SCRATCH

From Scratch

for Schools and Clubs

by Seamus O'Neill
& Stephen Howell

SCRATCH
FROM SCRATCH

for
Schools and Clubs

by
Seamus O'Neill
and
Stephen Howell

Design by We and Us Ltd. Educational Design Services

Introduction

Welcome to *Scratch from Scratch*! It's always exciting to begin a new book and learn exciting new skills, and this book is really special. *Scratch from Scratch* will teach you how to create something out of nothing, to write interactive stories and games using only your imagination, logical thinking skills and creativity. This book will teach you to *program*.

What is commonly called *computer programming*, or *software development*, or simply just *coding* has long been stereotyped as an activity that only 'nerds' or 'geeks' could learn and enjoy. Recently this perception of computer programming has changed quite a bit. In some countries, schools are teaching computer programming from early ages. Industry led groups like the TEALS program and Computer Clubhouse provide after school computer programming clubs. Community led schemes such as the CoderDojo movement see parents and children give up their Saturdays to learn how to program. This is a welcome change. Computer programming has become mainstream, and boys and girls who want to learn this exciting skill should be afforded many opportunities to do so.

Unfortunately, there is no one established and accepted syllabus for a young programmer starting off, or a curriculum to guide a teacher presenting software development for the first time. Luckily, an influx of fantastic introductory tools have arrived on the market, including Scratch, that make it possible to teach and learn programming in a fun, safe environment.

Experts will always have different (and evolving!) opinions of the best way to inculcate particular software development skills, but perhaps the goal of teaching programming shouldn't be to learn a particular language but to comprehend and adopt *Computational Thinking*. Proposed by Professor Jeannette Wing of Carnegie Mellon University in Pittsburgh, Computational Thinking is a different way of thinking about computers and computer programming. It allows curiosity and creativity inform our learning and understanding of computing concepts.

continued

Scratch from Scratch is the perfect way to start learning an exciting programming language aimed at younger learners. We've written the book to be accessible to young and old, but primarily with the teacher or mentor in mind. As this book is an evaluation edition of *Scratch from Scratch*, it is a limited run and just the first part of a planned three-part text. This edition covers the *Tools and Skills of Scratch*. Our next edition will comprise all three parts; *The Tools and Skills of Scratch*, *Numeracy in Scratch* & *Computational Thinking in Scratch*. We hope that teachers, parents and mentors of children will use this book to encourage younger learners to try computer programming. We've seen programmers as young as eight embrace Scratch and develop a love of programming.

We really hope you enjoy this text, and use it in your computer club or classroom. We welcome and value feedback, and you can get in touch with us through our contact email addresses: stephen.r.howell@gmail.com and seamus@weandus.ie.

We hope you enjoy learning Scratch!

Stephen Howell
Seamus O'Neill
October 2012

Software & Resources

This edition uses Scratch 1.4. This software is free and can be downloaded from http://scratch.mit.edu. It is available for Windows, Mac OSX and Linux. It can be installed or, where there is no administrative permissions to install software, it can be unzipped and run without installation.

Scratch 2.0 is web based and currently being developed. Although as of writing, a release date has not been set, we are confident that most if not all the tasks, skills and instructions of this edition will apply to Scratch 2.0. We will be updating the text when Scratch 2.0 is finally released.

All the Scratch projects used in this text can be downloaded as completed projects from the Scratch website user forums.
Go to http://scratch.mit.edu/users/scratchfromscratch to see the projects (they are named from task04.sb, task05.sb etc. as they are named in the text). You may need to register as a user on the forums to download projects. This registration is free.

Contents The Tools & Skills of SCRATCH

Introduction: Acknowledgements		**5**
Downloading SCRATCH etc.		6
The Tools and Skills of SCRATCH		7
The Scratch Interface		8 – 9
Skill 1	**Knowing the Names of the Scratch Tools**	
Motion, Control and **Looks Palettes**		**10 – 11**
Skill 2	**Dragging and Dropping**	10
Skill 3	**Editing a Text Field**	10
Skill 4	**Using a Drop List**	11
Task 1	Make the cat move: **[move]** block	11
Task 2	Make the cat move further: **[repeat]** block	11
Task 3	Animate the cat: **[next costume]** block	11
Task 4	Make the cat walk: **[green flag]** and **[wait]** blocks	12
Skill 5	**Importing a New Sprite; Giving it a New Costume**	12
Skill 6	**Copying Code from One Sprite to Another**	12
Task 5	Make the bat fly	12
Task 6	Click the bat to make him fly: **[when sprite clicked]** and **[forever loop]** blocks	12
Task 7	Bat, Cat, Hat, Rat: A Word/Picture Game for Infants **[when key pressed], [hide]** and **[show]** blocks	13
Task 8	Burst the ball: **graphics** and **size** blocks	14
Skill 7	**Understanding How a Script Runs**	14
Skill 8	**Using Graphics Effects**	14
Task 9	Investigate colour: **[set color effect]** and **[change color effect]** blocks	14
Sensing Palette		**15 – 16**
Skill 9	**Using Reporter Blocks**	15
Task 10	Click anywhere to stop the bat: **[if]** loop, **[mouse down?]** reporter and **[stop script]** blocks	15
Task 11	Set the start position of a sprite: **[go to x: y:]** and **[if on edge]**	16
Task 12	Put the cat in a spin: **[touching mouse?]** reporter	16

Operators Palette: the Paint Editor		**16**
Task 13	Put the cat in a double spin: **[not]** operator block	16
Task 14	Use random numbers; Copy and Edit a costume: **[pick random]** operator block	17
Skill 10	**Erasing Pixels in Paint Editor**	17
Skill 11	**Creating a New Sprite in Paint Editor**	18
Skill 12	**Changing the Colour of a Sprite**	18
Task 15	Create two disc sprites in Paint Editor	18
Variables Palette: (a) Variable Blocks		**19**
Skill 13	**Creating a Variable**	19
Task 16	Count the score every time green touches red **[touching color], [set variable], [change variable]**	19
Task 17	Create a text sprite and add script to it: **[show]** and **[hide]** blocks	20
Task 18	Broadcast and receive a message: **[broadcast], [when I receive]** blocks	20
Skill 14	**Importing a New Background from File Making a copy / Adding Title Text**	21
Task 19	Switch backgrounds when a broadcast is received: **[switch to background]** block	21
Skill 15	**Planning a Scratch Project with Multiple Sprites**	22
Task 20	Create a Scratch Project with Multiple Sprites: **[color touching color], [point in direction]** blocks	22 – 24
Skill 16	**Painting a Background in Paint Editor**	25
Task 21	Make a Game: It's Raining Bones: **[set], [change]** and **[score]** variable blocks	25 25 – 26
Skill 17	**Understanding X and Y in Scratch**	26
Task 22	A two-way spinner: **[set x], [set y], [x position], [y position], [greater than]** and **[and]** operator blocks	27
Task 23	A Boy and his Dog – the Movie: **[say]** and **[say for]** blocks	28
Sounds Palette		**29**
Task 24	Explore the Sound Blocks	29
Skill 18	**Recording in Scratch**	29

Task 25	Ask and Answer:	30
	[ask and wait], [answer] and **[join]** blocks	
Variables Palette: (b) List Blocks		**30**
Skill 19	**Understanding Lists**	31
Task 26	Make and use a list:	31
	[add item], [item of] and **[length of]** blocks	
Task 27	Create an easy quiz:	32 – 33
	concatenation explained	
The Pen Palette		**34**
Skill 20	**Using the Pen Blocks**	34
Task 28(a)	Write and draw freehand	34
Task 28(b)	Draw straight lines	34
Task 29(a)	Draw regular shapes	35
Task 29(b)	Draw the circumference of a circle	35
Task 29(c)	Shapes and repeating patterns	35
Task 30	Patterns and Spirals:	
	(a) Use a variable to create a design	36
	(b) Put a loop inside another	36
	(c) Make rows and columns with the stamp tool	36

Seamus O'Neill is a primary teacher and the co-author of Mathemagic, the popular maths textbook for primary schools. He is a computer programmer and he tutors teachers in Scratch programming in Navan Education Centre.

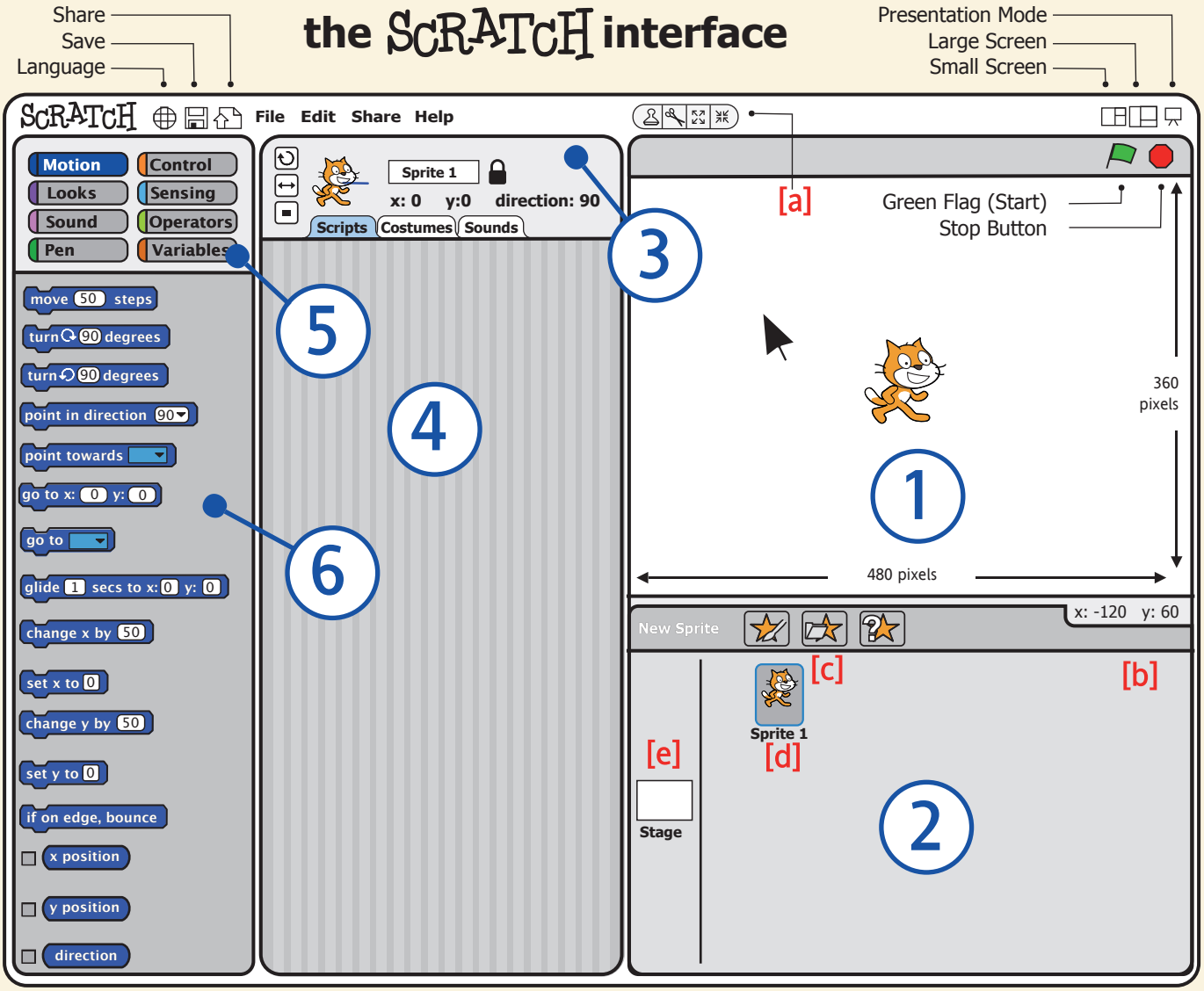
Stephen Howell is a Computing lecturer with the Institute of Technology Tallaght where he lectures on Interactive Media & Software Development.

The Tools and Skills

of

SCRATCH

the SCRATCH interface



Skill 1 Knowing the Names of the Scratch Tools

When you start Scratch you are presented with the interface which contains all the tools of Scratch and where all your Scratch programming activities take place. Your enjoyment of Scratch starts here and your first skill is to get to know the names of the tools and of the essential parts of the interface. You need enough information to get started right away.

1. the Stage: Perhaps the most noticeable feature of the interface at start-up is the large white area with the cat right in the middle. The cat is a **Sprite** and the white area is the **Stage**.

The cat is the start-up (or the default) sprite. **Can you see the cat in three different areas of the interface?** His start-up name is Sprite 1. Scratch is almost all about programming sprites.

Areas of the Interface

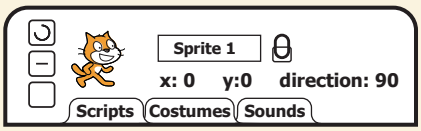
1: Stage
2: Sprite List
3: Current Sprite Info
4: Scripts Area
5: Blocks
6: Blocks Palette

The stage is 480 pixels wide and 360 pixels tall. Every point on the stage has a name, like *first name, surname,* except it's an **x** and **y** which are two numbers. To find out the x,y of any point in the interface (not just the stage) move the mouse-pointer around and look in the bottom right corner just below the stage. The pictured mouse (cursor) is at position -120, 60. The starting position of the cat is 0,0. **See can you bring the mouse to stop over 0,0?**

2. the Sprite List:

When you start Scratch the *sprite list* consists of only one sprite, the default cat sprite (Sprite 1). Most projects will have many sprites and each sprite will appear in the *sprite list* with its name under it.

3. the Current Sprite Info:



The *current sprite info box* contains all the components (f – l). These are all listed and explained below.

4. the Scripts Area:

(1) The *scripts area* is where you create your *scripts* or, in other words, do your *scripting* (programming).

- You create your script **on** whichever sprite is the *current sprite.* You can also create script on the *stage* – but otherwise, your script is always on a sprite.
- Drag code blocks from the *blocks palette* into the *scripts area.*
- Create script by snapping blocks together into stacks.
- Move or drag a stack by its top block.
- When removing a block from a stack all the blocks below it move with the block you are dragging – the bottom block will always move on its own.
- A white highlight shows you where you can insert blocks or stacks into an existing script.
- To remove a block or stack from the *scripts area* drag it back into the *blocks palette*.
- To run a script, click anywhere on the script or, click the green flag above the stage, if it is programmed to start the script.

(2) There are three tabs at the top of the *scripts area :* Scripts, Costumes, Sound

- At start-up, the scripts tab is the default. We have spent some time explaining *scripts.*
- Click the *Costumes tab* to change from *scripts area* to *costumes area* and click the *Sounds tab* to switch to *sounds area*. In Scratch, a *costume* is the look or appearance of a sprite (like 'frames' of a cartoon animation. We will explain a lot more about costumes and sounds later.

5. the Blocks:

There are eight colour-coded categories of *blocks*. By default, at start-up all the *Motion* blocks are visible in the *blocks palette.* Click on a block category to populate the *blocks palette* with that category.

6. the Blocks Palette:

The *blocks palette* has already been mentioned. The blocks of any category are displayed in the *blocks palette.* Drag blocks from the *blocks palette* to the *scripts area* and vice-versa.

