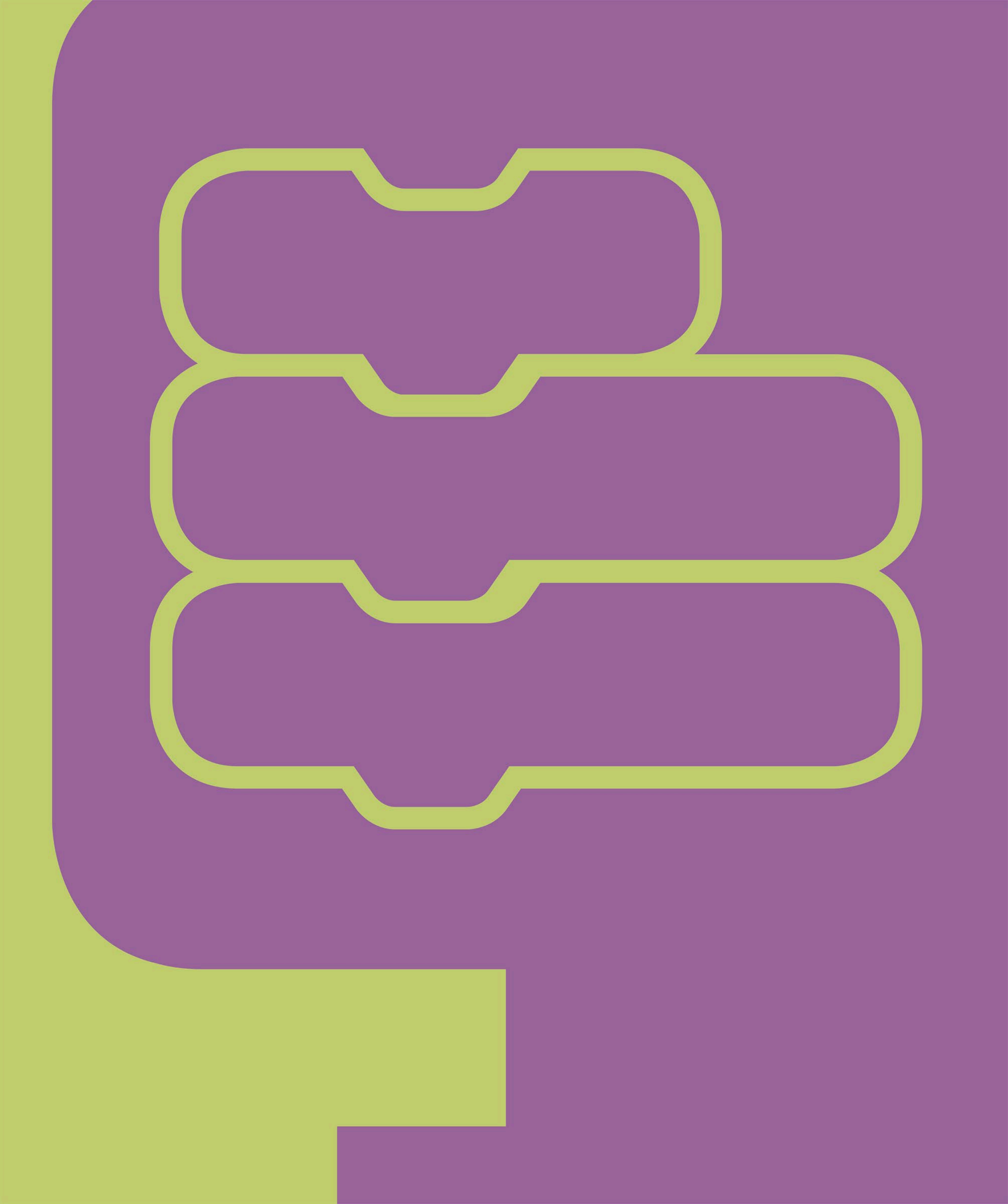


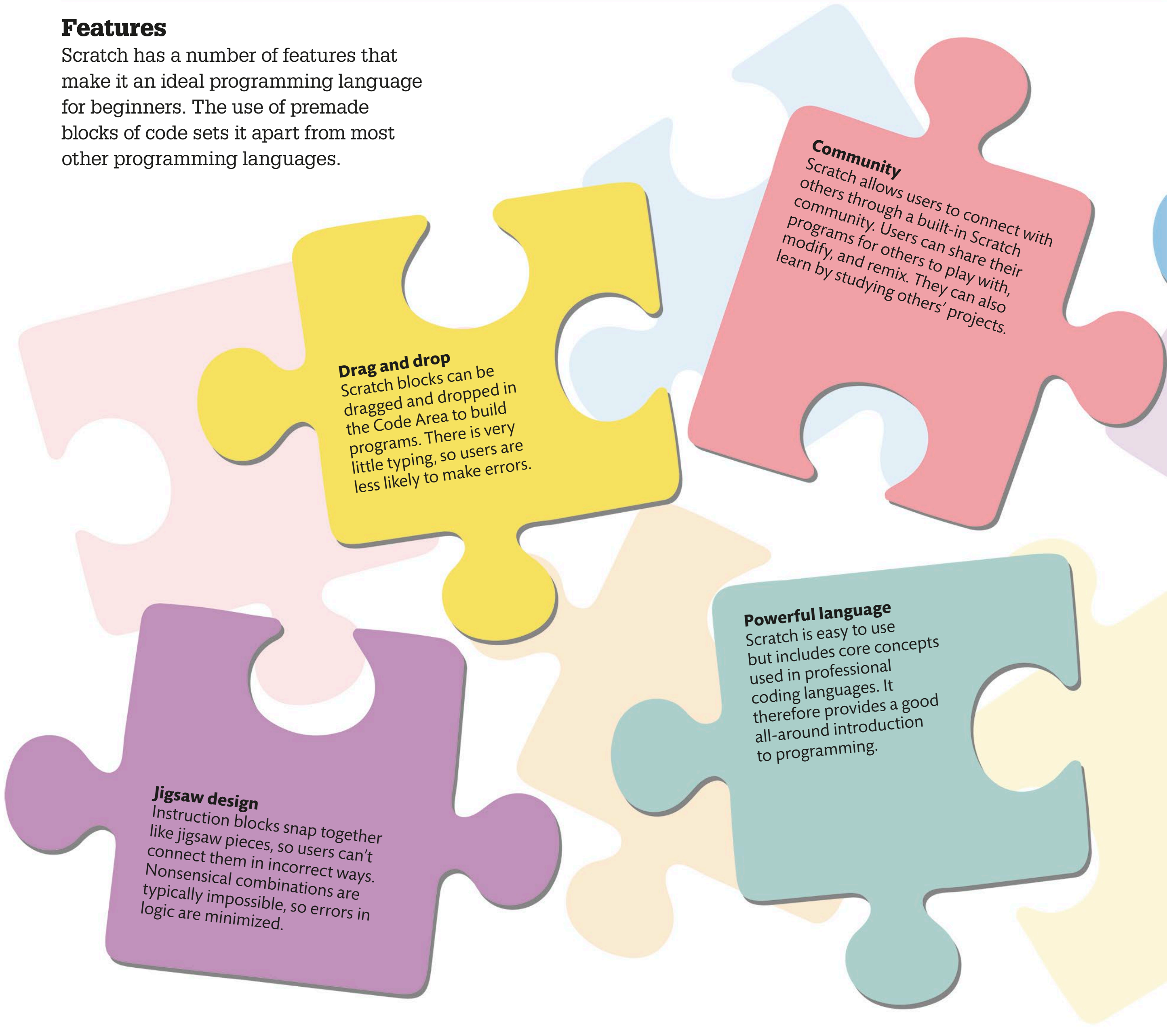
SCRATCH

What is Scratch?

