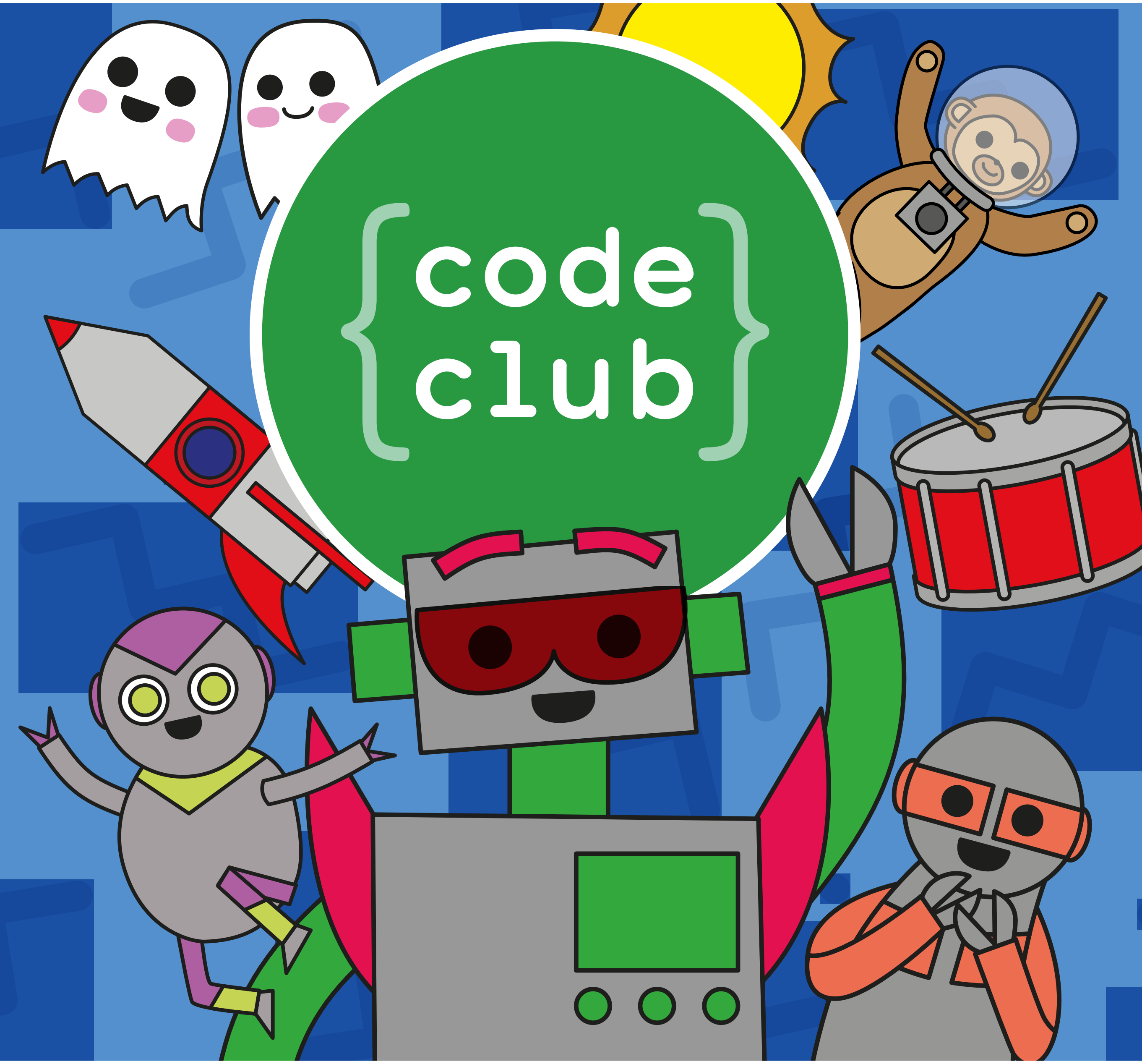
Simple coding for total beginners



Book of Scratch

Volume 1

{code club}
when clicked
say I can code!
I can code!
www.codeclub.org
{CODE CLUB}

Raspberry Pi
PRESS

First published in 2018 by Raspberry Pi Trading Ltd, Station Road, Cambridge, CB1 2JH

Writers: Rik Cross, Tracy Gardner
Illustrator: Timothy Winchester • Design: Critical Media
Editor: Phil King • Sub Editor: Nicola King
Publisher: Russell Barnes • CEO: Eben Upton
Projects tested by: Alexander King & the Code Club community

ISBN: 978-1-912047-67-3
Printed in China

The publisher, and contributors accept no responsibility in respect of any omissions, errors or issues relating to goods, software, viruses, or exposure to harmful web content on websites other than its own.

Contents

Learn to code with Scratch and Code Club!

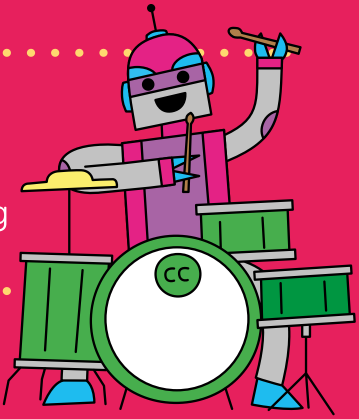
6 **Welcome**
Welcome to the first ever Code Club book

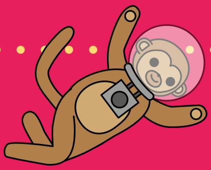
10 **Introducing Scratch**
Discover how to use Scratch and start coding

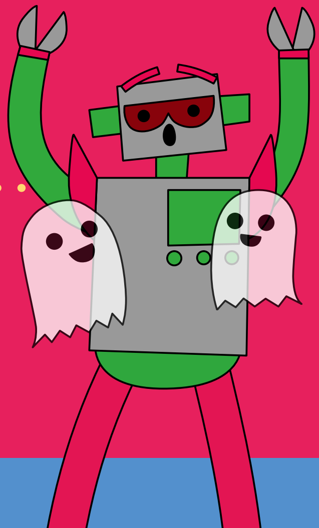
24 **Rock Band**
Make music with your first fun coding project

38 **Lost in Space**
Create an animation that's out of this world

50 **Ghost Catcher**
Build your own spooky ghost-catching game







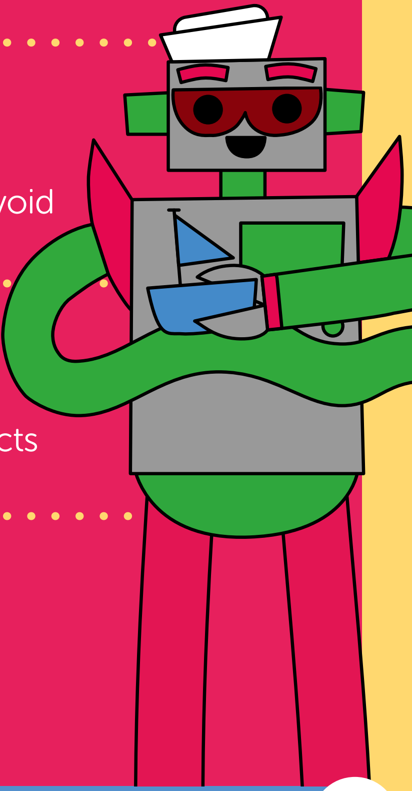
62 **Chatbot**
Code your own talking character to chat to

74 **On Target**
Learn how co-ordinates work with this fun game

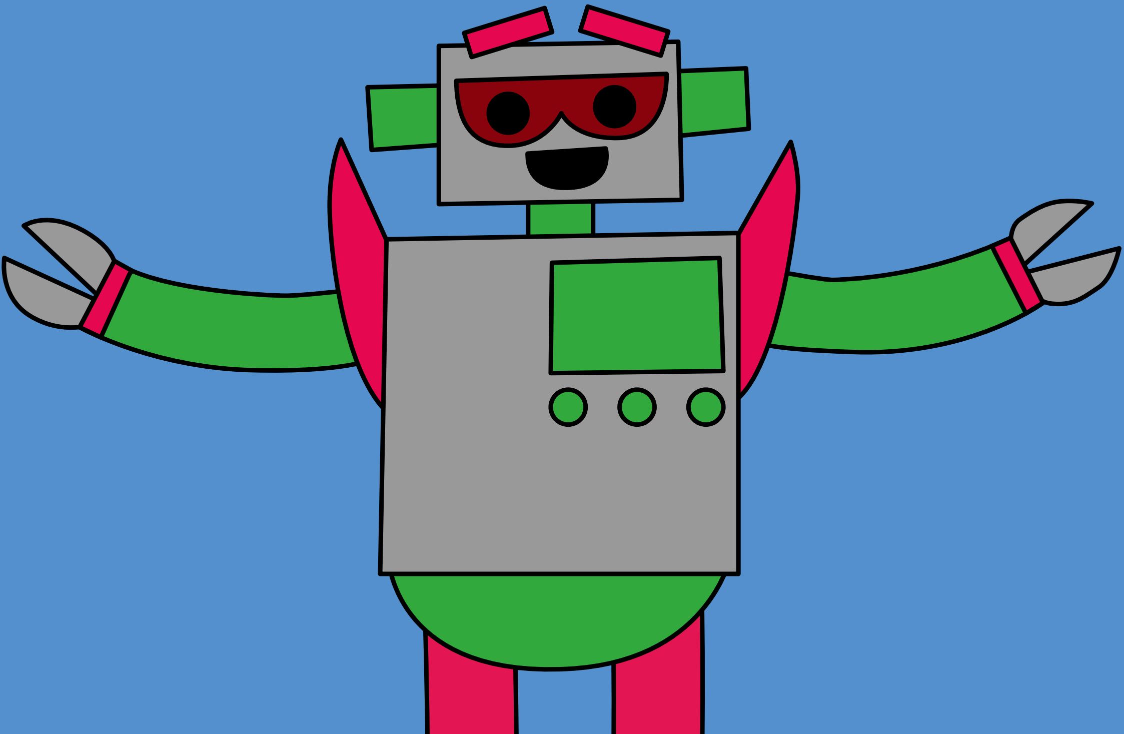
92 **Boat Race**
Create a cool racing game with obstacles to avoid

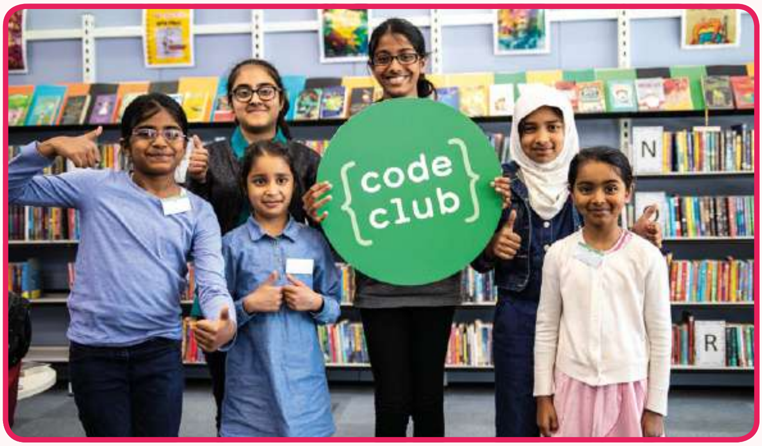
104 **Useful Code**
Handy code snippets to use in your own projects

110 **Puzzle Answers**
Check your answers here – no cheating now!



Welcome to the First Ever Code Club Book!





Code Club is a movement of free, fun computing clubs that meet in over 150 countries all over the world. At Code Club, hundreds of thousands of young people – just like you – learn how to create with technology and have made their own games, animations, websites, and more.

To get a computer to do things you want it to, you need to give it instructions in a language the computer understands. Creating those instructions is called coding or programming.

In this book we show you how to use a programming language called Scratch, which uses blocks to tell the computer what to do. Each block contains an instruction that the computer understands. You put blocks together to make your program. Simple.

Programming in Scratch is a great way to learn how to code. It's also really creative. You can create your own characters and backgrounds to make your project unique. You can remix and change existing projects. For example, you can make a game more difficult by speeding things up, or easier by slowing things down. The possibilities are limitless.

In each chapter you'll find instructions for building a cool project with Scratch. Our friendly Code Club robot will guide you through and give you some handy tips. There are tick-boxes to help you keep track of your progress (we

love tick-boxes) and you can give yourself a big pat on the back when you finish each project.

We've also included lots of challenges for you to change and personalise your projects and plenty of ideas to inspire you to create something new using the computing skills that you learn.

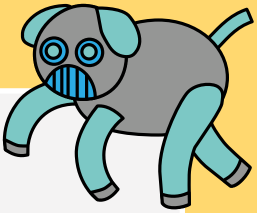
Coding can be hard and even the world's best computer scientists get stuck sometimes. That's why we've included some special upside-down hints that you can use if you're really stuck. Only to be used in emergencies!

Once you've completed the projects in this book, you can find loads more fun project ideas on our website **rpf.io/ccprojects**.

You could also ask your teacher to set up a Code Club in your school using the letter on the next page. Don't forget to sign it and to complete the blank space we left to tell your teacher why you love coding!

I really hope you enjoy this book and I can't wait to see what you create.

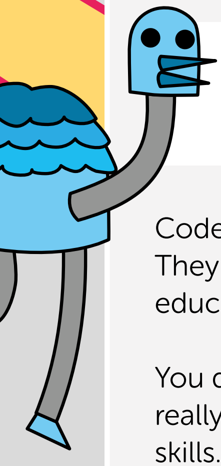
Maria Quevedo
Director of Code Club



Fill out this letter and give it to your teacher
if you would like to start a Code Club in your school.

Dear

I've been learning how to code at home using the Code Club Book of Scratch. I would love to keep coding at a Code Club in our school. I love coding because...



Code Club is a global network of over 12 000 coding clubs for 9 to 13-year-olds. They provide free online projects, training, and resources to help teachers and educators run lunchtime or after-school clubs.

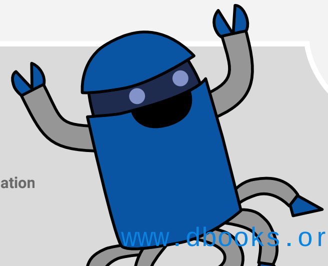
You don't need any coding experience to run a club: Code Club's projects are really easy to follow and help pupils and teachers develop their programming skills. They are really fun and a great starting point for creating awesome games, websites, and animations!

It would be so great to have a Code Club in our school, and I'd be happy to help!

Here's what other teachers say:

> "I started a Code Club to give pupils a chance to try different things, as well as to explore their own ideas. Pupils have a natural love of creativity, technology, and challenge – Code Club ticks all these boxes and has provided me with an excellent platform to embed Computing in a school setting."
>
> Matt Warne, Teacher at RGS The Grange

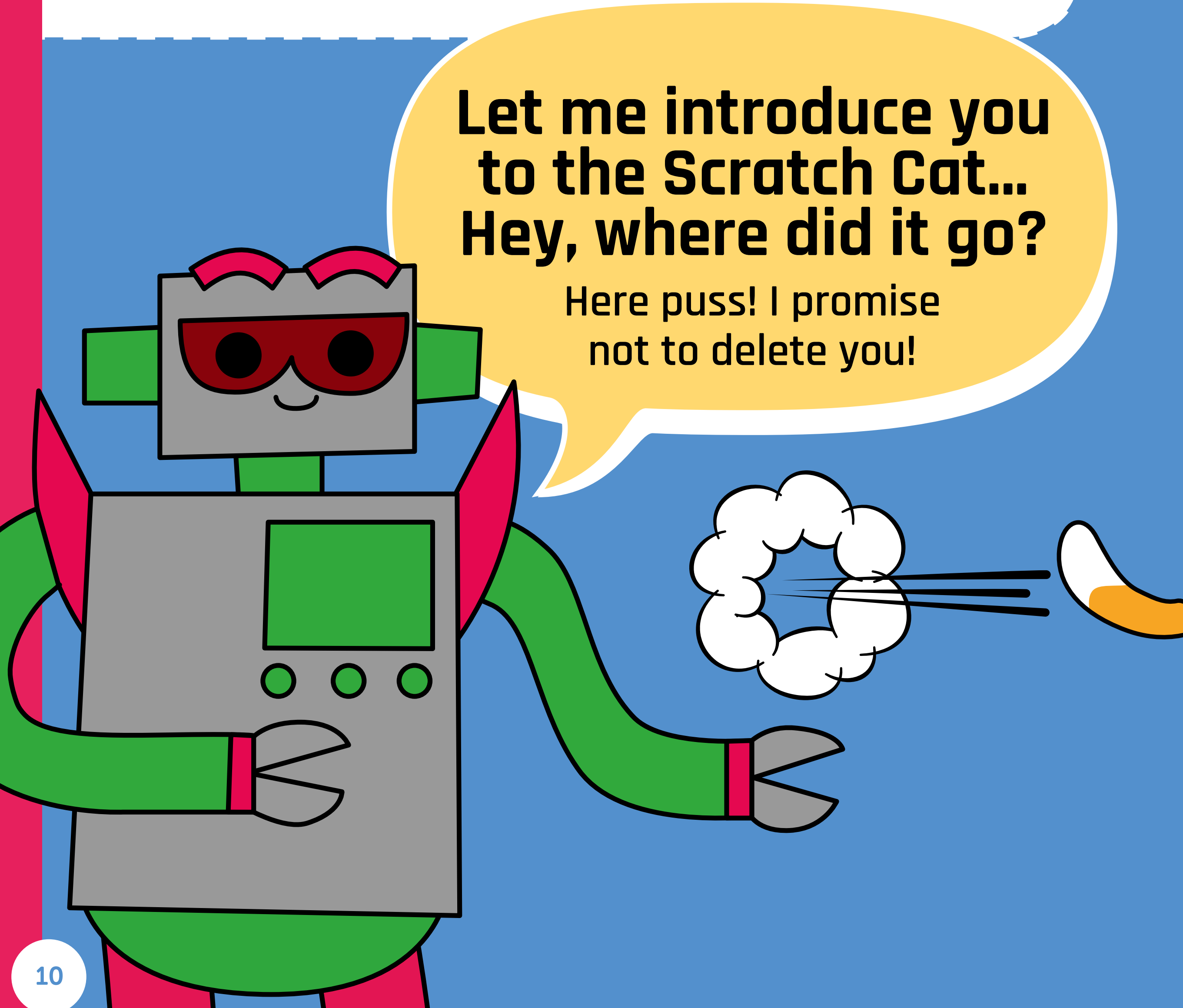
If you'd like to find out more, visit codeclub.org
From,



Code Club is part of The Raspberry Pi Foundation
UK registered charity 1129409

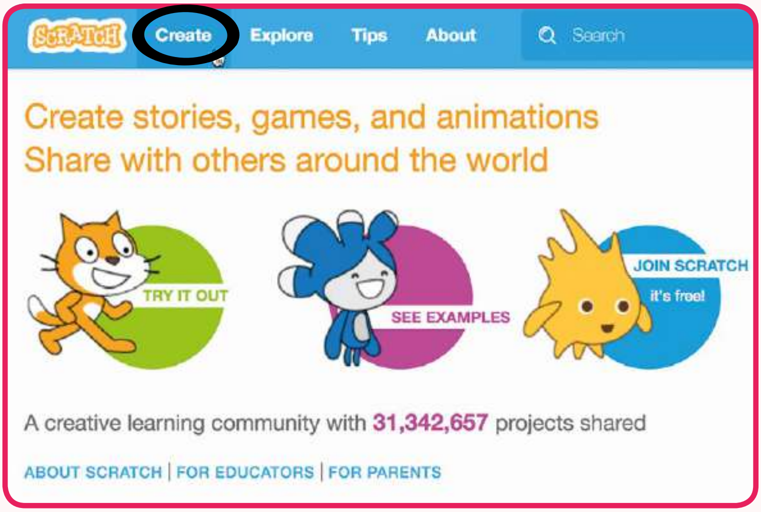
Introducing Scratch

Discover how to navigate Scratch's user interface and website to start coding and sharing projects


Let me introduce you to the Scratch Cat... Hey, where did it go?
Here puss! I promise not to delete you!

Scratch is a programming language that allows you to use code blocks to create animations, stories, musical instruments, games, and much more. It's a bit like programming using Lego!

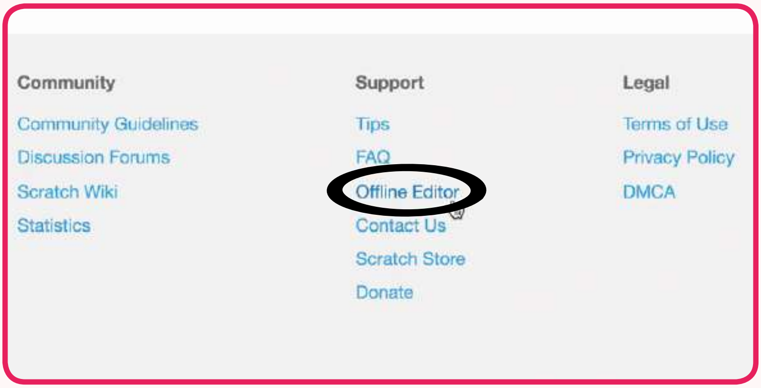
The easiest way to start programming in Scratch is to use the online editor. Visit **scratch.mit.edu** in a browser and click **Create** at the top of the page to get started.



Create stories, games, and animations
Share with others around the world

A creative learning community with 31,342,657 projects shared

There are lots of advantages to working online, but if you prefer to work offline (or don't always have an internet connection), you can click **Offline Editor** at the bottom of the homepage to download Scratch instead.



TIP!

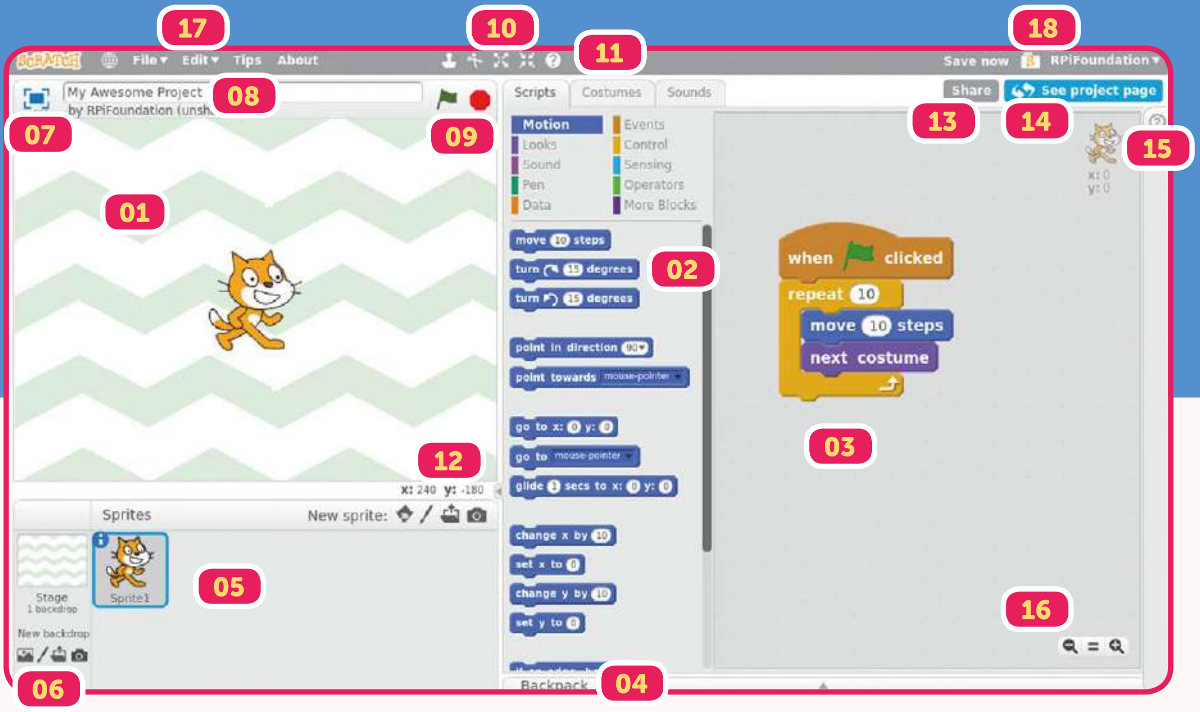
PROJECT FILES

To download a zip file of all the Scratch 2 (.sb2) project assets files for this book, go to:

rpf.io/book-s1-assets

The Editor

Find your way around the Scratch editor...



```
when [flag] clicked
repeat 10
    move 10 steps
    next costume
```

01: STAGE

A project contains 'sprites' which you add code to. Sprites appear on the stage and can be coded to move around, make sounds, and do lots of other things.

02: BLOCKS PALETTE

Code blocks can be used to control your sprites and stage backdrop. All blocks are colour-coded, and can be found in the categories at the top of the blocks palette.

03: SCRIPTS AREA

Drag blocks from the palette to this area and create scripts by clicking them together.

04: BACKPACK

Add scripts to your backpack to use them in other projects.

05: SPRITE LIST

This shows all of the sprites in your project. You can click the blue information icon on any sprite to change its name and how it behaves.

06: BACKDROPS

Change how your stage looks by adding new backdrops.

07: FULL-SCREEN

Make your stage full-screen so that others can see your creation in its full glory.

08: PROJECT NAME

09: START/STOP YOUR PROJECT

10: CURSOR TOOLS

Duplicate ⧉, Delete ✂, Grow ⤢, and Shrink ⤡ a sprite (by clicking an icon and then a sprite on the stage). Click the Block Help tool ?, then a block in the palette to learn more about it.

11: SCRIPTS/ COSTUMES/ SOUNDS TABS

Switch between coding your project, and adding costumes and sounds.

12: MOUSE POINTER CO-ORDINATES

13: SHARE

If you have a Scratch account, you can share your projects with the community.

14: SEE PROJECT PAGE

Add instructions and other notes to your project, and see how others in the community are interacting with it.

15: TIPS

Get project tutorials, tips on using Scratch, and learn more about how each block works.

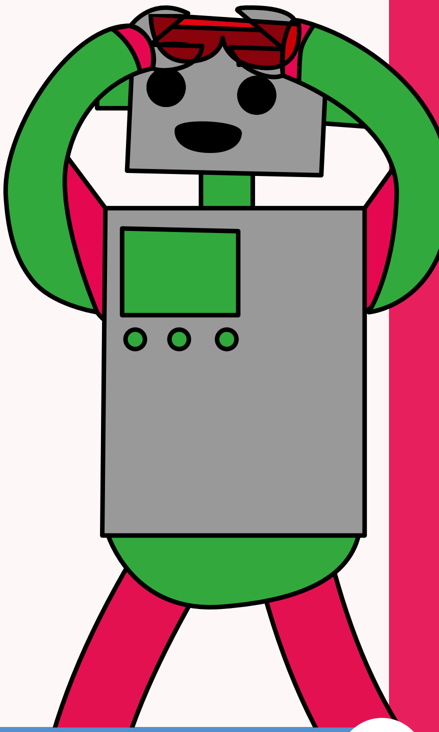
16: ZOOM

17: MENU

Use the menu to load, save, and browse your projects, and access loads of other useful options.

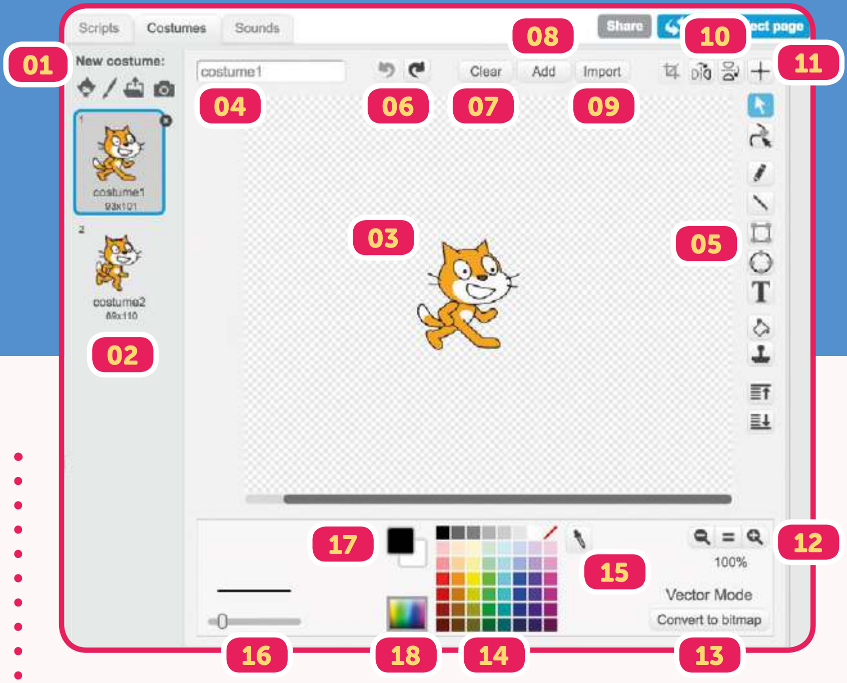
18: MY STUFF

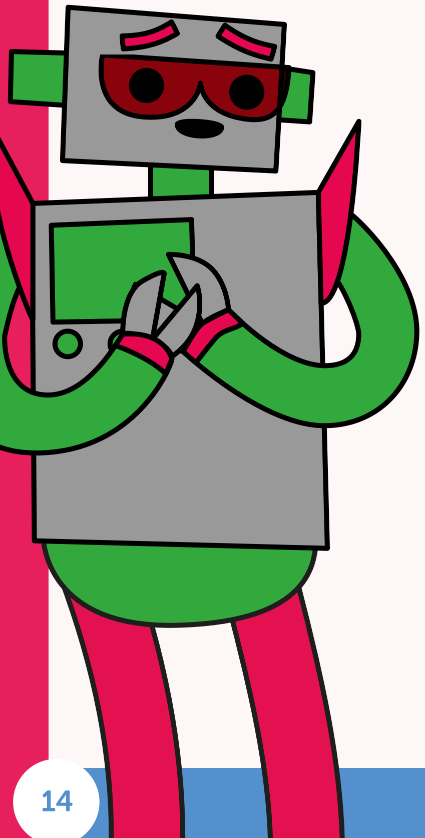
This is where your projects are stored online.



Costumes Tab

Click on this tab to open the paint editor





01: NEW COSTUME

Add costumes to a sprite by adding them from the Scratch library ◆ , drawing your own / , uploading an image from your computer ⏏ , or using your webcam 📷 to take a picture.

02: COSTUMES LIST

Your sprite's costumes will appear here, and you can click one to start editing it.

03: CANVAS

This is the canvas where you edit a costume.

04: COSTUME NAME

You can change the name of a costume, so that you can find it more easily.

05: TOOLS

You can use these tools to edit your image. You can add lines, shapes and text, as well as adding colour, and lots more.

06: UNDO/REDO

Use these arrows to undo or redo your last action.

07: CLEAR

Clear the current costume and start again!

08: ADD

Add another costume image from the Scratch library.

09: IMPORT

Add another costume image from your computer.

10: FLIP

Flip costume horizontally or vertically .

11: COSTUME CENTRE

Set your costume's centre, which is used when moving and rotating your sprite.

12: ZOOM

Use these icons to zoom in and out of your costume as you edit it.

13: BITMAP/VECTOR MODE

The paint editor has two modes – Bitmap and Vector. In Vector mode (shown here), the editor lets you to edit shapes after you have created them, and your costumes and backgrounds will look really good when you make them bigger. When you create a new costume, the editor will be in Bitmap mode by default. In Bitmap mode, you can't easily move or resize shapes you have drawn, but some people find it easier to get started with. When you edit an existing costume, the editor will be in the mode that the costume was created with.

14: COLOUR PALETTE

Use this palette to choose a colour.

15: COLOUR PICKER

Use this to pick up a colour on your costume.

16: LINE SIZE

Move this slider to change the line size used when drawing.

17: COLOUR SWITCH

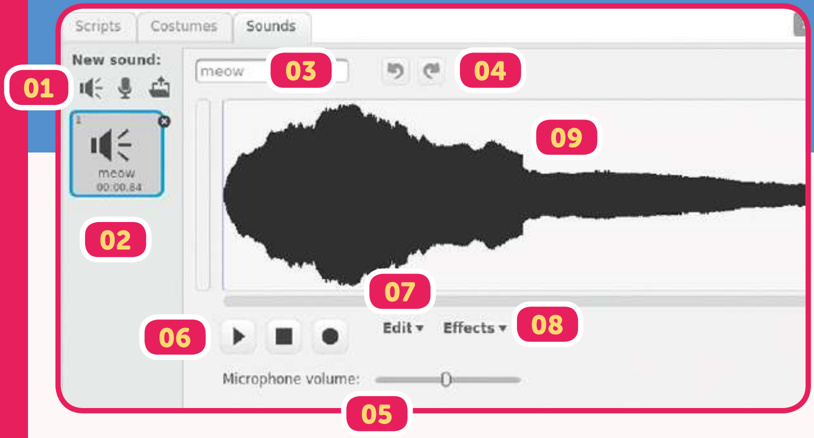
Switch between two selected colours.

18: SWITCH PALETTE

Change the colour palette to 'advanced', to give you access to more shades.

Sounds Tab

Change the sounds your sprites make



01: NEW SOUND

You can add sounds to a sprite (or the stage) from the Scratch library 🔈, by recording your own (if you have a microphone) 🎤, or by uploading a sound from your computer 📤.

02: SOUNDS LIST

Your sprite/stage's sounds appear here, and you can click one to start editing it.

03: SOUND NAME

You can change the name of a sound, so that you can find it more easily.

04: UNDO/REDO

Undo or redo your last action.

05: MICROPHONE VOLUME

Adjust your microphone volume to record quieter or louder sounds.

06: PLAYBACK CONTROLS

Listen to your sound, or record a new one.

07: EDIT

Remix your sound by cutting, copying, and pasting.

08: EFFECTS

Add effects to your sound, such as fading in and out or reversing.

09: SOUND WAVE

This is what your sound looks like! You can select a part of your sound to edit by dragging over it using the mouse.

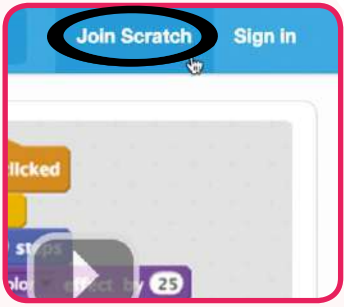
Creating a Scratch Account

Save and share your projects online

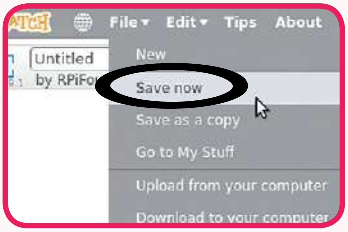
Creating a Scratch account will allow you to save your projects online, so that you can access them from any computer with an internet connection. You will also be able to share your projects with the Scratch community and comment on other projects. To create a Scratch account, click **Join Scratch**. When coding online...

- **Don't use your real name when creating a user name.**

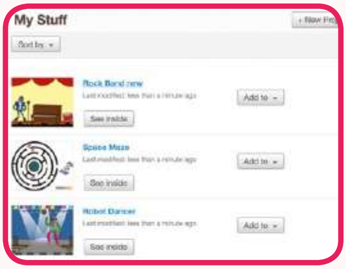
- **Be respectful of others when commenting on and remixing projects.**



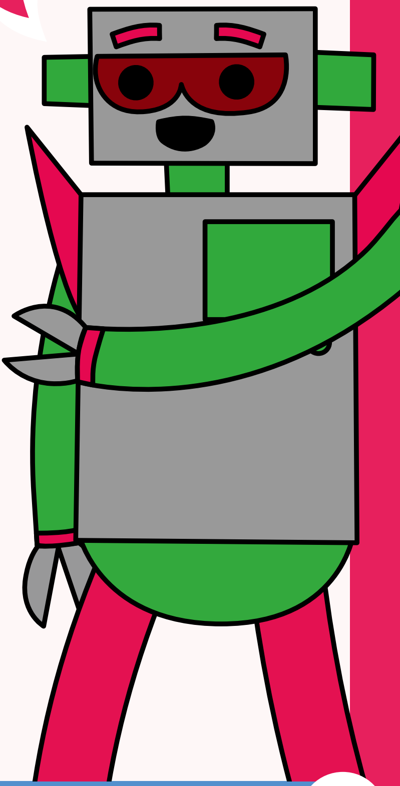
If you have a Scratch account, you can click **File** and then **Save now** to save your project. Once you've saved your project, it will appear in your **My Stuff** folder.



To access your stuff from within a project, click **File** and then **Go to My Stuff**. You should see a list of all of your projects.

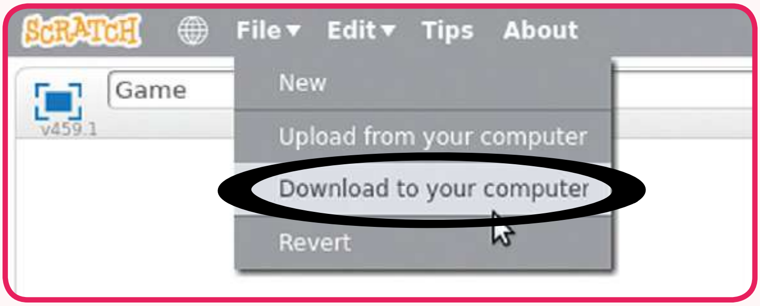


You'll need parental permission to set up an account if you are under 13 years of age. Read the community guidelines at **scratch.mit.edu/community_guidelines** before creating an account.

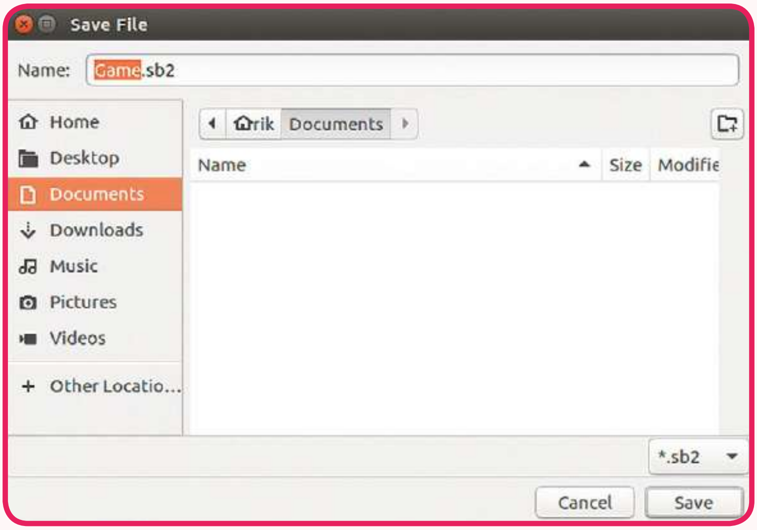


SAVING PROJECTS WITHOUT A SCRATCH ACCOUNT

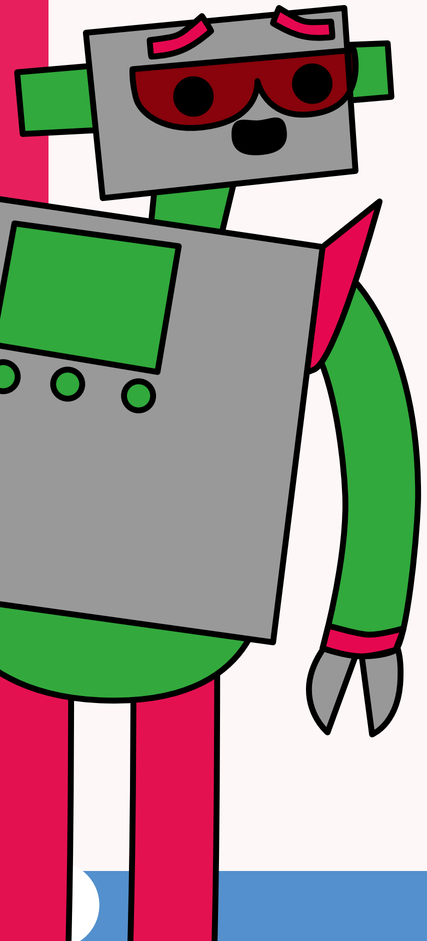
If you don't have a Scratch account, you can still save your Scratch projects by clicking **File** and then **Download to your computer**. You will then be asked where to store the Scratch project, which will be a .sb2 file. This will download your project from the Scratch editor.



To continue working on your project, go into the Scratch editor and click **File** and then **Upload from your computer**. Find your Scratch .sb2 file and click **OK** / **Open**. This will upload your project to the Scratch editor.


