Bellevue PI - Lesson #1

1.0 Introductions

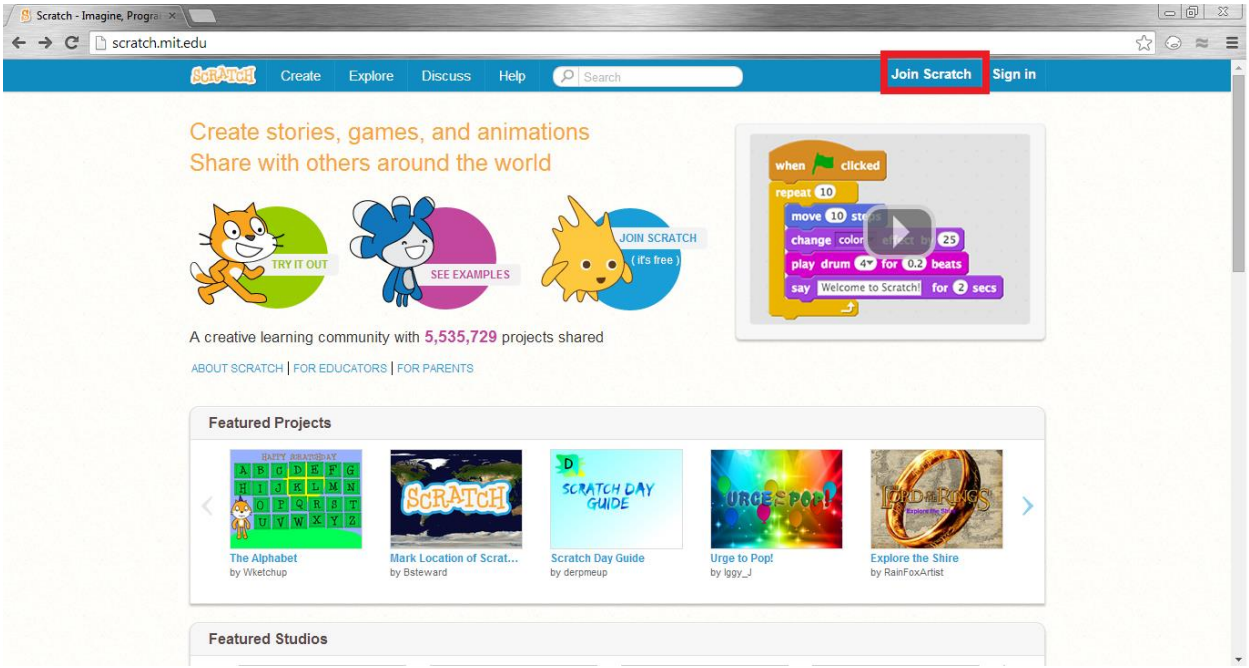
Hello students! This tutorial is designed to allow any student between first and eighth grade to easily follow along and be introduced to the world of programming. Programming, or the art of commanding your computer to do exactly what you want to, is a fascinating and rewarding experience that every student should try out. We here at Bellevue Programming Institute want to give you the joy of seeing something move on your computer, just because you told it to do so! While regular programming classes begin with complex theory and variables, our tutorials are designed so that the student can read them and have no fear in jumping right in and working with the code. One of our core beliefs is that students must have the motivation to try things out for themselves, rather than just watch other people teach.

Enough introductions! Let's get started with our "Hello World" project.

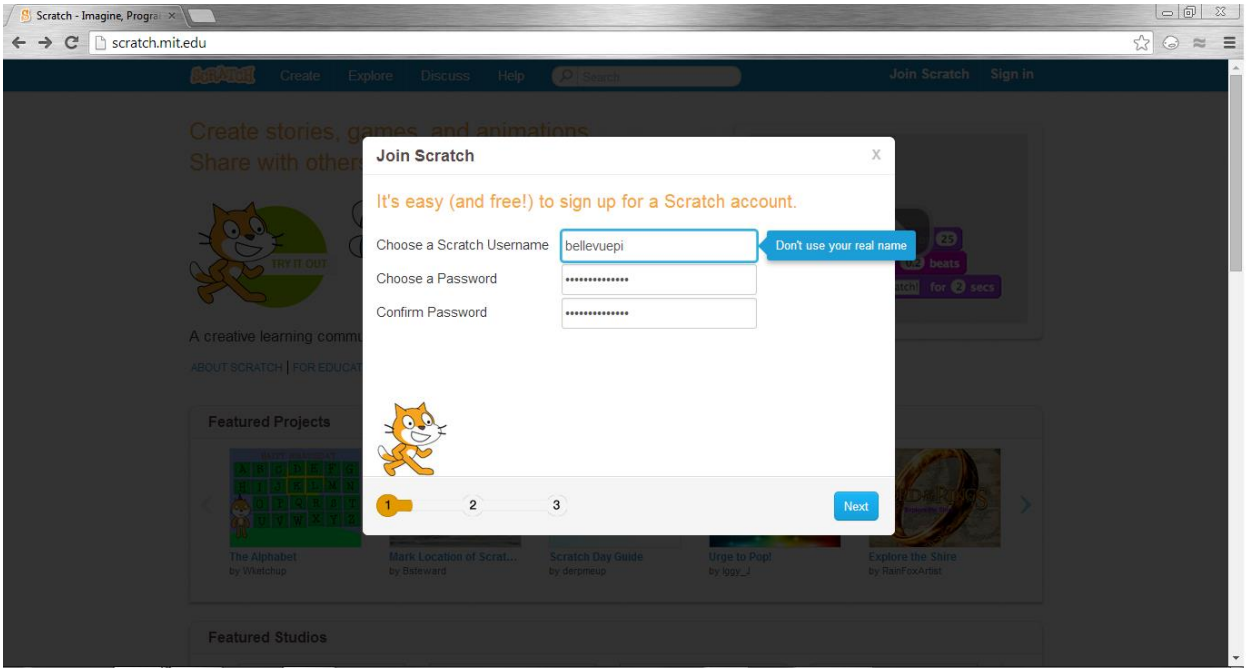
1.1 Setting Up Scratch

The primary program that we will be using in this course is Scratch, a visual programming language developed by geniuses at the Massachusetts Institute of Technology. To get started, get your parents' permission and go to the website scratch.mit.edu.

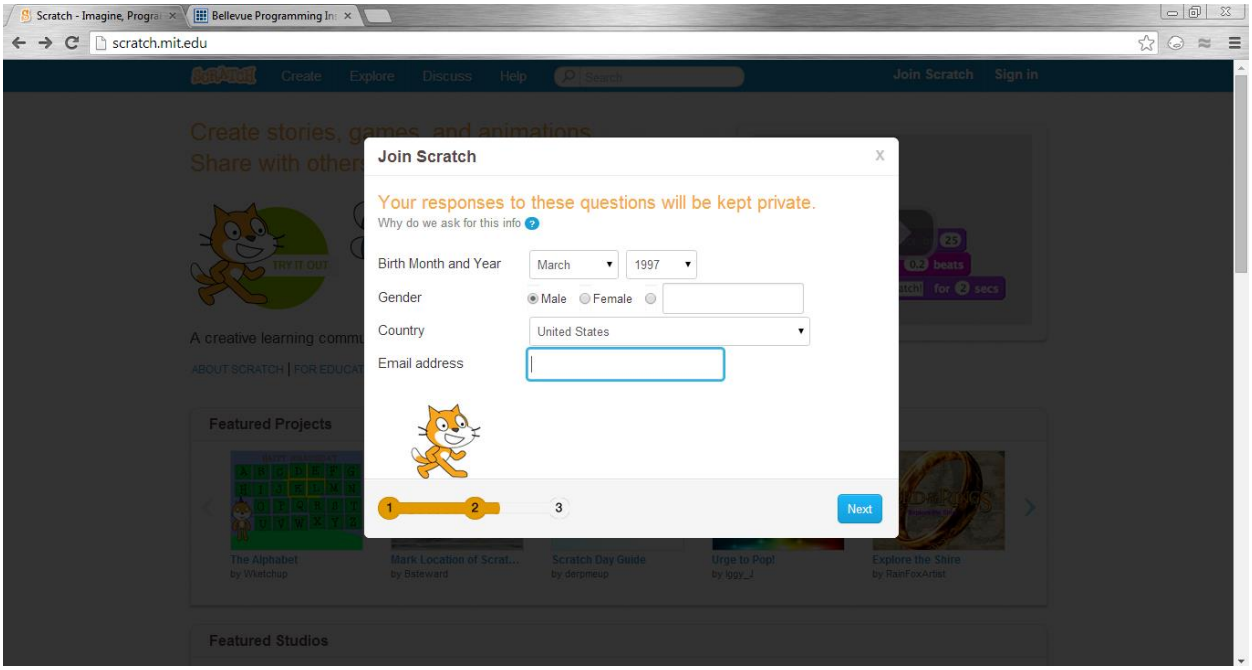
Step 1: Create your account by clicking on the button in the red box.



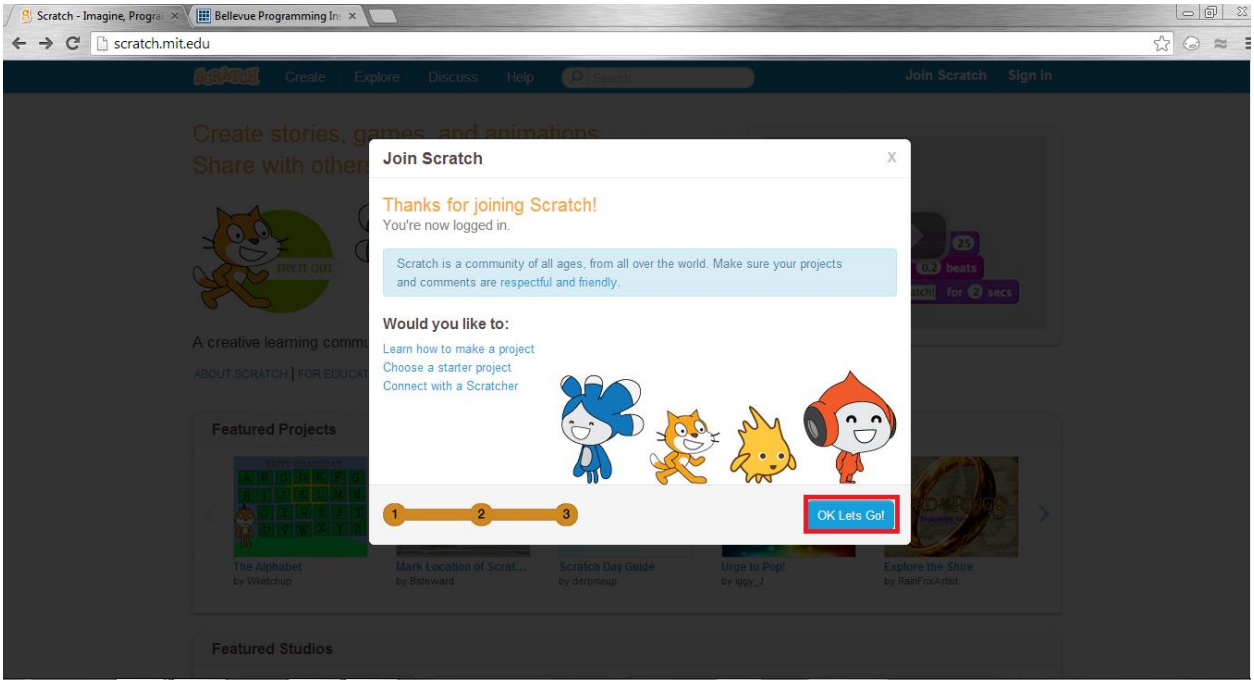
Step 2: Make up a username and password, and make sure to remember it!



