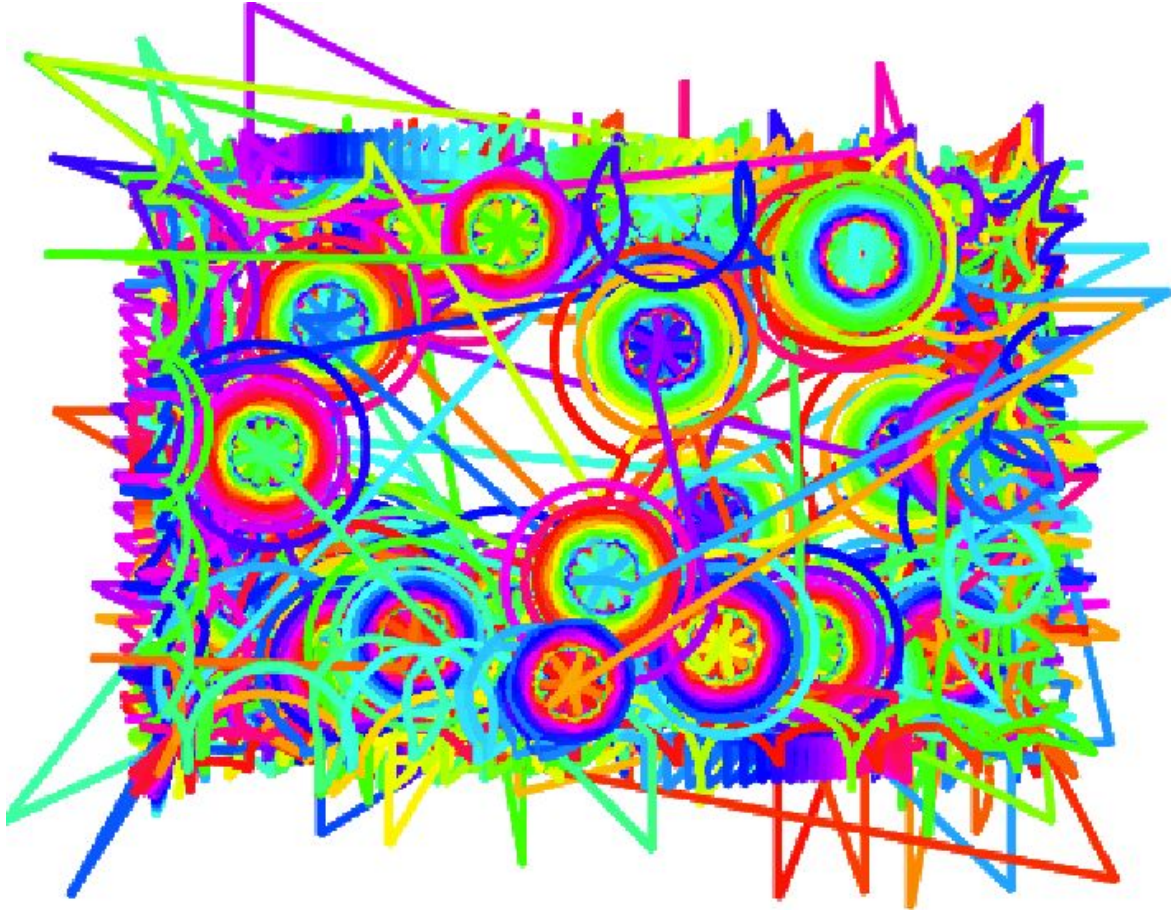


Phoenix



You will need:

Access to the [Scratch](#) website, from a computer or laptop.
(It might also work from an ipad or tablet)

This tutorial is mainly for children and people who have not used Scratch much before. If you get stuck, let us know in the comments section for the example project.

Setup:

If you haven't used Scratch before, click join scratch at the top and create an account. You might need your parent's email to do this, so ask for help. If you can't make an account, you can follow along, but you won't be able to save your work.

Once you've made an account follow the Phoenix Scratch studio [here](#). This is where you can find the template versions of all the workshops we create. Feel free to remix them and post the new things you create.

Wait, what is Scratch?

Scratch is a free, online, **block-based** code editor. You might have used it in school. If you've been to one of our family workshops, you've probably seen us using it there.

There are lots of different tutorials available for using Scratch. Here we are going to focus on showing you ways of using Scratch that link to the work we do at Phoenix. This will include things like animation and drawing.

