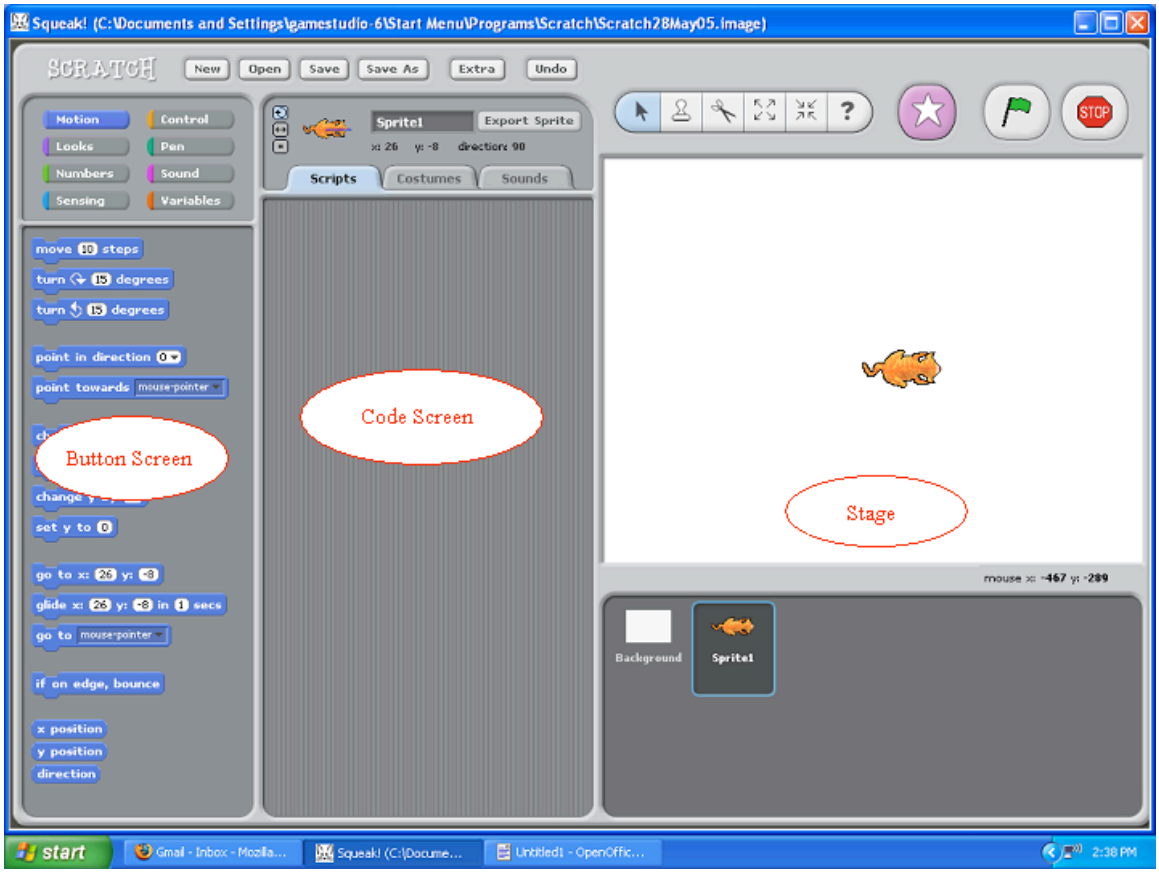
Scratch Tutorial

When you use Scratch, you will see a screen that looks like this:



There are three basic parts of the screen: the Button Screen, the Code Screen, and the Stage.

The Button Screen: This is where all of the buttons, or “blocks” that you can use to piece together on the code screen. At the top of the screen, there are eight different categories of buttons to select from. Clicking on any of these will show all the buttons in that category on the Button Screen.

The Code Screen: This is the area where you piece blocks together to “write” code. The code is basically an instruction recipe, which tells your Sprites what to do.

The Stage: This is where you can see your game, or progress on your game played out. It shows your background, as well as all your Sprites. Sprites are any characters or objects which you want to be able to program.

Sprites

What are these "sprite" things I keep talking about? They can be a lot of things. They can be characters in your game. They can be objects that your characters interact with. Basically, if you want something to be animated, move, or interact with anything else, you need to make a sprite instead of just putting it on your background.

