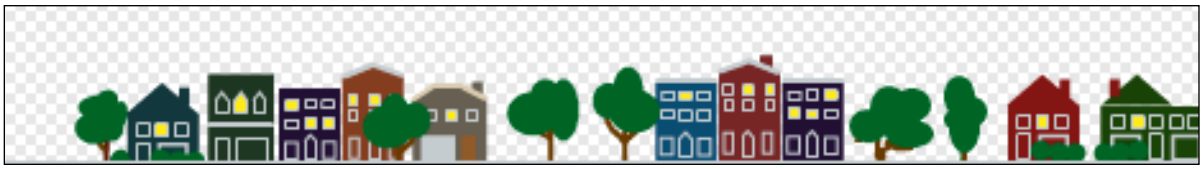
HOW TO CREATE ANIMATED LIGHTNING USING SCRATCH

In this exclusive Kids Code Mississippi tutorial, we'll teach you how to animate lightning strikes using the free programming platform, Scratch, which you can use online at www.scratch.mit.edu. Thanks, MIT!

Click here to see the finished result. Be sure to click the "see inside" and then activate turbo mode in the "edit" menu.

Step 1: Create a town sprite

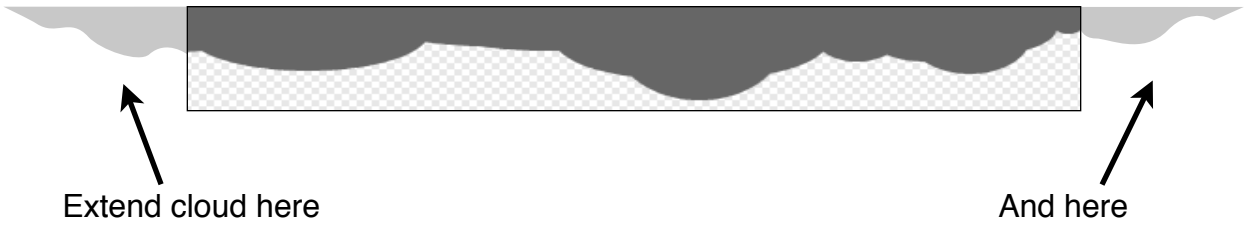
From the Sprite Pane, click the paint brush icon to bring up Scratch's Paint Editor. You can use either the built-in drawing tools (use "Vector Mode" for best results) or import an image created using the graphics program of your choice. Just be sure to size your image to fit the 480 pixel x 360 pixel screen and use the "upload backdrop from file" option in the Stage Panel.



Step 2: Create a cloud sprite

Next, create a new vector sprite in Scratch's Paint Editor. You can form the cloud shape with several overlapping ovals.

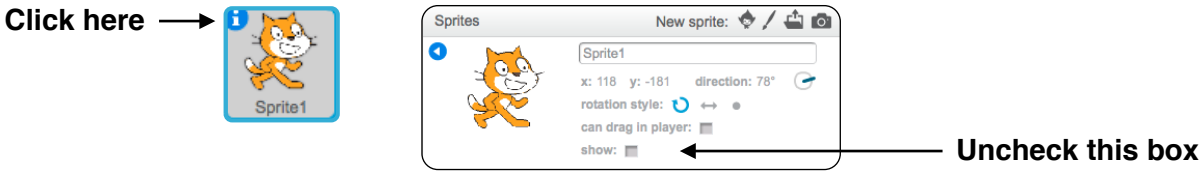
Note that the cloud sprite needs to be wider than the 480 pixel stage size to allow the sprite to move to the right and left. Getting it just right may take a few tries.



Step 3: Create scripts for lightning strikes and sound effects

In Scratch, all code has to be connected to a sprite or backdrop. **The first four scripts (below) are connected to the default Cat1 sprite.** It might seem odd, at first, but don't worry. We're going to hide the sprite.