Growing Art

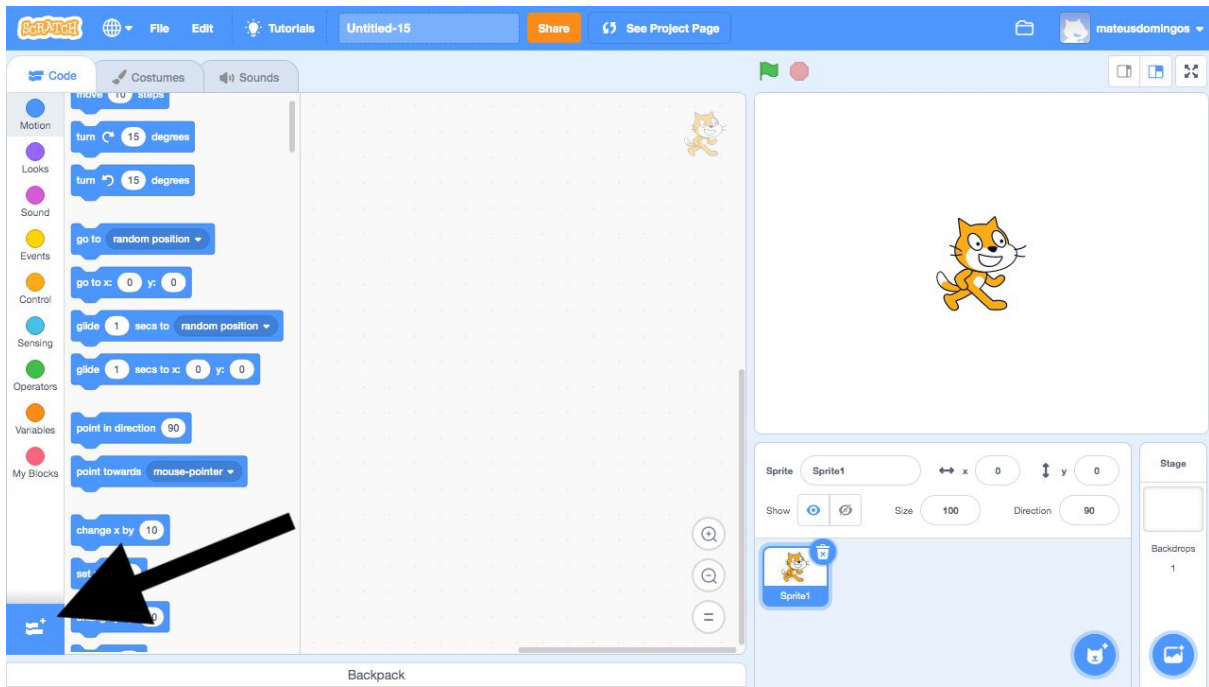
With Scratch we can make art that grows and changes.

