AN INITIATIVE BY

With the participation of UNESCO

United Nations Educational, Scientific and Cultural Organization

YouthMobile

AFRICA CODE WEEK

An Introduction to the Scratch Programming Language

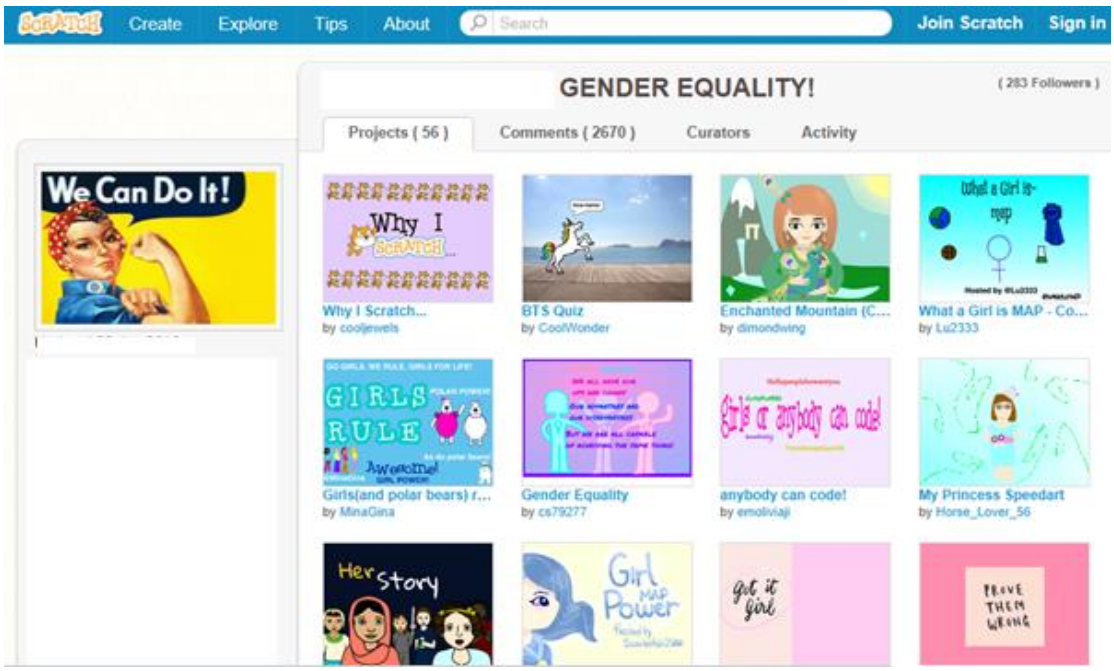
by

Brendan Smith, Camden Education Trust, Ireland

Table of Contents

Introduction to Scratch and the Art of Coding

How computer coding can function as an inter-disciplinary learning environment



In this section, we will introduce you to Scratch, a programming language developed at the MIT Media Lab in the United States that has captured the imagination of children everywhere.
Scratch makes it easy for users to create their own interactive stories, animations, games, music, art and to share these creations on the web.
So it is ideal for children aged eight to eleven years of age.

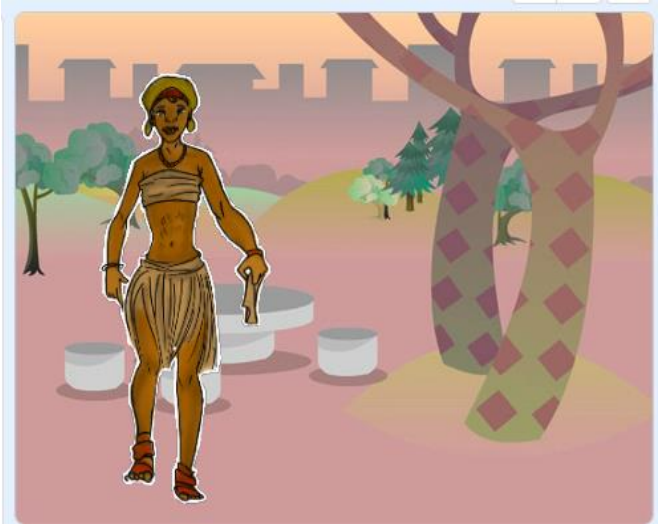
What is a program?

A program is a set of instructions that tells a computer or other electronic device what to do. These instructions or commands are written in an artificial (i.e. non-speaking) language. The script used is often referred to as code or computer code. Computer programing or coding is the process of writing code.

Where does the term Scratch come from?

Scratch is the **name** given to this very powerful but very child friendly programming language
The term Scratch was chosen by its' inventors because of its' similarities to a Hip Hop DJ's method of mixing different music tracks together to create new sounds.
For it is a key element of the Scratch website that members can download other people's computer games and change the details if they so wish (e.g. 'speed up' or 'slow down' animated characters in a game). The result is that a large number of

these online projects are remixes of other projects on the website http://scratch.mit.edu



A Fun Way of Learning

The Scratch language has similarities to children's building brick toys. It uses a simple structure of graphical **bricks** or **blocks** of computer code that snap and interlock together to build and control sound, music and images. Hence it is ideally suited for the enjoyment and learning by children, as it utilizes their natural inclination of tinkering with building materials to create different shapes, games and stories within a **new** and **exciting interactive** digital dimension. Gone are the days of the difficult and boring text syntax that traditionally constituted a computer program.

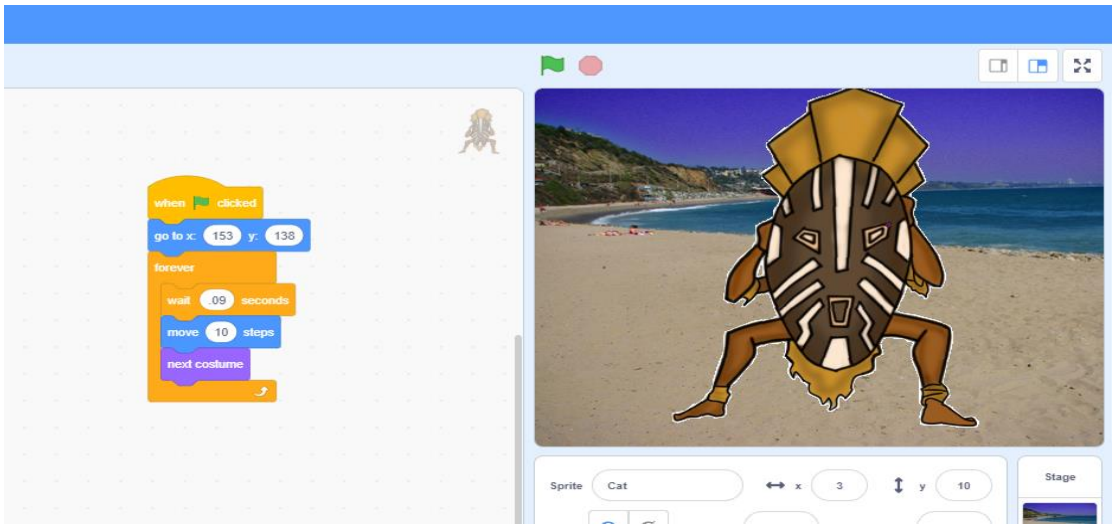
Educational Benefits: Numeracy, Literacy & Beyond

Scratch provides a unique environment for children and young people to develop and utilise their artistic and creative talents through the construction of eye-catching animations.

During the training sessions, children work together to complete certain projects, learn to plan, design and share tasks, are encouraged to show and explain their completed works to their fellow participants where critical analysis and compliments by fellow classmates are an integral part of the learning presentation and communication process.

Scratch allows children to develop their artistic and creative skills in a digital world that is both empowering and adventurous.

The teaching of Scratch utilizes so many different aspects of junior school curricula including art, languages, science and particularly numeracy and literacy.



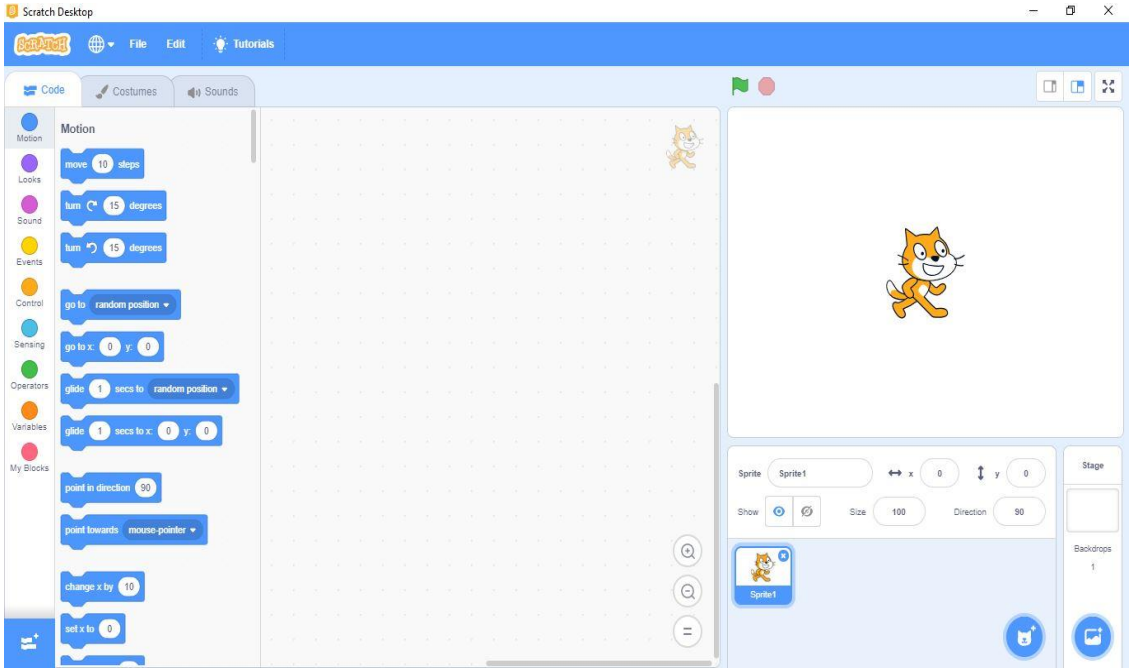
As children enjoy the challenge of creating and sharing Scratch projects, they learn important mathematical and computational ideas such as arithmetic (addition, subtraction, multiplication, division of numbers), geometry (branch of mathematics that deals with the measurement, relationships of points, lines, angles etc. via such terms as coordinates, shapes, size, relative position of figures), algebra (variables, symbols representing numbers for explaining quantities and numeric relationships) as well as additional concepts such as algorithms, while also learning to reason systematically, solve problems, work collaboratively and develop presentational skills.

Motivation is very rarely a problem for learners of Scratch

Scratch allows users to create a large variety of online projects that can reflect their own personal interests as well as being used as a resource in a range of subjects across the school curriculum e.g. creating interactive games, mapping out a tour of different countries for a geography class, producing a musical concert for a music class, building a dress-up doll with multiple clothes options within an arts class, or creating a digital story about a sporting match, a folk tale or an important science issue such as the causes of global warming.

With junior school children, the teacher can expand the artistic elements of Scratch by getting them to make clay models of their sprites whilst planning out their projects.

Lesson 1 - Introduction to Scratch Interface



Sharing & Storing Scratch Projects Online

A number of the projects used in these learning notes are stored online on the Scratch website.
Where this occurs, the specific web address for the individual project is provided.

This site should also be used to store the best examples of the projects created by your participating students, so that other mentors can used them as teaching resources in their classroom and as examples for other participants to emulate.

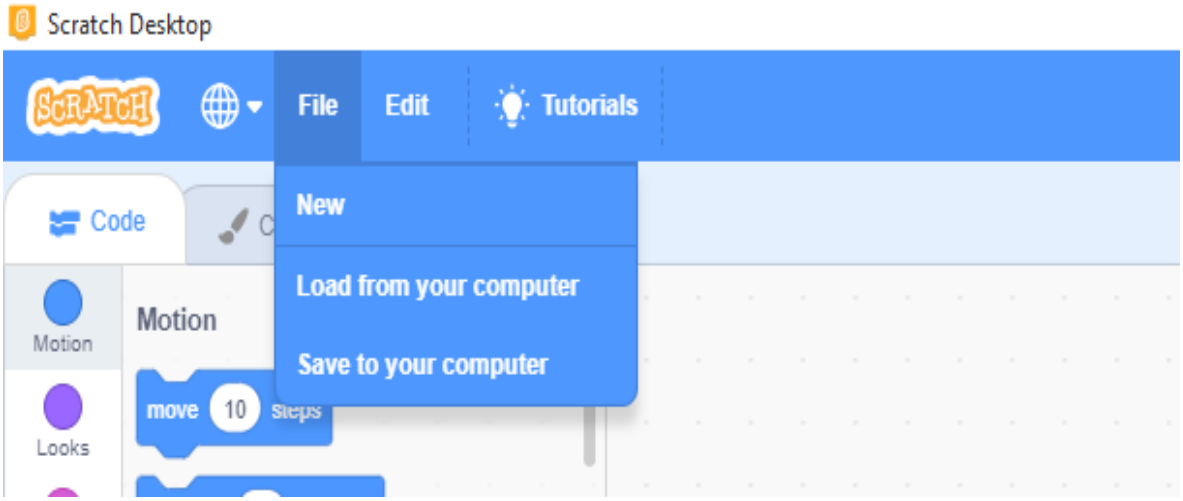
To access the Scratch website:
Type into the web address bar http://scratch.mit.edu
If you do not already have an account, click on **Join Scratch**

However you can join the existing Code for Africa community by going to the Sign In option and
Under **Username**, type in africacodeweek
Under **Password**, type in fionnfionn.

To **upload** a completed project from your computer or other smart device onto the Africa Code Week account on the Scratch website, first go to the Create option in the menu bar.
Then select under the **File** option Load from your computer
As a teacher, you may wish to create an account on the Scratch website for your class, or school or group that can be used to host samples from each of your participant's work.

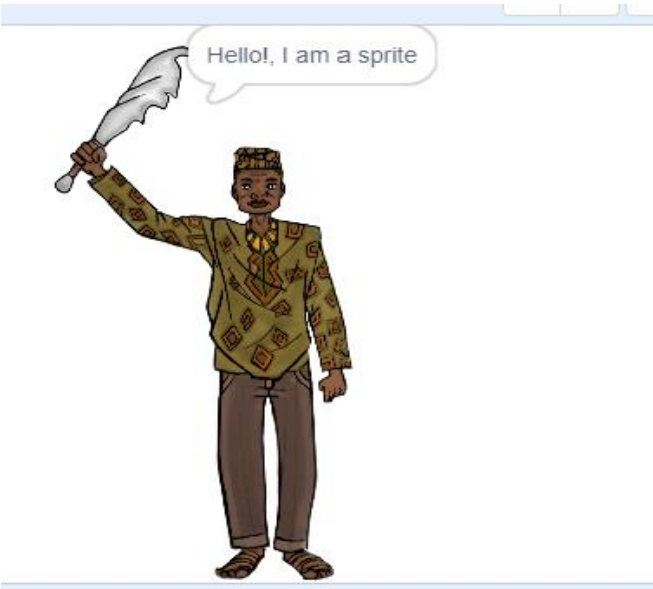
This course though will use the Scratch Offline Editor.

Go to a web browser such as Google Chrome, Safari or Firefox.
Type in on the address bar:
http://scratch.mit.edu/scratch3download/
and follow the instructions.
To familiarise yourself with the programme, click on Tutorials.



A Moving Talking Sprite

A **Sprite** is an animated character or object in your programme.



In Scratch, sprites can move around, be active or be objects that stay still. We will choose a sprite character that we will animate.

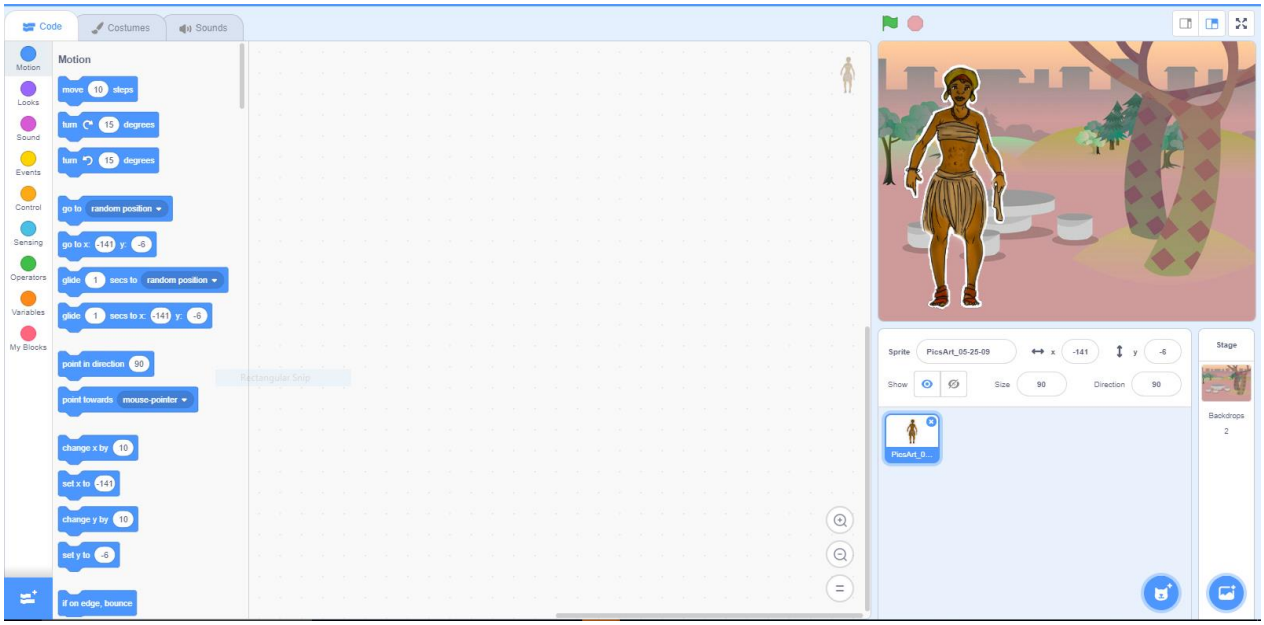
However it is important to realise that a sprite cannot do anything by itself.

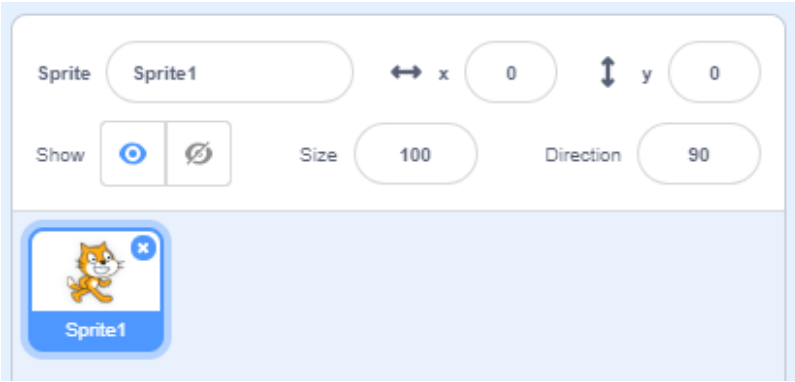
A sprite's action comes from a response to scripts inputted by the user into the **Script** (scripts or programme code) **Area**. These scripts are the instructions or commands that tell the sprite what to do and are written in a sequence. The user drags individual pieces of code from the Blocks' Palette into the scripts area. These blocks then fit together like pieces of children's building bricks to create the instructions.

So let us enter Scratch and program the sprite to talk and to move around the screen!

To open **Scratch** on your smart device, double click on the Scratch icon on your computer.

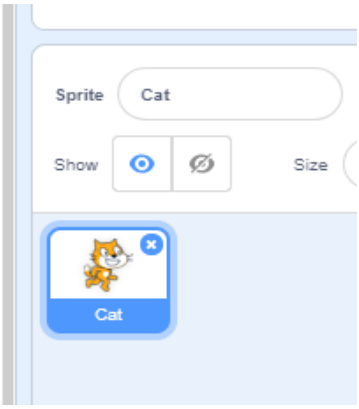
You will see the opening screen.





Take your time and familiarise yourself with the main features of this screen or what we will refer to as the Scratch Interface (Home Page).

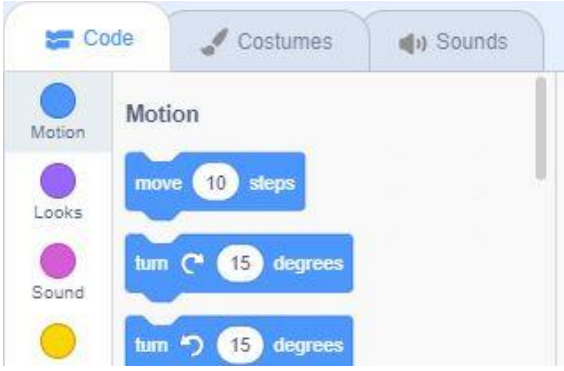
Notice that the small version of the cat is highlighted in blue signifying that it is the active component.



Placing Text

By placing blocks in the Script Area, the sprite will tell us his/her name.

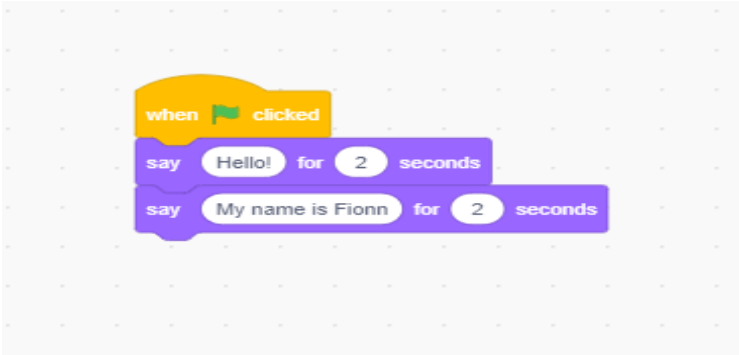
First go to the **Blocks' Category section** located at the left hand side of the Scratch interface which contains thematic folders of blocks of code such as Motion, Looks, Sound etc.
To the right of this section of the graphical user interface is what is referred to as the **Blocks Palette** that contains all the individual blocks of code.



Go to the **Events** folder.
Place the following block in the Script Area:

This block is a 'When' command.
This will mean that once the **Green flag** above the Stage is clicked, the Sprite will follow the commands that are placed in the Script area.

Go to the **Looks** section located in the Folders panel.
Select the blocks twice that **say Hello** for two seconds.
Type in the text Hello! and My name is Fionn (or your own choice of name) before placing both in the Scripts area.



Make sure the blocks click together.

It would be nice to personalise the sprite by giving it a name.

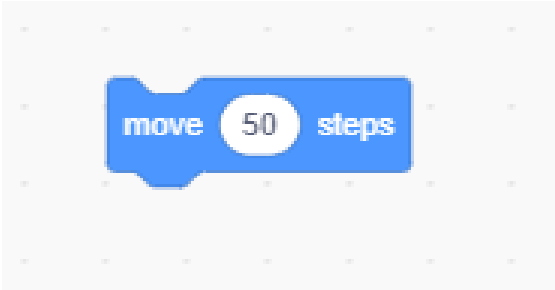
To do so, click on the  at the top left of the cat sprite icon and type in the text Fionn or your name choice in the relevant box.

Click the Green Flag at the top left hand side of the computer screen and see what follows.

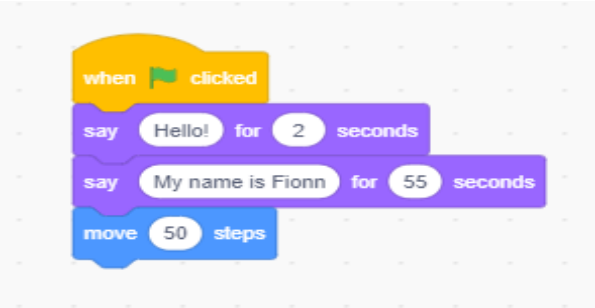
Now we need to get the cat moving.

Go to the Motion folder

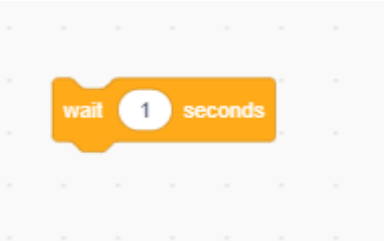
Select the Move block and change the number of steps to 50.



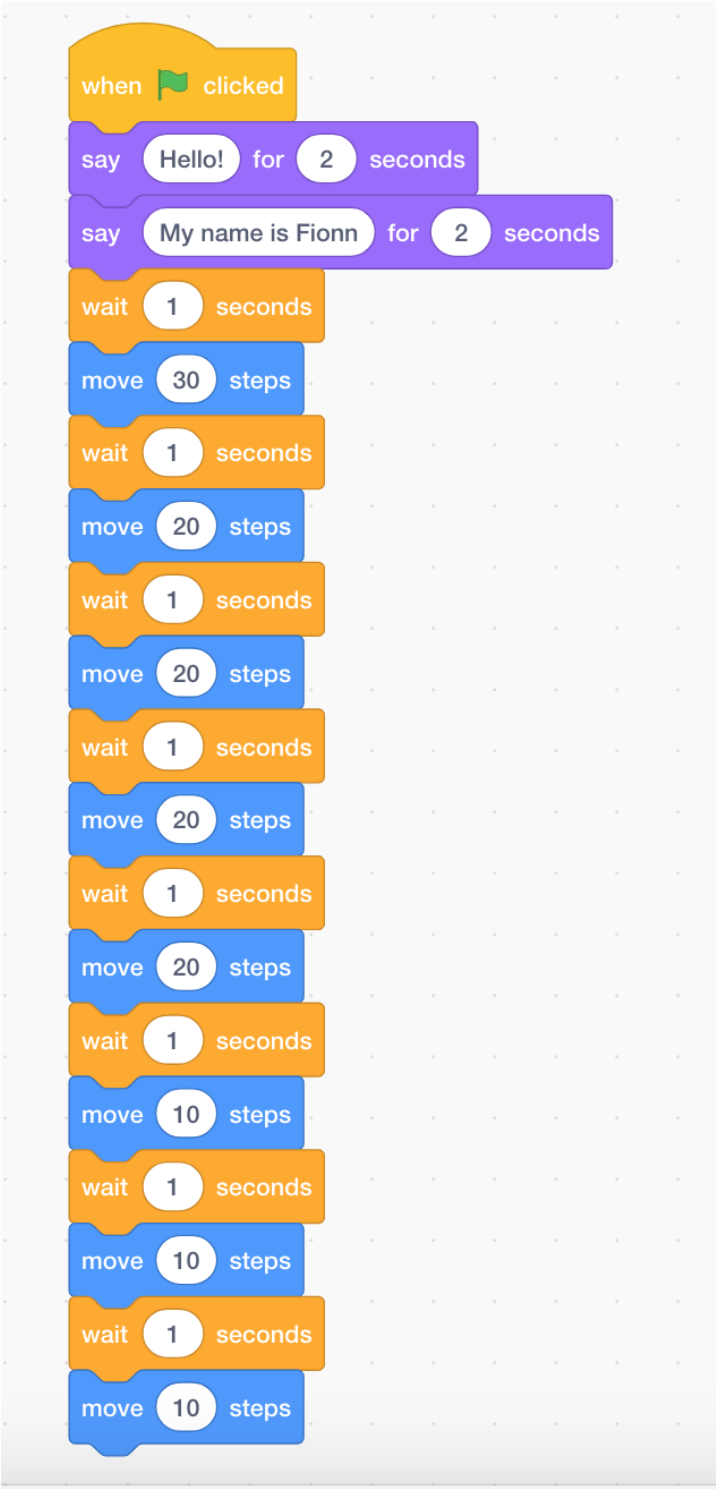
Attach this block to the rest of the blocks in the script area and start the program.



To increase movement of the sprite and to give the impression of walking, first go to the Control folder and select the Wait block



Place this block in the Script area with the addition of some extra Move blocks:



However, we now have a problem with the sprite.

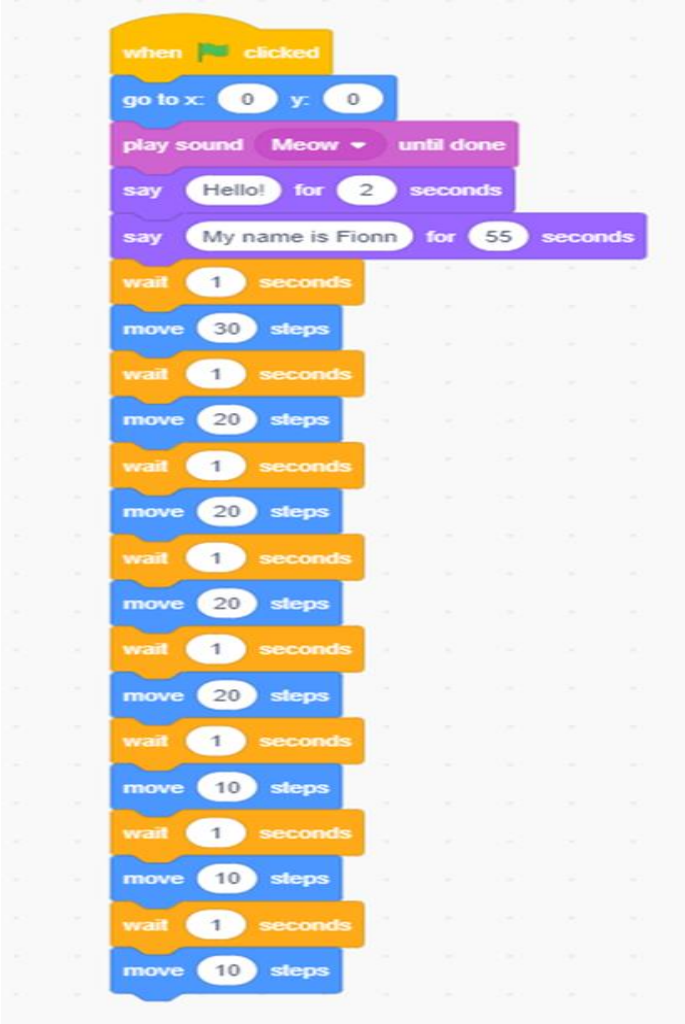
As you may have noticed, if we keep using this script, the cat will keep moving until it almost disappears off the screen.
So we have to put in an extra command that will bring him or her back to the centre of the screen on every occasion that we use this set of instructions.

The screen is divided into **X (horizontal)** and **Y (vertical) coordinates** based on the centre of the screen being **(X)0 (Y)0** and the numbers being positive or negative depending on their positioning.
Get your pupils to move the sprite around the screen and watch as the values of the X and Y coordinates change just above the sprite small icon at the right side of the Sprite Interface as the sprite changes position.

This function allows the user to position different sprites at different locations.
(See later explanatory box on Geometry).

Hence we can place a piece of code or block at the beginning of the set of commands that will instruct the cat to move back to the centre of the screen every time that we select the Green flag.

This will appear as follows:



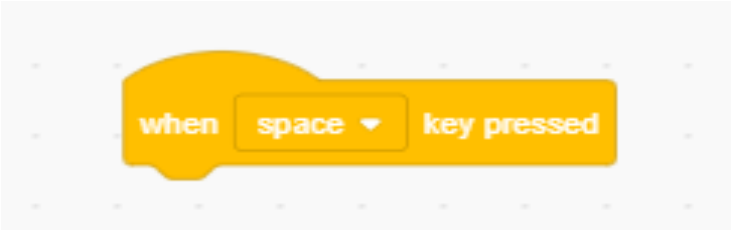
Test the effectiveness of this new piece of code by using the mouse to position the sprite towards the bottom or top of the screen before clicking on the Green Flag icon.

Different methods other than a Green Flag can be used to start a script using the 'When' commands in the Control folder

For instance, the Space Bar or Arrow Keys.

So replace

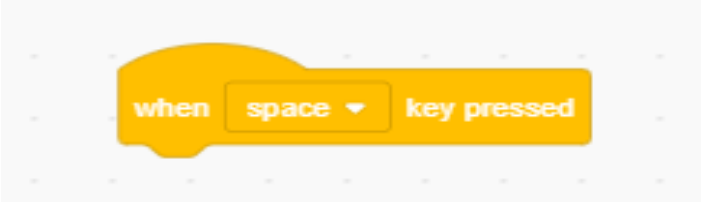
with



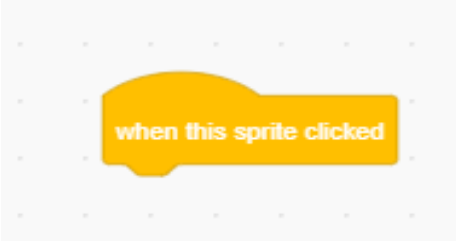
(from the Events category) in the Palette.

Now click the Space Bar on the computer keyboard to start the programme.

Replace



with



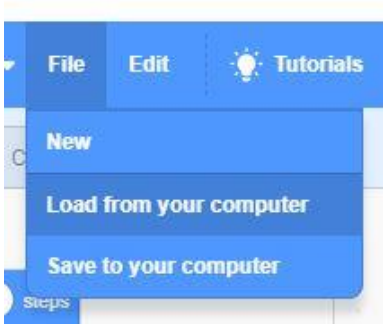
Then start the programme by clicking on the cat.

Revert back to the Green Flag block.

Saving the project

Go to the File folder

Select 'Save to your computer'



Type in Fionn, or your own name for your cat icon in the section Save As.....

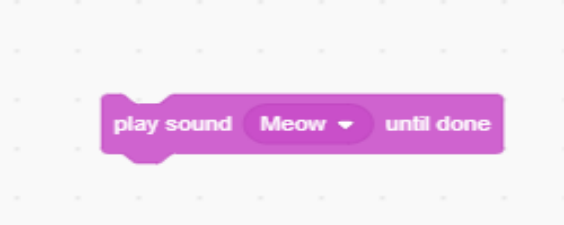
Then choose the option where you want to store it, such as Desktop or My Projects.