Scratch is a visual programming language that does not require users to type code. Instead, they build programs using colored blocks that represent instructions. Scratch focuses on the creative aspect of coding and allows users to create interactive games, stories, and other visual applications.

Features

Scratch has a number of features that make it an ideal programming language for beginners. The use of premade blocks of code sets it apart from most other programming languages.



Drag and drop
Scratch blocks can be dragged and dropped in the Code Area to build programs. There is very little typing, so users are less likely to make errors.

Community
Scratch allows users to connect with others through a built-in Scratch community. Users can share their programs for others to play with, modify, and remix. They can also learn by studying others' projects.

Jigsaw design
Instruction blocks snap together like jigsaw pieces, so users can't connect them in incorrect ways. Nonsensical combinations are typically impossible, so errors in logic are minimized.

Powerful language
Scratch is easy to use but includes core concepts used in professional coding languages. It therefore provides a good all-around introduction to programming.

Learning to program with Scratch

Scratch was created by the Lifelong Kindergarten Group at the Massachusetts Institute of Technology (MIT). It was first launched in 2007.

Scratch was designed to be fun and easy to use for beginners and to help them understand basic concepts and avoid errors. It is therefore widely used in education. Scratch has a highly visual interface with colored blocks of code that join together to form scripts, which can include images and sounds to create action onscreen. Scratch provides a powerful platform (see pp.30–31) for exploring programming.

Built-in assets
Scratch comes with a preinstalled library of sounds and images (called sprites—see pp.32–33) that makes it easy to start coding right away. Other programming languages lack in this regard, as images need to be created or uploaded before writing a program.

Color coding
Instruction blocks for movement, sound, control, and sensing (among others) are color coded, so they can be easily identified and read when creating a program.

Hardware support

The latest version of Scratch works on computers with Windows, macOS, and Linux. It can even be used on tablets. Scratch projects can use extensions to interact with hardware devices.

Raspberry Pi
Scratch can use a Raspberry Pi to connect to other sensors or motors.

micro:bit
Scratch can be used with a BBC micro:bit, which has a built-in LED display, buttons, and tilt sensors.

Lego®
Scratch can connect to Lego® WeDo and Lego® Mindstorms™ to work with motors, sensors, and robots.

Webcam
Scratch can access a webcam to layer images on a live video feed to create simple augmented reality applications.

GETTING SCRATCH

The Scratch developer environment is required for using Scratch. It can be accessed both online and offline.

In your browser
Visit the Scratch website at *https://scratch.mit.edu/* and click Join Scratch to create an account.

Offline
Scratch can be downloaded and used without an internet connection at *https://scratch.mit.edu/download.*

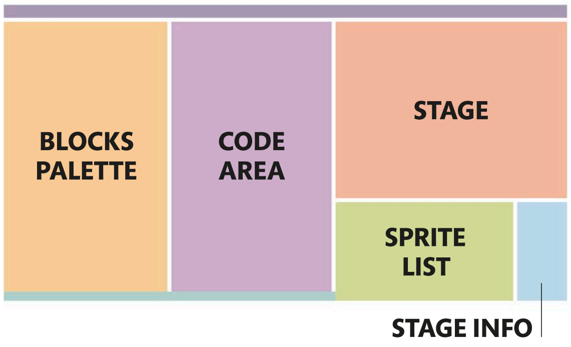
Scratch interface

The screen layout, or interface, in Scratch can be used to build programs, edit them, and view the output in the same screen. The interface is divided into several sections, each serving a particular purpose. This book uses Scratch 3.0—the latest version of Scratch.

Understanding the screen layout

The Scratch interface is divided into the following sections:

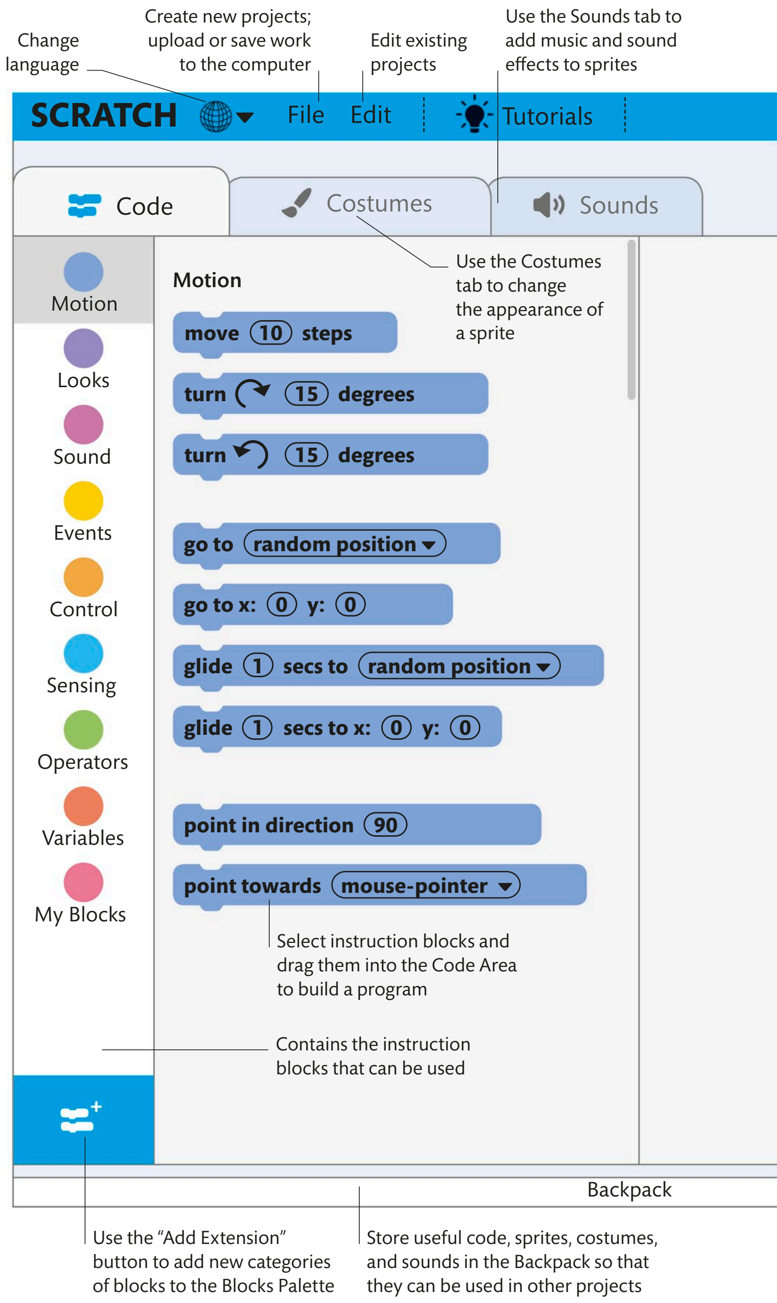
- **Blocks Palette:** This contains the instructions, or code blocks, required to build a program.
- **Code Area:** The instruction blocks are assembled here to create a script.
- **Stage:** Allows the user to interact with the program.
- **Sprite List:** Displays and manages all the images, or sprites, used in a program.
- **Stage Info:** Manages the background images.