Save File

Name: Game.sb2

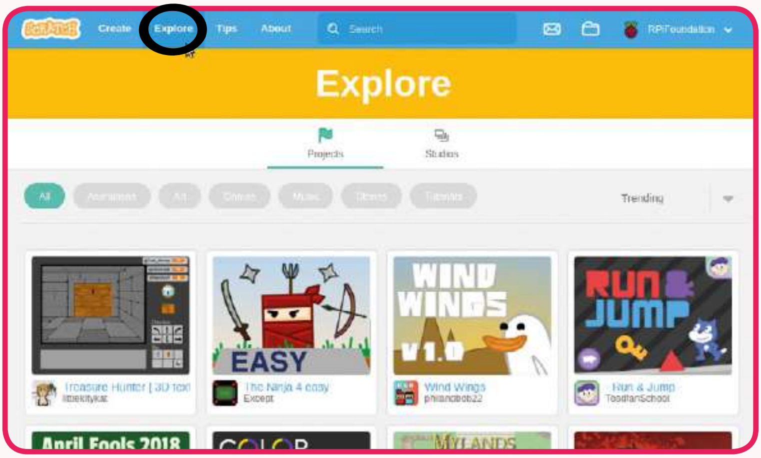


The Scratch community

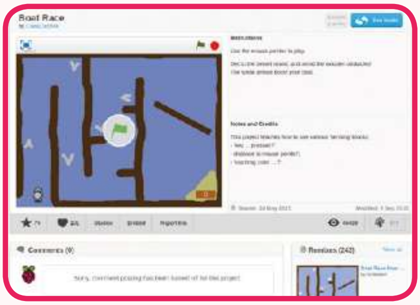
One of the great things about programming in Scratch is that you get to be part of a community of millions of people around the world, all creating and sharing their ideas with each other.

FINDING PROJECTS

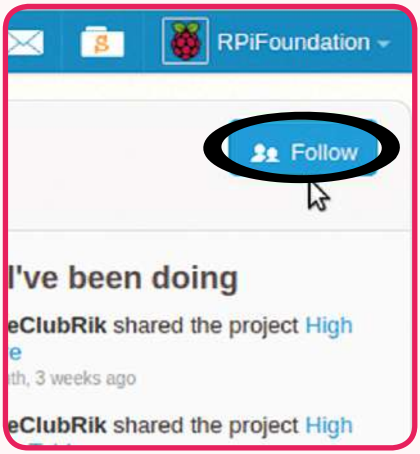
To see what others in the Scratch community are making, click **Explore** in the top menu of the website. You can look for popular or recently created projects, as well as searching by keyword, such as 'Games' or 'Tutorials'. You can use the search bar if you are looking for something in particular.



Once you've found a project you like, you can click the green flag to play it. Below the project are buttons to favourite/love a project or to report a project if it is inappropriate. You can also leave a comment, and click **See Inside** if you want to see the code.

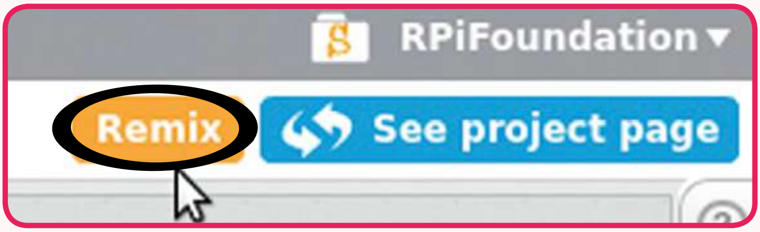


If you find someone whose work you like, you can click their user name and then click **Follow**. You will then be notified when they create something new.


RPiFoundation
Follow
I've been doing
eClubRik shared the project High
e
th, 3 weeks ago
eClubRik shared the project High

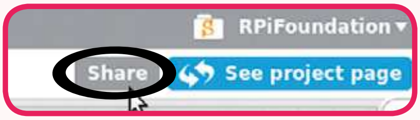
REMIXING

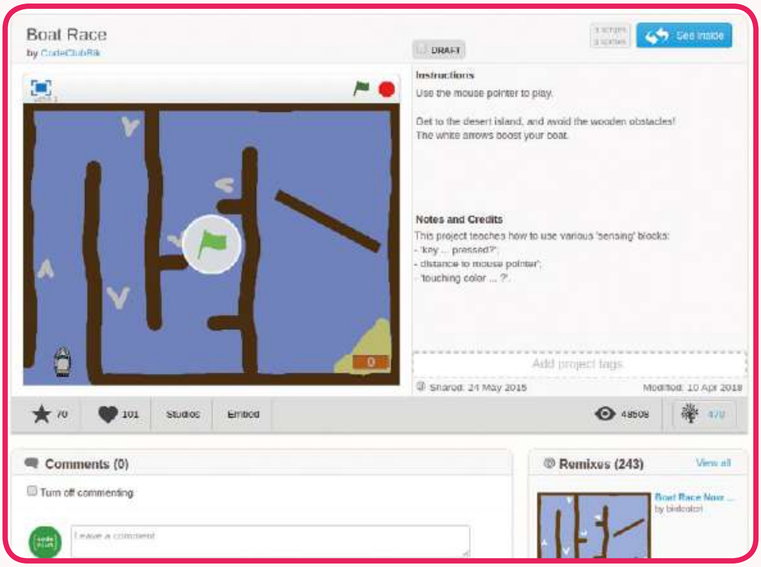
You can use other Scratch projects to get ideas, and use them as a starting point for your own creations. If you have a Scratch account, you can click **Remix** on a project to save your own copy.



Sharing

Sharing your projects with the Scratch community allows others to enjoy your awesome creations. Projects aren't shared with the community unless you want them to be, and you can share projects by clicking the Share button at the top-right.





Instructions

Use the mouse pointer to play.

Get to the desert island, and avoid the wooden obstacles!
The white arrows boost your boat.

Notes and Credits

This project teaches how to use various 'sensing' blocks:
- 'key ... pressed?';
- distance to mouse pointer';
- 'touching color ... ?'.

Shared: 24 May 2015 Modified: 10 Apr 2018

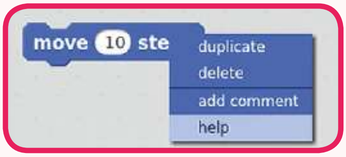
□ Turn off commenting

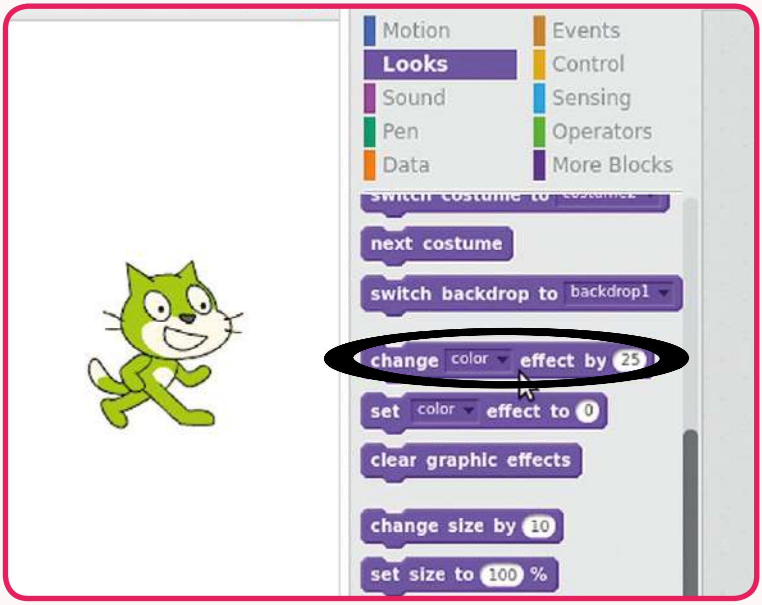
Before sharing your project, it's a good idea to check the project page to make sure the community have all the information they need to use your project. You can add instructions to tell others how to use your project, and credit other people who have helped you (especially if you've remixed a project).

Once shared, others in the community will be able to comment on your project, although you can disable comments if you prefer. Comments are really useful for improving your project by finding out what people do and don't enjoy. You can also see how many people have viewed, favourited, and loved the project, as well as how many have remixed your project.

Tips for Scratch coding

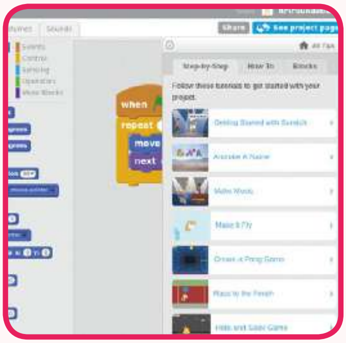
If you're not sure what a code block does, you can right-click and select **help** to learn more about it. You can also just click the block to see what it does before adding it to a script!





If you need a bit more help, Scratch has a help section that includes:

- **Step-by-step instructions for making animations, stories, music, and games**
- **A 'How to' section that shows you how to do specific things in your project**
- **A 'Blocks' section that explains what each of the blocks do**

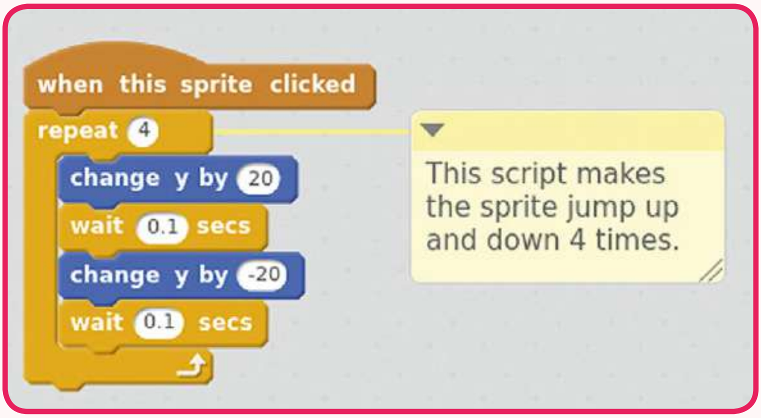


If you are not sure how to do something, you can also ask others for help. Maybe they had the same problem as you!

Test your code regularly, to make sure your code does what you want it to. You will find it much easier to fix problems in your code if you test each time you make a change to your code.

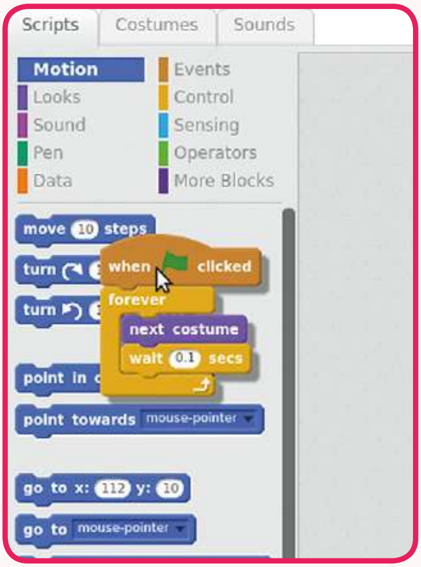
Get others to try out your projects, and ask them what they like about your project and what they would improve.

You can add comments to a script by right-clicking on a block and selecting **add comment**. It's a good idea to comment a script to explain what it does, so that others will know what your scripts do. It's also useful in case you forget what your code does!



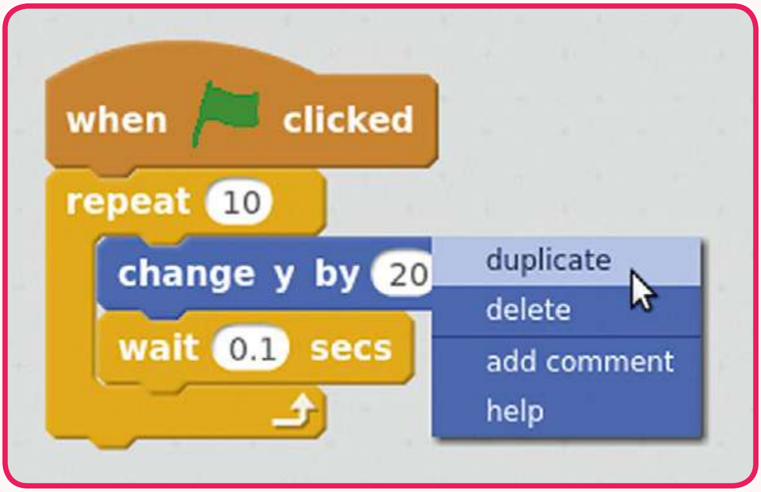
```
when this sprite clicked
repeat 4
    change y by 20
    wait 0.1 secs
    change y by -20
    wait 0.1 secs
```

This script makes the sprite jump up and down 4 times.



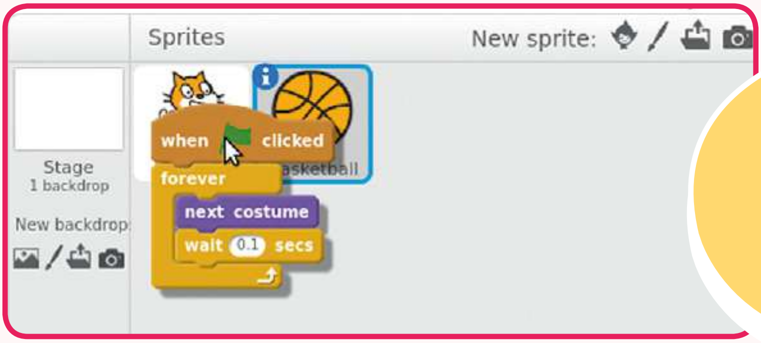
To delete blocks, drag them over the palette area. Don't worry if you accidentally delete blocks you need: you can click the **File** menu and then **undelete** to get them back!

You can right-click on a block and choose **duplicate** to make a copy of that block and the blocks attached below it.



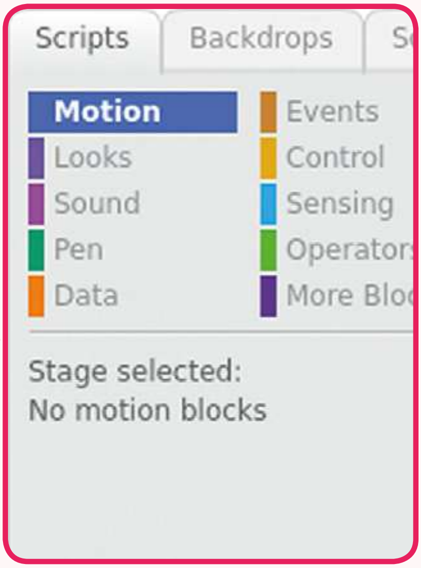
```
when [flag] clicked
repeat 10
    change y by 20
    wait 0.1 secs
```

Dragging blocks to another sprite makes a copy of them. This is useful if you need similar code in another sprite.



```
when [flag] clicked
forever
    next costume
    wait 0.1 secs
```

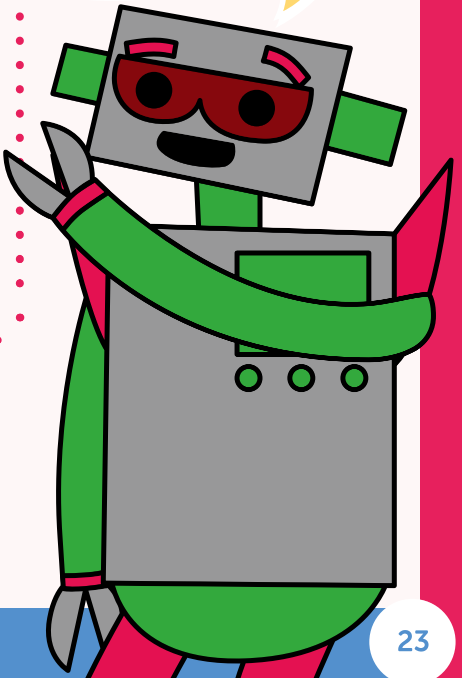
If you can't find the blocks you need to control a sprite, for example the Motion blocks, it may be that you have the Stage selected.



Stage selected:
No motion blocks

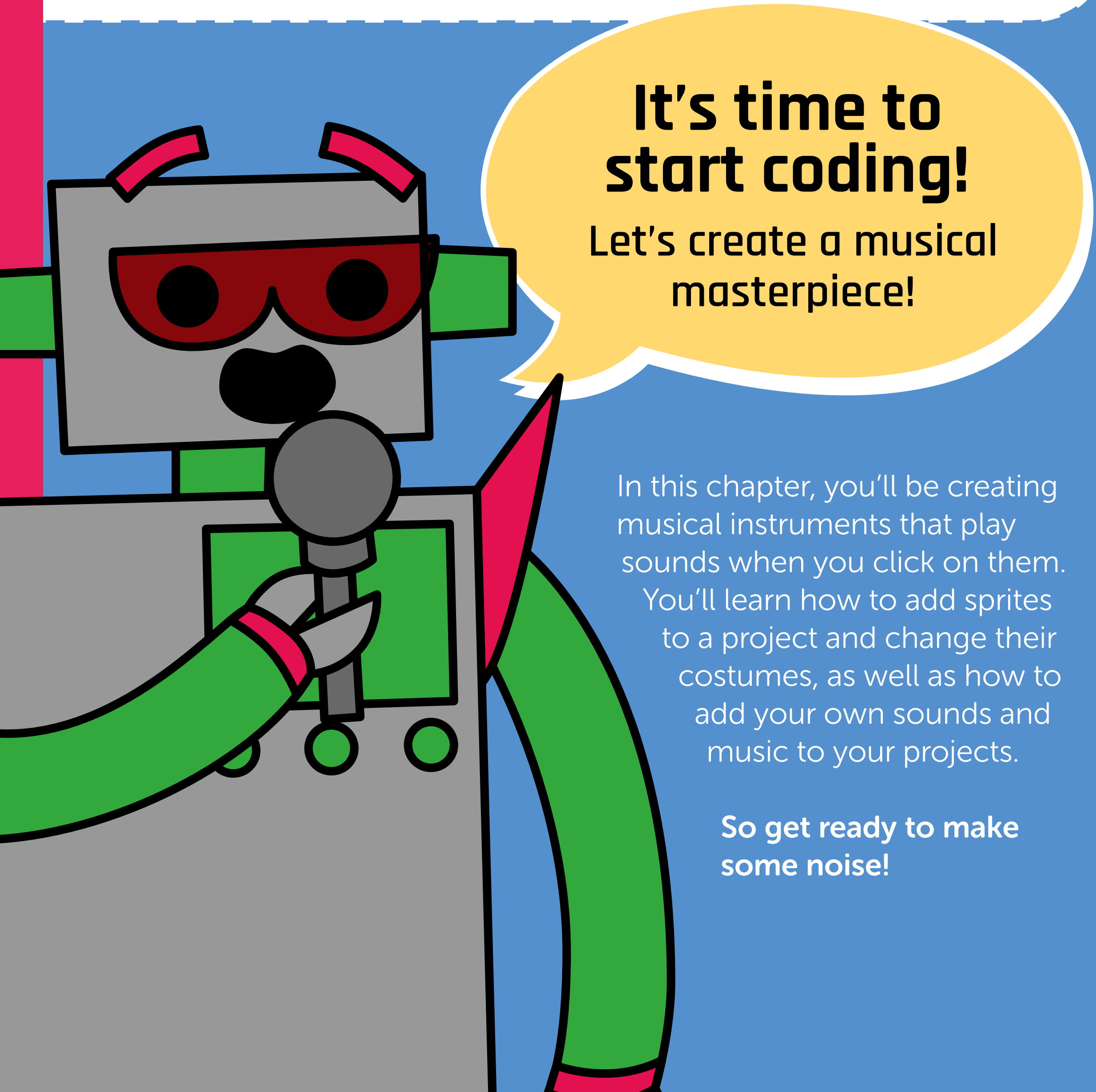
Can't wait to get coding?

Turn the page to start your first project...



Rock Band

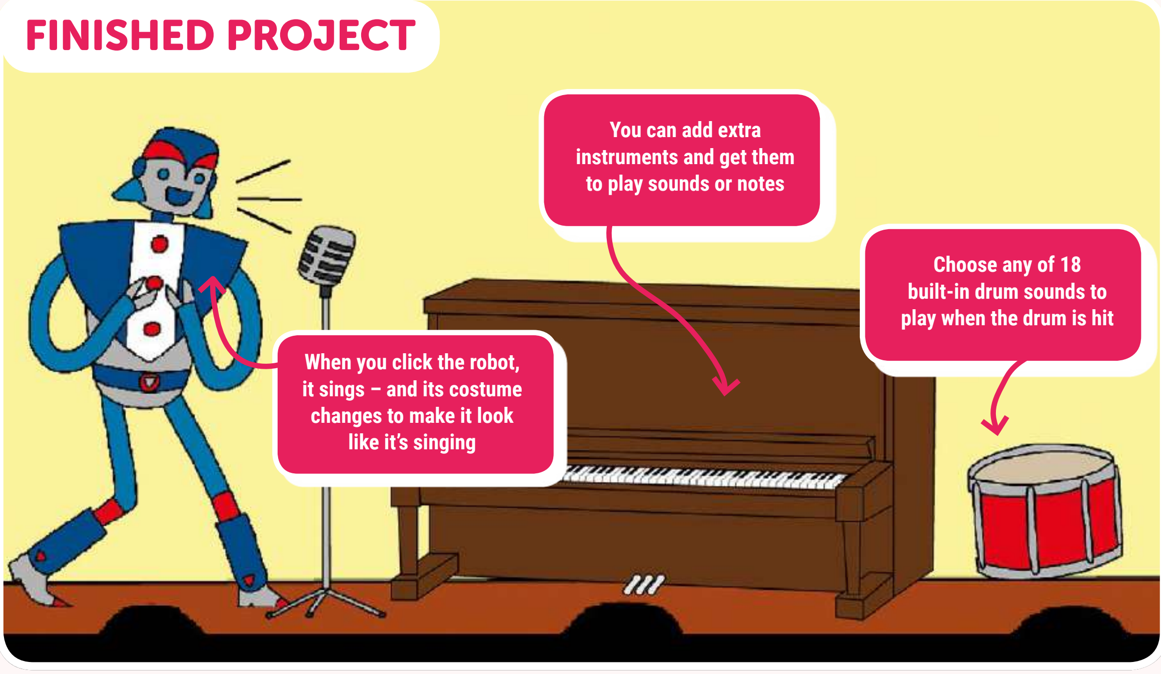
Create your own virtual rock band by coding a selection of musical instruments


It's time to start coding!
Let's create a musical masterpiece!

In this chapter, you'll be creating musical instruments that play sounds when you click on them. You'll learn how to add sprites to a project and change their costumes, as well as how to add your own sounds and music to your projects.

So get ready to make some noise!

FINISHED PROJECT



You can add extra instruments and get them to play sounds or notes

Choose any of 18 built-in drum sounds to play when the drum is hit

When you click the robot, it sings – and its costume changes to make it look like it's singing

STEP 1: SPRITES AND THE STAGE

Let's start by taking a look at the Scratch project.

☑ In a web browser, go to **rpf.io/book-rockband** to open the Rock Band Scratch project. Click Remix.

If you'd prefer to use Scratch offline, click **File → Download to your computer** in the Scratch online editor. You can then open the project in the offline editor. *[See the 'Introduction to Scratch' chapter for more information on using Scratch offline.]*

The **stage** is at the top-left of the editor, and is where the action happens. Think of it as a performance area, just like a real stage.

This project contains **sprites** which you can add code blocks to. Sprites appear on the stage and can move around, make sounds, and do lots of other things.

WHAT YOU'LL LEARN

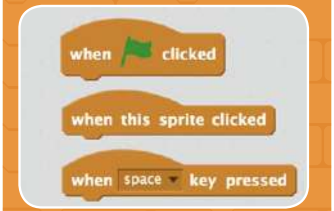
- Sprites
- Costumes
- Events
- Sequencing instructions
- Sound and music

TIP! PROJECT FILES

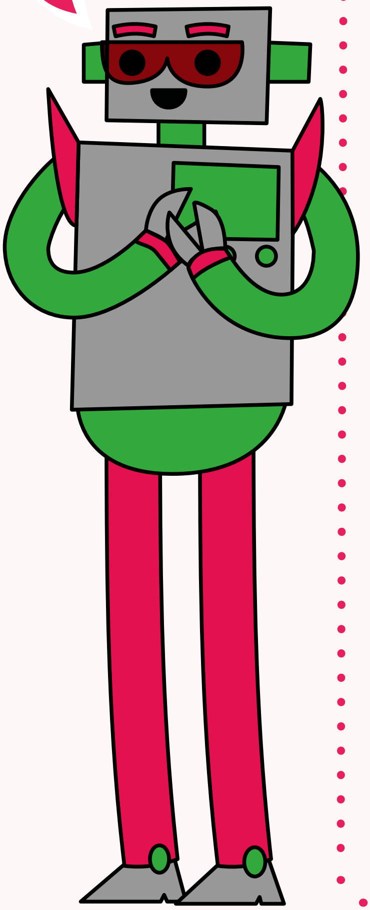
To download a zip file of all the Scratch 2 (.sb2) project assets files for this book, go to:

rpf.io/book-s1-assets

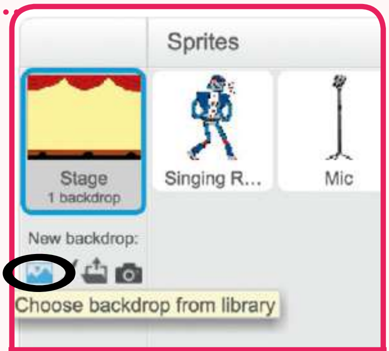
TIP! EVENTS



Events blocks are used to tell sprites when to run some code. Scratch has lots of Events blocks, for running code when a project starts, a sprite is clicked, a key is pressed, and more.



□ If you want to change the stage backdrop, click the **Choose backdrop from library** icon and select your own from the library.

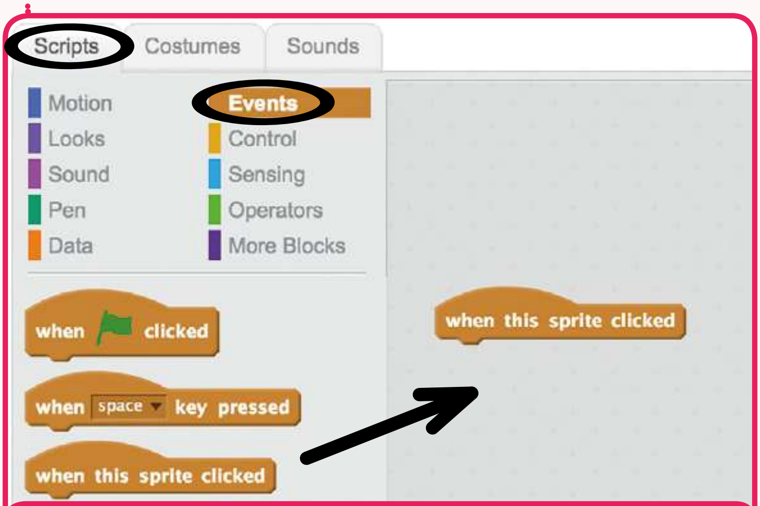


You can add a new backdrop to the project

STEP 2: CODE A DRUM

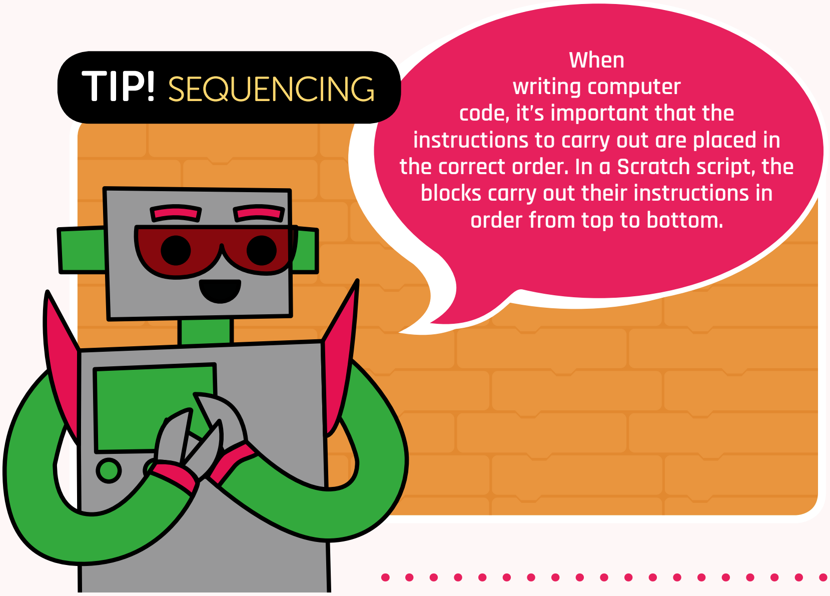
Let's code your drum to make music when it's hit.

□ Select your Drum sprite and click the **Scripts** tab. You should see lots of colour-coded blocks that can be used to control your robot. Click on the **Events** category and then drag a **when this sprite clicked** block from the blocks palette into the coding area to the right.



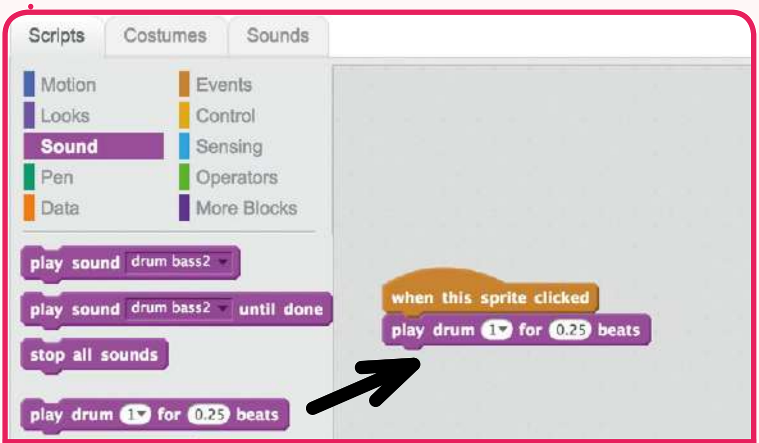
Drag the block to the right and drop it in the coding area

TIP! SEQUENCING



When writing computer code, it's important that the instructions to carry out are placed in the correct order. In a Scratch script, the blocks carry out their instructions in order from top to bottom.

☑ Any code that you attach to your Events block will be run **in order** when you click your drum sprite. To play a sound, click the purple **Sound** category in the Scripts pane, to show all the Sound blocks below. Drag a **play drum** block into the coding area, attaching it to the bottom of the **when this sprite clicked** block.



Drop the **play drum** block just underneath the **when this sprite clicked** block so that it connects to it

☑ TEST YOUR PROJECT

Click on your drum sprite and you should hear a sound.

CHALLENGE

HIT IT

Can you code your drum to make a sound when the SPACE bar is pressed?

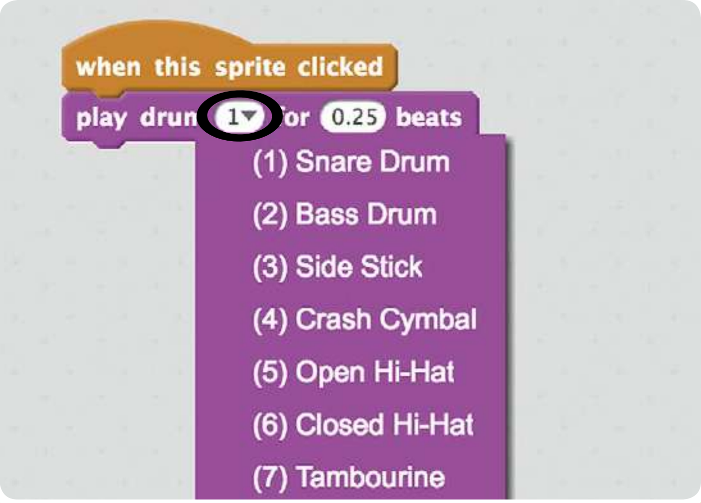
HINT!

You'll need to use a different Events block to get the sprite to react to a key press.

HOW TO...

ALTER YOUR DRUM

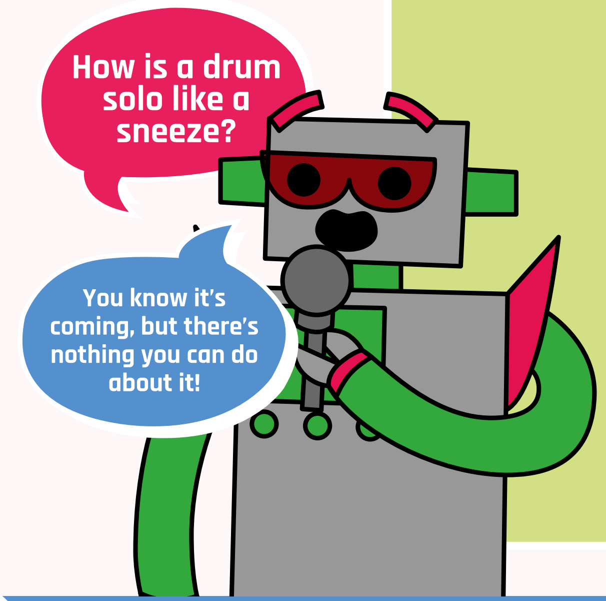
Want to change the sound that your drum makes when it's clicked?

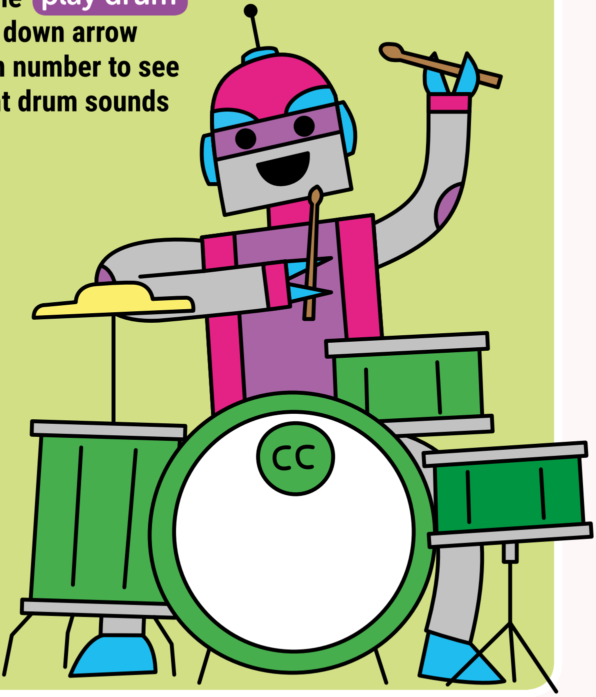


```
when this sprite clicked
play drum (1▾) for (0.25) beats
```

(1) Snare Drum
(2) Bass Drum
(3) Side Stick
(4) Crash Cymbal
(5) Open Hi-Hat
(6) Closed Hi-Hat
(7) Tambourine

It's easy to change the sound of the drum in the play drum block. Click the down arrow next to the drum number to see a list of different drum sounds to choose from.

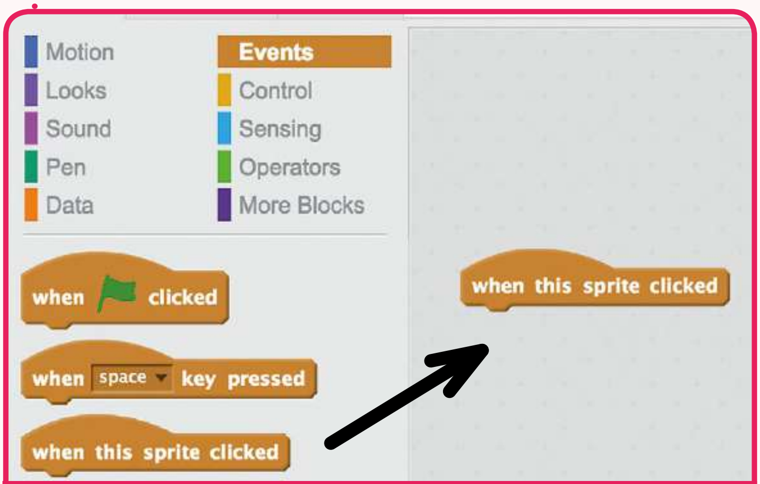

How is a drum solo like a sneeze?
You know it's coming, but there's nothing you can do about it!



STEP 3: ADD A ROBOT SINGER

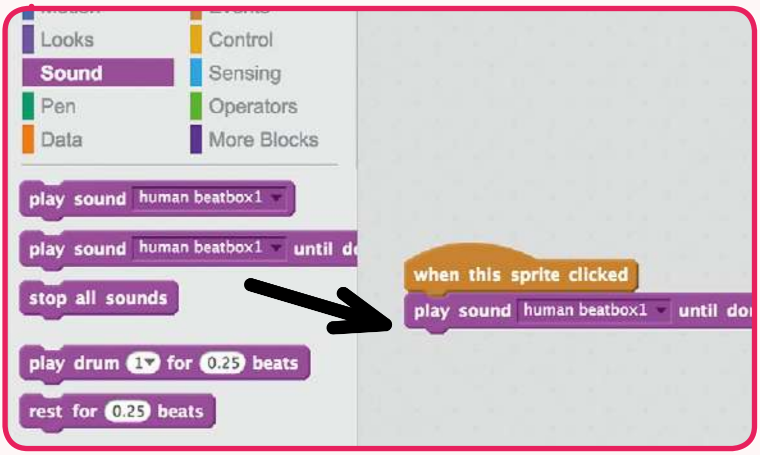
Let's code your robot sprite to make a sound when it's clicked.

□ Click on your robot sprite and then add a **when this sprite clicked** Events block from the blocks palette, just like you did with your drum.



As before, drag the block to the right and drop it in the coding area

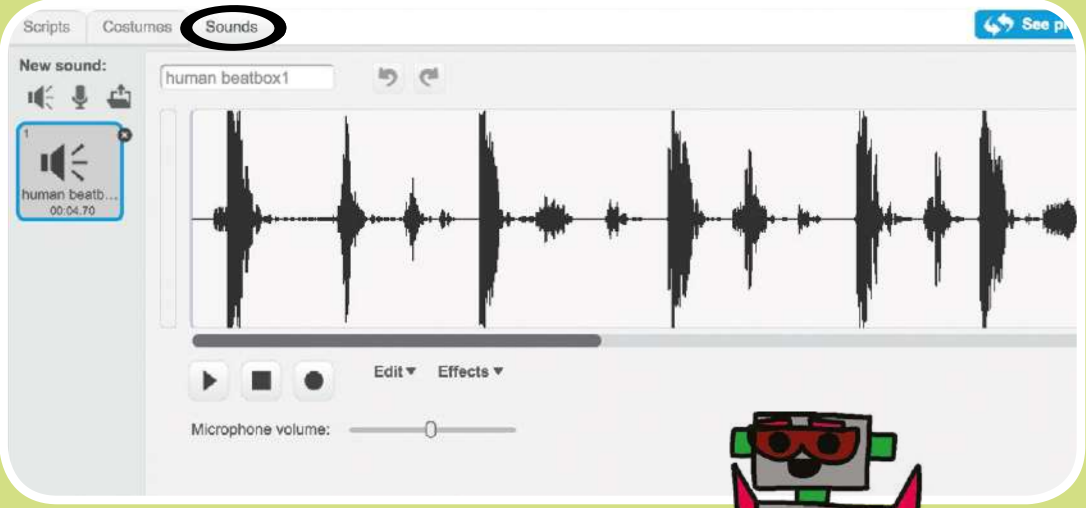
□ Drag a **play sound... until done** block into the coding area, attaching it to the bottom of the **when this sprite clicked** block.


Looks
Sound
Pen
Data
Control
Sensing
Operators
More Blocks
play sound human beatbox1
play sound human beatbox1 until d
stop all sounds
play drum 1 for 0.25 beats
rest for 0.25 beats
when this sprite clicked
play sound human beatbox1 until do

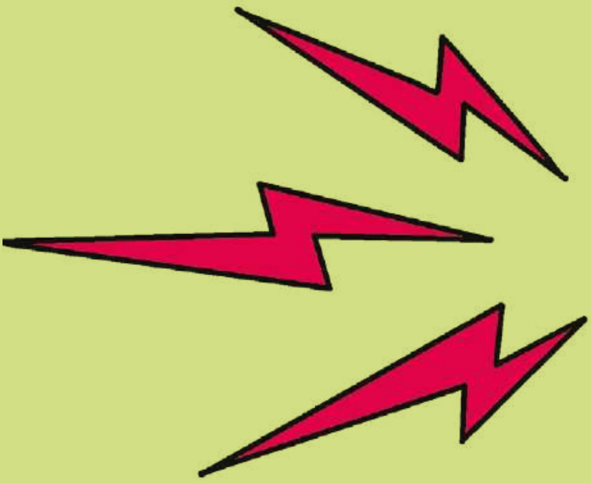
HOW TO...

