Raspberry Pi Activity 2: How to Make Raspberry Pi

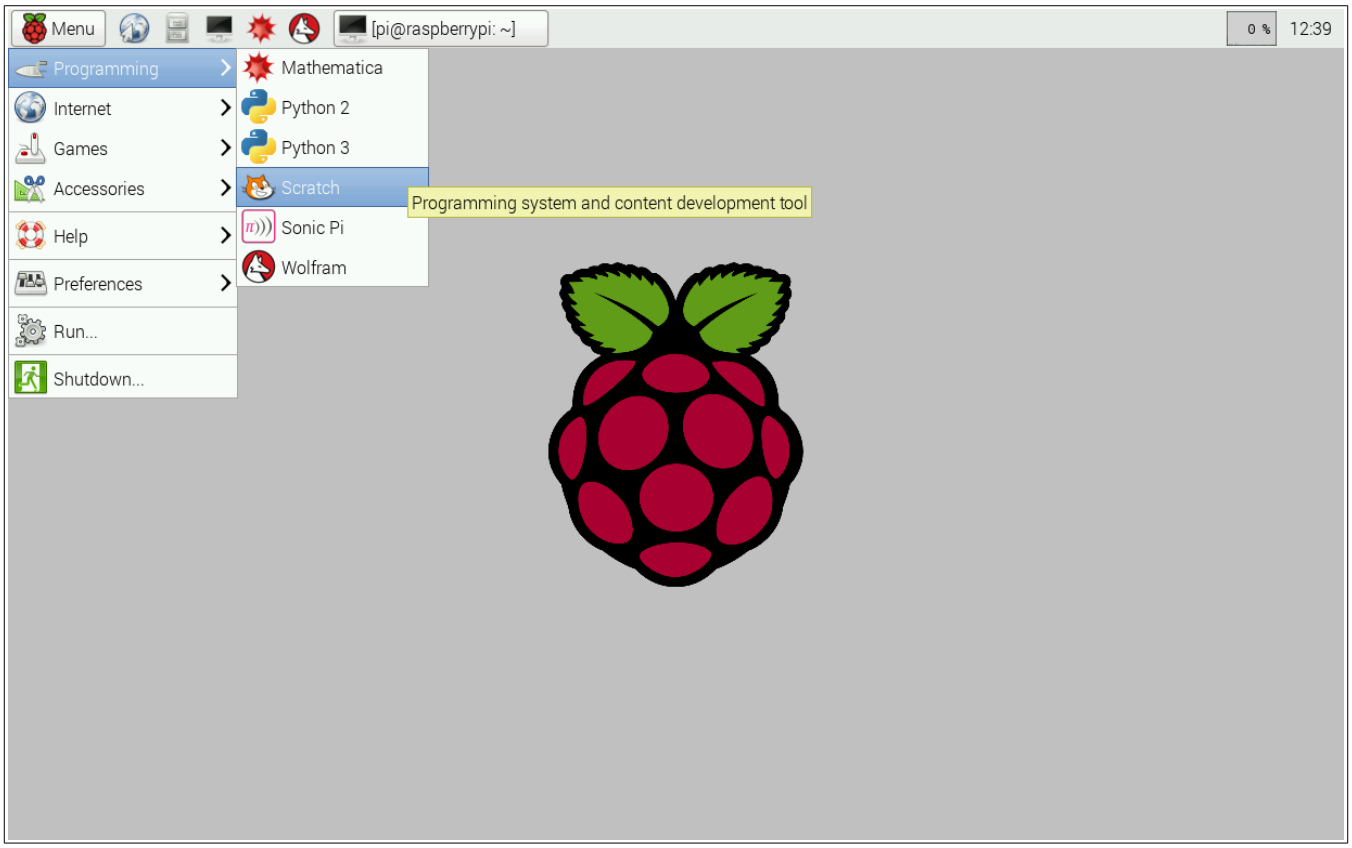
In this activity, you will be programming with the Raspberry Pi using a programming language called Scratch. You will need the following items:

- Raspberry Pi B v2 with power adapter;
- LCD touchscreen with power adapter and HDMI cable;
- Wireless keyboard and mouse with USB dongle;
- Wi-Fi USB dongle; and
- MicroSD card with Raspbian already installed (which was covered in the first Raspberry Pi activity).

