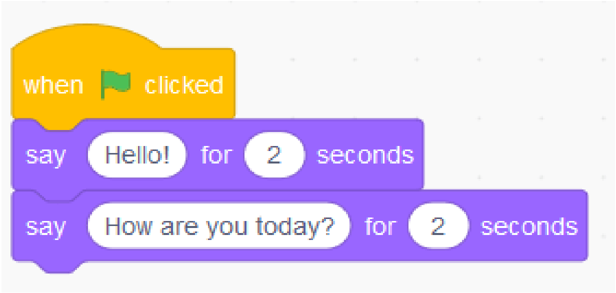
TRAINING MODULE

ON CODING FOR MIDDLE SCHOOL TEACHERS

Centre for Mathematics, Science and Technology Education in Africa (CEMASTEA)



```
when 🏁 clicked
say Hello! for 2 seconds
say How are you today? for 2 seconds
```

ISO 9001:2015 CERTIFIED

CEMASTEA –NAIROBI, KENYA 2023
CENTRE FOR MATHEMATICS, SCIENCE AND TECHNOLOGY EDUCATION IN AFRICA (CEMASTEA)
P. O. BOX 24214-00502 Karen – Junction of Karen Road and Bogani Road, NAIROBI – KENYA
Tel Numbers: 020 2044406 / 0706 722697 / 0780 797648
E-mail: director@cemastea.ac.ke
Website: http//www.cemastea.ac.ke

Approved for circulation

CENTRE FOR MATHEMATICS, SCIENCE AND TECHNOLOGY EDUCATION IN AFRICA
COORDINATOR ACADEMIC P
09 JUN 2023
P.O. Box 24214 - 00502, NAIROBI

for JACINTA AKATSA, HSC

CEO, CEMASTEA

Citation

CEMASTEA (2023). Training Module
on Coding for Middle School Teachers. CEMASTEA: Nairobi, Kenya

KEBS ISO 9001:2015 CERTIFIED

Acronyms and Abbreviations

CBC -Competency-Based Curriculum

CEMASTEA-Centre for Mathematics Science and Technology Education in Africa

PCIs-Pertinent and Contemporary Issues

Acknowledgment

The CEO of CEMASTEA would like to thank the following staff for their input in the development of this module.

1. Jacinta Akatsa -CEO
2. Makoba Kizito
3. Martin Mungai
4. George Kiruja
5. Nancy Nui
6. Francis Kamau
7. Philip Maate
8. Richard Jakomanyo
9. Clotilda Nyongesa
10. Rahab Chiira
11. Agnes Mwangi
12. Dr. Njoroge Mungai
13. Clara Mwangi
14. Esther Kamenwa
15. Lucy Mwaniki
16. Paul Lomosi

Introduction

One of the learning areas in grades 4 and 5 is science and technology. Giving students a foundational understanding of science and computing is essential. One of the sub-strands in this learning area is coding. Coding proficiency is a necessary ability in today's technologically advanced environment. It is crucial to introduce coding throughout the early learning years because it introduces students to computational and design thinking at a young age. In a research carried out by CEMASTEA on coding it was observed that teachers needed support to teach coding (CEMASTEA,2022). This informed the need to prepare materials on coding and train teachers. The module is consistent with the requirements of the curriculum designs for grades four and five. The following topics are covered in the module to address the proposed learning outcomes namely: an introduction to coding; patterns and games; Overview of Scratch; Animation using Scratch; Games and Graphics using Scratch; and Scratch Project. The interactive teaching method used in the module gives teachers the chance to get hands-on experience with coding from the beginning while creating numerous activities and projects.

Rationale

The Science and Technology curriculum designs for grades 4 and 5 and the baseline study by CEMASTEA (2022) served as the foundation for the content of this module. The creation of this module will enable teachers to acquire instruction in the fundamentals of coding.

Theme: *Enhancing teachers' capacity to teach coding at grades 4 and 5*

Expected training Outcomes

By the end of the training, the participant should be able to:

1. Identify the features of learning applications which imitate simple programming
2. Interact with patterns and games using available learning applications which mimic simple programming
3. Identify a learning platform for creating stories, games, and animations
4. Create simple animations using applications that mimic simple programming
5. Create simple games and graphics for enjoyment
6. Use available learning applications to find solutions to problems in the local environment

Expected training outputs:

1. A scratch project problem solving
2. A lesson to teach coding

Section 1: Introduction to Coding

Welcome to Section One of this Module. This section will equip you with the knowledge and strategies needed in learning coding. In today's digital era, coding has become an essential skill for children. To make the learning process engaging and enjoyable, exciting activities that combine game puzzles and mazes are used to introduce the learners to basic coding skills.

Reflection 1

Share your favorite childhood games and toys, and briefly discuss why you enjoyed.

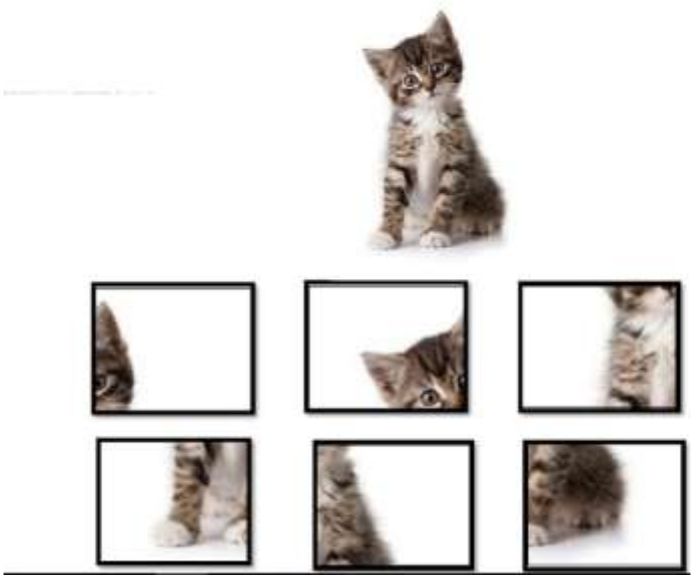
- You will realize that it is all connected to creativity, problem solving and playfulness.

Reflection 2

Share the strategies you used to make others learn the game.

- Effective teaching strategies you may have mentioned:
- ○ Providing clear simple instructions
- ○ Incorporating demonstrations (Hands on activities)
- ○ Encouraging creativity and self-expression
- ○ Offering scaffolding and guidance

All these are essential elements for coding. In grade four, learners were exposed to codes in a playful way. Do you think the tasks below can help develop problem-solving skills, logical thinking, and computational creativity?
Perform the following tasks.



Task 1

Jig-saw puzzle;

- Cut out the pieces and build the cat above
- Use the link to play the jig-saw game

https://www.jspuzzles.com/en/transportation/vehicle/truck/pickup-truck/2459888

Task 2

Tell your friend how to move the smiley face to the last box where the lollipops are.

Note: The smiley face can only move vertically or horizontally!

Start Here

☺	3	8	12
9	7	10	6
1	2	4	15
5	11	13	

End Here

Task 3

In the grid fill in the missing numbers using; 1,2,3,4. No number should be repeated horizontally or vertically**.**

	3		1
2			
			3

Task 4: Discussion

Discuss how Tasks 1,2 and 3 can help develop problem-solving skills and logical thinking.

Young children are curious and enjoy exploring and experimenting. A playful approach to coding taps into their innate sense of play and makes the learning experience enjoyable. Creativity, logical thinking, problem-solving, and playfulness are essential elements in coding.

What is coding?

Coding can be defined as the process of developing and implementing various sets of instructions to enable a computer to perform a certain task, solve problems, and provide human inter-activity (Balanskat & Engelhardt, 2014).

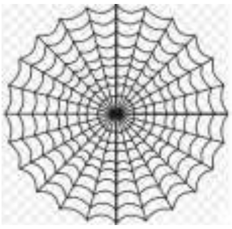
What is programming?

Programming is giving instructions to a computer or a device to perform specific tasks or solve problems. It involves writing a series of step-by-step instructions called codes that tell the computer what to do. Just like we use words to communicate with other people we use codes to communicate with computers.

A set of rules and commands that a computer understands is known as a programming language. There are many programming languages and coding apps available for young learners based on age appropriateness, educational value, and user-friendliness. These include Blockly, Swift Playgrounds, Kodable, Python, Tynker, Minecraft, Hopscotch, and Scratch. Scratch is widely used in schools and has a large online community where learners can share their projects.

Section 2: Patterns and Games

A computer code as defined earlier is a set of instructions that enables a computer to perform a particular task. Similarly, a pattern that uniquely identifies something can be referred to as a code. For example, a banana leaf has a pattern formed by the leaf veins which is unique to all banana leaves. A spider web contains a pattern that is unique to the type of spider making the web and so are nests of birds. All these are natural codes.



Spider web

Banana leaf

Weaver nest

Activity

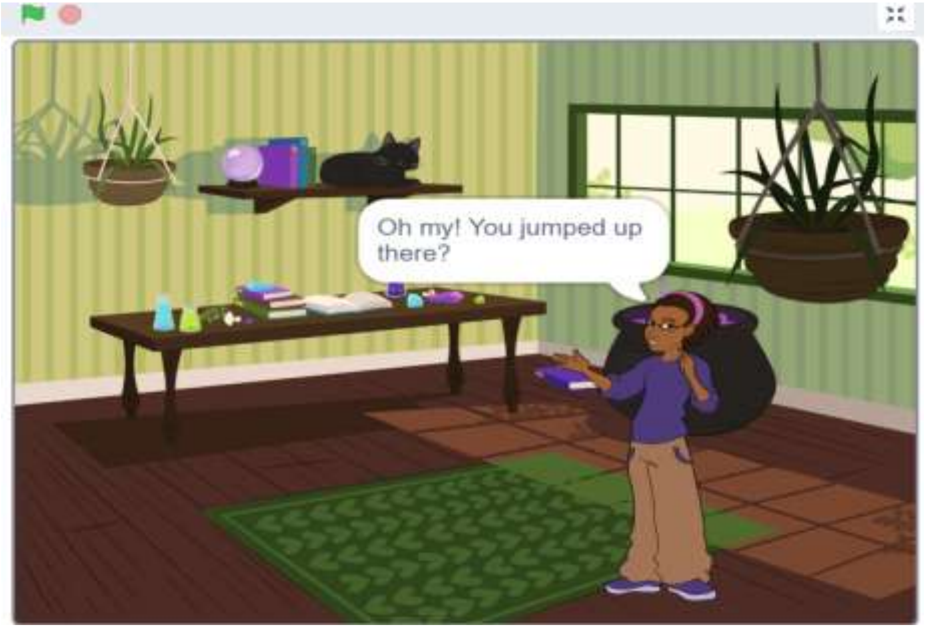
Click the link https://scratch.mit.edu/projects/394885714/fullscreen/ and observe the animation. Describe your observation.

a) Games and animations

In this section, we are going to play games or use animations created with scratch applications. The following shows examples of animations and games created using scratch.

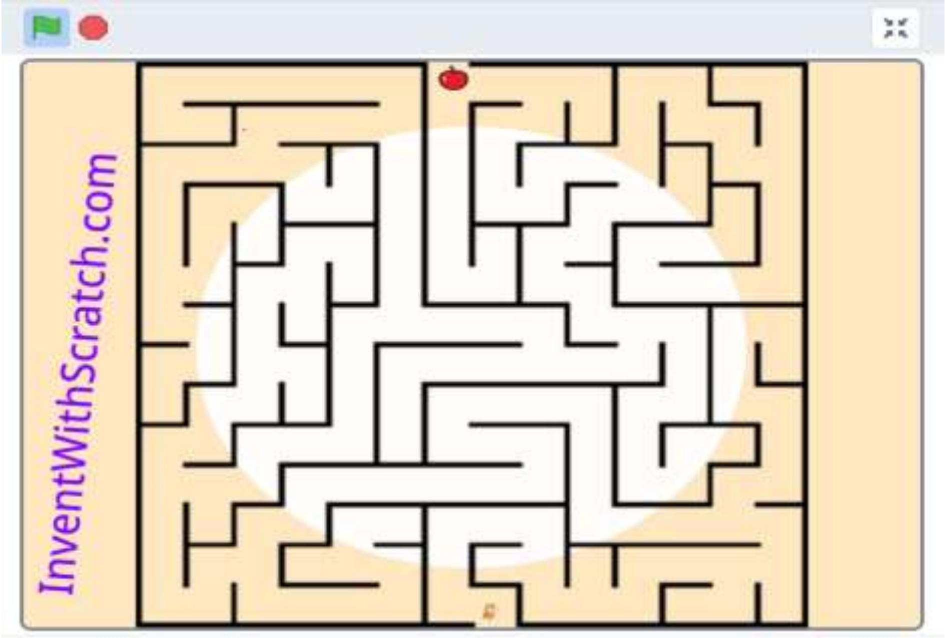
Click the link {my_story.sb3} to view an animation created using a coding application. Once clicked you will see a page as shown below:

Click the full control icon to maximize the image. Once the image is full screen click the green flag icon ⚑ to play the animation. Observe what happens and describe your observation.



Animated story

Click the link https://scratch.mit.edu/projects/11710850/editor/
You will access a maze. Maximize the maze so that it fills the page by clicking the full control icon. Click the green flag and use the arrow keys (for a laptop) to move the sprite (Cat icon) to the apple. For a smartphone or tablet you can drag and navigate the icon along the maze all the way to the apple.



Maze game

Section 3: Overview of Scratch

What is Scratch?