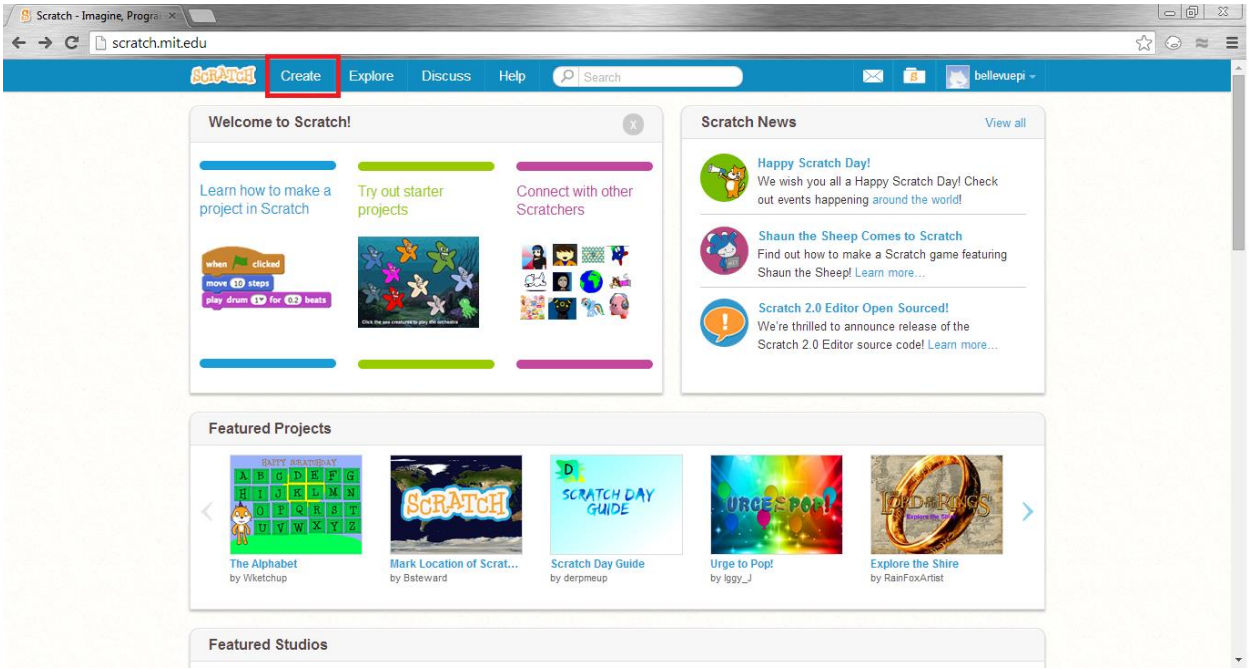
Step 3: With your parent's permission, complete this information



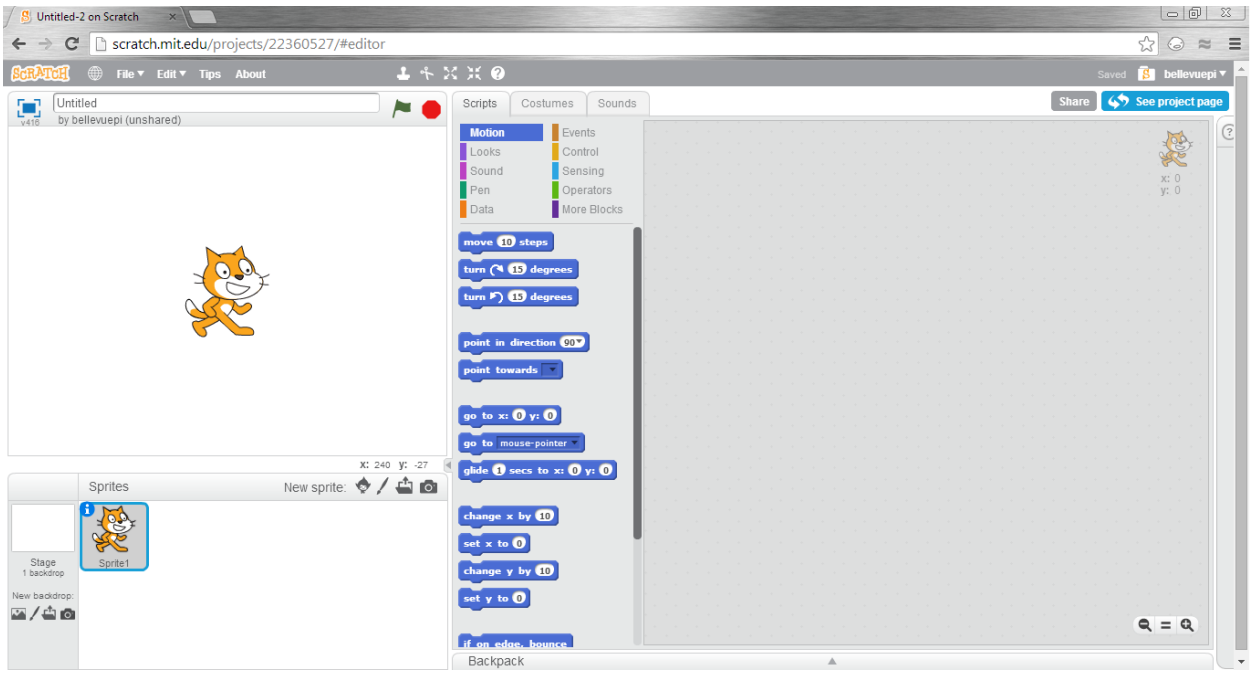
Step 4: You're ready to go! Click the button in the red box to return to the main screen.



Step 5: Click on the Create button.



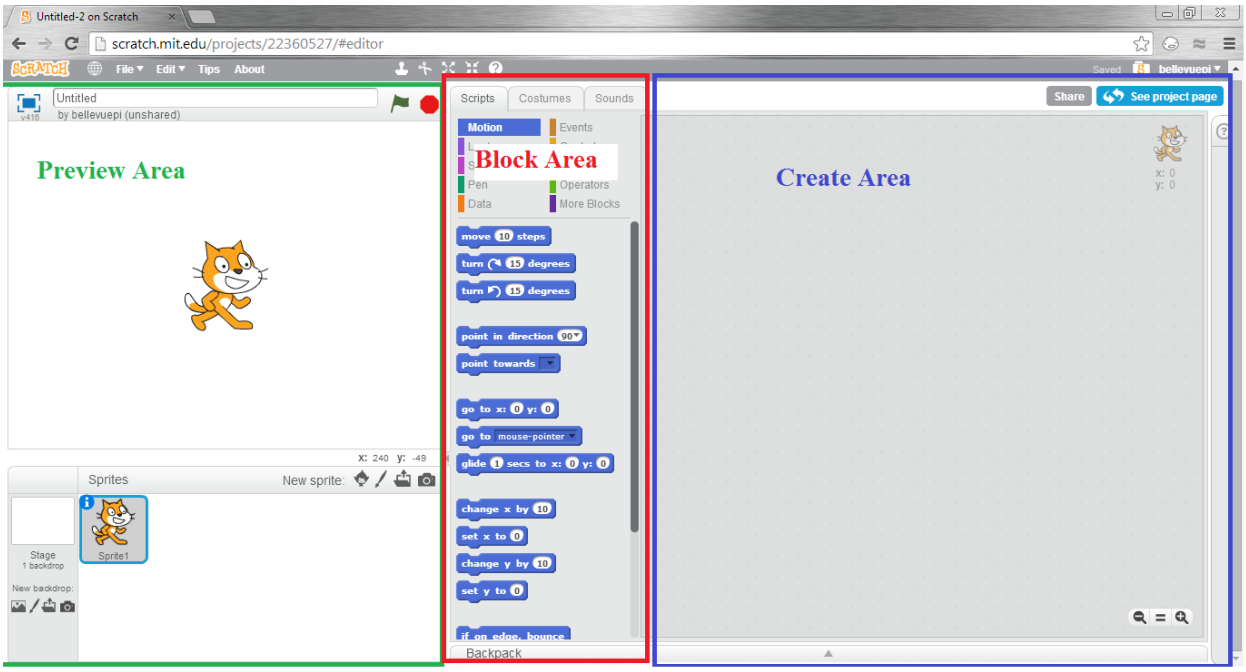
Step 6: Let's get started! You're on the main page for all of your scratch work. This is the primary location where you will be doing the magic of programming.



1.2 Hello World

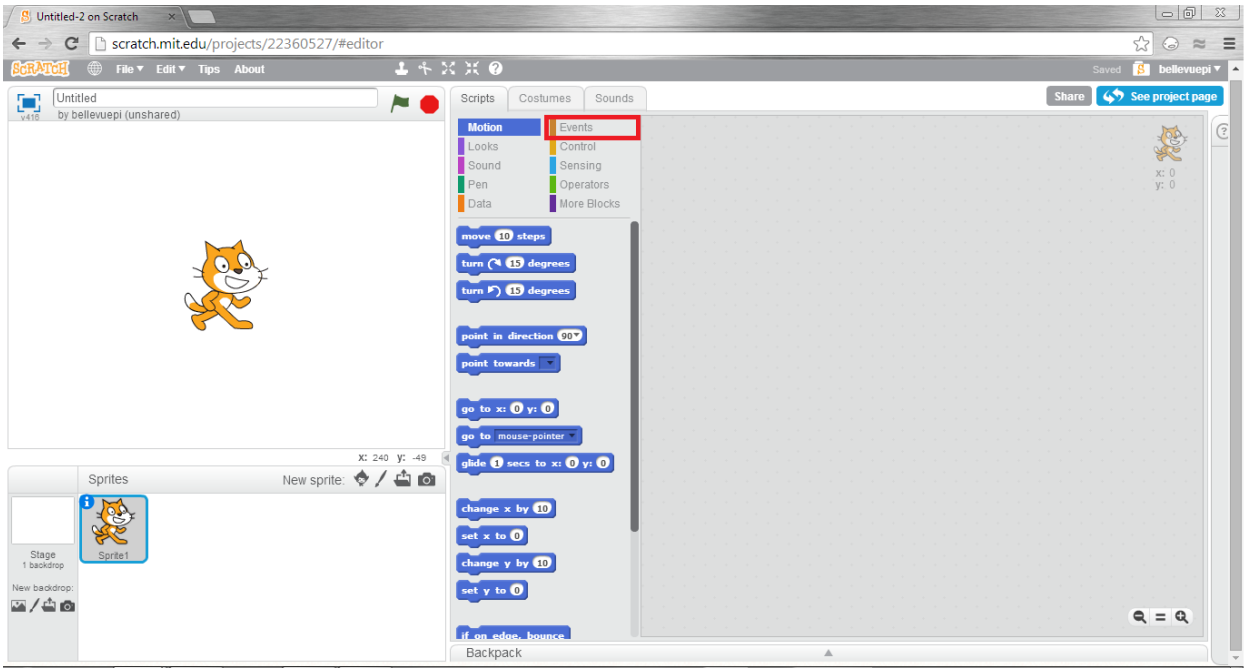
Now that we are at this step, it's time to make your first program. Traditionally, in the world of computer programming, programmers will create a "Hello World" project, which tests if the language that they are programming in will work properly. Therefore, we will make a similar project in scratch to get you warmed up!

As we said earlier, Scratch is a visual programming language, so that instead of doing lots of typing, you will be dragging blocks of items onto the program to command your world to work. The Scratch editor has three main sections: The Preview area on the left, the block area in the middle, and the Create area on the right.

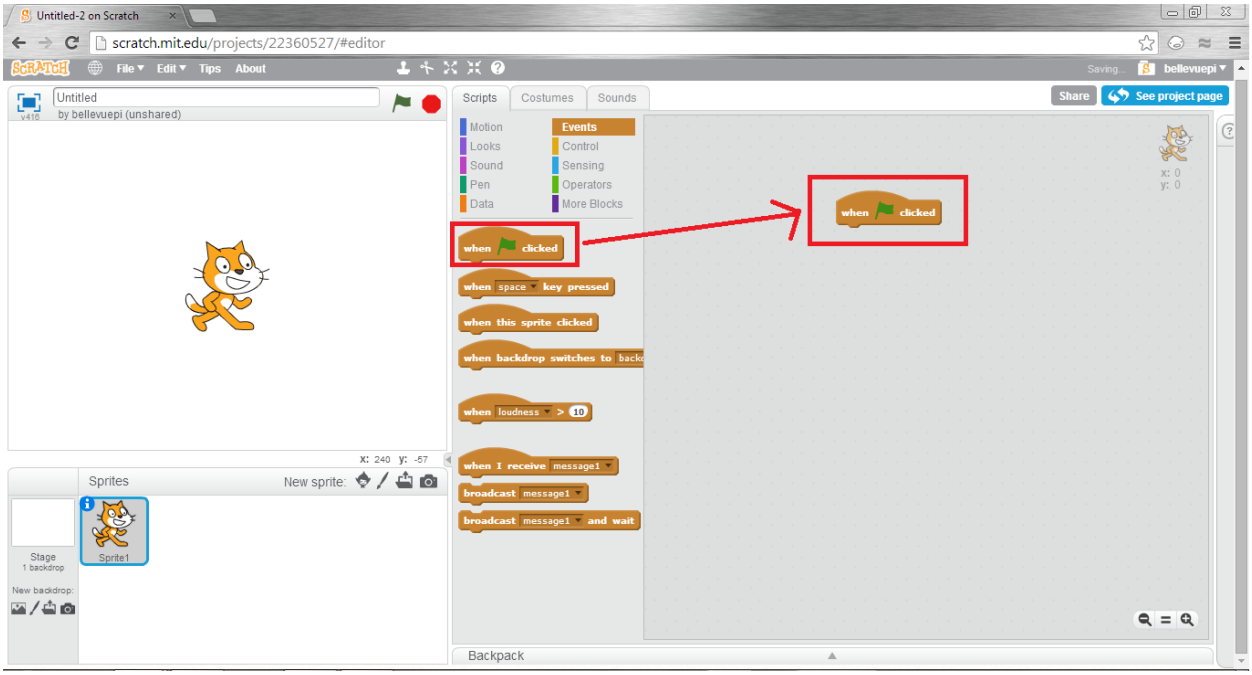


Confused? Don't be. Let's get started and make something work!

Step 1: Click on the "Events" tab in the Block area.

