[the academy_of_code]

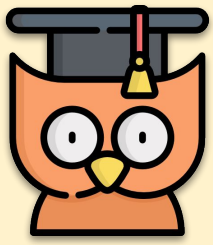
Grade 3

Scratch Games Development II

Games 1-5

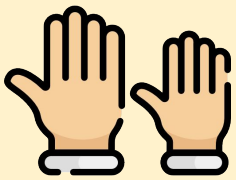
(for Units 1 & 2)

Game 1 - Hungry Shark - ADVANCED

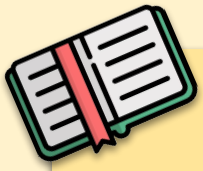


Learning Outcomes:

- Control the main character (shark) which is a shark with the mouse to move around the aquarium eating fish.
- Eating a yellow fish will score 1 mark while eating a red fish will reset the total score to zero.



REMEMBER: If you have any questions, stay in your seat and put up your hand. We love to help!



Grade 2 vs Grade 3 - Compare and Contrast

Have a look at both of the videos below.

On the left, **Hungry Shark** from *Grade 2*.

On the right is our more advanced Version.

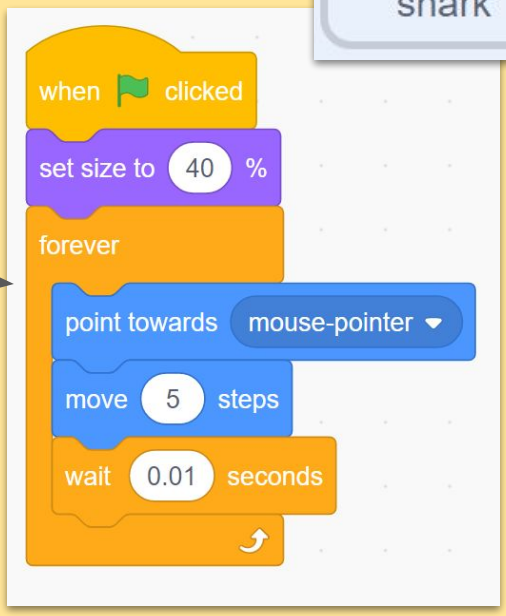
You should have the skills and know-how to work through this document yourself although we are always there to help. Although they have similar themes, they're two very different games.

Stage 1 - Shark's Basic Code

1. Start Scratch to create a new project. Pick an **underwater backdrop** and save the new project as **Hungry Shark Advanced**.

2. **Right-click** Sprite1 (the cat) to delete it. Then select the **shark1-a** sprite from the Animals folder. Rename the sprite as shark.

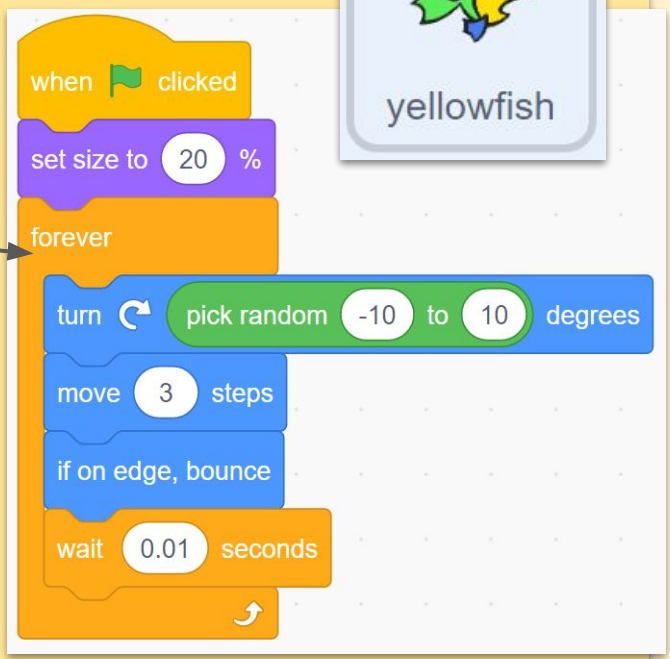
3. Create the starting script for the shark sprite.
Click the **Green flag** icon to start the program testing it by using the mouse to guide the shark moving around.



Stage 2 - Yellowfish's Basic Code

1. Select the **fish3** sprite from the Animals folder and rename the sprite as **yellowfish**.

2. Create the scripts for the yellowfish sprite.
Click the **Green flag** icon to start the program testing that **yellowfish** moves randomly while the shark sprite follows the mouse.

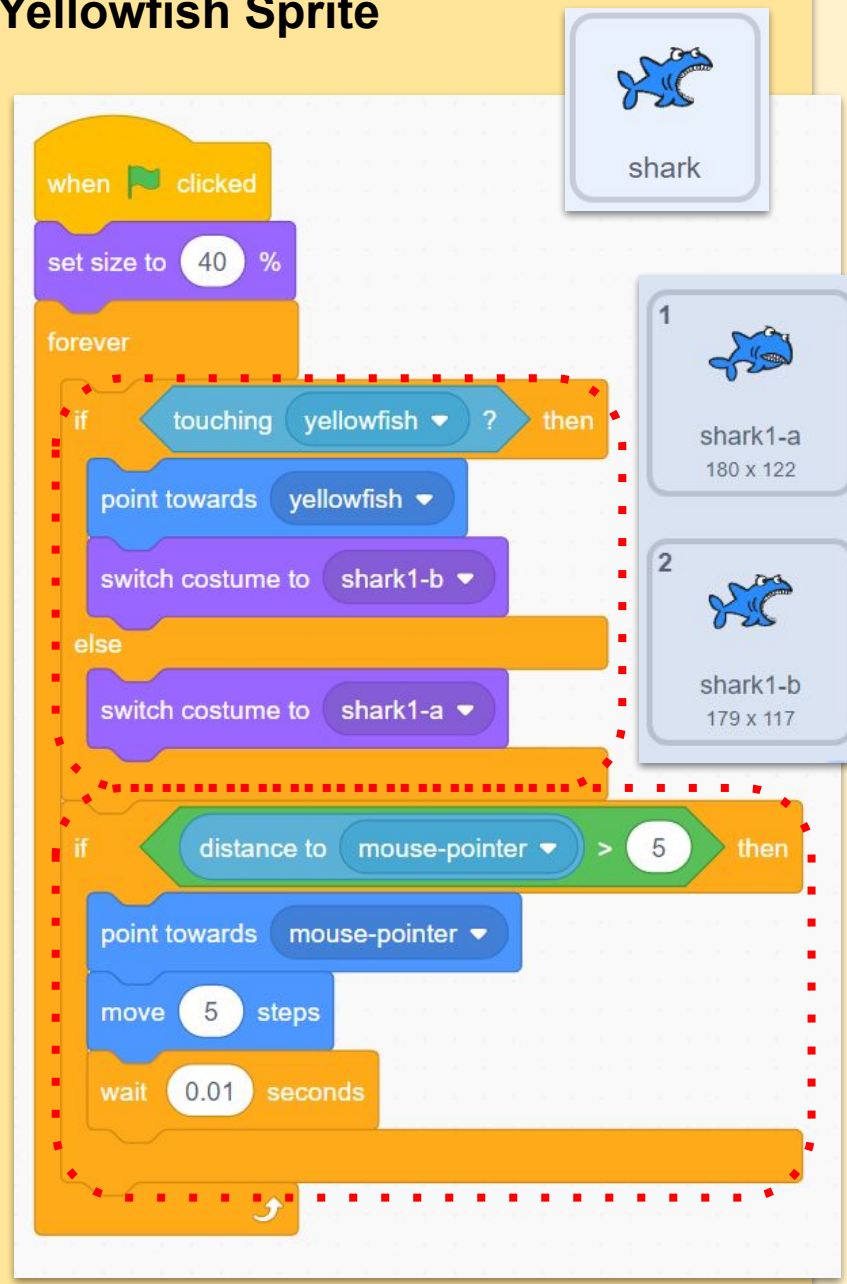


Stage 3: “Eating” the Yellowfish Sprite

1. Select the **Shark** sprite and then click the **Costumes** tab.

2. Click the **Code tab** and modify the script for the **shark** sprite by adding an if-else control block and an if control block.

3. **Save** the project again and click the icon to start the program. The shark should open its mouth as it touches the yellowfish.

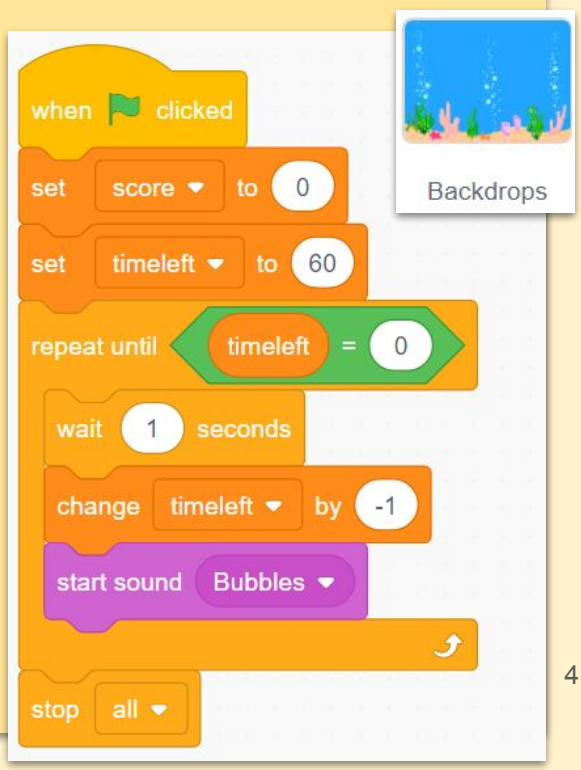


Stage 4: Setting Up the Counters

1. Click on **Variables** and **Make a variable** to create two variables **score** and **timeleft.** Make sure that the **For all sprites** option is checked.

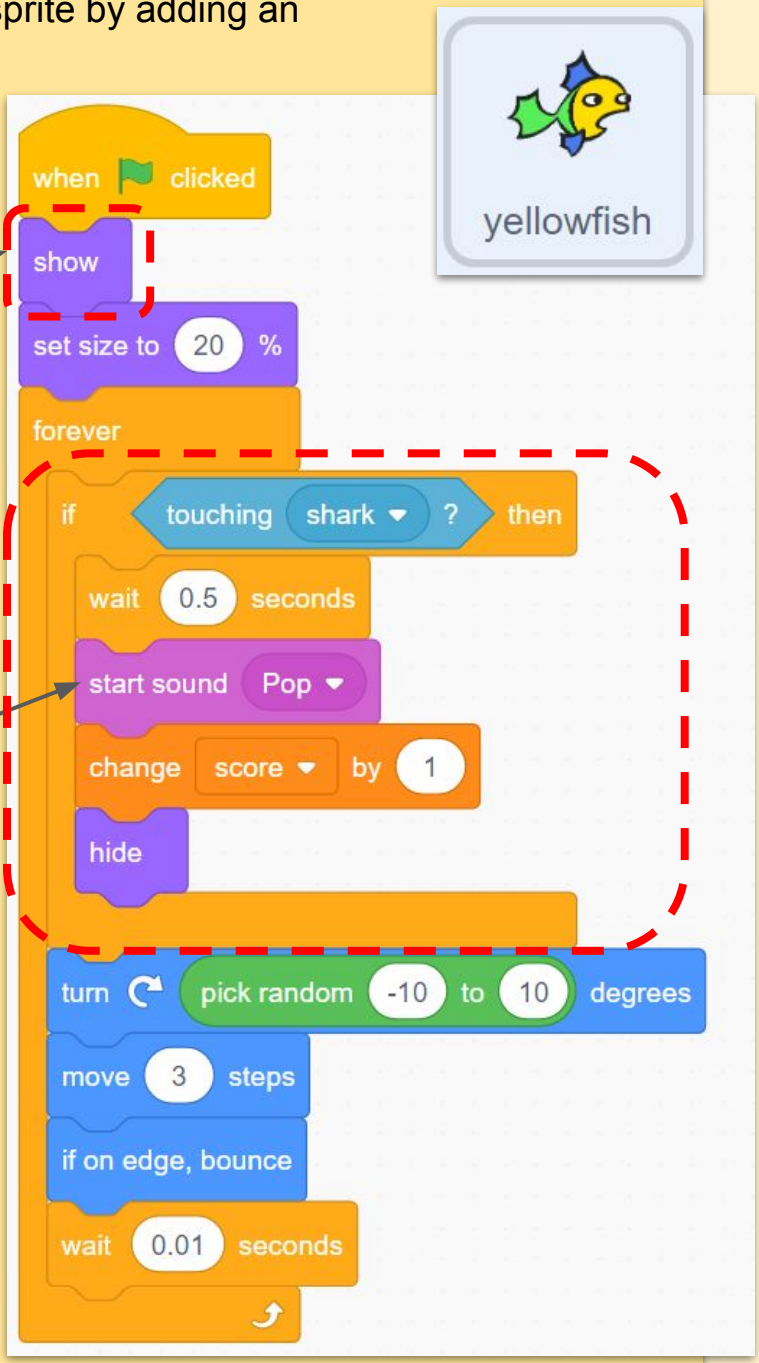
2. Click the Stage to select it and create the script for the Stage.

