Supporting Coding through the Competence-Based Curriculum & After-School Clubs

Scratch Pedagogical Guide

January 2021

REB Rwanda Education Board

vvob education for development

Rwanda Coding Academy

WEHUBIT

This project is funded by Belgium through the Wehubit programme implemented by Enabel

Supporting Coding through the Competence-Based Curriculum & After-School Clubs

Scratch Pedagogical Guide

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

Animation: An animation is a sequence of images in the motion of objects to create a video. On Scratch, users can make short movies, music videos, comical shorts, and more through a variety of techniques.

Backdrop: A backdrop is a background of a scratch project. It is like a costume, except that it is shown on the stage instead. They are in the backdrop's library. The Stage can change its look to any of its backdrops using the Switch Backdrop to () block.

Block Palette: The block palette is where the different script blocks are located. The different types of script blocks include motion, control, looks, sensing, sound, operators, pen, and variables.

Block: Blocks are puzzle-piece shapes that are used to create code in Scratch. The blocks connect to each other vertically like a jigsaw puzzle, where each data type (hat, stack, reporter, boolean, or cap) has its own shape, and a specially shaped slot for it to be inserted into, which prevents coding errors. There are ten categories of blocks: Motion, Looks, Sound, Event, Control, Sensing, Operators, Variables, List, and My Blocks.

Boolean: Booleans are conditions that can either be true or false. They are often indicated by the numbers 1 (True) or 0 (False).

Broadcast: A broadcast is a message that is sent through Scratch, activating scripts with matching hat blocks. Broadcasts are useful in games and animations, as they trigger the execution of specific scripts.

Cap block: A cap block is a block that is designed to stop a block from being placed underneath it. It looks like a Stack Block, except there is no bump underneath it and the bottom is flat.

Coding: Coding is the process of using programming language to get the computer do as desired. Each line of the code is a set of instructions for the computer.

Computer program: Is a collection of instructions that can be executed by a computer to perform a specific task.

Concurrency: Concurrency is an idea in computer science in which multiple computations are being performed at the same time. Scratch allows many scripts to run at the same time, though only one block can be busy at any one point.

Control blocks: Control blocks are one of the ten categories of Scratch blocks. They are color-coded gold and are used to control scripts.

Debug: A complex program is made up of lots of lines of code and it is normal for new programs to have some bugs (mistakes). An important part of programming is testing your program and 'debugging' (which means removing the bugs). Debugging is running your code line per line to see at which line the mistake occurs.

Event Block Palette: Event blocks are one of the ten categories of Scratch blocks. They are color-coded light yellow and are used to sense events, which trigger scripts to run. The block palette is where the different script blocks are located.

Event-driven programming: Programming in which the code is based on events. For example, a "when mouse moved" event can trigger all scripts when the mouse is moved. Events have their own attributes, called event attributes.

Execute instructions: Starting a computer program.

Hat block (Hat): a block that starts a script when a specific event occurs. Hats can be Control blocks, Events blocks or More Blocks. Hat blocks are useful in Event Based Programming.

Iteration: Repeating a set of commands.

Looks blocks are one of the ten categories of Scratch blocks. They are color-coded purple and are used to control a Sprite's appearance**.**

Loops: In Scratch, it is called a repeat loop. It is a program cause a block of code to be executed a fixed number of times when the user clicks the green flag.

Operators: An operator in a programming language is a symbol that tells the interpreter to perform specific mathematical, relational or logical operation and produce final results.

Parallelism: Parallelism is when actions occur at the same time. When scripts in Scratch are executed at the same time.

Pixel 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. An image on a computer display consists of tiny dots. 1 dot is a pixel.

Programming: In Scratch, programming is more commonly referred to as "scripting" because a script is a stack or combination of blocks. The blocks are the code of the project, meaning they tell it what to do.

Reporter block: A block that reports a value. These can be anything, from numbers to strings. Unlike a Stack block, reporter blocks cannot be placed directly above or below another block.

Operating System (OS): System software that manages the computer hardware, software, and provides common services for computer programs such as…. Examples are Windows, iOS, Ubuntu, Android.

Script or Script Block: A set of instructions made to automate computer tasks. A script is a collection or stack of blocks that all interlock with one another. The blocks and their order are very important, as they determine how sprites interact with each other and the backdrop.

Scripts Area: The area on the right side of the project editor where scripts are built. Blocks are dragged from the block palette to the scripts area, so the script area shows a stack of connected blocks.

Sensing blocks: Sensing blocks are one of the ten categories of Scratch blocks. They are color-coded light-blue and are used to detect things. They can be used to determine the location of the mouse-pointer, its distance from other sprites, and whether a sprite is touching another sprite.

Sound blocks are color-coded pink and are used to control sounds.

Sprite: An image on a Scratch computer program screen. Every Scratch program is made up of sprites and scripts that control them. Scripts are programmed to make sprites do things.

Stage: The stage is found on the right in Scratch. The stage is the background of the project, but can have scripts, backdrops (costumes), and sounds, like a sprite.

String: A variable representing a sequence of letters and / or numbers.

Talk bubble: An image of a bubble representing the speech or thoughts of a Sprite.

Variable: A changeable value recorded in Scratch's memory. Variables can only hold one value at a time, unlike lists. These values can be either integer (numbers) or strings (letters).

INTRODUCTION

This guide is for secondary Science, Technology, Engineering and Mathematics (STEM) and ICT teachers who are interested to learn about Scratch and coding in general. It focuses on the competences needed to integrate Scratch in STEM and ICT lessons and initiate after-school Scratch coding clubs in secondary schools. In these coding clubs, secondary school learners will learn about coding and Scratch in a social and fun way.

Quality education provides all learners with the capacities to become economically productive, develop sustainable livelihoods, contribute to peaceful societies and enhance individual wellbeing. In 2016, Rwanda has introduced a competency-based curriculum (CBC) with the purpose to shift the focus of instruction from knowledge to competences. Competences mean the application of knowledge and skills in daily life. The CBC puts emphasis on crosscutting competences such as problem-solving, collaboration, self-regulation, communication and creativity. Learning how to code will help learners to develop all these skills.