Lesson 3 - Placing Sounds in a Script

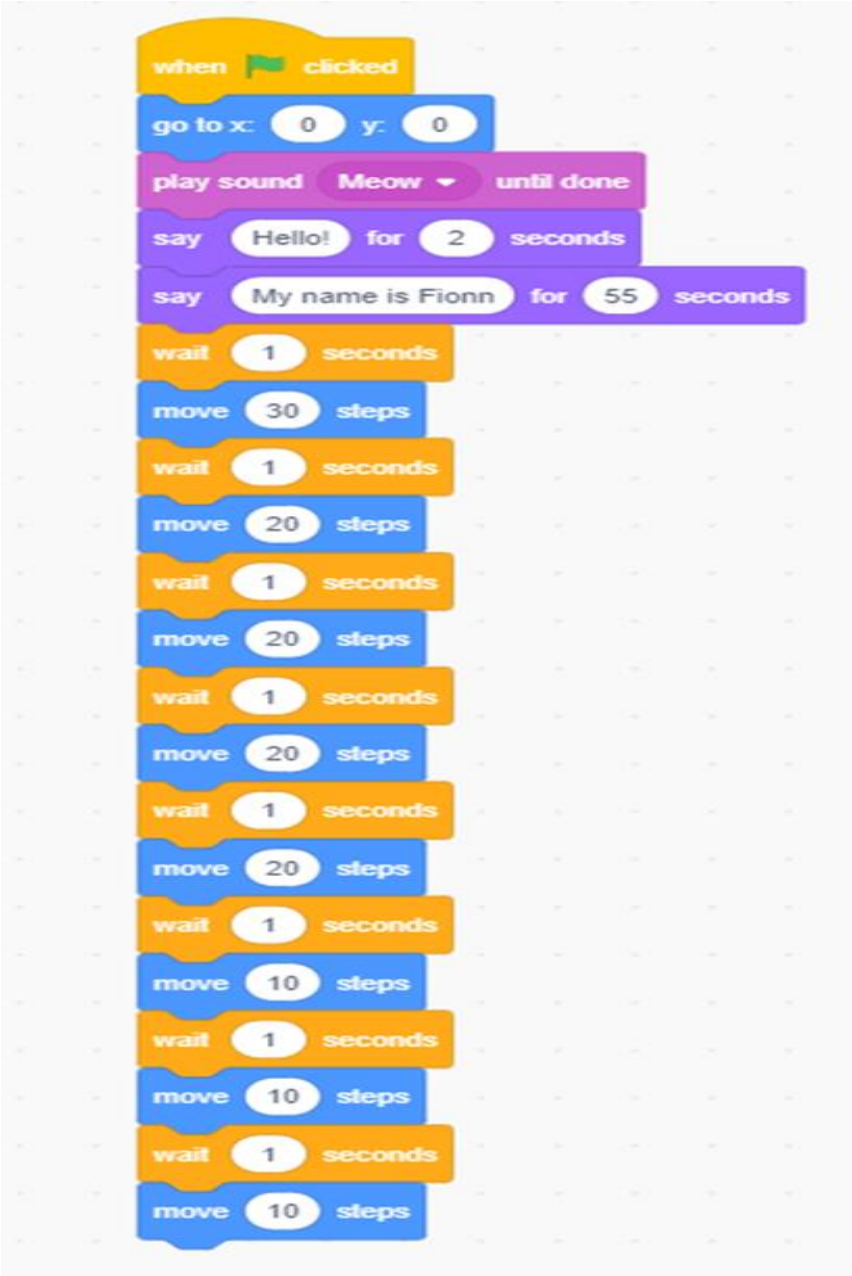
Making the Cat Purr!

Go to the Sounds category.

Select the following block:



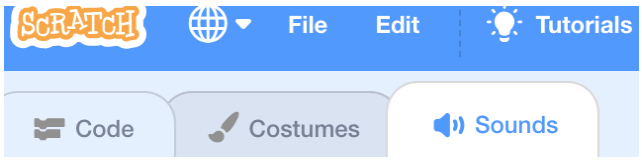
Place it in the set of instructions in the Script Area



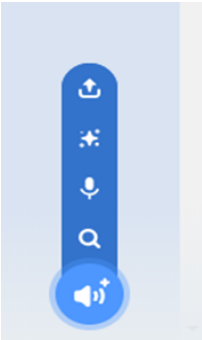
Play the programme.

Now replace the cat meow in the script with other animal noises.

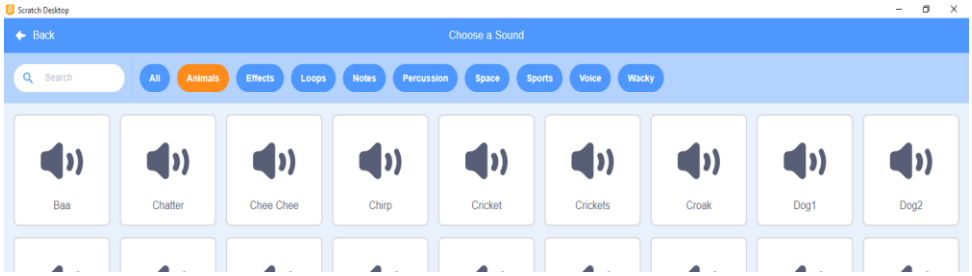
This can be done by going to the Sound library:



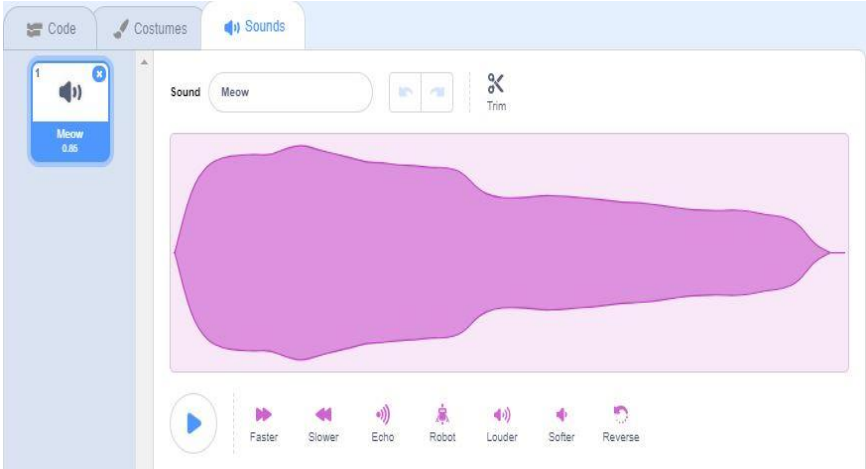
and clicking on the speaker icon at the bottom left hand corner of the page.



A whole repository of sounds now appears.



Click on your preferred sound which will then appear on the list of sounds on the left column (under the heading Code).



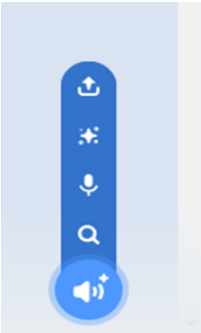
You can now include the new sound in your script by selecting the relevant block (& sound option) of code

Exercise

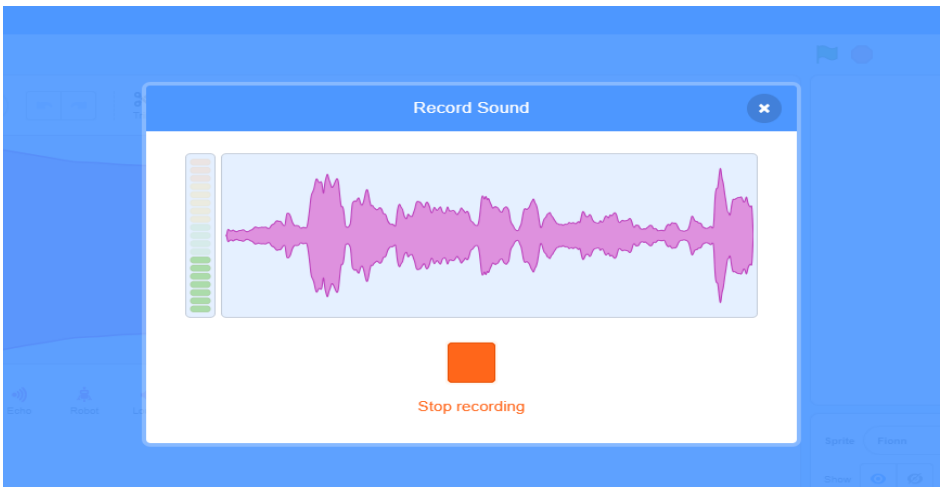
Wow! A Talking Cat!

Now let us get the cat to say the words that we have written into the programme, namely Hello and My Name is Fionn.

Click on the Microphone image in the row of icons to the bottom left of the screen as shown below:

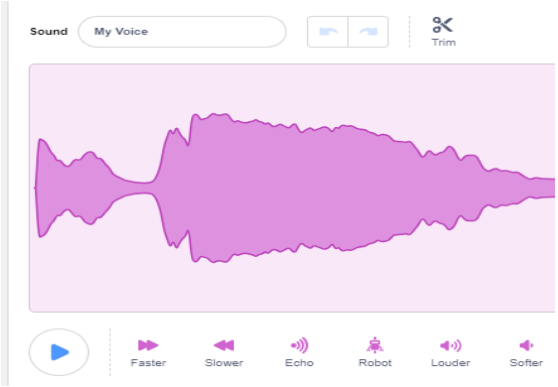


Then choose to record your voice by clicking on the RED button below



Say loud and clear, the word “Hello”. You can trim and edit the audio as required.

Change the title Recording1 to My Voice.



Return to the Script Area and select the Sound (pink) category.
Place the play sound My Voice block in the script just above the say hello for 2 secs block

Repeat this recording process for My Name is Fionn.

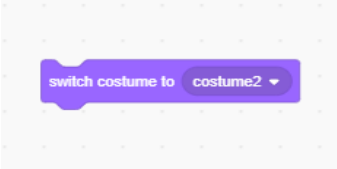
Lesson 4 - Making the Sprite Walk Better

Let us now make the cat more realistic in its movements by getting it to move its' legs when she/he walks.

This can be done by changing the physical look of an individual sprite under the section labelled Costumes.

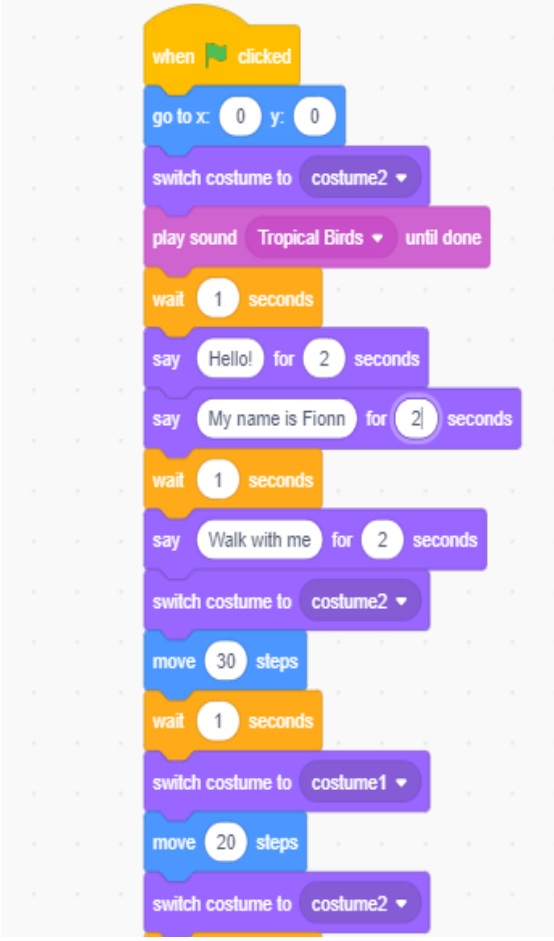
Go to Costumes and notice that there is a second image or Costume of the same sprite with the legs and arms in different positions than that of the first image.

After the first Move and Wait blocks located in the script, place in a 'Switch to costume block with Costume 2' taken from Looks.
The option Costume 2 is chosen by clicking on the inverted black triangle icon located to the left side of this block



Then, for the next Move and Wait blocks, place in a Switch to costume block with Costume 1.

Repeat this process all the way through the Script, thus alternating between Costume 1 and Costume 2.

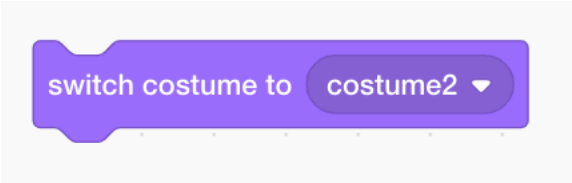


Play the Script by clicking on the Green Flag icon

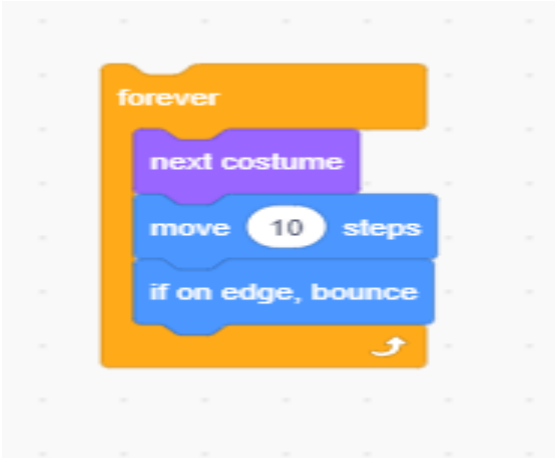
A Very Speedy Cat!

In this lesson students are introduced to the very important Forever loop block of code as we demonstrate how the cat can speedily move back and forth across the screen without stopping!

First, separate all the text in the code that is positioned underneath the first occurrence of the following block:



In place of the removed blocks, substitute the following new commands:



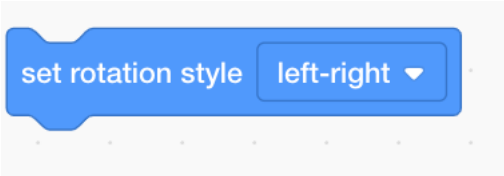
Explain to students the importance of the Forever loop in computer programming.

This piece of code states that the sprite will change costume and move forward one set of steps not just once but forever. Furthermore, the addition of if on edge bounce command means that the sprite will not disappear off the screen but turn each time it reaches the end of the screen in order to continue walking.

However you might now find that the sprite is walking upside down!

To ensure that the cat is standing upright and moving face front, go to the Motion category and place under

the following block:



Lesson 5 - Multi-Coloured Sprites

A Chameleon Cat! Changing the colour of a sprite.

Let us change the colour of the cat whilst it is walking.

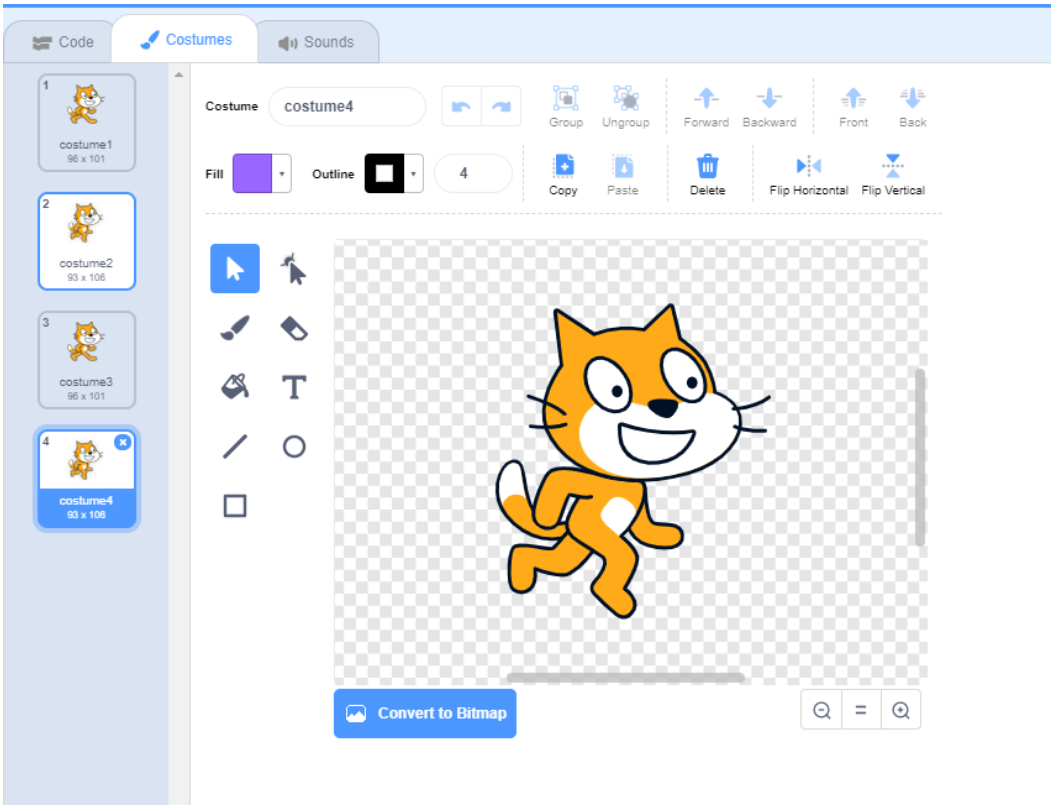
To do this, go to Costumes option and select Duplicate for Costume 1 by clicking appropriately on the keypad or on the top right side of the mouse.
Do the same for Costume 2.

Repeat this process for both Costume 1 and 2 until you have nine costumes.

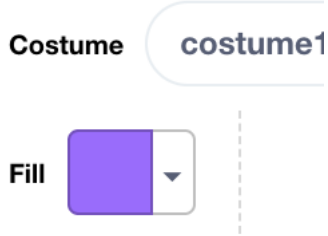
Change the Costume numbers in the script so that they appear chronologically.

Then click the icon for Costume 2

To the right of the icon costumes is the Paint Editor, which allows one to colour in existing sprites as well as to paint and draw new ones.



Choose a different colour from the palette box labelled Fill



located at the top left side of the screen.

Select a colour and use the paint bucket icon to bring this new characteristic into the existing Sprite on the screen.
Please ensure that the Paint Editor is in Bitmap Mode for use of the paint bucket tool.

Note: Introduce to the students some of the other features of the Paint Editor including duplication, text and erasing tools.

When finished, press okay.

Repeat this process to bring a new colour into all of the remaining costumes

Play the Script by clicking on the Green Flag icon and watch the cat of many colours walk!

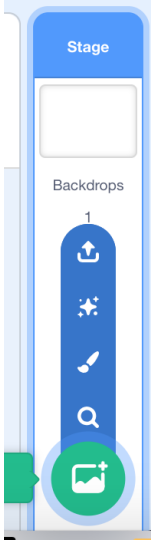
Lesson 6 - Changing the Backdrop (Stage)

At present, we are using a blank backdrop or background for the cat.
So let us bring some excitement into its' life by having the cat walk around a new landscape.

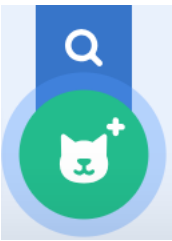
If you view the bottom left side of the screen, you will notice that Fionn the cat is presently highlighted in blue which signifies, as mentioned previously, that the Sprite is the action element in the Script.

Click on Stage.

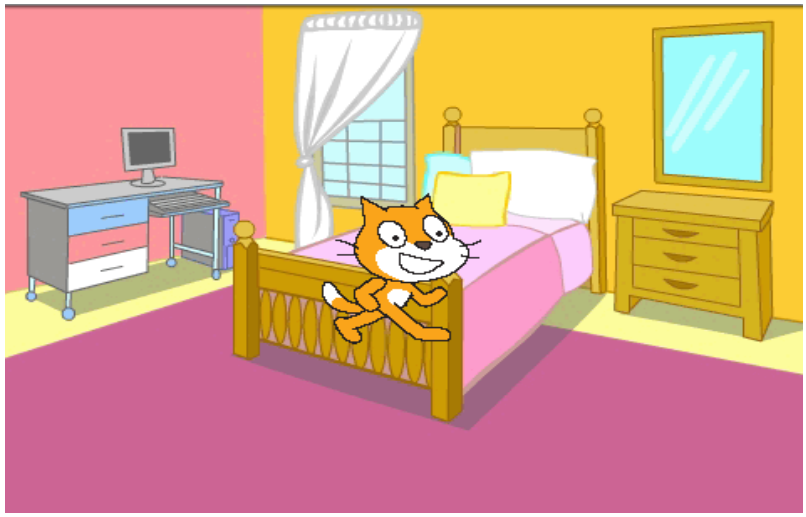
The Stage icon is now highlighted in blue signifying that it has now become the active element that the user can now change.



Go to the New Backdrops icon located under the Stage icon and select either of the bottom options in the row of five icons (see below).



Go to the folder labelled Indoors and select Bedroom 2.



Click on the Green Flag icon to start the programme.

You will now notice though that we have now got a problem, namely that Fionn the cat looks as if it is walking on air!

So first ask the students to come up with a solution on how to code the script in order to make the cat move across the floor of the bedroom.

We will of course have to put in some extra command code in a script for the cat that will ensure that the it walks along the floor.

So double Click on the Backdrop icon to move back to the Script of the Sprite (cat).

Move the Sprite to the bottom left-hand corner of the Stage screen.

GEOMETRY

Computer Images are made up of Pixels. They can be identified using X & Y Coordinates.

A computer screen or a picture is made up of basic units known as **Pixels**. It is an abbreviation of two words, picture and element.

Pixels are the smallest units of colour on a computer display or in a computer image that can be controlled or programmed.

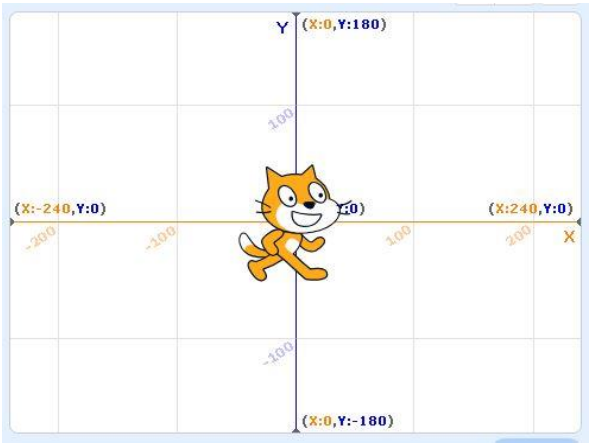
X, Y coordinates are respectively the **horizontal** (X) and **vertical** (Y) addresses of any pixel or addressable point on a computer display screen.

The x coordinate is a given number of pixels along the horizontal axis of a display starting from the pixel (pixel 0) in the centre of the screen. The y coordinate is a given number of pixels along the vertical axis of a display starting from the pixel (pixel 0) in the middle screen. Together, the x and y coordinates locate any specific pixel location on the screen.

X & Y coordinates are part of the branch of **mathematics** known as **Geometry** which is concerned with questions of relative position of figures, shape, size, and the properties of space.

To familiarise users with XY gridlines, go to the new Backdrop section, as had been undertaken previously above and select the last or second last icon in the row of five icons. Then choose the xy-grid, the third last of the screen options in the backdrop library.

The following screen now appears:



Allow students to view the screen and move the sprite around in order that they come to an appreciation of coordinates.

Explain their importance for instance in computer gaming (e.g. moving objects in a game to a start position).

Return to the Bedroom 2 backdrop.

Move the cat to the top right hand corner of the stage. The X and Y coordinates for the cat will appear under the stage as follows:

Once the cat is in position, the current X & Y coordinates for the cat will appear in the X and Y boxes (as shown in the image above)

Go to **Motion**.

Find the **Go to X: Y:** block

Move this block into the Script Area and position it directly under the first (Green Flag) block in the set of programme instructions.
As mentioned above, the X and Y coordinates for the cat will be shown in the block.

Click the Green Flag icon to start the programme.

Question: What do we have to do to get the cat to jump onto the bed?

The operator of course must instruct the sprite to move onto the bed by placing, in the correct spot in the programme, a Motion block that includes the correct X and Y coordinates.

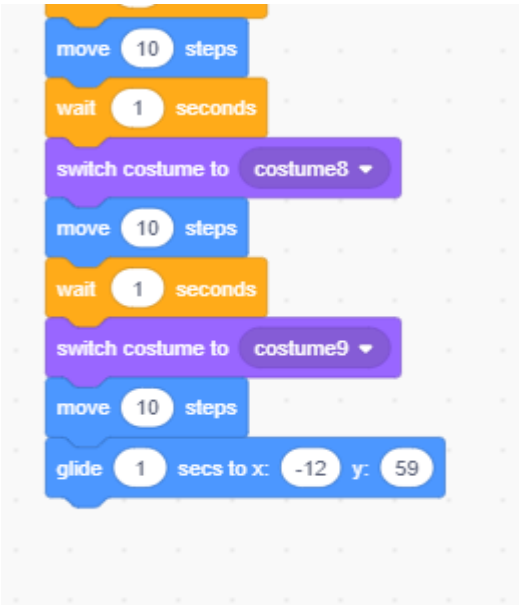
So first start the programme.
Look at the location where the cat stops and the programme ends.

Move the cat onto the bed.

Go to Motion.

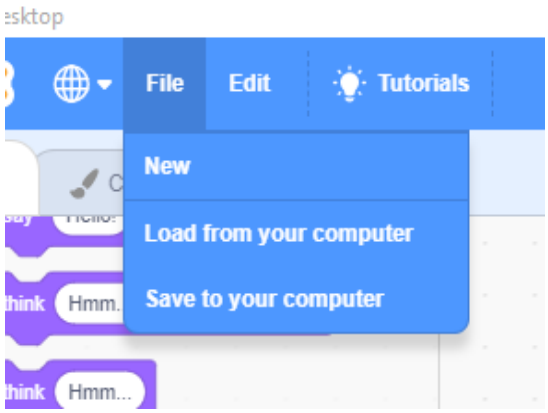
Find the **Glide 1 secs to X: Y:**

The new X & Y coordinates (of the cat on the bed) will be shown in the block of code. Move this block into the Script Area and position it in the correct location in the set of programme instructions (probably best to locate it at the end of the programme).



Click the Green Flag icon to start the programme.

Save your programme file with the name CatJumps by selecting Save to your computer in the **File** pull-down menu located at the top of the Scratch screen.



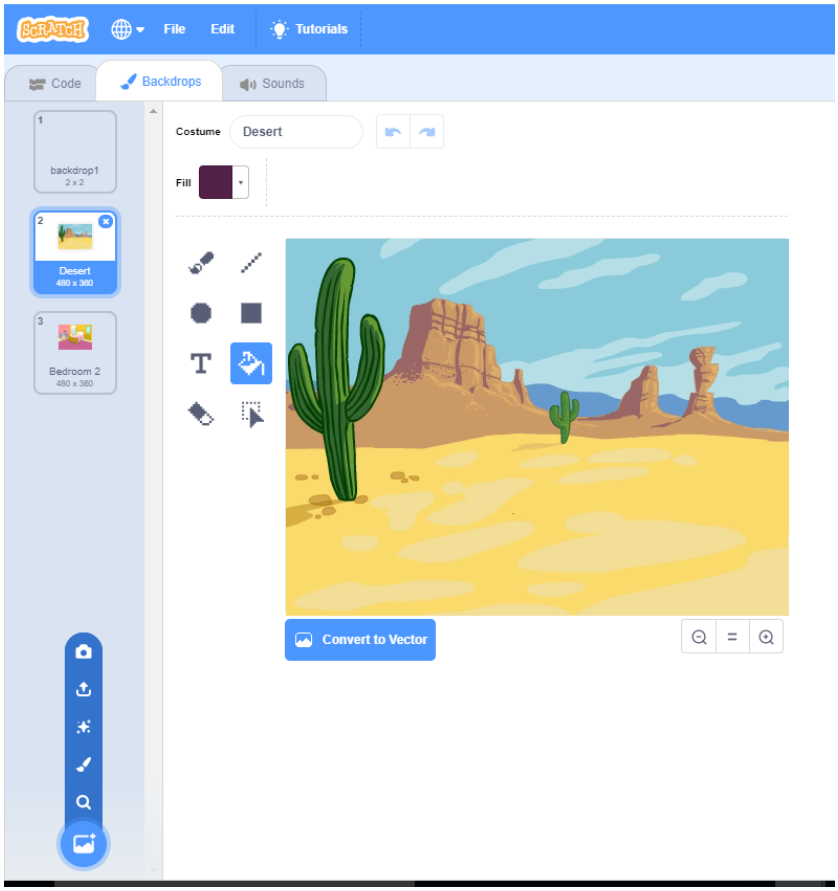
Exercise

Place some further blocks of code in the programme that will allow the cat to jump off the bed and walk a few steps.
Furthermore, ensure that the cat changes colour for each step of the remainder of his walk.

Exercise

Replace the backdrop with a new image from the Backdrop Library.

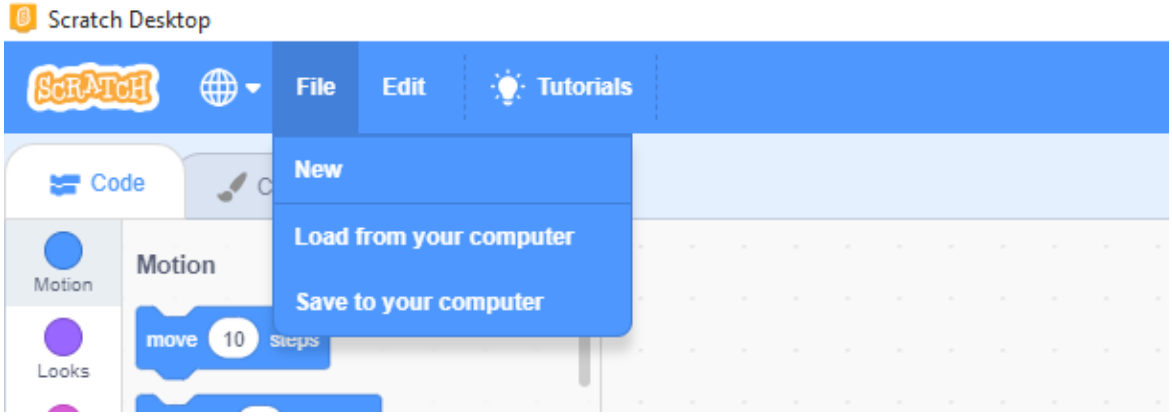
Notice that the new backdrop appears in a list of backdrops for your project on the left side of the screen.



To revert to a previous or new backdrop, just bring your cursor onto the appropriate backdrop image and click.
The new image is now highlighted, is the active element and becomes the backdrop that appears on the Stage screen.

Then re-do the instructions so that the sprite jumps on and off some object that is in your selected picture.
Note: the user has to click on the sprite image of the cat before it is highlighted (in blue) when code can be added.

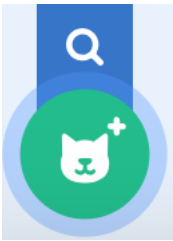
Save your new programme file with the name CatJumps2 by selecting **Save to your computer** in the **File** pull-down menu located at the top left side of the Scratch Interface.



Addition of a Flying Bird

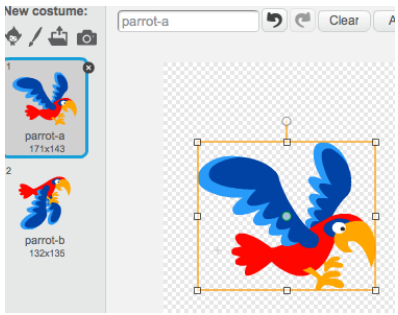
For this feature, we place a bird that continuously flies across the room near to the ceiling so as to be outside the reach of the cat

Select a new sprite by clicking on either of the icons below



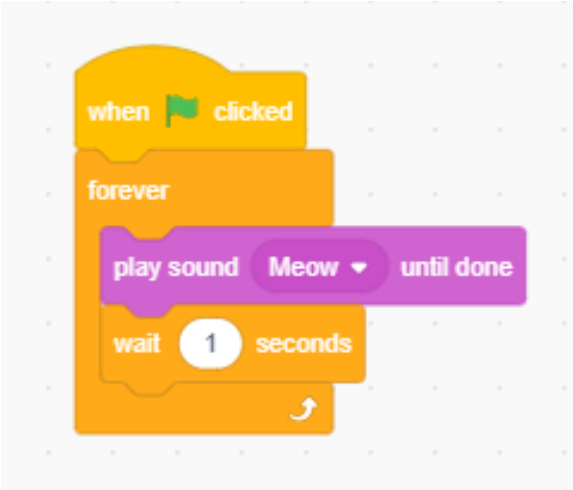
which appears in the listing below

Select a bird with two costume changes (so as to give the impression of wing movement).



Input the following code

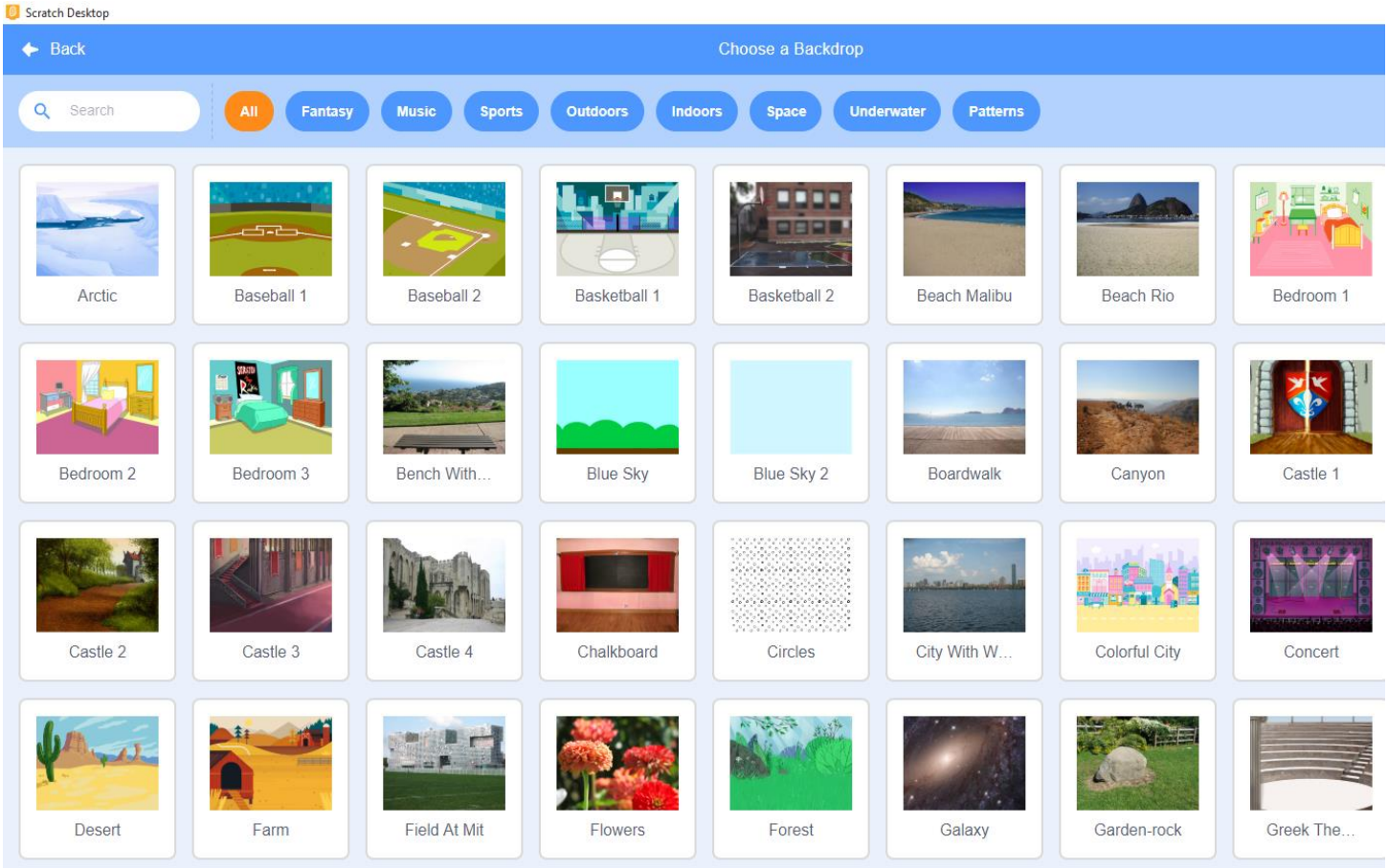
For the addition of sound, input the following additional script



Exercise: Walking along a road

Draw a new sprite e.g. a boy or girl
Draw different costumes (versions)
Draw a street scene (backdrop)
Write a programme to have the boy or girl walk along the street or road.
Draw in extra sprites that also have scripts allowing them to walk, run and cycle.
Place in a bird that is constantly flying back and forth across the sky.