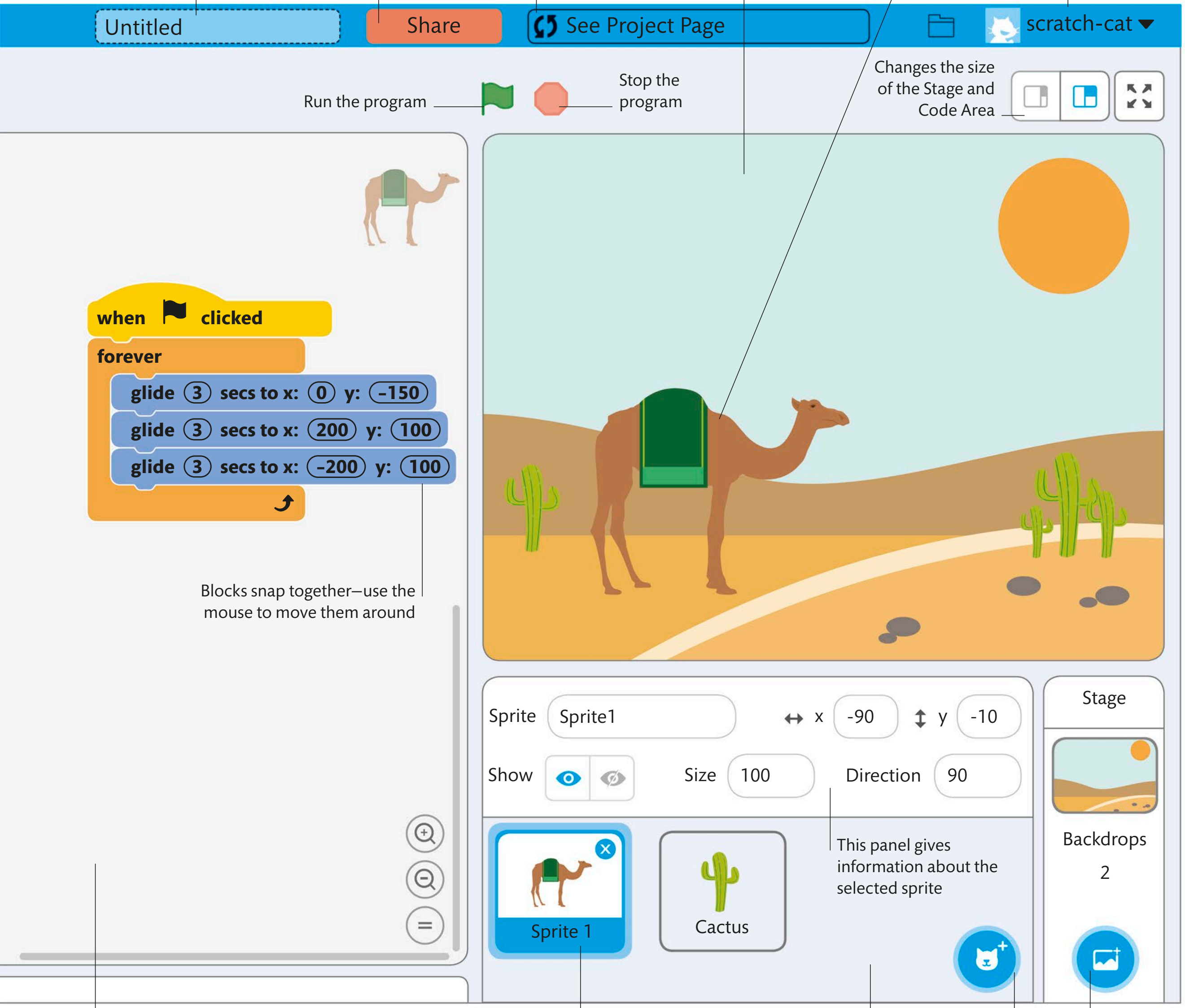
VERSIONS OF SCRATCH

There have been three versions of Scratch so far, each with a different screen layout. New features and instruction blocks were added at each update of the interface. These features may not work on earlier versions.

- **Scratch 1.4:** The interface was similar to Scratch 3.0, but the Code Area was called Scripts Area.
- **Scratch 2.0:** The Stage was on the left of the screen. Introduced sprite cloning and reorganized some blocks into the Events category.
- **Scratch 3.0:** Introduced Blocks Palette extensions and moved the Pen blocks into them.



Change language
Create new projects; upload or save work to the computer
Edit existing projects
Use the Sounds tab to add music and sound effects to sprites
SCRATCH
File
Edit
Tutorials
Code
Costumes
Sounds
Use the Costumes tab to change the appearance of a sprite
Motion
Looks
Sound
Events
Control
Sensing
Operators
Variables
My Blocks
Motion
move (10) steps
turn ↻ (15) degrees
turn ↺ (15) degrees
go to (random position ▾)
go to x: (0) y: (0)
glide (1) secs to (random position ▾)
glide (1) secs to x: (0) y: (0)
point in direction (90)
point towards (mouse-pointer ▾)
Select instruction blocks and drag them into the Code Area to build a program
Contains the instruction blocks that can be used
Backpack
Use the "Add Extension" button to add new categories of blocks to the Blocks Palette
Store useful code, sprites, costumes, and sounds in the Backpack so that they can be used in other projects



Name of the project

Share projects with the Scratch community

Access the community page of a project

The Stage shows the sprites moving and interacting with each other when a project is run

Click or tap a sprite on the Stage or in the Sprite List to select it

Edit profile and access saved projects

Run the program

Stop the program

Changes the size of the Stage and Code Area

```
when ⚑ clicked
forever
    glide (3) secs to x: (0) y: (-150)
    glide (3) secs to x: (200) y: (100)
    glide (3) secs to x: (-200) y: (100)
```

Blocks snap together—use the mouse to move them around

This panel gives information about the selected sprite

Sprite: Sprite1 ↔ x: -90 ↕ y: -10
Show Size: 100 Direction: 90

Drag instruction blocks into the Code Area and join them together to build a script for a sprite

A blue box highlights the selected sprite

This panel shows the sprites used in a program—select one to see its code in the Code Area

Add new sprites to the project

Change the backdrop

Sprites

Sprites are the basic components of Scratch. Similar to characters in a video game, they can move around the Stage, change their appearance, and interact with other sprites. Each sprite uses one or more images and is controlled by scripts.

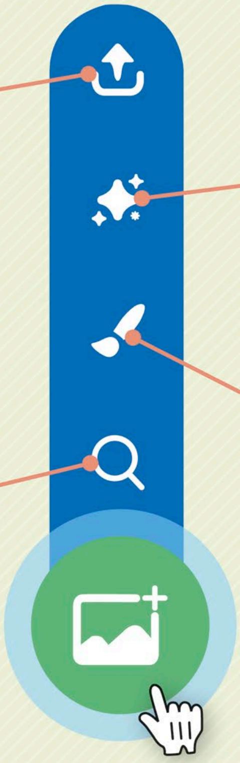
How do sprites work?

Most sprites have multiple images, called costumes, which can be used to animate them in a program. The Cat sprite, for example, has two costumes that show its legs in different positions. Switching between the costumes makes it look like the cat is walking on the Stage. Scratch comes with a preloaded library of sprites that can be used and modified in a program.

Default sprite
Every project starts with the Cat sprite. Delete it by clicking the "x" on its thumbnail in the Sprite List.

Creating a sprite

Scratch allows its users to add or create their own images. The Choose a Sprite button on the bottom right of the Sprite List reveals options to add, create, or upload sprites in a project.



Upload
Users can include their own images and upload them to the Scratch interface to be used as sprites. This includes images, such as photographs, which may be required to create or personalize games.