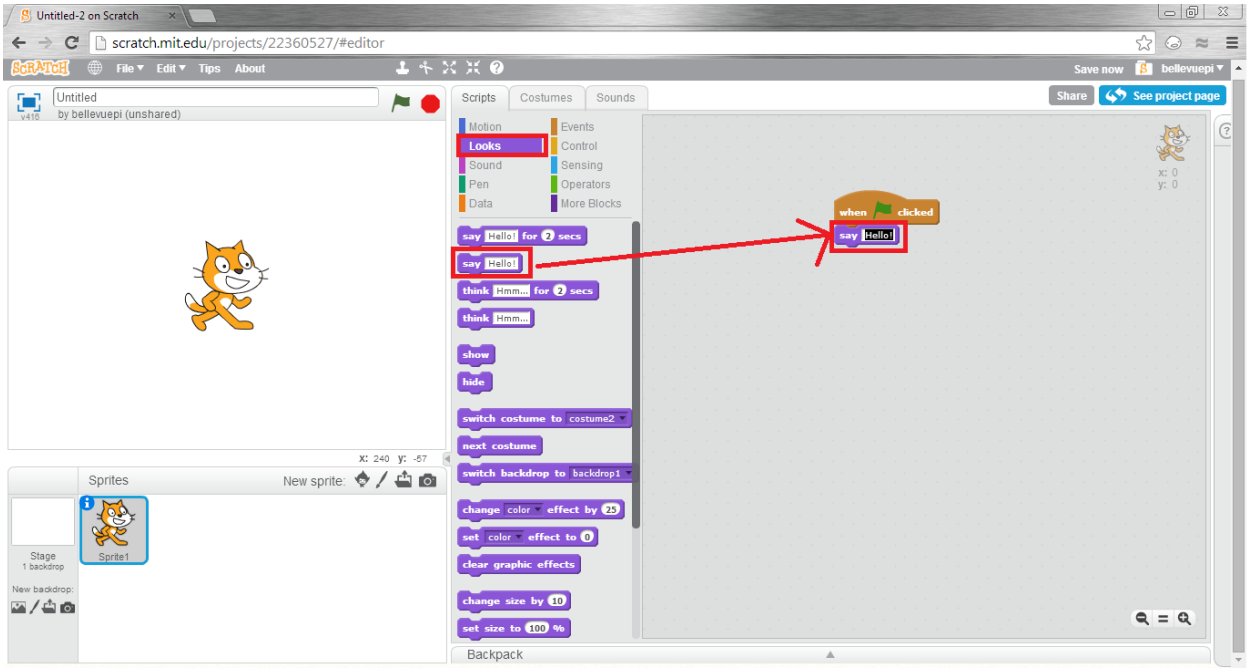


Step 2: Drag the "When Flag Clicked" block into the Create area.



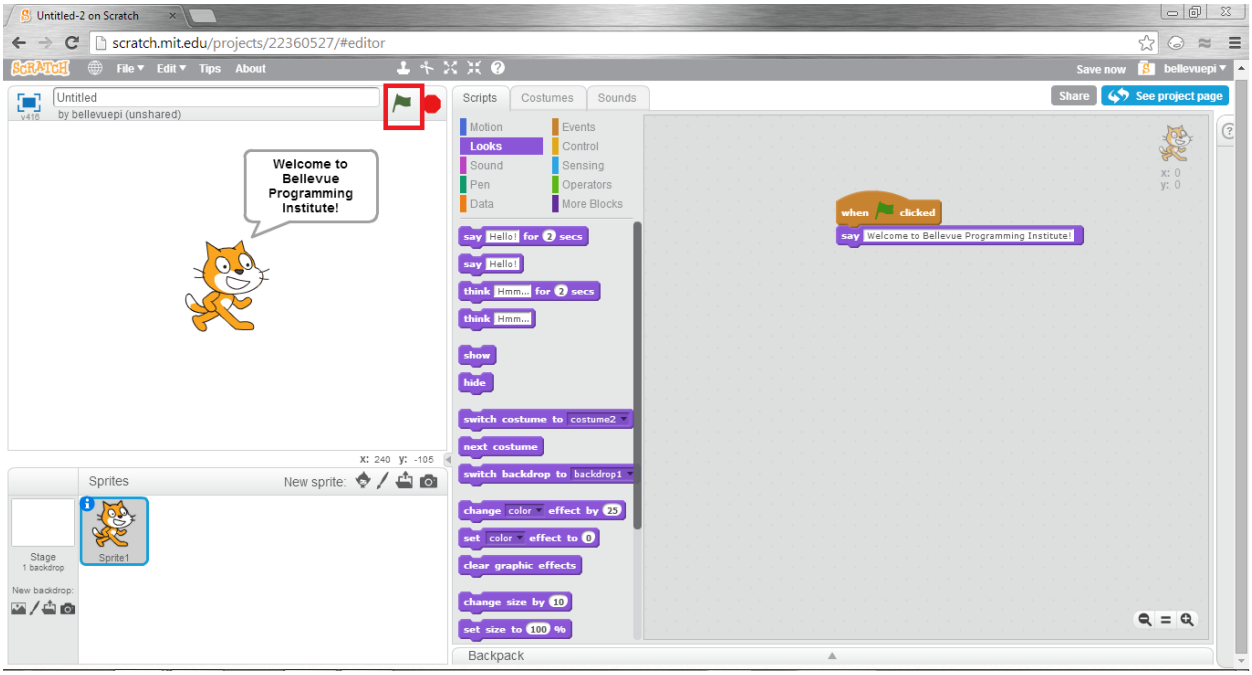
Step 3: Click on the "Looks" tab in the Block area.

Step 4: Drag the "Say Hello!" block directly under the "When Flag Clicked" block. If done correctly, the two blocks should click together.



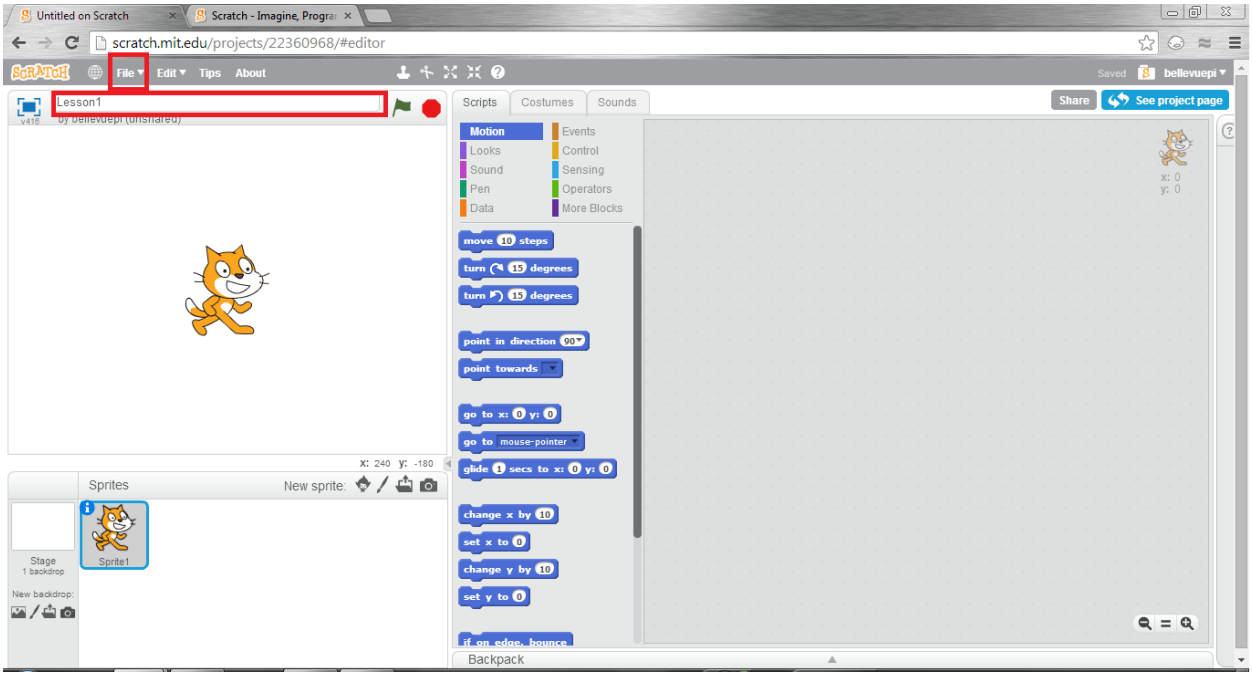
Step 5: Click on the text area of the "say..." block and edit the text to say what you want to say.

Step 6: Test your program out by clicking on the Green Flag in the Preview area of the workspace.



Step 7: Name and save your work by typing in the highlighted box "Lesson 1", and then clicking "File" and "Save". Later, you would be able to return to what you created here. Although these

first steps seem simple, later programs will require much more thinking, and you would want to keep your progress.



Congratulations! You have now begun your journey into the great world of programming, and have completed Lesson 1. Before you go on, we encourage you to explore Scratch and click around. Don't worry if you get confused; the next lessons will better explain what is going on in this world.

Bellevue PI – Lesson #2

2.0 Preview of the Lesson

Now that we understand the basics of how Scratch work, we are ready to test out some more functions of Scratch, as well as try to better understand how those functions work. At its core, Scratch is a place where students can tell stories. The animation and actions all build on that premise, that YOU are able to control what the people on the screen do. Therefore, in this lesson, we want to teach you the basics of moving around in the Scratch world.

2.1 What is an Action?

In Scratch, there are several different types of blocks. A quick non-complete rundown of them would be:

- Action blocks, which perform an action on the actor,
- Look blocks, which change how things look on the screen,
- Logic blocks, which perform a test to execute an action,
- Sensing blocks, which return some value of sensing,
- Input blocks, which allow the user to input information
- System blocks, which allow the system to run.

There are many other types of advanced blocks that do not seem to fall into a single category, that we will allow you to explore on your own. So far, in the "Hello World" (Lesson 1) Program, we used the system block of 'When Flag Clicked" and the Look block of "say...". The "say..." block is a look block because it only changes the way that things look on the screen, without changing any basic properties of the actor. The "When Flag Clicked" block is a system block

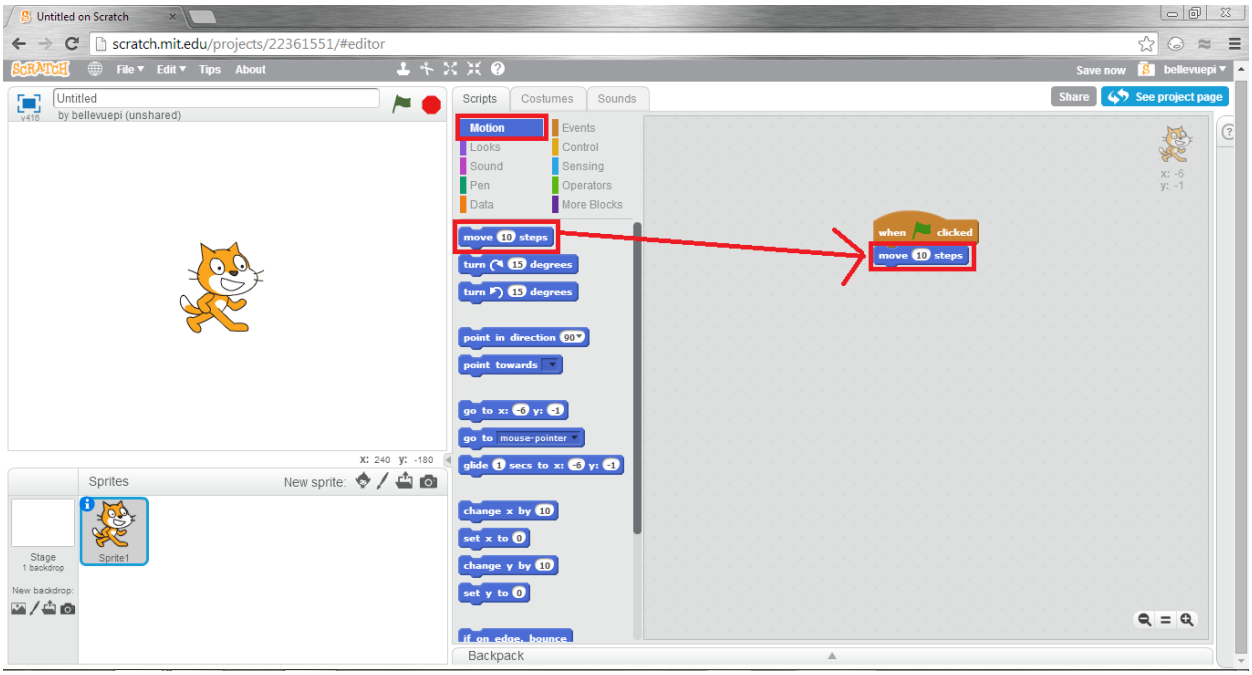
because when the system receives information about it, the actor is able to begin its work. In regular programming languages, the "When Flag Clicked" block is similar to the default run function, which is run when a program starts. Its function is to tell your computer, "Hey! let's get something started here!"