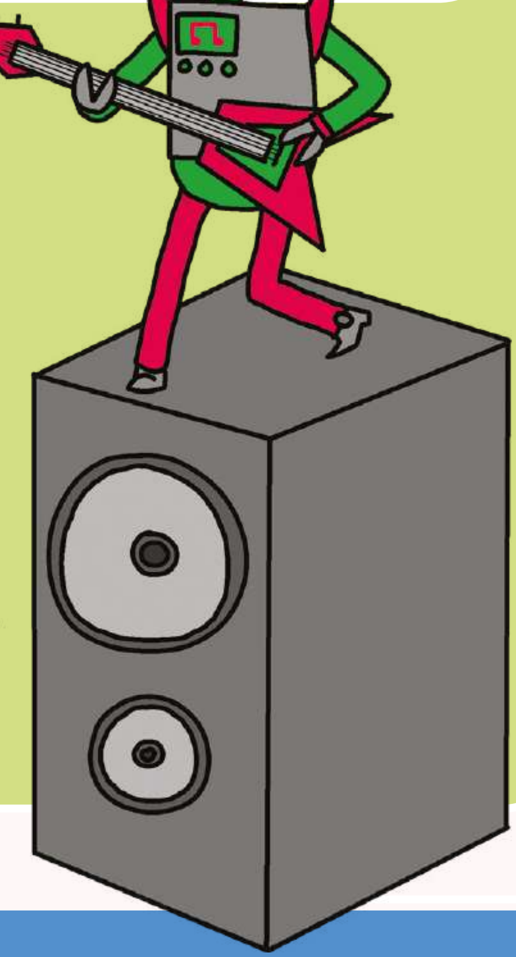
EDIT SOUNDS

Want to change the sound that your robot makes?



First, click on the 'Sounds' tab at the top of the editor. Using the Effects drop-down menu, you can make the sound louder, softer... or even reverse it! In addition, you can add other sounds from the Scratch library, record your own, or upload them, using the icons under 'New sound:'.

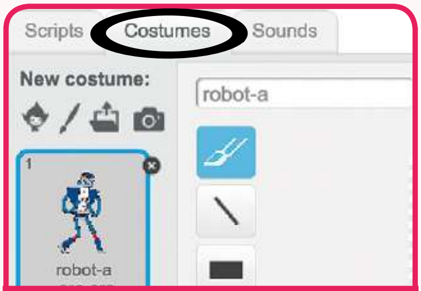




STEP 4: COSTUMES

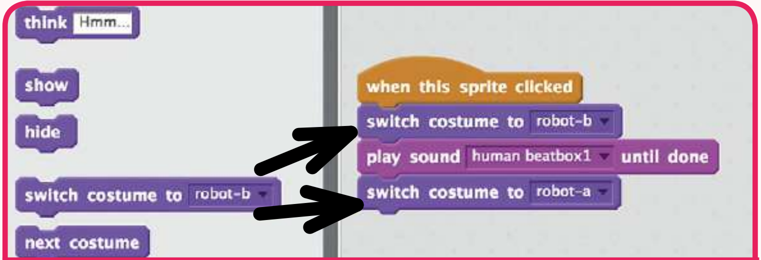
Let's make your robot look like it's singing!

☑ Click on your robot sprite and then click on the **Costumes** tab at the top of the editor. You'll see that the robot has two costumes.



Click the Costumes tab

☑ Click the **Scripts** tab to get back to your code. Click the **Looks** category and then drag two switch costume blocks into your code. Make sure that your robot first displays the **robot-b** costume, plays a sound, and then switches back to **robot-a**.

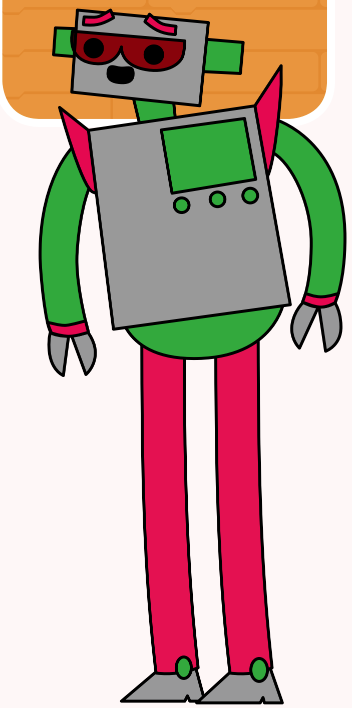


```
when this sprite clicked
switch costume to robot-b
play sound human beatbox1 until done
switch costume to robot-a
```

Add **switch costume** blocks above and below the **play sound** block

TIP! COSTUMES

Sprites in Scratch have a number of costumes, and you can code sprites to switch between costumes to change how sprites look. Scratch includes a library of costumes, or you can even draw your own.



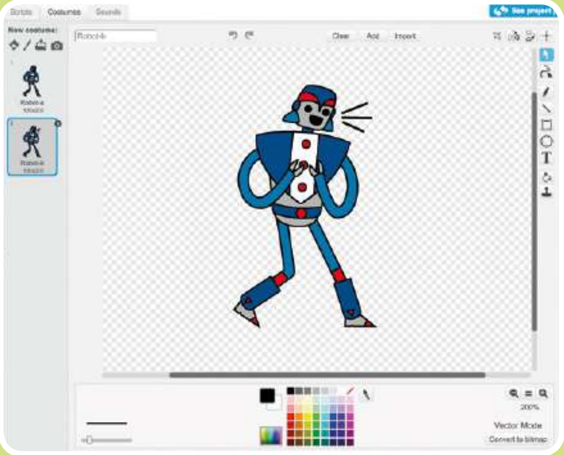
☑ TEST YOUR PROJECT

Click your robot to test it. The robot should now change costume, play a sound, and then change back to the first costume once the sound has finished playing.

CHALLENGE

EDIT COSTUMES

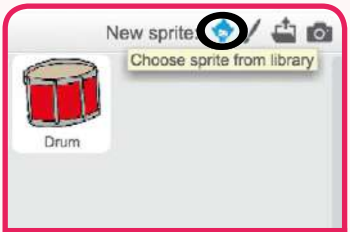
Want to change how the robot looks when it's singing? Click the Costumes tab, then select the robot-b costume. You can then use the paint editor tools to alter it. Currently, it simply has three lines coming from its mouth, drawn using the line tool. You can use editing tools, such as the pencil, to make more changes to your robot.



STEP 5: PLAYING A TUNE

Let's add a new piano sprite that plays a tune when clicked.

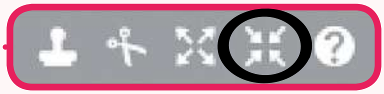
☑ Click the **Choose sprite from library** icon just below the stage to add a new sprite from the Scratch library.

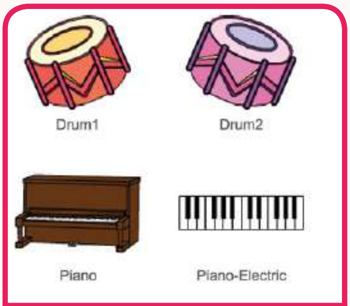


Click this icon to add a sprite from the library

☑ Click the **Music** theme, select the **Piano** sprite, and then click OK to add it to your project.

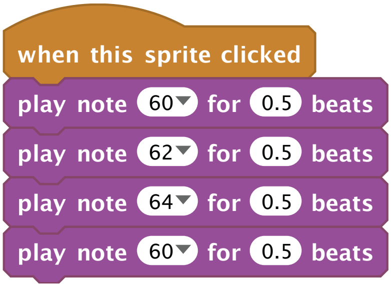
☑ The piano is too large to fit on the stage easily, so click the Shrink icon – in the tools to the right of 'About' in the top bar – and then click repeatedly on the piano on the stage to reduce its size.



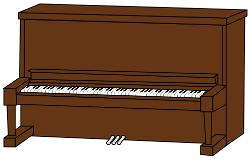


You'll find the piano in the Scratch sprites library

☑ Now add some **play note** blocks under a **when this sprite clicked** block to play a song/tune when the piano sprite is clicked.

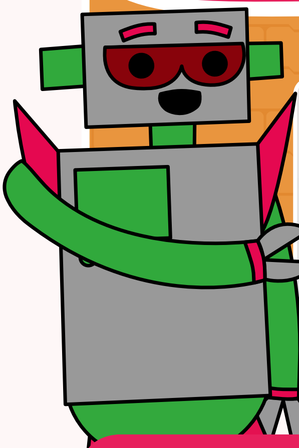


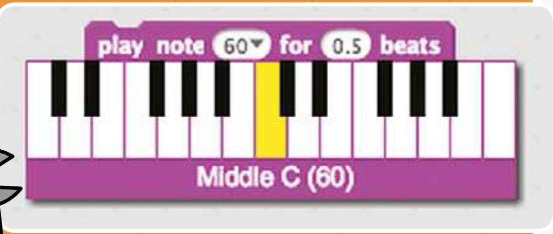
```
when this sprite clicked
play note 60▾ for 0.5 beats
play note 62▾ for 0.5 beats
play note 64▾ for 0.5 beats
play note 60▾ for 0.5 beats
```



TIP! PLAY NOTE BLOCKS

The numbers in the **play note** blocks relate to musical notes: number 60 is 'Middle C', and the higher the number the higher the note! If you click the arrow next to the number, a keyboard will appear below the block, to help you choose the notes for your tune.





```
play note 60▾ for 0.5 beats
```

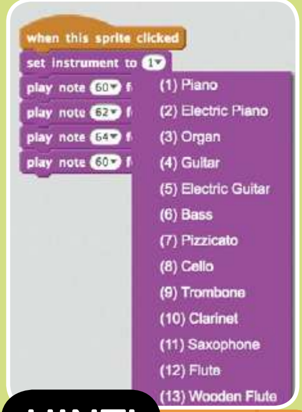
☑ TEST YOUR PROJECT

What music is played when the piano sprite is clicked?

CHALLENGE

CREATE YOUR OWN TUNE

Can you change the notes played, and create your own tune?



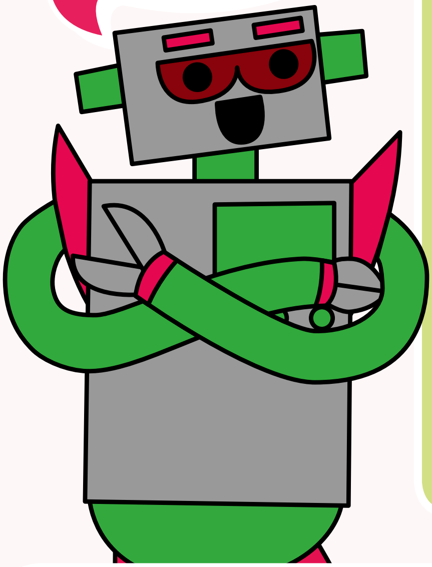
```
when this sprite clicked
set instrument to 1▾
play note 60▾ f
play note 62▾ f
play note 64▾ f
play note 60▾ f
```

(1) Piano
(2) Electric Piano
(3) Organ
(4) Guitar
(5) Electric Guitar
(6) Bass
(7) Pizzicato
(8) Cello
(9) Trombone
(10) Clarinet
(11) Saxophone
(12) Flute
(13) Wooden Flute

HINT!

You can change the numbers in the play note blocks to create your own tune, and even use the 'set instrument' block to choose a different instrument!

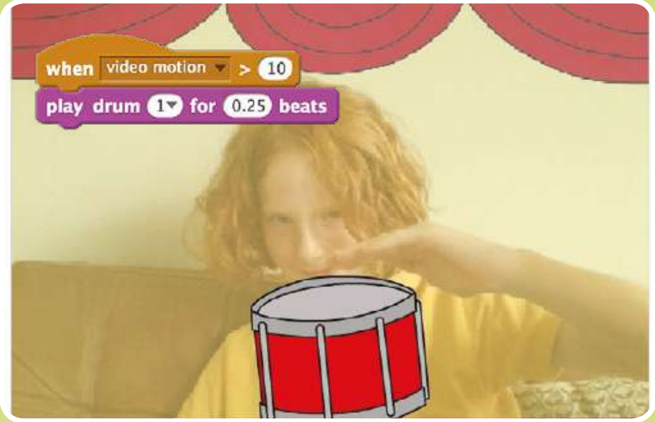
Grab a **when loudness** block, click the down arrow on it, and select **video motion**. Add a **play drum** block, then wave your hand to test it!



HOW TO...

USE WEBCAM INPUT

If you have a webcam, you can use it to play instruments when you move over them!

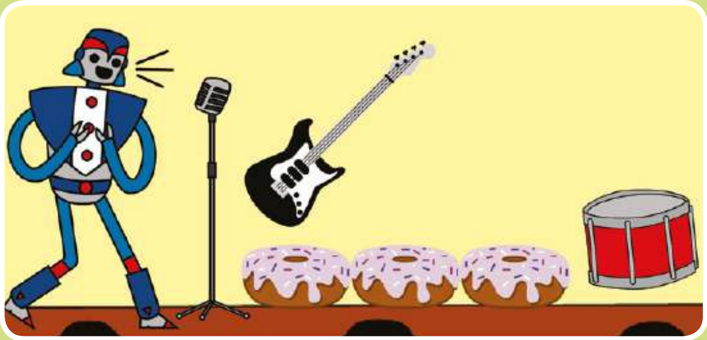


```
when video motion > 10
play drum 1 for 0.25 beats
```

CHALLENGE

MAKE YOUR OWN BAND

Can you use what you've learnt in this chapter to make your own band? Look at the available sounds and instruments to get some ideas, or you could even draw your own. Your instruments don't have to be sensible – you could make a piano made out of doughnuts!



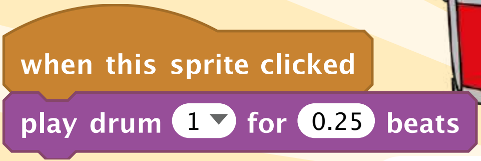
HINT!

As well as using costumes, backdrops, and sounds from the Scratch library, you can create your own – use the Paint or Record sound option.

ROCK BAND: FULL CODE LISTING

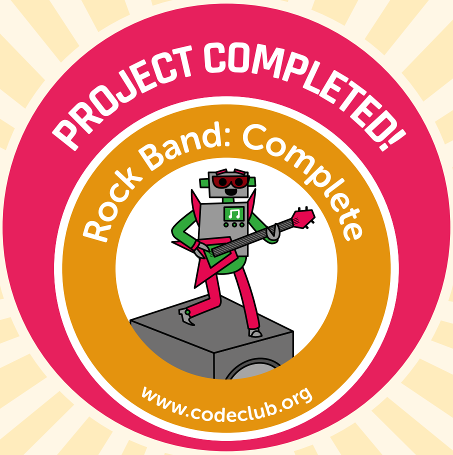
DRUM

When the drum sprite is clicked, a drum beat is played.



```
when this sprite clicked
play drum 1 for 0.25 beats
```

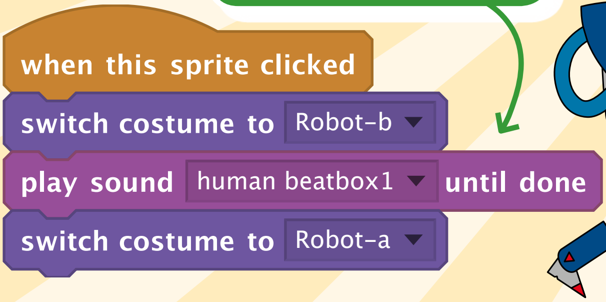
This Sound block plays the chosen drum sound for 0.25 beats



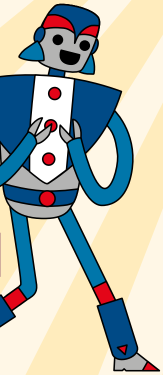
ROBOT SINGER

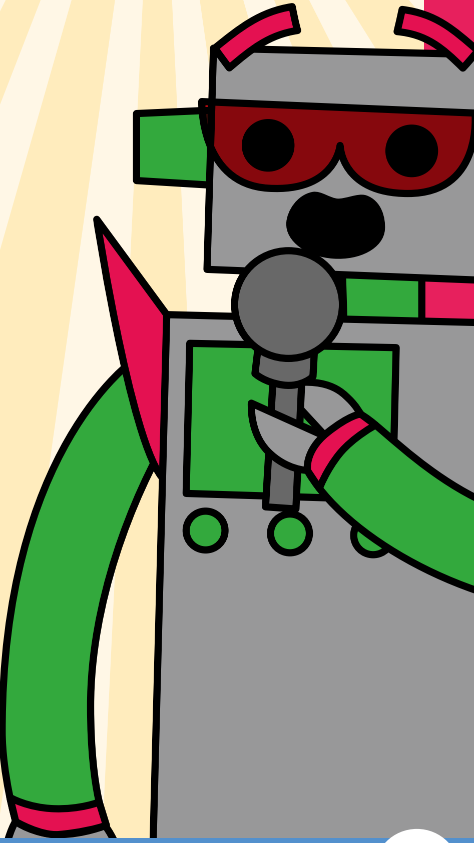
When the robot is clicked, it changes its costume before playing a sound. Once the sound has finished, the robot changes back to the first costume.

This block waits until the sound has finished playing before moving on to the next one



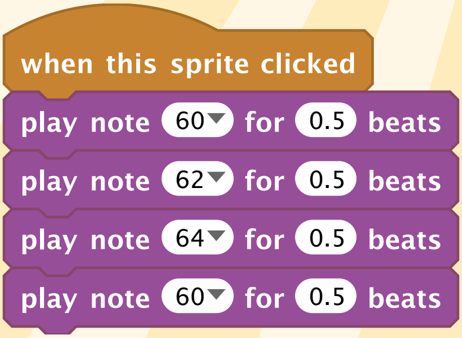
```
when this sprite clicked
switch costume to Robot-b
play sound human beatbox1 until done
switch costume to Robot-a
```





PIANO

When the piano is clicked, four notes are played one after the other.



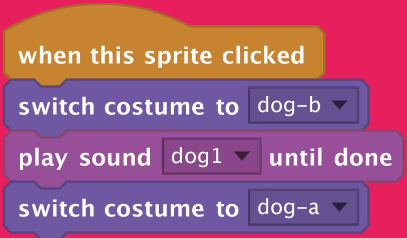
```
when this sprite clicked
play note 60 for 0.5 beats
play note 62 for 0.5 beats
play note 64 for 0.5 beats
play note 60 for 0.5 beats
```

Now You Could Make...

With the skills you've learnt, why not try these projects?

SOUNDBOARD

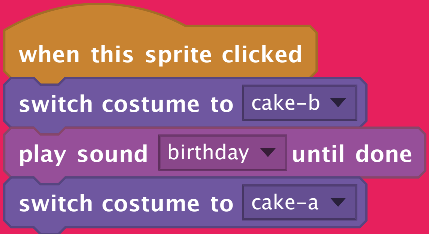
Fill the stage with lots of different sprites that make a noise or play some music when clicked.



```
when this sprite clicked
switch costume to [dog-b ▾]
play sound [dog1 ▾] until done
switch costume to [dog-a ▾]
```

INTERACTIVE BIRTHDAY CARD

Create an interactive birthday card for a friend. You could play them a song or even record your own personalised message.



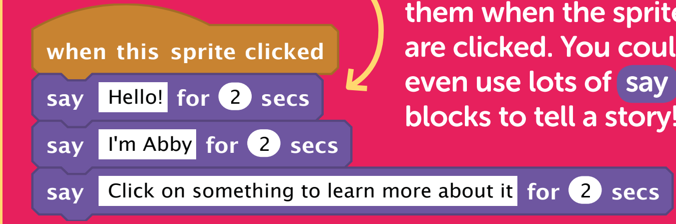
```
when this sprite clicked
switch costume to [cake-b ▾]
play sound [birthday ▾] until done
switch costume to [cake-a ▾]
```

Fancy heading out into space?

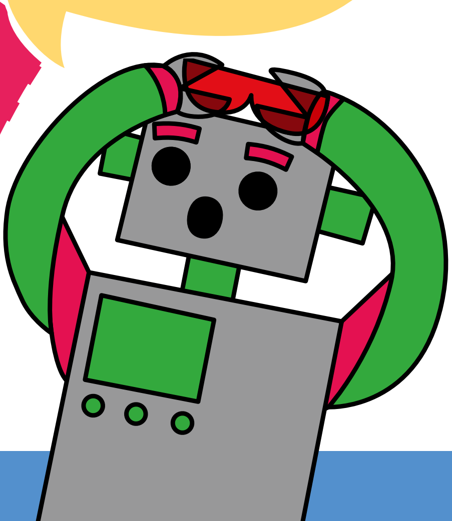
Turn the page to find out how...

ABOUT YOU

Create a project to tell people more about you. You could add sprites for your favourite hobbies and interests, and use say blocks to talk about them when the sprites are clicked. You could even use lots of say blocks to tell a story!

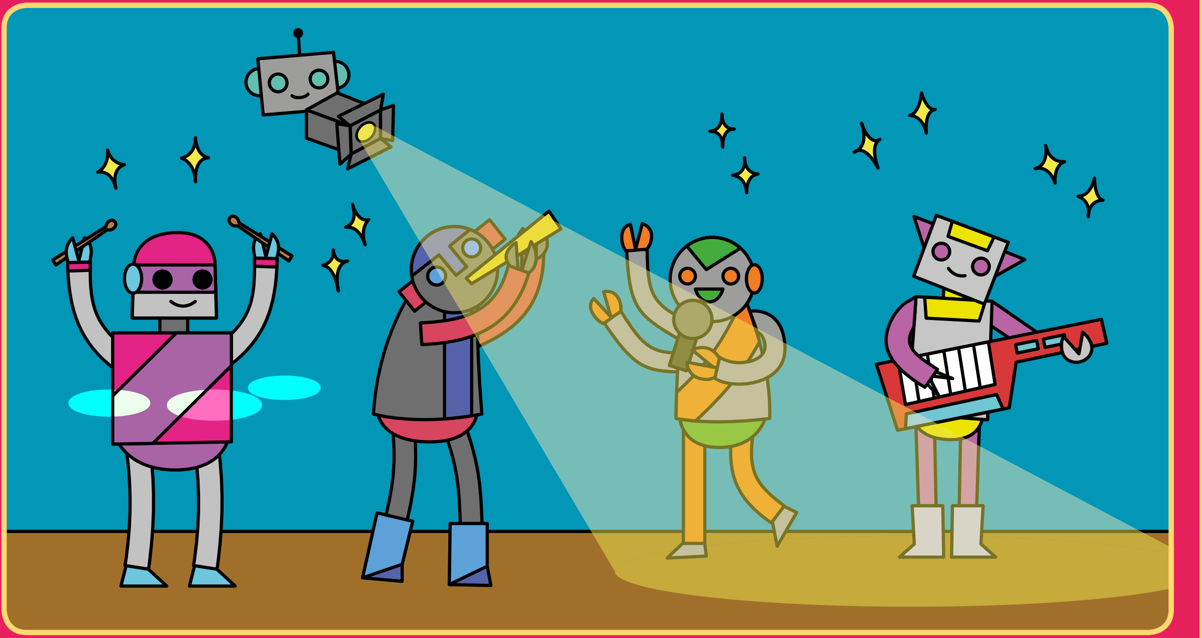


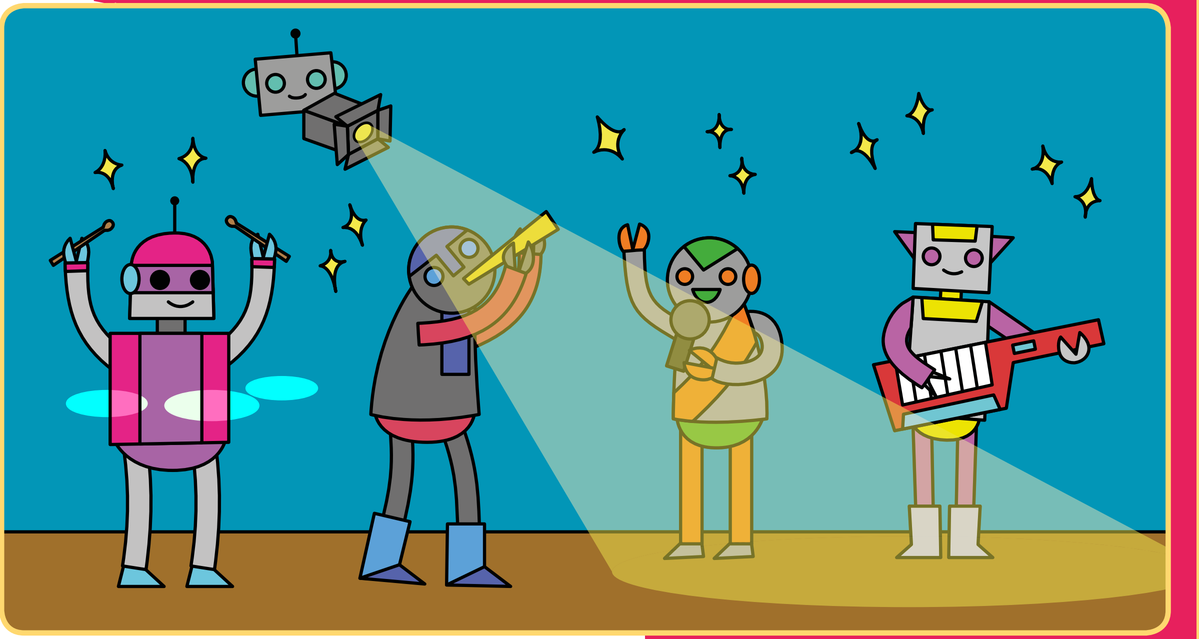
```
when this sprite clicked
say [Hello!] for (2) secs
say [I'm Abby] for (2) secs
say [Click on something to learn more about it] for (2) secs
```



Spot the Difference

There are ten differences between these two images. Can you spot them all? Answers on page 110.



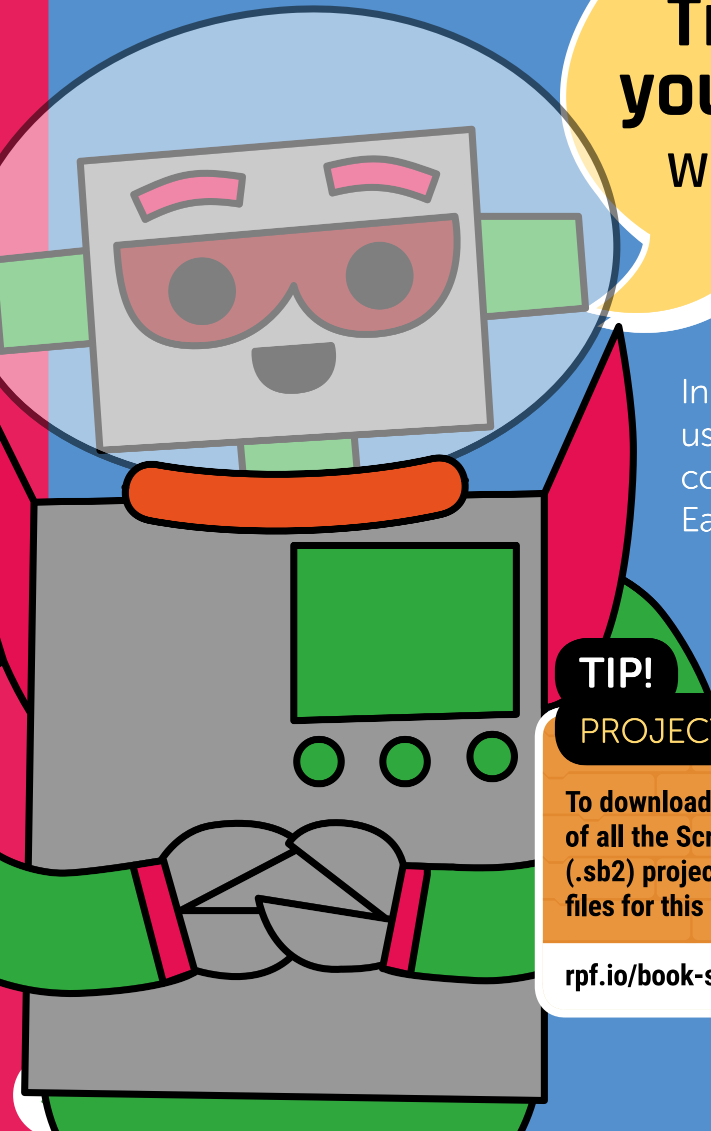


Lost in Space

Create your own space-themed animation, including spaceships, asteroids, and floating space-monkeys

Time to launch your next project!

We're heading to outer space for this one!



In this chapter you'll learn how to use loops to animate sprites. You'll code a spaceship that travels back to Earth, a floating monkey astronaut, an asteroid, and a shining star.

TIP!

PROJECT FILES

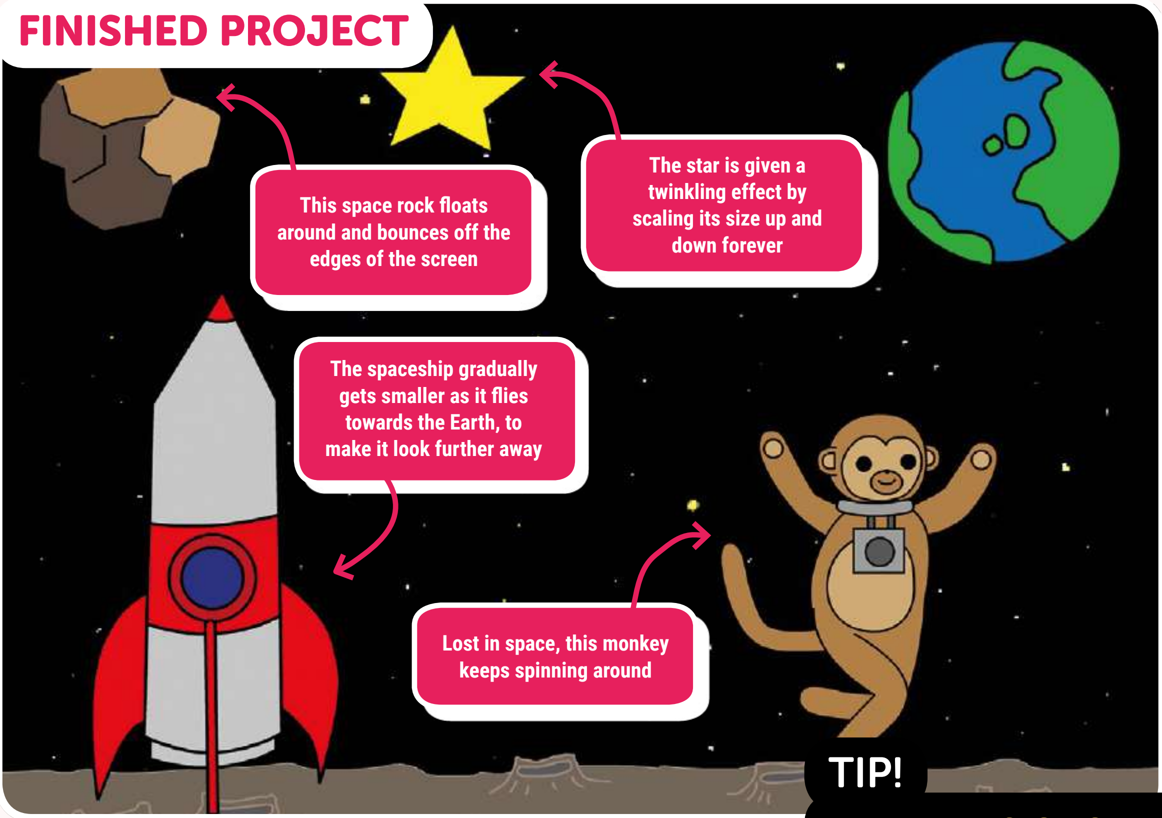
To download a zip file of all the Scratch 2 (.sb2) project assets files for this book, go to:

rpf.io/book-s1-assets

WHAT YOU'LL LEARN

- Moving sprites around the stage
- Repetition (loops):
  - repeat block
  - forever block

FINISHED PROJECT



This space rock floats around and bounces off the edges of the screen

The star is given a twinkling effect by scaling its size up and down forever

The spaceship gradually gets smaller as it flies towards the Earth, to make it look further away

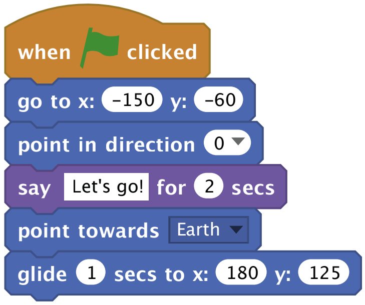
Lost in space, this monkey keeps spinning around

STEP 1: ANIMATE A SPACESHIP

Let's start by making a spaceship that flies towards the Earth.

☐ In a web browser, go to **rpf.io/book-lostinspace** to open the Lost in Space project.

☐ Click on the Spaceship sprite and add the following code:

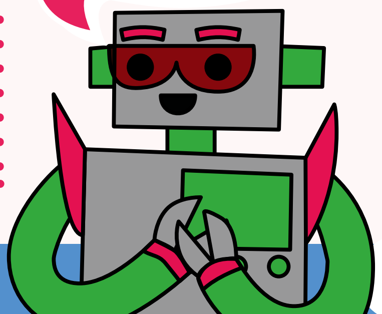


```
when ⚑ clicked
go to x: -150 y: -60
point in direction 0 ▾
say Let's go! for 2 secs
point towards Earth ▾
glide 1 secs to x: 180 y: 125
```

TIP!

WHEN FLAG CLICKED

Any code attached to a **when ⚑ clicked** block will be run when the project first starts. You can use this event to start code, rather than waiting for the user to click a sprite or press a key.



CHALLENGE

SPEED UP YOUR SPACESHIP

Can you make the spaceship move faster (or slower) towards the Earth?

HINT!

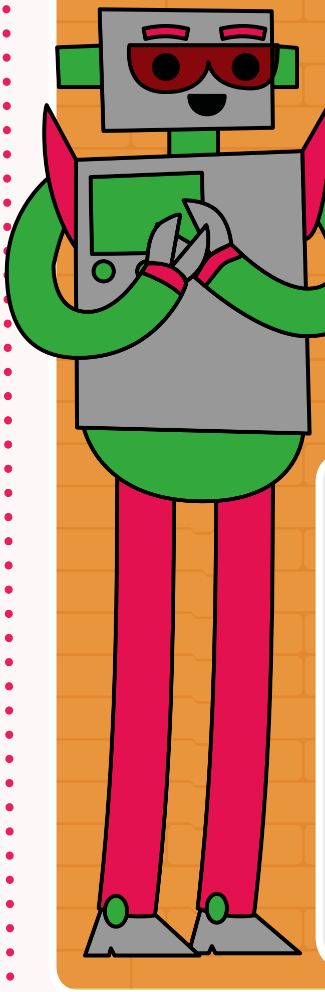
You'll need to change the number in the 'glide' block!

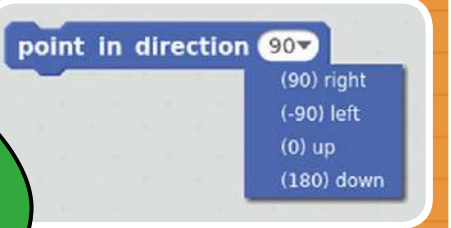
TIP! CO-ORDINATES

The numbers in the go to and glide blocks are x and y co-ordinates for setting a sprite's position on the stage. You'll learn more about co-ordinates in the 'On Target' chapter.

TIP! DIRECTION

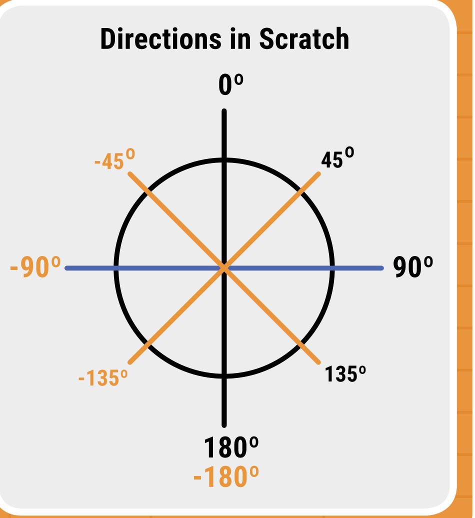
If you click the down arrow in the point in direction block, you can see that there are numbers that represent directions.





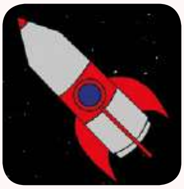
```
point in direction 90▾
  (90) right
  (-90) left
  (0) up
  (180) down
```

This number is the angle that a sprite is facing (in degrees). You can enter any number between 0 and 180 clockwise, or 0 to -180 anti-clockwise.



Directions in Scratch

What number would you need to enter in the point in direction block for the spaceship sprite to face this way?



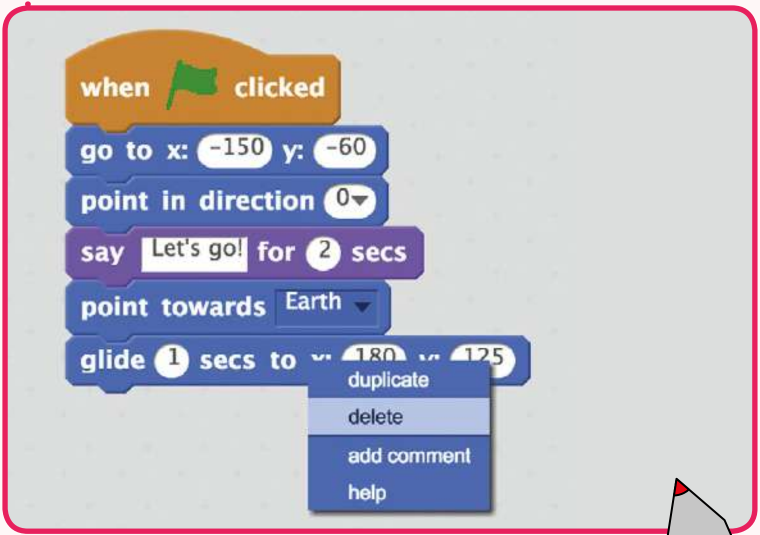
TEST YOUR PROJECT

To test your code, you can either click on the green flag just above the stage, or just click on the script itself. You should see your spaceship sprite speak, turn, and move towards the Earth.

STEP 2: ANIMATING USING LOOPS

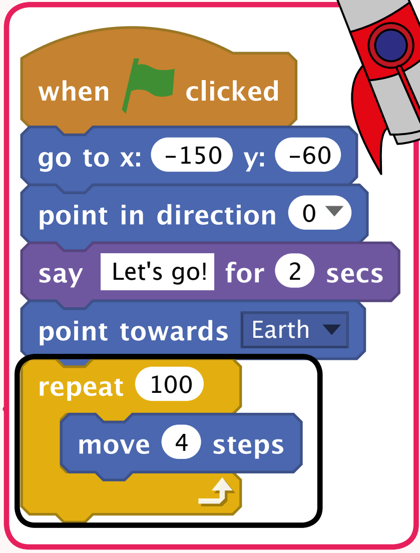
Now that you know how to write code to move sprites, let's use a 'repeat' block to create more interesting animations.

☐ Delete the glide block from your spaceship script by right-clicking on the block and clicking **delete**. You can also delete code by dragging it off the script area, back into the blocks palette on the left of the editor.



```
when [flag] clicked
go to x: -150 y: -60
point in direction 0
say Let's go! for 2 secs
point towards Earth
glide 1 secs to x: 180 y: 125
```

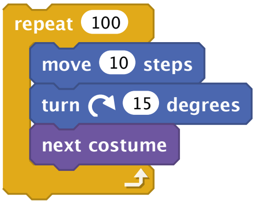
☐ Once you've removed the glide block, add a move block inside a repeat block instead. This code will move your spaceship a small amount, lots of times!



```
when [flag] clicked
go to x: -150 y: -60
point in direction 0
say Let's go! for 2 secs
point towards Earth
repeat 100
    move 4 steps
```

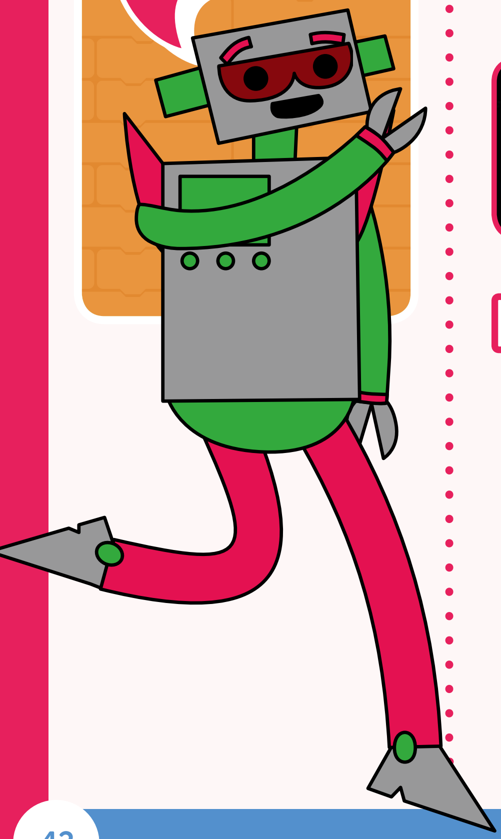
TIP!

