

4

Super Dodgeball

In this chapter, you'll make a game where you have to dodge bouncing balls and collect ice creams. As you build the game, you'll learn a new way to move sprites, how to make random numbers and use them to add suspense to a game, how to paint your own sprites, and how to copy and clone sprites. The chapter concludes with tips for changing the speed of the game.

- 64** Introducing Super Dodgeball
- 65** Setting up the variables
- 67** Preparing for the game start
- 68** Using coordinates to move
- 70** Adding more images
- 71** Renaming sprites
- 72** Making random numbers
- 73** Moving the ball
- 75** Copying and deleting sprites
- 78** Adding the energy meter
- 80** Painting in Scratch
- 82** Using Paint tools
- 84** Using colors
- 85** Using vectors in Scratch 2.0
- 86** Making the ice cream appear
- 88** Enabling the player to score
- 90** Tweaking the gameplay

Introducing Super Dodgeball

It can be hard to relax at the seaside with so many other people around, playing games and getting in the way. In Super Dodgeball, you have to avoid all the beachballs bouncing around.

Each time a ball hits you, your strength is sapped. Your energy is shown with a semi-transparent picture of your cat character, which shrinks as your strength decreases.

You score by collecting ice creams which pop up on the screen, but they soon disappear again if you're not quick enough. Ice cream also gives you a small dose of energy, to help you recover from being hit by the beachballs.

In this chapter, you'll learn how to:

- Move sprites in a new way, under keyboard control
- Design your own sprites
- Add randomness to your game
- Copy sprites
- Add sound effects
- Adjust the game's difficulty

Beware



To get this game working, you'll need to do some things differently in Scratch 1.4 to Scratch 2.0, but I'll tell you about those as you make your way through the chapter.



Setting up the variables

We'll use two variables in this game: one to keep track of the player's score, and one to keep track of the player's percentage strength remaining:

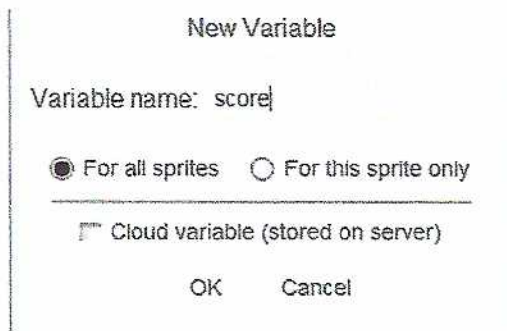
- 1 Click the **Data** button (Scratch 2.0) or the **Variables** button (Scratch 1.4) above the Blocks Palette



- 2 Click the **Make a Variable** button in the Blocks Palette

Make a Variable

- 3 Type **score** into the Variable name box. The standard options are correct for this variable (make the variable for all sprites, and don't make it a cloud variable). That means you can just press Enter or click the **OK** button



- 4 Repeat the process, but this time enter the name **strength** to create the *strength* variable

Don't forget



It's a new project, so start with a clean slate. Use the File menu and choose New or click the Create button at the top of the screen to start afresh!

Don't forget



Cloud variables are explained in Chapter 8, and are only available in Scratch 2.0.

...cont'd

- 5 When you make variables, reporting boxes are added to the Stage to display their values. They appear in the top left corner, but you can move them. Click the *score* box and drag it to the top right corner, and release the mouse button to position it there. We won't use the *strength* box, so you can leave that where it is



Hot tip



There's a lot of work going on to speed up Scratch on the Raspberry Pi, so it's worth testing the game to see whether you can display the score on screen all the time using the latest software update.

- 6 In the Blocks Palette, there are rounded blocks for each of your variables. The tickbox beside them decides whether the variable should be shown on the Stage. We'll show the player's strength using our energy meter instead of the variable box on the Stage, so untick the box beside the *strength* variable

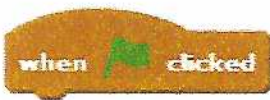


- 7 Displaying the variable boxes on the Stage can noticeably slow down your Scratch programs on the Raspberry Pi. For that reason, if you're using a Raspberry Pi, untick the box beside the *score* variable. It would be ideal to display it *all* the time, but we can make the game faster and more fun by just showing it at the end of the game

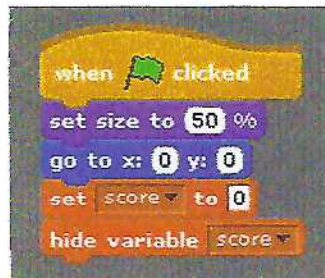
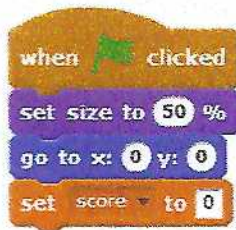
Preparing for the game start

Before you start bouncing balls and designing your dream ice cream, there are a few other things you need to prepare:

- 1 When the game starts, we need to position and size the cat correctly. Above the Blocks Palette, click the **Events** button in Scratch 2.0, or the **Control** button in Scratch 1.4. Drag the **when green flag clicked** block into the Scripts Area



- 2 Click the **Looks** button above the Blocks Palette and drag in the **set size to 100%** block. Change the number in it to 50. Shrinking the cat makes it easier to dodge the beachballs, which makes the game more playable
- 3 Click the **Motion** button. Drag in the **go to x:0 y:0** block. This ensures the cat starts in the middle of the Stage
- 4 Click the **Data** button (in Scratch 2.0) or the **Variables** button (in Scratch 1.4) above the Blocks Palette. Drag in the **set score to 0** block. This ensures the score starts at zero each time the game is played
- 5 Unless you're using a Raspberry Pi, your script is below (left). If you are using a Raspberry Pi, drag in the **hide variable score** block. Hiding the variable on the Stage during the game helps speed it up (below right)



Beware



You might think that some of these blocks are unnecessary because the cat's already in the middle and the score's already zero. But what happens when someone plays twice? The second time around, the score and cat position will carry on from the first game unless you reset them.