How to hide your sprite

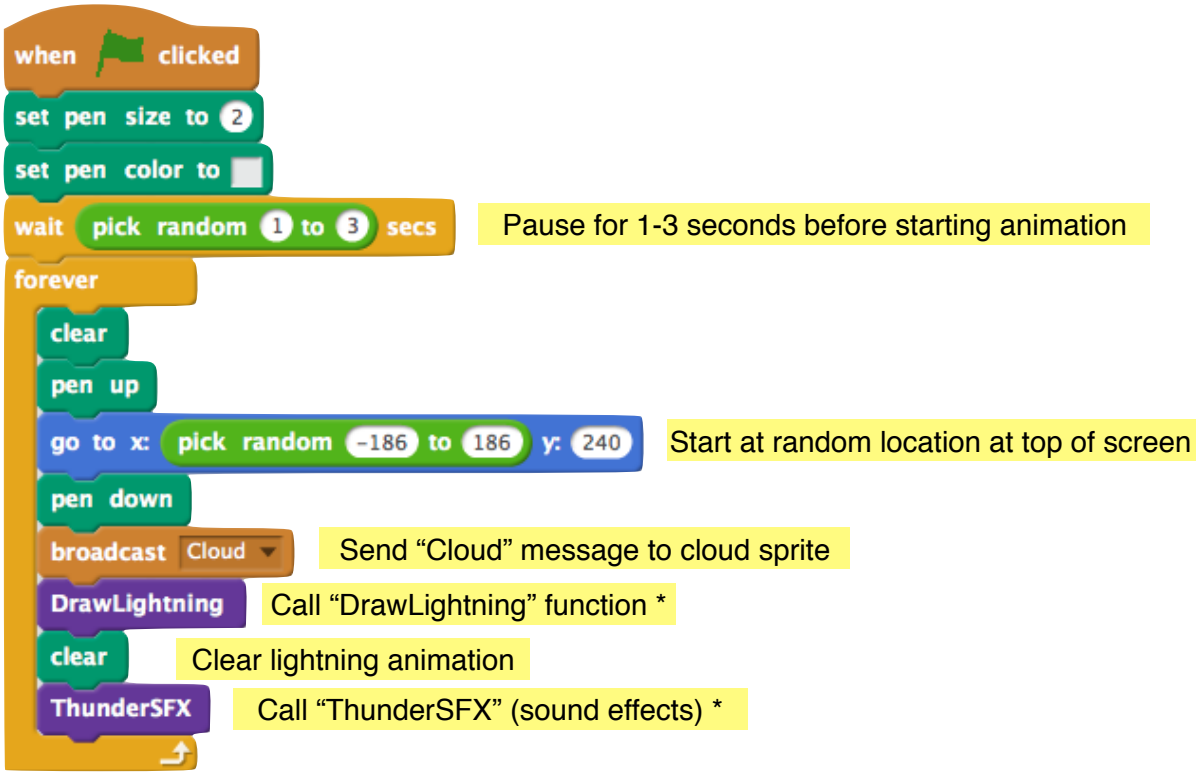


Now that we have our sprites ready, we're going to start adding code. Make sure that you have the Cat1 sprite selected so that you are adding code to the correct sprite.

The Main Script

The first script sets the starting conditions for the animation (setting the pen size and color), pauses for a few seconds and then begins a loop which will repeat until the program is stopped. The script uses a Broadcast Block to send a message to the cloud sprite and "calls" (triggers) two additional scripts via the purple blocks.

We're calling this script the "main script" because it's the one that runs when you hit the starting flag and because it contains the main loop that keeps the program running until it's stopped.



*	**Ignore the purple stack blocks for now. You'll add those later.**

The “DrawLightning” Script

The next script is built using the “More Blocks” feature of Scratch. This type of code is called a function or subroutine in other programming languages.

Functions are a great way to simplify your code because they allow you to “call” the same blocks of code from different parts of your program. That way you don't have to repeat the same blocks over and over. Functions can make it easier for others to understand your code by separating blocks into groups that perform special tasks. “DrawLightning” draws lightning. Easy, right?