Components of the Interface: (Main picture)

a. Toolbar. The four tools are:
Duplicate, Delete, Grow, Shrink

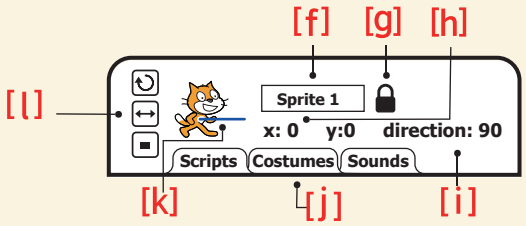
b. Mouse x / y Location

c. New Sprite Buttons:
(1) Paint New Sprite,
(2) Choose New Sprite from File,
(3) Get Surprise Sprite

d. Sprite Thumbnail: The highlighted sprite is the *current sprite* and the sprite thumbnail will also appear in the *current sprite info box.*

e. Stage Icon: Click the *stage icon* to switch the *scripts area* to the *stage scripts area*

Components of the Current Sprite Info Box



f. Sprite Name Box: Type a new name for the sprite

g. Sprite Drag Lock: Unlocked sprite is draggable

h. Sprite x / y Location

i. Sprite Direction

j. Tabs to Costumes / Sounds

k. Sprite Direction Line (blue)

l. Rotation Style Buttons:
(top) can fully rotate
(middle) can only face left-right
(bottom) can't rotate

Motion Blocks

a move 10 steps

b turn 15 degrees

c turn 15 degrees

d point in direction 90 — fixed list

(90) right
(-90) left
(0) up
(180) down

e point towards — dynamic list

f go to x: 0 y: 0

g glide 1 secs to x: 0 y: 0

h change x by 10

i change y by 10

j if on edge, bounce

k x position

l y position

m direction

n go to — dynamic list

o set x to 0

p set y to 0

Control Blocks

a when clicked

Dynamic names and lists change as the context changes.

b when Sprite 1 clicked — dynamic

c when space key pressed — fixed list (same as Sensing block **g** page 15)

d wait 1 secs

e broadcast — dynamic list

f broadcast and wait — dynamic list

g when I receive — dynamic list

h wait until

i repeat until

j forever if

k forever

l repeat 10

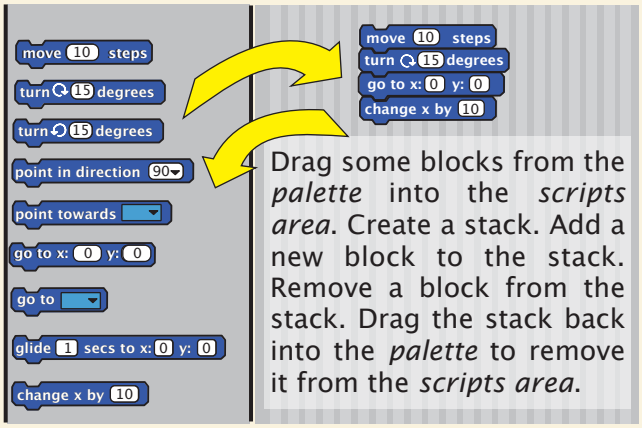
m if

n if else

o stop script

p stop all

Skill 2 Dragging and Dropping



Drag some blocks from the *palette* into the *scripts area*. Create a stack. Add a new block to the stack. Remove a block from the stack. Drag the stack back into the *palette* to remove it from the *scripts area*.

Skill 3 Editing a text field

move 50 steps

Some blocks like this [move] block have editable text fields. To change the value, click inside the rounded white area and type a number. Other blocks like the [say] block below, have a rectangular text field where you can type your own words.

say Hello!

Looks Blocks

a switch to costume costume1 ▾
dynamic list

b next costume

c ☐ costume #

d switch to background background1 ▾
dynamic list

e say Hello! for 2 secs

f say Hello!

o think Hmm...

g think Hmm... for 2 secs

h change color ▾ effect by 25
fixed list

i set color ▾ effect to 0
fixed list

j clear graphic effects

p show

k change size by 10

q hide

l go back 1 layers

r ☐ size

m next background

s set size to 100 %

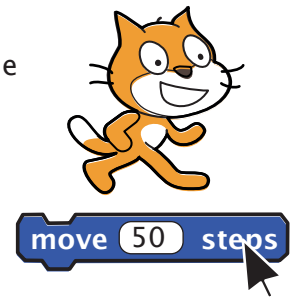
n ☐ background #

t go to front

Task 1 Make the cat move.

Drag out Motion block (a) and change its value to 50.

What happens when you click the code block?

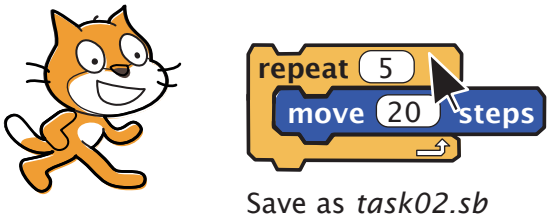


Save your work as *task01.sb*

Try these variations!
[Use Motion block (b) or Motion block (c)]

Task 2 Make the cat move further.

Drag out Control block (l). Blocks of this type are *loops*. You can insert other blocks into the 'mouth' of a loop block. Drag the [move] block into the [repeat] and change the values as shown. Click the [repeat] block to animate the cat.



Save as *task02.sb*

How many steps does the cat take now?

Task 3 Animate the cat.

Drag out Looks block (b).

When you click the block does the sprite move steps or just animate on the same spot?

Save as *task03.sb*

Skill 4 Using a Drop List

color
fisheye
whirl
pixelate
mosaic
brightness
ghost

set color ▾ effect to 0

Blocks with a small black triangle have a drop list. Click on the triangle to see the list. Some lists have **fixed** data such as the list of *effects* shown in the [set color] block on the left. Other lists are **dynamic** in the way they change when you create a new sprite or make changes to the code. For example, the list in the [point towards] block would contain the name of *Sprite 2* when you create a new sprite with the name 'Sprite 2'.

point towards ▾

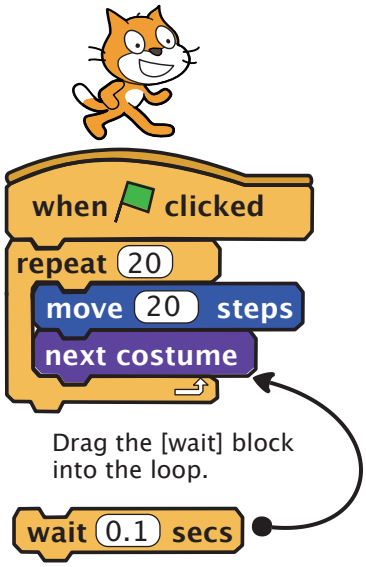
More about this list in SKILL 8 on page 14

Task 4 Make the cat walk.

You will need *Control* blocks (a) and (d) as well as the blocks you have already used in Tasks 1, 2 and 3. Change values as shown.

Drag the sprite to left edge of the stage before running the script.

Test
Click the green flag above the stage, to test the script **before** adding the [wait] block.



Drag the [wait] block into the loop.

Save as *task04.sb*

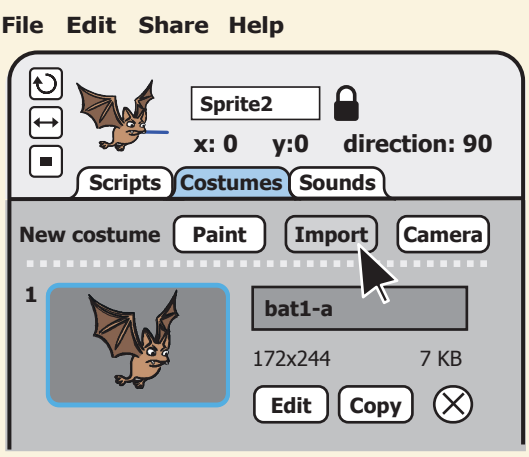
After testing, can you see why you need to include the [wait] block?

Skill 5 Importing a New Sprite Giving it a New Costume

1. Click the *new sprite from file* button.

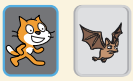
2. In the *Animals* folder choose **bat1-a**.
3. Click the *Costumes* tab.

[Notice that **bat1-a** has only one costume but the default **cat** sprite has two.

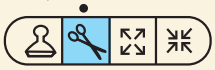


4. Click the *Import* button.
5. In the *Animals* folder choose **bat1-b**

Skill 6 Copying Code From One Sprite to Another



From the *scripts area,* drag the code you want to copy from the source sprite (1) to the target sprite (2) in the *sprite list*. When you see a faint grey rectangle around the target sprite release the mouse. Delete the cat sprite when you have copied the script. **How?** To delete a sprite, select the *Delete* tool and click the sprite on the stage or in the *sprite list.*



Task 5 Make the bat fly.

bat 1-a
sprite
costume 1

bat 1-b
costume 2
of the bat
sprite

Use the very same code as in Task 4 but first you must import the **bat** sprite and a second costume. Copy the code from the *cat* sprite to the bat sprite. Click the **green flag** above the stage to start the animation.

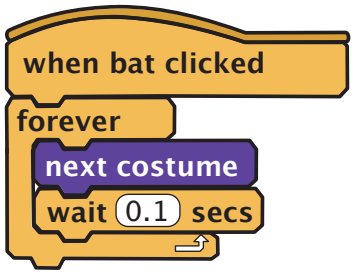
Save as *task05.sb*

Task 6 Click the bat to make him fly.

In the *sprite name box* change the name of Sprite 2 to bat. You will need Control blocks (b) and (k) as well as [next costume] and [wait].

Click the bat to see it flap its wings continuously on the same spot.

To stop the action click the stop button above the stage.



Save as *task06.sb*

When we introduce the *Sensing* blocks (page 15) **we can make this animation more interactive and have much more fun.**

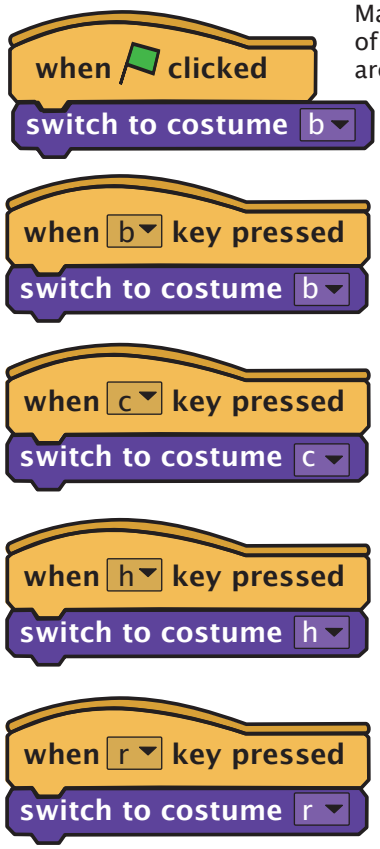
Task 7 Bat-Cat-Hat-Rat

Start a new project and delete the cat sprite.

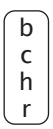
This is a word game for infants. These letters will be three sprites. The **B** sprite will have 4 costumes. The **A** and **T** sprites do not need multiple costumes, just the one you import in step 1.

PART 1: The words

1. Import three new sprites **B**, **A** and **T** which you will find in the *Letters /outline* folder.
2. With **B** as the *current sprite,* click on the *Costumes* tab. *Import* three new costumes, the letters **C, H** and **R**. **B** has 4 costumes in all now.
3. With sprite **B** still the *current sprite* . Click the *Scripts* tab. From the *Control* palette, drag out the blocks, [green flag], [when key pressed] and [switch to costume]. The first two are *Control* blocks (a) and (c) and the [switch to costume] block, is *Looks* block (a).
4. Use the *Duplicate tool* to make **three** copies of [when key pressed] and **four** copies of [switch to costume]. Click the small black triangle to select the keyboard **keys b, c, h** and **r**. Look at the pictures below.
5. Centre the word **BAT** at top centre of stage.



Make sure that all five of these script stacks are on the **B sprite**.



When you click on the small black triangle you should see this list of four letters.

First you see the letter **B**. When key **C** is pressed the costume you see is **C**. When key **H** is pressed the costume you see is **H** and so on. When key **B** is pressed you see the letter **B** again.

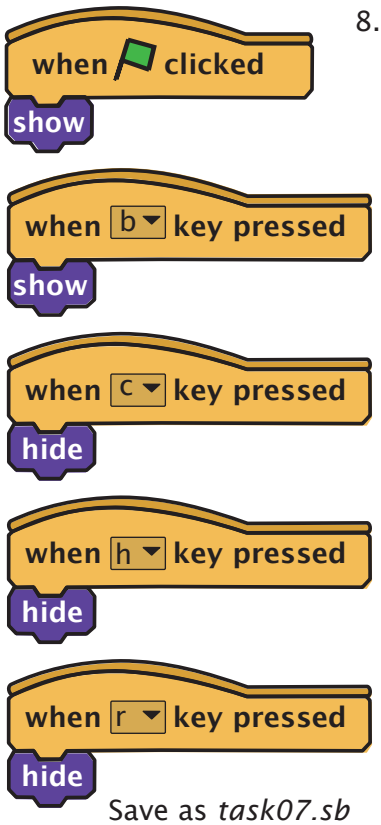
continued in column 2

PART 2: The pictures

6. Import four picture sprites: **bat1-a, cat3, mouse1** (*Animals* folder) and **partyhat1** (*Things* folder).

When imported, they will all be on top of each other in the centre of the stage – that's OK! When the *green flag* is clicked they will all become invisible except for the **bat**.

7. Click the **bat** to make it the *current sprite.* From the *Looks* palette, drag the [show] and [hide] bocks onto the stage. Create the following script for the **bat** sprite.



Save as *task07.sb*

8. *Copy* the scripts to each of the remaining three **picture** sprites. Make each one **hide** when the *green flag* is clicked. The only visible sprite should be the bat. **You need to make a small change to each of the other sprites to make them visible when their letter key is pressed.** All the other sprites will be invisible then.

What do you need to do to include the word *mat* plus a picture of a mat in the project?

Hint:
Look in the transportation folder for a picture of a mat!

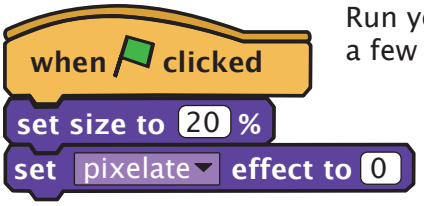
Task 8 Burst the ball

Start a new project. *Delete* the cat.

In this task we are going to simulate inflating a balloon or ball until it bursts. You will need *Looks* blocks (e), (h), (i),(k) and (s). Most of these blocks can change the look and size of a sprite.

Import **beachball1** from the *Things* folder.

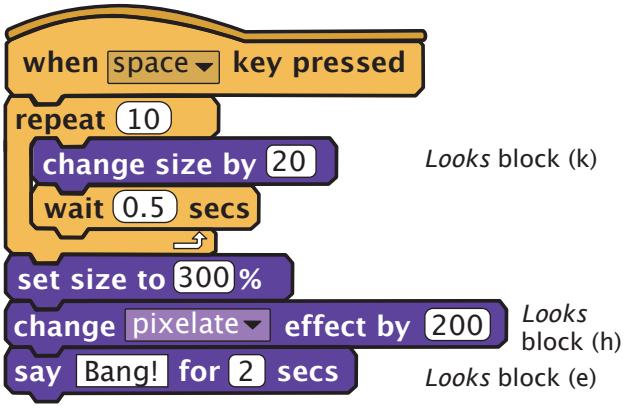
Put all this script on the **beachball1** sprite.



Run your script a few times.

Looks block (s)

Looks block (i)



Looks block (k)

Looks block (h)

Looks block (e)

Save as *task08.sb*

Skill 8 Using Graphic Effects

This is the list of graphic effects you see when you click the small black triangle on graphics blocks (h) and (i) in the *Looks* palette. The best way to understand what they do is experiment with them on a sprite.

color
fisheye
whirl
pixelate
mosaic
brightness
ghost

Task 9 investigates the colour effect.

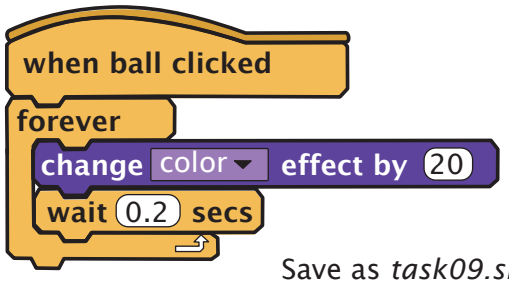
Task 9 Investigate colour

(a) Drag this block into the *scripts area* and change the value.

Here's a guide to each colour (by number) when the sprite's colour is **red**:

red	1	**blue** (light)	100
orange	15	**blue** (dark)	120
yellow	30	**indigo**	160
green (light)	45	**purple**	170
green (dark)	60	**pink**	180
		red	200

(b) Try this disco colour effect.



Save as *task09.sb*

Skill 7 Understanding How a Script Runs

Code tells a computer precisely what to do, step-by-step. In Scratch you create programmes using blocks of script which give precise instructions to a sprite, telling it to move, turn, change size, say something, vanish etc. You then click on the script or on the green flag to run the script and start the programme.