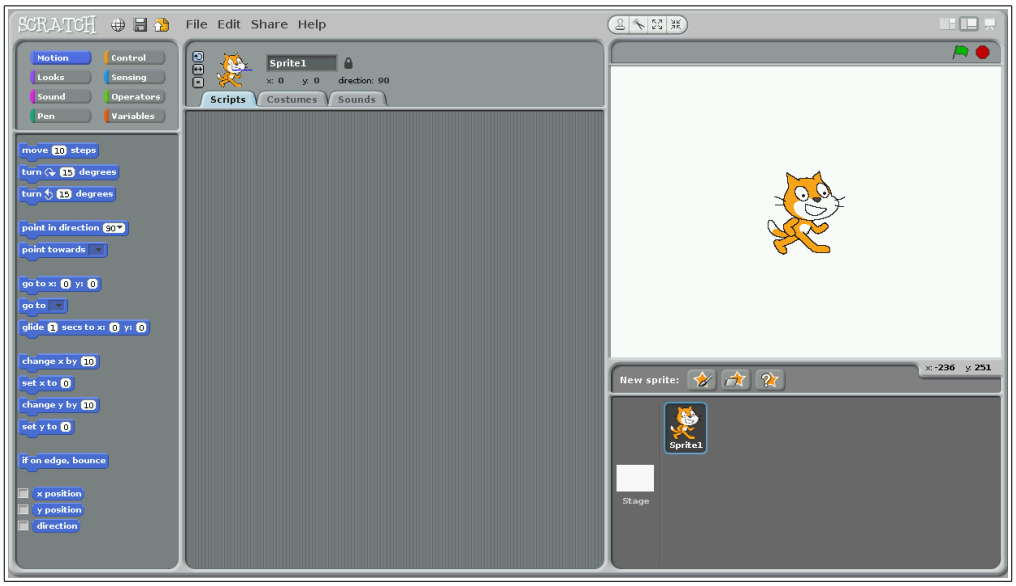
Although you won't actually design complex programs that solve interesting problems (yet), you will explore algorithm design and computer programming in Scratch, a visual programming language that replaces syntax with puzzle pieces. Unlike programming languages that are used in practice (e.g., C++, Java, Python), Scratch is intended for education and provides a great starting point for novice programmers. But don't get boxed in to the idea that Scratch is somehow not powerful In fact, it is actually quite powerful and allows you to create games, animations, and interactive stories.

Scratch

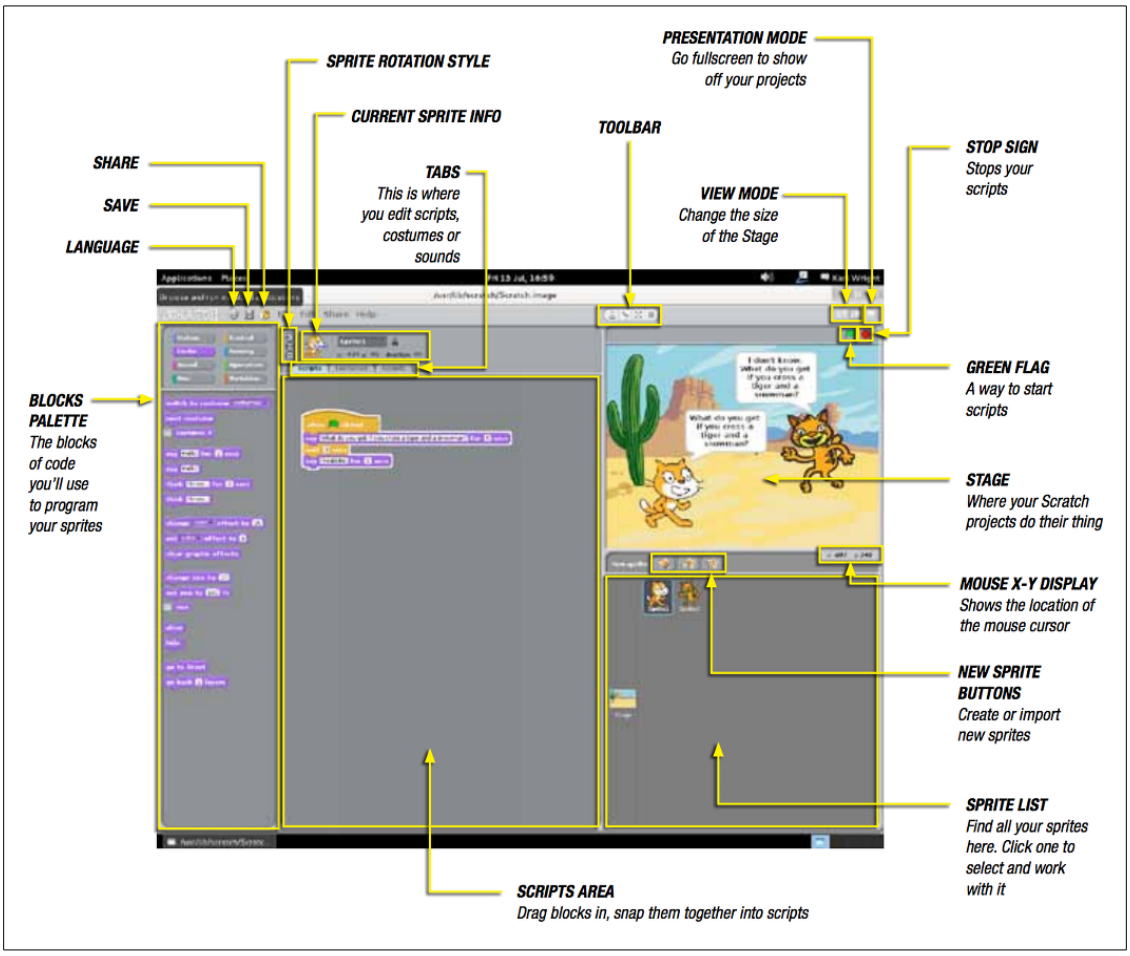
To begin, let's start Scratch from the menu:



