**Lab #6: High-level Language Programming: Introduction to Scratch**

Due: End of lab, Tuesday, March 4th, 2014.

## Notes to grader: Only grade questions 1 -6. 16 points each, divided equally among the parts. Note, question 7 is not graded. Answers provide for a few questions only as the answers are kind of open ended. Grader should work through questions 1-6 to understand how to grade. This will take maybe 30 – 45 minutes. Grading should probably be lenient for this lab.

## Instructions

* You will need your WFU-issued ThinkPad for this lab. This lab makes use of a programming language named *Scratch.* Download Scratch using the instructions under Resources -> Scratch in Sakai.

# Work Parts 0 and 1.

* Submit your answers to the assignment in Sakai using this document (you may delete the images if you like and Part 2).
* If you have time left over, work Part 2 but you do not need to submit for this lab. However, review and work this part for the next class.

## Part 0: Background

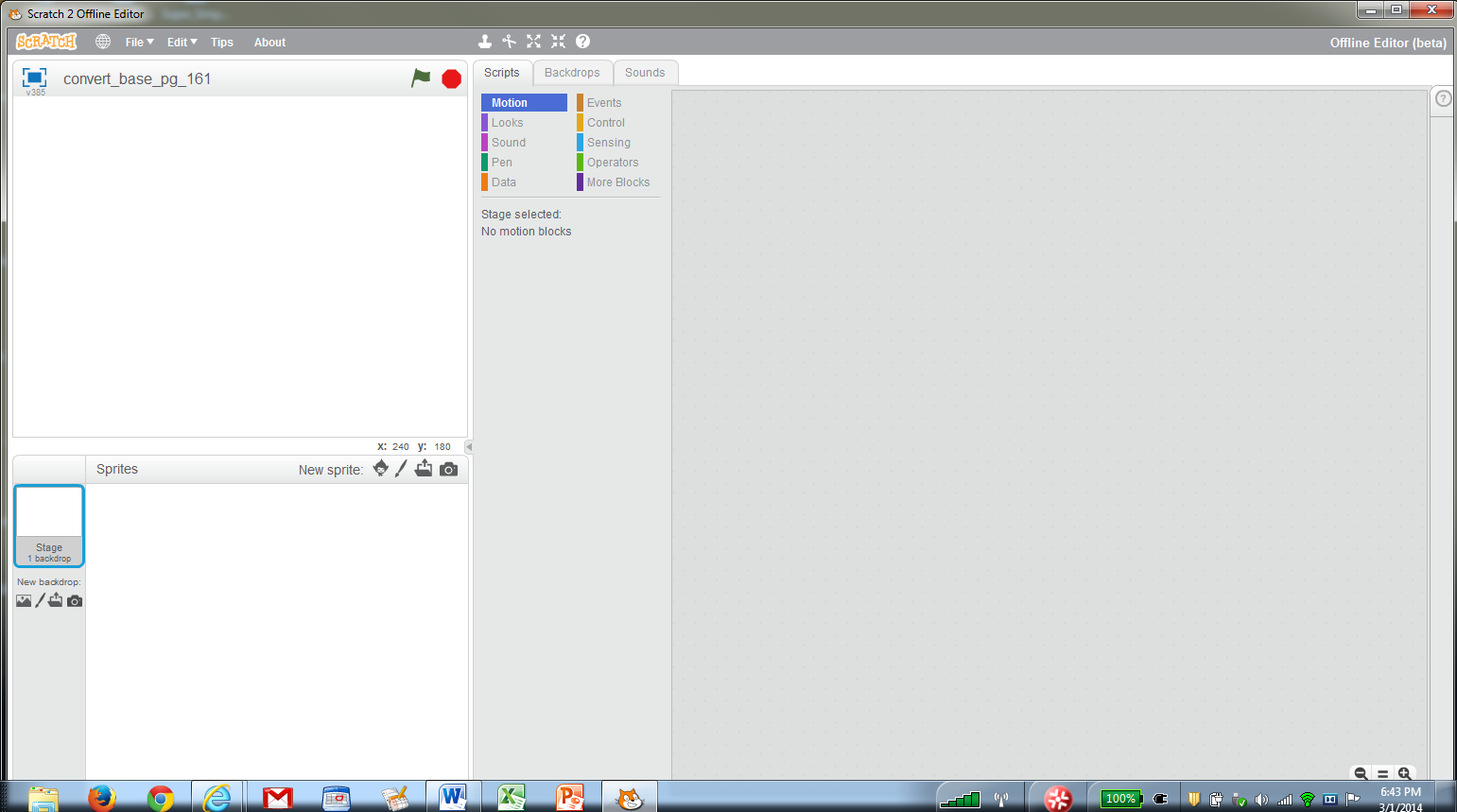
The purpose of this lab is to reinforce what you have learned about algorithms and pseudocode by translating them into a high-level computer language. Background comes from Chapters 6 and 7 (Computer Science Illuminated) and from lectures. For our purposes, the idea of programming is to take an algorithm in some form and translate it into *code that a computer can understand*. The computer in this case is the “compute agent” for the algorithm, much in the same way you might think of a chef as the compute agent for a recipe. We will use an excellent, introductory language, Scratch, that allows focus on problem solving and creativity rather than syntax by providing a graphical interface with drag-and-drop program components (statements). You will immediately see the ease of this approach as compared to machine or assembly code. However, you should remember that eventually your programs must be translated into machine code in order to run on your computer (not by you ☺).