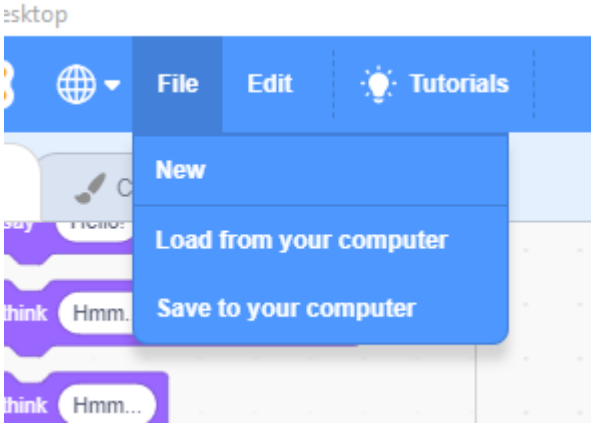
Note: In advance of the exercise above, demonstrate to the students how a photograph can be imported from the Internet or from the computer as a 'backdrop' and how it can be amended by using the Paint Editor tools.


Scratch Desktop
Back
Choose a Backdrop
Search
All
Fantasy
Music
Sports
Outdoors
Indoors
Space
Underwater
Patterns
Arctic
Baseball 1
Baseball 2
Basketball 1
Basketball 2
Beach Malibu
Beach Rio
Bedroom 1
Bedroom 2
Bedroom 3
Bench With...
Blue Sky
Blue Sky 2
Boardwalk
Canyon
Castle 1
Castle 2
Castle 3
Castle 4
Chalkboard
Circles
City With W...
Colorful City
Concert
Desert
Farm
Field At Mit
Flowers
Forest
Galaxy
Garden-rock
Greek The...

Lesson 7 - Barking Dog Chases Cat!

Let us have a barking dog chase after Fionn, our lovable cat!

Go to the File pull-down menu

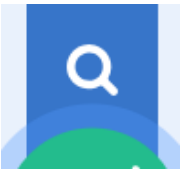


and choose Load from your computer

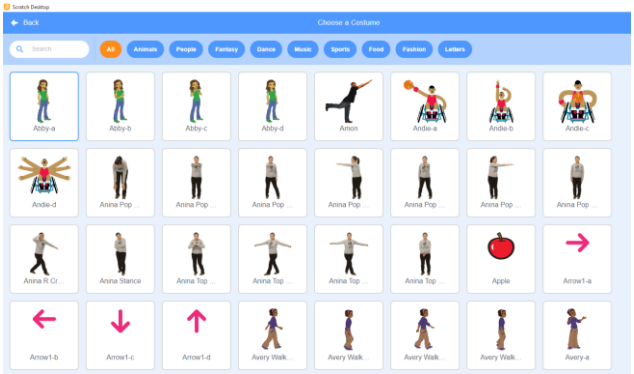
Then click on Select CatJumps2 to open your previous Scratch file

Go to

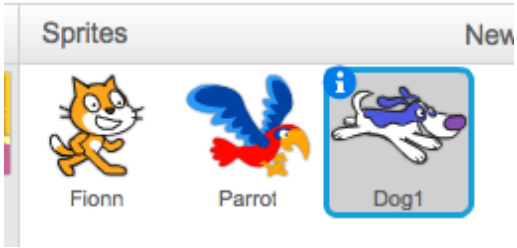
followed by



The following screen now appears:



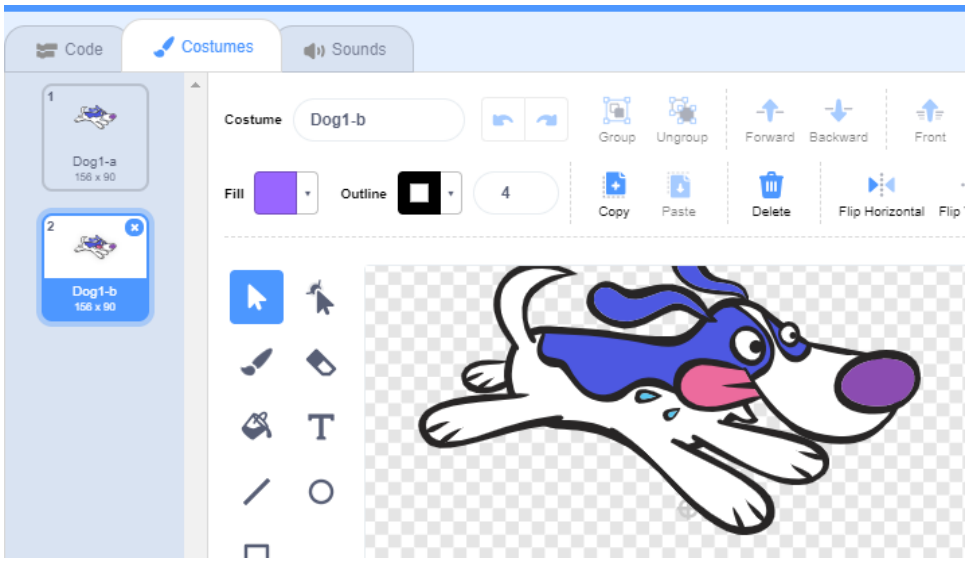
Then go to the Animal folder and select the first running dog image (dog1). Notice that the Dog icon is now highlighted in blue outline signifying that it is the sprite that is currently active.



We now have to build a script or programme to operate the dog sprite allowing him/her to run and to talk.

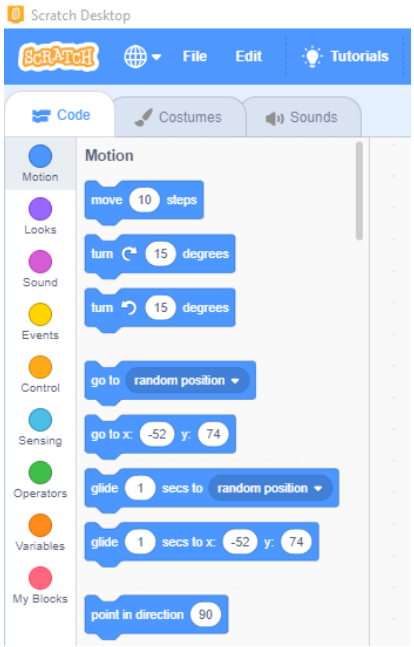
So in order to do so, we have at least one other version of the dog that will, in combination with the existing dog costume, gives the impression of motion.

Selecting Costumes shows that there are two versions already available for use in the programme.



Building a Script to operate the Dog

Go to the Blocks categories located on left side of the Scratch Interface.



Click on the Events folder

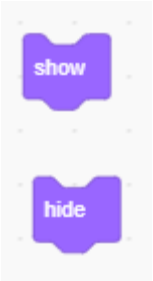
Place the following block in the Script area:

This will mean that once the Green Flag above the Stage is clicked, the sprite will follow the commands that are placed in the Script Area.

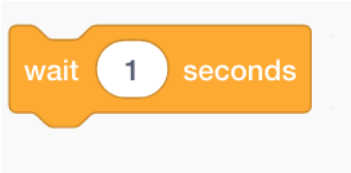
However remember that we now have two sprites, namely the cat and the dog.
We want to have the dog only appear on stage (screen) <u>after</u> the cat has jumped on the bed.
Hence we must have it hide when the programme begins, and only appear (show) in the scene at a certain location (X & Y) after a certain amount of time has elapsed (wait).

The following blocks in a combination sequence will allow the operator to undertake this task:

Show and Hide blocks in Looks

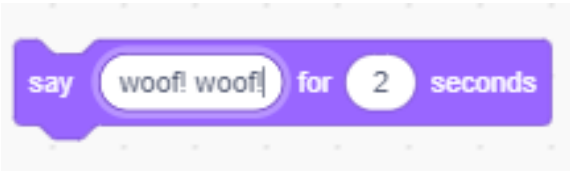


and the Wait block in the Control category

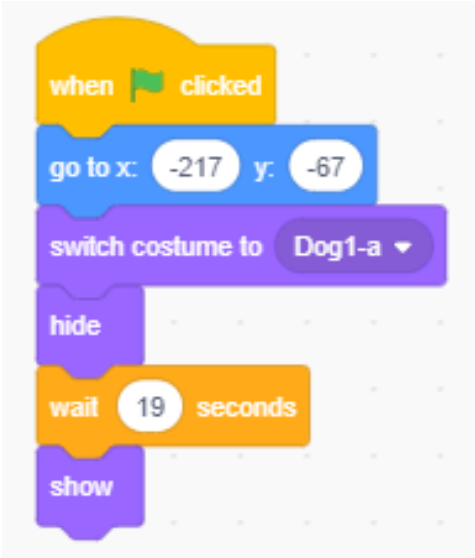


as well as the X and Y coordinates block in the Motion folder. The coordinates should reflect a position to the far left of the cat (on the same X axis) and close to the edge of the screen.

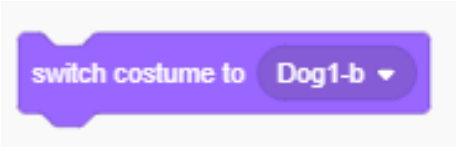
Get the dog to bark by selecting the say_____ block in Looks



To get the sprite to give the appearance of motion, we first have to place a Switch to costume command block towards the beginning of the script so that the first version of the dog, namely dog1-a, <u>always</u> appears once the programme starts.



After the first Move (in this case go to x ___ & y___ block) and Wait block located in the script, place in a 'Switch to costume' block with dog1-b taken from Looks.

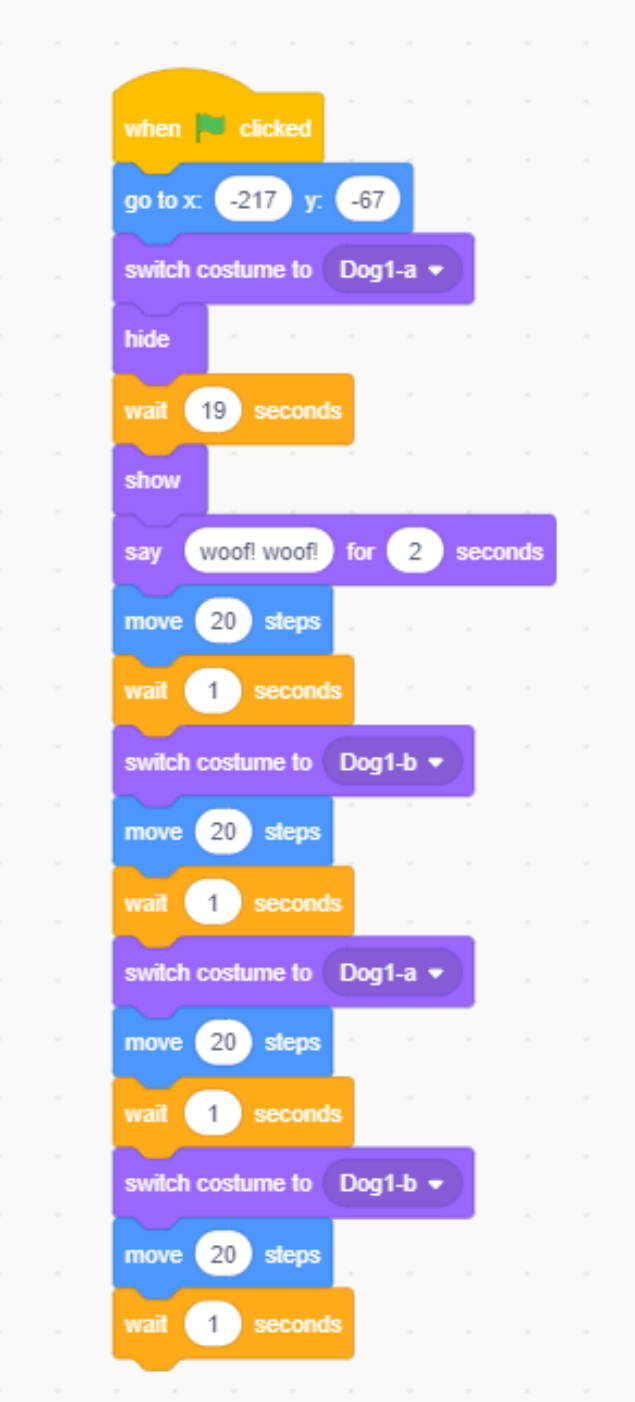


The option dog1-b is chosen by clicking on the black triangle icon located to the right side of this block

Then for the next Move and Wait blocks, place in a Switch to costume block with dog1-a.

Repeat this process all the way through the script, thus alternating in sequence between dog1-a and dog1-b.

The result should appear as is shown in the following screen:



A Scared Cat!

In order to give more authenticity to the project, Fionn the cat needs to shout "Help!" and put on a sad face when the dog appears and before it jumps on the bed.

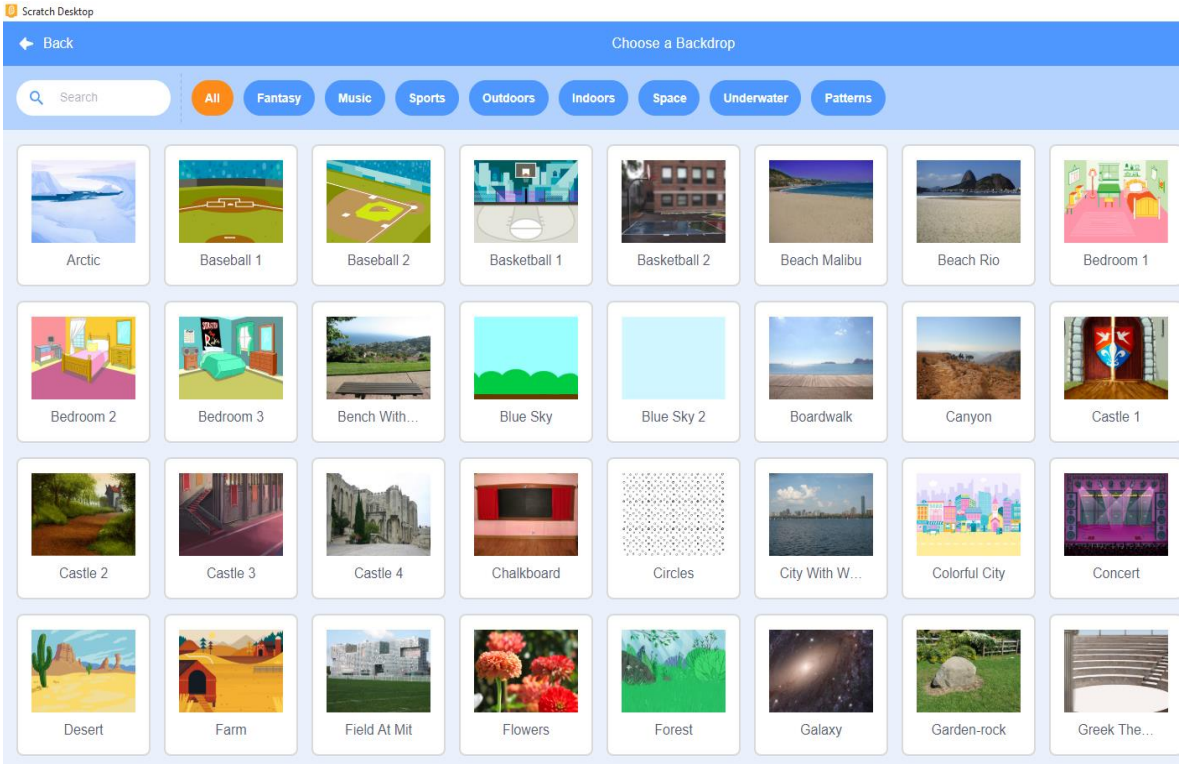
Whilst on the bed, why not have the cat's face full of tears!

So as an exercise get the students to create an extra costume change(s) for the cat and place in appropriate code within the Script Area.

Lesson 8 - Creating a Sprite

In this lesson students are introduced to the powerful Paint Editor facility of Scratch which allows users to create their very own sprites.

Click on the Stage icon at the bottom right of the screen and select the Beach Malibu scene from the Outdoors section of the Backdrop Library.

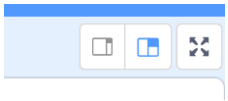


You can include the cat sprite in the beach scene. But relocate him/her to the bottom left hand corner of the screen, inputting a script that will have the cat walk back and forth across the beach.

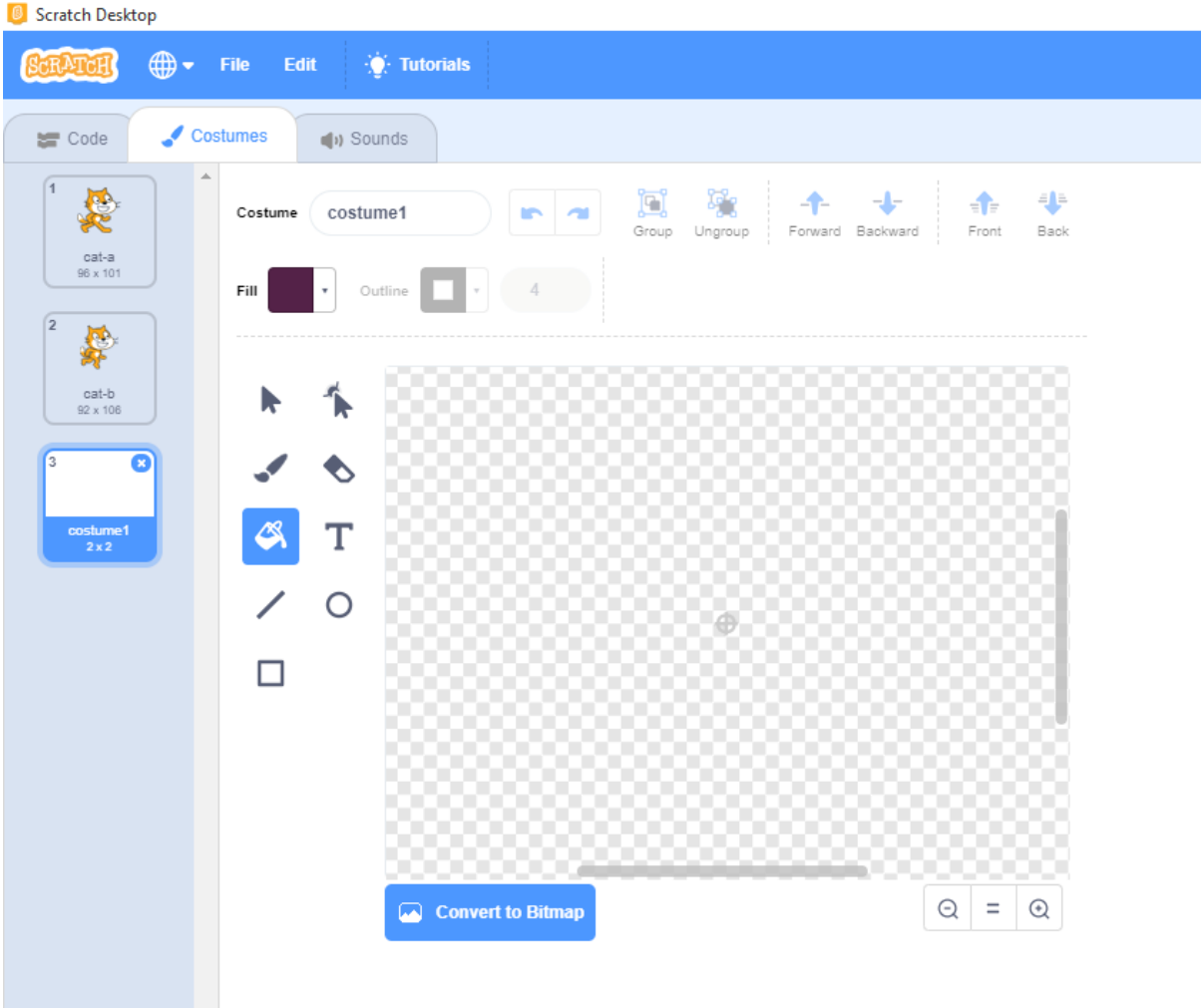
Now go to the new sprites section located underneath the stage on the right hand side.

Select the paint brush icon.

Decrease the size of the Stage by clicking on the middle icon on the top right side of the screen in order that the Paint Editor takes priority.



The **Paint Editor** is shown below:



Ensure that the Editor is in Bitmap Mode (bottom left hand side).

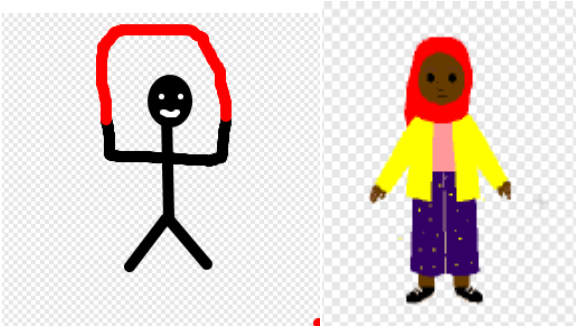
The built-in Paint Editor is a powerful easy-to-use drawing system.

Familiarize yourself with all its key features: colour palette box, erasure, brush. etc.

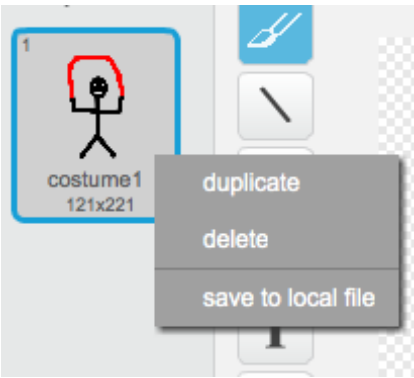
Now draw a skipping stickperson sprite using the colour palette (**Fill**) box

in combination the circle, line and brush options.
Ensure that the drawing is in the centre of the pixelated screen.

The sprite can be as simple or as detailed as you wish. So you can add on elements such as hair, eyes, mouth etc.

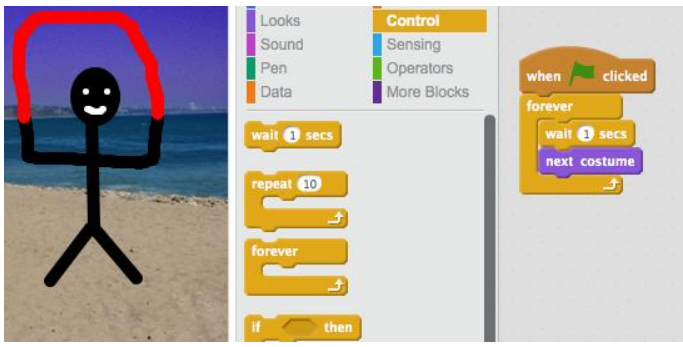


Once the stickperson drawing is completed, go to its small iconic costume representation. By clicking on the keypad or the top right side of the mouse, make a duplicate.



Edit this second costume version so that the sprite's arms, rope and other features are positioned differently to the first version.

Once completed use a simple script to give animation to the skipping sprite.



Lesson 9 - Cursor-controlled Sprites

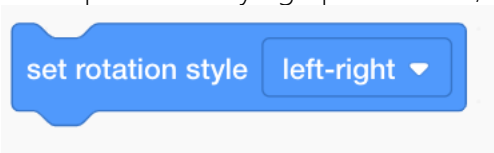
In this lesson students will become familiar with the programming of sprites to follow the movements of the cursor.

Tropical Bird on the Beach

Select a beach scene and a two costume sprite from the backdrop/sprite libraries:

Utilise the **Mouse Tracker** option located in point towards block_____ (Motion) and write up the following script

Change the numbers in the **wait** _____**seconds** block.
Notice the effect on the movement of the parrot.
To stop the bird flying upside down, use the middle option in the **Rotation** section:



Exercise:

1. Draw your own sprite bird with two costume changes using a stage contained in the Outdoors section of the Backdrop library.
2. Create a programme with the sprite controlled by the Mouse-pointer code.

Lesson 10 - The Psychedelic Sprite

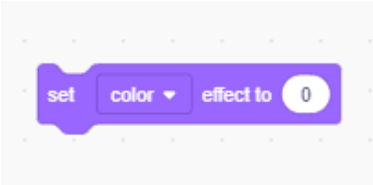
This lesson introduces students to the ***Sensing*** *category that will allow us in this case to change the Sprite's physical appearance by movements of the mouse (or keypad).*

Select a Sprite, preferably one a multi-colour body such as the Butterfly 2.

Go to the Code section of the sprite

Place into Script Area the code blocks as shown below:

The block below, as shown in the script above, is located in the Looks category



Go to the Sensing category to find the Mouse X and Mouse Y blocks which you drop into the **set color effect to** _____ as show in the script above.

Start the programme by clicking the Green Flag on the top hand corner of the screen.

Because of the instructions that you have typed in, which will move the mouse tracker left and right along the X axis, the sprite will change shape.
If you move the mouse along the Y axis, the sprite will change colour.

Exercise

Experiment with other effects by clicking the black arrow in the 'set… effect to' and choose other options besides Whirl and Colour.

Lesson 11 - Sprite Interaction

*This lesson will build on the familiarity gained in Lesson 9 with the **Sensing** commands to show how to code in an automatic interaction between two sprites using the **Touching** block of code.*

First, delete the cat.
Select Max from the People category of the sprites library.

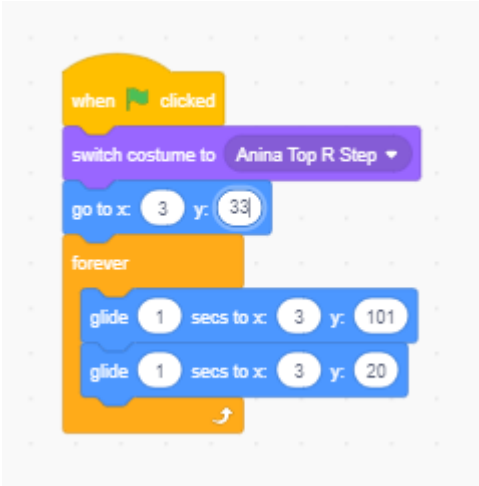
Then chose the Trampoline as a second sprite.

Under the Costumes section of Max, delete all versions of the Sprite except max-b and max-c.

These two versions should be fine to animate Max as she jumps up and down on the trampoline.
However you can of course use the Paint Editor to change one or more of the costumes in order to visually show the arms to be more raised even higher or lower. This can be achieved using the highlighted tool in the image below.

We of course want Max to jump up and down off the trampoline.

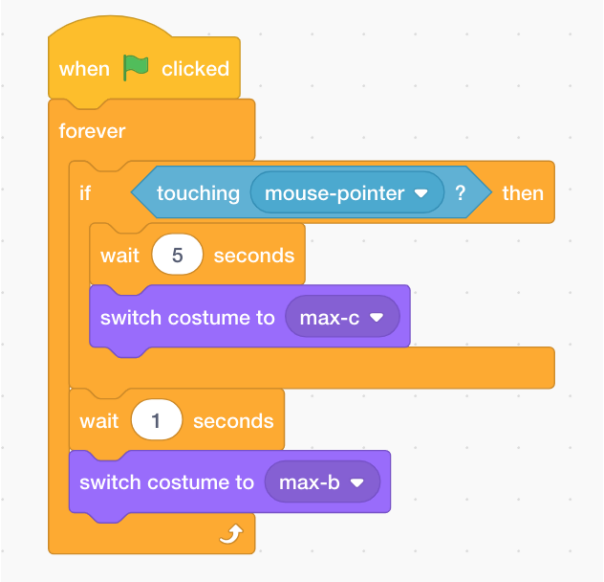
As we did in an earlier lesson, we achieve the impression of jumping by using the glide block from the **Motion** folder.
But this time we use <u>two</u> glide blocks as we want her to ascend to a certain point (Y axis) before descending to touch the trampoline.
This is achieved by the second glide block having a lower Y coordinate (number).



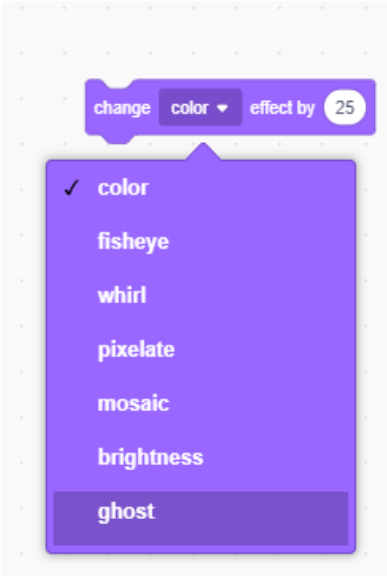
As Max touches the trampoline her arms will rise upwards. At the same time, the surface of the trampoline will it seems react to the weight of Max by stretching downwards until she jumps back up again.

We achieve this effect by the insertion of the **touching** command from the **Sensing** category in the scripts of both Max and the trampoline.

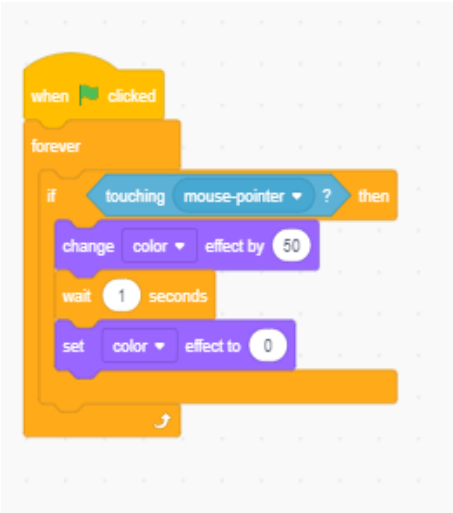
For Max:



For the trampoline, the use of the fisheye effect from the **Looks** category gives the impression of a bulge.



The script is as follows:



Please note that the fisheye effect has to be brought back to zero as otherwise the trampoline will continue to bulge more and more each time that it is touched by Max.

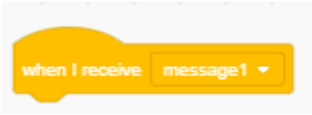
Exercise:

1. Make a Playground scene complete with a see-saw and a swing.
2. Draw in two children positioned at each end of the see-saw
3. Give the impression of the see-saw moving.
4. Draw in a child on the swing.
5. Give the impression of the swing moving.
6. Draw a boy or girl skipping. These characters can be simple stick people.

Lesson 12 - Two Sprites having a Chat

In this lesson students will become familiar with the ***Broadcast*** *command code which sends a message to some other part(s) of the programme (e.g. another sprite or a backdrop) instructing it to implement a change.*

Broadcast is used to allow communication between sprites and other elements of Scratch. The response from the recipient sprite will be initiated by a placing in a piece of code known as the



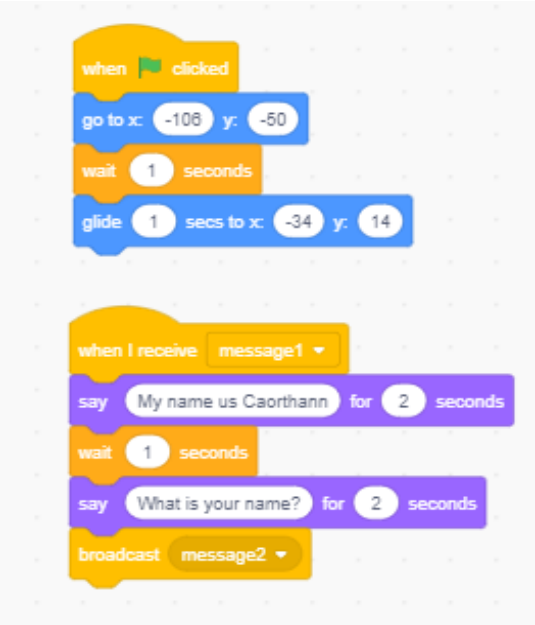
Select twice the Butterfly 2 sprite from the Scratch library. Choose also an appropriate backdrop.

Please note that if your sprite is turned the wrong way, click on the costume tab.

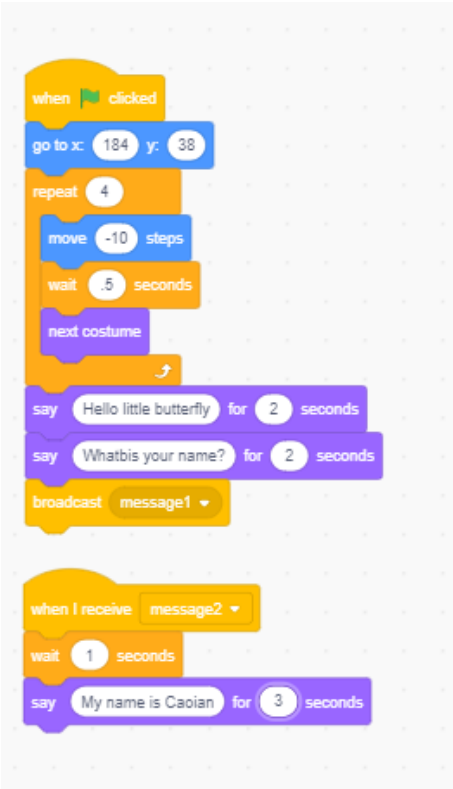
In the Paint Editor, click on the Flip horizontal (left-right) icon 

to reverse the direction that the sprite is facing.

Place the following code in the first Butterfly 2 sprite:



Place the following scripts in the second butterfly sprite



Continue with this type of coding until the conversation is completed.

Exercise:

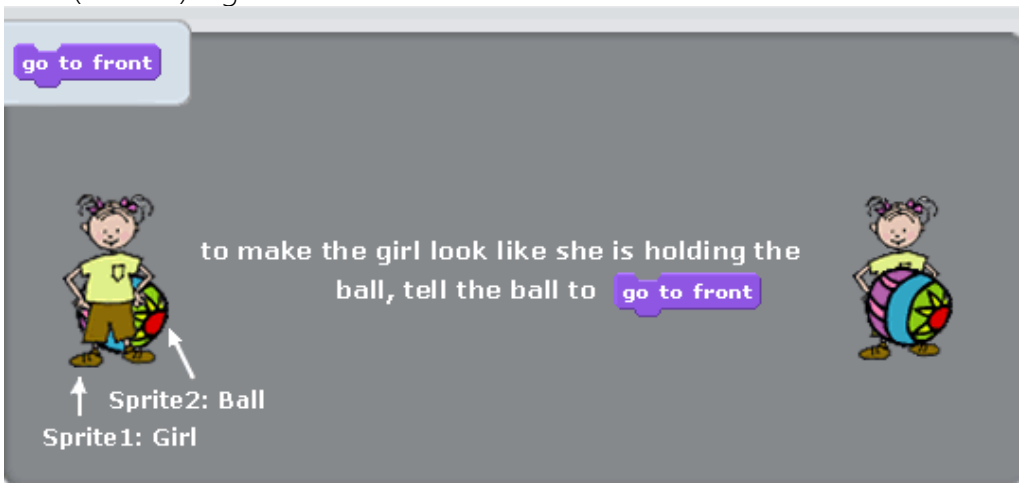
1. Create a programme where two people are chatting after meeting up whilst out walking.

Lesson 13 - Creating a Coral Reef

Lesson Objectives:

- To understand the significance of the **If** command (with its cause and effect impact) that is contained within the **Control** category
- To use the **Random** block of code (**Operators**) to change sprite positioning
- To use **Sensing** commands to give different effects when sprites come into contact with each other
- To add in a **music** score.

Explain that the '**go to front**' is very useful where there are two sprites positioned on top of each other and where you want to have one of them appear always at the front (or back) e.g.



The '**go back _ layers**' blocks of code are applicable when you have multiple objects on screen such as in an aquarium, populated with different fish and plants, where you need to position some objects to the rear whilst others would be to the front when they cross each other's paths.

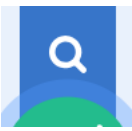
Part 2: Creating a Sea World

So let us now create our own colourful tropical sea world.