In this lesson, we will introduce the Action blocks, and describe what they do to the actor. These blocks are very important because they allow scratch to be animated, changing the position of the Actor on the screen.

2.2 Using Actions

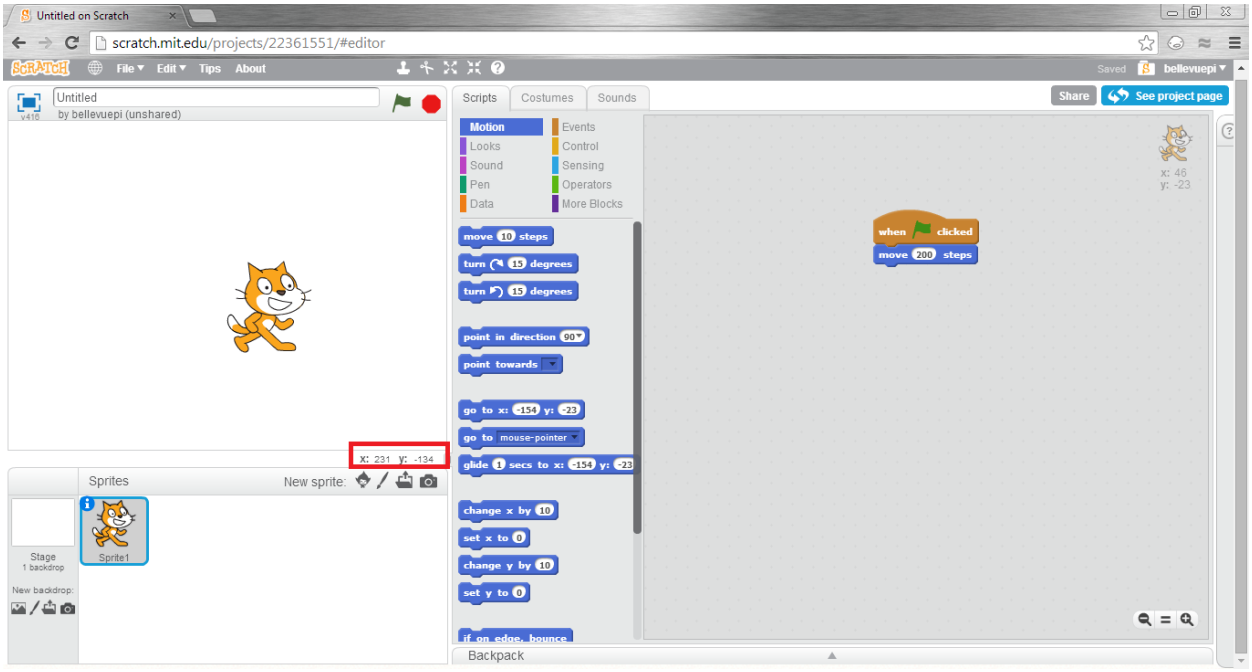
Many of the basic action blocks can be found under the Motion tab. Let's get something started to see how they work.

Step 1: Under the Events tab, drag a "When Flag Clicked" block into the Create area.
Step 2: Click on the Motion tab and look for the "move ... steps" block. Just like the "say..." block, drag this block directly underneath the "When Flag Clicked" block.



Step 3: Change the number to be some number other than 10 but less than 200.

Step 4: Click the Green Flag in the Preview area to see what you have created.



If your "move... steps" block moved a significant number of steps, you should see the Cat actor move some area across the screen. This basic action is the premise of most of the animation aspects of Scratch.

Let's see if we can understand why this works. If you look at the Preview area of the workspace, you will see in the lower right hand corner some odd numbers such as "X: 231 Y: -134". These numbers represent a coordinate grid where the Actor is free to move around in. Therefore, the Actor must have some trait that allows it to change where it is on the screen. A higher level question would be to wonder: What does the "move ... steps" block change about the actor?

2.3 Turning Around

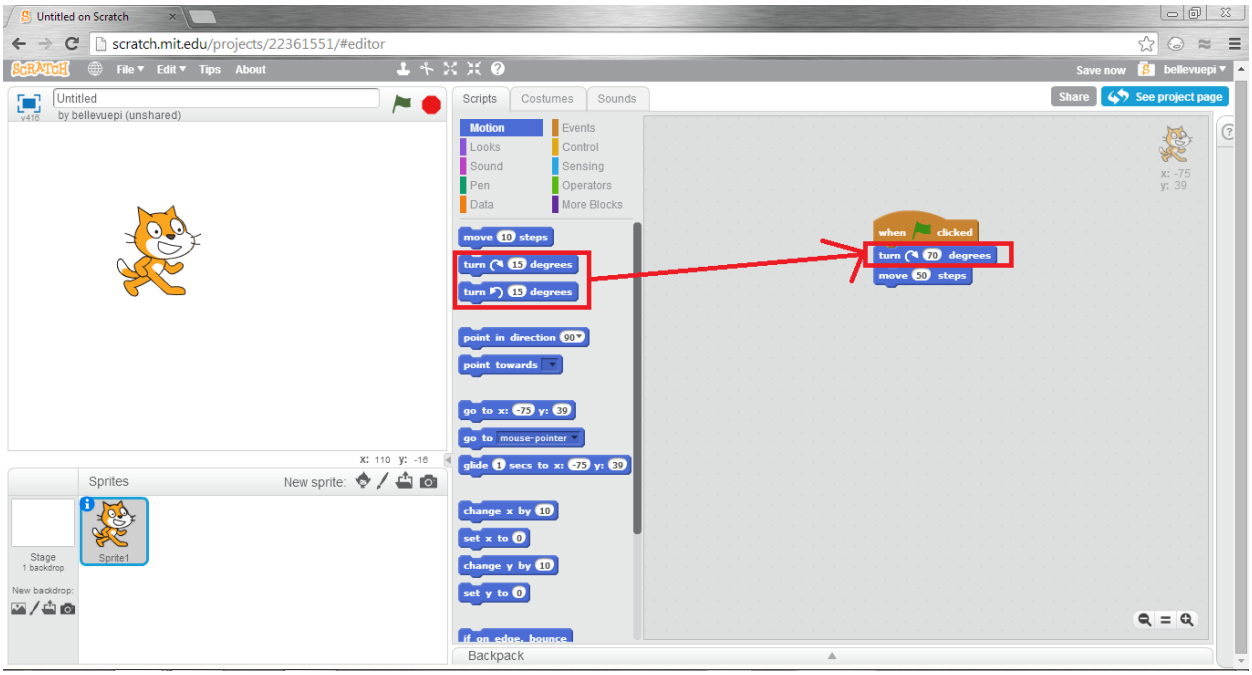
Let's test out another Action block.

Step 1: Click on the "move ... steps" block and drag it out of the screen. This will be the same as deleting an action.

Step 2: Under the Motion tab, look for the "turn... degrees" block. Choose either the clockwise or counterclockwise block, and drag it under the "When Flag Clicked" block.

Step 3: Change the number of degrees that the block says.

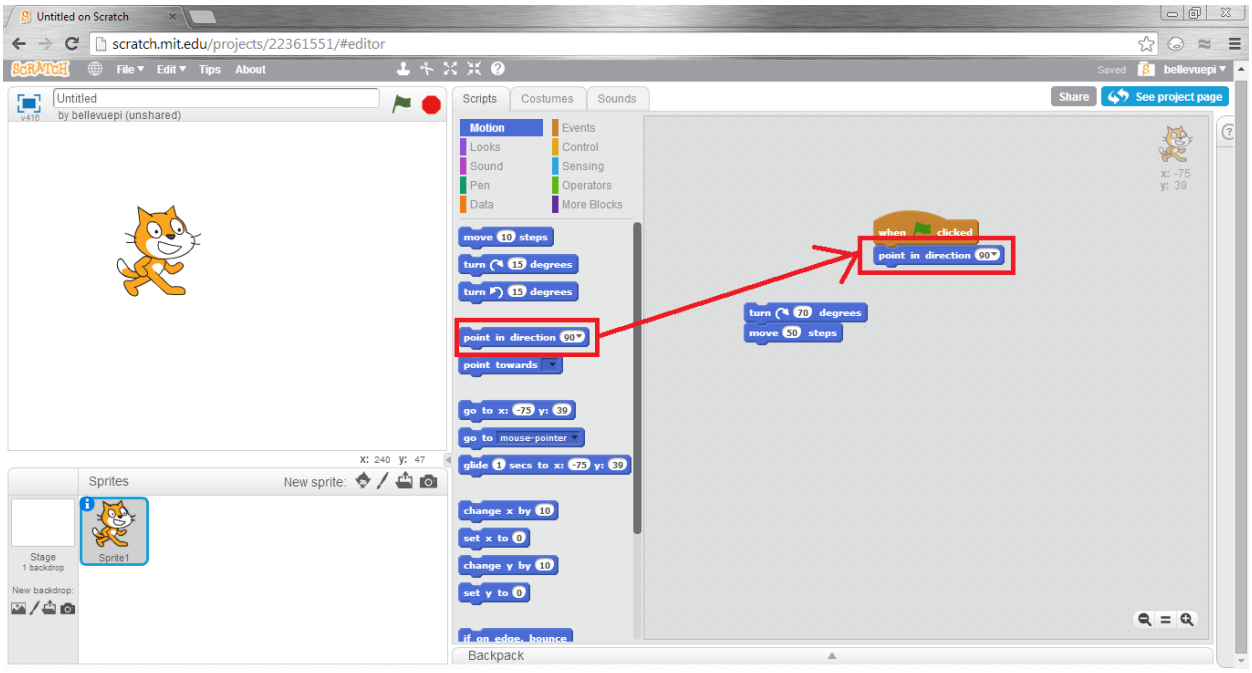
Step 4: Add a "move ... block" after the "turn ... degrees" block. Remember to change the number of steps to be what you want it to be!



Step 5: Press the Green Flag in the Preview area to see what your program has done.

What seems to have happened this time? Instead of moving straight forward, it seems like the actor has moved in a different direction. The "turn...degrees" block changes the direction that the actor faces. Afterwards, the "move ... steps" block merely moves the actor forwards in the direction that it is currently facing.

In order to reset the actor to face forwards, you can drag the "point in direction..." block back under the "When Flag Clicked", removing the chain of "turn ... degrees" and "move ... steps". Instead of dragging these blocks out of the Create area, just move them so that they are no longer connected to the "When Flag Clicked" block.



Try pressing the Green Flag. What happens to your actor?

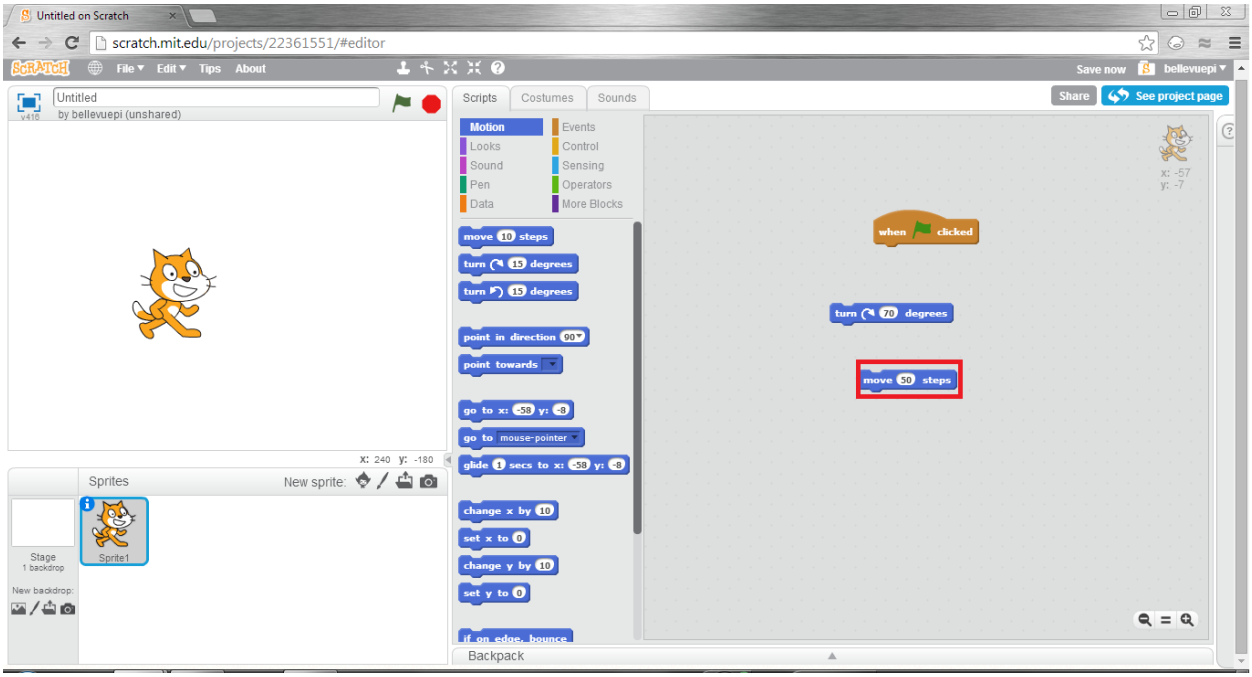
The "point in direction...” flag immediately changes your actor to face the direction specified. One important thing that Scratch uses is a coordinate system that comes from mathematics. Therefore, upwards is 0 degrees, right is 90 degrees, downwards is 180 degrees, left is 270 degrees. What do you think if you put in 360 degrees?

