3 Spiral Rider

Learn how to make your first game, including using the green flag to start scripts, using variables to store a number, moving a sprite with the keyboard, and letting Scratch make simple decisions about which blocks should run.

Introducing Spiral Rider

Poor Freddy Fish has bought a new T-shirt and the tag in the back is really itching. Chris the Crab has claws that could clip the tag off, if only the pair could meet up. Chris has crawled through a maze of passages in the rock. Can you help Freddy get there?

Behind this silly premise lies a game that will use what you have already learned about Scratch and take it further to help you make your first real game. You'll learn:

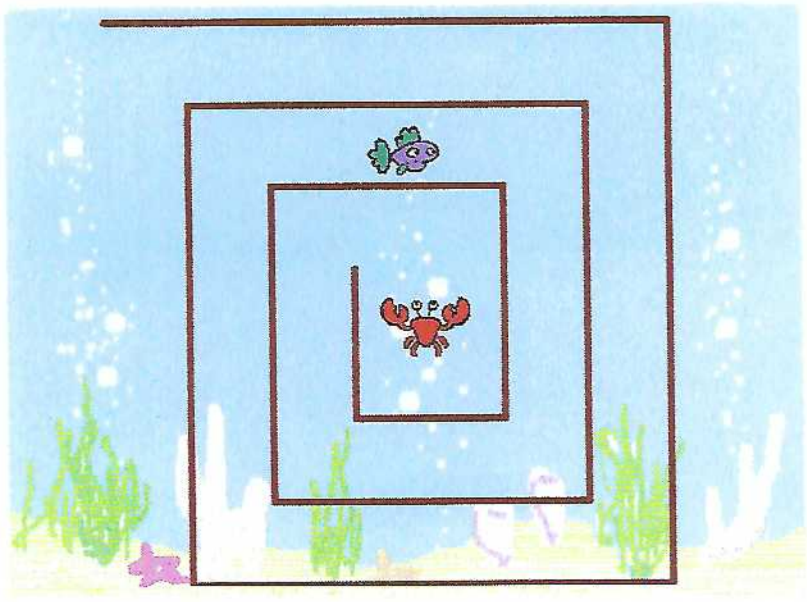
- How to keep track of a number, so you can make a square spiral with ever decreasing line lengths;
- How to use multiple sprites, and use the green flag to synchronize them;
- How to move sprites with the keyboard;
- How to make a sprite move automatically;
- How to detect when a sprite hits something; and
- How the graphic effects can be used on the background.

In this game, you control the fish using the cursor keys, and must turn at the right time on each corner to navigate the spiral successfully. If you hit the spiral, it's game over.

Hot tip

If you can't get the examples to work, or if you want to take a shortcut, you can download them. The examples for Scratch 1.4 can be downloaded at the author's website at **www.sean.co.uk** and the publisher's website at **www.ineasysteps.com/resources/downloads**

To find the examples for Scratch 2.0, visit the author's profile on the Scratch website. You can find it at **http://scratch.mit.edu/users/seanmcmanus/** or follow the link from **www.ineasysteps.com/resources/downloads**



Using the green flag

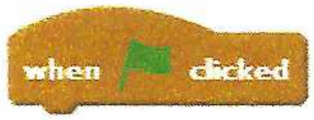
So far we've started our programs by just clicking the stack of blocks we want to run. There are a few problems with this:

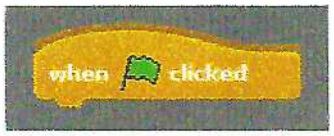
- When someone else uses the program, they might not know which stack of blocks they should run first.
- It's not user-friendly. Someone who wants to just play your game should be able to do so without having to look at its blocks.
- It's difficult to synchronize different sprites to start at the same time.

Scratch provides a simple solution to problems like this with the green flag, a button above the Stage used to start programs. Scripts on different sprites can detect when that button is clicked, and can then start at the same time.

People using Scratch programs know how to start programs using the green flag too. When you share programs through the Scratch website, they're shown with a prominent green flag button in the middle of the Stage, like a play button on a video.

There is a block you can use at the top of your stack of blocks to start them when the green flag is clicked:





It has a curved top because no other blocks can connect to the top of it: it is always the first block in its stack. Blocks like this are called hat blocks.

This block is categorized differently in the two versions of Scratch:

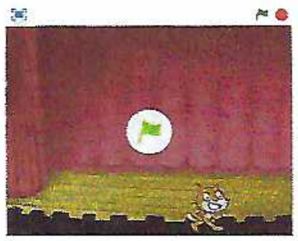
- In Scratch 2.0 it's a brown Events block. Events blocks are used for detecting when things happen, such as buttons being clicked. The Events category is new in Scratch 2.0.
- In Scratch 1.4, it's a yellow Control block. In Scratch 1.4, Events blocks and Control blocks are mixed in together.

Don't forget

This block has a different color in Scratch 2.0 and Scratch 1.4. In my screenshots, I'll use Scratch 2.0, so remember where to find this block if you're using a Raspberry Pi.

Hot tip

You can still click a stack of blocks to run them, even if there is a "when green flag clicked" block at the top of them, and this can be useful for testing scripts.