SCRATCH starts at the top block and runs down through the script to the bottom. Sometimes SCRATCH has to complete a loop (or several loops) along the way. It may have to complete a loop *if a condition is met* or *else* to complete a different loop. There are blocks to create *forever* loops, *repeat* loops and *conditional* loops. You have used a *repeat* loop as recently as in Task 8. You will use *if* loops from Tasks 9. (*if else* see p. 27)

Look again at task 8 and see what you can learn about how a Scratch script runs.

This script does *not* run from start to finish. There is a point where the script stops and waits for the user to do something. What has the user to do to re-start the script?

What size is the sprite after you click the green flag?

What is the loop meant to simulate? (There's a hint in the next question.)

Can you count each round of the loop while the ball (or balloon) is inflating?

What size is the sprite after the loop but before it becomes 3-times its normal size?

Sensing Blocks

a touching ? — dynamic list

the default list: edge; the names of sprites will be listed here also

b touching color ?

c color is touching ?

d ask and wait

m answer

e mouse x

n mouse y

f mouse down?

g key space pressed ?

h This is a fixed list of all key-board keys.

up	d	3
down	e	4
left	f	5
right	g	6
space	. . . etc.	7
a	0	8
b	1	9
c	2	more...

i distance to — dynamic list like bock (a)

j reset timer

o timer

k loudness

p loud?

l x position of Sprite 1 — dynamic list like bock (a)

Skill 9 Using Reporter Blocks

Perhaps the first thing you notice abut the *Sensing* palette is that most of the blocks have hexagonal or round-ended shapes. Blocks like these are known as *Reporters* and they are designed to fit into other blocks. Reporters with round ends fit into blocks with **round** or rectangular holes. **Hexagonal** reporters fit inside blocks with hexagonal holes. There are some *reporters* in the *Motion* and *Looks* palettes as well as in the palettes we have not used yet.

Task 10 Click anywhere to stop.

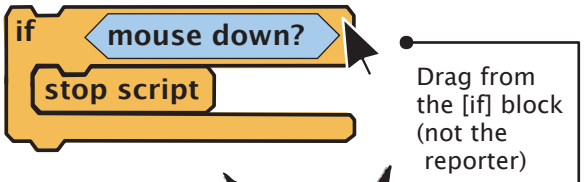
Open the project **Task06.sb**. Drag the following blocks into the scripts area. Follow the three steps below.

if — *Control* block (m)

stop script — *Control* block (o)

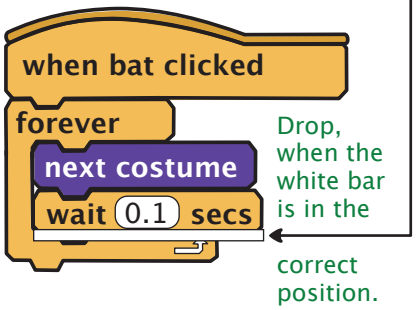
mouse down? — *Sensing* block (f)

1. Lock the three blocks together. The [if] block makes a *conditional* loop.

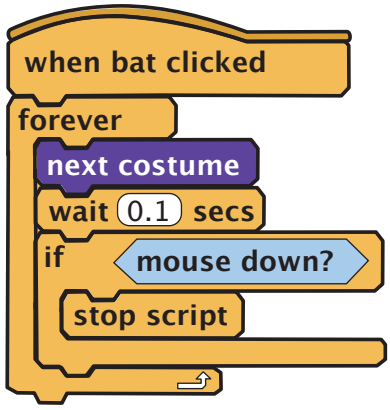


2. Drag the [if] loop into the **Task06** script. When a white bar appears **under** the [wait] block drop the loop. The [if] loop must be below the [wait] block.

Look out for the white bar.



3. The new script should look exactly like this. The [forever] loop will stretch to take in the [if] loop.



Save as *task10.sb*

Click the bat to start the animation.
Click anywhere else to make it stop.

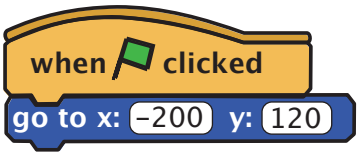
Can you figure out the logic of the code and what each block does?

Task 11 Set the start position of a sprite.

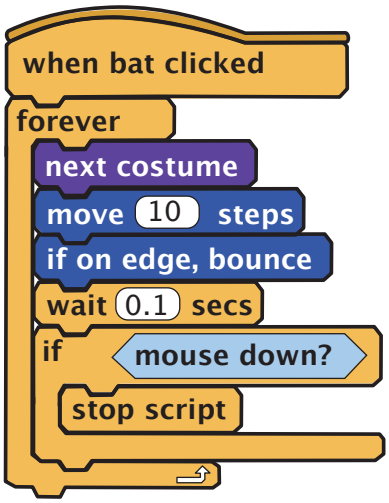
In this project you will make changes to the code in **Task 10**. You will use *Motion* blocks (f) and (j) for the first time and you will also need *Control* block (a). Follow these steps.

1. Shrink the bat sprite until it is quite small. Click the sprite with the *Shrink Button* about 40 times! Click on the stage to get your pointer back.

2. Create this new script and edit the values.

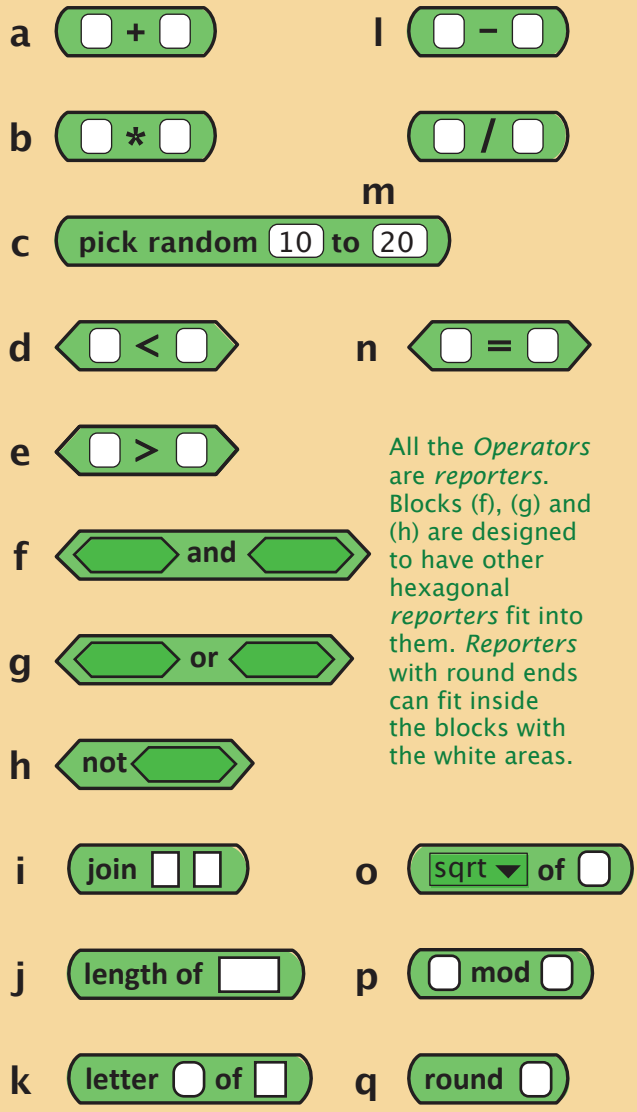


3. Insert the two new blocks so that the script looks exactly like this. Click the *Rotation Style Button* to make the sprite *only face left-right*.



Save as *task11.sb*

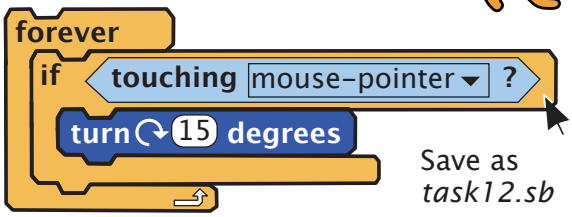
Operators



All the *Operators* are *reporters*. Blocks (f), (g) and (h) are designed to have other hexagonal *reporters* fit into them. *Reporters* with round ends can fit inside the blocks with the white areas.

Task 12 Put the cat in a spin.

In this task you will use *Sensing* block (a) for the first time.



Save as *task12.sb*

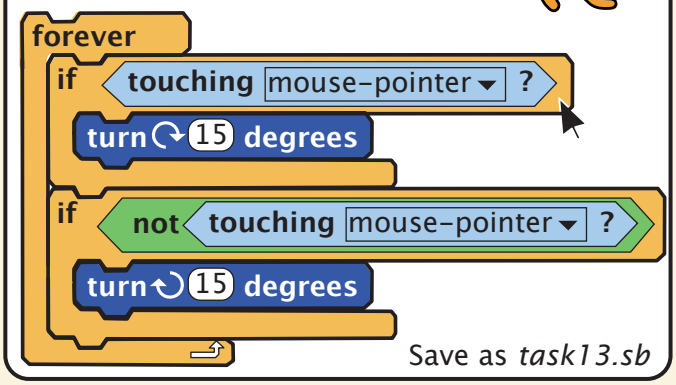
Move the mouse over the sprite. There's no need to click. If it stops and starts that is because, as the sprite turns, it may not be touching the mouse-pointer all of the time.

Task 13 Put the cat in a double spin.

In this task you will use *Operator* block (h). Fit *Sensing* block (a) inside the *Operator* block.

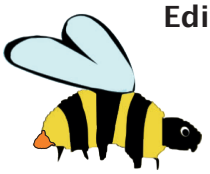
Create the script and then click anywhere on the stack to start the animation.

Can you explain how it works?



Save as *task13.sb*

Task 14 Use random numbers to make a bee float about. Copy and Edit a costume.



You'll find the **bee1** sprite in the *Animals* folder

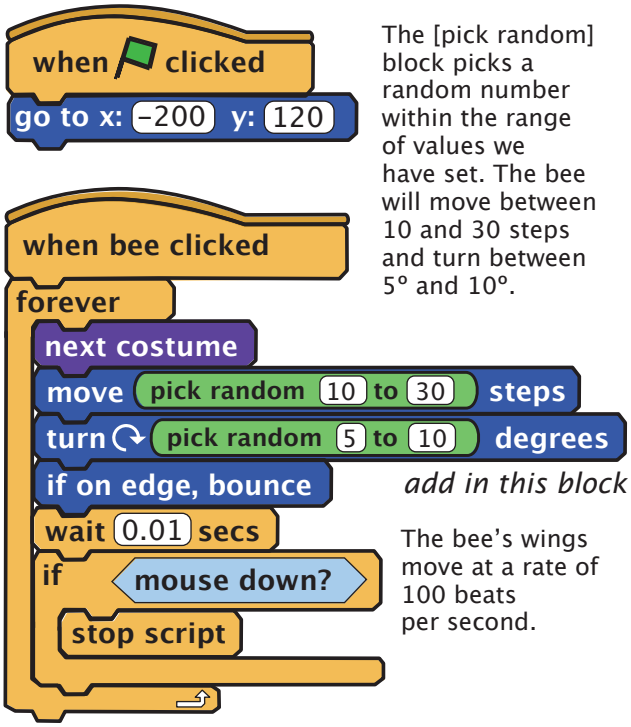
1. Open **Task11** and save it as *task14.sb* before making any changes. Import the **bee1** sprite. Shrink its size and give it the name *bee*. Copy all the script from the **bat** sprite to the *bee* sprite and only then, delete the bat. Click the *Costumes* tab.

2. Have you noticed that the sprite has only one costume? Click the *Copy button* and now there's a new costume, **bee2**. You might ask, 'What's the point in having two identical costumes?

3. Still in *Costumes*, click the *edit button* and the **Paint Editor** will open. **[See Skill 10 below.]** Select the *Eraser tool* and choose a middle-sized brush in the *Options* area. Erase the wings of costume **bee2** and click *OK*. The idea is, every time the [next costume] block loops it should create the effect of the bee's wing movement. We also want to make the bee move randomly so we need to makes changes to the script. We will use *Options* block (c).

4. Now we are ready to make changes to the script. You need to insert *Motion* block (b) [turn clockwise] just after the [move] block. Look at the picture opposite. Fit the [pick random] *reporter* block into both the [move] and [turn] blocks and edit the values.

continued



The [pick random] block picks a random number within the range of values we have set. The bee will move between 10 and 30 steps and turn between 5° and 10°.

add in this block

The bee's wings move at a rate of 100 beats per second.

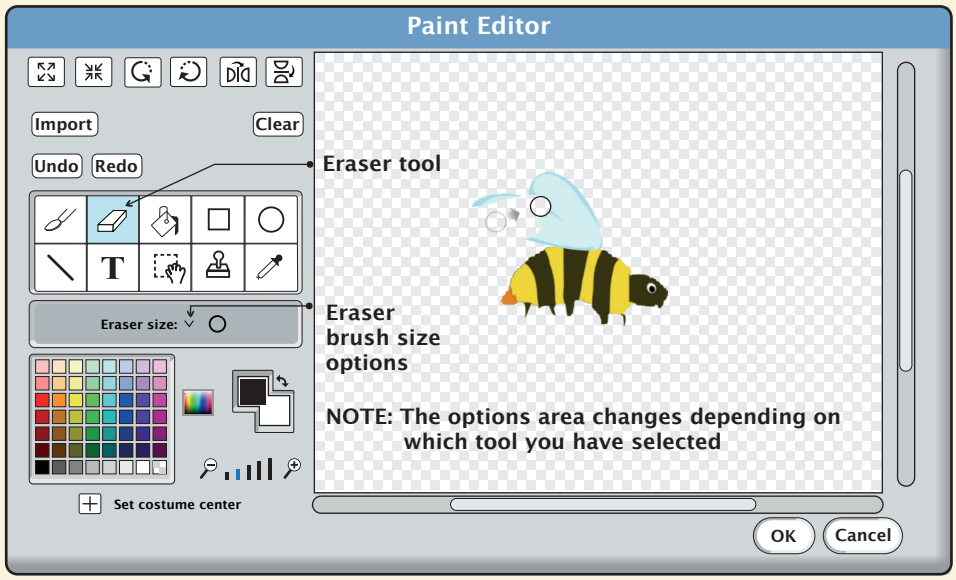
You have already saved it as *task14.sb*. Save and click the green flag to try it out.

5. One way to improve this project

(i) Make the bee always start with wings. Add the following block, *Looks* block (a) to the [green flag] stack.

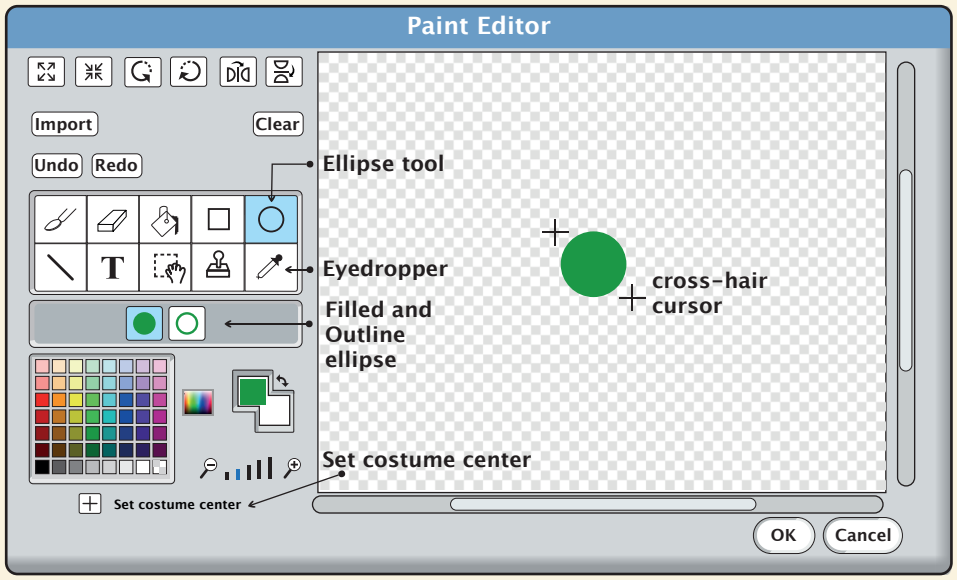
(ii) Click the Rotation Style Button ↔ to prevent the bee turning upside down.