Go to New within the File menu
Delete the cat sprite
Select the Underwater stage within the Nature folder

Go to the sprite plus menu 

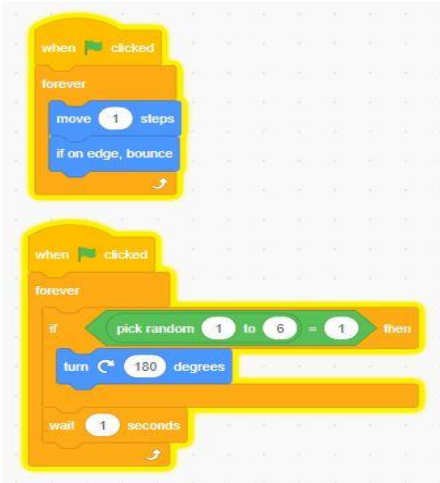
and click on



to gain access to the sprites library.
From the animal folder, choose the Shark sprite.

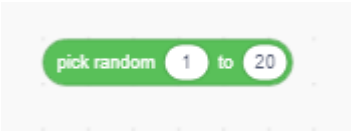
Choose the small Fish sprite option from the Animals folder and duplicate it.
Delete all but one costume for each. Please ensure each of the two fish sprites uses a different costume in order to ensure that each represents a separate species.

Input the following code:



Rather than have the sprites change direction (i.e. only turn around) when they 'bounce' off the edge, which has been solely used up until now, the latter piece of code allows the sprites to turn around (e.g. at 180 degree or right-to-left turn) randomly within a certain range (e.g. 1 to 26).

Use of the

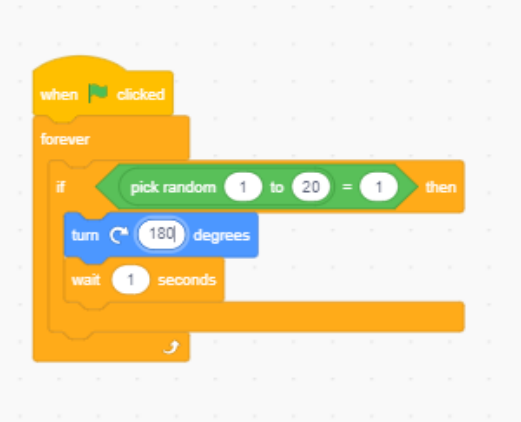


in combination with

located in the **Operators** folder signifies that this change in direction will occur once in every twenty. So a lesser number range (e.g. 1 to 6), will lead to the turning around movement of the sprite occurring more frequently.

The use of the powerful IF THEN command in the script means that if a certain listing that is inputted occurs, then the corresponding stated reaction contained within the code will automatically take place.

Used of Forever with the If then blocks of code ensures that this change in direction of the sprite will be a continuous process.



Exercise:

Get students to experiment with changes to the random numbers and with the turn _ degree block.

Use the same code as with the previous sprite (above).

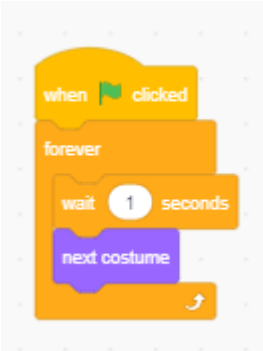
Addition of Seaweed & Anemones

To give the programme a more authentic ambient of a living coral reef, anemones, seaweed or coral should be added to the scene.

Draw a seaweed sprite by clicking the icon 

Then use the paint brush tool from the menu on the left to create a seaweed plant. Create two or three costume versions for the seaweed sprite with each version having their blades(fronds) in different positions in order to give the look of a swaying movement when animated through coding.

Input the following code:

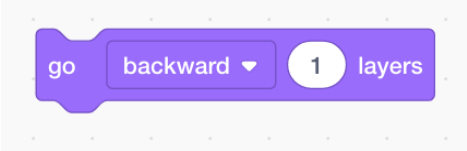


Draw one or two anemone sprites.

Creating a 3D effect: Using 'Layers'

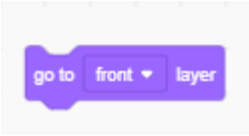
However, you may notice that the seaweed plants are all positioned at the front, thus blocking off the views of the moving fish.

To reposition some of the seaweed plants towards the back, thus giving the effect of the fish moving in front of and between individual plants, go to Looks and select the following block:



Each layer represents one specific sprite area of movement. The higher the number, the more marine sprites will be found swimming in front of the seaweed.

To have the seaweed located at the front of the screen, use the following motion block:

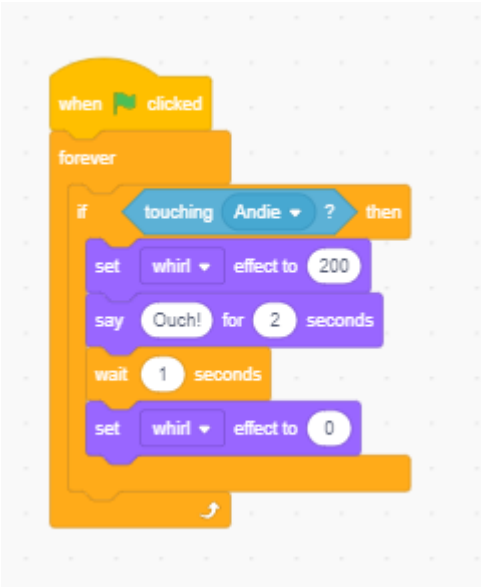


When Sea Creatures Collide

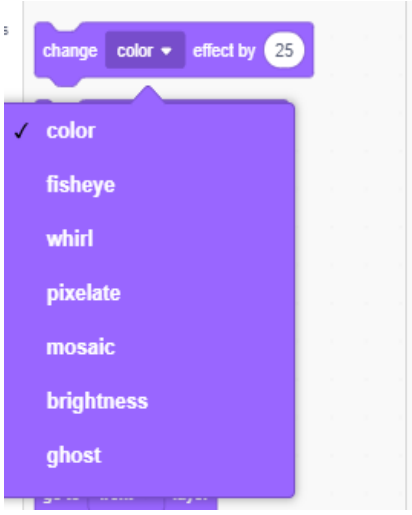
The Sensing category contains blocks of code that allow sprites to interact with each other.

Select a sprite whose code allows it to actually cross the path of another sprite.

Add the following code to this sprite:



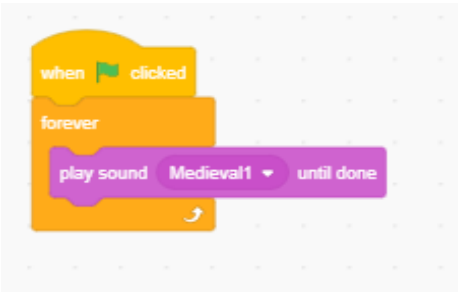
'Whirl', the aforementioned special effect used in the example above, will change the physical shape of the sprite. The higher the number, the greater the distortion to the sprite.
But it is important to add on to the block of code, after a wait of one or two seconds, the additional block set whirl effect to 0 or clear graphics effects. Otherwise, the sprite will remain permanently distorted.
The 'whirl' is one of a number of special effects in this command block located within Looks as shown below:



Get the students to experiment with these options and in changing the number in the set whirl effect to __

An Ambient Aquarium Sound

Finally, select a suitable music score to match the mood of slow moving fish.



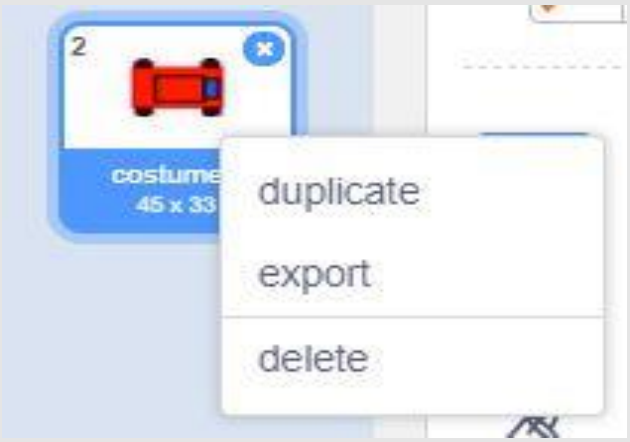
Exercise

Get the students to draw:

a) A woodland or jungle scene populated by many different moving animals such as birds, mammals, and insects or
b) A street scene populated by walkers, shoppers, cyclists, buskers etc.

Saving one's own Sprite creation

Sprites created in one project can be saved so that they be used in other projects. Bring the cursor onto the sprite that you want to save.



Select export this sprite.
Then chose the location of where you want to Save the sprite (e.g. desktop, Scratch Sprite 'Things' folder), followed by inputting Filename and then Save.

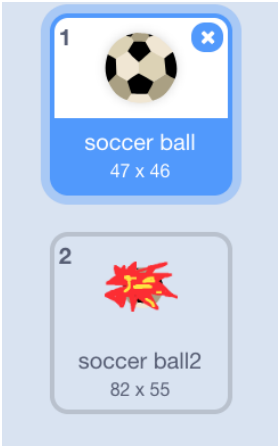
Lesson 14 - Target Ball

Lesson Objectives:

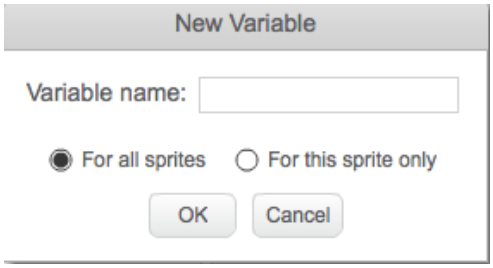
- To learn the use of **variables**
- To use the **Random** block of code (**Operators**) to change sprite positioning.
- To show how an action can be programmed to occur when a sprite is clicked.

Target Ball - Coding Plan Summary The ball moves at speed randomly across the stage. User tries to target the ball, registering a score for each hit.

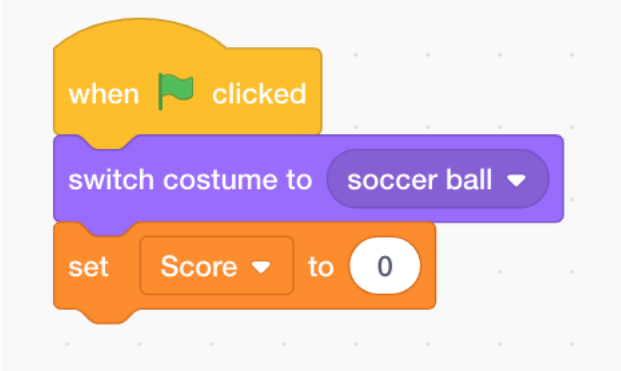
Make a second costume of the ball sprite that gives the impression that it is exploding.

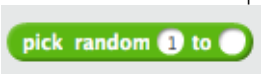


Now do to the **Variable** folder, select Make a Variable option.



Type in Score under variable name and choose For all sprites.
Then place the following block into the Script area setting the Score to zero.



Now we want the sprite to move across the stage in set pattern. So we use the block 

in Operators for the X (horizontal) and Y (vertical) coordinates. Random numbers are numbers that do not form any pattern and are unpredictable.

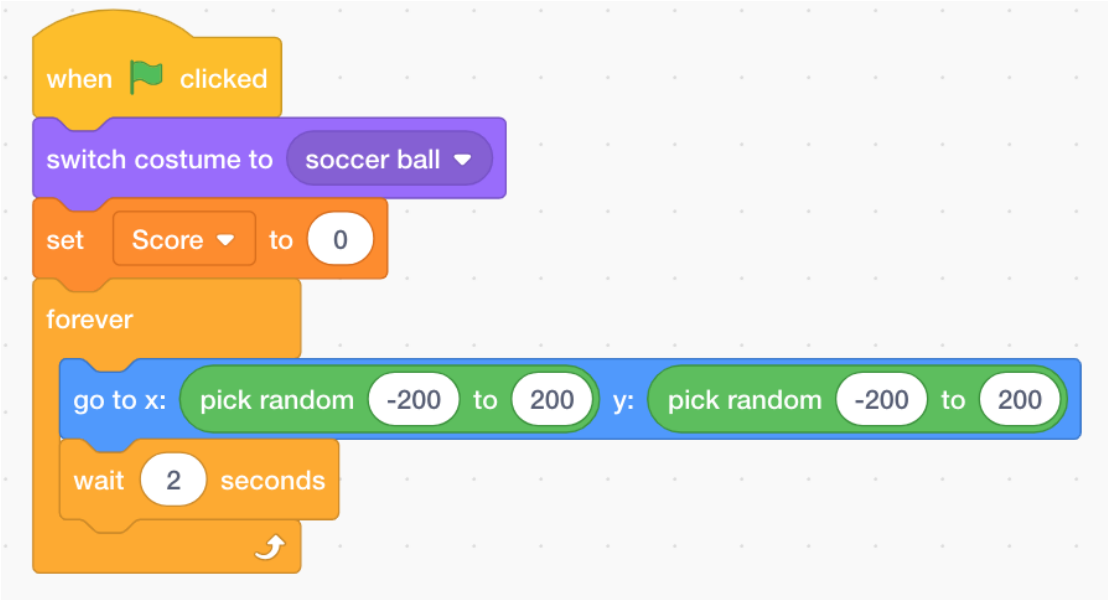
But the two white boxes in 

allow the user to set a range for X and Y. So in this case we will have the sprite move across the maximum area of the stage by using the following block

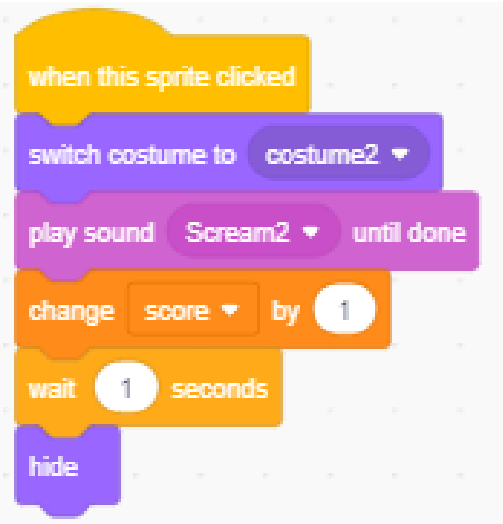
to ensure that the random movement is on-going use the Forever block.

However it has to be used in conjunction with 

. Otherwise the speed of the cat would make it very difficult to register a score.



Now we input a separate piece of code into the script that gives a score every time that the ball is hit.



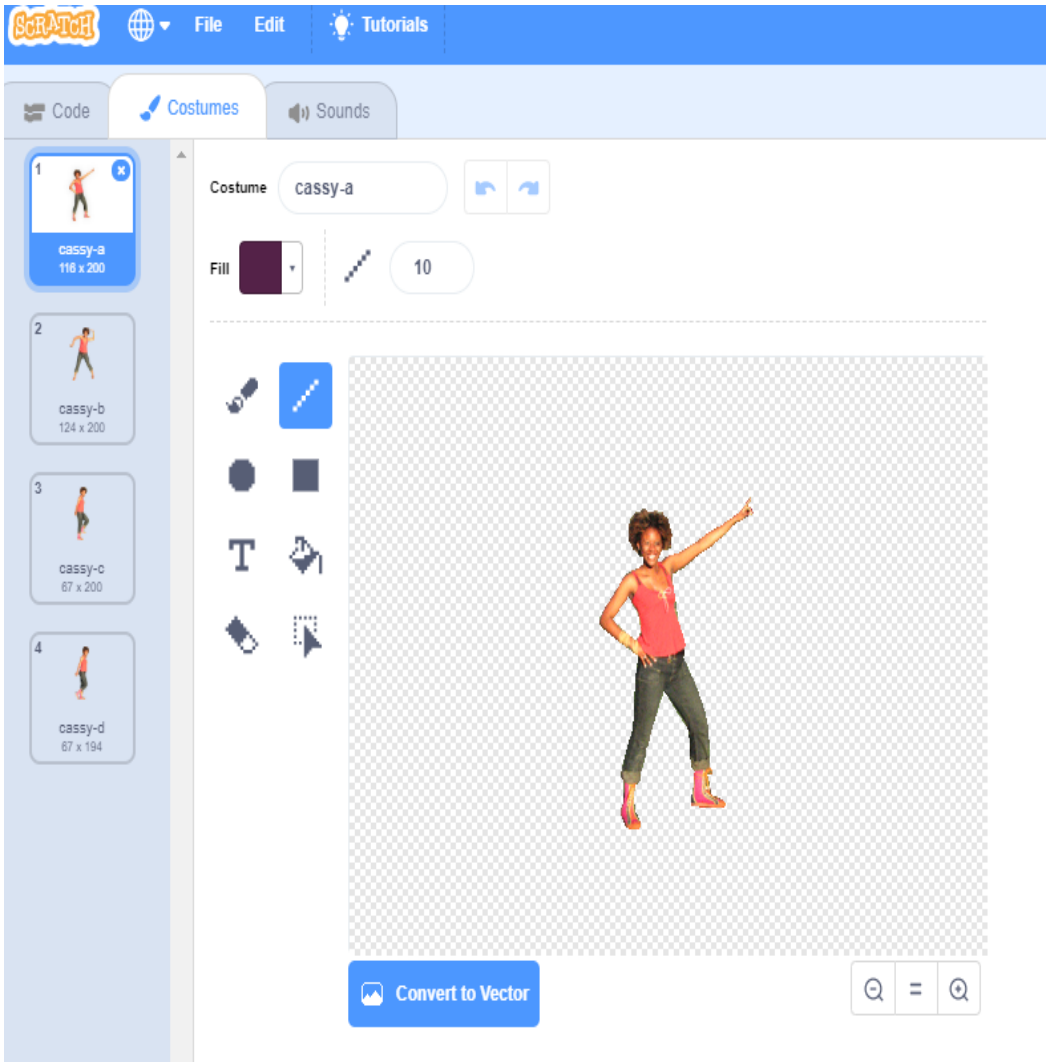
The use of Show and Hide above signifies that the ball disappears every time that it is hit. Of course this means that the game is very short ending when the sprite is hit once.

So in order to make the game longer and more fun, copy the sprite five or six times. Then give each of these ball sprites a different set of X and Y coordinates in order that they are all in different locations when the game starts.

Lesson 15 - Dancing Sprites

In this lesson, students learn how to create a dance animation through a combination of commands from the Looks, Control and Sound categories.
First, click on the Stage icon located under the Stage section of the Scratch screen.

Go to  and choose an appropriate Backdrop for the dancing sequence you are about to make.
Then choose a new sprite that has a choice of different positions.
Next, go to the Costume section and import a number of different versions (costumes) of the same sprite.



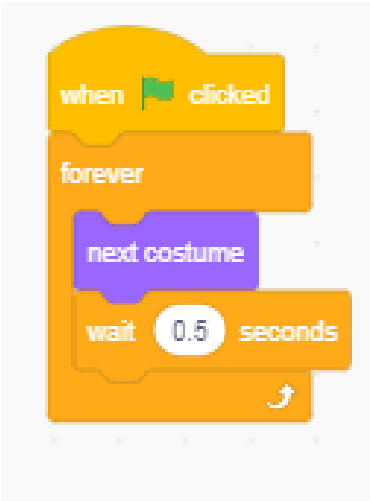
Now let's animate!

First select

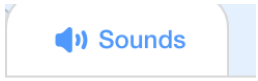
followed by the Forever block.
This command will let the dance programme run continuously until the user selects the red circle **Stop** icon located at the top right side of the Stage screen

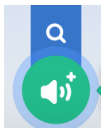
Next select the Wait command which the user should reduce in duration from 1 second (secs) to 0.5 seconds as otherwise the sprite will be dancing too slowly.

Go to Looks, select Next Costume and place it within the Forever block.



Run the programme.

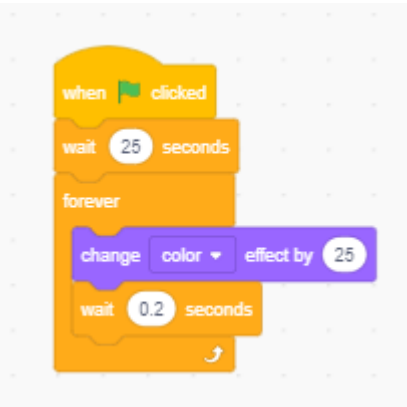
In order to add sound, go to 

, click on  and select an appropriate dance sound such as Hip-Hop in Loops.

In the sprite's Script Area, add the following:

Run the program.

Now add the following script to introduce a disco light effect to the stage



Exercise

Get the students to start a new programme with a different backdrop, different music and three dancing sprites.
Encourage them to draw for instance a few traditional dancers in different poses and stagger (use Wait block) their appearance onstage. Locate (from the Web etc.) some appropriate music and make your very own musical!

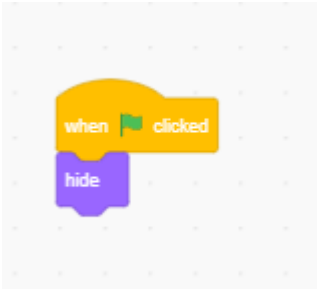
Lesson 16 - Drawing Shapes

In this lesson students learn how to understand how to draw geometric shapes using a series of scripts.
They also learn the concept of Chance using a Random selection block from Operators.

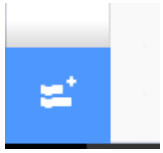
Creating a Script to draw a Square

This time we do not need a sprite. But as all of the Starch commands cannot function without a sprite, we need to hide it.

Use the following commands to make the sprite disappear:

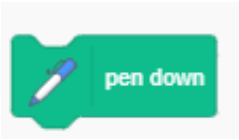


In order to draw in Scratch, we have bring the cursor to the bottom left corner of the Interface and select the add extension option represented by the icon

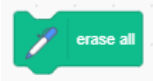


Then click on the Pen section which will automatically add this option to your Block thematic categories.

From Pen, select a pen size and colour.
We also give instructions for the pen to a) start 

and b) finish 

drawing.

It is also important to clear previous drawings from the stage area and to recommence the drawing process on a blank canvas once the Green Flag is selected to restart the project.
So use the command 

Adding on **wait 1 second** block will allow the viewer to better appreciate visually both the commencement of the drawing and the actual process of the formation of a new object.
To ensure that the geometric object that we are about to draw, namely a square, is of sufficient dimensions for easy viewing, use a sizeable number of steps from Motion e.g. 100 steps.

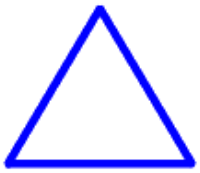
To code in the command that will make the four lines form a box, we first use the Repeat command. In the construction of a square it is **Repeat 4** times.

Please note also that for each geometric shape, the angle is proportionate to the number of sides i.e. 360 degrees divided by the number of sides.
For instance, a square is 360 divided by four = 90;
A triangle is 360 divided by three = 120 and
A circle is 360 degrees divided by 360 = 1

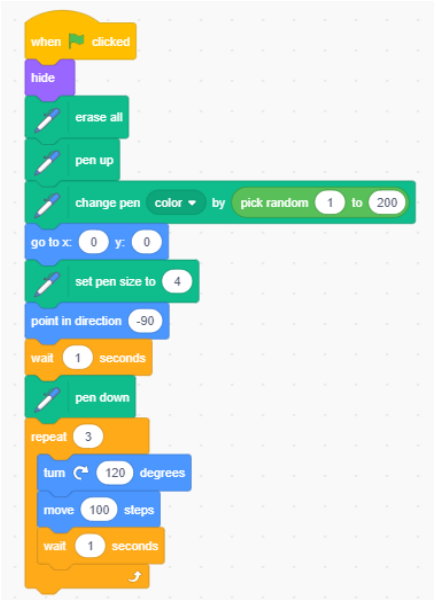
So we have to turn the lines using a degrees block in the Motion category

We can also change the colours for each drawing by picking the random option in the Operators category. The spectrum of colours go from 1 to 200. Hence choosing a high random range e.g. 1-200 (see image below) for colour variation will allow the programme to randomly select from the full range of colours each time that it is run.

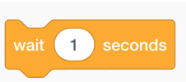
Let's draw a **triangle**



First ask how many sides are in a triangle?
So what then would the angle (degree) settings be?

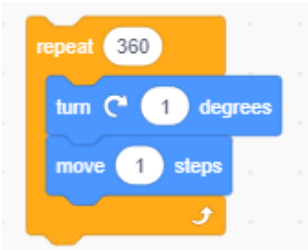


Let's draw a **circle**.
How many turns (degrees) in a circle?

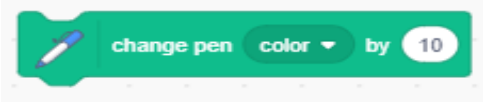
So turn one degree at a time. Do not use the 

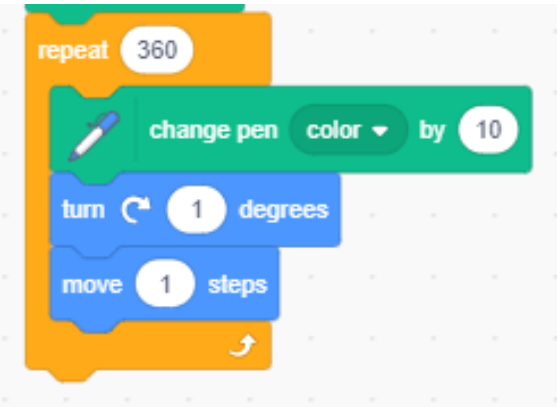
block in the script. This is because using it will mean that it will take 360 seconds for the circle to form.

Add inside the loop as follows:



to give





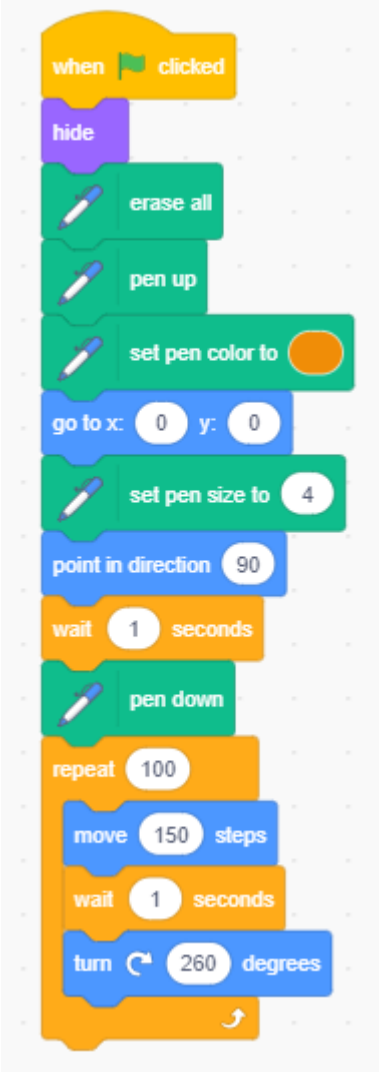
and enjoy the colourful effect.

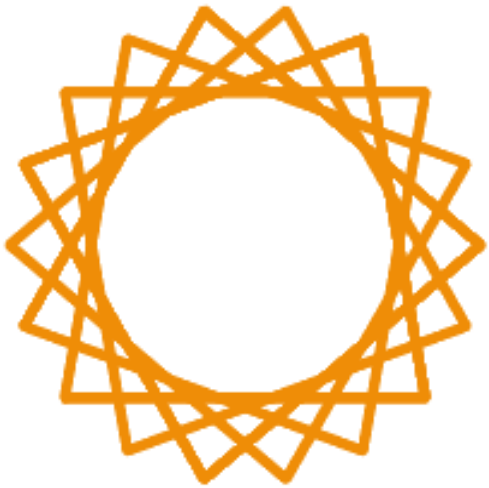
Exercise

1. Draw a Pentagon
2. Write a programme that draws three different shapes that appear at different times at different locations on the Stage.

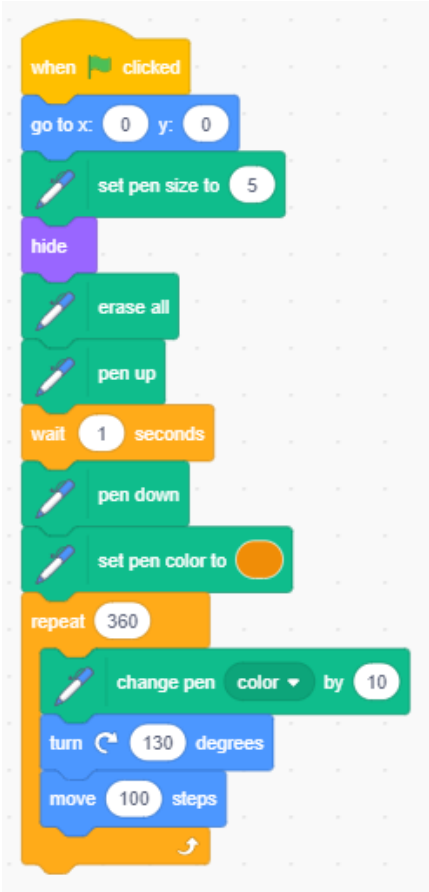
Shapes - others

Experiment with different angles and 'repeats' in the programme.
For instance, select the code below and admire the results

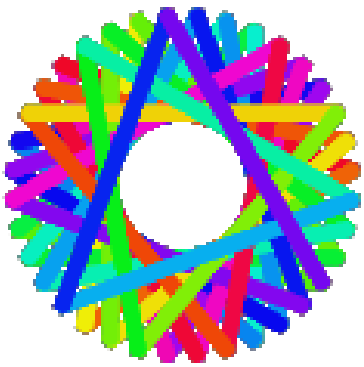




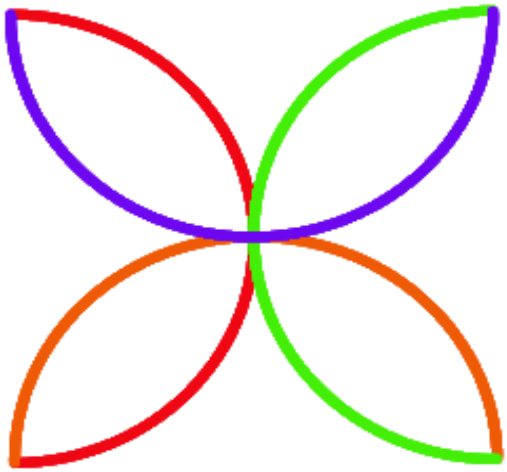
Input the following code



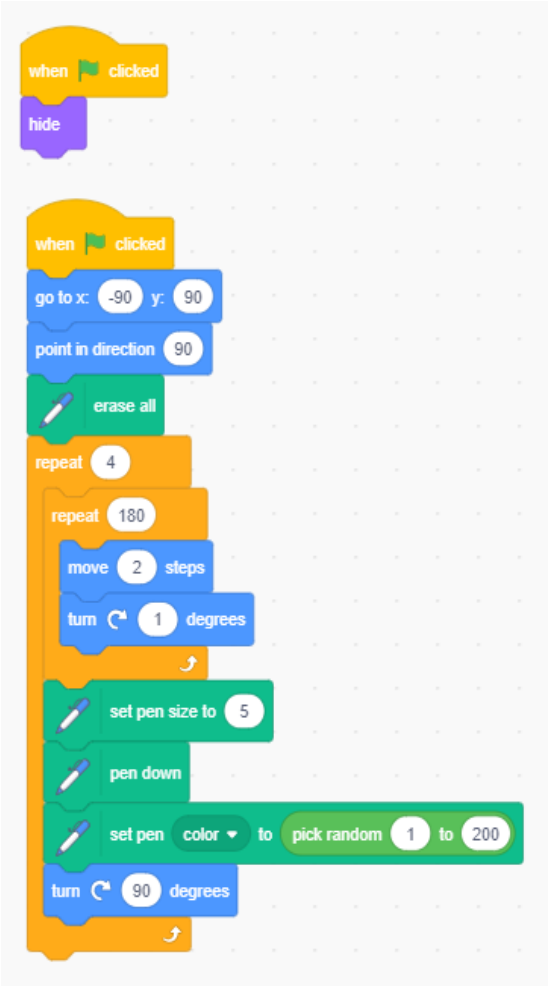
to give



Create A four petal Flower



The effect is achieved by making four half circles (180 degrees) and turning right (90 degrees) at the completion of each one.

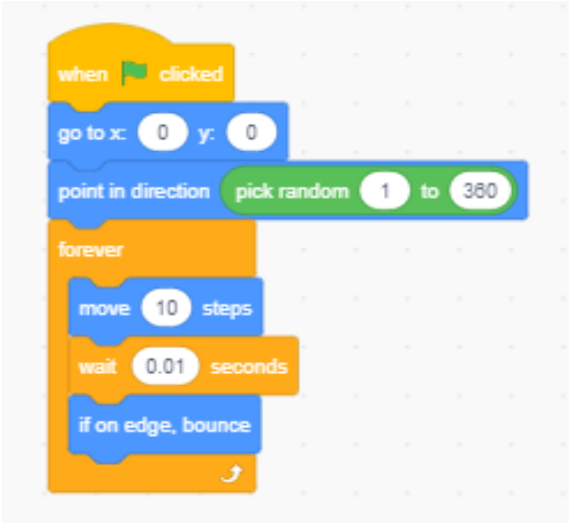


Lesson 17 - Bouncing Ball

In this lesson, students become familiar with **Degrees** and **Variables of Space.**

First draw a Ball using the Paint New Sprite option

Then start writing a script as follows:



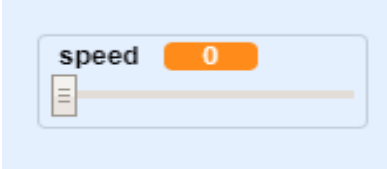
Experiment with the speed of the ball by increasing and decreasing the number in the move _ steps box

We can also control the speed of the ball during a game by using the **Variable category.**

Select Make a Variable (for all sprites) and type in the word Speed in the option Variable Name

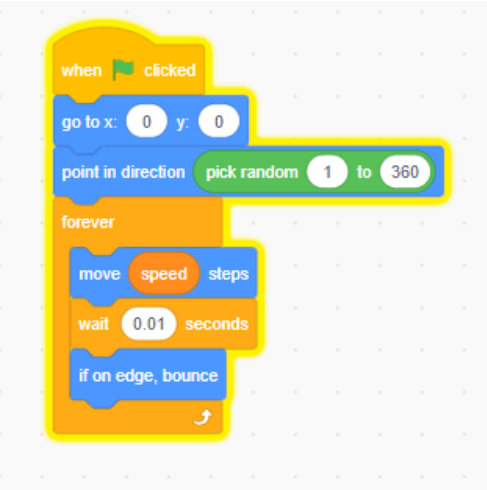
A Speed box appears on the Stage

Right click on the icon (if using the mouse) and select the Slider option, which allows the user to adjust the speed of the ball when in motion.

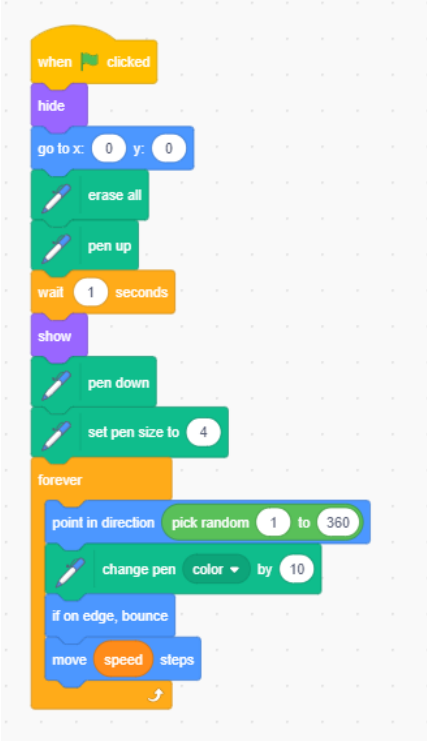


Drag the Speed block from the Variable category and position it in the white (number) box in the move _____ steps block.