Above: When you share a program on the Scratch website, it has a big green flag button so users know how to start it.

Creating variables

Hot tip

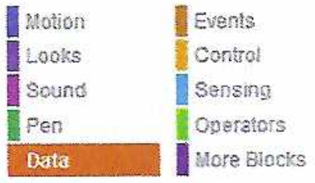
The pop-up box looks a bit different on Scratch 1.4, but includes the same form elements, except for the cloud variable option, which is new in Scratch 2.0.

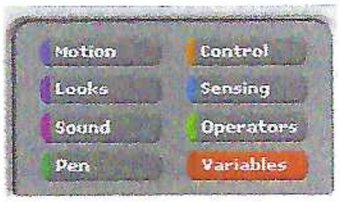
We're going to draw our spiral starting at the outside and working our way inwards. The way we draw a spiral is similar to how we draw a square: we draw a line, turn 90 degrees and repeat until we reach the middle. The big difference is that the lines get shorter as we move towards the middle.

That means we need a way to remember the length of the line, and to shorten it. Variables are used in Scratch (and other programming languages) to store information we want to remember, so we can reuse it and change it. You could use a variable to keep track of a score, the number of lives left, the player's name, or the right answer in a word guessing game.

Start a new project, and click on the cat sprite, which we will use to draw our spiral. Let's make a variable to store the length of the lines in the spiral:

1. In Scratch 2.0 (left, below), the blocks for variables are categorized as Data blocks, so click **Data** at the top of the Blocks Palette. In Scratch 1.4 (right, below), the blocks are categorized as Variables blocks, so click **Variables** at the top of the Blocks Palette

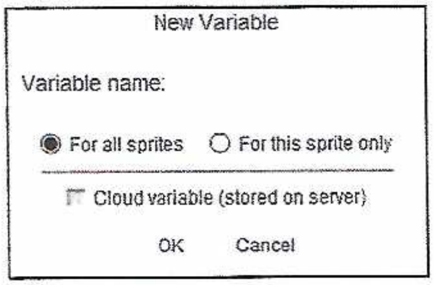




Don't forget

They're called variables because the information they store (their values) can vary. Their name stays the same though. The number in the score variable might change, for example, but the name of the variable will stay *score*.

2. Click the **Make a Variable** button in the Blocks Palette. A pop-up box appears, like this



New Variable

Variable name:

● For all sprites ○ For this sprite only

☐ Cloud variable (stored on server)

OK Cancel

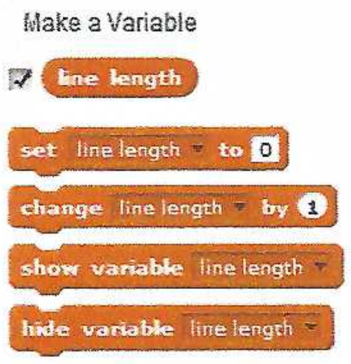
3 The first thing we need to do is name our variable. You can use spaces in Scratch variable names. Type **line length** into the box, but don't press Enter yet

4 When you make a variable, you can choose whether it can be used by all sprites or just the current sprite. You can't change this later, so pause for thought here. We'll make this variable for this sprite only, because other sprites don't need it and shouldn't use it

5 In Scratch 2.0, you can make a cloud variable. This means the variable is shared between everyone using your program on the Scratch website. If 10 people play your game, for example, they could all see (and change) the same high score number if it's in a cloud variable. Leave the Cloud variable box unticked. (For more on cloud variables, see Chapter 8)

6 Click the **OK** button

7 The Blocks Palette now contains some new blocks for managing your variable, shown below. The block with rounded ends has a tickbox beside it. When ticked, it shows your variable and its value on the Stage. We don't need that for this variable, so untick it. The blocks to set and change the variable values have menus in them that you can use to choose which variable you'd like to change, and a box where you can type a number or some text



Hot tip

If a variable is set up for just one sprite, other sprites can't use its value or change its value. That stops them causing errors by interfering with a variable they shouldn't, so it's considered a good idea where possible.

Don't forget

Keep variable names short but meaningful so you can easily understand your program. A good variable name is something like *alien speed*.

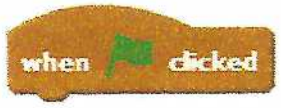
Beware

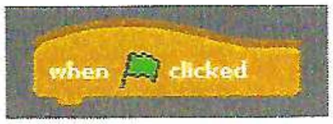
Two sprites can have variables with the same name that work completely independently. This happens, for example, if you give a sprite a variable for that sprite only, and then duplicate that sprite. See duplication in Chapter 4.