2.4: Does Order Matter?

Now, let's test to see if order matters in this program.

Step 1: Delete the "point in direction... block" by dragging it out of the Action area.

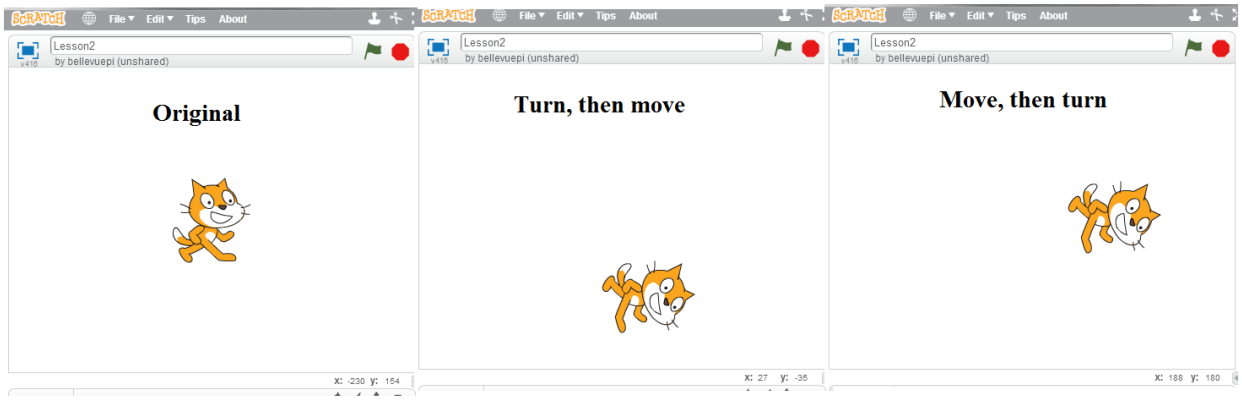
Step 2: Break apart the previously made blocks of turn and move, so that they are no longer connected to each other. You can do this by dragging only the "move .. steps" block away from the "turn...degrees" block.



Step 3: Drag the "move ... steps under the "When Flag Clicked" block, and attach the "turn...degrees" block under it.

Step 4: Press the Green Flag.

If your "move...steps" block was large enough, then you should notice a significant change in your actor. If not, see the below picture:



What happened? It seems that by changing the order of the action blocks, we have changed where the Actor ends up in the location, even though we didn't change anything about the action blocks themselves.

This is a very important concept in programming: ORDER MATTERS. The order that you execute actions will typically have an effect on what actions are executed. This is because many actions change properties that the Actor has. Order is the difference between "Turn towards the door, then walk 50 steps" and "Walk 50 steps, then turn towards the door". Clearly, the first action chain will get you closer to the door, while in the second chain could leave you standing in a completely different area of the room!

2.5: Gliding towards success

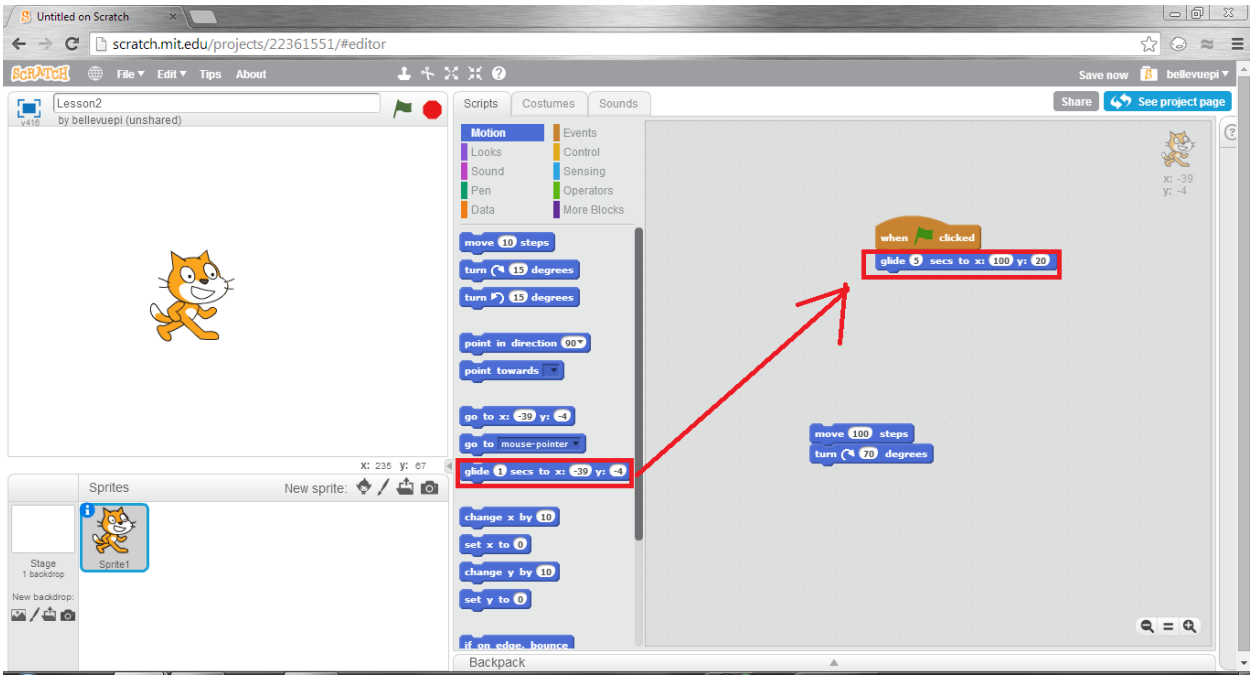
One of the odd results of our "move...steps" block is that it is not smooth; it seems like the Actor instantly teleports from one location to another. Clearly, this is not wanted when you want to see the Actor slowly move from one area to another! Luckily, there is another block to solve this problem.

The "glide ... secs to x:.. y:..." is an interesting block, not only because it allows for you to see the intermediate steps of the Actor as it moves, but also because it allows you to specify a specific location to move towards. Test it out using the following steps:

Step 1: Drag away the "move...turn.." blocks to another portion of the Action area.

Step 2: Drag in the "glide...secs to x:... y:..." block and place it directly under the "When Flag Clicked" block.

Step 3: Edit the three numbers of the "glide...secs to x:... y:..." block to be whatever you want it to be.



Step 4: Press the Green Flag to test out your new program.

Now, your Actor is able to smoothly move from one location to another. Interestingly, your actor is able to move to any point on the screen in this manner, by your changing of the x and y values. Because each point corresponds with a specific point on the screen, you can move your actor anywhere with this block.

2.6: Extension Activity

There are many different ways to use the Action blocks, so this is the time for you to try some out for yourself. Try putting some of the blocks together so that your actor moves in a square, moving the same distance for each length and also facing in the correct direction as it moves. Afterwards, try out some ideas of your own!

Bellevue PI: Lesson #3

3.0 Preview of the Lesson

Although it is interesting to make the Actor move where you tell it to move, it is even more interesting to allow the computer to determine where the Actor moves! Better yet, it is possible for you to tell the computer to execute actions automatically that moves the Actor from one location to another.

3.1 An Introduction to Logic

The core of basic programming is logic, which is defined as the principles that are always true and command a computer to perform a specific task. Although things in the real world may change from time to time, logic is entirely concrete. There is no way of changing the values of "true" or "false" to mean something different. For example, the number 2 will always be less than the number three. If you would state that "2<3", I would tell you that that always would be a true statement. No matter what happens, 2 will never become larger than 3. If, however, you say that "4<3", I would tell you that you are wrong! Clearly, four things are more than three things, so "4<3" is false.

All of logic is founded on these ideas that things can be true or false. However, it isn't very interesting to go around saying "Three is less than four!" all day long. People would probably give you weird looks and ask if you are okay. Instead, we want to use the fact that this statement is true to have the computer do something for you.

There is a very useful function in Scratch, and in most programming languages, called the "if...else..." block. In this block, the computer will perform a logic test to see if it should be doing something. The statements after "if..." are the things being tested, and the statements after "else..." are the actions that the computer should take. The computer would only execute the "else..." actions if the "if..." block is a true value.

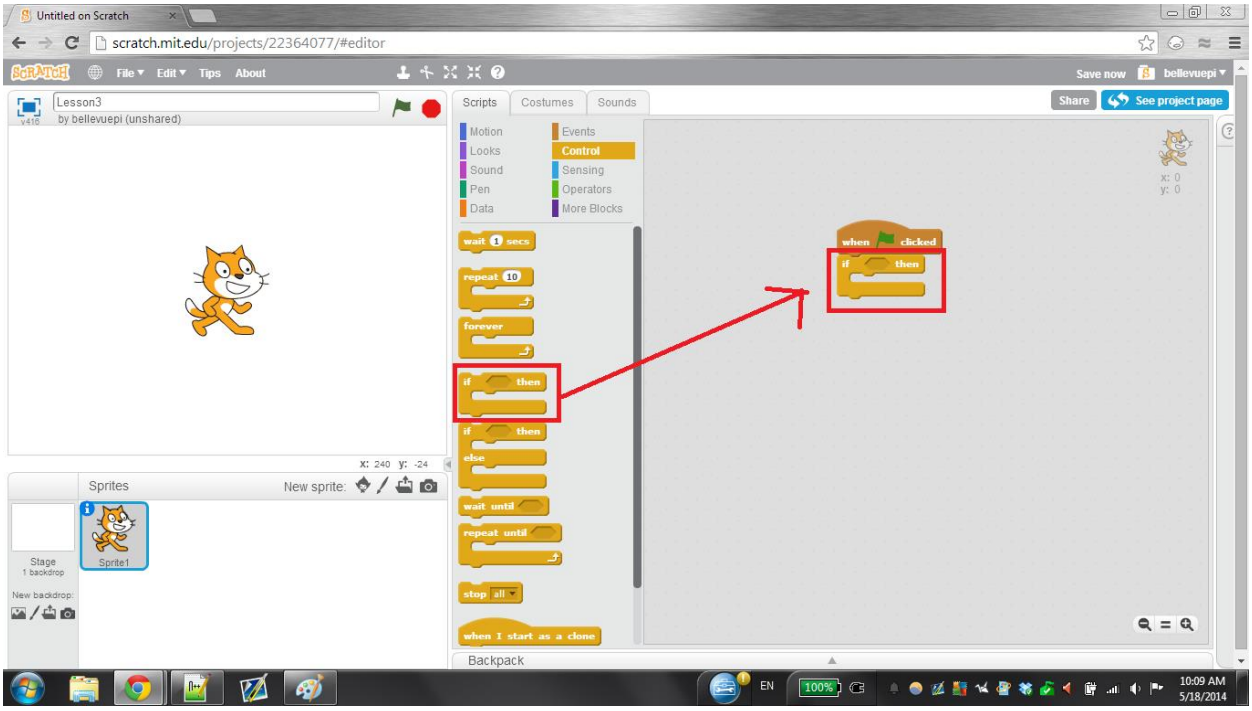
Perhaps this is a bit too complicated all of a sudden. How about a real world example? Suppose your mother asks you to wash the dishes, while you are busy learning about Scratch. You might groan at the extra chore, and your mother reconsiders and says: "If there are less than 5 dishes in the sink, you need to wash all of them. Otherwise, I will take care of it." After this, you should run over to the sink and count the number of dishes there. Surprise, there are eight dishes in the sink! Because eight is not less than five, you do not need to wash all of the dishes and can happily continue learning about programming.

When you ran over to the sink, you were seeing if there was a condition that could be satisfied. You unconsciously checked if it was true or false, and only if it was true, you would perform the action required. Similarly, the computer works in this manner. It will perform a check to see if the condition is true or false, and ONLY if it is true will it do an action. Let's start playing with Scratch to see how this works.

