Making a Scratch game

TechResort Mini Makers #1126

What do I need?

- A laptop with an internet connection
- *An email address to sign up for a Scratch account (optional)*

What is Scratch?

Scratch is a **programming language** that allows you to **drag and drop** different blocks together to write **code** that can control certain objects and events, to make animations or games!

How are we going to be using Scratch?

In this session, we are going to be using Scratch to make a **game**, where the player (a small object) has to make their way through a **maze** to a certain point, which unlocks the next level.

Firstly, we'll need to **design** our levels, then we can **code** the object so that it can move around the maze.

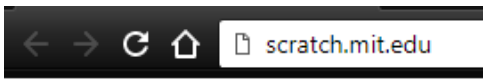
Once we've done that, we'll be able to play our games!

If you've read this page – and you know what Scratch is and what we'll be doing – then turn over and get going!

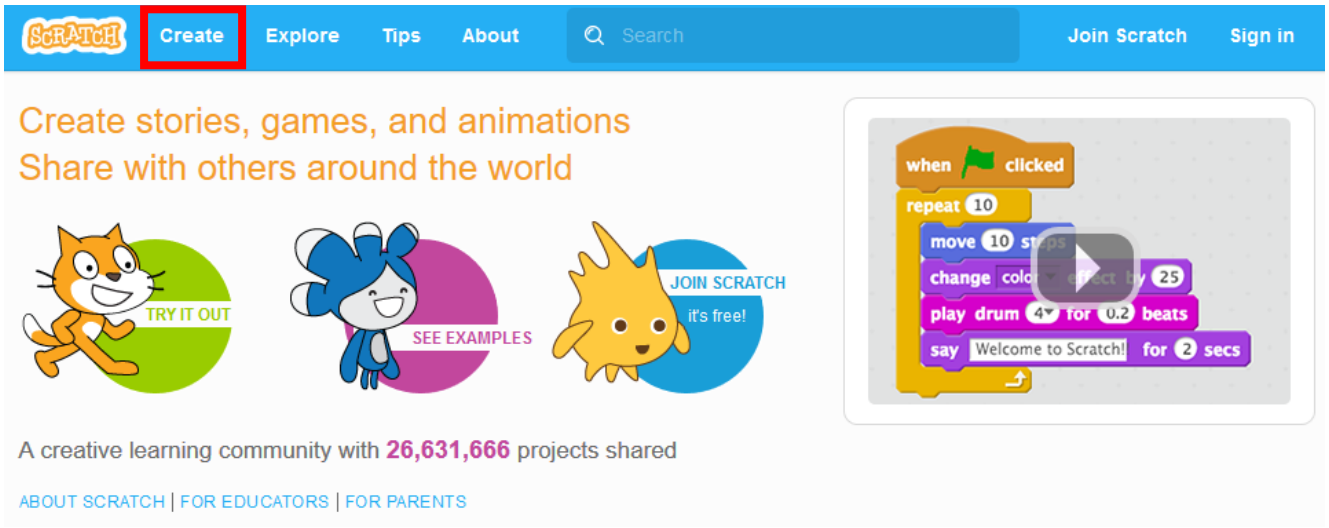
Firstly, we need to open up Scratch.