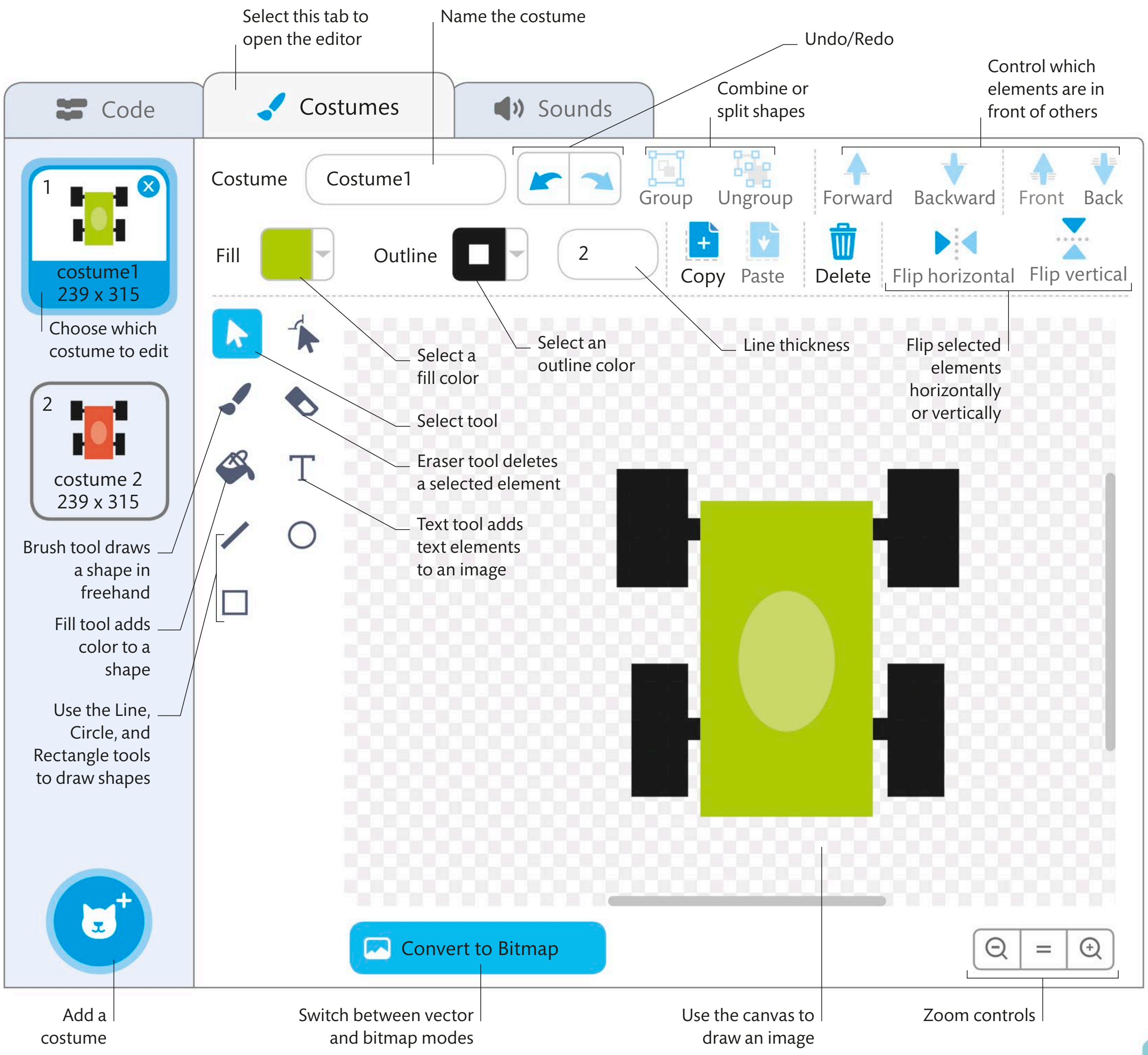
Surprise
Scratch also has an option to add a random sprite from the Sprite library. It can be a quick way to add sprites to experiment with or to generate new game ideas.

Choose a Sprite
Scratch contains a large library of sprites that can be used. The sprites are divided into categories such as Animals, People, and Fantasy. Hovering over a sprite in the library cycles through its costumes.

Paint
Users can create their own images in Scratch's Paint Editor (shown on the right). This editor is quite versatile and can be used to edit the costumes of existing sprites in the Sprite library.

Painting a sprite's costume

The Paint Editor in Scratch can be used to make new sprites or create additional costumes. By default, the editor uses the vector mode, which stores images as shapes and lines, making them easier to edit. The user can switch to the bitmap mode, which stores the color of every bit in the image. The Paint Editor shown below is in the vector mode.


Select this tab to open the editor
Name the costume
Undo/Redo
Combine or split shapes
Control which elements are in front of others
Code
Costumes
Sounds
Costume
Costume1
Group
Ungroup
Forward
Backward
Front
Back
Fill
Outline
2
Copy
Paste
Delete
Flip horizontal
Flip vertical
1
costume1
239 x 315
Choose which costume to edit
Select a fill color
Select an outline color
Line thickness
Flip selected elements horizontally or vertically
Select tool
Eraser tool deletes a selected element
Text tool adds text elements to an image
2
costume 2
239 x 315
Brush tool draws a shape in freehand
Fill tool adds color to a shape
Use the Line, Circle, and Rectangle tools to draw shapes
Convert to Bitmap
Add a costume
Switch between vector and bitmap modes
Use the canvas to draw an image
Zoom controls

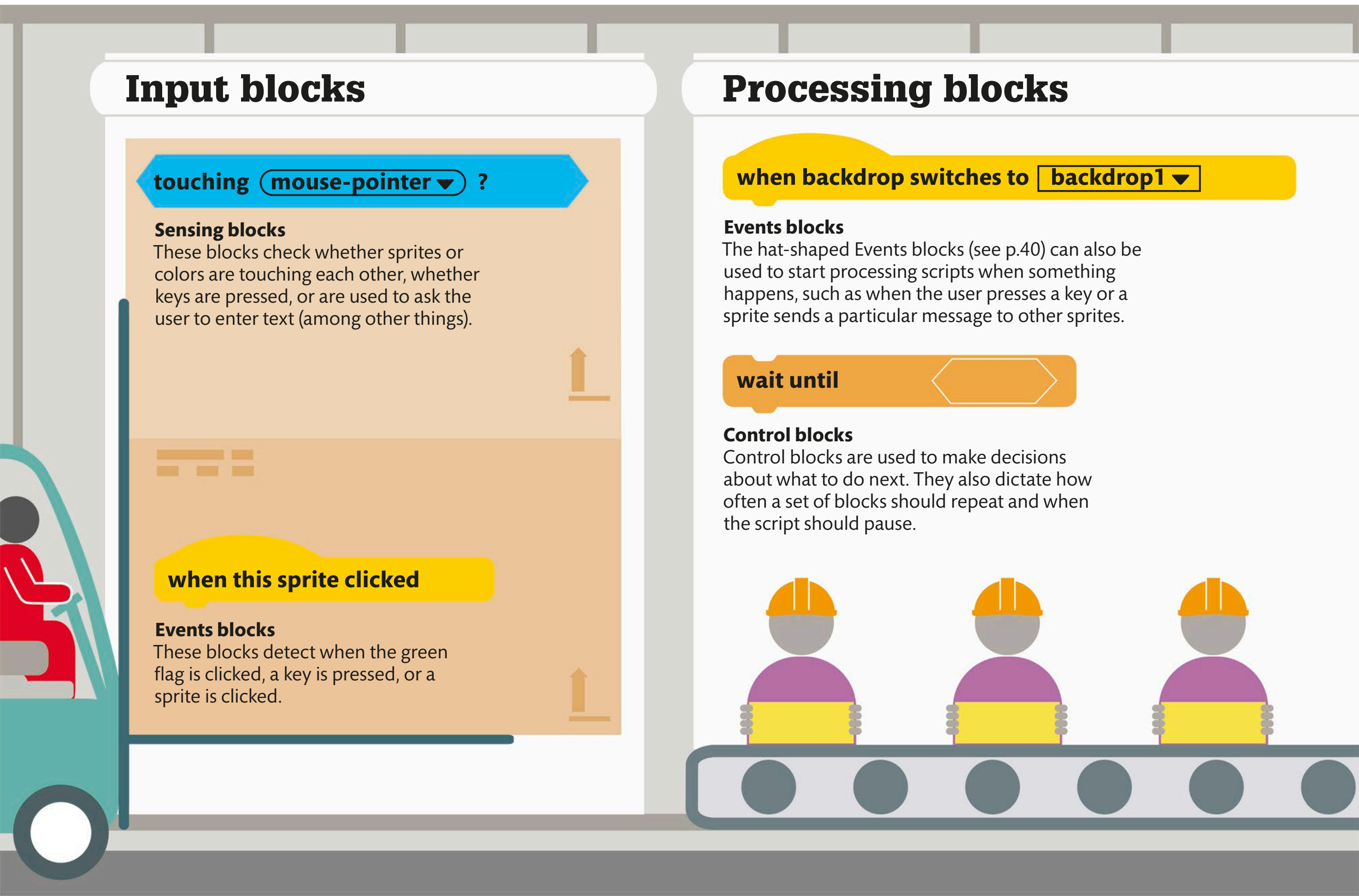
Colored blocks and scripts

Scratch instructions come in color-coded blocks that can be assembled into chunks of program called "scripts." These blocks can be used for collecting input, processing information, and displaying the output onscreen.