Skill 10 Erasing Pixels from a Sprite in the Paint Editor



1. Select the *Eraser* tool
2. Click the 'v' in the *Options* area and choose a middle-sized brush from the options available.
3. Use the eraser as you would use an ordinary eraser, moving carefully back and forth until you erase the pixels you don't want.
4. Click OK to return to the Scratch interface.

Skill 11 Creating a New Sprite in the Paint Editor



A. In Scratch click the

Paint New Sprite **button**

B. Draw a green disc

1. Select the *Eyedropper* and pick a dark green colour.
2. Select the *Filled ellipse* tool.
3. Press and drag the cross hair cursor from top left to bottom right. (To make a perfect circle hold down the **Shift Key** while you drag). Release the mouse.
4. Click the *Set costume center* button.
5. Click OK.

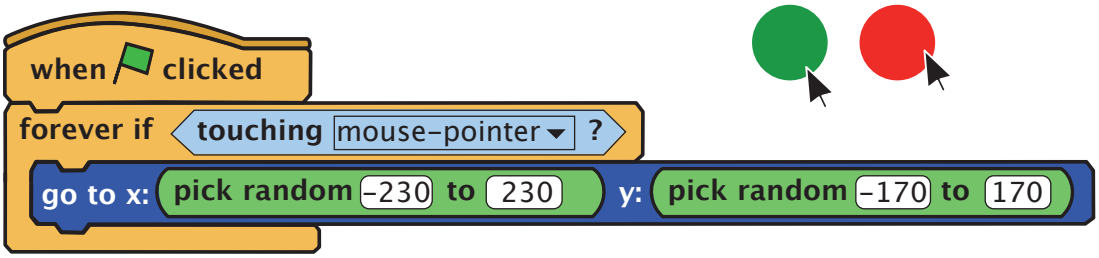
Skill 12 Changing the Colour of a Sprite in the Paint Editor

Change the green disc to a red disc.

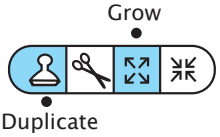
1. Make the green ball sprite the *current sprite.* It will be visible in the *current sprite info box.* The *current sprite* is also the one that is bounded by a blue box in the *sprite list.*
2. Click the *Costumes* tab and click *Edit.* You should see the green ball in the centre of the 'canvas'.
3. Select the *Eyedropper* tool and pick a **red** colour in the *colour palette.*
4. Take the *Fill* tool (the paint bucket) and touch the green ball with the tip of the pouring-paint. The ball will turn red. Click OK.

Task 15 Create two sprites – one a green disc and the other a red disc. This is the first part of a ball touching game

1. Create a green disc sprite in the *Paint Editor* and give it the name *greenBall.*
2. Add the following script to the *greenBall* sprite.



3. Duplicate the *greenBall* sprite. How? To duplicate a sprite, select the *Duplicate tool* (the stamper) and click the sprite you want to copy either on the stage or in the *sprite list.*



4. Give the new sprite the name *redBall* and change the sprite's colour to **red**.
5. Copy the code from the *greenBall* sprite to the *redBall* sprite.

The two sprites should react to the mouse-pointer in the same way since they're both coded with the same script. Your task is to get both discs to overlap.

If it takes too long to get the discs to touch, it means that the game is too difficult. It shouldn't be too easy either. To make the game a little easier however, use the *Grow tool* to make the two discs larger. Have you improved the game?

Save as *Task15.sb*

Variables

Variable List Blocks **f** to **m** can be seen on page **30**.

Make a variable

Delete a variable

a ☑ score

b change score by 1

c set score to 0

d show variable score

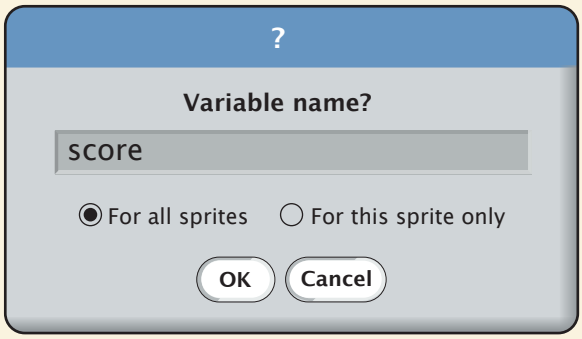
e hide variable score

The *Variable* blocks become visible only after you click [Make a variable]. The first thing you do to make the variable is to give the variable a name. If your new variable is going to manage the score in a game you might decide to give it a name such as *score*. If it's going to keep a count of the lives of a sprite, you might give it the name *lives*. Naming a variable is up to you.

The *Variables* blocks also includes [Make a list] which we will include later (see p. 31). For now we will add a scoring feature to make the ball-touching game more interesting.

Skill 13 Creating a Variable

1. Click [Make a Variable] and type a name for your variable. Then click OK.



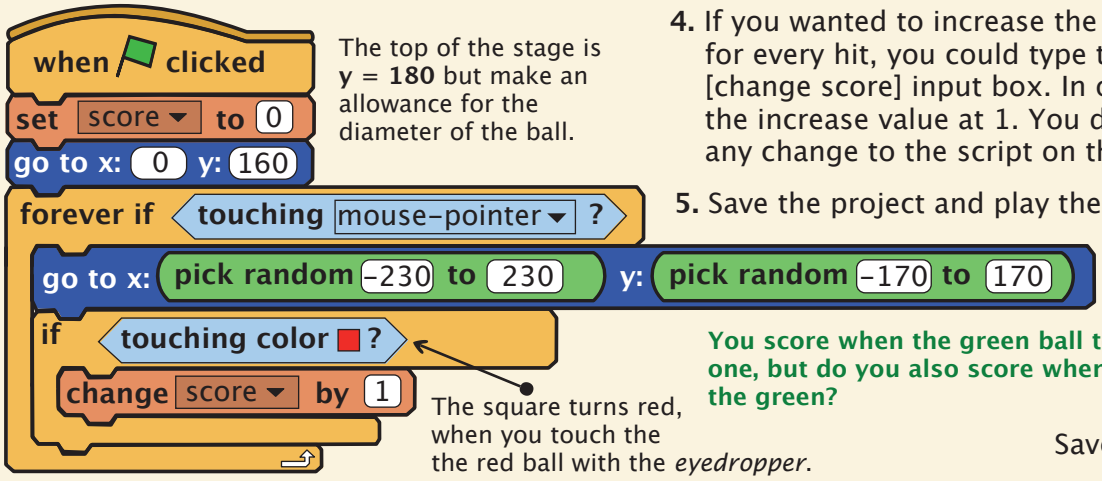
2. *Variables* block (a) has a small tick-box on its left. To see the score monitor on the stage check the square. Tick the box.

3. Block (b) can be used to increase the score every time a point is won, a hit is made or a goal is scored. If you start a game with, for example, 10 lives, block (b) can also be used to decrease by one if a life is lost.

4. *Variables* block (c) is used to set the starting score or set the initial number of lives.

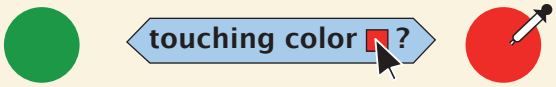
Task 16 Count a score every time green touches red.

1. Make changes to the script on the *greenBall* as follows: (See the picture below.)
   (i) Put the [set score] block at the start.
   (ii) Put a [go to] block next, to give the **greenBall** a starting position.
   (iii) Insert the [if] loop with [touching color] **inside** the [forever if] loop. It must be **after** the [go to]/[pick random] block.
   (iv) Insert the [change score] block into the 'mouth' of the [if] loop.



The top of the stage is **y = 180** but make an allowance for the diameter of the ball.

The square turns red, when you touch the the red ball with the *eyedropper*.



2. When you click on the **colour square** on the [touching color] block, you see your pointer changed into an eyedropper.

3. With the *Eyedropper*, touch the colour you want the **greenBall** to hit. Touch red because you want to score when the **greenBall** hits the **redBall**.

4. If you wanted to increase the score by 5 points for every hit, you could type the value 5 into the [change score] input box. In our example we left the increase value at 1. You don't need to make any change to the script on the **redBall**.

5. Save the project and play the game.

You score when the green ball touches the red one, but do you also score when the red touches the green?

Save as *Task16.sb*

Task 17 Create a text sprite and add script to it.

It is a good idea to instruct the user to click the *green flag* to start your Scratch programme. To create your instruction you can create a text sprite in the Paint Editor as follows:

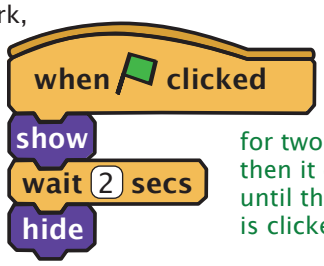
1. Click the *Paint new sprite* button and use the *Text tool* in the Paint Editor to type...

2. Use the small black square to move the text. Change the font type and size in the *Options area*. Click OK when done.

3. Add the following script to your text sprite.

To see it work, create an instruction to start *Task 14*.

Save as *task 17.sb*



The text appears on the stage for two seconds then it disappears until the green flag is clicked again.

Task 18 *Broadcast* a message from one sprite to be *received* by another sprite.

There are two parts to this task. In **Part A** you will create a sprite to use as a start button to a game or animation. You will add a *broadcast* script to the button which will call on sprites to run their scripts. In **Part B** you will add script to a sprite to make it *receive* a *broadcast*.

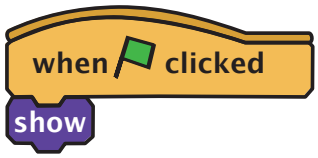
A. Create a new sprite and add a broadcast.

1. Open **Task 14** and ***Save as*** *task18.sb*.

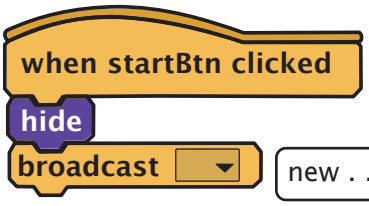
2. Click the *Paint new sprite* button and in the *Paint Editor* draw a small black box. Use the *Text tool* to type the word START in white text. Click OK. This is your start button. Give it the name *startBtn*.

3. Drag the new sprite to the bottom right corner of the stage.

4. In the *scripts area* add this script to the new start button sprite.



Every time the *green flag* is clicked the button is visible.

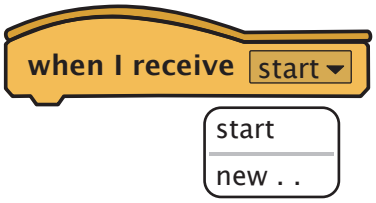


After the *startBtn* has been clicked it is made invisible.

5. Click the black triangle. Click *new...* and give the *Message* a name. Type *start*.

B. Make a change to the bee sprite script so that it receives the broadcast.

1. In the *scripts area* of the bee sprite replace the [when bee clicked] block at the top of the stack, with the [when I receive] block.



2. Click the black triangle. The new name you gave the message is in the list of broadcasts. Click the word *start* in the drop down list.

Explanation: After the *green flag* is clicked to set up your animation or game, the *start button* will become visible in the bottom right corner of the stage. When you click the start button it broadcasts the *start* message to any sprite that is programmed to receive the broadcast. The bee sprite is programmed to receive the *start* message and when it does, it runs its script. After clicking the button, it becomes invisible to clear the stage – even before it broadcasts. In Scratch, invisible sprites are still on the stage, it's just that you can't see them. (Look back at Task 7).

Click the *green flag* then click the start button to see your bee animate.

Save as *task18.sb*

Skill 14

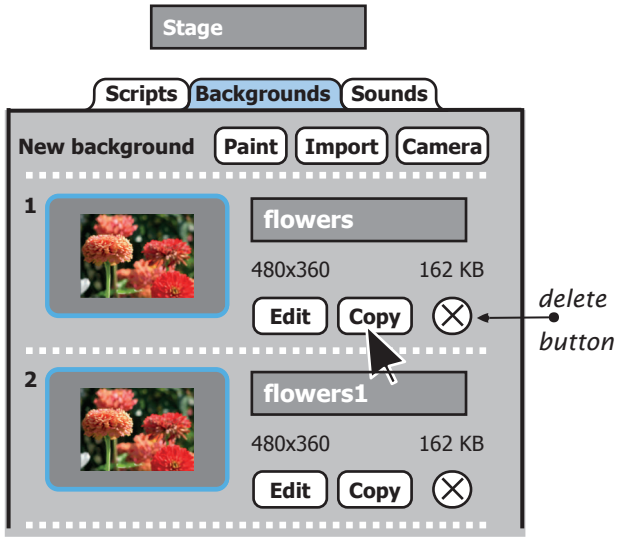
Importing a New Background from File/Making a Copy/ Adding Title Text

Importing

1. In Scratch click the *stage icon* – the white rectangle on the left side of the *sprite list* – and select the *Backgrounds* tab.
2. Click the *Import* button to get the *Import Backgrounds* folders.
3. In the *Nature* folder choose **flowers** and **OK.**
4. You may *delete* the white default background.

Copying a background

Click the *Copy* button and this will create a copy of your background with the name, **flowers1**.



Rename backgrounds as ***bg1*** and ***bg2***

Adding Title Text

Select *bg1* and click the *Edit* button. This will open the *Paint Editor*. Choose a chunky font type, size up to about 60pt and contrasting colour. Centre the text. (The title text font in the picture is a font called *Blippo.*

NOTE about Text in Scratch Paint Editor

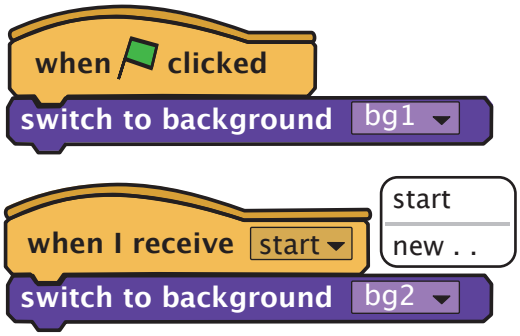
Backgrounds and *Costumes* can only have one block of text. If you want to add a second block, it must be a separate sprite (see *Task 17)*

Task 19 Switch backgrounds when a broadcast is received.

Practice Skill 14 by adding backgrounds and some title text to Task 18.

Adding scripts to the background

Click the *Scripts tab* to access the *scripts area* of the background. To switch from **bg1** to **bg2** add the following scripts. The new block here is Looks block (d). It has a *dynamic list* containing **bg1** and **bg2.**



Explanation: When the *green flag* is clicked the background with the title ***Buzzy Beez*** is the starting background. The button which, when clicked, broadcasts the *start* message is in the bottom right corner of the stage. Click the button and it broadcasts *start* to both the background and the bee-sprite as both have [when I receive] block. The background switches to **bg2** and the bee animates. Try it out now. Save as *task 19.sb.*

Skill 15 Planning a Scratch Project with Multiple Sprites

(a) What do I want to create in Scratch?
(b) Do I have the tools?
(c) Do I know the skills?

(a) I want to create a scene with *bees* flitting among flowers. Bees will stop on the flowers and will appear to come and go at random.

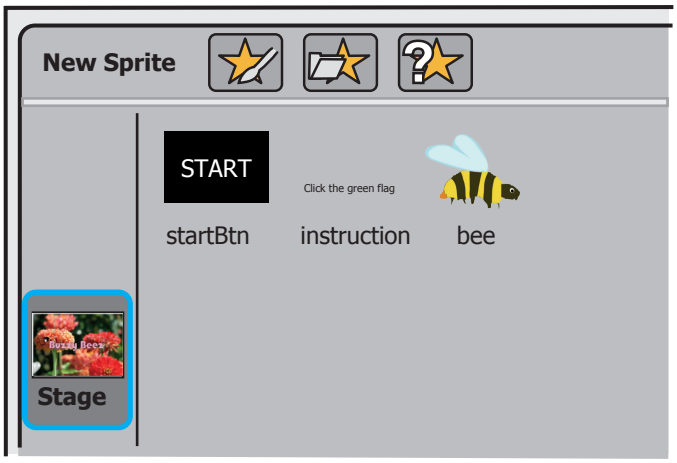
(b) I will build upon task 19 re-using the existing scripts and making use of some new blocks.

(c) When I click the *green flag* the scene will be set up and then clicking a button will start the animation.

I will use the [colour touching colour] block – *Sensing* (c) and [wait] blocks to make the bees stop on the flowers. I will use a different colour and the [touching colour] block – *Sensing* (b) and [wait] blocks, to make the bees disappear then reappear in a different location.