- Open a **web browser** like **Google Chrome**

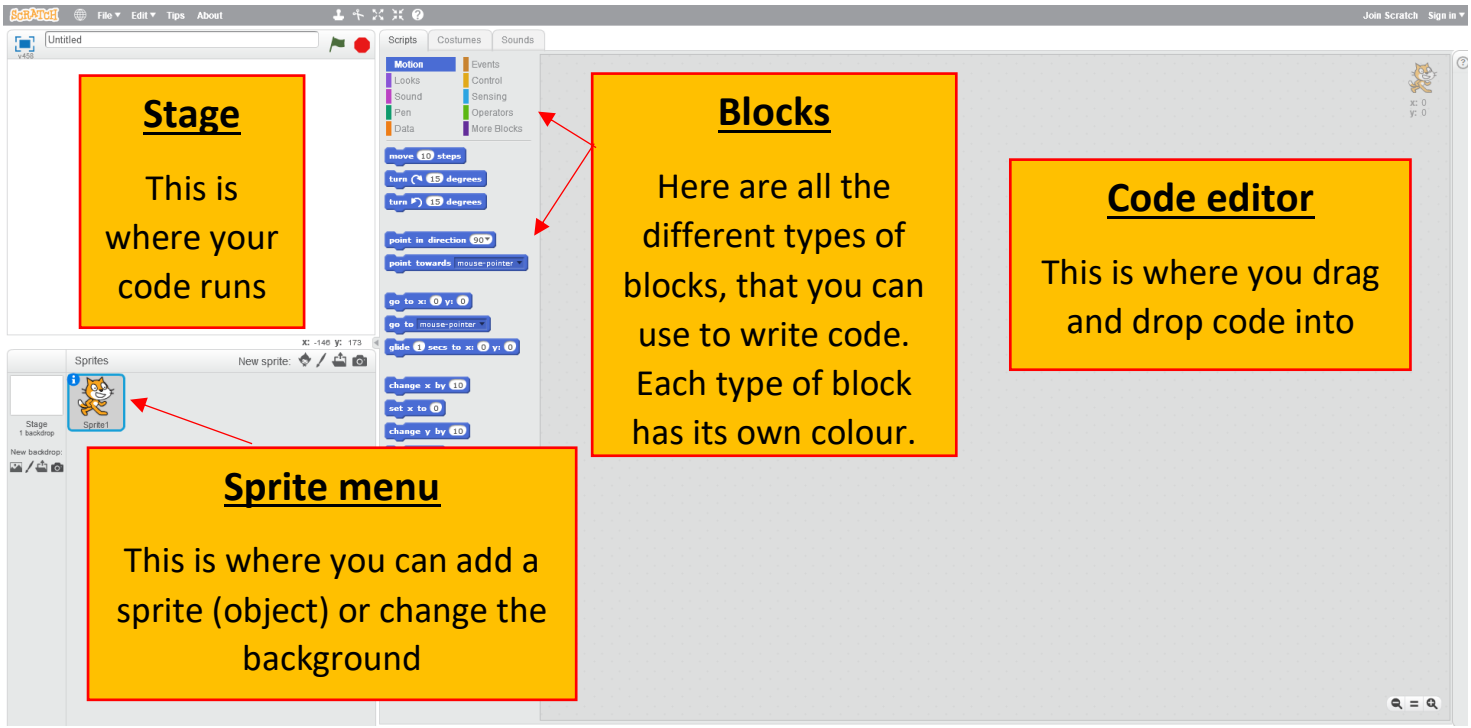
- In the **address bar** (at the top), type in **scratch.mit.edu** and press the **ENTER** key on the keyboard



- Click on **Create** at the top



You'll now be taken into the **Scratch Editor**.



Note that we're using the web-based version of Scratch so you'll be able to continue at home, but there are also offline versions that can be downloaded.

We're trying to make a game where the player, who controls a small object, has to make their way through a maze, and reach a certain point.

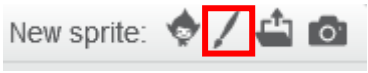
The object will also have to dodge some obstacles along the way, to make it more of a challenge.

Designing the object

Let's start by making a **new** small object (scratch calls them **Sprites**) for the player to control.

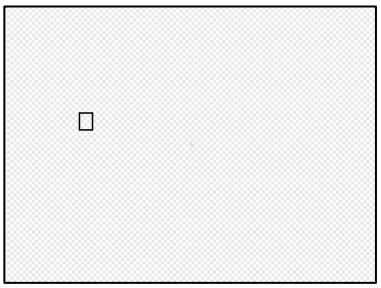
- We can start by deleting **Sprite1.** Right-click it and select **delete**.

- Now, we can make a **new sprite** – in the **sprite menu**, look for the **paintbrush** to paint a new sprite



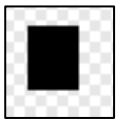
- Now pick the **rectangle tool**.