Using coordinates to move

Last chapter, you learned how to use directions to move a sprite under keyboard control. Spiral Rider used a loop to move the sprite continuously, and used the controls to change its direction.

For this game, we'll use a different technique. Each time a cursor key is pressed, the program will change the sprite's x or y coordinate slightly, making the sprite appear to move.

This is how to use this technique:

- To go right, change the x coordinate by a positive number
- To go left, change the x coordinate by a negative number
- To go up, change the y coordinate by a positive number
- To do down, change the y coordinate by a negative number

Follow these steps to add the player's movements:

- 1 Click the **Control** button above the Blocks Palette



- 2 Drag the **forever** block into your cat's script. Join it at the bottom of the script

- 3 Drag the **if** block into your script, and drop it inside your **forever** block



- 4 Click the **Sensing** button above the Blocks Palette

- 5 Drag the **key space pressed?** block into your script and drop it into the frame of your **if** block. Click the menu in the block to select the **right arrow** key

Don't forget



You can refer back to the grid reference chart in Chapter 2 if you can't visualize how this movement is working.

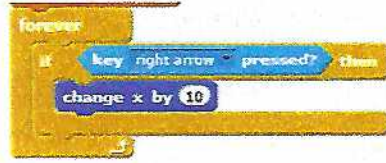
Beware



Last chapter, you used the "when space key pressed" hat block. That's unsuitable for games that need fast fingers, because when you hold down a key, there's a delay before it repeats. Using the Sensing block as we have in this project enables the player to hold down the key to dash across the Stage.

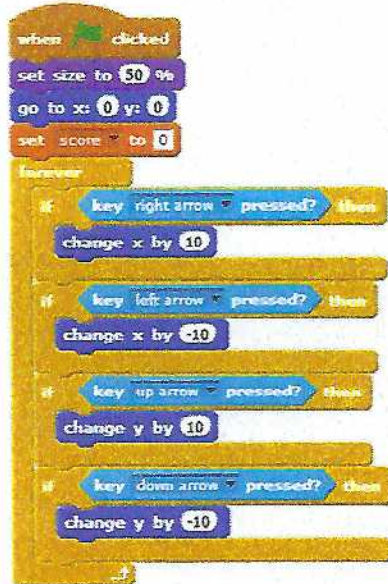
6 Click the **Motion** button above the Blocks Palette. Drag the **change x by 10** block into the bracket of your **if** block. This block makes the sprite move right and will be run if the right arrow key is pressed

7 Right-click the **if** block and click **duplicate** in the menu. A copy of your **if** bracket and all its contents follows the mouse pointer. Click underneath your **if** bracket to add a copy of it there



8 Click the Sensing block in the copy to change the key to the **left arrow** key. Change the number in the **change x by 10** block to -10

9 Repeat this process to add controls for moving up and down. Instead of using the **change x by 10** block, use the **change y by 10** block. Remember, use 10 to go up, and use -10 to go down



10 Test it by clicking the green flag and trying the cursor keys. You should see the cat move in four directions

Hot tip



When you duplicate a block, it also duplicates the blocks joined inside or underneath it.

Don't forget



To get rid of a block you no longer need in the Scripts Area, drag it into the Blocks Palette.

Hot tip



Scratch won't let sprites walk off the edge of the Stage.

Adding more images

You've already learned how to add sprites to your project (see Chapter 3) and how to change the background (see Chapter 1), so use those skills to add the following:

- 1 Add the Beachball sprite. You can find it in the Things category in Scratch 2.0 (where it's called Beachball) and the Things folder in Scratch 1.4 (where it's called beachball1)



- 2 Add another cat sprite. It's in the Animals category in Scratch 2.0 and the Animals folder in Scratch 1.4. In Scratch 2.0, it's called Cat1. If you're using Scratch 1.4, you can choose to use either cat1-a or cat1-b



- 3 Add the background beach malibu in Scratch 2.0 or beach-malibu in Scratch 1.4. It's filed in the Nature Theme (Scratch 2.0) or folder (Scratch 1.4)
- 4 Your Stage should look something like this, although it doesn't matter where the sprites are positioned for now

Beware



Don't use the Bouncy Ball in Scratch 1.4 for this project. It looks the same but includes some scripts already, which we don't need.



Renaming sprites

As we add more sprites, it's important that we can easily understand which is which. In Scratch 1.4, sprites are just called Sprite1, Sprite2, and so on. Scratch 2.0 uses more descriptive names, but we now have sprites called Sprite1 and Cat1 which look identical. That could quickly get confusing. You can give sprites a name that makes it easier to write and understand your programs.

Rename your original cat sprite from Sprite1 to Cat, the ball to Beachball (only necessary in Scratch 1.4), and the second cat to Energy.

Renaming sprites in Scratch 2.0

- 1 Click the sprite in the Sprite List. A small information button appears in the corner of the sprite. Click it
- 2 Click in the box where it says Sprite1, delete that text and replace it with the new name
- 3 Click the **Back** button to the left of the sprite. It's a blue blob with a white triangle in it



Renaming sprites in Scratch 1.4

- 1 Click the sprite in the Sprite List to select it
- 2 Above the Scripts Area is a representation of the sprite, with its name beside it. Click in the name box, delete that text and type your new sprite name



Hot tip



Where you rename the sprite, you can also see its position on the Stage, direction, and rotation style (see Chapter 2). In Scratch 2.0, you can also see its visibility (with a tick beside "show" indicating it's visible).