Scratch is a visual programming language that allows learners to create their own interactive stories, games, and animations. As the learners design their own projects in Scratch, they develop the skills of thinking creatively, reasoning systematically, and working collaboratively.
https://www.uab.edu/icac/images/Scratch_Guides/Intro_to_Scratch.pdf

Definition of terms as used in Scratch

a) Interface - Scratch software

b) Programming **-** the process of creating sets of instructions or commands that tell a computer what to do.

c) Coding **-** refers to the process of writing instructions or commands in logical and sequential steps known as codes that a computer can understand and execute.

d) Code blocks **-** puzzle-piece-like shapes which represent a set of commands that instruct the computer on the task to be performed by the character in the story.

e) Sprite - the main character(s) in the story created. They can be made to move around, change the appearance or even react when they touch something.

f) Animation - display of a sequence of images of the sprite which create an illusion of movement

g) Gaming **-** the art of creating a game in Scratch

h) Backdrops **-** background displaying the scene of the character(s) in the story.

i) Project **-** story/game created in Scratch

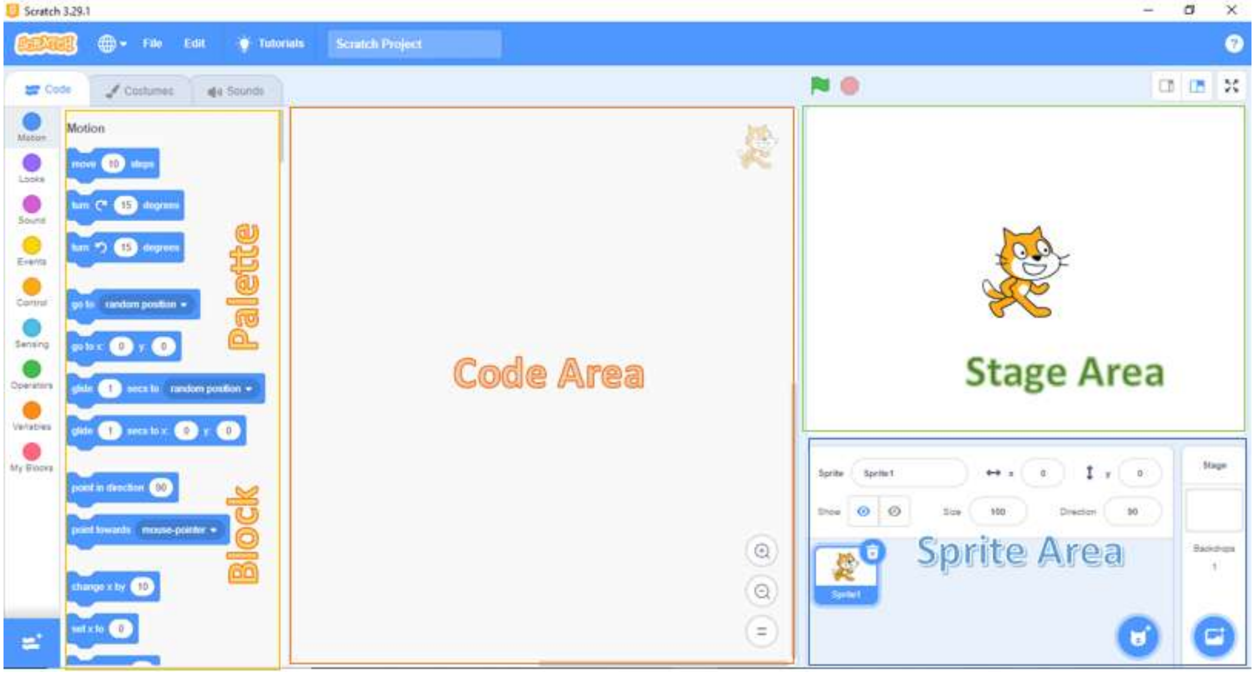
j) Costume **-** A costume is a different appearance for the same sprite allowing a sprite being controlled by one or more scripts to appear to change its looks.

Getting started with Scratch

Double click on the Scratch icon to get started.

This brings you to the interface home page which has 4 main elements namely;

1. Block Palette - also known as the programming palette
2. Code area - also known as the scripting area
3. Stage area
4. Sprite area



.

***Block palette**:* This contains instruction blocks (code blocks) used to program the sprite to do or say something.

Coding area**:** This is the space where the code blocks are dragged and dropped as you write a program(story/games).

Sprite area: This is the area or space where the selected sprites appear and can be interchangeably controlled for display in the coding area. It defines the boundaries within which the sprite can move and interact with other objects. It is represented by a rectangular shape that encloses the sprite's image or costume

Stage: This is the primary area where the action of the program is displayed as performed by the sprite. The stage may be of different backgrounds.

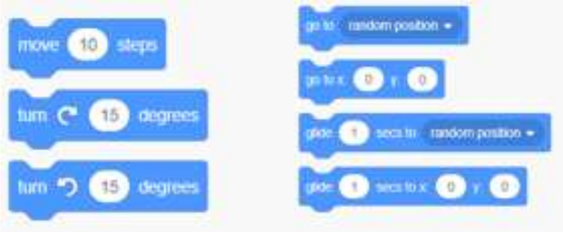
The code blocks

In writing a program, a set of commands is used to instruct the computer how to perform a certain task or how to solve a certain problem. This is done by typing the commands. However, in Scratch, code blocks are used to represent these commands. These code blocks are represented by colorful puzzle-piece-like shapes that fit together to create a script or program. The code blocks have notches at the top and bumps on the bottom.

To create a story/game/animation, you need to stack the blocks in the appropriate order. This is done by dragging a code block from the appropriate category in the block palette and dropping it in the coding area as you build a set of commands for the story/game/animation. The categories include **motion**, **looks**, **sound**, **events**, and **control** among others. Examples of these code blocks in these categories are given below.

Motion Blocks:

These blocks in Scratch are used to move or turn sprites around the stage. For example:

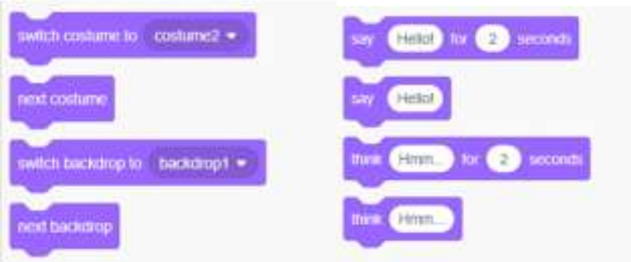


```
move 10 steps
turn ↻ 15 degrees
turn ↺ 15 degrees

go to random position ▾
go to x: 0 y: 0
glide 1 secs to random position ▾
glide 1 secs to x: 0 y: 0
```

Looks Blocks:

These code blocks are used for the purpose of controlling a sprite’s appearance (size, costume, or visibility). They are also used to add speech bubbles and change the backdrop.

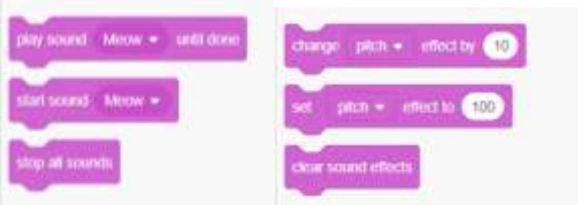


```
switch costume to costume2 ▾
next costume
switch backdrop to backdrop1 ▾
next backdrop

say Hello! for 2 seconds
say Hello!
think Hmm... for 2 seconds
think Hmm...
```

Sound Blocks:

These code blocks are useful in adding music and sound effects. For example,



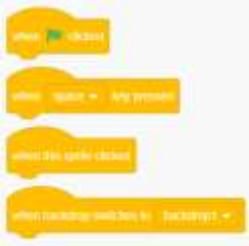
```
play sound Meow ▾ until done
start sound Meow ▾
stop all sounds

change pitch ▾ effect by 10
set pitch ▾ effect to 100
clear sound effects
```

Event Blocks:

These are the most important code blocks in Scratch. These blocks are used to start the code. They always come at the beginning of the code.

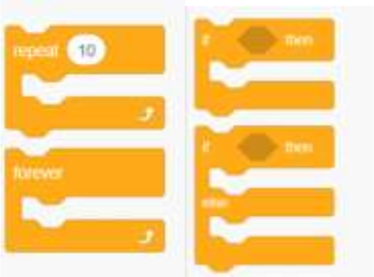
For example, *when the green flag* *is* *clicked* block.



Control Blocks:

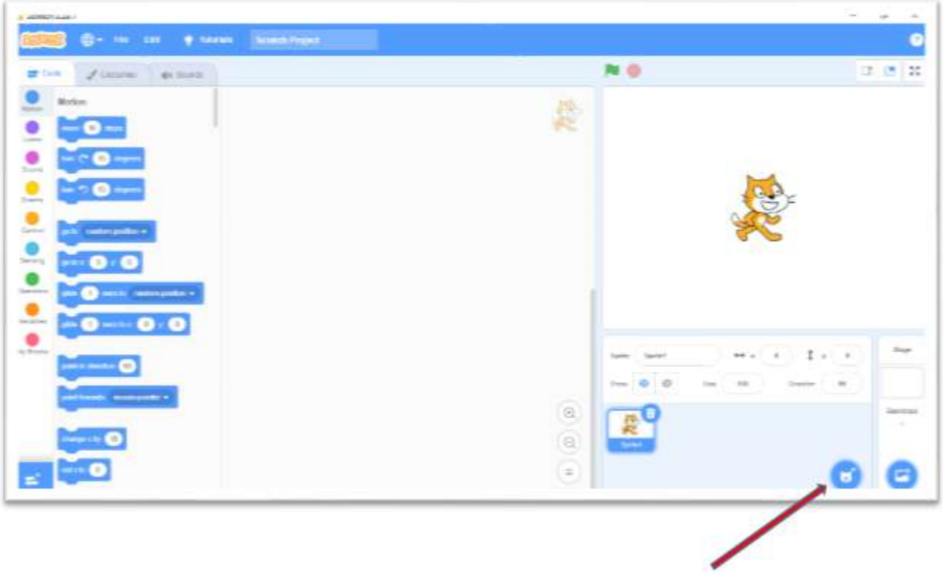
Control blocks are used in Scratch to give certain conditions. Some of the commonly used control blocks are as shown below.

Repeat Blocks and *If then else* Block:



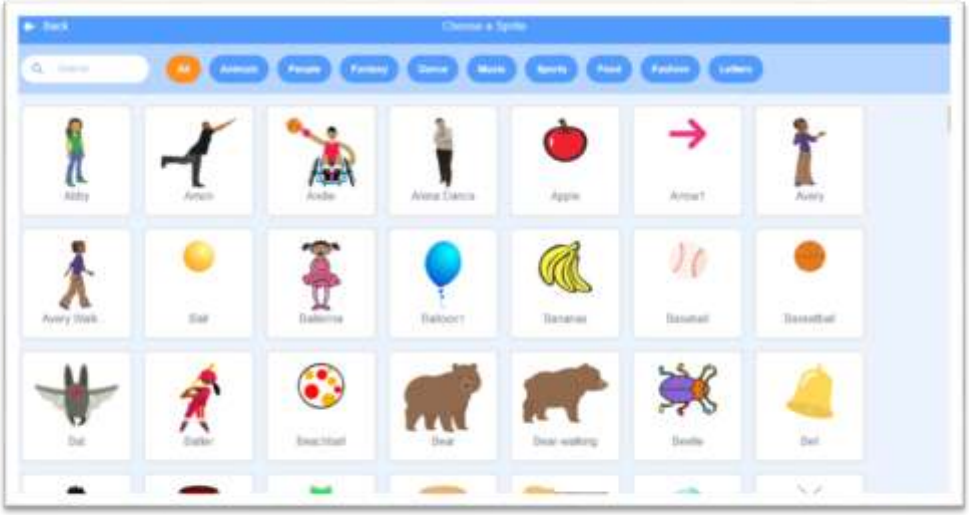
Accessing other sprites from the interface

A story may have more than one sprite. Scratch provides a library of sprites from which other sprites can be accessed. To change or add the sprites, you can select a different sprite from the library or you can create your own. This is done by clicking on the "Choose a Sprite from Library" button located below the stage area.

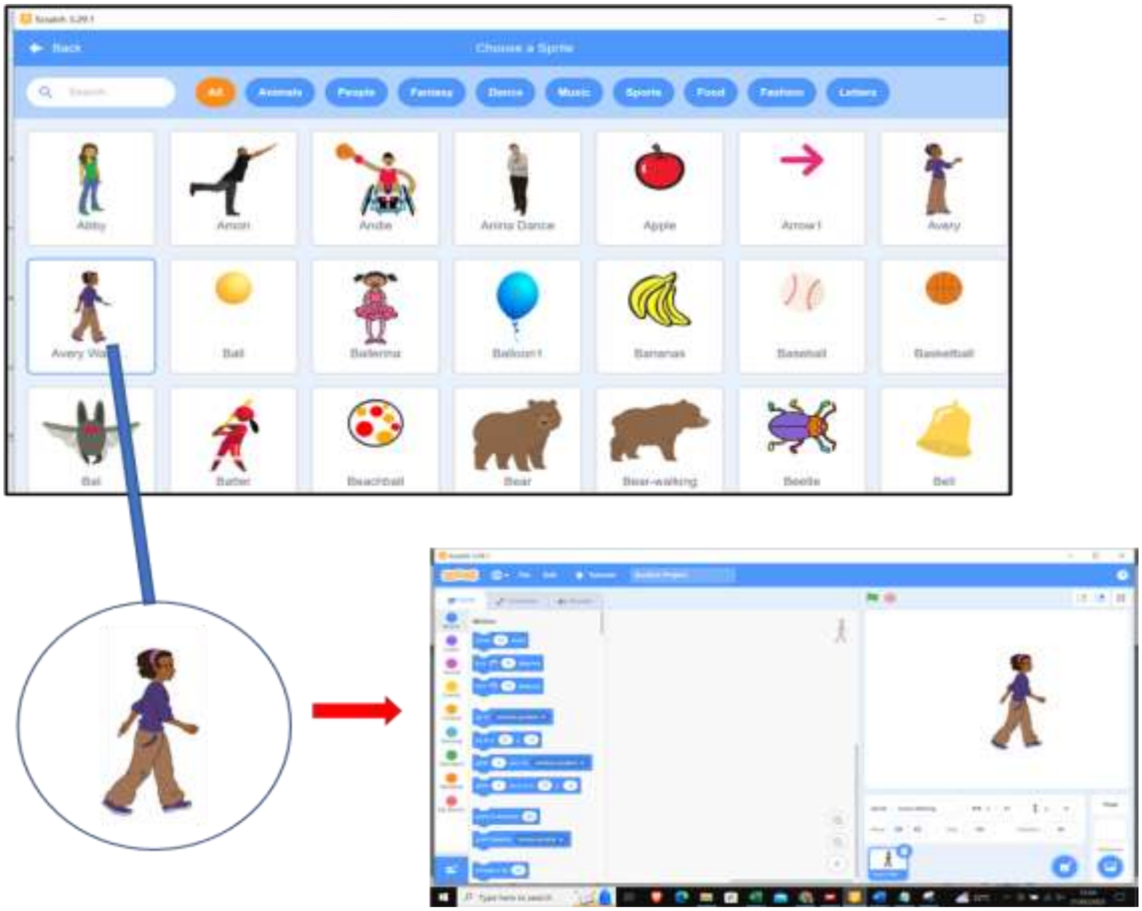


Accessing other sprites

Browse the sprite library: In the Sprite library, you'll find various categories and options. You can choose from.