The script code now reads as follows:



Generating a Comet tail on the Bouncing Ball



Exercise

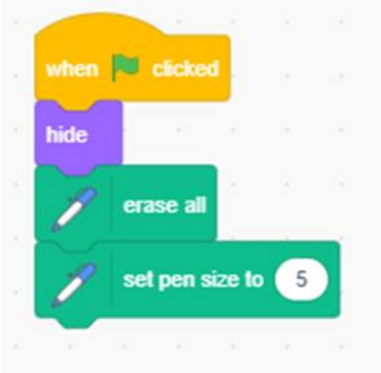
1. Write a programme with a number of bouncing balls
2. Write a marine-themed programme that has a number of different bouncing sea creatures such as jellyfish and sharks.

Lesson 18 - Drawing Free Hand

In this lesson, students learn how to create a programme that will allow a user to draw free hand.

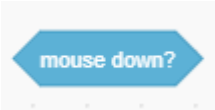
First, as with the programmes to draw shapes, the user has to make the cat disappear (hide). This is because it would be very difficult to draw properly with a sprite such as a cat being used as the drawing tool.

The user then uses the clear block followed by the set pen size block in order to have a drawing pen of sufficient size.

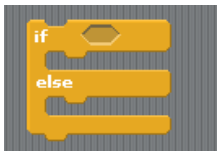


The user can become the drawing pen by coding in his/her movement of the mouse interface unit as we did in an earlier lesson when we had the movement of an animal sprite being controlled by the mouse tracker.
But we need now to ensure that the programme takes account of the fact that the user's movement of the motion is not continuous. Otherwise we would have the screen being filled by one never-ending line.

Hence we have to code in a set of instructions so that the artist can raise his/her pen at any time to stop drawing and to continue drawing at some other point on the screen.
This we achieve by coding in a script that is primarily based around the



from the Sensing folder and the



from the Control folder

This will give

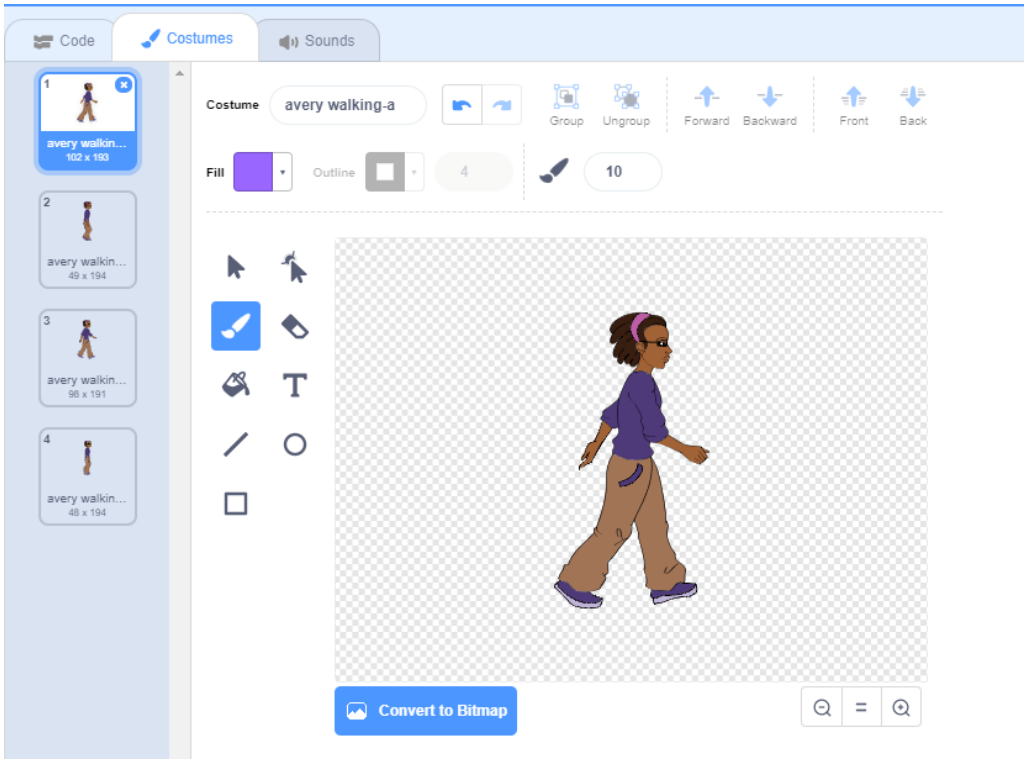
The final script will read as follows:

Lesson 19 – Walking the Dog

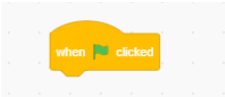
*In this lesson students are introduced to the **Timer** feature, allowing changes to backdrops and sprites to occur at certain designated times.*
Students will also use the 'point in direction' block, which controls the direction that a sprite faces and moves.

The lesson is based on a girl or a boy taking a dog for a walk-through different landscapes.

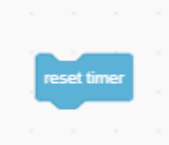
Select from the People section in the sprite library an appropriate sprite with multiple walking costume changes.



Go to the backdrop (backdrops) library of the Stage option and select three different scenes from the Outdoors folder.

Under the 

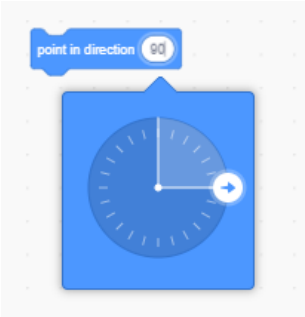
command, we add two additional blocks over and above what the user is familiar with from previous lessons.



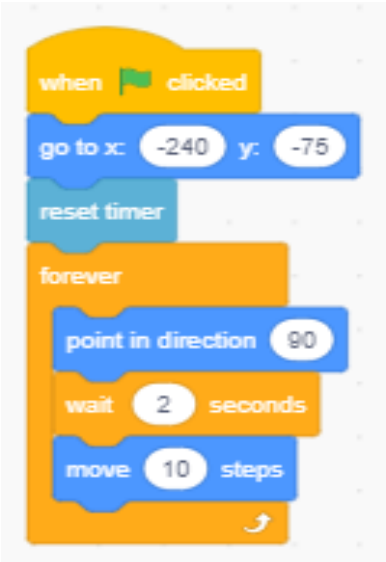
First, choose the

from the Sensing Folder which brings the time back to zero when the Green Flag is clicked.

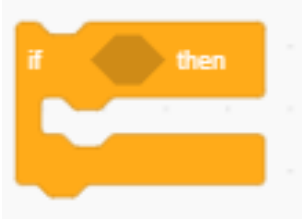
The second additional feature is the



block which codes in the direction that the sprite will be facing and walking. This is important as the storyline will include the girl in this case turning and walking back whens she gets to the end of his journey.
Thus the initial script would be

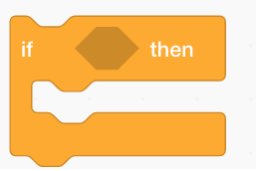


We now place underneath the above blocks additional code that will reposition the sprite back at the extreme left (end of screen) along the X axis when she reaches the extreme right of the screen. This is because we want the girl to be seen walking through different backdrops (from scene to scene) as if it is one long continuous journey.
To do this we use a block of code that allows something to happen when the end of the screen is reached. This is the

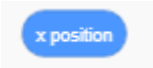


command.

Go to the **Operators** folder and select the **Greater Than** option block 

which is placed in the spacing between the 

block in the above script.

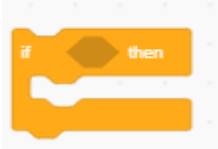
Type in the digits 226 in the right hand box which represents the pixel number (X axis) on the far right of the screen.
So place 

from Motion in the left hand box of

We use the 

as the sprite moves only along the horizontal (X) axis.

From the same folder take 

and insert within the 

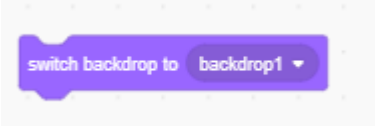
block.

Notice that the X coordinate has a high minus number that will, when the programme is running, give the impression of the girl sprite walking onto the stage (screen).

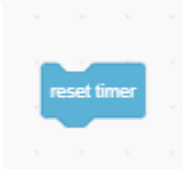
The code will now appear as

Now we go to the Stage script where we will code in three backdrops that will change based on a time parameter control.

The main commands for these blocks are:



(Looks)

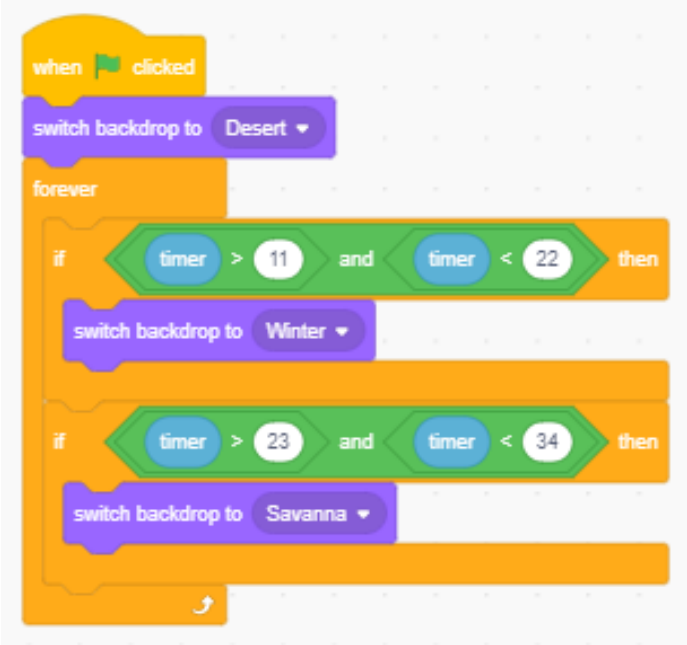


(Sensing)

and combining

(Operators)

to give:

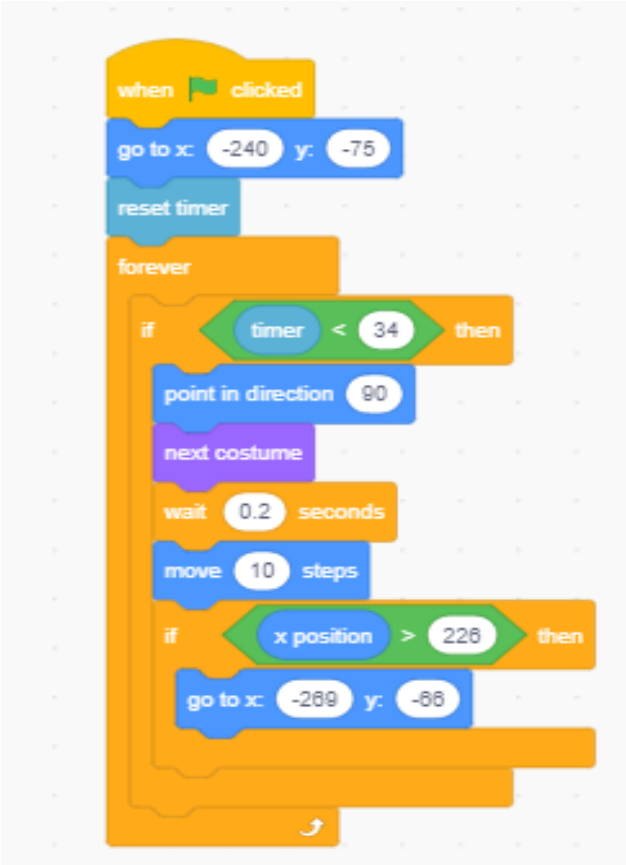


Please note that the numbers inputted are based on the length of time it takes for the girl to move along the X axis from one end of the screen to the other. Obviously it is critical that the coder synchronises the timer with the re-positioning of the sprite back to the left of the screen every time that she reaches the end of the screen. This will probably involve a bit of 'trial and error' on behalf of the student.

So in order to complete this part of the programme, we will input an additional piece of coding that will limit the amount of time that the girl sprite is walking facing right so as to allow her to turn around when she gets to the end of the last backdrop and walk a few steps back (to the left).

First we measure the amount of time it takes the sprite to walk from the beginning of the first scene to the end of the last scene. In this particular project, it was 34 seconds which is inserted with the Timer into the if ____then block

and placed into the code as follows



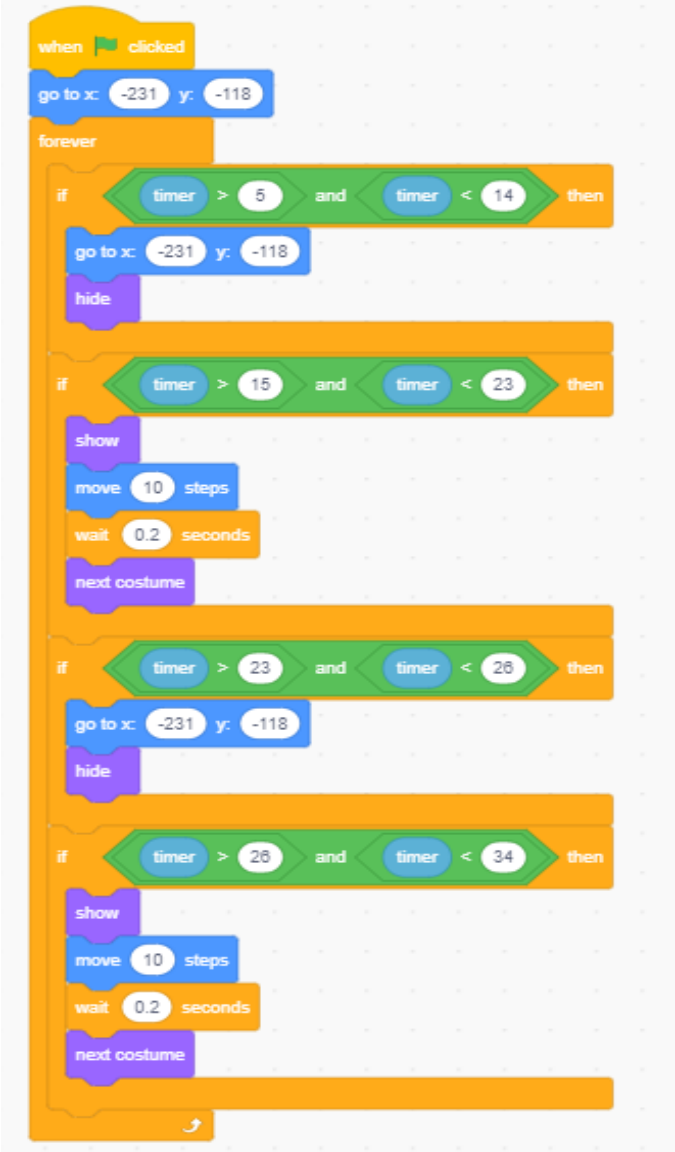
There is now needed an additional block which will reverse the direction of the walking sprite and end the programme:

Now we introduce a second sprite, namely a dog  that barks.

Place in a similar code to that used in the girl's script.

However we will have the dog appear after the girl calls for her pet to join her on the walk.
Hence, as the animal only shows a few seconds after the girl has begun her travels and in answer to the girl's call for him to join her whilst always walking a few steps behind his owner, we need to code into the script a command(s) that will hide the dog before it reaches the end of the screen (please note the backdrops change in response to the actual times when the girl reaches the end of the screen). The dog also re-appears in the next screen only after the girl has moved a few steps forward. Thus we use the Show and Hide option from the Looks folder.

As was done with the script for the girl sprite, once the end of the last screen is reached, input code that reverses the direction of the sprite to walk a few steps back. The coding of the timer parameters for the dog has to be synchronised with that of the girl script.



Finally, we should also input code to allow a bit of chat going between the girl and the dog.

Lesson 20 – Planning & Designing A Game[i]

Students need to be informed of the importance of first ***planning*** *out their proposed game with pencil/pen and paper rather than rushing in to happily write up scripts They should understand the need to define the purpose of the game, how this is to be achieved and the role of each of the game elements such as the different sprites and how they relate (effect) to each other.*
The written plan (or Algorithm) is then used to build the game.

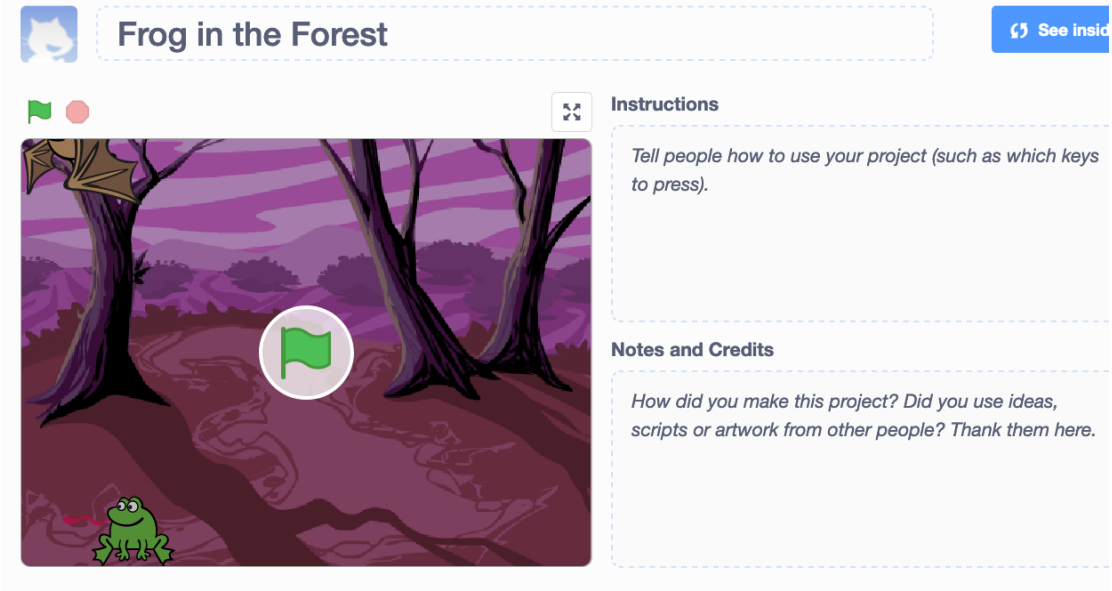
However the tutor should first provide an example of different types of games that could then be used as starting points or templates for the students' own creations.

Sample Game **Guideline** of *'Shark Attack!'*

- *How many characters (Two Sprites to start with)*
- *Role of characters (Little Fish sprite, Bad Shark sprite)*
- *Operations of characters (one sprite's movement is controlled by the four 'arrow keys'; the second character will 'follow' the first sprite using the 'point towards' block of code)*
- *What backdrop would be most suitable for the type of game that I am creating?*

Uploading a Game to the Scratch Website

Samples of projects from all students should be uploaded onto the school, class or group's online Scratch account that you helped them set up on the Scratch website with the very best of their projects stored also on your own Scratch project account. Hence it is important that a brief well-written explanatory note explaining the theme and the workings is provided for each uploaded project in order to benefit online users.
When your project is uploaded, go to the project page and complete the sections on the right of the screen labelled *Instructions*, *Notes & Credits* and *Add Project Tags*.



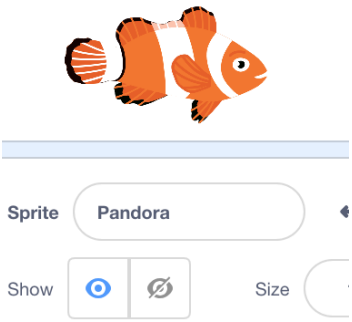
Lesson 21 – Game: Shark Attack!

This lesson introduces students to a) programming the ***arrow keys*** *to control the movement of sprites (a popular characteristic of many computer games) and the b)* ***Stop all*** *code.*

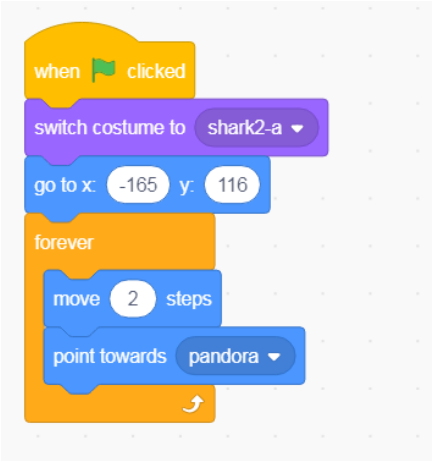
The lesson also re-introduces a powerful ***Touching*** *command and the* ***if*** *command (with its cause and effect impact).*

<u>Game Play - Coding Plan Summary</u> Shark chases a little fish, who desperately tries to escape its clutches. When the shark touches it, the little fish disappears as if eaten by the shark which changes to a new costume to simulate the swallowing process. The shark happily says "Yummy!" The game then ends.

Select from the Animal folder a) Shark sprite with two costume changes ('open' & 'shut' mouths) and b) a little fish sprite.
Give names to the shark and the little fish. In this example, I have called the former Kracken and the latter Pandora.

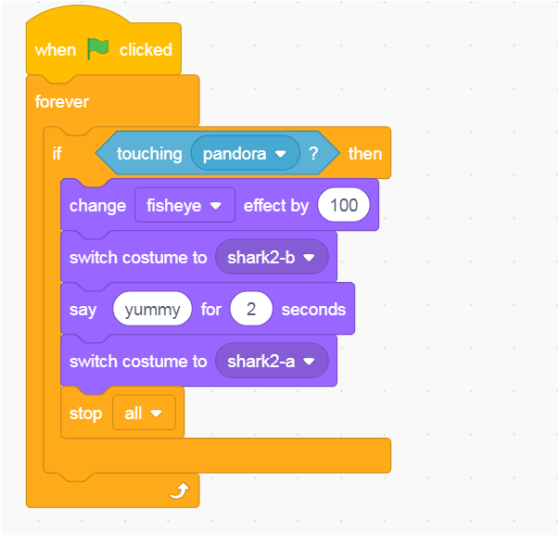


Type in the following two pieces of code in the Kraken (shark) script:



The *forever* loop block of code and the content contained within will mean that, no matter where Pandora is located, the shark *will always follow*.

The '*move 2 steps*' should be of sufficient speed for the initial version of this Shark Attack game as it will allow the small fish the opportunity to escape the clutches of the Kraken.

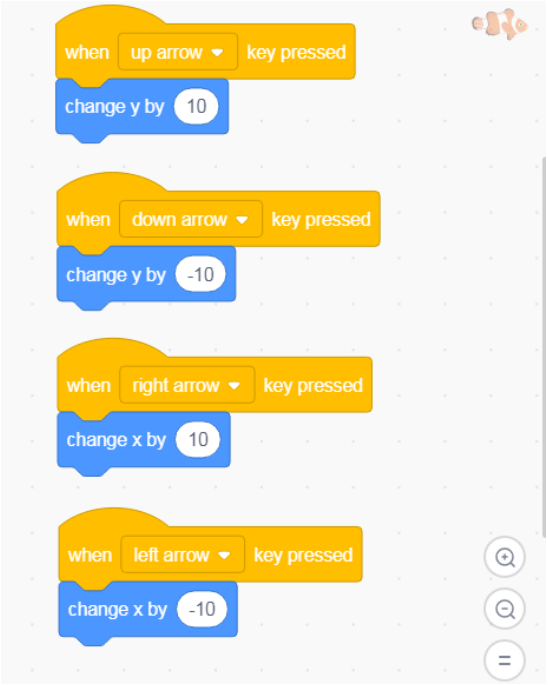


In the above piece of code, the touching option is found appropriately in the SENSING folder.

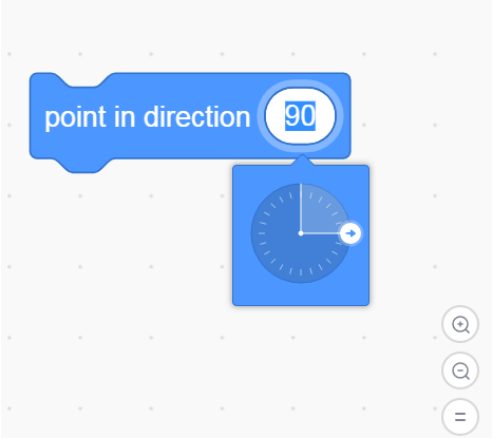
The additional pieces of code from the Look folder will give the impression of the Shark eating the little fish by alternating between costumes with the a) closed b) open mouth options (costumes).

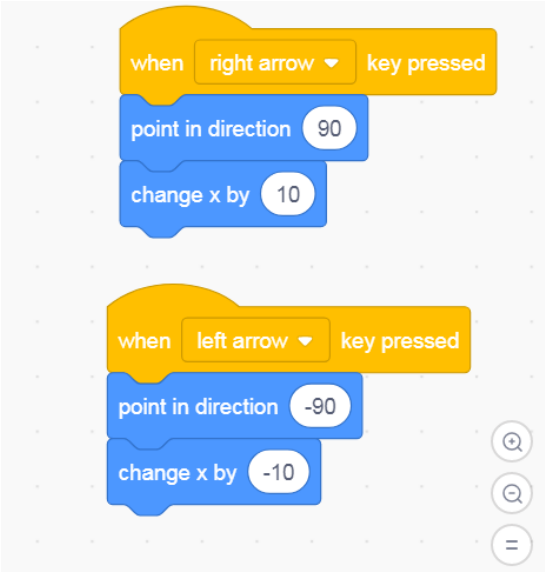
Pandora's (little fish) script

Type in the following two pieces of code in the Pandora (fish) script:

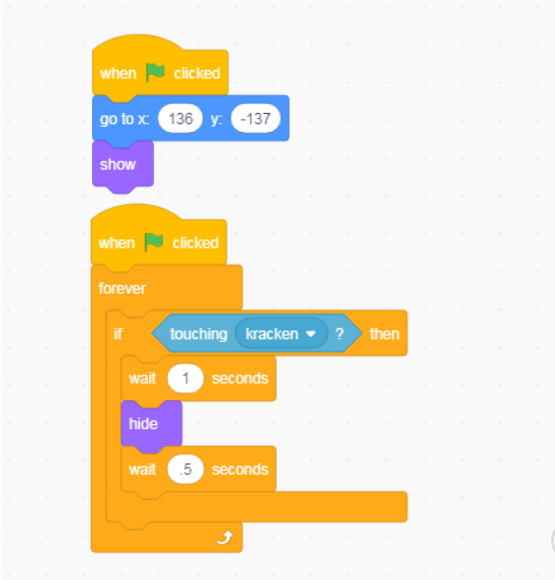


To ensure that the fish is always facing in the correct direction when it is being moved by the left and right arrow keys, the **point in direction** block from the Motion category needs to be added into the script as follows:



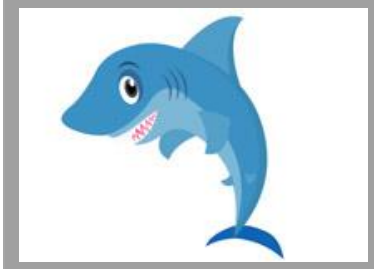


Finally, the inclusion of the *Show* and *Hide* blocks as well as two *Wait* options (shown below) will mean that Pandora disappears after it is 'eaten' by the Shark.



As you can see from the code above, the use of ***Stop All*** will bring the "Shark Attack!" game to an end.

Sharks under Threat



There is fossil evidence of sharks going back 400 million years. This is remarkable especially when one realises that the first dinosaurs only appeared on the planet 243 million years ago.

Today we know of more than 1,000 species of sharks with new species being discovered every year.

But the very existence of these remarkable fish is threatened due to the activities of humans. With 70 million killed every year, many species of sharks are now endangered and could become extinct in the near future.

Saving these creatures is key to restoring the life of the Earth's oceans.

Exercise

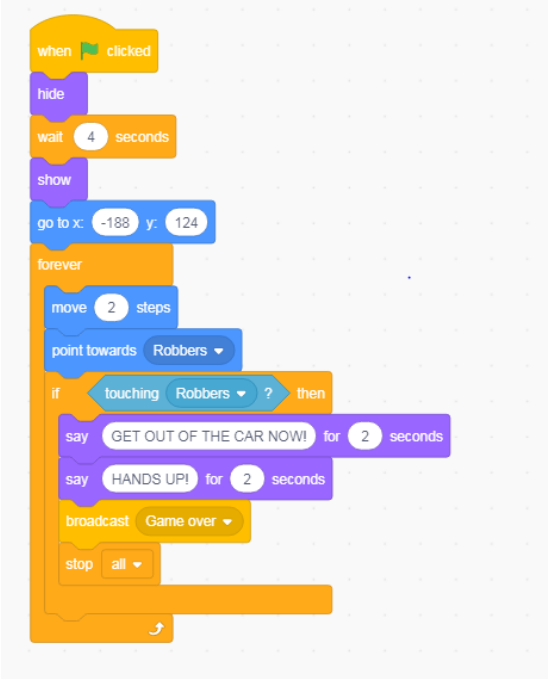
Car Chase

Get students to use the commands and the coding processes taught in the Shark Attack! Lesson to create a Car Chase project.

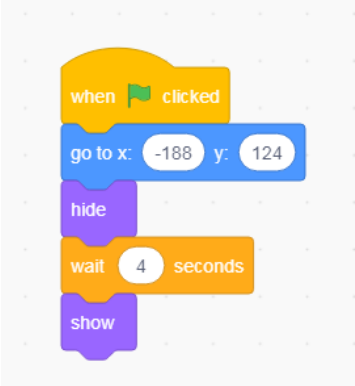
Code for a Car Chase Game

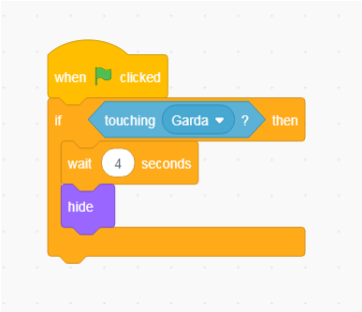
First have the students use the **Paint Editor** to draw a police car and a (thief's) car as well as a series of suitable backdrop road scenes.

Have the students use a similar script to that below for the Police car:



Get the students to use a similar script for the thief's car to below:



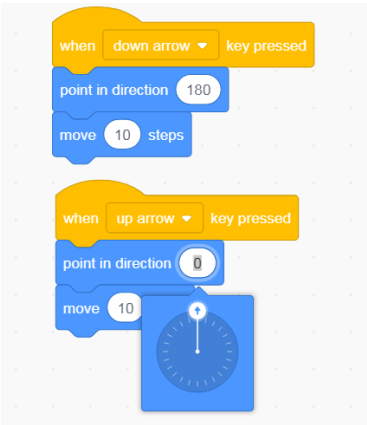


Note: Programming the Four Arrow Keys to control the direction as well as the movement of the car

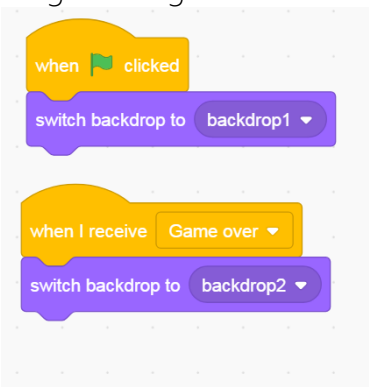
In Shark Attack! specific blocks of codes were used to have the little fish being chased *facing left* when moving in a *left direction* and *facing right* when moving in a *right direction*.

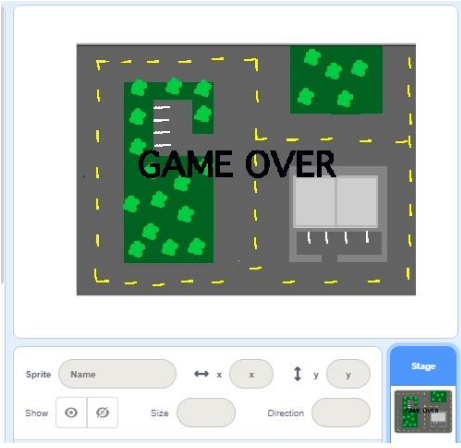
This time we also need to have the car being chased facing up (north) and down (south) when travelling in those directions.

See below:



Programming in different Backdrop Scenes





Lesson 22 - Shark Attack Advanced

This lesson increases the complexity of the 'Shark Attack!' game by introducing a ***competitive*** *element into the project in the form of a* ***timer*** *(Sensing command), the* ***random selection*** *(the concept of chance), the use of simple* ***variables*** *in the form of* ***'lives'*** *and the creation of different* ***levels****.*

Note: In Mathematics, a **variable** is a value that may change within the scope of a given problem or set of operations. Here in this instance, we use variables to represent the number of lives of one of the sprites, which changes as a result of interaction with another sprite.

Game Play - Coding Plan Summary

Shark chases a little fish. The latter has three lives and has to stay alive for 30 seconds when the game will automatically end. Each time the shark touches the fish, the latter loses one life.

After ten seconds, the game moves onto a different level which involves some changes to the backdrop, an increased speed for the pursuing shark and the appearance of a crab who, if he touches the fish, will lead to a reduction in lives for the latter.

If the fish is still alive after 30 seconds, he is declared the winner.

As with the previous lesson (Shark Attack!), we will use a Shark (Kraken), a small fish (Pandora) and an 'Underwater' backdrop from the Scratch *Outdoor* folder.

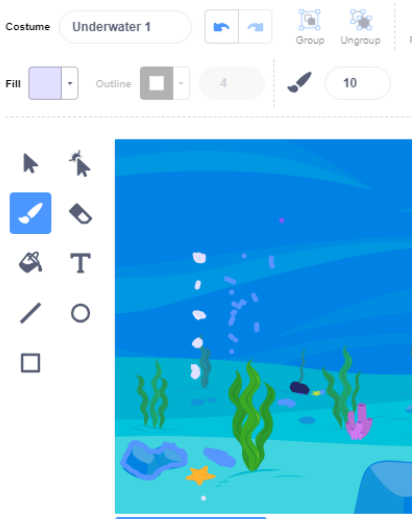
So first upload the previous *Shark Attack!* Project.

This time though, we will significantly amend the scripts of both the shark and the little fish; bring in different versions of the *Underwater* backdrop that will represent levels of increased difficulty and relay *You Win*! and *You Lose!* messages to signify the ending of the game.

Start

Click on the *Backdrop* icon, then duplicate the *Underwater* backdrop once.

Edit this new level (Level 2), by drawing in some additional features such as extra or larger blades/stems on the seaweed, coral and rocks as well as changing the backdrop colour to a deeper shade of blue.



When the editing is completed, duplicate Level 2 twice more.
With the second new level (Level 3), use the edit option to enter the *Paint Editor*. Then type in the relevant wording in large bold lettering such as *You Lose!* By using the (T)ext icon.
With the third new level (Level 4), type in the text *You Win!* in large bold lettering
To reposition the text to a suitable location, click on the T(ext) icon. Bring the cursor onto the little black rectangle that appears at the top left hand corner of the onscreen text (see below). A 'hand' icon then appears allowing the operator to drag the text to a new location.

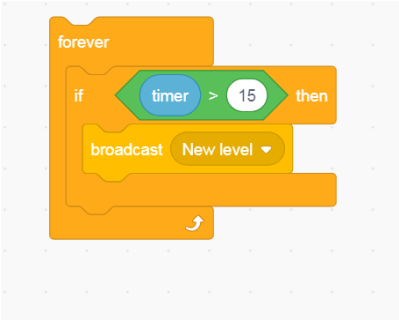
Coding the additional *Levels* into the script

To code in the extra three levels into the programme, we first move to the script for the little fish sprite.
Then place in a *Timer* which will allow us to change the backdrops based on a certain duration of time.