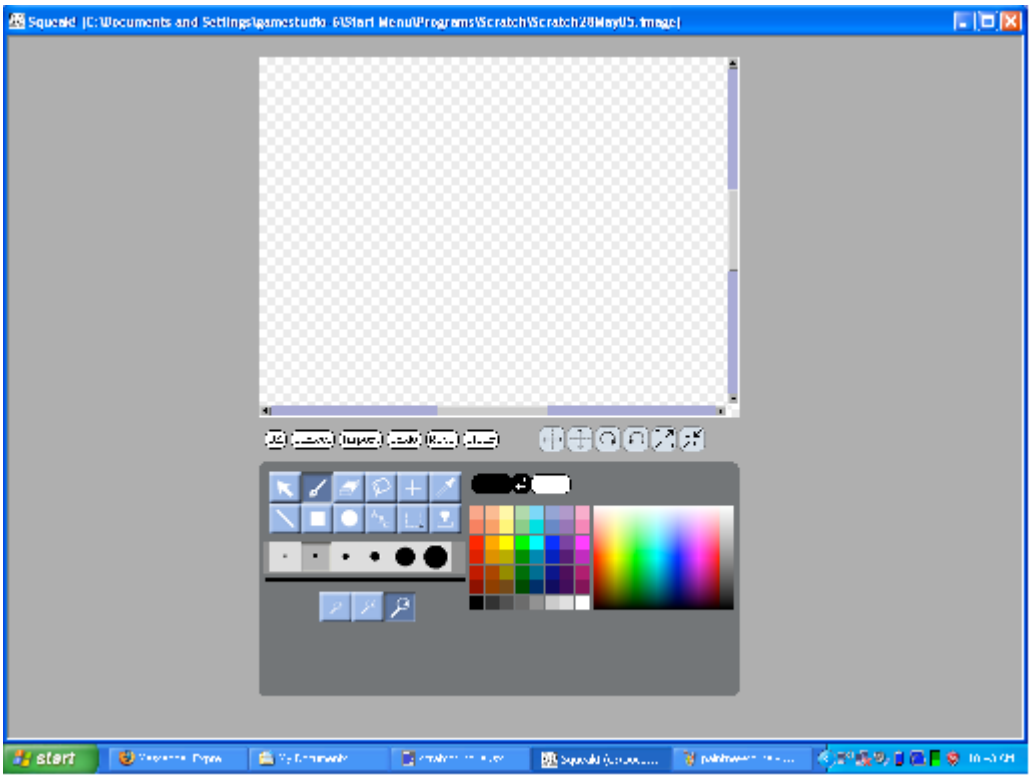
To make a new sprite:

Click on the purple button with the star on the top of the Stage part of the screen.

This will bring up a new window. To draw your Sprite:

Click on the button that says "Paint New Sprite"

This will open up a screen which allows you to draw your sprite however you want it to look:



By default, the screen is set at the largest zoom. You can zoom out or in using the three magnifying glass buttons:

Pointer Tool: Allows you to click on buttons and objects

Paint Brush: Allows you to paint with the selected color

Eraser: Erases previously drawn lines or objects

Paint Bucket: Paints an entire area with the selected color

Guide Tool: Use as a guide to see if things are properly lined up

Eyedropper: Click on a color to copy it

Line Tool: Draws a straight line

Box Tool: Draws a perfect box

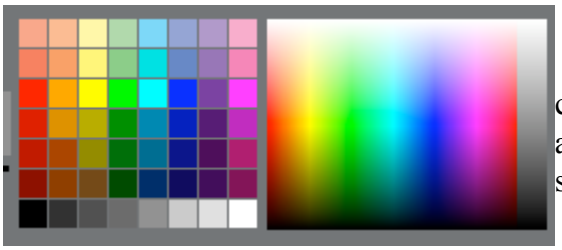
Oval Tool: Draws a perfect oval

Text Tool: Use to type text

Select Tool: Use to select an area to move or copy

Copy Tool: Copies a selected area

This tool allows you to change the width of your brush.



The color palette allows you to pick the color you are using. The one on the left side are common colors, on the right is the whole spectrum available.

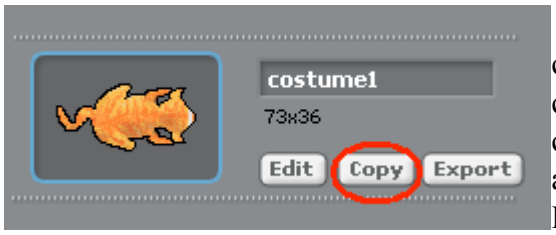
When you are done, click the OK button to return to the main screen

Animation

Now that you have created a sprite, you are ready to animate it. You will be creating a sort of "virtual flip book." You'll make similar drawings of the same character, and then then program the computer to "flip" them for you.