When learning to program, particularly at this stage of experience, it is informative to both analyze and synthesize code. That is, read and study what others have done then write some of your own. You will be asked to describe what existing code is doing, modify code, and write your own.

## Part 1: Jumping into Scratch

Rather than trying to explain everything up front, let’s jump right in and see what is going on. Since you have downloaded Scratch by now, start it running as shown in the instructions. You should have the Scratch 2 Offline Editor running. If not, ask your lab neighbor or lab instructor for help. Once you have it running, connect to the following in a web browser. This works for Internet Explorer.

<http://scratch.mit.edu/help/videos/>

Run the *Introductory Tutorial* called *Make your sprite move forward*. After watching this tutorial, run it again and follow along by doing the same activities in your Scratch Editor as are done in the tutorial. NOTE: When you start your Scratch Editor, you may not have a sprite shown in the lower left window of the editor. If not, select a new sprite by clicking on the sprite icon next to *New sprite:* at the top of the same window. 

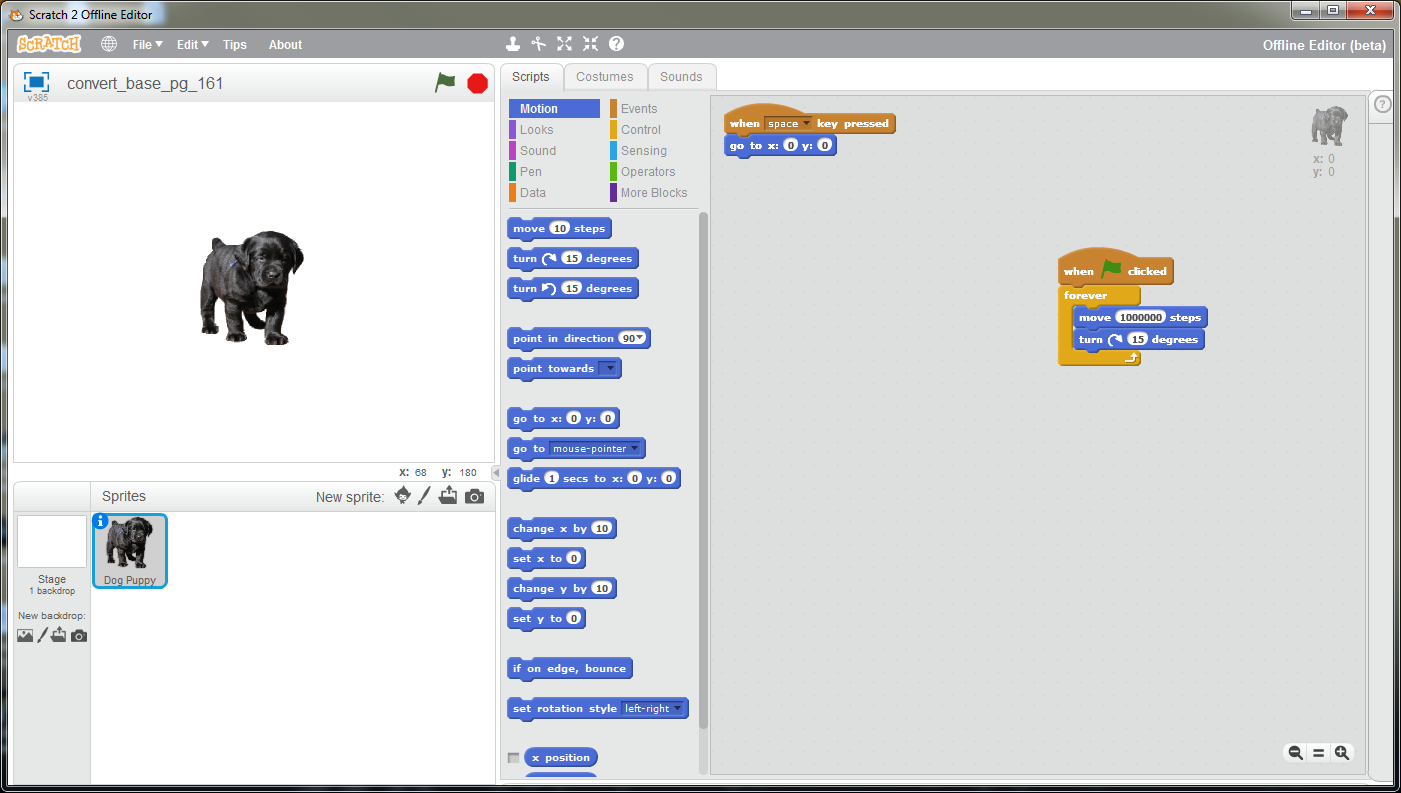
Stop and start the tutorial as needed to follow along.