Don't forget



Short names are best: longer names are cut short in the Sprite List.

Beware



Make sure each sprite has a different name.

Hot tip

Click this script to run it, and you should see the ball jump to a different position and rotate to face a different direction each time.

Hot tip

Making the game unpredictable like this makes it more fun to play repeatedly than it would be if everything was always in the same place. It adds suspense too: nobody knows where the balls and ice cream will appear!

Don't forget

If you think back to the grid layout you saw in Chapter 1, the x coordinates run from -240 to 240, and the y coordinates run from -180 to 180. That's why we've used those numbers in our pick random blocks.

Making random numbers

One of the Operator blocks is used for making random numbers, and you can drag it into a number space in another block. Follow these steps to position the beachball in a random starting position, and facing a random direction:

- 1 Click the Beachball in the Sprite List, to make sure you're adding scripts to the right sprite. Click the **Scripts** tab to show the Scripts Area
- 2 Drag the **go to x:0 y:0** Motion block to the Scripts Area
- 3 Click the **Operators** button above the Blocks Palette. Drag the **pick random 1 to 10** block and drop it into the hole for the x coordinate. Do the same for the y coordinate

go to x: **pick random 1 to 10** y: **pick random 1 to 10**

- 4 Click the numbers in the first **pick random** block and change them to -240 and 240
- 5 Click the numbers in the second **pick random** block and change them to -180 and 180

go to x: **pick random -240 to 240** y: **pick random -180 to 180**

- 6 Use a similar process to make the ball point in a random direction too. Use the **point in direction** block, and pick a random number from -179 to 180

go to x: **pick random -240 to 240** y: **pick random -180 to 180**
point in direction **pick random -179 to 180**

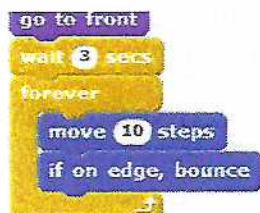
Moving the ball

The script to move the ball uses mostly blocks you've seen before, and just two new ones. Use your random positioning blocks as the starting point for the script to move the ball:

- 1 Add a **when green flag clicked** block above them
- 2 Click the **Looks** button above the Blocks Palette and drag in the **show** and **go to front** blocks. Join these underneath your random positioning blocks. The **go to front** block ensures that when the sprite hits the cat, the ball will be in front, so it looks more like a direct hit



- 3 Click the **Control** button above the Blocks Palette and drag in the **wait 1 secs** and **forever** blocks. Adjust the time delay to 3 seconds. The ball can appear anywhere, so the **wait** block gives the player time to move if the ball appears close to their starting position. The ball doesn't move or sap strength until after that grace period
- 4 Click the **Motion** button above the Blocks Palette and drag the **move 10 steps** and **if on edge, bounce** blocks into the **forever** block's bracket. The **if on edge, bounce** block changes the direction of the ball when it hits the edge of the Stage



Hot tip



In *Shop Cat* (see Chapter 10), the "if on edge, bounce" block is used to make cars move left and right across the Stage.

...cont'd

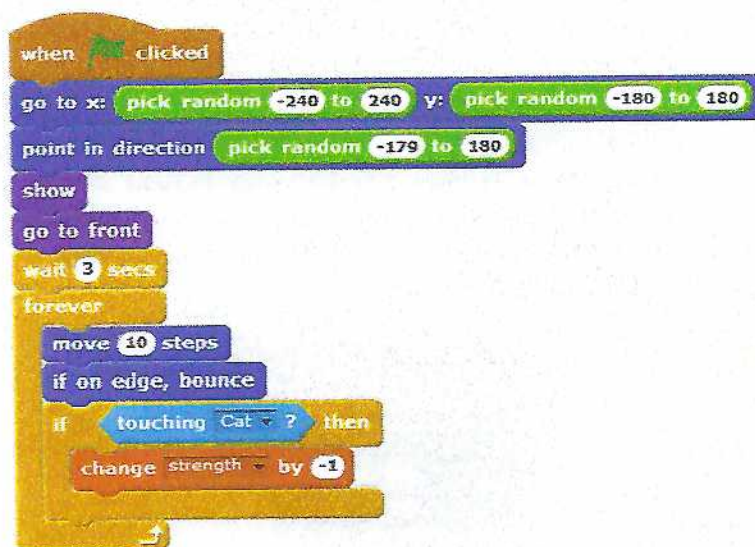
Hot tip



For testing purposes, click the Data or Variables button above the Blocks Palette and tick the box beside strength to show the strength variable on the Stage. You should see the number go down as the ball hits the cat. Untick the box in the Blocks Palette again to hide the strength.




- 5 Click the **Control** button above the Blocks Palette and drag in the **if** block
- 6 Click the **Sensing** button above the Blocks Palette and drag in the **touching?** block and set it to detect the Cat
- 7 Click the **Data** button (Scratch 2.0) or the **Variables** button (Scratch 1.4). Drag in the **change [variable name] by 1** block. Click the menu in the block to change the variable to the *strength*, and edit the value to -1
- 8 Click the green flag above the Stage and you should see the ball appear in a random location, and start moving in a random direction. You can move the cat, so why not see if you can dodge the ball?



Right: the completed script.

