MAGBOOK

BUILD A GAME IN 30 MINUTES!

Coding for Kids Scratch

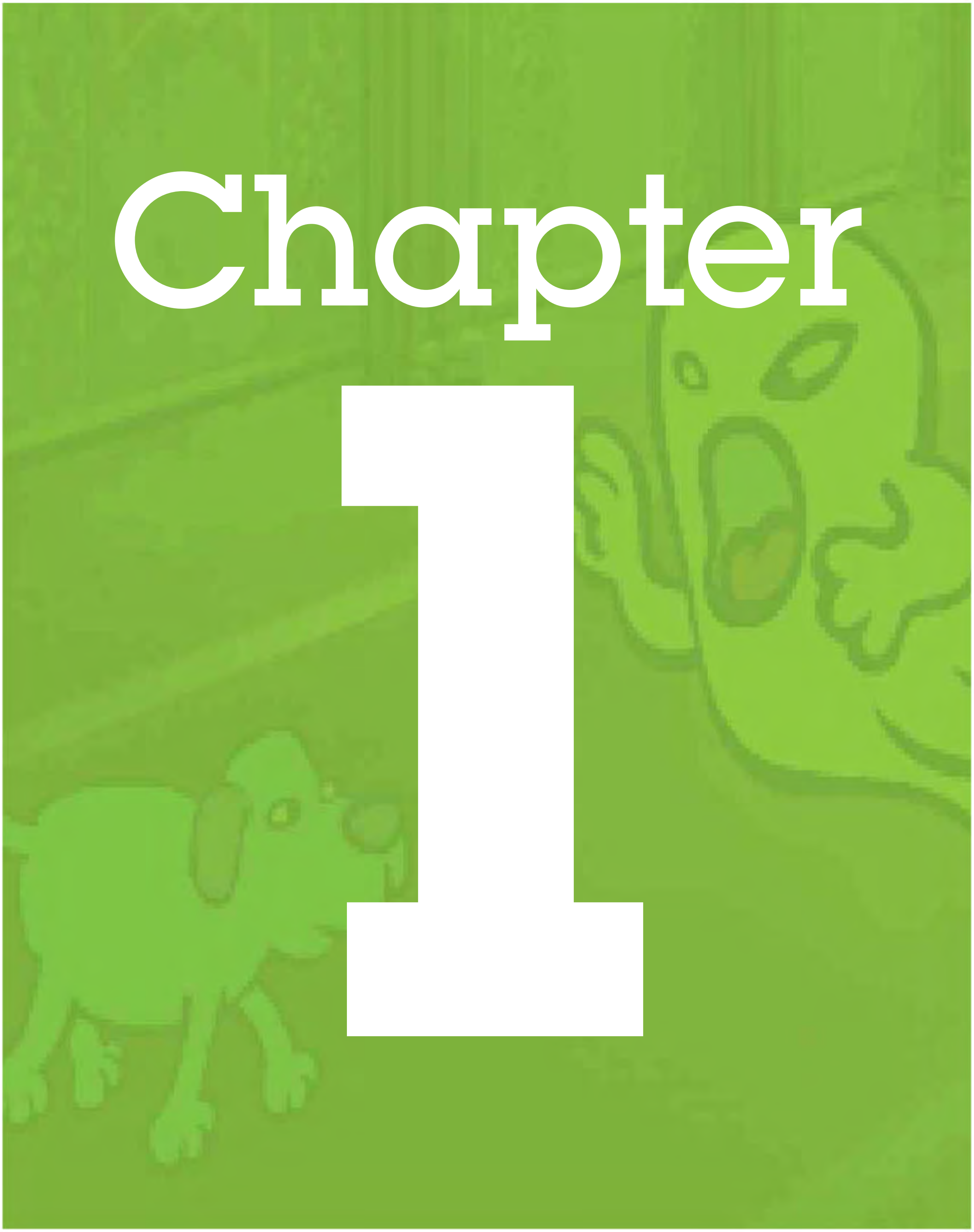
Step-by-step friendly advice

Animate stories

Make your own racing game

Create your own quiz

Chapter 1



Welcome

Anyone can code. Certainly, writing the next Minecraft or programming complex simulations from scratch will require a deeper knowledge, but anyone and everyone has the potential to learn some basic coding skills, then take those skills and write a simple program. This book can help you and your kids take that potential further. Read it, follow the projects and get to grips with the fundamentals of programming, and you and they can learn to code.

We live in a world where technology and everyday life have never been more tightly interwoven, and that technology – the hardware, websites and services we use all the time – is dependent on software. An understanding of how that software is made is as valuable in the 21st century as an understanding of engineering was in the 19th and 20th, and it's only going to grow more important.

Forget all that stuff of grave importance, though, or you may miss the fact that coding can be fun. You can make something in less than an hour, watch it work, then go back and make it better. As long as you have a computer – and almost any laptop or PC will do – you can build something brilliant, bizarre or even useful, and the tools won't cost you a penny. Coding is creative. It pushes your imagination, your ability to improvise and your ability to plan.

Most of all, it doesn't have to be difficult. In this book, we'll show you how you can use visual tools such as Scratch to build programs object by object or block by block, so that kids as young as seven or eight can make their own funny animations or playable arcade games. While they're doing it, they'll absorb fundamental concepts that will help them develop their skills later on. We also introduce SmallBASIC – a simplified version of the classic coding language, which is free to use and surprisingly easy to learn. By the time this book is over, we'll have started using Visual Basic, a tool that many professional programmers use every day.

The projects in this book are fun, so that kids and adults will enjoy making them, and playing them once they're done. They're also easy to customise, so that novice programmers can take what we've put together, change it and make their own mark. That's important, because programming isn't about using technology, but about taking it apart, seeing how it works and making it better. We hope that you and your kids will take these projects, improve them and make them your own. There's no better start on a programmer's journey.

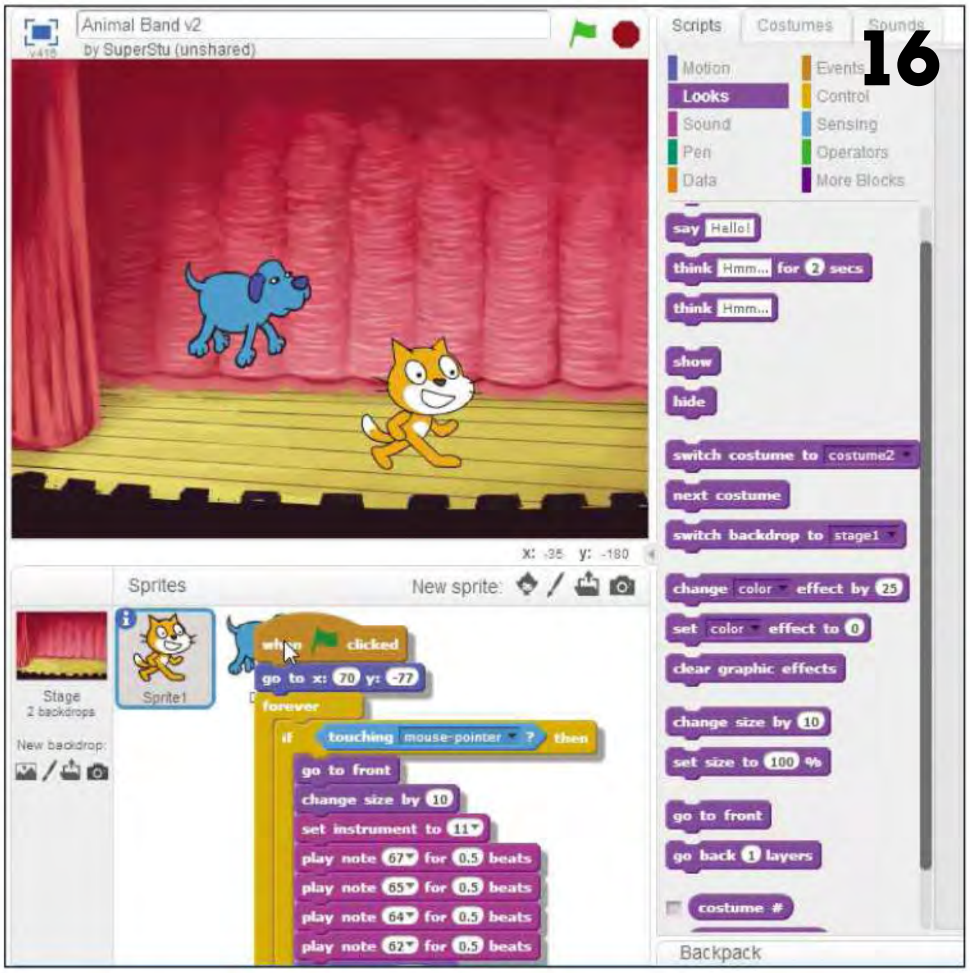
Stuart Andrews
Editor

Stuart Andrews is a technology journalist specialising in PCs, games and business and educational IT. He writes for a range of computing and technology magazines and websites, including *ICT Reviews*, *PC Pro*, *Cloud Pro*, *ComputerActive*, *TrustedReviews* and *The Sunday Times*. His first computer was a Sinclair ZX Spectrum, but he now uses a range of tablets, Chromebooks and Windows 8 PCs, none of which have rubber keys. You can follow him on Twitter at @SATAndrews.

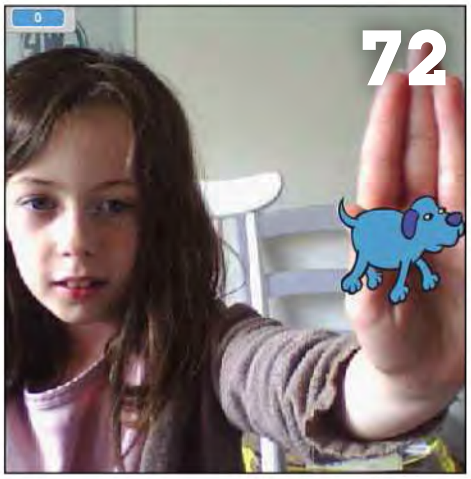
Andrew Dixon has been teaching ICT for eight years. A Computer Science graduate, he's always had a passion for the unlikely combination of programming and playing rock guitar. He is currently Head of ICT and Computing at Summerhill School in the West Midlands and was nominated for Teacher of the Year in 2009. A self-confessed coding geek and gadget junkie, Andrew is always pursuing new and emerging technologies in education and writes regularly for *ICT Reviews* and *PC Pro*. You can follow him on Twitter at @ADXeventide.

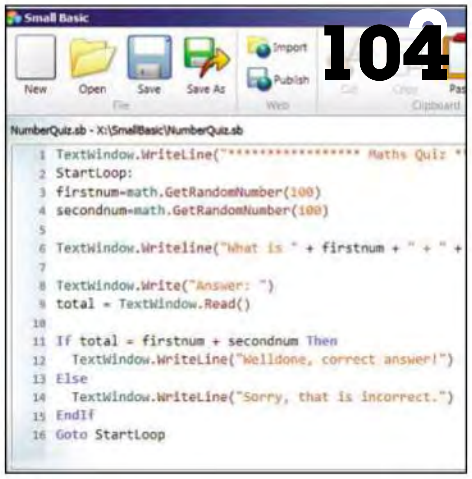
Many of the projects in this book use Scratch. Scratch is developed by the Lifelong Kindergarten Group at the MIT Media Lab. See scratch.mit.edu

Contents



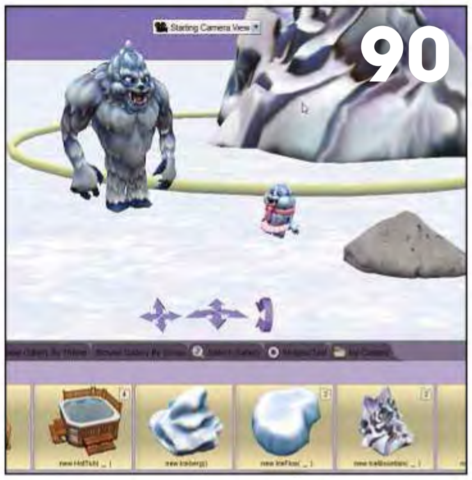
```
when [flag] clicked
go to x: 70 y: -77
forever
    if touching mouse-pointer ? then
        go to front
        change size by 10
        set instrument to 11
        play note 67 for 0.5 beats
        play note 65 for 0.5 beats
        play note 64 for 0.5 beats
        play note 62 for 0.5 beats
```





NumberQuiz.sb - X:\SmallBasic\NumberQuiz.sb

```
TextWindow.WriteLine("****************** Maths Quiz *
StartLoop:
firstnum=math.GetRandomNumber(100)
secondnum=math.GetRandomNumber(100)

TextWindow.Writeline("What is " + firstnum + " + " +

TextWindow.Write("Answer: ")
total = TextWindow.Read()

If total = firstnum + secondnum Then
  TextWindow.WriteLine("Welldone, correct answer!")
Else
  TextWindow.WriteLine("Sorry, that is incorrect.")
EndIf
Goto StartLoop
```



Section 1 Start coding

The ability to program has never been as important as it is right now. In a digital age, an understanding of code and how it works is an incredibly useful skill. It can transform you from someone who uses other people's software into someone who can make it. It can help you get to grips with real computing, and it can be a fun and interesting pastime. It might even one day help you get a job. Right now, this knowledge might seem out of reach, but with the aid of the projects on the next few pages and an easy, graphical programming tool called Scratch, we're going to help you through the fundamental concepts, and steadily teach you how to code. Before you even know it, you'll have taken your first steps into a world of programming and be hungry for more.

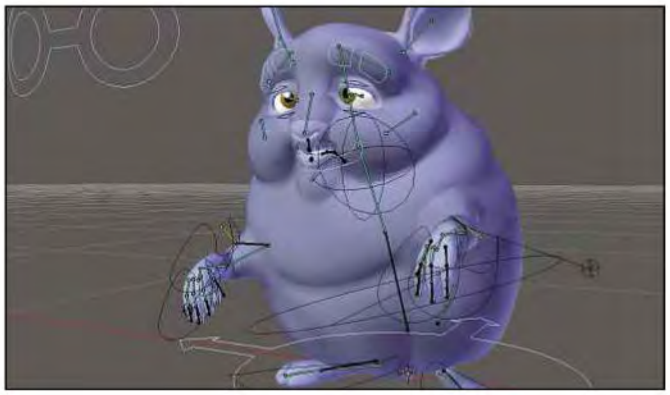
IN THIS SECTION

Why learn to code?

In a digital age, being able to code has become a vital skill. Discover how to create something brilliant using just a PC, a screen and free software

Code is everywhere, and not just where you might think. When you're running apps on a smartphone or playing games on a console, it seems obvious that the app you're using or the game you're playing has been put together by programmers, using lines of code to stitch together every last feature, every button you tap and everything you see on the screen, so that it all does the job it's meant to do.

This is why learning to code is important. It can take you from someone who can use technology to someone who can create technology or make amazing things with it. It could also be the passport to an incredible future. You could one day be helping to build the next iPhone, create a blockbuster film or help a Formula One racing team make faster cars, all by using your coding skills.


▲ Computer-generated movies exist because programmers developed the software to produce them, and work with artists to code more lifelike or advanced effects. Blender is a free 3D graphics package, developed by hundreds of programmers working together.

Most importantly, coding can be fun. Often, you're creating something brilliant from nothing, using just a computer, a screen and some free software. Change something in your program and you can see the result on the screen, and it's pretty hard to make any mistakes that you won't be able to fix. Coding can be like solving a puzzle. You know what you want to do, or that something isn't working properly, and it's exciting to find a way to make it better. Whatever you're doing, you can do something brilliant, and make it your own.

WHAT IS CODING?

When it comes to computing, code means a set of written instructions for a computer, usually arranged in a structure called a program. When a user runs the program, the code tells the computer what to show on the screen, how to process any data that the program uses, and what to do if certain things happen; for example, if a button is pressed. Programming – or coding – is the art of writing those instructions so that the computer can understand them, and the program functions as it should. It's also the art of arranging those instructions so that the program works as smoothly and as quickly as it can, and doing all this in a way that other programmers can follow if they need to look at or change your code.

What can this book do to help?

This isn't one of those books that tells you all about programming but not how to do it. Nor is it one of those books that gives you lots of code to type in, but doesn't tell you how it works or what it's doing. Instead, we're going to take you through a series of projects that will introduce the most important concepts, help you use the key building blocks of code, and enable you to create fun programs that you can then go back and change. We're going to start off easy and slowly add the more complex

Without programmers, we wouldn't have amazing games like Minecraft. Learn to code, and you might help create the next big gaming hit.

"You might start as the young apprentice, but you'll finish feeling like a coding Jedi Master"

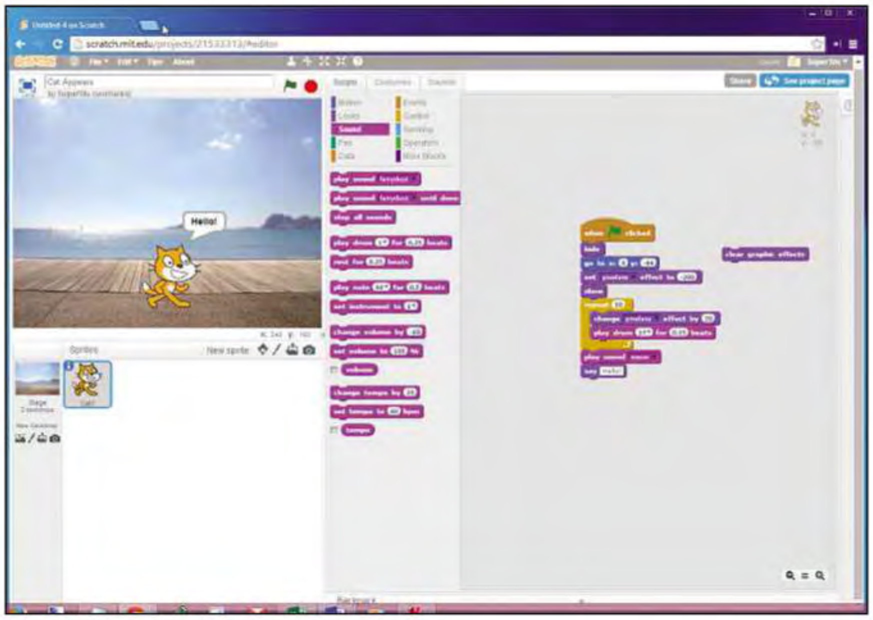
stuff, so that you're never left drowning in a sea of jargon, or having your mind boggled by big chunks of code that make your eyes and brain hurt. You can start the book knowing almost nothing, but by the end you'll feel confident enough to explore the world of coding further. You might start as the young apprentice, but you'll finish feeling like a coding Jedi Master.

The projects are designed so that they can be completed by younger children with a little parental help, or by older children working on their own.

How does it work?

Our projects kick off with Scratch – a highly visual, easy-to-use programming tool that was designed to introduce the main concepts of programming, and help young novice coders build something good with minimum fuss. We'll then move onto SmallBASIC, a refined version of the classic coding language, designed to get young programmers used to working with a proper text-based programming environment. Finally, we'll look at projects that use Visual Basic Express – a free version of the same tool being used by millions of professional programmers around the world.

Along the way, we'll tell you what you're going to learn, take each project apart, and pick out all the vital bits of code that make the program work as it should, or that you might want to come back to and change later on. After all, these aren't our fully finished projects – they're starting points for your own. ●



Coding doesn't have to be difficult. Use easy tools like MIT's Scratch, and almost anyone can do it.

Introducing Scratch

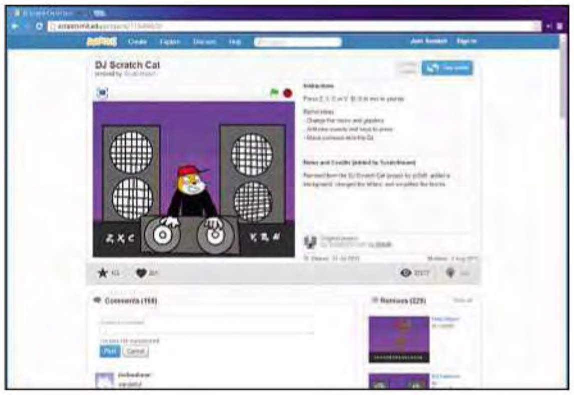
With a drag-and-drop approach, Scratch is the perfect way to start coding

WHAT YOU'LL LEARN

- What Scratch is and what it does
- How to join in and start using it
- What a sprite is, and how you use it

Originally developed by computer scientists at America's Massachusetts Institute of Technology (MIT), Scratch is a simple, visual programming language that you can use to create cartoon animations, interactive stories and simple games. It's designed for kids aged eight to 16, but it's a good way for someone of any age to learn the basics of programming. While a Scratch program might not look like what you'd think of as a program, with its chunky blocks you drag and click together, it still works like one and uses the basic parts that you'd find in a real program.

The great thing about Scratch is that it takes away a lot of the complexity of programming and leaves you free to think about how the program needs to work and what it needs to do. You don't need to worry about writing your code in the right way so that your computer can understand it. You just drag blocks into the Scripts space and click them together, a bit like blocks of Lego. Make a mistake and put things in the wrong order, and you can easily separate the blocks, change, delete or move them around.



▲ Scratch projects are designed for sharing. You can try other people's, and transform them with your own 'remix' projects.

It's hard to imagine a more intuitive way to code.

Scratch doesn't just give you all the building blocks for a program, but also a whole grab-bag of great stuff that you can use in them. You'll find a wide range of cartoon characters to star in your program – everything from dogs and dinosaurs to ghosts, aliens and penguins. You'll find a selection of background scenery, and a library of musical instruments, drums and sound effects. And if Scratch doesn't have what you're looking for, it's easy to import your own stuff from your computer or use the simple, built-in tools to make new characters, scenes and sounds.