The Rwandan government has formulated its Vision 2050. In this vision, the Government has expressed the ambition to transform Rwanda to a high-income country based on services (instead of agriculture and industry). A key condition to achieve this vision is quality education, in particular in the fields of STEM and ICT. Competences in STEM and ICT will enable students to find fulfilling work or start their own business.

However, important challenges remain. One challenge is that many girls (and their parents) and teachers still think that STEM and ICT are for boys, and that girls are more suitable for humanities or caring jobs. Although there are currently more girls enrolled in secondary education than boys, boys continue to outperform girls. This is particularly true in ICT and STEM. As a result, many girls in Rwanda hesitate to choose ICT- and STEM-related fields as they come to believe that these subjects are more suitable for boys.

Scratch is being used by educators all around the world. From pre-primary to higher education and beyond, across subject areas, educators of all backgrounds and experience are helping learners engage in creative computing with Scratch. Our goal is to help teachers connect, share, and learn from each other—and to help learners who are new to Scratch join the fun.

LEARNING OUTCOMES

At the end of this learning trajectory, participants will be able to:

- Understand key principles of coding.
- Develop the concept of computational creation through Scratch programming.
- Use different concepts and features of Scratch to create animated stories.
- Create stories and games with Scratch.

- Integrate Scratch into ICT/STEM lesson plans for student-centred and competency-based lessons.
- Develop communication, critical thinking, problem solving, collaboration, and creativity skills with their learners.

PREPARATORY UNIT

0.1 OVERVIEW

During this learning trajectory, you will set up Scratch coding clubs in your school. The main competences that you need for this are:

- Motivating learners, especially girls, to join the clubs and be active in them.
- Having the digital literacy skills and technical competences to explain basic concepts of coding in Scratch.
- Pointing your learners to resources to continue developing their coding skills.
- Facilitating clubs in a learner-centred way, focusing on collaboration, problem-based learning, and self-regulation.

This guide consists of 10 modules. In the preparatory unit, you will get ready to start working with Scratch: downloading the software, creating an account and adding students to your Scratch virtual class. In the modules 1-7, you will learn the key functionalities of Scratch. Guided exercises and a challenge at the end of each module will help you to practise what you have learned. In Module 8, we discuss the pedagogical aspects of coding clubs. These will help you to set up a coding club in your school and ensure that it is learner-centred and collaborative. In the final module, we will discuss gender in STEM and ICT education. Many learners, parents and teachers still believe that STEM and ICT are subjects for boys. Coding clubs can help to raise awareness that ICT and coding are for boys and girls.

0.2 IMPORTANCE OF CODING

Computers are not clever, but they are very obedient. They will do exactly what you tell them to do. Coding is the art of instructing a computer what to do. Programming languages are used to communicate to a computer what you want it to do. Coding involves feeding the computer step-by-step commands.

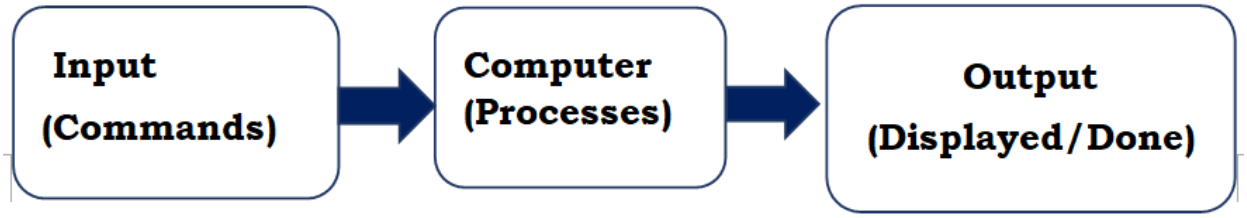


Figure 1: The coding process

There are many coding languages. Based on the methods of feeding instructions to the computers, we divide programming languages into two categories: **block-based coding** and **text-based coding.**

In text-based coding, instructions are fed to the computer through typing a text. Examples are Python, Java, JavaScript, C and C++. The Block based coding is where instructions are mainly represented as blocks. Examples are **Scratch**, Blockly and Elemental (still in development).

0.3 WHAT IS SCRATCH?

Scratch is a block-based, visual programming language that is used by learners all over the world. This visual language is in the shape of blocks, and it allows users to create online projects, games, apps, and many other things.

Scratch helps learners who are new at programming to become fluent with digital media and learn to bring their ideas to life. Through Scratch, learners can demonstrate and deepen their understanding of key topics of the curriculum and develop digital literacy skills. Scratch offers teachers the opportunity to integrate coding concepts into every day's teaching and learning activities and help their students develop computational thinking skills and life skills known as the 4 Cs (Creativity, Communication, Collaboration and Critical thinking).

One of the most interesting facts of this visual programming language is the involvement of the community. When someone finishes a project, or whenever a user wants to, they can share and discuss their creations with other members of the community.

You will be surprised to learn how easy it is to program by using Scratch's blocks. With a very user-friendly interface and attractive colours, Scratch is the ideal platform to begin the process of learning how to code. Then, moving to a programming language like Java, C++, or even Python will be a much more natural process for your learners.

Watch this **intro video** about Scratch:
https://www.youtube.com/channel/UCOUJqZk5znHVCvPMvQY53Fg

0.4 GETTING STARTED WITH SCRATCH

Scratch can be used online and offline. In this guide we will use **Scratch version 3.0**.

- **Online Scratch** is available at http://Scratch.mit.edu

- **Offline Scratch** can be downloaded from https://Scratch.mit.edu/download

Here are the steps to use Scratch offline:

Step 1: Go to https://Scratch.mit.edu/download.
Step 2: Select your Operating System (e.g. Windows)
Step 3: Click the *Direct download* button and save it to your computer to download it.
Step 4: Click on *install* and follow the steps.