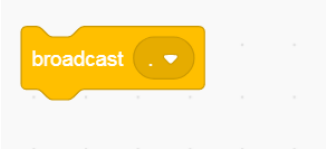
Go to *Control* folder and select:

Then go to the *Operators'* folder and the '*greater than*' block of code

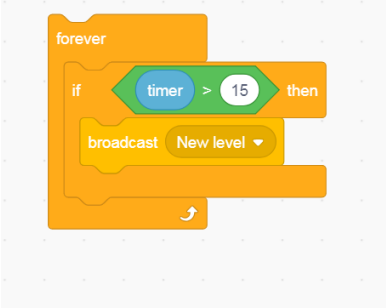
In the first blank white box in the above, place in TIMER from the *Sensing* folder.
In the second blank white box, input a number that will represent a time duration measured in seconds:



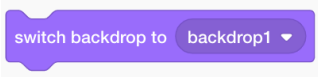
Within the *forever if* block above place, from the *Events* folder, the *broadcast* option.



To ensure that the *Backdrop* changes after 15 seconds has elapsed, select *new* from the broadcast block of code and type in the text "new level".

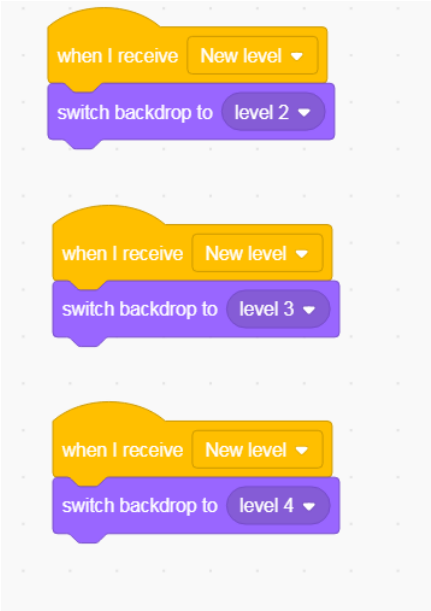


Broadcast is a very important piece of Scratch command code as it sends a message to some other part of the programme instructing it to implement a change.
But for it to work, it has to have a corresponding ***I receive*** command that in this case will be positioned in the **Backdrop** script.

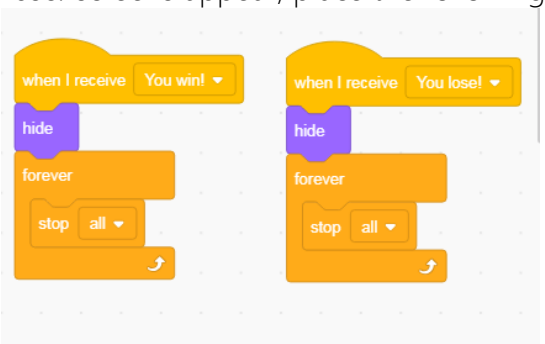
Note: There is an alternative and easier way to change the backdrops, namely the direct use of 

in the sprite rather than in the backdrop script. But familiarisation by students with the importance of the Broadcast commands is encouraged.

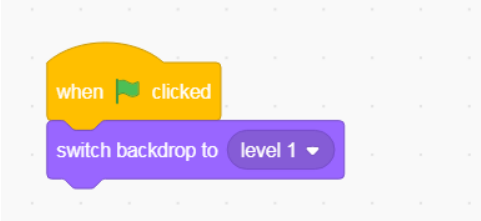
So go to the backdrop script. Input the ***When I*** **receive**_____ command followed by the ***switch to backdrop level*** ___ piece of code found in the *Looks* folder as shown below:



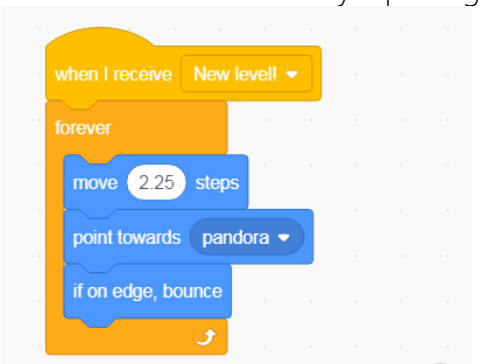
To ensure that that the game ends and all elements stop when the *You Win!* or *You Lose!* screens appear, place the following code in the Pandora (or Kraken) script:



The user also has to ensure that the correct screen appears every time that the game is played or restarted by inputting the following code into the *Backdrop* script:

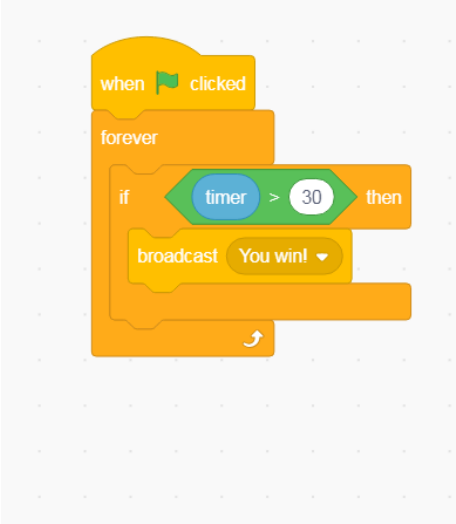


A characteristic of the new level will be the fact that the shark will increase in speed, this making the game more challenging for Pandora (user).
This feature is achieved by inputting the following code into the Shark's script:

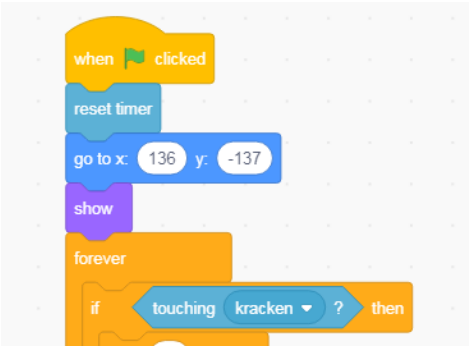


You can of course increase the speed even further by changing the digits in the *move__steps* block.

The little fish sprite has to stay alive for the specific period of time that the user has chosen. If he manages to stay alive for this inputted time duration, then he/she is declared the winner.
This is achieved by inputting the following piece of code into the little fish script that is based on a 30 second game duration:



The *Timer* has to be reset to 0 every time that the game restarts.
This is achieved by placing the *RESET TIMER* option into the little fish code as follows:

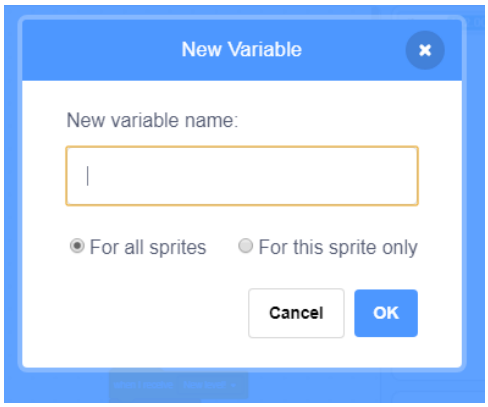


Click on the *Green Flag* to check out that the game is functioning okay.
Notice that a *Timer* indicator box appears on the top left hand side of screen:

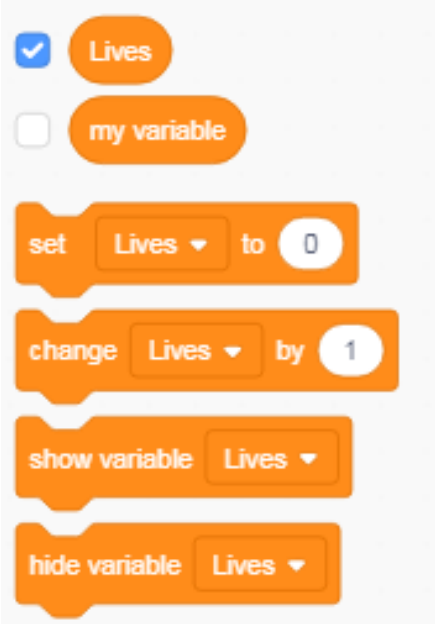
However for the timer to be shown on the stage, the little *box* to the left of the Timer code in the *Sensing* folder must contain a black marking.
If this is not the case, click on this box.

The Use of Variables

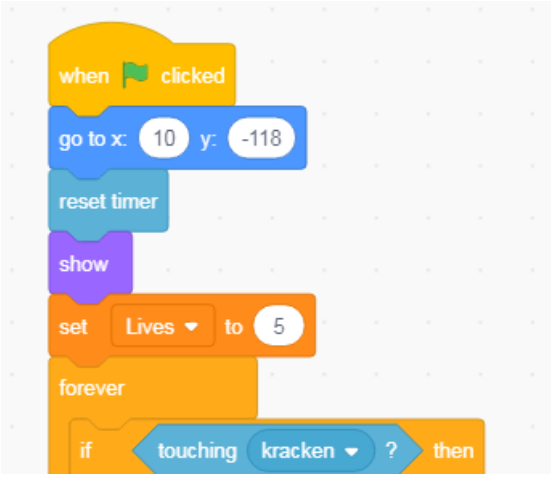
To programme in extra LIVES for the little fish sprite, go to the *Variable* folder and select the *Make a Variable* option.
You will be asked to type in a name.



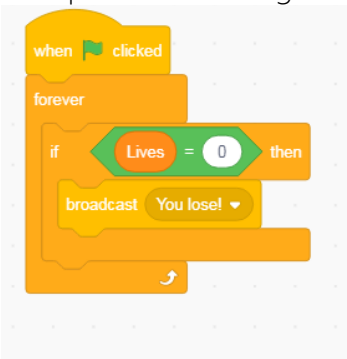
Type in the word *lives*
The **Lives** icon now appears in the *Variable* folder



Type in the number 5 in the **Set Lives to__** block before placing this code into the script of the little fish as follows:



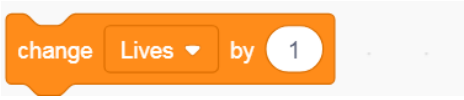
The little fish now has five lives.
If he/she loses all of his/her lives *before* reaching thirty seconds, then the game is over.
So input the following code:

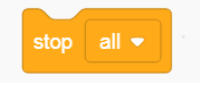


which is made up of blocks from the *Control*, *Variables* and *Operators* folders.

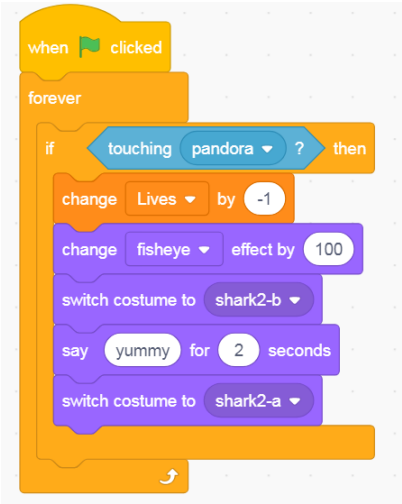
The little fish loses a life each time that it is touched by the shark.

This result is achieved by inserting within the *touching* block section of the shark's script.

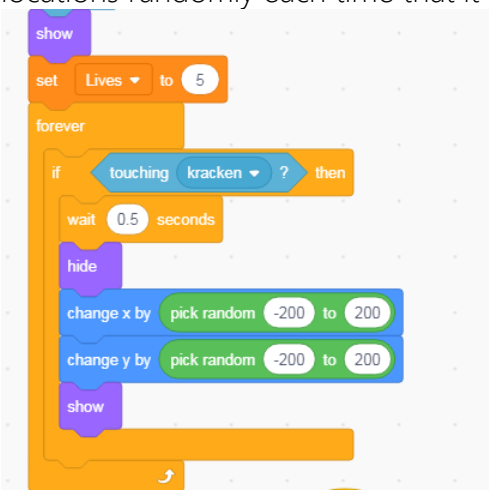


Furthermore, remove the 

block from the script and delete.
This is because, unlike in the previous *Shark Attack!*, this game does not end the first time that the little fish is touched by the shark due to the fact that the latter now has 5 lives.



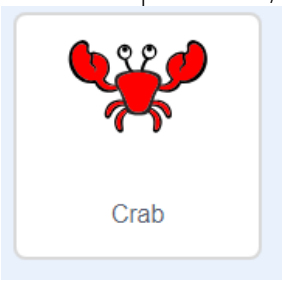
To give a fighting chance to the little fish in its attempts to escape the clutches of the shark, we need to add on a piece of code that will allow it to appear in different locations randomly each time that it is caught (touched) by the shark.

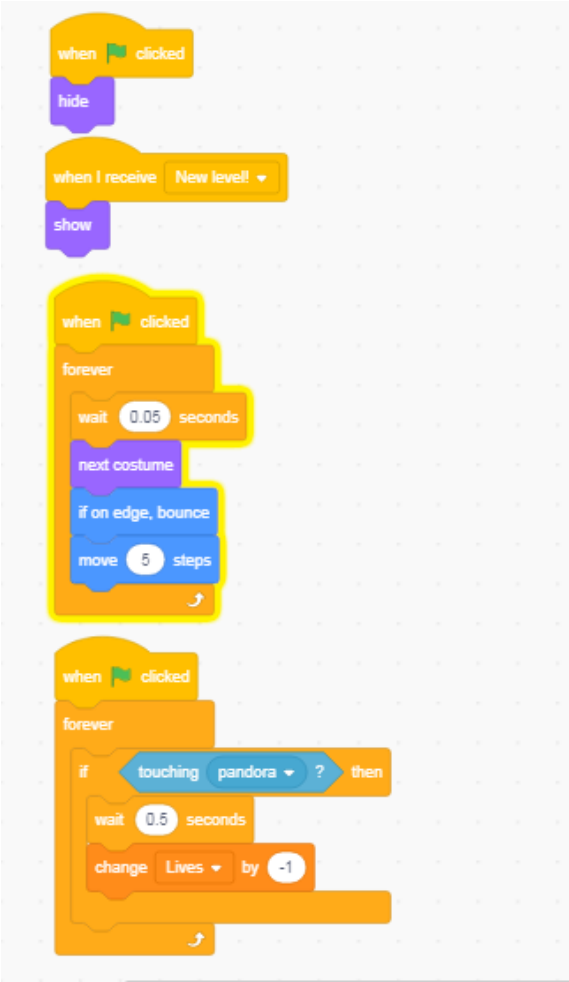


The wider the number range in the *random* (green) block, the wider the area that the little fish will reappear (i.e. *show*) after each time that he is touched by the Shark.

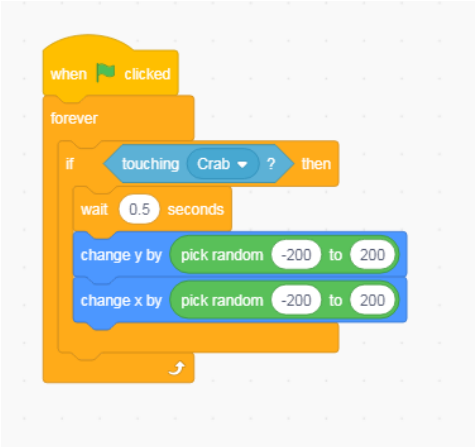
To add even more difficulty to the game, create another sprite that appears when the second Level is reached. This sprite will be coded so that if he touches the little fish, it too reduces the amount of Lives by one.

In the sample below, we use a *crab* sprite.





Return to the little fish (Pandora) script, where we add the following instructions that will lead to a loss of one life each time it is touched by the crab.



Exercise

Chasing Dinosaurs

Get the participants to create a game of one dinosaur chasing another dinosaur using *Lives* and *Levels*. It will involve one sprite moving from a forest, to a desert to a water, field or mountain scene. From the second scene, get them to introduce a second dinosaur (or another exotic creature) who will start chasing the main protagonist.

Lesson 23 – Tennis Solitaire

This lesson introduces a simple game modelled on one of the great classic computer games known as ***Pong****. It will introduce students to another* ***Sensing*** *command based on* ***touching colour****. They will also learn how to* ***change*** *the* ***direction*** *of a Sprite using a combination of* ***Motion*** *and* ***Operator*** *commands.*

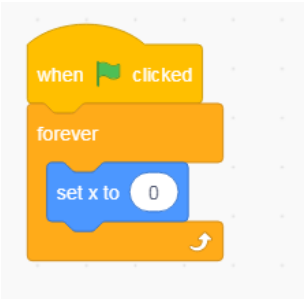
After opening up Scratch, delete the cat sprite.

The Paddle script

Go to the new sprite  icon and draw a black paddle using the *line* drawing tool and a suitable *Brush size*.

Rename the sprite '*paddle*'.

Go to the script for the paddle.
Input the following pieces of code:



The **Set x to 0** block is in the *Motion* folder.

Select 

from the Sensing folder and place it in the 0 of 

.

This ensures that the mouse pointer will always only move right or left across the screen along the X axis.

The Goals – *Setting the Stage*

To create a Goal-line, go to the *Stage* icon and choose the *Backdrops* option.
Then select the *Line* drawing tool, a large *Brush size* and the colour RED from the paint *palette*.
Bring the cursor to the bottom of the screen and draw a thick red line from right to left across its length.

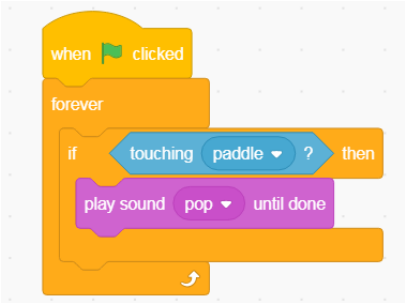
Position the paddle slightly above the RED goal-line.

The Ball script

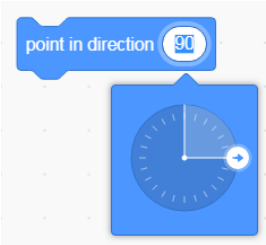
Go to the sprite column  and select a ball sprite from the gallery. Alternatively you can draw your own ball. If you do decide to do so, do not choose red as a colour as we will soon be using red as the colour of the goal line.
However it is recommended that students use an existing sprite from the gallery for this project.

We now input a motion script for the ball

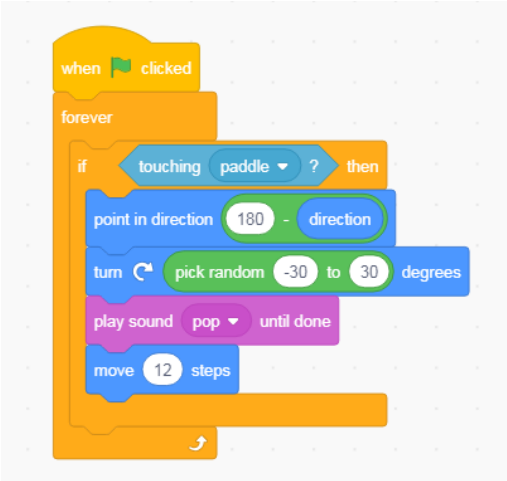
followed by a separate command block that will let the ball bounce in a general upwards direction with a nice sound effect if it touches the Paddle.
To achieve this, first input the script below:

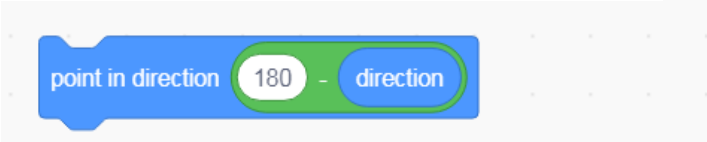


However, to ensure that the ball bounces *upwards off the paddle rather than downwards*, we need to use a ***point in direction*** block where downwards (i.e. 180) movement

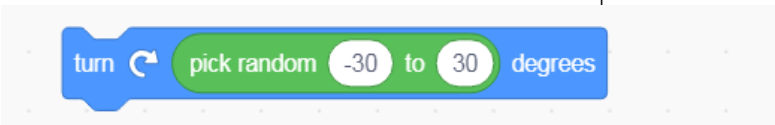


is negated by subtracting from its present trajectory (*direction*) using the appropriate code in the *Operators* folder.

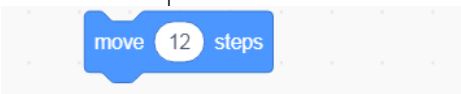




Variation in direction in it's upwards movement is achieved by the use of a *random number selection* which, by the parameters' range inputted, ensures a change in direction each time that the ball touches the paddle

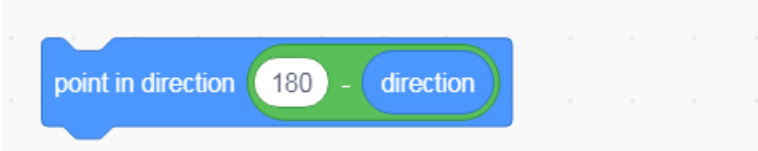


Its' upwards thrust is helped by *increasing the amount of steps* that it moves *after* it touches the paddle.



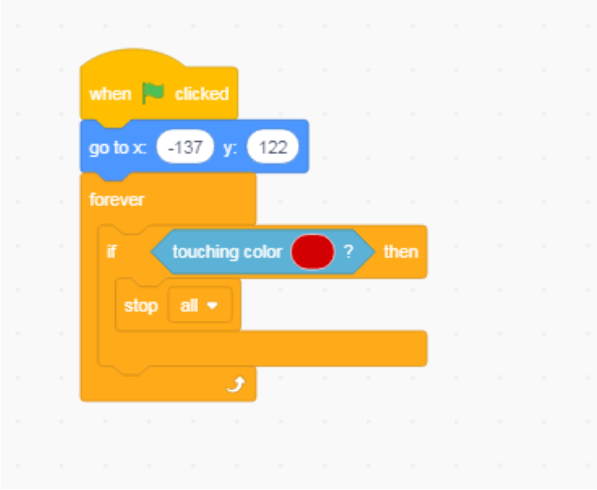
The final code sequence for this command should be as follows:

The arrangement of the coding blocks, as shown is the example above, is very important and critical to the correct playing of the game.
For instance if



appears *after* the move **12 steps** block, then the ball sprite will keep on moving downwards or sideways rather than upwards.

To ensure that the game correctly ends when the red line is touched, input the following:



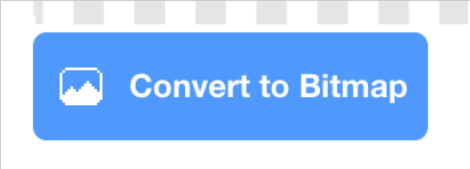
<u>NOTE</u>: For the above code block to work appropriately, the red colour chosen for the **touching colour** should have same colour value (***colour, saturation*** and ***brightness)*** as that of the red colour value for the red line.

Lesson 24 – Adventure Games: The Amazing Maze!

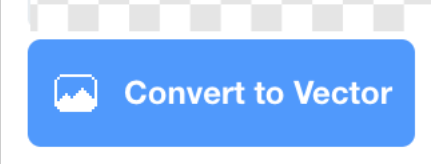
Adventure games are based on a main character undertaking a journey across many different lands where he/she encounters obstacles or dangers that have to be overcome or circumnavigated in order to reach the final destination and claim a treasure or reward.
The Maze is a popular variant of this type of game.
*This lesson reinforces many of the important commands learnt in the previous few chapters such as **broadcasting** and **touching colour**.*

Bitmaps and Vectors – A brief overview

Bitmaps and Vectors are different types of two-dimensional graphics which are used in the Paint Editor of Scratch.
What sets them apart is the way in which they store their information.



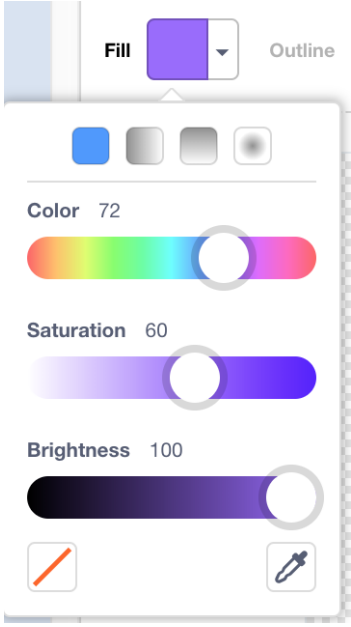
Bitmap images store colour information. It stores the colour of each and every individual pixel that makes it up.
Bitmap images store the colour information of every individual pixel that makes up the above image.



Vector images store mathematical formulae that draw lines and curves.

Selecting Colours in the Paint Editor

It is critical that, when choosing colours in the Paint Editor, users write down the *numbers* that appear at *Colour*, *Saturation* and *Brightness* (see graphic below) Otherwise problems will arise when selecting colours in the Sensing blocks for building a script.



Creating Levels

First select the Stage icon, click on Edit and colour the backdrop black.
Copy this black backdrop three more times.

In one backdrop, use the TEXT tool to type in You Win!
In a second backdrop, use the TEXT tool to type in You Lose

Go to one of the remaining backdrops (Basic Level)
Using the Rectangle tool, the colour white and the outline icons, draw a number of different boxes of different positions spread across the stage. In the top far corner, place a rectangle box coloured yellow.

Go to the remaining backdrop and rename it Advanced Level
Colour in the main area Green.
Using the Rectangle tool, the colour white and the solid colour icon, draw two cloud and two castles type shapes.

The Sprites

First select a main character sprite whose movement will be controlled by the four arrow keys.

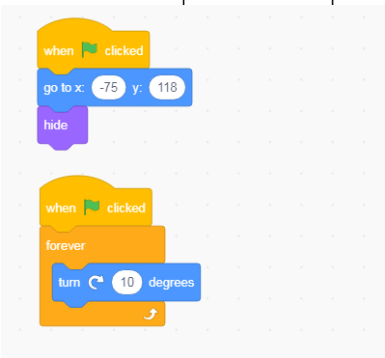
Go to new sprite  icon. Draw a coloured ball using the *circle* drawing tool.
Using the paint brush icon, draw a number of uneven lines emanating from the circle area.

Name this sprite *Hero*.

Once again, use the  to draw another coloured ball (RED) using the *circle* drawing tool. Then with the paint brush draw a number of scraggly BLUE lines emanating from the circle area.

Alternatively select appropriate sprites from the gallery. <u>This might be easier and less problematic f</u>or you as a mentor as it might avoid individual students forgetting the specific numbers of their colour selection (Colour, Saturation and Brightness), for the sprites that they created.

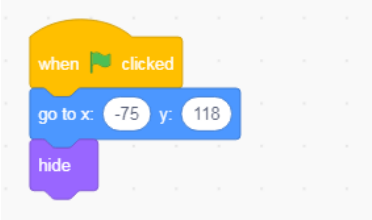
Go to the script for this sprite and input the following:



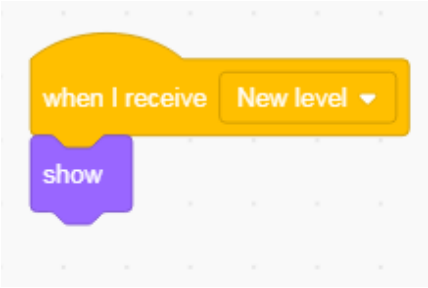
This sprite will represent an obstacle for the HERO sprite that if he touches will led to him losing the game.

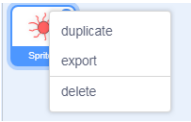
Hence to increase its difficulty for Hero, we will have the second sprite constantly rotating which is achieved by the use of a *Turn Degree* block in the *Motion* folder in combination with a *Forever* block

Furthermore, this sprite will *only* appear in the Advanced Level. Hence the use of HIDE in the opening code

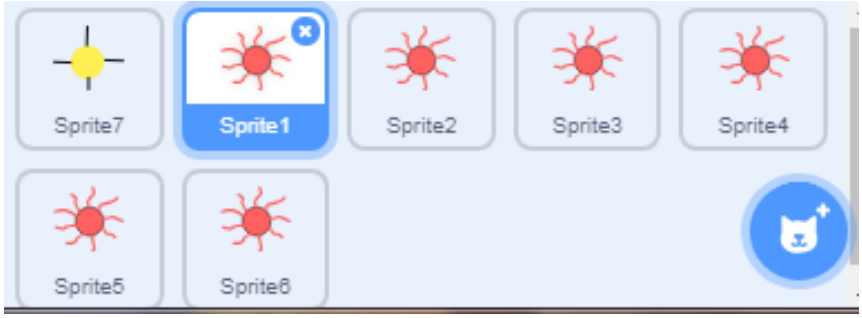


and by inputting the following block:

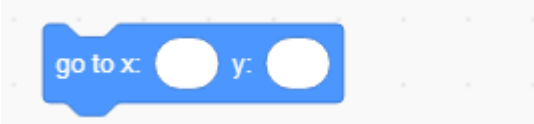


Duplicate 

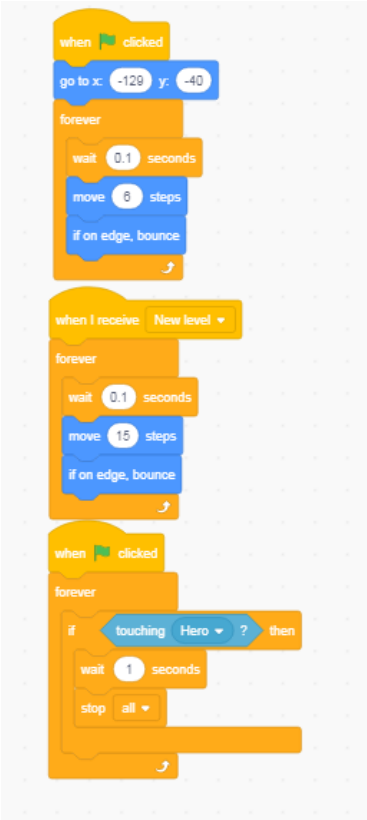
, this sprite six times.



Of course the position of each of these duplicated sprites has to be different and should be located to give maximum obstacle impact to the Hero sprite.
So change the entry in each of the individual blocks of code

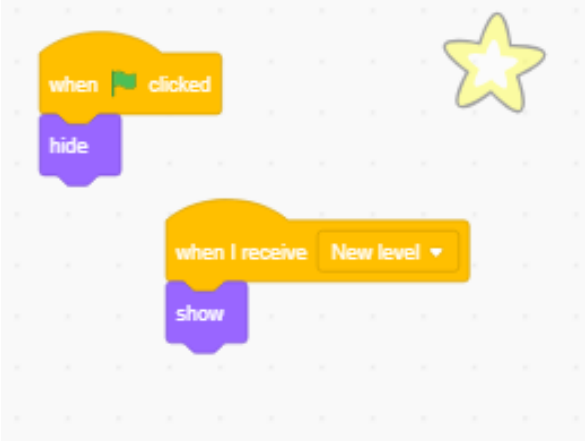


We will now bring in another obstacle sprite  that will be programmed to move across the screen, to increase in speed when the Advanced Level is reached and when touched by Hero will led to the game **ending** and the message **You Lost** appearing on a black screen.
This is achieved by the use of the following three pieces codes to form its script:

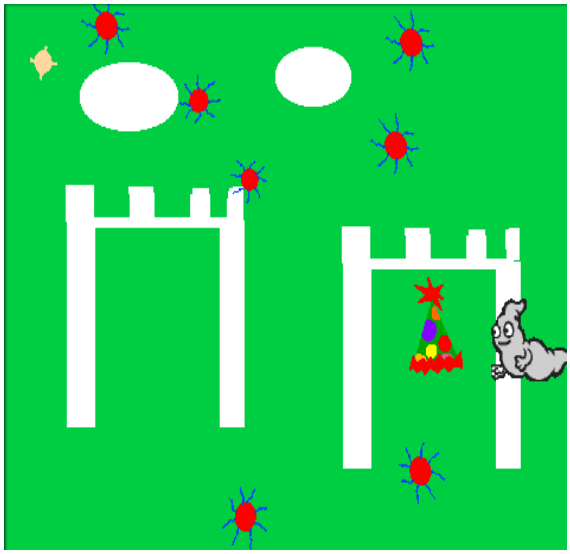


To increase the level of difficulty for the Hero sprite, the Ghost sprite can be replicated a few times, with the X and Y coordinates for obvious reasons being different for each version.

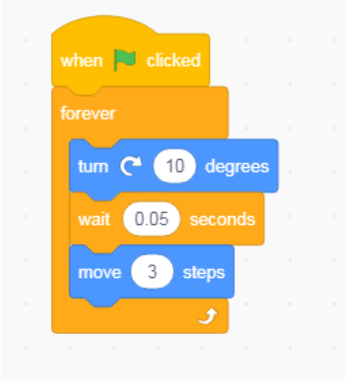
The final sprite addition will serve as a PRIZE that if reached by the Hero sprite will lead to the message You Win! screen appearing and the game ending.



The final positioning of the sprites in the Advanced Level should be similar to:

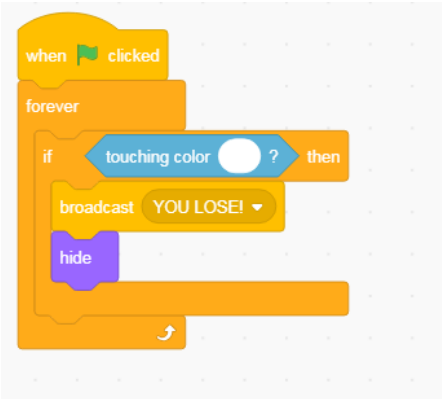


The intensity of the difficulty for the Hero in securing the PRIZE can be increased for instance by adding two blocks of code to the script of the red circle sprite positioned in front of the PRIZE that will allow it to move in a circular motion.

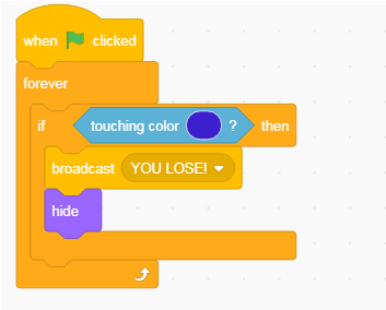


The script for the Hero should include:

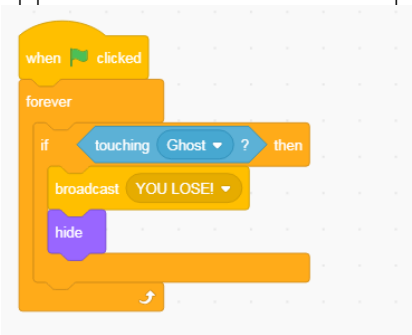
a set of instructions that if it touches the *white* maze walls, the Hero will disappear.



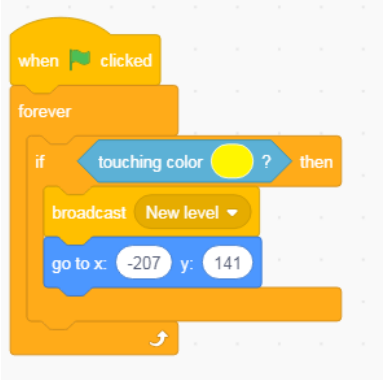
a set of instructions that if it touches *blue* spokes of the *mines*, the message 'You Lose' will appear and the Hero will disappear.



a set of instructions that if it touches the *Ghost* sprite , the message 'You Lose' will appear and the Hero will disappear.

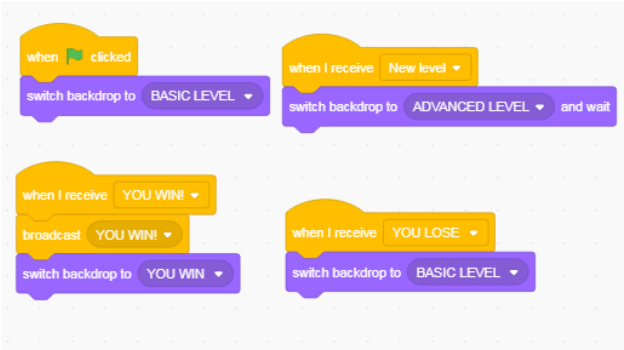


a set of instructions that if it touches the *yellow* box, the game will move onto the next level.



a set of instructions that if it touches the prize in the second level, the message 'You Win' will appear and the game will end.