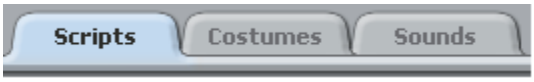
The first thing you'll need to do is create more costumes for your sprite. To do this, you'll need to select the costumes tab in the code part of your window:

You want to create an animation, which means creating costumes that appear similar with small changes. To do this easily, you should copy your first costume.



When you click on the copy button, a second costume will appear under the first. Now you can click on the Edit button for your second costume, and the same screen will appear that allowed you to draw your first costume. However, it will already have your first costume drawn in it, so all you have to do is make your changes.

Now that you have at least 2 different costumes (you can have as many as you want) you can animate them. To do this you need to write a script for this Sprite, so click on the Script tab.



In this area, you can drag all of the commands that you want over and click them together into a logical statement. For an animation, you will need an initiating action (from the control selection) costume changes (from the looks selection) and waits (from the control selection.)

Here, our initiating action is "When Green Flag clicked." Notice it is smooth on the top because it is for starting code. We also have "set costume to _____" which allows us to change from costume to costume. Finally, we put waits in between each change so that when the script runs we will actually be able to see each costume. You can change the length of the wait by clicking on the 1 and entering any number you would like.

For our initiating action, we could have also used:

The “When space key is pressed” means that the code will run when the user pressed the space bar. We could also change this to any other key on the board by clicking on the arrow next to the word “space.” The next one, “when sprite1 is clicked” means the code will run when the user clicks on the sprite with the mouse. Finally, the last one uses variables, which we will cover later.

What about if we want the animation to repeat? Then we can put a repeat loop around our code, like this:

Here, we still need to have an initiating action to tell our code to run. Then we need to make sure that everything we want to repeat is inside of our repeat loops. We can change the number of times it repeats by clicking on the 10 and entering a new number.

We could also make our loop repeat forever by putting out code inside of a forever loops:

Motion in Scratch

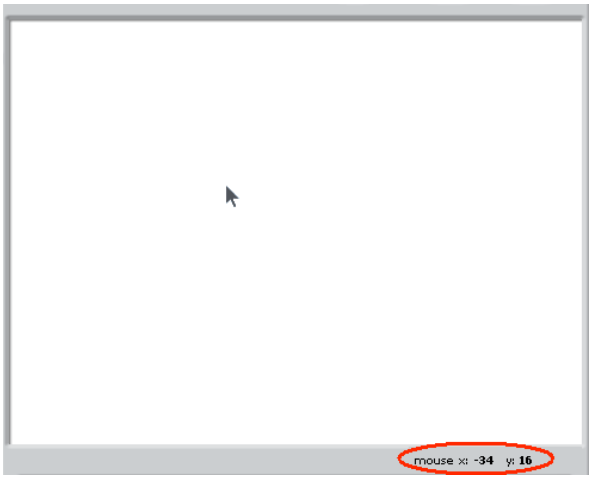
Scratch also has build in commands for moving sprites. All of these are located under the Motion section of the code screen. There are two basic ways to move in Scratch: using steps and degrees, or using a Cartesian Coordinate System (x,y).

Steps and degrees can be used by pointing the sprite in the direction (in degrees) that you want it to travel, and then telling it to move the number of steps you want it to move.

As usual, we need an initiating action. Then we simply point our sprite in the direction we want, and move it the number of steps we want. We could also add on move direction changes, movements, waits, and also repeat or forever loops if we wanted.

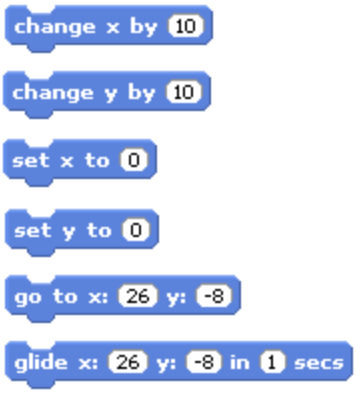
You can also use these to change the direction:

The other way to move your Sprite in Scratch is to utilize the (x,y) coordinate systems. On your stage, if you hold your mouse over any area of the stage, it will tell you the (x,y) coordinates of that position.



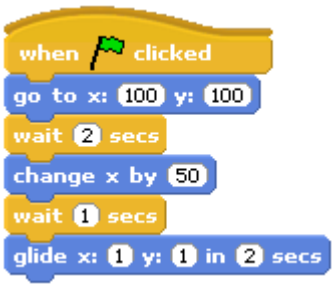
The coordinates work like the coordinates that you had to do in high school, with the center (0,0) being at the center of the stage.