After a short while, the Scratch interface will be displayed:



Although the interface looks a bit complicated, it really isn't. We won't make use of everything at first; however, here is an overview of the interface components:



The Scratch interface is quite busy; however, there are three main areas that you will find yourself interacting with often.

The **blocks palette** panel provides a variety of useful constructs that allow you to write programs. In total, there are eight different block types, accessible by clicking the groups in the top-left of the interface:

- *Motion*: anything related to the movement or placement of sprites (graphics);
- *Looks*: anything related to the appearance of sprites;
- *Sound*: anything related to incorporating sound in your programs;
- *Pen*: anything related to the pen which allows drawing on the canvas (background);
- *Control*: anything related to controlling the flow of programs;
- *Sensing*: anything related to detecting things (like movement, collisions, etc);
- *Operators*: anything related to math functions and string handling; and
- *Variables*: anything related to the declaration and upkeep of variables.

The **scripts area** is where you define your computer programs. This is done by dragging various blocks from the blocks palette and connecting them to make a program. It's almost like solving a jigsaw puzzle. In fact, blocks have different types of notches and ridges that allow them to match up only to certain other blocks. This helps simplify the design of programs.

The **stage** is where your programs are executed. It's where to look to see if your code works...or not. On the stage, we can place sprites (graphics), variables, text, and drawings. At the top-right of the stage, a green flag and a red stop sign are used to start and stop your programs. The stage implements a two-dimensional coordinate system, where *x* and *y* represent the horizontal and vertical axes respectively. On our system with the LCD touchscreen, the center of the stage is at the point (0,0); the top-left corner is at (240,180); the bottom-right corner is at (240,-180). Note that the stage is actually much larger (i.e., the cat sprite could technically be moved out of the viewable area of the stage).

The first program

Let's create a simple program. Your task will be to move the cat sprite on the canvas. One useful block in the control blocks group is the **when green flag clicked** block. It is used to specify what to do when the green flag at the top of the stage is clicked (in other words, what to do when your program starts). We can add it to the scripts area by dragging it from the control blocks group in the blocks palette. Let's also add the first instruction to **move 10 steps** (pixels) in the direction the cat is facing (i.e., to the right). For this, we can utilize a block in the motion blocks group. Drag the move steps block to the scripts area until it snaps in place beneath the green flag block:

Click on the green flag in the top-right corner of the stage to run this program. You will notice that the cat moves very quickly a very small distance to the right. Click the green flag several times so that the cat repeatedly moves to the right a bit more.

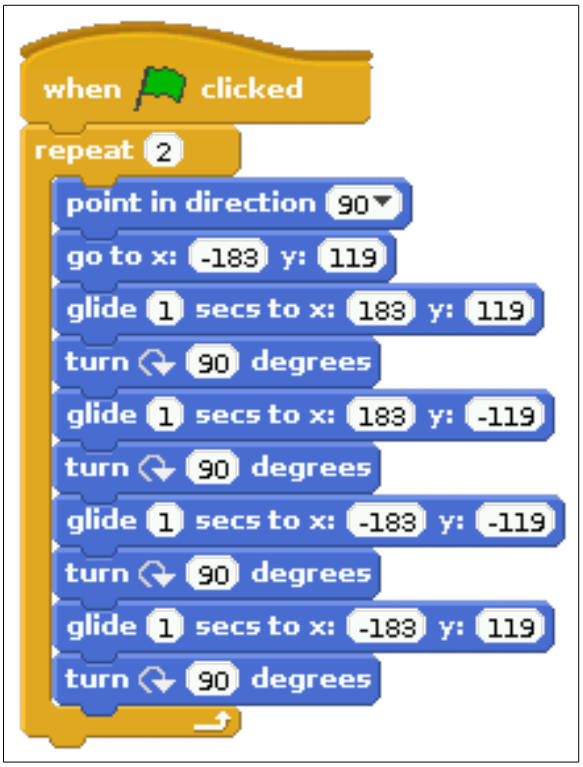
Modify the program to move a different number of steps by changing the value in the move 10 steps block to something like -100 (which will move the sprite 100 pixels in the opposite direction that the cat is facing). Click the field that specifies the number of steps and replace 10 with -100:

Clicking the green flag a few times moves the cat to the left quickly. This movement is too quick and quite joggy. In the motion blocks group, a **glide 5 secs to x,y** block allows motion to be more specifically defined. Let's replace the move steps block with this new block. To remove the move steps block, drag it away from the green flag block (you can just put it to the side if you anticipate using it again, or drag it back to the blocks palette to *trash* it). Tweak the values as you wish in the new motion block and watch the cat move smoothly to the specified coordinates:

You can actually click on the cat sprite to move it anywhere on the stage; then try running your program again.

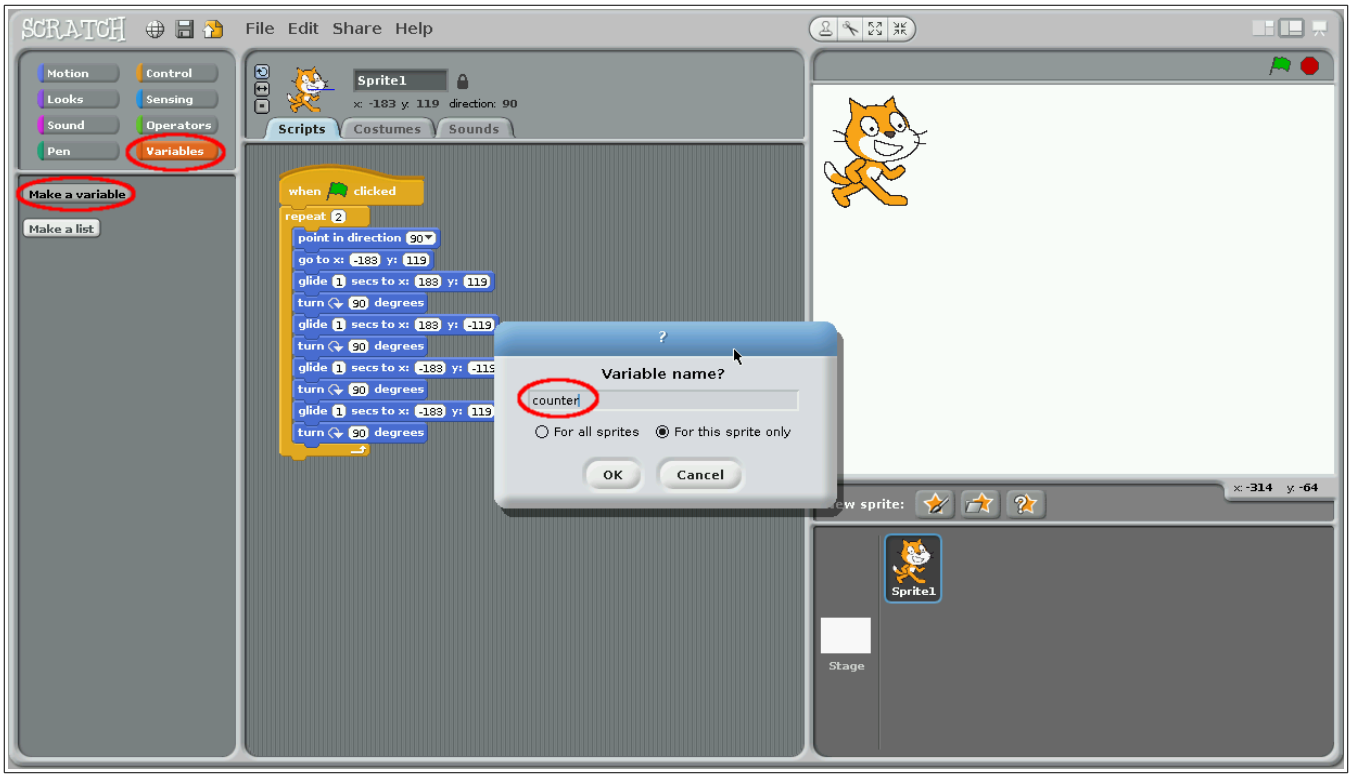
Improvement (or maybe just more fun?)
Let's combine blocks to form a more complicated program. Create the following program and run it:

You'll notice that the program moves the cat around the perimeter of the stage, pointing in the direction of travel as it does so. How could the program be modified to repeat this some number of times (like 2)? There is a **repeat** block in the control blocks group that can be used to repeat an action some number of times. Modify your program as follows:

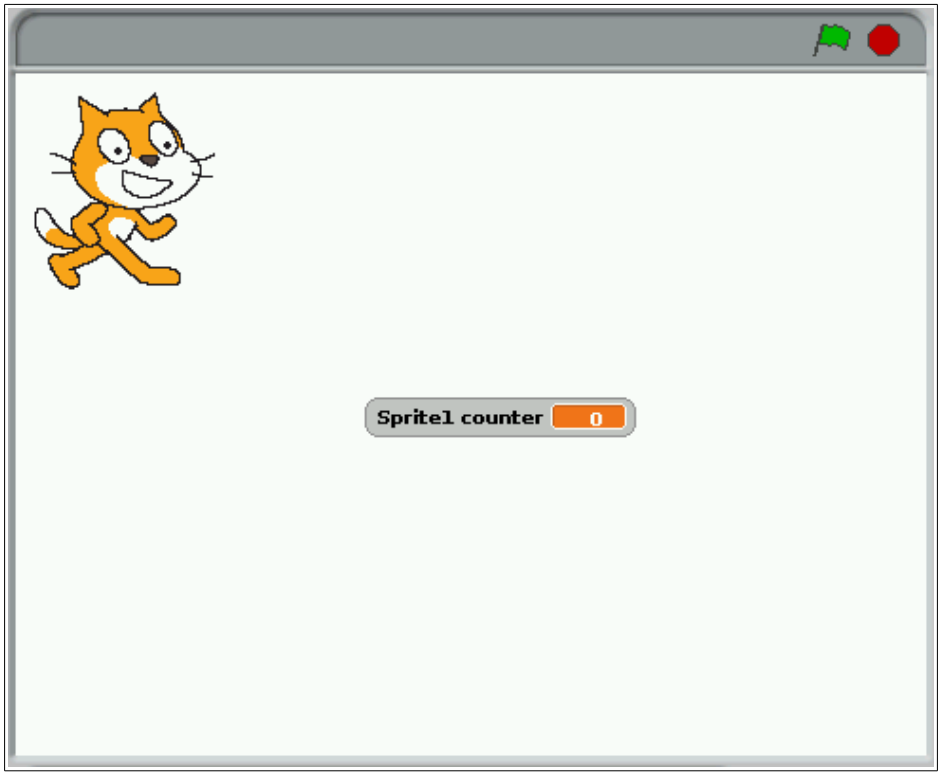


Some of you will notice that the first two motion blocks (**point in direction 90** and **go to x,y**) can be moved out of and above the repeat block. Try it. You probably won't notice much difference, but these two blocks serve to initially orient and position the cat sprite. This really only needs to be done once at the beginning of the program.

It would be neat to count the number of 90 degree turns that the cat makes during its journey. To do this, let's define a variable (called `counter`) that will be updated each time the cat turns. Defining variables can be done by selecting **make a variable** in the variables blocks group.



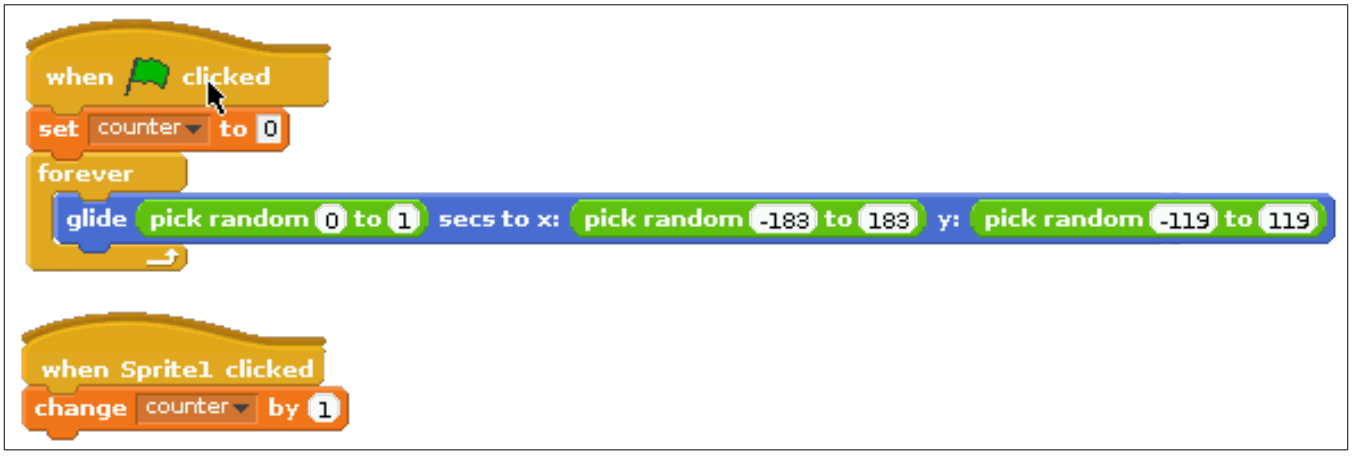
This adds the variable on the stage. Drag it to the center of the stage so that it doesn't get in the cat's way as it moves around:



To use the counter in your program, it will first need to be initialized (with the value 0) and then incremented each time the cat makes a 90 degree turn. We can modify our program as follows:

A first game?

