Programming with Scratch

The Basics

Created By: Bridget O’Mara & Nyles Fowler

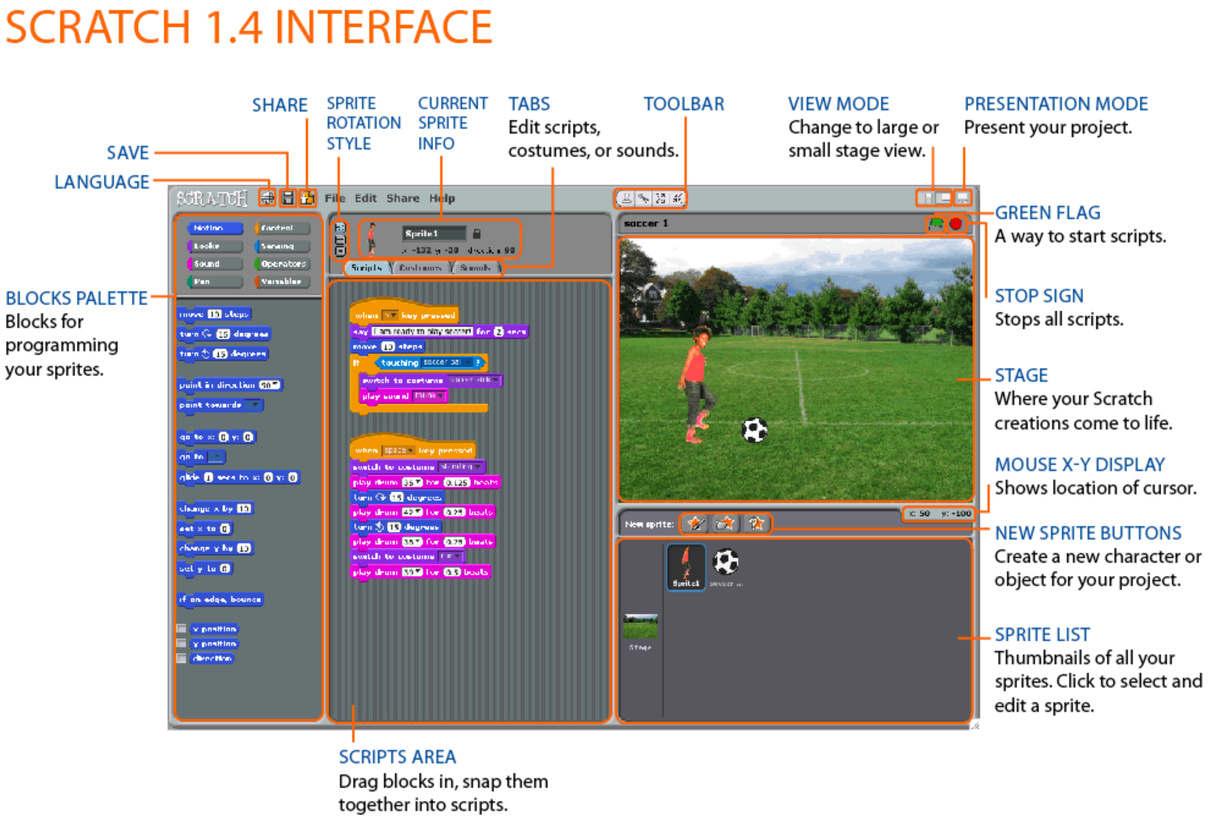
REGIS
UNIVERSITY
College for Professional Studies

Table of Contents

Introduction to Scratch

Scratch is a way to introduce programing in a fun and inventive way to anyone interested in how computers work. With the scratch program you can create games, videos, and your own virtual world. In order to accomplish such feats though it is important to understand how scratch works.

Scratch Program Diagram:



SCRATCH 1.4 INTERFACE

SHARE

SPRITE ROTATION STYLE

CURRENT SPRITE INFO

TABS
Edit scripts, costumes, or sounds.

TOOLBAR

VIEW MODE
Change to large or small stage view.

PRESENTATION MODE
Present your project.

SAVE

LANGUAGE

GREEN FLAG
A way to start scripts.

STOP SIGN
Stops all scripts.

BLOCKS PALETTE
Blocks for programming your sprites.

STAGE
Where your Scratch creations come to life.

MOUSE X-Y DISPLAY
Shows location of cursor.

NEW SPRITE BUTTONS
Create a new character or object for your project.

SPRITE LIST
Thumbnails of all your sprites. Click to select and edit a sprite.

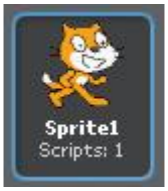
SCRIPTS AREA
Drag blocks in, snap them together into scripts.

The First thing you should know about scratch is that it is a drag-drop program. This means that to use the code blocks (in the Blocks Palette) you must drag the block over to the Scripts Area using the mouse.

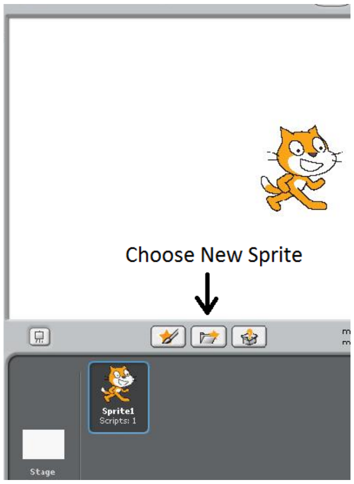
Sprites

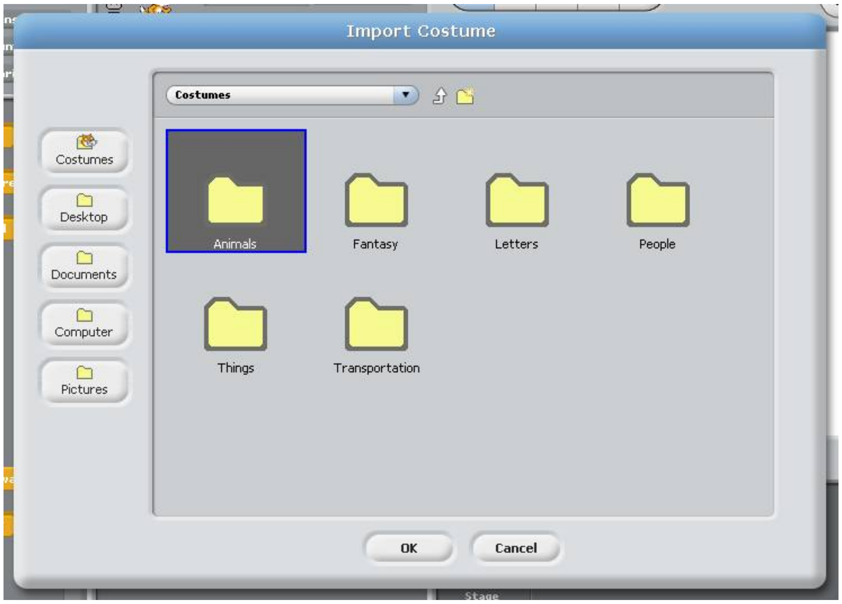
When you open scratch you will notice a happy cat standing in the screen.

Notice that this cat is labeled as sprite1

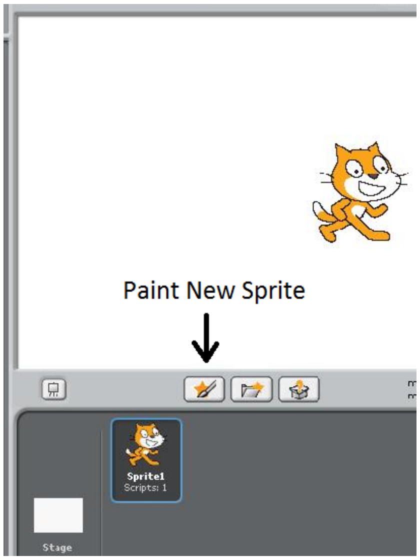


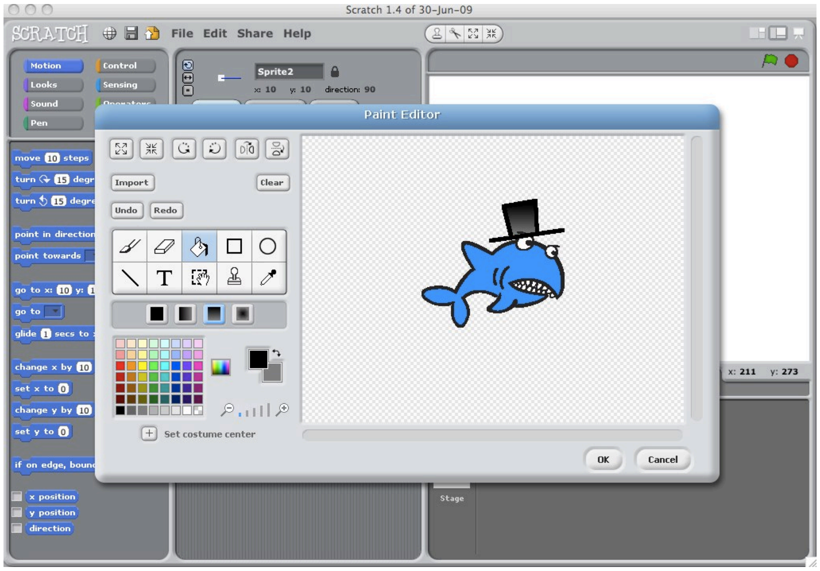
The sprite doesn’t have to be a cat, if you click on the “choose new sprite from folder” icon you can select another sprite from the folders seen on the right below.



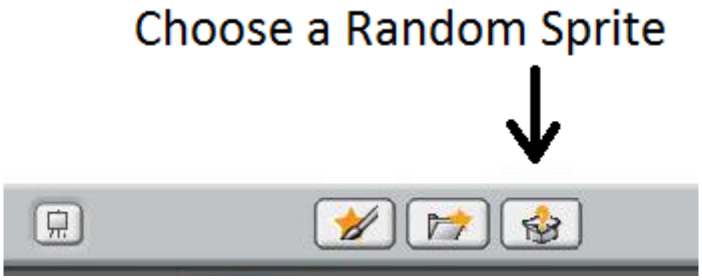


You can make your own sprites too! Click on the “paint new sprite” icon. The paint editor will appear, and you can draw your own sprit!

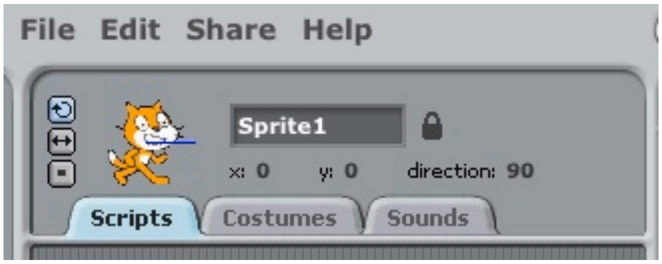




The last icon is the “surprise sprite” icon. Click it and you’ll get a completely random sprite.



Now that we know about sprites we can move on to learn about scripts, costumes, and sound.

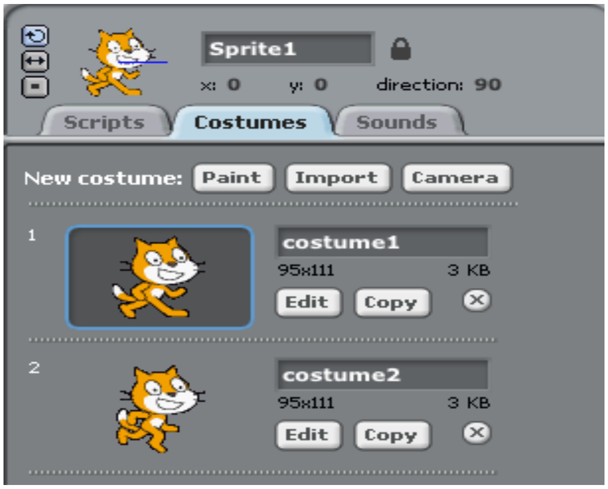


Scripts

Scratch works as a drag-drop program. This means that the code blocks (find these in the blocks palette, p. 1) are dragged over to the blocks palette with the mouse. The script area allows for people to make commands for their sprite to follow. They can make the sprite move, change color, or sing a song if they wish. You can think of this area as a place where you make own recipe with ingredients of your choice for a sprite to follow.

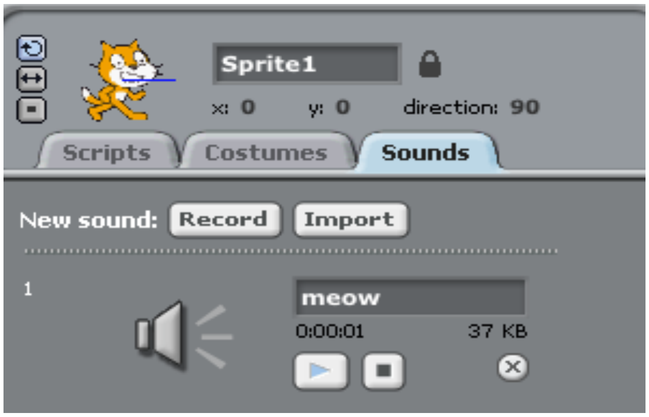
Costumes (if you don’t see this button try clicking on your sprite and see if it appears)

The costumes button allows for you to change your sprite’s entire appearance. This tool is nice if you want to create short videos. Adding a new costume is similar to adding a new sprite, you can either import a new costume or create your own.

