could be made better.)

How It Works

The program is written in Processing. Take a look at the code now. Notice these parts as you look through the file:

- A series of declarations defining values and variables used in the program.
- A `setup( )` that specifies a few basic items, and then does the initial layout of the board.
- A `draw( )` function of just a few statements which perform the game.
- A series of functions that are either called in `draw( )` or are called by functions called in `draw( )`, etc.

Except for the loaded font, there's nothing more. *Play the game for at least 5 minutes.*

To see how it works notice that `setup( )` just lays out the canvas. Recall that the `draw( )` function is being called over and over again. If the image looks static – nothing is happening – then it's still being completely redrawn every few milliseconds.

Continuing with the `draw( )`, `showselect( )` simply prints out the program at the top.

The rest of `draw( )` is three if-statements followed by `execute( )`. These are the only active parts of the program. To understand how they work, notice that only four things can be done:

- Click a Turtle Name, which is `clicktype == 1`
- Click a Command, which is `clicktype == 2`
- Click the Run/+ buttons, which are the `clicktype == 3`
- Execute the actual commands, moving the turtles around

Find these four parts in `draw( )`.

For the first two and the beginning of the third parts, all of the action happens in the `mousePressed( )` function which you will study later. The remainder of the third step is fetch instructions, decode instructions, and assign work to the right turtle to do it. Notice that if the next instruction is for a turtle that is already busy, the assignment waits until the turtle is done with his work.

The `execute( )` function simply does the work once it has been assigned; note in each call, it just moves things along one step.

The workings of the program will become clearer as the discussion continues.

Sample Question

This assignment asks a series of questions about the Turtlebot program, which you will answer in a Word document. Here is an example of a question and (part of) the answer.

Q. Suppose we add a new turtle button for a turtle named Bud. What changes would have to be made from the original program?

A. A new call to the `tbutton( )` function must be added in the `setup( )` after line 59. [Refer to the numbered copy of the original code.] The new call has the form

```
tbutton(100,340, budc, color(255), "Bud");
```

where "`budc`" is the color of the Bud button, defined in the declarations after line 20.

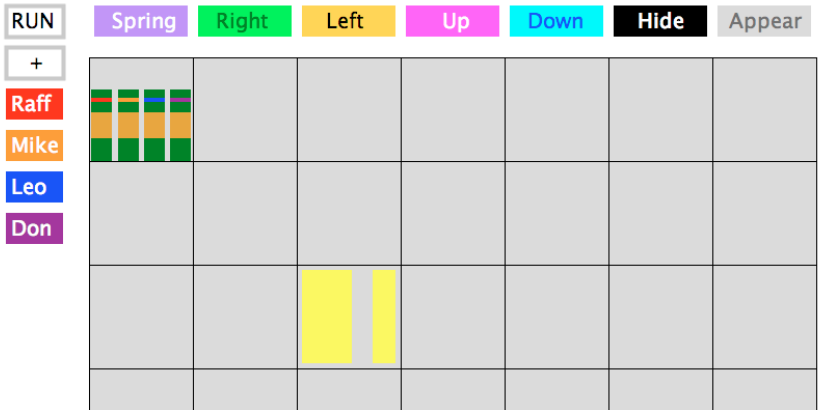
There are more changes that need to be made to add a turtle to the processing of the program, of course, but this is all that's necessary to show the button.

Why is this a good answer? It is explicit … it tells what needs to be done, and someone could do it to check your answer. Did you try it? Also, it showed the exact code.

Questions

Answers should include the exact code wherever appropriate.

1. This screen was produced by running a program; what clicks created this result,



   and why is the yellow square so strange?

2. Calling the `grid( )` function is the first thing that happens in `draw( )`. Explain what happens when `grid( )` is moved to the last line of `draw( )`, and explain why it happens.

3. To change the UP button, which is now pink or magenta, to a light blue, say `color(240, 240, 255),` with black writing, what would change in the program?

4. When the user clicks on Leo, which lines in the `mousePressed( )` function run? And what variable (besides `i`) has its value changed, and to what?

5. Assuming we have added the "Bud" button, as described in the Sample answer above, what changes are required to the program so that Bud could be *selected*?

6. Using the mouse to select turtles and to choose commands causes a lot of annoying mouse motion. What code (included it in your answer by copy and paste) would be needed to select turtles by typing keys? Notice that adding key input does not mean that you need to remove the ability to click a turtle; both can be available.

7. Mike thinks he's so tough and important that his commands should be listed as if they were placed on a pedestal:

   What changes to the program are needed to accommodate Mike's ego? Show your commented code, and an illustration of it operating.

8. If Mike is programmed to move 3 squares down, and three squares right, what values would `guy` and `inst` variables have? Where does the content come from?

9. The + button doesn't do much now. When more code is added to the command list, the new instructions are executed, which continues the action, because the `pc` – the marker that tells which command is being executed – doesn't get reset to the start of the program. Add `pc=0;` to restart execution at the beginning. This should happen at the very end of the `mousePressed( )` function, where the + button is recognized and `clicktype=1;` is set. In addition, at this place you must:
   - Erase the turtles in their current position.
   - Reset the positions of the turtles to square 1.
   - Redraw the turtles in their new positions.

   Make those changes and notice that you can now restart the computation. Answer by copy/pasting the code you added – just the relevant lines, commented – into your document.

10. Explain how the `spring` variable makes a turtle jump, and include in your explanation how it avoids making all turtles jump.

Wrap Up You have studied a "complex" program and figured out how it works – maybe not completely, but the basic idea. You can navigate the code, and identify the place where a specific activity happens. You recognize that this 400 line program is not gibberish, but is a careful description of what a computer must do to implement a game.