- Over the drawing area, click and drag to make your shape



- If you want to fill it in, look for the **paint bucket** tool.

- Click inside your object to fill it in.



- You've made your first **sprite** in Scratch!

Once you've done this, go over to the next page to start making backdrops for our maze.

Ask an elf if you are unsure what to do

Now that we've designed the object, we need to make a **backdrop**. This will be the **maze** that the player has to try and navigate.

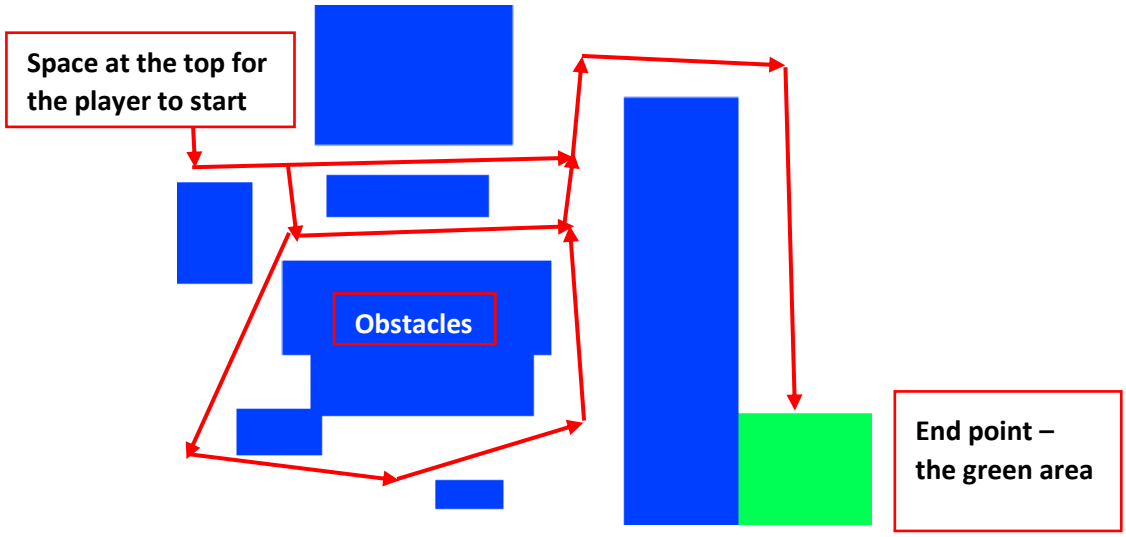
Designing the backdrop

Near the sprite menu, look for **Stage**. Click on this and it will let you edit the **backdrop**.

Now, the maze design is up to you. There are only a few rules that you have to follow for our code to work later on:

- **Have the obstacles in a single colour. This must not be the same as the colour of your sprite.**
- **Have the 'completed' section in a different colour to the player sprite and the colour of the obstacles.**
- **Your sprite (that the player controls) will always start in the top left of the screen, so make sure there is a gap there**
- **Make sure there is enough space for the player (the sprite) to fit through all the gaps, and that there is at least one route to the end**

For example:



Remember that we'll want levels, so that once the player has finished one, they progress to the next.

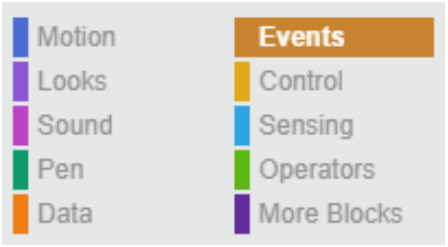
- **Add two more backdrops so that there are at least three – following the same rules as last time**

- Remember to look at the example on the last page if you are stuck

Now, we can start writing some code for our **sprite**.

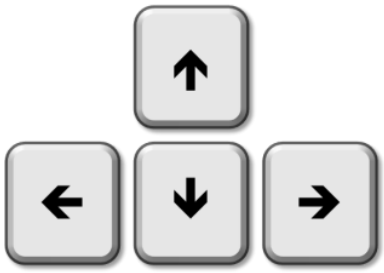
- Click on it in the **sprite menu** near the bottom of the screen

- Under the **Events** category, look for the block that says *when space key pressed*



- Drag this out into the coding area.