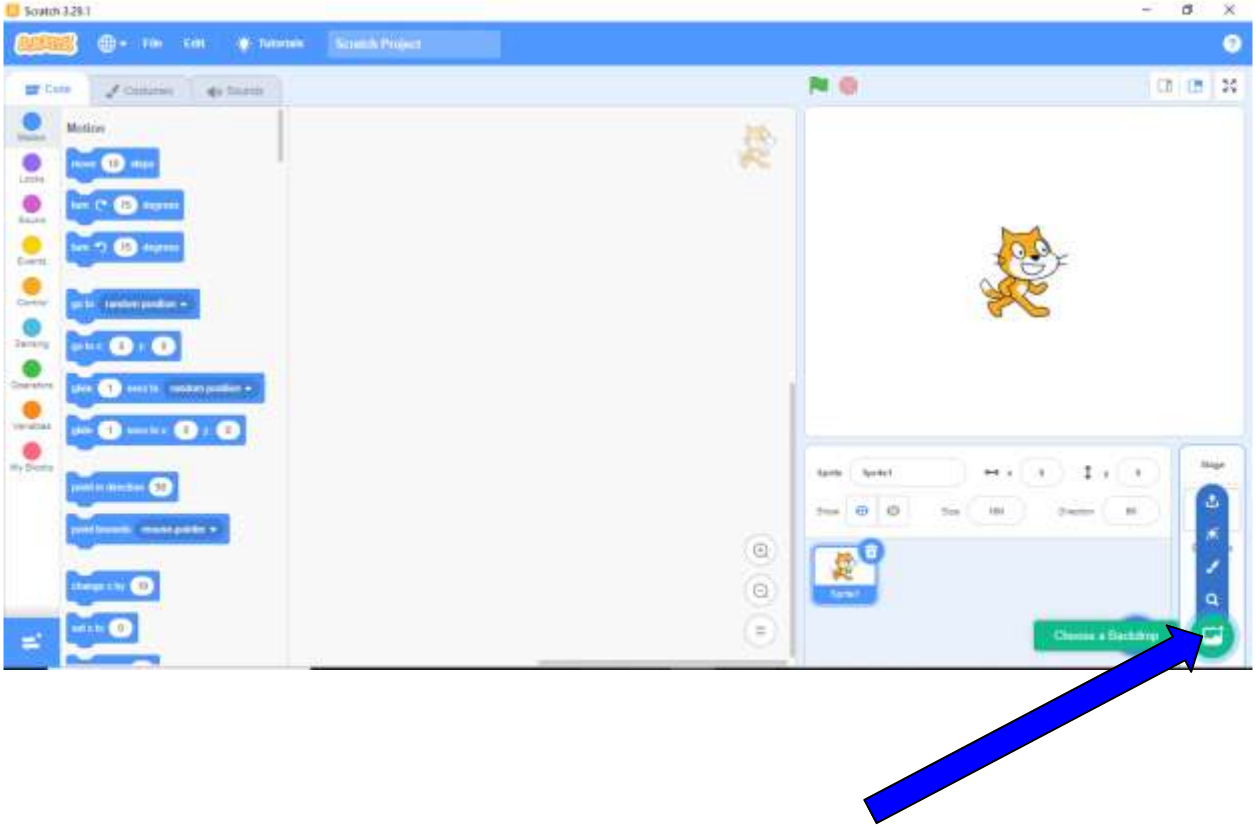


For example, we can select the sprite as indicated in the diagram below and see how it changes on the stage area.

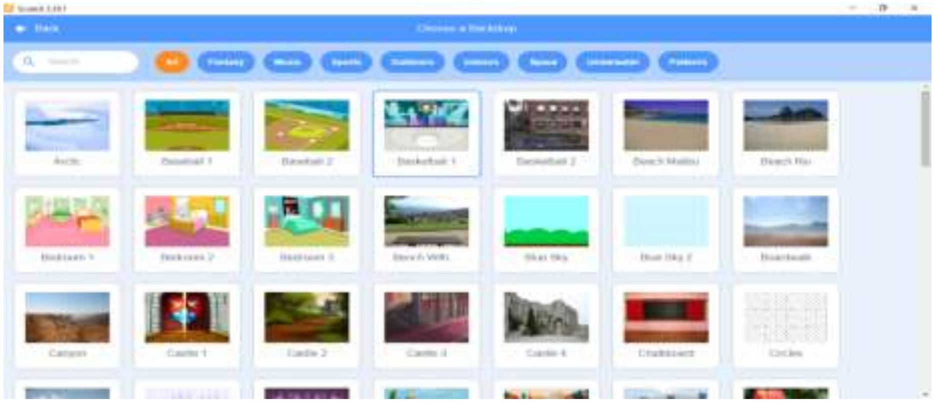


Changing Backdrops on the scratch interface.

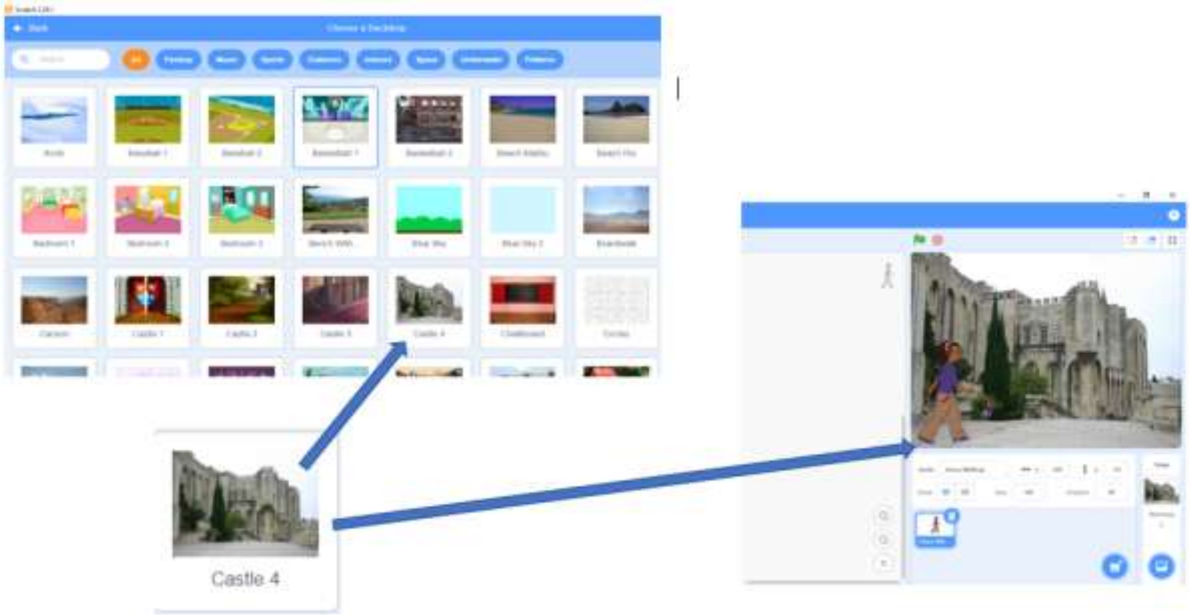
In Scratch, a backdrop is an image that can be shown on the stage. Scratch provides a variety of options for backdrops. To change or add a new backdrop we click on the “choose a backdrop’ button on the bottom right corner of the sprite area.



Browse the Backdrop library: In the backdrop library, you'll find various categories and options. You can choose from



For example, we can select a backdrop as shown below and see how it changes in the stage area



Activity 1

a) Using the ‘choose a sprite’ icon, select a different sprite of your choice for your project.

b) Select an appropriate backdrop for your sprite in (a)

Section 4: Animation using Scratch

Introduction

Welcome to animation in sscratch. In this section, we will focus on how to animate and create an animated project based on pertinent and contemporary issues (PCIs).

Reflection

1. What is animation?
2. What are some of the area where animations are applicable?

You may have come up with the following;

- Animation is the process of manipulating still figures (sprites in scratch) to create the illusion of movement. In scratch programming, sprites (icons) are manipulated to create a combination of live action.
- Animations are applied in various areas such as; education, entertainment and gaming, advertisement and marketing, architecture and engineering.
- Animation can be applied to facilitate learners' understanding of various aspects outlined in the Competence Based Curriculum (CBC), such as Pertinent and Contemporary Issues (PCIs).
- The process of creating an animation by learners can harness the core competencies, such as creativity and imagination.

How to animate

Animation in scratch programming can be summarized into two major steps.

I. Storyboarding consists of imagining and creating a series of actions to be illustrated by the sprite.

II. Programming the sprites to illustrate the actions.

Figure 1 indicates the spiral of creative learning by Resnick (2007), which shows the interaction between the two steps.

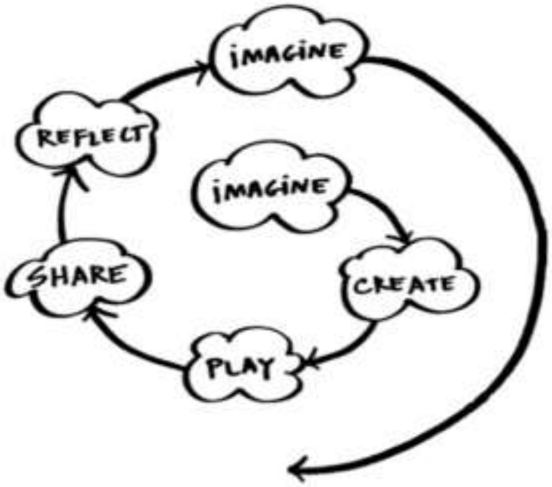


Figure 1: The spiral of creative learning "The kindergarten approach to learning" (Resnick, 2007).

The following is an example of animation based on the two steps.

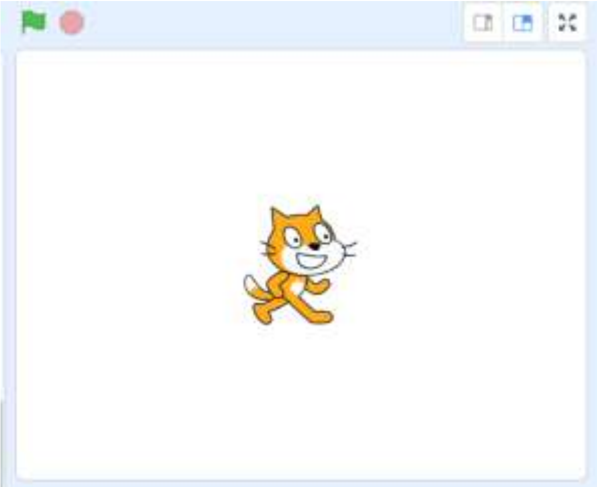
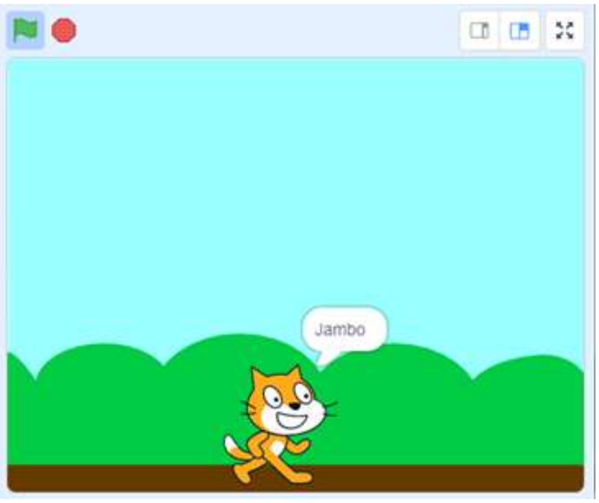
Imagine

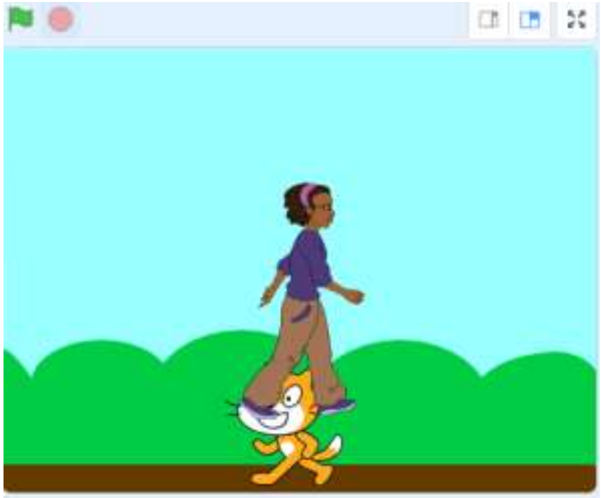
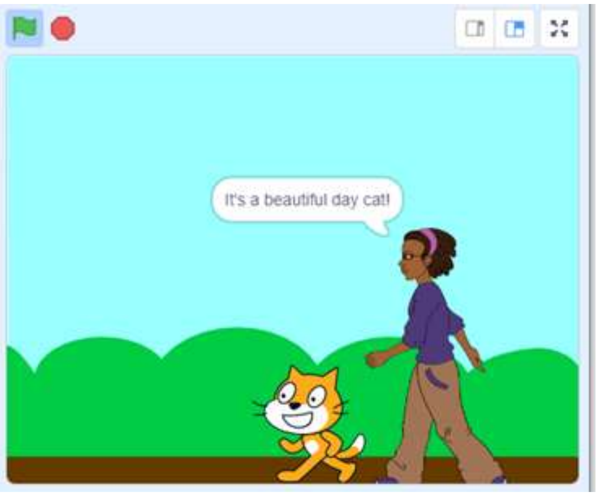
Reflection

- Imagine the actions you want your sprite to perform?