Drawing a spiral

We'll use our new variable to draw a spiral on the Stage. Follow these steps to add the blocks needed to your cat sprite:

1. 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 to start your script

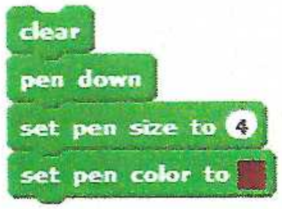




2. It looks best if our spiral seems to magically appear from nowhere, so let's make the cat invisible. Click the **Looks** button above the Blocks Palette and add the **hide** block to your stack

3. To put our cat in its starting position (the top left of the screen), click the **Motion** button above the Blocks Palette and drag the **go to x:0 y:0** and **point in direction 90** blocks into your stack. Change the values in the **go to** block to x: -180 and y: 170 by clicking and editing the numbers

4. Click **Pen** above the Blocks Palette, and drag in these blocks. This clears the screen of any drawing that's already there, and gets the pen ready to draw this time



```
clear
pen down
set pen size to 4
set pen color to
```

5. We need to set the starting value of our *line length* variable. I've chosen 340, because the height of the Stage is 360, so a maximum spiral height of 340 fits comfortably. Click the **Data** button above the Blocks Palette in Scratch 2.0, or the **Variables** button in Scratch 1.4. Drag the block **set line length to 0** into your script. Then click the box in the block and edit the number 0 to 340

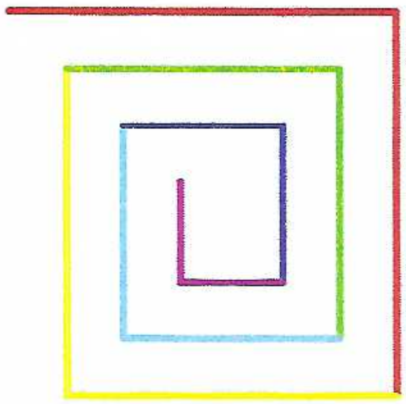
Hot tip

The "go to x:0 y:0" block might have different numbers in it in the Blocks Palette. That's okay, because you can always change them.

Don't forget

The "when green flag clicked" block is a brown Events block in Scratch 2.0, and a yellow Control block in Scratch 1.4.

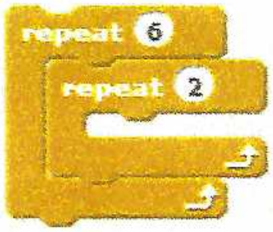
6. Our spiral uses two loops, one inside the other. The inner loop draws a line, turns 90 degrees and then does the same again, so it repeats its blocks twice in total. The line length is then shortened, and the process repeats, six times in total. In the picture below, I've changed the pen color each time the line length is shortened. Lines of the same color are drawn by the same inner loop



Hot tip

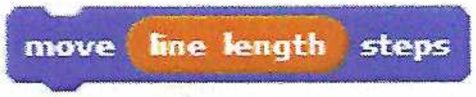
You can see the complete program for drawing a spiral on the next page.

7. Click the **Control** button above the Blocks Palette and drag the **repeat 10** block into your stack. Drag another **repeat 10** block in, and set the outer loop to repeat six times, and the inner loop to repeat twice



8. Click the **Motion** button above the Blocks Palette, and drag in **move 10 steps** and **turn clockwise 15 degrees**. Change the number in the **turn** block to 90 degrees

9 You can use a variable in place of a number. Instead of moving 10 steps, or any other fixed number, we want to move the same number as we've stored in our *line length* variable. To do this, we can replace the number in the **move** block with the variable name. Click the **Data** or **Variables** button above the Blocks Palette, and drag the rounded block containing the *line length* variable name into the **move 10 steps** block



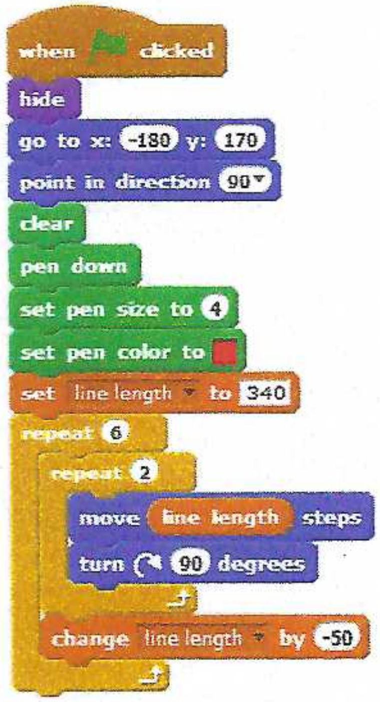
Hot tip

The shapes of the blocks give you a hint how they can be used. The "move 10 steps" block has a rounded hole, and the block for the *line length* variable has rounded edges, so you can easily see that one fits inside the other.

10 After the inner loop ends, and an L shape has been drawn, we want to shorten the length of the lines. Drag in the Data or Variables block **change line length by 1**. Make sure it joins underneath the inner **repeat** block, and doesn't go inside its bracket. Change the number in it to -50

11 Click the green flag button above the Stage to see your spiral appear!

You can see our finished script to the right

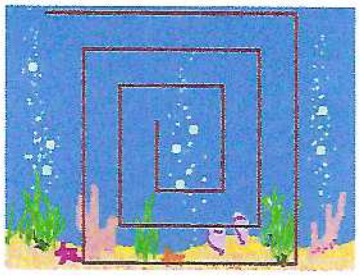


```
when [flag] clicked
hide
go to x: -180 y: 170
point in direction 90
clear
pen down
set pen size to 4
set pen color to 🟥
set line length to 340
repeat 6
    repeat 2
        move line length steps
        turn ↻ 90 degrees
    change line length by -50
```