We're going to use the arrow keys in order to move the player about.



- Look for the **down arrow** next to the word *space* – click it and look for **up arrow** – so the block will now say *when up arrow key pressed*

- When we press the **up arrow**, we want to move the player **up**.

- Under the **Motion** category, look for 

and connect it so that it snaps underneath the *when up arrow key pressed*

- Your code should now look like:

Our code changes the 'y' co-ordinate. This can make something move up, or move down. To move up, you use positive numbers. To move down, you use negative numbers (less than 0).

Testing time

Now click the green flag button near the top of the stage to run your code. Press the up arrow, and your object should move up!

Now that your object moves up, **your challenge is to make it move down**.

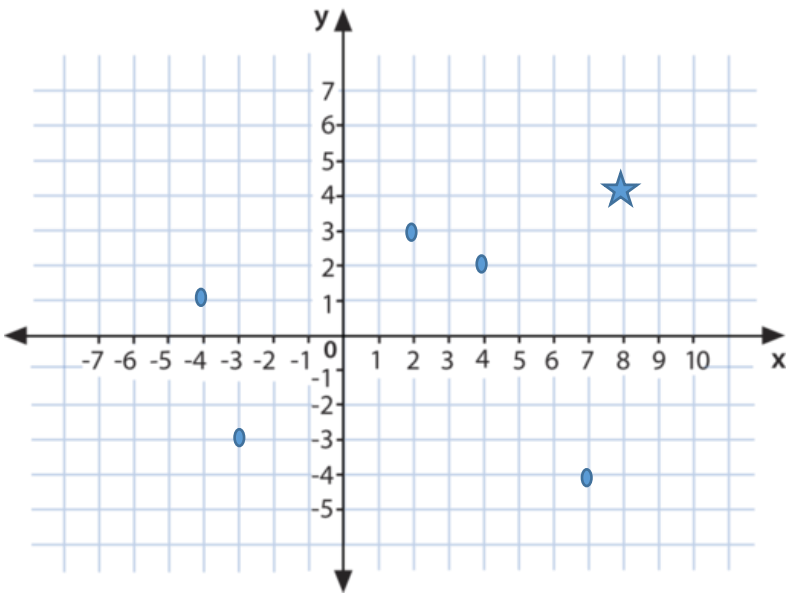
To move it up, we told Scratch to **change y by 10**. This changes the **y co-ordinate** by **+10**, making it move up.

A co-ordinate system works by giving each point an X and a Y reference.

- What would happen if you changed 10 to 5? Why?
- What would happen if you changed 10 to -10? Why?

Just to ensure you understand how coordinates work, have a look at the grid below. Each point on the grid is referenced by how far to the right or left it is (the x co-ordinate) or how far up and down it is (the y co-ordinate). The x co-ordinate is always shown first, and that y one second. So the position (8,4) means that you move right to position 8 and up to position 4. It's a bit like going along a corridor before you go up the stairs.

Remember to go along the corridor (x axis) and then up the stairs (y axis).



Which 5 points have I marked on the grid using circles?

- Each point on the grid has a coordinate, for example, (8, 4) is marked with a star.
- The Scratch stage uses a coordinate system to track sprites.

- You can see the x and y values of your mouse position on the bottom right corner of the stage.

Now you should have the code for moving your object up, and moving it back down again.

To make sure your object can move properly, we'll need to make sure it can move left and right as well – these are quite similar to moving up and down.

Notice that this time we'll be using the **x** axis in order to control left/right movement.

- The code for moving left should look like this:

- **See if you can make the object move to the right!**

Once you've done this, you should be able to move your object left, right, up and down. Let's test this!

Testing time

Now click the green flag button near the top of the stage to run your code. Press all of the arrow keys to make your object move.

If your object doesn't move in the right direction, try:

- Making sure that the **Motion** blocks link up to the right **Events** blocks
- Making sure that you are using the correct **arrow key**
- Making sure that you are using the correct **x** and **y** movements
- Asking somebody sitting next to you or an elf

Now that we know that our movement code works properly, we can make it move onto the next level when you reach the end point.