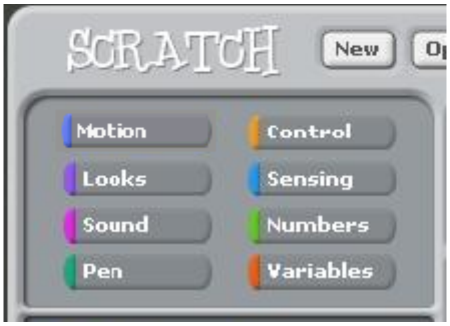


Sound

The Sound button allows you to record your own sound and use it in your scripts.



Now you've probably have noticed a bunch of buttons that look like this:



Here is what these tabs are:

The Control Tab

The control tab tells your sprite when to do something and how long to do something. What your sprite does depends on the other tabs (motion, looks, sound, pen, sensing, numbers, and variables).

The Motion Tab

The motion tab tells your sprite how far to move and what direction to move.

The Sensing Tab

The sensing tab allows for interactions between sprites, colors, time, and other variables. Feel free to experiment a great deal with this tab!

The Looks Tab

The looks tab can change the size, color, or costume of your sprite. This tab can also allow your sprites to communicate with each other in a comic book-like way.

The Numbers/Operators Tab

/

The numbers (might be labeled as operators instead) tab contains code blocks which make conditions for your sprite, so they only perform a certain action if the conditions are met.

The Sound Tab

The sound tab, as you may have guessed, creates sound within the script.

The Pen Tab

The pen tab leaves a trail behind your sprite as it moves. You can think of it as though your sprite is drawing on the screen.

The Variables Tab

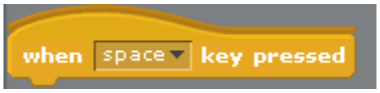
The variables tab allows you to add an extra feature to your script such as time, points, or lives.

Now that we have most of the basics covered, let's do something with our sprite!

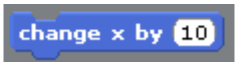
Lesson 1: Making the Sprite Move

Now remember that Scratch is a drag-drop program, meaning that to use the code blocks (in the Blocks Palette) you must drag the block over to the Scripts Area using the mouse.

To make the sprite move you'll need the following blocks:



```
when space key pressed
```



```
change x by 10
```

The orange block can be found under the control tab and the blue box can be found under the motion tab. Drag both of these blocks over to the script area and connect them. (Connect them by dragging the blue block over to the orange block until you see a white highlighted area, then let go of the block).

Click on the down arrow seen on the orange block and select "right arrow", your sprite should now move to the right whenever you press the right arrow.

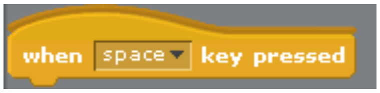
Notice that the blue box says "change x by 10". The change x part refers to the x-axis, like on a grid or a graph. The change x block moves your sprite to the right if the number x is changed by is positive (like 10), if the number is negative your sprite will move to the left.

If you want your sprite to move to the left when you hit the left arrow follow these steps:

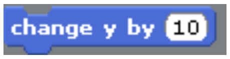
1) Right click on the orange block you have just connected with the blue block
2) Select "duplicate"
3) Place the duplicate blocks under the ones already in the scripts area
4) Change "right arrow" to "left arrow"
5) Change the number 10 in the blue box to -10

Click the left arrow key, your sprite should now move to the left

To get your sprite to move along the y axis (up and down) you will need the orange block we just used as well as a new block:



```
when space key pressed
```



```
change y by 10
```

Notice the blue block now says "change y by 10". When the number is positive in this block it means that the sprite is moving up, when the number is negative the sprite moves down.

Select the "up arrow" for the key pressed in the orange block

Connect the blue and orange blocks together

Press the up arrow key on your keyboard, your sprite can now move up!

To get your sprite to move down we will duplicate the blocks once more by right clicking on the orange block and selecting the “duplicate” option.

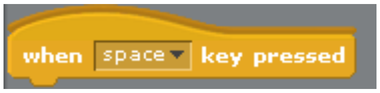
Next, we will change “up arrow” to “down arrow” in the orange block

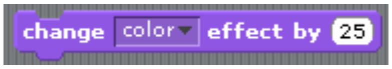
Finally, change the “10” in the blue block to “-10”.

Press the down arrow key on your keyboard, your sprite can now move in all directions!

Lesson 2: Changing the Sprite's Color and Costume, and Size

To change the sprite's color you will need the following blocks:


when space key pressed

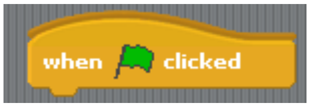

change color effect by 25

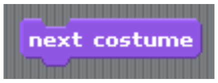
Connect the two blocks and press space. Your sprite's color will change!

To change the sprites costume you must first make a new costume. So click on the costumes button and select "import". You can select any costume you want, we will be selecting a sprite called cat 1-b from the animal folder.

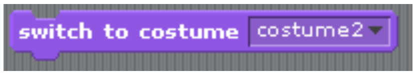
>Picture

Now that we have two costumes we want to be able to switch between them. We will need the following blocks:


when clicked


next costume

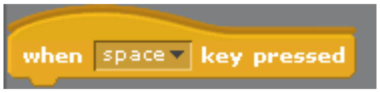
or

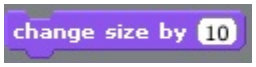

switch to costume costume2

Put either of the two purple blocks with the orange one. The only difference is that the one on the right allows you to switch to a specific costume, while the one on the left will go to the next costume in line.

If you choose to use the purple block on the right remember to make sure the costume it is switching to is the one you want.

To change the size of the sprite select these blocks:


when space key pressed


change size by 10

Connect these two in the scripts area.

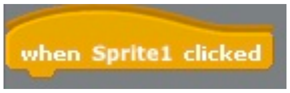
Because you already have the space bar being used to change color you will need to set the "key pressed" to something else. You may select whatever button you wish for this, we will use the "Z" key. When finished simply press the key you selected (for our example this key would be Z) and see your sprite grow!

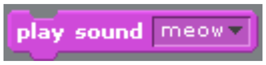
To get your sprite back to normal change the “10” in the purple block to “-10” or select the shrink sprite button

>Picture of shrink sprite button

Lesson 3: Sound

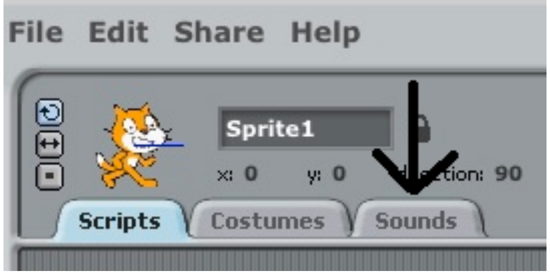
To get sound we suggest the following blocks:





Connect these two together in the scripts area and click on your sprite. You should hear it meow. You can get it to make other sounds too though.

Click on the grey Sounds button



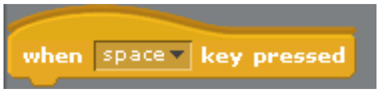
Now click record. You will see a box appear. This box will record any sound you make. Click the red stop button to start recording, when you are done hit the stop button again and click “okay.”

Click the down arrow on the purple block that says “play sound meow”, you should now see that you can select your recorded sound to play.

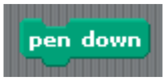
Lesson 4: Using the Pen Tool

Part 1: Getting started

To use the pen tool you will need the following blocks:



when space key pressed



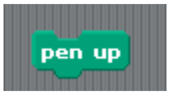
pen down

It should be noted that this lesson is assuming that you already have a script that allows your sprite to move around, if your sprite cannot move around yet please see **Lesson 1: Making the Sprite Move** above.

1) Connect these two blocks together in the scripts area
2) Change the key pressed in the orange block to X
3) Press X and move around, you will notice a trail following your sprite wherever you go

Part 2: Turning the pen tool off

To turn off the pen tool you will again need the orange block that says "when ___ key pressed" (the blank should have a "space" in it by default. You will also need a new block:



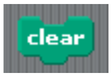
pen up

As you may have guessed, "pen up" turns off the pen tool while "pen down" turns the pen tool on.

1) Go ahead and connect this block with the orange block that says "when ___ key pressed".
2) Now change "key pressed" to "V".
3) Press "V" and move around, your sprite should no longer leave a trail wherever it goes

Part 3: Clearing the Pen trail

To clear the pen trail you have made again select the orange block saying "when ___ key pressed", you will need the following new block:



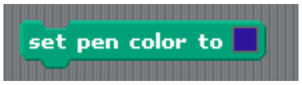
clear

1) Can you guess step one? If you guessed connecting this block to the orange block your right! Connect the two blocks together.
2) Next step is changing the "key pressed" to "C".
3) Press "C", any trail you made should now be gone.

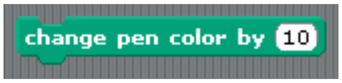
Part 4: Changing the Color of the Pen

Note: You must have at least completed “Part 1” of this lesson before preceeding.

To change the color of the pen, select the orange block that says, “When ___ key pressed” and the new block:


set pen color to

or


change pen color by 10

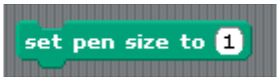
The difference between these blocks is that the one on the left will allow you to select a specific color. The one on the right, however, will go through the colors of the rainbow.

1) Connect either the right block or the left block with the orange block. (Do not connect both)
2) Change “Key pressed” to “A”
3) If you chose the left block click the color in that block and choose your favorite color instead (if your favorite color is already in there choose your second favorite color). If you did not choose the left block then move on to the next step.
4) Make sure the pen is on by pressing “X” and moving around, if you see a trail behind you move on, if not go to “part 1” of this lesson.
5) Press “A” and move around. If you chose the left block for this your pen’s color will now be your favorite color. If you chose the right block your pen should be slightly different from before because it is moving through the colors on the rainbow, keep pressing “A” to see other colors appear

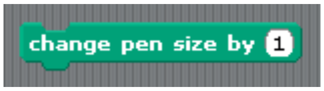
Part 5: Changing the Size of the Pen

Note: You must have at least completed part one of this lesson before preceeding.

To change the size of the pen select the “When ___ Key pressed Block” once more and the new block:


set pen size to 1

or


change pen size by 1

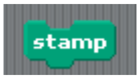
The difference between these two blocks -like the color blocks- is that the one on the left set the pen to a specific size while the one on the right changes it from what it already is.

1) Connect either the left or right block to the orange block (do not connect both)
2) Change “key pressed” to “S”
3) Make sure the pen is on by pressing “X” and moving around, if you see a trail behind you move on, if not go to “part 1” of this lesson.
4) If you selected the left block change the “1” to your favorite number (if 1 is your favorite number change it to your second favorite number). If you did not select the left block move to the next step.

5) Press “S” and move around, your pen should now be a different size than before. If you chose the right block you can continue to press “S” to make your pen larger.
6) If you chose the right block you can shrink your pen by replacing the “1” with a negative number.
   Remember that you can clear the pen trails by pressing C! (if you can’t then see part 3 of this lesson).

Part 6: The Stamp Tool

The stamp tool allows you to make copies of your sprite. To use it select the “When ___ Key Pressed” block and the new block:

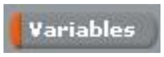


1) Connect the new block with the orange block
2) Now change the “Key Pressed” to “D”
3) Press “D” and move around, you should see that a copy of your sprite has been made!

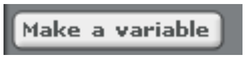
Unlike the other pen features, the stamp tool can be used even if the pen is off (turned up).

Lesson 5: Adding a Variable

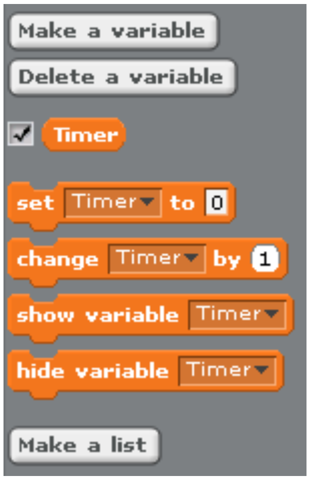
1) To add a variable is fairly simple, First click “Variables”



2) Now click “Make a Variable”



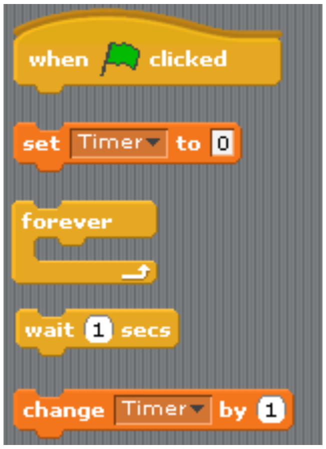
3) Now just type in the name that you want to give your variable, we recommend using the name “Timer” because that’s what the following parts of this lesson will use.
4) You should see options that say “For all sprites” and “For this sprite only” for now keep it as “For all sprites” and click “okay”. You should now see the following on your screen:



Note: if you named your variable something other than timer the variable blocks will show you the name you put down instead of “timer”.

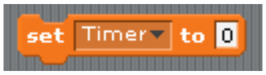
Part 2: Doing something with the variable

There are a great number of things you can do with a variable. At the moment we will show you how to make a timer with a variable. You will need the following blocks:



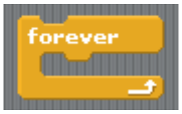
```
when ⚑ clicked
set Timer ▾ to 0
forever
wait 1 secs
change Timer ▾ by 1
```

You may have noticed a few new blocks here, before we continue we'll introduce you to a few of them.