In Scratch the default sprite is a cat . We are going to animate the cat to talk and move around the screen using the ***‘move’*** code block and ‘***say’*** code block. The cat sprite is usually on a blank backdrop (background), we will change and have the cat move around a new backdrop by introducing a ***‘go-to’*** code block. Next we will make the cat jump for a specific number of times by introducing a new code block known as ***‘repeat’*** code block.

At this point we will save the existing project before introducing a second sprite known as ‘Avery Walking’ .We will reposition, turn and resize ‘Avery Walking’ and let her take a walk with ‘the cat’ as they chat. This will include the use of ‘***show’*** and ‘***hide’*** code blocks. Let us now apply visuals to the imaginations on a storyboard.

	
Make the Sprite cat to say ‘Jambo’ and move around on a blank backdrop	Change the backdrop and make the cat to turn and Jump

 Introduce the second sprite- “Avery walking”	 Turn and increase the size of Avery. The two sprites take a walk as they chat.

Create

Reflection

- How do you animate the sprites based on the imagination above?

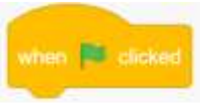
This involves writing commands to actualize the intended action as outlined on the storyboard. The commands can be adjusted to reflect changes made progressively on the storyboard.

Step 1: Make the Sprite cat to say ‘Jambo’ and move around on a blank backdrop

Open Scratch by double-clicking on the scratch icon on the computer's desktop.

On the left-hand side of the Scratch interface in the Blocks palette section, click on the Event category ● Events

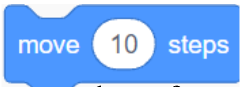
Drag and drop the following code block into the Script Area.



when 🏴 clicked

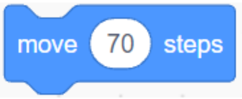
Click on the Motion category ● Motion

Drag and drop the ***‘move’*** code block on the script area



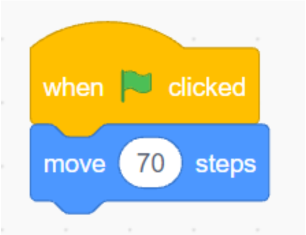
move 10 steps

Click inside the text area written ‘10’ steps and change the number of steps to ‘70’



move 70 steps

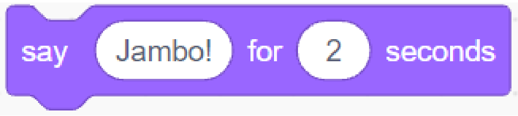
Attach the ***‘move’*** code block to the previous ***‘when-clicked’*** the command block as shown below.



```
when ⚑ clicked
move 70 steps
```

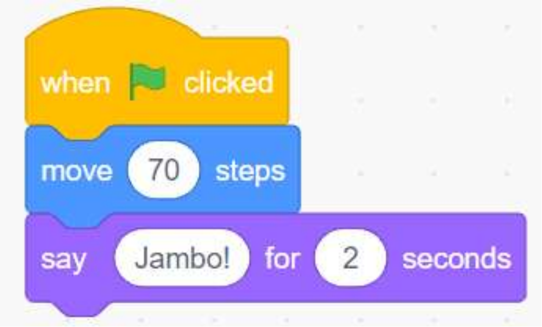
Next, click on the Looks category ● Looks

Drag and drop on the Script Area the command that says *Hello* for two seconds. Replace the word ‘Hello!’ with ‘Jambo!’



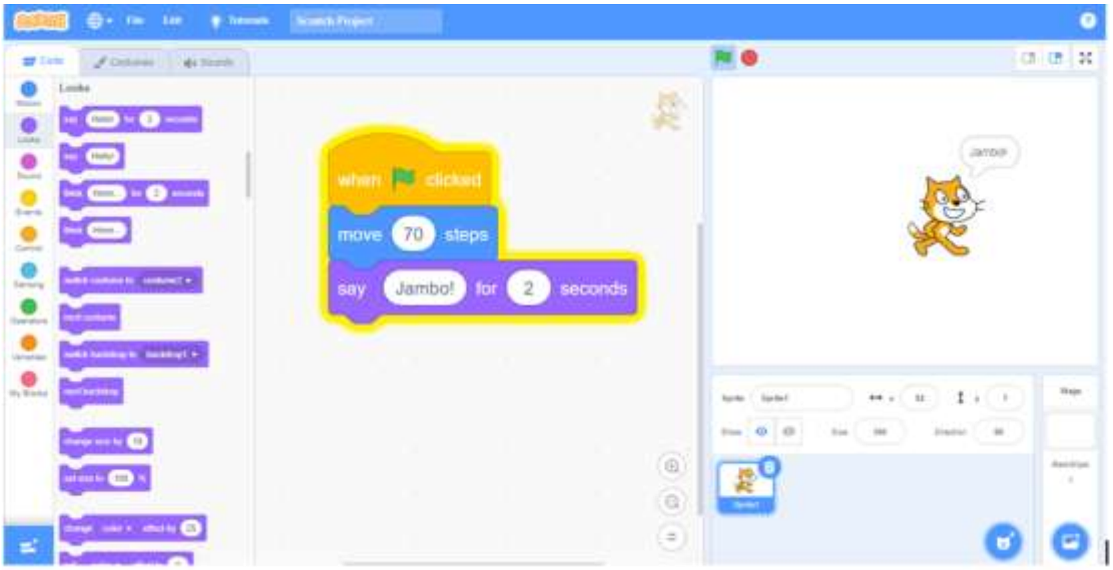
```
say Jambo! for 2 seconds
```

Attach the block to the previous two block commands as shown below



```
when ⚑ clicked
move 70 steps
say Jambo! for 2 seconds
```

Click the Green Flag ⚑ on the top left-hand side of the screen and observe what happens to the cat sprite.

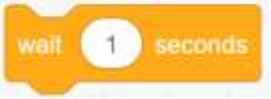


```
when 🏳 clicked
move 70 steps
say Jambo! for 2 seconds
```

In order to create the illusion of the cat walking and to increase the sprite's movement we will use the *‘wait’* block.

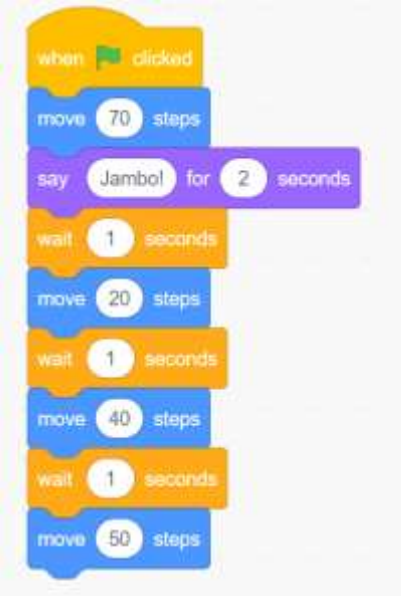
Click on the Events categor ● Events

Drag and drop the *‘wait’* code block and attach to existing code block



```
wait 1 seconds
```

Drag and drop three more *‘move’* code blocks and adjust the number of steps to 20, 40, and 50 as shown in the code block below.

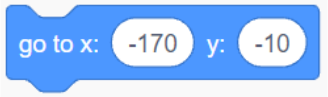


```
when 🏳 clicked
move 70 steps
say Jambo! for 2 seconds
wait 1 seconds
move 20 steps
wait 1 seconds
move 40 steps
wait 1 seconds
move 50 steps
```

After adding more move code blocks, you will notice the cat will keep on moving until it's almost at the edge of the stage. To avoid the sprite disappearing off the stage, we will add one last code block, the ***'go-to'*** code block.

Click on the motion category ● Motion

Drag and drop the ***'go-to'*** code block on the script area



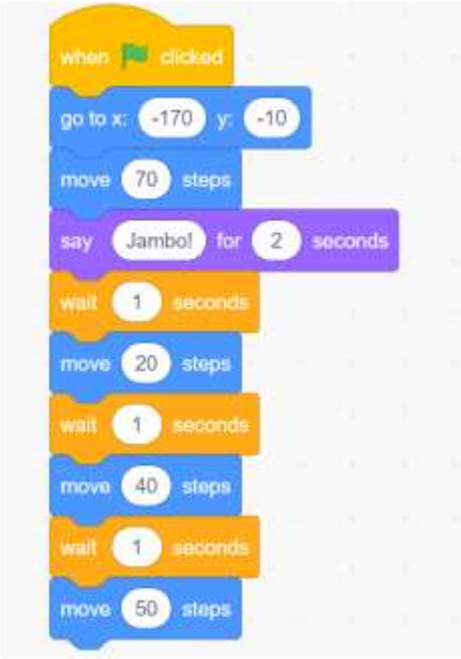
```
go to x: -170 y: -10
```

The code block has different X and Y coordinates depending on the position of 'the cat' on the stage. It allows one to position the sprite at different points on the stage.

The stage is partitioned into two sets of coordinates: the X-axis represents the horizontal position, and the Y-axis represents the vertical position.

Move 'the' cat randomly to different positions on the stage and observe how the X and Y coordinate change on the block palette.

Add the ***'go-to'*** code block after the ***'when-clicked'*** code block to bring the cat back to the point of preference on the stage area.

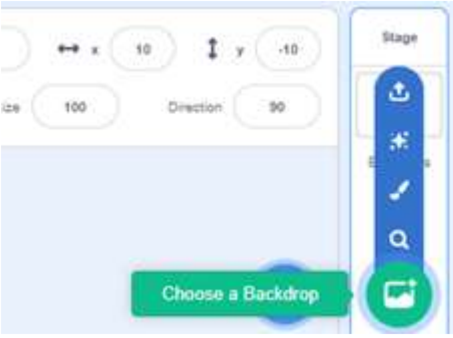


```
when 🏴 clicked
go to x: -170 y: -10
move 70 steps
say Jambo! for 2 seconds
wait 1 seconds
move 20 steps
wait 1 seconds
move 40 steps
wait 1 seconds
move 50 steps
```

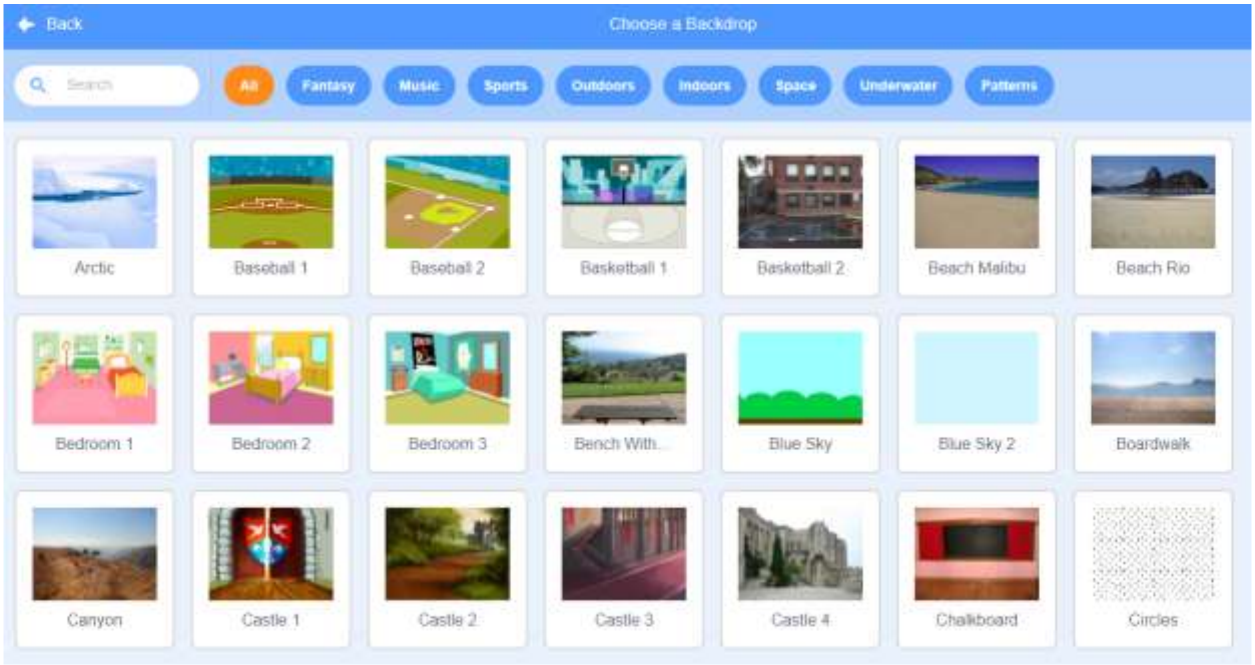
Click the Green Flag on the top left-hand side of the screen and observe what happens to the cat sprite.

