CMSC 100 – Fall 2010
Programming Assignments #2 and 3
Handed out Thursday, 9/23/10
Due Thursday, 10/7/10 (PA #2), and Thursday, 10/21/10 (PA #3)

Notes:

- You should submit your Scratch program (the ".sb" file) via Blackboard, by the end of the due day (11:59PM)
- For assignment 3, you should also submit a written description of (1) what your program does (i.e., what we should see when we run it), (2) any instructions for running it (i.e., what we should click on or type in to make it work), and (3) **how the program satisfies the requirements listed in the assignment below**. (i.e., what are the sprites, conditionals, loops, etc.) Please make this last part (program requirements) very explicit (e.g., one bullet per requirement, such as "Sprites: My world includes a turtle, a spaceship, and a chair.") **This description should be submitted as a hardcopy in class**, also no later than classtime on the due date.
- The third assignments are very open-ended. The purpose is to get you to try out the different things that Alice can do, and develop your own "world" and storyline.
- Your grade will be based on whether you satisfy the requirements listed below, whether your program works "as advertised" in the description you submit, and the effort you appear to have put into the design of the program (creativity, trying out different Scratch concepts, etc.) Please note that in some cases, your program may not run exactly the same way on our machines as it does in yours, so the description document is very important for us to know how your program was intended to work.
- Please feel free to send me or the TA email or talk to us during office hours if you get stuck or want some help figuring out how to design and implement your program.
- You have several weeks for each assignment, but please don't put them off until the last minute. Leave yourself some time to experiment and familiarize yourself with the Scratch interface.

Programming Assignment #2

In this assignment, you will be constructing a maze world for a sprite you control to move through. There will be a couple of things that can happen to the agent along the way until it reaches its goal. You will need to follow the following steps.

1. Start by reading the help sections on sensing, operators, and variables.
2. Draw a maze on your stage background (click on stage in the Sprite pane, then click on the Background tab and click on edit). You should use a single color to draw the walls (e.g., black). Alternatively, you can use a drawing program to save an image of your background and import it into Scratch (using the import button in the backgrounds tab). Note: your maze doesn't have to be sophisticated at all so

don't spend all your time making the most crazy maze you can imagine. So long as there are walls, that's good enough!

3. Create a Sprite that will represent your protagonist. You can import an image from their library (or elsewhere) or draw one yourself.
4. Set up a reset script that places your protagonist at the starting point of your maze (can be anywhere you want) when the green flag is pressed.
5. Set up 4 control blocks that start with "when (right/left/up/down) arrow key pressed."
6. To each of these blocks you should attach a code block that makes your protagonist move in the appropriate direction. Note: You should use the commands "change x by __" and "change y by __"
7. Now you need to make the movement code blocks more sophisticated. In this case, you don't want your protagonist to be able to move through walls (that defeats the purpose of a maze!). As is such, you need to add conditional logic that makes sure that the protagonist doesn't move in the designated direction if it will run into a wall. Here is a hint about how to do this, since your walls are all the same color, you can make use of the "touching color" item (found in sensing) and select the color of your wall. This value will be true if your sprite is touching the specified color and false otherwise. Therefore, one thing you can do is first move your protagonsit in the correct direction, test if they are touching the wall color and if they are, then move the sprite back in the opposite direction.
8. Add a new sprite that represents your maze goal and place it somewhere in the maze.
9. Add logic that tests whether the protagonist is touching the goal sprite. You should use a conditional and a "touching ___" value to test if the sprites are touching. You could potentially put this conditional in either sprite's script section, but each way may require a slightly different approach. When the protagonist it touching the goal sprite, you should have him say a victory message, such as "I win!"
10. Make it so that once the protagonist is touching the goal, it can no longer move in any direction until the reset script is activated (the green flag is pressed). To do this, you will need to make use of a variable that is accesible by all sprites. You may want to call it something like "gameOver." Once the protagonist reaches its goal sprite, you should change the value of the variable. In your movement code blocks, you should then only allow movement when the variable is not set to the game over value (use a conditional for this). Remember to also add to your reset code block a statement that sets the game over value to not being true.
11. Add an obstacle sprite. This sprite is just like the goal sprite except that if the protagonist touches it, then the protagonist loses and its game over (preventing movement until the reset script is activated). The agent should display an appropriate defeat message when this happens.
12. Add a bonus sprite. A bonus sprite is a sprite that the agent gets points for collecting. You can place it anywhere in the maze. In this case, when the agent touches this bounus sprite, the sprite should disappear from view. You should make use of the constume system to do this and make a second costume just be a touch of paint that is the background color of the maze (e.g., white). Make sure that when the reset script is activated that the bonus sprite reappears.

13. Add a score varaible that is increased for collecting the bonus sprite. When the game starts the score should hold a value of zero and once the bonus sprite is collected the score should be increased. **Note**: You need to make sure that the protagonist cannot collect extra points once the bonus sprite disappears. This can potentially happen even though you have changed the constume to something invisible. Therefore, you may want to make a new variable and add a conditional that increases the score value only when the bonus sprite hasn't been collected. Your approach to doing this should be similar to how you prevent movement once the game is over.

Tips:

- You can hide and show variable values by checking them in the variables tab. You should always have the score variable checked so that you can see it. The others you can uncheck or check for testing purposes.
- Observe the video demo of the final assignment from our course webpage.
- It might be worthwhile to break your code up into broadcasted procedures, but this is up to you!

Extra Credit:

- Add animation to when your protagonist moves (3pts)
- Make Obstacle sprites that move on their own accord to make for more difficult dodging (5pts)
- Add sounds to program (2pts)

Programming Assignment #3

Design your own Scratch world from scratch (no pun intended). This can be some kind of interactive cartoon, a game, or anything else of the sort. The only requirements are as follows:

- At least 5 sprites must be used
- Broadcast messages must be used to initiate behavior in one of the other sprites
- At least one conditional ("if" statement)
- At least one way for the user to provide input (via keyboard hits, mouse clicking, inputting text, etc.) that affects the program
- At least one variable must be used such that different things happen based on the value of variable and the variable must change through the program (either through user input or some other means)
- At least one loop must be used.
- At least two sounds

Tips:

- Did you know you can have multiple stage backgrounds that can be change programmatically? You might want to consider changing the background as part of your program

Extra Credit:

- Utilize a list in a dynamic way (meaning the list changes throughout the course of the program in a relevant way) for 10 extra points. In a game, for example, you could have a score board. See the help screen on control for more info. (10pts)