0.5 CREATING A TEACHER ACCOUNT

A teacher account lets you make classes, add students, give content to your students, help and grade them.

0.5.1 Creating Your Teacher Account

To create an account, go to **https://Scratch.mit.edu/educators/register**. You will be asked to create a username and password.

Make sure that your username **does not contain your name or personal information, like your school, location, or email address.** Within the Scratch community, users are asked to not share personal information through their usernames. It is important that you and your students follow these guidelines. Accounts that do not respect these guidelines will be deleted.

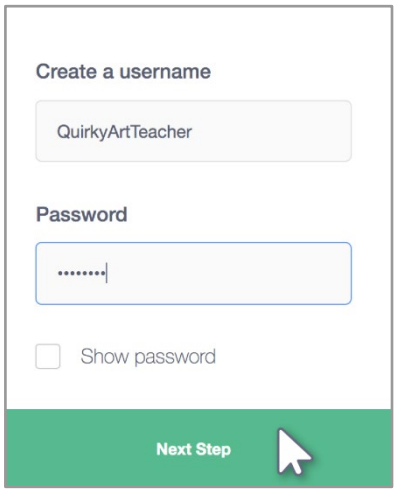
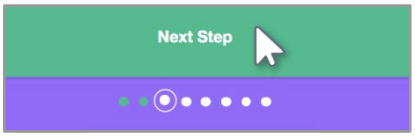
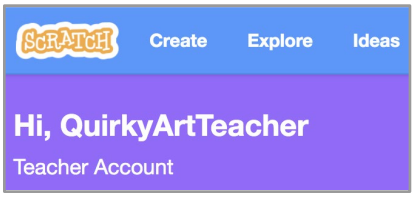
Creating your teacher account	
	Tips for creating your username: • Incorporate the name of the subject you teach ex: Physics class • Use a tool or term from the subject you teach ex: Metamorphic Rocks • Add an important date, to create a unique username ex: Physicsclass2020 • Make it easy to remember, for example Physics Teacher • Make a note of your username and password.
	• Click through each step to complete registration.
	• You will receive an email from the Scratch team with a link. • Check your email inbox and click on the link to confirm your email address. • Check your spam folder if you do not see the email. • Once you have confirmed your email address, your account will be reviewed.
	• When your account has been reviewed and approved, you will receive a welcome email. • Click on the button in that mail to confirm your account. • Congratulations!! You can now log into your account at Scratch.mit.edu!

Table 1: Creating a Teacher's account

0.5.2 Creating Classes

With classes you can manage groups of students and create studios where students can add projects.

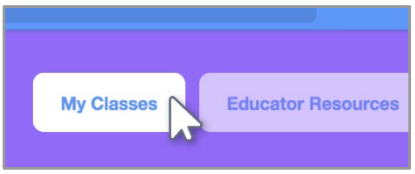
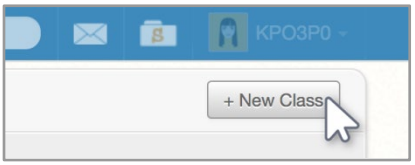
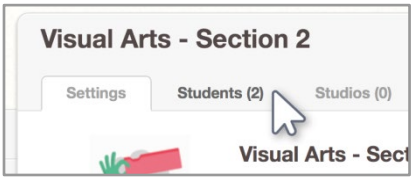
Creating Classes	
	• Once you have successfully logged into your Teacher Account, you will see a bar at the top of the screen with three options. • Select My Classes.
	• To create a class, click the + New Class button at the top right of the page. • Enter a class name and description. Do not include real names and locations, like the name of your school, district or sector.
	• Once you have created a class, you can add students by clicking on the Students tab.

Table 2: Creating classes

0.5.3 Adding Students to Your Class

There are three methods to add students to your class. With the first method you can add an individual student to a class while methods 2 and 3 are more suitable to add several students to a class at once.

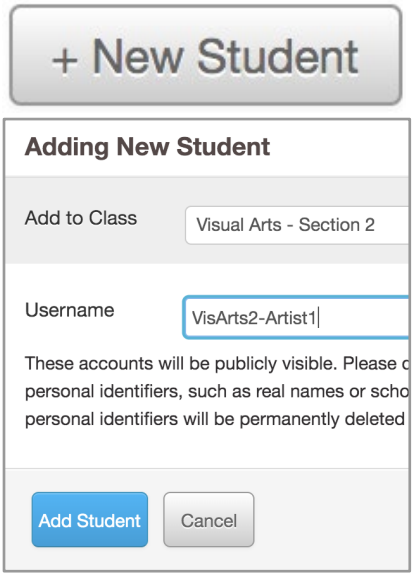
Method 1: Adding Individual Students	
	• Click + New Student to add students individually. You will be prompted to create a username for this student. • Make sure that the usernames you create do not contain identifying information about yourself, your students, or your school. • The password for this student username will automatically be set as your username for your account. • Have students log into their accounts and change their passwords to a password that they can remember.

Table 3: Adding students to your class

Tip: Create a naming rule as a guideline for generating usernames. For example, you may want each name to have a club as prefix, and the student's number on your roster (ex: Umucyoclub-17). In this example, Umucyo club is the name of the club and 17 is the number of the student and all you need is to put it together.

In order to record your students' usernames you can download a sheet available at this link. You need to download and save this Google Document from Google. If you are using Windows, it is best to download the document as a .docx [File > Download >Microsoft Word (.docx)].