To get started, click create, or remix this project,

<https://scratch.mit.edu/projects/378739186>

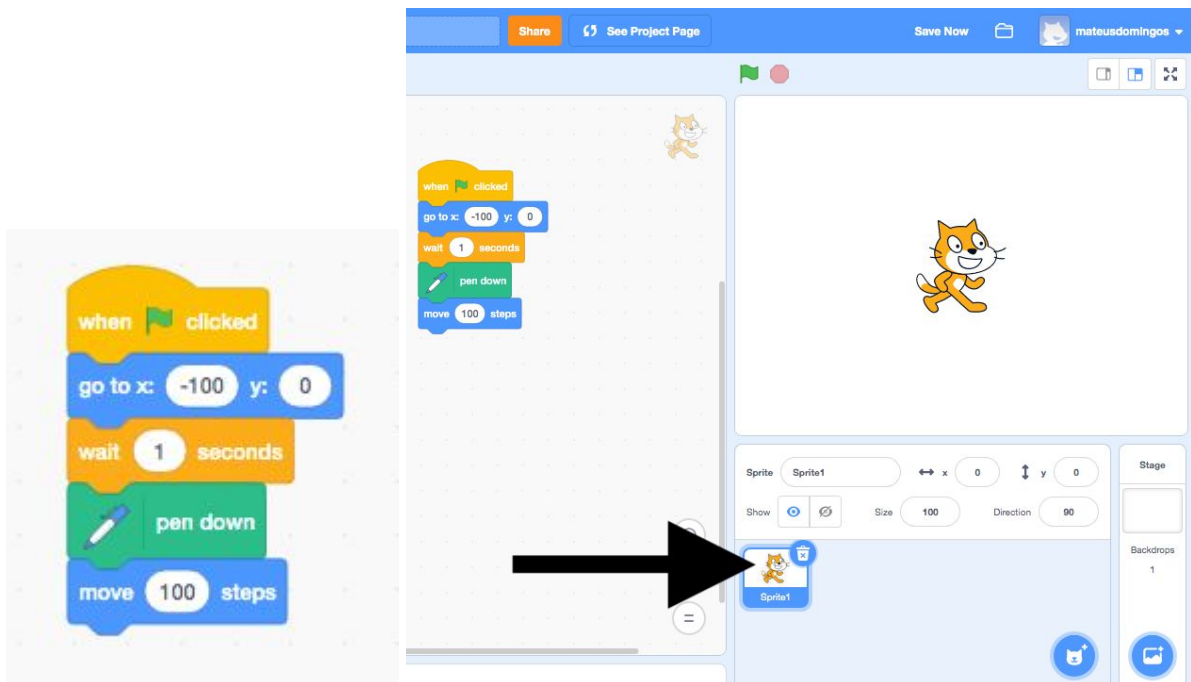
Step 1

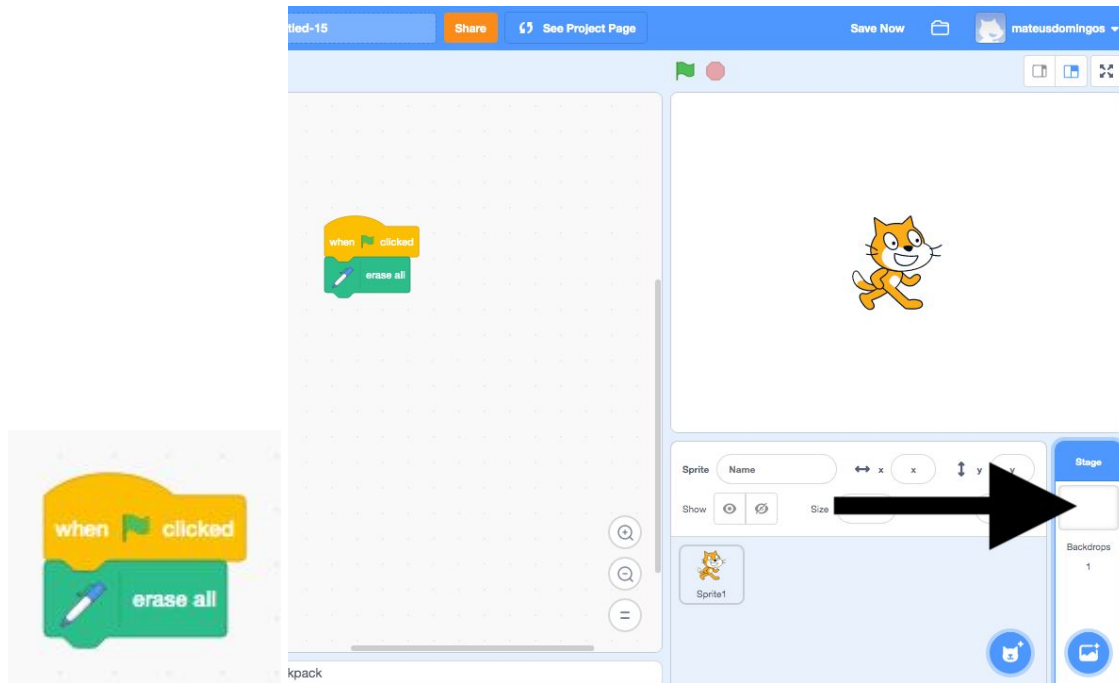
Click 'Add Extension' at the bottom left and choose 'Pen'.



This allows us to create a program that will start drawing.

To start off we will build two blocks of code. One for our Sprite and one for the Stage.





Try to copy these blocks as shown in the image. You can find each of these blocks in the menu on the left. Blocks are split up by category. You'll find most of the ones for this tutorial in, **Events**, **Control**, **Motion** and **Pen**.

Click the Green Flag to run the program and see what happens.

Let's have a look at what our code does so far:

- First of all we move the sprite to a starting point.
- Then we wait for a moment. This gives the Stage code a chance to clear the screen.
- Next we put the pen down.
- Finally, we move the sprite 100 steps to draw a line.

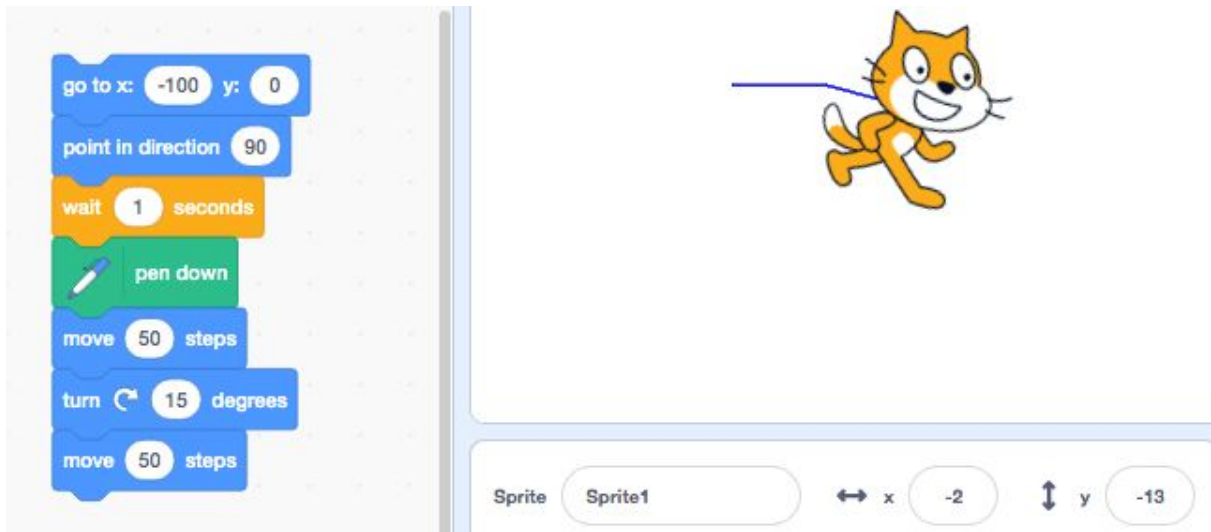
Hint

The blocks are stored in different sections. When you are looking for a block we have used, match the colour of the block to the circle on the left side. For example, movement related blocks are always blue.