Changing the background

We have our spiral, but we can make our game look much more interesting by adding in an underwater background:

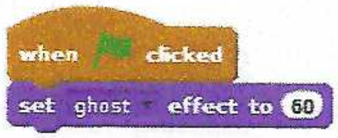
1. Add one of the underwater backgrounds to your project. Scratch 2.0 has three to choose from in its Underwater theme. Scratch 1.4 has just the one, in the Nature folder



2. The background I've chosen is quite vibrant, and makes it hard to see what's going on in the game. We can tone it down using a graphic effect. To do that, we will add blocks to the Stage. Click the Stage icon beside the Sprite List. Open the Scripts Area by clicking the **Scripts** tab above the Blocks Palette in Scratch 2.0 or above the Scripts Area in Scratch 1.4

3. Click the **Events** button (Scratch 2.0) or **Control** button (Scratch 1.4) above the Blocks Palette. Drag the **when green flag clicked** block into the Scripts Area

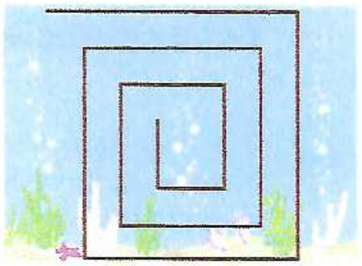
4. Click the **Looks** button above the Blocks Palette. Drag in the block **set color effect to 0** and join it to **when green flag clicked**



```
when [flag] clicked
set ghost effect to 60
```

5. In the **set effect** block, click the menu to choose the ghost effect. Change the number in the block to 60

6. When you click the green flag, the background lightens (see right)



Hot tip

The ghost graphic effect uses a number from 0 to 100, which is a percentage. When the ghost effect on the background is at 100%, the background is invisible.

Hot tip

You can also apply the graphic effects to sprites, as you'll see later on. The ghost effect makes sprites transparent.

Adding sprites

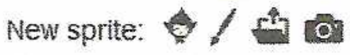
Our game will use three sprites: the invisible cat sprite, which draws the spiral, Freddy the Fish, and Chris the Crab. The way you add sprites is different in Scratch 2.0 and Scratch 1.4.

Adding sprites in Scratch 2.0

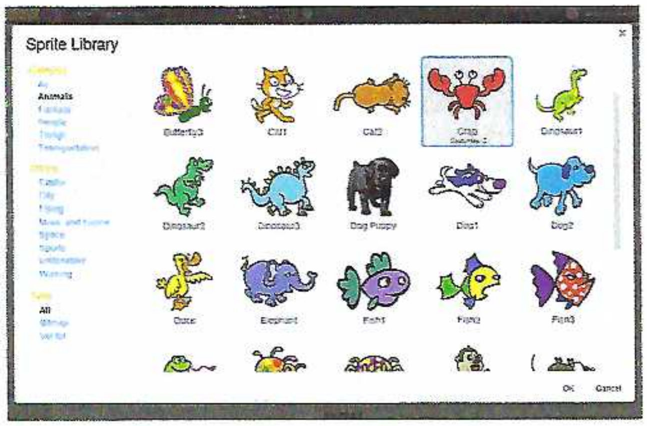
Hot tip

There are two types of sprites in Scratch 2.0: vectors and bitmaps. Vectors look better when you change their size. When you have a choice, pick a vector over a bitmap.

1. Between the Sprite List and the Stage is a row of buttons for making new sprites. You can choose a sprite from the library, paint a new sprite (see Chapter 4), upload a sprite from a file if you have a picture you've already made, or use a webcam to take a photo to use as a sprite. Click to choose a sprite from the library



2. The Sprite Library opens, and looks similar to the library of backgrounds you used previously. You can click categories and themes on the left to browse the different sprites available. Click the Animals category, click the Crab and then click **OK**



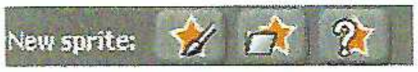
Hot tip

You can also double click a sprite in the Sprite Library to open it.

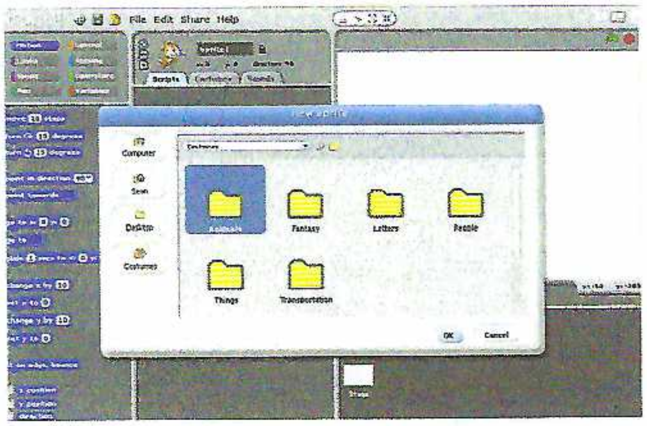
3. Repeat the process, but this time select the sprite Fish1, which is also in the Animals category. Use the scrollbar on the right of the Sprite Library to see more sprites