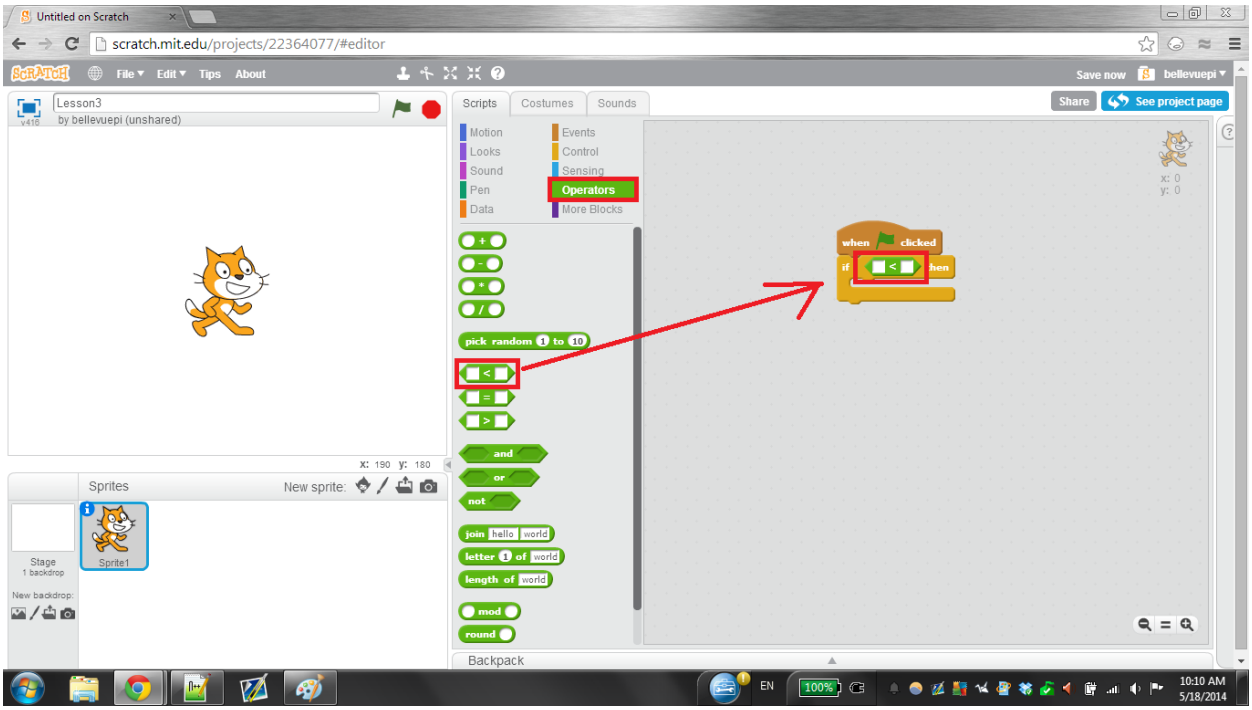
3.2: Testing out Logic

Step 1: Drag a "When Flag Clicked" block into the Action area.

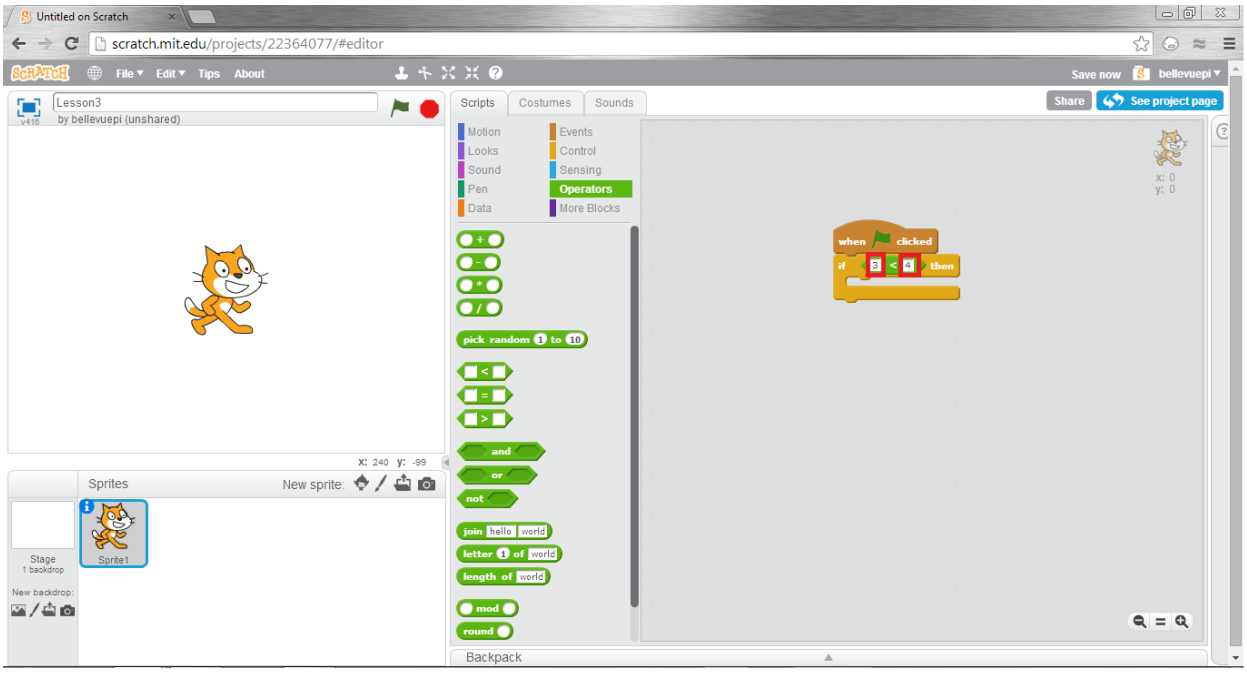
Step 2: Under the Control tab, drag a "if...then..." block under the "When Flag Clicked" block.



Step 3: Under the Operators tab, find the "... < ..." block. Drag it into the "if...then..." block, in the space directly after "if..."

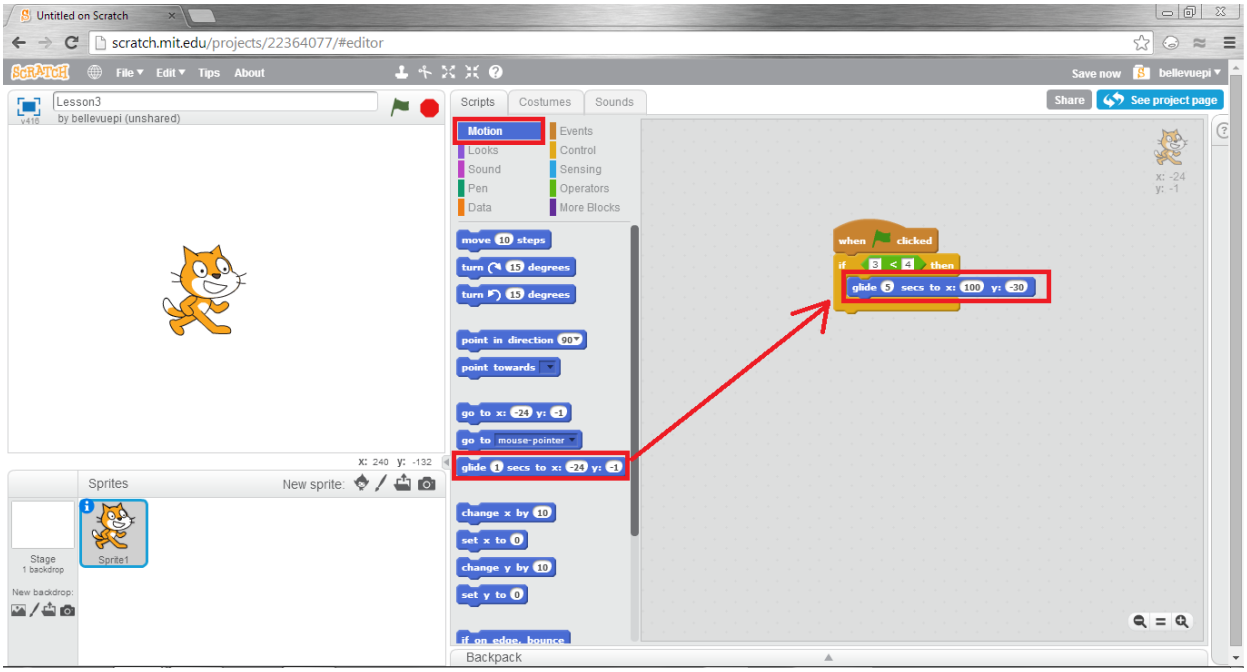


Step 4: Put in the numbers "3" and "4" in that order in the 2 empty blocks of the "... < ..." block.



Step 5: Under the Motion tab, Select the "glide...secs to x:.. y:..." block and drag it inside of the "if...then..." block.

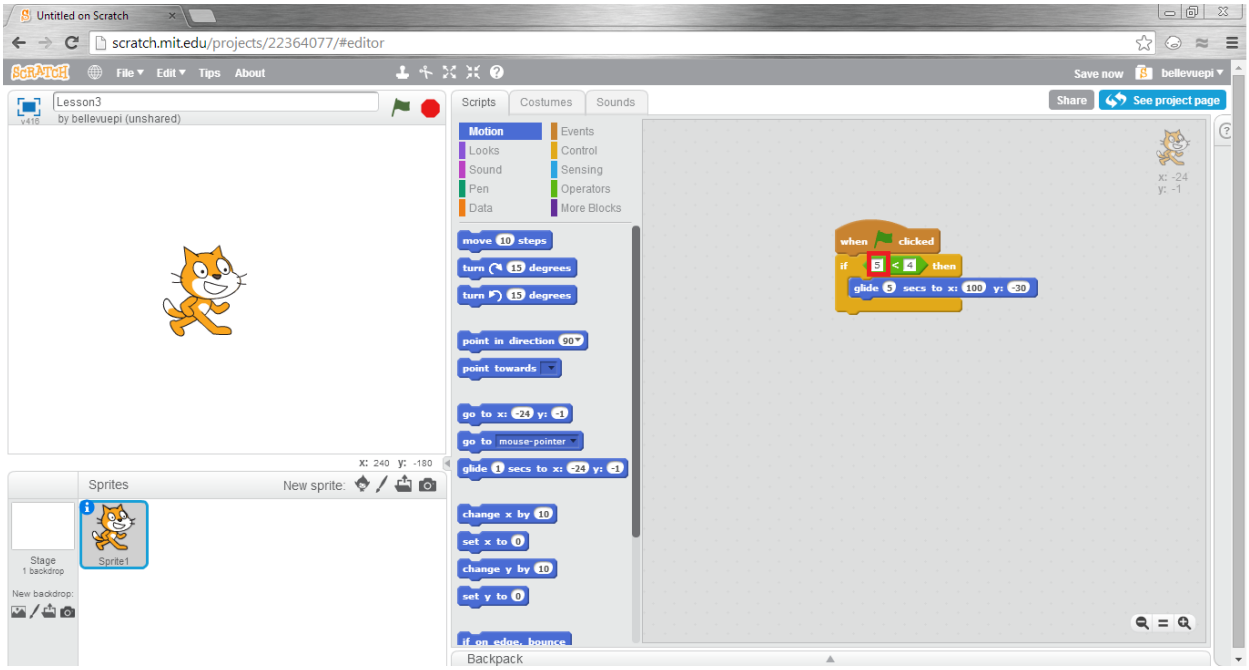
Step 6: Change the conditions of the "glide...secs to x:... y:..." block to be what you want them to be.



Step 7: Press the Green Flag in the Preview area to see what you have made!

Wait a minute... this looks exactly like what had happened before. It doesn't look like the Actor has changed in any way. Is logic therefore useless?

Let's try something else. Change the number "3" to "5" and press the Green Flag again. Does the same thing still happen?



If done correctly, after pressing the Green Flag, you should see the fascinating result of... nothing happening!

Consider what had just happened in the computer. When Scratch saw the statement "5<4", it immediately determined that this statement was false. Therefore, if something is false, it would not do the following statements, and therefore not do anything. Likewise, when you saw that there were more than 5 dishes in the sink, you realized that the condition was false, and did not

wash the dishes (unless if you are an especially good child!). Based on the condition, you determined if an action should be done.

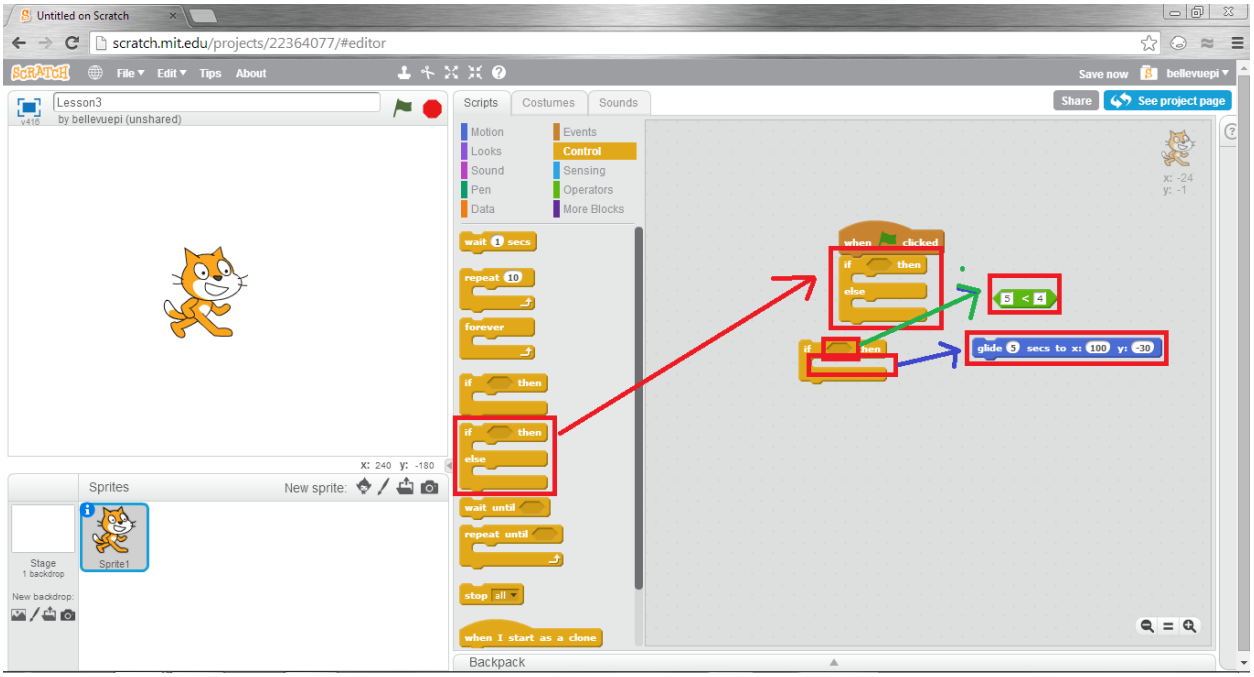
3.3 Another Logic Block

However, what if you mom said, "If there are less than 5 dishes in the sink, then wash all of the dishes, but otherwise ask me to come over?" In this statement, there is still only one condition phrase, but there is now a choice of two action phrases. If the condition phrase is true, then you should perform the first action. Otherwise, you should perform the second action.