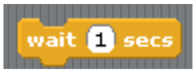
```
set Timer ▾ to 0
```

This block sets your timer to a specific value. We will keep it at zero for now.



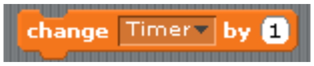
```
forever
```

This is a forever loop, everything in this loop is repeated until you click the stop button 🛑



```
wait 1 secs
```

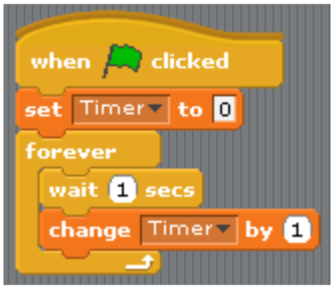
This block tells your script to wait a specific amount of seconds before performing an action. We will keep it at one.



```
change Timer ▾ by 1
```

This block tells the script to change the timer's value by a specific amount. For example if we change the timer by one when it is currently at 0 the timer will then have a value of 1. We will keep the number changed as one for now.

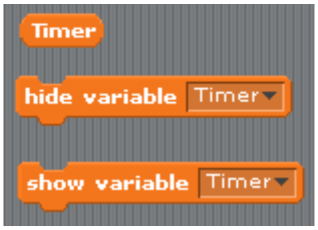
Put the blocks together in the following way:



```
when [flag] clicked
set Timer to 0
forever
    wait 1 secs
    change Timer by 1
```

Now whenever you click the green flag the timer will start at zero and move up one every second.

You may notice other blocks that look like this:



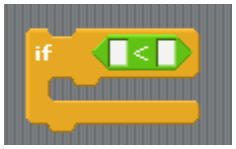
```
Timer

hide variable Timer

show variable Timer
```

The Timer block (top) is mostly combined with operators/numbers, what this means will be explored more in the next lesson.

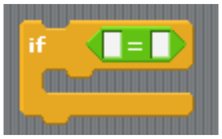
The hide variable block (middle) simply hides the variable (or Timer) from view, whereas the show variable block (bottom) shows the variable.

Lesson 6: Numbers/Operators

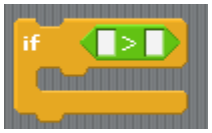
The numbers/operators function adds conditions to your script. What this means is that whenever you use something like an “if” block the number/operator blocks complete the statement, for example:



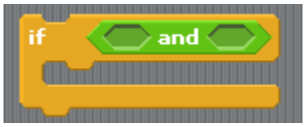
This script is saying: “if _ is less than _ then…”



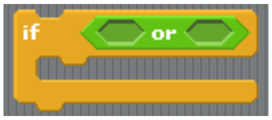
This script is saying: “if _ is equal to _ then…”



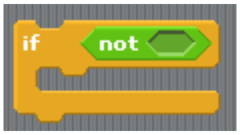
This script is saying: “if _ is greater than _ then…”



This script is saying: “if _ and _ then...”

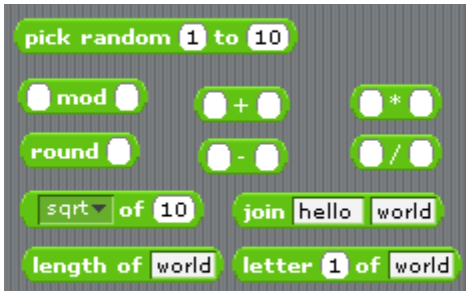


This script is saying: “if _ or _ then…”

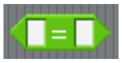


This script is saying: “if not _ then…”

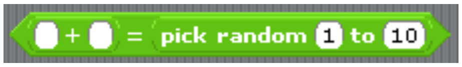
It is important to notice the shape of each of these operators as well. See how they are hexagons? They fit into the “if” statement because of this, whereas these operators would not fit into the “if” statement:



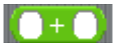
These operators fit into slots with more of a square shape like in the “_is equal to_” block:



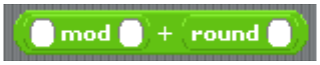
→ put in two blocks from above →



You can also fit these operators within one another:



→

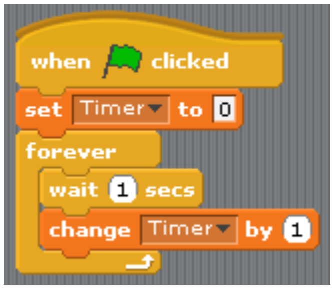


Now these are the blocks that can make your script become complicated fast!

Now that we know a little more about operators, let’s do something with them. Remember the timer we made in the variables section? We are going to make the timer count down.

Lesson 5 should be completed before moving on

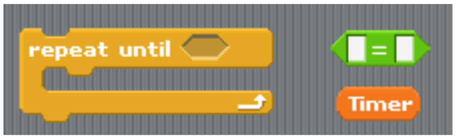
1) You should have a script that looks like this:



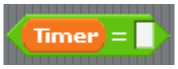
```
when [flag] clicked
set Timer to 0
forever
    wait 1 secs
    change Timer by 1
```

If you do not have this script please go to lesson 5

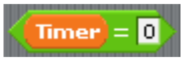
To get your timer to countdown you will need the following blocks:



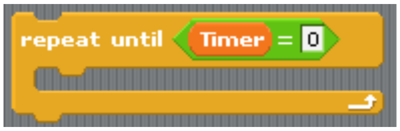
2) Place the “Timer” block into one of the areas in the “_ = _” block



3) In the other area of the “_ = _” block type in “0”

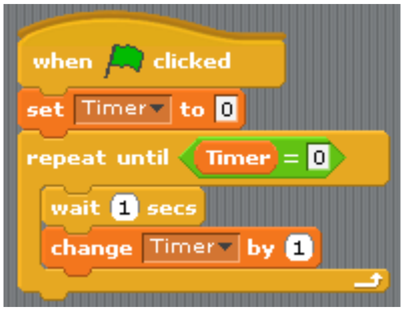


4) Place the "_ = _" block into the "repeat until _" block



```
repeat until <(Timer) = 0>
```

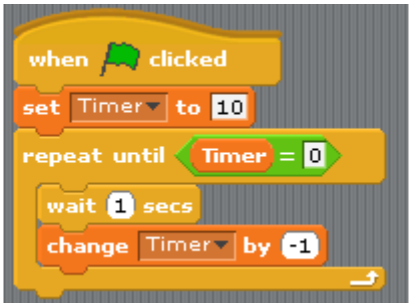
5) Replace the "forever loop" block with the "repeat until _" block. Do this by:
Detaching the forever loop from the rest of the blocks
Delete the "forever loop" block
Place the "repeat until _" block under the "set timer to _" block
Place the blocks "wait _ secs" and "change _ by _" in the "repeat until _" block:



```
when ⚑ clicked
set Timer to 0
repeat until <(Timer) = 0>
    wait 1 secs
    change Timer by 1
```

6) All we need to do now is change a few numbers. Where it says "set Timer to 0" replace the "0" with "10"

7) Where it says "change Timer by 1" replace the "1" with "-1":

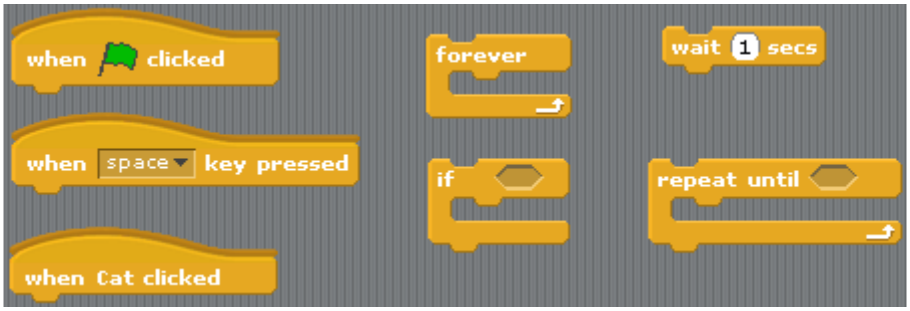


```
when ⚑ clicked
set Timer to 10
repeat until <(Timer) = 0>
    wait 1 secs
    change Timer by -1
```

You're all done! Hit play and see your timer countdown from ten until it reaches zero. Enjoy!

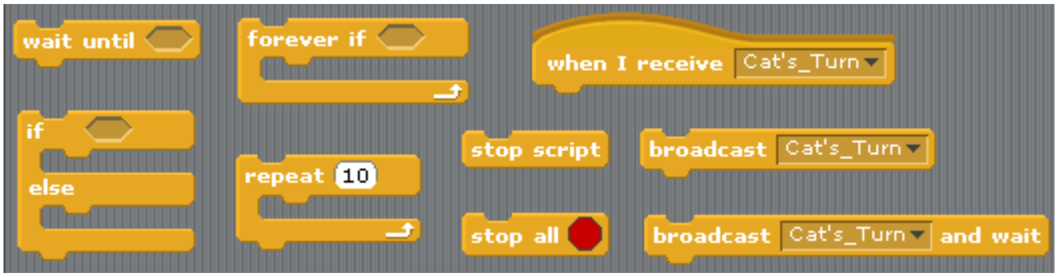
Lesson 7: What the other Control Blocks do:

The Control blocks we have seen so far are:



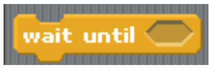
```
when ⚑ clicked

when space key pressed

when Cat clicked

forever

if < >

wait 1 secs

repeat until < >
```

We are now going to look at the other control blocks and explain what they do. Here are the other Control Blocks:



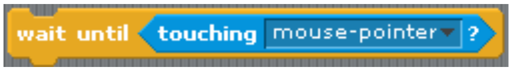
```
wait until < >
forever if < >
when I receive [Cat's_Turn]
if < >
else
repeat (10)
stop script
broadcast [Cat's_Turn]
stop all
broadcast [Cat's_Turn] and wait
```

The "wait until _" block:



```
wait until < >
```

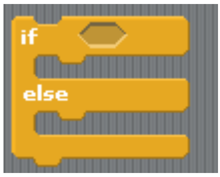
This block is very similar to the "wait _ secs" block, the only difference is that it can make your sprite wait for anything instead of just time. For example:



```
wait until <touching [mouse-pointer] ?>
```

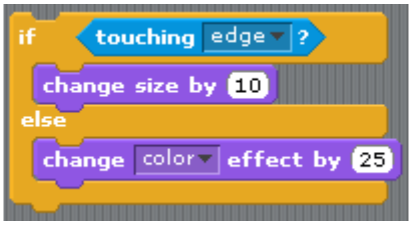
This arrangement of blocks tells your sprite to wait until it is touching the mouse to perform an action.

The "if _ else" block:



```
if < >
else
```

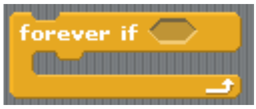
The "if _ else" block is much like the "if _" block, but with a twist. If the condition placed near "if" is not met then whatever is under "else" will happen, for example:



```
if <touching [edge] ?>
  change size by (10)
else
  change [color] effect by (25)
```

This arrangement of blocks is saying that if your sprite is touching the edge then it will change its size by 10 (so it will get bigger), but then if it isn't touching the edge it will change its color. You can see this for yourself by arranging the blocks like we have above and double-clicking on the whole script.

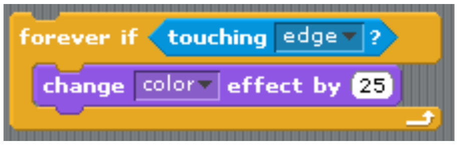
The "forever if _" block:



```
forever if < >
```

You may be able to guess what this block does, so go ahead and make a quick guess.

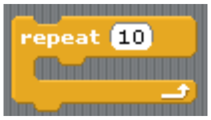
If you guessed that this block is something like an "if _" block and forever loop put together you are absolutely correct! This block will repeat an action forever if a condition is met, for example:



```
forever if <touching [edge v] ?>
    change [color v] effect by (25)
```

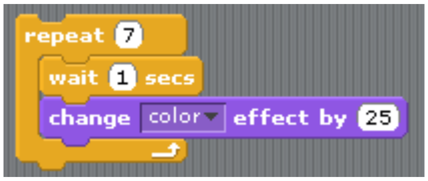
This arrangement of blocks is saying if your sprite is touching the edge then change its color forever. Go ahead and try this with your sprite touching and not touching the edge.

The “repeat _” block:



```
repeat (10)
```

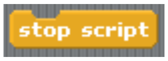
The “repeat _” block is very similar to the “repeat until _” block. The only difference is that it repeats for a given number of times instead of repeating until a condition is met. For example:



```
repeat (7)
    wait (1) secs
    change [color v] effect by (25)
```

This arrangement of blocks will repeat the process of waiting 1 second and changing the color of the sprite 7 times. Go ahead and try for yourself!

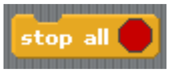
The “stop script” block:



```
stop script
```

This block tells the script it is attached with to stop.

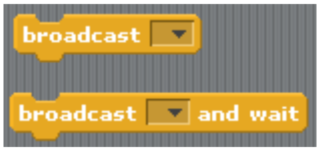
The “stop all” block:



```
stop all
```

This block tells everything to stop.