3. Save your program



Stage 5: Eating and Scoring

1. Modify the scripts for the yellowfish sprite by adding an **if then block**.



2. The **Show** block ensures that the fish appears again at the beginning of each new game.

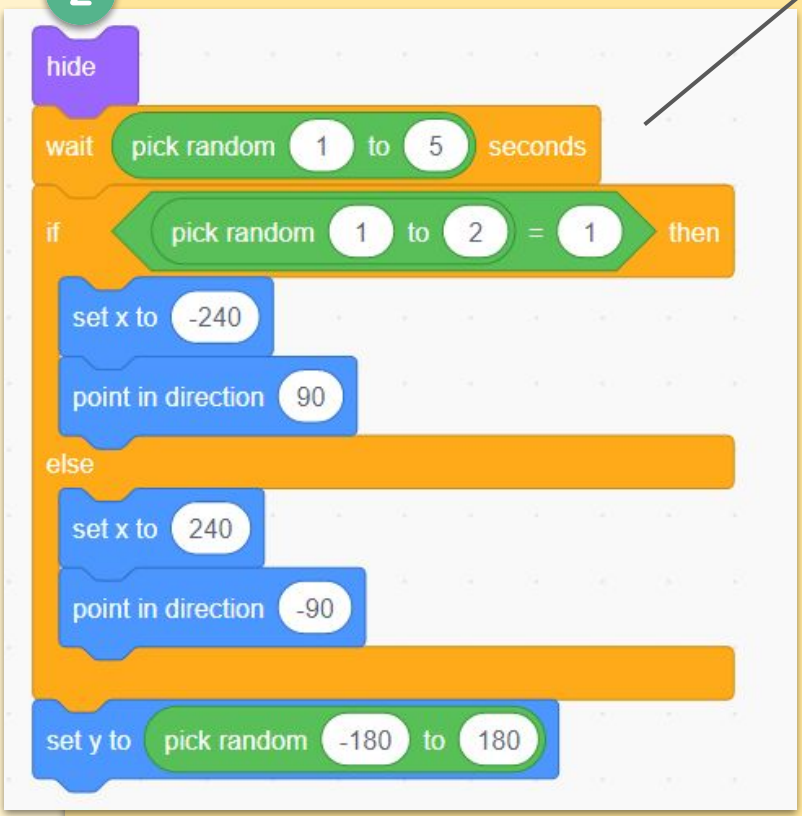
3. The Pop sound can be imported from the Effects folder.

4. Click the **Green flag** icon to start the program. As the shark opens its mouth to eat the yellowfish, a pop sound is played and the counter score is increased by 1. However, the yellowfish sprite is no longer seen again.

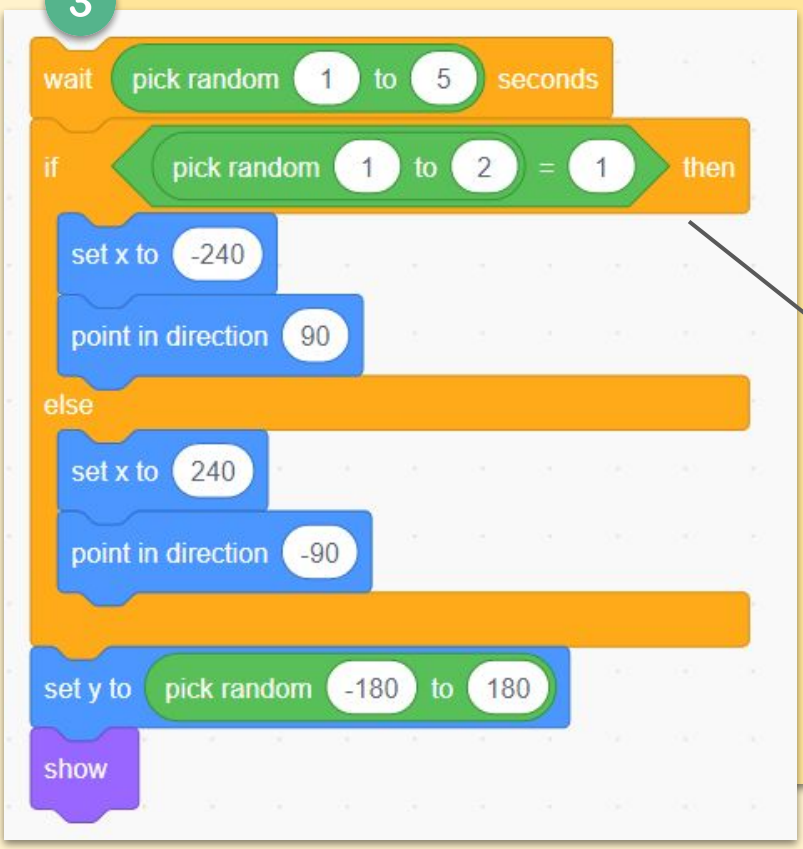
Stage 6: Adding More Fish

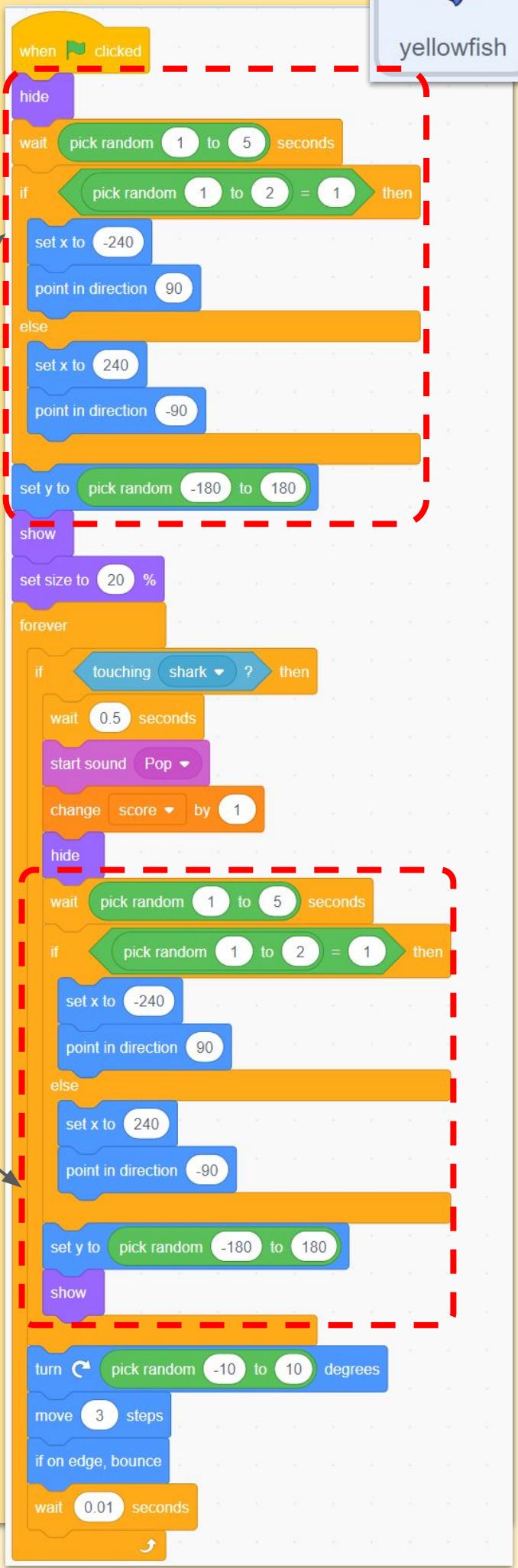
1 Select the yellowfish sprite. Add the following block of scripts just after the  control.

2



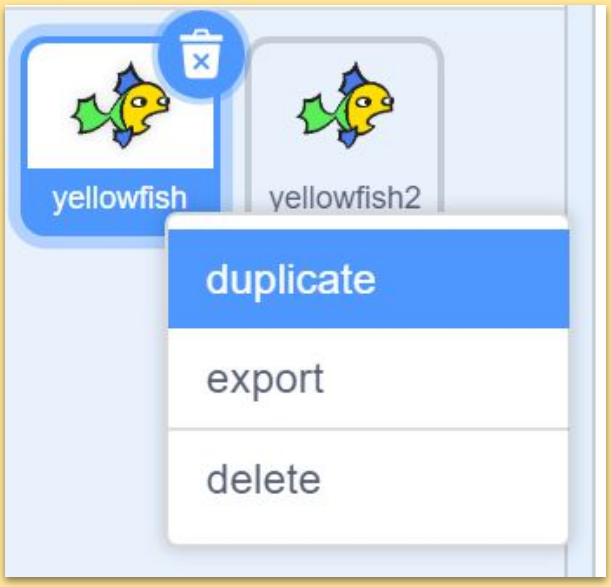
3





Stage 6: Adding More Fish (cntd)

4 **Once you've finished the code** on the previous page, **right click** on the fish to duplicate it a few times. Rename the duplicates as **yellowfish2, yellowfish3** and **yellowfish4.**



5 Click the **Green flag** icon to start the program. Fish appear one by one after the program has started, and reappear after having been eaten. The counter also increases by 1 whenever a fish has been eaten.