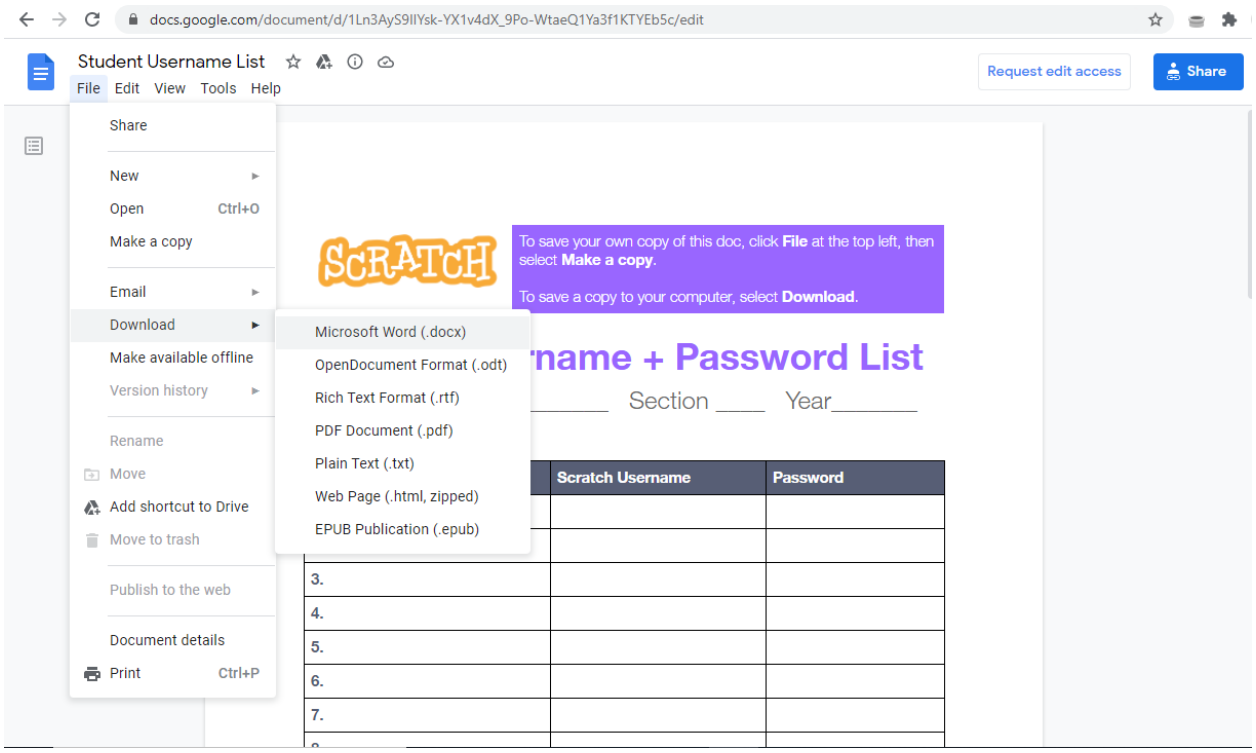


Figure 2: Downloading the Student Username list

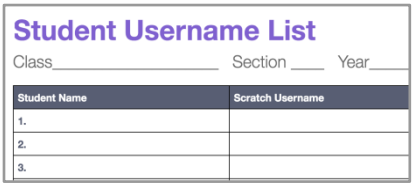
Method 2: Student Sign-up Link Create a sign-up link which allows students to create their own usernames and passwords	
	• Clicking the Student Sign-Up Link button will generate a link which will allow your students to join the class you've created.
	• Share this link with your students via email. When they click on it, they will be asked to create a username and password.
	• Use the sign-up sheet to record the usernames and passwords your students have created. • You can access the sheet via https://docs.google.com/document/d/1Ln3AyS9IIYsk-YX1v4dX_9Po-WtaeQ1Ya3f1KTYEb5c/edit?usp=sharing

Table 4: Adding Students by sign-up link

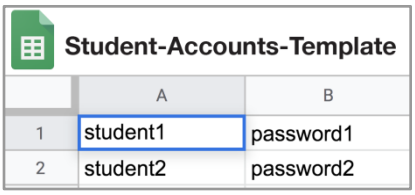
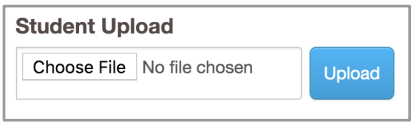
Method 3: CSV Upload Create usernames and passwords for all your students at once using a spreadsheet	
	• Click the CSV Upload button on the class page.
	• Using the template provided, create a username and password for each of your students. Download the template at https://docs.google.com/spreadsheets/d/1OxBTdfOzb74ZcKIm10_QJcvAqSEcgmHV9eCHOZ2oJ3c/edit?usp=sharing • Save this Google Sheets document as a CSV (File > Download > .csv)
	• Once you have created usernames and passwords for each student, click CSV Upload, to upload your file. • Your student accounts will appear under the Students tab for the class.

Table 5: Adding students by CSV Upload

Now that we have created accounts for yourself and your students, it is time to start exploring Scratch!

MODULE 1

INTRODUCTION TO SCRATCH

OVERVIEW

In this module we give an overview of Scratch Studio elements. You will learn where to find the different building blocks to make Scratch code and discover the logic behind interlocking. We will also introduce operators, sequential execution of instructions and multi-threading.

LEARNING OUTCOMES

At the end of this module, you will know how to:

1. Drag blocks to make scripts.
2. Animate by changing costumes.
3. Add and change a backdrop.
4. Use operators.
5. Debug your code.

1.1 THE SCRATCH INTERFACE

The Scratch interface is the screen you see when you start Scratch. It consists of multiple elements (Figure 3). This might seem complex the first time you are using Scratch. But don't worry, we will explain all elements of the interface in this Module.