Program flow

A program can receive information (input), do something with it (process it), and then deliver the result (output). In a game, the input can be the player's key presses and the output is the movement onscreen. A program may receive input from users, other computer systems, or sensors. Output, on the other hand, can be given on the screen, by a printer, or by sending information to another system.

In Scratch, instructions in a program are built through code blocks. These instructions always run from top to bottom unless told otherwise.



Input blocks

```
touching (mouse-pointer ▾) ?
```

Sensing blocks
These blocks check whether sprites or colors are touching each other, whether keys are pressed, or are used to ask the user to enter text (among other things).

```
when this sprite clicked
```

Events blocks
These blocks detect when the green flag is clicked, a key is pressed, or a sprite is clicked.

Processing blocks

```
when backdrop switches to [backdrop1 ▾]
```

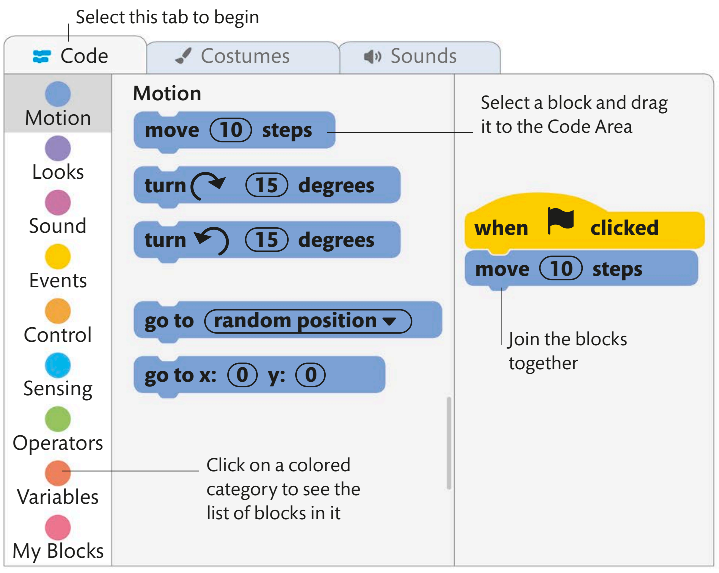
Events blocks
The hat-shaped Events blocks (see p.40) can also be used to start processing scripts when something happens, such as when the user presses a key or a sprite sends a particular message to other sprites.

```
wait until < >
```

Control blocks
Control blocks are used to make decisions about what to do next. They also dictate how often a set of blocks should repeat and when the script should pause.

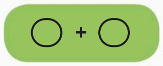
Using the Blocks Palette

The Blocks Palette can be found on the extreme left of the Scratch interface. It contains nine different types of blocks and an "Add Extension" button that can be used to add more blocks to the palette. The blocks can be accessed by switching between the colored categories and scrolling through the list of blocks that appear.

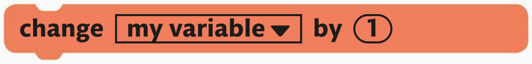


Creating scripts
To create a script, click or press a block and drag it into the Code Area. Drop the block below another block, and they will snap together to make a script. If the blocks fail to snap together, it means they cannot be used that way or they are not close enough to attach.

Output blocks



Operators blocks
These blocks are used for math and comparing numbers and pieces of text, as well as analyzing text. They can also be used to generate random numbers and are great for adding surprises to a game.



Variables blocks
Variables blocks are used to store information, such as the current score of a game. They can also be used to store text. Certain blocks in this category can increase or decrease a variable's number.

move (10) steps

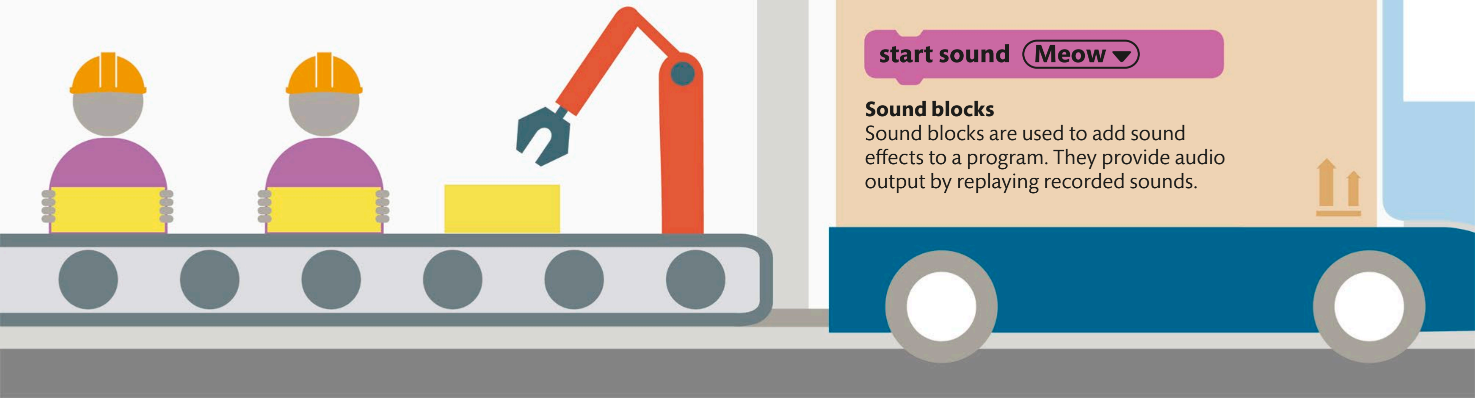
Motion blocks
Motion blocks display the output of a program by moving and controlling the sprites on the Stage.

switch costume to (costume 1 ▾)

Looks blocks
These blocks can alter a sprite's costume, change the background image, and display information in a speech bubble. They can also change a sprite's size and visibility and apply special effects.

start sound (Meow ▾)

Sound blocks
Sound blocks are used to add sound effects to a program. They provide audio output by replaying recorded sounds.

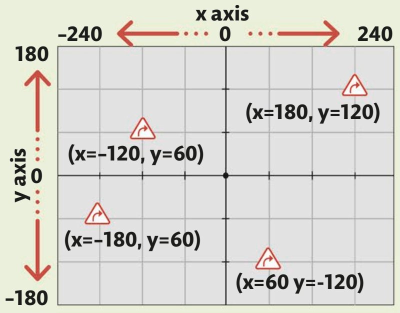


Output using movement

Scratch is an ideal language for programming simple games and applications that move images around the screen. It has a set of blue Motion blocks that can be used to control a sprite's movement.

Coordinates

In Scratch, any position on the Stage can be pinpointed using the x and y coordinates. The x axis runs from -240 on the left to 240 on the right, and the y axis runs from -180 at the bottom to 180 at the top. When writing a program, coordinates can be used to place a sprite in a particular position.



x and y grid
The Stage here has been marked with grid lines every 60 steps. Try these positions in the Motion blocks (below) for moving a sprite using coordinates.