**Question 1.**

Click here to get a sprite if one isn’t visible

1.a) What happened when you entered a negative number in the move block?

1.b.) Enter 100 in the move block. Keep clicking the green flag. What happens? Does it ever completely disappear?

Clear the blocks you moved to the editor window by dragging them back onto the palette or by right clicking on them and choosing delete. If your sprite is off the window, you can drag it back to the center by left-clicking your mouse on the sprite and dragging it. Run the tutorial called *Making your sprite spin* and follow along as before (note, to return to the menu of videos, click outside the white boundary of the current video). In this figure, you can see the blocks for the tutorial AND in the upper left you see some additional blocks. By putting these in, you can make the sprite go back to the center of the window by hitting the space bar. 

**Question 2.**

2.a)Hit the Stop button (you should do this before making any changes for this question and for others). Change the steps in the move block to 50 and run the program. What happens? If needed, drag the sprite to a location in the window that allows you to easily see the movement.

2.b) Replace the *forever* block with a *repeat* block (pull the blue blocks out of the *forever* block. Then drag the *forever* block away, right click on it and delete). Change the steps to 10 in the *move* block and keep the degrees as 15 in the *turn* block. Enter 24 in the *repeat* block and run the program. What happens?

2.c.) Without watching, how many revolutions would you expect your sprite to make? Explain.

One since 24 times 15 equals 360.

Delete your sprite and select a new one (see above). Run the tutorial *Making your sprite change color.*

**Question 3.**

3.a.) Make the changes suggested at the end of the tutorial and run the program. What happens when you hit the space bar?

3.b). Stop the program then run it, leaving the changes in place. What happens when you hit the up arrow?

Delete your sprite and make select a new one. Run the tutorial *Make your sprite follow the mouse.* Make the change suggested at the end of the tutorial, i.e., make your sprite run away from the mouse.

**Question 4.**

4.a) What change(s) did you make?

Stop the program and change the *forever* block to a *repeat* block with 10 steps. Run the program.

4.b) What happens?

Delete your sprite and make select a new one. Run the tutorial *Make your sprite spin when you say something.*

**Question 5.**

5.a) Make the change suggested at the end of the tutorial. What change(s) did you make?

5.b) Change the operator in the *if* block to a less than (<) so that you check loudness < 10. Run the program. What happens?

5.c) Change the operator in the *if* block back to greater than. Replace the *turn* block with a *point towards mouse-pointer* block. Move the mouse then clap your hands. What happens?

Uncheck the *loudness* block. Delete your sprite and make select a new one. Run the tutorial *Make a simple game.* Follow along and program the game (note that you may not have the specific sound used in the demo. Just pick another one). **Be sure to save your program, which you can do under the File drop down menu.**

**Question 6.**

6.a) As suggested at the end of the tutorial, think of a simple change that you could make during the lab period. Describe your change to the game?

No correct answer here. Whatever the student describes is fine if it makes sense.

6.b) Describe your change to the code. Did it work? If not, explain what you think might be wrong.

No correct answer here. As long as something is here and it makes some sense.

## Part 2: Summing program.

For this exercise, you will implement in Scratch a simple algorithm. You will be introduced to variables and selection. The program should do the following. Start when the green flag is pressed. Ask the user to input two numbers. Sum the number. If the output is greater than or equal to zero, print the sum. If the sum is negative, write an appropriate message. Download the program Sum\_Two\_Nums.sb2 from Sakai. Read over the code and understand it. Test numbers that produce a negative sum, a positive sum, and a sum that equals zero. Writing pseudocode for this problem is suggested.

Students did not have to do this question. So, do not grade.

**Question 7.**

7.a) Which situation does not work (i.e., does not result in an answer)?

When the sum is zero, no message is printed.

7.b) Where is the error in the program?

It only tests for less than zero or greater than zero.

7.c) Fix the error. Hint: Change the condition in the first *if* statement by replacing it with an *operator* block that has an *or* operator in it. Then test for greater than zero or equal to zero. Did you make it work?

7.c) Replace the two *if* statements with an *if-then-else* construct as discussed in class when covering pseudocode selection constructs. Did you make it work?

7.d) Select the *Data* blocks. What are the three variables? What happens when you check and uncheck the boxes next to them.

7.e) Change the program to input 3 numbers and sum them and output using the same rules as above.