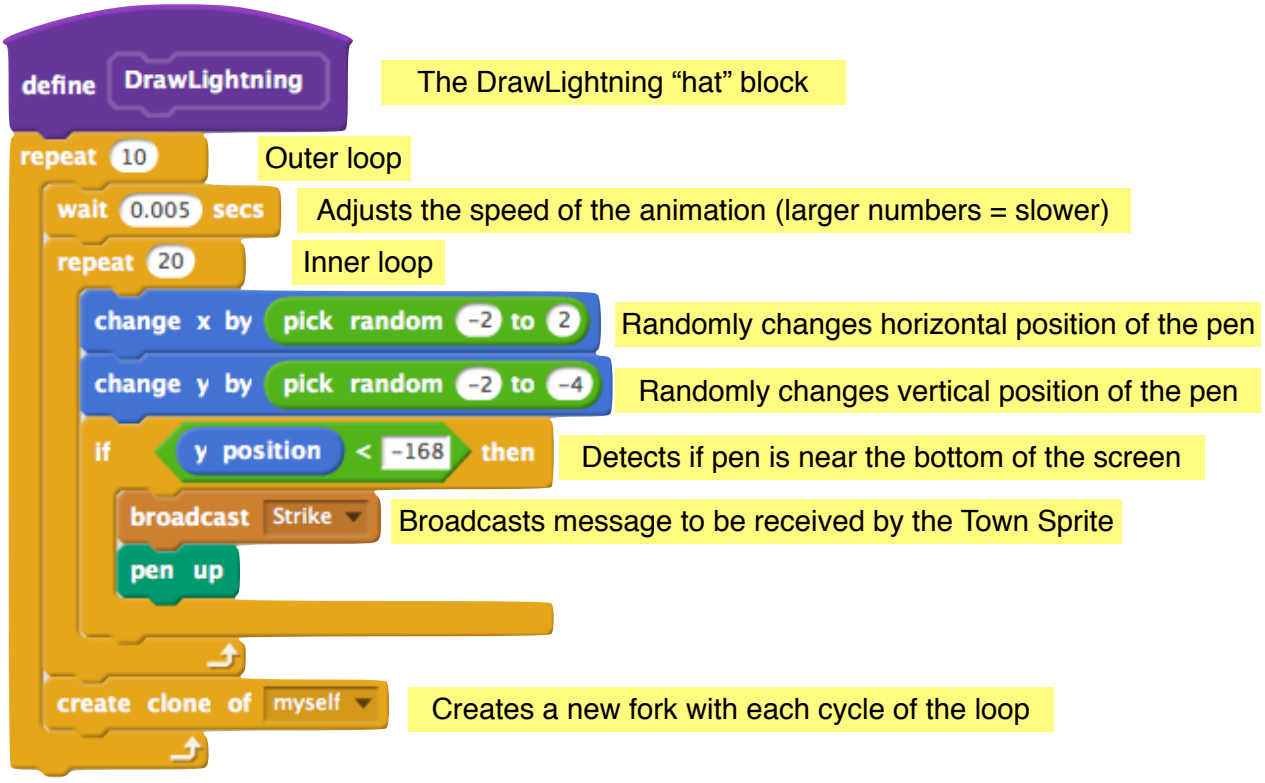
These blocks work a little differently than other blocks in Scratch. First, you click on the More Blocks button in the Block Palette. Then, you click the Make a Block button, which will open up a new pop-up window.

Name your block “DrawLightning” by typing in the pink area, and then hit “ok.”

You should now see two new blocks – a “hat” block in the Scripting Area and a stack block in the Block Palette. You will define DrawLightning by dragging new code blocks underneath it, as shown below. This script uses nested loops (one inside the other) to draw the lightning in a downward zig-zag path.

The outer loop uses Pick Random blocks to change the x (left-right) and y (up-down) direction of the lightning by a random distance, while the inner loop detects when our lightning has reached the lower part of the stage (-180 is the bottom). When it does, the “Strike” message is broadcast and the pen raised so that we can move to a new starting point for the next lightning bolt without drawing a line back up to the top of the screen.

The outer loop does one other thing: Every time the script loops, it creates a lightning “clone” to branch off the main bolt.