Moving sprites using coordinates

These Motion blocks can be used to move a sprite to a particular position on the Stage using coordinates. The **go to x: y:** block and **set x to** and **set y to** blocks are often used to set a sprite's starting position.

```
go to x: (0) y: (0)
```

Set sprite position
Makes a sprite jump to a particular point on the Stage. The numbers in the block can be edited to choose different coordinates.

```
glide (1) secs to x: (0) y: (0)
```

Move sprite in given time
This block smoothly moves the sprite to a particular point. The time taken for this journey can be specified in the input area for seconds.

```
change x by (10)
```

Alter x position
Changes the x position by the number in the block without changing the y position. It is used to move a sprite sideways.

```
set x to (0)
```

Change x position
Moves the sprite to a particular x position without changing its y position. The sprite will jump straight there.

```
change y by (10)
```

Alter y position
Changes the y position by the specified number without changing the x position.

```
set y to (0)
```

Change y position
Moves the sprite to a particular y position without changing its x position. As with the similar x block, the sprite jumps straight there.

```
x position
```

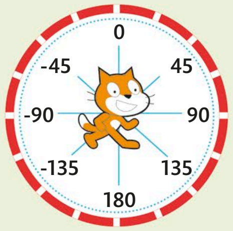
Show x position
This block does not move a sprite, but shows the sprite's x position when clicked. Drop it into other blocks to use this coordinate in a script.

```
y position
```

Show y position
Does not move a sprite, but shows its y position. It can also be used with other blocks. For example, a sprite can be made to say its y position (see p.38).

Moving sprites using directions

Scratch calls each position on the Stage a step. A sprite can be moved by pointing it in a particular direction and then making it walk forward. The direction 90 degrees will make a sprite face right. This is the default direction for most sprites.



```
move (10) steps
```

Move sprite
This block moves the sprite 10 steps across the Stage. However, since this is just one movement, the sprite will appear to jump, not walk.

```
turn ↺ (15) degrees
```

Rotate sprite counterclockwise
This block changes the sprite's direction by 15 degrees the other way. The value in the degrees input can be changed by the user.

```
turn ↻ (15) degrees
```

Rotate sprite clockwise
Changes the sprite's direction by 15 degrees clockwise. As with all blocks with input areas, the number of degrees can be changed as required.

```
point in direction (90)
```

Change sprite's direction
Sets the sprite's direction to a specific number. The numbers are measured from 0 at the top to 180 going clockwise and -180 going counterclockwise.

Drawing with pen blocks

Each sprite has a pen, which it can use to draw a line as it moves around the Stage. The thickness (or size) and color of the line can be changed as required. The **pen down** block is used to draw, while the **pen up** block turns off the pen. The Pen blocks are an extension in Scratch 3.0. They can be found under the Add Extension section of the Blocks Palette.

```
pen up
go to (random position ▾)
pen down
change pen [color ▾] by (10)
move (100) steps
turn ↻ (120) degrees
move (100) steps
turn ↻ (120) degrees
move (100) steps
turn ↻ (120) degrees
```

Stops the sprite drawing as it moves to the start position

Turns the pen on

The script changes the pen color each time it runs

100 is the length of the side

120 is the angle to turn for a triangle

Drawing a triangle
Try this script to draw a triangle using the pen and movement blocks. Click on the script to run it. The **erase all** block in the Blocks Palette can be used to wipe the Stage.

Output using looks and sounds

In a game, sprites often mutate or play sound effects to tell players what is going on. Changing a sprite's appearance or playing sounds can be useful in other programs as well. It can be used to warn users or get their attention to look at something important.

Displaying messages

In Scratch, sprites can display messages through speech and thought bubbles. These are created using the **say** and **think** blocks from the Looks section of the Blocks Palette. The holes in these blocks can be used to change the message to be displayed or to drop another round-ended block in it.

```
say (Hello!)
```

Speech bubble
This block displays a speech bubble containing "Hello!" until a new **say** or **think** block is used.

```
say (Hello!) for (2) seconds
```

Timed speech bubble
Using this block, a message can be displayed for two seconds before it disappears. Both the message and its duration can be changed.

```
think (Hmm...)
```

Thought bubble
This block uses a thought bubble to display a message until a new **say** or **think** block is used.

```
think (Hmm...) for (2) seconds
```

Timed thought bubble
This displays a thought bubble that disappears after two seconds. Again, it is possible to change both the message and its duration.

Changing a sprite's appearance

The Looks blocks can be used to show a sprite's reaction to a game event by giving it special effects. They can also help to display a message. There are even blocks to make a sprite visible or invisible on the Stage.

```
clear graphic effects
```

Remove effects
In Scratch, each sprite can have its own special effects. This block removes all special effects applied to a sprite.

```
switch costume to (costume1 ▾)
```

Change costume
This block changes a sprite's costume to a particular image. The menu can be used to choose which costume to display.

```
set size to (100) %
```

Change size
Makes the sprite's size a particular percentage, considering its default size to be at 100 percent.

```
hide
```

Hide sprite
Makes a sprite invisible on the Stage. It can still move around using the Motion blocks.

```
next costume
```

Show next costume
Useful for animation, it switches to a sprite's next costume or goes back to the first one, depending on the sprite's current costume.

```
change [color ▾] effect by (25)
```

Change effect
Increases (or decreases) a special effect using a positive (or negative) number. Both the number and special effect can be changed.

```
show
```

Show sprite
Makes a sprite visible on the Stage if it has previously been made invisible with the hide block.

```
change size by (10)
```

Alter size
Changes the sprite's size by the percentage entered in the block. Using a negative number shrinks the sprite.

```
set [color ▾] effect to (0)
```

Set effect
Used to give special effects a particular value, no matter what the current value is. Used with 0, this turns off the effect.

MUSIC BLOCKS

The Music blocks in Scratch are an extension and need to be added using the Add Extension button on the bottom left of the Blocks Palette. They make it possible for programmers to use blocks to play musical notes. It is not necessary to know the number for each note, because clicking the hole for the note number will display a piano keyboard to help enter the required music.

Scratch's **set instrument** block has 21 built-in instruments

```
set tempo to (120)
set instrument to ((20) Synth Lead ▾)
play note (60) for (0.25) beats
play note (62) for (0.25) beats
play note (64) for (0.25) beats
play note (62) for (0.25) beats
play note (67) for (1) beats
```

Playing sounds

Sounds are a great way to provide feedback in a game or as an alert in a program. Before a sound can be used, it has to be added to the sprite from the **Choose a Sound** button under the Sounds tab. Programmers can either use a sound from the Scratch Sound library or record or upload a sound of their own.

```
play sound (Meow ▾) until done
```

Pause script to play sound
Sets a sound to play, then pauses the script until it is finished. The menu in the block can be used to choose a different sound.

```
start sound (Meow ▾)
```

Play sound in background
Starts playing a sound but does not pause the script. The sound plays in the background while the script runs.

```
stop all sounds
```

Stop all sounds
This block stops all the sounds, no matter which sprite started them or how many sounds are playing.

```
change [pitch ▾] effect by (10)
```

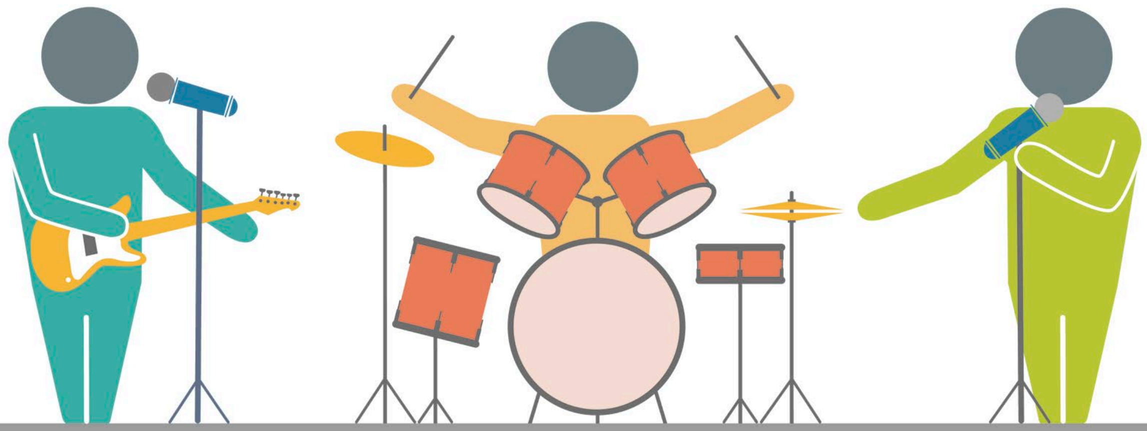
Alter pitch
Changes the pitch of a sound effect. Positive numbers make the pitch higher and negative ones make it lower. The stereo setting can be adjusted as well.

```
set [pitch ▾] effect to (100)
```

Reset pitch
Resets the pitch or changes it to a value specified by the programmer. It can also be used to adjust the stereo left/right setting.

```
clear sound effects
```

Remove sound effects
This block resets all of the sound effects previously applied to sprites or backdrops.