Step 2: Change the backdrop and make the cat to turn and Jump

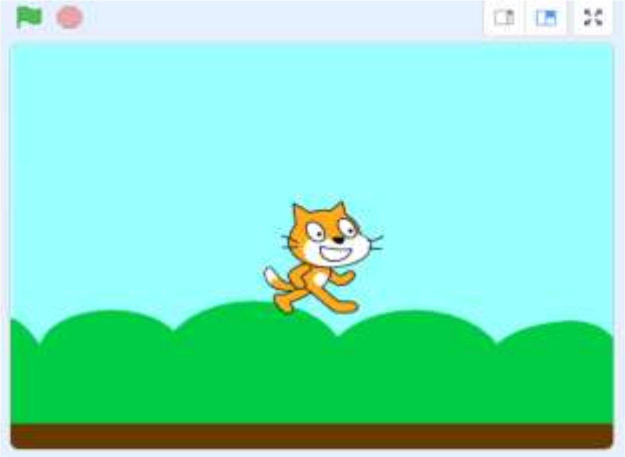
Go to stage at the bottom left and click on the new Backdrops icon located under the Stage icon



Click on the *Blue-Sky* backdrop

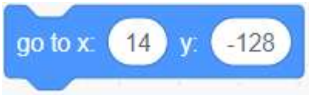


Click on the green flag and observe what happens



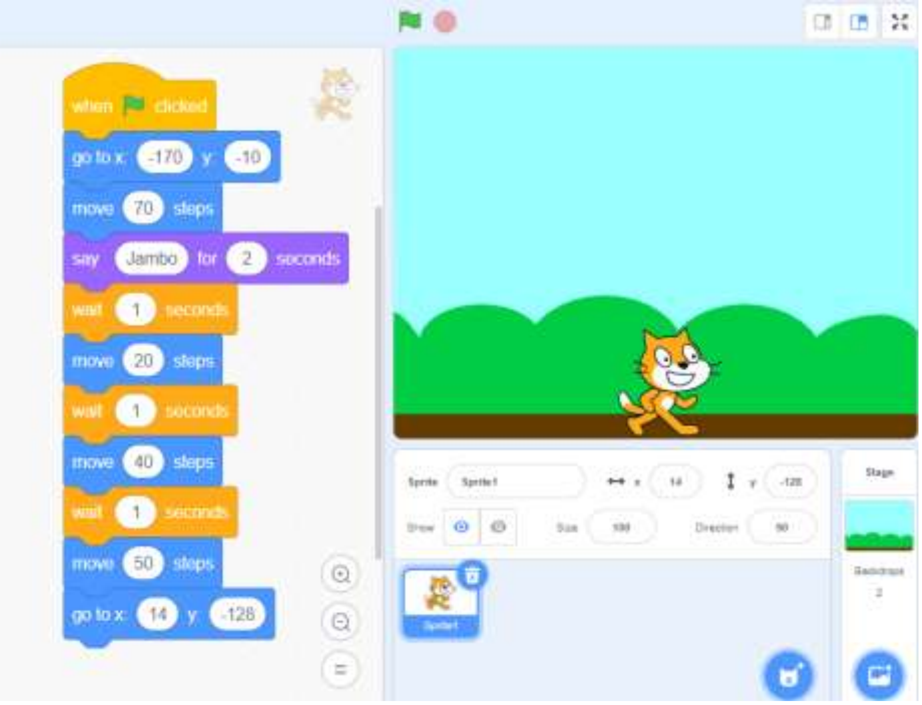
‘The cat’ seems to be floating above the new backdrop.

Drag and drop the ***‘go-to’*** code block on the script area to place the cat on the ground



```
go to x: 14 y: -128
```

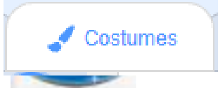
‘The cat’ will now be positioned at the bottom of the backdrop.



```
when 🏴 clicked
go to x: -170 y: -10
move 70 steps
say Jambo for 2 seconds
wait 1 seconds
move 20 steps
wait 1 seconds
move 40 steps
wait 1 seconds
move 50 steps
go to x: 14 y: -128
```

Sprite: Sprite1 | x: 14 | y: -128
Show | Size: 100 | Direction: 90

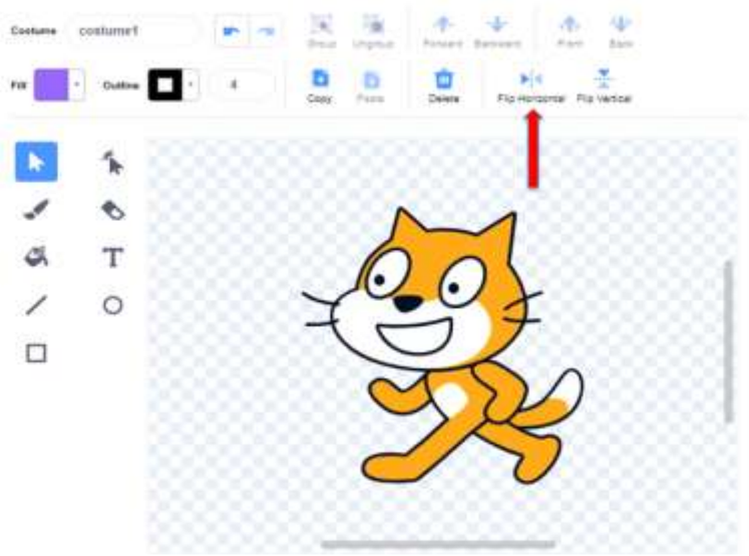
Now let's make ‘the cat’ to turn then jump up and down



Click on the *costume* tab

Click on the *Flip Horizontal* tab

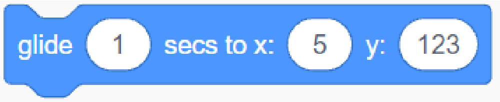
The cat will turn to face the opposite direction as shown below.



Click on the *code* tab to go back to the coding area

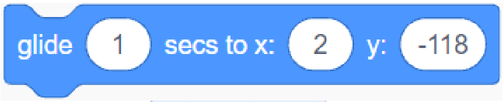
Move 'the cat' sprite at the top of the stage area. The X and Y coordinate will change accordingly as observed in the motion area.

Drag and drop the ***'glide'*** code block on the script area.

An example code would look like this 

```
glide 1 secs to x: 5 y: 123
```

Move 'the cat' sprite at the bottom of the stage area. The X and Y coordinate will change accordingly as observed in the motion area.

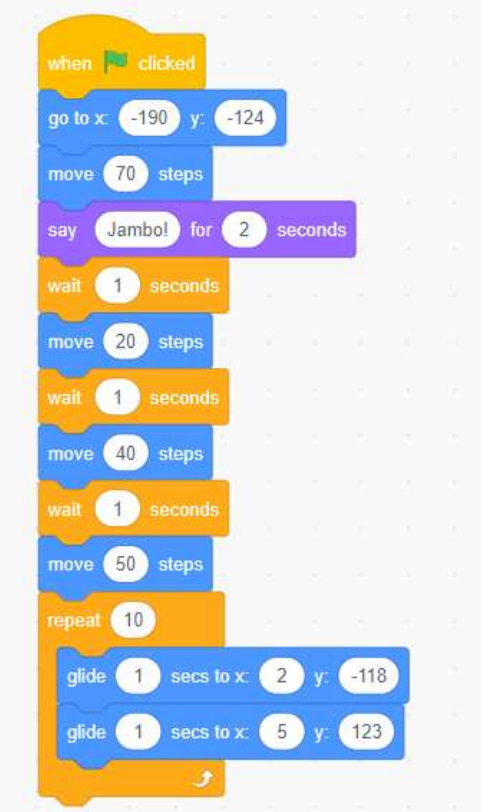
An example code would look like this 

```
glide 1 secs to x: 2 y: -118
```

Note the Y coordinate has changed to a negative. Click on the green flag and observe what happens

To make the cat jump up for a specific number of times, click on the *Control* category

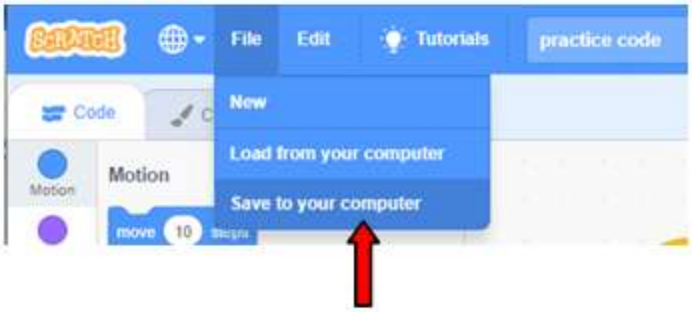
Drag and drop the *‘repeat’* code block to the existing code blocks such that the *‘glide’* code blocks are enclosed as shown.



```
when 🏴 clicked
go to x: -190 y: -124
move 70 steps
say Jambo! for 2 seconds
wait 1 seconds
move 20 steps
wait 1 seconds
move 40 steps
wait 1 seconds
move 50 steps
repeat 10
    glide 1 secs to x: 2 y: -118
    glide 1 secs to x: 5 y: 123
```

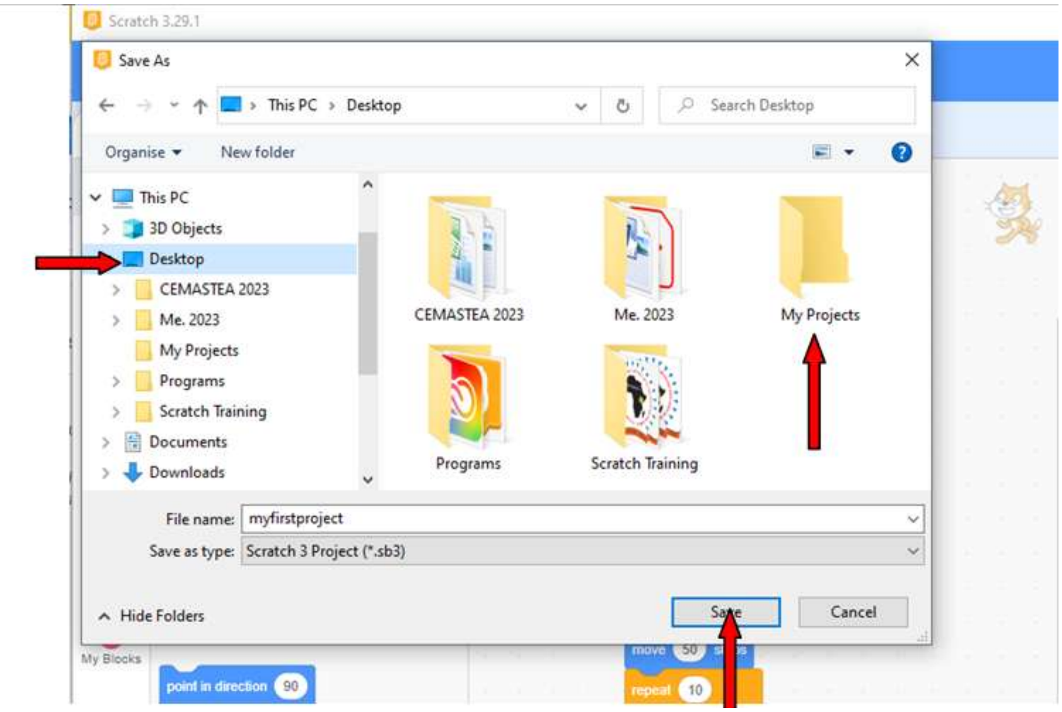
Let’s now save our project. Click on the *file* menu at the top right of the screen

On the drop down menu, click on *Save to your computer*



On the left-hand side of the dialog box, choose the location to save your project in this case, the Desktop.

Click on the Desktop option and create a folder named My Projects.



File name: myfirstproject

Save as type: Scratch 3 Project (*.sb3)

Double click on the folder *My Project* to open.

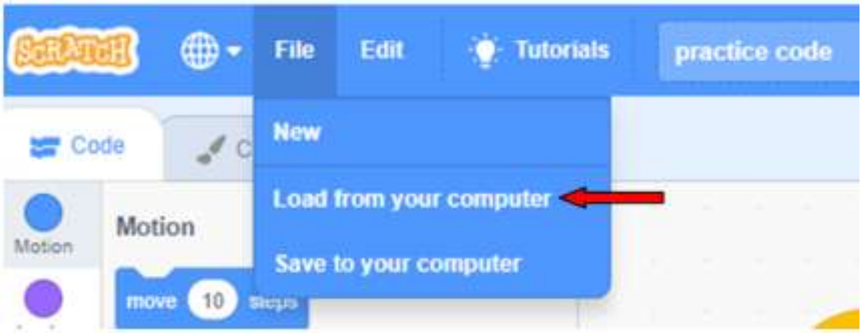
Place the cursor in the *File name* input box and key in the file name *myfirstproject*

Click on the Save button Save

Step 3: Introduce the second sprite- "Avery walking"

Click on the *file* menu at the top right of the screen to open the project we were working on in the previous section *myfirstproject*.

On the drop down menu, click on *Load from your computer*

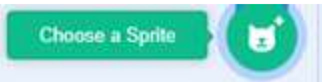


A dialog box will appear, click on the desktop and locate your folder *My Projects*

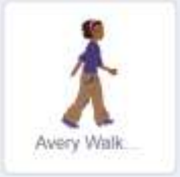
Open

Select your project named *myfirstproject* and click open.

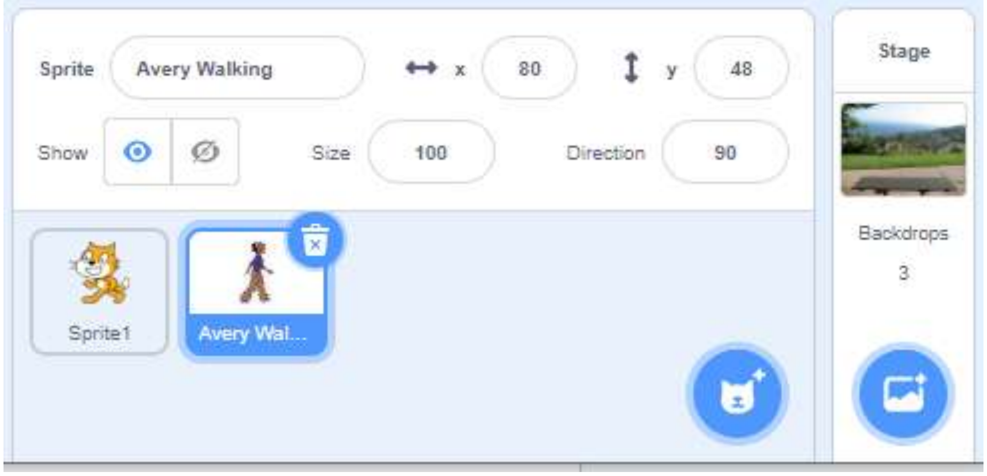
To introduce a new sprite, click on the Choose a Sprite icon



Click on the ‘Avery Walking’ Sprite



The ‘Avery Walking’ icon is highlighted in blue, showing that the Sprite is active.