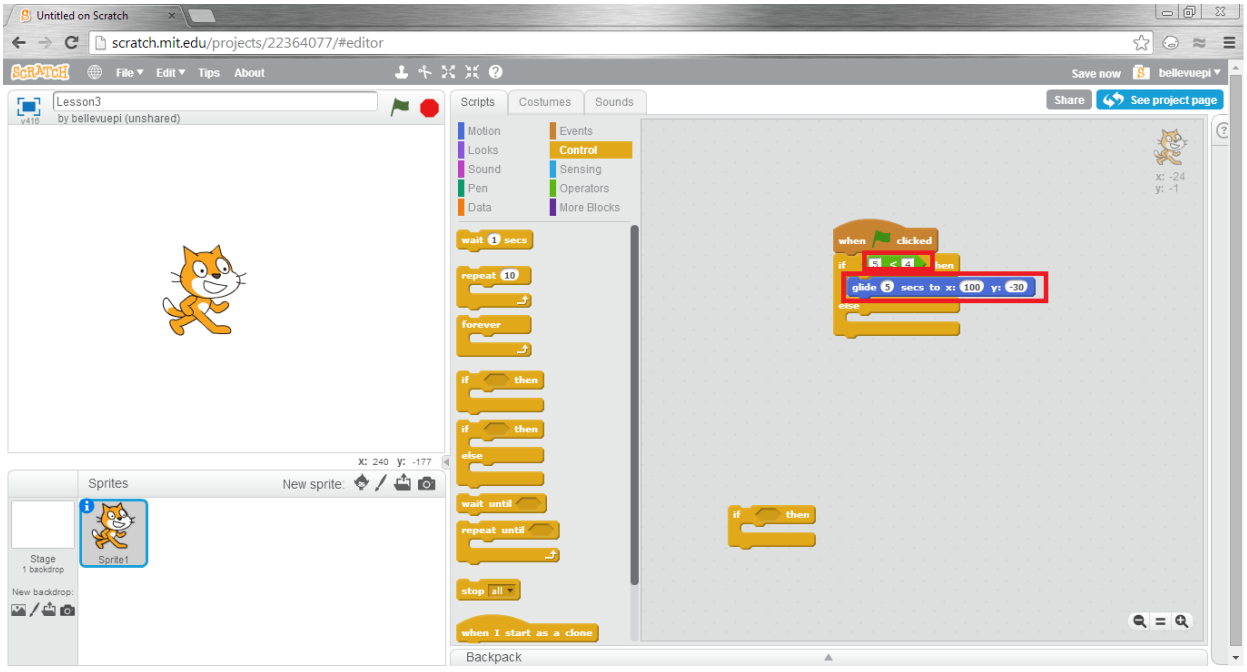
Scratch visualizes this with the "if...then...else..." block, which is very useful for performing a different action if false. Let us test it out!

Step 1: Drag the green "... < ..." and the blue "glide... secs to x:... y:..." block out of the orange "if... then..." block, but keep them in the Action area. Also drag the orange "if... then..." block away from the "When Flag Clicked" block.

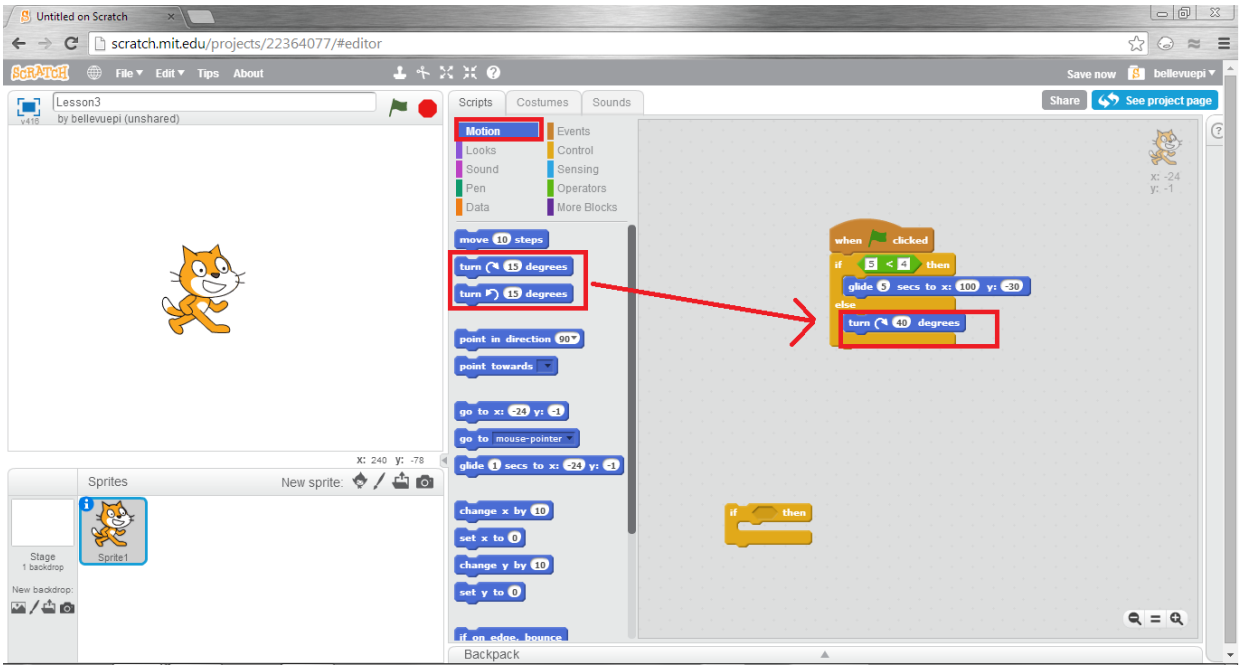
Step 2: Drag the "if... then... else..." block under the "When Flag Clicked" block.



Step 3: Drag the green "... < ..." block into the "if..." section, and the "glide...secs to x:... y:..." under the "then..." section.



Step 4: Under the Motion tab, drag the "turn...degrees" block into the "then..." section. Make sure to edit the values to what you want them to be!



Step 5: Press the Green Flag to test out your new program.

What happens this time? Because your previous condition is still false, instead of not doing anything, the Actor turned some number of degrees instead. If you change the 5 back to a 3, then the statement would become true again, and the actor would glide to a position again.

3.4: Extension

Logic is at the core of all programming, and is extremely useful. These basic ideas of "true", "false", "if...then..." and "if...then...else.." are the very start of making the computer more responsive to your requests!

Your extension activity is to use multiple conditions after the "if...then...else..." blocks, similar to the order idea made in Lesson 2. See if there are interesting combinations that you could create.

Also, try placing a logic block inside of another logic block. Does it work? What do you think would be the purpose of this?

Bellevue PI Lesson #4

4.0: Preview of the Lesson

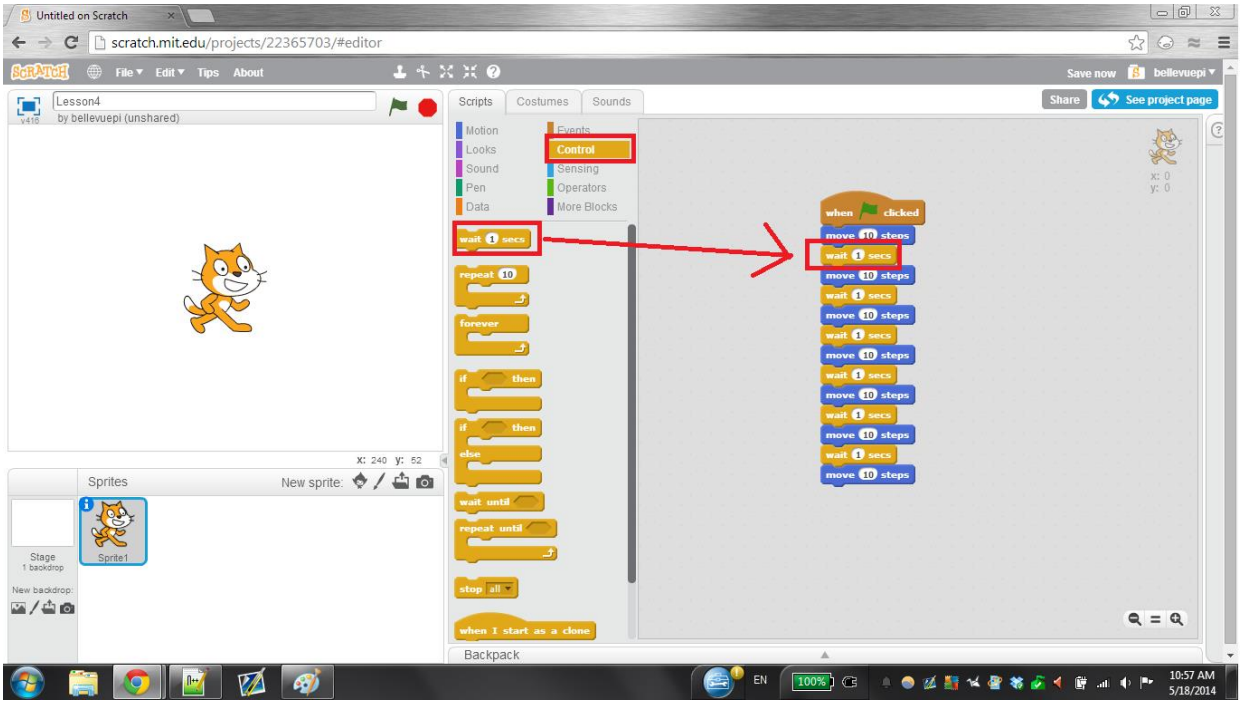
After learning basic logic, you have gotten a glimpse into how the computer responds to your actions. However, there are many more steps to learn. We will explore two more components of logic: the "repeat.. ..." function and the "not..." operator.

4.1: What does it mean to repeat?

Let's reconsider the "glide...sec to x:... y:..." block. How do you think the computer know how to make the Actor appear at every single position along the way? In addition, what is something that looks similar to the glide block, but just isn't exactly there?

One of the first blocks that we used was the "move...steps" block, but unfortunately instead of showing every single step, it would skip from the first step to the last step. However, consider the following: If your glide function moves the Actor forwards 30 steps, wouldn't that be exactly the same as moving the Actor forwards 10 steps three times in a row? Or maybe forwards 5 steps six times in a row? Let's try it out.

Step 1: Drag the "When Flag Clicked" tab into the Action area.
Step 2: Drag 7 of the "move 10 steps" blocks under the "When Flag Clicked" block, all directly under each other.
Step 3: Under the Control tab, select the "Wait 1 secs" block and insert one in between every two blue "move 10 steps" blocks.



Step 4: Press the Green Flag in the Preview area.

What seems to happen this time? For comparison, you can remove the long chain of blocks and replace it with one single "move 70 steps" block. The difference is that when there are different sections, the Actor moves slowly, showing every intermediate position, while the single block seems to teleport the Actor from one position to another. Therefore, it is possible to claim that the "glide..." block is just lots of really small "move...steps" blocks put together!

However, as you probably realized, it is really hard and boring to keep on dragging the same blocks over and over again. Is there a way for the computer to do this for you, leaving you with more time and energy to do some cooler activities?