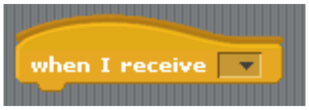
The broadcast blocks:



```
broadcast [ v]

broadcast [ v] and wait
```

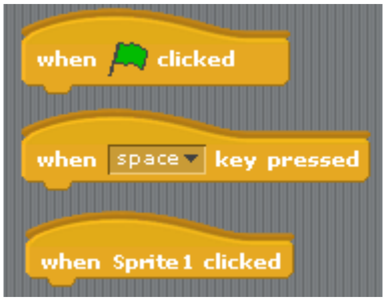
These blocks are a bit tough to explain, but they basically send a signal that lets your sprite know that they need to do something. These blocks are often used with the “When I receive _” block, which will be explained soon.

The “When I receive _” block:



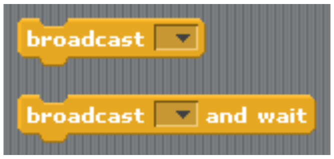
```
when I receive [ v]
```

This block is called a Hat block. It’s called this because it can only go on top of a script, like a hat. We have seen a few other hat blocks:



```
when [flag] clicked

when [space ▾] key pressed

when Sprite1 clicked
```

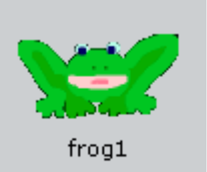
The “when I receive _” block is typically used with the broadcasting blocks:



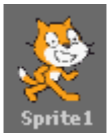
```
broadcast [ ▾]

broadcast [ ▾] and wait
```

Remember how the broadcasting blocks send signals? Well the “when I receive _” block captures those signals. To get a better idea of what this all means we are going to do an exercise where we make a conversation between two sprites:

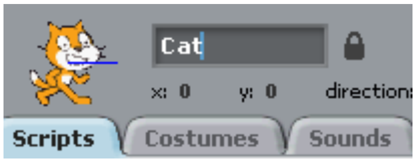
1) Click (add sprite option) and double click the “animal” folder.
2) Select the Frog:



3) Now we are going to change the Sprite’s names. First let’s change the Cat’s name. Click on sprite1:



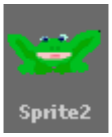
4) Go to the box that says “Sprite1” (above the Scripts Area) and replace “Sprite1” with “Cat”:



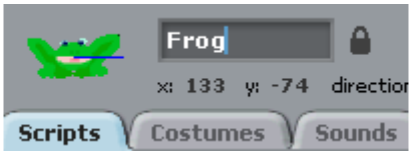
Cat

x: 0 y: 0 direction:

5) Now click on sprite2:



6) As we did with the cat, go to the box that says “Sprite2” and replace it with “Frog”

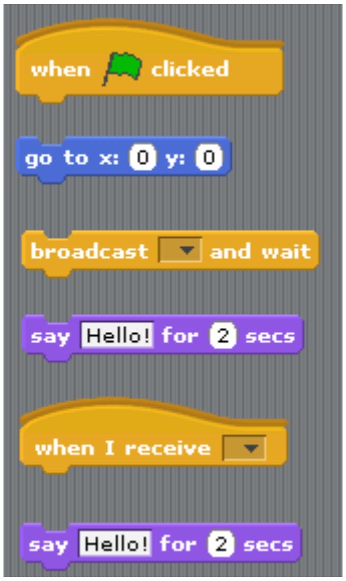


Frog

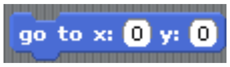
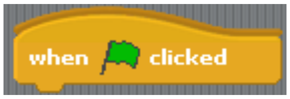
x: 133 y: -74 directior

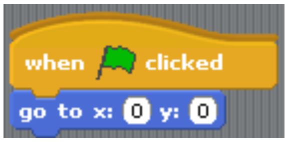
7) Now we need to get the scripts ready for both our sprites. Let's get the cat first, double-click on the cat

8) Place these blocks onto the Cat's Script Area:



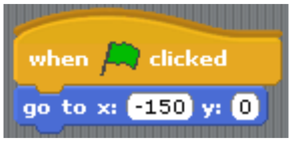
```
when [flag] clicked

go to x: 0 y: 0

broadcast [ ▼] and wait

say Hello! for 2 secs

when I receive [ ▼]

say Hello! for 2 secs
```

9) First we need to make it so the cat and the frog aren't overlapping, so connect  go to x: 0 y: 0 to  when [flag] clicked like so:

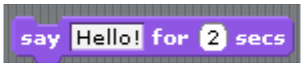
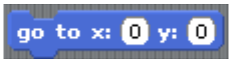


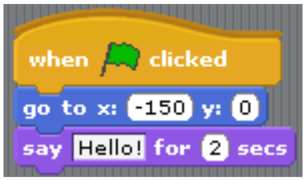
```
when [flag] clicked
go to x: 0 y: 0
```

10) Change the "0" by the x to -150:

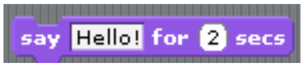


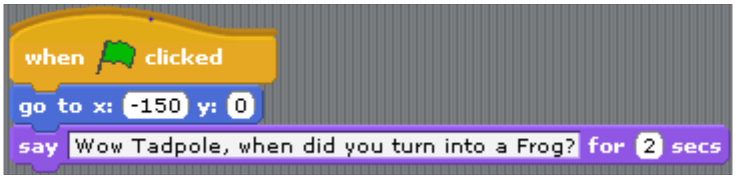
```
when [flag] clicked
go to x: -150 y: 0
```

11) Now place one of the  say Hello! for 2 secs blocks under the  go to x: 0 y: 0 block:

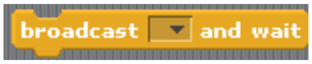
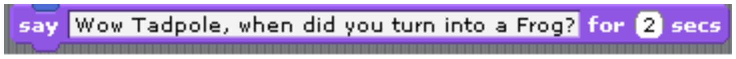


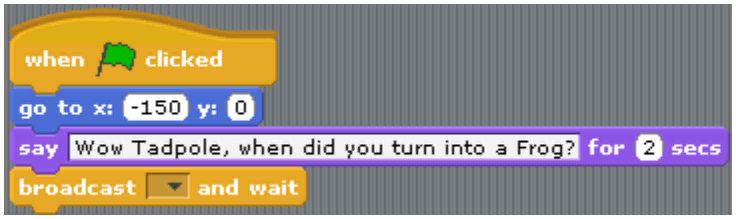
```
when [flag] clicked
go to x: -150 y: 0
say Hello! for 2 secs
```

12) Change the "Hello" in the  `say Hello! for 2 secs` block to "Wow Tadpole, when did you turn into a Frog?":

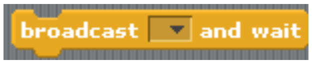


```
when [flag] clicked
go to x: -150 y: 0
say Wow Tadpole, when did you turn into a Frog? for 2 secs
```

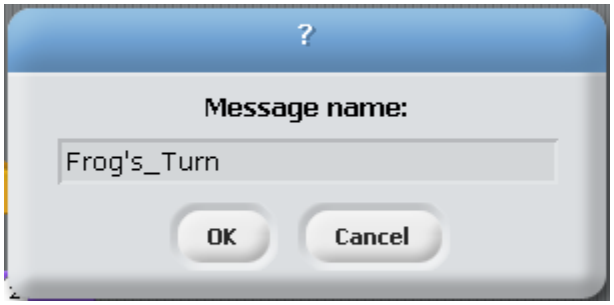
13) Place the  `broadcast [▼] and wait` block under the  `say Wow Tadpole, when did you turn into a Frog? for 2 secs` block:



```
when [flag] clicked
go to x: -150 y: 0
say Wow Tadpole, when did you turn into a Frog? for 2 secs
broadcast [▼] and wait
```

14) Click on the down arrow in the  `broadcast [▼] and wait` block and click "New"

15) A message name box should appear, type "Frog's_Turn" in it:



?

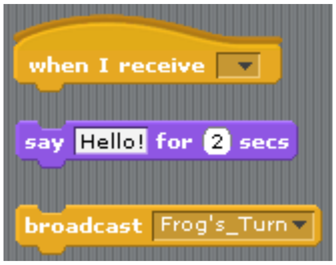
Message name:

Frog's_Turn

16) Click "OK"

17) We need to do a few things with Frog's script now so double click on Frog.

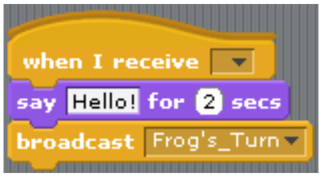
18) For Frog we need the following script Blocks:



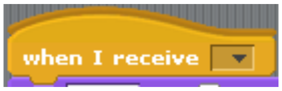
```
when I receive [▼]

say Hello! for 2 secs

broadcast Frog's_Turn ▼
```

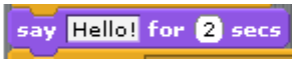
Drag and place these blocks on the script's Area

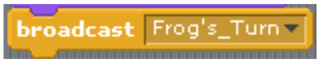
19) You can put all three of the blocks together like so:

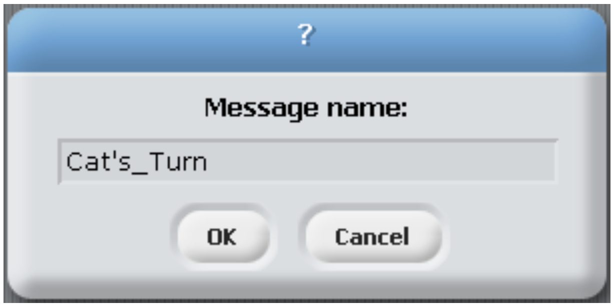


```
when I receive [▼]
say Hello! for 2 secs
broadcast Frog's_Turn ▼
```

20) In the  block click the down arrow and select “Frog’s_Turn”:

21) Replace the “Hello!” in the  block type in “About a week ago. I can play leap frog now, want to play?”

22) Finally, click on the down arrow in  and click “new”, in the message name box type in “Cat’s_Turn”:



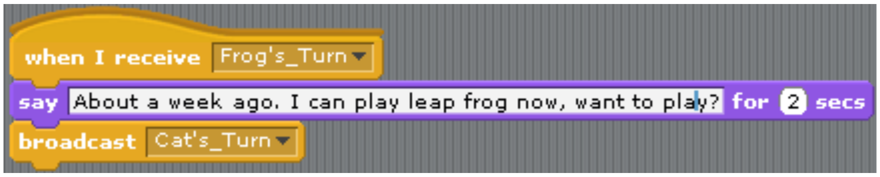
?

Message name:

Cat's_Turn

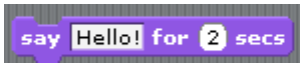
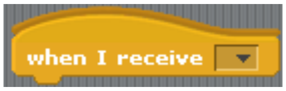
Click “OK”

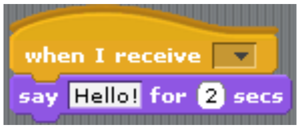
23) Your script for Frog should look like:



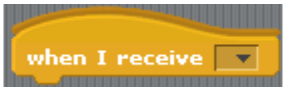
```
when I receive Frog's_Turn
say About a week ago. I can play leap frog now, want to play? for 2 secs
broadcast Cat's_Turn
```

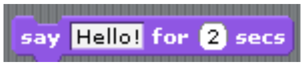
24) We need to go back to Cat’s script now, double click on cat.

25) Connect the  block to the  block:

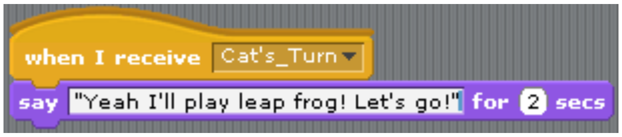


```
when I receive
say Hello! for 2 secs
```

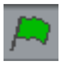
26) Click on the down arrow in the  block and select “Cat’s_Turn”

27) Change the “Hello!” in the  block to “Yeah I’ll play leap frog! Let’s go!”

28) Cat’s script should now look like:



```
when I receive Cat's_Turn
say "Yeah I'll play leap frog! Let's go!" for 2 secs
```

29) Click  and see what happens!

And that’s how the broadcasting blocks and receiving blocks work!

* You can continue the conversation between Cat and Frog by using the method above, feel free to get creative!*

Lesson 8: Using the "Sensing" tab

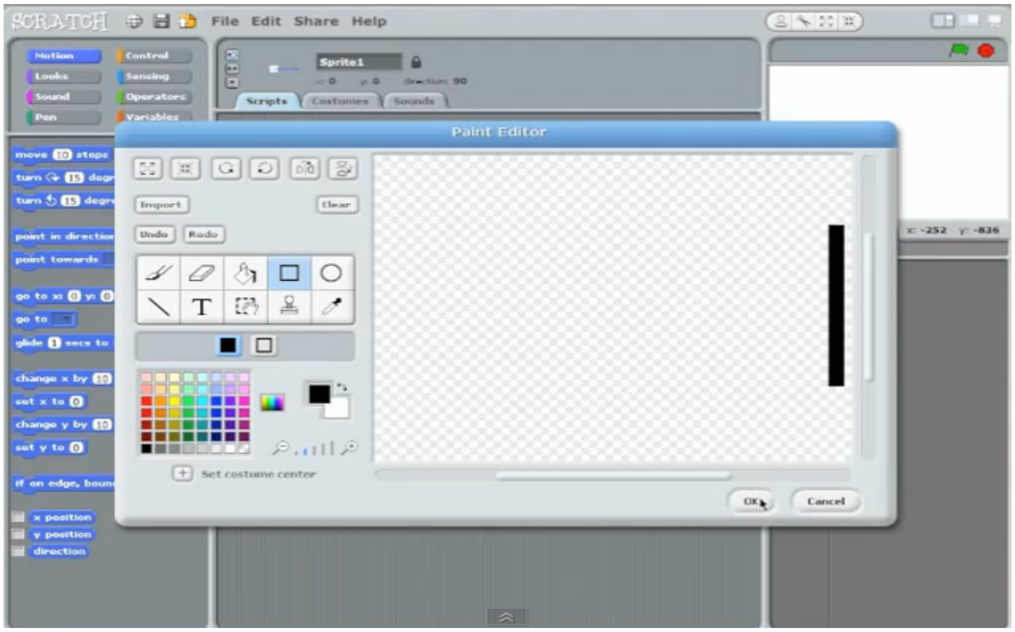
The sensing tab has lots of cool stuff in it.