Copying and deleting sprites

Now we've got one ball bouncing around, let's make it a bit more challenging! You can copy (or 'duplicate') your beachball sprite, including its scripts, so that when you play the game, you have several balls all bouncing around randomly. I recommend three:

- 1 Above the Stage is a toolbar containing four buttons. In Scratch 2.0, they're in the dark stripe at the top of the screen (right, top). In Scratch 1.4, they're in a rounded panel (above)

- 2 Click the first icon, which looks like a stamp. Your mouse pointer turns into a stamp
- 3 Click the Beachball on the Stage. A new sprite is added to your project, with the same scripts as the one you've duplicated
- 4 You can instead right-click on a sprite in the Sprite List to open a menu with the option to duplicate the sprite



- 5 You might want to delete a sprite, perhaps because you made too many duplicates, or because you want to tidy up sprites you created to try out the Paint Editor (see later in this Chapter). Click the scissors icon in the toolbar (see Step 1) and then click the sprite you want to delete

Hot tip



Use the two icons with four arrows on them to resize sprites (the one on the left enlarges, and the other one shrinks). Click one and then click the sprite on the Stage repeatedly to change its size. In Scratch 2.0, the tool stops when you move your pointer off the sprite. To stop using the tool in Scratch 1.4, click an empty part of the Stage, Sprite List or Scripts Area.

Beware



In most programs, a scissors icon is for cutting something you want to reuse later. In Scratch, it's for deleting sprites completely. If you delete one by mistake, open the Edit menu at the top of the screen and click Undo.

Hot tip



Click the green flag and you should see several balls bouncing around.

Beware



Cloning is only available in Scratch 2.0. If you're using Scratch 1.4, you can't try this.

Don't forget



Duplicating sprites is when you make a copy of a sprite while you're writing the game. Cloning is when Scratch makes a copy of a sprite while the game is running.

Hot tip



For another example of cloning sprites in Scratch 2.0, see Going Batty in Chapter 9.

...cont'd

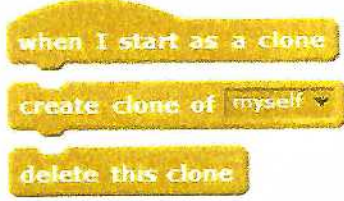
Cloning sprites (Scratch 2.0)

Scratch 2.0 introduced a new feature: the ability to clone sprites. This means that a sprite can make a copy of itself or another sprite while the game is running. This is a more elegant solution than duplicating sprites yourself, for four reasons:

- The program does the donkey work of making the copies, so it saves you time.
- The clones can be made at any time while the game is running, so you can respond to what the player does. You could have monsters or treasure multiplying, for example, depending on what the player does.
- There's just one set of scripts to edit when you want to make changes. With duplicated sprites, you would have to make changes to the scripts on all of them, if you wanted to change the speed of the balls, for example. With cloned sprites, there's just one sprite to change and the copies of that are made during the game.
- It's easier to understand how the program works, because there are no identical scripts in the program while you're editing it.

There are three main blocks that are used for managing clones:

- **create clone of myself:** This block creates a copy of the sprite. It also enables you to create a copy of another sprite by clicking the menu in the block and choosing another sprite.
- **when I start as a clone:** This hat block is used at the top of a script you would like to run when the clone is created.
- **delete this clone:** This block is used to delete a clone. No other block can be joined underneath it.



Let's look on the next page at how we can use cloning to make the game start easy and get gradually harder, as new balls appear during gameplay.

- 1 First, delete any duplicates you made of the Beachball sprite. Click the scissors icon in the dark stripe at the top of the screen, and then click the duplicate sprite on the Stage or in the Sprite List. Repeat until there's just one beachball left
- 2 Click the Beachball in the Sprite List. In the Scripts Area, remove the **when green flag clicked** hat block from your script. Click the second block in the stack (the **go to** block), and drag it away from the hat block. Keep the **when green flag clicked** block in the Scripts Area
- 3 Click the **Control** button above the Blocks Palette. Drag in the **when I start as a clone** hat block, and join this to the top of your script

```
when I start as a clone
go to x: pick random -240 to 240 y: pick random -180 to 180
point in direction pick random -179 to 180
show
```

- 4 Add the following blocks to the **when green flag clicked** block. You can find the **hide** block by clicking the **Looks** button above the Blocks Palette. The rest are Control blocks

```
when green flag clicked
hide
repeat 3
  create clone of myself
  wait 10 secs
```

The **hide** block is there because the original beachball is only used to make copies of itself. It's the clones that do the bouncing. When the green flag is clicked, the beachball makes three clones of itself, ten seconds apart. When each clone is created, it uses the same script you used previously to move to a random position, pause briefly and then start bouncing around.

Hot tip 

You can change the wait time to adjust how long there is before each ball appears, and you can increase the number of balls by changing the number of times the loop that creates them repeats. Cloning makes it simple to make changes like this, which would be quite fiddly if you were using duplicated sprites instead.

Hot tip

After you've added the energy meter, play the game to test it. The meter should shrink as the ball hits the player, and when it disappears completely, the game should end.

Don't forget

Scratch makes it easy to have several sprites doing different things at once. In this program, the energy meter is always adjusting its size to match the *strength* variable and checking whether to end the game. At the same time, the balls are bouncing and the player is moving. Each sprite can have a script that starts when the green flag is clicked.



Above: the energy meter's ghost effect means the background shines through.

