JavaScript: Lab 4 (from Tutorial 5)

Due Thurs, Nov 29 (100 pts)

Your Turn 1:
1. (18 pts) Make sure you have the coin toss game running and you understand it! (Steps 0 – 29, p 1-4)
2. Make sure you’ve got the random width and height working, steps 1-6 page 5
3. Make sure you’ve got the setToRed and setToBlack functions working with setTimeout (pages 7 and 8)

Your Turn 2 (for the final project):
1. Make sure you’ve got the MoveLeft, MoveRight, MoveUp and MoveDown functions working for your game’s hero
2. Make sure you’ve got the ChangePos function working with your goodie, and make sure you’ve got it working with the setTimeout call inside it so that it moves automatically to random places.
3. Make sure you’ve got the MoveAcross function working with setTimeout so that your bad guy moves across the screen automatically
4. Make sure you’ve got the score working so that if your hero moves over the goodie (with MoveLeft, MoveRight, MoveUp, and MoveDown), the eatcount goes up by 1 and the paragraph’s innerHTML is changed to the new eatcount

Final Project:
For your final project, you will be adding to the functions you’ve got so far. Here is what my final project was: I had a zombie (my hero) wandering around a wasteland trying to catch and eat cute chipmunks (which moved to random places on the web page). Whenever the zombie caught a chipmunk (by moving over the chipmunk), an alert box pops up saying that you ate another chipmunk, eatscore increases by 1, and the paragraph updates. Also, a mini zombie chipmunk forever follows the zombie hero around from then on. If the eatscore gets to 10, you win. If, however, the zombie gets hit by the bunnymonster (movingAcross automatically), then an alert box pops up saying you lost and you lose the game.

You can pick your own scenario. I’ve just been watching too much walking dead.

What you still need to add:
In your html, create the scene you want by adding a background image - mine was a barren wasteland, dead trees, abandoned cars, etc. Make sure all images are at a very low z index so they’re below the hero, the monster, and the goodie. Make sure the paragraph for the eatscore is easy to see and make sure the buttons for moving the hero are somewhere easy to use.

Once you have the board looking the way you want, you’ll have to modify the js as follows:

1. Start function:
   You want both the goodie to start moving around randomly and the monster to start falling simultaneously when the game starts. You can either have both starting when you click a start button, or when the web page loads in the browser. If you want a start button, you should add a button to your web page somewhere. Otherwise you can just change the function called in the body onLoad command (above).
   The problem is that with either a button or with the onLoad, only one function in your JavaScript can be called. Yet you want 2 (or possibly more) functions to start simultaneously. The solution is a startall function in your JavaScript
   a. In your JavaScript, create another function. This is a very simple function, and I called it function startall().
   b. Inside your startall function, call your other two functions. My starting function looks like this:
   
   ```javascript
   function startall() {
     ChangePos('c1')
     MoveAcross('m1')
   }
   ```
   c. Now in your html code, make sure you either create a start button that, when clicked on, calls the startall function, or you can add to the body tag as follows:
   ```html
   <body onload = "startall()">
   Which will start those two functions automatically when you load the web page..
   d. Save and test. Both the monster and the goodie should start moving on the web page automatically.

2. Adding the goodie to follow the hero after the hero has “caught” the goodie (aka an army of zombie chipmunks):
a. In your MoveLeft function, where you check to see if the hero is over the goodie, add code that will make the goodie follow the hero after the hero catches the goodie (e.g., make a zombie chipmunk follow the zombie when the zombie eats the chipmunk) as follows:
   i. First, inside the if condition that checks if the hero is over the goodie in the MoveLeft function, set a variable to the innerHTML of the div of your hero (‘z1’) using document.getElementById. That variable will hold whatever tags are inside the div. At first it will only hold the hero image tag, but as each goodie is acquired, it will hold the hero image tag and more and more goodie image tags.
   ii. Next create a variable that holds an image tag with your goodie (or a version of it) as the src. Give the image tag a width and a height, and a height. Mine looked like this:
   k="<img src = ‘zombiechipmunk.gif’ width = ‘75’ height = ‘90’ alt = ‘zombie chipmunk’>”
   iii. Add that image tag variable to the variable containing the innerHTML of the div.
   iv. Set the div’s innerHTML ot the variable that holds both the old innerHTML and the image tag.

b. Repeat for your MoveRight, MoveUp and MoveDown functions.

c. Save and test. Now every time your hero goes over a goodie, the goodie should follow the hero around as you move the hero.

3. Winning the game:
   a. You currently have an eatscore variable that goes up by 1 each time the hero is over the goodie. Add an if condition around that eatscore increase as follows:
      If the eatscore is < 10, increase the eatscore and change the paragraph to the new eatscore (currently you also have an alertbox popping up – you can leave that or get rid of that if it’s slowing down your game)
      Otherwise (else) have an alert box pop up that says, “You win!”

4. Losing the game:
   One final part to the game: If the hero gets hit by the monster, the game should end and the user should get a message, ‘You Lose’ in an alert box.

   You will again need to check if the hero and the monster are in the same place. You always check for this in the object that is moving the most – in the previous example, the hero was moving faster than the goodie, so the code to check if the two objects were over each other was in the move hero functions. In this case, the monster is moving faster, so the check should be in the MoveAcross function.

   a. In the moveAcross function, after the monster has moved but before the setTimeout, add a check for whether the monster is over the hero by checking the hero’s left and top variables with the monster’s left and top variables. If the monster is over the hero, use an alert to say, “You Lose”, change the paragraph score’s innerHTML should change to You Lose, and change the score variable to 20 (you change it to a really large number so that everything stops moving about).

   b. Save and test. Again, you’ll have to play with the range to make sure that the monster is over the hero when the You Lose alert pops up.

That’s it for the base version! There are a bunch of things you can do to make the game a bit nicer if you like:

1. The buttons are a pain in the neck. Follow the keypress tutorial so that you can use keypresses to move the hero.
2. After you move the hero over the present, the goodie gets added to the “army of goodies” behind the hero. But, as the game stands now, the goodie also remains where it is so you can run over the same goodie in the same location again and again and get a bunch of goodies from that one goodie. To change this, once a goodie has been acquired, you can set its top and left variables to large negative numbers and move the goodie to that location (so that it, in essence, disappears off the screen). Now, technically, a better coding way to do this would be to change the present’s visibility property to “hidden” as follows:
   document.getElementById(‘pres1’).style.visibility = “hidden”

   and then you’d have to a: change the visibility back to “visible” in the changePos function and b: check to see when you run the hero over the goodie that the visibility property isn’t hidden. That would be more elegant code. But it’s quicker and works just as well to just move the goodie off-screen temporarily.

3. The way the game is set up now, each time you add a goodie, you add a goodie that is the same size. You could generate a new random number for the width and a new random number for the height each time. Then the goodie army would have random size goodies.

4. You could add other goodies or other monsters that run simultaneously, and have them work with the hero as well.