Task 20 a Project with Multiple Sprites

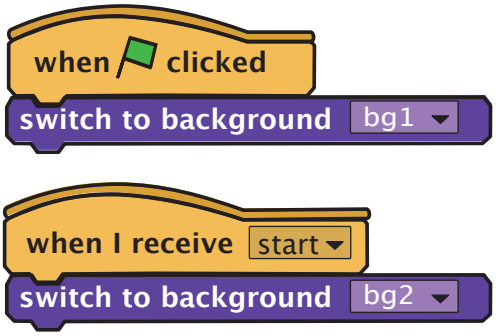
1. Open **Task 19** and take a look at the *sprite list.*



The *sprite list* contains four thumbnails, the *stage* and three sprites:
the *start button*
the *instruction* to click the green flag
the *bee*
The order may not be the same as in the picture.

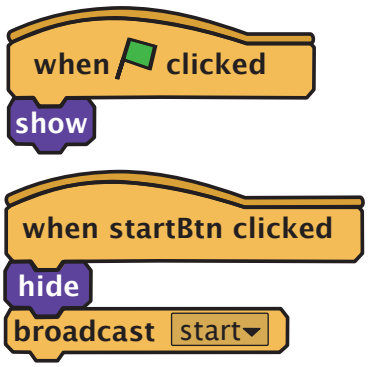
2. Each sprite has script attached.
Take a look at each sprite's script.
Click the thumbnail and then click the *Scripts* tab.

(a) the *Stage* thumbnail and scripts



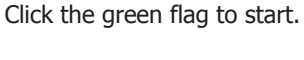
This script was first created in Task 19.
(i) When the *green flag* is clicked the background with the title *Buzzy Beez* is set as the current background
(ii) When the start button broadcasts '*start*' the stage switches to the animation background – no title, no start button.

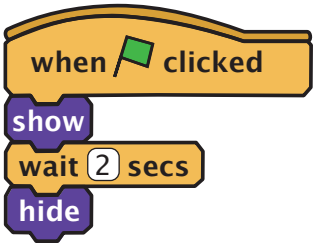
(b) the **startBtn** thumbnail and scripts



This script was first created in Task 18.
(i) When the *green flag* is clicked the button can be seen.
(ii) When the button is clicked, it broadcasts '*start*' and then it disappears.

(c) the **instruction** thumbnail and scripts





This script was first created in Task 17.
The instruction is visible for 2 seconds then it disappears.

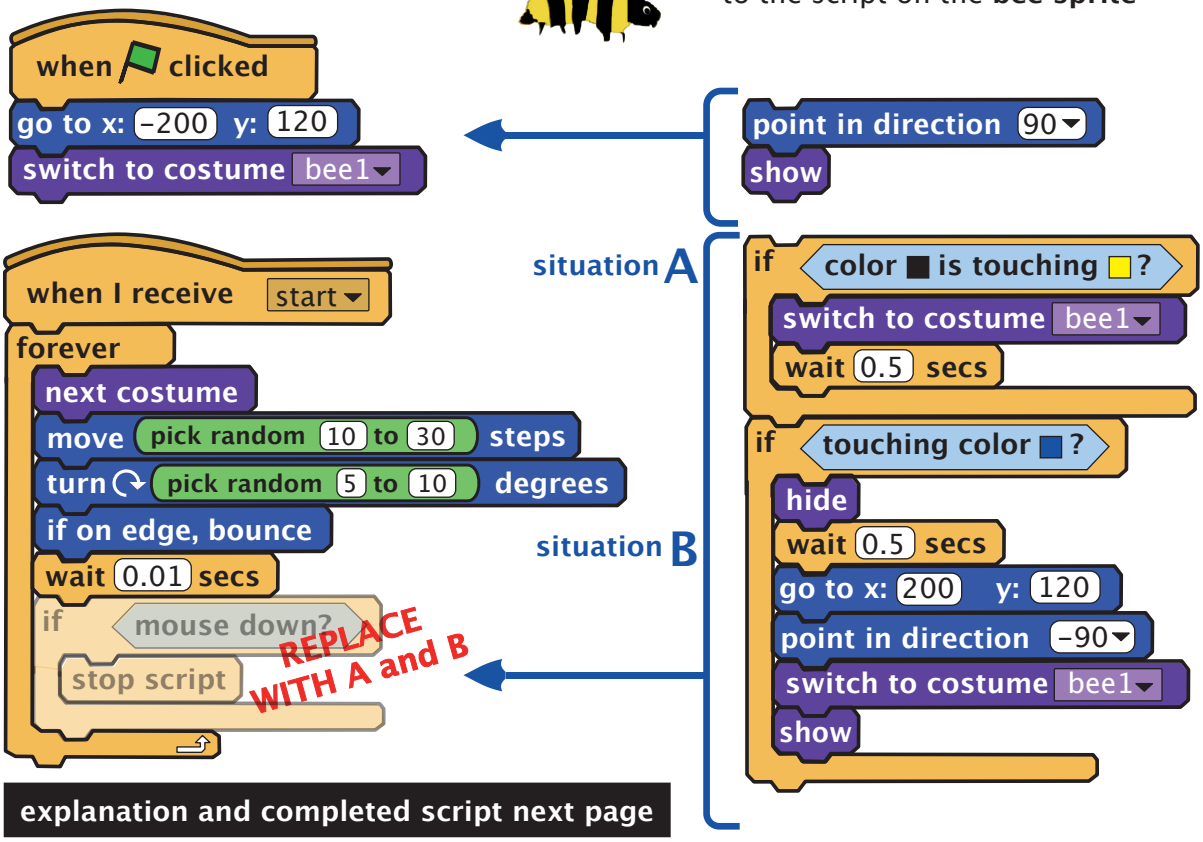
There is no need to make any change to the *Stage*, *startBtn* or *instruction* scripts.

(2) continued next page

(2) continued from previous page

(d) the **bee** thumbnail and scripts script on the **bee sprite** at **Task 19.**

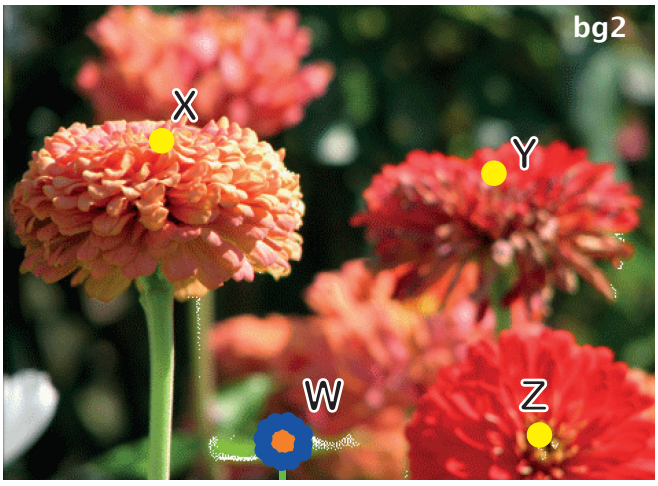
script to be added to the script on the **bee sprite**



explanation and completed script next page

(e) In the original script we used a conditional loop to stop the action *if* the mouse was clicked. Now, we intend to programme the *bee* sprite to stop on flowers and to simulate going and coming. To do this we will paint colours on the *animation background* (**bg2**) in the *Paint Editor*. Then we can use colour *Sensing* blocks (b) and (c) to control the activities of the bee.

(f) You get the *Eyedropper tool* when you click on the small squares in the colour-touching blocks. **Situation A:** Click the first of the two squares and hold the tip of the eyedropper on the head of the bee to 'pick' its colour; click on the second square and hold the tip of the eyedropper on one of the yellow circles X, Y or Z. **Situation B:** Click the small square and hold the tip of the eyedropper on the blue colour of the flower you have just drawn.



(g) Paint colours for the touching blocks

(i) Click the *Stage* thumbnail and the *Backgrounds* tab (illustration page 21). Click on the thumbnail of **bg2** and the *Edit* button beside it, to work on **background2** in the *Paint Editor*.

(ii) Select the *Filled ellipse* tool and draw 3 small yellow circles on the flowers like X, Y and Z.

(iii) Change the fill colour to blue and draw a small blue flower (W) with an orange-coloured centre.

(iv) Click OK.

point (3) next page

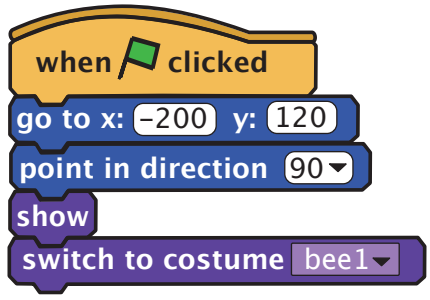
3. Make a duplicate of the **bee** sprite.

When a sprite is duplicated, any scripts on the sprite are duplicated also.

The scripts on duplicates are identical to the script on the original. The random numbers in the [move] and [turn] blocks will make the bees move differently.

Explaining the script: reading code step-by-step

Keep in mind that this is the script of only one bee sprite and that there are several other sprites that have their own scripts. They are dynamically linked in the same project with this sprite.



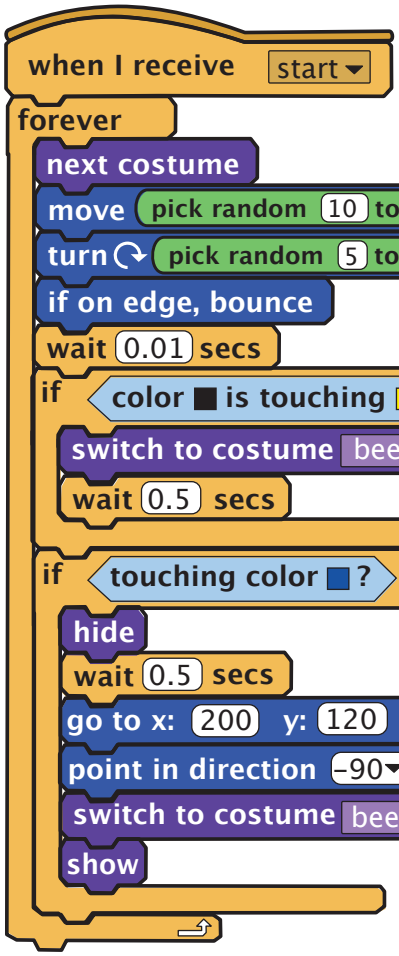
1. One click of the green flag will have an effect on several sprites simultaneously (the bees, start button, instruction).

1.1 The bee is positioned on the top, left side of the stage.

1.2 The bee is pointing to the right.

1.3 Each time the programme starts the bee is visible.

1.4 Each time the programme starts the bee has wings (in costume *bee2* the bee has no wings)



2. Clicking the *start button* broadcasts a *start* message. Here the *bee sprite* is receiving the broadcast and responding to it.

2.1 This is a **[forever]** loop.

2.1.1 The bee changes costume on each loop to simulate flying.

2.1.2 The bee moves 10 to 30 steps to its right.

2.1.3 The bee turns 5 to 10 degrees clockwise.

2.1.4 If at the edge, the bee turns in towards the stage.

2.1.5 This is the pause between each flap of the bee's wings.

2.2 If the black colour on the head or body of the bee sprite makes contact with the yellow of a flower this loop runs.

2.2.1 When stopped, the bee has wings.

2.2.2 This is the bee stopped on the flower for half of a second.

2.2. end

2.3 If the *bee* makes contact with the blue flower this loop runs. (It makes no difference what part or colour of the bee.)

2.3.1 The bee seems to leave the scene, it is invisible.

2.3.2 The bee is invisible for half of a second.

2.3.3 This is where the bee will re-appear (right side of the stage).

2.3.4 The bee will point to the left.

2.3.5 The bee will have wings.

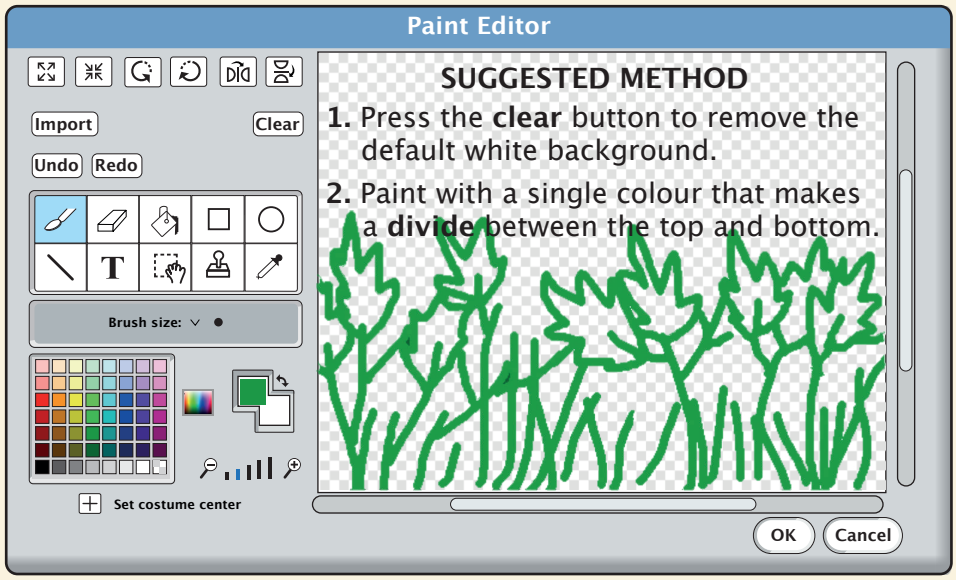
2.3.6 The bee is back.

2.3. end

2.1. return

Save as *task20.sb*

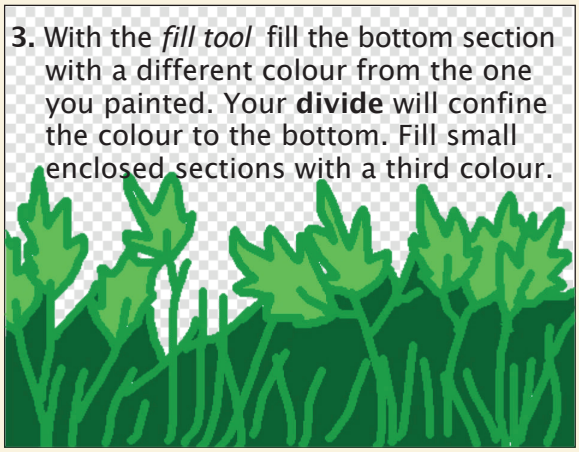
Skill 16 Painting a Background in Paint Editor



A. (i) **In Scratch click the *Stage Icon*** (it's on the left of the *sprite list*).
(ii) **Select the** *Backgrounds* tab.
(iii)**Click** *Edit* to get the Paint Editor.

B. Paint a background
Tools useful in painting a background are:

Paintbrush
Fill tool
Line tool
Stamp tool



If you are able to create you own painted background in *Paint Editor*, try replacing the imported background with your own in Task 20.

Task 21 Make a Game: *It's Raining Bones*

The idea behind this game:

Bones keep falling randomly from the sky. The player moves a plate *left* and *right* using the *arrow keys*, to collect 10 bones to win.

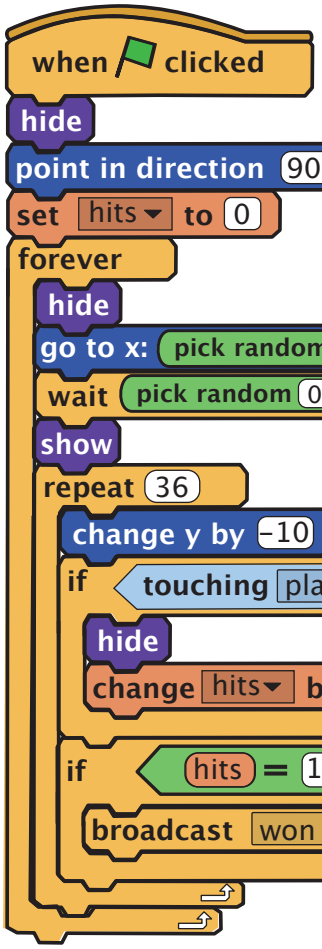
1. Create three sprites, a bone, a plate and a text sprite with the words "You've WON!".
2. The main script will be on the bone sprite. Try scripting it yourself before turning this page to see the completed bone script. But before that, **script the plate and the text sprite. There are four scripts on the plate.**
3. Programme the plate to be visible and the text sprite to be invisible, when the *green flag* is clicked. In a separate script, make the plate to [move -20 steps] using the left arrow key and [move 20 steps] in a positive direction using the right arrow key.
4. When scripting the bone, we will use a [broadcast] to tell the plate to disappear and to tell the text "You've WON!" to make its appearance. We will give the broadcast the name '*won*'. Make the plate [hide] and the text sprite [show] when they [receive '*won*'].
5. Make a variable and give it the name '*hits*'.