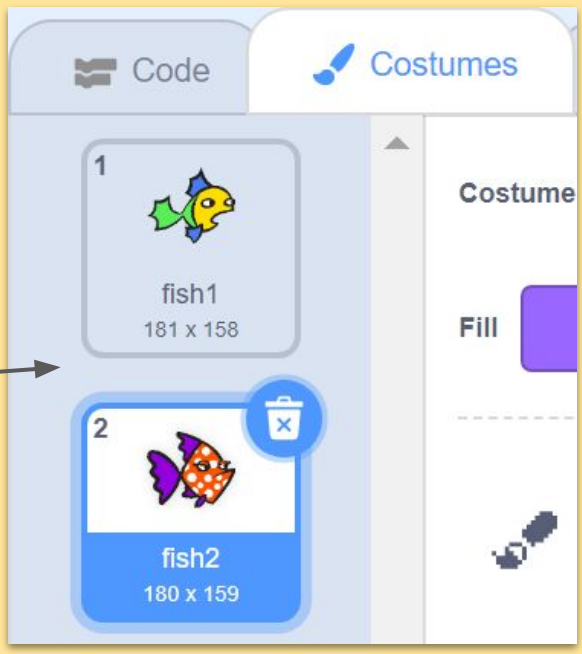
Stage 7: Creating the Redfish Sprite

1 Right-click the yellowfish sprite again to **make a duplicate** of it. Rename the newly created sprite as **redfish**.

2 Select the red fish (fish2) from the costumes tab.

3 Click on the trash can beside **fish1** to delete it.

4 Save your program

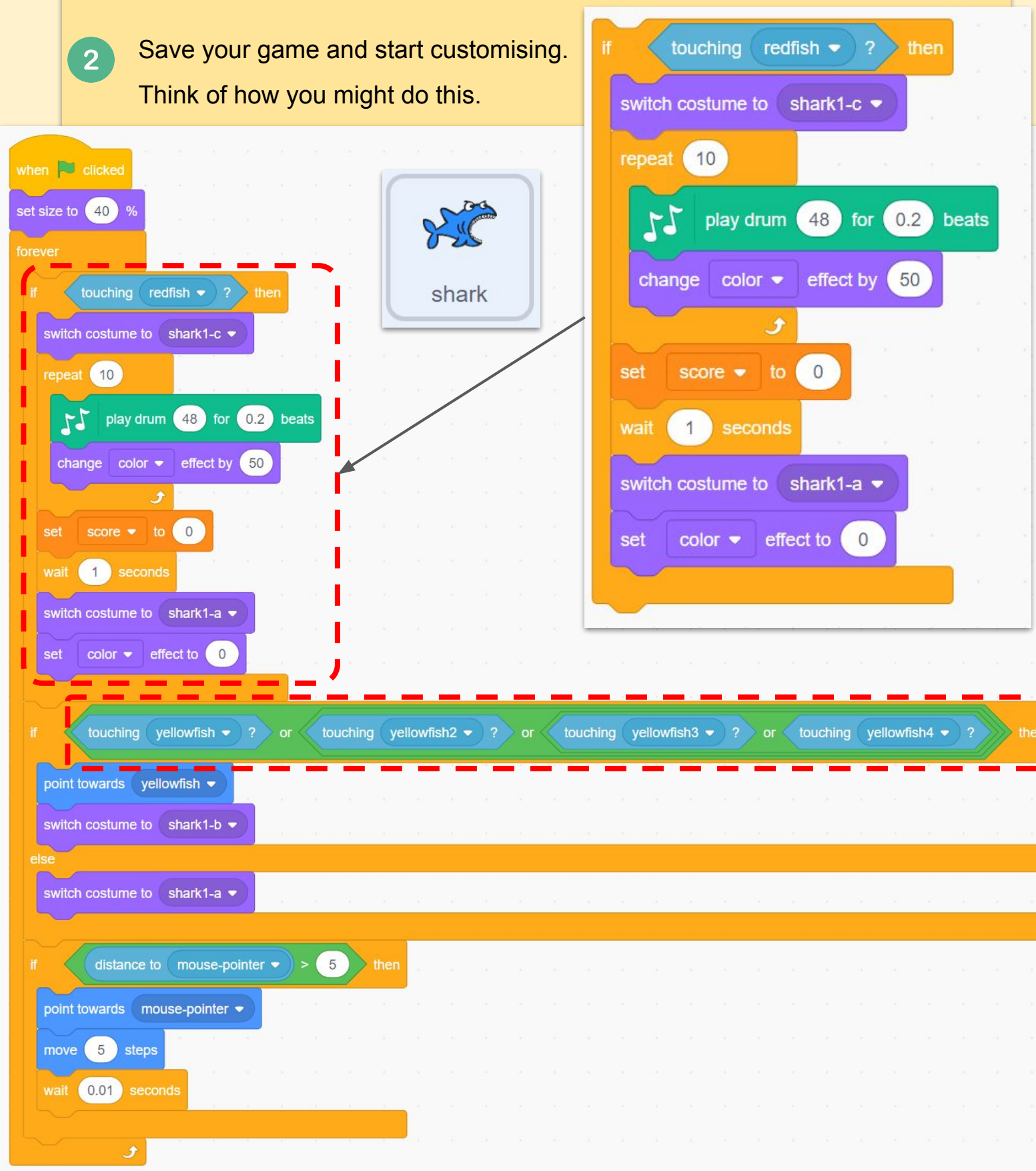


Stage 8: Finishing it off

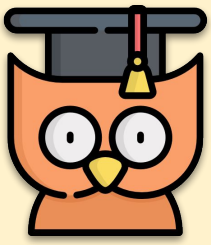
1. Add the **if then block** into the forever block for your Shark

2. Save your game and start customising.

   Think of how you might do this.

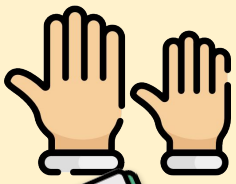


Game 2 - Pong

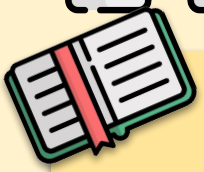


Learning Intentions:

- Making a two player pong style game with a scoring system for both players
- Adding extra, more advanced features to the game



REMEMBER: If you have any questions, stay in your seat and put up your hand. We love to help!



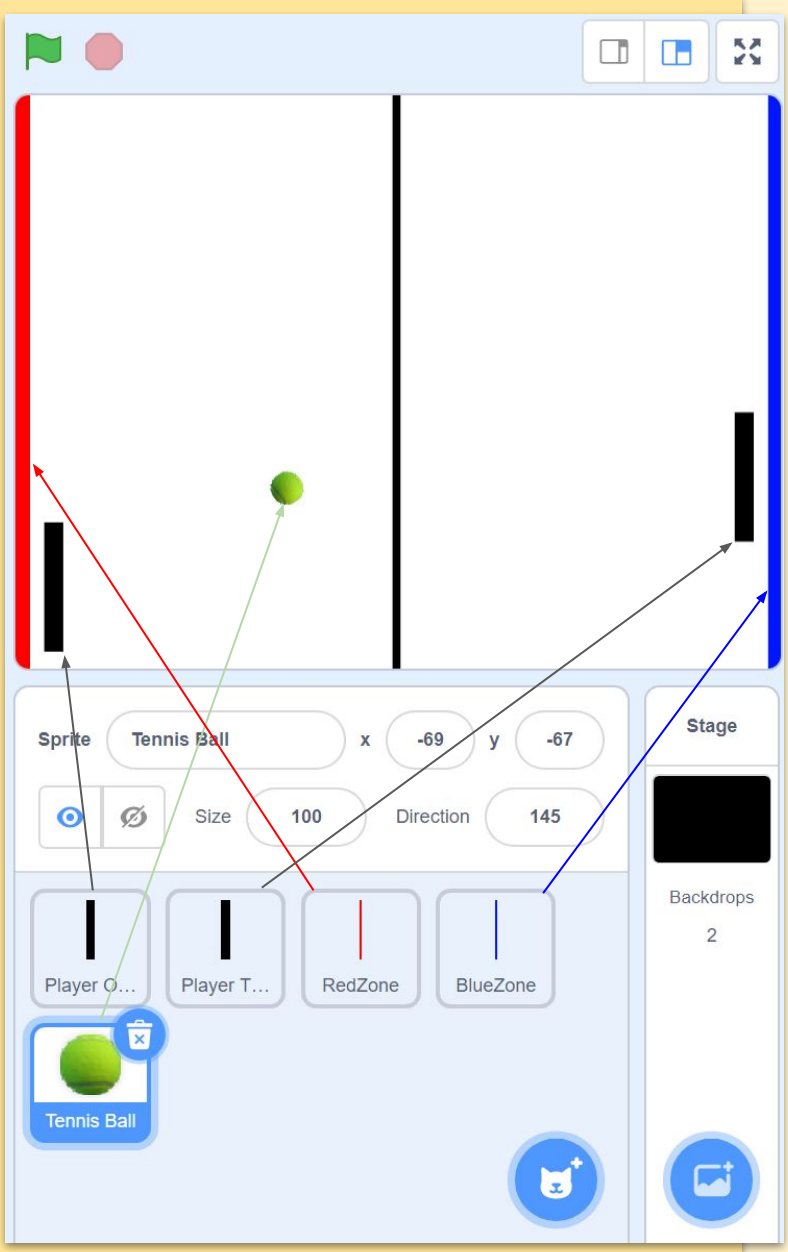
Pong

In Grade 2 we “held your hand” and gave you a lot of the Scratch code. In Grade 3, we’re expecting you to be using Scratch to a high level and to show us that you’re not afraid take it to this level.

Pong is one of the earliest arcade video games. It is a table tennis sports game featuring simple two-dimensional graphics. The game was originally manufactured by Atari, which released it in 1972.

This lesson details how to make the **Basic version** but your tutors will need to see that, once you have finished this, you can take it to the next level. Have a look at our more advanced example here for inspiration.

Let's Get Coding

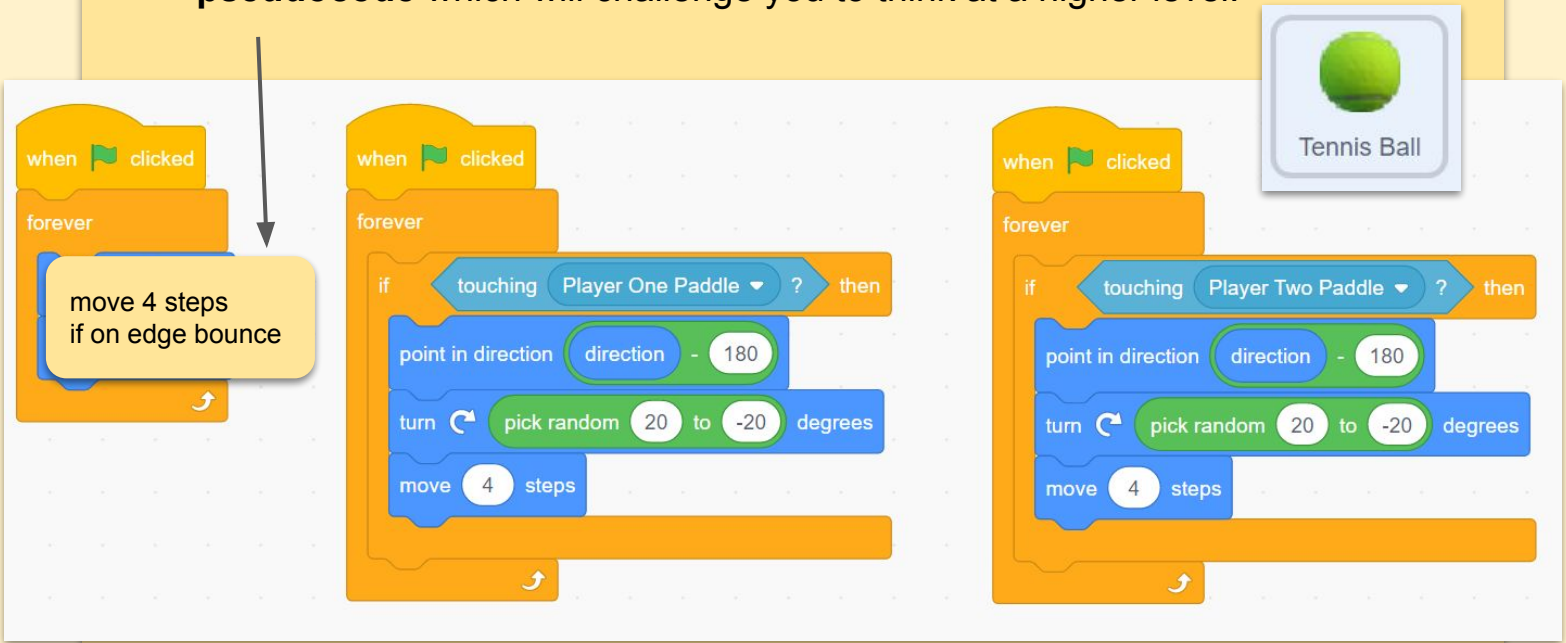


1. **Paint a new Backdrop**

   Create a stage with a line in the middle to represent a net.