We have a lot of different options using the (x,y) coordinates:



We can chose to simply change the x or y coordinates by a set amount, to set the x or y value to a specific number, to go to a specific coordinate, or to glide to a coordinate. As usual, we can combine these blocks in an infinite number of ways. Waits and repeat or forever loops can also be added to spice things up. However, as usual, you have to start out any script with an initiating action.

How about a sample?

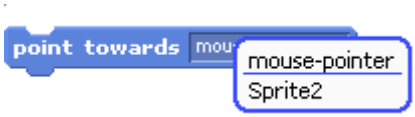


When this code id initiated, the Sprite will go to (100,100), wait 2 seconds, change its X value by 50, wait 1 second, and then glide to (1,1)

Another great block that can be used with either method is the 

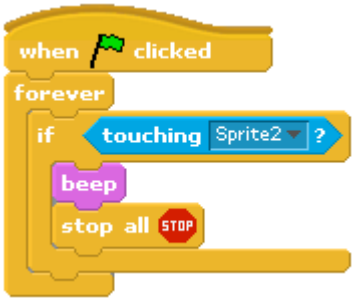
block. This is great for creating continuous motion. However, if you want it to always bounce, make sure to put it inside a forever loop!

We can also set our Sprite to point in a particular direction, such as towards wherever the mouse is or towards another sprite. If the mouse, or the other Sprite moves, then our Sprite will change its direction to compensate – if we remember our forever loop, that is.



Sensing in Scratch

The commands under the section Sensing allow us to have our Sprites interact. Lets say that I am making a game where the goal is for two Sprites to never touch. Well then I need to write a script telling the program this. In order to do this, I am going to need to use an If loop.



Here is a script for Sprite1. Anything contained inside of the if loop will happen whenever our "if" occurs. The if statement can be found in the Sensing section, as can other choices for sensing. The purple "beep" can be found under the sound section along with other sounds, and will simply play a beep sound. The "stop all" stops everything in the game. Notice how the If loop is inside of a forever loop, so that it will always happen after Go is clicked.

We can also have our If statement be another element from sensing, "touching color," which will execute anything in our If loop when the Sprite is touching that color.

Notice that in If loops, only blocks that are shaped like long diamonds will fit into the space provided in the If block. This is because these are the only blocks which will work and make sense in this spot.

Variables and Numbers

Now for some complex stuff. In order to have some more complicated interactions, we will need to use variables. Variables allow us to use a placeholder of some sort, such as "score" to keep track of a value that varies and relate it to another value that we wish to vary with our first value.

Let's say that we wanted to make a game where a cat was chasing a mouse that was moving randomly. First of all, how do we make the mouse move randomly? We will need some blocks from the Numbers section.

This script utilizes the "pick random" block from th Numbers section. Every time the loop is repeated a random number is picked for how far to move and turn the sprite. It also tells the sprite to bounce if it is on the edge.

Now in order to make our cat follow our mouse, we need to create variables to track the

location of our mouse. To do this, go to the variables section of the Button Screen. Then select the option New Variable (for all).

A screen will appear to allow you to name your variable. You should name it something representative of how you will be using it so that you can remember it later if you have a lot of variables. Let's name this variable MouseX because it will represent the X value of our mouse Sprite.

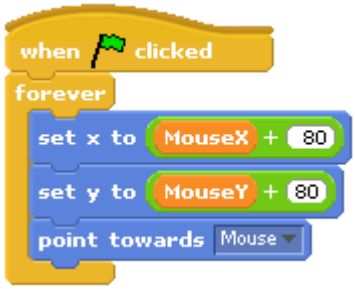
When you click OK, some new blocks will appear in your button screen. We will be able to use these when we write our script.

The first thing that we will need to do is define the value of our variable. We can do this by writing a script where we set it equal to the value we want it to track, in our case the x position of our mouse.

Here, we use one of the new blocks created when we made our variable. We defined our MouseX variable to be the x position. Note that it is important that this script be in the code for the Mouse sprite, so that the value is set at the x position of the Mouse Sprite and not the x position of the Cat Sprite. As always, we need an initiating action, and here we need a forever loop so that the variable continually changes as the x position of the Mouse changes.

Similarly, make a variable called MouseY and have it be equal to the y position of the Mouse.

Now finally we can go into the code for the Cat Sprite and write a script to make it follow the Mouse Sprite.



Here we use the variables that we defined to have the Cat Sprite follow the coordinates of the mouse by 80 units for both the X and Y value. We also need a “point towards” block so that the cat's head will actually orient towards the mouse while moving. A forever loop is again needed so that the Cat's position continues to change as the Mouse's position changes.