Sprite Avery Walking x 80 y 48

Show Size 100 Direction 90

Sprite1 Avery Wal...

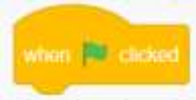
Stage

Backdrops 3

Let us now code ‘the cat’ to take a walk with ‘Avery Walking’ as they have a chat.

On the block section click on Events category

Drag and drop the ***‘when- clicked’*** block code in the Script area.

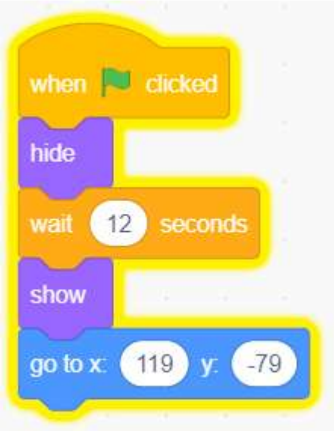


On the Looks category, we have the ‘***show’*** and ‘***hide’*** code blocks.

These code blocks will hide ‘Avery Walking’ so that she appears on the stage after ‘the cat’ has completed its jumps.

Add a *‘wait’* code block to input the number of seconds ‘Avery Walking’ will wait before appearing. The number of seconds depends on how long ‘the cat’ code program takes.

The code will look something like this:



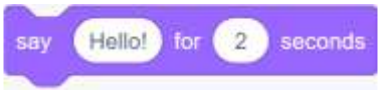
```
when ⚑ clicked
hide
wait 12 seconds
show
go to x: 119 y: -79
```

Activity

1. ‘Avery Walking’ will appear as if she is floating, add the X & Y coordinate block to place her on the ground.
2. Turn ‘Avery Walking’ to face the same direction as the cat.

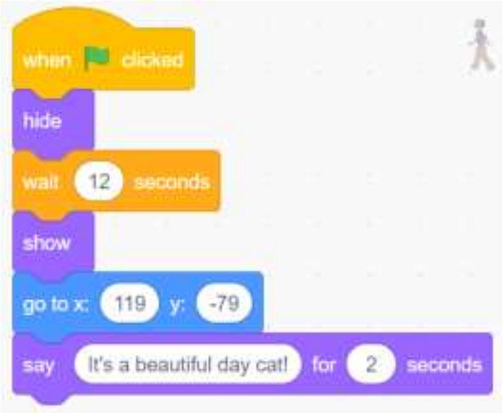
Now let's get Avery to say “*it’s a beautiful day cat!*’

While still on code for ‘Avery Walking’, click on the Looks category.

Drag and drop the code block 

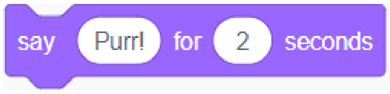
```
say Hello! for 2 seconds
```

Click in the input box and change from ‘Hello!’ To ‘It's a beautiful day cat!’

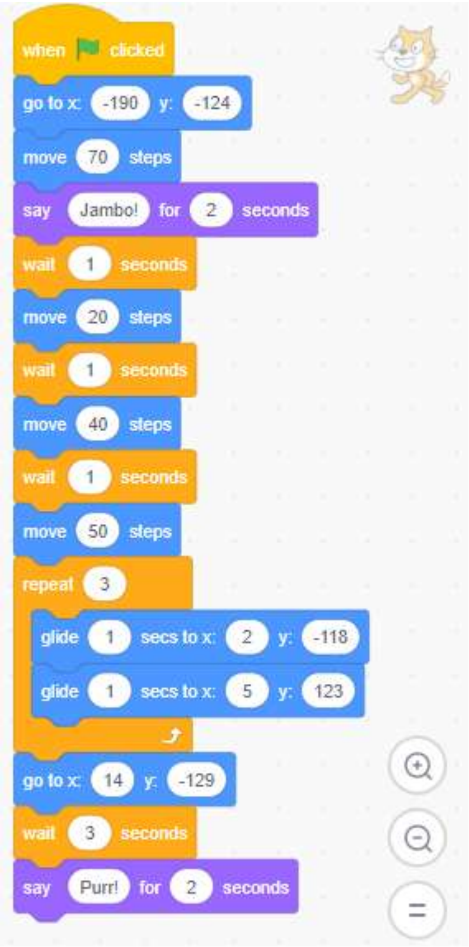


```
when ⚑ clicked
hide
wait 12 seconds
show
go to x: 119 y: -79
say It's a beautiful day cat! for 2 seconds
```

Click the Green Flag on the top left-hand side of the screen and observe what happens.

Add a code where ‘the cat’ will respond with a ‘purr!’ 

```
say Purr! for 2 seconds
```

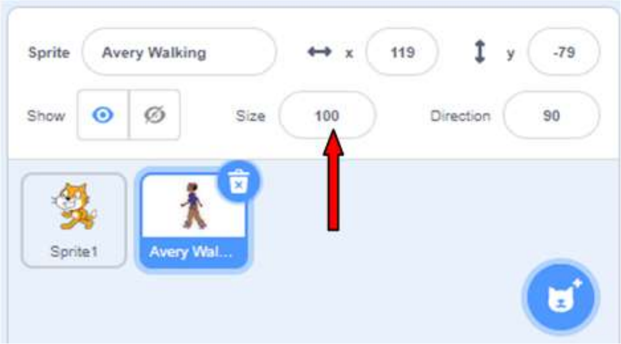


```
when [flag] clicked
go to x: -190 y: -124
move 70 steps
say Jambo! for 2 seconds
wait 1 seconds
move 20 steps
wait 1 seconds
move 40 steps
wait 1 seconds
move 50 steps
repeat 3
    glide 1 secs to x: 2 y: -118
    glide 1 secs to x: 5 y: 123
go to x: 14 y: -129
wait 3 seconds
say Purr! for 2 seconds
```

Step 4: Turn and increase the size of 'Avery Walking'. Make 'Avery Walking' with 'the cat' take a walk as they chat.

On the Sprite List section, click on the 'Avery Walking' icon to activate it

Click on the Size input box. Change from 100 to 120. Observe what happens.



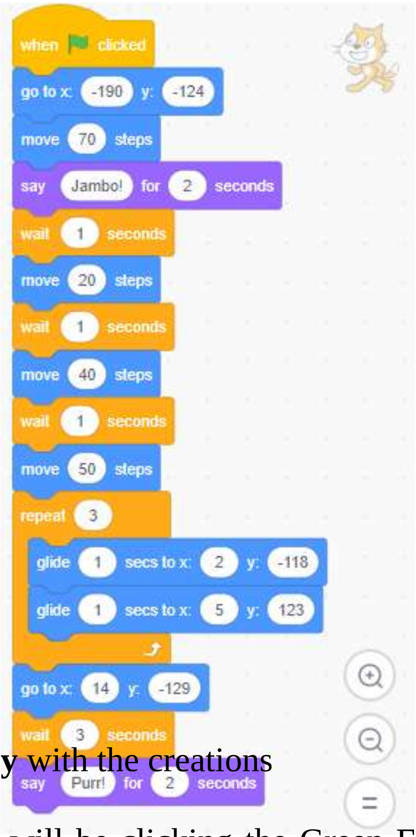
Sprite: Avery Walking   ↔ x: 119   ↕ y: -79

Show   Size: 100   Direction: 90

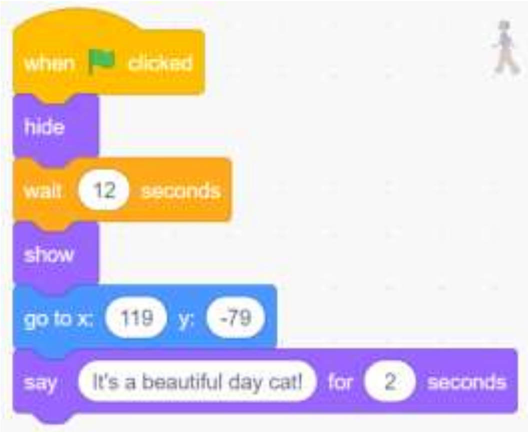
Activity

Build up ‘Avery Walking’ and ‘the cat’ code blocks to make them walk as they talk.

Final code for Sprite 1: CAT and ‘AVERY WALKING’



```
when 🏴 clicked
go to x: -190 y: -124
move 70 steps
say Jambo! for 2 seconds
wait 1 seconds
move 20 steps
wait 1 seconds
move 40 steps
wait 1 seconds
move 50 steps
repeat 3
    glide 1 secs to x: 2 y: -118
    glide 1 secs to x: 5 y: 123
go to x: 14 y: -129
wait 3 seconds
say Purr! for 2 seconds
```



```
when 🏴 clicked
hide
wait 12 seconds
show
go to x: 119 y: -79
say It's a beautiful day cat! for 2 seconds
```

Play with the creations

We will be clicking the Green Flag on the top left-hand side of the screen and observe what happens to the animation we have created.

It involves a process of trial and error. Explore and test boundaries of animation by;

- Change the sprites and backdrop
- Make the sprite glide in different directions
- Add a third sprite and animate it as you like
- Continue the conversation between Avery and the cat.
- Trying out different codes blocks →

Share the ideas and creations with others

We will share our ideas on;

- The changes we have made to the sprites and backdrop
- The reason making the changes
- Additional code blocks that you might have used.

Reflect on the experiences of animating.

Imagine new animation ideas and project to do

We will come up with new animation ideas by drafting a storyboard

Project task

Create a storyboarding on any of the following Pertinent and Contemporary Issues (PCIs) for animation:

1. Climate change
2. Affordable housing
3. Care for the environment and animals

Or any other of your choice

Additional Learning Resources

The following two links can be used to support learners on the CBC sub-strands on care for the environment and animals.

- Tree simulator: https://youtu.be/St244bVXC-c
- How to make a virtual pet: https://youtu.be/irhNLRWwhv0

The following are introductory links to animation through scratch programming.

- Create animation with scratch https://youtu.be/1GmaQAQvgPc
- How to make animation in scratch programming: https://youtu.be/k4zMuBf-7Vs
- How to make a jumping game in scratch: https://youtu.be/1jHvXakt1qw

Section 5: Games and Graphics Using Scratch

Introduction

Scratch games promotes learner's computational thinking, creativity, and problem solving skills. Further, it promotes communication and collaboration as learners work in groups. Learners can use Scratch to create their own stories, games, and animations, and share with others. Scratch introduces the learners programming.

Scratch enhances learner's high order thinking skills as they come up with creative solutions to problems. Coding enables learners to figure out how to break down complex ideas into smaller manageable parts. As learners make games, animations, or stories, they have to figure out how various parts of their coding can work together. This develops their problem-solving skills that can be applied to other areas of their daily life.

Imagine

We are going to create a maze game using scratch application. First we will sketch on a piece of paper a maze with a clear path from start to exit. We will use the sketch to draw the maze on a canvas on scratch application with the help of your mouse pointer.

We will code two sprites; a Sprite 1 the cat and a packet of milk. The goal is to make the 'cat' sprite move along a path in the maze till it reaches the 'milk' sprite. For this game we wanted the 'Milk' to make a sound when 'Sprite 1' touches 'milk'. The code block to use will include the Motion, Looks, Sound, Control, Sensing and Events.

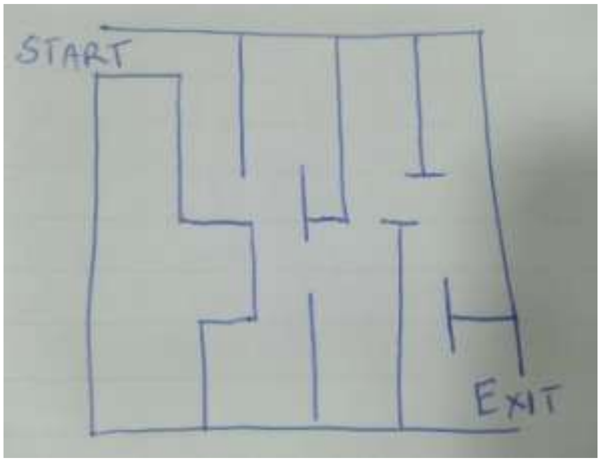
The sensing blocks will be used to tell or highlight how sprites interact with their surroundings. The broadcast block will broadcast messages to let one part of the code or project communicate with another part. An if-then block code will be used to ensure that the sprite only uses the designated path to the milk. A forever code block will be used to repeat the sequence of commands to ensure the continuity of the game.

Create

Let's now create the maze and code the maze game.

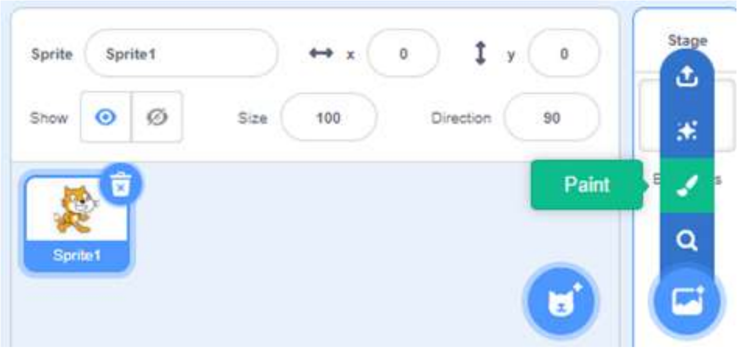
1. **How create the maze**

On a piece of paper sketch a maze with a clear path from start to exit.