Adding the energy meter

We're going to show the player how much energy they have left using an energy meter, which is a picture of the player's character (the cat), that changes its size as the player loses or gains energy:

- 1 Click the Energy sprite in the Sprite List to select it
- 2 Above the Blocks Palette, click the **Events** button in Scratch 2.0, or the **Control** button in Scratch 1.4. Drag the **when green flag clicked** block into the Scripts Area
- 3 In Scratch 2.0, click the **Data** button above the Blocks Palette. In Scratch 1.4, click the **Variables** button
- 4 Drag in the **set score to 0** block and join it to the **when green flag clicked** block. Change the variable in the block to *strength*, and the number to 100. This ensures the strength is set to its maximum value as the game starts
- 5 Click the **Motion** button above the Blocks Palette and drag in the **go to x:0 y:0** block. Change the numbers in it to x:-190 and y:120. This puts the energy meter in the top left corner of the screen
- 6 Click the **Looks** button above the Blocks Palette and drag in the **set color effect to 0** block. Change the effect to ghost, and the number to 50. This makes our energy meter semi-transparent, so it doesn't look like a character in the game (see Figure 1, facing page)
- 7 Click the **Control** button above the Blocks Palette. Drag in the **forever** block, and add an **if** block inside its bracket (see Figure 2, facing page)
- 8 The **if** block is used to check whether the player has run out of strength, which will be when the *strength* variable is less than 1. Click the **Operators** button above the Blocks Palette and drag the **<** block into the **if** block (see Figure 3). This checks whether the number on the left is less than the number on the right

9 Click the **Data** or **Variables** button above the Blocks Palette. Drag the *strength* variable block into the left of the Operator block. Type a 1 into the right of that block. This will check whether the strength is less than 1, and will do whatever is inside the **if** bracket if it is

10 Click the **Looks** button above the Blocks Palette and drag the **say Hello! for 2 secs** block into the **if** bracket. Change the text in it to Game Over

11 Click the **Control** button above the Blocks Palette and drag the **stop all** block into the **if** bracket. This will stop the game if the *strength* variable is less than 1

12 Finally, you need to make the energy meter change its size. Click the **Looks** button above the Blocks Palette and drag in the **set size to 100%** block. It goes inside the **forever** bracket, but outside the **if** bracket. Click **Data** or **Variables** above the Blocks Palette, and drag the *strength* variable block into its number space (see Figure 4, right)

Figure 1.



Figure 2.



Figure 3.

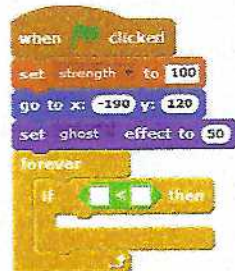


Figure 4.



Beware



If you just check whether *strength* is equal to zero, the game might never end. A ball could be sapping the player's energy, and could take the *strength* variable into negative numbers before the program checked whether the strength was zero and stopped the game.

Hot tip



If you're using a Raspberry Pi and you hid the score on the Stage to speed the game up, click the Variables button above the Blocks Palette and drag the "show variable score" block into your script, immediately above the "stop all" block.

Hot tip



You can use the Paint Editor to customize the sprites that come with Scratch too. Click your sprite in the Sprite List to select it, then click the Costumes tab above the Blocks Palette. In Scratch 2.0, that takes you straight into the Paint Editor. In Scratch 1.4, you open the Paint Editor by clicking the Edit button beside the costume you want to redesign.

Painting in Scratch

Scratch includes a Paint Editor you can use to design your own sprites and backgrounds, or to edit the existing ones. There isn't an ice cream sprite provided with Scratch, so you'll need to use the Paint Editor to make one for the Super Dodgeball game.

The layout of the tools is different in Scratch 2.0 and Scratch 1.4.

Painting in Scratch 2.0

To create a new sprite, click the **Paint new sprite** button above the Sprite List. It's the second of the four buttons.



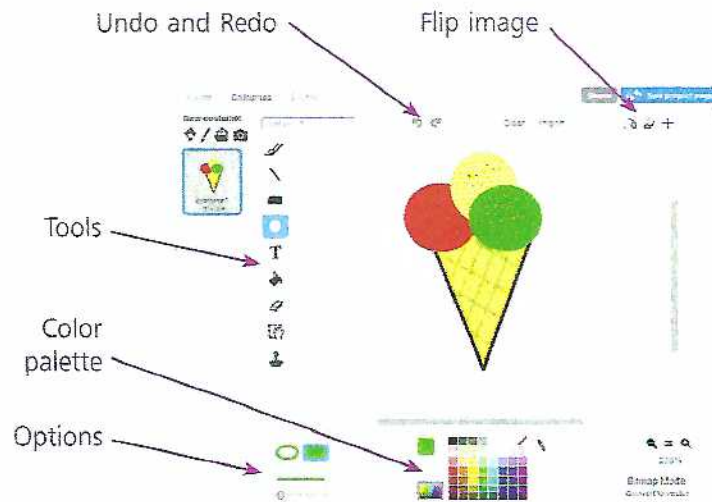
The Paint Editor opens on the right of the screen. You can increase the amount of space available for drawing by shrinking the size of the Stage. To do that, click **Edit** at the top of the screen and then click **Small stage layout**. When you've finished painting, repeat the process to return the Stage to its normal size.

The chequered area is the canvas, and the painting tools are arranged around the sides of it. Refer to the picture below as you learn about the tools on the following pages.

Beware



In Scratch 2.0, your changes are saved automatically. If you want to play around with the Paint Editor, start a new sprite, rather than editing one you will need later.