*	**Add your "DrawLightning" stack block to the main script now.**

The "ThunderSFX" Script

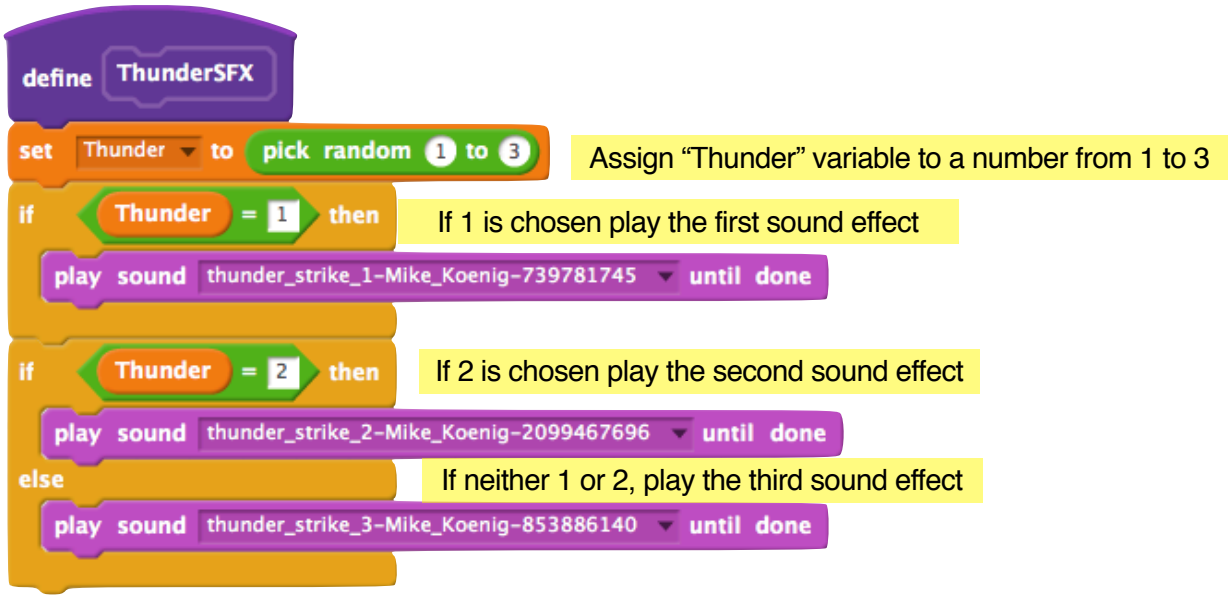
Now it's time to use the More Blocks feature to define a new script called "ThunderSFX" using the same process as above.

Before we can define ThunderSFX, we have to create a Variable. Click the "Make a Variable" button in the data palette and name it "Thunder." By default, the variable will show up on the stage. We don't want the variable to be visible, so uncheck the box next to the Thunder block in the data palette.

We'll use a Pick Random block and two Control Blocks (an "if" and an "If/Else") to assign three different sound effects. We'll use the random numbers to decide which sound to play. You'll need to find and download the sound effects (I found mine at soundbible.com) and then import each one. Click on the Sounds Tab, and click the "upload sound from file" button.

You can eliminate the silence at the beginning of each sound by selecting that portion of the audio waveform and hitting delete on your keyboard.

Here are the code blocks you'll need to define ThunderSFX:

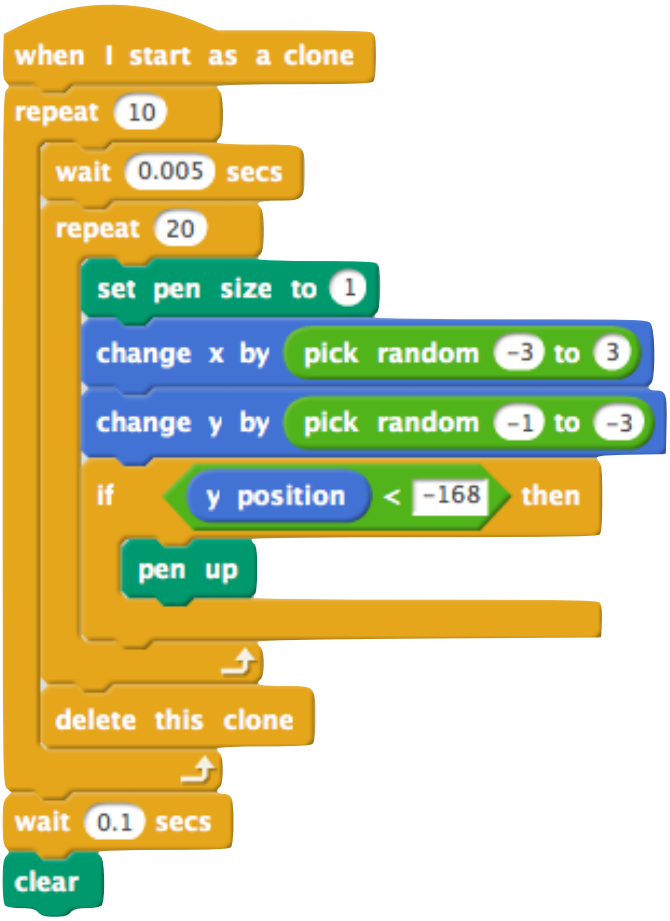


*	**Add your "ThunderSFX" stack block to the main script now.**

The “Clone” Script

Now for the last script that will be attached to the Cat1 sprite. This one gets activated every time the “create clone of myself” block is executed in the DrawLightning function.

This script is defined in a very similar way to the main lightning script. You can play around with the numbers to see which gives you the best results. The “delete this clone” block keeps the path of the forks relatively short compared to the main bolt.



Step 4: Create scripts for Town sprite brightness and color effects

The Town sprite uses two scripts and they are both very simple. One sets up the starting settings for when the green flag is clicked. The other lights up the town when a lightning bolt approaches the ground. These scripts use the Brightness effect, found in the Looks Palette.

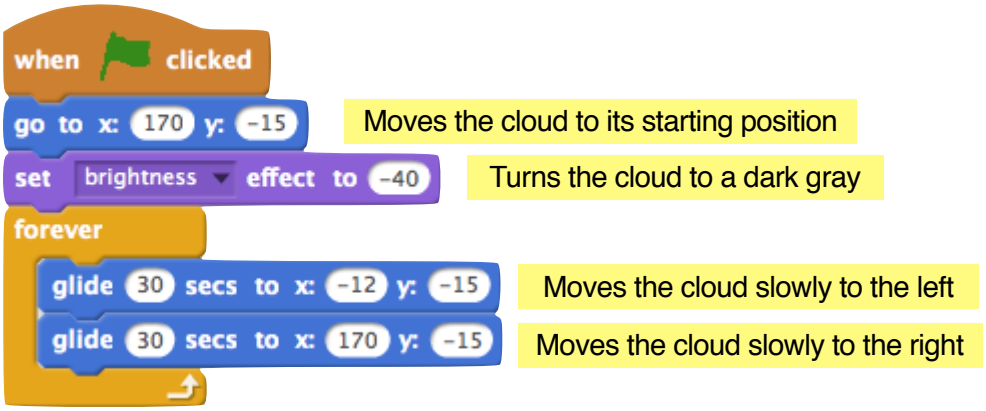
This value turns all but the brightest colors to black

This value displays the normal sprite colors

After a brief pause, the sprite returns to darkness

Step 4: Create scripts for Cloud sprite brightness and color effects

We're in the home stretch. The Cloud sprite uses two scripts. The first script sets up the brightness of the cloud and its starting position and also animates the movement of the cloud. (You'll need to adjust the exact numbers to work for your cloud.) The second script lights up the cloud in response to each lightning strike.

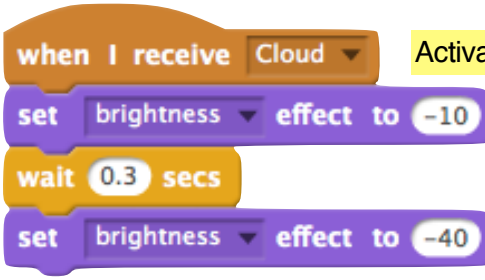


Moves the cloud to its starting position

Turns the cloud to a dark gray

Moves the cloud slowly to the left

Moves the cloud slowly to the right



Activated when "Cloud" message broadcast.

Brightens the cloud to a lighter gray

After a brief pause, returns the cloud to darkness

Step 4: Activate “Turbo Mode” and run program

Now you’re almost ready to run your program. Before you do, be sure to select “Turbo Mode” in the edit menu. And then click the green flag to start the animation.

Did everything work? If not, it’s time to make some tweaks or do some debugging.

What now?

Use your imagination to make this program even better. For example, you could try using random number ranges for the number of repeats in the nested loops in the main script. Or you could turn the animation into a game where you move a new sprite to avoid being struck by the lightning. The possibilities are endless.

Did you enjoy this tutorial? Let us know! And be sure to share any new spin-offs you make from this tutorial. Just drop us a line at kidscodems@gmail.com.