Finally, the code for the Stage has to be written to respond (via the inclusion of **When I receive block**) to the **broadcasts** contained in the HERO sprite:

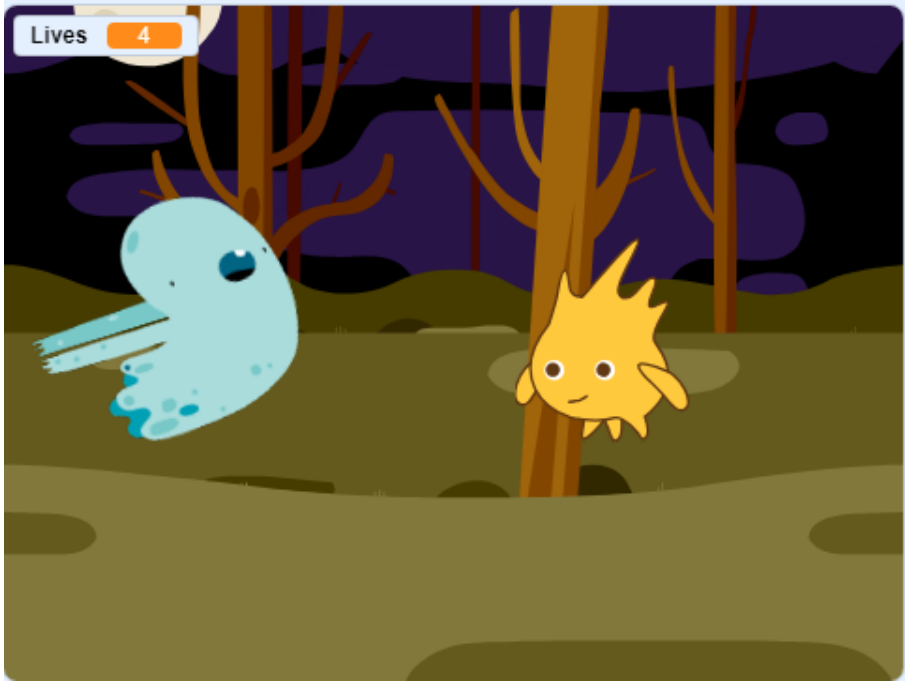


Exercise:

Say Cheese! Get the students to construct a Maze project whereby a mouse has to avoid being entrapped by mousetraps and caught by cats in order to reach the finishing line to claim a lovely big slice of cheese.

Lesson 25 – Demon Chaser

Demon Chaser



Game Play - Coding Plan Summary

- 'Good Sprite' moves by arrow keys but has to keep moving
- Demon moves randomly so that the Good Sprite does not know where he will appear
- The Good Sprite Starts with 5 lives
- If the Demon touches the Good Sprite, the Good Sprite loses a life
- If the Good Sprite's lives become equal to zero, then the game stops and a 'Game Over' backdrop appears
- When the Game restarts the backdrop is reset and the lives are initialised back to 5.

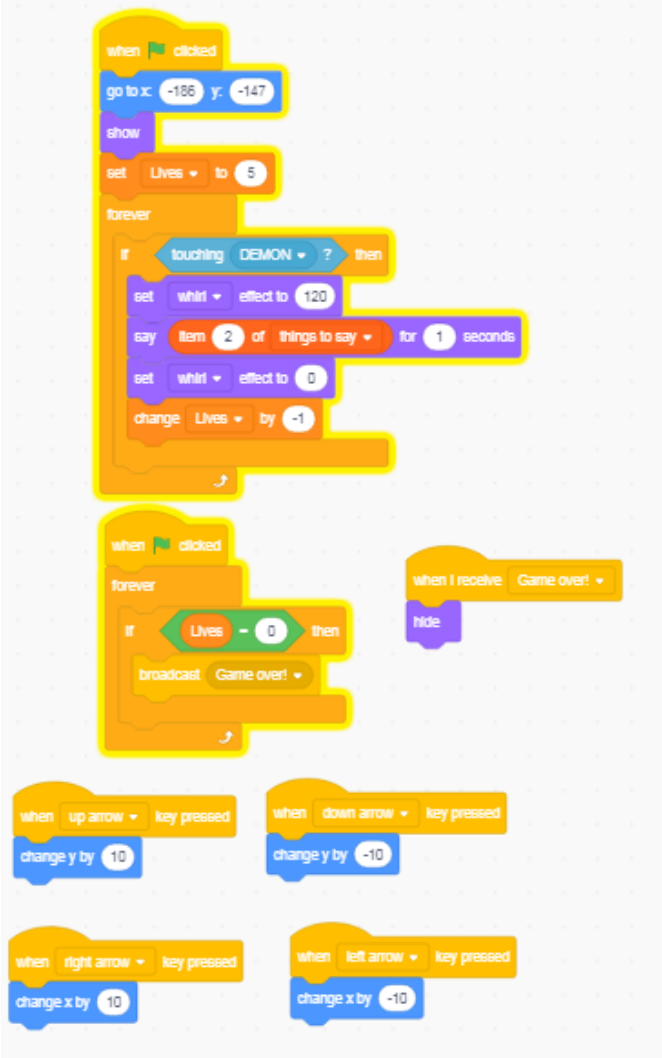
Key Command Blocks: *'Forever if_____', 'When I Receive' & 'Broadcast' (Control folder), 'Say' (Looks), 'Touching (a sprite)' (Sensing), 'Pick random ___ to ___' Make a Variable, 'Make a List' (Data), '___or ___' (Operators), 'Turn ___ degrees', If on edge, bounce' (Motion), 'Play sound ____' (Sounds).*

Make a List option in the *Data* folder allows the developer to compile for instance a list of comments or words that a sprite would say when something specific happens such as being touched by another sprite during a game.

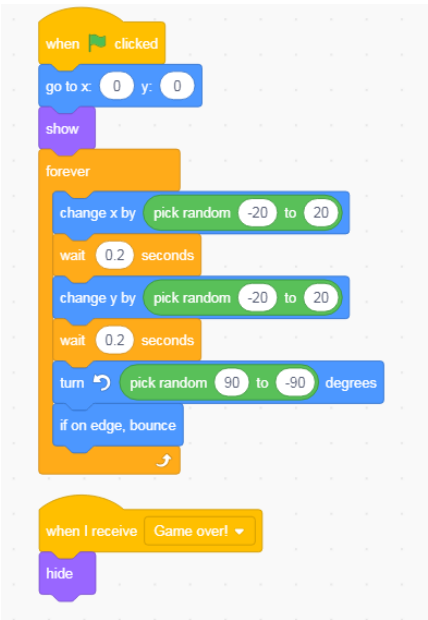
The game requires two sprites and two versions of a backdrop. The latter would have one screen displaying the text Game Over.

Improvements: *After completion of the above guidelines, get the students to make some adjustments that they feel might improve the game*

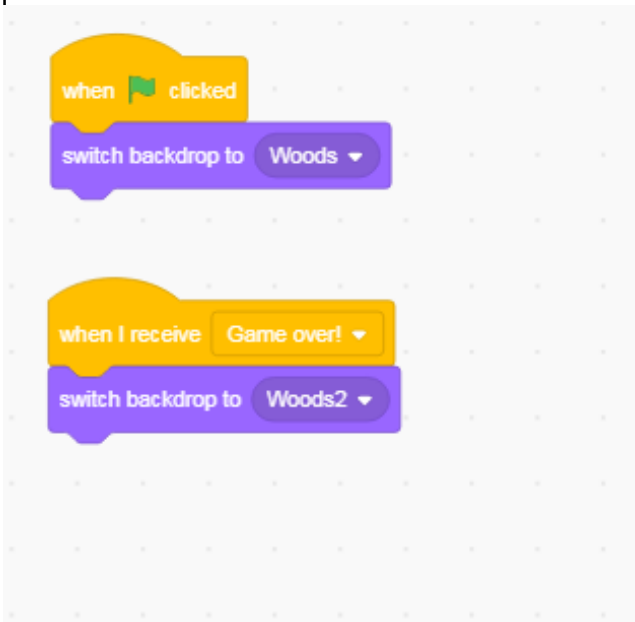
Code (Instructions) For Gobo (the 'Good Sprite')

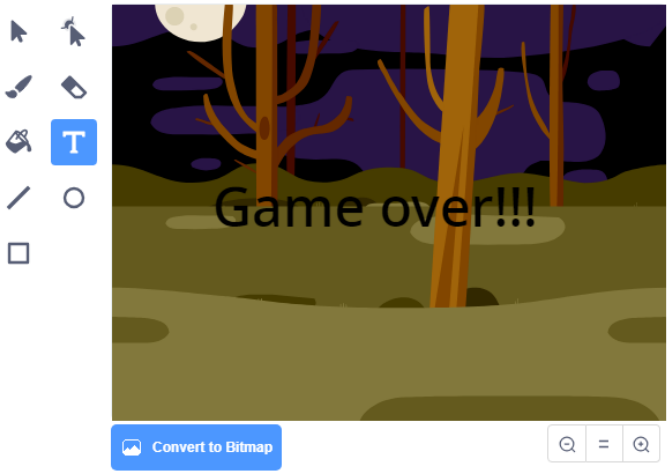


Code for Demon



Code for Backdrop





Exercise

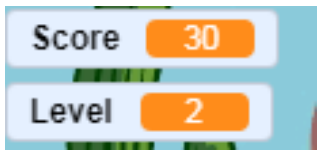
Get the students to create their own game, based on what they have already learnt from this and other lessons, for the next class using sprites drawn by themselves. Encourage each individual or group to explain and demonstrate their own project creations to the full class.

Lesson 26 - Extending the Demon Chaser Game

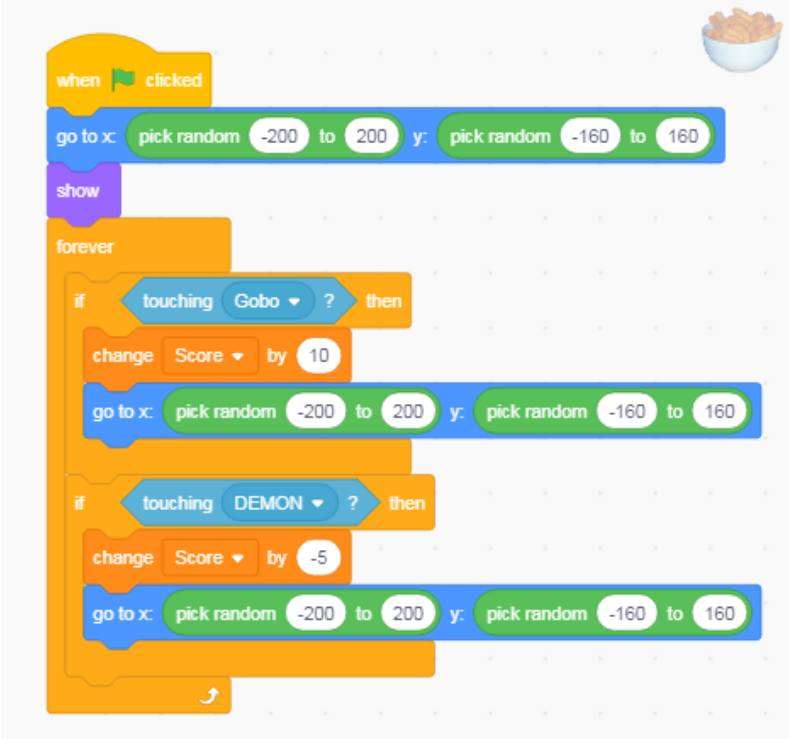
Game Play – Coding Plan Summary
In this lesson, students will extend the Demon Chaser game by adding treasure items for the hero sprite (Gobo) to collect, maintaining a score and changing levels when the score reaches certain values. A player will lose the game if the number of lives reaches zero and will win the game if level 3 is reached.

First we add a new sprite for the treasure (in this case a bowl of cheese puffs).

Then we add two new variables, one called "Score" and one called "Level"

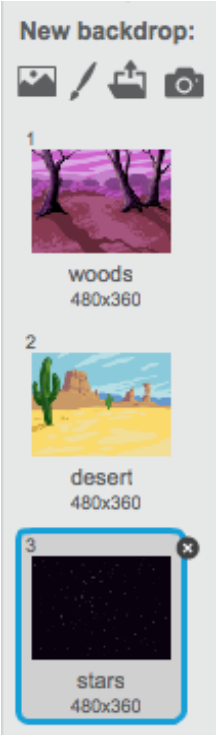


We add the following script to the "treasure" sprite:

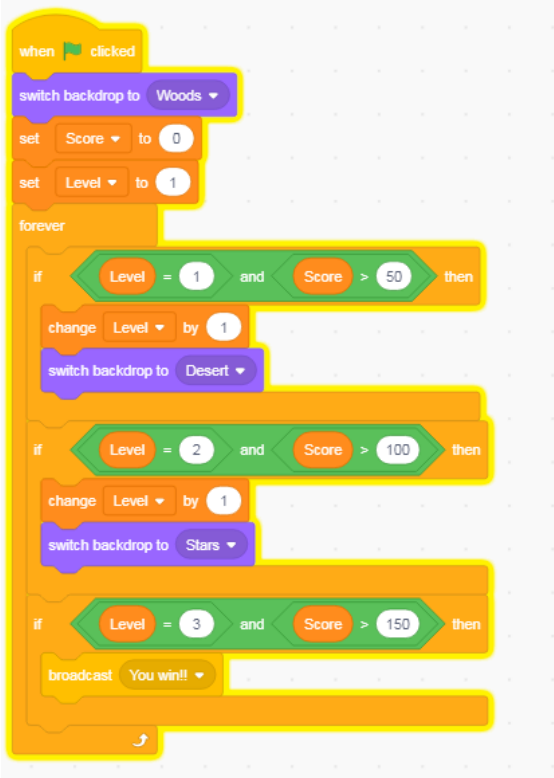


This script moves the treasure sprite to a random part of the screen when the game starts. Each time Gobo touches the treasure, the score is increased by 10 points and the treasure is moved to a new random position. If the Demon touches the treasure then the score is reduced by 5 points and the treasure moves to a new random position.

Now we remove the original "woods1" backdrop that had the "Game Over" message. We add 2 new backdrops to the stage. When the level changes, we will change the backdrop.

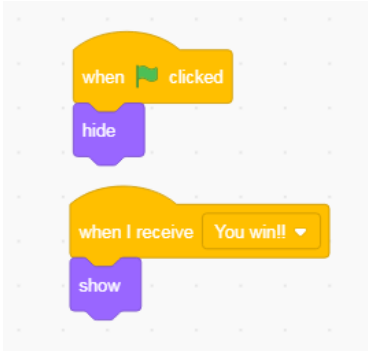


We add the following script to the stage to control the levels:

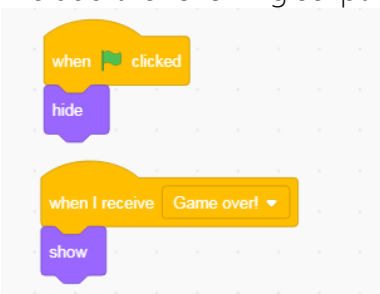


When the green flag is clicked, we set the score to 0, the level to 1 and the backdrop to "woods" (the first backdrop). We add a *forever loop* to keep checking the score and to increase the level if necessary. If we are on level 1 and the score is greater than 50, then we increase the level by 1 and we switch to the next backdrop. Similarly, if the level is 2 and the score is greater than 100, we go to level 3 and change to backdrop 3. If the level is 3 and the score is greater than 150 then the player wins and we broadcast a "You Win" message.

Now we add 2 new sprites for indicating whether a player has won or lost the game. Using sprites for this allows us to simply show a "win" or "lose" message over the current backdrop:
We add the following script to the "You Win" sprite:

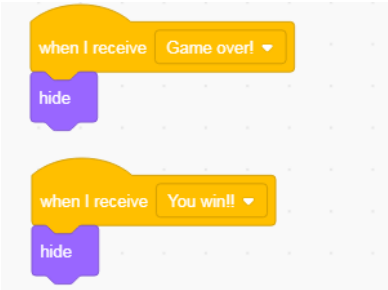


When the game starts, we hide the "You Win" sprite. If we receive the "You Win" message broadcast by the stage, then we show it.
We add the following script to the "You Lose" sprite:

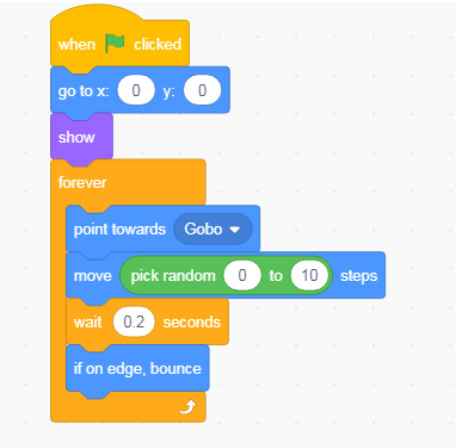


When the game starts, we hide the "You Lose" sprite. If we receive the "Game Over" message broadcast by the Good Sprite, then we show it.

We add the following scripts to the "Good Guy", "Bad Guy" and "treasure" sprites, to hide them when the game is over:



To make the game a bit more challenging, we can make the Bad Guy actively follow the Good Guy instead of just moving about randomly. We can also make him move more quickly as the levels increase. To do this we change the script for the Bad Guy to the following:



In this script, we continuously point the Bad Guy towards the Good Guy and move him a random number of steps towards the Good Guy. The maximum number of steps the Bad Guy can move is 10 times the current level. This means that he tends to speed up as the levels get higher, making the game more difficult on each new level.

Exercise

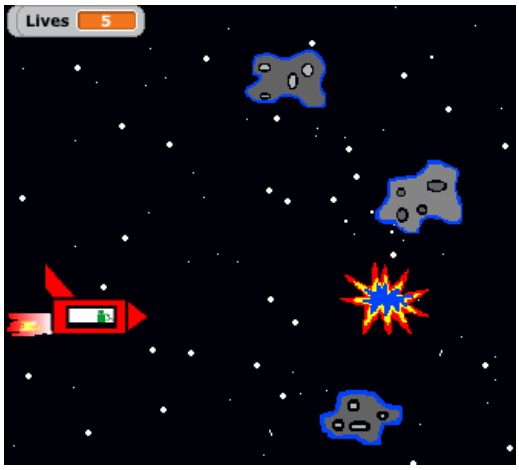
Get the students to create an Adventure Game based on a legendary mythological hero from their own county/culture having to fight off demons and monsters in the mountains, in the seas and in the caves in order to gain a treasure.

Lesson 27 – Shooter Games: Asteroids

Shooter games are based on an environment whereby a main character secures points for touching another character or set of characters. This genre is often quite challenging and can test a player's speed and reaction time. Oftentimes, the player-character, if he/she survives obstacles, doesn't lose lives and claims a certain score, can then moves forward onto another level or mission.

Asteroids

In this simplified version of the classic game Asteroids, students will be introduced for the first time on how to create a sprite that imitates the rapid action motions of a laser (or bullet).



Game Play - Coding Plan Summary Spaceship enters an asteroid field. The pilot has to blast the oncoming astral rocks order to save the ship. The ship can only survive three direct hits before it is destroyed But for every direct hit, the pilot gains a point.

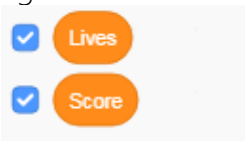
First, select the *Stars* backdrop from the *Nature* folder.

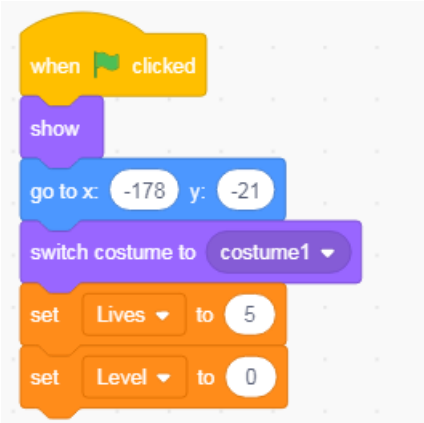
Spaceship sprite

Draw (or select from the sprite gallery) a spaceship with three costumes, one of which signifies an explosion.

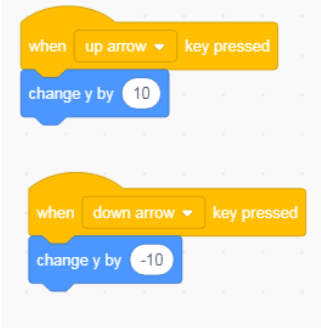
The first two costumes when coded in will give the impression that the spaceship is moving rapidly through space due to the small difference in the size and the shape of the spaceship particularly the width or length of the rocket booster flames.

Set up *Lives* using the *My Variable* option in the *Variables* folder.
Select a maximum of 5 lives.
Create a *Score* monitor using the same procedure using zero (0) as the starting figure.

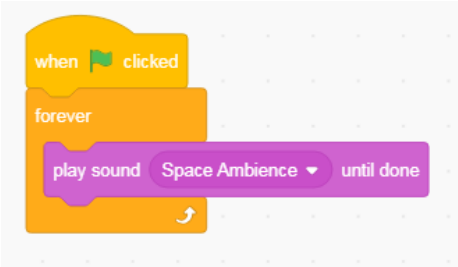




Control the *movement* of the spaceship using the *Up* and *Down Arrow* keys only.



Place in a suitable Sound effect code

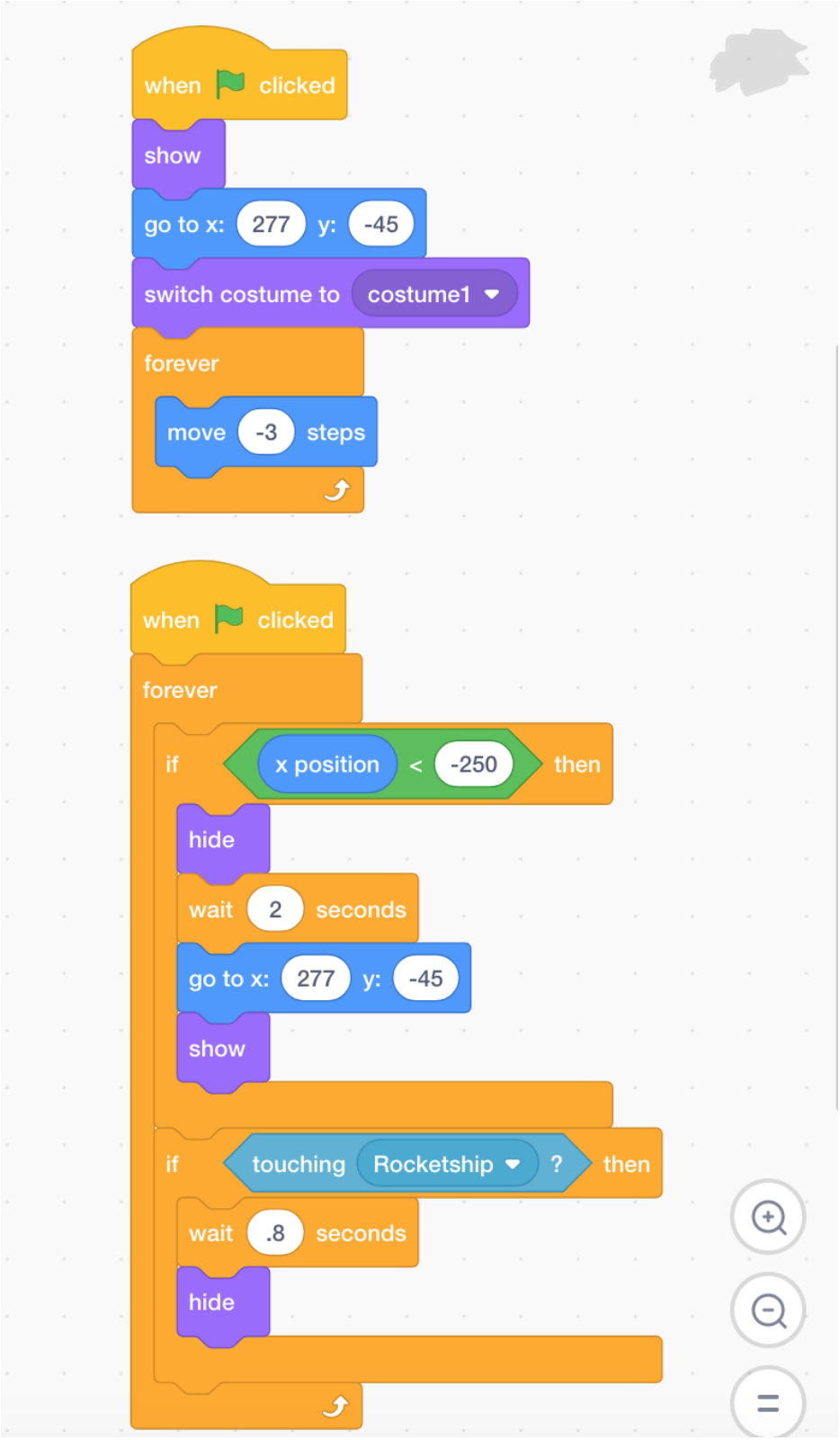


Draw an **asteroid** sprite with two costumes.

In the first costume, the asteroid should be grey.
Select the following numbers when drawing:
Colour: 25; Saturation: 0; Brightness: 65.

The second costume should represent an explosion.
Now replicate the asteroid at least four more times.
Each asteroid script should have a different X and Y setting to ensure that the sprites do not start from the same location nor have the same flight path.

Furthermore it would be atmospheric if these moving space rocks disappear just prior to reaching the left edge of the screen and to reappear a second or two later moving out of the right side.
This is done by using the following code (with each rock sprite having different X and Y coordinates):



Now we return to the Spaceship script as we have to programme in commands that will signify the impact of a collusion with an asteroid.
As there are multiple asteroids, it would take a lot of code to recognise each of the individual sprites. However we collectively identify all the asteroids by using the identifying colour grey.
So the main coding script for the spaceship should now read as follows:

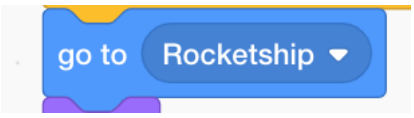
```
when [flag] clicked
switch costume to rocketship-a
show
go to x: -173 y: -22
set Lives to 3
set Score to 0
forever
  if <touching color [ ] ?> then
    switch costume to rocketship-d
    wait 1 seconds
    hide
    wait 1 seconds
    switch costume to rocketship-a
    show
    wait 1 seconds
    change Lives by -1
  else
    show
    switch costume to rocketship-a
    wait .8 seconds
    switch costume to rocketship-c
```

Finally, input code that will stop the game once all the spaceship's lives are lost:

```
when [flag] clicked
forever
  if <Lives = 0> then
    stop all
```

The Laser Sprite

Draw a new sprite that consists of a short relatively thick line.
Copy its costume six or seven times.
Give each costume a different colour.
This will give the effect of *firing* when coded in.

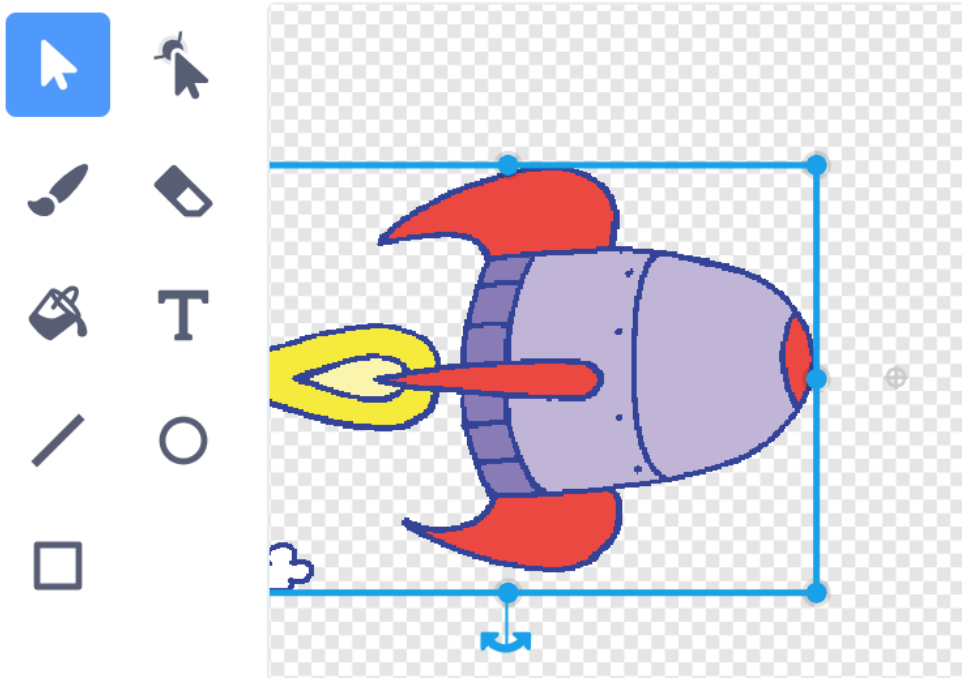
Align the Laser sprite with the Spaceship by using the 

in the Motion category.

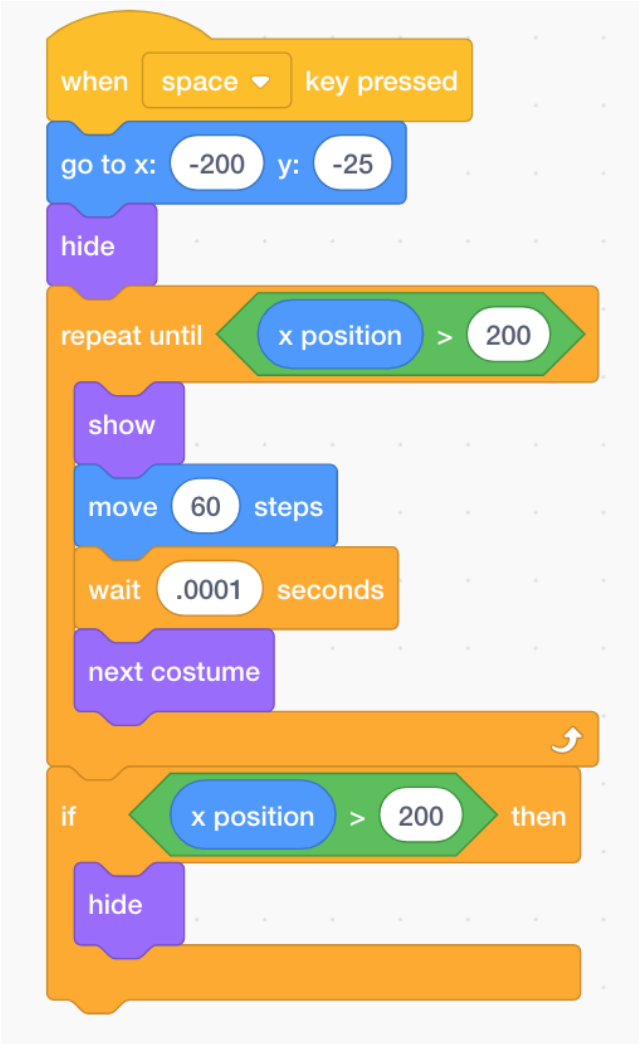
However you need to go into the costumes' *Paint Editor* of the Rocketship sprite to get a better alignment between both sprites.

After doing so, move the Rocketship sprite out of the middle of the screen in the *Paint Editor* using the *Arrow* positioning icon located at the Top left hand corner of the tools menu within *Vector* mode.
However if the sprite comprises multiple parts that may move independently of each other using this tool, go to the tool  located at the top right of the menu which allow it to be manipulated as one entity.

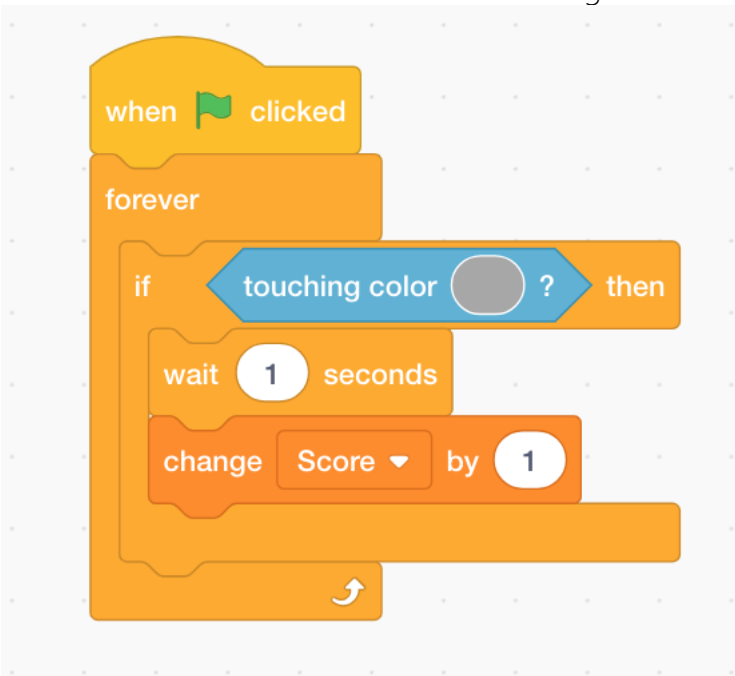
You will see a very small ***target*** marking. Move the spaceship sprite to the left and in front of the target.



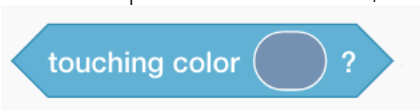
Run the full script project and see if it works.
You may though have to experiment a few times with moving the rocketship sprite in the Paint Editor until you feel the alignment between both sprites is satisfactory.
Now, we build a code that will give the visual impression that, when the spacebar is touched, a laser is shot from the spaceship in a straight firing line towards the direction of the asteroid.



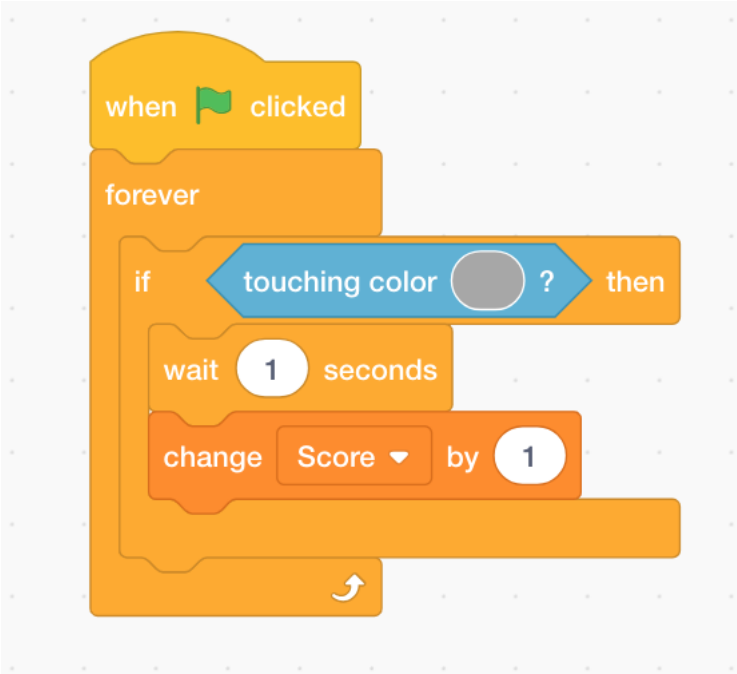
The code below includes commands that registers the laser hits of an asteroid:



However please ensure that, in the



block, you input the same numbers in the colouring box as you did when colouring the asteroid. Otherwise no score will be registered when you touch an asteroid. These numbers are:
Colour: 25; Saturation: 0; Brightness: 65.