Step 2

Let's add another step to our code. Try changing adding another block of **movement**, with a **turn**.

We also need to a **go to** block at the top to make sure the sprite starts in the right place.



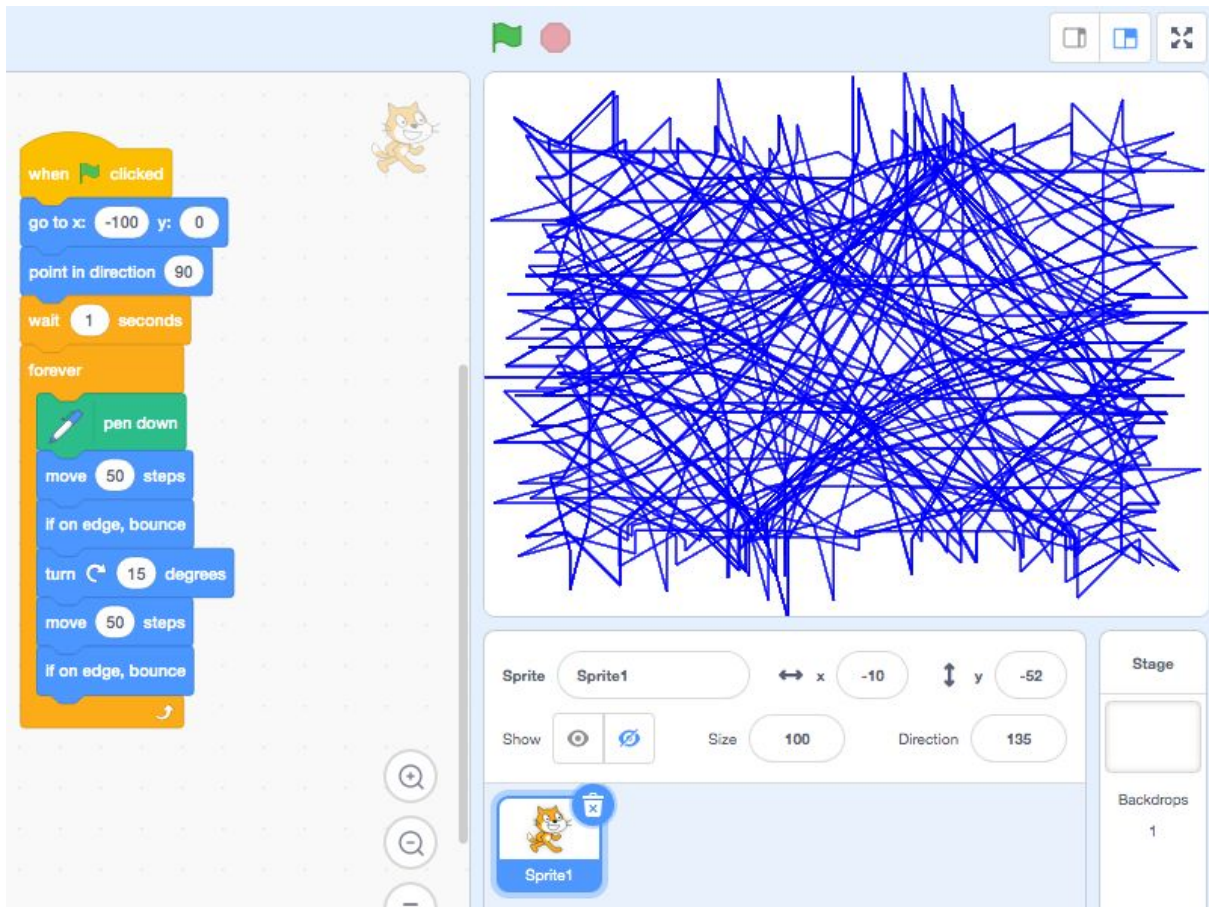
Hint

By now you have probably started to realise that you can make different adjustments to the numbers in these blocks. Spend some time changing the values and seeing what happens!