2. Create **5 new sprites** as in the example and place them into position. Name them:
   - Player One Paddle
   - Player Two Paddle
   - RedZone
   - BlueZone
   - Tennis Ball

3. Add the following code for the ball. It contains **pseudocode** which will challenge you to think at a higher level.



More Code

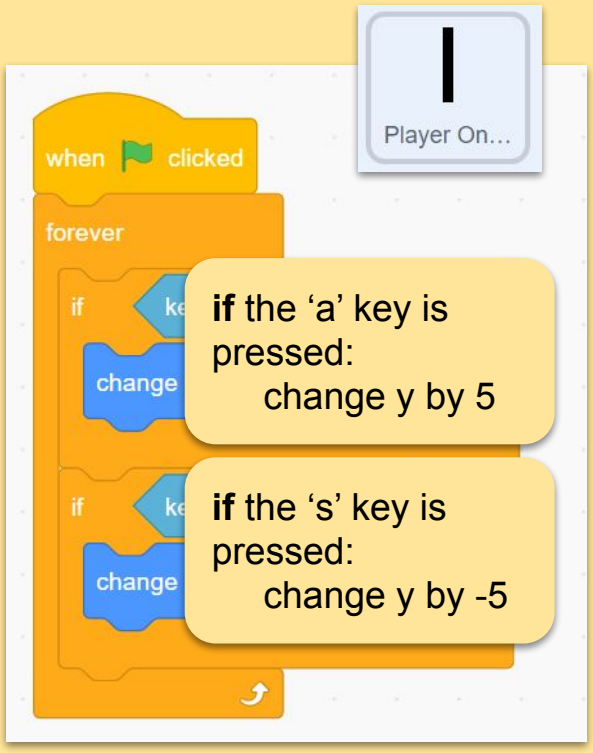
1. Now it's time to add the instructions for the paddles.

   Use the **'a'** and **'z'** keys to control *Player One Paddle* and the **UP** and **DOWN** arrows to control *Player Two Paddle*. This means that two people can play using the same keyboard. You can choose to use different keys if you wish.

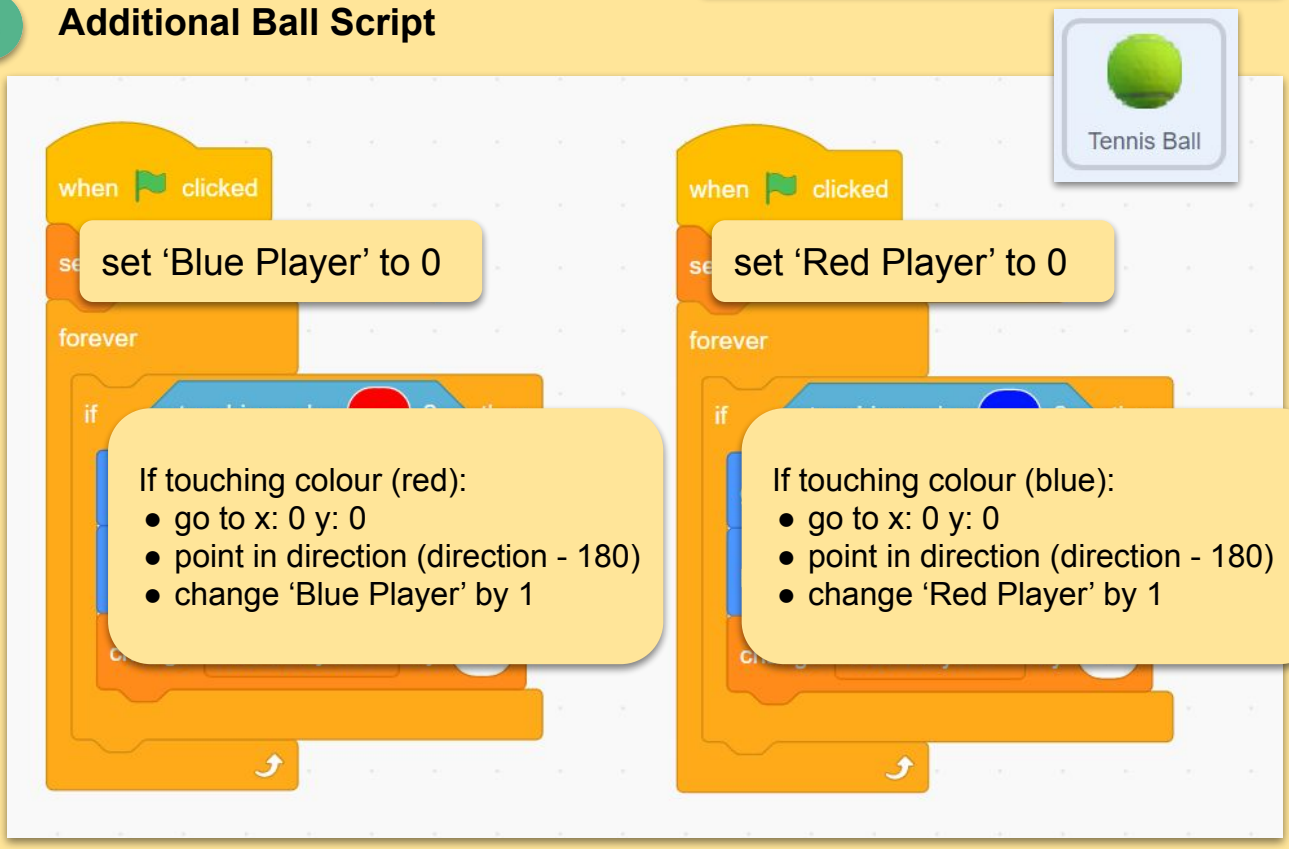
2. **Player Paddle Scripts**

   Complete the script on the right and complete the code for the other player

3. Once complete for both Player One and Two, test the game with a partner.



4. **Additional Ball Script**



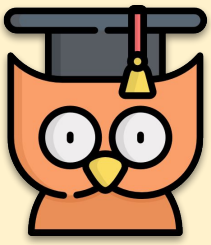
Customising the Game

In Grade 2, we “held your hand” and gave you a lot of the code. In Grade 3, we’re expecting you to be using Scratch to a high level and to show us that you’re not afraid take it to this level. There are hundreds of versions of this game on Scratch. Have a look and see other peoples’ projects to get ideas for your game.

Think about:

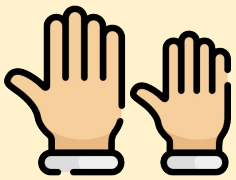
- How could you improve your game?
- What alterations have you made already?
- How would you make the game more user-friendly?
- How would you make the game appeal more to adults?
- What did you find easy about making the game?
- What did you find challenging about making the game?
- What advice would you give to a student just starting to make the game for the first time?

Game 3 - Snow Skater

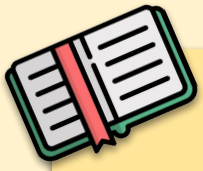


Learning Outcomes:

- Making snow skater
- Bringing the basic template to the next level
- Getting a time and score variable



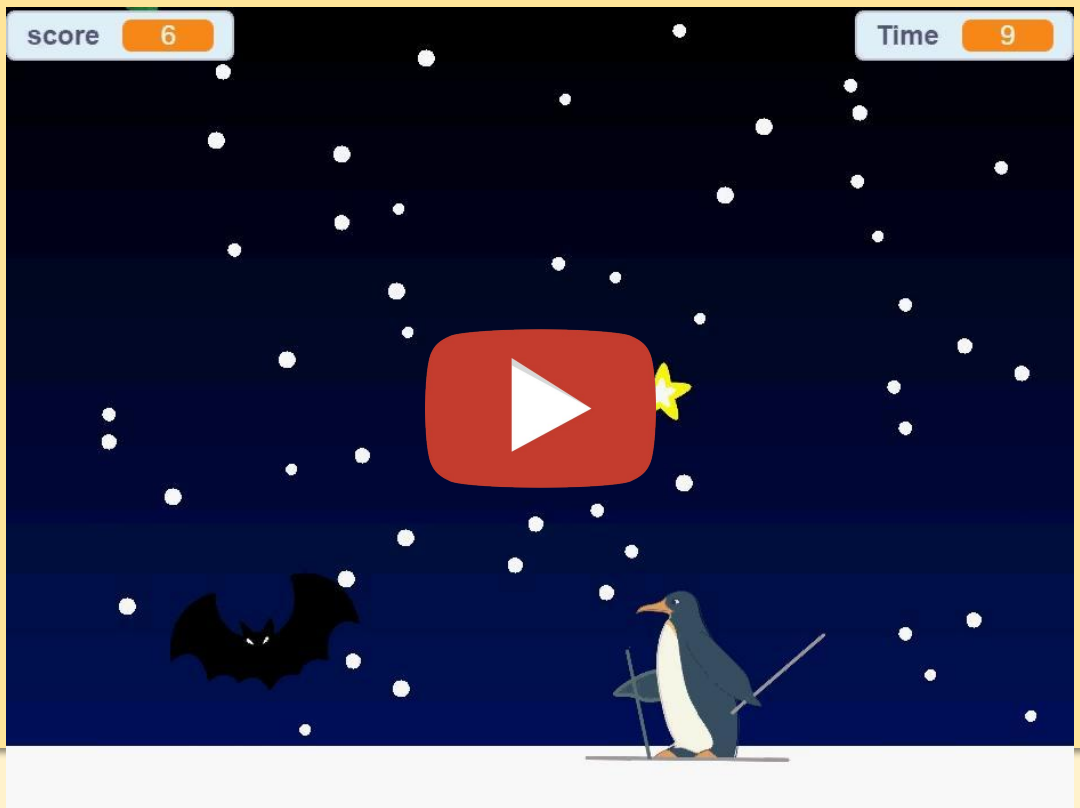
REMEMBER: If you have any questions, stay in your seat and put up your hand. We love to help!



Snow Skater

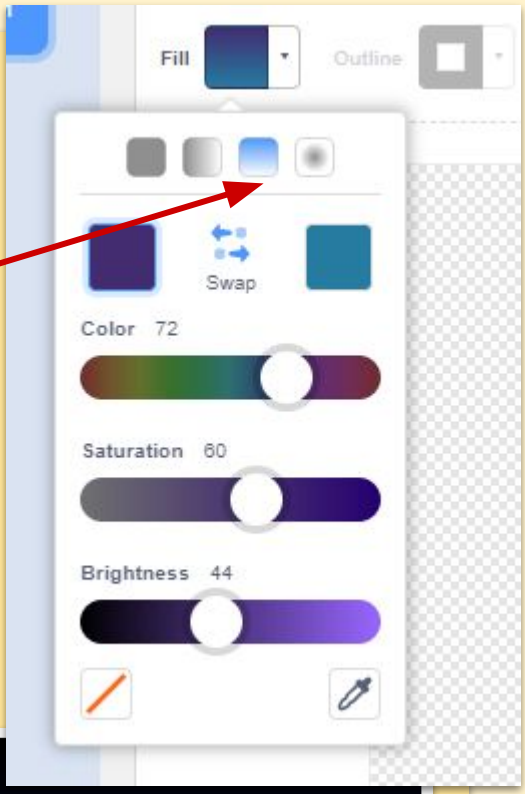
In this lesson we're going to make a game which will use **keyboard input**. We will also introduce the concept of **cloning**.