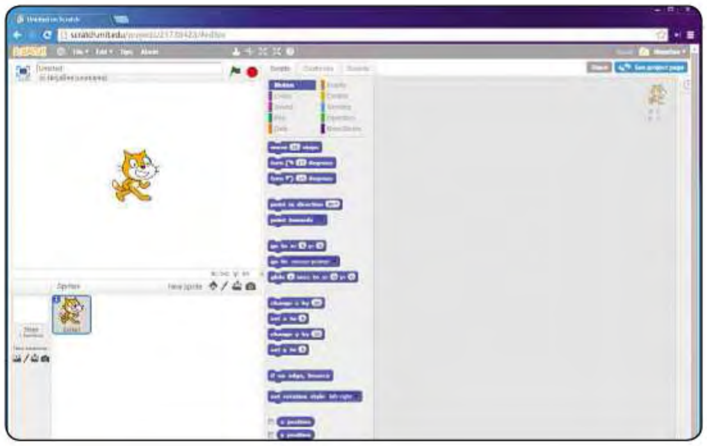
The other great thing about Scratch is that you don't have to learn on your own. The Scratch website (scratch.mit.edu/) is the centre of a huge community, where you can try other users' programs or get help, hints and tips on making your own. It's also where you can eventually share your own programs, so that when you've made something that you're proud of you can let your friends and family try it out. Plus, with millions of Scratch users out there, you could make a name for yourself as a Scratch coding superstar! ●

ABOUT SPRITES

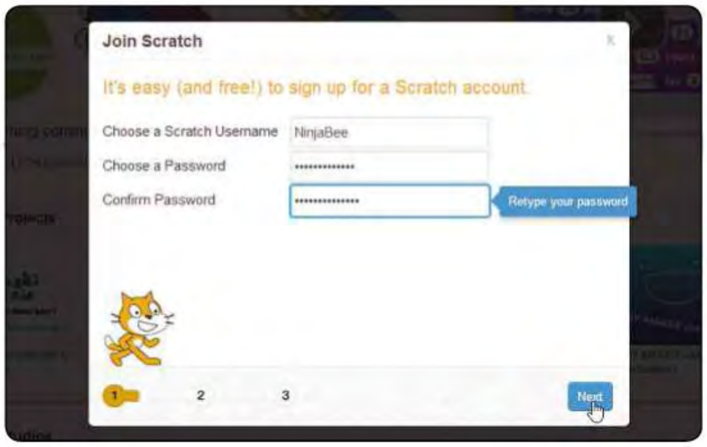
The most important things in any Scratch project are the sprites. Sprites are the characters or objects that move around or do things in your program. They're the heroes and the villains, the actors in your cartoons, the cars and spaceships that you might set racing around. You program what the sprites do and when they do it by giving them instructions in scripts. These tell your sprites what to do and where to go, what to say or what noise to make. They also tell your sprite what to do when something happens – for instance, when it hits another sprite.

The default sprite – the one that automatically appears in any new project – is the Scratch cat, but he doesn't have to be the star of your program. You can use any sprite from the large library, or make your own. The great thing about sprites is that, once you've added one and built a script, you can duplicate it, change it and use it again in the same project, or even export it – make a copy and send it out of your project – so that you can use it in a completely different project.

Signing up for Scratch



STEP 1 You don't need to install anything to start coding with Scratch. Switch on your computer, fire up your browser, and type scratch.mit.edu into the browser and the page loads. Click on the Create button or on the Try it Out button below, and the Scratch editor loads, complete with step-by-step instructions to try your first sample program.



Join Scratch

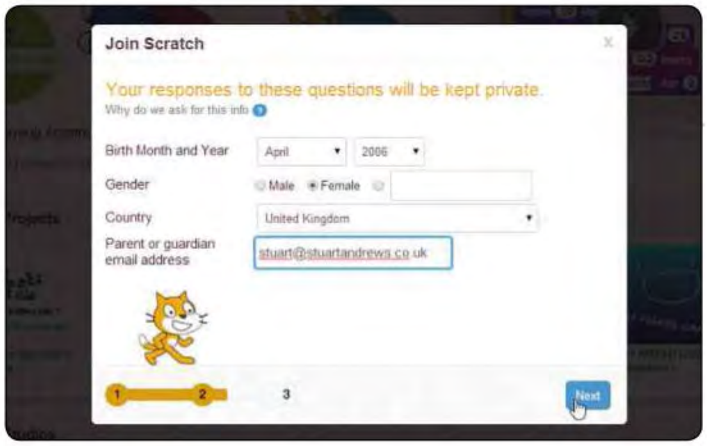
It's easy (and free!) to sign up for a Scratch account

Choose a Scratch Username: NinjaBee

Choose a Password: ••••••••••••••

Confirm Password: •••••••••••••• Retype your password

STEP 2 To really get to grips with Scratch, though, you need to sign up. Click on the Join Scratch button in the bar at the top. Now, go to the window that pops up and choose a Scratch Username, then enter a password. You'll need to enter the same password in each of the two boxes to confirm it. Press Next.



Join Scratch

Your responses to these questions will be kept private

Why do we ask for this info

Birth Month and Year: April 2006

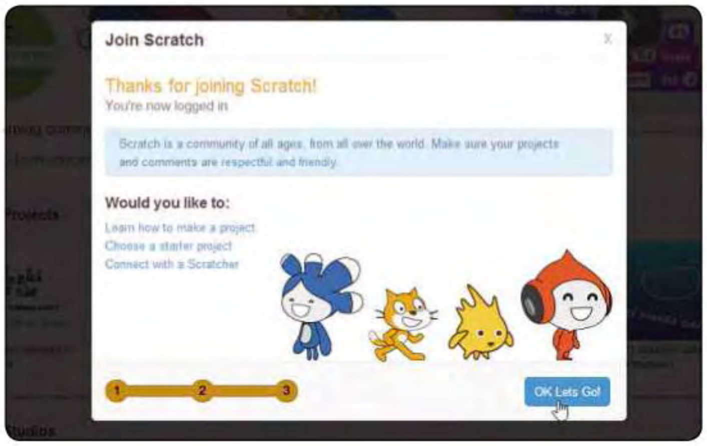
Gender: Male [✓ Female]

Country: United Kingdom

Parent or guardian email address: stuart@stuartandrews.co.uk

STEP 3 Now you need to enter the month when you were born and the year, your gender, the country where you live and an email address. This can be a parent's email address if you don't have one of your own. When you've entered all these things, click Next.

STEP 4 You're all signed up! Now it's time to get started. Under 'Would you like to:' on the left-hand side, you'll find links to take you to the sample program we mentioned earlier, choose a basic starter project to be getting on with, or get help and advice from the Scratch community's friendly Welcome Committee. The projects here do a nice job of explaining how to use Scratch, and how to behave while you're using it. Click the OK, Let's Go button to go back to the Scratch homepage, and start working on your first Scratch project.



Join Scratch

Thanks for joining Scratch!

You're now logged in

Scratch is a community of all ages, from all over the world. Make sure your projects and comments are respectful and friendly.

Would you like to:

Learn how to make a project

Choose a starter project

Connect with a Scratcher

Scratch basics

Scratch is very easy to use once you know your way around its simple interface and discover how to connect blocks of script together

WHAT YOU'LL LEARN

- What the different parts of the Scratch inferface do
- How to select instruction blocks and connect them in a stack
- All about the Toolbar and the Costumes window

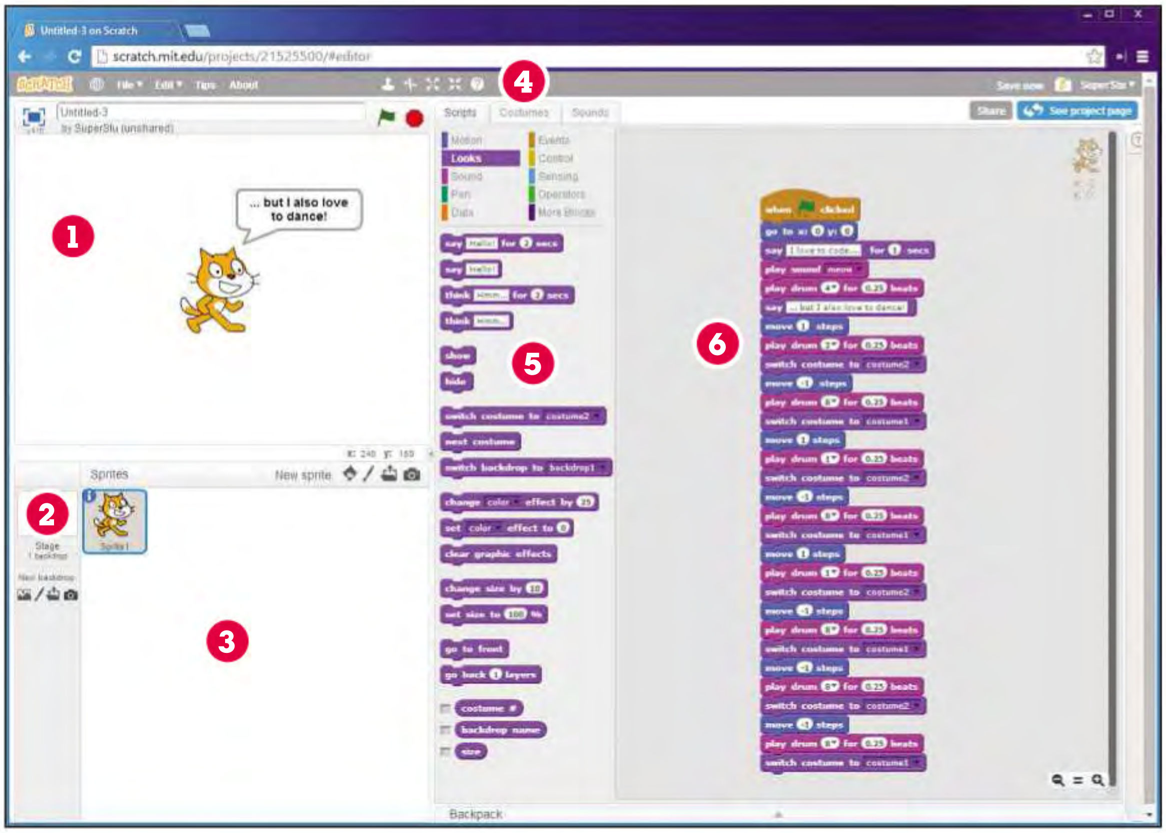
Two things make Scratch the perfect choice for novice programmers: its block-based system for building scripts, and its very simple, logical interface. The Stage (1), the area where your project plays while you're working on it, takes up the top-left corner of the window. Beneath that sits the Backdrops area (2), where you can add background scenery for the Stage, and the Sprites area (3), where you add and delete your sprites, and select them for editing.

The right-hand side of the screen is where all the work really happens. It switches between three windows, covering Scripts, Costumes and Sounds, and you can flip between the three by clicking the tabs (4) at the top of the area. Scripts is where you'll spend most of your time, dragging blocks from the Blocks Palette (5) on the left into the Scripts area on the right (6).

Putting scripts together

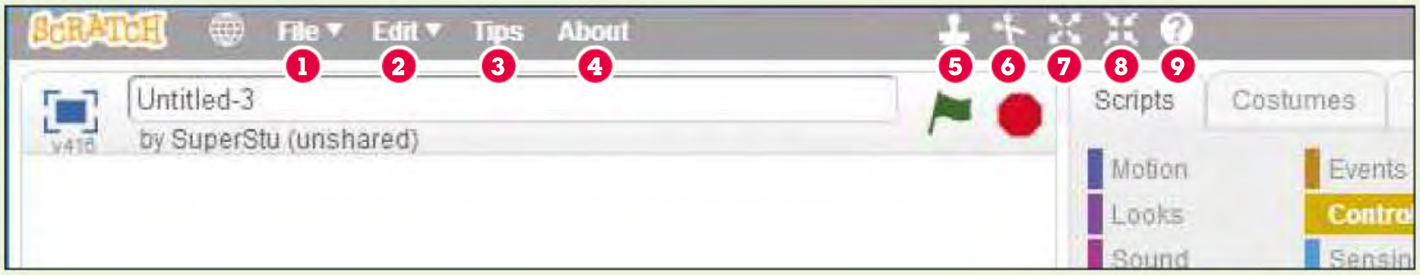
You control what a sprite does by connecting different blocks of script together. All you need to do is click on a block in the Blocks Palette, hold the left mouse button down, then drag it into the Scripts area and release the mouse button. Drag another block and release it just above or below the last one, and the two will join together in a stack. A white highlight above or below the block to which you're joining will tell you where the new block will go.

Stacks of script run from top to bottom, and once



```
when [flag] clicked
go to x: 0 y: 0
say [I love to code...] for 1 secs
play sound meow
play drum 4 for 0.25 beats
say [... but I also love to dance!]
move 1 steps
play drum 3 for 0.25 beats
switch costume to costume2
move -1 steps
play drum 8 for 0.25 beats
switch costume to costume1
move 1 steps
play drum 1 for 0.25 beats
switch costume to costume2
move -1 steps
play drum 8 for 0.25 beats
switch costume to costume1
move 1 steps
play drum 1 for 0.25 beats
switch costume to costume2
move -1 steps
play drum 8 for 0.25 beats
switch costume to costume1
move -1 steps
play drum 8 for 0.25 beats
switch costume to costume2
move -1 steps
play drum 8 for 0.25 beats
switch costume to costume1
```

THE TOOLBAR



The Toolbar at the top of the screen has all the most important Scratch menus, plus some buttons you'll need to use while working on a project.

1 **File:** Clicking on File opens up a dropdown menu, with options to start a new Scratch project, save your current project, save your current project as a copy, go to the My Stuff page to look at other projects, upload a project you've saved on your computer, or download a project so that it's saved on your computer. The Revert option resets the whole project to the state it was in when you launched Scratch and opened it. It's a useful option if you've made a total mess of things, but use it carefully as you could lose hours of work!

2 **Edit:** The Undelete option is the biggie here. Scratch makes it a bit too easy to delete a huge block of instructions in one go. If you do it accidentally, you'll need to click Edit then Undelete to bring it back. The Small Stage Layout option offers you a different interface that gives you more space for the Scripts area. This can be useful when you're working on a complex project, and you can always click it again to go back to the normal layout. The Turbo mode option runs your project at a faster speed. You can switch it on and off by clicking it again.

3 **Tips:** Opens up a Tips window, with a couple of built-in step-by-step project walkthroughs, and advice on the different types of block and what they all do.

4 **About:** Takes you to the About Scratch webpage, which tells you more about Scratch and how to use it.

5 **Duplicate:** Click on this and your mouse pointer becomes a rubber stamp. Clicking on a Sprite on the Stage will create an exact duplicate, complete with costumes and scripts. Clicking on a block of script in the Script area will make another copy of the block. You can also duplicate sprites and blocks by right-clicking on them and then left-clicking Duplicate in the menu.

6 **Delete:** Click on this and your pointer becomes a pair of scissors. Clicking on a Sprite will now delete it, and the same goes for a block of script in the Script area. You can also delete sprites and blocks by right-clicking on them, then left-clicking Delete in the menu.

7 **Grow:** Click on this button, then click on a sprite in the Stage window, and it will grow bigger.

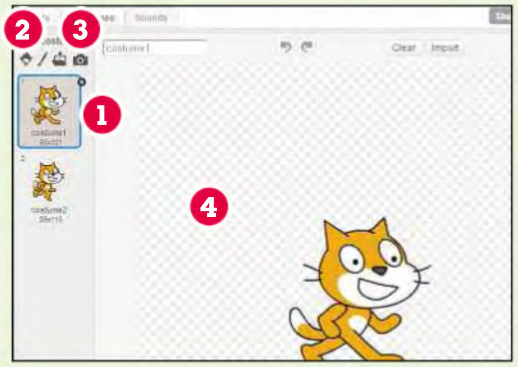
8 **Shrink:** Click on this button, then click on a sprite in the Stage window, and it will get smaller.

9 **Block Help:** Click here, then click on a block in the Blocks Palette or Script area. A tips window will appear, telling you what it is and how it works – a helpful reminder.

they're joined they stay joined unless you click and drag a block down away from the block above. When you do that, any blocks joined to it at the bottom will move away at the same time. This is important when it comes to deleting or replacing an individual block. If you don't drag it away from the blocks above, then drag away the blocks below, you'll delete every block in the script.

You can switch between the different types of block by clicking on the categories – Motion, Looks, Sound and the rest – at the top of the Blocks Palette. Click on Control and you'll notice that some blocks have a kind of C or E shape. These blocks are designed to work closely with other blocks and change what they do, and will fit in above and below existing blocks. Just drag them into the Scripts area, but watch the white highlights to see where they'll slot in. And when you add blocks to a C- or E-shaped control block, watch the highlights to check that they're going to fit in the right place. ●

THE COSTUMES WINDOW



Clicking on the Costumes tab opens the Costumes window. Costumes allow you to clothe your sprites with different graphics, so that you can have different frames, to make your sprite look like its walking or running, or different expressions, to make it look happy, sad, angry or frightened. Existing Costumes for the current sprite are listed on the left 1, or you can add your own Costumes from the library 2, from a camera 3, or by painting your own in the Costume editor 4. You can also use the Costume editor to change the existing Costumes (we'll talk about this in more detail later on).

My first Scratch program

It's time to write your very first Scratch program, and make the Scratch cat magically appear using some simple but fun effects

WHAT YOU'LL LEARN

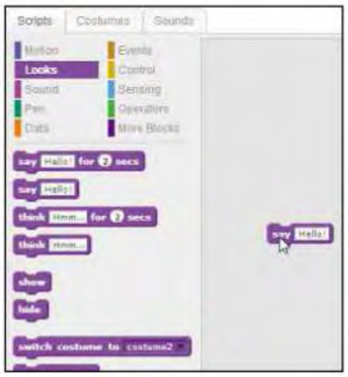
- How to build a simple Scratch script
- How to bring in backdrops
- About simple visual effects
- How to use loops in a Scratch program

TOP TIP

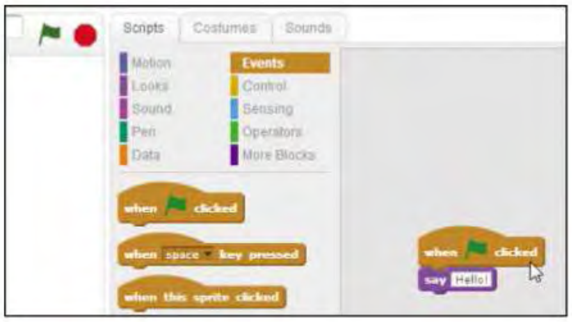
To save a Scratch project, type a name into the textbox above the Stage area, then click on File to open the File menu and click Save Now.

For our first program, we're going to keep things simple. We'll use the Cat1 sprite that appears with every new Scratch project, and we'll do nothing more complex than add a background and drag a few blocks into place.

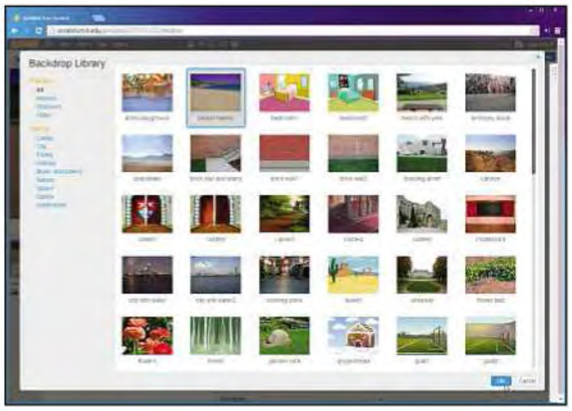
STEP 1 Our leading actor today is the ever-friendly Scratch cat. He wants to say 'Hello!' Click on the Looks category in the Blocks Palette, then drag and drop the Say Hello! Block into the Scripts area. Click on the block, and the cat will say 'Hello' from the Stage.



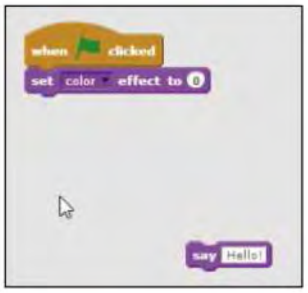
STEP 2 Now go to the Events block and drag the When Clicked block with the picture of the green flag so that it sits on top of the 'say Hello' block. Press the red Stop button above the Stage, then the green Go flag. You've just written your first program, and clicking the green flag starts it running.



STEP 3 You've got a friendly cat, but not the most interesting program. Let's add a backdrop to give our feline friend some scenery. Go the Backdrop area to the left of the Sprites area, and click on the 'Choose new backdrop from library' option. Click on the beach Malibu picture, and click OK. The new background is a start, but let's make things a bit more exciting.

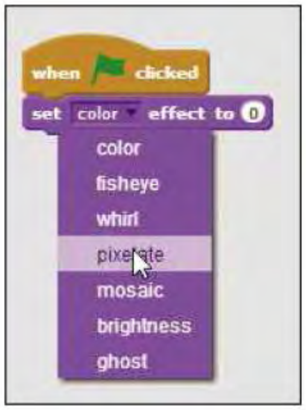


STEP 4 How about a cat that magically appears on the beach? Click on the Scripts tab, then drag the 'say Hello!' block away from the green flag block, and leave it somewhere at the



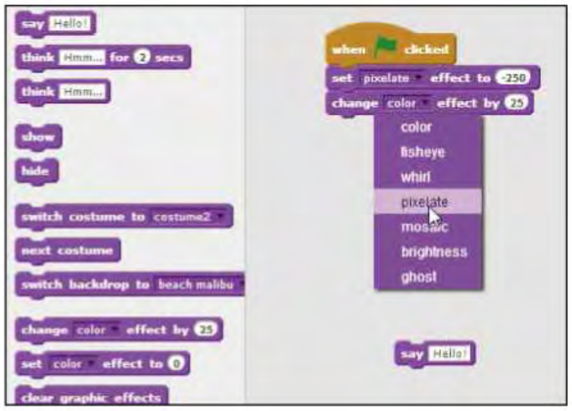
bottom of the area. Now click on the Looks category in the Blocks Palette, and drag the block that says 'set color effect to 0' into place beneath the green flag block.