Modify your program (you can save the current version first if you wish) so that it looks like this:



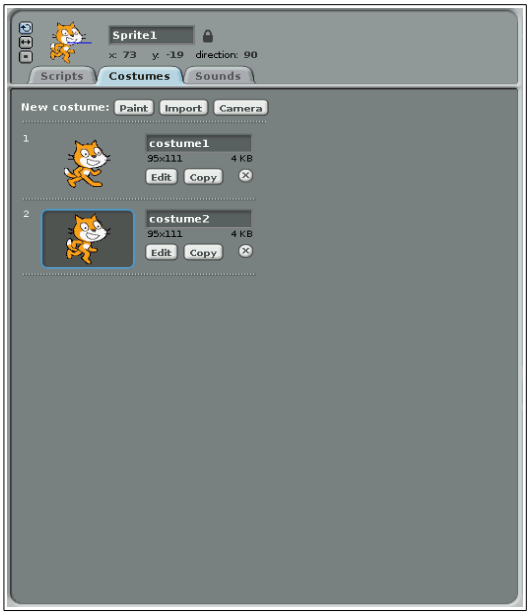
Notice that we now have two block groups in the scripts area. One that is executed when the green flag is clicked; and another that is executed when the cat sprite (Sprite1) is clicked. What does the program do?

Did you know?

You can change the name of the cat sprite at the top of the scripts area. Make sure that the cat is selected in the **sprite list** below the stage.

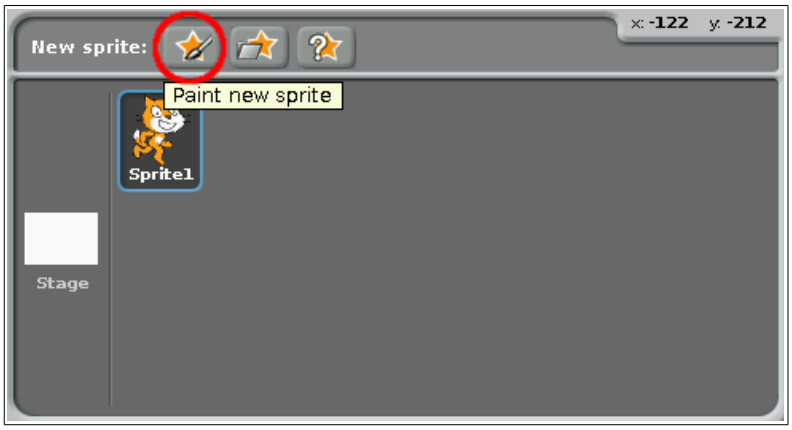
Playing with sprites

At the top of the scripts area, there are several other tabs that provide sprite costume and sound tools. Click on the **costumes** tab. You will notice that the cat actually has two sprites, one named costume1 and another named costume2:



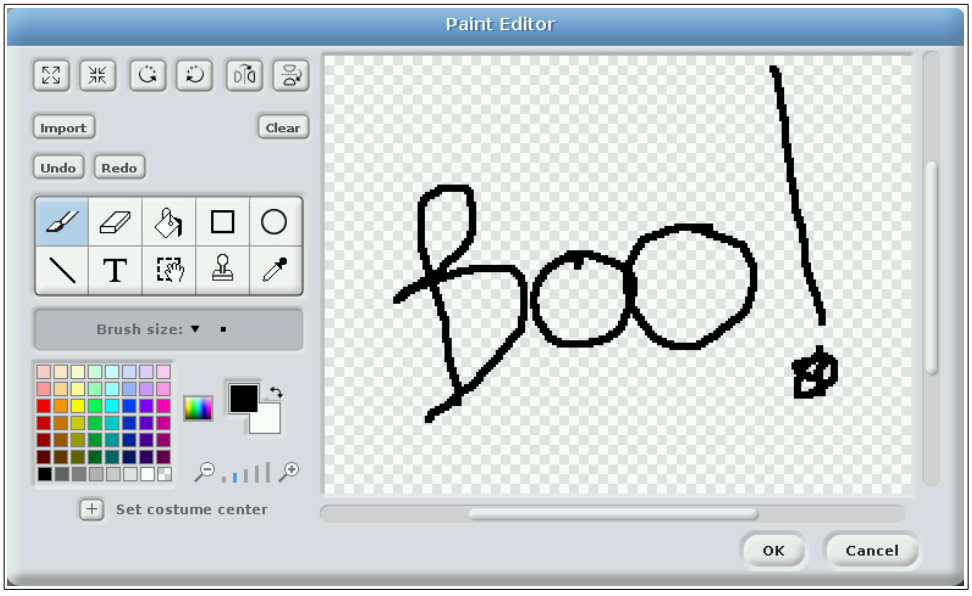
By alternating them, we can make it look as if the cat is walking or running. Modify your program as follows:

You can also create your own sprites via the paint new sprite button in the sprite list:

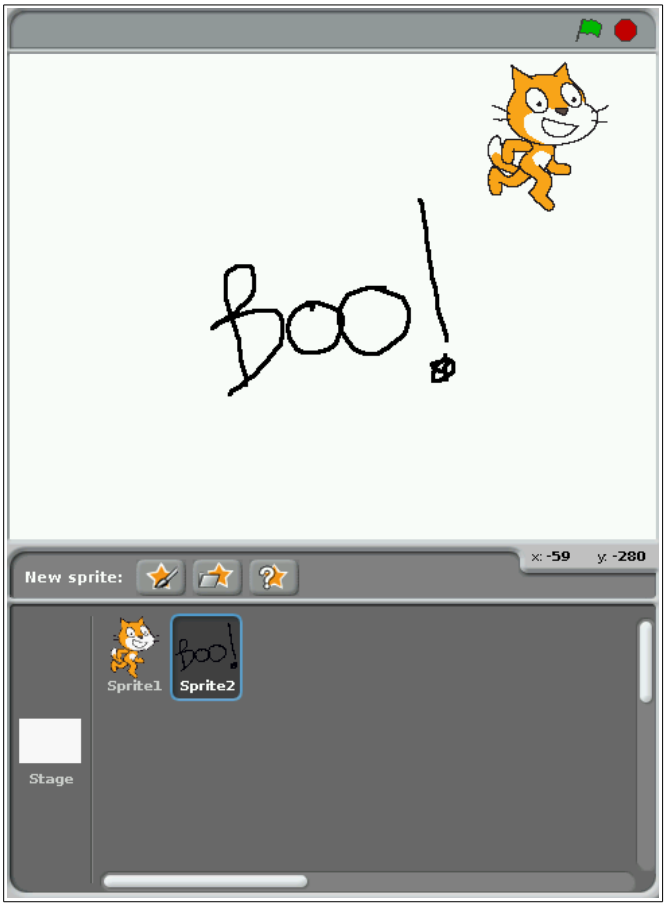


This displays a **paint editor** that can be used to create a new sprite of your design. The other buttons to the right allow saved sprites to be loaded (Scratch comes with many different sprites) and a random sprite to be added to the sprite list.

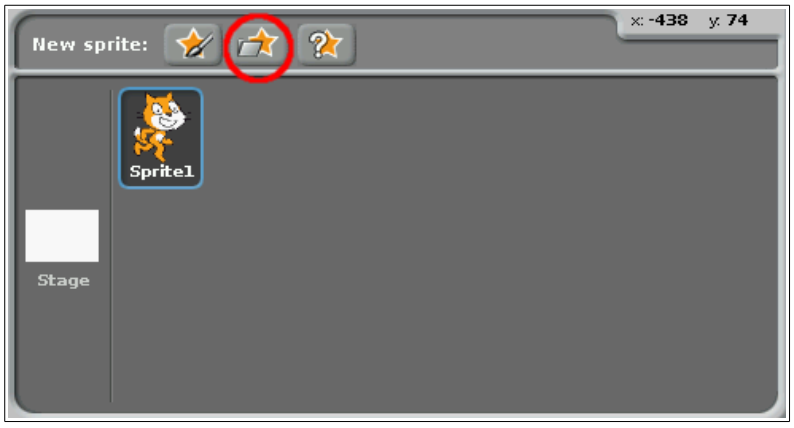
Try creating one now:



Click OK when done. This will drop your new sprite in the sprite list and on the stage:

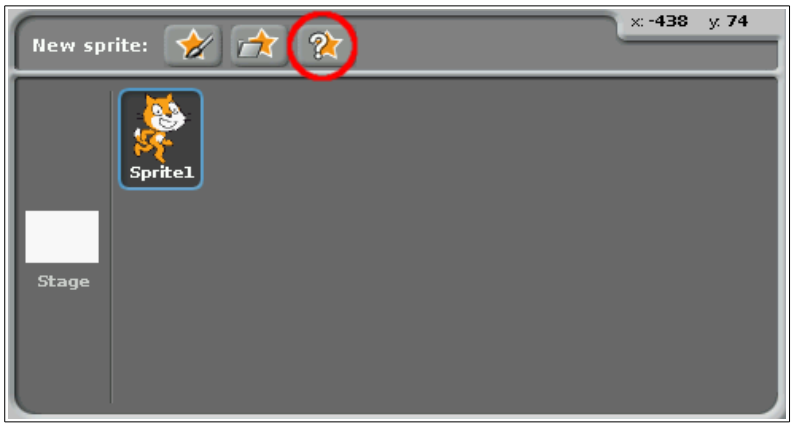


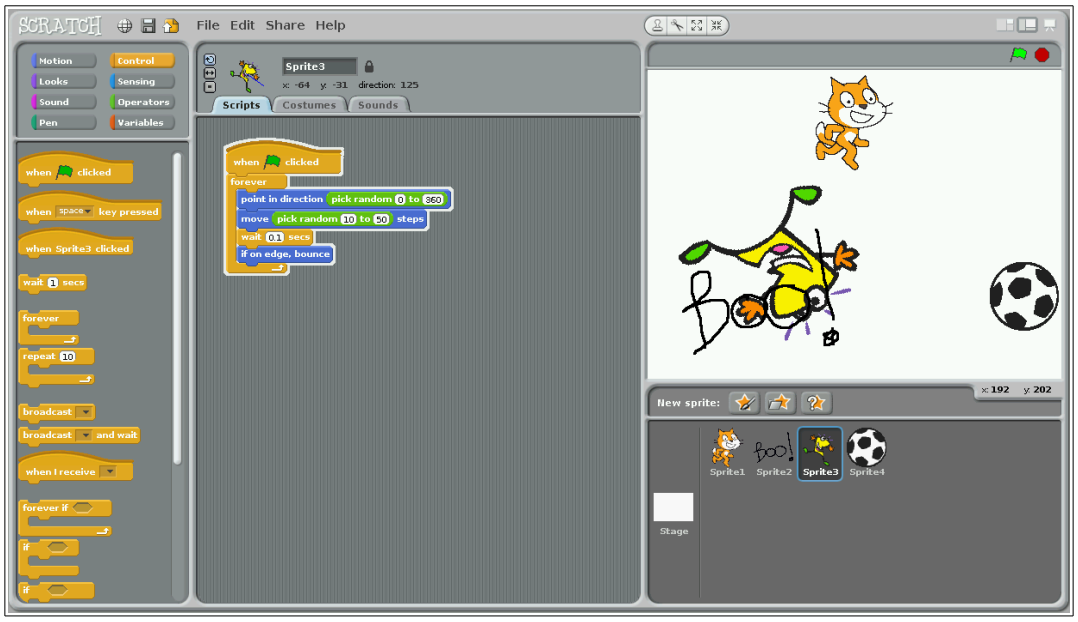
As mentioned earlier, you can also load a saved sprite if you click on the middle button in the row at the top of the sprite list:



Feel free to select any sprite from any category that you wish:

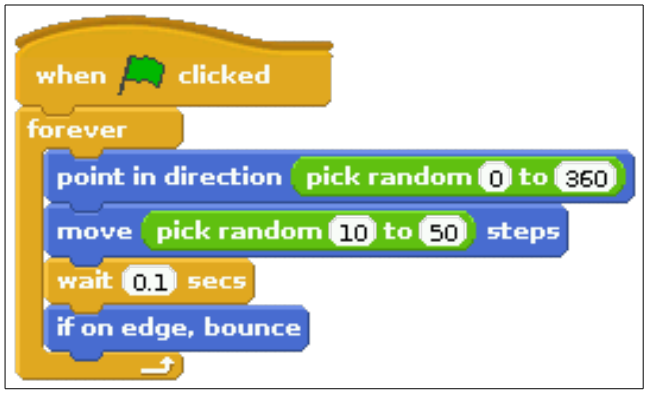
Again, you can have a random sprite brought to the stage too by clicking the right-most button:





You can click on any sprite in the sprite list, and its scripts load in the scripts area. This allows you to have a separate program for each sprite in the sprite list. Think about what this means. You can separately control each sprite while they all run their programs simultaneously!

Try clicking on one of the sprites in the sprites area and create the following program for it:



What does this program do? What does the **if on edge bounce** block do?

Experiment and explore
Feel free to experiment in Scratch. Add new sprites, remove old ones, and create scripts for the sprites. Play around and try different blocks to see what they do.

A final program?
If you are feeling less creative and want someone else to tell you what to do, why don't you try creating a program that:

(1) Randomly moves two sprites around the stage;
(2) Any time that a sprite hits the edge of the stage, it should bounce away from the edge; and
(3) Any time a sprite collides with another sprite, it should say, "Watch out!"

Things to try and figure out
Can you figure out how to:

- Resize (grow or shrink) a sprite (hint: there's an easy way and a hard way)?
- Modify a sprite's costume (hint: it can be done via the scripts area)?