- *How many sprites can you see in this?*
- *How many variables can you see?*
- *How can you bring this game to the "next level"?*



Making the Backdrop

1. Select the Backdrops tab.

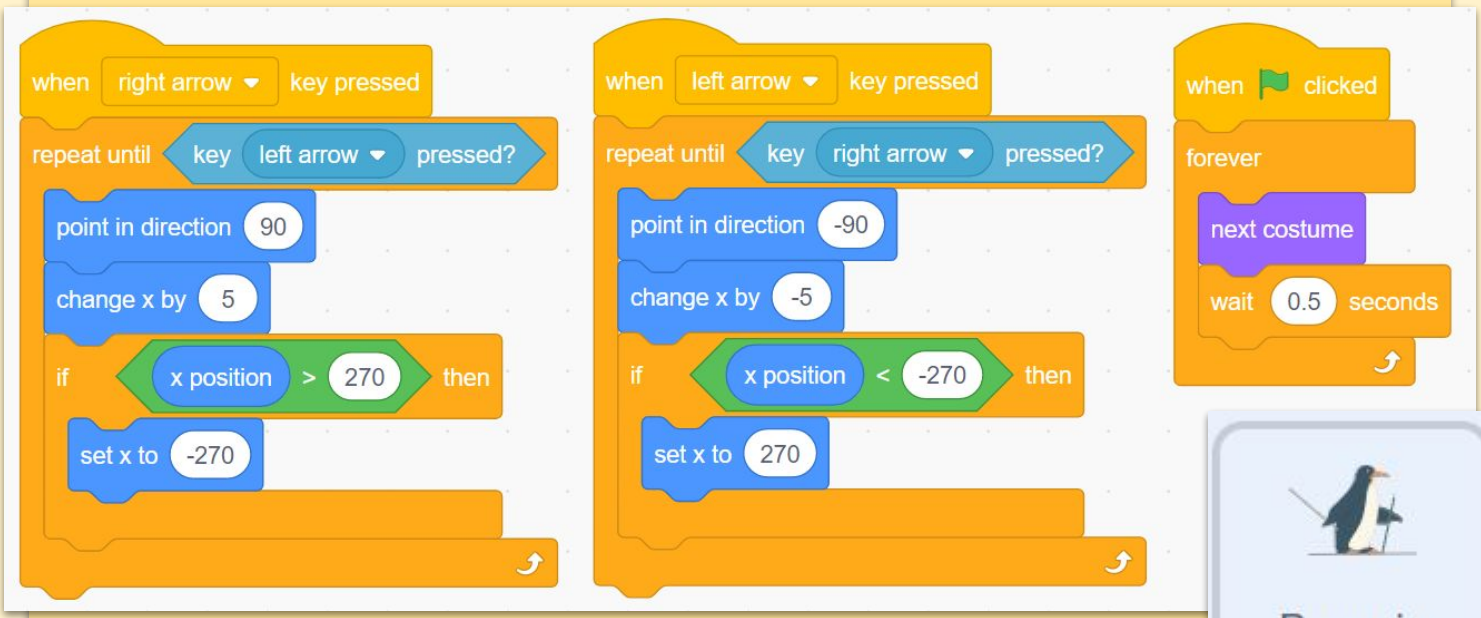


2. Select the Fill tool and choose a background colour. To make things more interesting, you can blend two colours using the **gradient tool**. Choose two different colours, black and blue.

3. Now using the fill or **rectangle tool**, **draw a white ice rink** for your Player to skate on.

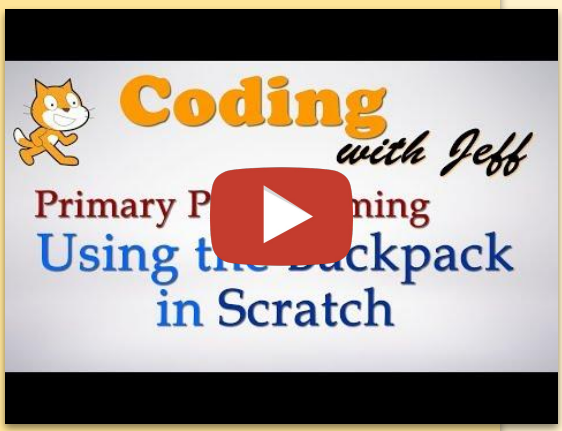
4. Make a **variable called 'time'**. Put the following code on the **stage** (although it could go anywhere and would work the same)

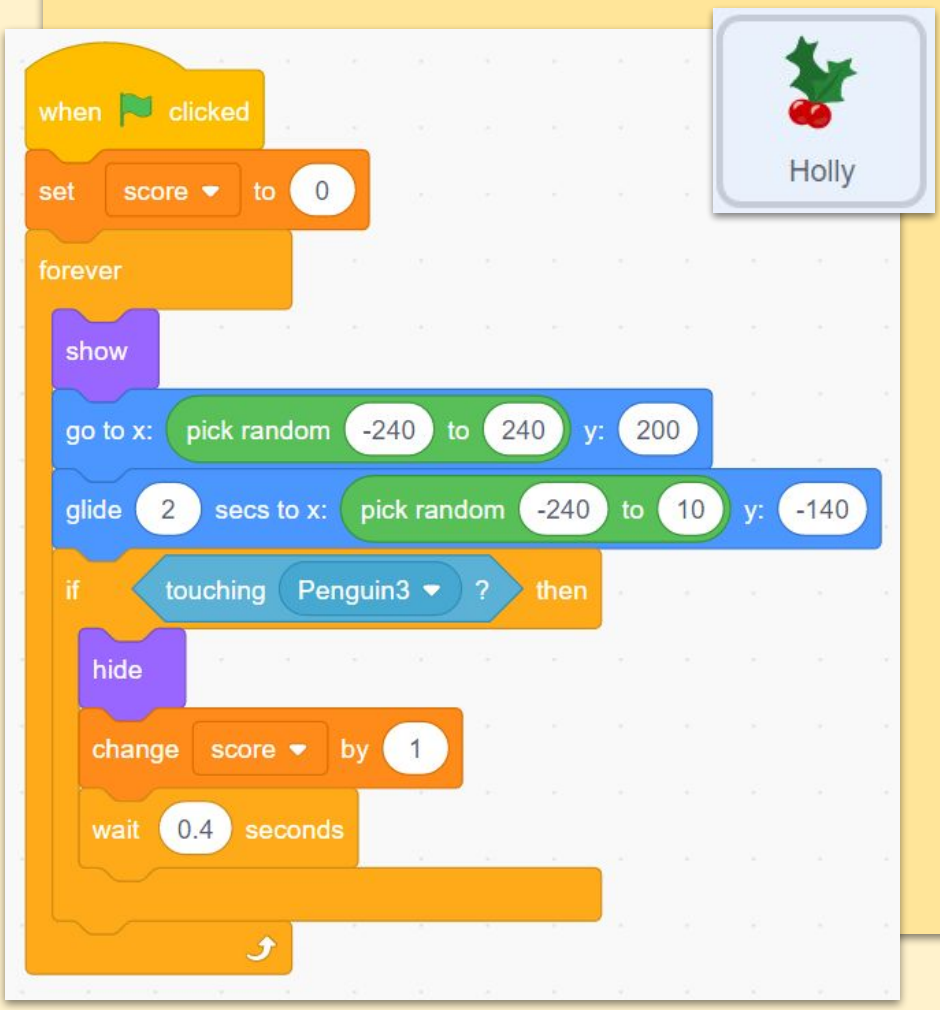
The Penguin's Code



Coding the Goals (Stars/Holly)

Have you seen a code like this before?
Which game used a very similar code to this?



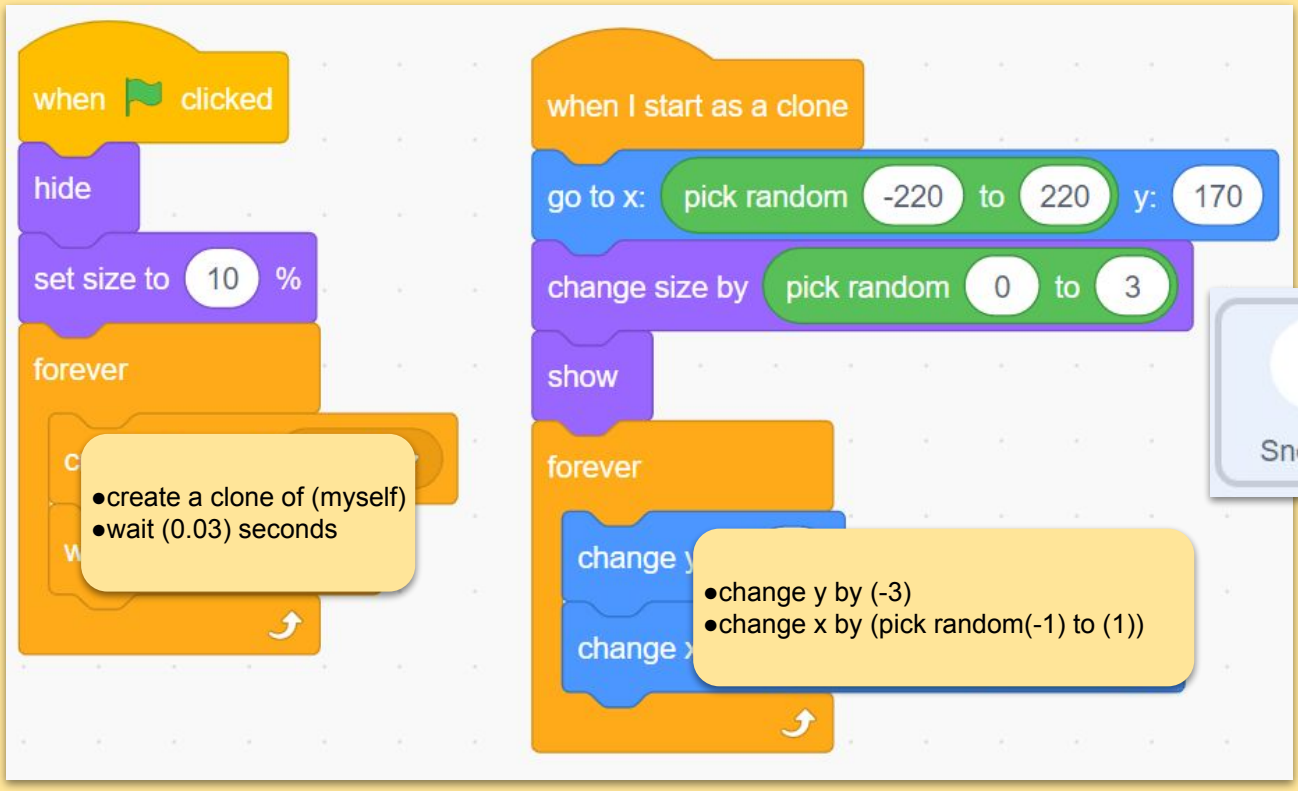


If you've found out which one, you can take the code from *that* game and put it in your **backpack**.