STEP 5 Where a block has an arrow on it, it means there's more than one option. Click on the arrow, and select Pixelate from the list of effects. Now we need to change the strength of the effect. Click on the white area with the number in it, and type in -250.



```
when [flag] clicked
set color ▾ effect to 0
```

STEP 6 Now for the clever bit. Drag the 'change color effect by 25' block from the Blocks Palette and place it underneath your existing blocks. Change the effect from 'color' to 'pixelate'. This will start transforming your cat from a blocky mess into his old feline form. A sound effect adds to the fun, and makes the change take a little longer. Click on the Sound category and drag the 'play drum 1 for 0.25 beats' block into place.



```
when [flag] clicked
set pixelate ▾ effect to -250
change color ▾ effect by 25

say Hello!
```

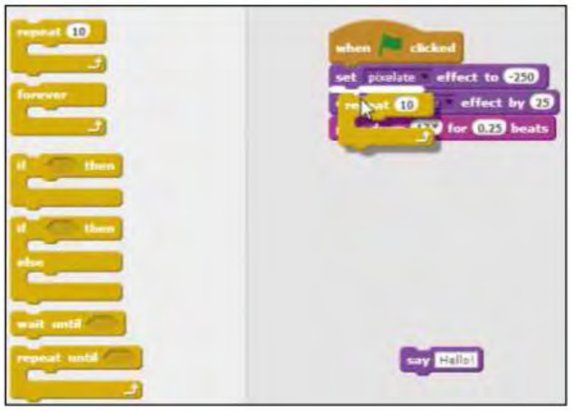
MESS AROUND

Your cat doesn't have to say 'Hello'. Click on the white box next to 'say' on the purple block and you can type in anything you like. Try changing the drum noise in step 7 to different sounds, and try changing the duration (the number before the beats) to different values, like 0.1 or 0.5. See how the project works with different effects, like whirl or ghost. Just remember to alter the setting in both the Set and Change purple blocks, and change the strength of the effect to see what that does. Make the cat appear your way!

LOOPS

When you use the Repeat block, you're creating a loop. Loops are one of the most common structures you'll find in a program. They tell the computer to keep on doing something until a certain condition is reached – in this case, until the loop has run ten times.

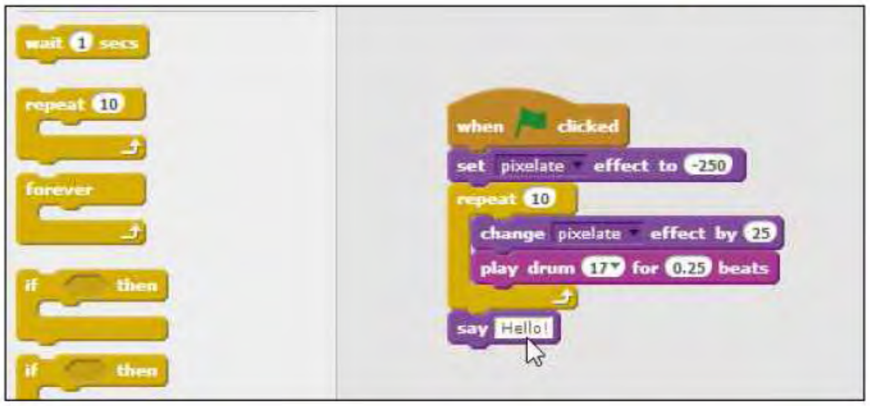
Using loops is a good way to keep your program working at a repetitive task without having to keep adding the same instructions yourself. In more complex programs, they play a large part in controlling how the program and the different parts of it behave.



```
when [flag] clicked
set pixelate ▾ effect to -250
repeat 10
change pixelate ▾ effect by 25
play drum 17 ▾ for 0.25 beats

say Hello!
```

STEP 7 Click the arrow next to drum 1 and select drum 17, the Vibraslap. Now, we could just keep adding the last two blocks, doing so ten times over to bring the cat back to normal (as that's how many steps it would take to turn the Pixelate effect from -250 to 0). Luckily, there's a quicker way. Click on the Control category in the Blocks Palette, then drag the 'repeat 10' block to the Scripts area and drop it in below the first 'set pixelate…' block.

STEP 8 Now grab the 'change Pixelate effect…' block and drop it inside the C of the 'repeat' block. See how the 'Play drum…' block comes with it? Drop it in place, then drag the 'say Hello!' block up again so that it connects to the rest of the program. You're program is done! Press the green flag above the Stage to see it in action. ●



```
when [flag] clicked
set pixelate ▾ effect to -250
repeat 10
    change pixelate ▾ effect by 25
    play drum 17 ▾ for 0.25 beats
say Hello!
```

The animal band

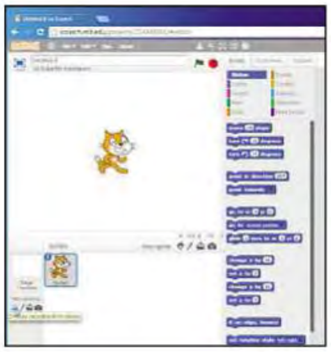
It's time to get interactive as you put an animal band on the stage

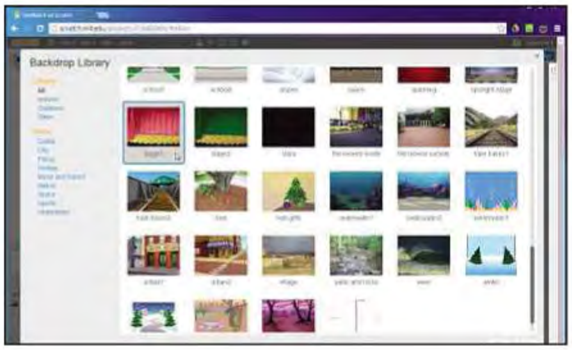
WHAT YOU'LL LEARN

- How to make sprites interact with the mouse and pointer
- How to make music in Scratch
- How to work with values and coordinates
- How to copy scripts from one sprite to another

Our first Scratch program was fun, but it wasn't very clever or interactive. The most useful and interesting programs involve interaction – they respond to an action from the user, whether that's moving the mouse, tapping a key on the keyboard or clicking on a button on the screen. That's why our next project goes a little deeper, with a musical bunch of animals, all primed and ready to perform at the first touch of the pointer.

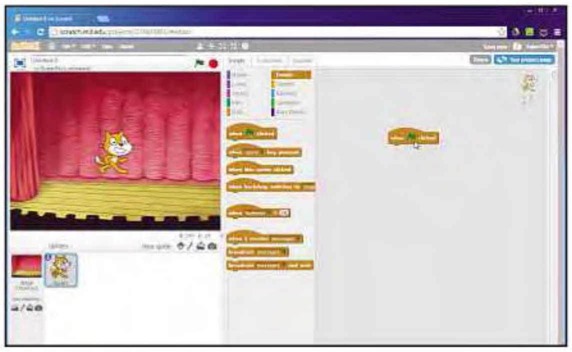
STEP 1 This time we'll start with a backdrop. Go to the Backdrop area next to the Sprites area, and click on the 'Choose backdrop from library' button. Scroll down until you can see the Stage backdrop with the red curtain, then click to select it and press OK. The stage is here, and with Scratch cat on the boards it already has our first performer.



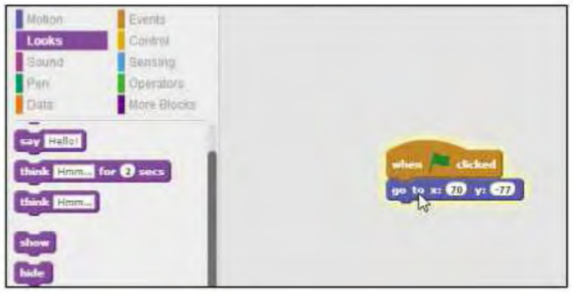


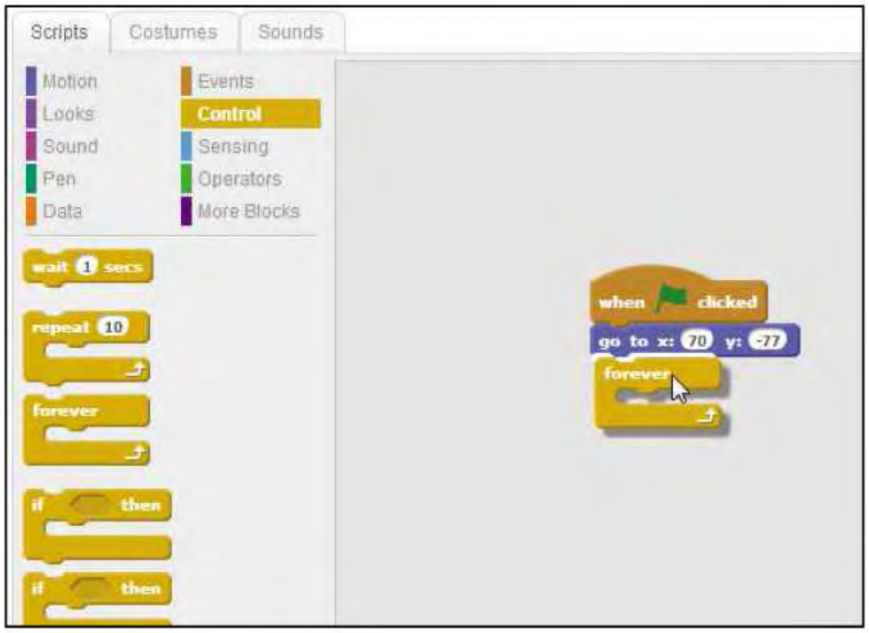
TOP TIP
To open a Scratch project you worked on earlier, click on File and select Go to My Stuff. Remember to save first if you're working on something else, though.

STEP 2 Let's make sure our first musician hits the right spot every time. Click on the Scratch cat sprite in the Sprites area to select it. Now click on the Scripts tab, then the Events category. Drag the 'when green flag clicked' block from the Blocks Palette and drop it onto the Scripts area.

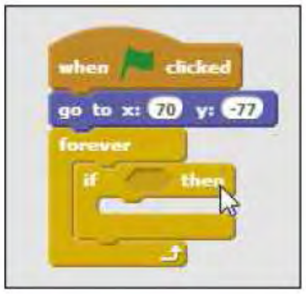


STEP 3 Now click on the Motion category and drag the 'go to x: y:' block onto the Scripts area and stack it underneath the green flag block. By putting numbers – or coordinates – next to the x and y, we can control exactly where the cat appears on stage. Here, click on the number next to x and type 70, then click on the number next to the y and type -77. Click on the block, and our cat moves into the right position.



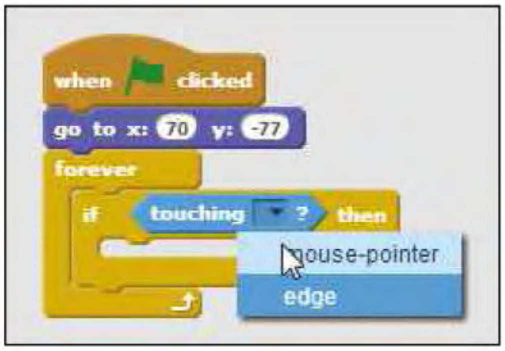


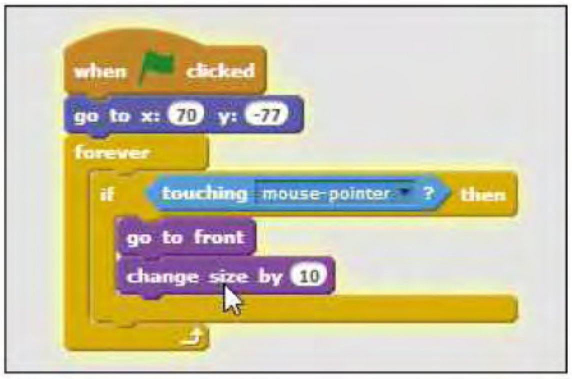
STEP 4 Now for the interactive bit. Click on the Control category, then drag the 'forever' block onto the Scripts area and connect it at the bottom. The 'forever' block keeps running the blocks inside it over and over, no matter what else is happening in the script. We're going to use it to make sure this script keeps looking for our user to interact with the program.



STEP 5 Grab the 'if x then x' block from the Blocks Palette and drop it inside the C of the 'forever' block. The 'if x then x' block tells the script that if a certain 'thing' happens, it's to do whatever instructions we give it inside the C of the block. We'll define what that 'thing' is in the next step.

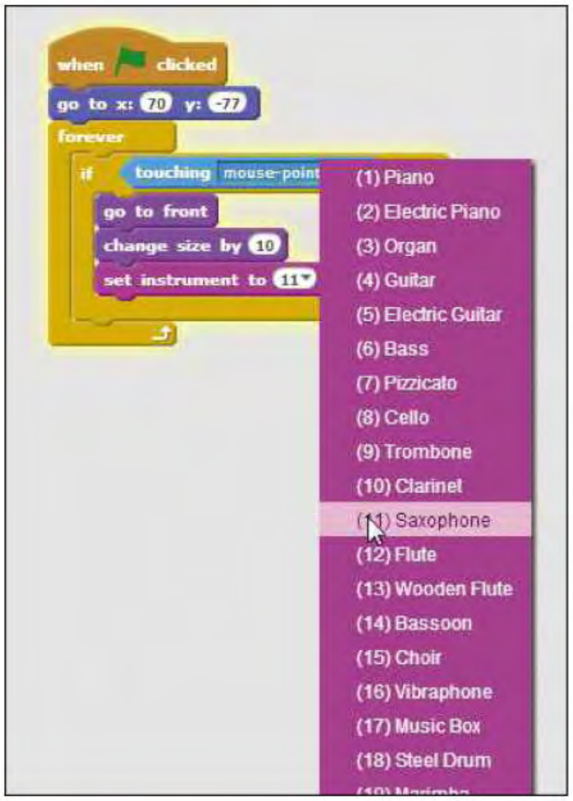
STEP 6 Click the Sensing category, then drag the 'touching' block and drop it onto the little slot between the 'if' and 'then' of the 'if then' block. You have to be careful here, so watch for the highlight to make sure it's going in the right place. Click the arrow next to the question mark, and select 'mouse-pointer' from the dropdown menu.

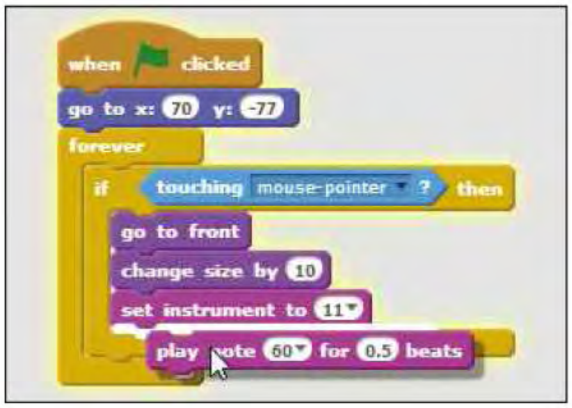




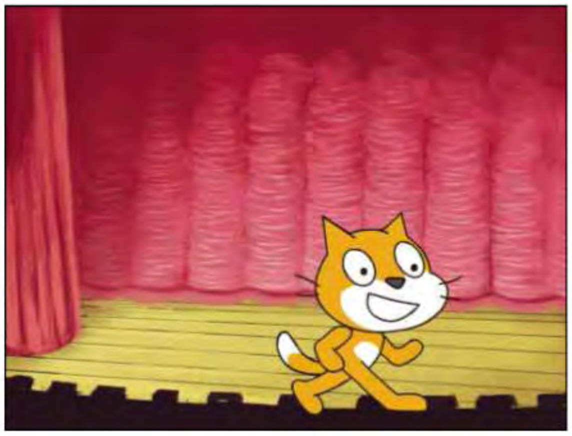
STEP 7 Our script is now looking to see whether the mouse pointer touches the sprite, so we need to tell it what to do when that happens. First, a simple visual effect. Click on the Looks category, and drag the 'go to front' block, then the 'change size by 10' block into the C shape of the 'if then' block.

STEP 8 It's time to get our star making music. Click on the Sound category, drag the 'set instrument to x' to block onto the Scripts area, and drop it right beneath the 'change size by x' block. Again, watch the highlight to make sure it's going in the right place. Click on the arrow next to the 1 and set the instrument to 11, the Saxophone.



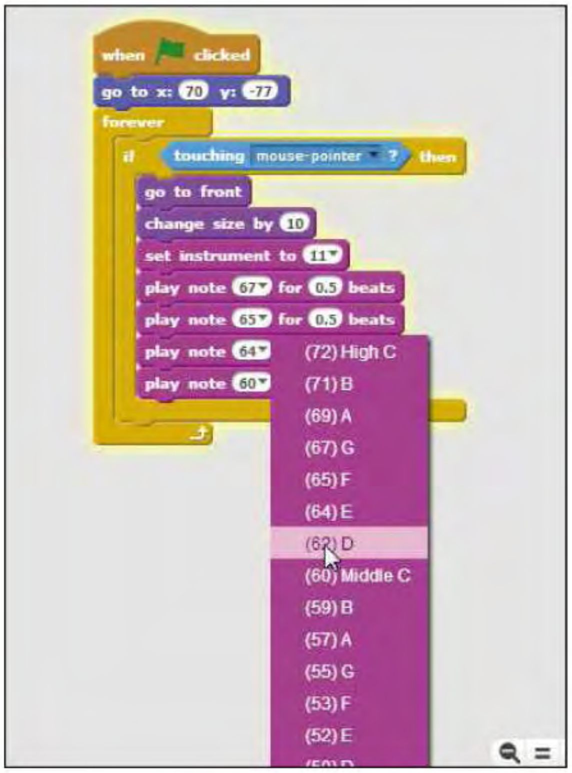


```
when [flag] clicked
go to x: 70 y: -77
forever
    if touching mouse-pointer ? then
        go to front
        change size by 10
        set instrument to 11
        play note 60 for 0.5 beats
```

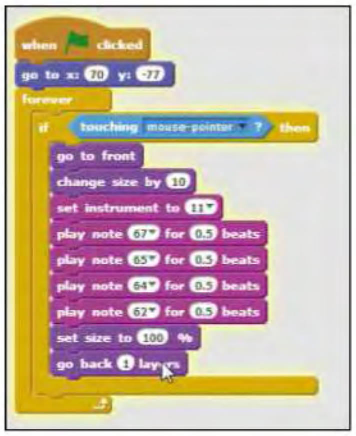


STEP 9 That tells the script what noise to make when we touch the sprite with our pointer, but what about which notes to play? Drag the 'play note x for x beats' block into the Scripts area and stack it underneath the last block. Click the arrow next to the first number, and set it to 67, or the musical note G. Leave the number next to beats alone for now.

STEP 10 Repeat that last step three more times, stacking each 'play note x for x beats' block below the last. Now go through and change the values for each note to 65, 64 and 62. This sets our cat saxophonist to play four notes of a simple downward scale.



```
when [flag] clicked
go to x: 70 y: -77
forever
    if touching mouse-pointer ? then
        go to front
        change size by 10
        set instrument to 11
        play note 67 for 0.5 beats
        play note 65 for 0.5 beats
        play note 64
        play note 60
```



```
when [flag] clicked
go to x: 70 y: -77
forever
    if touching mouse-pointer ? then
        go to front
        change size by 10
        set instrument to 11
        play note 67 for 0.5 beats
        play note 65 for 0.5 beats
        play note 64 for 0.5 beats
        play note 62 for 0.5 beats
        set size to 100 %
        go back 1 layers
```

STEP 11 If you try running the script now, by clicking on it or clicking the green flag, you'll notice that our cat keeps getting bigger. Let's stop that once he's played his piece. Click on the Looks category, then drag the 'set size to x%' block and the 'go back x layers' block in beneath the 'notes' blocks, but still within the C shape of the 'if then' block. Set the number in the first block to 100 and keep the number in the second to 1.

STEP 12 That's one band member sorted, but we need to add some others. Go to the Sprites area and click the 'Choose sprite from library' button next to New Sprite. Select the Dog2

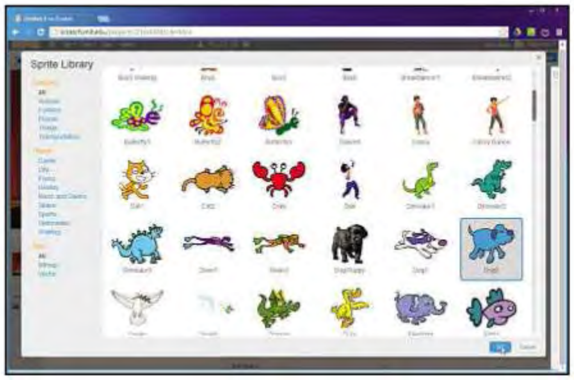
COORDINATES

When Sprites need to go to a certain place or move around the Stage, we control where they go and what they do using coordinates. There are two coordinates for each position. The first, the x coordinate, tells the sprite where to go horizontally, or left and right along the Stage. The second, the y coordinate, tells the sprite where to go vertically, or up and down inside the Stage. x: 0, y: 0 is the centre of the Stage, so you use minus numbers when you want a Sprite to go left or down of centre, and positive numbers to place them up or right of centre. The same goes when you want them to move, which we'll look at later on.

TOP TIP

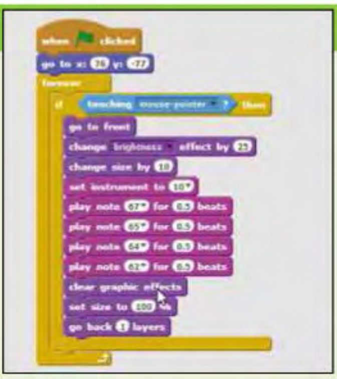
ASSIGNING VALUES

In Scratch, many blocks work with values – a number you type in or select from a menu that controls how the block behaves. Many will have a ready-set value, like 10 or 50, but for the purposes of the book we'll refer to these values as x or y. Altering the values can be a good way to experiment with a Scratch project. If you don't like what happens, just put the value back to what it was before.