Lesson 28 – Two Player Games

Teamwork is a core element in the teaching of Scratch. Hence regular project works involving two or more students are recommended.
Likewise, participants are encouraged to produce games that involve multiple players as today's modern computer gaming tends to be played in an interactive online social networking environment.

Soccer Striker

Shooting type games are a popular genre.

<u>Game Play - Coding Plan Summary</u>
One player, the Striker, tries to score 5 shots. The other player, the Goalkeeper, tries to save 5 shots. Whoever reaches their target first wins the game.
The goalkeeper moves by use of the W and S keys. The striker and the ball move along the Y axis by use of the Up and Down arrow keys. To shoot, the striker clicks on the spacebar. The positioning of these four motion keys facilities two players using the keyboard at the same time.
The programme resets immediately after shots are on target, off target or when shots are saved. This continues until either of the two players reaches five on the save/score display monitor.
This game also contains our first encounter with a Question & Answer sequence.
It was originally developed by eight-year-old Philip.

Key Command Blocks: 'Ask a Question' & 'Answer' sensing commands, 'Touching colour', the 'If' command, Variables & Operators, 'When I Receive' & 'Broadcast'.

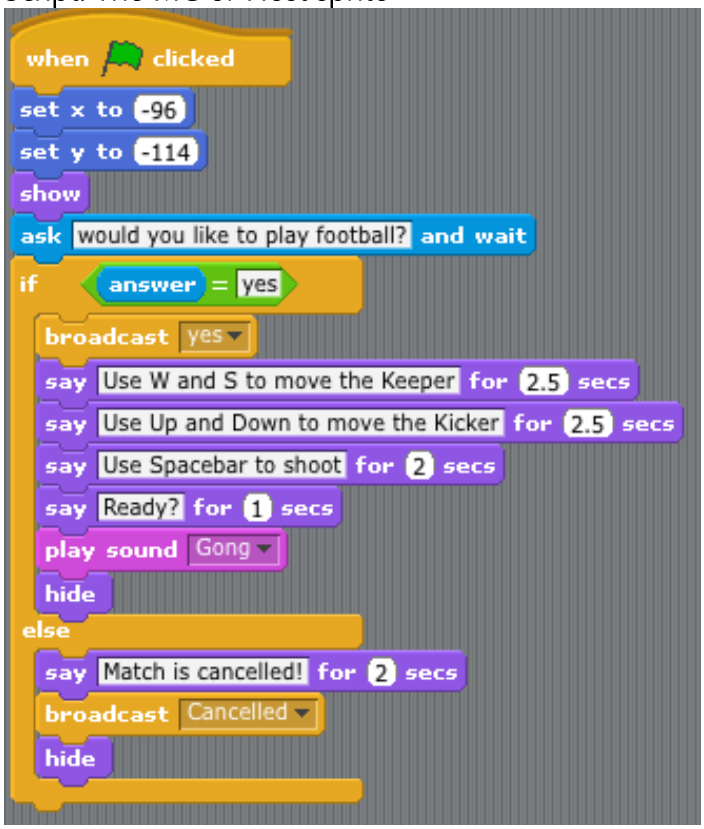
Note that the *Ask command* (*Sensing* folder) allows the user to input an answer. The veracity or falseness of the reply and the resulting responses are controlled by the commands inserted.

Draw out a suitable section of a soccer pitch.

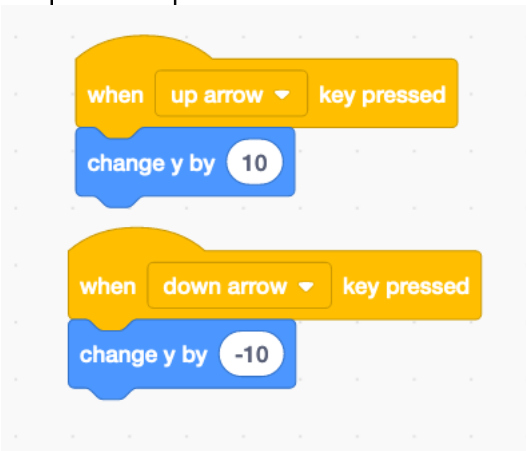
Introduce sprites that will take on the roles of a striker, a goalkeeper, a football and an official host.

Note: In the scripts below, all X and Y positions for the striker, a goalkeeper, a football and the host are all specific to the demonstration project only and will not conform to any other project. They exist only as guidelines.

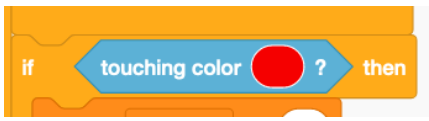
Script: The MC or Host sprite



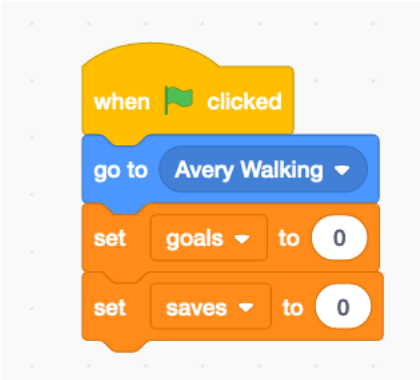
Scripts: Ball sprite

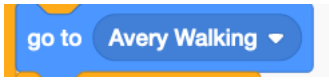


```
when [space] key pressed
repeat until <(goals) = (5)> or <(saves) = (5)>
    move (-20) steps
    if <touching [Amon] ?> then
        change [saves] by (1)
        broadcast [message1]
        wait (.5) seconds
        go to [Avery Walking]
        stop [this script]
    if <touching [edge] ?> then
        broadcast [m2]
        wait (.5) seconds
        go to [Avery Walking]
        stop [this script]
    if <touching color (red) ?> then
        change [goals] by (1)
        play sound [Goal Cheer] until done
        broadcast [m3]
        wait (.5) seconds
        go to [Avery Walking]
        stop [this script]
```

Note: The 

in the above script refers to the *colour* of the goal nets which have to be touched by the ball in order to register a score.

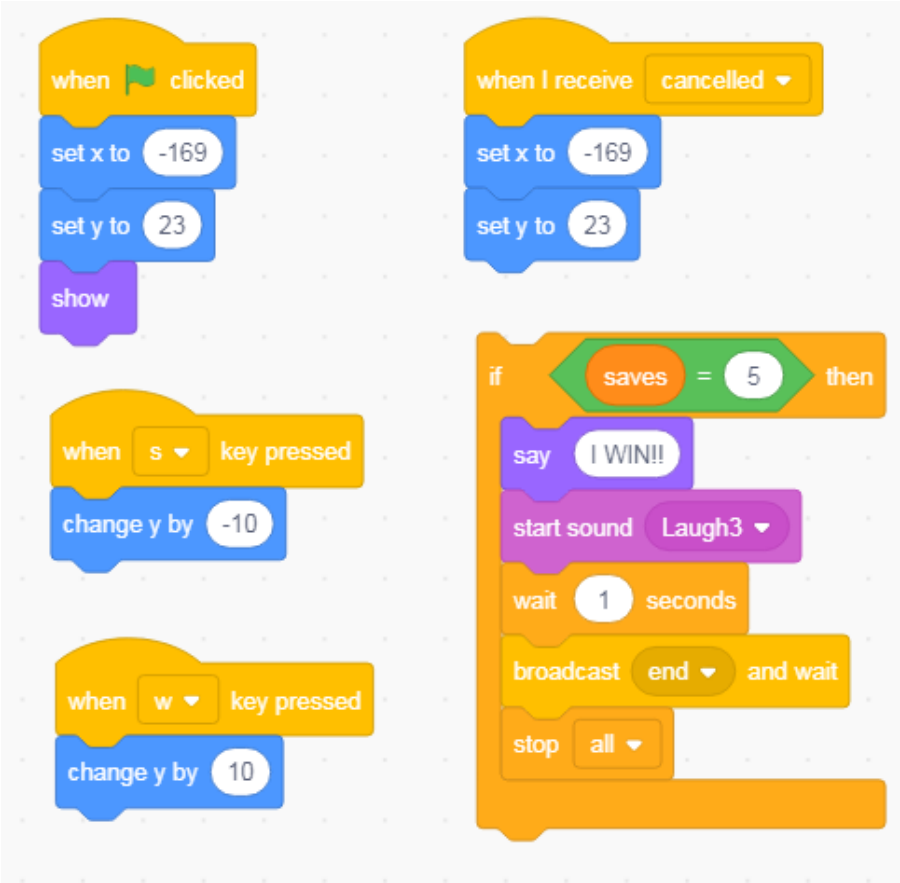


In the above text, the 

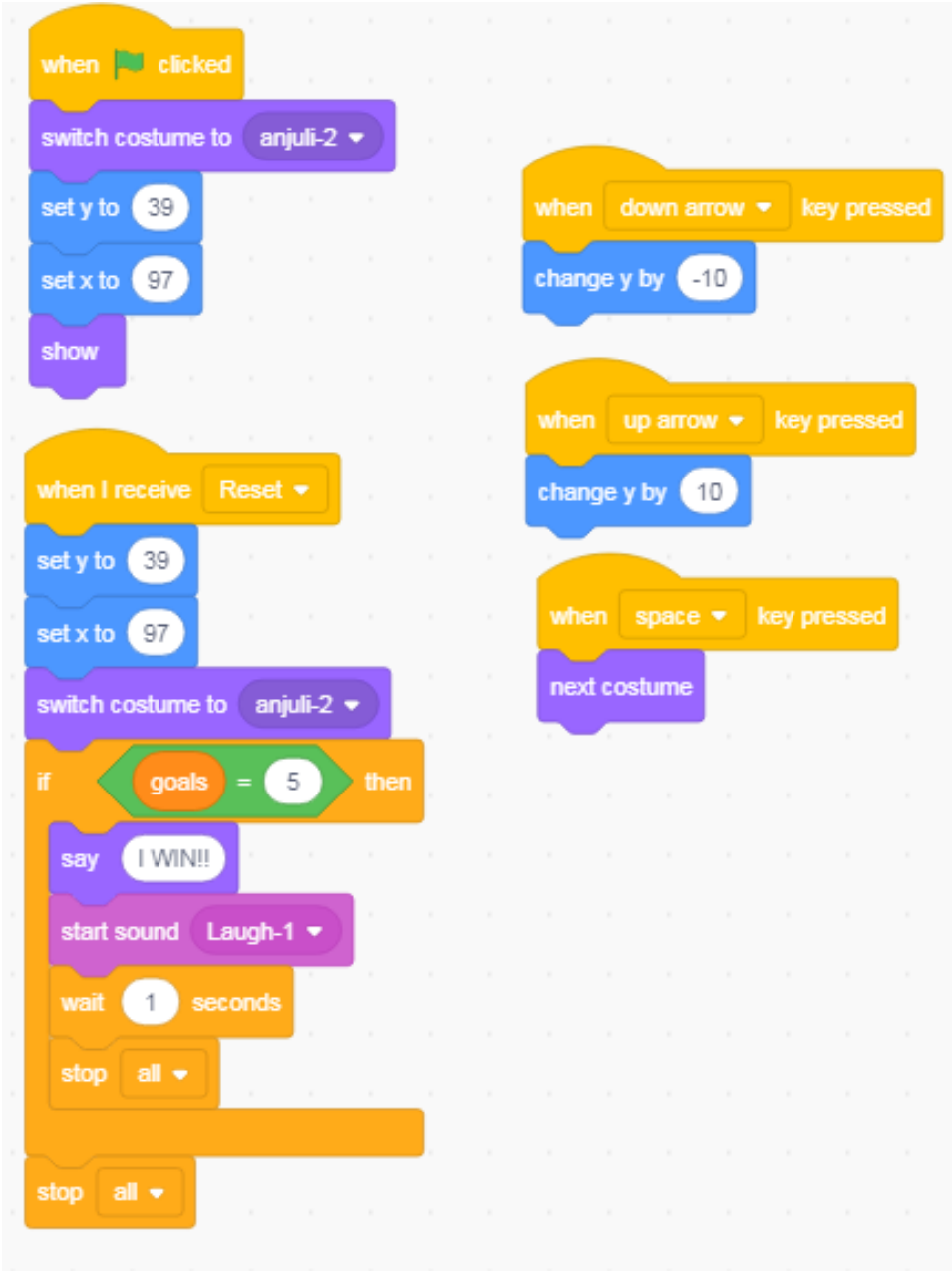
block appears beside the striker. However you will probably need to go into the costumes' *Paint Editor* of the striker's sprite in order to have the ball positioned at the feet of the striker.

After doing so, move the striker's sprite out of the middle of the screen in the *Paint Editor* using the *Arrow* positioning icon located at the top left-hand corner of the tools menu within *Vector* mode. However if the sprite comprises multiple parts that may move independently of each other using this tool, go to the tool  located at the top right of the menu which allow it to be manipulated as one entity. Once you move the sprite, you will see a very small ***target*** marking. Move the striker sprite until its feet are behind the ball sprite.
Looking at the stage will help you get the proper alignment.

Scripts: Goalkeeper sprite


when clicked
set x to -169
set y to 23
show
when s key pressed
change y by -10
when w key pressed
change y by 10
when I receive cancelled
set x to -169
set y to 23
if saves = 5 then
say I WIN!!
start sound Laugh3
wait 1 seconds
broadcast end and wait
stop all

Script: Striker sprite



Exercise

Create a *hockey* or other sporting game based on the same or similar structure to the above Soccer Striker game.

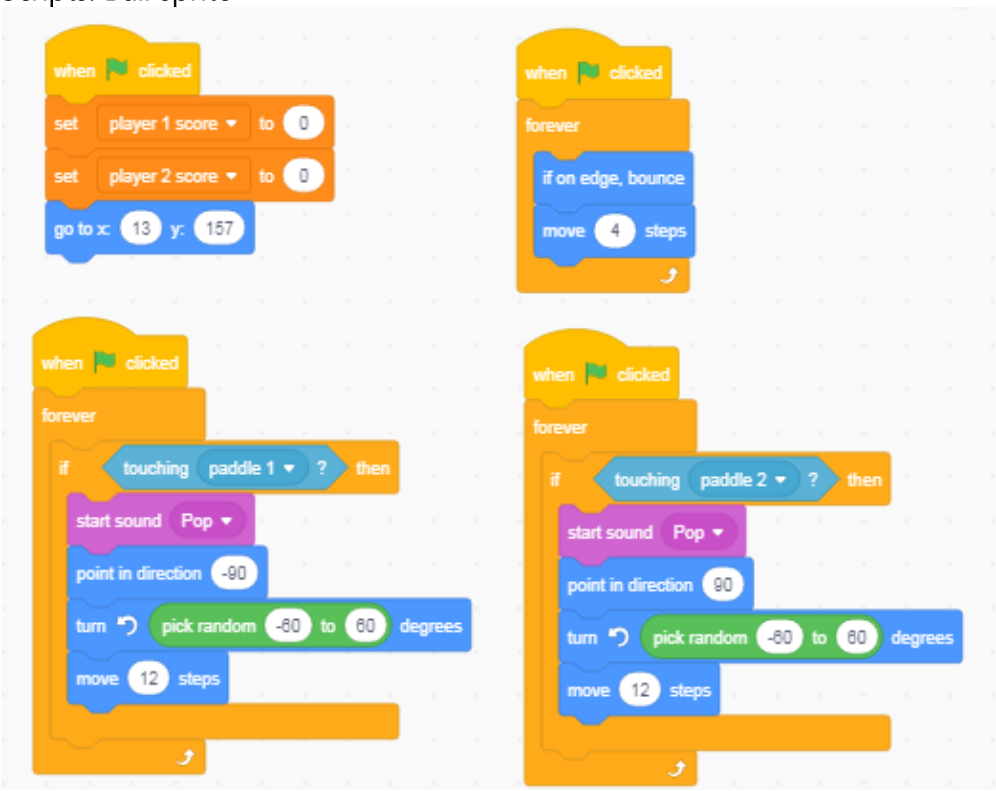
Lesson 29 - Two Player Games: Tennis for Two

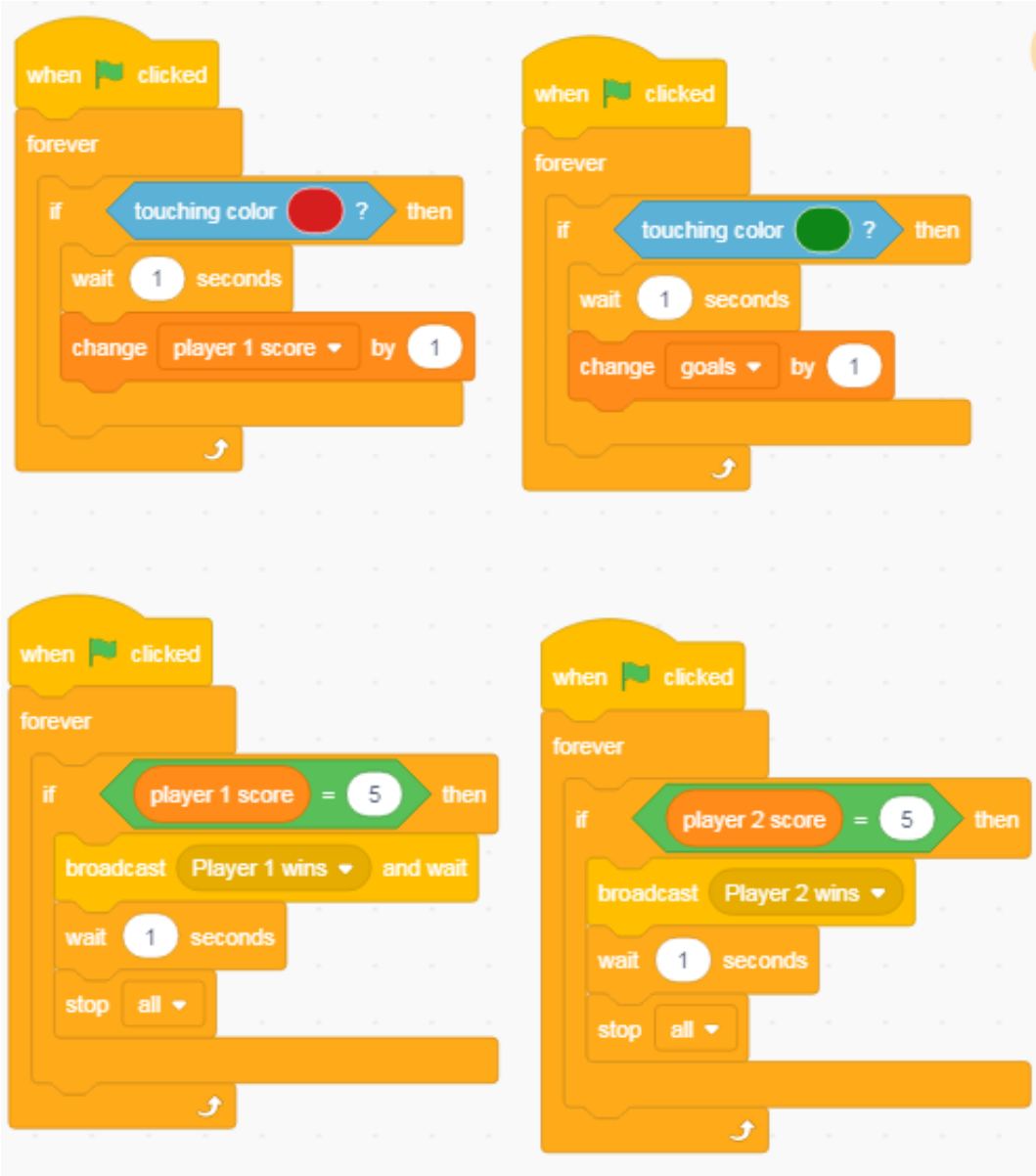
Even more than the single player version used in an earlier lesson, this two player Tennis game is closer to the original classic game known as ***Pong****.*

<u>Game Play - Coding Plan Summary</u>
The purpose of the game is for one of the two players to be declared the winner by getting the ball to touch the coloured line on the opponent's side five times. The Paddles act both as defensive shields and as offensive shooters. Each of the two paddles is controlled by a separate set of two adjacent keys (Up/Down Arrows & W/S).

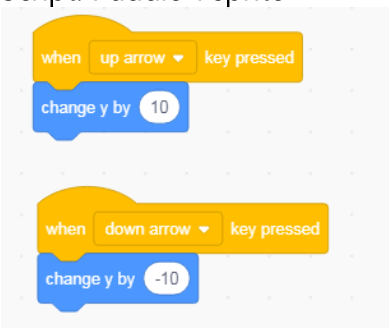
Key Command Blocks: *'forever if', 'When I Receive' & 'Broadcast' & Stop All (Control folder), 'Touching (a sprite)' (Sensing), 'Change by ____' & 'Set ___to' (Variables), Pick random __ to __ ' (Operators), Point in Direction __', Turn (Motion).*

Scripts: Ball sprite

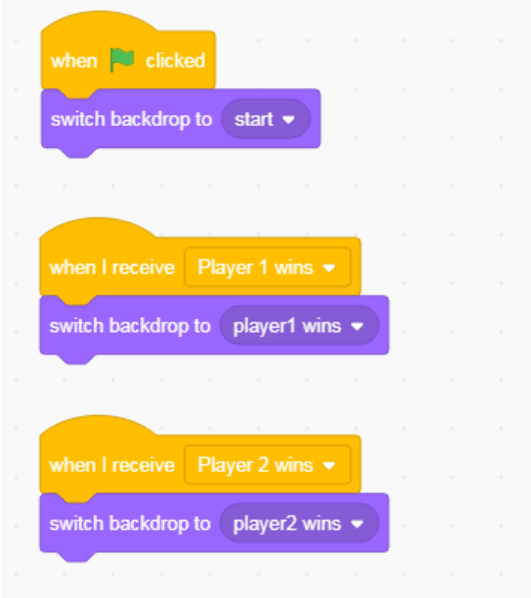



when clicked
forever
if touching color ? then
wait 1 seconds
change player 1 score by 1
when clicked
forever
if touching color ? then
wait 1 seconds
change goals by 1
when clicked
forever
if player 1 score = 5 then
broadcast Player 1 wins and wait
wait 1 seconds
stop all
when clicked
forever
if player 2 score = 5 then
broadcast Player 2 wins
wait 1 seconds
stop all

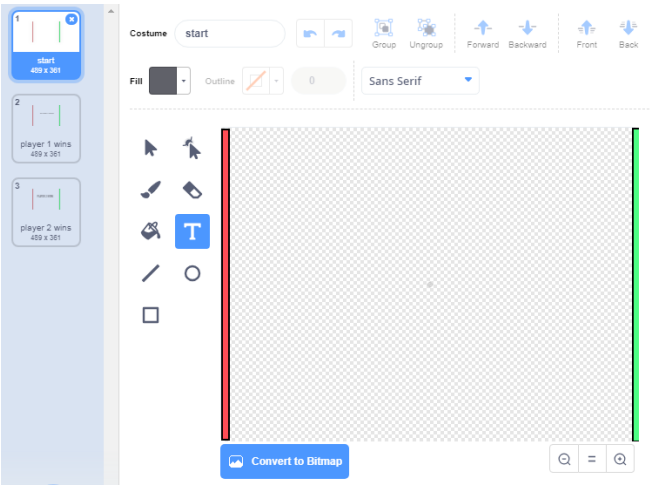
Script: Paddle 1 sprite



Script: Stage sprite



Script. Stage costumes: a) Standard version with 2 coloured goal lines at opposite ends of the screen, b) Player 1 Wins! and c) Player 2 Wins!



Lesson 30 – Geography Quiz: Travelling across a Continent

*Coding a **Quiz** using Scratch is a popular activity in schools. The themes range from general knowledge to a specific subject. In the latter area, it can give an exciting new dimension to the teaching of languages, sciences, history and geography. The students can be given the opportunity to research the questions and answers of their chosen subject as well as to build an exciting **question and answer** project that will capture the interest of people participating.*

The sample project below is of a tour of Europe. The questions here are based on naming the countries. But it could be instead based on naming the main languages, capitals, the head of states, popular tourist destinations, mountains, rivers, lakes etc.

<u>Project Play - Coding Plan Summary</u>

The tour guide is Daire who travels across Africa stopping at each country to ask a player to type in its name onscreen.

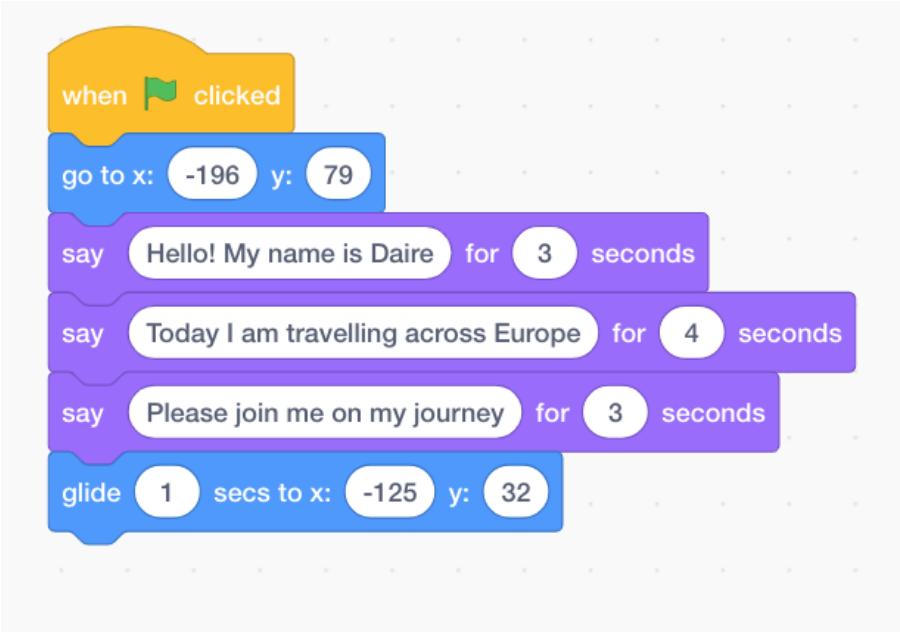
The respondent cannot move from the country until he/she types in the answer correctly. Once the correct answer is inputted, Daire travels to the next country and the next asking the question until all selected countries are visited. This process is continued until all countries are visited.

Key Command Blocks: 'Forever if', 'If ___ Else', 'Repeat until___', 'When I Receive' & 'Broadcast' & Stop All (Control folder), 'Say' (Looks), 'Touching (a colour)', 'Ask', 'Answer' (Sensing), 'Change by ___' & 'Set ___to' (Variables), '___or ___' (Operators), Turn (Motion), 'Play sound ___' (Sounds).

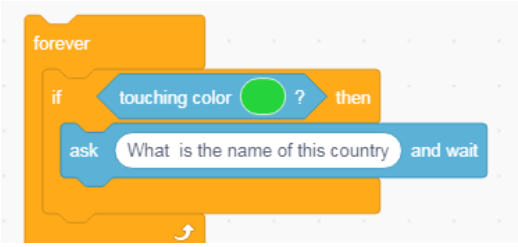
One Sprite (Daire) and one backdrop (map of Europe) with only a script for the former.

Scripts:

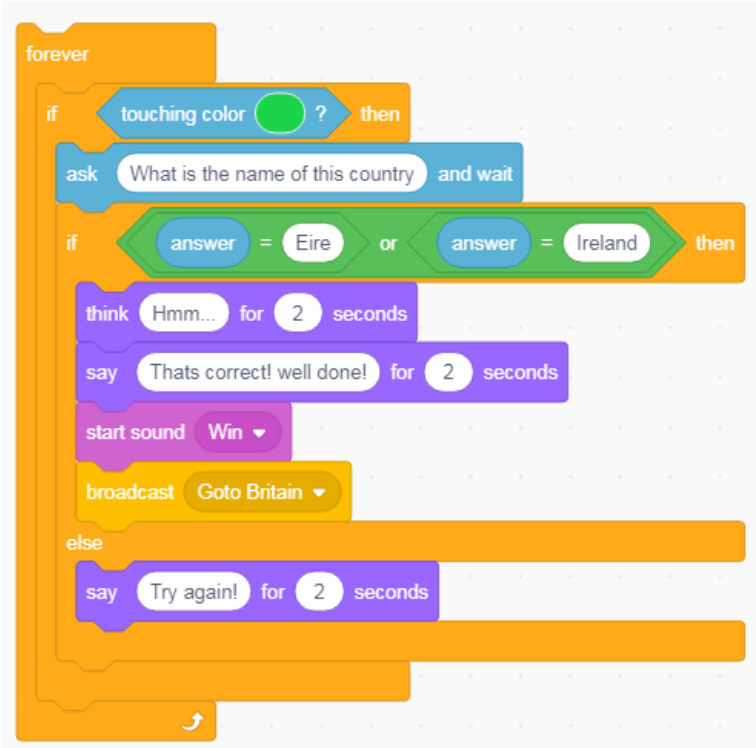
Part 1



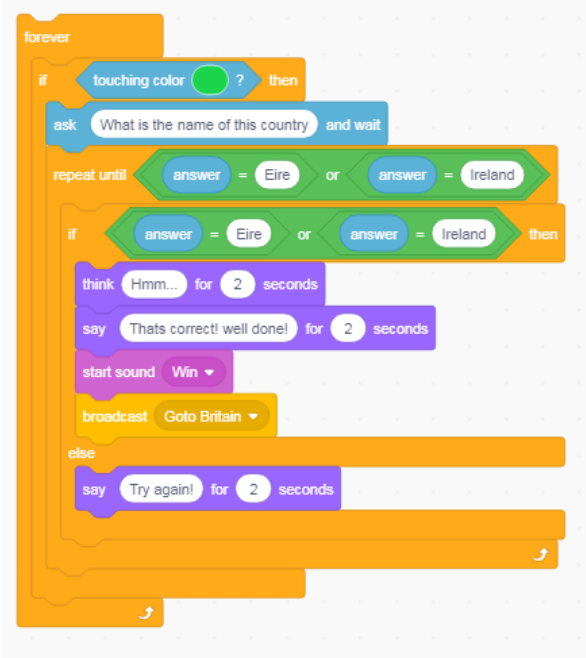
Part 2



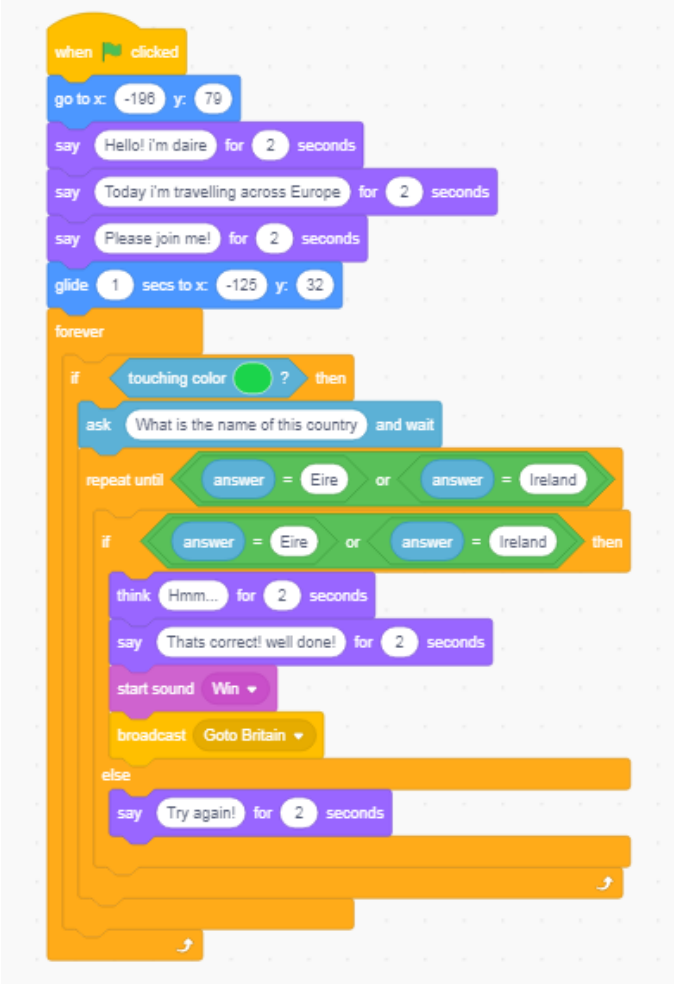
Part 3 (containing Part 2 above)



Part 4 (containing Part 3 above)

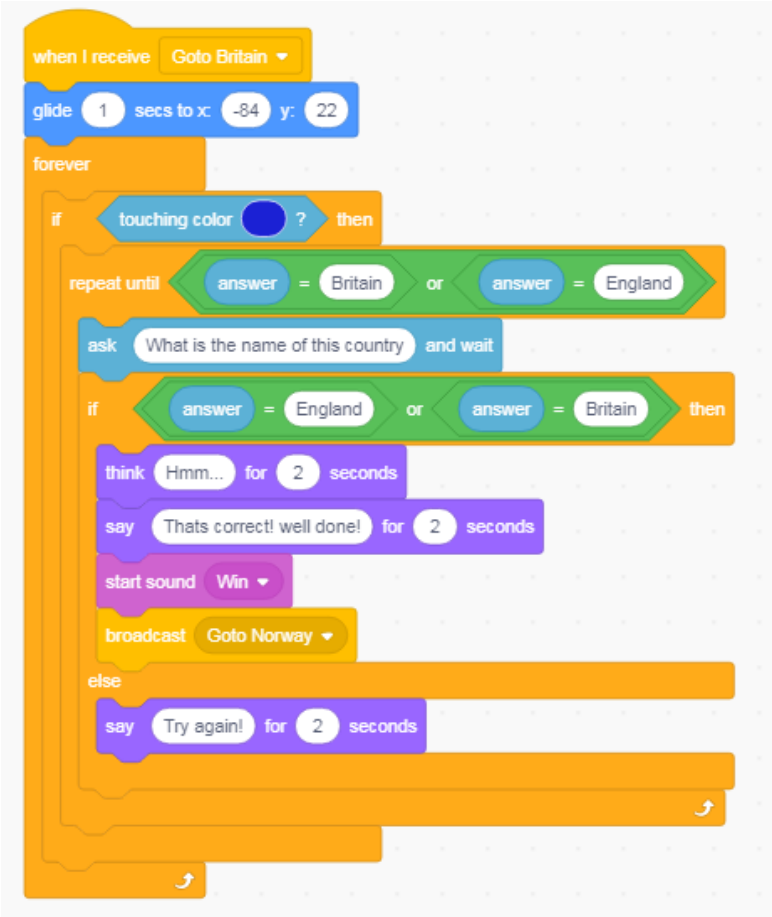


Part 4 (containing Part 1 and Part 3)



Note: Éire or Eire is the Irish name for Ireland

Script for the country *Britain*



Follow the same procedure for all the other marked countries.

Exercise

Get the children in the class or the participants in the session, as individuals or in groups, to make a similar project based on the continent of Africa.
This will involve a lot of advanced research and planning in order to obtain the necessary relevant information (e.g. capitals of countries, languages spoken, geographical features such as main rivers, mountains, etc).

[i] Note: All Lesson Plans compiled by Brendan Smith, Camden Education Trust, Galway, Ireland.