Fun with Scratch!

Pong Game

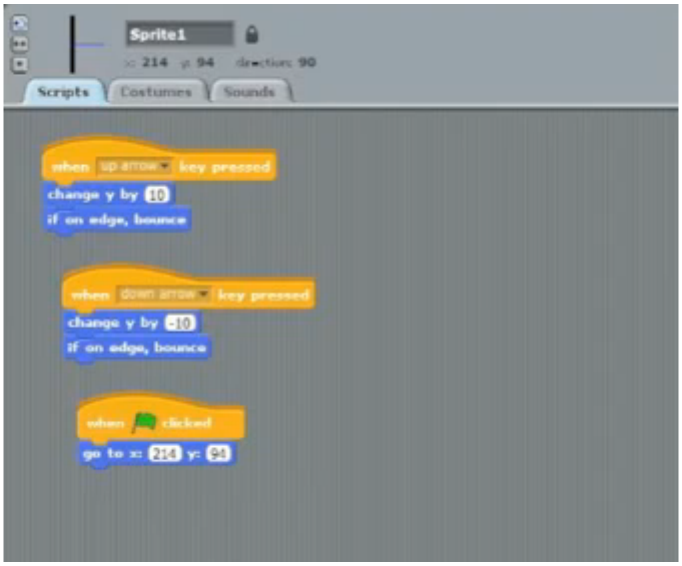
Creating the Panels:

1. First, right click on the cat that is there when you start up Scratch and click the “delete” button.
2. Then Click New sprite: and within that you will create a rectangle to serve as the panels in the game. It should look similar to this:



3. The next series of steps will assist you in making it and so you can control the panels with keys on the keyboard. First click the Control option and there you will select when space key pressed and instead of the “space” button you will select the “up arrow”.
4. Then, click the Motion option and there you will select change y by 10 and attach it to the bottom of the when space key pressed control.
5. You will complete repeat steps 3 and 4 for the “down arrow” control but you will change y by -10 instead of +10.
6. You can now use the arrow keys to test the movement of the panel.

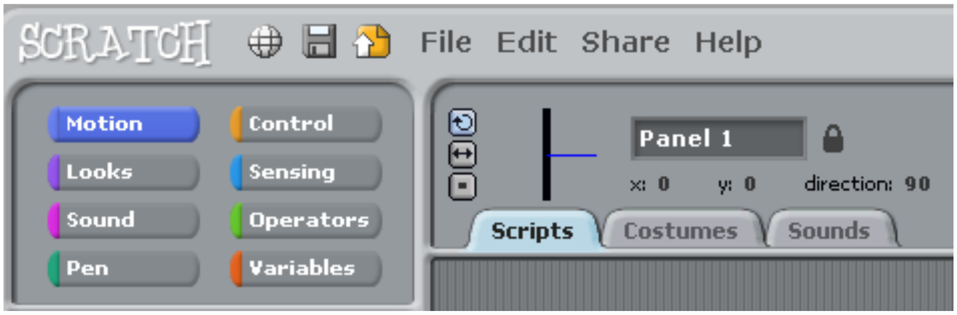
7. In order to make it and so the panel doesn't go past viewable space, click the Motion option and select if on edge, bounce. Do this for21 both the up and down arrow controls.
8. Then, in order to make the panel start at the correct position when the game is started, click the Control option and select when 🏁 clicked.
9. Then, click the Motion option and select go to x: 0 y: 0, attach it to when 🏁 clicked and set the coordinates to 214, 94. Your script should look like this:



Sprite1
x: 214 y: 94 direction: 90

```
when up arrow key pressed
change y by 10
if on edge, bounce

when down arrow key pressed
change y by -10
if on edge, bounce

when 🏁 clicked
go to x: 214 y: 94
```

10. Next, you will want to name your panel so navigate to the upper portion of the page and change the name of the sprite to "Panel 2" because it's on the right side of the page. Below is a picture of where you will rename the panel.

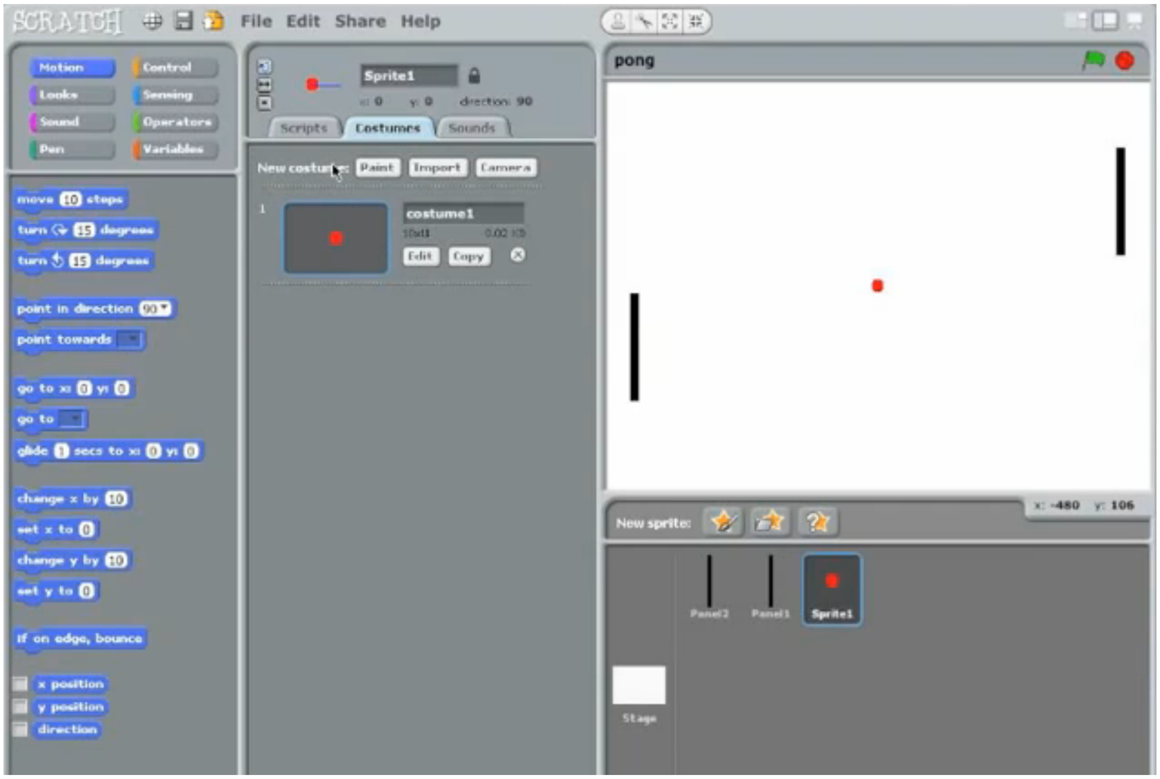


Panel 1
x: 0 y: 0 direction: 90

11. All you need to do now in order to get your second panel is right click the panel you made and click “duplicate”.
12. Select the duplicated panel and rename it to “Panel 1”.
13. Then, you will need to make slight modifications to the script by first changing the starting point coordinates to -214, -94. This will force the panel to start at the bottom left of the screen when the game is started.
14. Next, you will need to change the controls of the second panel and most common 2nd player controls would be “W” for up and “S” for down. Now you can test the progress of the game by clicking the button and using the controls you set to move both panels up and down.

Creating the Ball:

1. First click the button in order to start the creation of the ball. Use the color red and the circle option and create a generally small circle (you can change the size of the ball at any time if there are issues). Your view should now look like this:

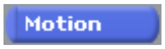
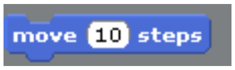


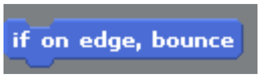
2. Now that the ball is created you will start the script. Click the Control button and there you will select the when clicked option.

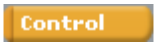
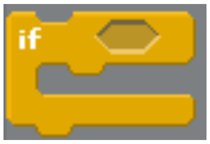
3. Then, you want the ball to start at a specific place so you click Motion and select the go to x: 0 y: 0 option and set it at 0,0. Attach this to the "when clicked" script.

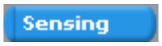
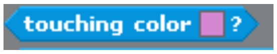
4. In order to make it and so the ball moves around and not just horizontally select point in direction 90▼ and set it to -45 degrees and attach it to the bottom of the "go to" script.

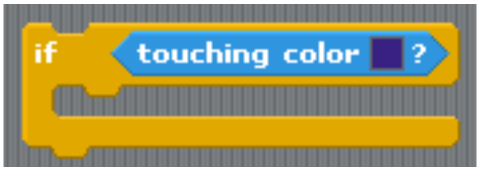
5. So the ball continuously moves, you will click Control and select the repeat until option and attach it to the bottom of the script that you have so far created.

6. Then, click  and select  and move it inside of the "repeat until" script so the ball moves, but instead of 10 steps change it to "5" so it's not too fast.

7. And so the ball bounces of each surface, attach  to the "move 5 steps" script but still within the "repeat until" script.

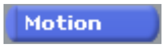
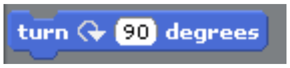
8. You will go to  and select the  option and place it within the "repeat until" loop.

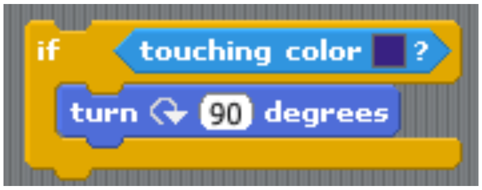
9. Because you want the ball to bounce off of certain surfaces you will go to  and choose the  option and place it in the "if" loop like this:



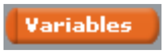
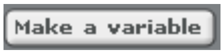
```
if touching color ?
```

10. If you click on the color box you will be able to select whichever color you want the ball to bounce off of and in this case you want it to bounce of the panels, so simply match the color with the color of the panels.

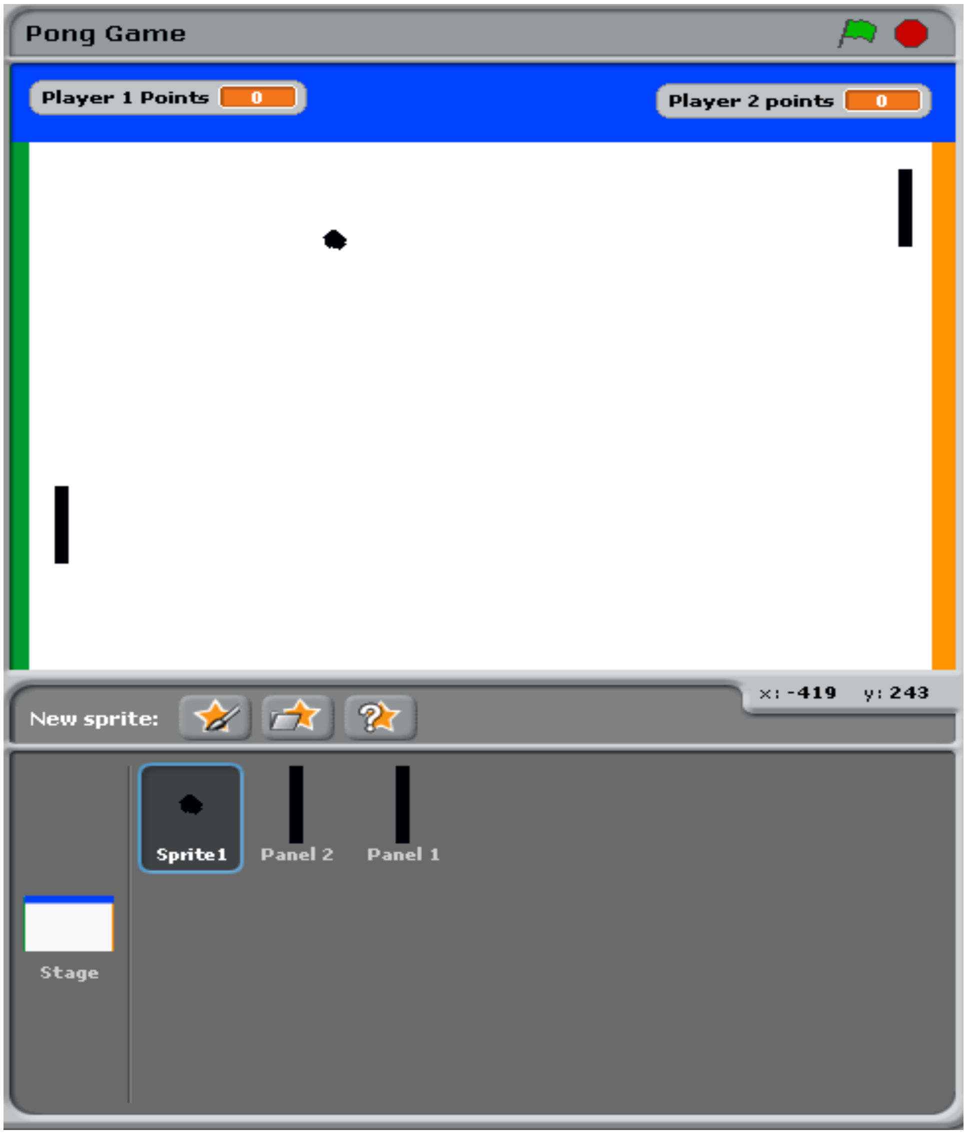
11. And so the ball bounces of the panels the way you want it to, go to  and place a  inside of the "if" loop like this:



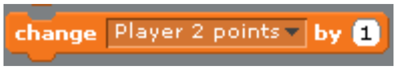
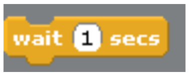
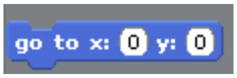
```
if touching color ?
  turn 90 degrees
```

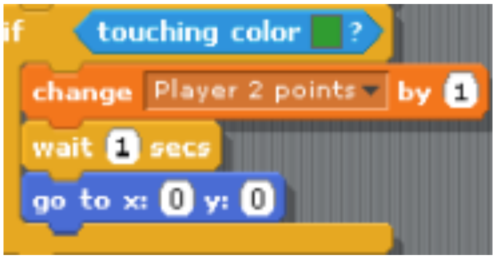
12. In order to track points within this games you will have to set up variables. Do this by going to  and selecting . The two variables that need to be created are "Player 1 Points" and "Player 2 Points"

13. Now we will be creating a better layout for the pong game and you will do this by clicking on "stage" in the bottom right corner of Scratch. You will then go to backgrounds and click the "edit" option. Here you will design rectangles of your choosing to end up looking something like this:



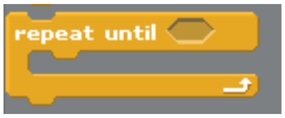
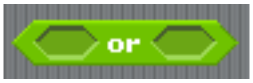
14. Then, go back to the ball script and now you have to make it and so the ball bounces off of the three colors. Repeat steps 8-11 for the color blue as it will act the same as the color black.
15. For the two colors on the sides (in this example green and orange) you will need to change script a bit to make sure points are counted. To start this off follow steps 8 and 9 for the two colors on the right and left.

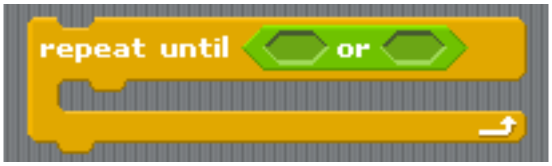
16. You will then go to Variables and chose the  change Player 2 points by 1 and put it inside of the "if" loop. You will also add  wait 1 secs and  go to x: 0 y: 0 to the script and so if you score the ball knows to start back in the middle and wait 1 second before starting up again. This script will look like this:



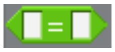
```
if touching color 🟩 ?
  change Player 2 points by 1
  wait 1 secs
  go to x: 0 y: 0
```

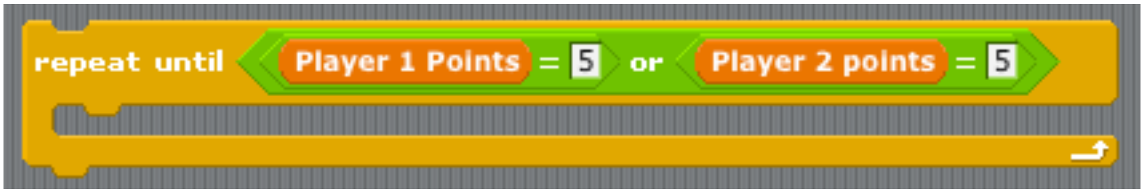
17. Make sure to do this for both right and left with the corresponding colors so the points are recorded correctly.

18. Then, in order to make sure there is a score limit, within the  repeat until loop go to Operators and choose the  or option and place it into the "repeat until" loop like this:



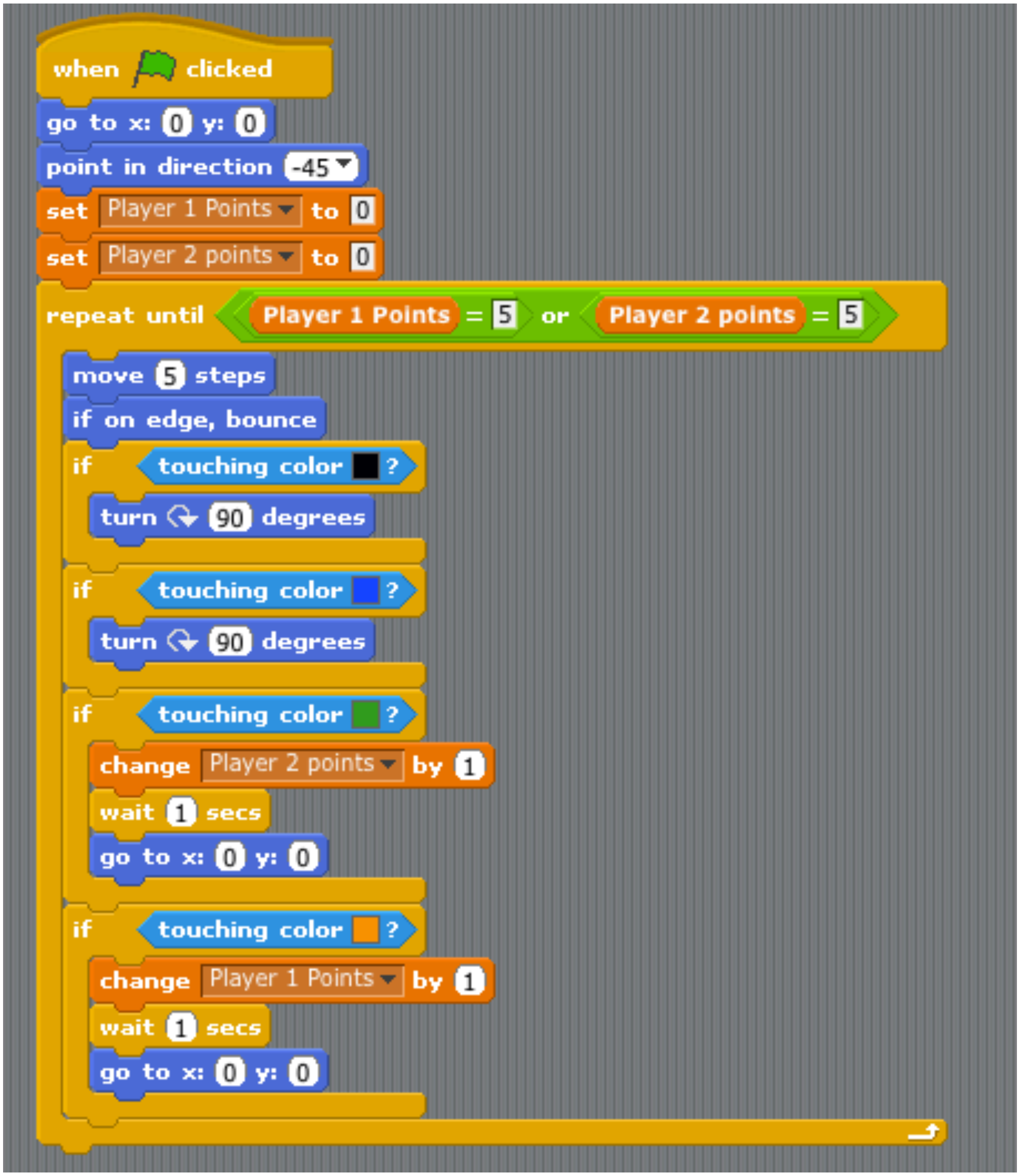
```
repeat until < > or < >
```

19. Put a  = in both of the slots and set it up like this:



```
repeat until Player 1 Points = 5 or Player 2 points = 5
```

20. The ball script should look like this:

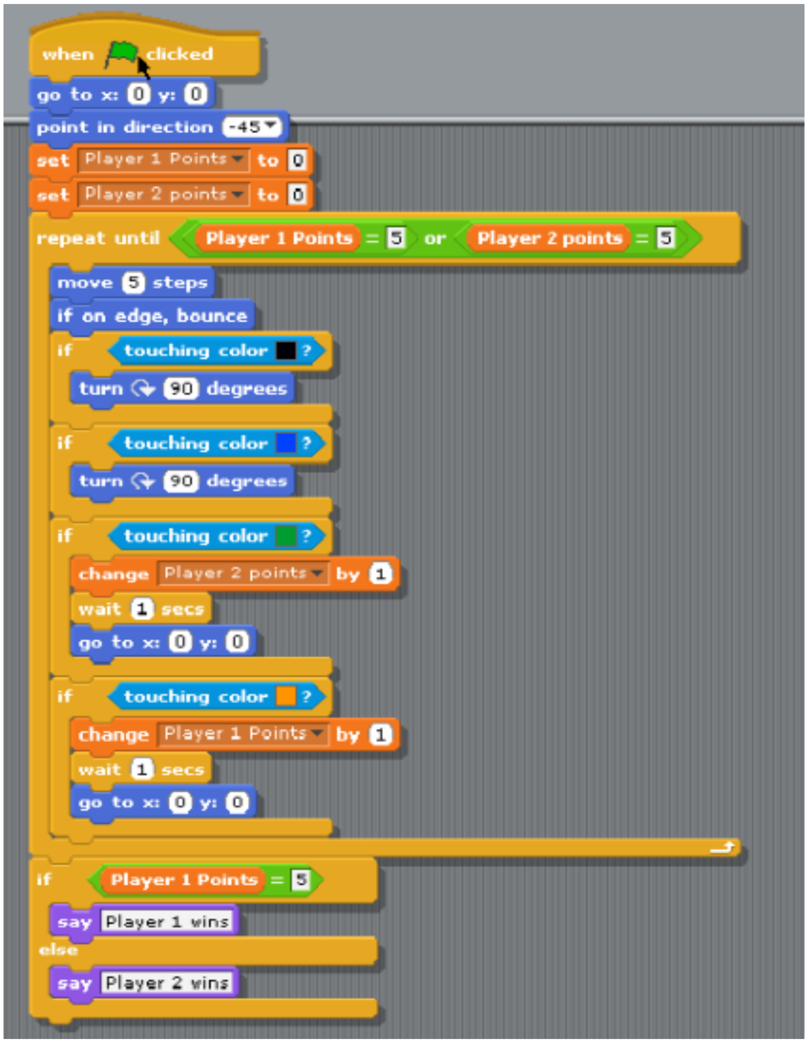


```
when 🏴 clicked
go to x: 0 y: 0
point in direction -45 ▾
set Player 1 Points ▾ to 0
set Player 2 points ▾ to 0
repeat until < < Player 1 Points = 5 > or < Player 2 points = 5 > >
    move 5 steps
    if on edge, bounce
    if < touching color ⬛ ? >
        turn ↻ 90 degrees
    if < touching color 🟦 ? >
        turn ↻ 90 degrees
    if < touching color 🟩 ? >
        change Player 2 points ▾ by 1
        wait 1 secs
        go to x: 0 y: 0
    if < touching color 🟧 ? >
        change Player 1 Points ▾ by 1
        wait 1 secs
        go to x: 0 y: 0
```

21. To end the game and announce which player wins you will add one more addition to your script. Go to Control and choose the if else loop.

22. Go to Operators and choose the □ = □ option and place it next to the “if” in the loop. Put “Player 1=5” within the equals option.

23. Lastly go to Looks and choose the say Hello! option and put “Player 1 Wins!” in the first slot and “Player 2 Wins!” in the second. The final script of the ball should look like this:



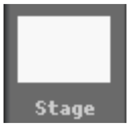
```
when 🟩 clicked
go to x: 0 y: 0
point in direction -45
set Player 1 Points to 0
set Player 2 points to 0
repeat until < Player 1 Points = 5 or Player 2 points = 5 >
    move 5 steps
    if on edge, bounce
    if touching color ■ ?
        turn ↻ 90 degrees
    if touching color 🟦 ?
        turn ↻ 90 degrees
    if touching color 🟩 ?
        change Player 2 points by 1
        wait 1 secs
        go to x: 0 y: 0
    if touching color 🟧 ?
        change Player 1 Points by 1
        wait 1 secs
        go to x: 0 y: 0
if Player 1 Points = 5
    say Player 1 wins
else
    say Player 2 wins
```

YOU ARE NOW READY TO PLAY!!

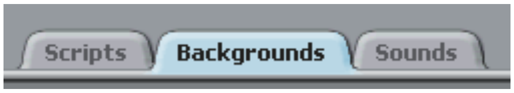
Make a Short Video

The video we are going to make is a short little film with two sprites named Boy4 and Gobo1. They are on the moon and Boy4 is doing the moon walk.

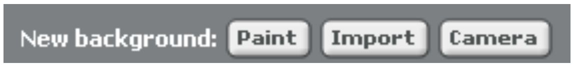
1) The first thing we will do is get a moon for our background. Click on the Stage.