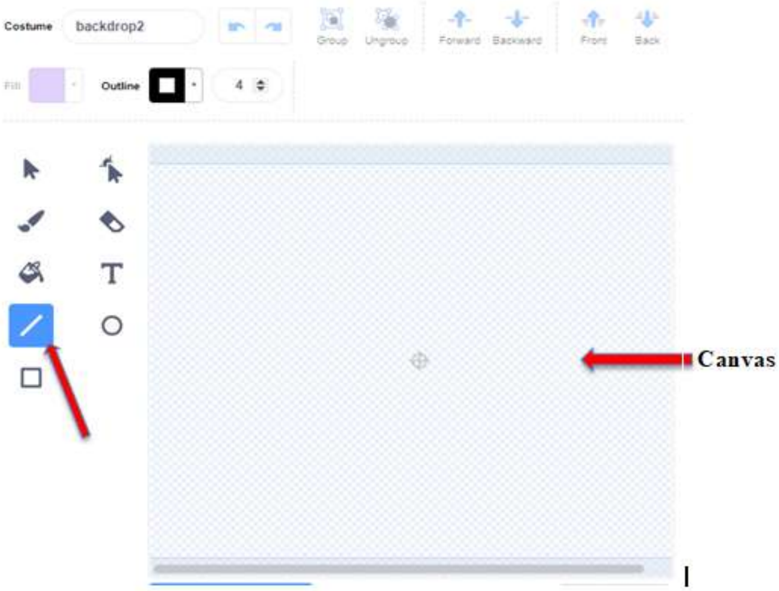


Open the scratch application

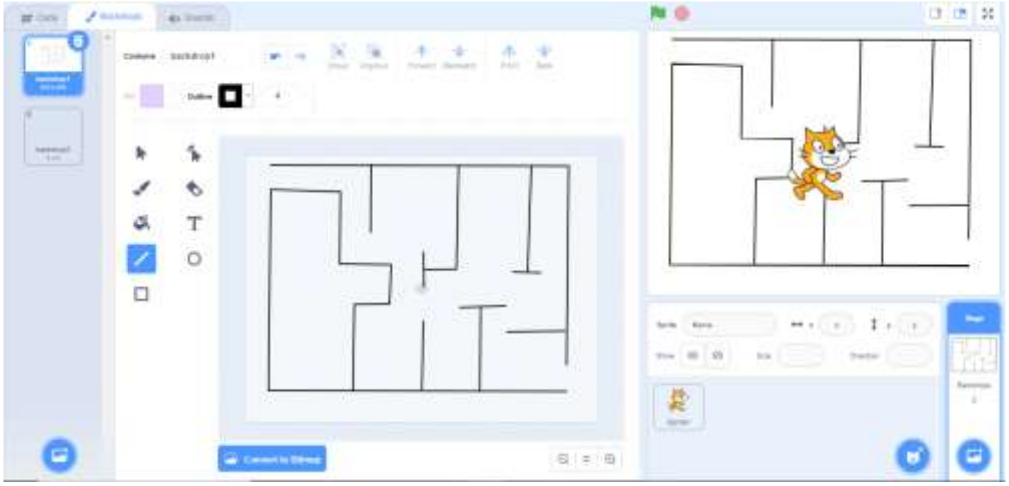
On the bottom right corner click on the paintbrush on the pop up menu.


Sprite: Sprite1 x: 0 y: 0
Show Size: 100 Direction: 90

Select the line icon highlighted in blue


Costume: backdrop2
Fill Outline 4

Draw your maze on the canvas with the help of your mouse pointer. As you draw, your maze will appear as a backdrop

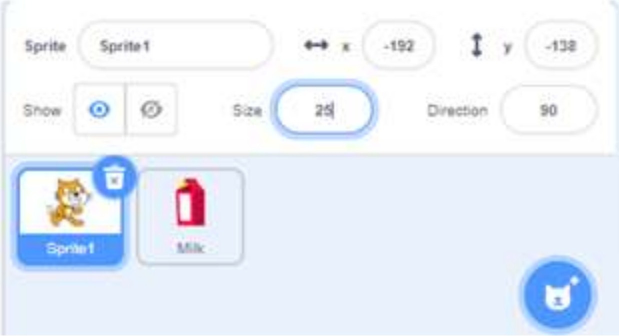


2. How to create a maze game

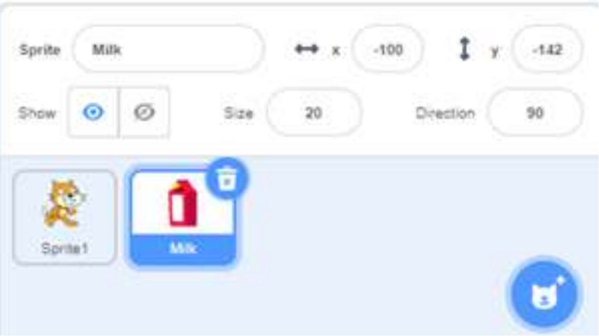
Click on the code button to start creating the game

Remember in Scratch the default sprite is a cat. Choose the second sprite ‘Milk’

Change the size of the Sprite 1 to 25 and Milk 20



Sprite: Sprite1 x: -192 y: -138
Show Size: 25 Direction: 90



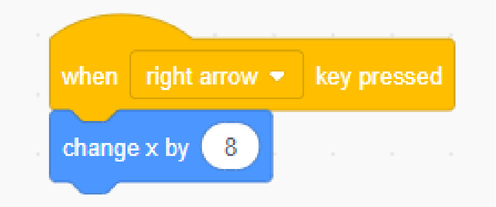
Sprite: Milk x: -100 y: -142
Show Size: 20 Direction: 90

Coding Sprite 1: the cat

Click on ‘the cat’ Sprite

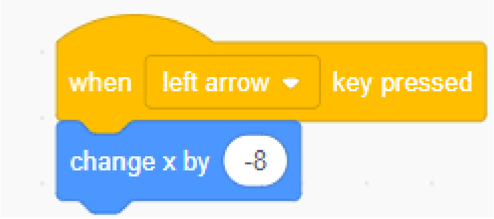
Change the sprites position using x and y coordinates for the sprite to move in the different directions

a. Right arrow



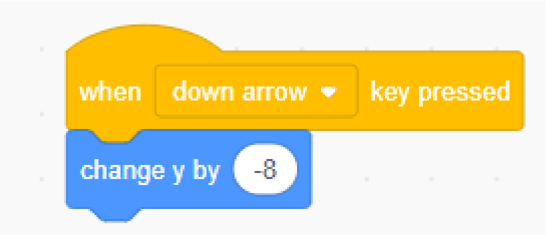
```
when right arrow key pressed
change x by 8
```

b. Left arrow



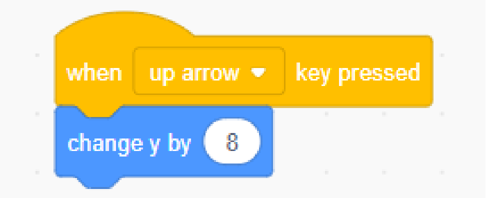
```
when left arrow key pressed
change x by -8
```

c. Down arrow



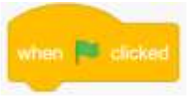
```
when down arrow key pressed
change y by -8
```

d. Up arrow



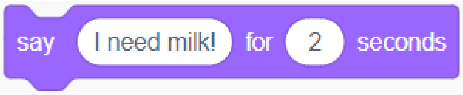
```
when up arrow key pressed
change y by 8
```

Drag and drop the following block into the Script Area.



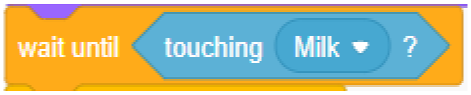
```
when 🏁 clicked
```

Drag and drop on the Script Area the command that says *Hello* for two seconds. Replace the word 'Hello!' with 'I need milk!'



```
say I need milk! for 2 seconds
```

Add the ***'wait until- touching'*** code Block



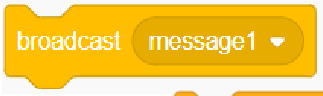
```
wait until touching Milk ?
```

This code gives a condition, ‘the cat’ only goes on after it touches ‘Milk’

Click on the Control category

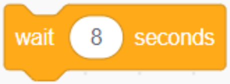
Control

Drag and drop the *‘broadcast’* code block



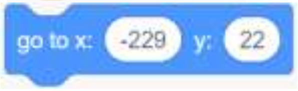
```
broadcast message1 ▾
```

Add the *‘wait’* code block and change it to 8 seconds



```
wait 8 seconds
```

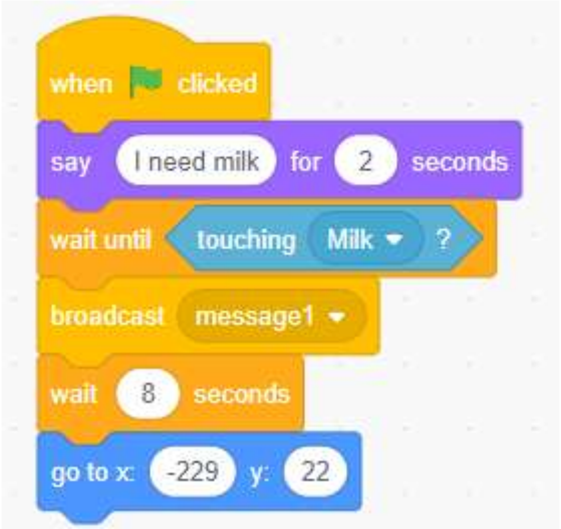
Drag and drop the *‘go-to’* code block on the script area



```
go to x: -229 y: 22
```

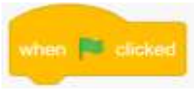
Click the Green Flag on the top left-hand side of the screen and observe what happens.

Activity: Can you tell what ‘the cat’ is expected to do from the following code?



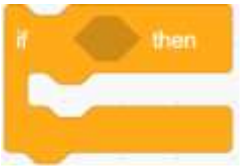
```
when 🏁 clicked
say I need milk for 2 seconds
wait until touching Milk ▾ ?
broadcast message1 ▾
wait 8 seconds
go to x: -229 y: 22
```

Drag and drop a second *‘when- clicked’* block into the Script Area



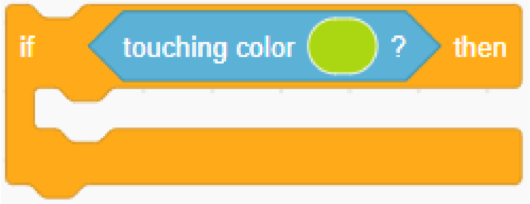
```
when 🏁 clicked
```

Add the *‘if-then’* code block



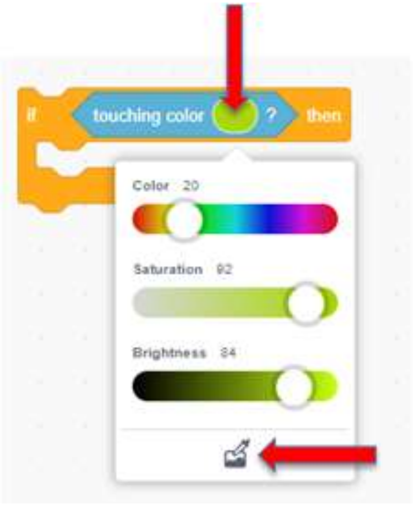
```
if ⬡ then
```

Drag and drop the ‘*touching color’* code block on the *‘if-then’* code block

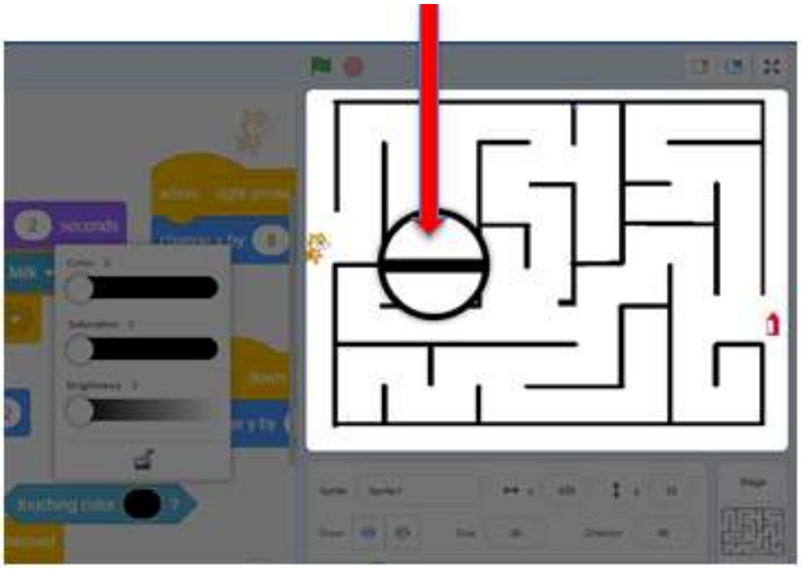


```
if touching color ● ? then
```

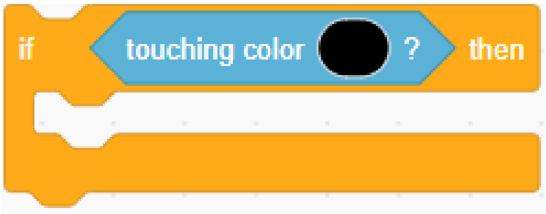
Click on the color, a drop down menu will appear.



At the bottom of the color code, click on the paint icon.

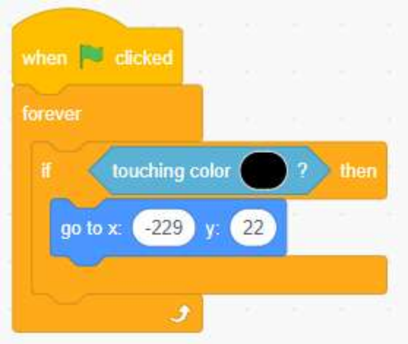


Move the mouse cursor on the maze. A magnifying glass will appear, click on any line of the maze. The color on the code will change to reflect the color of the maze.



Drag and drop the ‘*go to x: y’ code* block inside the *‘if-then’* code block. This will take the cat back to the starting point when it touches the boundaries/line of the maze.

Drag and drop the *‘forever’* code block to encapsulate the *‘if-then’* code block

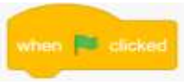


```
when ⚑ clicked
forever
    if touching color ● ? then
        go to x: -229 y: 22
```

Click the Green Flag on the top left-hand side of the screen and observe what happens.

Let's add one more condition to control the use of a mouse pointer.