Step 3

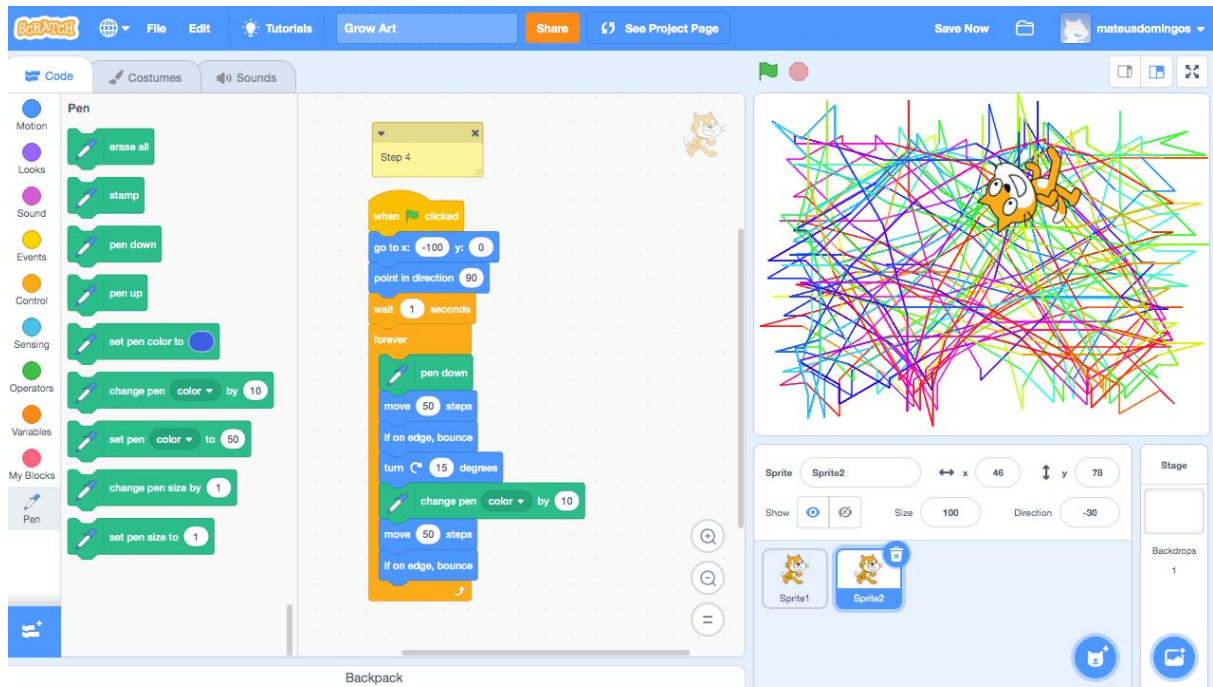
Now we are going to supercharge our code. We do this using the **forever** block. This wraps around other blocks and forces the code to be repeated.

We also need to think about what happens when the sprite hits the edge of the screen. For now let's use the **if on edge, bounce** block. This keeps our drawing where we can see it!



Step 4

So now, let's take a look at how we can change the pen, to make the drawing more interesting. If you are following from the example project, click on Sprite 2.



Open up the Pen blocks and try adding in some extra steps. **Change pen color by "50"** is a good one to start with.

You might notice that the part of that block that says **color** can also be swapped. You can also choose brightness and saturation. These are a bit trickier to handle, so it is best to stick with **color** to begin with.

It is also a good idea to add some extra blocks to the start of the code to make sure the colour starts as you expect it to.