REPEAT BLOCKS



```
repeat 100
    move 10 steps
    turn ↻ 15 degrees
    next costume
```

A repeat block runs the code inside it repeatedly, a set number of times, or until a certain condition is met. Repeating code lots of times is sometimes called a 'loop', as the code loops back to the start of the repeat block once it gets to the end. A forever block repeats the code inside it forever.

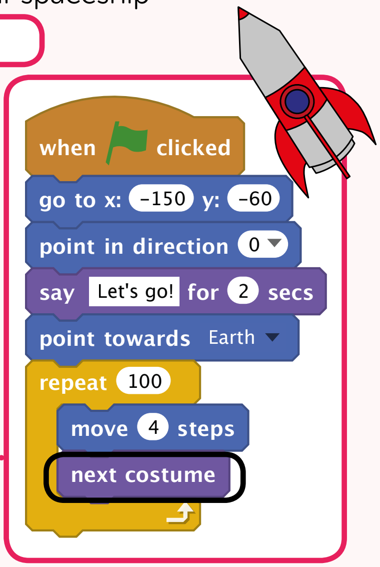


□ If you click the green flag to try out this new code, you'll see that it does pretty much the same thing as before.

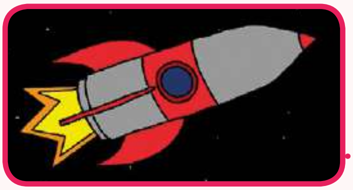
In your new code, how many times does your spaceship move? ▭

How many steps does your spaceship move each time? ▭

□ You can add more code to your loop, to change how your spaceship looks as it moves. Add the next costume block (from the Looks category), to repeatedly change the spaceship's costume as it moves.

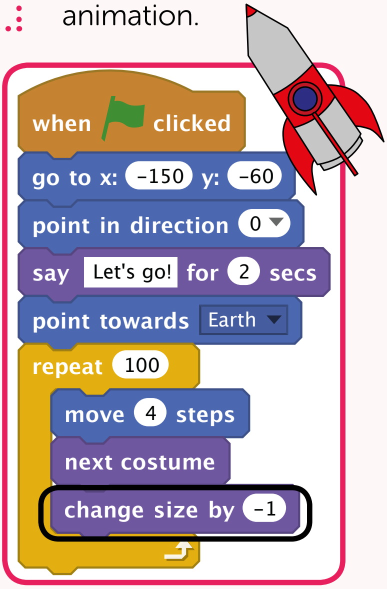


```
when 🏴 clicked
go to x: -150 y: -60
point in direction 0 ▾
say Let's go! for 2 secs
point towards Earth ▾
repeat 100
    move 4 steps
    next costume
```



□ Click the green flag to test your new animation.

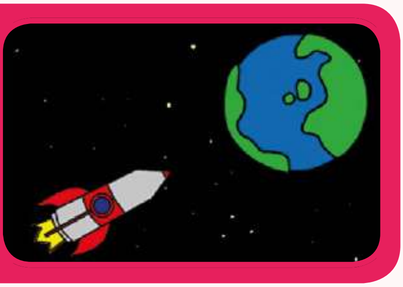
□ As well as changing the spaceship's costume, you could also make it appear to get smaller as it moves towards the Earth.



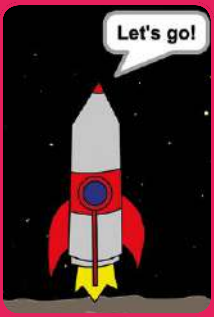
```
when 🏴 clicked
go to x: -150 y: -60
point in direction 0 ▾
say Let's go! for 2 secs
point towards Earth ▾
repeat 100
    move 4 steps
    next costume
    change size by -1
```

TEST YOUR PROJECT

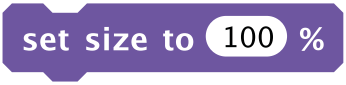
Your spaceship should slowly get smaller as it moves towards the Earth.



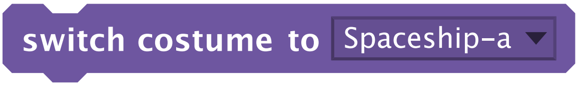
What happens if you click the flag a second time? Does your spaceship start the right size? You might also notice that sometimes your spaceship starts out using the wrong costume.



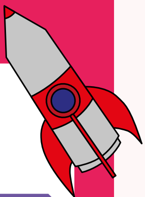
Can you add these blocks to the start of your animation to fix the problem?



```
set size to 100 %
```



```
switch costume to Spaceship-a
```



DEBUG

DEBUG YOUR CODE

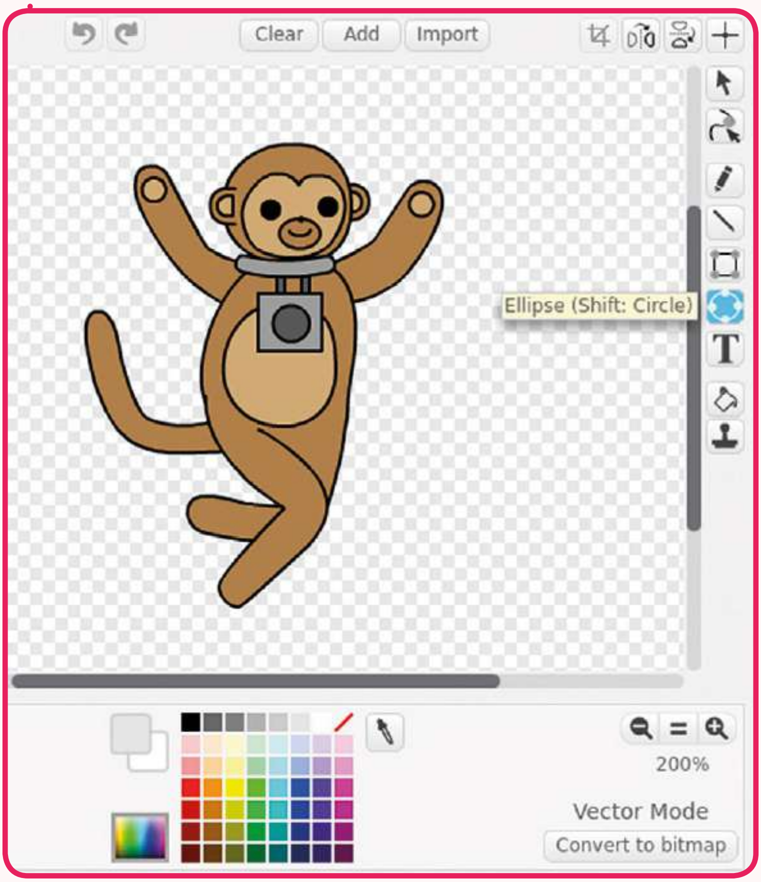
Problems with your code are called ‘bugs’, and spotting and fixing those problems is known as ‘debugging’. When writing code, you might often find that your projects don’t do what you want them to do first time.

Having a bug in your code is nothing to worry about – it happens to programmers all the time! In fact, fixing bugs is a great time to learn more about coding and how your project works.

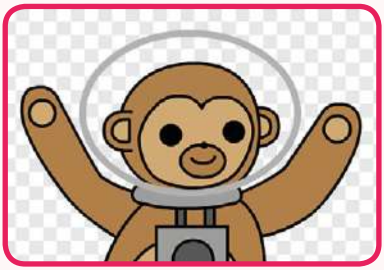
STEP 3: FLOATING MONKEY

Now we'll add a monkey to your animation, who's lost in space!

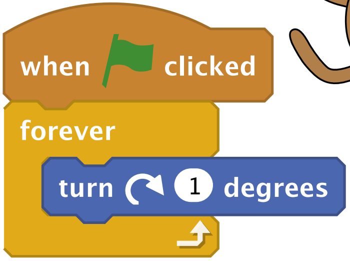
□ Let's start by making the monkey look more like an astronaut! Click on the Monkey sprite and then click the **Costumes** tab. Click the **Ellipse** tool in the paint editor and choose a colour which will show up against the stage backdrop.



□ Use the Ellipse tool to draw a space helmet around the monkey's head, by clicking and dragging the mouse.



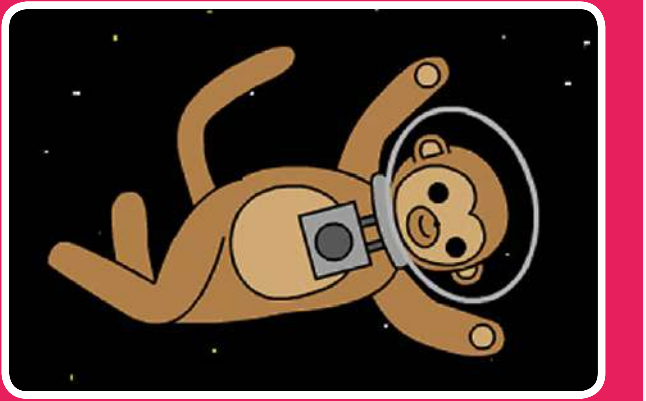
☑ Next, click the **Scripts** tab and add code to the monkey, so that it spins slowly in a circle forever.



```
when 🏴 clicked
forever
    turn ↻ 1 degrees
```

☑ TEST YOUR PROJECT

Click the flag to test your monkey sprite.



As you've coded the animation to run forever, you'll have to click the red stop button (next to the green flag) to stop this animation.

CHALLENGE

IMPROVE YOUR MONKEY ANIMATION

Can you make your monkey sprite spin faster?

Can you make the sprite get smaller as it spins, so that it looks as though it's floating away?

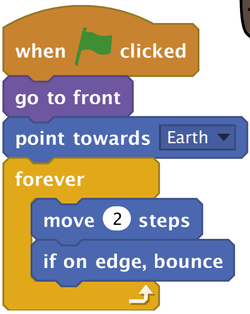
HINT!

Change the number in the 'turn' block to change the speed at which the monkey spins, and use a 'change size' block to make the monkey get smaller, just as you did with your spaceship.

STEP 4: BOUNCING ASTEROIDS

Let's add some floating space-rock to your animation.

☑ Click on the Asteroid sprite and add this code to make the asteroid bounce around the screen.

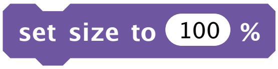


```
when 🏴 clicked
go to front
point towards Earth ▾
forever
    move 2 steps
    if on edge, bounce
```

DEBUG

DEBUGGING YOUR STAR SPRITE

If your star sprite ends up getting too big or too small, you can add a 'set size' block at the start of your script to reset its size.



```
set size to 100 %
```

CHALLENGE

MAKE YOUR OWN ANIMATION

After you've finished your space animation, click File and then New, to start a new project.

Use what you've learnt in this project to make your own animation. It can be anything you like, but try to make your animation match the setting.

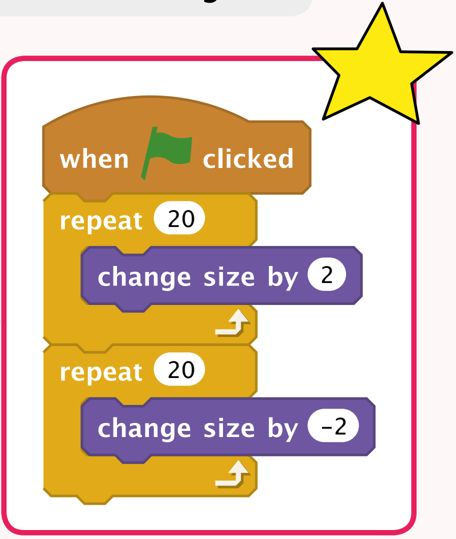
TEST YOUR CODE

If you click the green flag to test your asteroid animation, you should see it bounce around the stage.

STEP 5: SHINING STAR

Let's combine loops to make a shining star.

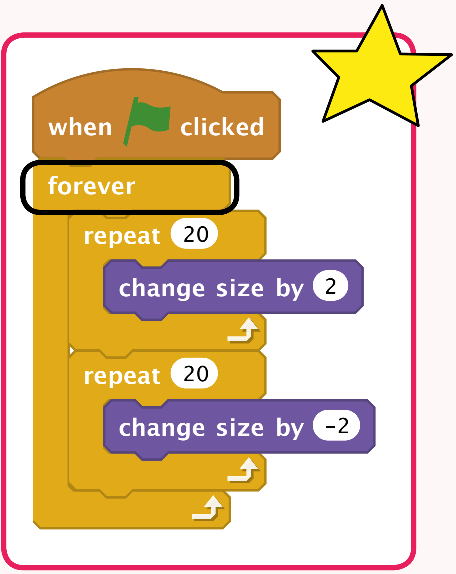
□ Click on the Star sprite and add this code to make the star slowly get larger and then smaller again.



```
when [green flag] clicked
repeat 20
    change size by 2
repeat 20
    change size by -2
```

□ Test your code; your star sprite should slowly get larger and then smaller.

□ To make the star change size repeatedly, you can add a forever block around the code.

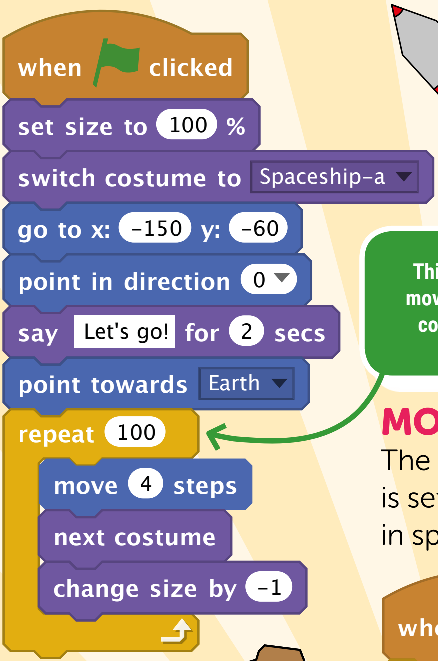


```
when [green flag] clicked
forever
    repeat 20
        change size by 2
    repeat 20
        change size by -2
```

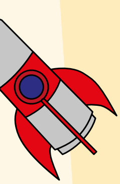
LOST IN SPACE FULL CODE LISTING

SPACESHIP

The spaceship launches and then heads for Earth.



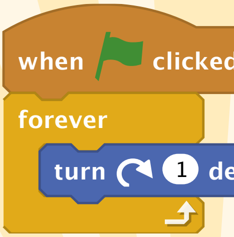
```
when ⚑ clicked
set size to (100) %
switch costume to [Spaceship-a ▼]
go to x: (-150) y: (-60)
point in direction (0 ▼)
say [Let's go!] for (2) secs
point towards [Earth ▼]
repeat (100)
    move (4) steps
    next costume
    change size by (-1)
```



This loop makes the spaceship move repeatedly while switching costumes and getting smaller

MONKEY

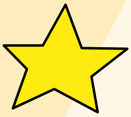
The astronaut monkey is set to spin forever in space!

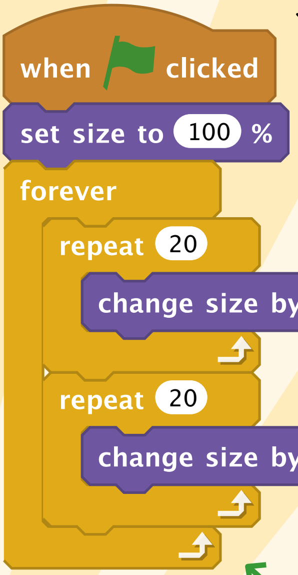


```
when ⚑ clicked
forever
    turn ↻ (1) degrees
```

STAR

The star twinkles in the night sky.



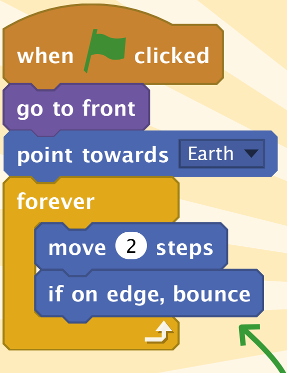


```
when ⚑ clicked
set size to (100) %
forever
    repeat (20)
        change size by (2)
    repeat (20)
        change size by (-2)
```

Two repeat loops cause the star to get bigger, then smaller again

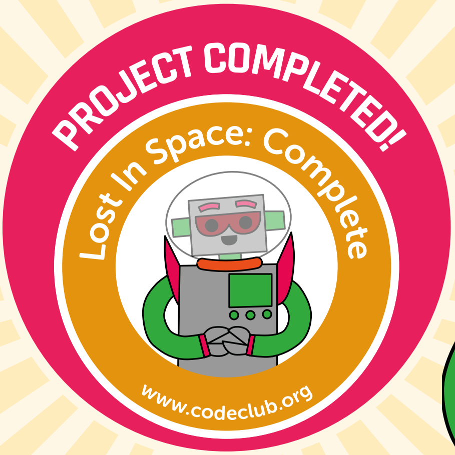
ASTEROID

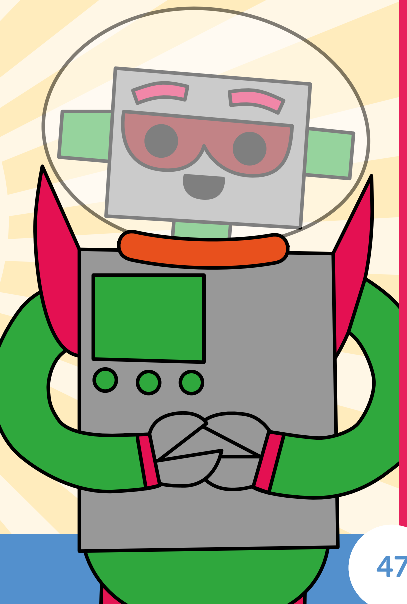
This piece of space-rock bounces around the screen.



```
when ⚑ clicked
go to front
point towards [Earth ▼]
forever
    move (2) steps
    if on edge, bounce
```

Whenever the sprite hits the edge of the stage, it'll bounce off



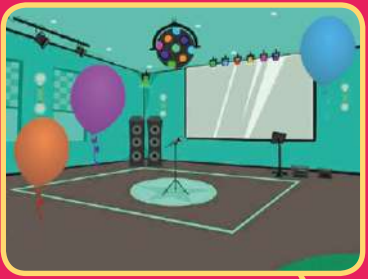


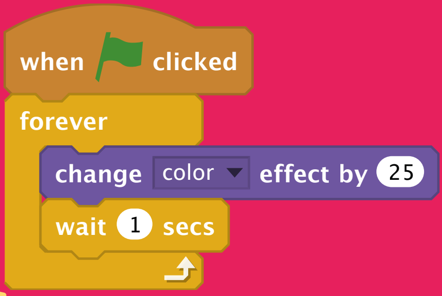
Now You Could Make...

With the skills you've learnt, why not try these projects?

PARTY

Animate balloons and create multicoloured disco lights. You could even create some party music.

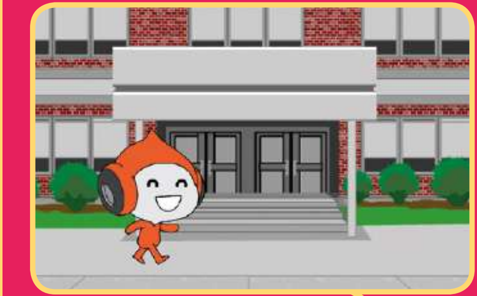


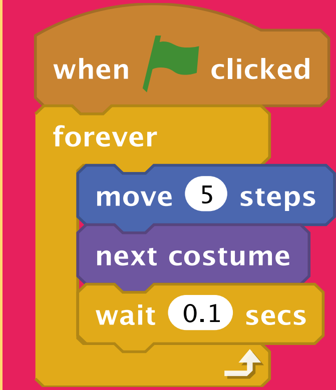


```
when ⚑ clicked
forever
    change color ▾ effect by 25
    wait 1 secs
```

WALKING SPRITES

Some sprites, such as 'Pico walking', have a set of costumes for creating a walking animation.

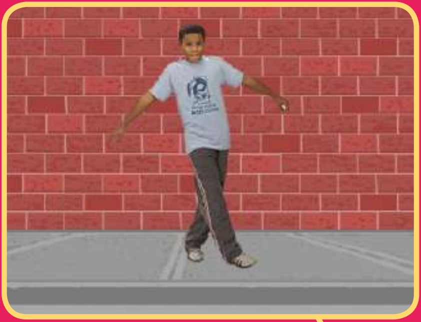


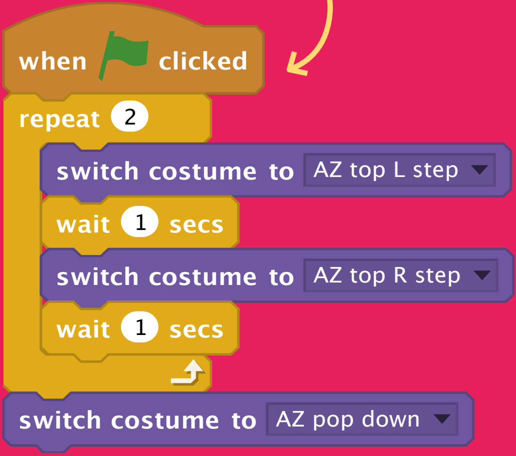


```
when ⚑ clicked
forever
    move 5 steps
    next costume
    wait 0.1 secs
```

DANCE MOVES

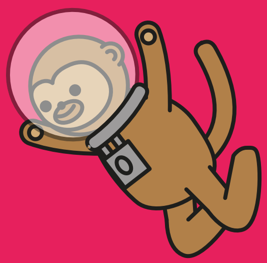
Code a sprite to dance along to some music by changing costumes and moving around the stage.





```
when ⚑ clicked
repeat 2
    switch costume to AZ top L step ▾
    wait 1 secs
    switch costume to AZ top R step ▾
    wait 1 secs
switch costume to AZ pop down ▾
```

Lost In Space



Answers on p110

Can you find all the words in the grid, including a lost monkey?

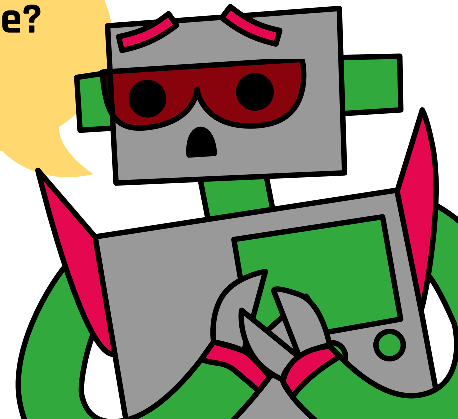
B	E	O	J	E	A	E	C	A	T	E	S	P	S	R
R	B	Y	E	U	R	P	O	A	S	R	S	O	C	T
A	B	S	A	M	P	O	M	T	L	T	A	O	R	T
P	E	C	R	B	A	I	E	S	E	N	R	E	L	U
T	A	G	N	M	L	O	T	C	A	A	P	D	I	N
A	V	S	T	M	O	N	K	E	Y	T	P	I	T	E
O	M	O	O	U	M	I	J	R	R	E	U	O	S	U
A	L	O	T	P	N	E	G	K	U	C	E	R	O	T
N	N	E	B	U	L	A	U	A	C	L	K	E	N	R
X	O	X	E	T	L	A	U	T	R	I	O	T	E	M
G	A	R	K	A	V	O	N	R	E	P	U	S	U	R
Y	R	E	X	R	O	E	T	E	M	S	T	A	R	E
R	A	Y	P	E	E	X	A	U	T	E	K	C	O	R
T	C	A	E	T	E	C	K	U	U	S	C	K	P	E
T	Y	R	Y	M	S	S	G	R	L	O	O	E	N	E

WORDS TO FIND

ASTEROID
COMET
ECLIPSE
GALAXY
JUPITER
MERCURY
METEOR
MONKEY
MOON
NEBULA
PLANET
ROCKET
SATURN
STAR
SUPERNOVA

Want to code a ghoulish game?

Turn the page if you dare...

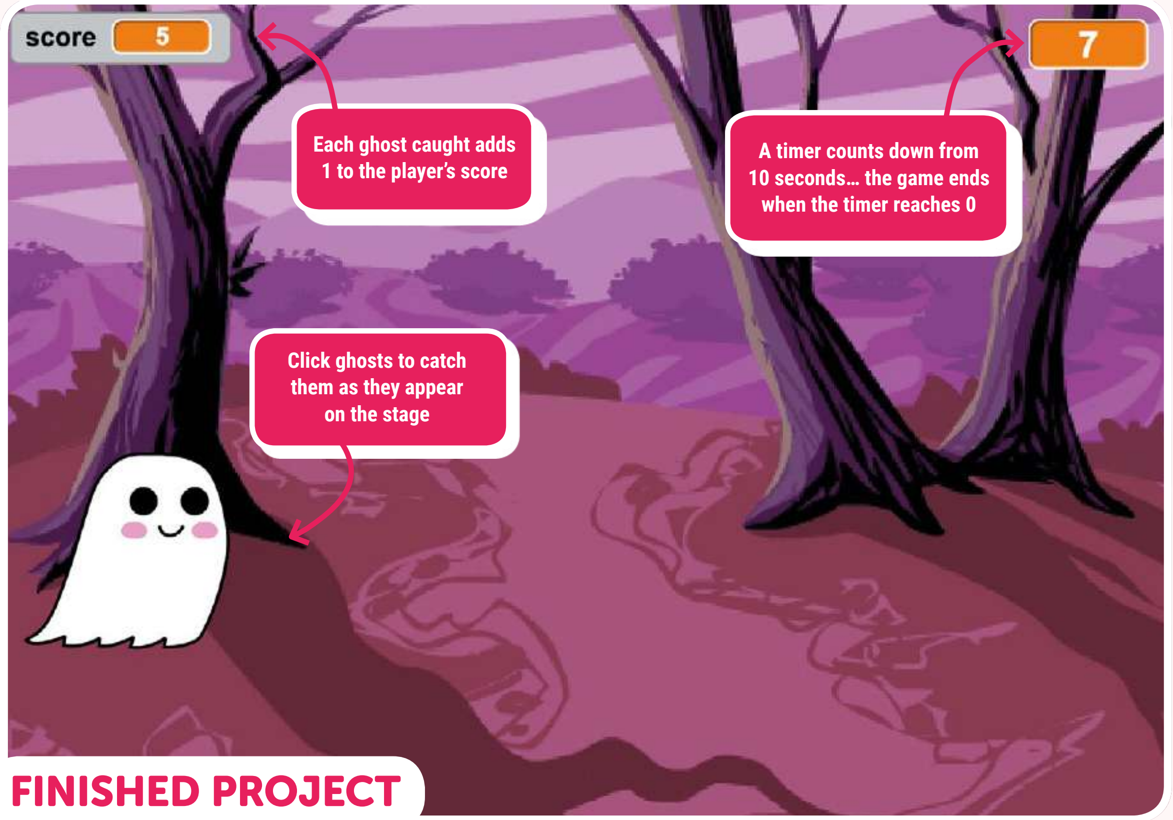


Ghost Catcher

Create a ghost-catching game, in which players score points by clicking on sprites as they move around the stage

Let’s go catch some ghosts!
Have fun making your own spooky game!

You’ll make use of a ‘variable’ to keep track of the player’s score as they gain (and lose) points. You’ll also create a timer, so that players are in a race against time.



Each ghost caught adds 1 to the player's score

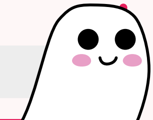
A timer counts down from 10 seconds… the game ends when the timer reaches 0

Click ghosts to catch them as they appear on the stage

FINISHED PROJECT

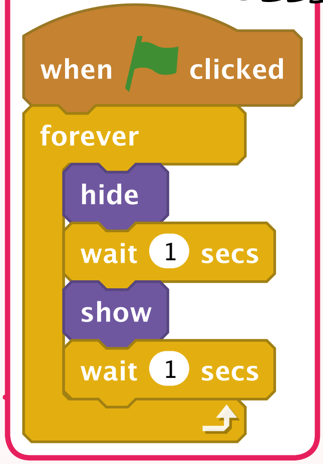
STEP 1: ANIMATE A GHOST

Let's start by animating a ghost.



☐ Open a web browser and go to **rpf.io/book-ghostcatcher** to open the Ghost Catcher project.

☐ Click on the Ghost sprite, and add code to make it repeatedly appear and disappear forever.



```
when 🏴 clicked
forever
    hide
    wait 1 secs
    show
    wait 1 secs
```

TIP! PROJECT FILES

To download a zip file of all the Scratch 2 (.sb2) project assets files for this book, go to:

rpf.io/book-s1-assets

WHAT YOU'LL LEARN

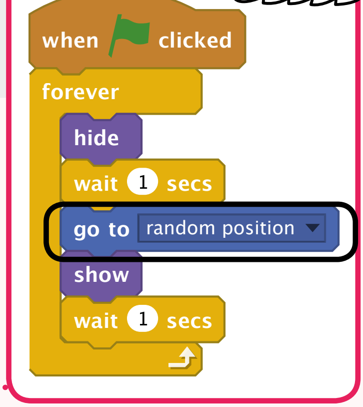
- Variables
- Random numbers

☑ TEST YOUR PROJECT

Click the green flag to test your code. You should see your ghost appear and disappear every second.

STEP 2: RANDOM GHOSTS

Move your ghost around the stage, so that it's harder to catch!

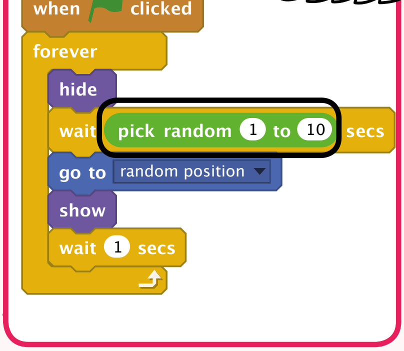


```
when 🏁 clicked
forever
    hide
    wait 1 secs
    go to random position
    show
    wait 1 secs
```

☑ Instead of staying in the same position, you can let Scratch choose a random position for the ghost sprite before it appears each time.

☑ Test your code. Does your ghost sprite move around the stage?

☑ Your ghost always waits exactly 1 second before appearing and disappearing. To change this, grab a **pick random** block from the green Operators category and place it inside the first **wait** block, replacing the 1.



```
when 🏁 clicked
forever
    hide
    wait (pick random 1 to 10) secs
    go to random position
    show
    wait 1 secs
```

CHALLENGE

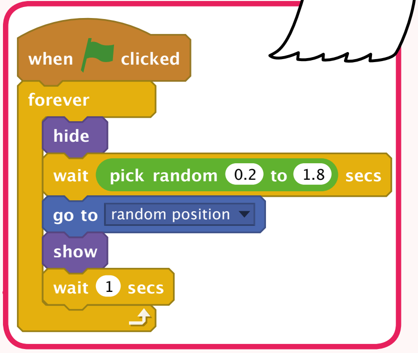
MORE RANDOMNESS

Can you make your ghost appear on the screen for a random amount of time? Can you make the ghost a random size each time it appears?

HINT!

You'll need to add another 'random' block to your second 'wait' block. For a random ghost size, add a 'random' block to a 'set size to' block.

☑ Your ghost will now wait anywhere between 1 and 10 seconds before appearing, which is a long time! Change the numbers in your **pick random** block until you're happy with how often your ghost appears.

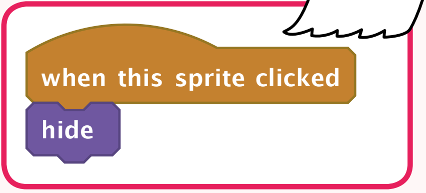


```
when ⚑ clicked
forever
    hide
    wait (pick random (0.2) to (1.8)) secs
    go to [random position ▾]
    show
    wait (1) secs
```

STEP 3: CATCHING GHOSTS

Let's allow the player to catch ghosts!

☑ Add code to allow the player to catch a ghost.

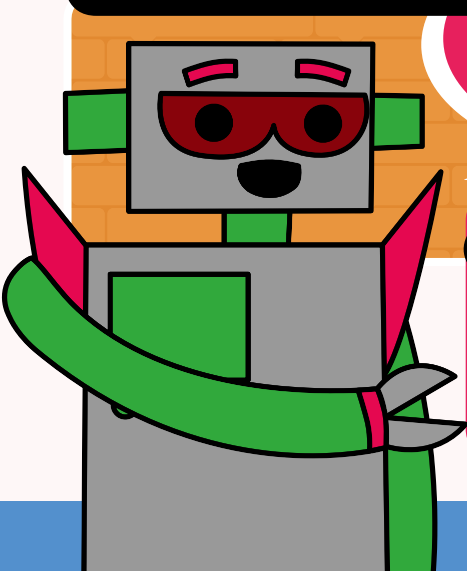


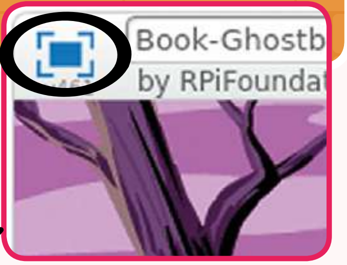
```
when this sprite clicked
hide
```

☑ Test out your project. Can you catch ghosts as they appear on the stage?

TIP! FULL-SCREEN MODE

If you find it difficult to catch the ghosts, you can play the game in full-screen mode by clicking the button above the stage.





CHALLENGE

ADD A SOUND

Can you play a sound each time a ghost is caught?

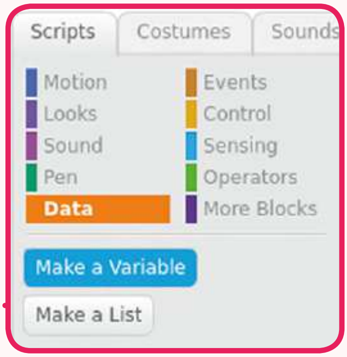
HINT!

You'll need to add a 'play sound' block to your 'when this sprite clicked' script.

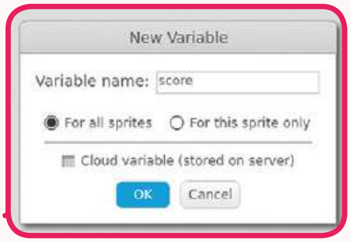
STEP 4: ADD A SCORE

Let's make things more interesting by keeping score.

☐ To keep the player's score, you'll need to create a variable. Click the bright orange **Data** category in the blocks palette and then click **Make a Variable**.



☐ Type **score** as the name of the variable, make sure that it is available for all sprites, and click **OK** to create it.

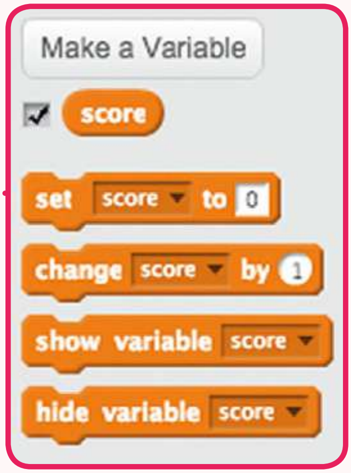

New Variable

Variable name: score

● For all sprites ○ For this sprite only

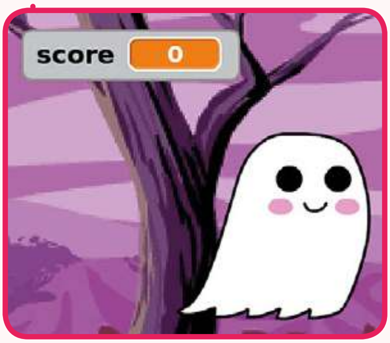
☐ Cloud variable (stored on server)

☐ You should now see lots of code blocks that can be used with your score variable.

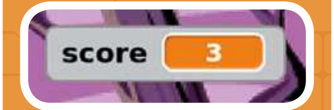


```
☑ score
set score ▾ to 0
change score ▾ by 1
show variable score ▾
hide variable score ▾
```

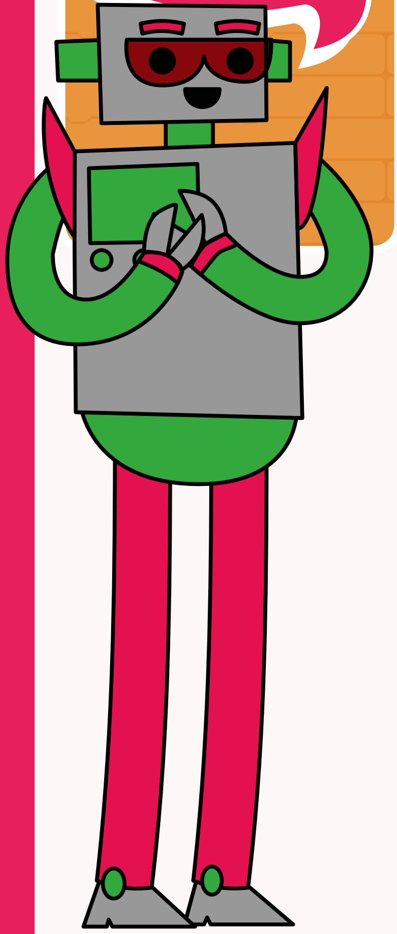
☐ You'll also see the score in the top-left of the stage.



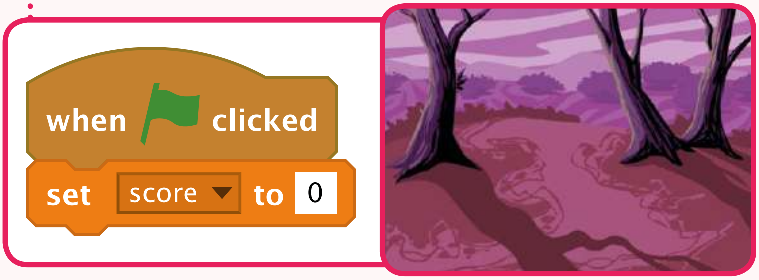
TIP! VARIABLES



A variable is a place in a computer's memory to store data, such as numbers or text. Each variable is given a name, so that the stored data can be accessed and changed later.

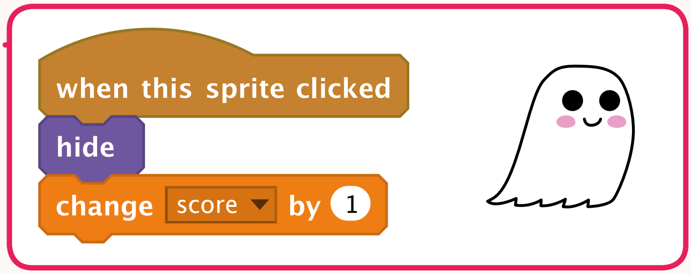


□ When a new game is started (by clicking the flag), you should set the player's score to 0. Add this code to the Stage in order to set the score at the start of the game.



```
when 🏴 clicked
set score ▾ to 0
```

□ Whenever a ghost is caught, you need to add 1 to the player's score. Add this code to your Ghost sprite.



```
when this sprite clicked
hide
change score ▾ by 1
```

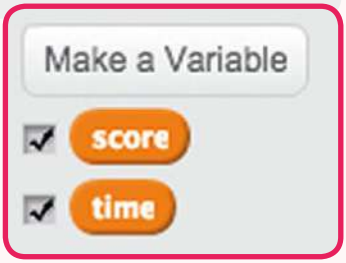
□ TEST YOUR PROJECT

Test your program and try to catch some ghosts. Does your score change each time you click on a ghost?

STEP 5: ADD A TIMER

You can make your game more interesting, by only giving your player 10 seconds to catch as many ghosts as possible.

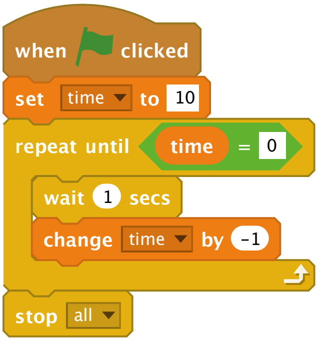
☐ You can use another variable to store the remaining time left. Make a new variable called time.



☐ This is how the timer should work:

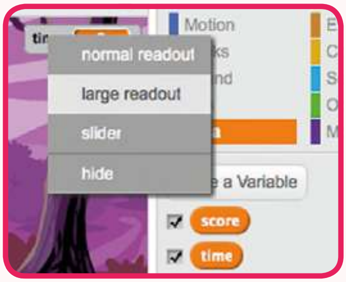
- The timer should start at 10 seconds;
- The timer should count down every second;
- The game should stop when the timer gets to 0.