The completed script on the bone sprite is on the next page.

continued from previous page

It's Raining Bones

In the game plan, it would seem that we need several bones, but there's only one. Here's the basic *bone* script.



1. One click of the green flag will have an effect on several sprites simultaneously (the bone, the plate and the text sprite).

1.1 The bone is hidden for an instant when the green flag is clicked.

1.2 The sprite is pointed to the right.

1.3 The *hits* count (variable) is set at zero.

1.4 This [forever] loop keeps the bones falling continuously.

1.4.1 As each loop begins, the bone sprite is invisible. (**Q**: Why are there 3 [hides]?)

1.4.2 It goes to a random **x** point at the top of the stage.

1.4.3 It falls instantly or waits 1 or 2 seconds before falling.

1.4.4 Just before falling, the bone sprite appears at the top of the stage.

1.4.5 This [repeat] loop will run 36 times before the next [forever] loop.

1.4.5.1 The bone falls 10 pixels (steps) in each loop. 36x10 = 360

1.4.5.2 **IF** the player 'catches' the bone with the plate ...

1.4.5.2.1. ... the bone disappears (**Q:** When will it reappear?)

1.4.5.2.2. ... a *hit* is recorded

1.4.5.3 **IF** the player 'catches' the **10th bone** ...

1.4.5.3.1 the message 'won' is broadcast to the text sprite to appear

In what ways could you improve the game to make it more interesting?

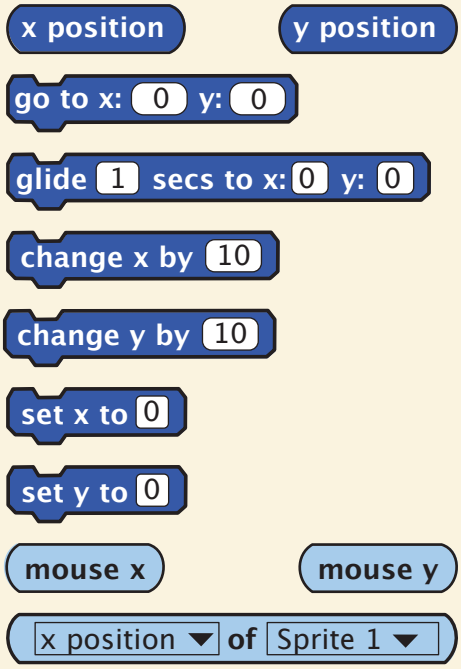
Save as *task21.sb*

Skill 17 Understanding x and y in Scratch

MOTION and SENSING blocks with x and y

x: 0 y:0 direction: 90

x: 80 y:5



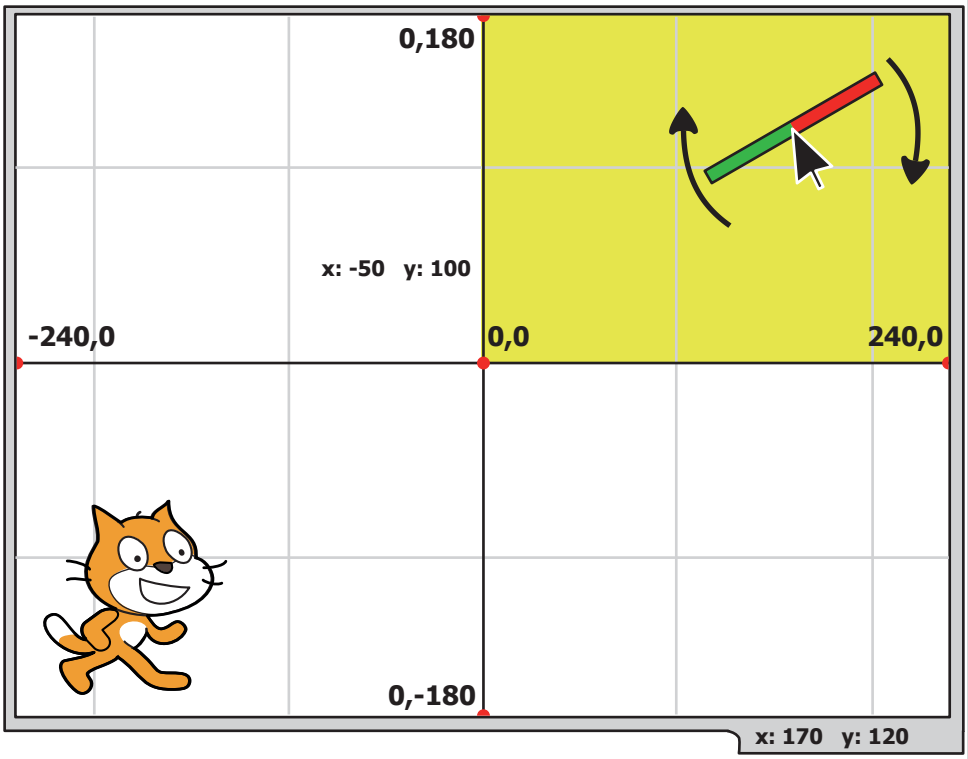
There are two places where **x** and **y** values are shown in the interface. One gives you the exact point of the tip of the mouse-pointer, the other gives you the exact location of the centre of the *current sprite*. When you want to give a sprite a starting position or when you want to direct it *how to move* or *where to move to*, you use the *Motion* and *Sensing* blocks with **x** and **y**.

In **Task 21** you gave the bone a random starting point at the top of the stage using the [go to] block. Then you used the [change y] block to make the bone fall. You used *minus values* in both of the blocks. **Question: Why did you use minus values?**

At this stage, you need to understand that numbers less than *zero* are *negative* (or *minus*) numbers. Numbers that are greater than zero are *positive* (or *plus*). You also need to understand that a sprite can move in either a *positive* or *negative* direction (up, down, forwards or backwards). Refer back to **page 8** to revise what you know about every point on the stage. If you think you understand your **x**'s and **y**'s then you will enjoy the next task.

Task 22 A two-way spinner

When the stage is sectioned into four quarters, with perpendicular lines going through the centre-point **0,0** there is only one section of the 4 where the **x** and **y** of each point is *positive*. In each of the other three sections, either the **x** or the **y** or both the **x** and **y** are *negative*.



You might find all of that difficult to understand, so this task will make it fun and help to explain it!

We want to programme a sprite to spin clockwise **IF** the mouse position is *positive* for both **x and y**. **ELSE** we want it to spin anticlockwise.

1. Use the default cat sprite or draw a simple spinner in the *Paint Editor*. Set the sprite's *centre of rotation*.

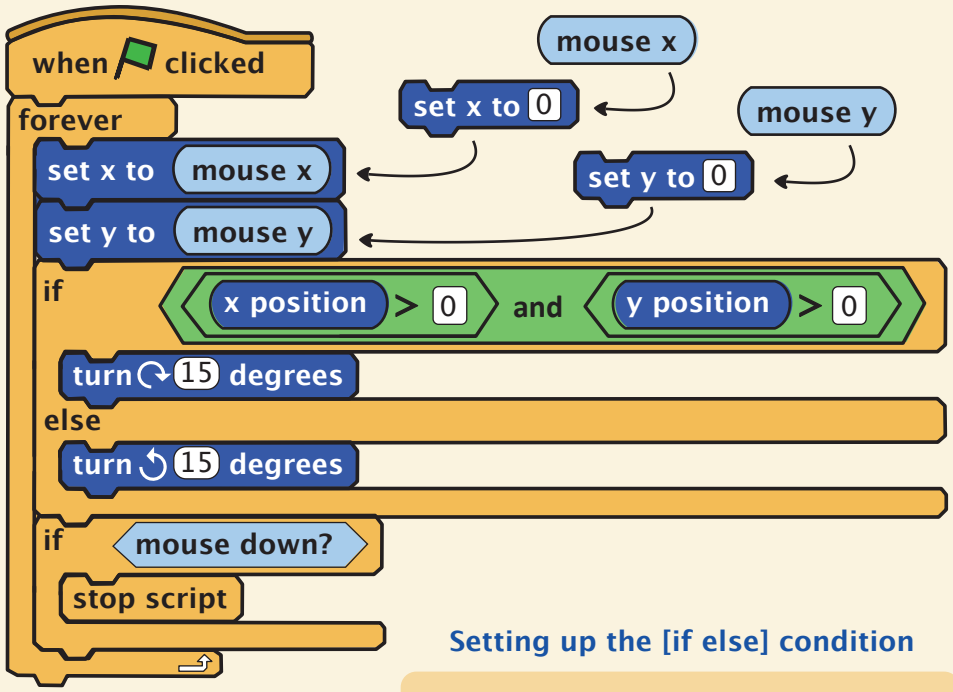
2. To make the sprite follow the tip of the mouse-pointer drop the [mouse x] reporter into the [set x to] block and do the same for **y**.

3. Setting up the [if else] condition is not as easy. **(i)** drop the blue [x position] and [y position] reporters into the green [greater than] blocks. **(ii)** Drop the two [greater than] blocks into the [and] reporter. **(iii)** Drop the [and] block into the hole in the [if else] block. **(iv)** Make the sprite turn 15° clockwise **if** the first condition is met, **else** make it turn 15° anti-clockwise.

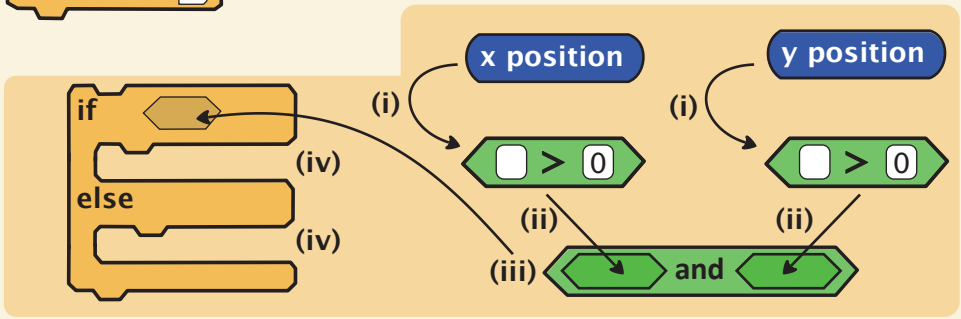
Remember: The symbol > means *greater than* (the open end is greater than the closed end.)

4. The second [if loop] lets you click the mouse to stop the spinner.

Save as *task22.sb*



Setting up the [if else] condition

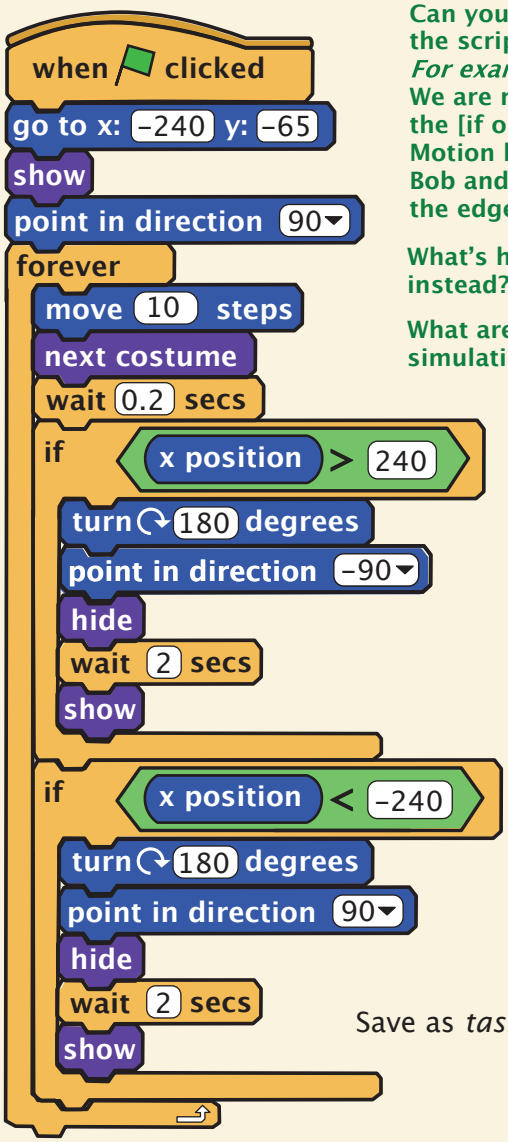


Task 23 A Boy and his Dog – the Movie

You are going to create a movie with a boy and his dog out walking. The boy appears from the left side of the stage and his dog comes along *two seconds* behind him. Start by importing or drawing a new background. We have drawn our background in *Paint Editor*. Follow the steps below. You now have more thinking to do with less help.

1. From the *People* folder, choose a new sprite **boy4-walking-a**. There are four more costumes for **boy4 -walking**. Import all of them. Use the *shrink* button to reduce his size. We have called him Bob. You can give him whatever name you like.

2. Give the Bob the following script and test it out. You might need to adjust the **y** value, as it depends on the size of the sprite and which is the most appropriate position to place Bob in your background.

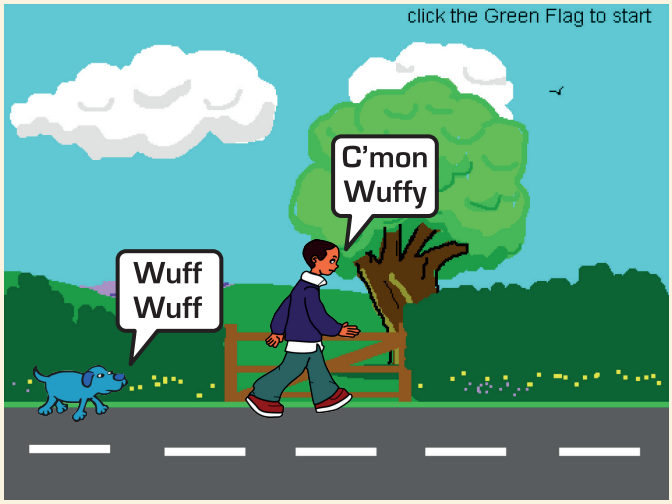


Can you interpret the script?
For example:
We are not using the [if on edge] Motion block when Bob and Wuffy reach the edge of the stage.

What's happening instead?

What are we simulating?

Save as *task23.sb*



3. You will find **dog2-b** in the *Animals* folder. Import **dog2-c** as a second costume. We have called the dog Wuffy. You can call him whatever name you like. Shrink him in proportion to Bob.

4. Copy Bob's script to Wuffy, for a start. Change Wuffy's **y** value so that he appears to walk on the same horizontal plane as Bob. Now for the movie-making fun!

5. Bob and Wuffy appear at the same time. Your first task is to **make Wuffy appear two seconds behind Bob**. How do you achieve that?

6. Bob doesn't like it when Wuffy fails to keep up with him. You want Bob to say "C'mon Wuffy." but just once! Make Wuffy reply a few times with "Wuff. Wuff." and then stop him. Which of these code blocks suits your needs. Experiment.

How will you decide *when* Bob says his words? Can you create a short pause before Wuffy barks his reply?

7. At some point, you might want to make Wuffy walk past Bob. Wuffy's way of showing off is to wag his tail.

Firstly, how do you programme Wuffy to walk past Bob ?(Wuffy should still be two seconds behind Bob at the start.)

Secondly, how do you make Wuffy wag his tail? Remember, he only wags his tail after he walks past Bob (and not before that).

8. Add one extra feature to the movie using the skills you have learned up to this point **or** see if you can add a sound using Sound block (a). Add more sound to your movie if you like.