MESS AROUND

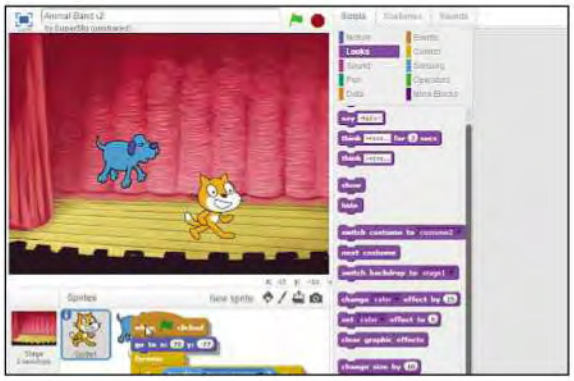
You can add any sprite you want to your band, and alter the instrument they play and the notes they play on it just by altering the values – the numbers – in the 'set instrument' block and 'play note' blocks. If you want, you can also add visual effects to your sprites so that they change colour or brightness when you touch them with the pointer. Just drag a 'change x effect by x' block into the stack beneath the 'go to front' block, then drag a 'clear graphic effects' block underneath the 'play note x for x beats' blocks.



```
when [flag] clicked
go to x: 76 y: -77
forever
  if touching mouse-pointer ? then
    go to front
    change brightness effect by 25
    change size by 10
    set instrument to 10
    play note 67 for 0.5 beats
    play note 65 for 0.5 beats
    play note 64 for 0.5 beats
    play note 62 for 0.5 beats
    clear graphic effects
    set size to 100 %
    go back 1 layers
```

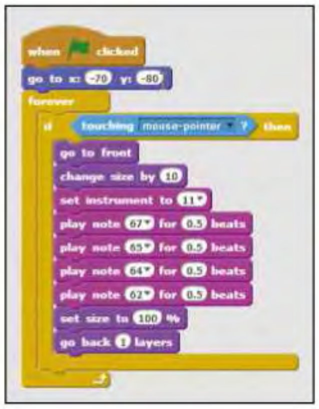
sprite, then click OK. Hey presto, a dog appears on the stage.

STEP 13 At this point, you're probably thinking "Oh no, I've got to make a whole new script for the dog!" Actually, you don't. Click on the Cat sprite, then right-click on the block of script and select Duplicate. You'll now have the whole block attached to your pointer. Drag it down to the



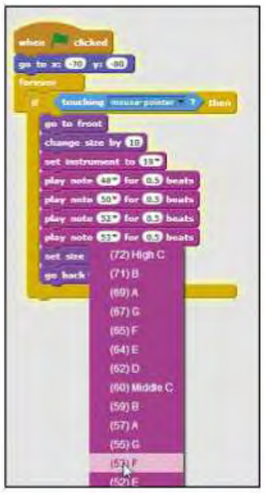
Sprites area, move the pointer over the dog and click. Now go back to the Scripts area, right-click on the extra block of script and select Delete.

STEP 14 The script you had attached to your cat is now attached to your dog, but we don't want him to appear in exactly the same place or play exactly the same notes through the same instrument. That means we need to adjust the values – the numbers – in the script. Click on the dog sprite, then go to the Scripts area and change the x: and y: values in the 'go to' block to -70 and -80. Just click on the numbers and type in new numbers.



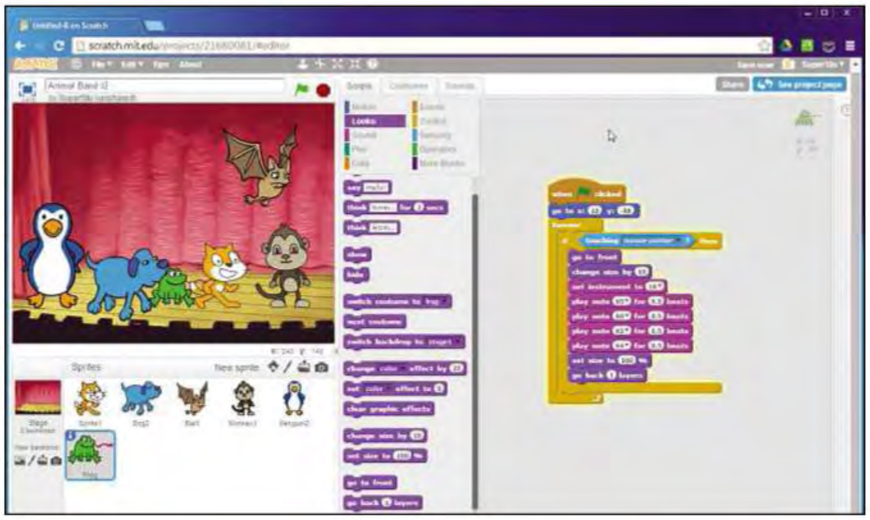
```
when [flag] clicked
go to x: -70 y: -80
forever
  if touching mouse-pointer ? then
    go to front
    change size by 10
    set instrument to 11
    play note 67 for 0.5 beats
    play note 65 for 0.5 beats
    play note 64 for 0.5 beats
    play note 62 for 0.5 beats
    set size to 100 %
    go back 1 layers
```

STEP 15 Now try setting the instrument in the set instrument block to 19, the marimba, then change the four notes to 48, 50, 52 and 53. Our canine chum now plays a cheerful upward scale all of his own.



```
when [flag] clicked
go to x: -70 y: -80
forever
  if touching mouse-pointer ? then
    go to front
    change size by 10
    set instrument to 19
    play note 48 for 0.5 beats
    play note 50 for 0.5 beats
    play note 52 for 0.5 beats
    play note 53 for 0.5 beats
    set size
    go back
```

STEP 16 Now, keep adding sprites until your band is assembled and ready to play. You can keep duplicating and dragging the same script from one to the next, remembering each time to add a new starter position, a new instrument and a new set of notes to play. Within minutes, your animal band will be up on stage and ready to perform. ●



Animate a Scratch cartoon

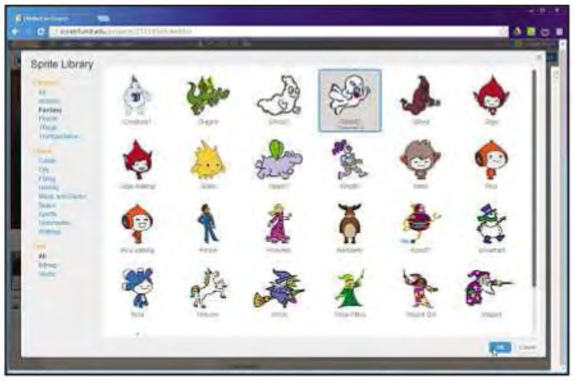
Now you know some of the basics, you're ready for something more ambitious. How about a short scratch-powered cartoon?

WHAT YOU'LL LEARN

- How to animate a character
- How to make one sprite trigger actions in another
- How to duplicate stacks to save time and effort

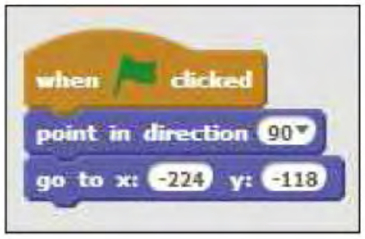
Scratch is perfect for creating animations, as it has a library of characters, backdrops and sounds built in, and all the script blocks you need to make use of them. To see how it can be done, we're going to create a simple cartoon, but you can use the techniques you learn here to make something longer and even more exciting.

STEP 1 First, we need to get our sprites and backdrops ready. Click on the 'Choose backdrop from library' button, select the Castle4 backdrop and click OK. Next, right-click on the cat sprite in the Sprites area and select Delete. Now, using the 'Choose sprite from library'



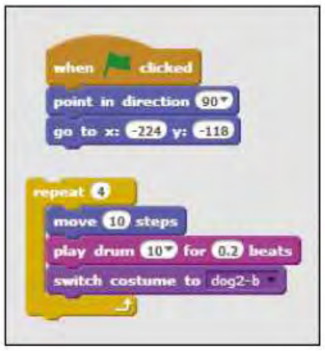
button, go and grab the Dog2 sprite, then the Ghost2 sprite. You can find them faster by clicking on the Animals and Fantasy categories on the left-hand side of the Sprite Library window.

STEP 2 Click on the Dog2 sprite in the Sprites area, and let's get scripting. Click on the Events category and drag the 'when green flag clicked' block into the Scripts area, then click on the Motion category and drag in the 'point in direction x' block and the 'go to x:x y:y' block. Keep the first block as it is, but in the second set x to -224 and y to -118. This makes sure our dog is facing the right direction – it becomes important later – and puts him in exactly the right spot.



```
when [flag] clicked
point in direction 90
go to x: -224 y: -118
```

STEP 3 Here's how we animate him. Click on the Control category, then drag a 'repeat x' block into the Scripts area. Don't attach it to our stack right away. Now go to the Motion category, and drag the 'move x steps' block into the C shape of the 'repeat' block. To control the speed of the animation, we'll use a sound. Go to the Sounds category, and pull in the 'play drum x for y beats' block. Set the drum sound to 10 and the beats to 0.2. Finally, go to Looks, and drag in the 'switch costume to x' block. Set the costume to dog2-b.



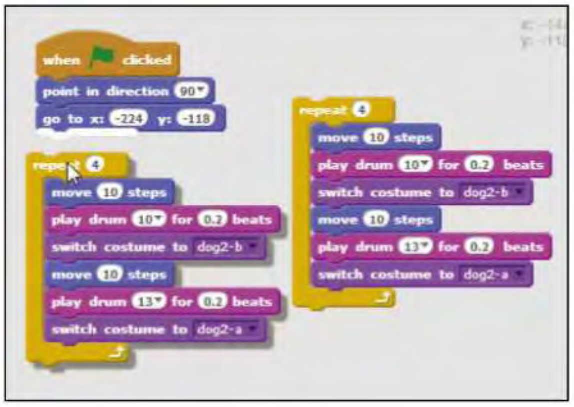
```
when [flag] clicked
point in direction 90
go to x: -224 y: -118

repeat 4
    move 10 steps
    play drum 10 for 0.2 beats
    switch costume to dog2-b
```

STEP 4 That sorts out one step of this two-step walk. For the next step, simply repeat the sequence of 'move x steps', 'play drum' and 'switch costume to x' blocks, then make some changes. Set the drum sound to 13 and the beats to 0.2,

TOP TIP
DUPLICATING
One of the great things about Scratch is that you can build stacks of blocks for the most common tasks going on in your program, then keep duplicating and reusing them. It's a lot easier than doing it all block by block.

ANIMATION

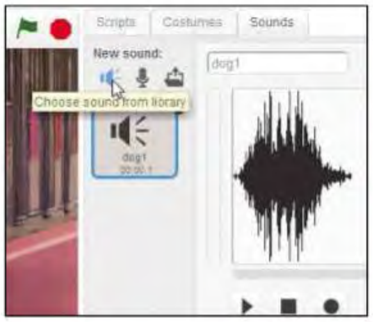
To animate a character, you need to use a sprite with more than one costume, where each costume can represent a frame of animation. With our dog, for example, each costume has the legs in different positions, which means that swapping from one costume to another will make it look like the dog is walking. The more costumes you have for a walking animation, the smoother and more realistic the animation will look. The old video games that used sprites sometimes used hundreds of frames just for one character's walks, runs and jumps!

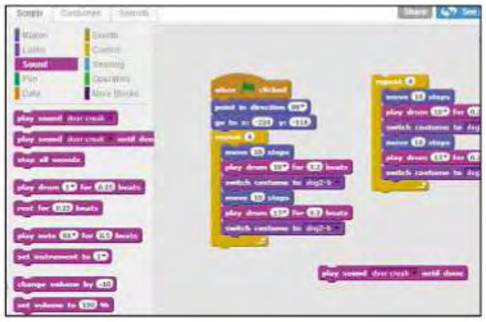


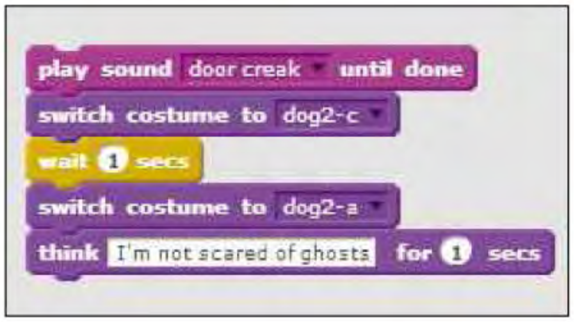
```
when [flag] clicked
point in direction 90
go to x: -224 y: -118

repeat 4
    move 10 steps
    play drum 10 for 0.2 beats
    switch costume to dog2-b
    move 10 steps
    play drum 13 for 0.2 beats
    switch costume to dog2-a

repeat 4
    move 10 steps
    play drum 10 for 0.2 beats
    switch costume to dog2-b
    move 10 steps
    play drum 13 for 0.2 beats
    switch costume to dog2-a
```

and switch the costume back to dog2-a. Finally, set the value of the 'repeat' block – the number of times it will repeat – to 4. That's our doggy animation block done, but don't attach it to the main script block, as we'll need it several times. Instead, right-click on it and duplicate it, then attach that version.

STEP 5 It's time to give our canine star the creeps. First, we'll add a noise. Click the Sounds tab, then click the 'Choose sound from library' button – it's the first one on the left under New Sound in the top-left corner. Click on the Effects category, select the 'door creak' sound, and press OK. Now, click the Scripts tab, and drag the Play sound x until done block into the Scripts area and start a new stack with it. Make sure the sound is set to the door creak.

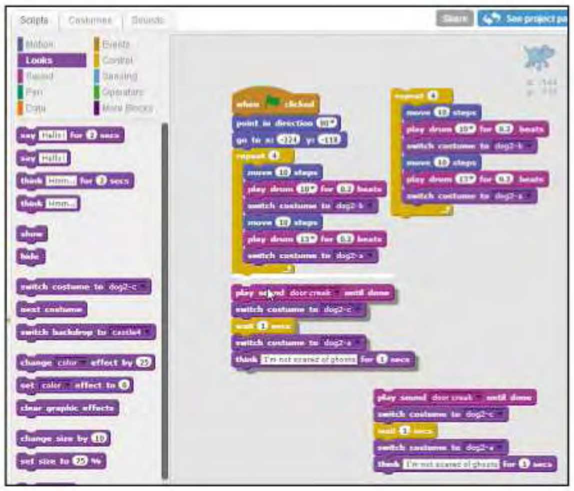






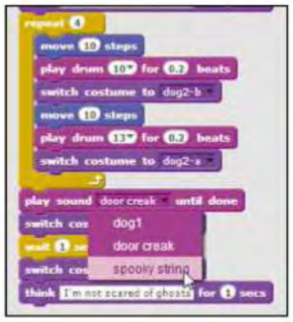
```
play sound door creak until done
switch costume to dog2-c
wait 1 secs
switch costume to dog2-a
think I'm not scared of ghosts for 1 secs
```

STEP 6 Click on the Looks category, drag in the 'Switch costume to x' block, and set the costume to 'dog2-c'. Click on the Control category, and drag in the 'wait x secs' block. Leave it set to 1. Pull in another 'switch costume to x' block, and set it to switch costume to 'dog2-a'. Finally, drag in a 'think x for x secs' block from Looks, and change the "Hmmm…." To "I'm not scared of ghosts…." and the time to 1 second.



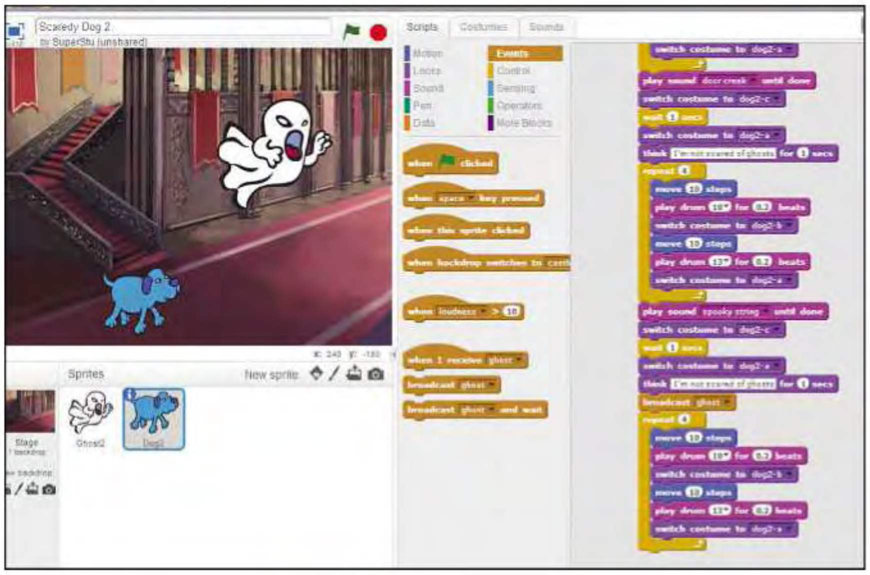
STEP 7 Again, we'll want to use this block more than once, so duplicate it, then drag the duplicate into place at the bottom of your main script. Give it a quick runthrough by clicking the script or pressing the green flag button at the top of the Stage area.

STEP 8 Now for the clever bit. We can save ourselves some work by reusing the two stacks we've made. Duplicate the animation stack again and drag it to the bottom of our main stack. Then duplicate the second stack – the one

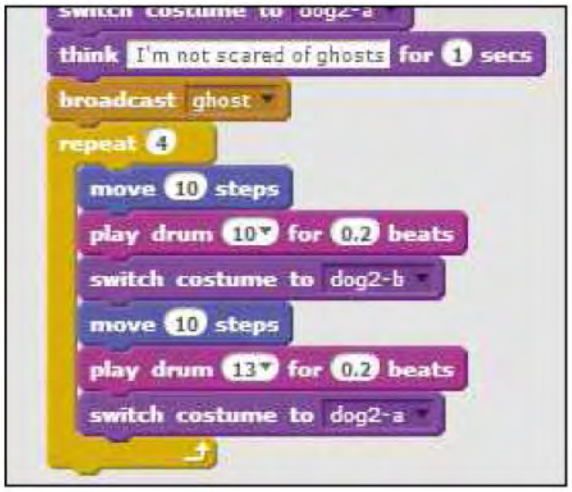


```
repeat 4
    move 10 steps
    play drum 10 for 0.2 beats
    switch costume to dog2-b
    move 10 steps
    play drum 13 for 0.2 beats
    switch costume to dog2-a
play sound door creak until done
switch cos
wait 1 se
switch cos
think I'm not scared of ghosts for 1 secs
```

that ends with ‘I’m not scared of ghosts…’ – and drag that below it. To add a little variety, click on the Sounds tab, add a second sound, Spooky String from the Instruments category, just like we did in step 5. Change the sound in the second ‘I’m not scared…’ block to the Spooky String sound.



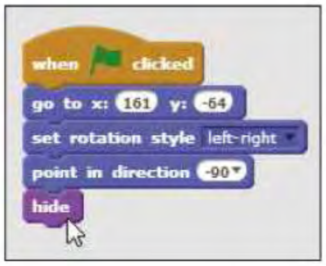
```
    switch costume to dog2-a
play sound door creak until done
switch costume to dog2-c
wait 1 secs
switch costume to dog2-a
think I'm not scared of ghosts for 1 secs
repeat 4
    move 10 steps
    play drum 10 for 0.2 beats
    switch costume to dog2-b
    move 10 steps
    play drum 13 for 0.2 beats
    switch costume to dog2-a
play sound spooky string until done
switch costume to dog2-c
wait 1 secs
switch costume to dog2-a
think I'm not scared of ghosts for 1 secs
broadcast ghost
repeat 4
    move 10 steps
    play drum 10 for 0.2 beats
    switch costume to dog2-b
    move 10 steps
    play drum 13 for 0.2 beats
    switch costume to dog2-a
```



```
switch costume to dog2-a
think I'm not scared of ghosts for 1 secs
broadcast ghost
repeat 4
    move 10 steps
    play drum 10 for 0.2 beats
    switch costume to dog2-b
    move 10 steps
    play drum 13 for 0.2 beats
    switch costume to dog2-a
```

STEP 9 It’s time to bring our second character into play. In Scratch, you can use the Broadcast block to send a message to another sprite, so that they can do something in return. Go to the Events category, and drag the ‘broadcast’ block to the bottom of your stack. Click the arrow next to the box with ‘message1’ in it, and select ‘new message…’ Type in ‘ghost’ as the message name and click OK.

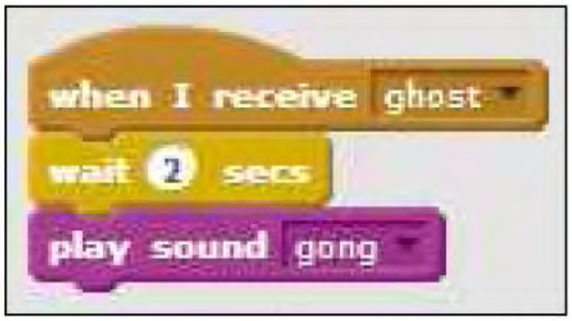
STEP 10 We want our dog to keep on walking, so duplicate and drag the animation stack to the bottom of the main stack. Now click on the ghost in the Sprites area.



```
when clicked
go to x: 161 y: -64
set rotation style left-right
point in direction -90
hide
```

First, we need to set up a simple script to get him in the right position. Drag in the ‘when green flag clicked’ block, then the ‘go to x: x y: y’ block. Set x to 161 and y to -64. Drag in the ‘set rotation style’ block and set it to ‘left-right’, then the ‘point in direction x’ block. Set the direction to -90. Finally, go to the Looks category and drag in a ‘hide’ block. We don’t want our ghost to be visible at first.

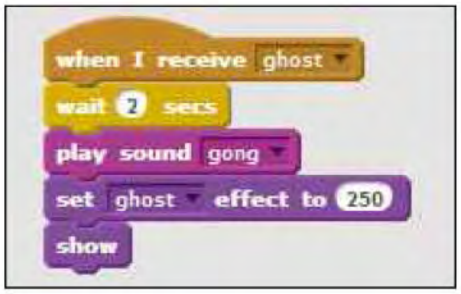
STEP 11 Now we need to decide what the ghost does when he gets the message. Go to the Events category, and drag in the ‘when I receive’ block to start a new stack. It should already be set to ‘ghost’. For the sake of suspense, go to the Control category and add a ‘wait x secs’ block, setting the value to 2. When our ghost appears, we want a shock, so add a new sound, just as we did before. This time we want the gong, which you can find in the Percussion category. Once you’ve added it, click on Scripts, then the Sound category, then drag in the ‘play sound’ block and set it to ‘gong’.