Add the following new script to your Stage. The ☐ = ☐ block is found in the Operators category.



```
when ⚑ clicked
set time ▼ to 10
repeat until time = 0
    wait 1 secs
    change time ▼ by -1
stop all ▼
```

☐ Drag your time variable display to the right side of the stage. You can also right-click on the variable display and choose **large readout** to change how the time is displayed.

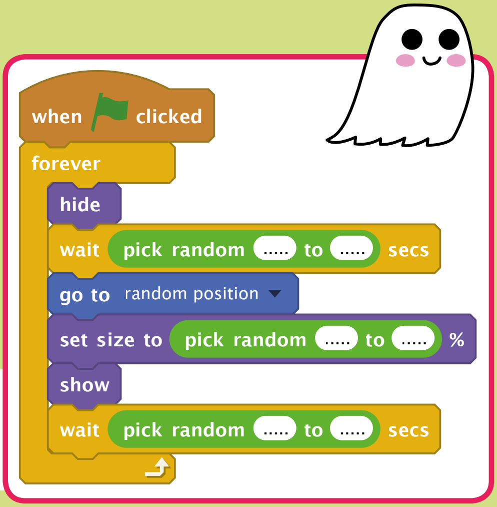


CHALLENGE

MORE RANDOMNESS

Ask a friend to test your game. Change the numbers in your game if they found it too easy or too hard.

What numbers did you decide on?



```
when [flag] clicked
forever
    hide
    wait (pick random ..... to .....) secs
    go to random position ▾
    set size to (pick random ..... to .....) %
    show
    wait (pick random ..... to .....) secs
```

HINT!

Give the player less time.

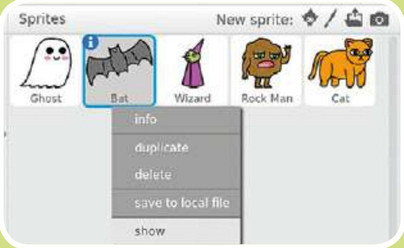
Make the ghosts appear less often.

Make the ghosts smaller.

CHALLENGE

MORE OBJECTS

Can you add other objects to your game? You can right-click on the sprites in the sprite list and click ‘show’ to make them appear on the stage. You don’t have to use those sprites, though: you can add any other sprites you want from the Scratch library.



Before you get started, you could complete the table below.

	What size will it be?	How often will it appear?	What happens when it has been caught?	How many points will you score (or lose) for catching it?
GHOST	Between 40% and 80%	Between every 0.2 and 1.8 seconds	Plays a 'pop' sound	1 point scored

Enter the Crypt!

Solve the fiendish cryptic clues to find monsters. Place them in the grid to reveal another ghastly creature in the shaded squares.

Answers on p110

1 ▨□□□□

2 □□▨□□□□□

3 □▨□□□

4 □□▨□□

5 □□□▨□

6 □□□▨□

7 □□▨□□□

The hidden creature is a...

....

CLUES

1 Charming host conceals apparition

2 We're wolfing down food, hairy howler

3 Evil spirit hidden in crude montage

4 Mum, my ancient Egyptian is bandaged

5 Ugly cave dweller takes a stroll outdoors

6 Rude, vile rascal with horns!

7 'I've got a bun! Yippee!' yelled Australian swamp monster

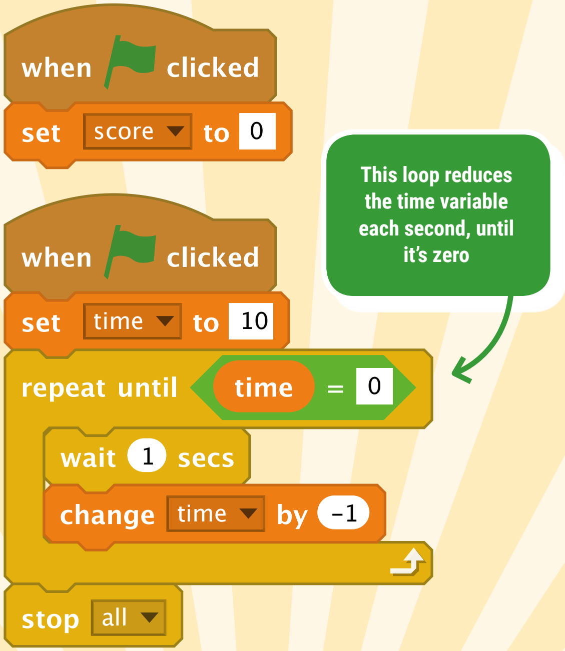
HINT!

All the answers are concealed within the clues – look carefully and you'll find them!

GHOST CATCHER FULL CODE LISTING

STAGE

The Stage scripts reset the score to zero and handle the timer.

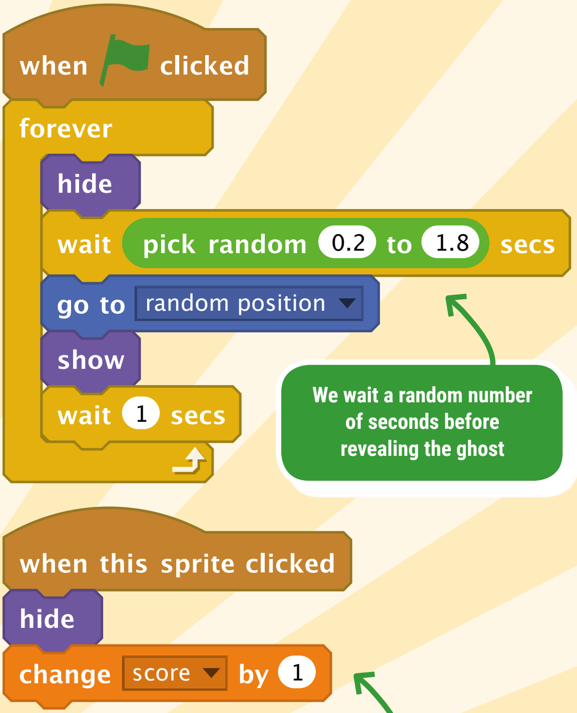


```
when flag clicked
set score to 0

when flag clicked
set time to 10
repeat until time = 0
    wait 1 secs
    change time by -1
stop all
```

This loop reduces the time variable each second, until it's zero

GHOST

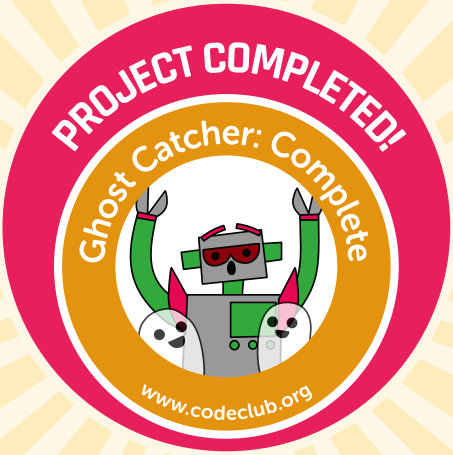
The Ghost sprite has two scripts: one to make it appear in a random position, and another for the player to 'catch' it.



```
when flag clicked
forever
    hide
    wait pick random 0.2 to 1.8 secs
    go to random position
    show
    wait 1 secs

when this sprite clicked
hide
change score by 1
```

We wait a random number of seconds before revealing the ghost

When the player clicks the ghost, 1 is added to the score

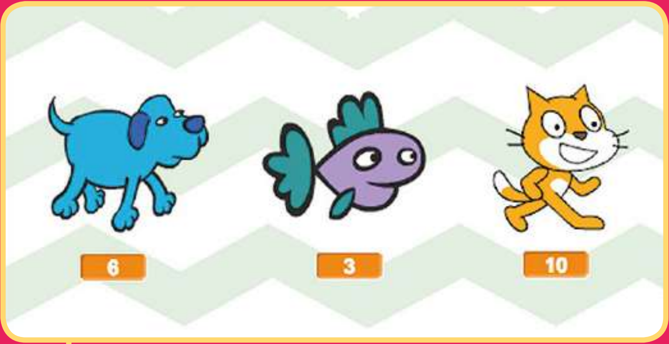


Now You Could Make...

With your new coding skills, you could try these projects...

VOTING APP

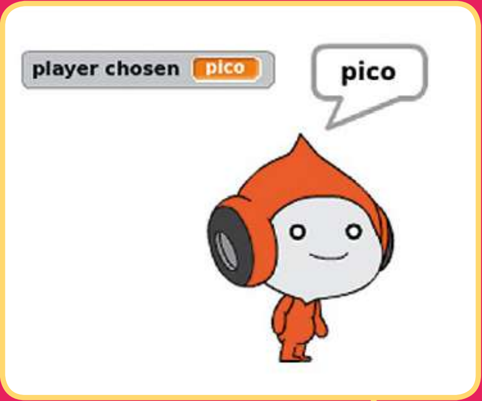
Create a sprite and a variable for each choice, and let your friends vote on their favourite! You could even add a reset button to set the votes back to zero.



```
when this sprite clicked
change [dog votes] by (1)
set [fisheye] effect to (50)
play sound [dog1] until done
set [fisheye] effect to (0)
```

PLAYER CHOOSER

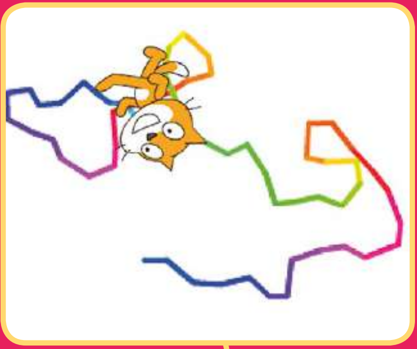
Allow players to randomly choose a character by randomly changing its costume when the sprite is clicked.



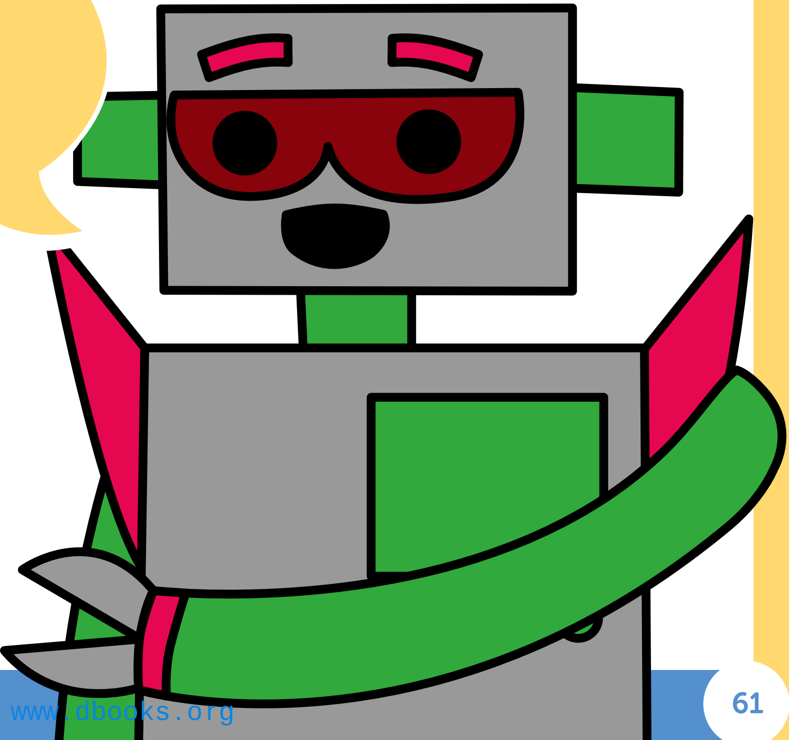
```
when this sprite clicked
switch costume to (pick random (1) to (4))
set [player chosen] to ([costume name] of [Player])
say (player chosen) for (2) secs
```

RANDOM ART

Use pick random blocks with blocks from the Pen category to create unique works of art!



```
when this sprite clicked
pen down
repeat (pick random (40) to (100))
    move (pick random (10) to (20)) steps
    turn ↻ (pick random (-90) to (90)) degrees
    change pen color by (pick random (1) to (10))
pen up
```

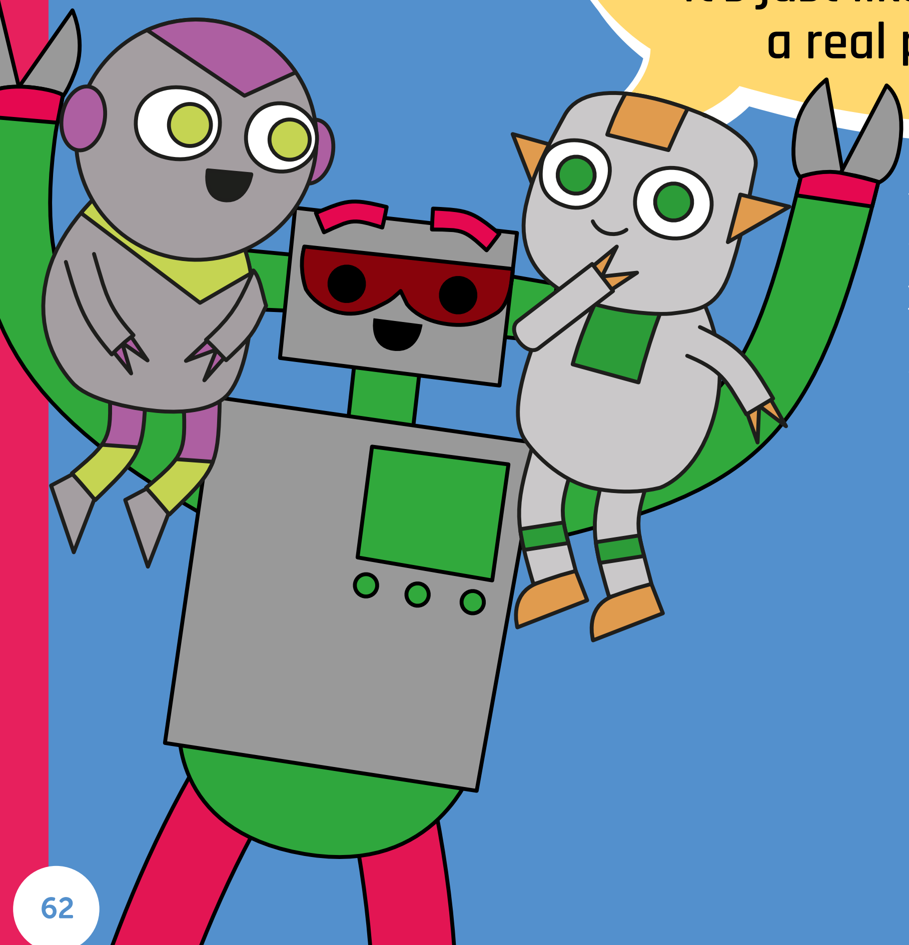


Need to talk to someone?

Turn the page to create a chatbot...

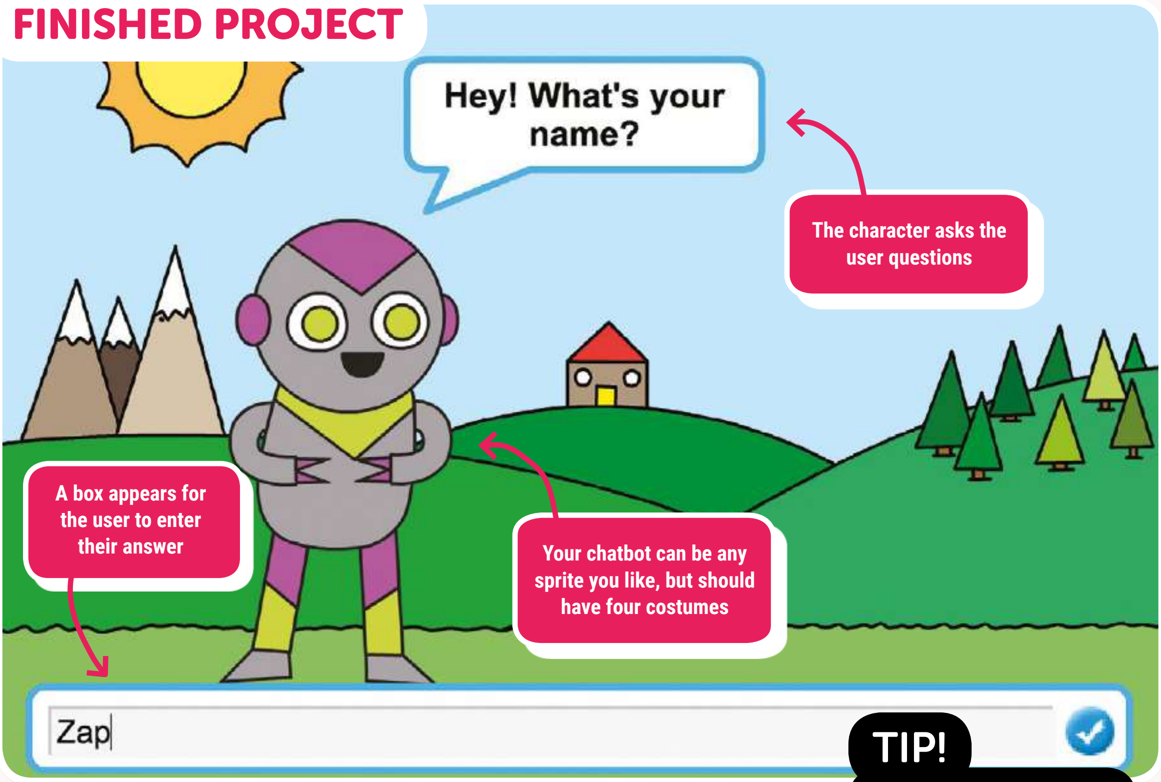
Chatbot

Create your own talking character that asks questions and responds to the answers you give it


Program your own chatbot!
It's just like talking to a real person!

You will learn how to add 'selection' to your code by using **if** and **if...else** blocks to change how your character responds, depending on the answers given.

FINISHED PROJECT



The character asks the user questions

A box appears for the user to enter their answer

Your chatbot can be any sprite you like, but should have four costumes

STEP 1: YOUR CHATBOT

Choose your character's personality and look.

Before you start making your chatbot, you need to decide on its personality. Think about:

- What is their name?

  ....

- Where do they live?

  ....

- Are they happy? serious? funny? shy? friendly?

  ....

- What do they like and dislike?

  ....

TIP!

PROJECT FILES

To download a zip file of all the Scratch 2 (.sb2) project assets files for this book, go to:

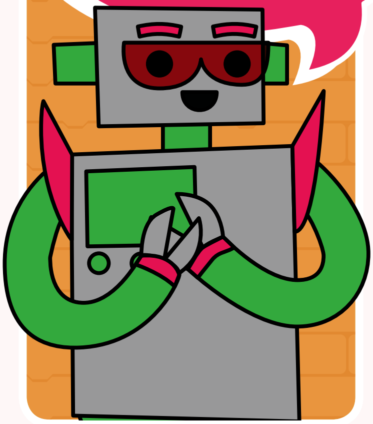
rpf.io/book-s1-assets

WHAT YOU'LL LEARN

- **Selection (if and if...else blocks)**
- **Keyboard input using the ask block**
- **Using the join block to join text together**

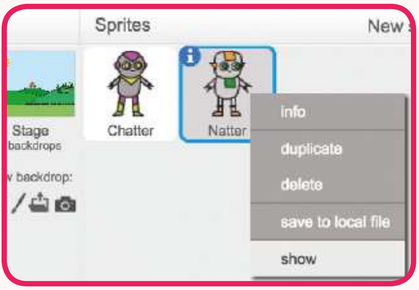
TIP! CHOOSING YOUR OWN SPRITE

If you prefer, you can choose a different sprite from the Scratch library (or even draw your own). For this project, the sprite you use should have four costumes, such as the sprites above.

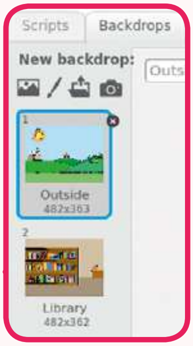


☐ Open a web browser and go to **rpf.io/book-chatbot** to open the Chatbot project. Click the Remix button.

☐ There are two characters in the sprite list: Chatter and Natter. If you prefer to use the Natter sprite, then you can right-click and **show** the sprite. You can also right-click to **hide** the Chatter sprite.



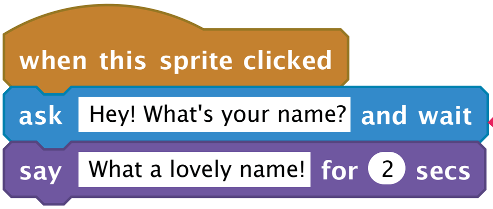
☐ Choose a stage backdrop to match your chatbot's personality. There are already two to choose from, or you can select a different backdrop from the Scratch library. We're sticking with the Outside backdrop



STEP 2: A TALKING CHATBOT

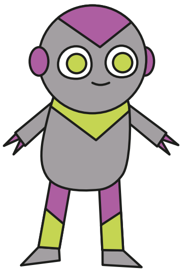
Now that you have a chatbot with a personality, let's program it to talk to you.

☐ Click on your chatbot character, and add this code:

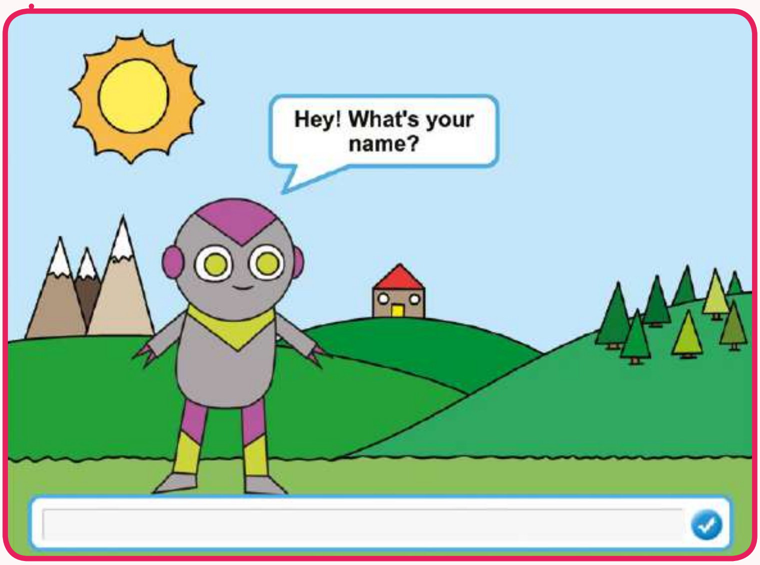


```
when this sprite clicked
ask [Hey! What's your name?] and wait
say [What a lovely name!] for (2) secs
```

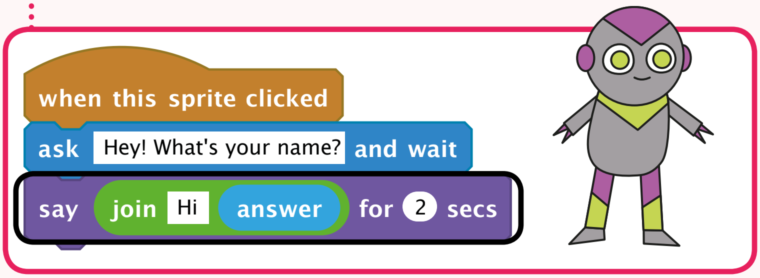
The ask block waits for the user to enter an answer



☑ Click your chatbot to test it out. When you are asked your name, type it into the box along the bottom of the stage.



☑ Your chatbot simply replies 'What a lovely name!' every time. You can personalise your chatbot's reply, by making use of the user's answer. Change the chatbot's code, so that it looks like this:



```
when this sprite clicked
ask [Hey! What's your name?] and wait
say (join [Hi] (answer)) for (2) secs
```

DEBUG

DOES IT WORK?

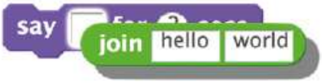
Test out this new program. Does it work as you expected? Can you fix any problems that you can see?

HINT!

You can try adding in a space somewhere!

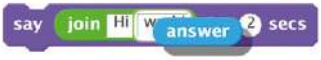
TIP! COMBINING BLOCKS

To create that last block in the script, you'll need to first drag a green join block, and drop it on to the say block.



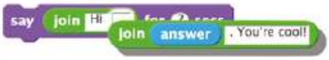
```
say (join [hello] [world]) for (2) secs
```

You can then change the text 'hello' to say Hi, and drag the light blue answer block (from the Sensing category) onto the text 'world'.



```
say (join [Hi] (answer)) for (2) secs
```

If you want to add text after the answer, you can use another join block inside the second field of the first one.

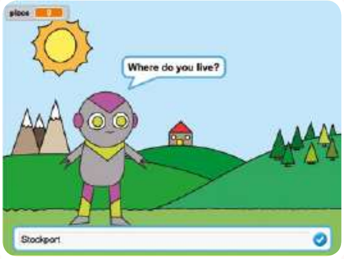


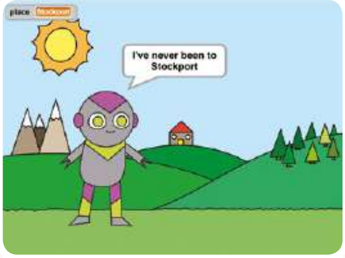
```
say (join [Hi] (join (answer) [. You're cool!])) for (2) secs
```

CHALLENGE

MORE QUESTIONS

Can you code your chatbot to ask another question? Can you store their answer in a variable?

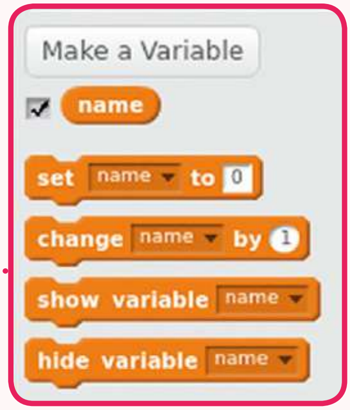




HINT!

You'll need to use another 'ask' block to ask another question, and another variable to store the answer.

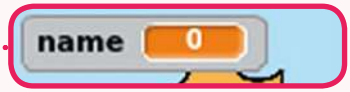
☐ If you store the answer in a variable, you'll be able to make use of it throughout your project. Create a new variable called **name**.


Make a Variable

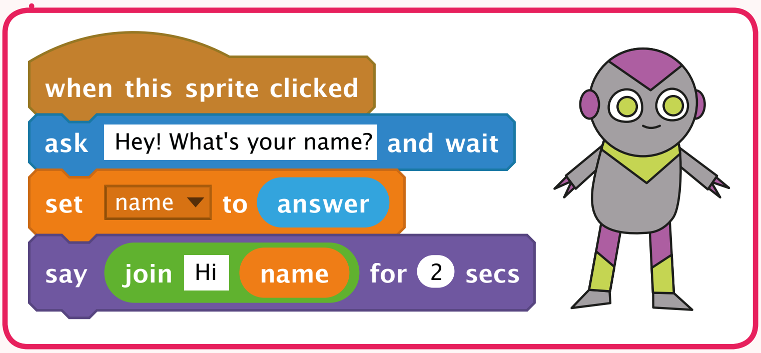
☑ name

```
set name ▾ to 0
change name ▾ by 1
show variable name ▾
hide variable name ▾
```

You should also see your new variable in the top-left of the stage.


name 0

☐ Once you've created your new variable, edit your chatbot's code to look like this:



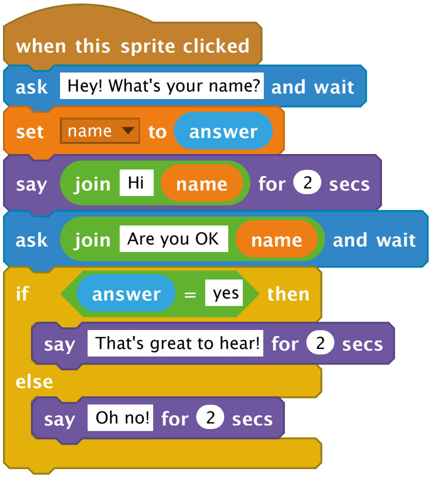
```
when this sprite clicked
ask Hey! What's your name? and wait
set name ▾ to answer
say join Hi name for 2 secs
```

☐ If you test your program again, you'll notice that the answer is stored in the **name** variable, and is shown in the top-left of the stage. (To hide this, just untick the tick-box next to **name** in the blocks palette.)

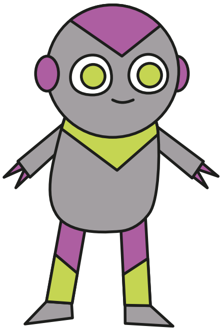
STEP 3: MAKING DECISIONS

You can program your chatbot to decide what to do, based on the user's responses.

☐ Let's get your chatbot to ask the user a question which has a yes or no answer. Here's an example, but you can change the question if you like:



```
when this sprite clicked
ask [Hey! What's your name?] and wait
set [name] to (answer)
say (join [Hi] (name)) for (2) secs
ask (join [Are you OK] (name)) and wait
if <(answer) = [yes]> then
    say [That's great to hear!] for (2) secs
else
    say [Oh no!] for (2) secs
```

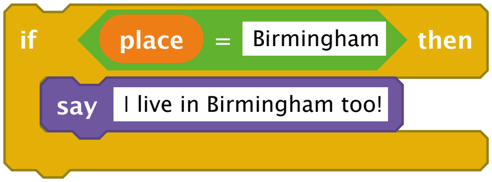


Notice that now you've stored the user's name in a variable, you can use it as much as you like.

TIP! IF AND IF... ELSE BLOCKS

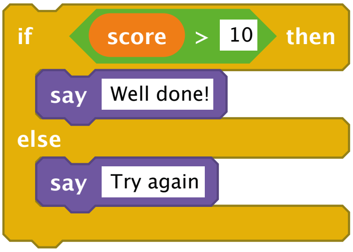
So far, the scripts you've written have performed exactly the same task each time they are run. if and if...else blocks allow your scripts to decide what to do next.

An if block includes a condition, and the code inside the if block is run only if the condition is true. If the condition is false (not true), then the code inside the if block is skipped.

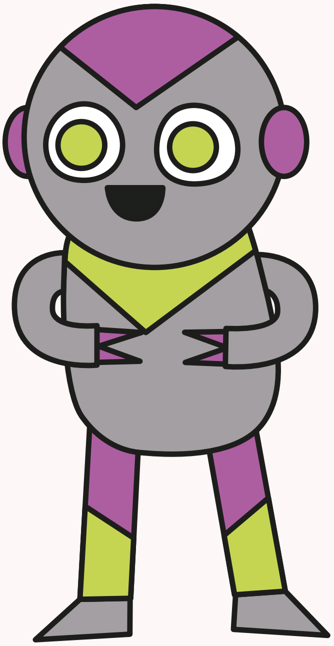


```
if <(place) = [Birmingham]> then
    say [I live in Birmingham too!]
```

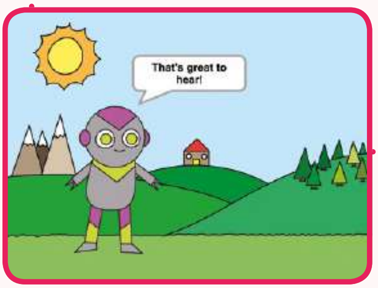
An if...else block will always run either the first or second set of blocks. If the condition is true, then the first set of blocks is run. If the condition is false, the second set of blocks is run instead.

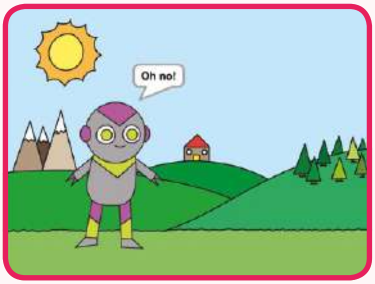


```
if <(score) > [10]> then
    say [Well done!]
else
    say [Try again]
```



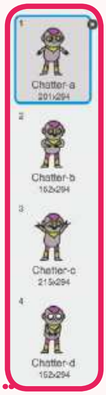
□ If you test your code, you'll now see that you get a response when you answer yes or no. Your chatbot should reply with 'That's great to hear!' when you answer yes (which is not case-sensitive), but will reply with 'Oh no!' if you type anything else.



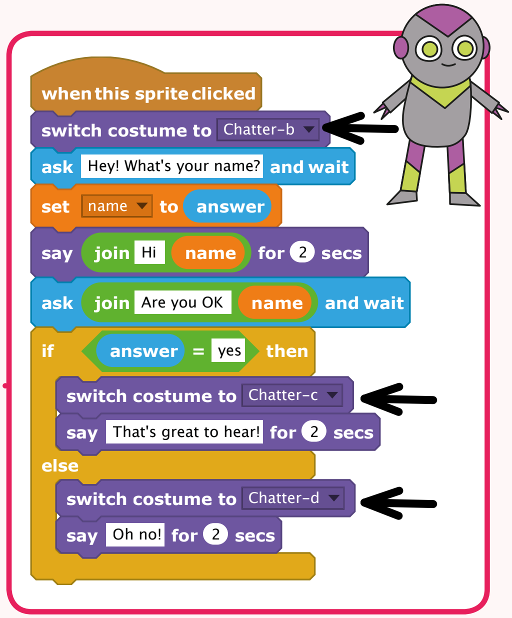


□ You can put any code inside an if or else block, not just code to make your chatbot speak. For example, you can change the chatbot's costume to match the response.

If you have a look at your chatbot's costumes, you should see that there are four of them. (If not, you can always add more yourself!)



□ You can use these costumes as part of your chatbot's response, by adding this code:

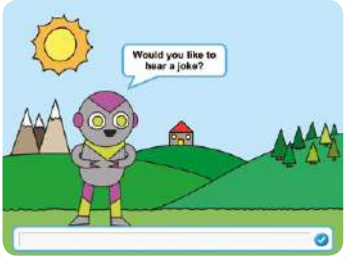


```
when this sprite clicked
switch costume to Chatter-b
ask Hey! What's your name? and wait
set name to answer
say join Hi name for 2 secs
ask join Are you OK name and wait
if answer = yes then
    switch costume to Chatter-c
    say That's great to hear! for 2 secs
else
    switch costume to Chatter-d
    say Oh no! for 2 secs
```

CHALLENGE

MORE DECISIONS

Program your chatbot to ask another question – something with a yes or no answer. Can you make your chatbot respond to the answer?

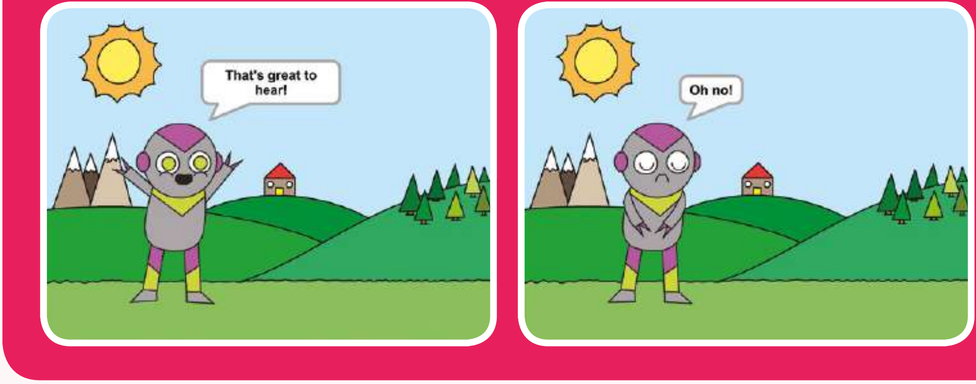


HINT!

You'll need to add another 'ask' block, with another 'if...else' block to respond to the answer.

☑ TEST YOUR PROJECT

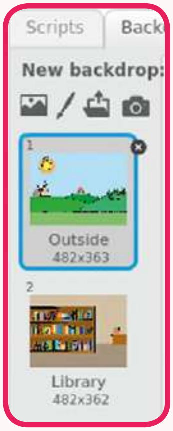
Test out your program and you should see your chatbot's face change depending on the answer you give.



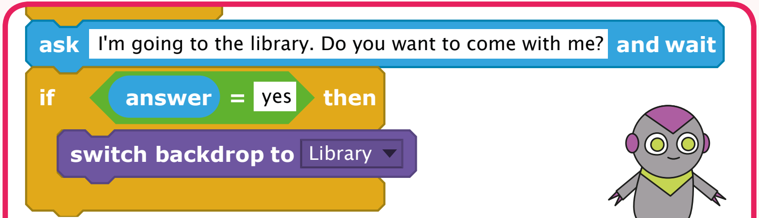
STEP 4: CHANGING LOCATION

You can also program your chatbot to change its location.

☑ Click on your stage and then click the **Backdrops** tab. You should see that your stage has two backdrops. Add another backdrop to your stage if you can only see one.

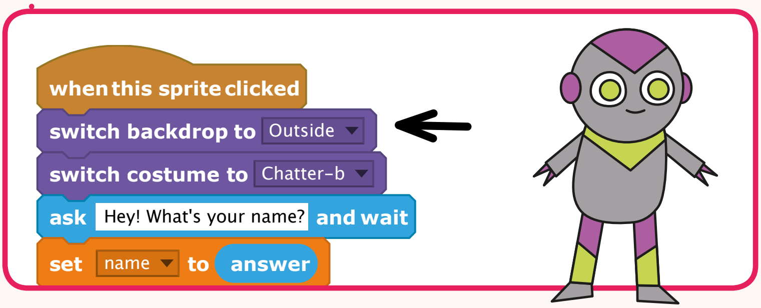


☑ You can now program your chatbot to change location, by adding this code to your chatbot:



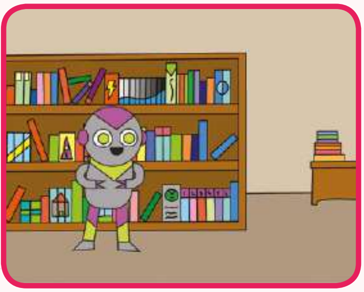
```
ask I'm going to the library. Do you want to come with me? and wait
if answer = yes then
    switch backdrop to Library
```

☐ You also need to make sure that your chatbot is in its original location when you start talking to it. Add this block to the top of your chatbot code:



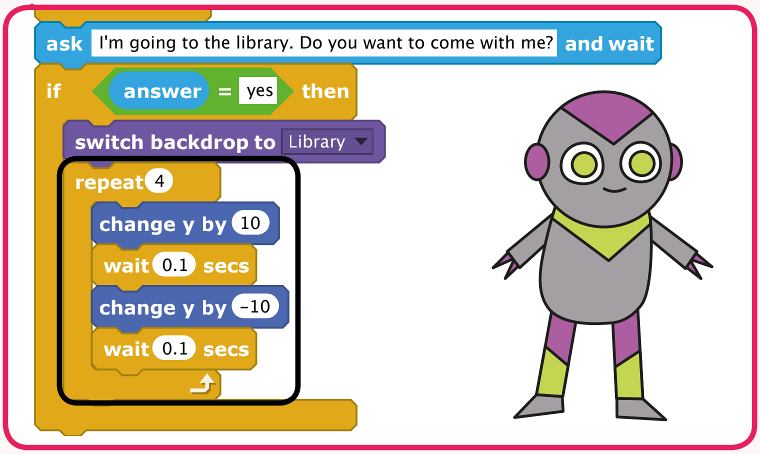
```
when this sprite clicked
switch backdrop to Outside ▾
switch costume to Chatter-b ▾
ask Hey! What's your name? and wait
set name ▾ to answer
```

☐ Test your program, and answer **yes** when asked if you want to go to the library. You should see that the chatbot's location has changed.

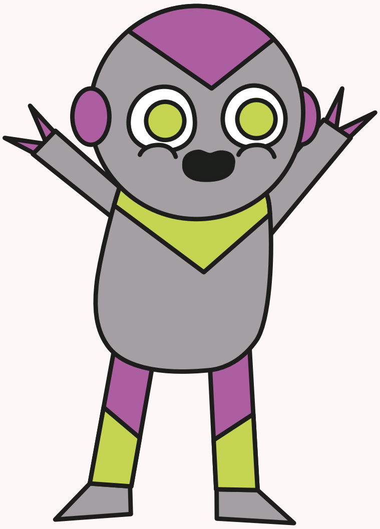


Does your chatbot change location if you type **no**? What about if you type **I'm not sure**?

☐ You can also add this code inside your **if** block, to make your chatbot jump up and down four times if the answer is **yes**:



```
ask I'm going to the library. Do you want to come with me? and wait
if answer = yes then
    switch backdrop to Library ▾
    repeat 4
        change y by 10
        wait 0.1 secs
        change y by -10
        wait 0.1 secs
```

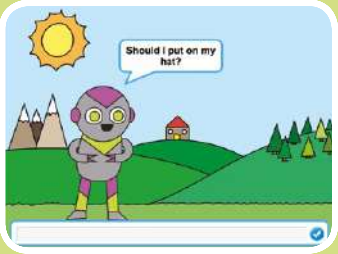


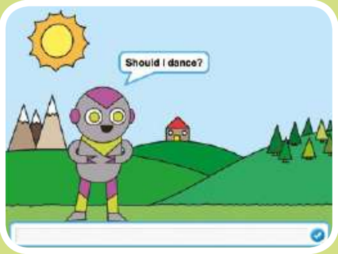
☐ Test your code again. Does your chatbot jump up and down if you answer **yes**?