Adding sprites in Scratch 1.4

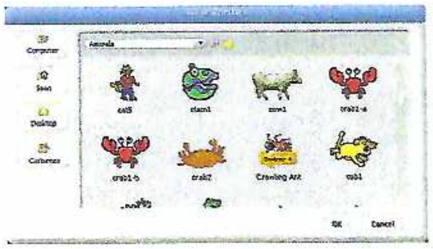
1. Above the Sprite List are three buttons you can use to add a new sprite. From the left, they enable you to paint a new sprite (see Chapter 4), choose a new sprite from a file, or get a surprise sprite. Click the middle button to get a sprite from a file



2. The file browser opens, ready for you to browse the sprites supplied



3. Double click on the Animals folder and use the scrollbar to find crab1-a. Click it and then click **OK**



4. Repeat the process, but this time select the sprite fish2, which is also in the Animals category

Hot tip

In the file browser, you can use the buttons on the left to go to your computer or Desktop to find your own pictures to use as sprites in Scratch.

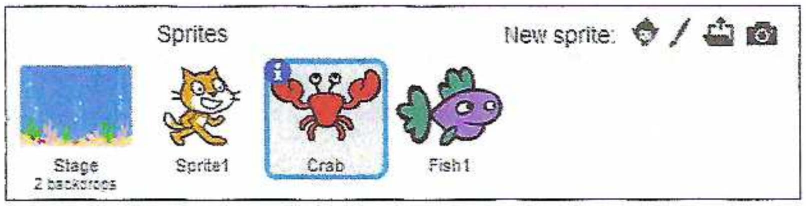
Hot tip

The "surprise sprite" button can give you some great ideas for games you could make, by bringing together all kinds of different people, creatures and things!

Animating the crab

When the game starts, we need to position the crab in the middle of the spiral and set its size. To make it look more lifelike, we can make it bob from side to side during the game. Follow these steps to create the script for the crab:

1. Click the crab in the Sprite List to make sure you're adding your script to the right sprite



2. Click the **Events** button (Scratch 2.0) or **Control** button (Scratch 1.4) above the Blocks Palette. Drag the **when green flag clicked** block into the Scripts Area to start your script

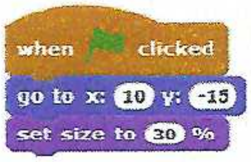
3. Click the **Motion** button above the Blocks Palette and drag the **go to x:0 y:0** block

4. Change the values in the **go to** block to x: 10 and y: -15 by clicking and editing the numbers

5. Click the **Looks** button above the Blocks Palette and drag the **set size to 100%** block

6. Click the number in the block and change it to 30. The choice of 30% wasn't scientific: I arrived at it through trial and error, after trying several different numbers. I wanted the crab to be as big as possible, but still fit in the middle of the spiral

7. Check your script so far (right)



```
when [flag] clicked
go to x: 10 y: -15
set size to 30 %
```

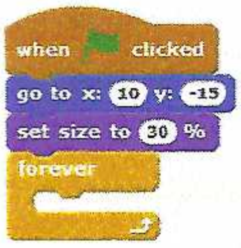
Hot tip

You could position the crab by just dragging it on the Stage, but it's better to make sure everything is in its right place every time your program runs.

Beware

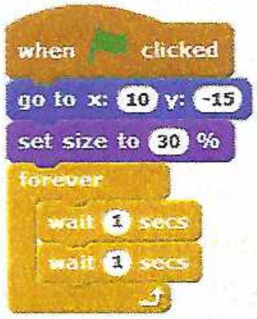
When you have more than one sprite in a program take care to make sure you're adding your scripts to the right sprite.

8. Click the **Control** button above the Blocks Palette and drag the **forever** block into your stack. We'll use this to make our crab hop left and right all the time the game is playing



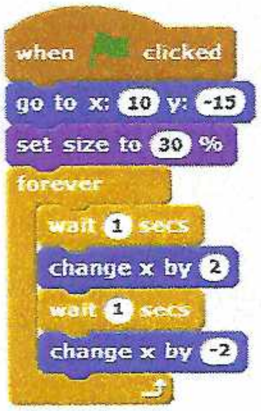
```
when [flag] clicked
go to x: 10 y: -15
set size to 30 %
forever

```

9. Drag two **wait 1 secs** blocks into the bracket of the **forever** block, so they're joined to each other



```
when [flag] clicked
go to x: 10 y: -15
set size to 30 %
forever
    wait 1 secs
    wait 1 secs
```

10. Click the **Motion** button above the Blocks Palette, and drag in two **change x by 10** blocks. Position one after each of the **wait 1 secs** blocks. Click the number 10 in the first **change x by** block and change it to 2. Change the number in the other one to -2



```
when [flag] clicked
go to x: 10 y: -15
set size to 30 %
forever
    wait 1 secs
    change x by 2
    wait 1 secs
    change x by -2
```

11. Test it works by clicking the green flag button above the Stage. You can click and drag the fish out of the way

Hot tip

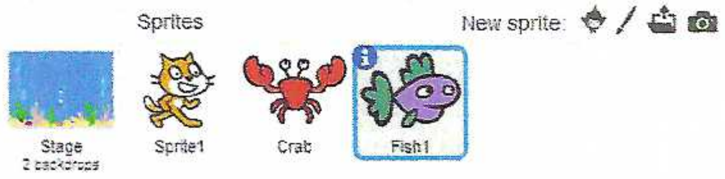
You don't have to add blocks in the order they will be used by the program. We added two "wait" blocks in one go, and then two "change x" blocks in one go. That saved time switching between the categories in the Blocks Palette.