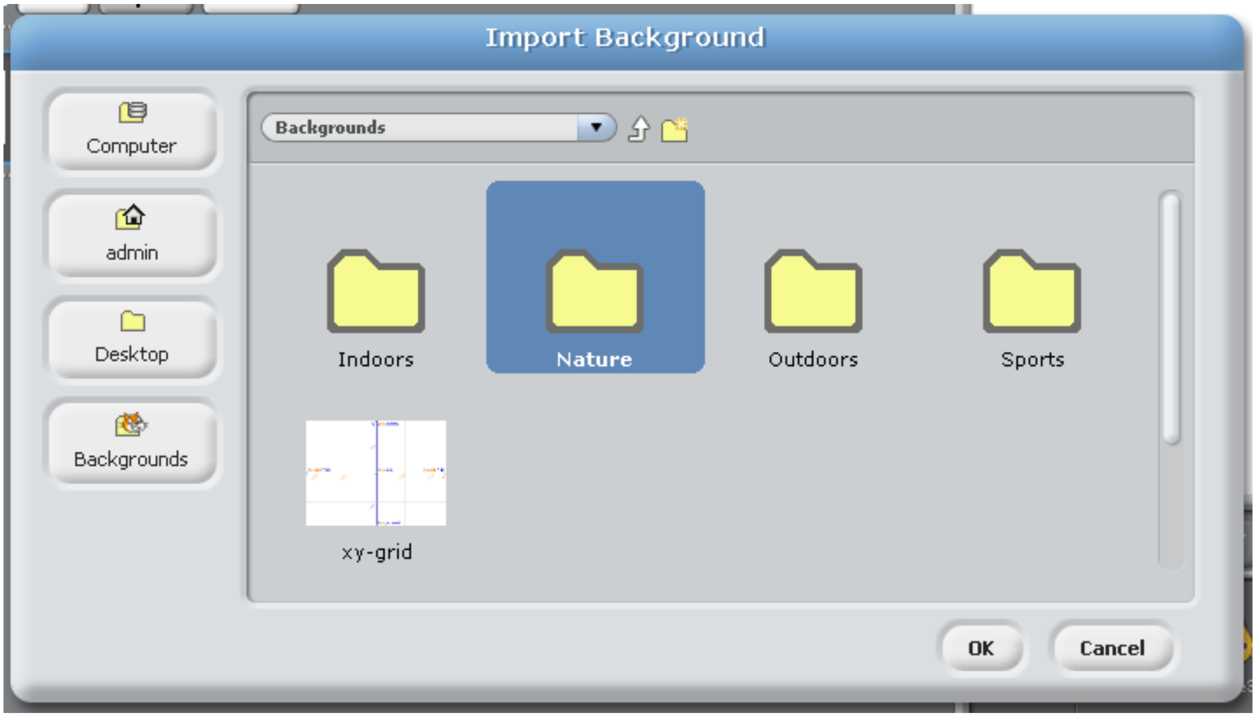
2) Select "backgrounds" (between "scripts" and "sounds")



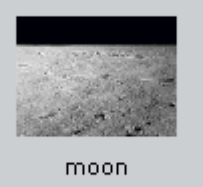
3) Select "import"



4) Double click on the "Nature" folder



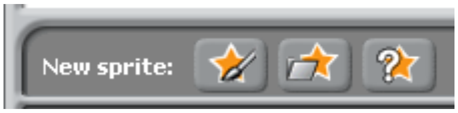
5) Find the background that says "moon" in the nature folder and double click it



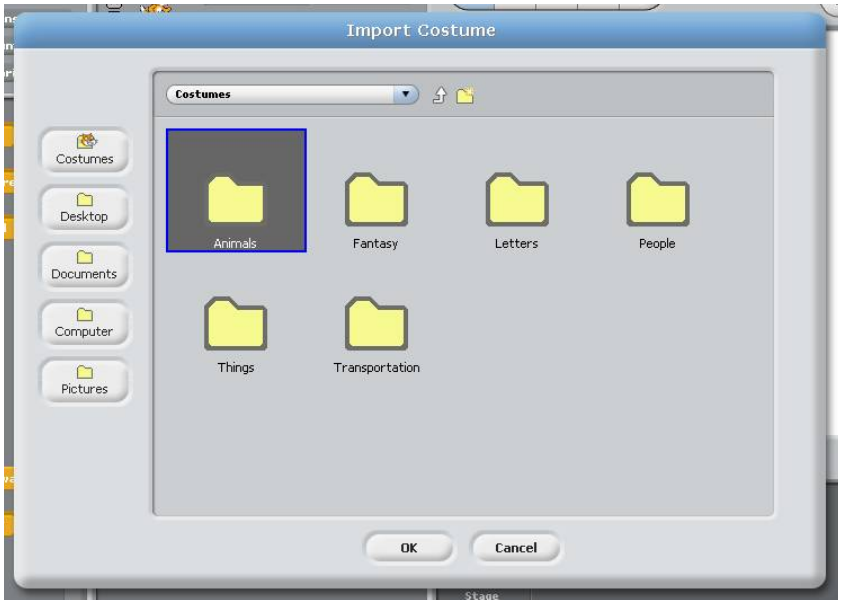
6) Your background is now the moon! (Note: it is possible that the moon picture is in the "Outdoors" folder instead). Time to move on to the next step, double click on the cat.

7) Delete the cat by right clicking on it and selecting "Delete".

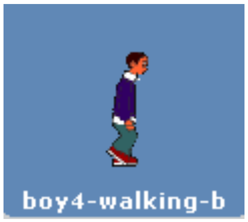
8) Click on the “choose new sprite from file” button. (Middle button in the picture below).



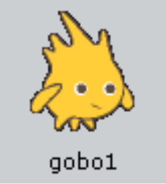
9) Double click on the “People” folder



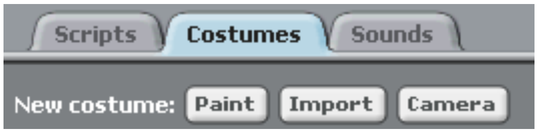
In the fantasy folder there should be a sprite labeled “boy 4-walking-b”. Double click on this to add him as a sprite



10) Next we need to add another sprite. Click on the “add sprite” option and double-click the fantasy folder this time. We are going to find the sprite labeled “gobo1” and add him.

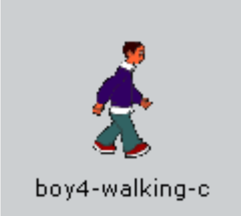


11) Now we need to get a few costumes for each of our sprites. Click on boy4 and then click on the "Customs" option.

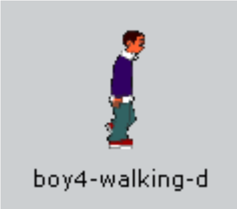


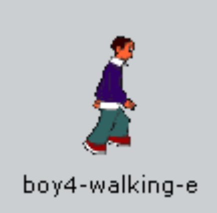
12) Select "Import" and then double click on the fantasy folder.

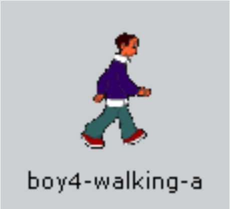
13) has a few different costumes, find and select the one that says -walking-c :



14) We will need to add 3 more costumes for Boy4. For each costume we need to select "import" and double click the people folder as we did before. These are the other three costumes we need:

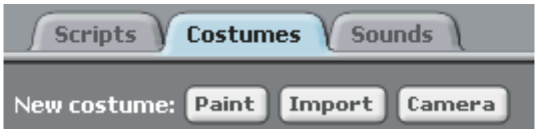






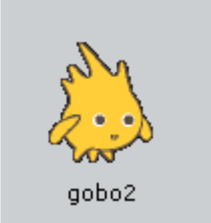
15) Now let's get the costumes for Gobo1. Click on Gobo1.

16) Click the costumes option.

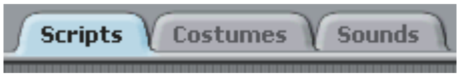


17) Select "Import" and double click the fantasy folder

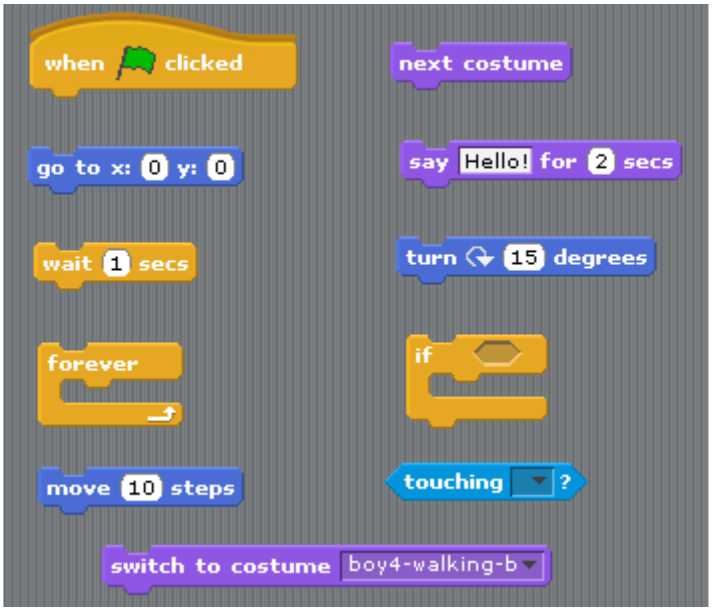
18) Find the costume labeled as Gobo2 and select it:



19) Now that we have the costumes settled we can make the scripts for each of our sprites. Click on and then click on "Scripts"



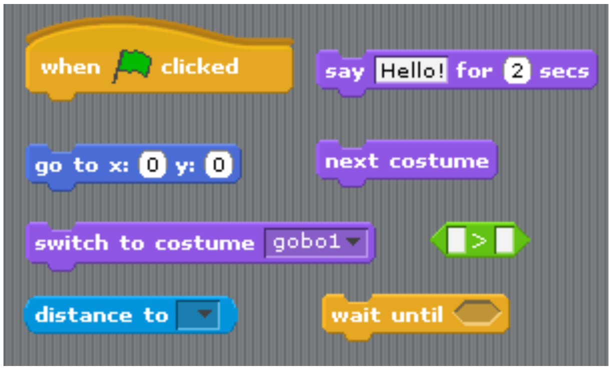
20) For we will need the following code blocks:



```
when [flag] clicked
go to x: 0 y: 0
wait 1 secs
forever
move 10 steps
next costume
say Hello! for 2 secs
turn ↻ 15 degrees
if
touching ?
switch to costume boy4-walking-b
```

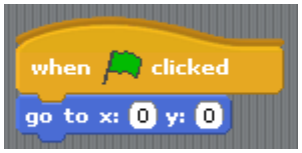
Drag and place these blocks on ’s script

21) Now we need to get Gobo1’s script set up. Click on Gobo1.
22) Click on “Scripts”
23) Drag and place the following blocks onto Gobo1’s script:



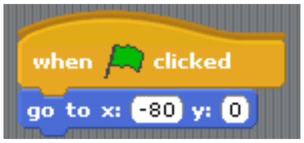
```
when [flag] clicked
go to x: 0 y: 0
switch to costume gobo1
distance to
say Hello! for 2 secs
next costume
[ ] > [ ]
wait until
```

24) Let’s start putting everything together now! Click on again so we can see his script. Take the “Go to x: y:” block (blue) and place it under the block with the green flag:



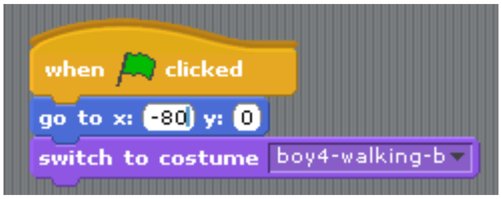
```
when [flag] clicked
go to x: 0 y: 0
```

25) Change the x to -80 in the blue block



```
when [flag] clicked
go to x: -80 y: 0
```

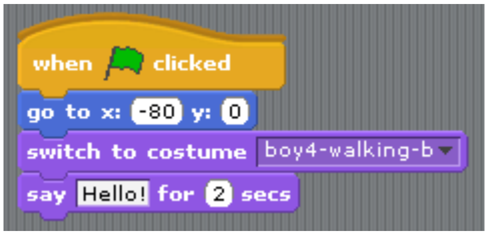
26) Place the “Switch to costume” block under the “go to x: _ y: _” block:



```
when ⚑ clicked
go to x: -80 y: 0
switch to costume boy4-walking-b ▾
```

make sure the costume being switched to is the first costume (in this case titled “Boy4”)

27) Place the “say _ for _ sec” block (purple) under the “switch to costume _” block:

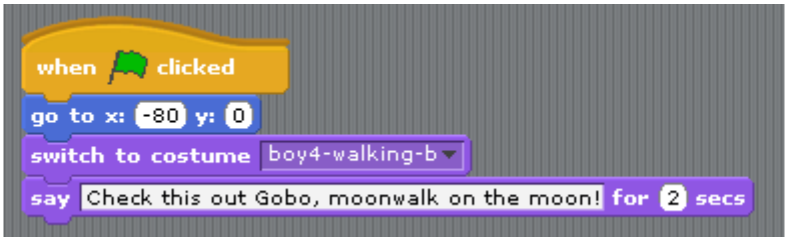


```
when ⚑ clicked
go to x: -80 y: 0
switch to costume boy4-walking-b ▾
say Hello! for 2 secs
```

28) Now Boy4 says “Hello!” whenever the green flag is pressed, except we don’t want him to say this. Click in the box that says “Hello!” and type the following:

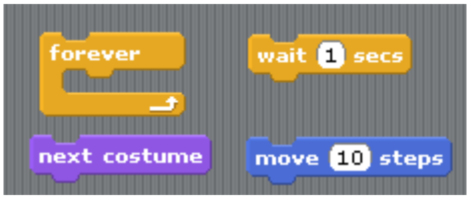
“Check this out Gobo, moon walk on the moon!”