Sound Blocks

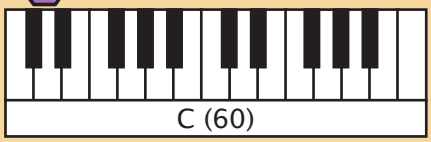
a play sound meow — dynamic list *e.g.* Imported sounds and recordings can be used.

b play sound meow until done — dynamic list

c stop all sounds

d play drum 48 for 0.2 beats — fixed list

e play note 60 for 0.5 beats — fixed list



f rest for 0.2 beats

g set instrument to 1 — fixed list

h change volume by -10

i set volume to 100 %

l ☐ volume

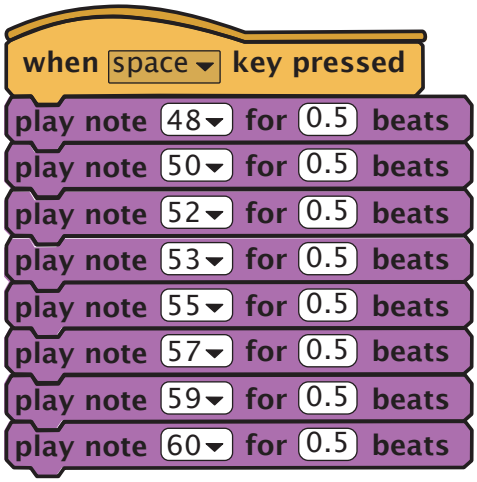
j change tempo by 20

m ☐ tempo

k set tempo to 50 bpm

Task 24 Explore the Sound Blocks

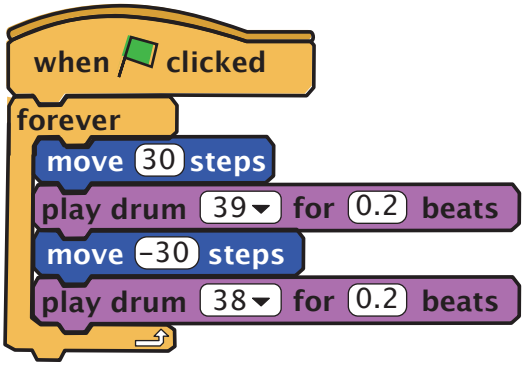
(a) Try out the music scale. Block (e)



Experiment with different instruments. Block (g)

Experiment with rests, volume and tempo using Block (f) and Blocks (h) to (k).

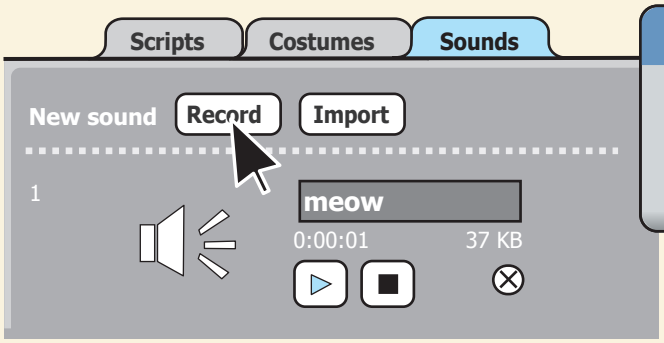
(b) Import a background and a dancing sprite, such as Anjuli, Cassy, Dan or the Breakdancer. All can be found with several costumes in the *People* folder.



Experiment with drums. Block (d).

Skill 18 Recording in Scratch

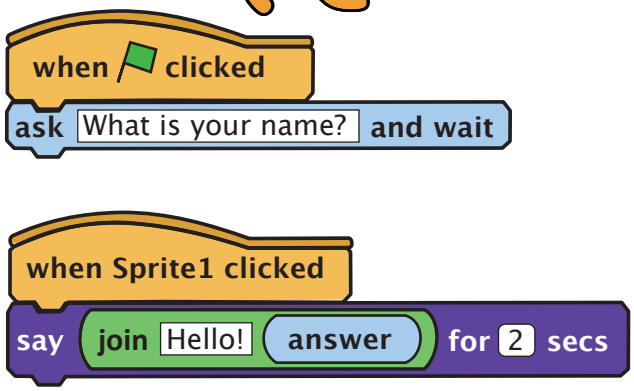
Scratch has its own sound recorder under the *Sounds* tab. If your laptop does not have an inbuilt microphone you will need to attach an external device to use the recorder.



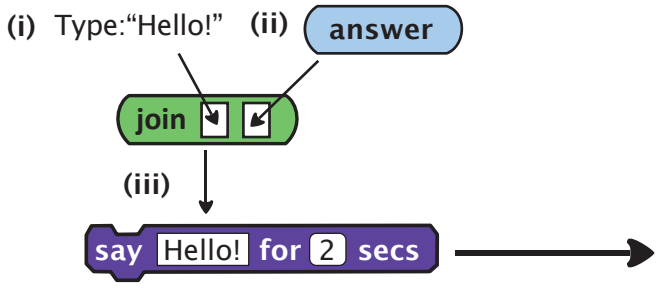
You can add sounds to sprites or to the background.

Task 25 Ask and Answer

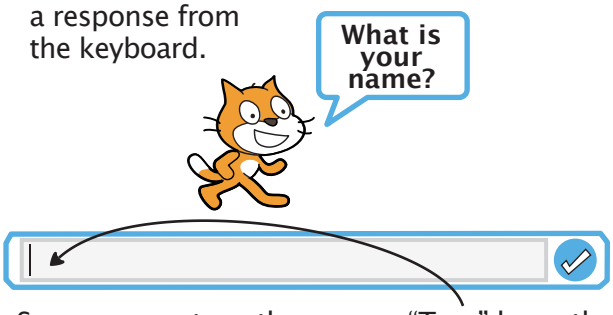
It is quite common that a computer programme communicates with the user by requiring some form of user input. Scratch is no different. Set up the following script on the default sprite and then think about how it all works.



How to make the composite [say] block



1. When you use the [ask] *Sensing* block, it gives the sprite a speech balloon with the value (or words) you have typed into it. In this task you typed "What is your name?". It also makes a text input box appear at the bottom of the stage. The cursor is ready for the user to type a response from the keyboard.



Suppose you type the answer "Tom" here, then click the check mark or press the Enter key.

2. When you typed your response, you created a special variable. This variable is in the *Sensing* palette (not in the *Variables* palette). It has a single purpose, which is to store the keyboard input. It is given the name "*answer*" and it has its own [answer] reporter block. After you typed your answer to the question, "*answer*" has the value "Tom"

3. To put the variable to use, we have made the cat sprite greet you by name. When you click the cat sprite it says "Hello! Tom" for two seconds. The [join] block makes it possible to combine the [say] block with the [answer] reporter. There is more about the [join] block on page 32.

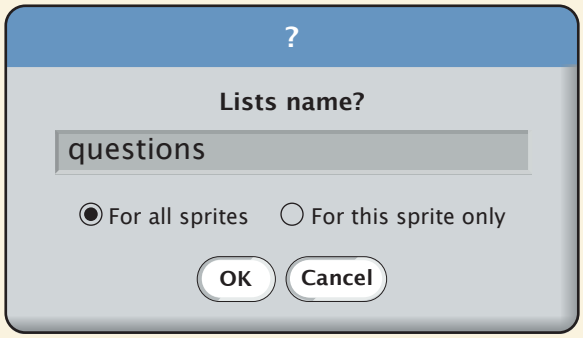
4. Note: The value in the *answer* variable changes each time the user is 'asked' for input.

Save as Task25.sb

Variables List Blocks

Variable Blocks **a** to **e** can be seen on page **19**.

Skill 19 Understanding Lists



Variables and Lists are in the same palette, because lists are somewhat like variables. A variable can hold only one value at a time. A list can hold several values, all at the same time. When you click the [Make a list] button the next thing you do is to give your list a name and click OK.

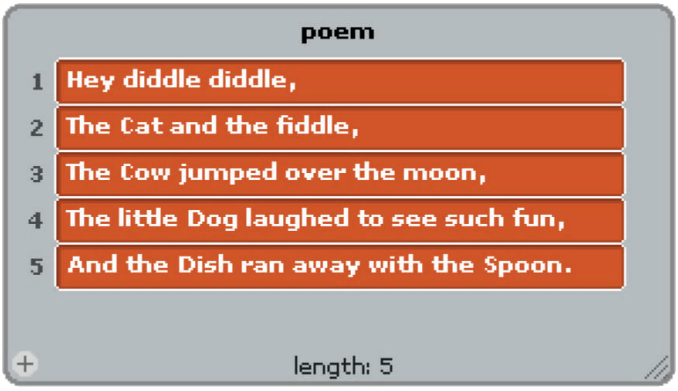
empty list monitor

Resize the list monitor by dragging from bottom right

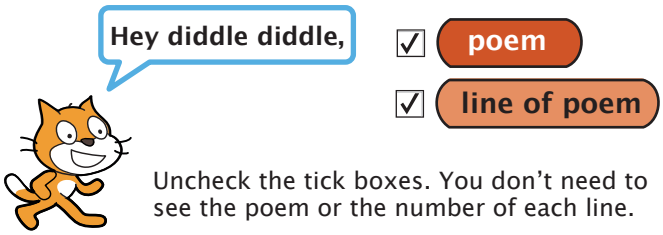
Immediately all the List blocks (f) to (m) appear and a list monitor pops up on the stage. You can type directly into the list monitor. Let's suppose you have given the list the name *questions*. You have not added any *items* to *questions* yet so the list is empty, with *length* 0. To add *items* to *questions*, click the plus + button on the bottom left of the list monitor. If you have pre decided that you are going to add ten questions to *questions* you could click the + ten times to get ten empty input boxes. As you fill in the *items*, a scroll bar appears on the right of the monitor.

Alternatively, you can add *items* (things) to the list using block (g), the [add item] block. Type in the small input window and click the block. You can right-click on a list monitor and *import* a list you have already prepared as a plain *.txt* file. We will fill in the list directly, using the *list monitor*.

Task 26 Make and use a list

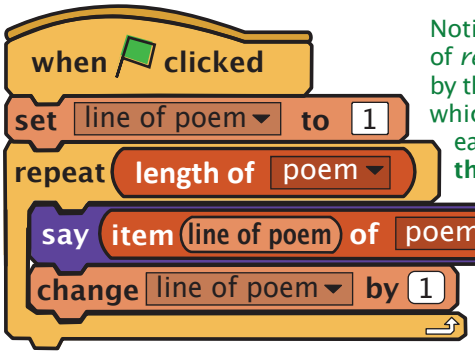


1. Create a new project. Click *Variables* and click *Make a list*. Give your list the name *poem* and click OK.
2. When you look in the bottom left corner of the *list monitor* that has just appeared on the stage you will see + (plus) sign. Click the + and you will get an orange-coloured input box. Click in the box and type "*Hey diddle diddle,*". Stretch and resize the *list monitor* as you need.
3. Continue to type the remaining lines of the poem. Use the scroll bar and resizing handle as necessary. Use the tab key to move on to the next input box.
4. Create a *variable* and call it *line of poem*. Now you have two separate sets of of blocks, one for the *variable* and the other for the *list*.
   At this stage, save your work as *task26.sb*



Uncheck the tick boxes. You don't need to see the poem or the number of each line.

5. Add the following script to the cat sprite then save again and run the programme.



Notice how the number of *repeats* is determined by the *length* of your list, which is **5**. The cat says each *item* – **a line of the poem** – while the loop iterates through **1** to **5**.

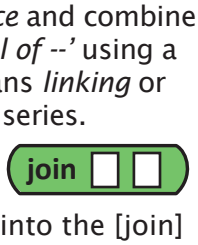
Make the cat introduce himself with the words:
"Welcome to my poetry recital."
and make him sign off with the words:
"Thank you for listening."

Task 27 Create an easy quiz

When creating a quiz, all you need do is make two lists of words. Give one list the name *questions* and the other the name *answers*. A list *item* can be a whole sentence, if you like, but for this quiz we will get by with a single word for each question and a single word for each answer.

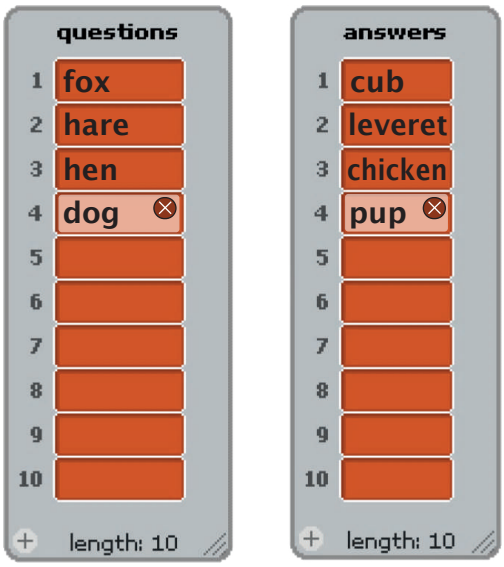
Eg. Your quiz master could ask the question:
"What is the capital city of **France**?"
Scratch will select the list item *France* and combine it with the words '*What is the capital of --'* using a method called *concatenation.* It means *linking* or *joining* words together in a chain or series.

The main block used for this is the [join] *Operator*. You can type text or insert other blocks directly into the [join] block. You can even insert one [join] block into another and into another again!



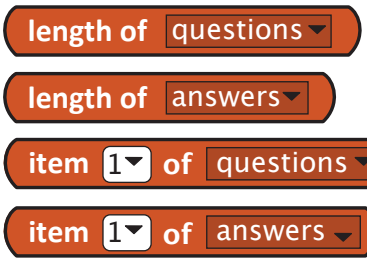
The plan is to make the cat sprite the Quiz Master. The cat will ask a question and the player will type in a response and either click the check mark or press ENTER.

1. Create two lists. One list is a set of questions. The other list is a corresponding set of answers. For the example, we have created a list called *questions* and we populated the list with the names of ten common *animals*. Our second list is called *answers* and it is populated with the names of the *young* of each of the ten animals.



It is important that both lists be the exact same *length* and that the *index number* of each question should correspond with the *index number* of each answer.

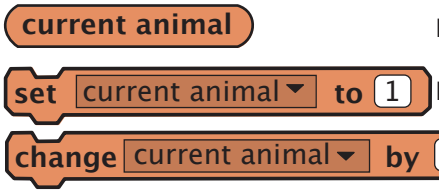
After you create the lists you have a set of list blocks. The blocks [length of list] and [item of] have a small triangle to open drop down lists with the names *questions* and *answers*. When you see the *length of questions* or *length of answers* understand that *length* means the number of *items* that are in each list.



block (l)
block (l)
block (k)
block (k)

You can see below❋, that you can drop the reporter block (a) into block (k).