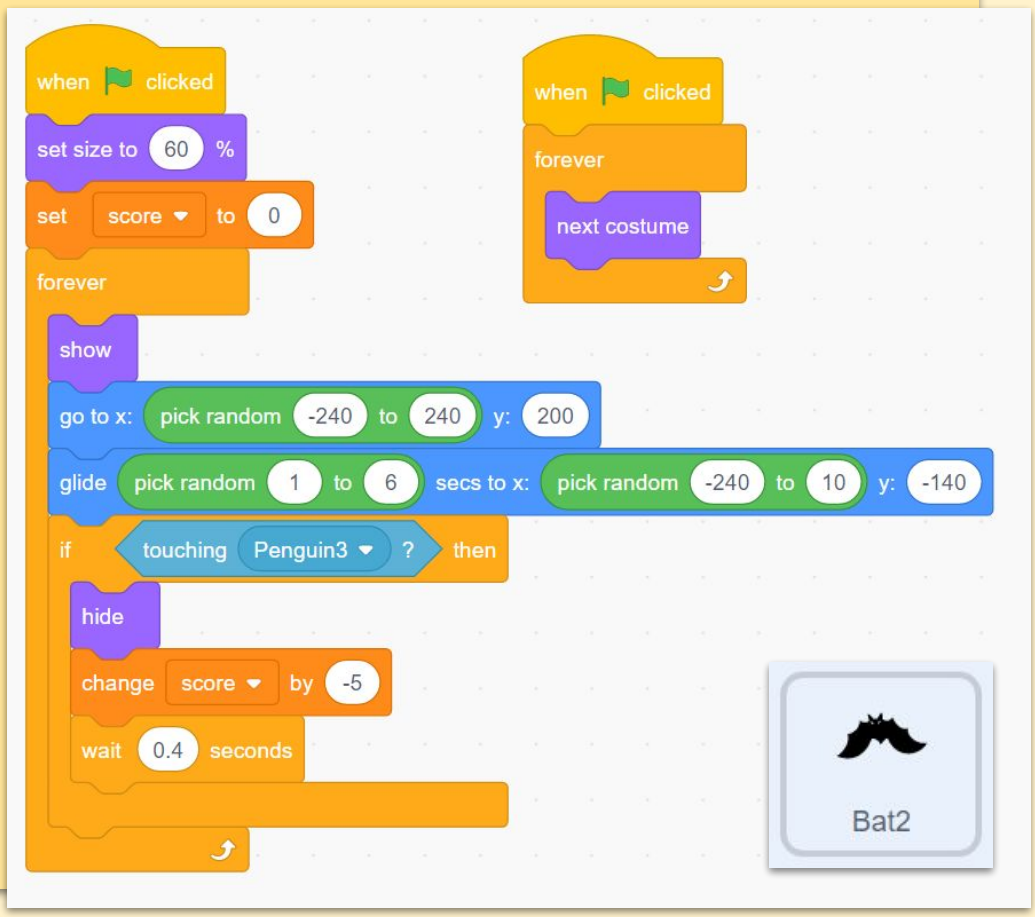
You'll need to make a **score variable**.

Cloning

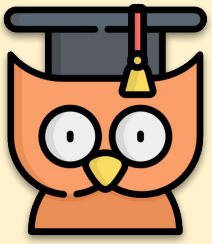
Copy this code to our **snowflake**.



Taking the Holy/Star code from your **backpack**, put it on a bat/enemy but change it so it looks like the code here.

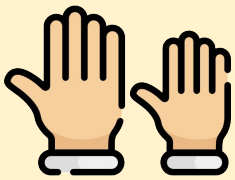


Game 4 - ‘Basic’ Maze Game



Learning Outcomes:

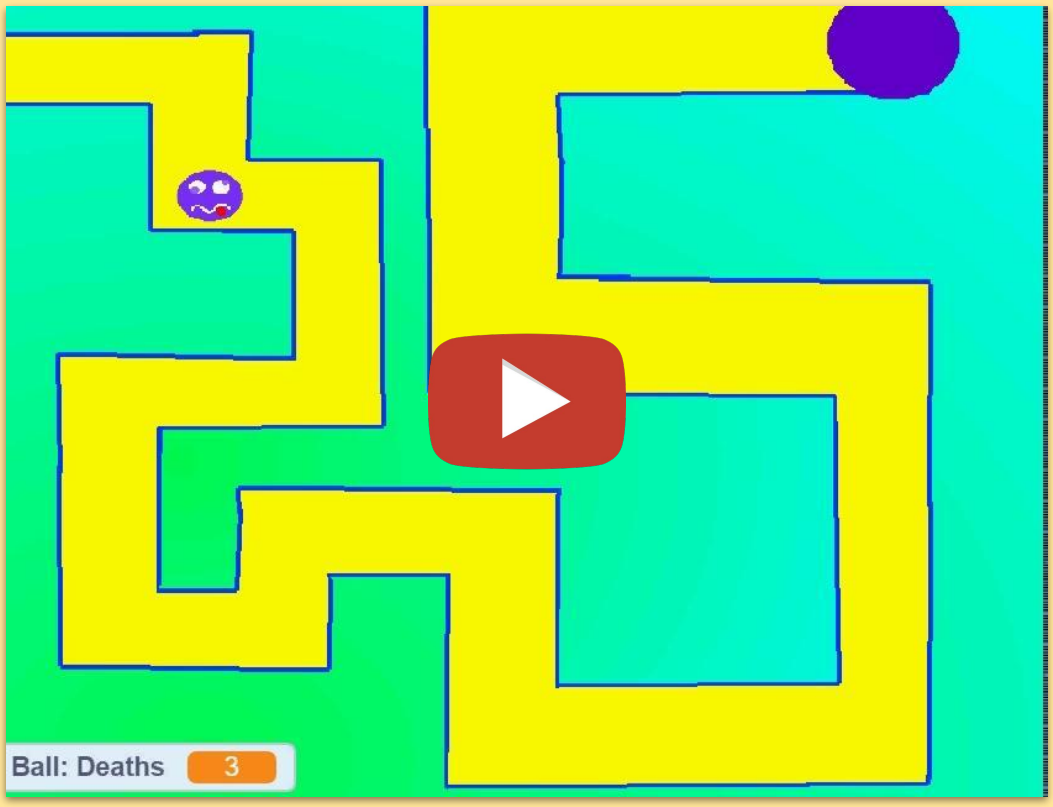
- To make a basic maze game based on a simple template
- Making multiple levels
- Using the ‘touching colour’ code block
- Having intro, direction and multiple level backdrops



REMEMBER: If you have any questions, stay in your seat and put up your hand. We love to help!

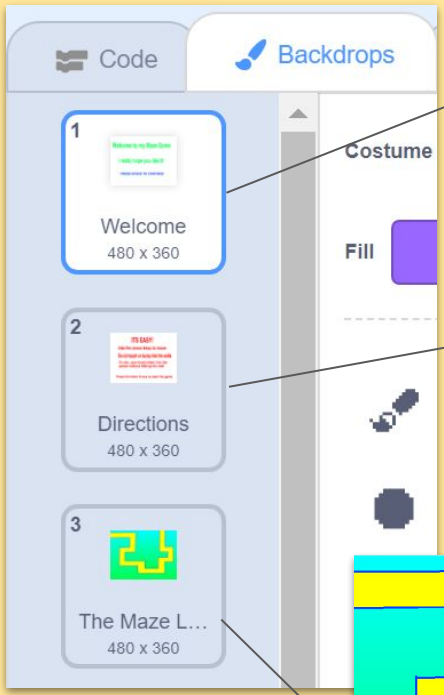
The Final Product

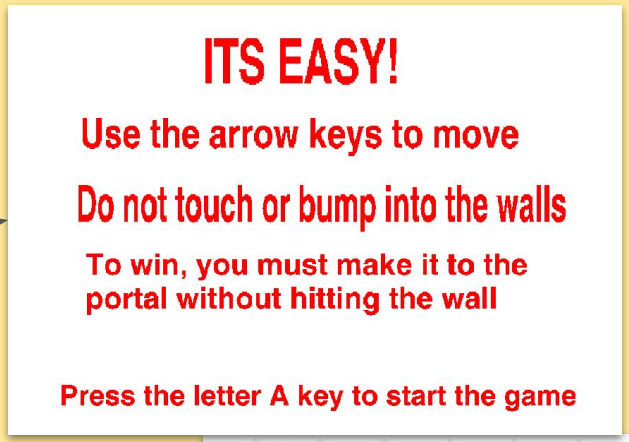
Our plan for this was to make a **Scary maze game** where the player would direct the sprite around a maze for a few levels leading to a scary picture at the end, but you can chose to follow your own path! As with any good game, start it out easy and have it get harder as the game progresses. **AND REMEMBER:** Make sure your game **works**, that it is **completable** and that it is **bug-free**, or else nobody will want to play it!

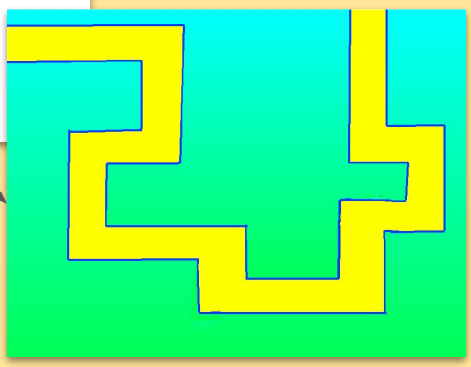


Making an Intro Screen and Starting Maze

1 Make a *Welcome*, *Directions* screen and Level one of your maze. Your character will be controlled by the arrow keys and must be small enough to actually fit through your maze.