4.2: Repeating Again and Again

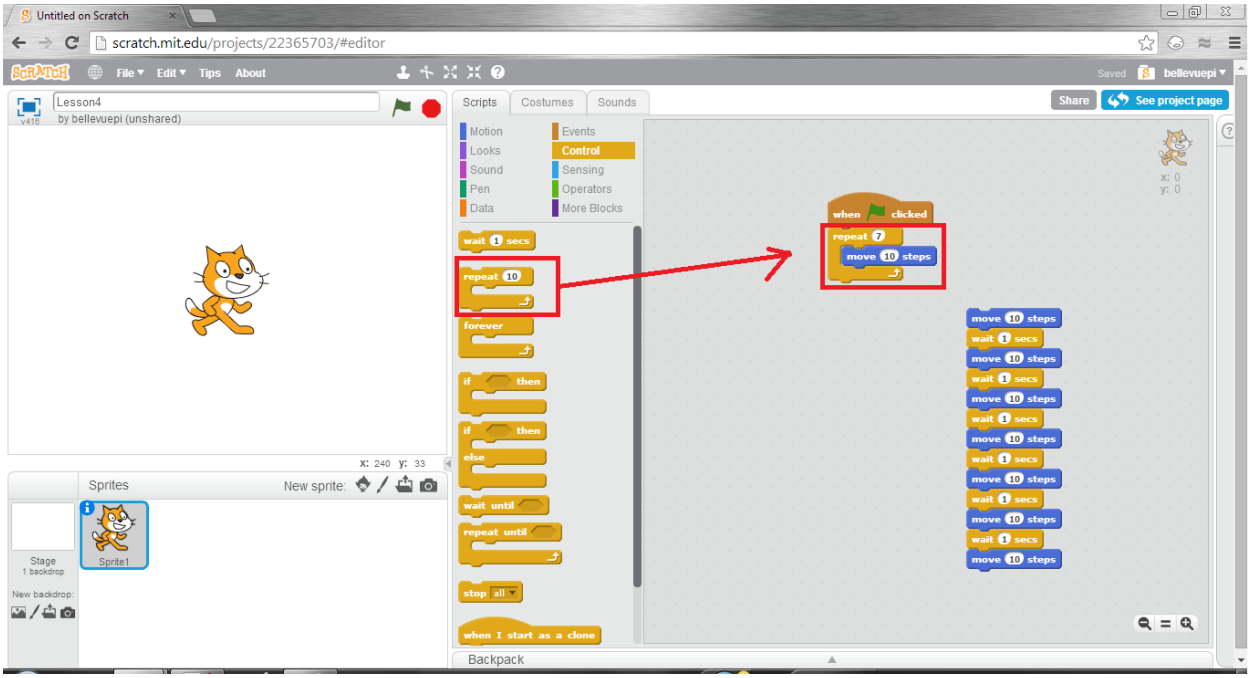
Astute students may have noticed the "repeat ... ." block right under the "wait... secs" block. This block does exactly what it sounds like: It repeats an action however many times you want it to. Let's try it out to make something similar to what we just did.

Step 1: Drag away the long "move..." and "wait.." blocks to the side.

Step 2: Under the Control tab, select the repeat ... ." block, and place it directly under the "When Flag Clicked" block.

Step 3: Edit the number to be 7

Step 4: Drag a single "move ... steps" block inside of the "repeat... ...." block



Step 5: Click the Green Flag and see the effects of your program.

As you should be able to see, the Actor performs the same actions as before, but you don't have to do nearly the same amount of repetitive tasks as before. The repeat function allows for tedious tasks to be performed over and over again until something is done.

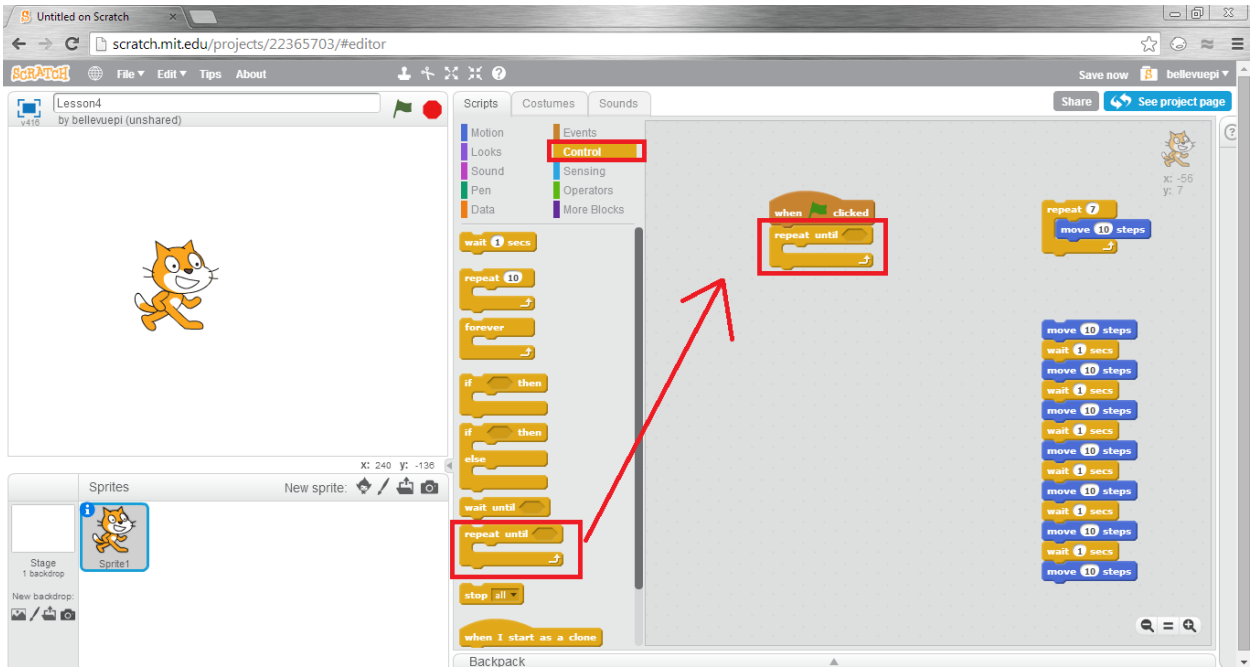
4.3: Not Doing Things

Just as important as doing things is, sometimes it becomes necessary to see if something won't happen. For example, suppose your mom asks you to clean the table only if it is not clean right now. For a regular person, they would only check the table every once in a while to see if it is clean or not, but computers can't tell. Instead, they would need to constantly check if the function is not being done.

This action can be accomplished in two ways: In an if/then/else loop, where it is repeated over and over again in the else portion. However, more conveniently we can use the "not..." block. This block has one function only: It turns true things false and false things true. Therefore if I ask you "Is 3 NOT less than 4", you would say False! Because "3<4" is true, "not 3<4" is false. Conversely, if you say is 5 NOT less than 4, you would say true! Because "5<4" is false, "not 5<4" is true!

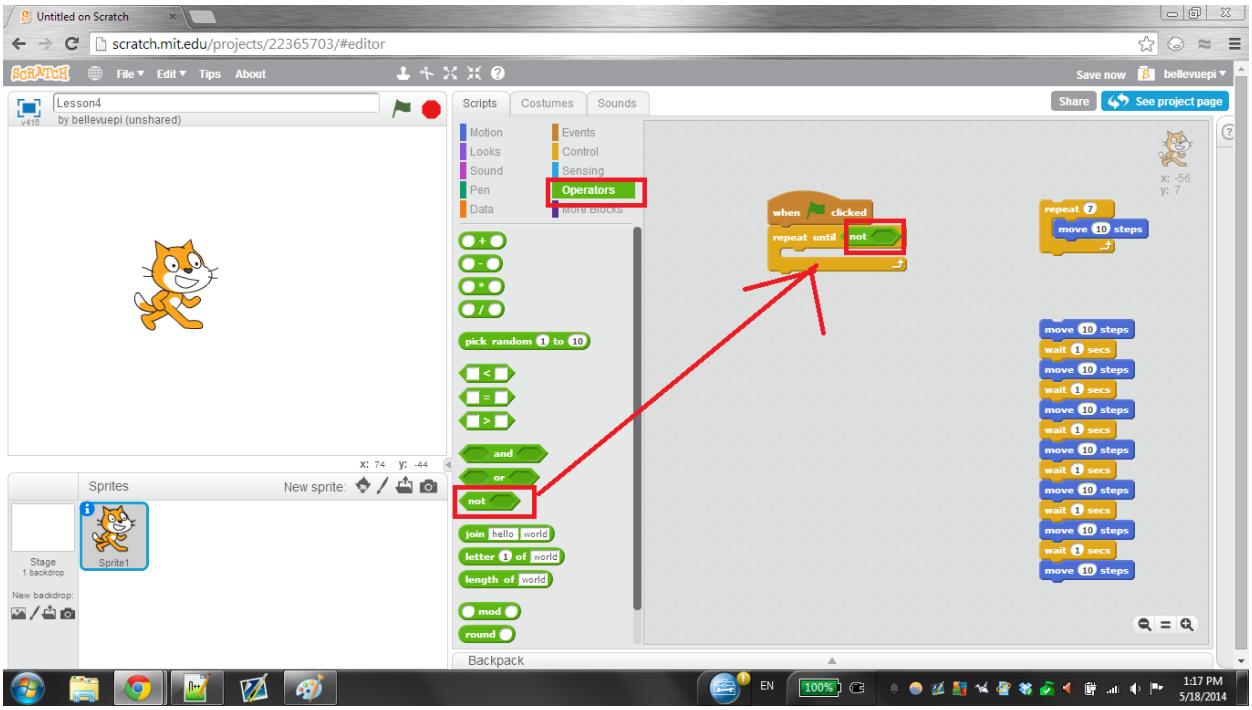
Let's use the NOT block now, in conjunction with the "repeat until..." block. There are lots of new concepts here, and hopefully you will take some time to understand each new block on your own.

Step 1: Delete all of the blocks under the "When Flag Clicked" block
Step 2: Under the Control tab, drag in the "Repeat until..." block and place it under the "When Flag Clicked" block.



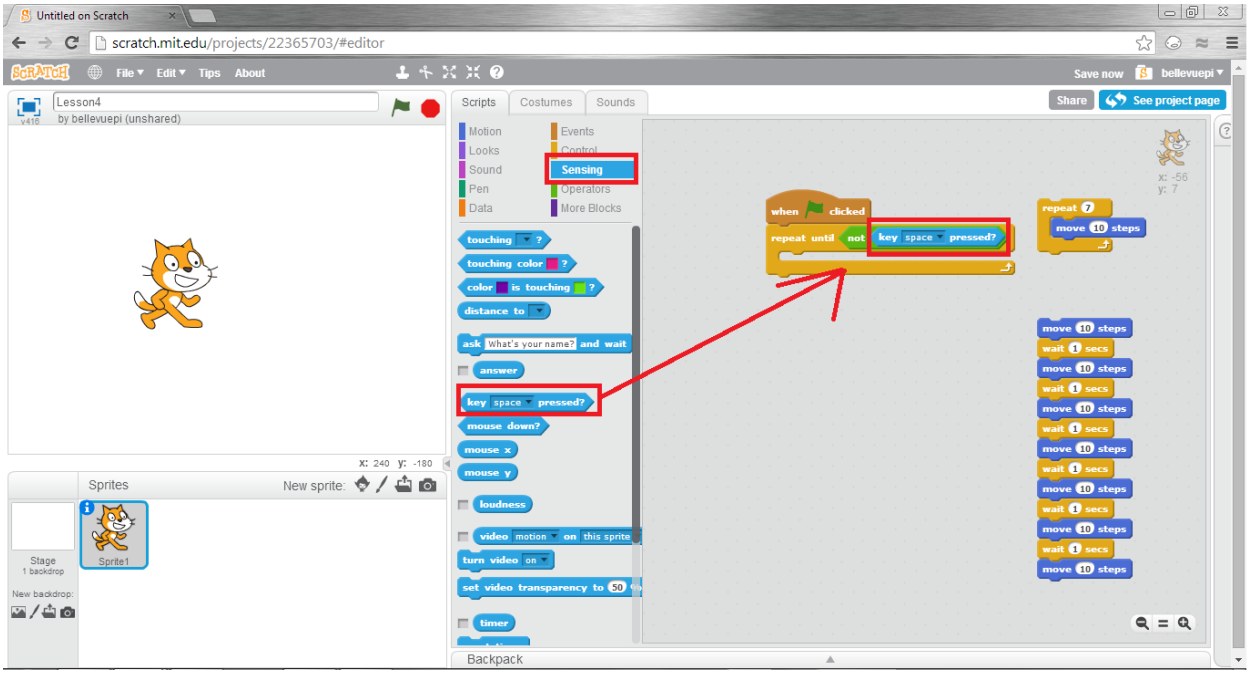
Step 3: Under the Operators tab, select the "not..." block and drag it into the "repeat until ..."

block.



Step 4: Under the Sensing tab, drag the "key Space pressed" block and place it into the "not..."

block.



Step 5: Under the Motion tab, drag the "move 10 steps" block into the "repeat until ... ..." block.

Step 6: Hold the Space key down, and then press the Green Flag. Try releasing the space bar. What do you notice?

4.4 Extension

We will allow for you to explore the rest of these blocks on your own. For a final project, please consider ways to nest loops in order to make more things happen! Try using nested loops and investigating the “repeat forever” loop. Why do you think programmers might be interested in using such a loop?