CHALLENGE

MAKE YOUR OWN CHATBOT

Program your chatbot to ask another question – something with a yes or no answer. Can you make your chatbot respond to the answer?





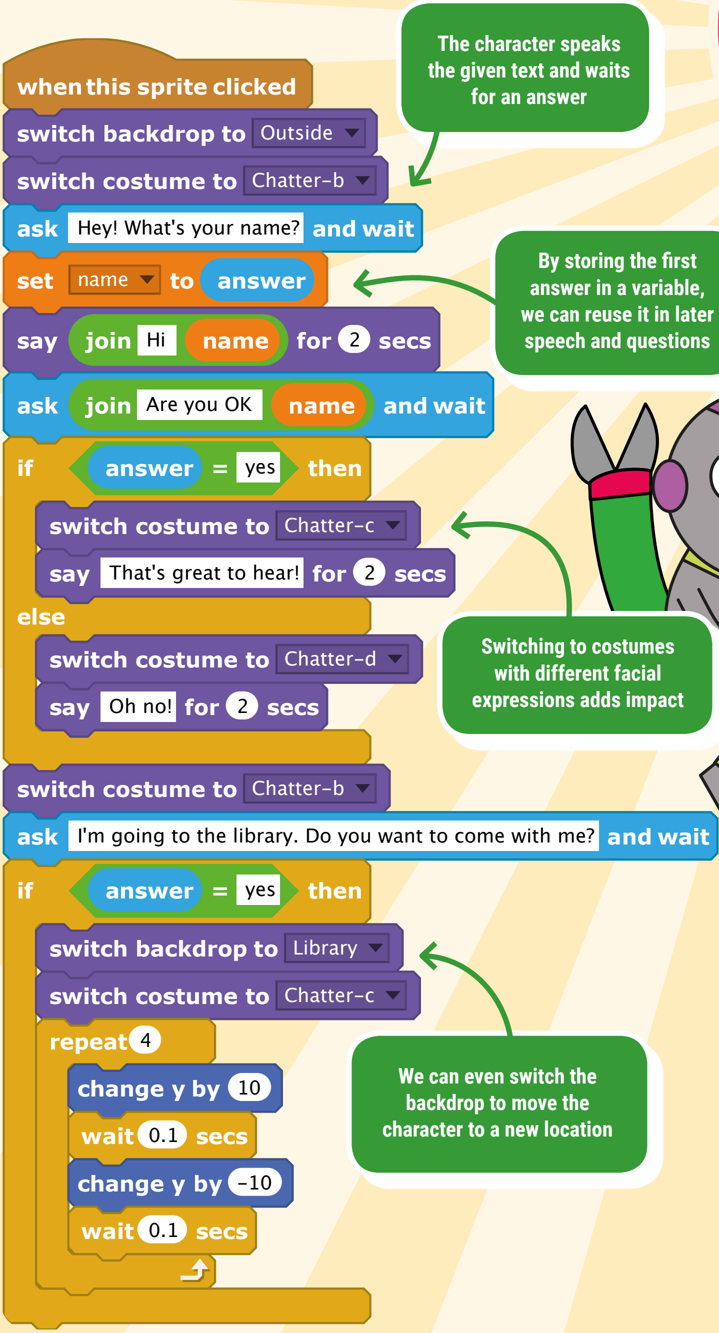
Once you've finished making your chatbot, get your friends to have a conversation with it! Do they like your character? Did they spot any problems?

Draw your own sprite and take a photo of it to use in your Scratch project!

CHATBOT FULL CODE LISTING

CHATTER

The Chatter bot asks questions and responds to the answers.



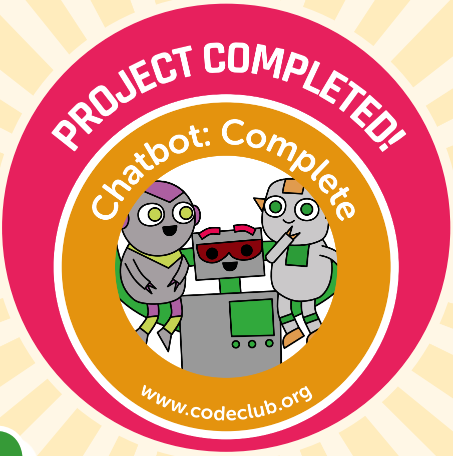
```
when this sprite clicked
switch backdrop to [Outside ▾]
switch costume to [Chatter-b ▾]
ask [Hey! What's your name?] and wait
set [name ▾] to (answer)
say (join [Hi] (name)) for (2) secs
ask (join [Are you OK] (name)) and wait
if <(answer) = [yes]> then
    switch costume to [Chatter-c ▾]
    say [That's great to hear!] for (2) secs
else
    switch costume to [Chatter-d ▾]
    say [Oh no!] for (2) secs
end
switch costume to [Chatter-b ▾]
ask [I'm going to the library. Do you want to come with me?] and wait
if <(answer) = [yes]> then
    switch backdrop to [Library ▾]
    switch costume to [Chatter-c ▾]
    repeat (4)
        change y by (10)
        wait (0.1) secs
        change y by (-10)
        wait (0.1) secs
    end
end
```

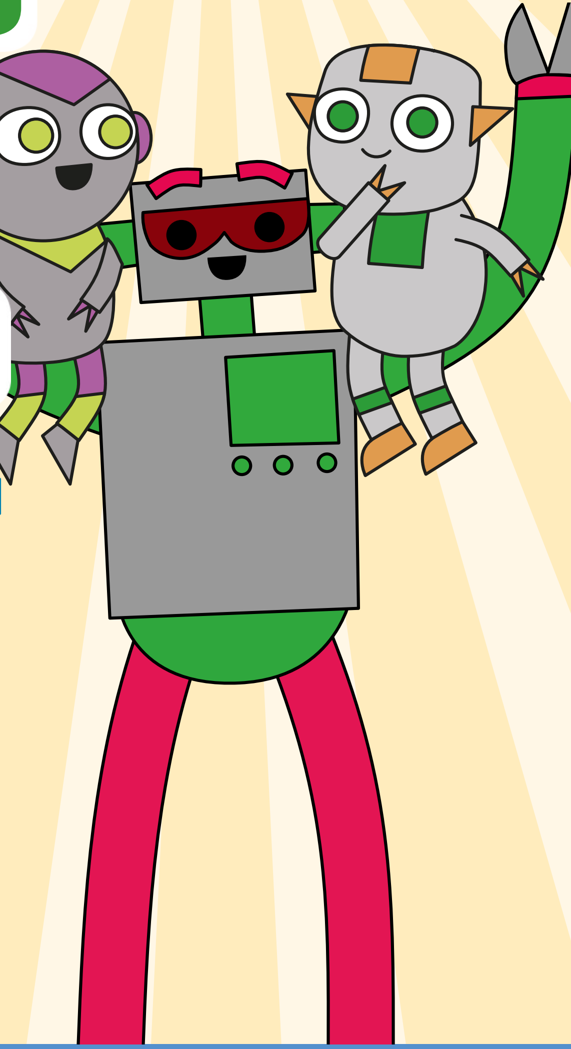
The character speaks the given text and waits for an answer

By storing the first answer in a variable, we can reuse it in later speech and questions

Switching to costumes with different facial expressions adds impact

We can even switch the backdrop to move the character to a new location



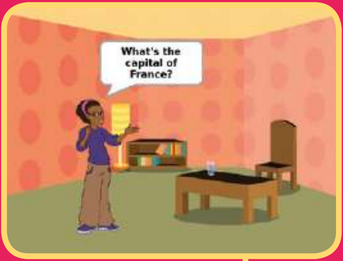


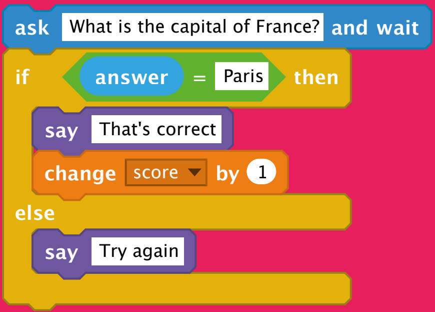
Now You Could Make...

With the skills you've learned, try making these projects...

QUIZ

Create a quiz that asks questions, and checks whether the player's answer is correct. A point is added to the player's score if they get a question correct.

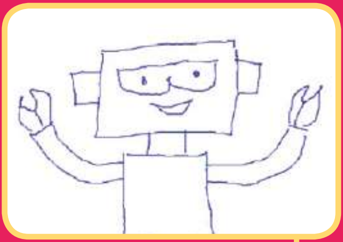


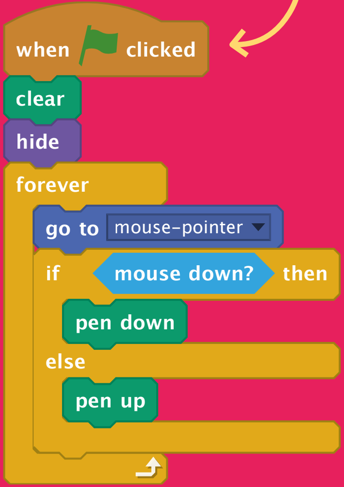


```
ask [What is the capital of France?] and wait
if <(answer) = [Paris]> then
    say [That's correct]
    change [score v] by (1)
else
    say [Try again]
```

PAINT APP

Use your mouse to draw on the stage! The hidden sprite will follow the mouse pointer, and the pen only draws if the mouse button is pressed.

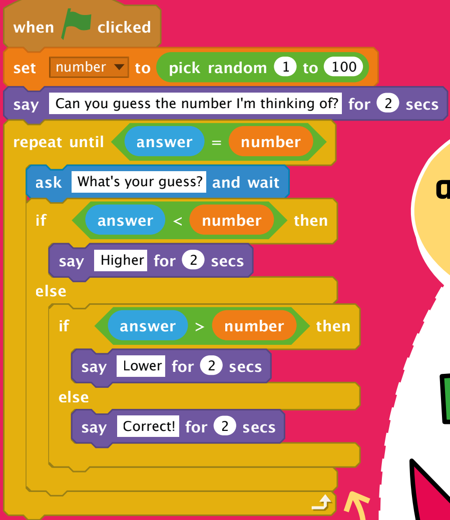




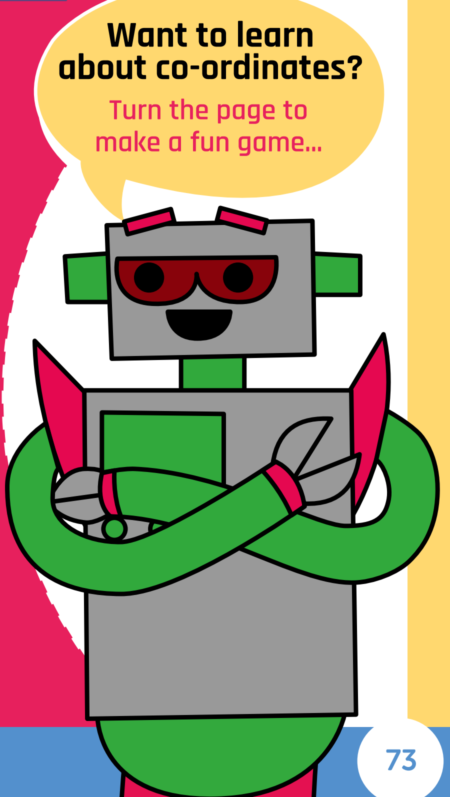
```
when ⚑ clicked
clear
hide
forever
    go to [mouse-pointer v]
    if <mouse down?> then
        pen down
    else
        pen up
```

GUESSING GAME

A number between 1 and 100 is randomly chosen, and the player must try to guess the chosen number. You could even adapt the game to keep track of the number of guesses taken, so that you can play against your friends.

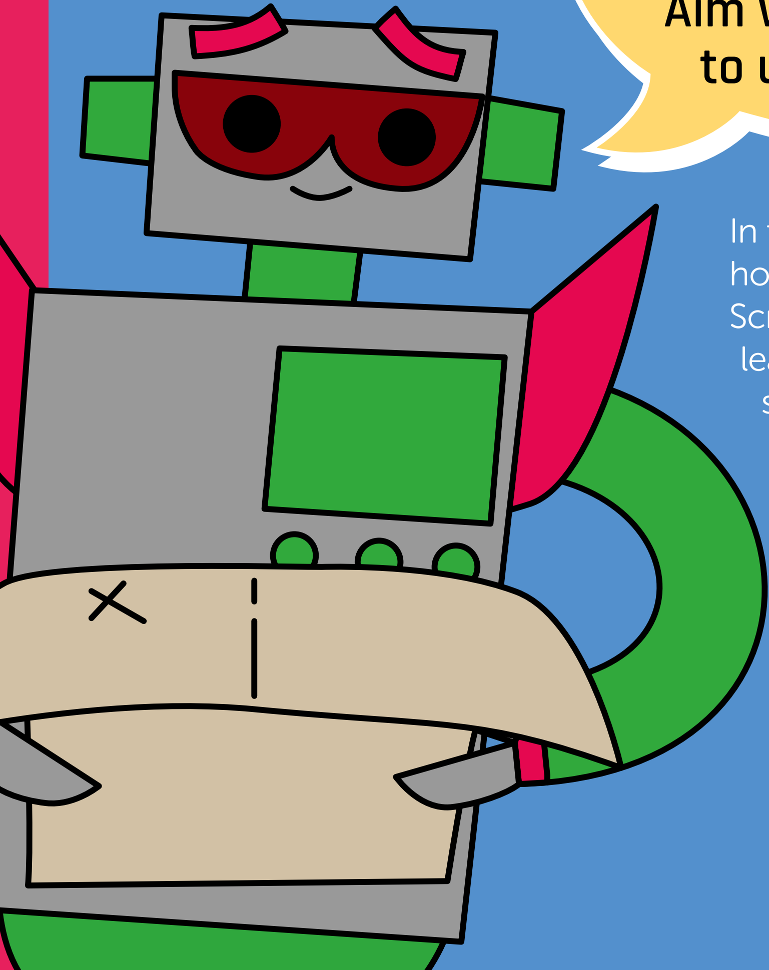


```
when ⚑ clicked
set [number v] to (pick random (1) to (100))
say [Can you guess the number I'm thinking of?] for (2) secs
repeat until <(answer) = (number)>
    ask [What's your guess?] and wait
    if <(answer) < (number)> then
        say [Higher] for (2) secs
    else
        if <(answer) > (number)> then
            say [Lower] for (2) secs
        else
            say [Correct!] for (2) secs
```


Want to learn about co-ordinates?
Turn the page to make a fun game...

On Target

Learn how co-ordinates work in Scratch with a fun game

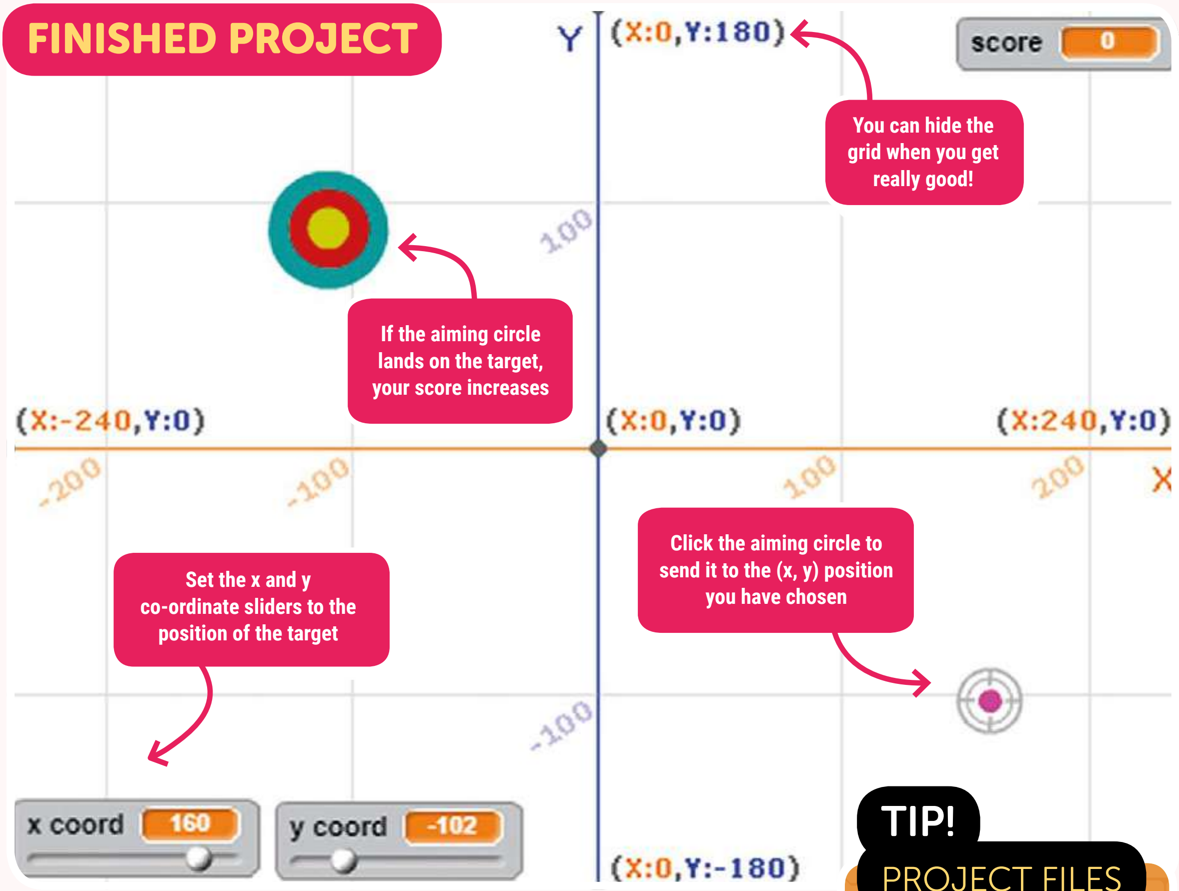


Make a target shooting game!

Aim well and learn how to use co-ordinates!

In this chapter, you'll be learning how the co-ordinate grid works in Scratch by making a game. You'll learn how to accurately position sprites on the stage using x and y co-ordinates. You'll also learn how to work with variable slider inputs. Get ready to hit some targets!

FINISHED PROJECT



You can hide the grid when you get really good!

If the aiming circle lands on the target, your score increases

Click the aiming circle to send it to the (x, y) position you have chosen

Set the x and y co-ordinate sliders to the position of the target

STEP 1: THE CO-ORDINATES GRID

Let's start by adding a co-ordinates grid backdrop.

☑ Open a web browser and go to **rpf.io/book-ontarget** to open the On Target Scratch project. Click Remix.

The project contains two sprites: a target for you to try to hit, and an aiming circle that will move to the co-ordinates you select. The target sprite is hidden at first; you'll use it later.

Scratch uses co-ordinates to allow you to accurately position sprites on the stage. There's a backdrop to help you understand the co-ordinates grid.

TIP!

PROJECT FILES

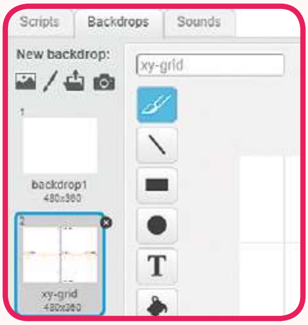
To download a zip file of all the Scratch 2 (.sb2) project assets files for this book, go to:

rpf.io/book-s1-assets

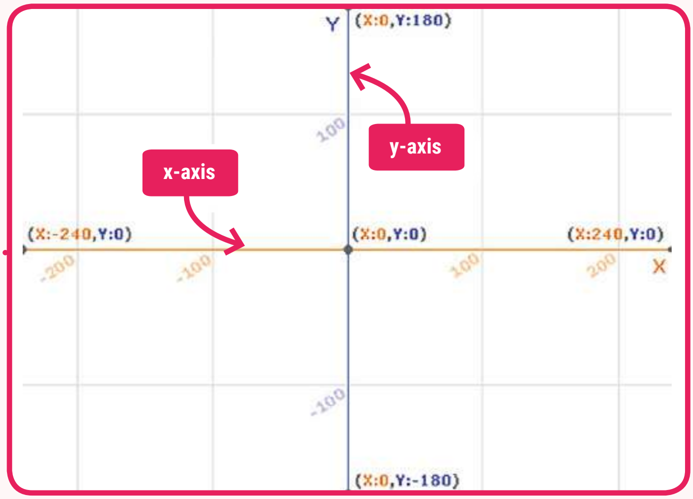
WHAT YOU'LL LEARN

- **x and y co-ordinates**
- **Positioning a sprite**
- **Slider inputs**

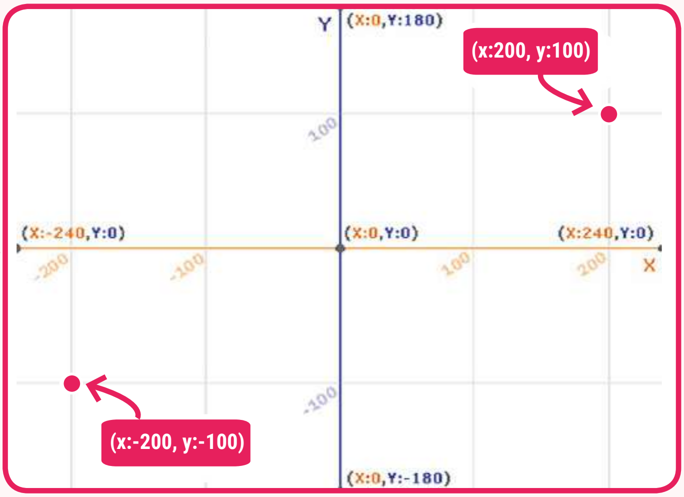
☐ Add the **xy-grid** backdrop to your project (keep the blank backdrop).



The co-ordinates of the stage run from **-240** to **240** along the x-axis, and **-180** to **180** along the y-axis. The co-ordinates of the centre are **(x:0, y:0)**.

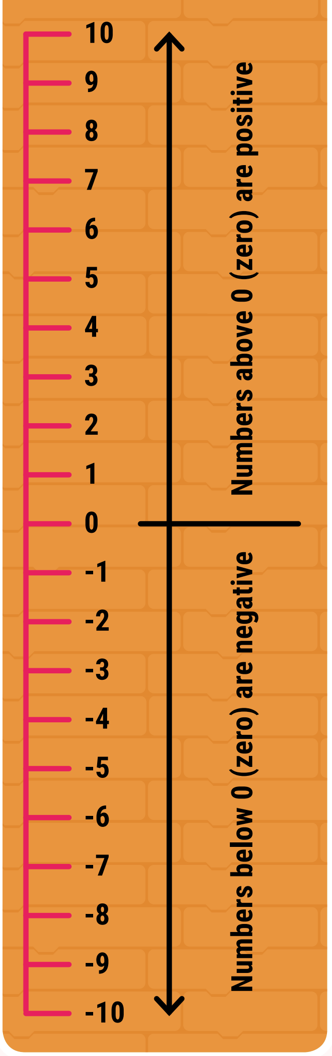


The position **(x:-200, y:-100)** is towards the bottom left on the stage, and the position **(x:200, y:100)** is near the top right.



TIP!

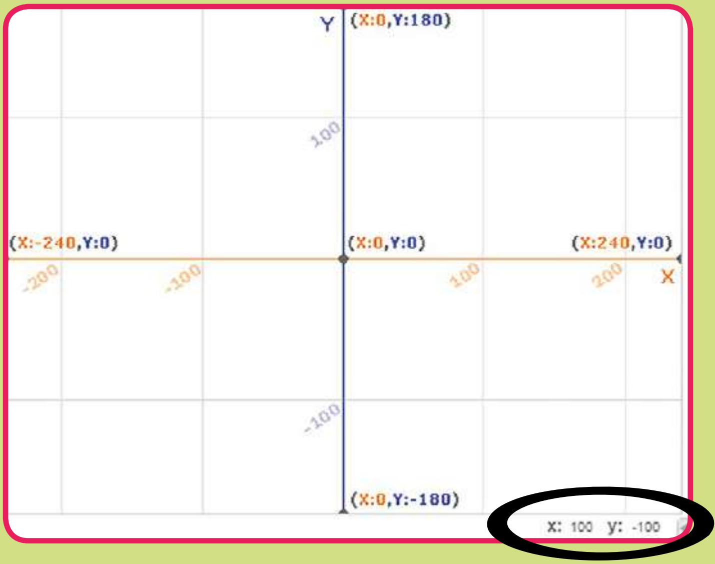
NEGATIVE NUMBERS ARE SMALLER THAN ZERO


10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
Numbers above 0 (zero) are positive
Numbers below 0 (zero) are negative

HOW TO...

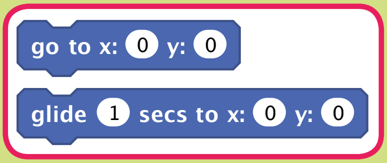
USE CO-ORDINATES

Try moving the mouse pointer around the stage and notice how the co-ordinates shown in the bottom right-hand corner change.

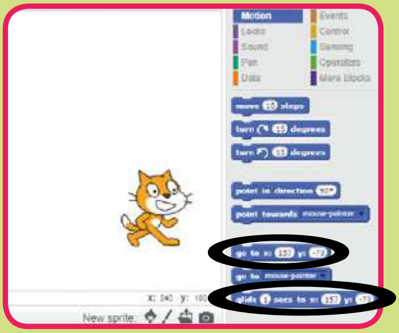


You can use this to cheat in the game we're making! But if you switch to full-screen mode, you don't see the co-ordinates of the mouse cursor.

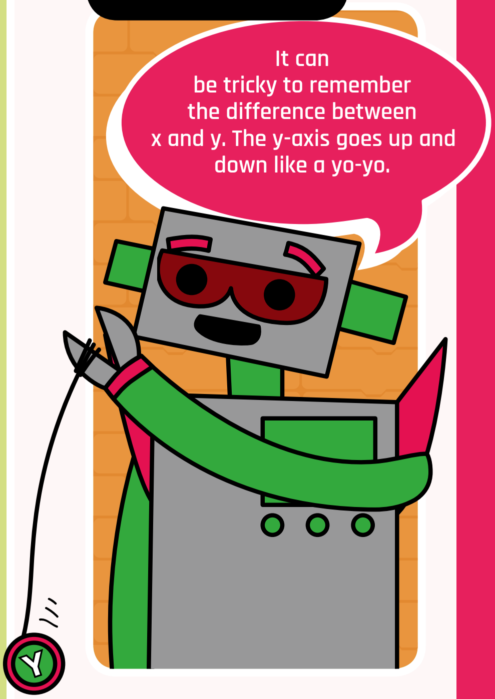
The go to and glide Motion blocks take their default inputs from the current position of the sprite. This means you can move a sprite to the position you want it to go to and then just drag the block to the coding area. This is easier that working out the co-ordinates and entering them yourself.



```
go to x: 0 y: 0
glide 1 secs to x: 0 y: 0
```



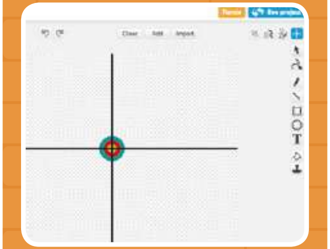
TIP! X AND Y

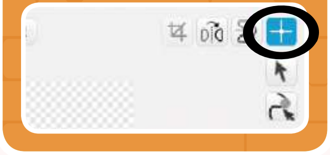


It can be tricky to remember the difference between x and y. The y-axis goes up and down like a yo-yo.

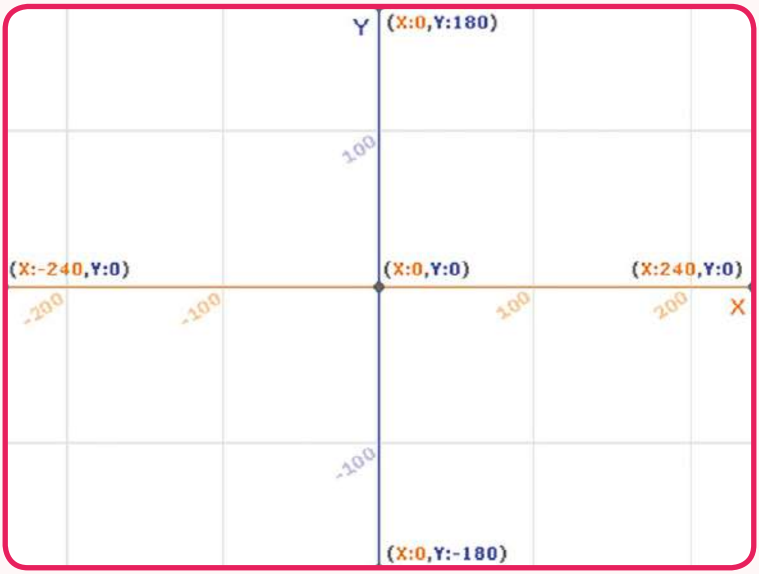
TIP! SET CENTRE

The co-ordinates are based on the centre of the sprite. You can set this using the crosshair tool when you edit a costume for a sprite.





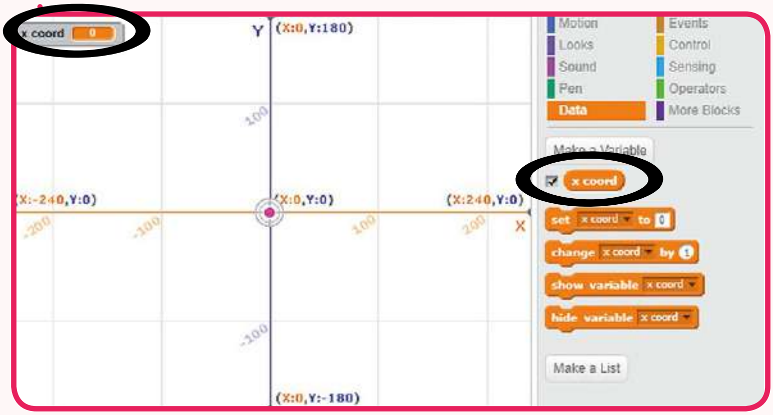
Add letters to the grid below to mark the following positions: **A:** (x:50, y: 50); **B:** (x:-100, y: -100); **C:** (x: -150, y: 100); **D:** (x: 175, y: -30)



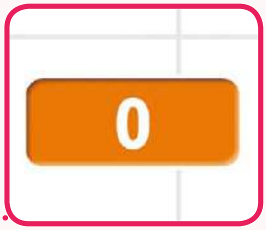
STEP 2: AIM AT (X, Y) CO-ORDINATES

Now let's send the aiming circle to (x, y) co-ordinates.

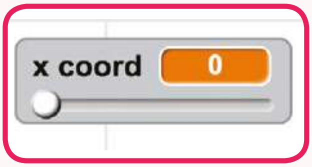
☐ Add a variable called **x coord** to your Aim sprite and choose 'For all sprites'. A **monitor** for your variable will appear on the stage.



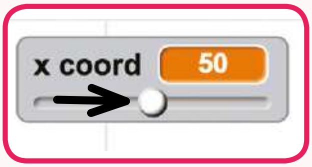
☐ Double-click on the variable monitor for the **x coord** variable and it will change to just showing the number; this is called the 'large readout'.



☐ Double-click on the variable monitor again and it will turn into a slider.



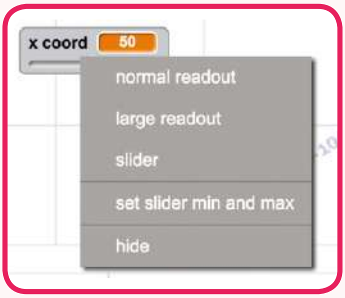
☐ Drag the slider and watch the number change.



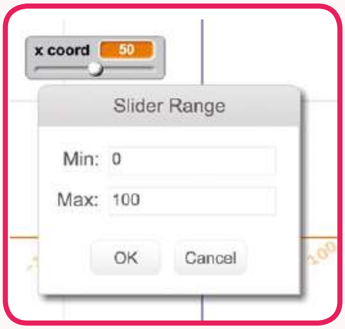
The current smallest number for the slider is ___ and the largest number value is ___ .

You're going to use the slider to represent an x co-ordinate, so it needs to be able to change between -240 and 240.

☐ Right-click on the x coord variable monitor on the stage and choose **set slider min and max**.



☐ Set the **Min** to **-240** and the **Max** to **240**.



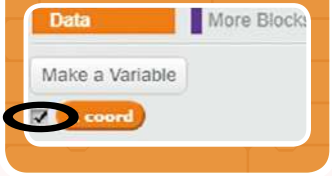
Min: 0

Max: 100

TIP!

VARIABLE MONITOR

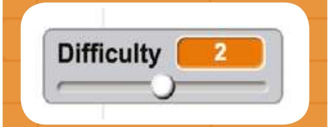
When you create a new variable, a 'variable monitor' appears on the stage showing its current value. You can show or hide the monitor on the stage by clicking the tick-box next to the variable.



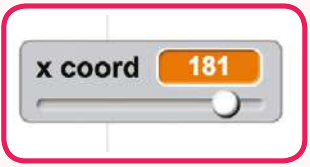
TIP!

SLIDER INPUTS

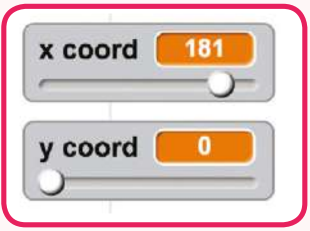
A slider allows you to set a variable by moving a control. Sliders are useful for creating number inputs in Scratch.


Difficulty 2

☑ Try the slider out. Now you can set the x coord variable to values from -240 to 240, which corresponds to the range of the x-axis in Scratch.

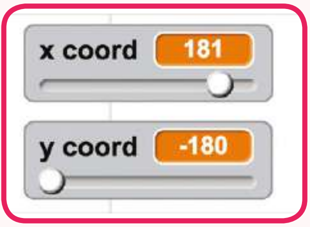

x coord 181

☑ Now add a **y coord** variable for the y co-ordinate and change to the slider setting.


x coord 181

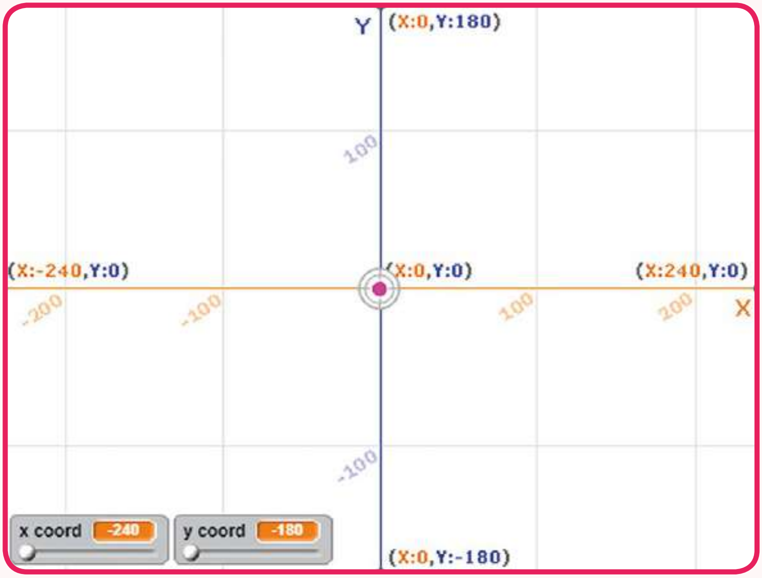
y coord 0

☑ Set the Min to -180 and the Max to 180 to match the range of the y-axis.


x coord 181

y coord -180

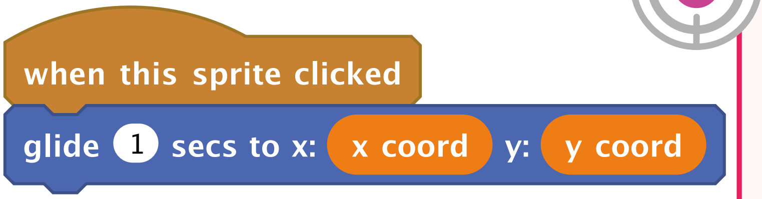
☑ Drag your x and y sliders to the bottom left of the stage. Make sure you place x on the left and y on the right, as co-ordinates are given in this order.



x coord -240

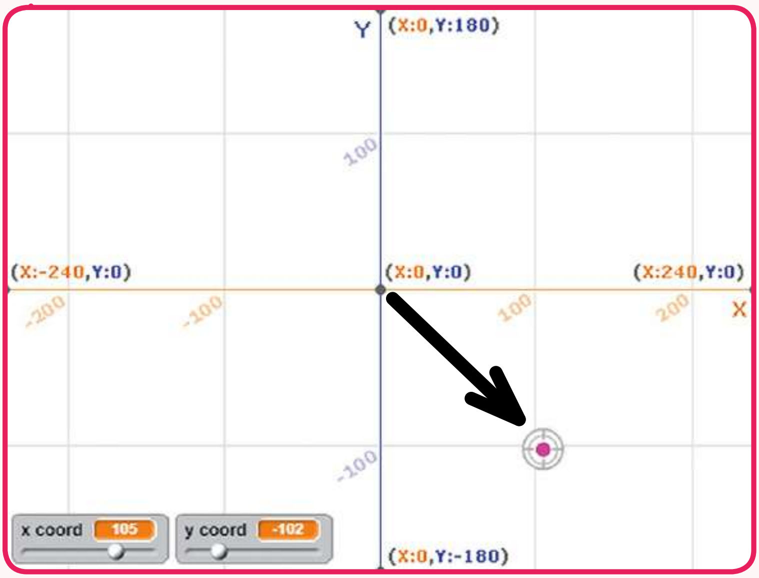
y coord -180

□ Now add a script to your Aim sprite so that when you click it, it glides to the x coord and y coord shown on the variable sliders.



```
when this sprite clicked
glide 1 secs to x: x coord y: y coord
```

□ Spend some time changing the x and y co-ordinates and then clicking on the aiming circle to get it to move to the position you have chosen. Make sure you understand how changing the x and y sliders will change the position of the aiming circle.



x coord 105

y coord -102

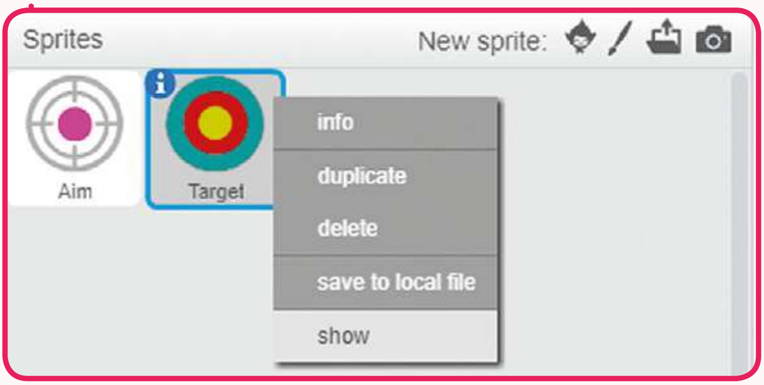
TIP! TINY MOVES

You can click a slider either side of the knob to increase or decrease the value by 1 at a time. Try it! This is useful for accurate positioning.

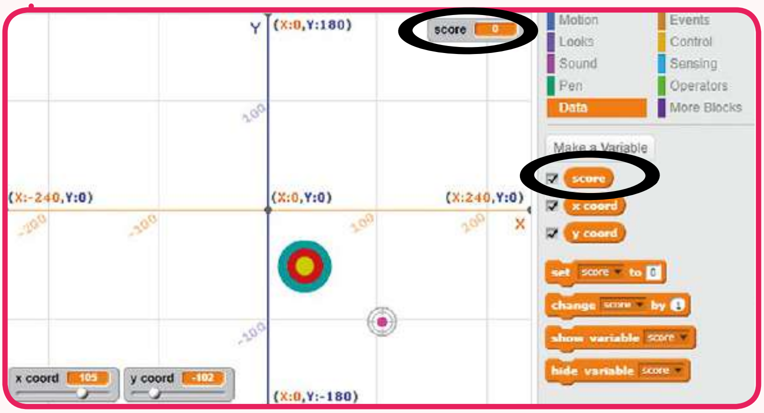
STEP 3: CAN YOU HIT THE TARGET?

Now let's see if you can set the co-ordinates correctly to aim at the target. You'll score a point each time you hit the target.