...cont'd

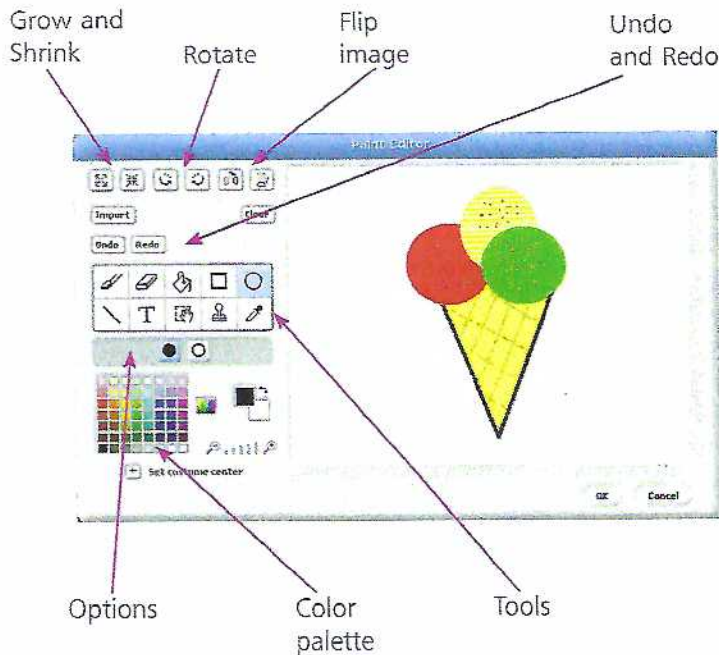
Painting in Scratch 1.4

In Scratch 1.4, you create a new sprite by clicking the **Paint new sprite** button above the Sprite List. It's the first of the three buttons.



The Paint Editor opens in a new window, with the controls down the left and the canvas you paint on on the right.

Refer to the picture below to orientate yourself as you read about the tools available on the following pages.



Hot tip



Each sprite can have a number of different pictures (or 'costumes'). You'll learn how to use these in Chapter 5.

Hot tip



In Scratch 1.4, you can use Ctrl + Z to undo. In Scratch 2.0, you have to click the Undo button.

Hot tip



The chequered pattern on the canvas represents transparency, where the background will show through your sprite.

Hot tip

Paintbrush is the tool that's selected when you first open the Paint Editor.

Hot tip

The button to set the costume center is important if you will be rotating your sprite. It's the plus icon in the top right in Scratch 2.0, and the button under the palette in Scratch 1.4.

Hot tip

If you hold down the Shift key while you're creating a rectangle, the tool creates a perfect square. To make a perfect circle, hold down the Shift key at the same time as drawing an ellipse.

Using Paint tools

To select one of the tools, just click its icon in the Paint Editor. I've shown icons for both versions of Scratch, where they differ:

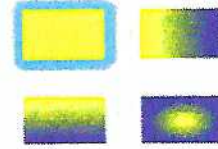
- **Paintbrush:** Simply hold down the mouse button as you move your pointer over the canvas, and you leave a line behind you. In the options, use the slider in Scratch 2.0 or the menu in Scratch 1.4 to change the thickness of the brush.
- **Line:** Click at the start of the line, hold the mouse button down, and then move the mouse to the end of the line. When you release the mouse button, a straight line is drawn between where you clicked and where you released the mouse button.
- **Rectangle:** Click to position one corner, hold the mouse button down, and then drag to the opposite corner and release the button. In the options, you can choose whether the shape is filled or hollow. Immediately after drawing the shape, you can use the handles on it to resize it in Scratch 2.0.
- **Ellipse:** This works in a similar way to the rectangle tool. Click, hold down the mouse button and drag the mouse pointer to create an ellipse. The ellipse expands to fill the space between where you click and where you release the mouse button.
- **Text in Scratch 2.0:** Click where you would like to place text on the canvas and then type what you'd like to write. Press Enter to start a new line. Press the mouse button when you've finished and a box appears around the text.



You can reposition the text by clicking and dragging inside the box, resize it by dragging one of the drag points on its sides and corners, and can rotate it by dragging the small circle above it. When you click outside the box, the editing ends. In Scratch 2.0, you can have multiple text boxes on a sprite, but you can't reposition them or edit them later (unless you're working in vector mode, coming right up!).



- **Text** in Scratch 1.4: To reposition your text, click and drag the box that appears in its top left corner. You can edit and move the text again later by selecting the Text tool again, but you can only have one piece of text per sprite. Use Enter to start a new line when you're typing your text. In the options area, you can change the font and text size.



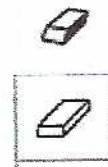
Above: Fades are available in the fill options.

Scratch

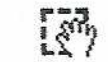
- **Fill:** This will fill a shape on your image with color. In the options area, you can choose a different pattern to use, with the foreground color fading into the background (see right).



- **Eraser:** This works like the Paintbrush, except that it deletes your image. Click once to delete just one spot, or click and drag to remove larger parts of the image. You can change the size of the eraser in the options area.



- **Select:** If you want to modify, move or delete a part of your image, use this tool to select it first. It can only select areas that are rectangular. Click in one corner of the area you'd like to select, drag the mouse pointer to the opposite corner, and release the mouse button. You can then click inside the selected area and drag it to move it, press the Delete key on the keyboard to remove it, and flip it upside down or reverse it horizontally. In Scratch 1.4, you can also rotate it, enlarge it or shrink it.



- **Stamp:** The stamp tool enables you to copy part of your image and then paste it somewhere else in it. You could use it to make a sprite's two eyes identical, for example. After selecting the tool, click and drag a rectangular box on the image to select part of it. In Scratch 2.0, click in the middle of the box and drag the copy to position it. Click outside it to drop it in place. In Scratch 1.4, move the mouse pointer and click to stamp the image in place. You can stamp the same image fragment multiple times (in Scratch 1.4 only).