Your script should now look like:



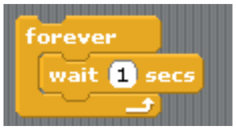
```
when ⚑ clicked
go to x: -80 y: 0
switch to costume boy4-walking-b ▾
say Check this out Gobo, moonwalk on the moon! for 2 secs
```

29) Next we need to make boy4 do the moon walk. For this part we will need to look specifically at these four blocks:



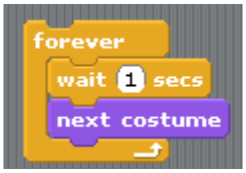
```
forever
wait 1 secs
next costume
move 10 steps
```

Place the “wait _ secs” block into the “forever loop” block:



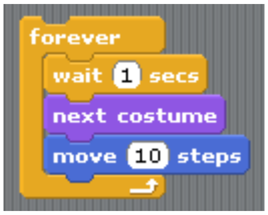
```
forever
  wait 1 secs
```

30) Now place the “next costume” block under the “wait _ secs” block:



```
forever
  wait 1 secs
  next costume
```

31) Finally, place the “move _ steps” block under the “next costume” block:



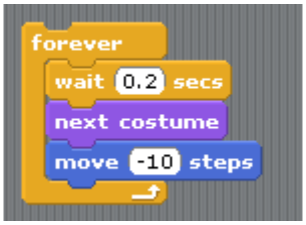
```
forever
    wait 1 secs
    next costume
    move 10 steps
```

Make sure that the blocks are placed in the order shown above, the script reads from the top down so this is important

32) Change the “1” in the “wait _ secs” block to “0.2”

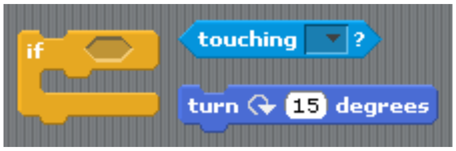
33) Change the “10” in the “move _ steps” block to “-10”

This part of your script should now look like this:



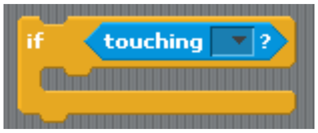
```
forever
    wait 0.2 secs
    next costume
    move -10 steps
```

34) Now we will look specifically at these blocks:



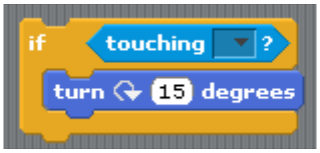
```
if
touching ?
turn 15 degrees
```

35) Place the “touching _” block into the slot near the “if” block:



```
if touching ?
```

36) Place the “turn _ degrees” block into the “if” block:

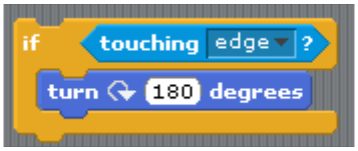


```
if touching ?
    turn 15 degrees
```

37) Click the down arrow in the “touching _” block and select “edge”

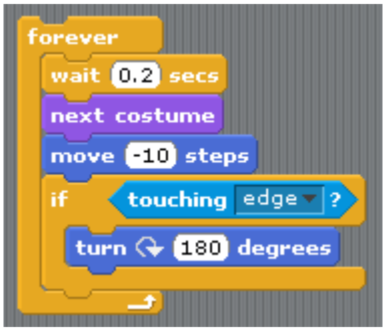
38) Change the “15” in the “turn _ degrees block” to 180

This part of your script should now look like this:



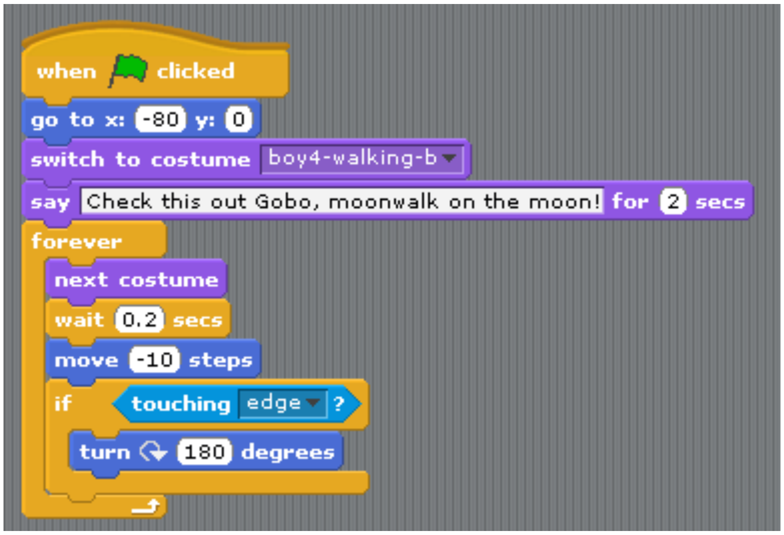
```
if touching edge ?
    turn 180 degrees
```

39) Place the “if” statement you have just created into the “forever loop” block:



```
forever
    wait 0.2 secs
    next costume
    move -10 steps
    if touching edge ?
        turn ↻ 180 degrees
```

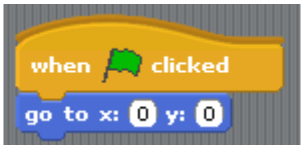
40) Place the “forever loop” under the “say _ for _ secs” block:



```
when 🏴 clicked
go to x: -80 y: 0
switch to costume boy4-walking-b
say Check this out Gobo, moonwalk on the moon! for 2 secs
forever
    next costume
    wait 0.2 secs
    move -10 steps
    if touching edge ?
        turn ↻ 180 degrees
```

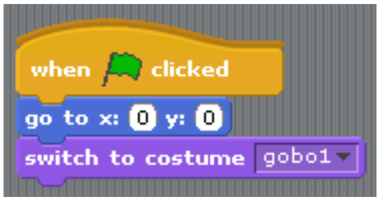
41) Alright! We are done making ’s script, now it’s time to make Gobo1’s script. Double click on Gobo1 and then click on “Scripts”

42) Place the “go to x:_ y:_” block under the block with the green flag:



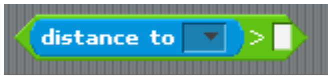
```
when 🏴 clicked
go to x: 0 y: 0
```

43) Place the “switch to costume _” block under the “go to x:_y:_” block



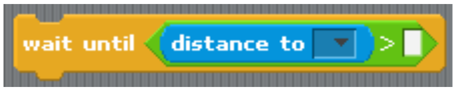
```
when 🏴 clicked
go to x: 0 y: 0
switch to costume gobo1
```

44) Place the “distance to _” block in the left end of the “greater than” block:



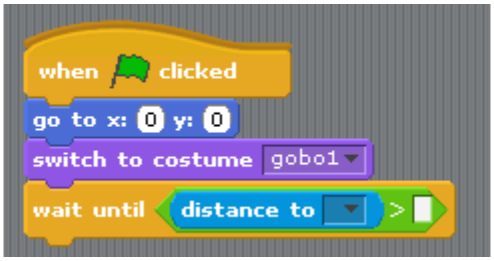
```
distance to ▾ > □
```

45) Place the “greater than” block in the “wait until” block:



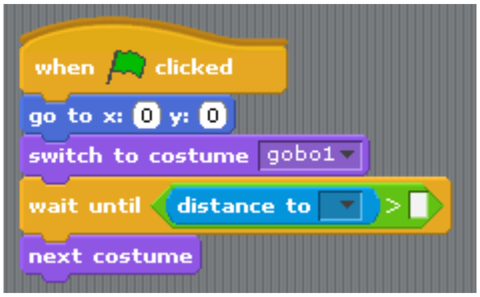
```
wait until distance to ▾ > □
```

46) Place the “Wait until” block under the “switch to costume _” block:



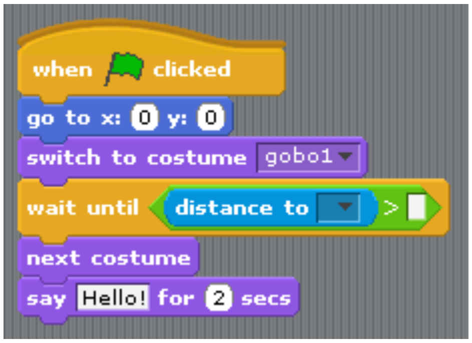
```
when clicked
go to x: 0 y: 0
switch to costume gobo1
wait until distance to > 
```

47) Place the “next costume” block under the “wait until” block



```
when clicked
go to x: 0 y: 0
switch to costume gobo1
wait until distance to > 
next costume
```

48) Place the “say _ for _ secs” block under the “next costume” block



```
when clicked
go to x: 0 y: 0
switch to costume gobo1
wait until distance to > 
next costume
say Hello! for 2 secs
```

49) Now we need to change a few things. Make sure the “x” and “y” are set to “0” in the “go to x:_y:_” block.

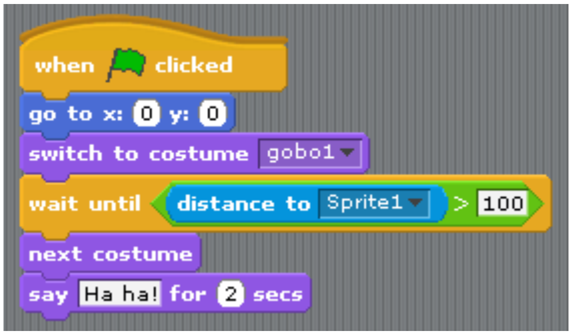
50) Next make sure that in the “switch to costume _” has the first costume selected (in this case that costume is “Gobo1”)

51) In the “distance to _” block click on the down arrow and select “sprite1”

52) In the right end of the operator block (green) type in “100”

53) In the “say _ for _ secs” block type in “Ha ha!” Where it says “Hello!”

After all of that your script should now look like this:



```
when clicked
go to x: 0 y: 0
switch to costume gobo1
wait until distance to Sprite1 > 100
next costume
say Ha ha! for 2 secs
```

54) Click the green flag and see your first mini video come to life on scratch

Linux Guide

The raspberry pi uses a programming language called “Linux”. Below is a guide to changing a few things within Linux to make using the raspberry pi easier.

Raspberry Pi

To Login to the Pi:

Username: pi

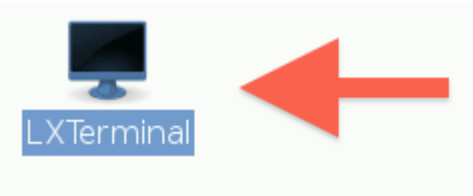
Password: raspberry

pi@raspberrypi: startx

Restarting the Pi:

- In the Linux (LX) terminal type in sudo reboot and press enter

Linux/LX terminal:



Time

Set time zone to mountain time by:

- Entering “sudo dpkg-reconfigure tzdata” (without quotes) into the LX Terminal
- Select US
- Select Mountain
- Restart the pi

Sound:

To get sound:

- Restart the pi (open the LX Terminal and enter sudo reboot, or unplug and plug in the system)
- Type in the username + pass for the pi (Username: pi, Pass: raspberry) Do NOT type in startx
- Type in “sudo nano /boot/config.txt” (without quotes) hit enter
- If there is no place where it says “hdmi_drive=2” then add it as a line to the configuration file

- If there is a line where it says “hdmi_drive=2” make sure it is uncommented, do this by deleting the # sign in front of it
- Press control-x
- Press Y
- Press enter
- Type in Sudo reboot

Sound should work on the pi now

Keyboard

To change the keyboard to a US layout:

- Restart the Pi
- From the command line type “sudo nano /etc/default/keyboard” (without quotes)
- Then find where it says XKBLAYOUT=”gb” and change the gb to the two letter code for the US (two letter code is us)
- Press control-x
- Press Y
- Press enter
- Type in Sudo reboot

The keyboard should now be set as an average US keyboard instead of an average United Kingdom keyboard

Websites to find out more

There are an infinite number of fun things to do with scratch and they can’t all be covered here unfortunately, but we have a couple of sites that can be visited online to find out more about scratch:

http://learnscratch.org/

Learnsrcatch.org is a great site to learn more about scratch. It has tutorials and videos to help guide you along with their lessons, definitely a helpful resource to have.

http://scratch.mit.edu/

This is my favorite scratch site. It has its own built in scratch program that is very similar to the one on the raspberry pi (actually it has a few more features than the one on the raspberry pi). On this site you can view games and videos others have made with scratch and then see how they did it so you can make it for yourself!

Python:

http://www.codecademy.com/

This website is great for learning python, java, and many other programming languages. You may have to set up an account but it is free (and if you have facebook you can connect through that instead of making an account). You earn badges as you complete new lessons on this site and it's all interactive. When I wanted to learn Python this is the site I used and it was a blast.

Linux and your raspberry Pi:

You may have noticed a few irritating things about your Pi, there's a good chance that whatever it is can be fixed by changing something in the Linux programming language. The site below is a great guide for solving these problems:

http://elinux.org/R-Pi_Troubleshooting