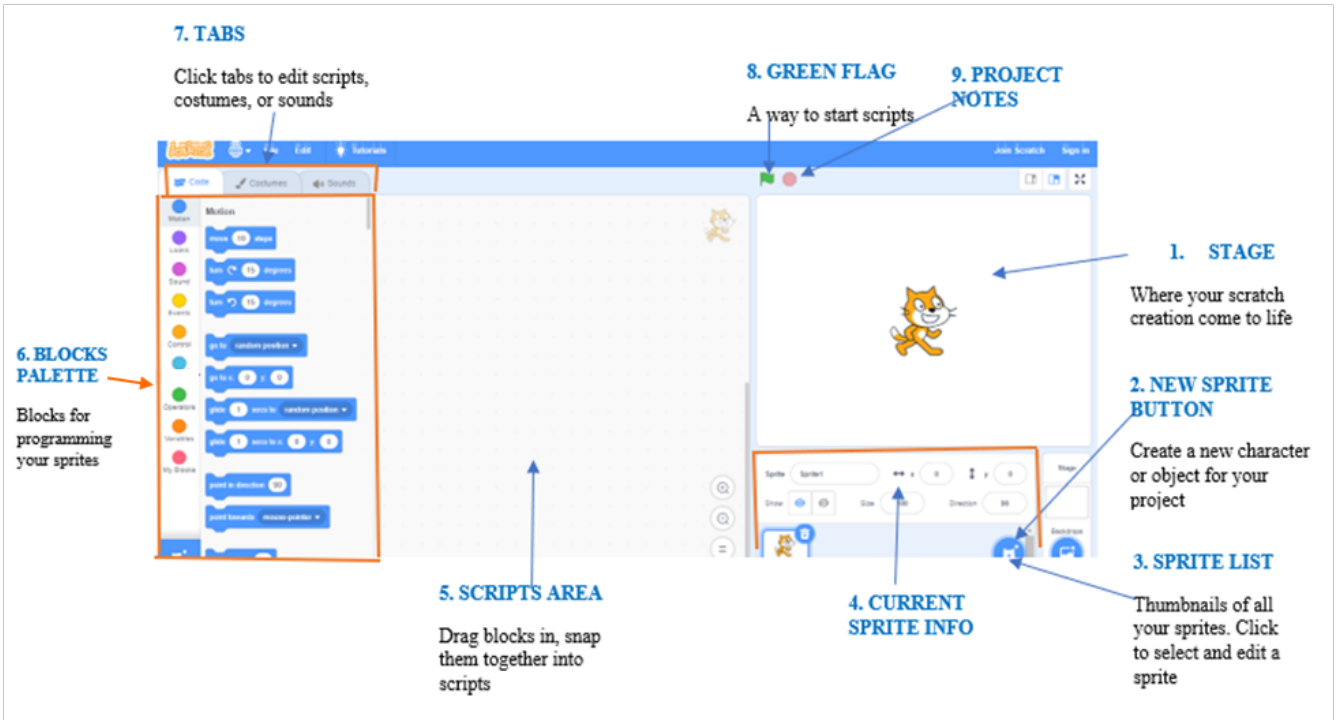


Figure 3: Overview of Scratch Studio Elements

1.1.1 Menu

When you open Scratch, you will see the Main menu. Here you can create a new project, share your project and do many other things.

In the menu box you find:

- The **Language menu** allows you to select the language used on the Scratch blocks.
- The **File menu** allows you to save and import (open) files or projects which are stored on your computer.
- The **Edit menu** allows you to turn on the turbo mode. In the process of writing codes, turning on the turbo mode will make your scripts run quicker. To activate the mode, click shift + green flag. You will see that the project becomes much faster.
- The **Tutorials menu** will give you a series of videos that introduce Scratch and show all options and possibilities.
- The **join Scratch** and **sign in** will allow to open accounts and log in to Scratch. These are the menus on Scratch headers but most of them have other options.

1.1.2 Stage

The **Stage** is where you see your stories, games, and animations come to life (Figure 3). When you add blocks of codes and animate the sprite, everything appears on the stage. Sprites move and interact with one another on the stage. The stage is 480 units wide and 360 units tall. It is divided into an x-y grid. The middle of the stage has an x-coordinate of 0 and an y-coordinate of 0. We will look into more detail in the XY grid in Module 2.

1.1.3. New Sprite Button

*A **sprite** is an element or a character that you want to use in your story. There is a list of sprites that you can use in Scratch, but you can also create your own sprite. When you start a new Scratch project it begins by default with a single cat sprite. To create new sprites, click on the buttons as shown in*

Figure 4.

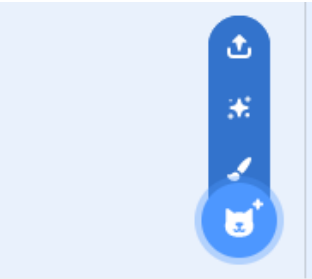


Figure 4: New Sprite button

The **Sprite List** appears and displays thumbnails (small pictures) of all sprites in the project (Figure 5). For each sprite, you can see the sprite's name, scripts and costumes.

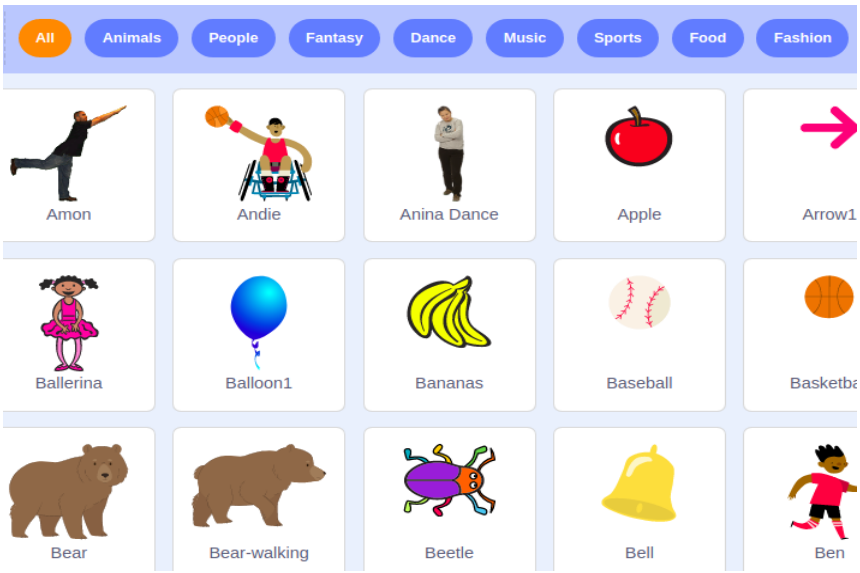


Figure 5: Sprite list with different categories of sprites

To see and edit a sprite's scripts, costumes, and sounds, click on the sprite's thumbnail in the Sprite List – or double-click on the sprite itself on the Stage. The selected sprite is highlighted and outlined in blue in the Sprite List.