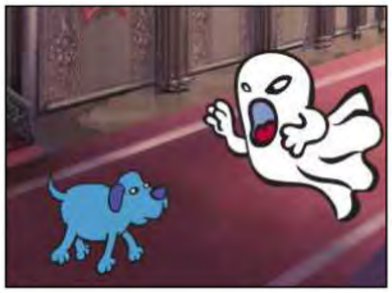
```
when I receive ghost
wait 2 secs
play sound gong
```

CONTROL FLOW

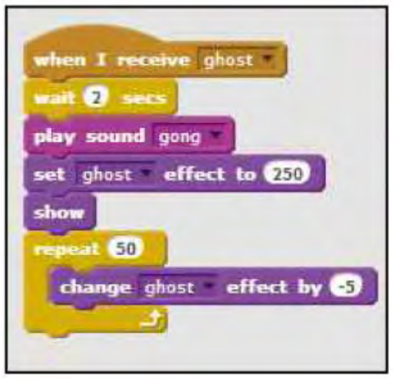
In programming, the control flow is the order in which the individual statements or instructions in a program will run, and control flow statements are instructions which affect that order. We’ve already used the loop, which tells the program to run the same instructions again and again, either forever or a certain number of times. The If x then x block is another control flow statement, as it tells the program to do one thing if a certain thing is happening, or not. The Broadcast and When I receive blocks used here are more examples, as they tell one sprite to send a message that triggers an action from the other sprite. The When I receive block then defines what that action is.



```
when I receive ghost
wait 2 secs
play sound gong
set ghost effect to 250
show
```

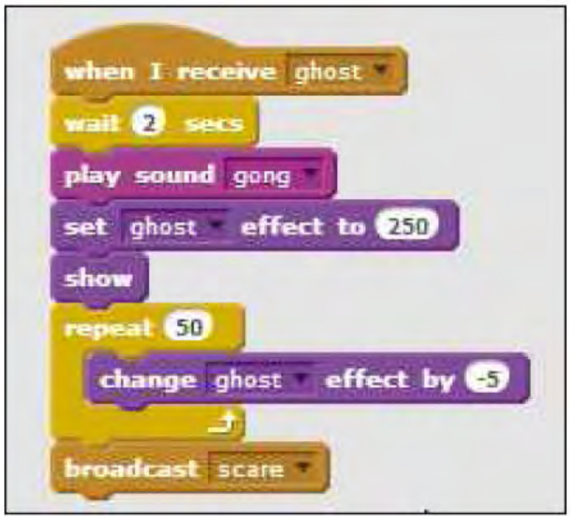


STEP 12 A simple script makes our ghost appear. Go to the Looks category, and drag in the 'set x effect to x' block. Set the effect to 'ghost' and type in 250 as the value. Now drag in the 'show' block underneath. The ghost effect tells our scary spirit to remain invisible. Next, go to Control and drag in a 'repeat' loop. Return to the Looks category, and drag the 'change x effect by x' block. Set the loop to repeat 50 times, and set the effect to 'ghost' and the value to -5. This will make our ghost appear in spooky style.



```
when I receive ghost
wait 2 secs
play sound gong
set ghost effect to 250
show
repeat 50
    change ghost effect by -5
```

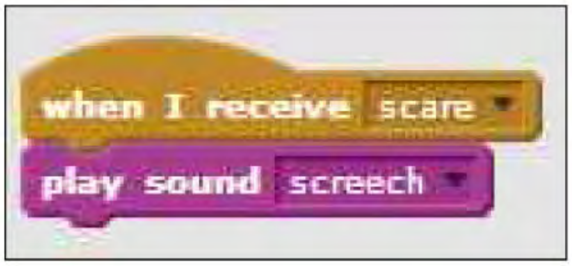
STEP 13 That should scare our canine hero, so how will he react? Again, we can use the 'broadcast' block to find out. Go to the events category and drag a 'broadcast' block



```
when I receive ghost
wait 2 secs
play sound gong
set ghost effect to 250
show
repeat 50
    change ghost effect by -5
broadcast scare
```

to the bottom of the stack. As in step 9, select a new message and this time call it 'scare'. Now, click on the dog sprite so we can define how he takes his shock.

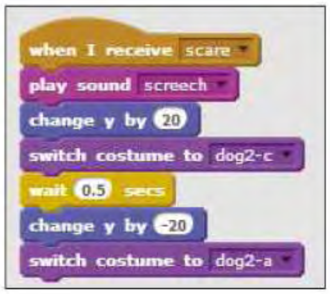
STEP 14 Drag the 'when I receive' block from the Events category into the Scripts area to start a new stack. Set the message to 'scare'. First, our dog will scream, then jump in fear. Add a new sound – screech from the Electronic category – then



```
when I receive scare
play sound screech
```

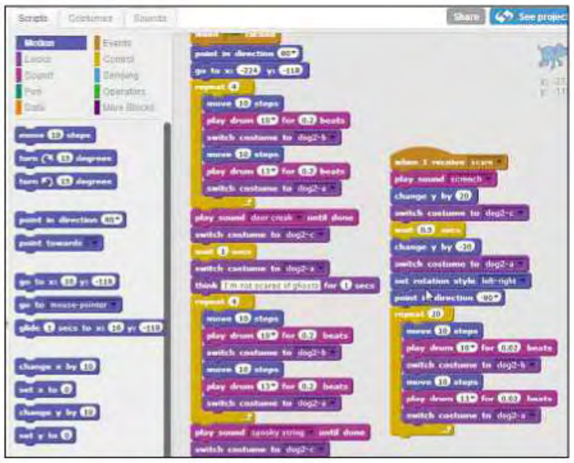
click on the Scripts tab, then the Sound category. Drag in the 'play sound' block and set the sound to 'screech'.

STEP 15 For the jump, go to the Motion category and drag in the 'change y by x' block. Set the value to 20. Next, go to the Looks category, drag in a 'switch costume to x' block from the Looks category, and set the costume to 'dog-2c'. Now go to the Control category and drag in a 'wait' block, and type in 0.5 to set the value. Finally, go back to the Motion category, pull in another 'change y by x' block, and set the value to -20. Go to Looks, drag in a second 'switch costume to x' block, and set the costume back to 'dog2-a'.



```
when I receive scare
play sound screech
change y by 20
switch costume to dog2-c
wait 0.5 secs
change y by -20
switch costume to dog2-a
```

STEP 16 Finally, let's have our dog turn tail and run away. Go to Motion, drag in a 'set rotations style' block and set it to 'left-right'. Next, grab the 'point in direction' block and set it to -90. Now we can reuse the animation block we set up earlier. Drag it into place at the bottom, then set the repeat value to 20, and the beats value for the two 'play drum' blocks to 0.02. This makes our dog run a faster, and sends him speeding home to end our short cartoon. ●



```
point in direction 90
go to x: -274 y: -118
repeat 4
    move 10 steps
    play drum 10 for 0.2 beats
    switch costume to dog2-b
    move 10 steps
    play drum 11 for 0.2 beats
    switch costume to dog2-a
play sound door creak until done
switch costume to dog2-c
wait 1 secs
switch costume to dog2-a
think I'm not scared of ghosts for 1 secs
repeat 4
    move 10 steps
    play drum 10 for 0.2 beats
    switch costume to dog2-b
    move 10 steps
    play drum 11 for 0.2 beats
    switch costume to dog2-a
play sound spooky string until done
switch costume to dog2-c
```

```
when I receive scare
play sound screech
change y by 20
switch costume to dog2-c
wait 0.5 secs
change y by -20
switch costume to dog2-a
set rotation style left-right
point in direction -90
repeat 20
    move 10 steps
    play drum 10 for 0.02 beats
    switch costume to dog2-b
    move 10 steps
    play drum 11 for 0.02 beats
    switch costume to dog2-a
```

Shark vs Food

In the battle of Shark vs Food, only shark can win

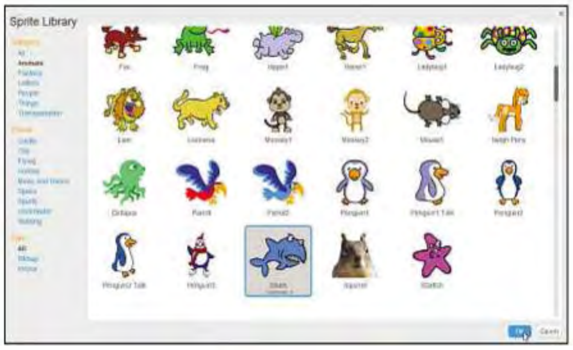
WHAT YOU'LL LEARN

- More on using conditional statements like, if and then
- How to make and use clones of your sprites
- How to use operators and get random numbers into programs
- How to put keyboard commands in a project

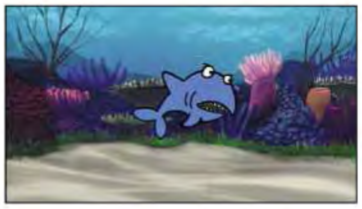
In the last two projects, we've started working with conditional statements. These statements tell our program to proceed in one direction or another, depending on whether a certain condition is met. The 'if, then' block is a great example. Imagine you have a runner waiting for the starting pistol at the beginning of a race. The 'if, then' block might tell the runner if the starting pistol fires, run as fast as you can. Otherwise, hold your position on the starting blocks and wait. Conditional statements work in exactly the same way.

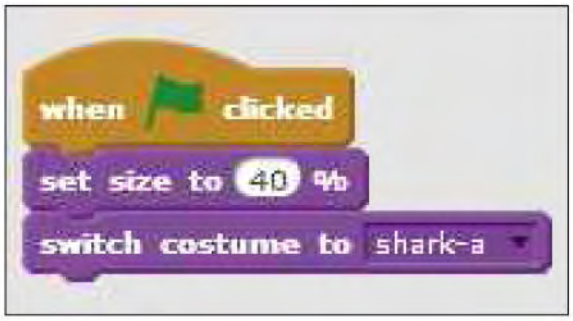
We're going to use another conditional statement in this next project, as well as discover a handy method of making many sprites form one and play with some more motion blocks and sounds. It's a tale of sharks, fish and overeating, which we're calling Shark vs Food.

STEP 1 We need to do some preparation first. Start a new project, then right-click on the Scratch cat in the Sprites area; now select Delete to dump it from the project. Go to the Stage area and click the 'Choose backdrop from library' button. Find the Underwater 2 backdrop, click on it and press



OK. Now it's time to find our hero. Go back to the Sprites area and click on the 'Choose sprite from library' button. Click on the Animals category then select the Shark sprite. Click OK.

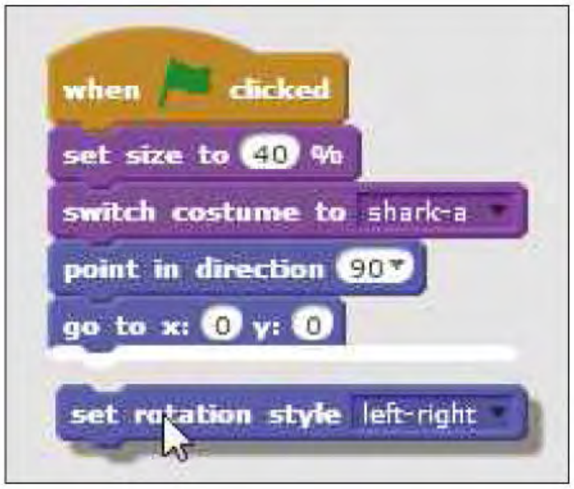




```
when [flag] clicked
set size to (40) %
switch costume to [shark-a ▾]
```

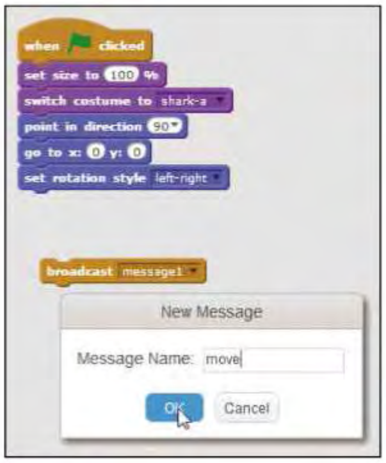
STEP 2 Now to get our hero in position, not to mention make him a more manageable size. Make sure the Scripts tab is active, then drag the old 'when green flag clicked' block into the Scripting area. Now go to the Looks category and pull the 'set size to' and 'switch costume to' blocks into place beneath. Set the size to 40% and make sure the costume is set to shark-a.

STEP 3 Next, hit the Motion category and grab the 'point in direction', 'go to x: x y: y' and 'set rotation style to' blocks. Drag them into position at the bottom of the Script block. Leave the direction set at 90 and the rotation stuck at left – right, but set the x: and y: coordinates to 0 and 0. All this block does is ensure that our shark starts each run of the program in the same place, at the same size, in the same costume and facing the right direction.

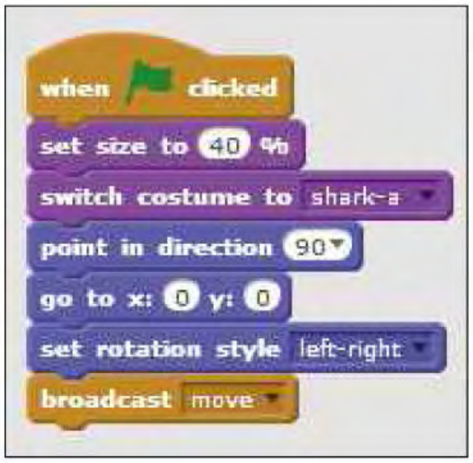


```
when [flag] clicked
set size to (40) %
switch costume to [shark-a ▾]
point in direction (90 ▾)
go to x: (0) y: (0)
set rotation style [left-right ▾]
```

TOP TIP
The Set rotation style block fixes Scratch's tendency to flip sprites over when they hit the edge if you're using the If on edge, bounce block. Keep your sprites standing the right way up!



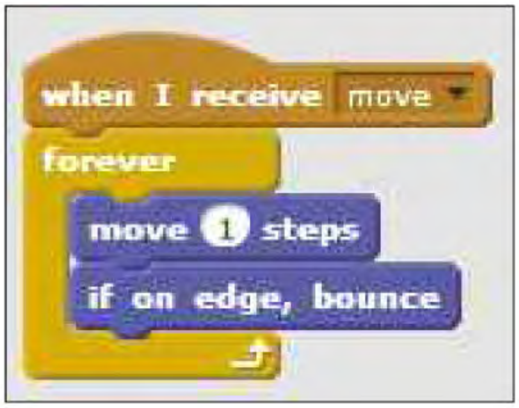
```
when [flag] clicked
set size to 100 %
switch costume to shark-a
point in direction 90
go to x: 0 y: 0
set rotation style left-right

broadcast message1
```



```
when [flag] clicked
set size to 40 %
switch costume to shark-a
point in direction 90
go to x: 0 y: 0
set rotation style left-right
broadcast move
```

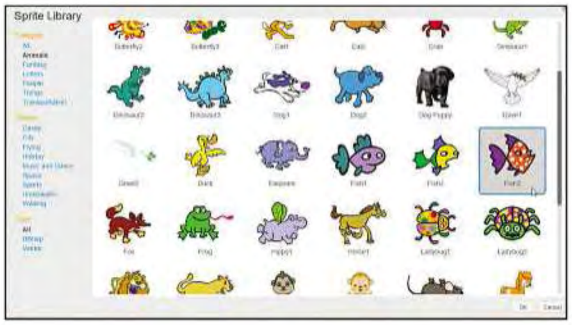
STEP 4 Now we want to pull in a Broadcast message block from the Events category – you'll remember them from the last project. Drag the block into the Scripting area then click on the message and type in 'move'. This is going to call in a new routine we'll continue using to move the shark around the screen.

STEP 5 We can start this new routine by dragging in a 'when I receive' block from Events and switching the message to 'move'. Beneath this, we need a Forever loop, and inside this we place two Motion blocks. The first is a 'move x steps' block, which we set to tell the shark to move forward 1 step. The second is an 'if on edge, bounce' block. You probably know what that one does.

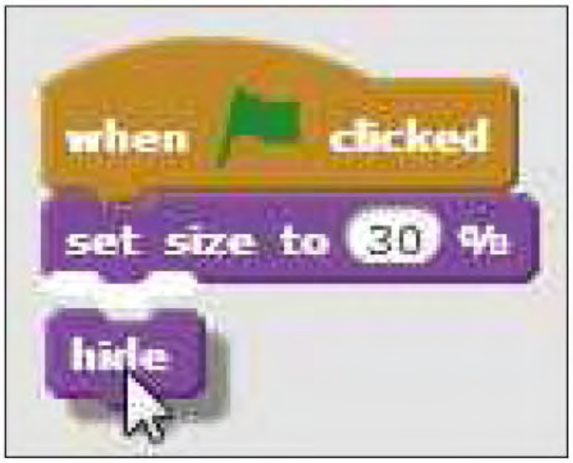


```
when I receive move
forever
    move 1 steps
    if on edge, bounce
```

STEP 6 That's the shark up and running, but what about the food? Go back to the Sprites area and click the 'Choose sprite from library' button. Go to the Animals category and select Fish3, then click OK. Our new fishy foodstuff will appear on the stage.

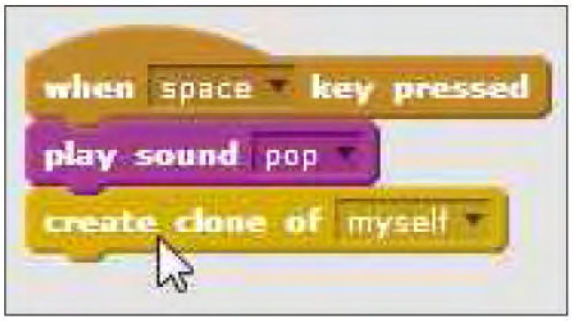


STEP 7 We're going to do something clever with this new sprite. Rather than add lots of fish and script them individually, we're going to use this one to generate clones. We can then define how these clones behave and what will happen if they come into contact with the shark. First, drag a 'when green flag clicked' block into the Scripting area, and add



```
when [flag] clicked
set size to 30 %
hide
```

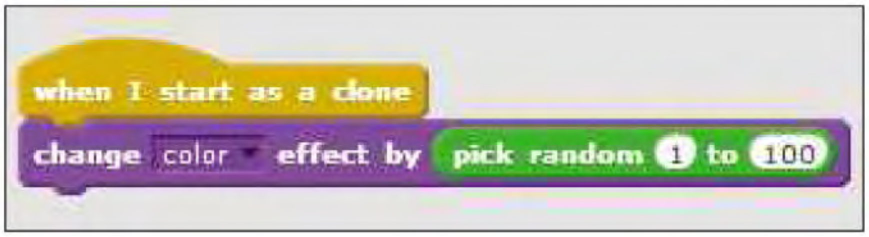
a 'set size to' block from Looks beneath. Set the size to 30%. Now grab the 'hide' block and drag this into place. That keeps our original 'motherfish' safely off the screen.



```
when space key pressed
play sound pop
create clone of myself
```

STEP 8 Now for the cloning bit. Go to Events and pull in the 'When space key pressed' block, leaving it at 'space'. Go to the Sound category and add a cheerful 'play sound' block, with it set to 'pop', then go to Control and add the 'create clone of' block, leaving it set to 'myself'. Now a new fish will spawn every time you tap the spacebar while the program is running.

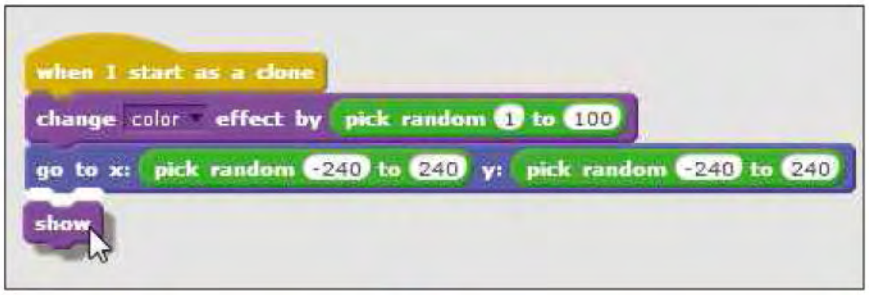
STEP 9 It would be dull if all the fish looked exactly the same and appeared in the same place, but we can fix this as each clone is spawned. Drag in a 'when I start as a clone' block from Control, then go to Looks and add a 'change colour effect by' block beneath it. Normally, we'd type in a number here, but we can make the colour random to



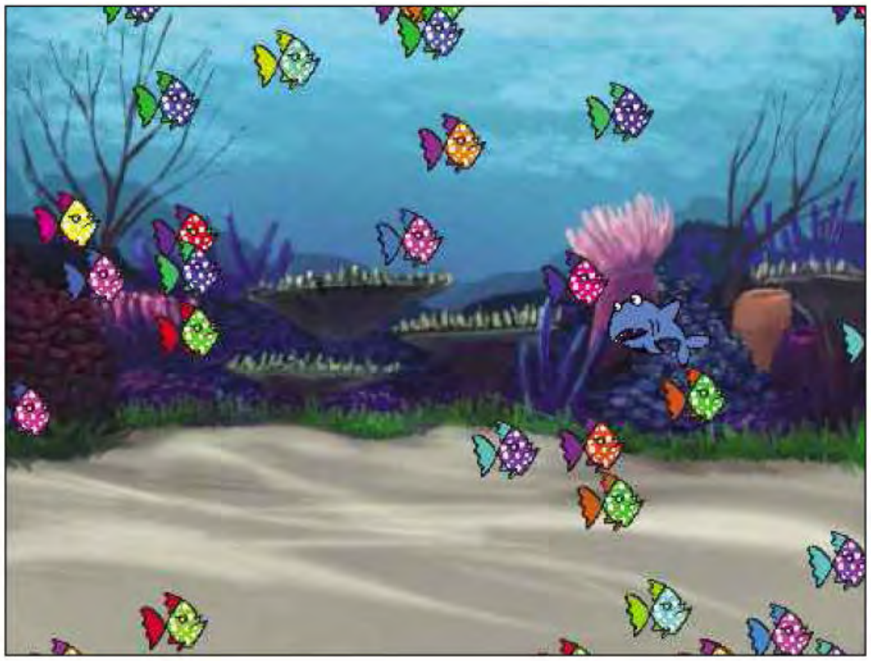
```
when I start as a clone
change color effect by pick random 1 to 100
```

keep our fish varied. Go to Operators and find the 'pick random' block, then drag it in where the number should be. Set the numbers to 1 and 100. Now, every time a fish spawns, its colour will change by a random number between 1 and 100.