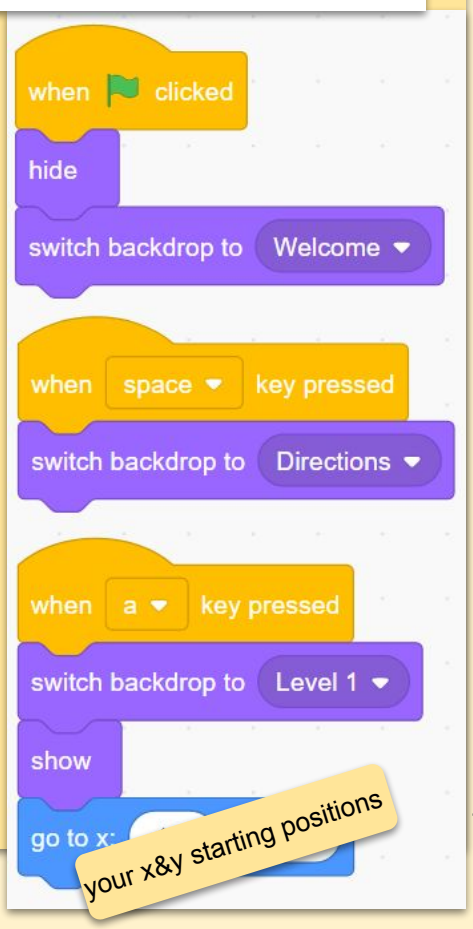




2 **Pick or Design a Player Sprite**

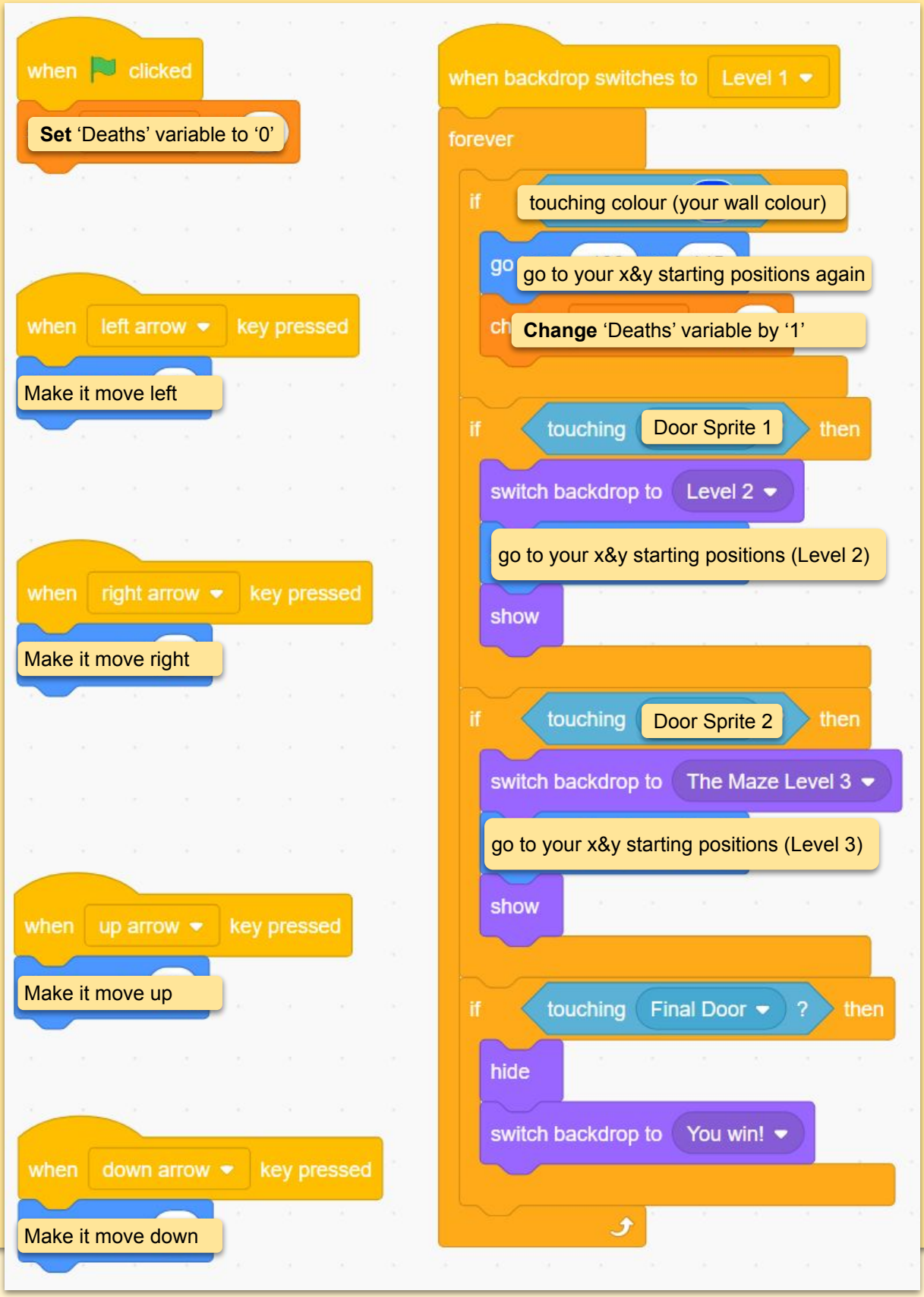
Once you've picked/ designed a player, add the code on the right

NB!!! The numbers in '**go to x: ( ) y: ( )**' will be the part of your maze where the sprite starts from!



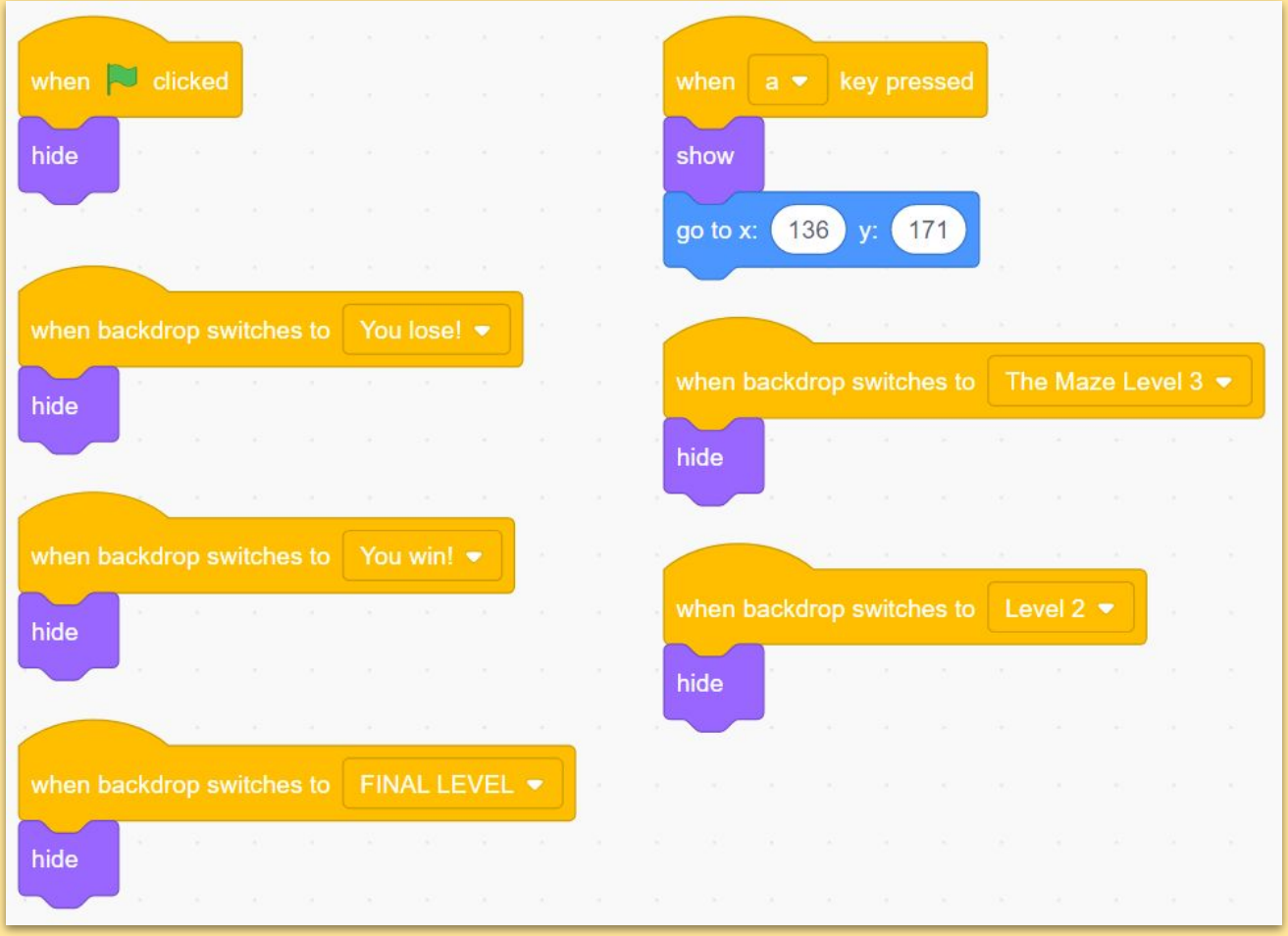
Getting things moving - More Player Code

We need to get the **player**'s sprite moving but also **set conditions** on what happens if it hits a wall or a portal (to go to the next level). Try and figure out the pseudocode below on your player sprite to get things moving.



Showing and hiding different elements

Below is a sample code for a **door/portal** which will appear at x:136 y:171 in Level One **(your numbers will be different)**. It will only be displayed in Level One as, in our example, when you press the a key, level one begins.

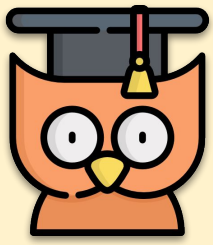


Before you say “I’m finished”

Have you:

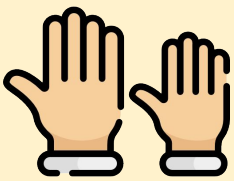
- Added in more levels
- Added in a timer
- Considered having something for your player to collect for points
- Considered having some moving walls (the same colour as the walls) that reset our player sprite
- Added in a ‘next level’ or a ‘You lose’ screen
- Sanity-tested and debugged all levels and featured.

Game 5 - ‘Advanced’ Maze Game

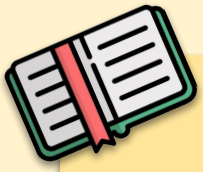


Learning Outcomes:

- To make a more advanced maze game based on a template
- Sending **Broadcast Messages**
- Having **Best Time, High Score** and **timer** features.

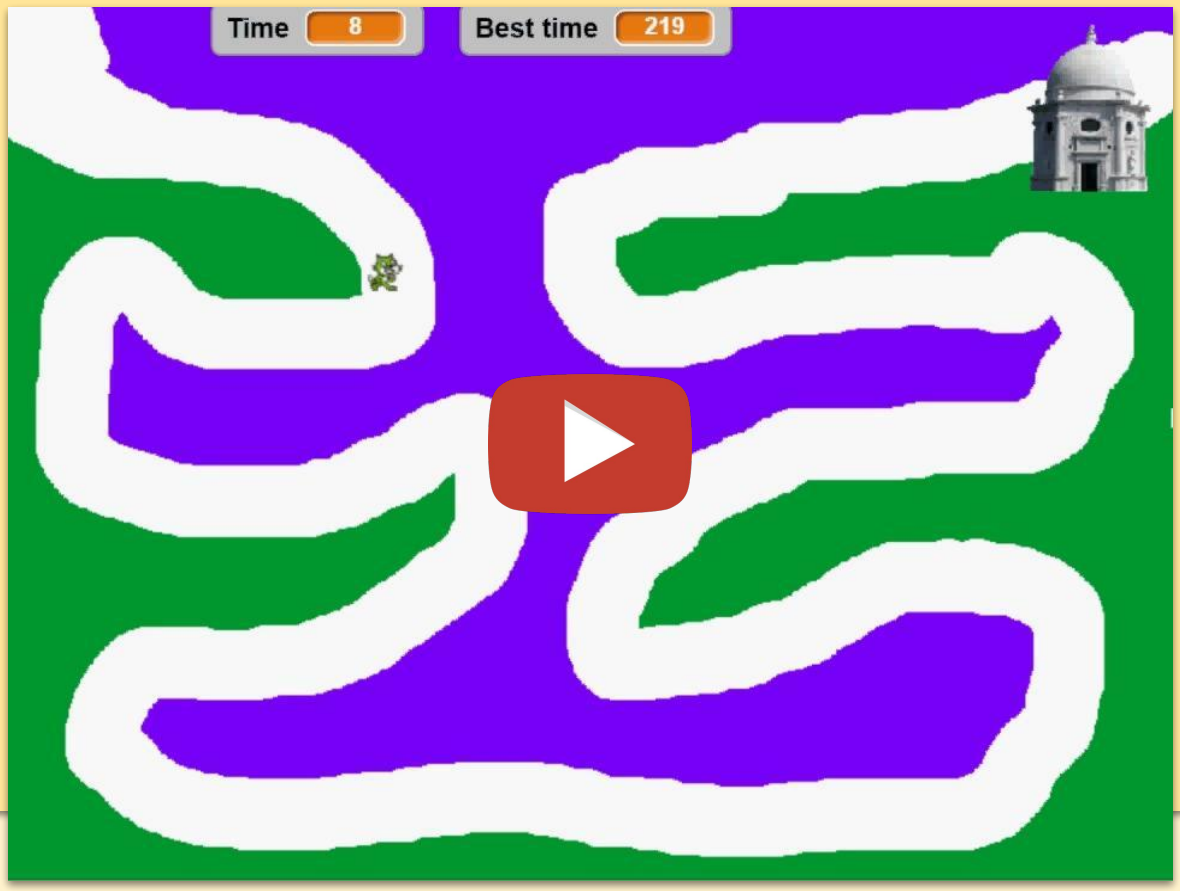


REMEMBER: If you have any questions, stay in your seat and put up your hand. We love to help!