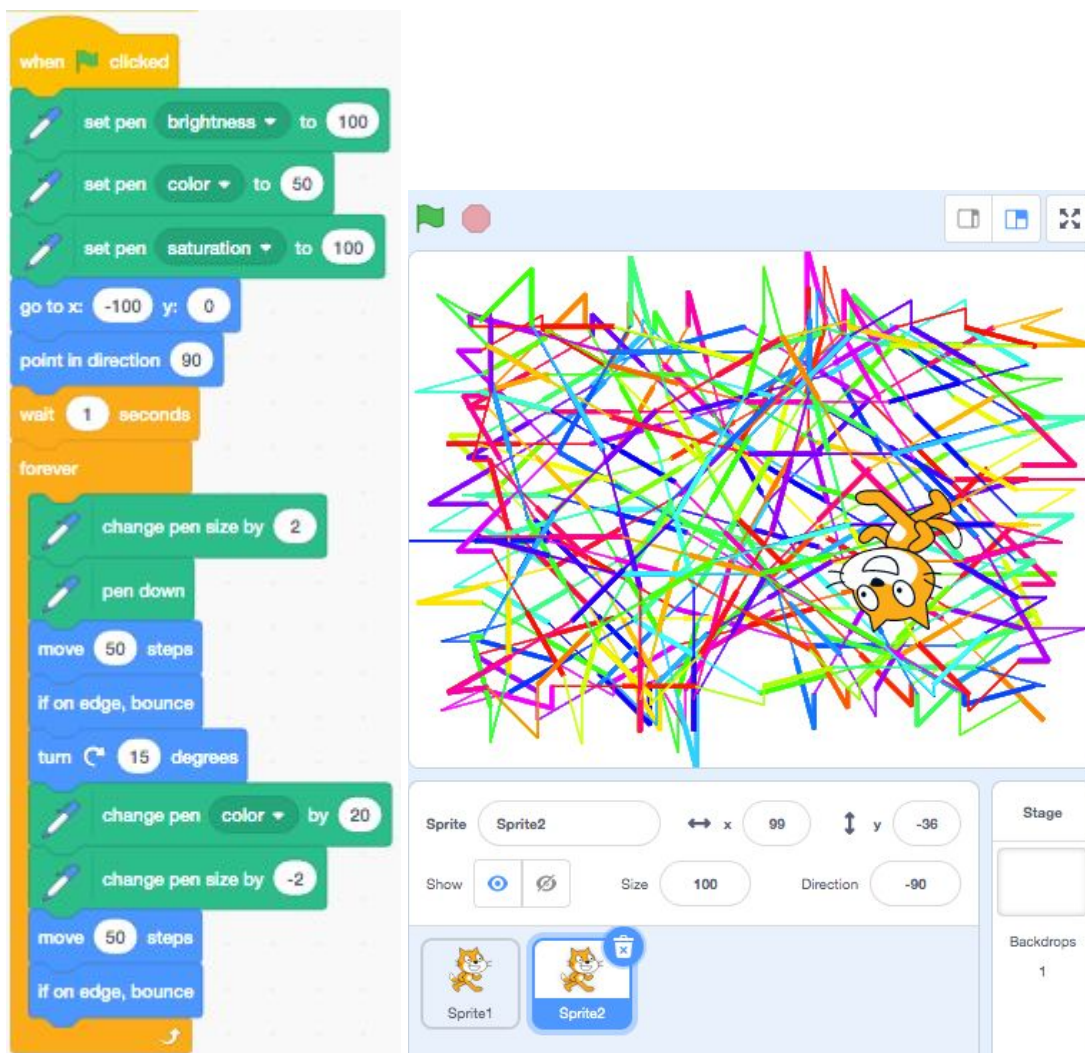
Hint

The colour value is always a number between 1 and 100. If you are changing the value by 20 each time it will go, 1, 21, 41, 61, 81, 1, 21, 41... and so on.

Step 5

Have you tried putting minus numbers into the blocks yet? Doing this means you can change colour in the other direction, move backwards and other useful things.

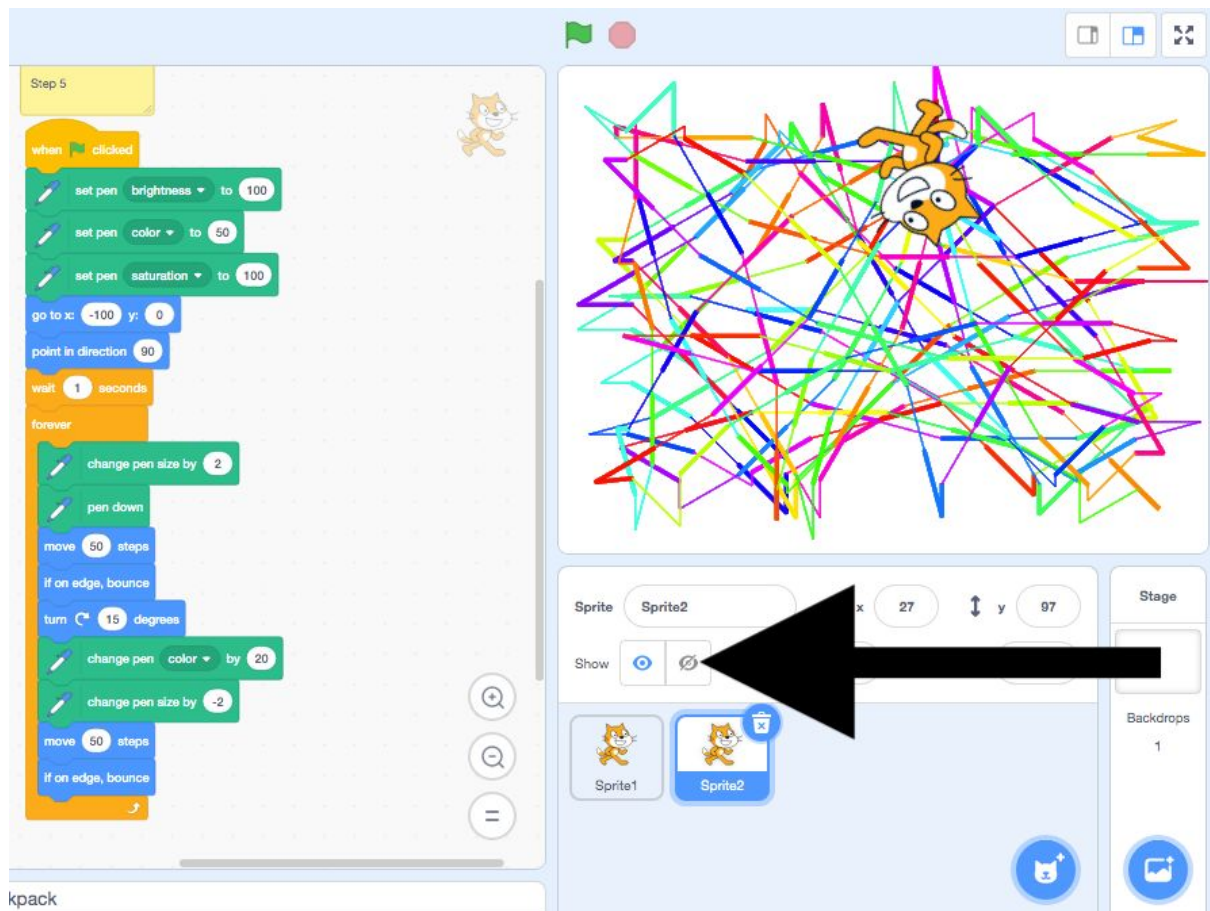
Try adding some blocks like this, to make a change and then change it back. We've made the pen go big and then small.