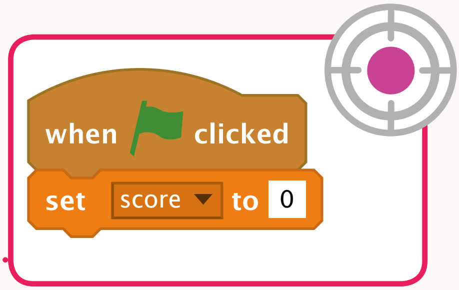
☐ Right-click on the Target sprite below the stage and choose **show**. The sprite will appear on the stage.



☐ Add a **score** variable for all sprites and drag its stage monitor to the top right.



☐ Add a script to the Aim sprite to set the score to 0 at the beginning of a game.



```
when ⚑ clicked
set score ▾ to 0
```

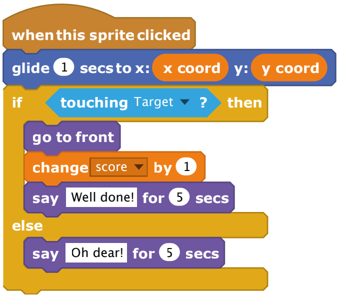
DEBUG

If your Aim sprite ends up behind your Target, add a go to front block before changing the score.

TIP!

The go to front Looks block puts a sprite on top of all the other sprites.

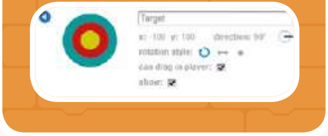
☑ Add code to the Aim sprite to check whether it is touching the target after gliding. Either reward the player by saying 'Well done!' and adding a point to the score, or if they didn't hit the target, you can say 'Oh dear!'.



```
when this sprite clicked
glide 1 secs to x: x coord y: y coord
if touching Target ? then
    go to front
    change score by 1
    say Well done! for 5 secs
else
    say Oh dear! for 5 secs
```

TIP! DRAG IT

If you want to try this out in full-screen mode, then you'll need to allow the target to be dragged. Click on information (i) for the Target sprite and click the box next to 'can drag in player'.



Target

x: 100 y: 100 direction: 90°

rotation style:

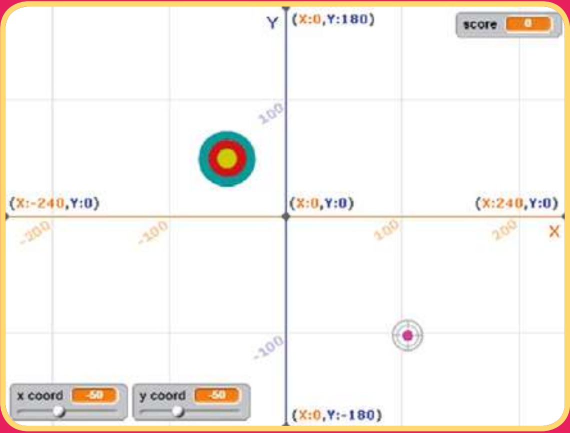
can drag in player: ☑

show: ☑

☑ TEST YOUR PROJECT

Drag the target to a new position on the stage. Set the x and y co-ordinates to where you think the target is. Click on the aiming circle to move to the co-ordinates you have chosen and see if you got it right.

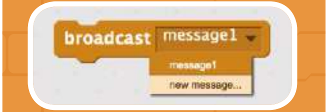
If you click on the aiming circle now, will it touch the target? _________



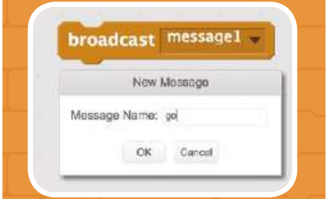
If you succeed then you will see a 'Well done!' message.

TIP! BROADCAST

To create a new broadcast block message, click its drop-down arrow and select ‘new message…’.

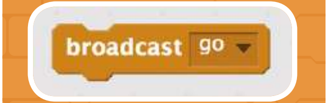


Now type a message into the Message Name field and click OK.



Message Name: go

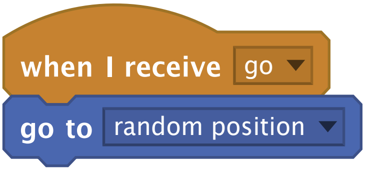
The new message will now appear in the broadcast block and will be also available in its drop-down list.


```
broadcast go
```

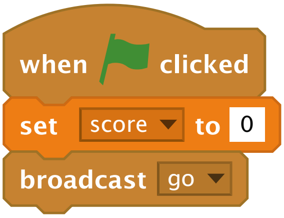
STEP 4: MOVING TARGET

Now let’s get the target moving to a random position at the start of the game and at the end of each turn.

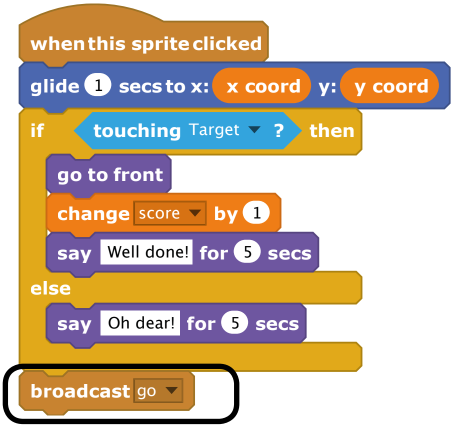
☐ Add a script to the Target sprite to go to a random position when it receives a **go** message.


```
when I receive go
go to random position
```

☐ Add a block to the when flag clicked script of the Aim sprite to broadcast a **go** message.


```
when [flag] clicked
set score to 0
broadcast go
```

☐ Add code to the Aim sprite’s when this sprite clicked script to broadcast a **go** message at the end of a turn.


```
when this sprite clicked
glide 1 secs to x: x coord y: y coord
if touching Target ? then
    go to front
    change score by 1
    say Well done! for 5 secs
else
    say Oh dear! for 5 secs
broadcast go
```

TEST YOUR PROJECT

Now you can try playing the game. Click the green flag to start. The target moves to a new position. Set the x and y sliders and then click the aiming circle to send it to that position.

Did you hit the target? Have another go. Keep trying until you are good at it.

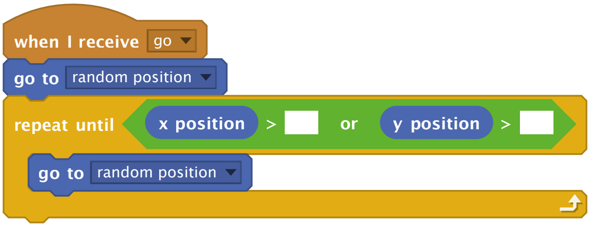
CHALLENGE

HIT IT

Sometimes the target ends up on top of the sliders. That's annoying! Click the green flag lots of times without playing the game until you see the target on top of the sliders.

Can you add code to the target sprite so that it moves to a new position if it ends up on top of the sliders?

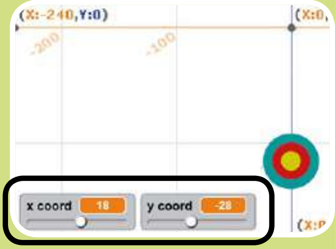
Start with this code and fill in the positions.



```
when I receive [go ▼]
go to [random position ▼]
repeat until <<(x position) > [ ]> or <(y position) > [ ]>>
    go to [random position ▼]
```

You need the centre of the target to avoid landing in the highlighted rectangle.

Test your code again by clicking the green flag lots of times and make sure it doesn't land on the sliders.



TIP!

You can move the mouse to check the co-ordinates of positions on the stage.

HINT!

> means 'greater than'. Remember -100 is bigger than -200! You need to check that the x position is greater than around 0 and the y position is greater than around -115.

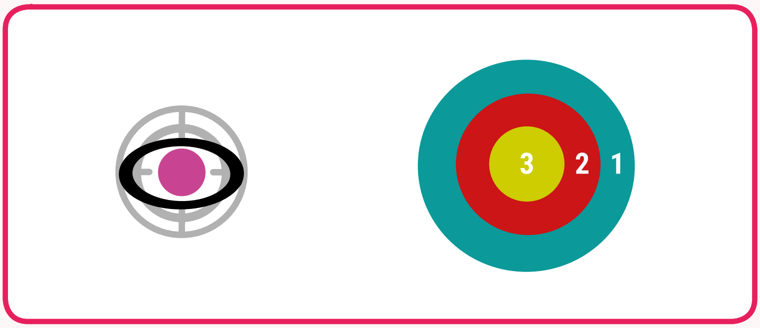
TIP! TOUCHING COLOUR?

The first colour in the color is touching? block is the colour on the sprite that the script belongs to; the second colour is on another sprite. Click on the colour box that you want to change and then click on that colour anywhere on the stage or editor.

STEP 5: MORE POINTS FOR ACCURACY

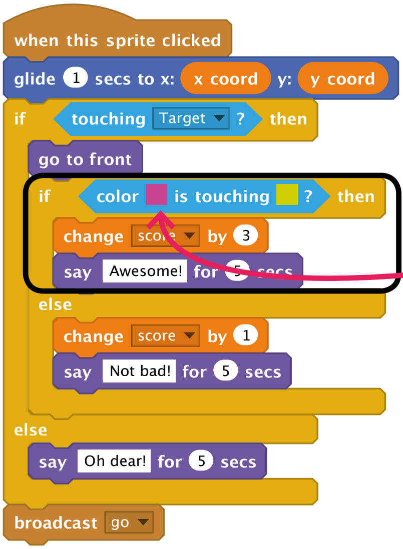
Now let's increase the score if you get your aim closer to the centre of the target.

☐ You're going to use the color is touching? block to detect which part of the target the pink circle in the centre of the aiming circle is touching.



You'll get 3 points if it's touching the yellow circle, 2 points for red, and 1 point for blue.

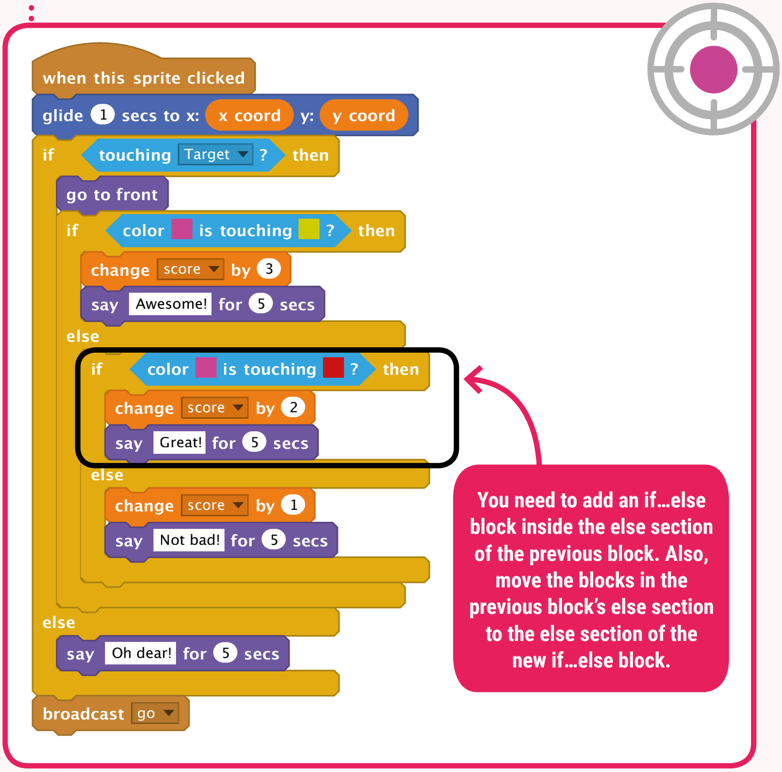
☐ Update the code on the Aim sprite so that it checks whether the centre of the sprite is touching the target's yellow centre and rewards the player with points and a different message:



```
when this sprite clicked
glide 1 secs to x: x coord y: y coord
if touching Target ? then
    go to front
    if color 🟪 is touching 🟨 ? then
        change score by 3
        say Awesome! for 5 secs
    else
        change score by 1
        say Not bad! for 5 secs
else
    say Oh dear! for 5 secs
broadcast go
```

Click on the first colour and then click on the pink colour in the centre of the Aim sprite. Click on the second colour and then click on the yellow colour in the centre of the Target

☐ Update the code on the Aim sprite to detect when the pink circle touches the red ring to give 2 points.



```
when this sprite clicked
glide 1 secs to x: x coord y: y coord
if touching Target ? then
    go to front
    if color 🟪 is touching 🟨 ? then
        change score by 3
        say Awesome! for 5 secs
    else
        if color 🟪 is touching 🟥 ? then
            change score by 2
            say Great! for 5 secs
        else
            change score by 1
            say Not bad! for 5 secs
else
    say Oh dear! for 5 secs
broadcast go
```

You need to add an if…else block inside the else section of the previous block. Also, move the blocks in the previous block's else section to the else section of the new if…else block.

You don't need to check for the red ring being hit if you know that the player hit the yellow ring, so this code goes in the **else** section.

CHALLENGE

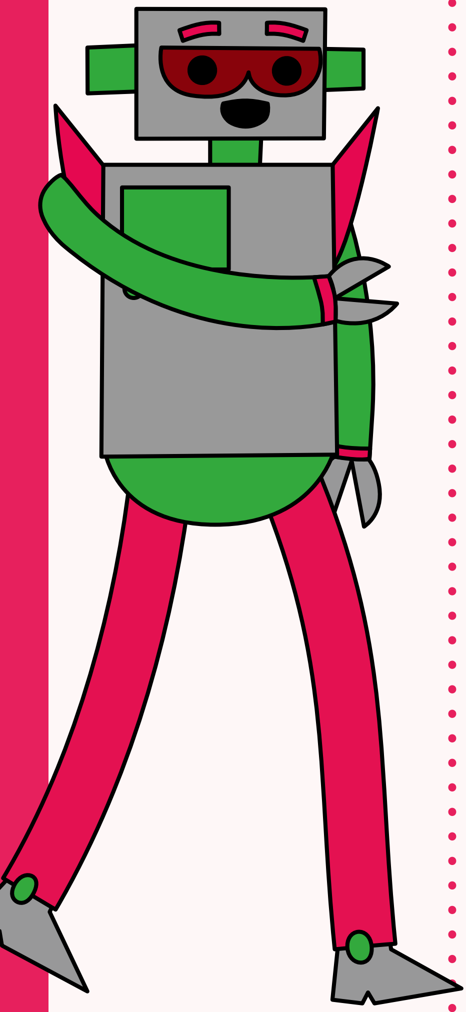
☐ **Become a co-ordinates expert! Keep practising until you are really confident using co-ordinate grid positions in Scratch.**

☐ **Add a 'turns' variable and see how many points you can score in 10 turns.**

☐ **Can you add instructions to your game that explain how co-ordinates work? You can record your own voice or type text into a sprite.**

HINT!

The x-axis runs from -240 on the left to 240 on the right. The y-axis runs from -180 at the bottom to 180 at the top. Remember, y goes up and down like a yo-yo.

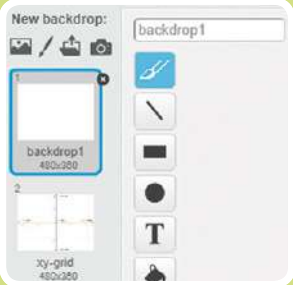


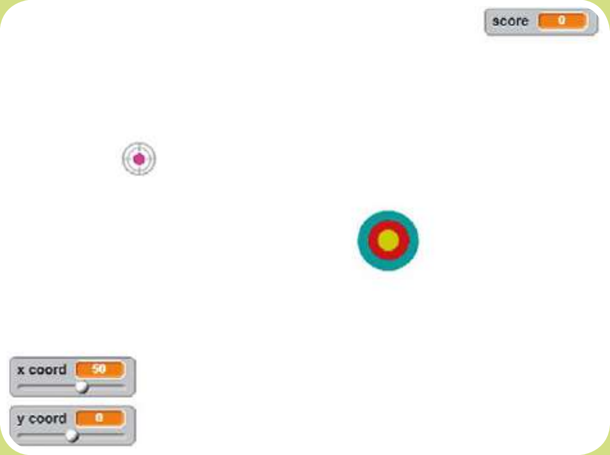
CHALLENGE

TOO EASY FOR YOU?

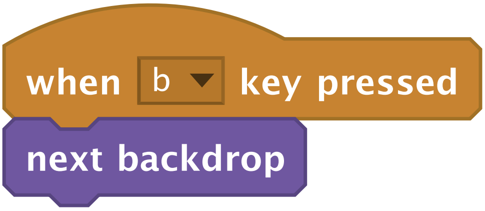
☑ Try making the Aim sprite or Target smaller so you have to be more accurate.

☑ Try changing the backdrop to the plain one without the grid.





☑ If you like, you can add a script to the stage to switch between the backdrops when you press a key:



```
when b key pressed
next backdrop
```

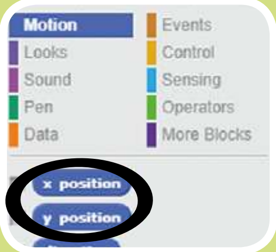
Hide the grid and switch to full-screen mode so that you can't cheat by looking at the co-ordinates of the target. If you find you're not hitting the target, switch back to the grid backdrop and have a bit more practice.

HOW TO...

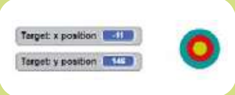
WORK WITH X AND Y POSITIONS

Scratch has built-in variables for the x and y position of a sprite.

Click Scripts and then Motion and you will see the x position and y position variables near the bottom.

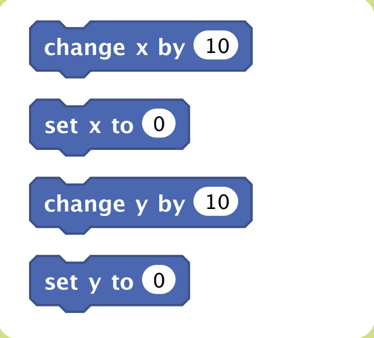


Just as with variables you create, you can click the tick-box to show these variables on the stage.



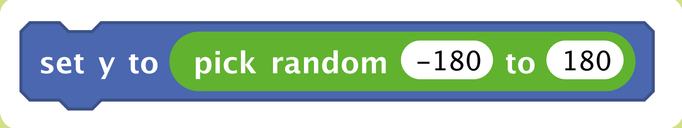
The variables will update when you drag the sprite around the screen.

You can change the x and y position of a sprite separately using set and change blocks.



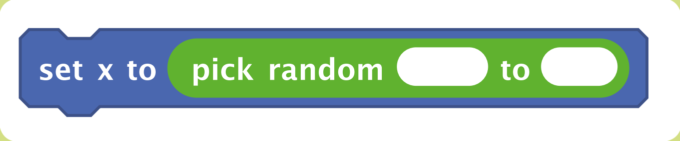
```
change x by 10
set x to 0
change y by 10
set y to 0
```

To send a sprite to a random y position, use:



```
set y to pick random -180 to 180
```

What numbers do you need in the following code to send a sprite to a random x position?:

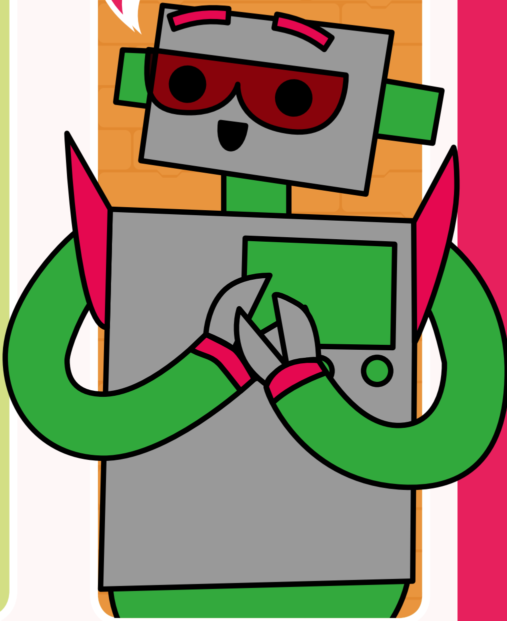


```
set x to pick random ( ) to ( )
```

TIP!

PICK RANDOM

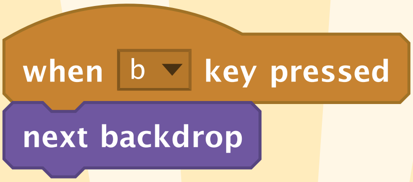
The pick random block selects a random number ranging from the value given in the first field to the value in the second field. If both values have no decimals, it will report a whole number.



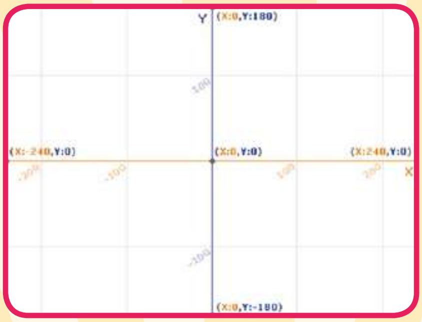
ON TARGET FULL CODE LISTING

STAGE

A key press changes the backdrop.

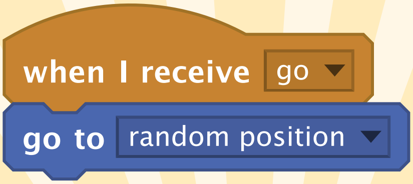


```
when [b v] key pressed
next backdrop
```

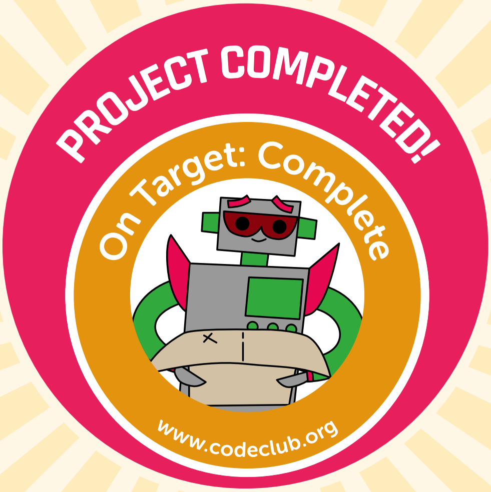


TARGET

It's sent to a random position.

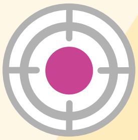


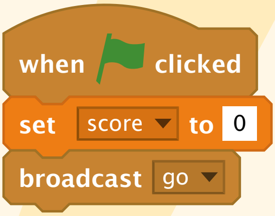
```
when I receive [go v]
go to [random position v]
```



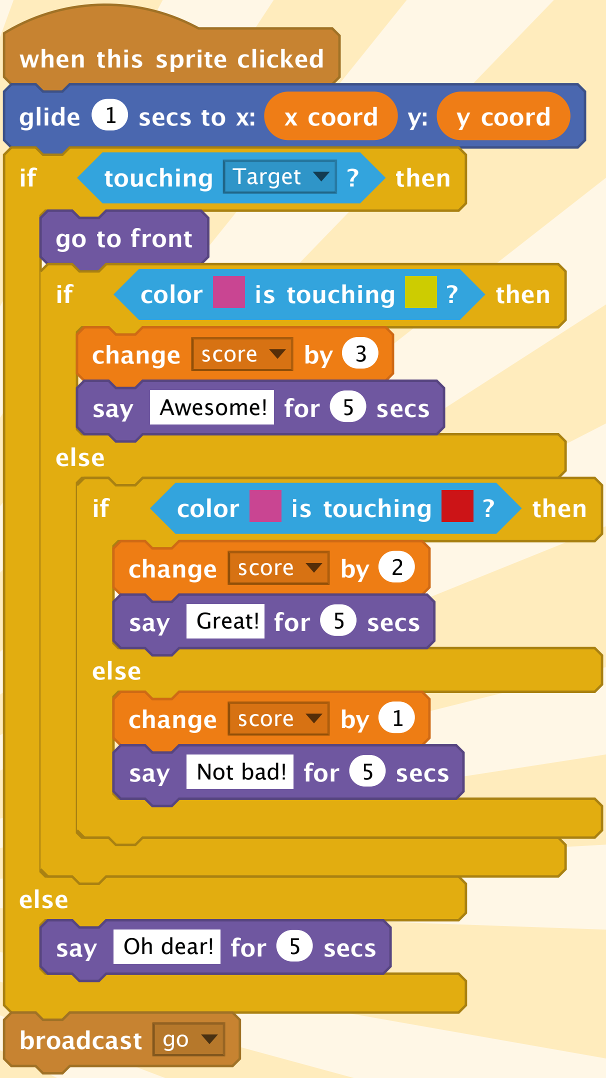
AIM

When clicked, it's sent to the co-ordinates of the sliders.





```
when green flag clicked
set [score v] to (0)
broadcast [go v]
```



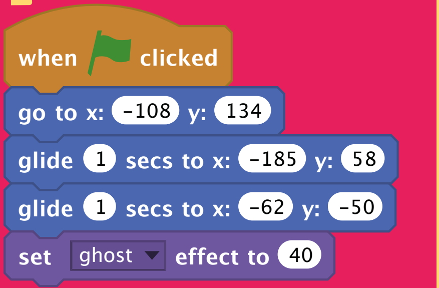
```
when this sprite clicked
glide (1) secs to x: (x coord) y: (y coord)
if <touching [Target v] ?> then
    go to front
    if <color [🟪] is touching [🟨] ?> then
        change [score v] by (3)
        say [Awesome!] for (5) secs
    else
        if <color [🟪] is touching [🟥] ?> then
            change [score v] by (2)
            say [Great!] for (5) secs
        else
            change [score v] by (1)
            say [Not bad!] for (5) secs
        end
    end
else
    say [Oh dear!] for (5) secs
end
broadcast [go v]
```

Now You Could Make...

With your new-found knowledge, you could try these projects...

GLIDING GHOSTS

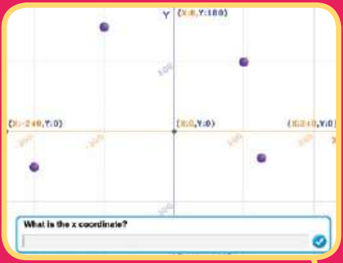
Create an animation that uses co-ordinates to position sprites accurately.



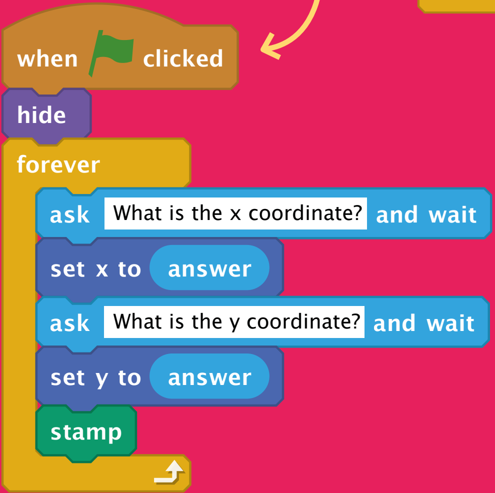
```
when 🏳 clicked
go to x: -108 y: 134
glide 1 secs to x: -185 y: 58
glide 1 secs to x: -62 y: -50
set ghost ▾ effect to 40
```

GRID PLOTTER

Make a maths app that allows you to ask the user for co-ordinates and then stamp a sprite to plot the given co-ordinates.



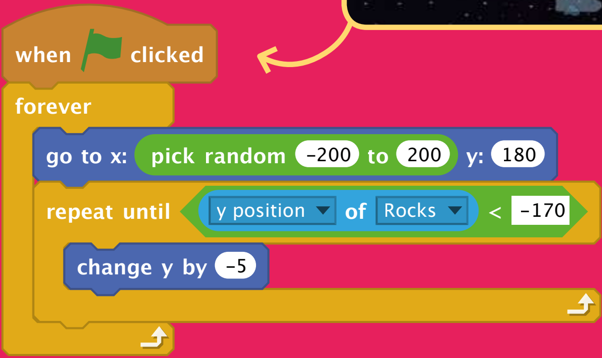
What is the x coordinate?



```
when 🏳 clicked
hide
forever
    ask What is the x coordinate? and wait
    set x to answer
    ask What is the y coordinate? and wait
    set y to answer
    stamp
```

FALLING ROCKS

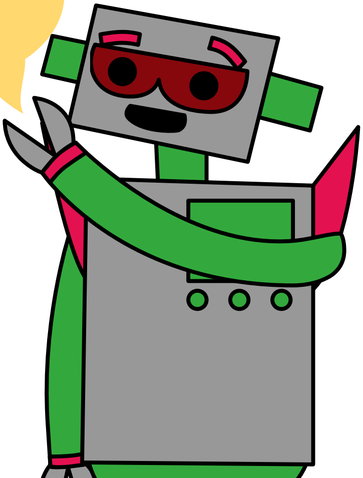
Code a game where rocks always fall from the same y position (height), but random x positions.



```
when 🏳 clicked
forever
    go to x: pick random -200 to 200 y: 180
    repeat until y position ▾ of Rocks ▾ < -170
        change y by -5
```

Want to make a boat race game?

Turn the page to find out how...



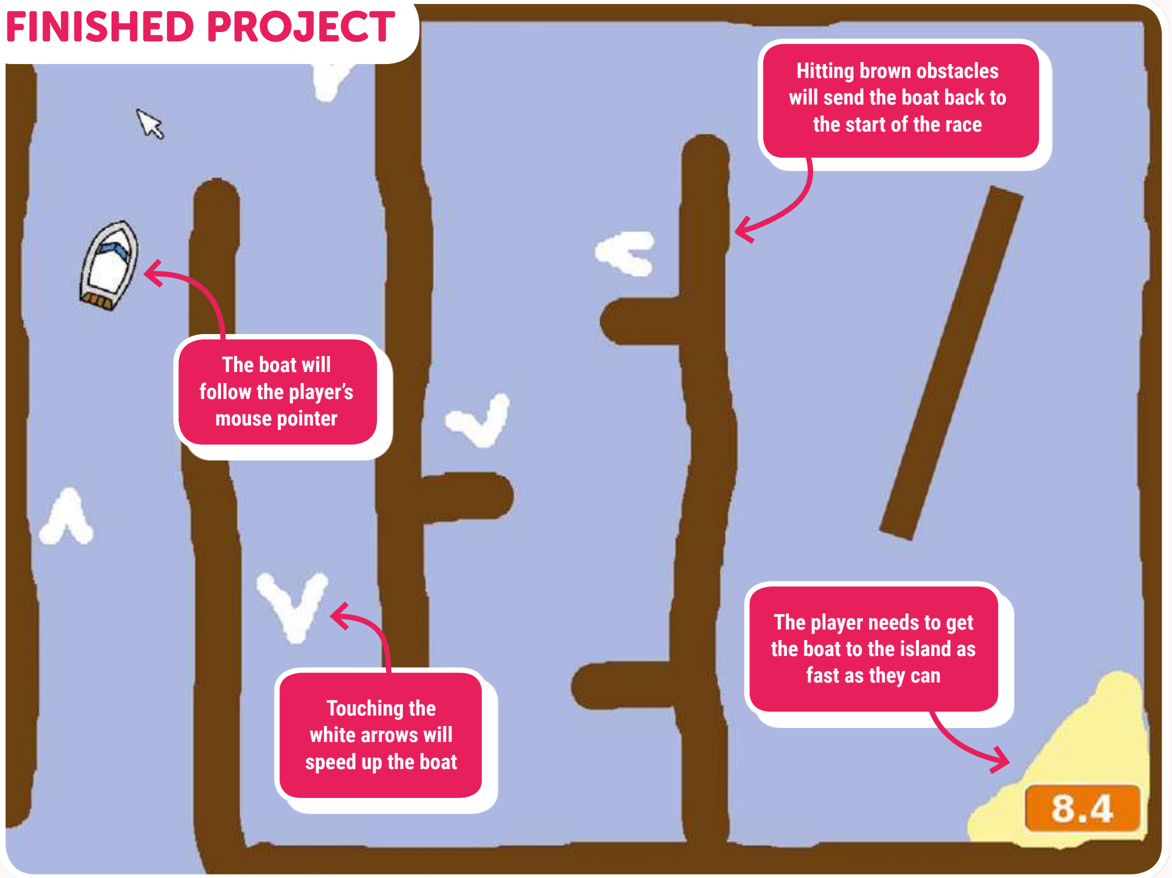
Boat Race

Make your own racing game featuring colour-sensing collision detection and a timer

Code an exciting racing game!
Add speed boosts, a timer, and more!

In this chapter, you'll learn how to control a boat sprite with the mouse. You will also discover how to sense when it hits an obstacle, by using **touching color** blocks.

FINISHED PROJECT



Hitting brown obstacles will send the boat back to the start of the race

The boat will follow the player's mouse pointer

Touching the white arrows will speed up the boat

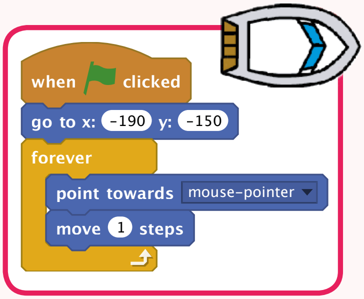
The player needs to get the boat to the island as fast as they can

STEP 1: CONTROLLING YOUR BOAT

Program your boat sprite to follow the mouse pointer.

□ In a web browser, go to **rpf.io/book-boatrace** to open the Boat Race project. Click the Remix button.

□ You are going to control the boat with your mouse. Add this code to your Boat sprite:



```
when ⚑ clicked
go to x: -190 y: -150
forever
    point towards mouse-pointer ▾
    move 1 steps
```

WHAT YOU'LL LEARN

- Sprite movement using the mouse

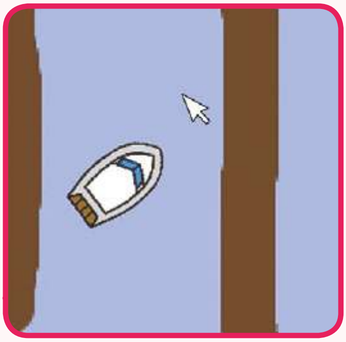
TIP!

PROJECT FILES

To download a zip file of all the Scratch 2 (.sb2) project assets files for this book, go to:

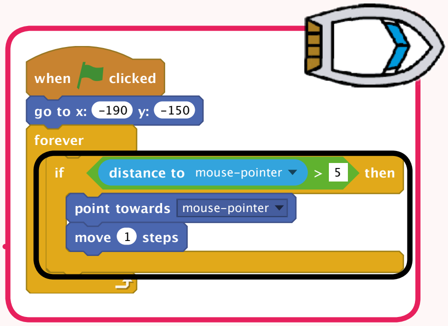
rpf.io/book-s1-assets

☐ Test out your boat, by clicking the flag and moving the mouse. Does the boat sail towards your mouse pointer? When done, hit the red Stop button.

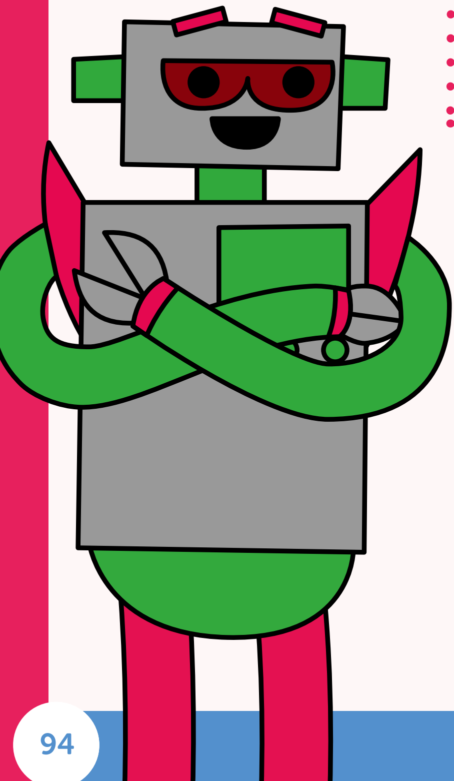


☐ Have you noticed that the boat glitches if it reaches the mouse pointer? To stop this happening, you'll need to add an if block to your code, so that the boat only moves if it is more than 5 pixels away from the mouse.

Note: This uses a □ > □ Operator block with a distance to Sensing block.



```
when 🏴 clicked
go to x: -190 y: -150
forever
    if distance to mouse-pointer ▾ > 5 then
        point towards mouse-pointer ▾
        move 1 steps
```

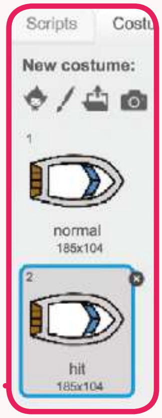


☐ Test out your boat again, to check that the problem has been fixed. When done, hit the Stop button

STEP 2: CRASHING

Your boat can sail through the wooden barriers! Let's fix that.

☐ You'll need two costumes for your boat: one normal costume, and one for when the boat crashes. Right-click on your boat costume to **duplicate** it, and name your costumes **normal** and **hit**.



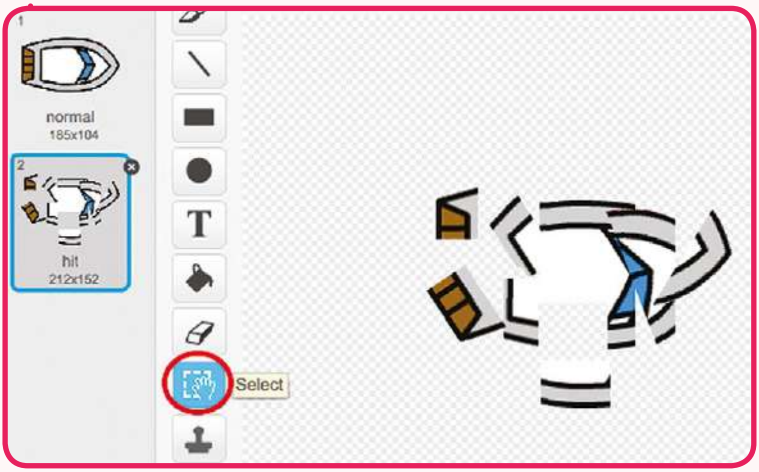
Scripts Costu

New costume:

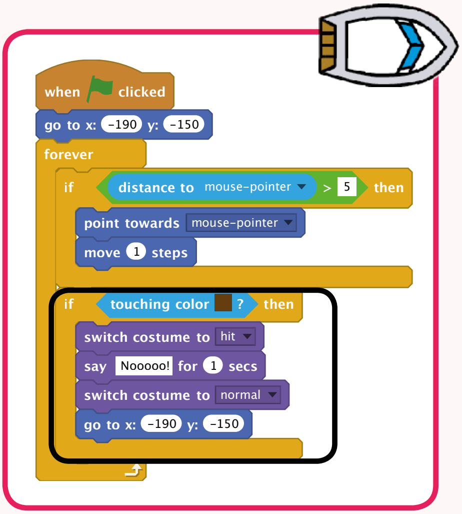
1 normal 185x104

2 hit 185x104

□ Click on your **hit** costume, and choose the **Select** tool to grab bits of the boat and move and rotate them around. Make your boat look as if it's crashed.



□ Add this code to your boat, inside the **forever** loop, so that it crashes when it touches any brown wooden bits.



```
when ⚑ clicked
go to x: -190 y: -150
forever
    if < (distance to mouse-pointer) > 5 > then
        point towards mouse-pointer
        move 1 steps
    if < touching color 🟫 ? > then
        switch costume to hit
        say Nooooo! for 1 secs
        switch costume to normal
        go to x: -190 y: -150
```

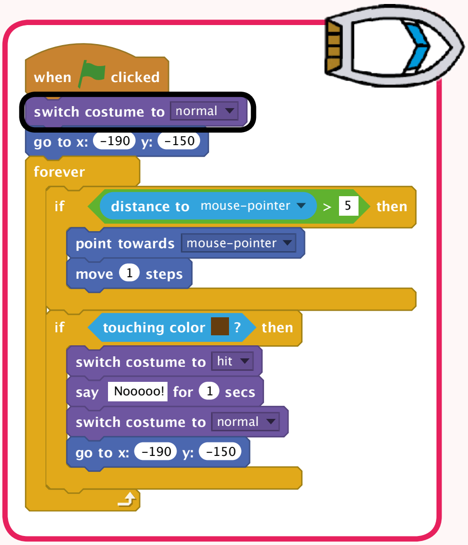
This code is inside the **forever** loop, so that your code keeps checking if the boat has crashed each time it moves.

Note: To set the correct colour, click the colour square in the **touching color** block, then click a part of the brown scenery on the stage.

TIP! SELECT TOOL

Using the Select tool, click and drag to select an area of the sprite. Drag the selected area to move it, or click its top 'handle' and drag left/right to rotate it.