STEP 10 We can repeat this trick when setting each fish's position. Pull in a 'go to x: x y: y' block, and put the same 'pick random' operator in where you'd normally type the x and y coordinates. Set the ranges from -240 to 240 for each one. That ensures that a fish might appear anywhere on the screen. Finally, go to Looks and add a 'show' block. This tells the fish to become visible once spawned,

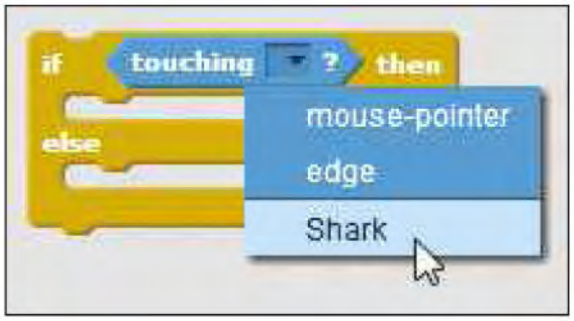


```
when I start as a clone
change color effect by pick random 1 to 100
go to x: pick random -240 to 240 y: pick random -240 to 240
show
```



coloured and in the right place. Try running the program and pressing Space a few times to see.

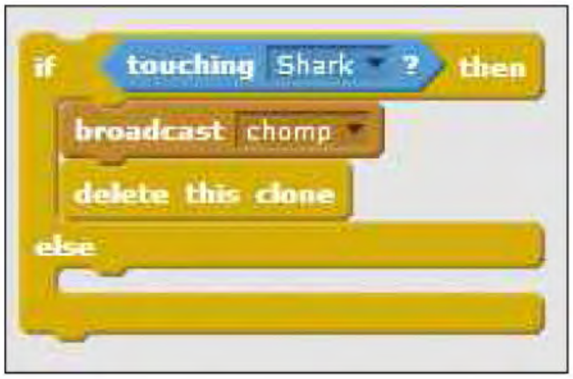
STEP 11 Pull a Forever loop underneath the 'show' block. Now it's time for our next conditional statement. The 'if, then, else' block tells the fish to look for a particular condition, but if it's not happening, to keep on doing something else. In this case, we want to know if the fish is touching the shark. Bring the 'if, then, else' block in from the Control



```
if touching ? then
else
```

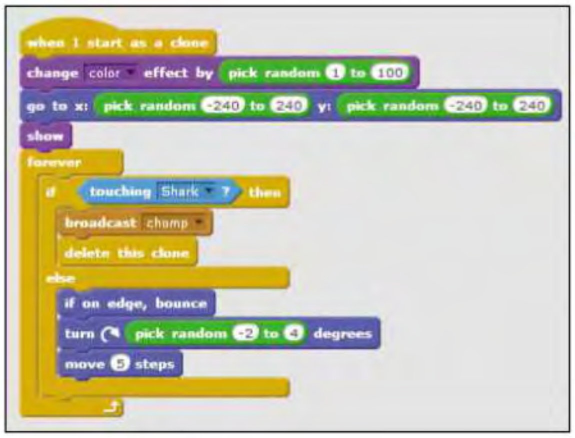
category, then go to Sensing and drag a 'touching' block into the top slot. Change the sprite to the Shark.

STEP 12 We want two things to happen if a fish touches the shark. First, we want to broadcast a new message, which we'll call 'chomp'. Second, we want the fish to disappear, since it's been chomped. You'll find the 'broadcast' message block in the Events category and the 'delete this clone' block under Control.



```
if touching Shark ? then
  broadcast chomp
  delete this clone
else
```

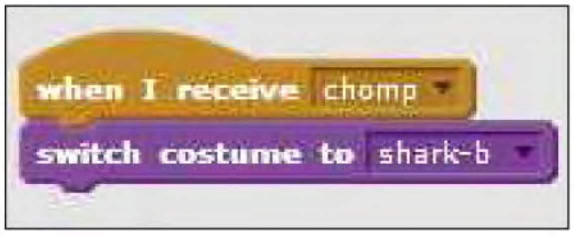
STEP 13 What do we want the fish to do otherwise? Swim around! Go to the Motion category and drag an 'if on edge, bounce' block, a 'turn x degrees' block and a 'move x steps' block into the bottom C-shape of the 'if, then, else' block. Use another 'pick random' block in the 'turn x degrees' block and set the numbers to -2 and 4. Leave the 'move x steps' block set to 5. Now drag this whole block inside the Forever block



```
when I start as a clone
change color effect by pick random 1 to 100
go to x: pick random -240 to 240 y: pick random -240 to 240
show
forever
    if touching Shark ? then
        broadcast chomp
        delete this clone
    else
        if on edge, bounce
        turn ↻ pick random -2 to 4 degrees
        move 5 steps
```

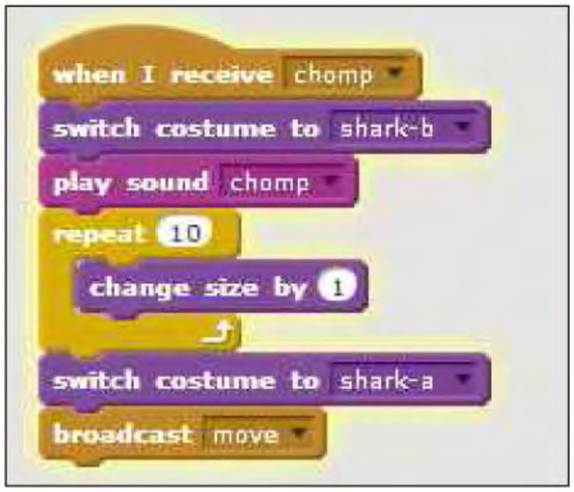
we put in place earlier. That's our fish clones ready to go.

STEP 14 What happens to the shark when it eats a fish? It's going to chomp it and grow bigger. Go to the Sprites area and click on the Shark sprite, then start a new block of script with a 'when I receive' message block. Change the message to 'chomp'. Now go to Looks and drag in a 'change costume to' block, and set the costume to 'shark-b' – a nice jaws-wide-open pose. Click the Sounds tab followed by the 'Choose sound from library' button. Select the 'Chomp sound under Animal' button and click OK, then drag a 'play sound' block into place and set it to 'chomp'.



```
when I receive chomp
switch costume to shark-b
```

STEP 15 Now to make our shark grow. Add a Repeat block and leave it set at 10. Go to Looks, and add a 'change size by' block and leave it set to 1.



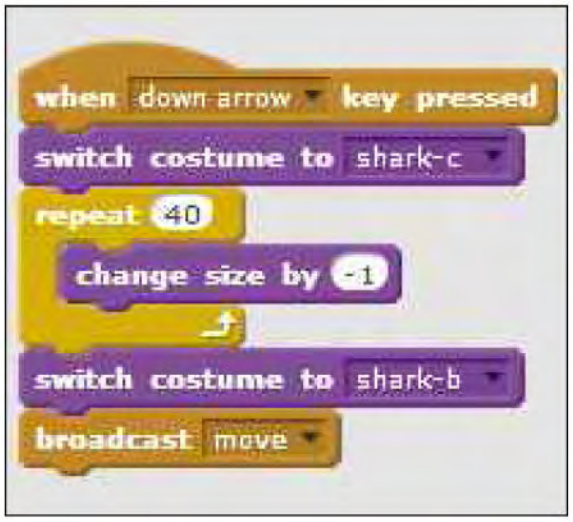
```
when I receive chomp
switch costume to shark-b
play sound chomp
repeat 10
    change size by 1
switch costume to shark-a
broadcast move
```

IF, THEN, ELSE

The 'If, then, else' block is a more complex version of the 'If, then' conditional statement. It tells the program to check whether a condition has been met then what to do if it has, and also what to keep doing if the condition hasn't been met. However, it only checks once and then stops both checking and performing the 'else' actions. If you want your Scratch program to keep on checking, make sure you put your conditional statements inside a 'forever loop' block.

Now add a second 'switch costume' block and set it back to 'shark-a'. Finally, add a 'broadcast' message block and set it to 'move'. This will make our shark grow steadily for a brief time, then switch back to its original costume and continue moving. The more fish you spawn with your spacebar, the more your shark will eat and grow.

STEP 16 That's a problem when our shark gets too big. We need another key to make him feel a little queasy and shrink him back to a more manageable size. That's exactly what this last block of script puts into action. By now you know where all the blocks are – just make sure you have the Shark sprite selected, then drag them into the Scripting area. ●



```
when down arrow key pressed
switch costume to shark-c
repeat 40
    change size by -1
switch costume to shark-b
broadcast move
```

Your first Scratch game

It's time to take your Scratch skills up a notch or two by learning how to code this simple arcade shoot-em-up game

WHAT YOU'LL LEARN

- How to put together a simple arcade game
- How to make keyboard controls work
- How to use variables to control the game

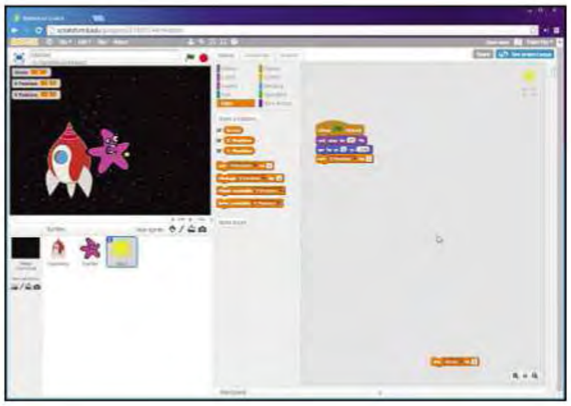
TOP TIP
You can define whether variables are or aren't visible on the screen by ticking and unticking the boxes next to them in the Scripts Palette.

By now, you should be ready to tackle something bigger, so we're going to build our first Scratch game. Old-school arcade games are a good programming exercise. They involve a lot of interaction, we all know what the rules are and it's fairly easy to piece together how they should work. What's more, you get something you might want to play. This one is going to be a shoot-em-up, where marauding invaders turn up at the top of the screen and make their way down, while a spaceship at the bottom of the screen tries to dodge them or riddle them with bullets.

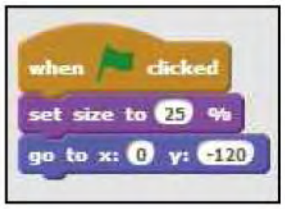
Before we start coding, let's think about what we need. Item one is the spaceship, which needs to move left and right, and fire bullets. Item two would be our invaders, and they need to appear at the top of the screen and move downwards towards the bottom. Item three would be the player's bullets, which need to travel upwards from the spaceship towards the invaders. That means we need to work with at least three sprites.

On top of this, we need to plan for three possible interactions. First, if a starfish invader hits the spaceship, the spaceship explodes and it's game over. Second, if the player fires the spaceship's cannons, a bullet needs to be released and travel up the screen. Finally, if a bullet hits a starfish, the starfish has to explode and the player's score has to go up. Now we know the basic elements, we can get them all working in the game.

STEP 1 First, we need to get our most basic elements together. Right-click on the Scratch cat sprite to delete it, then go to the backdrop area, click the 'Choose backdrop from library' button, then select the Stars backdrop. Now, we need to get hold of our three sprites: the Spaceship, the Starfish and Star2. For all three, click the 'Choose sprite from library' button, find and select them in the library, then click OK.

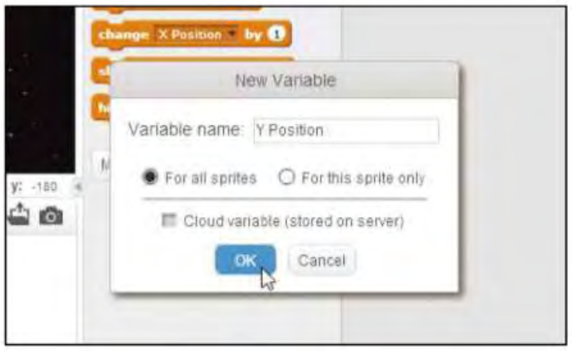


STEP 2 Click on the spaceship, and let's get scripting. The basic stack is pretty simple. First, we need the 'when green flag clicked' block, then we get a 'set size to x' block from the Looks category, and set the size to 25%. This makes our spacecraft a more manageable size. Now we drag in a 'go to x: x y: y' block from the Motion category, and set x to 0 and y to -120. This tells the spaceship to start at the bottom of the Stage.



```
when [flag] clicked
set size to 25 %
go to x: 0 y: -120
```

STEP 3 We're going to use some variables to control the score and where our sprites and any clones appear on the screen. Click on the Data category, then click the Make a Variable button. Call the first variable 'Score' and click on the little



New Variable

Variable name: Y Position

● For all sprites ○ For this sprite only

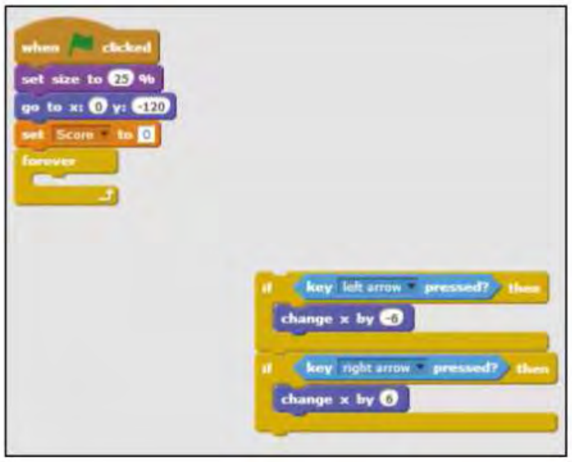
☐ Cloud variable (stored on server)

OK Cancel

```
change X Position by 1
```

round radio button to make it available 'For all sprites'. Now repeat this step twice more, creating one variable called X Position and one variable called Y Position. Now drag the 'set variable to x' block into the Scripts area and place it at the bottom of the stack. Set the variable to 'Score' and leave the value at 0.

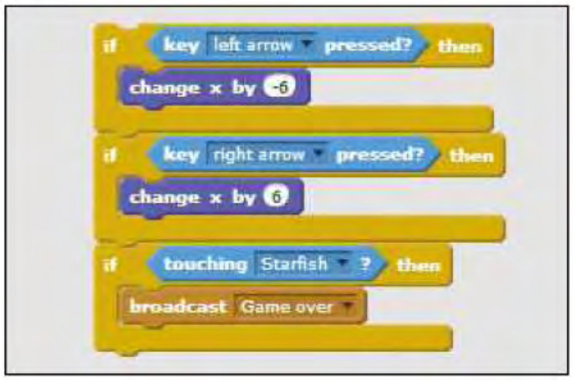
STEP 4 Now to control the spaceship's movement. Drag in a 'forever' block from Control and place it at the bottom of the stack. Next, drag an 'if then' block into the Scripts area and place it on its own. Go to the Sensing category, and drag the 'key x pressed?' block into the space at the top of the 'f then' block. Now go to Motion and pull in the 'change



```
when 🏴 clicked
set size to 25 %
go to x: 0 y: -120
set Score to 0
forever

if key left arrow pressed? then
    change x by -6
if key right arrow pressed? then
    change x by 6
```

x by x' block. Set the key to the 'left arrow', and ask x to change by -6. Now duplicate this little block, and set the key to the 'right arrow', and ask x to change by 6. Stack the two blocks together. The first 'if' block tells the spaceship to move 6 pixels to the left if you press the left arrow, while the second tells it to move 6 pixels to the right when you press the right arrow.

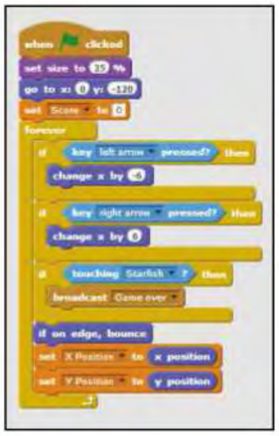
STEP 5 Drag in another 'if then' block. This time, we need the 'touching?' block from the Sensing category, and a 'broadcast message' block from Control. Set the 'touching?' block to look for the Starfish sprite, and set up a new message,



```
if key left arrow pressed? then
    change x by -6
if key right arrow pressed? then
    change x by 6
if touching Starfish ? then
    broadcast Game over
```

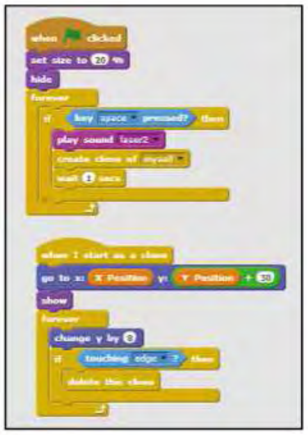
which we'll call 'Game over'. Attach this block to the other two 'if then' blocks, then drag the whole stack inside the C shape of the 'forever' block.

STEP 6 We need three more blocks to finish this script. The first, 'if on edge, bounce' from the Motion category, tells our spaceship to bounce off if it hits the left- or right-hand edge of the screen. The other two set the two variables we set up earlier. Go to the Data category, and drag the 'set variable to x' block in twice, making sure each goes inside the 'forever' block. Set the first to X Position and the bottom to Y Position, but for the values go to the Motion category and drag in the 'X Position block' for the first, and the 'Y Position' block for the second. You'll find both at the bottom of the Blocks Palette. These variables will store the current location of the spaceship.



```
when 🏴 clicked
set size to 35 %
go to x: 0 y: -120
set Score to 0
forever
    if key left arrow pressed? then
        change x by -6
    if key right arrow pressed? then
        change x by 6
    if touching Starfish ? then
        broadcast Game over
    if on edge, bounce
    set X Position to x position
    set Y Position to y position
```

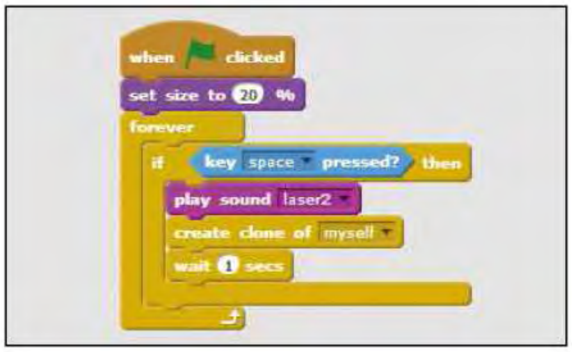
STEP 7 That's it for now for the spaceship; let's start work on the bullet. Click on the Star2 sprite and start a new stack with the 'when green flag clicked' block. Bring in a 'set size to x%' block and set it to 20, then a 'hide' block. Next, we need a 'forever' block, and inside that place an 'if then' block. Grab the 'key x pressed' block and drop it into place, and set the key to 'space'.



```
when 🏴 clicked
set size to 20 %
hide
forever
    if key space pressed? then
        play sound laser2
        create clone of myself
        wait 1 secs

when I start as a clone
go to x: X Position y: Y Position + 30
show
forever
    change y by 8
    if touching edge ? then
        delete this clone
```

>

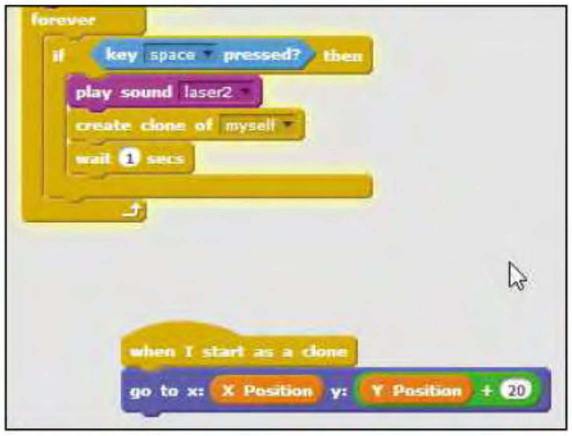
STEP 8 Now click on the Sound tab, then the 'choose a sound from library' button, and add the laser2 sound from the Electronic category to this sprite. Return to the Scripts menu, and drag the 'play sound' block into the 'if then' block, before setting the sound to 'laser2'. Beneath it, stack the 'create clone of...' block and set it to 'myself', then a 'wait x secs' block, which you should set to 1. This little stack tells a bullet to fire when the space key is pressed,



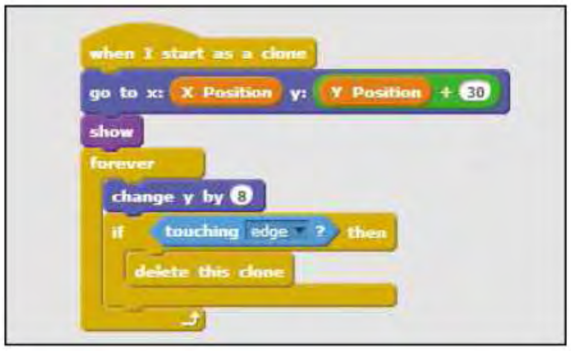
```
when ⚑ clicked
set size to 20 %
forever
  if key space pressed? then
    play sound laser2
    create clone of myself
    wait 1 secs
```

which actually means making the laser2 sound and creating a clone of the sprite. The 'wait' block limits the rate of fire.

STEP 9 This next script tells the cloned sprite where to appear and what to do. It starts off with a 'when I start as a clone' block, then it's time to use our variables. Pull out a 'go to x: y:' block from the Motion category, then go to Data and drag the block for the X Position variable into the space next to the x:. Now go to the Operators category and drag out the block at the top with two circular spaces separated by a plus. Drop it next to the y:. Now go back to Data and drag the 'Y position' block into the first circular space, then click in the second circular space and type 20. This tells our little bullet to appear wherever the spaceship is, but just above it.