The Final Product

We’re going to make an advanced maze game**,** a game which has a player controlled sprite and a goal. It also has a timer and a best time log. Like Grade 2’s **Hungry Shark**, it uses up and down arrows to control the **y axis** but also left and right arrows to control the **x axis.** The goal is to get the player sprite home in the quickest time possible.



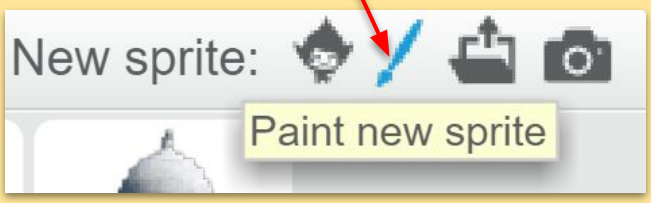
Stage 1: Drawing the Maze

1. Name your project (For example: *Home Runner*)

2. **Choose** and **rename** your sprite.

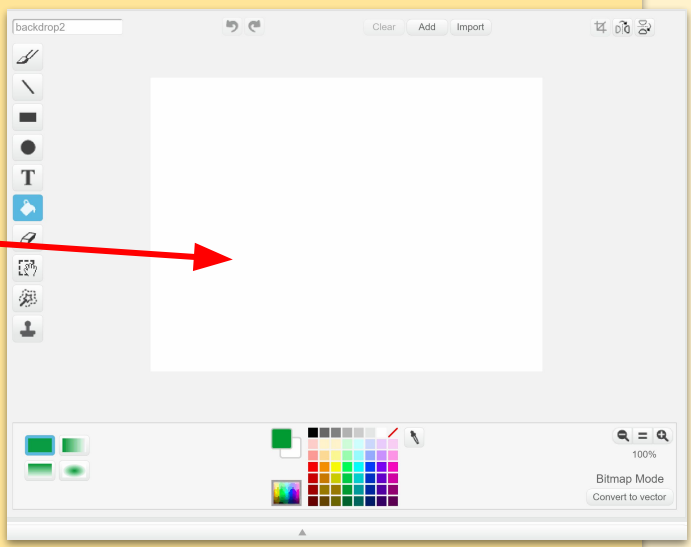
Unlike the others games we made, **DON'T** load a background image. Our background will actually be a custom sprite that doesn't move and takes up the entire window. Our project needs three sprites in total, **the player, the goal** (home) and **the map.**

3. Click on the paintbrush to *Paint a new sprite*



Making our Maze

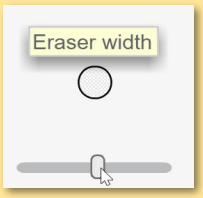
1. Click the  paint can



2. Paint the entire area green.

3. Select the Eraser 

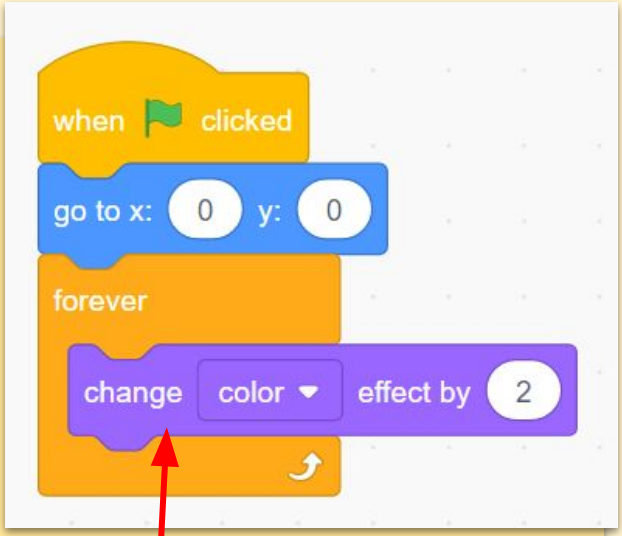
4. Increase the Eraser Width

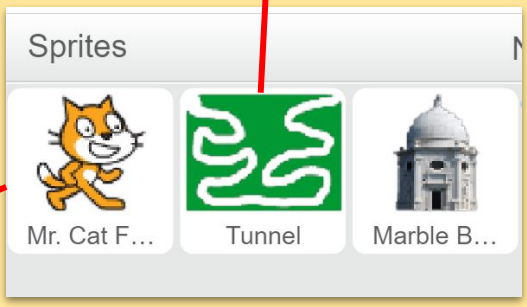


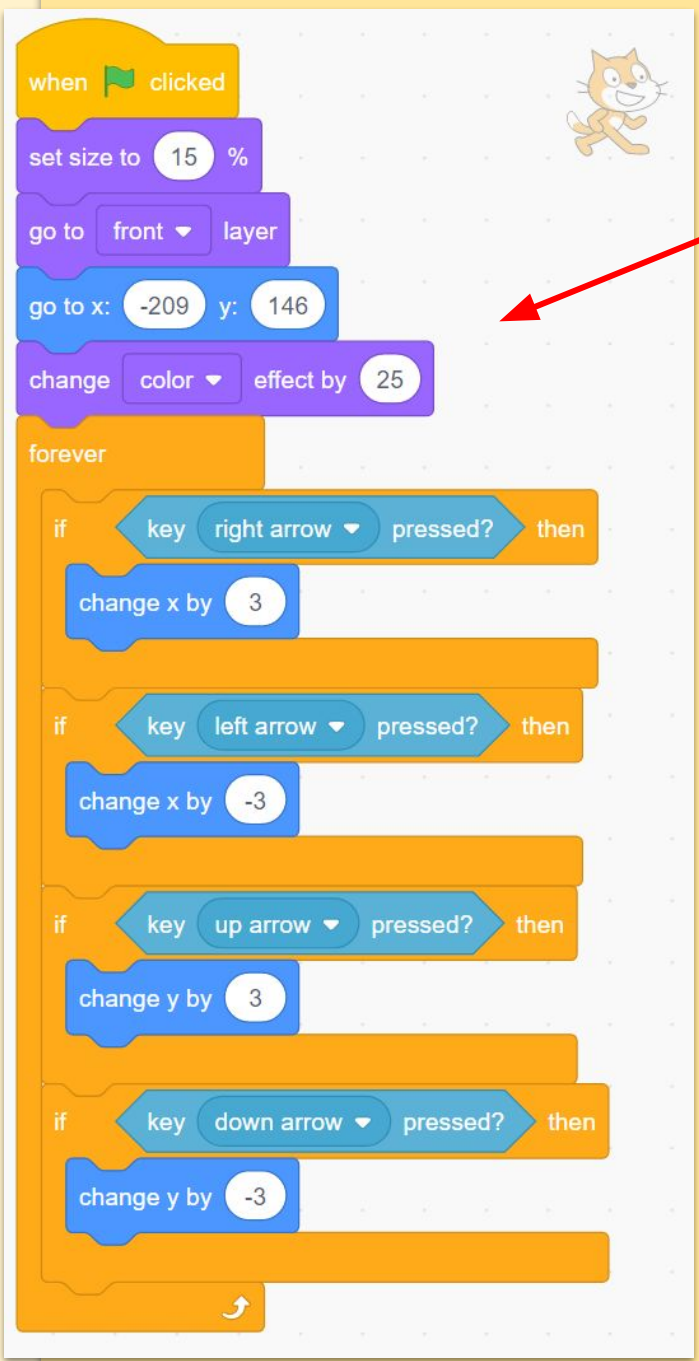
5. Using the **Eraser Tool** carve out a maze wide enough for your sprite to fit through. Your maze should split the window into two sections, grass and water.

Stage 1a: Code for the Maze

Our maze doesn't actually need any code to function properly, but if you like, this code will change the colour in a psychedelic way.







Stage 2: Coding the Player (cat)

The code on the left is the starting code to get the cat moving. Make sure and help those around you if you get it finished early.

Pair Programme

Help a Friend,
Make a Friend!

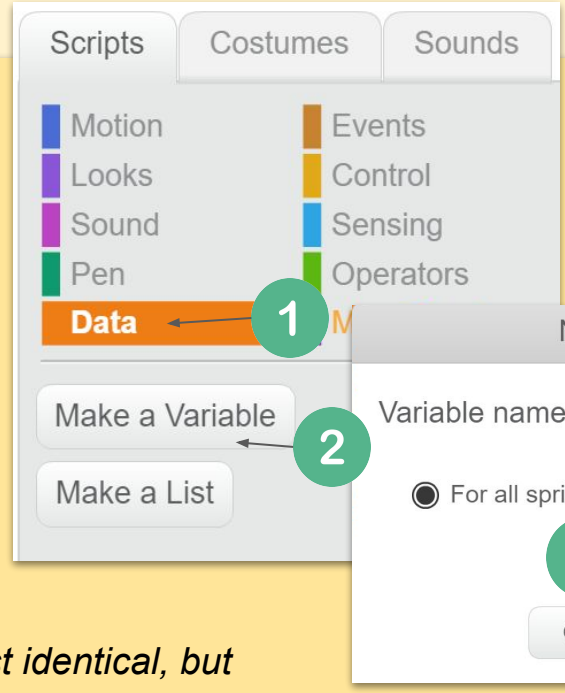
Stage 2: More Player Code - Crack the Pseudocode

See if you can crack the pseudocode on the right . We've left some of the block colours in to make it easier for you.

Pair Programme

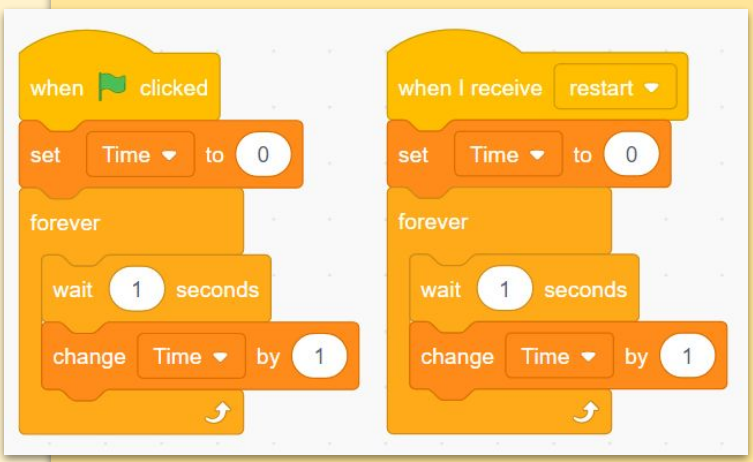
Help a Friend,
Make a Friend!

Stage 3a: Time



Your maze game needs a **time variable.** We need to make a variable called time but also to code this into our game.

Add these code blocks to the **player** script. *What will they do? They're almost identical, but why do we need both of these?*



Expert Tip

A **broadcast** is a message that is sent through the Scratch programme. It allows sprites to 'talk' to each other. One sprite will broadcast a set of commands and another will receive them.

Stage 3b: Keeping Track of Time and Best Player

We're going to make this game more competitive by adding a **Best Time** feature.

Make a Variable called *Best Time* and one called *Best Player* and drag them next to the *Time* display on the stage. Then make the following code. *But what does it do?*