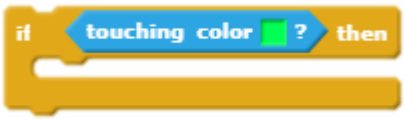
First of all, Scratch needs to know **when** the object is moving – it can do this by using something called a **broadcast** statement.

- Under the **Events** category, look for block that says **broadcast** *message1*
- See if you can **rename** it from *message1* to *moving*

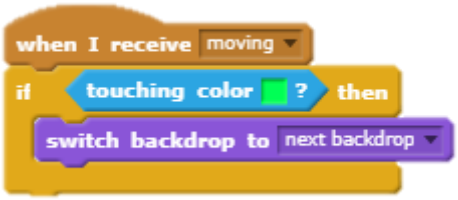
- Drag **4** of these blocks out, and put them under each of your **Motion** blocks.
- Now, we need to make it do something when we hit the “end point”
- We need to use a **when I receive** block from the **Events** category

- Now, we need to use an **if** statement to check when the player has reached the “end point”. We carefully made the end point a special colour and now we’ll use this colour.
- You’ll need to use **sensing** block to check to see if it touching the “end point”. You’ll need to pick the colour of your end section. To pick the colour, click in the coloured block, then click your end point.

- Now drag this inside of the **if** statement to make it look like the one on the right:



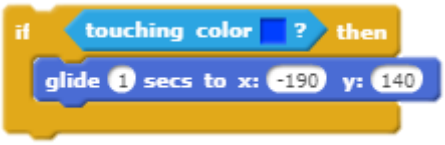
- Having detected the end point we need to tell our code what to do when we get there. Check the **Looks** section to look for a block that says **switch backdrop to next backdrop** and add it to your **if** statement.



If the player touches one of the obstacles, we'll want to **reset** to the start, so they can have another go.

We'll want to use a similar **if** statement to check to see if the player is touching one of the obstacles.

- Try coding an **if** statement like this one:

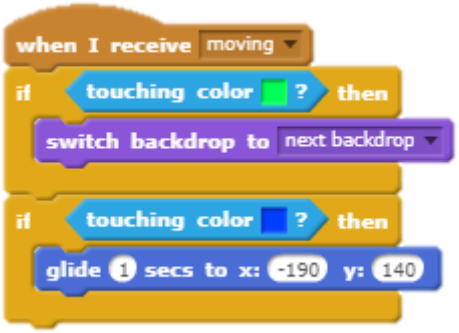


This uses the co-ordinates x=-190 and and y=140 as these co-ordinates are in the top-left corner of the stage. What happens if you change these values?

If you've used a different colour to **blue** then you'll need to change what colour is used as part of the **sensing** block.

If you aren't sure how co-ordinates work, try reading page 6 or ask an elf.

- You'll now need to attach it to your **when I receive *moving*** block like this:



Testing time

Now click the green flag button near the top of the stage to run your code. Try to reach the "end point" – and see if you reset back to the start when you touch an obstacle.

Although it resets to the start when it hits an obstacle, it'll need to reset when you start the game, as well.

- Drag out a **when** clicked block and attach one of the **glide** blocks that you have used before.

Now, every time you start the game, it should reset the player's sprite back to the start.

Testing time

Now click the green flag button near the top of the stage to run your code. No matter where you are on the stage, you will reset back to your starting position.

Well Done

You've made a cool Scratch game – now try playing it!

Here's some ideas to make your game even better:

- More levels (backdrops) – try making them really difficult!
- A timer, to count how many seconds it takes to complete a level
- A "fails" counter, to see how many times you hit an obstacle
- Multiplayer – have two different players with different objects, so you can make it a race to the finish!

Bonus – signing up for a Scratch account to carry on coding with Scratch – click on 'Join Scratch' in the top right corner to sign up for an account, then follow the instructions.

You'll need an email address to do this.

When you have logged in, your work will automatically be saved into your new account so that you can carry on another time.