Just as a sprite can change its appearance by switching costumes, the Stage can change its appearance by switching **backgrounds**. To see and edit the scripts, backgrounds, and sounds associated with the Stage, click on the Stage icon at the left of the Sprite List.

Deleting a sprite

If you want to delete a sprite, right-click on the sprite and select "delete" from the pop-up menu.

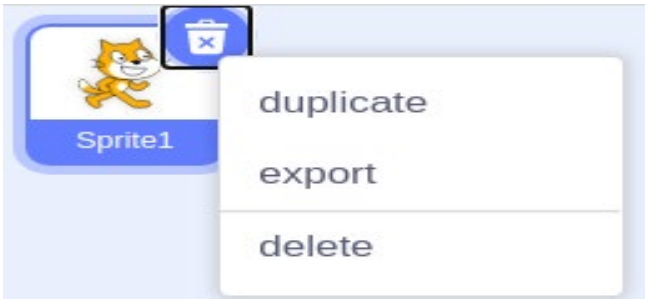


Figure 6: Pop-up menu after right click on the current sprite to delete it

1.1.4 Current Sprite Info

The sprite info shows a sprite's name, x-y position (horizontal and vertical position), appearance, size and direction. You can type in a new name for the sprite. The sprite's direction indicates the position of the sprite and which direction the sprite will move to.

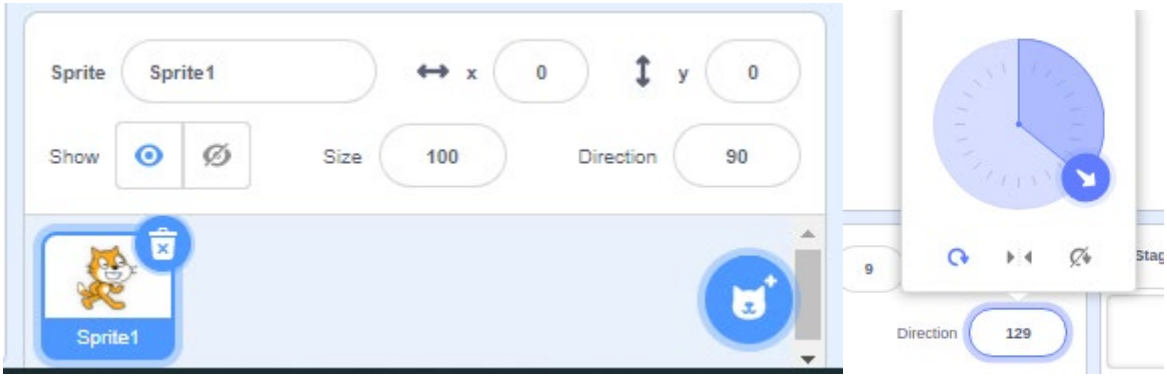


Figure 7: Current Sprite info

The blue arrow line on the thumbnail (Figure 7) shows the sprite's direction. You can drag this line to change the sprite's direction. Double-click on the sprite to set the direction back to its original state (direction=90).

1.2 THE BLOCK PALETTE AND SCRIPTS AREA

To program a sprite, you **drag blocks from the Blocks Palette to the Scripts Area**. To run a block, click on it. Create scripts (programs) by connecting blocks together into **stacks**.

1.2.1 Connecting Blocks

When you drag a block around the Scripts Area, a white/grey area highlight indicates where you can drop the block and form a valid connection with another block.

Figure 8: Connecting blocks in a script

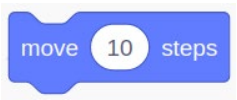
To move a stack, pick it up from the top block. If you drag out a block from the middle of a stack, all the blocks beneath it will come along with it.

To copy a stack of blocks from one sprite to another, drag the stack to the thumbnail of the other sprite in the Sprite List.

Figure 9: Copy a stack of blocks to another sprite

1.2.2 Blocks with Editable Text Files

Some blocks have a white field in which you can edit values.



To change the value, click inside the white area, delete the existing number, and type a new number.

You can also drop rounded blocks inside these areas as shown below:

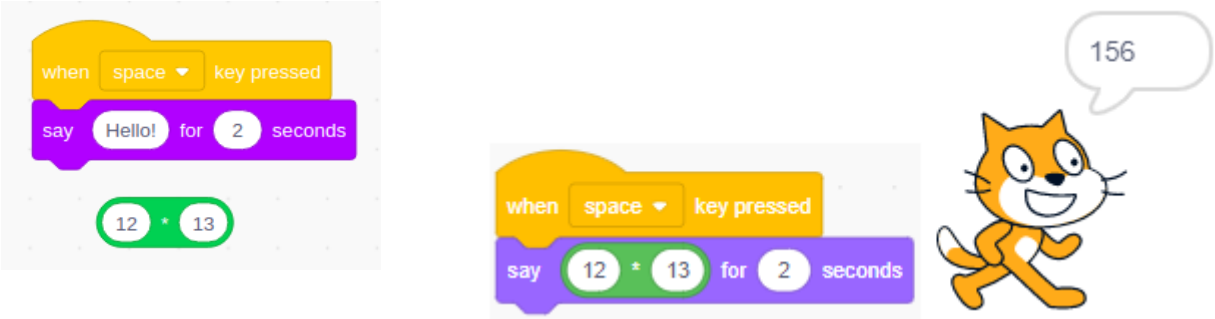


Figure 10: Dragging the multiplication operator to replace hello (Left). The sprite says the result from multiplication operator (right)

1.2.3 Blocks with Drop Down Menu

Some blocks have pull-down menus. Just click on them to see the contents, and then click on the value you want to select.