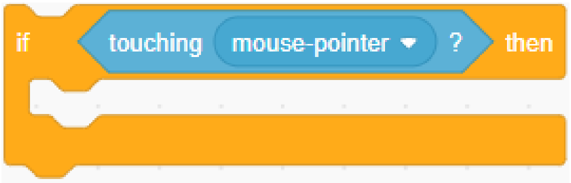
Drag and drop a third *‘when- clicked’* block into the Script Area



```
when ⚑ clicked
```

Add the *‘if-then’* code block

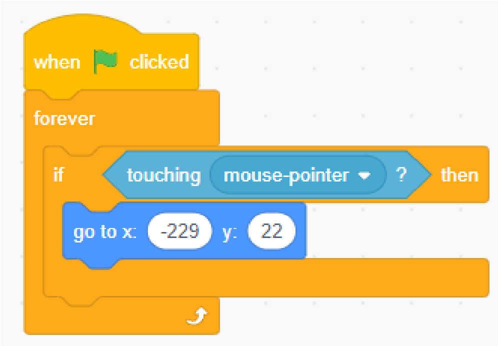
Drag and drop the *‘touching mouse-pointer’* code block on the *‘if-then’* code block



```
if touching mouse-pointer ▾ ? then
```

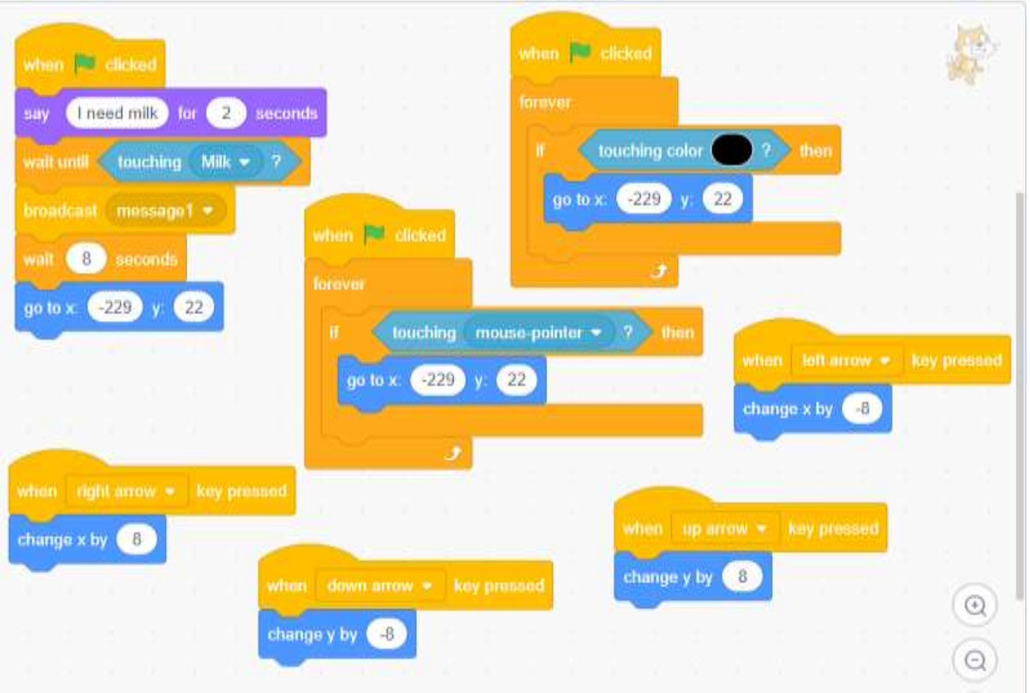
Drag and drop the ‘*go to x: y’* code block inside the *‘if-then’* code block. This will restrict moving ‘the cat’ using a mouse pointer.

Insert the *‘forever’* code block inside the *‘if-then’* code block



```
when 🏴 clicked
forever
    if <touching (mouse-pointer ▾) ?> then
        go to x: (-229) y: (22)
```

Final code should look something like this:



```
when 🏴 clicked
say (I need milk) for (2) seconds
wait until <touching (Milk ▾) ?>
broadcast (message1 ▾)
wait (8) seconds
go to x: (-229) y: (22)
```

```
when 🏴 clicked
forever
    if <touching color (●) ?> then
        go to x: (-229) y: (22)
```

```
when 🏴 clicked
forever
    if <touching (mouse-pointer ▾) ?> then
        go to x: (-229) y: (22)
```

```
when (left arrow ▾) key pressed
change x by (-8)
```

```
when (right arrow ▾) key pressed
change x by (8)
```

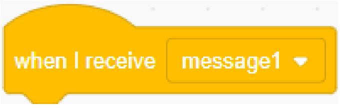
```
when (up arrow ▾) key pressed
change y by (8)
```

```
when (down arrow ▾) key pressed
change y by (-8)
```

Coding ‘Milk’ sprite

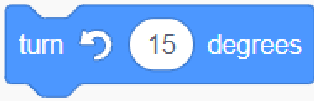
Click on the ‘Milk’ sprite to highlight and code it.

Drag and drop a second ***‘when I receive’*** code block into the Script Area



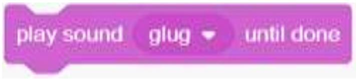
```
when I receive (message1 ▾)
```

Attach an anticlockwise ***‘turn degrees’*** code block

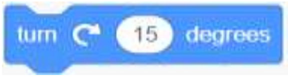


```
turn ↺ (15) degrees
```

Click on the sound category ● Sound

Drag and drop the *play sound-until done* code block 

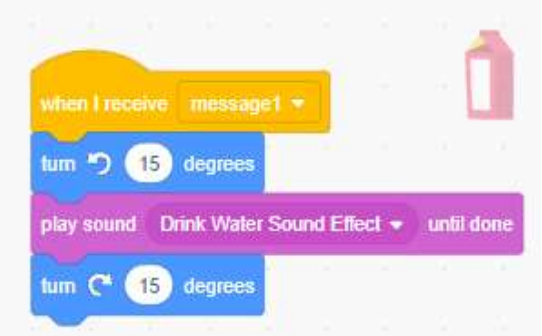
```
play sound glug until done
```

Attach a clockwise ***'turn degrees'*** code block 

```
turn ⟳ 15 degrees
```

Click the Green Flag on the top left-hand side of the screen and observe what happens.

Final code should look something like this:



```
when I receive message1
turn ⟲ 15 degrees
play sound Drink Water Sound Effect until done
turn ⟳ 15 degrees
```

Now let's **Play** with the game, explore and test the boundaries of the game.

Share the ideas about the game with others;

- What was good about the game
- The challenges faced when coding
- Areas of improvement

Reflect on the experiences of coding a game in scratch.

Project task

Imagine new game ideas you can create. Draft the ideas in a storyboard then create a game on any of the Pertinent and Contemporary Issues (PCIs)

Suggestions:

- Equality Puzzle: Design a puzzle game that promotes diversity and inclusion.
- Food Waste Challenge: Create a game that educates players about the issue of food waste.

- Cybersecurity Defender: Design a game where players take on the role of a cybersecurity expert defending a virtual network from cyber threats.
- Climate Challenge: Create a game where players must navigate a world affected by climate change.

Scratch Project

In this section we are going to discuss how to use scratch to find a solution to a problem in your local environment.

Example

A primary school is located along a busy highway. Learners cross this road to access the school in the morning and in the evening while going home. The problem is how to guarantee the safety of learners while crossing the busy highway.

Solution

Safety rules/ steps that learners can follow while crossing the road.

Steps/ rules that learners can follow:

a. Move to the road edge and stop
b. Check left, right and left again before crossing
c. Look around while crossing to spot any oncoming car or motorbike
d. If the road is clear, then cross.

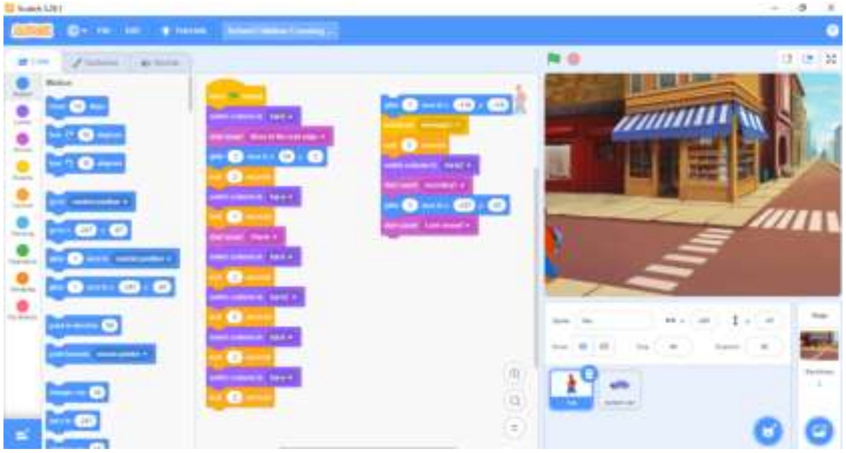
Trial code to solve the problem using paper

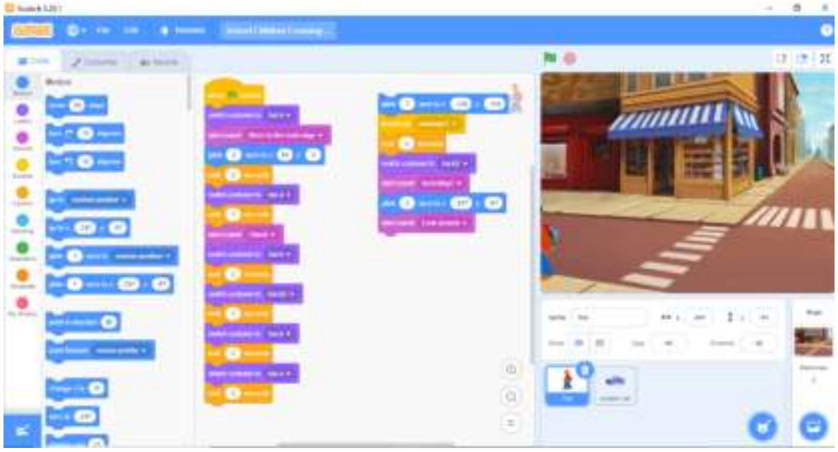
Manila papers can be used to form blocks labeled with instructions that will be used to solve the problem. The instructions need to be arranged in a logical manner as follows:

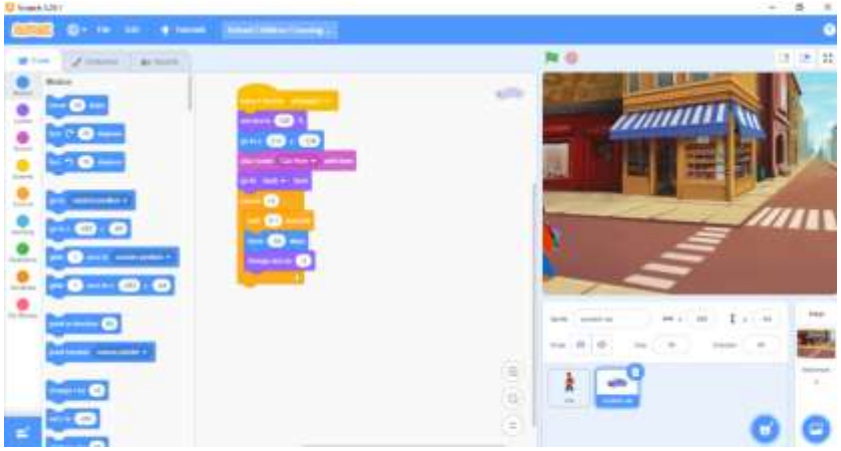
MOVE TO THE ROAD	STOP AT THE ROAD EDGE	CHECK LEFT
CHECK RIGHT	CHECK LEFT AGAIN	MOVE TO CROSS

Solving the problem using scratch

The paper blocks with instructions to solve the problem can be used to form code blocks on the scratch. The codes should have atleast one block from the *motion, looks, sound, events and control* button. The code blocks should be dragged to the code area and arranged in a logical manner as follows:







Activity 1

a) In groups discuss and name one problem found in your environment.

b) In your note books, write, draw, and describe how you will solve the problem.

c) Using Manila paper and scissors cut out all the blocks needed to solve your problem.

d) Label them with the instructions you will use to solve your problem.

e) Arrange them in a logical manner.

f) Draw your arrangement in your notebook

Activity 2

Solving your problem with scratch

In the previous activity, you identified a problem and came up with a solution, you then drew code blocks needed to solve the problem.

a) Using your drawings, find the code blocks on scratch and drag them to the code area.

b) Your code should have at least one block from the Motion, looks, sound, events and control button

Arrange your code blocks in a logical way and make any changes you need to make.

Conclusion

The teaching and learning of coding undoubtedly promote several CBC fundamental skills, such as problem-solving, communication, and collaboration. Coding encourages student cooperation, creativity, and problem-solving abilities. Furthermore, coding has demonstrated efficacy in teaching fundamental programming ideas as well as promoting computational thinking nd learning.In this module the focus was on developing coding skills uisng Scratch application.

Scratch is a programming language developed by the Lifelong Kindergarten Group at the MIT Media Lab. It is designed to teach children the basics of programming through a visual interface. Scratch allows users to create interactive stories, games, and animations by dragging and dropping blocks of code. For teachers in Grade 5, we have seen how Scratch can be a valuable tool for introducing students to the world of coding. By using Scratch, teachers can engage students in a fun and interactive way, while also teaching them important coding concepts. You are encouraged to join https://scratch.mit.edu/ and be part of the online community on the scratch platform to showcase your learner's projects as well as learn from others in the global community.

We hope that this module had enabled you to improve your efficacy to teach codingto your learnrs.

References

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. Communications of the ACM, 52(11), 60-67.

Balanskat, Anja, and Katja Engelhardt. 2014. *Computing Our Future: Computer Programming and Coding-Priorities, School Curricula and Initiatives across Europe*. European Schoolnet.

World bank 2019. Chidren learning to code: Essential for 21st Century Human Capital