Enabling keyboard control

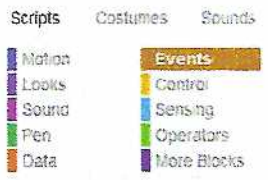
There are a couple of different techniques you can use to make a sprite move in a game, as you'll see in other projects in this book. For this game, we'll make the fish swim all the time, so players have to steer it to stop it hitting the spiral. When a player presses one of the cursor keys, we'll change Freddy's direction.

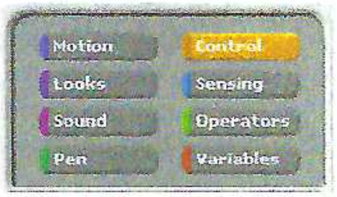
There is a block you can use to start a script when a particular key is pressed on the keyboard. In Scratch 2.0, this is an Events block and in Scratch 1.4, it is a Control block:

1. Click the fish in the Sprite List to make sure you're adding your script to the right sprite

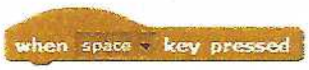


2. In Scratch 2.0 (left, below), click the **Events** button above the Blocks Palette. In Scratch 1.4 (right, below), click the **Control** button above the Blocks Palette





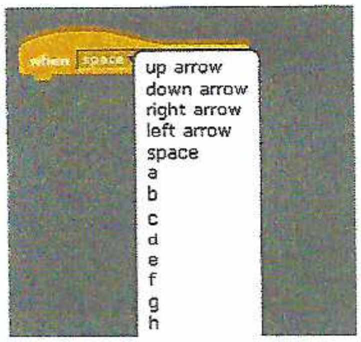
3. Drag the **when space key pressed** block into the Scripts Area. Like the **when green flag clicked** block, this is a hat block and has a curved top because nothing can go above it. This block always starts off the stack of blocks it is connected to



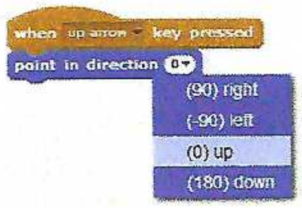
Don't forget

The "when space key pressed" block is a brown Events block in Scratch 2.0, but it's a yellow Control block in Scratch 1.4.

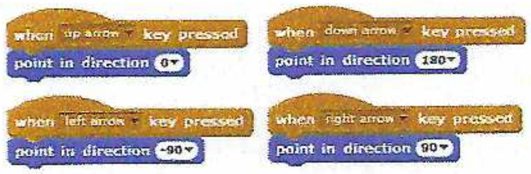
4 Click the menu in this block, and you can choose which key you want to detect. Click **up arrow**



5 Click the **Motion** button above the Blocks Palette. Drag in the **point in direction 90** block, and join it to your **when up arrow key pressed** block. Click the direction menu to open it and then click **(0) up**



6 Repeat these steps to detect the down, left and right arrow keys, and to change the direction of the fish when they're pressed. It's okay to have several different and unconnected scripts for a sprite. Here's what the finished movement key detection scripts should look like


when up arrow key pressed
point in direction 0
when down arrow key pressed
point in direction 180
when left arrow key pressed
point in direction -90
when right arrow key pressed
point in direction 90

7 Press the cursor keys to check the fish changes direction. Note that it won't actually move around the screen yet

Hot tip

To save time switching between different parts of the Blocks Palette, you could drag in four "when space key pressed" blocks, drag in four "point in direction" blocks, and then organize them in the Scripts Area.

Making the fish move

Our script for the fish puts it in the starting position (the bottom left of the screen), points it in the right direction (up), and adjusts its size so it can fit in the spiral. The script then keeps it moving until it hits either the spiral or the crab. If it hits the spiral, we display a message that says "Ouch". Otherwise, we display a happy message because the player has reached the crab.

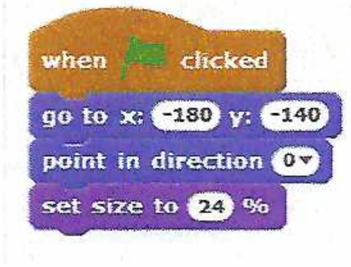
Here's how you make the script for the fish:

Hot tip

The optimum size of the fish was discovered by experimentation. I played the game a few times to work out a size that fitted in the spiral, but was still big enough to make it a challenge to dodge the walls.