The image displays a Scratch script on the left and its corresponding stage view on the right. The script begins with a 'when clicked' event, followed by three 'set pen' blocks: 'set pen brightness to 100', 'set pen color to 50', and 'set pen saturation to 100'. This is followed by a 'go to x: -100 y: 0' block, a 'point in direction 90' block, and a 'wait 1 seconds' block. A 'forever' loop contains the following blocks: 'change pen size by 2', 'pen down', 'move 50 steps', 'if on edge, bounce', 'turn 15 degrees', 'change pen color by 20', 'change pen size by -2', 'move 50 steps', and 'if on edge, bounce'. The stage view shows a dense, chaotic drawing of many overlapping, multi-colored lines. A small orange cat sprite is visible in the center of the drawing. The Scratch interface shows the sprite is 'Sprite2' with a size of 100 and a direction of -90.

This kind of thing means you can keep control of changes you make, instead of the pen getting too big and filling the screen or too small and disappearing.

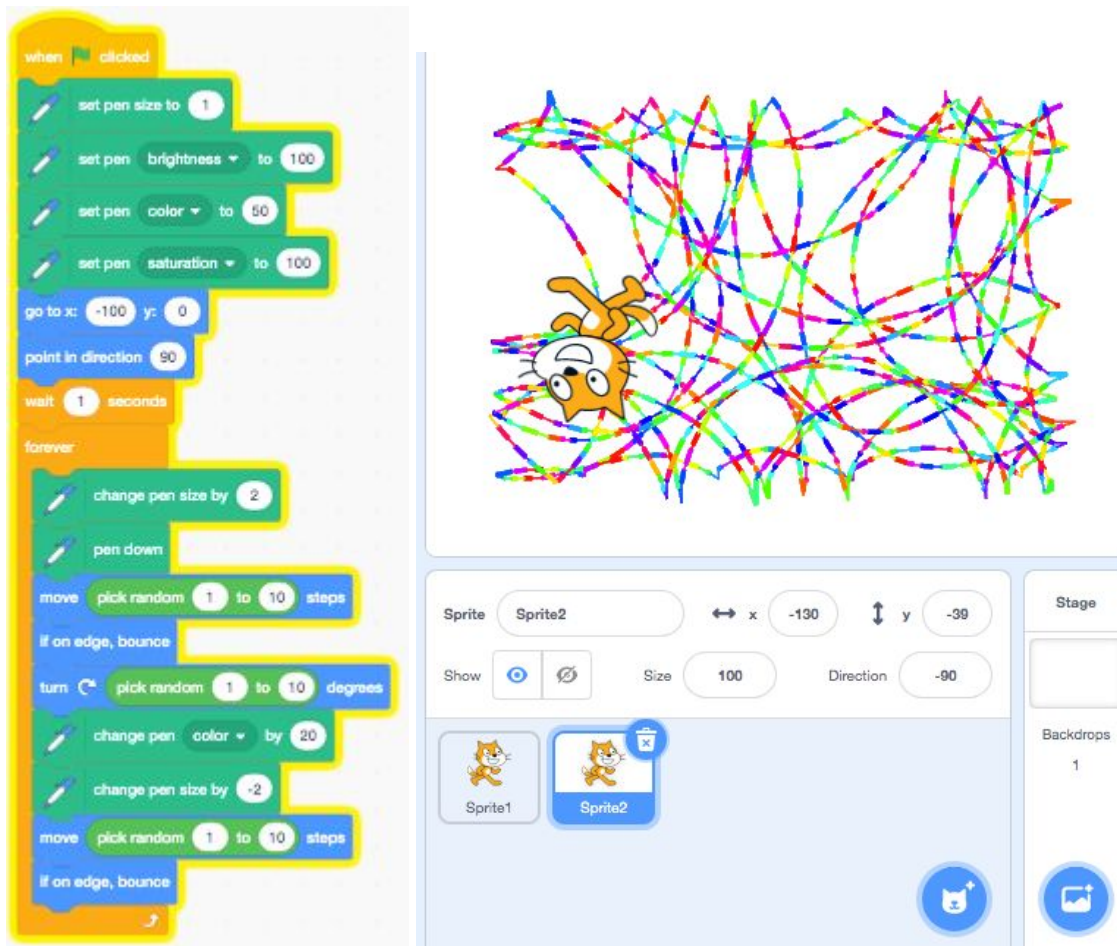
So far we have left the Scratch Cat visible, as it's useful to see where the pen tool is. Feel free to hide these now, using the closed eye icon. You can easily show these again by clicking the open eye.