Managing program flow

When writing code, programmers not only have to tell the computer what to do, but also when to do it. In Scratch, the Control and Events blocks are used to manage when an instruction is carried out.

Event-driven programming

In event-driven programming, the program's actions are started by events, such as user input, sensor input (see pp.46–47), or messages sent by other programs or parts of the program. The Events section of the Blocks Palette contains blocks that can start scripts when something happens. Also called hat blocks due to their shape, these Events blocks provide many more ways to start scripts than simply clicking on them.

```
when ⚑ clicked
```

Use green flag to start
Provides an easy way for users to start the program. Copies of this block can be used to start multiple scripts simultaneously.

```
when this sprite clicked
```

Use mouse click to start
This block starts the attached script when the sprite is clicked. It is ideal for creating on-screen buttons for users to click on.

```
when [loudness ▾] > (10)
```

Use sounds to start
When the microphone detects a volume more than 10 (on a scale of 0 to 100), the script can be activated.

```
when I receive [message1 ▾]
```

Use message to start
Scripts can send messages to each other (see pp.48–49). This block starts a script when a particular message is received.

```
when [space ▾] key pressed
```

Use key press to start
Starts a script when a key is pressed. The menu in the block can be used to select the required key.

```
when backdrop switches to [backdrop 1 ▾]
```

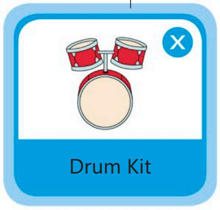
Use background to start
This block is particularly useful in story-based projects. It enables scripts to start when the scene (or background image) changes.

Making a clickable drum

This example uses an Events block to make a simple clickable drum. When the drum is clicked or tapped on the Stage, the script plays a sound and briefly changes the image (or costume) to show that it is playing.

1 Add the Drum Kit sprite
Hover over the Choose a Sprite icon in the Sprite List and select the magnifying glass to see the library. Sprites are listed alphabetically. Scroll down to find the Drum Kit sprite and click or tap on it to add it.

The blue highlight indicates this is the selected sprite



2 Add the script
Select the Code tab and add the following script to the Drum Kit sprite. The sprite already has the required sound. Click the sprite on the Stage to hear it play, and see its playing costume for half a second.

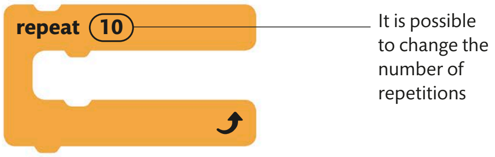
```
when this sprite clicked
switch costume to (drum-kit-b ▾)
start sound (drum bass1 ▾)
wait (0.5) seconds
switch costume to (drum-kit ▾)
```

This costume indicates that the Drum Kit is playing

Reverts to the original costume

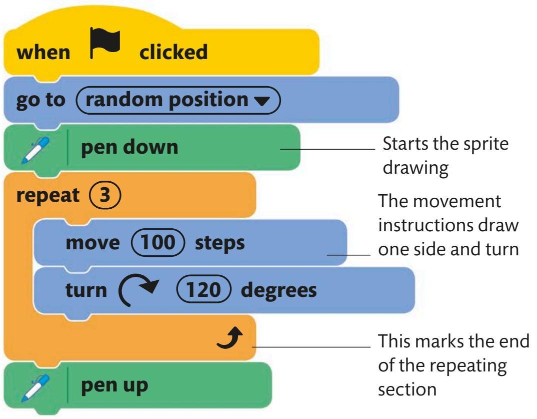
Using loops to repeat

A loop is a part of a program that needs to be repeated. In Scratch, these blocks are placed within the bracket of a **repeat** block, so that Scratch knows where the repeating section starts and ends. The bracket automatically stretches to make room for longer sets of instructions.



```
repeat (10)

↻
```



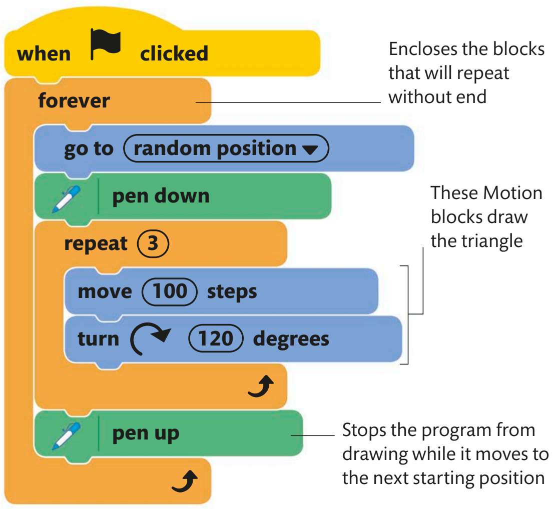
```
when 🏴 clicked
go to (random position ▾)
pen down
repeat (3)
    move (100) steps
    turn ↻ (120) degrees
↻
pen up
```

Starts the sprite drawing
The movement instructions draw one side and turn
This marks the end of the repeating section

Drawing a triangle
In the previous example of drawing a triangle (see p.37), three copies of the instructions for moving and turning were added. The script above, however, uses a loop, which is much easier to read and write.

Repeating forever

Sometimes a program needs to repeat forever, until explicitly stopped. For example, a simple animation or game can play indefinitely. The **forever** block repeats a set of instructions without ending. This block has no nub to join other blocks underneath it, since it never ends. To end the script, click the red Stop button. A **stop all** block may also be used to stop the script.



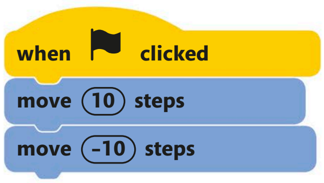
```
when 🏴 clicked
forever
    go to (random position ▾)
    pen down
    repeat (3)
        move (100) steps
        turn ↻ (120) degrees
    ↻
    pen up
↻
```

Encloses the blocks that will repeat without end
These Motion blocks draw the triangle
Stops the program from drawing while it moves to the next starting position

Drawing infinite triangles
This script will draw triangles in random positions forever. As shown above, a repeat loop can be placed within a forever loop. A loop inside a loop is called a nested loop.

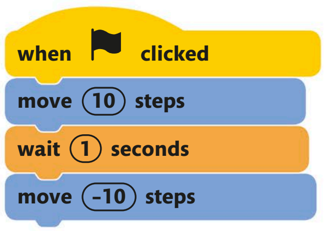
INTRODUCING DELAY

It is not always desirable to have a program run as fast as possible. In many cases, a program may need to be slowed down so that users can easily see what is going on and have time to respond. Games are often artificially slowed down in order to ensure that players can keep up.



```
when 🏴 clicked
move (10) steps
move (-10) steps
```

Normal movement
In this example, the program runs so fast that the sprite's movements are not visible. Nothing seems to happen.



```
when 🏴 clicked
move (10) steps
wait (1) seconds
move (-10) steps
```

Delayed movement
Introducing a **wait** block makes it possible to see the sprite move from right to left and back again.