1. Click the fish in the Sprite List to make sure you're adding your scripts to the right sprite

2. The first few blocks will be familiar to you by now, so drag them into the Scripts Area and change the values in them. The fish should start at x:-180 y:-140, pointing in direction 0 (up), and with a size of 24%

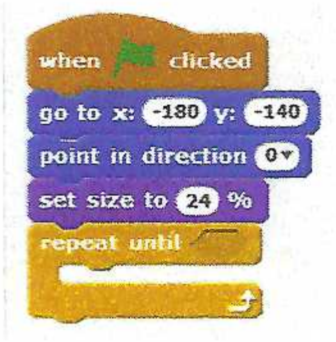


```
when [flag] clicked
go to x: -180 y: -140
point in direction 0
set size to 24 %
```

Don't forget

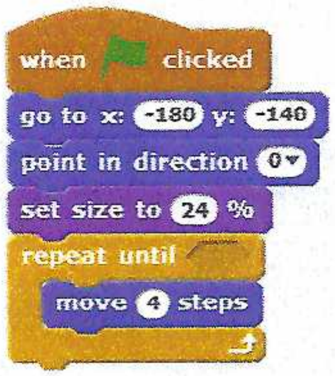
The color of the blocks usually gives you a clue where to find them in the Blocks Palette. The "when green flag clicked" block is a brown Events block in Scratch 2.0 and a yellow Control block in Scratch 1.4, though.

3. To keep the fish moving until it hits something, you use the **repeat until** block. It's a Control block. Drag it in and join it to your script



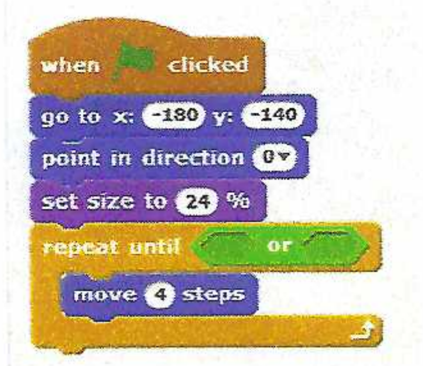
```
when [flag] clicked
go to x: -180 y: -140
point in direction 0
set size to 24 %
repeat until
```

4 Inside the bracket of the **repeat until** block, add a **move 10 steps** block, and change the number in it to 4. This is all you need inside the **repeat until** bracket. Remember you've added four other scripts to change the fish's direction when a key is pressed



```
when [flag] clicked
go to x: -180 y: -140
point in direction 0▾
set size to 24 %
repeat until < >
    move 4 steps
```

5 The **repeat until** block has a diamond-shaped hole in it. This is where you tell Scratch that you want it to repeat the loop until the fish hits the spiral or the crab. Click the **Operators** button above the Blocks Palette, and drag the **or** block into the diamond shaped hole



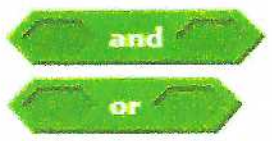
```
when [flag] clicked
go to x: -180 y: -140
point in direction 0▾
set size to 24 %
repeat until < < > or < > >
    move 4 steps
```

Hot tip

To speed the game up, change the number 4 in the "move 4 steps" block to a higher number.

Hot tip

The "or" Operator block is used because we want to move the fish until it hits the crab *or* the spiral. There is another Operator block you can use to check whether two things are both true. If we had used the "and" block instead of the "or" block, the fish would move until it was touching both the spiral and the crab.

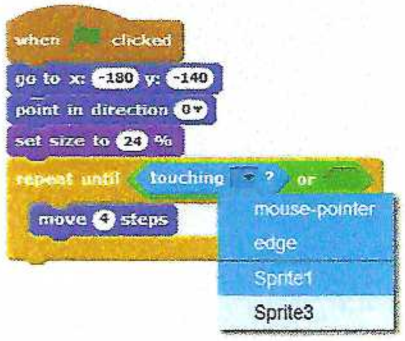


```
< > and < >
< > or < >
```

Hot tip

Glance at the Sprite List if you're not sure which sprite is which: you can see the sprite's picture and name there.

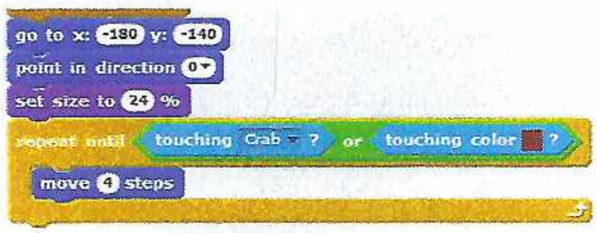
6 To see whether the fish has hit the spiral or the crab, you use two Sensing blocks. Click the **Sensing** button above the Scripts Area. Drag the **touching?** block and drop it onto the left of the green block. The **touching?** block is used for checking whether a sprite is touching another object, such as another sprite or the mouse pointer. Click the menu in it, and select Crab (in Scratch 2.0) or Sprite2 in Scratch 1.4. Both refer to the crab, but the two versions of Scratch use different names for the sprites you added


```
when [flag] clicked
go to x: -180 y: -140
point in direction 0
set size to 24 %
repeat until < touching [ ] ? > or < >
  move 4 steps
```

Beware

Dropping the Sensing block on top of the Operator block is tricky. Drag it, but don't release the mouse button until just one of the green holes in the Operator block is glowing. If the whole green block is glowing and you release the mouse button, your Sensing block will replace the Operator block instead of combining with it.