2. Create a *variable* to keep track of each question as it is asked. We have named our sample variable *current animal*. The following three blocks will be required in the single script we create.



block (a) p.19
block (c) p.19
block (b)

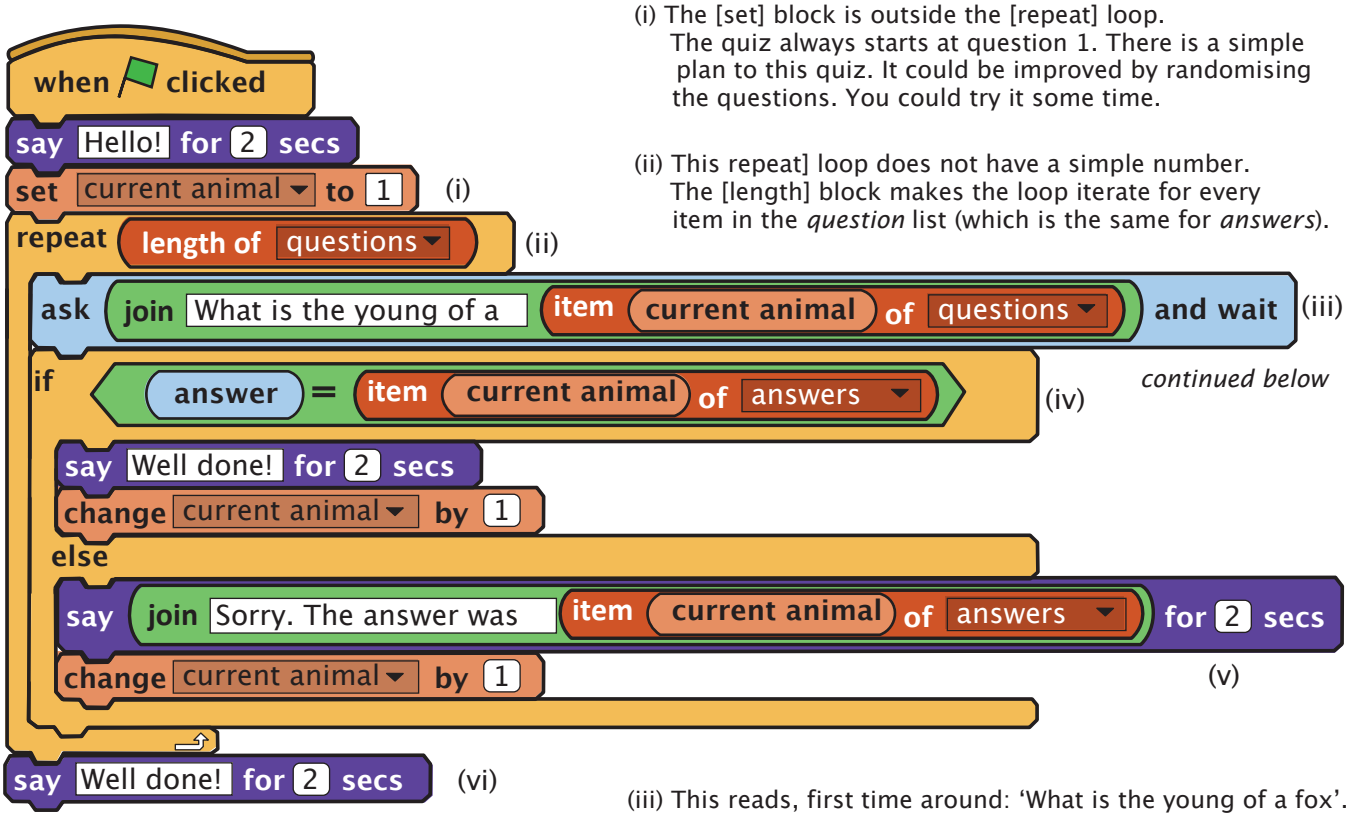
At the start of the quiz, *current animal* is animal number 1. In the second loop *current animal* is animal number 2. etc.

❋ 

This combination means *question number one, question number two etc.* as it is *iterates* through the loop.

Task 27 Create an easy quiz (continued)

3. Put it all together in one script.
Try to make sense of each block of code.



(i) The [set] block is outside the [repeat] loop. The quiz always starts at question 1. There is a simple plan to this quiz. It could be improved by randomising the questions. You could try it some time.

(ii) This repeat] loop does not have a simple number. The [length] block makes the loop iterate for every item in the *question* list (which is the same for *answers*).

(iii) This reads, first time around: 'What is the young of a fox'. The [ask] block is a combination, all made possible with just one [join] block. **Always** follow your text with a space, otherwise the concatenated text will be joined into your typing without a space. There's a missing **question mark**. The question mark would have required another [join] block. See if you can fix that problem.

(iv) In Task 25 you understood that when you type your reply in the input box, your reply is stored as a variable (which is overwritten when you type your next reply). The [if] condition enquires 'if your answer is equal to the current answer in the *answer* list.' If it is, the script moves on to say 'Well done'! and moves on to the next question.

(v) If you gave the wrong answer the cat sprite will say 'Sorry. The answer was cub.' or whatever. As this is a simple quiz, answers like *puppy* or *chick* will be regarded as incorrect, if the list contains *pup, chicken.*

(vi) The final 'Well done!' is after the round of questions as it's outside the loop.

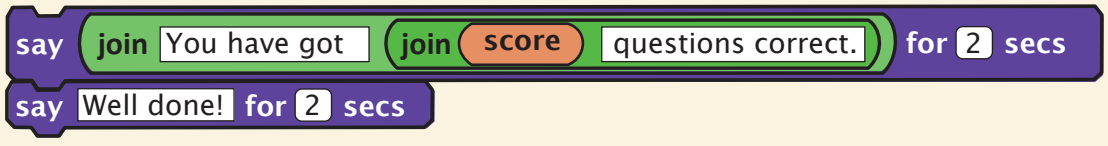
Adding a score

To add a score all you need do is create a new variable. Give it a suitable name *e.g.* **score**

You have to *set* the starting value of *score* to zero. You have to decide where to put the [set] block. You could give ten points for each correct answer.

Where do you put your code to *change* the score on a correct answer? What do you do in the event of an incorrect answer.

Your final code blocks would look something like the ones below but would they be inside or outside the [repeat] loop?



How else could you make your quiz more interesting?

Pen Blocks

a clear

b stamp

c pen down

d pen up

e set pen color to

f change pen color by 10

g set pen color to 10

h change pen shade by 10

i set pen shade to 50

j change pen size by 1

k set pen size to 1

Skill 20 Using the Pen Blocks

The Pen is probably the easiest Scratch tool of all to use. We have left it until last because when you know how to use all the other scripting tools and skills it increases your enjoyment of drawing with the pen. Now you can actually programme the pen to make it do what you want. You can easily explore all sorts of shapes and designs with the Pen blocks. You can write, draw freehand, create regular shapes and surprising patterns.

The [clear] block is like an eraser, as it clears all pen lines and drawings.

The [stamp] block is great when you want to copy a sprite on the stage. You cannot add script to a stamped copy!

Over the following tasks you can get plenty of practice with all the pen blocks and be as creative as you like.

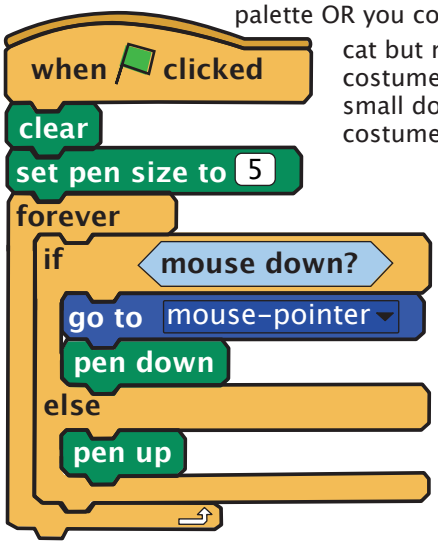
(a) Experiment in task 28(a), with pen colour, pen shade and pen size blocks.

(b) Extend task 28(b): Which letters of the alphabet, consisting of straight lines only, can you draw with the pen?

Task 28 (a) Write and draw freehand

When you try out this script you will see that it's not easy to draw with a cat-shaped pen. To correct this problem, you could make the cat disappear using use the [hide] block from the Looks palette OR you could keep the cat but make a new costume consisting of a small dot. Use the new costume as your pen.

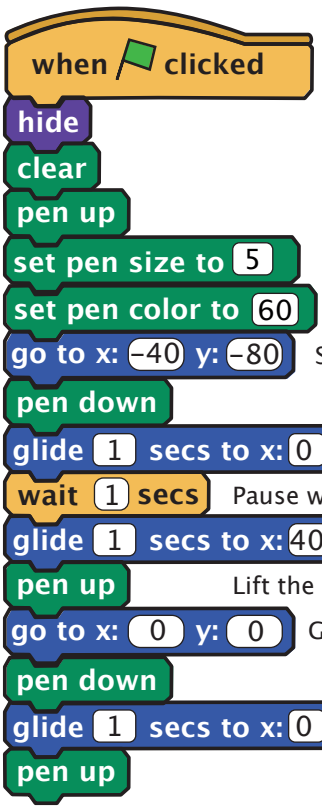
hide



Where in the script is the best place to put the [hide] block?

The pen, like all script must be on a sprite, but you don't need to see the sprite to use the pen. The pen has to be down if you want to see what what you're drawing. Use the [clear] block to start with a clean slate each time.

Task 28 (b) Draw straight lines



Drawing from one x,y to another x,y doesn't require turning or change of direction.

Starting at this point

Draw to this point

Pause without lifting the pen

Draw to this point

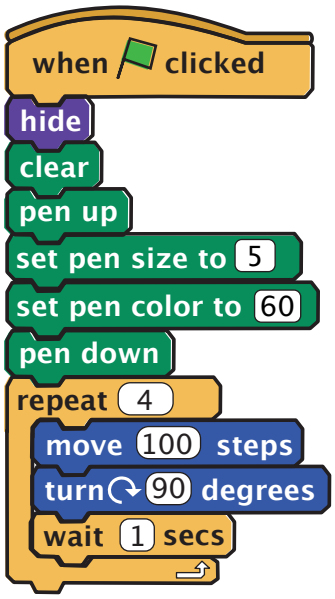
Lift the pen

Go to a new starting point

Draw to this point

Which capital letter do you draw with this script?

Task 29 (a) Draw regular shapes



Which values in this script tell you that it makes a square?

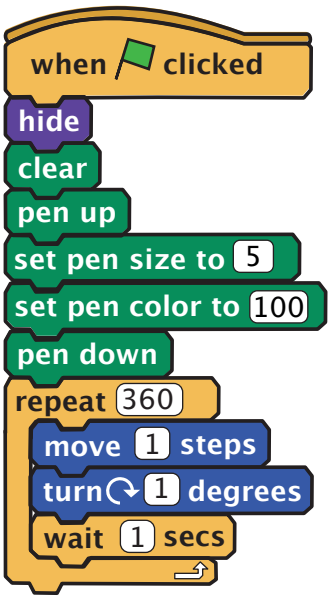
What would you change to make a triangle or a hexagon?

Where in the script could you put the [pick random] block to make your shape-drawing programme more interesting?

Pen colour codes

Numbers from 0 – 200 are a tint of a colour. Have fun exploring colour with the [pick random] blocks .

Task 29 (b) Draw the circumference of a circle



Which value is responsible for the size of the circle?

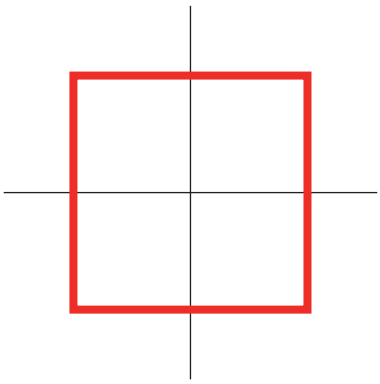
How would you make the circle bigger or smaller?

How would you make a semi-circle?

Task 29 (c) Shapes and Repeating patterns

Can you programme the pen to draw these pictures?

(i) Programme the pen to draw a zig-zag pattern. Keep the code short by using a [repeat loop].



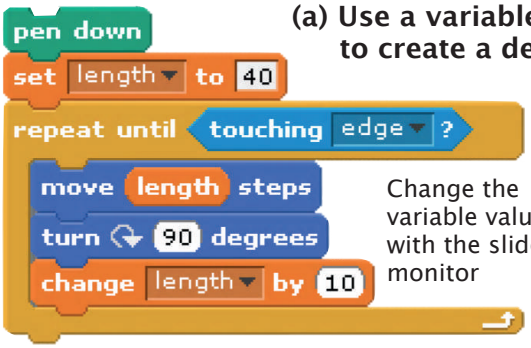
(ii) Draw a square that is centred on the point 0,0

(iii) Draw the letter 'R' with a semi-circle in the round part.

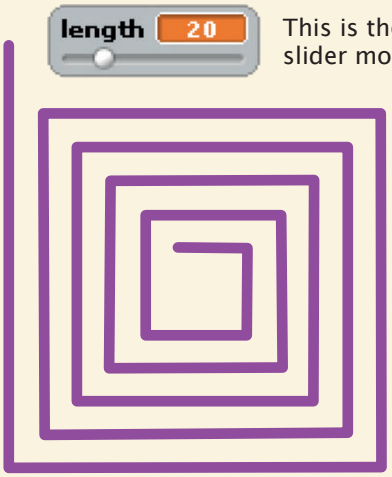
Task 30 Patterns and Spirals

Missing code: Complete the code that goes before the [pen down] block in tasks (a) and (b).

(a) Use a variable to create a design

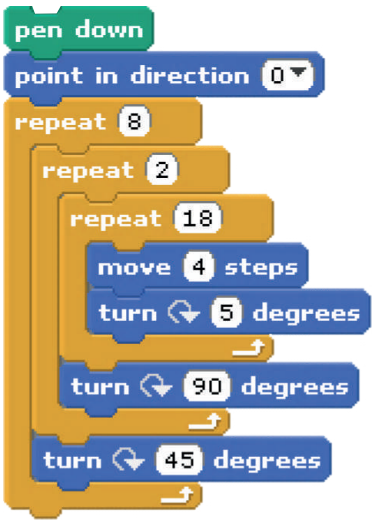


Change the variable values with the slider monitor



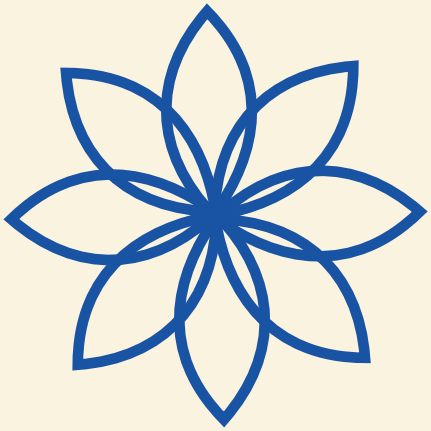
This is the slider monitor

(b) Put a loop inside another loop

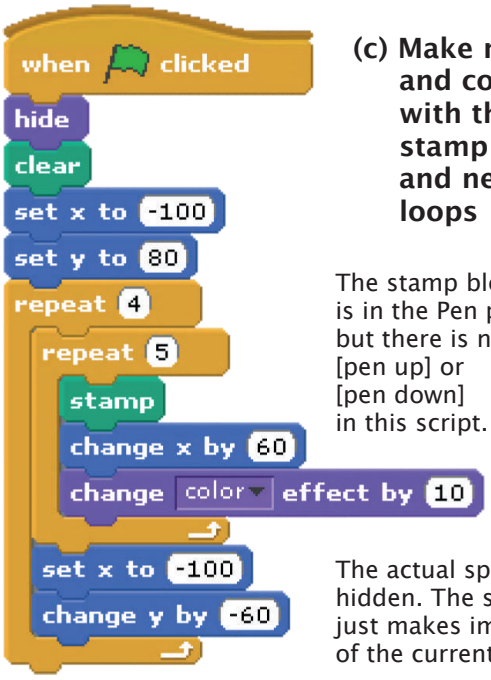


A loop inside another loop is called a nested loop

Experiment with the steps and degrees of turning

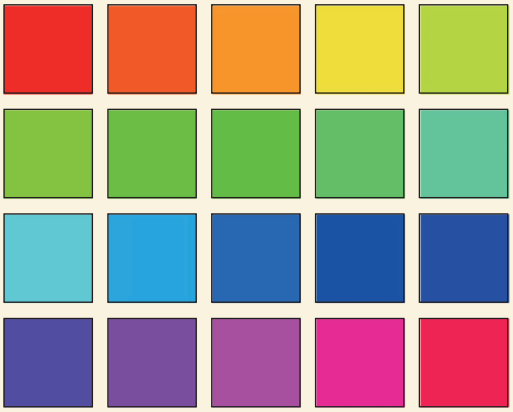


(c) Make rows and columns with the stamp block and nested loops



The stamp block is in the Pen palette but there is no [pen up] or [pen down] in this script.

The actual sprite is hidden. The stamp just makes images of the current sprite.



Scratch from Scratch is the perfect way to start learning an exciting programming language aimed at younger learners. We've written the book to be accessible to young and old, but primarily with the teacher or mentor in mind. As this book is an evaluation edition of *Scratch from Scratch,* it is a limited run and just the first part of a planned three-part text. This edition covers the *Tools and Skills of Scratch.*

Our next edition will comprise all three parts; *The Tools and Skills of Scratch, Numeracy in Scratch & Computational Thinking in Scratch.* We hope that teachers, parents and mentors of children will use this book to encourage younger learners to try computer programming. We really hope you enjoy this text, and use it in your computer club or classroom.

ISBN 978-0-7144-1903-9

This book is a limited edition. Copies can be obtained by e-mailing or texting. E-mail your order to: seamus@weandus.ie stating also your name, address and contact number. Please send your text order to: 087-2308320 or 087-8245959 Price per copy is €15 +p&p.