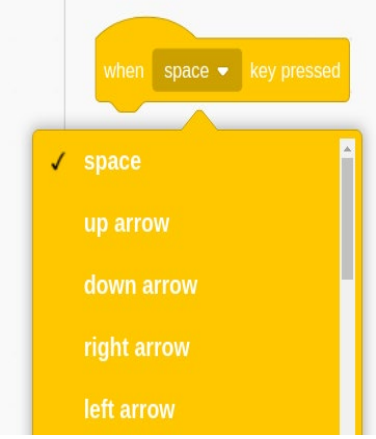


Figure 11: Example of block with pull-down menu

1.3 TABS

There are 3 tabs just below the headers and you can click on each to change the appearance of the sprite. The sections are the **code, costumes, and sounds (**see Figure 12)**.**

- The code section holds the block palette that helps in building the functionality of each sprite.
- The costume section helps in designing and decorating the sprite.
- The sound section gives you the option to add sounds to your project.

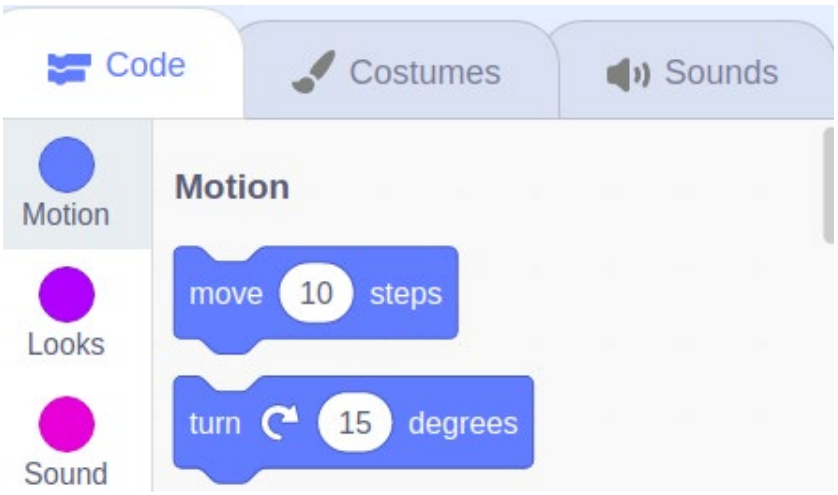


Figure 12: Code, Costumes and Sounds tabs

1.3.1 Code Tab

Code is the default tab we see when we open a Scratch project, it helps us to access the **block palette** (Figure 13).

It is important to realize that a Sprite cannot do anything by itself. A Sprite's actions are a response to scripts that are added by the user into the Script's (or program code) area. These scripts written in sequence are the instructions that tell the sprite what to do.

The user drags individual pieces of code (blocks) from the ***Block Palette*** into the Script area. These blocks then fit together like bricks to create the instructions for the Sprite.

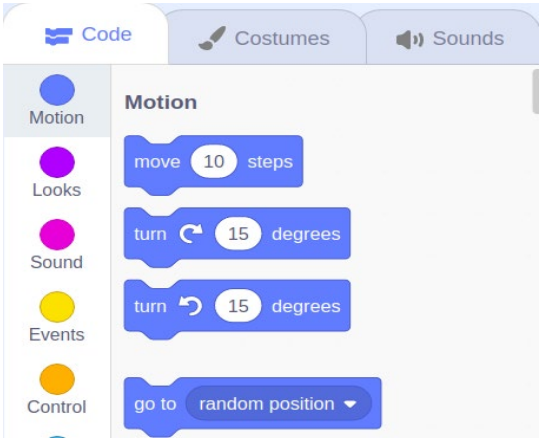


Figure 13: The Code tab

We can still add a new block palette by clicking on the add extension button at the bottom of the block palette.

Figure 14: Adding a new block

A new **palettes extension** appears, and you can choose the one that you can top up with the existing palette blocks. This will show you a page with different categories of blocks to extend your current palette. Some of these are quite advanced and complex, so for now we will be sticking to our default block palette.

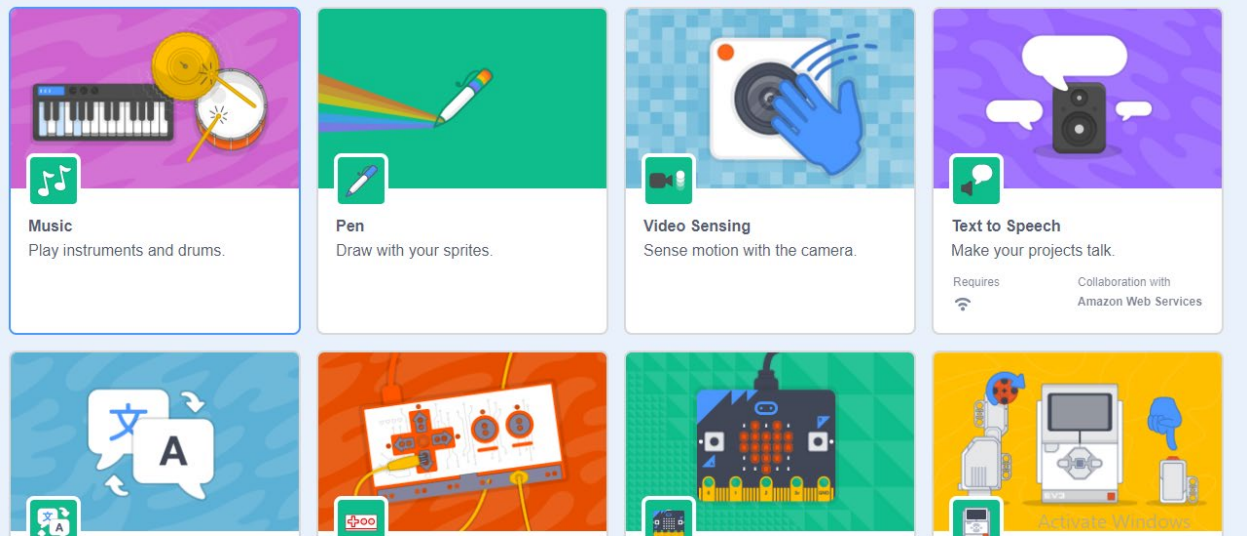


Figure 15: List of available blocks

1.3.2 Costumes Tab

Click on the **Costumes Tab** to see and edit the sprite's costumes (Figure 16). By default, the cat Sprite has two costumes. The sprite's current costume is highlighted. To switch to a different costume, simply click on the thumbnail of the costume you want.