Step 6

Let's move on to something really cool now. The places where we have been changing values, to control movement, colour and other things, can also hold more than simple numbers.

Things get exciting when we add in the operator block, **pick random from 1 to 10**.



You might notice now that the pen is moving a bit slower. This is just because as we add more steps, there are more things to be calculated.

Part of what makes these drawings fun, is just watching them appear. It doesn't matter too much if it starts going more slowly, it just means you can watch it more clearly.

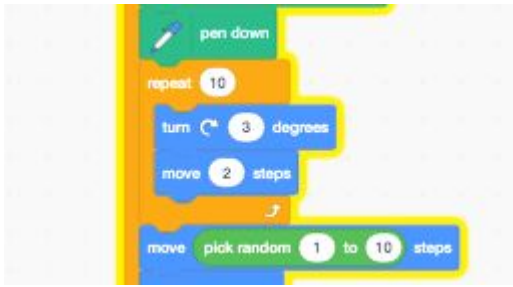
Step 7

These are all the basics you need to start experimenting with drawing in Scratch.

Using what you've learnt so far you can start to build more sophisticated drawing machines.

If you are using the example project, the following ideas are explored in Sprite 3.

- You could experiment using the **repeat 10** control block within the **forever** block.

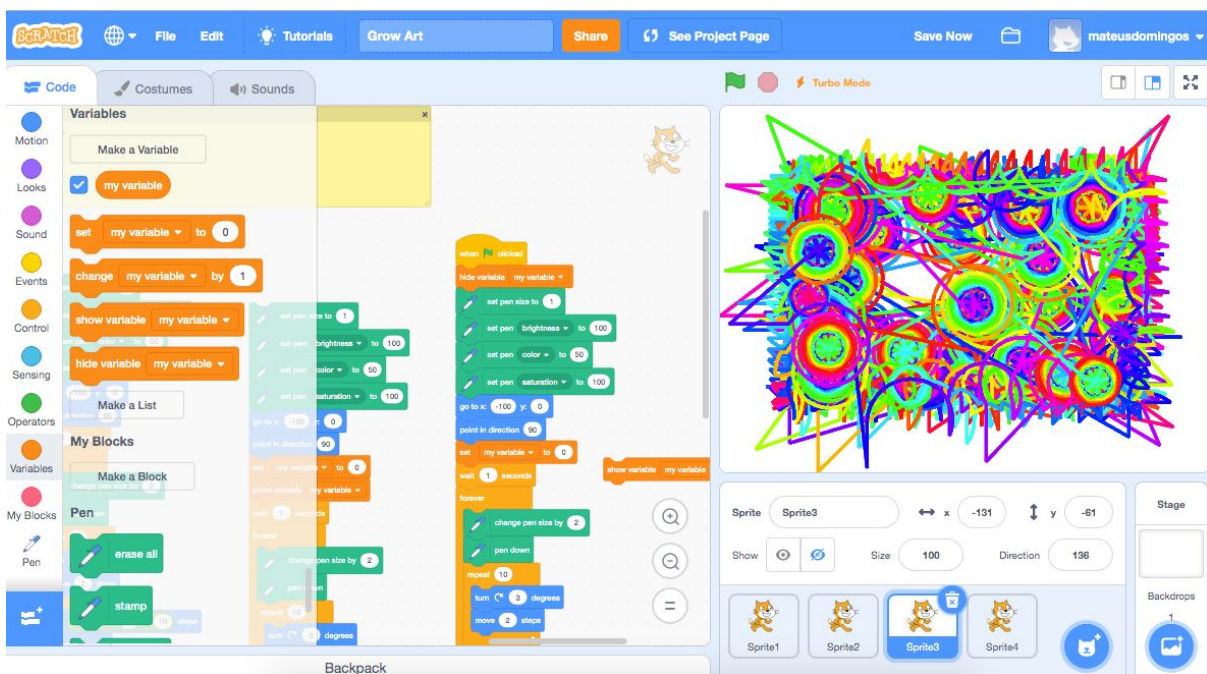


- Right click on the sprite to duplicate it. You now have two pens, following the same code! See if you can make them mirror each other.
- Whatever you make, be sure to save your projects and add them to the Phoenix Scratch Studio. Here you can show other people what you have made, ask questions and find out more!

What next?

We'll have more tutorials uploaded soon, but if you want to continue learning Scratch now there are lots of ideas [here](#).

If you are more familiar with scratch you could try adding in **variables** and using the **broadcast** messages block to make different pens do certain things together.



Further reading

This kind of art is sometimes called generative art. There are lots of different ways of creating it using a computer, but you can also try it at home on paper. Set yourself some simple rules and start drawing. An example might be-

- Start anywhere and draw a straight line in any direction.
- If the line touches an edge 'bounce' it off.
- If the line touches another part of the line, change the colour and continue the line.

If you want to find out more about how different artist use these kinds of ideas, here are some examples to look at:

[Briget Riley](#)

[Kerry Strand](#)

[Grace C Hertlein](#)

Phoenix.org.uk
@PhoenixLeic