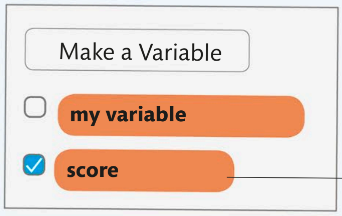
Manipulating data

Programs are often used to manage and process data. This data is either provided by the user or collected from other computer systems. In Scratch, the Operator blocks are used to manipulate numbers and text stored in variables.

Variables

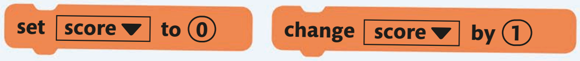
Many programming languages use variables to store information. A variable can store one piece of information—either text or a number. In a game, for example, two variables might be used to store the player's name and score.

1 Make a variable
To create a variable in Scratch, select Variables in the Blocks Palette and click on the Make a Variable button. Give the new variable a meaningful name, such as score, so that the code is easy to understand. Usually, variables need to be created for all sprites, which means that all sprites can see and change the variable.



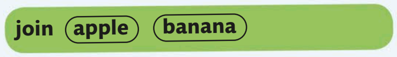
The new variable will be displayed here

2 Use blocks with your variable
Use the **set [variable name]** block to reset the variable's value—for example, **set score to 0**. The **change [variable name] by** block can be used to increase or decrease the value.

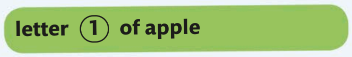


Strings

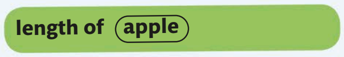
Programmers often call a piece of text in a program a "string." For example, a string can be a name, an answer to a question, or an entire sentence. In Scratch, any variable can store a number or a string, and it can store different values at different times.



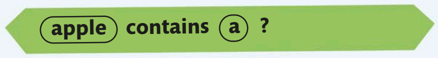
Join strings
This block can be used to join two strings. The strings are joined without a space, so the result for this example will be "applebanana". Variable blocks can also be used in place of words typed into the block.



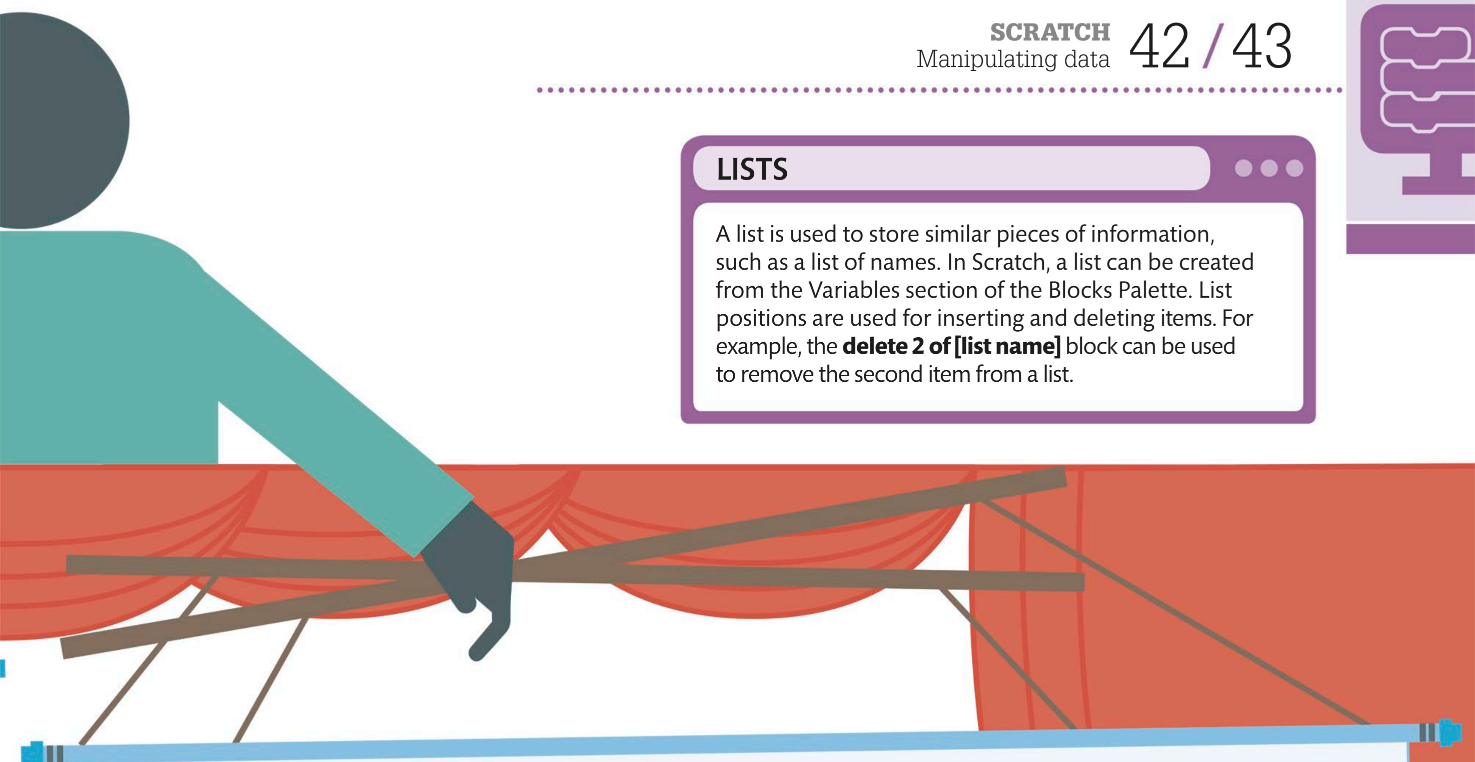
Extract letters
This block extracts one letter from a string. In this example, the first letter of the string "apple" is extracted.



Count a string
The number of characters in a string can be counted using this block. The result for the block can be viewed by clicking on it. It can also be dropped into other blocks to use in a script.



Check strings
This block checks whether the second string input is in the first one and gives the answer as true or false. It is also possible to check for more than one letter: **apple contains app?**



LISTS

A list is used to store similar pieces of information, such as a list of names. In Scratch, a list can be created from the Variables section of the Blocks Palette. List positions are used for inserting and deleting items. For example, the **delete 2 of [list name]** block can be used to remove the second item from a list.

Numbers

Operator blocks are a core part of Scratch and can be used for arithmetic operations, comparisons, and to pick random numbers. Some operator blocks even work with strings.

Used for addition

```
(7) + (2)
```

Used for subtraction

```
(7) - (2)
```

```
(7) * (2)
```

Computers use the symbol * for multiplication

```
(7) / (2)
```

Computers use the symbol / for division

```
pick random (1) to (10)
```

Chooses a random number between 1 and 10

```
when ⚑ clicked
go to x: (0) y: (0)
erase all
set [length of sides ▾] to (pick random (10) to (100))
set [number of sides ▾] to (pick random (3) to (8))
set [angle to turn ▾] to ((360) / (number of sides))
pen down
repeat (number of sides)
    move (length of sides) steps
    turn ↶ (angle to turn) degrees
end
pen up
move (150) steps
```

Length of the shape's sides is randomly chosen, between 10 and 100

The angle to turn is set at 360 degrees divided by the number of sides

This repeats the code to draw each side

Drop the variable blocks into hole in the Motion block

Moves the sprite out of the way after drawing

Drawing random shapes
The program above draws a random shape each time the green flag is clicked. Start by making the variables **length of sides**, **number of sides**, and **angle to turn**. Click the green flag several times to create random art.