Beware

In Scratch 2.0, you can only use the Stamp tool to paste one copy and then you have to start over, selecting the area you want to copy. After you've used it, the Select tool is chosen, so remember to choose the Stamp tool again if you want to use it again straight away.

Using colors

There is a palette of 56 colors you can click to choose a color for the art tools in the Paint Editor. The palettes are organized slightly differently in Scratch 2.0 (left, below) and Scratch 1.4 (right, below).

Hot tip



In Scratch 2.0, the slider beside the richer palette enables you to change the brightness of the palette. Click it and drag it down to fade the colors towards black.



The palette includes a transparent color, which enables you to erase parts of your drawing so the background shines through. In Scratch 2.0, it's represented by a red diagonal line through a white box, and is in the top right of the palette. In Scratch 1.4, it's the chequered box in the palette, the last box in the bottom row.

When you click the rainbow colored box beside the palette, the palette offers a much richer range of colors for you to pick from.



Beware



At the time of writing, the transparent ink can only be used with the paintbrush and fill tools in Scratch 2.0. In Scratch 1.4, you can use it with all the tools except Text.

The two overlapping boxes beside the palette are used to switch between choosing the foreground and background colors. These are used to choose the two colors in a fading pattern fill. Click the overlapping boxes to alternate between the foreground and background and then choose your color. Unless you're using a fading fill, you probably won't need to use this feature.

The pipette tool is on the right of the palette in Scratch 2.0, and on the toolbar in Scratch 1.4 (see icon on the right). When you click it, you can choose a color by clicking it on the canvas. This is the perfect tool if you need to precisely match a color you've already used in your drawing.



Using vectors in Scratch 2.0

Scratch 2.0 introduced support for vector images, which look better when they're resized, but are much more difficult to draw. Here's how to create them.

- Click the **Convert to Vector Mode** button at the bottom of the Paint Editor. The vector drawing tools are on the right.
- Click the pencil tool to draw on the canvas. Press the mouse button and drag the mouse pointer to make a line.
- The line, rectangle and ellipse tools enable you to draw in the same way as you do in the bitmap mode.
- You can fill rectangles and ellipses. To draw a shape with the pencil that you can fill, you need to join up your line's start and finishing points. You can fill the pencil line itself too.
- The select tool (with an arrow icon) enables you to edit shapes you added to the canvas previously. Click a shape and you can drag it to reposition it, or click and drag the handles on its sides and corners to resize it.
- With the reshape tool (see right), you can click the round control points that appear on a line (see below) and drag them to reshape the line.
- To copy a shape, click the stamp tool in the toolbar and then click the shape you'd like to copy.



Hot tip



Bitmap images store all the dots that make up the picture, and this image type is used in Scratch 1.4 and Scratch 2.0. A vector image stores instructions, explaining where the lines are and how to draw the image. Only Scratch 2.0 supports vector images.

Don't forget



Instead of using the paint tools to color in dots on the canvas to build up your picture, as you did in the Paint Editor previously, the vector tools create lines and shapes which you can edit later.

Beware



If you convert a vector image to a bitmap, the outlines become fuzzy and you'll lose the ability to edit the vector shapes. Don't do it, unless you're sure you know what you're doing!

Hot tip



If you don't want to design your own sprite now, you can use one of the sprites that come with Scratch to finish this game. They include the healthier food options of bananas, apples and oranges.

Hot tip

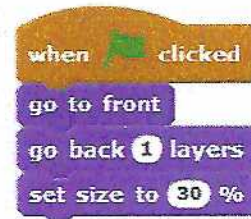


You can copy a block or set of blocks from one sprite to another. Drag them from the Scripts Area and drop them on the Sprite List, on top of the sprite you want to copy them to.

Making the ice cream appear

After all that hard work, you deserve an ice cream! You need to add two scripts to the ice cream sprite: this first one makes it appear in random locations for a short while and then disappear:

- 1 Drag in the **when green flag clicked** block. I'm sure you know where to find it by now! It's an Events block in Scratch 2.0, and a Control block in Scratch 1.4
- 2 Click the **Looks** button above the Blocks Palette. Drag in the **go to front** and **go back 1 layers** blocks. This means the ice cream will appear behind the beachballs, which looks most natural
- 3 It's easier to draw sprites large, so my ice cream sprite was huge. I used the **set size** block to reduce its size to 30%. You can use a similar block, and adjust the size to what you need
- 4 Click the **Control** button above the Blocks Palette and drag the **forever** block in to your script
- 5 Click the **Motion** button above the Blocks Palette and add the **go to x:0 y:0** block
- 6 Click the **Operators** button above the Blocks Palette and add two **pick random 1 to 10** Operator blocks in the number spaces in the **go to** block. Edit the numbers as you did for the beachball to position the ice cream in a random position on the Stage, where x is a random number between -240 and 240, and y is a random number between -180 and 180



- 7 Click the **Looks** button above the Blocks Palette and drag the **show** and **hide** blocks into your script

```
when clicked
  go to front
  go back 1 layers
  set size to 30 %
  forever
    go to x: pick random -240 to 240 y: pick random -180 to 0
    show
    hide
```

- 8 Click the **Control** button above the Blocks Palette and drag two **wait 1 secs** blocks into your script, after each of the **show** and **hide** blocks
- 9 Click the **Operators** button above the Blocks Palette and drag a **pick random 1 to 10** block into each of the wait blocks. Adjust the numbers in the **pick random** blocks, so the ice cream is shown for between 5 and 10 seconds, and hidden for between 1 and 3 seconds

```
when clicked
  go to front
  go back 1 layers
  set size to 30 %
  forever
    go to x: pick random -240 to 240 y: pick random -180 to 0
    show
    wait pick random 5 to 10 secs
    hide
    wait pick random 1 to 3 secs
```

Hot tip 

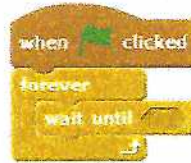
The ice cream in Super Dodgeball not only provides a scoring mechanism and a way to replenish strength. It also forces players to move around the screen, so they can't just find a safe place and stay there!