☐ You should also make sure that your boat always starts a new game looking like it's 'normal'. Add this block to the start of your boat's script (outside of the **forever** block).



```
when 🏁 clicked
switch costume to normal ▾
go to x: -190 y: -150
forever
    if <(distance to mouse-pointer ▾) > 5> then
        point towards mouse-pointer ▾
        move 1 steps
    if <touching color ■ ?> then
        switch costume to hit ▾
        say Nooooo! for 1 secs
        switch costume to normal ▾
        go to x: -190 y: -150
```

CHALLENGE

WINNING!

Can you add another if block to your boat's code, so that the player wins when they get to the desert island?

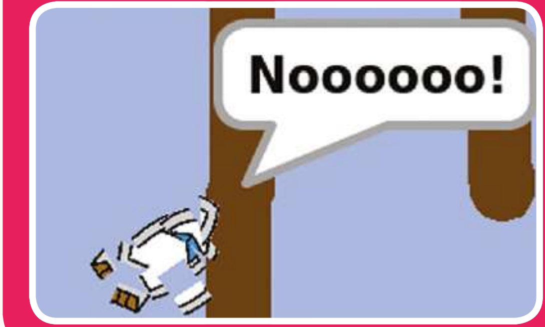
When the boat gets to the yellow desert island, it should say 'YEAH!' and then the game should stop.

HINT!

You'll need to use a stop all block to stop all scripts when the boat finishes.

☐ TEST YOUR PROJECT

Now if you try to sail through a wooden barrier, your boat should crash and move back to the start. When finished, click the red Stop button.



CHALLENGE

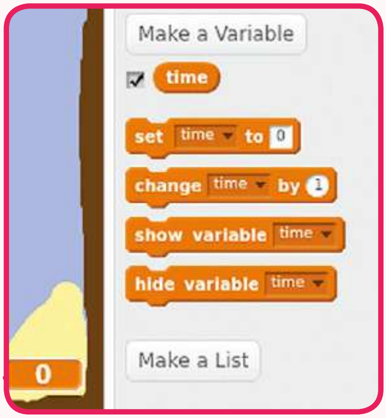
SOUND EFFECTS

Can you add sound effects to your game, for when the boat crashes, or reaches the island at the end? You could even add background music (see the previous 'Rock Band' project if you need help with this).

STEP 3: TIME TRIAL

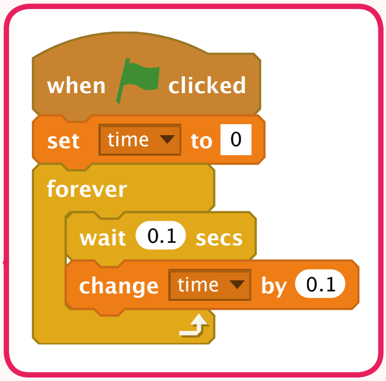
Let's add a timer to your game, so that the player has to get to the desert island as fast as possible.

☐ Add a new variable called time to your stage. You can also change the display of your new variable. If you need help, have a look at the 'Ghost Catcher' project.



```
Make a Variable
☑ time
set time ▾ to 0
change time ▾ by 1
show variable time ▾
hide variable time ▾
Make a List
```

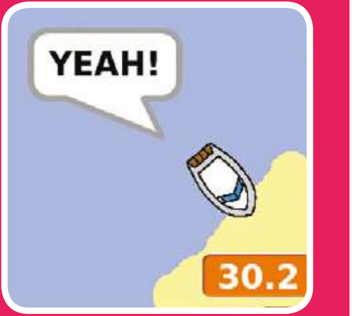
☐ Add this code to your **Stage**, so that the time variable counts up, starting at 0:



```
when 🏴 clicked
set time ▾ to 0
forever
    wait 0.1 secs
    change time ▾ by 0.1
```

☐ TEST YOUR PROJECT

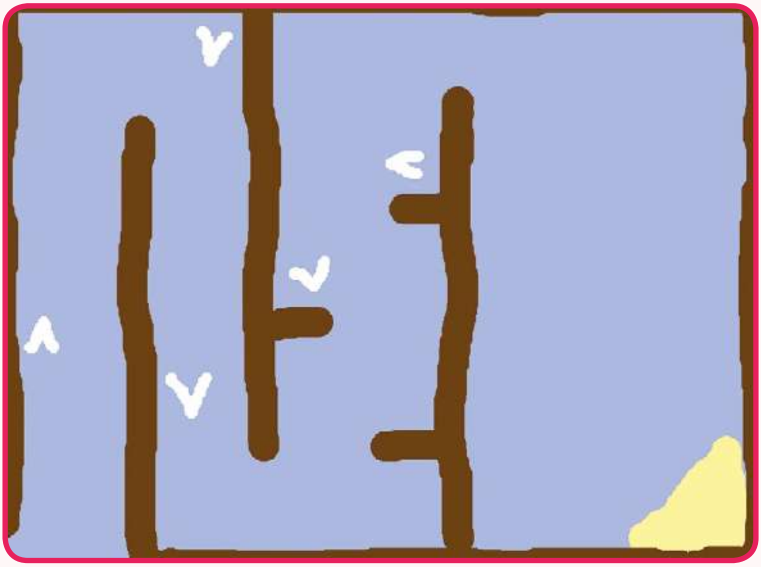
That's it! Test out your game and see how quickly you can get to the desert island!



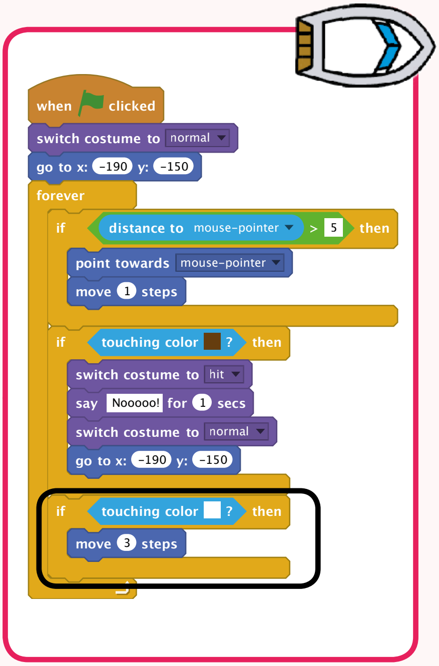
STEP 4: OBSTACLES AND POWER-UPS

This game is far too easy – let's add things to make it more interesting!

☐ First let's add some 'boosts' to your game, which will speed up the boat. Click the Stage, then the Backdrops tab, and add some white booster arrows.

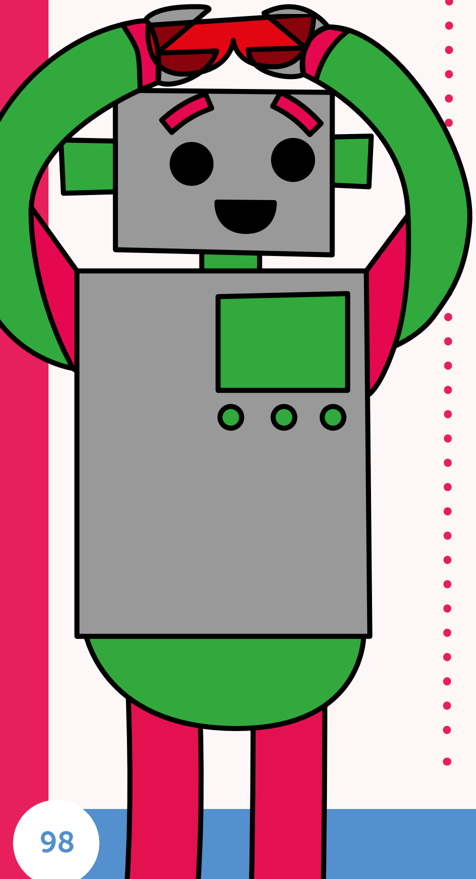


☐ You can now add some code to your boat's forever loop, so that it moves 3 extra steps if touching a white booster.



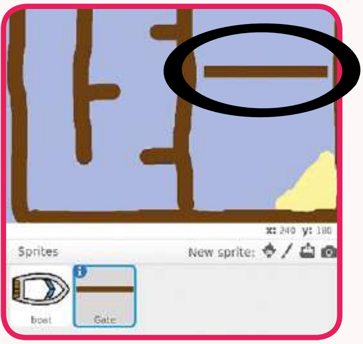
```
when ⚑ clicked
switch costume to normal
go to x: -190 y: -150
forever
  if distance to mouse-pointer > 5 then
    point towards mouse-pointer
    move 1 steps
  end
  if touching color 🟫 ? then
    switch costume to hit
    say Nooooo! for 1 secs
    switch costume to normal
    go to x: -190 y: -150
  end
  if touching color ⬜ ? then
    move 3 steps
  end
end
```

☐ Test your new code. Does your boat speed up when it touches a white booster?

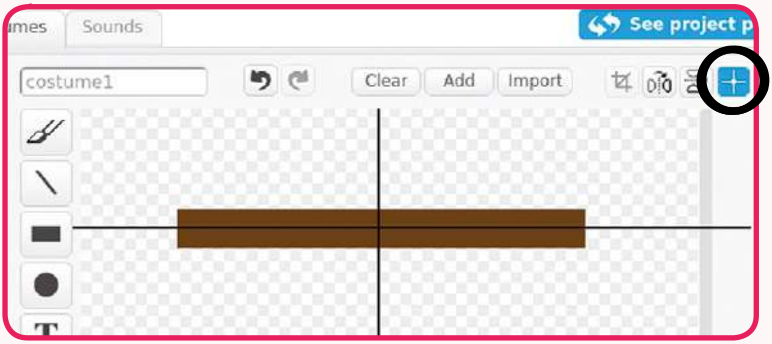


☑ You can also add in a spinning gate, which your boat has to avoid. Draw a new sprite called **Gate**, which looks like this...

Make sure that the colour of the gate is the same as the other wooden barriers.



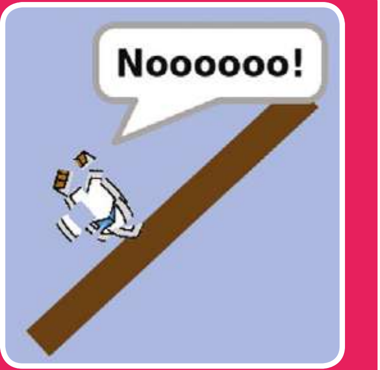
☑ Set the centre of the gate sprite by clicking the **Set costume centre** button and clicking in the centre of the rectangle.



☑ Add code to your gate, to make it spin slowly forever. Tip: Look at the code for the monkey sprite in the 'Lost in Space' project.

☑ **TEST YOUR PROJECT**

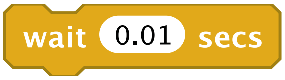
Test out your game. You should now have a spinning gate that you must avoid.



CHALLENGE

MORE OBSTACLES!

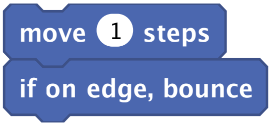
- You could add green slime to your backdrop, which slows the player down when they touch it. You can use a wait block to do this:



```
wait 0.01 secs
```

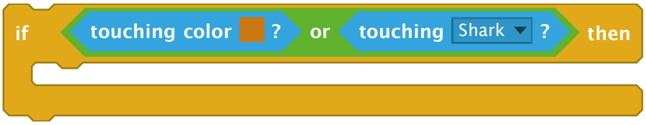
- You could add another moving object, like a log or a shark!

- These blocks may help you:



```
move 1 steps
if on edge, bounce
```

- If your new object isn't brown, you'll need to add to your boat code:



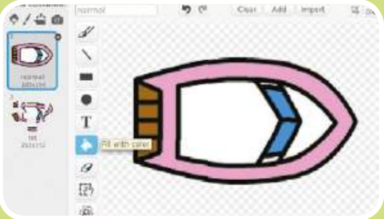
```
if < <touching color ■ ?> or <touching Shark ▼ ?> > then

end
```

CHALLENGE

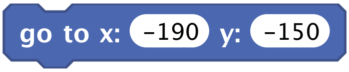
MORE BOATS!

Can you turn your game into a race between two players?

- Duplicate the boat sprite and change its colour.

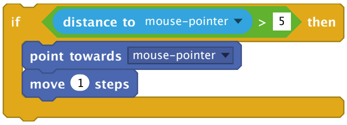


- Change Player 2's starting position, by changing this code:



```
go to x: -190 y: -150
```

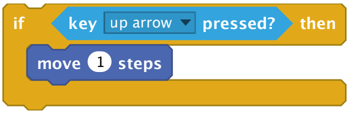
- Delete the code that uses the mouse to control the boat:



```
if < (distance to mouse-pointer ▼) > 5 > then
    point towards mouse-pointer ▼
    move 1 steps
end
```

Replace it with code to control the boat using the arrow keys.

- This is the code you'll need to move the boat forward:



```
if < key up arrow ▼ pressed? > then
    move 1 steps
end
```

You'll also need code to turn the boat when the left and right arrow keys are pressed.

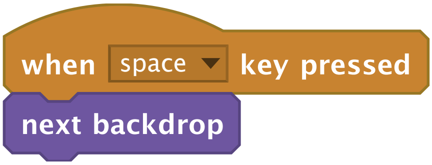
CHALLENGE

MORE LEVELS!

Can you create additional backdrops, and allow the player to choose between levels?

What will your new level look like? Sketch it out below and label the finish and any obstacles.

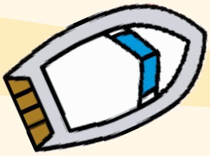
Here's some code you can add to your Stage to switch between levels:



```
when space ▼ key pressed
next backdrop
```

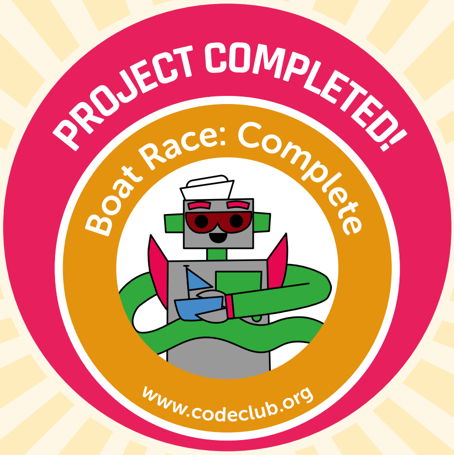
Draw a backdrop idea...

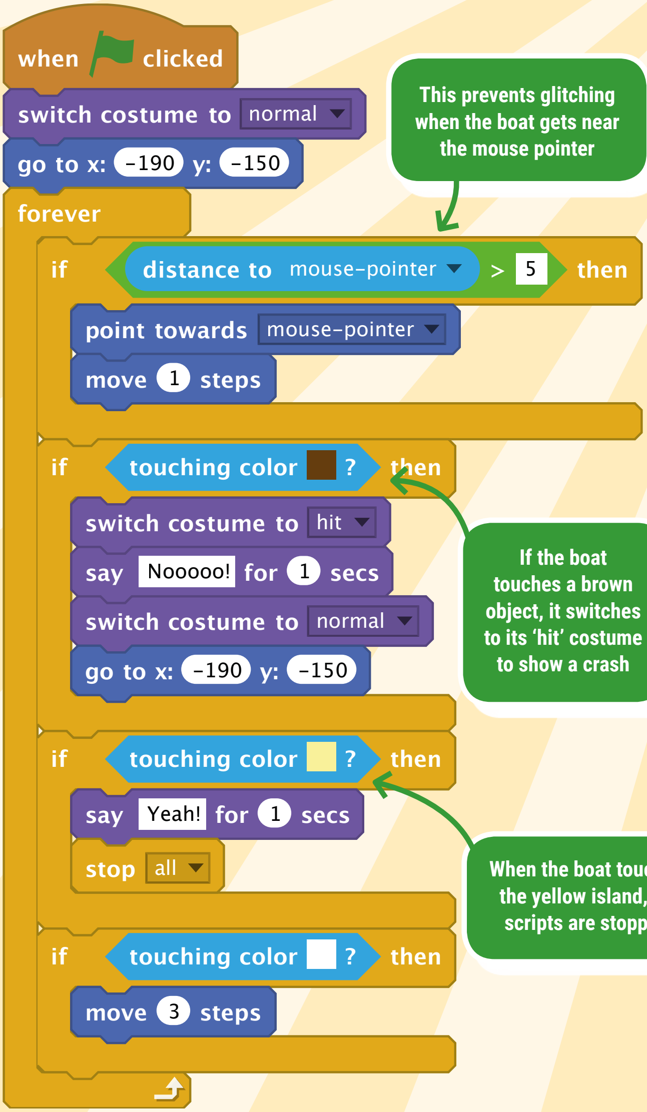
BOAT RACE FULL CODE LISTING



BOAT

Steered using the mouse pointer, the boat must be guided safely around the course.





```
when 🏴 clicked
switch costume to [normal ▼]
go to x: (-190) y: (-150)
forever
    if <(distance to [mouse-pointer ▼]) > [5]> then
        point towards [mouse-pointer ▼]
        move (1) steps
    end
    if <touching color [🟫]?> then
        switch costume to [hit ▼]
        say [Nooooo!] for (1) secs
        switch costume to [normal ▼]
        go to x: (-190) y: (-150)
    end
    if <touching color [🟨]?> then
        say [Yeah!] for (1) secs
        stop [all ▼]
    end
    if <touching color [⬜]?> then
        move (3) steps
    end
end
```

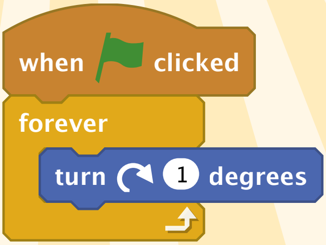
This prevents glitching when the boat gets near the mouse pointer

If the boat touches a brown object, it switches to its 'hit' costume to show a crash

When the boat touches the yellow island, all scripts are stopped

GATE

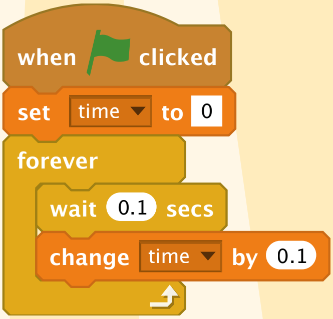
This continually spinning gate provides a tricky obstacle.



```
when 🏴 clicked
forever
    turn ↻ (1) degrees
end
```

STAGE

This code uses a variable to manage the on-screen timer.



```
when 🏴 clicked
set [time ▼] to [0]
forever
    wait (0.1) secs
    change [time ▼] by (0.1)
end
```

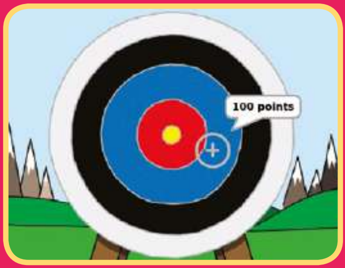
Now You Could Make...

You'll find lots more cool projects at rpf.io/ccprojects, including...

ARCHERY

Create an archery game, in which you have to shoot arrows as close to the bull's-eye as you can.

rpf.io/archery



BEAT THE GOALIE

Create a football game in which you have to score as many goals as you can in 30 seconds.

rpf.io/beat-the-goalie

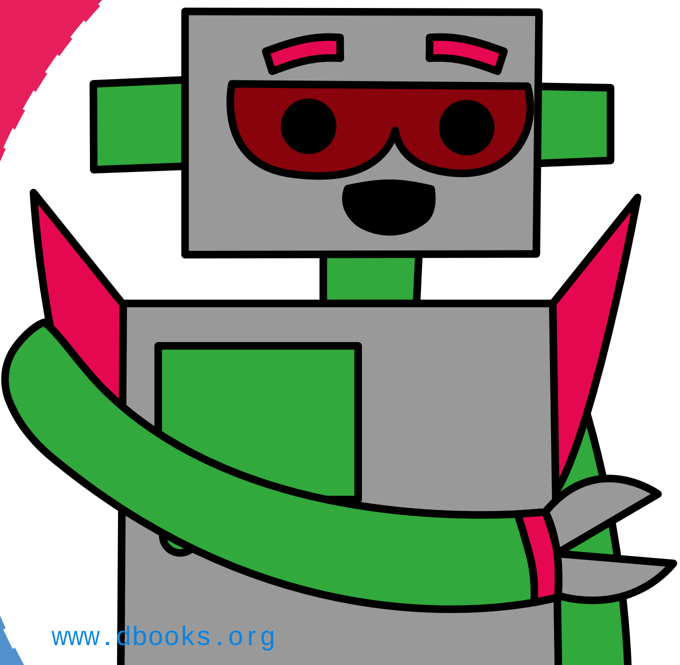
ADA'S POETRY GENERATOR

Learn how to create randomly generated poems! You will be using variables and selecting random items from lists in this poetic programming project.

rpf.io/ada-poetry

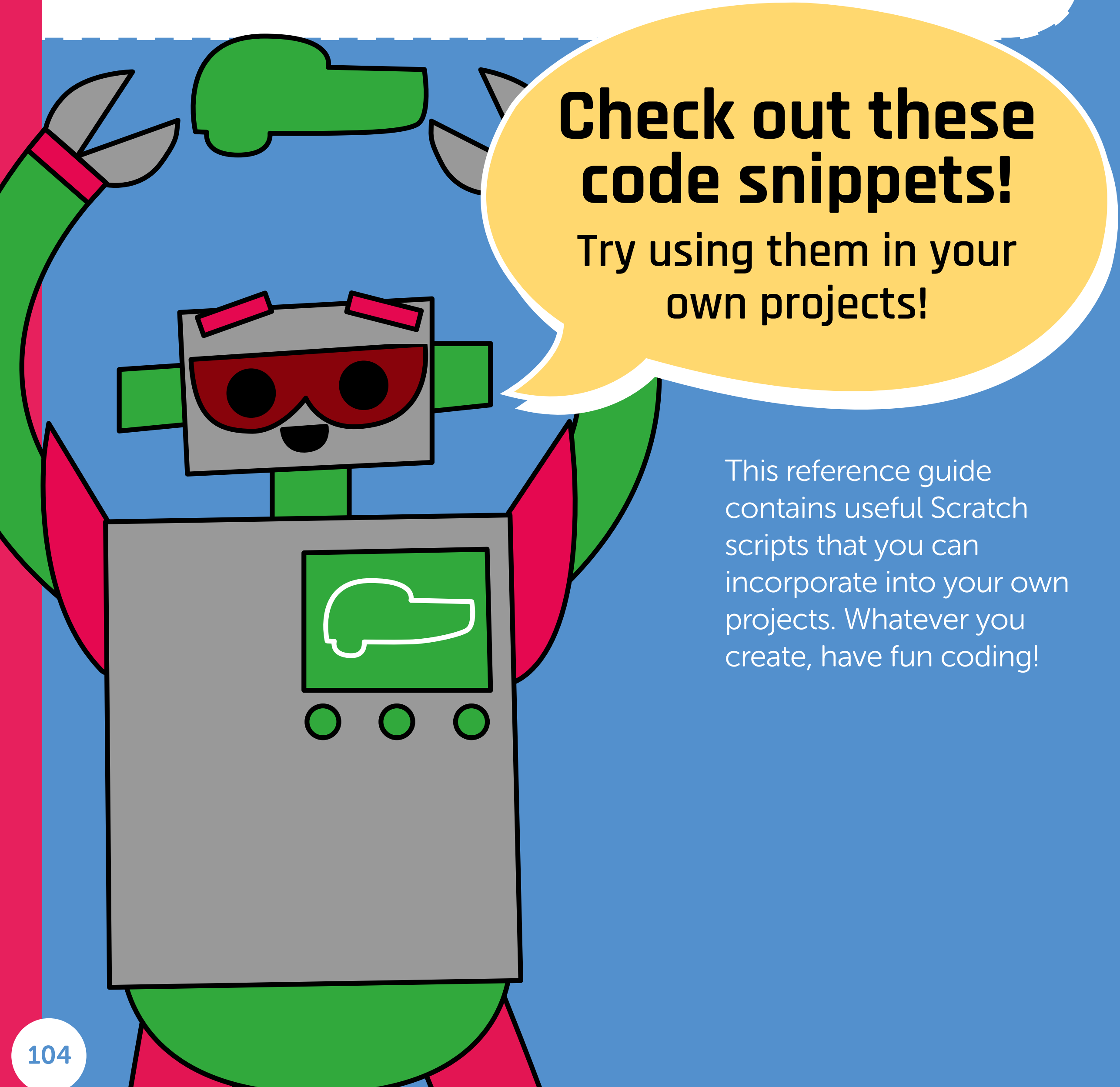
Want some handy code snippets?

Turn the page to find some useful scripts...

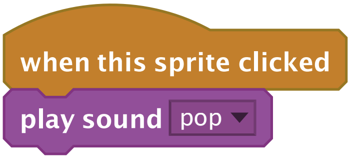
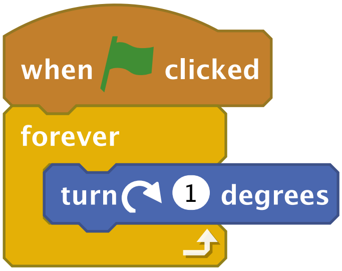
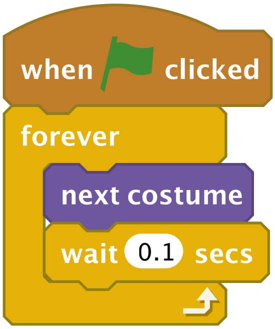
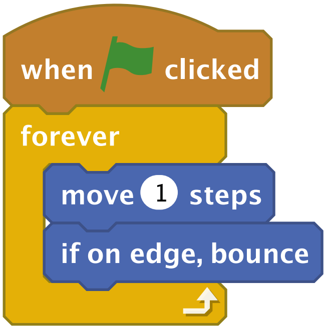


Useful Code

This chapter lists some useful code that you can use in your projects

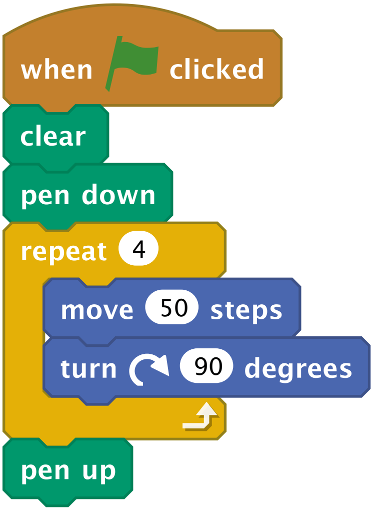

Check out these code snippets!
Try using them in your own projects!

This reference guide contains useful Scratch scripts that you can incorporate into your own projects. Whatever you create, have fun coding!

DESCRIPTION	CODE
Playing a sound	 when this sprite clicked play sound pop ▾
Spinning sprite	 when ⚑ clicked forever turn ↻ 1 degrees
Animating sprite costumes	 when ⚑ clicked forever next costume wait 0.1 secs
Bouncing sprite	 when ⚑ clicked forever move 1 steps if on edge, bounce

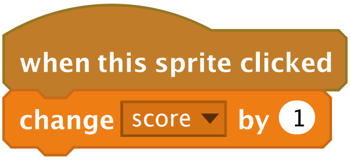
DESCRIPTION **CODE**

Drawing a square

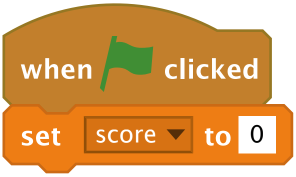


```
when ⚑ clicked
clear
pen down
repeat 4
    move 50 steps
    turn ↻ 90 degrees
pen up
```

Keeping score

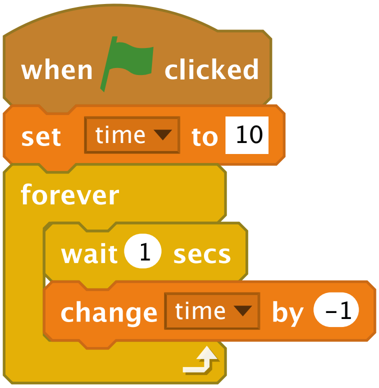


```
when this sprite clicked
change score ▾ by 1
```



```
when ⚑ clicked
set score ▾ to 0
```

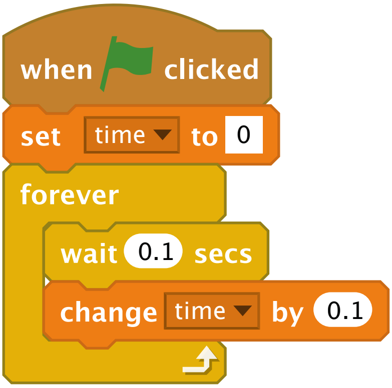
Timer counting down



```
when ⚑ clicked
set time ▾ to 10
forever
    wait 1 secs
    change time ▾ by -1
```

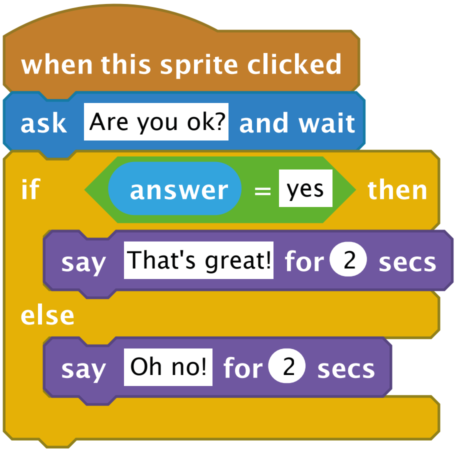
DESCRIPTION | **CODE**

Timer counting up



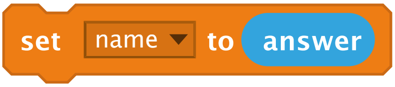
```
when 🏁 clicked
set time ▾ to 0
forever
    wait 0.1 secs
    change time ▾ by 0.1
```

Asking a question and responding to the answer



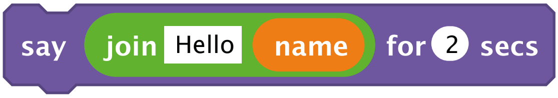
```
when this sprite clicked
ask Are you ok? and wait
if answer = yes then
    say That's great! for 2 secs
else
    say Oh no! for 2 secs
```

Storing the answer to a question in a variable



```
set name ▾ to answer
```

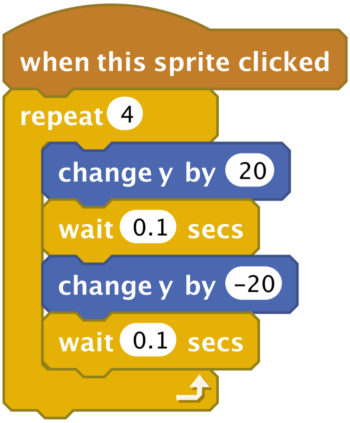
Joining text together



```
say join Hello name for 2 secs
```

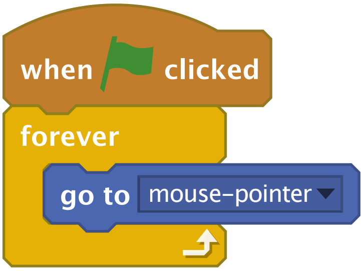
DESCRIPTION	CODE

Jumping sprite



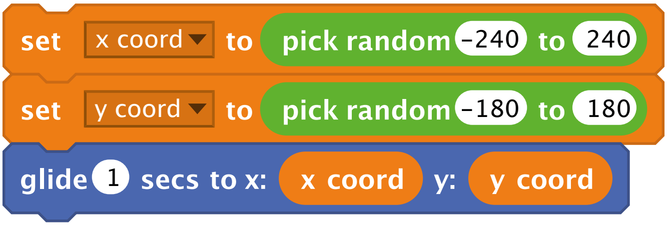
```
when this sprite clicked
repeat (4)
    change y by (20)
    wait (0.1) secs
    change y by (-20)
    wait (0.1) secs
```

Following the mouse



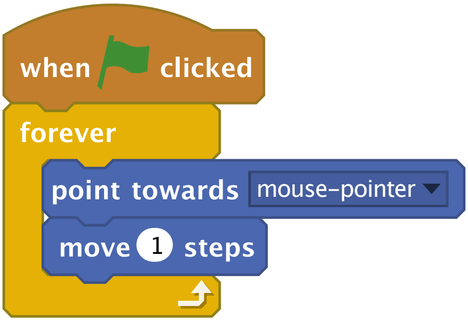
```
when ⚑ clicked
forever
    go to [mouse-pointer ▾]
```

Glide to random stage co-ordinates



```
set [x coord ▾] to (pick random (-240) to (240))
set [y coord ▾] to (pick random (-180) to (180))
glide (1) secs to x: (x coord) y: (y coord)
```

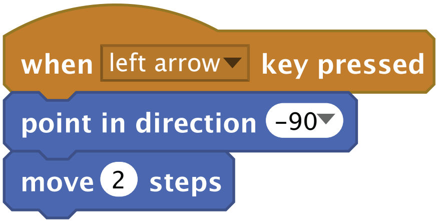
Movement towards the mouse



```
when ⚑ clicked
forever
    point towards [mouse-pointer ▾]
    move (1) steps
```

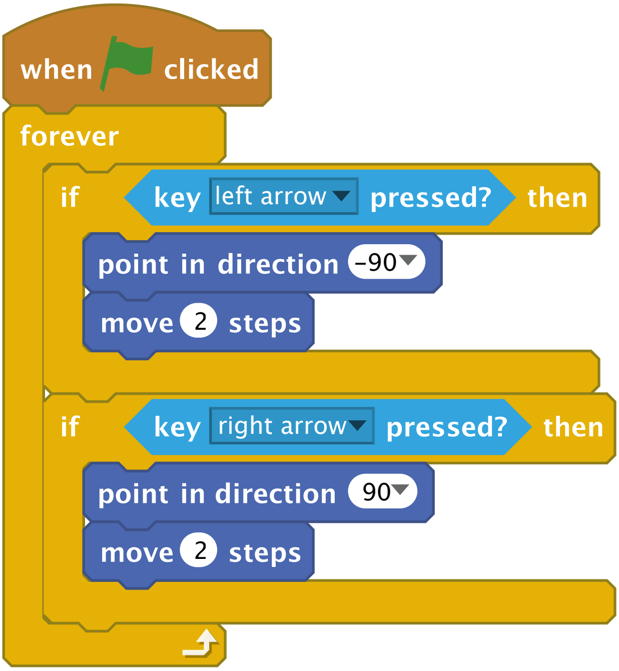
DESCRIPTION	CODE

Movement using the keyboard



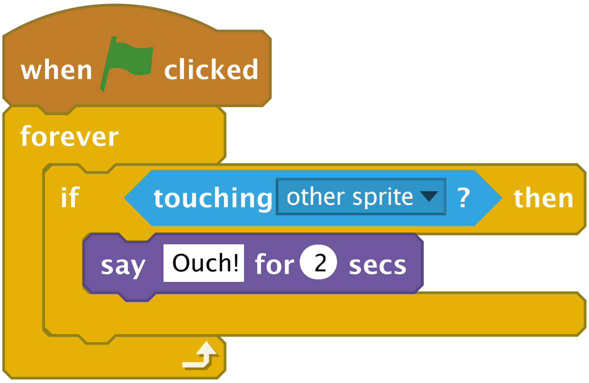
```
when [left arrow ▾] key pressed
point in direction (-90 ▾)
move (2) steps
```

or...



```
when 🏁 clicked
forever
    if <key [left arrow ▾] pressed?> then
        point in direction (-90 ▾)
        move (2) steps
    if <key [right arrow ▾] pressed?> then
        point in direction (90 ▾)
        move (2) steps
```

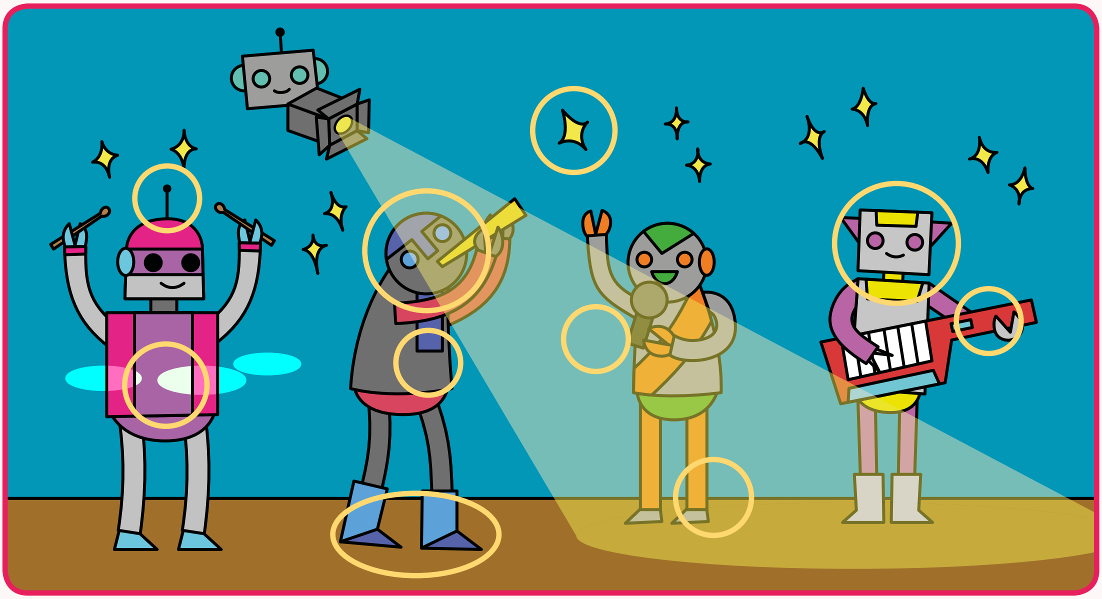
Check to see if a sprite has hit another sprite



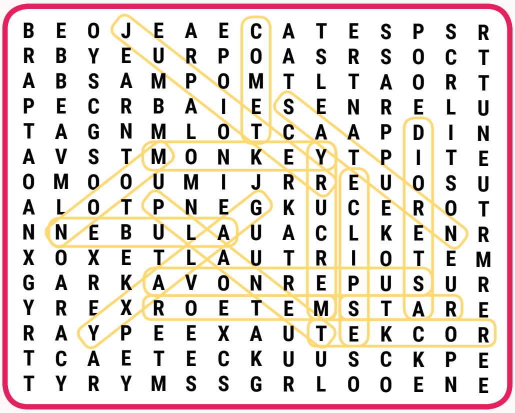
```
when 🏁 clicked
forever
    if <touching [other sprite ▾] ?> then
        say [Ouch!] for (2) secs
```

Puzzle Answers

SPOT THE DIFFERENCE



LOST IN SPACE



B	E	O	J	E	A	E	C	A	T	E	S	P	S	R
R	B	Y	E	U	R	P	O	A	S	R	S	O	C	T
A	B	S	A	M	P	O	M	T	L	T	A	O	R	T
P	E	C	R	B	A	I	E	S	E	N	R	E	L	U
T	A	G	N	M	L	O	T	C	A	A	P	D	I	N
A	V	S	T	M	O	N	K	E	Y	T	P	I	T	E
O	M	O	O	U	M	I	J	R	R	E	U	O	S	U
A	L	O	T	P	N	E	G	K	U	C	E	R	O	T
N	N	E	B	U	L	A	U	A	C	L	K	E	N	R
X	O	X	E	T	L	A	U	T	R	I	O	T	E	M
G	A	R	K	A	V	O	N	R	E	P	U	S	U	R
Y	R	E	X	R	O	E	T	E	M	S	T	A	R	E
R	A	Y	P	E	E	X	A	U	T	E	K	C	O	R
T	C	A	E	T	E	C	K	U	U	S	C	K	P	E
T	Y	R	Y	M	S	S	G	R	L	O	O	E	N	E

ASTEROID
COMET
ECLIPSE
GALAXY
JUPITER

MERCURY
METEOR
MONKEY
MOON
NEBULA

PLANET
ROCKET
SATURN
STAR
SUPERNOVA

ENTER THE CRYPT

1 G H O S T

2 W E R E W O L F

3 D E M O N

4 M U M M Y

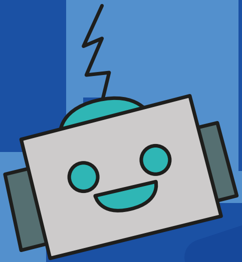
5 T R O L L

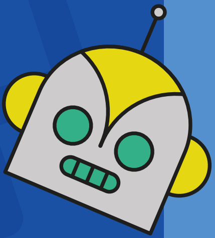
6 D E V I L

7 B U N Y I P

Code Club Book of Scratch

Volume 1





Learn to code using Scratch, the block-based programming language. In each chapter you'll find instructions to build cool games, animations, and interactive stories. Your friendly robot guide will aid you step-by-step through each project and give you handy tips along the way.

Code Club is a global network of free coding clubs where young people aged 9-13 build and share their ideas with code. There are currently more than 12 000 clubs in over 150 countries.

codeclub.org

Raspberry Pi PRESS