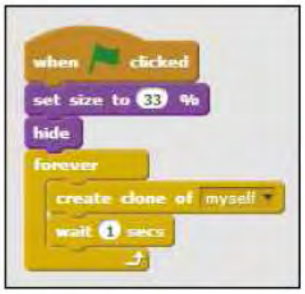
```
forever
  if key space pressed? then
    play sound laser2
    create clone of myself
    wait 1 secs

when I start as a clone
go to x: X Position y: Y Position + 20
```



```
when I start as a clone
go to x: X Position y: Y Position + 30
show
forever
  change y by 8
  if touching edge ? then
    delete this clone
```

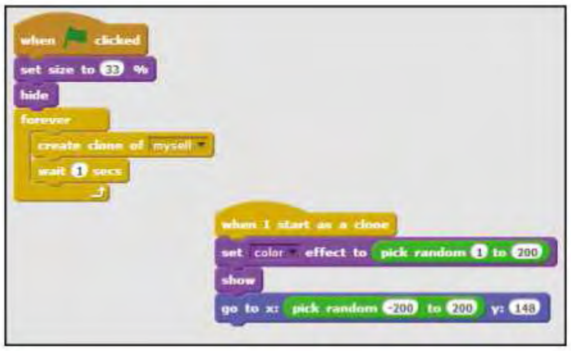
STEP 10 Next, stack the 'show' block, then a 'forever' block. Inside it, add a 'change y by..., block from the Motion category, and set the value to 8. Beneath that, add an 'if then' block. Grab another 'touching?' block from Sensing, and set it to 'edge'. Now get a 'delete this clone' block from Control, and drag it inside the 'if then' block.

STEP 11 We have a spaceship that moves and a bullet that it can fire upwards. All we need now is an enemy to shoot at. Click on the Starfish sprite and let's get coding. First, we create a new stack under a 'when green flag clicked' block. Pull in a 'set size to...' block and set the size to 33%. Add a 'hide' block underneath. Next comes a 'forever' block, and inside this we put a 'create clone of...' block, setting the sprite to clone as 'myself'. Beneath this, stack a 'wait' block, and set it to 1 secs. This stack tells the program to create a new starfish every second.



```
when ⚑ clicked
set size to 33 %
hide
forever
  create clone of myself
  wait 1 secs
```

STEP 12 This next stack tells these starfish what to do. First, a 'set x effect' block gives each one a different colour. Set the effect to 'color', then go to the 'operators' block and pull in the 'pick random' block. Enter 1 and 200 as the values. Next,

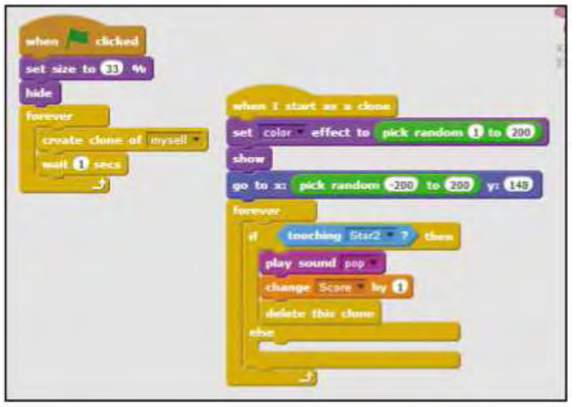


```
when ⚑ clicked
set size to 33 %
hide
forever
  create clone of myself
  wait 1 secs

when I start as a clone
set color effect to pick random 1 to 200
show
go to x: pick random -200 to 200 y: 148
```

TOP TIP
Can't remember which category to get a certain block from? They're all colour coded according to their category. If you see a bright orange block, it will always come from the Data category, while bright green blocks will always be Operators.

we want a 'show' block, to make each Starfish appear. However, we want each to appear in a different place. Drag the 'go to x: y:' block into place at the bottom of the stack, and pull another 'pick random' block in next to the x:. Set the values in the 'random' block to -200 and 200, then set the y value in the 'go to' block to 148. This tells each starfish clone to pop up anywhere at the top of the screen, with a little space so that they don't appear right at the sides.

STEP 13 This is where things get a bit complicated. We want out starfish to keep travelling downwards, unless it gets hit by a bullet or hits the bottom of the screen. The secret is to use two 'if, then, else' blocks, with one actually nested inside the other. Drag a 'forever' block to the bottom of the stack, then put the first 'if, then, else' block inside it. Next to the 'if', put a 'touching ?' block from the Sensing category, and set it to 'Star2'. Inside this block, drag the

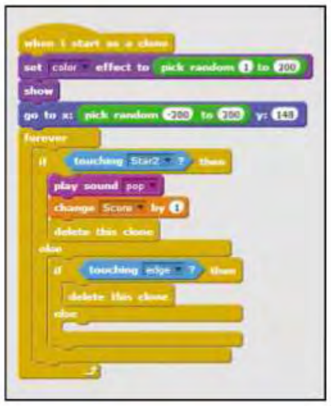


```
when [flag] clicked
set size to 33 %
hide
forever
    create clone of myself
    wait 1 secs
```

```
when I start as a clone
set color effect to pick random 1 to 200
show
go to x: pick random -200 to 200 y: 148
forever
    if touching Star2 ? then
        play sound pop
        change Score by 1
        delete this clone
    else
```

'play sound' block with it set to 'pop', then go to the 'data' block and find the 'change variable to x' block. Change the variable to 'Score' and set it to change by 1. Finally, pull a 'delete this clone' block into place. If a bullet hits a Starfish, it will make a pop noise, increase the score by 1, then disappear.

STEP 14 Now drag a second 'if, then, else' block inside the bottom space of the first. This time, pull a 'touching?' block into place next to the 'if' and set it to 'edge'. Go to the Control category, and pull a 'delete this clone' block into the space underneath. If a starfish hits the bottom edge of the screen, it will now disappear.



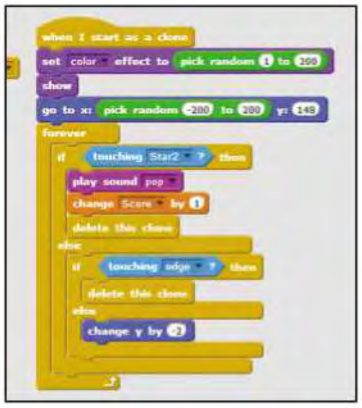
```
when I start as a clone
set color effect to pick random 1 to 200
show
go to x: pick random -200 to 200 y: 148
forever
    if touching Star2 ? then
        play sound pop
        change Score by 1
        delete this clone
    else
        if touching edge ? then
            delete this clone
        else
```

ABOUT VARIABLES

Variables are a common element of any programming language. They work like the values that we type into a block of script to tell a sprite to go to a specific point or a 'repeat' block to repeat ten times. But with a variable, the number is stored elsewhere by the program, and retrieved every time a block asks for it. In effect, the number gets stored in a box with a label. We tell the program what to store in the box, and it only has to ask for the label to use whatever number is in it.

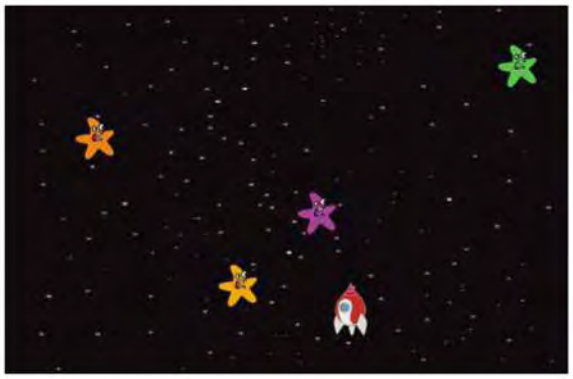
This has a lot of advantages. First, you can share a variable between different parts of the program. For example, you could use a variable to track the horizontal position of one sprite so that the program knows when another sprite is directly above it. We'll use this with the 'X Position' block when we want our spaceship to fire bullets. We can also change and make calculations with variables. We could control the size of a series of oranges by defining a variable called size, then making the number a little bit bigger with every orange. The more complex your program is, the more you'll want to use variables, and the more useful they become.

STEP 15 All we need to do now is get our starfish moving. Just get a 'change y by…' block from the Motion category and drag it into the space underneath the bottom X. Set the value to -2.



```
when I start as a clone
set color effect to pick random 1 to 200
show
go to x: pick random -200 to 200 y: 148
forever
    if touching Star2 ? then
        play sound pop
        change Score by 1
        delete this clone
    else
        if touching edge ? then
            delete this clone
        else
            change y by -2
```

STEP 16 Finally, we need to sort out what happens when our spaceship gets hit by a falling starfish. Remember the 'Game over' message we set to broadcast earlier? Create a new stack on any of the sprites with a 'when I receive message' block. Set the message to 'Game over'. Drag in a 'say x for x secs' block, and set it to say GAME OVER for two seconds. Then go to the Control category, and grab a 'stop' block. Leave it set to 'all'. If your ship is now hit by a starfish, the game will stop. Not very exciting, we know. If only there was something we could do… ●



Remix the game

Not satisfied with your finished game? The great thing about programming is that you can go back and fix it, or even make it more spectacular!

WHAT YOU'LL LEARN

- How to change or remix your Scratch projects
- How to add music to a game
- How to edit sprites and costumes
- How using variables can transform your game

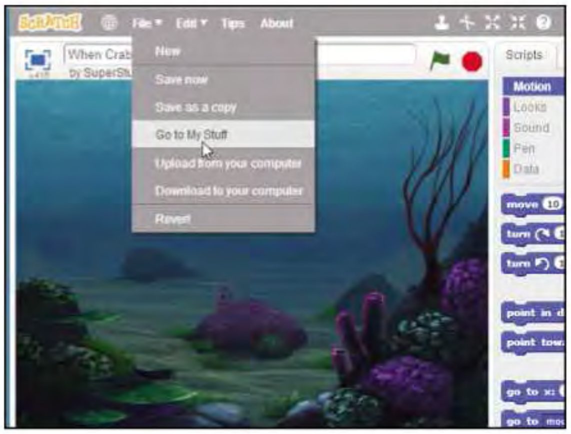
TOP TIP

The more sprites you have onscreen and the more animation, the more processing power your game needs to run, and the slower it might run on some less powerful computers.

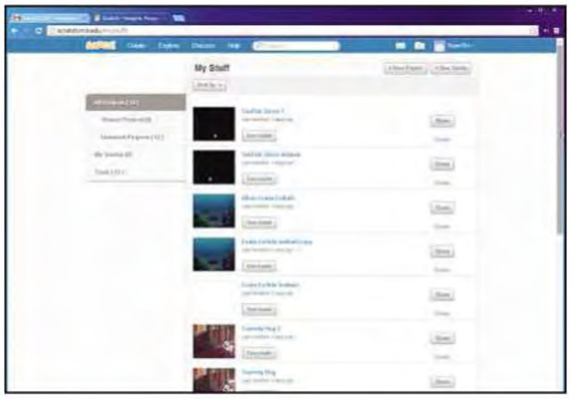
Talk to any programmer and they'll tell you that few projects are ever really finished – there's always something you can do to improve them. With this in mind, we're going to take another look at our first Scratch game. It's functional and playable as it is, but what would it be like if we changed the graphics and added music, or made it a little more exciting? It's time to find out.

Load up the game

You can load any project you've already created by clicking on the My Stuff folder in between the Messages icon and Your profile icon on the Scratch web page. You'll see all your existing projects listed. Alternatively, you can click on the File menu in any open Scratch project, then select Go to My Stuff. Open a project from the list by clicking on it, then click the See Inside button to go back, edit and remix it.



Click on Go to My Stuff on the File menu in any open Scratch project to see all your existing projects listed.

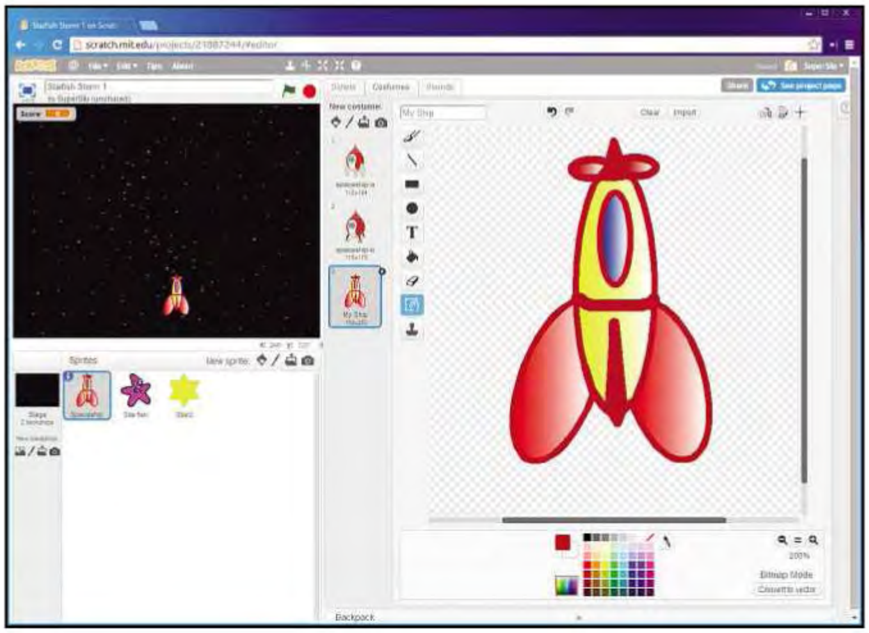


You can load any project you've already created by clicking on the My Stuff folder.

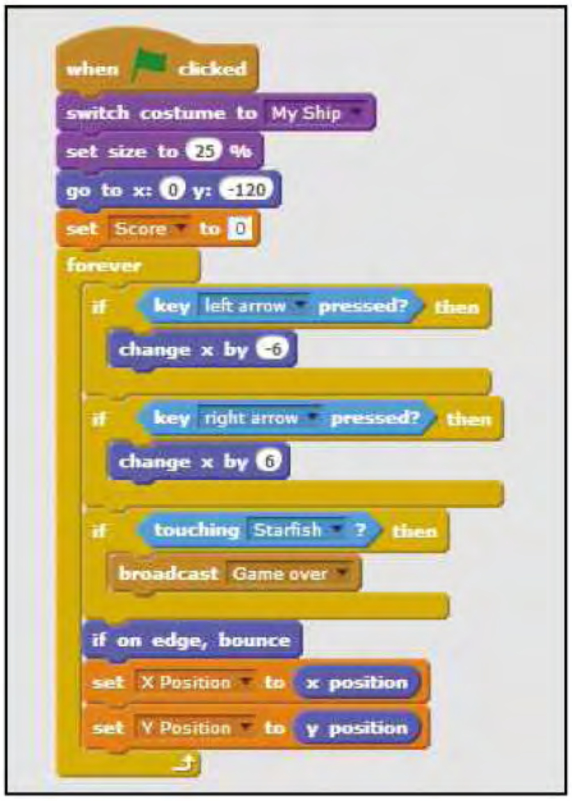
Transform your spaceship

There's nothing wrong with Scratch's built-in spaceship, but was it really built for a fast-paced action game? No. If you want to change the way it looks, you have two options. The first is to create a brand-new sprite. Next to the 'Choose sprite from library' button at the top of the Sprites area there's a paintbrush button: 'Paint new sprite'. Click this and you'll bring up the Costumes Editor, where you can use the tools provided to draw a new sprite.

Once you've finished, clicking on the blue 'i' symbol in the top left of the sprite in the Sprites area will bring up a window where you can type the new sprites name, set its starting direction (which way it faces) and set whether it can be rotated so that it will face in any direction, only face left or right, or only face in one direction, using the three buttons marked rotation style. Your new sprite will be saved when you next save your project. The only downside of this approach is that you'll need to copy all the scripts for the



▲ Why not give your exisiting spaceship a new costume?



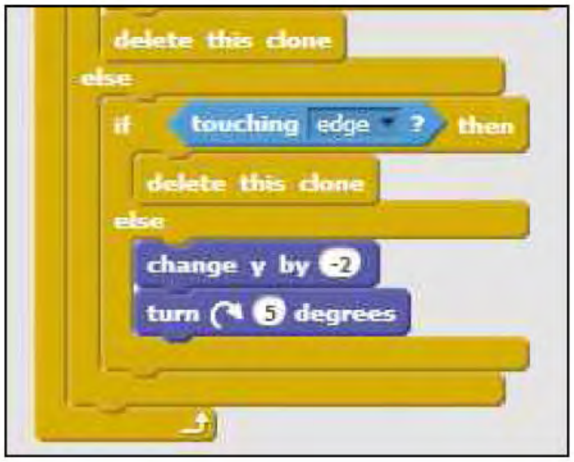
```
when [green flag] clicked
switch costume to [My Ship]
set size to (25) %
go to x: (0) y: (-120)
set [Score] to [0]
forever
    if <key [left arrow] pressed?> then
        change x by (-6)
    if <key [right arrow] pressed?> then
        change x by (6)
    if <touching [Starfish] ?> then
        broadcast [Game over]
    if on edge, bounce
    set [X Position] to (x position)
    set [Y Position] to (y position)
```

old spaceship over to the new one, and change any references made to the spaceship to the new sprite's name.

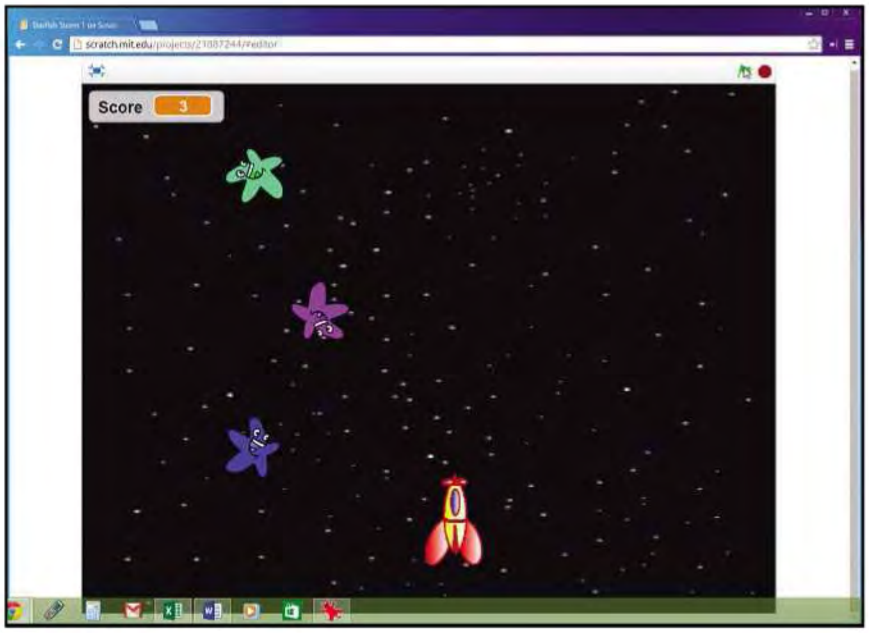
The alternative? You can give your existing Spaceship sprite a new costume. Simply click on it in the Sprites area, then click the Costumes tab above the Blocks Palette. Click the 'Paint new costume' button above the two existing costumes, and start drawing. When you're finished, you can just delete the old costumes by clicking on them in the list, then clicking the x on the right-hand side of the highlight box. Alternatively, you can just ensure your sprite is highlighted in the list of costumes, then add a 'Switch costume to x instruction at the start of the Sprite script, and set it to the new costume.

Add some animation

We can make our starfish even more fearsome by adding a little animation to the Sprite. All it takes are a few additions to the Starfish script. Click on the Starfish sprite in the Sprites area, then on the



```
        delete this clone
    else
        if <touching [edge] ?> then
            delete this clone
        else
            change y by (-2)
            turn ↻ (5) degrees
```



▲ The starfish now turn around and around as they make their descent.

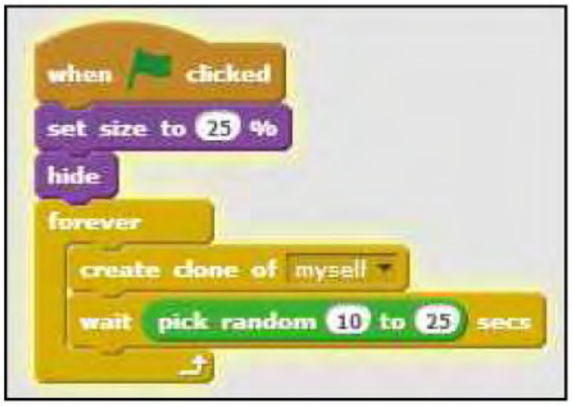
Scripts tab to edit its scripts. We need to adjust its default behaviour – what the sprite does when it isn't being touched by a bullet, and it hasn't touched the edge of the screen. See where the block says 'change y by -2'? That's it.

We're not going to use different frames of animation this time. We just want our starfish to turn around and around as they fall. We do this by adding a 'turn x degrees' block below the 'change y by -2' block, and set the rotation to '5'. Our starfish now turn smoothly as they make their menacing descent.

Make a new enemy

Our starfish put up a decent fight, but any good arcade game needs more than one enemy, so you might want to add this killer super-starfish.

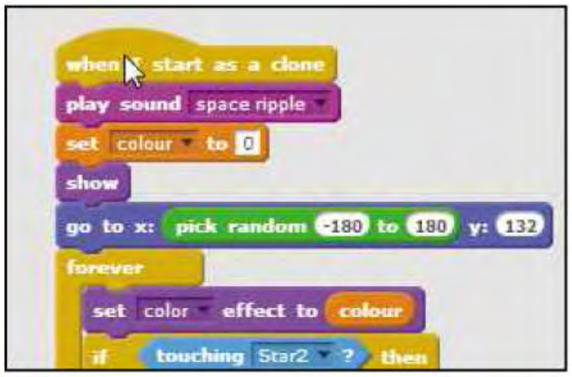
STEP 1 You can start off by duplicating the original starfish. Click in the 'I' in the top left of the highlight box that appears when he's selected if you want to change his name. Here, we're calling him SuperStar. It's once you get to the scripts, however, that you need to make some changes. First, take a look at the 'when Green flag clicked' stack. This script is still designed to keep on producing clones, but here we've put in a 'pick random' block in the 'wait'



```
when [flag] clicked
set size to 25 %
hide
forever
    create clone of myself
    wait pick random 10 to 25 secs
```

block instead of the usual 1 second value. This means a new super-starfish will now spawn at random intervals between 10 and 25 seconds, making him a more dangerous and unpredictable opponent. We've also changed the size of the sprite to just 25%, to make him smaller and harder to hit.