Enabling the player to score

The other script you need to add to your ice cream detects when the player reaches it, and then increases the score and strength:

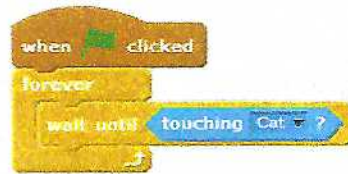
1 Drag in the **when green flag clicked** block

2 Click the **Control** button above the Blocks Palette and drag the **forever** block in to your script

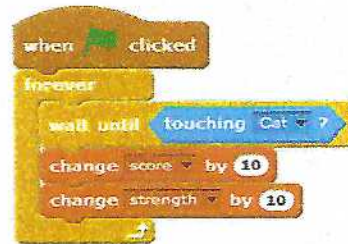


3 Add the **wait until** block into your script. It's another Control block

4 Click the **Sensing** button above the Blocks Palette. Drag the **touching?** block into the space in your **wait until** block, and then click the menu in it to choose the Cat. This block will pause this script until the cat is touching the ice cream



5 If the cat's touching the ice cream, we need to increase the *score* and the *strength* variables! Click the **Data or Variables** button above the Blocks Palette. Drag in two copies of the **change score by 1** block. Click the menu in one of the blocks to choose the *strength* variable, and change the number to 10 in both blocks



Don't forget



A sprite can have more than one script triggered by the green flag. We're using two on the ice cream.

- 6 At the moment, it's possible for the player's strength to be more than 100, if they collect an ice cream before being hit by a ball, for example. We need to make sure the strength doesn't go above 100. Click the **Control** button above the Blocks Palette and drag in the **if** block
- 7 Click the **Operators** button above the Blocks Palette, and drag the **>** block into the space in the **if** block
- 8 Click the **Data** or **Variables** button above the Blocks Palette. Drag the *strength* variable into the left of the **>** block, and type 100 in the right of it
- 9 Drag the **set score to 0** block into the bracket of the **if** block. Change the variable in it to *strength*, and the value in it to 100. That will ensure that if the *strength* variable is even higher than 100, it is instead set to be exactly 100
- 10 Finally, click the **Looks** button above the Blocks Palette and drag the **hide** block into the end of your **forever** loop. This will hide the ice cream if the cat touches it, making it look like the cat has picked it up

```
when green flag clicked
  forever loop
    wait until touching Cat
    change score by 10
    change strength by 10
    if strength > 100 then
      set strength to 100
    hide
```

Hot tip

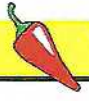


The **>** Operator block checks whether the number on the left is higher than the number on the right.

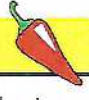
Hot tip



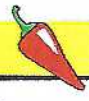
If you need to stop a variable going above or below a certain value in your own game, you can modify the blocks used here to do that.

Hot tip

Some of these ideas can be used in other Scratch games you make too.

Hot tip

You could stop the ice cream from replenishing the player's strength, if the game's too easy.

Hot tip

Don't be afraid to experiment and see what happens if you change numbers around, or add in your own effects. Click File and choose Save as a copy (in Scratch 2.0) or Save As (in Scratch 1.4), and your project will be saved with a new name, leaving the previous version untouched. If your game stops working, you can always go back to the previous version you made.

Tweaking the gameplay

If you find the game is too easy or too difficult, there are several things you can do to adjust the speed and difficulty of the game:

- To slow down the ball, change the number in its **move 10 steps** block to a lower number.
- To speed up the ball, change the number in its **move 10 steps** block to a higher number.
- To give players more time to react when a ball appears near them, change the number in the **wait 3 secs** block to a higher number.
- To make it easier to dodge the balls, make them smaller. You can do this by adding a **set size to 100%** block under the **when green flag clicked** block in their scripts. Change the number from 100% to something smaller. Remember that if you're using duplicated ball sprites (rather than cloned sprites in Scratch 2.0), you'll need to make this change for each one.
- To make it easier to catch the ice cream, make it bigger. Change the number in its **set size to 30% block** to a greater number.
- To give players more time to catch the ice cream, you can adjust the random numbers used to decide how long it's on screen. After its **show** block, change the random waiting time to a minimum of 10 seconds and a maximum of 20 seconds, to give players twice as long.
- To make the energy last twice as long, change how much energy the ball saps when it touches the cat from 1 to 0.5. Click the ball sprite and edit the **change strength by -1** block to **change strength by -0.5**. If you're using duplicated ball sprites, you'll need to change each ball sprite's script.

When you design games, you'll often find you need to adjust things like this to make the difficulty just right: a bit challenging, but not so difficult that it's not fun to play.

Professional game designers get other people to test their games, and watch them to see how easy or difficult they find them. Lots of people who use Scratch put their games on the Scratch website for others to try and give them feedback on too (see Chapter 11).