7 Drag the **touching color?** block into the other side of the **or** block. Click the square of color inside your **touching color?** block, and then click your spiral on the Stage. This will put the color of your spiral into the block


```
go to x: -180 y: -140
point in direction 0
set size to 24 %
repeat until < touching Crab ? > or < touching color [ ] ? >
  move 4 steps
```

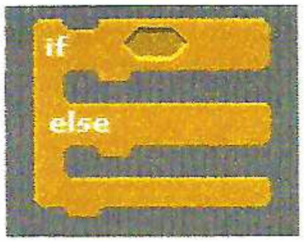
8 As always, test it works! Click the green flag button above the Stage. You should now find you can move the fish around and it'll keep moving until it hits either the spiral or the crab

Adding Game Over messages

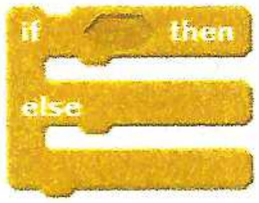
There are two ways the game can end: the fish is touching the crab (which means the player won), or the fish is touching the color of the spiral, which means they crashed. We can use an **if... then... else** block to display a different message, depending on how the game ended:

Beware

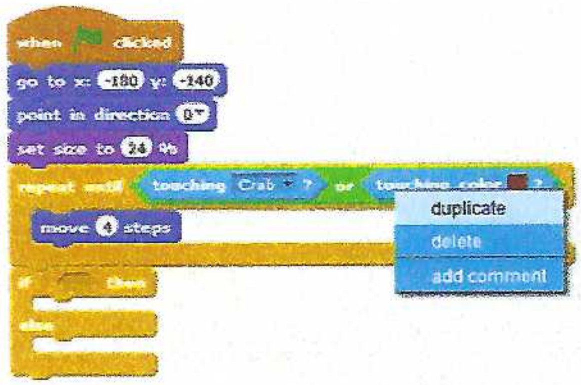
The "if... then... else" block is just called "if... else" in Scratch 1.4. It does exactly the same job, but it doesn't have the word "then" on it.



1. The **if... then... else** block is a Control block. Drag it into the Scripts Area and join it to the bottom of your **repeat until** block



2. Right-click the **touching color?** block you used in your **repeat until** block. A menu opens. Choose **duplicate**



3. As you move the mouse, a copy of the **touching color?** block will move with it. Click the mouse button to drop the block into the diamond-shaped hole of the **if... then... else** block

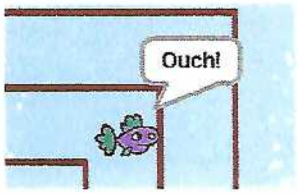
Hot tip

To understand how the "if" block works here, try reading the program aloud. Use the word "otherwise" instead of "else". The meaning becomes clear: If touching color red, then say "Ouch" for 2 seconds, otherwise say "You won!" for 2 seconds.

4 Click the **Looks** button above the Blocks Palette. Drag a **say Hello! for 2 secs** block into both of the brackets in your **if... then... else** block

```
say Hello! for 2 secs
```

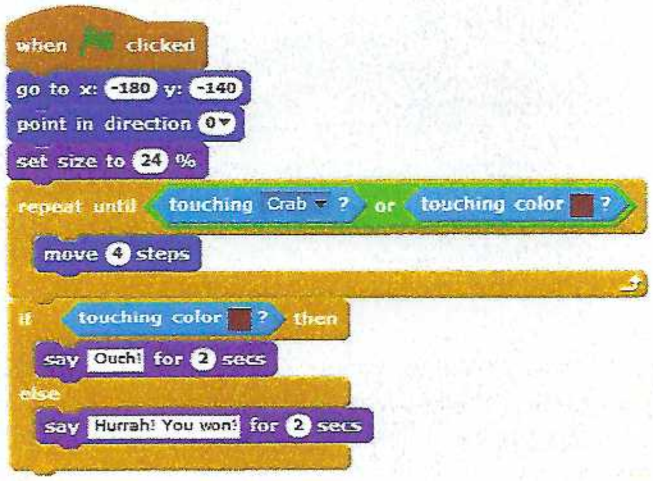
5 This **say** block displays a message in a speech bubble. After the time specified (2 seconds is the standard), it disappears again. Click the "Hello!" text in the first **say** block, and change it to "Ouch!". Change the text in the second **say** block to "Hurrah! You won!". The way the **if... then... else** block works is that it checks whether something is true, and if it is, it runs the blocks in its first bracket, and otherwise it runs the blocks in its second bracket. We're checking whether the fish has hit the spiral, so if it has, the first **say** block will display the message for losing the game. If the fish isn't touching the spiral, it must be touching the crab, so the program shows the congratulations message



Hot tip

The game is surprisingly tricky, because it's easy to catch the fish's tail when it turns the corner. It looks simple, but it might take you a few goes to complete it!

6 Your final script looks like this



```
when [flag] clicked
go to x: -180 y: -140
point in direction 0
set size to 24 %
repeat until <touching Crab ? or touching color ■ ?>
    move 4 steps
if <touching color ■ ?> then
    say Ouch! for 2 secs
else
    say Hurrah! You won! for 2 secs
```

Hot tip

You can customize this game by designing different shapes for the fish to navigate through, and by changing the images.