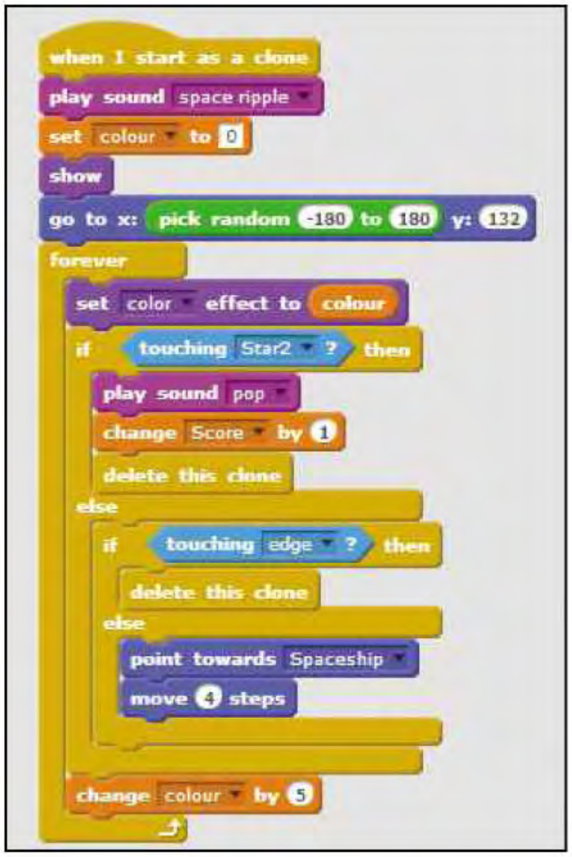
STEP 2 Look at the 'when I start as a clone' stack, and you'll see more changes. First, we've added a sound to the sprite: the space ripple from the Electronic category. If we place this here, before the 'forever' loop, this spooky sample will play every time a super-starfish appears. We've also created a new variable called 'colour', which we're using for a cool colour-change effect. We've added a 'set color



```
when I start as a clone
play sound space ripple
set colour to 0
show
go to x: pick random -180 to 180 y: 132
forever
    set color effect to colour
    if touching Star2 ? then
```

effect to x' block to the start of the 'forever' loop, and set the value to use the Colour variable. By putting a 'change colour by x' block at the end of the loop and setting it to 5, we can make the clone change colour every time its script runs through the loop, which happens pretty fast as he moves down the screen.

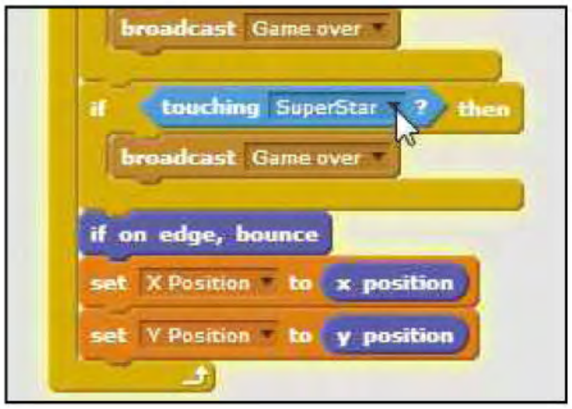
STEP 3 Finally, this enemy has a different style of movement. He mercilessly hunts down our player's spaceship. We do this by replacing the old 'change y by -2' block at the bottom of the second 'if, then, else' block so that it uses a 'point towards' block, set to the Spaceship sprite, then moves 4



```
when I start as a clone
play sound space ripple
set colour to 0
show
go to x: pick random -180 to 180 y: 132
forever
    set color effect to colour
    if touching Star2 ? then
        play sound pop
        change Score by 1
        delete this clone
    else
        if touching edge ? then
            delete this clone
        else
            point towards Spaceship
            move 4 steps
    change colour by 5
```

steps towards that target. This tells the sprite to keep pointing at the spaceship, wherever it's moving, then move steadily towards it. Our spaceship will now have to shoot it to survive.

STEP 4 To make super-starfish a credible threat, we also need to change the spaceship's script. If you look, we've added a fourth 'if then' block to the stack, identical to the one that looks to see whether the spaceship is touching the original starfish.

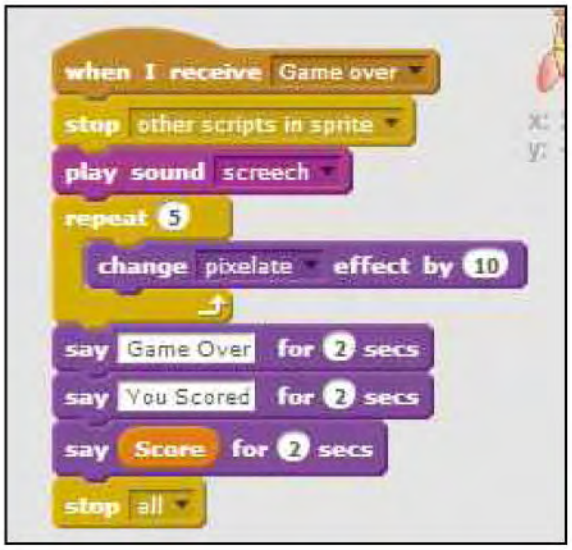


```
    broadcast Game over
  if touching SuperStar ? then
    broadcast Game over
  if on edge, bounce
  set X Position to x position
  set Y Position to y position
```

Set this to the SuperStar sprite, and it's game over if our spaceship makes contact with our super-starfish.

Making Game Over mean something

At the moment, the Game Over message is a bit of a damp squib. For a start, we could make our spaceship explode, using the same pixelate effect we used in our 'When Crabs Collide' project. Look at the new script, and you can see that we've changed the 'stop' block to read 'other scripts in sprite', then added the screech sound and a 'play sound' block to play it. We then run a short loop with a Pixelate effect that turns our spaceship into a pile of chunks. We then add three 'say' blocks, each one running for two seconds. The first says 'Game Over', the second says 'You Scored', and the third looks at the current value of the Score variable and reports the score. Add a 'stop all' instruction at the end, and it's a much better Game Over.

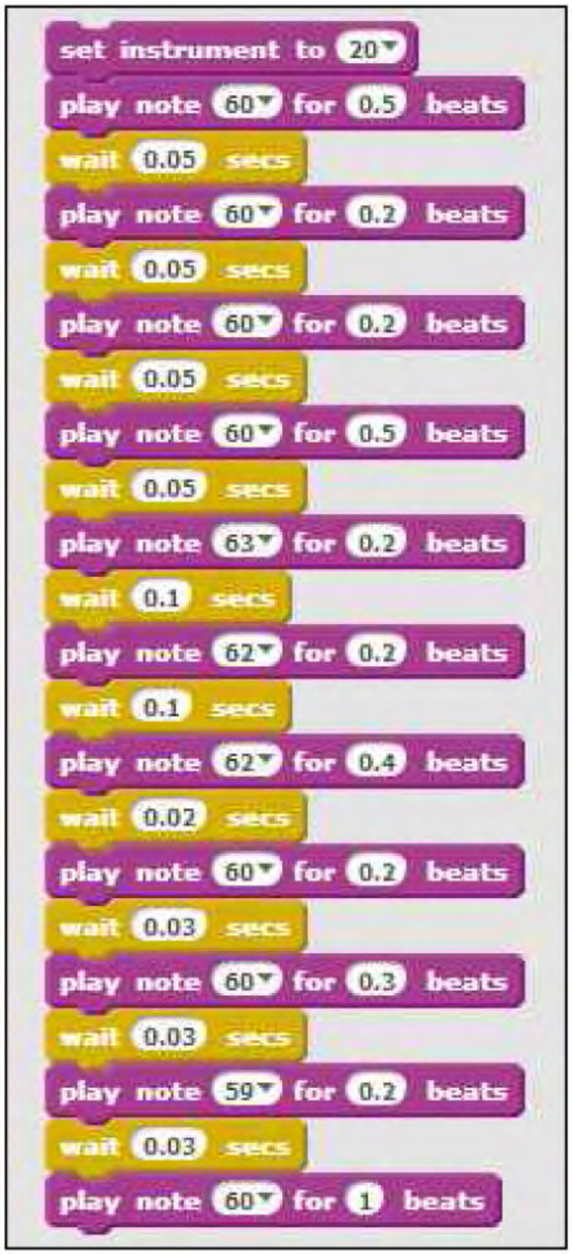


```
when I receive Game over
stop other scripts in sprite
play sound screech
repeat 5
  change pixelate effect by 10
say Game Over for 2 secs
say You Scored for 2 secs
say Score for 2 secs
stop all
```

Have a go at making the Game Over message a bit more exciting.

Music

Still, we could make it even better. This little stack adds the classic funeral march used by so many vintage arcade games. Just stick it in between the



```
set instrument to 20
play note 60 for 0.5 beats
wait 0.05 secs
play note 60 for 0.2 beats
wait 0.05 secs
play note 60 for 0.2 beats
wait 0.05 secs
play note 60 for 0.5 beats
wait 0.05 secs
play note 63 for 0.2 beats
wait 0.1 secs
play note 62 for 0.2 beats
wait 0.1 secs
play note 62 for 0.4 beats
wait 0.02 secs
play note 60 for 0.2 beats
wait 0.03 secs
play note 60 for 0.3 beats
wait 0.03 secs
play note 59 for 0.2 beats
wait 0.03 secs
play note 60 for 1 beats
```

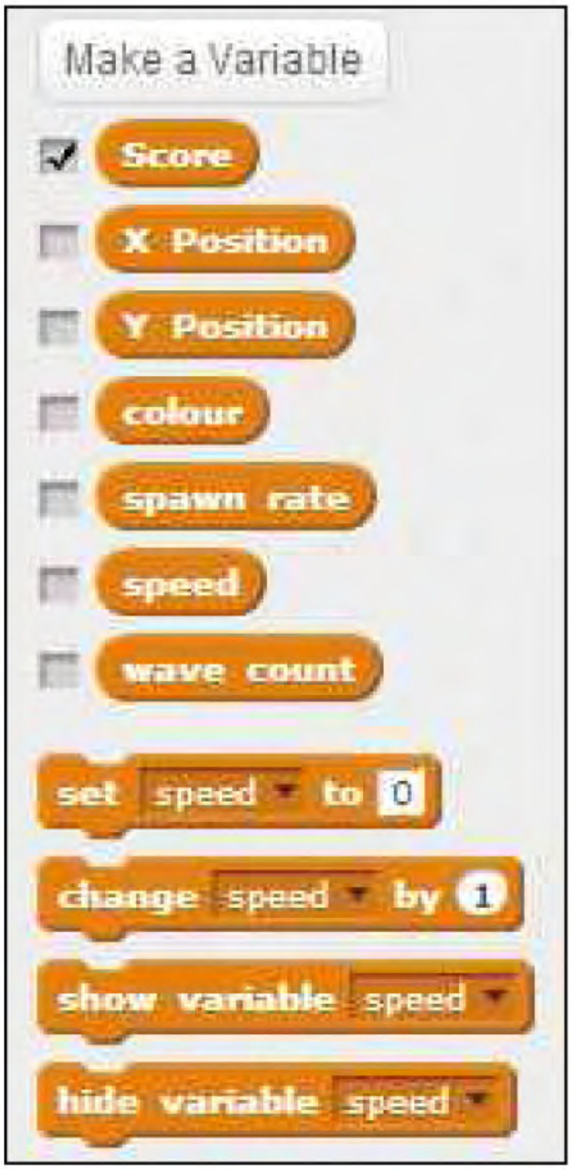
Even better, why not add some music in the form of the classic funeral march?

Pixelate animation and the Game Over messages, and it should work pretty well.

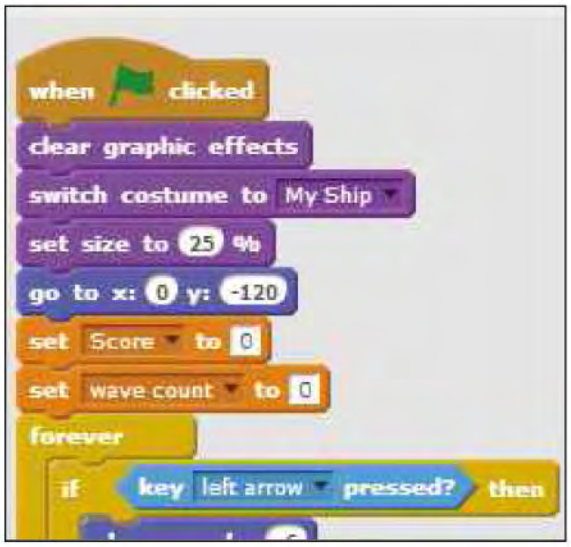
Make some waves

Classic arcade games used to have aliens that came in waves, increasing the difficulty as the player goes on. How can we make this happen here? Using variables, of course.

STEP 1 We use one – wave count – to work out how many starfish the player has destroyed. When the count hits 20, the game starts the next wave. We then use another two – spawn rate and speed – to tell the program how quickly to spawn new starfish, and how fast to make them move.



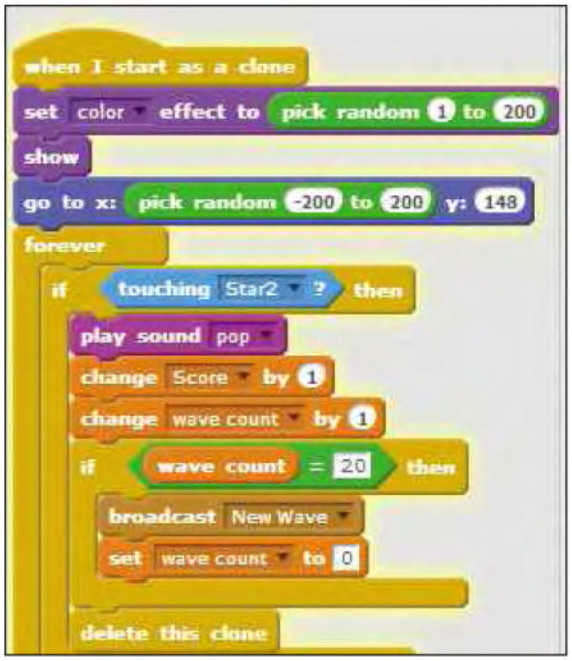
```
Make a Variable
[✓] Score
[ ] X Position
[ ] Y Position
[ ] colour
[ ] spawn rate
[ ] speed
[ ] wave count

set speed to 0
change speed by 1
show variable speed
hide variable speed
```



```
when [flag] clicked
clear graphic effects
switch costume to My Ship
set size to 25 %
go to x: 0 y: -120
set Score to 0
set wave count to 0
forever
    if key left arrow pressed? then
```

STEP 2 Once we've made our variables, we need to put them in a handful of scripts. First, go to the Starship script and add a 'set x to x' block from Data, setting it to wave count and 0.

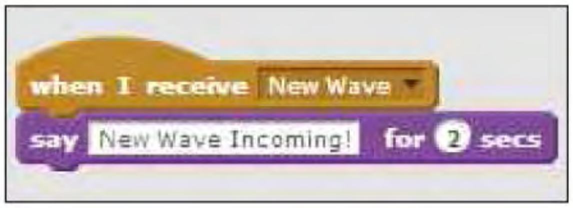
STEP 3 Next, go to the Starfish sprite, and underneath the 'change score by 1' block put in an identical block reading 'change wave count by 1'. Underneath this, we add an 'if then' block from the Control category, with an 'x=x' block from the Operators. Drop the Wave Count variable into the first space, then type 20 in the second. In the space underneath, put in a 'broadcast message' block, with a new message called 'New Wave', then a new 'set



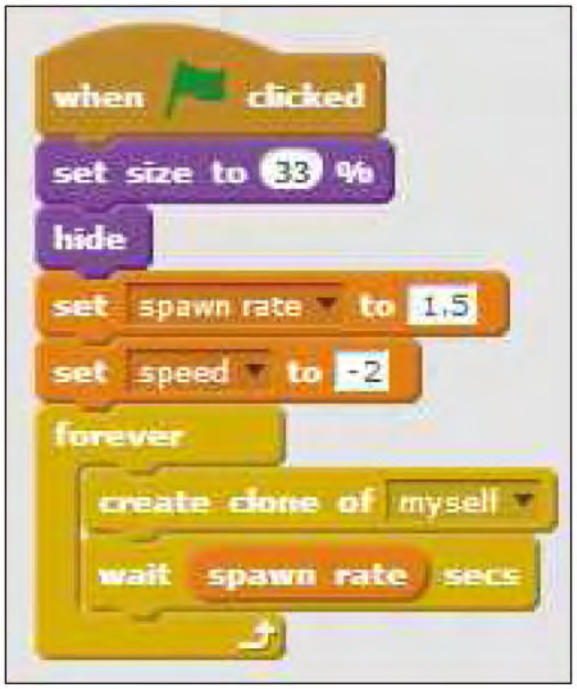
```
when I start as a clone
set color effect to pick random 1 to 200
show
go to x: pick random -200 to 200 y: 148
forever
    if touching Star2 ? then
        play sound pop
        change Score by 1
        change wave count by 1
        if wave count = 20 then
            broadcast New Wave
            set wave count to 0

        delete this clone
```

wave count' block to reset the Wave Count variable to 0. This tells the starfish to keep adding 1 to wave count every time a starfish is destroyed, and when the wave count hits 20, to broadcast a new wave message. Then it resets the wave count to 0.

STEP 4 Now what happens when the New Wave message is broadcast? Well, first we add a whole new block to the Spaceship sprite, which tells it to say 'New Wave Incoming!' when it gets the message. Second, we need to make a few more changes to both Starfish scripts. In the first



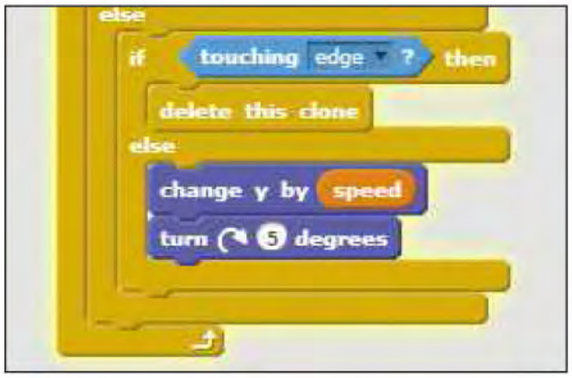
```
when I receive New Wave
say New Wave Incoming! for 2 secs
```



```
when clicked
set size to 33 %
hide
set spawn rate to 1.5
set speed to -2
forever
    create clone of myself
    wait spawn rate secs
```

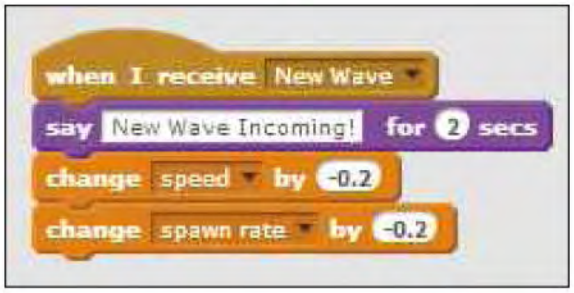
script, you can see we've added new 'set x to x' blocks from the Data category to set the spawn rate to 1.5 and the speed to -2. This tells the game to start off by releasing a new starfish every 1.5 secs, with the speed pretty slow. If we change the value in the 'wait' block to use the Spawn Rate variable, it will use this to decide how long to wait.

STEP 5 In the second script, we now need to change the 'change y by -2' block near the bottom to 'change y by the Speed variable'. This allows us to speed things up when we have a new wave of starfish.

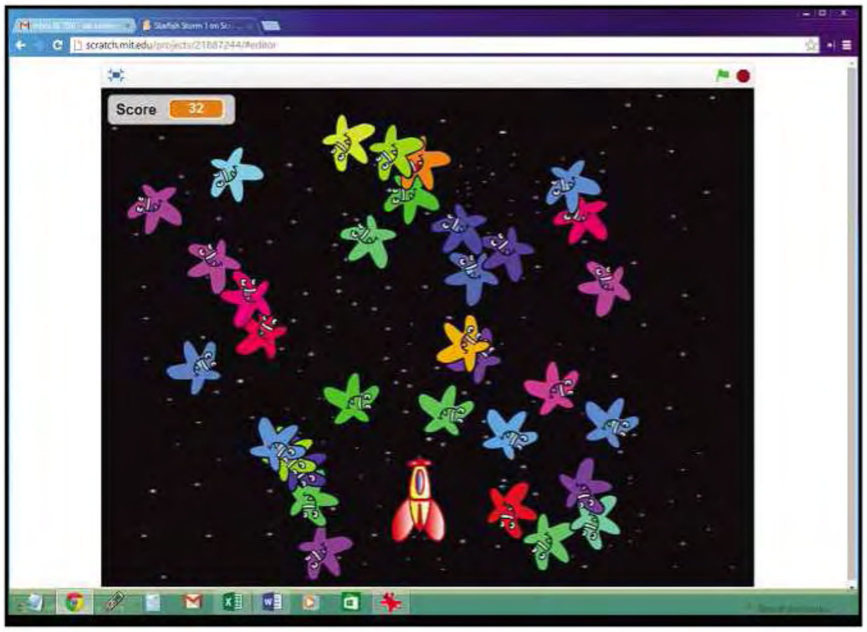


```
else
    if touching edge ? then
        delete this clone
    else
        change y by speed
        turn 5 degrees
```

STEP 6 If we now go back to the 'when I receive New Wave' stack in the Spaceship sprite, we can set the Speed variable to change by -0.2, slightly increasing the speed our starfish move at. We can also add another block to change the spawn rate by -0.2, increasing the rate at which our starfish are spawned. The game will now get more difficult with each new wave. ●



```
when I receive New Wave
say New Wave Incoming! for 2 secs
change speed by -0.2
change spawn rate by -0.2
```



▲ Here's our improved game with its waves of starfish that increase in difficulty as the player goes on.

WHAT NEXT?

You can carry on enhancing the game if you want. Why not change backgrounds with each new wave, add a new intro music to the game, or add a high score chart? Don't worry if you don't know how to do these things right now. We'll be covering them in later projects.