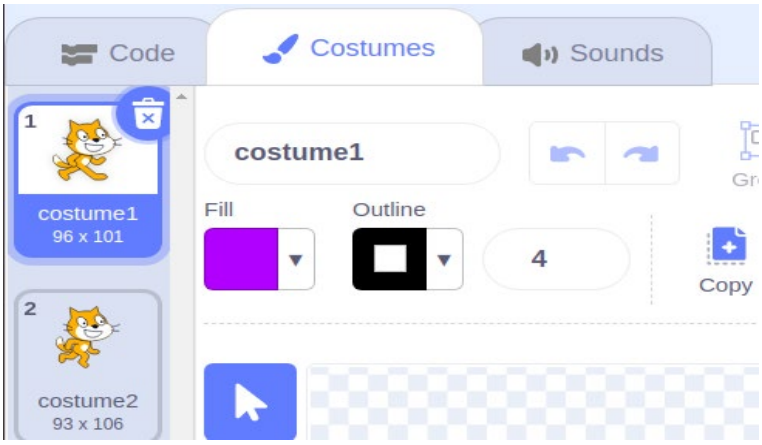


Figure 16: The costumes which comes with the cat sprite

Create a new sprite costume

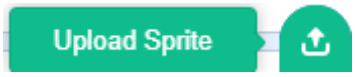
there are two ways to create a new Sprite or give it a new costume:

1. Click 

to paint a new costume in the Paint Editor. When you are inside the paint option, you can also take a picture.

Click 

to take photos from a webcam. Each time you click the button (or press the spacebar), it takes a photo.

2. Click 

to import an image file from your computer. Scratch recognizes many image formats: JPG, BMP, PNG, and GIF (including animated GIF).

After inserting or creating a sprite, you will use it to create a game and a story. In the game plan, you have described what each sprite will do. That is good time to see if the sprite you described is already on the stage. In the process of instructing a sprite to do something, you will need to use the sensing block.

Each costume has a costume number (displayed to its left). You can rearrange the order of the costumes by dragging the thumbnails. The costume numbers update if you change their order. Right-click on a costume thumbnail to convert the costume into a new sprite, or to export a copy of the costume.

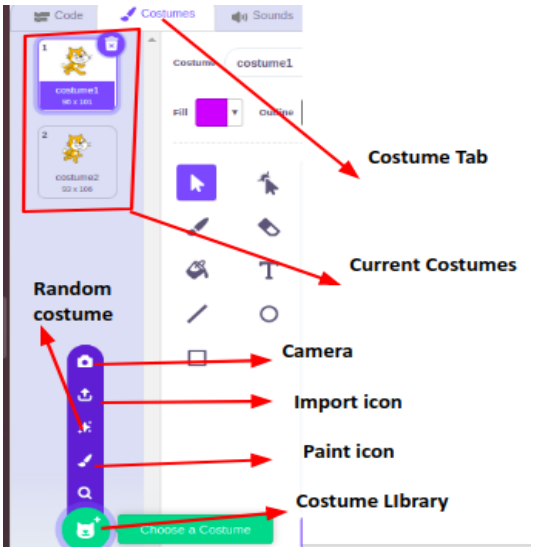


Figure 17: Costume tab and location of icon to create a new costume

1.3.3 Sounds Tab

Click on the Sounds Tab to see the sprite's sounds.

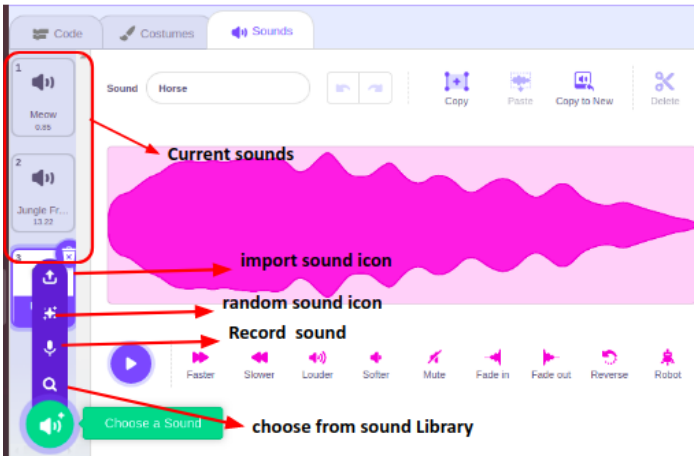


Figure 18: Sounds Tab

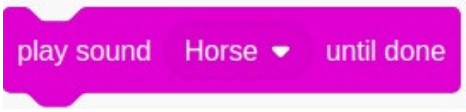
You can record sounds and import sound files. Scratch can read MP3 files.

1.4 SCRATCH BLOCKS

There are three main types of blocks in the Blocks Palette: Stack blocks, Hat blocks and Reporter blocks. Knowledge of these block types will help you in building your projects.

1.4.1 Stack Blocks

Stack blocks have bumps on the bottom and/or notches on the top. You can combine these blocks together into stacks.

Description	**Examples**
Stack blocks have bumps on the bottom and/or notches on the top.	
Some stack blocks have an input area inside them, where you can enter values such as numbers or text.	
Some stack blocks have a pull-down menu where you can choose an item.	

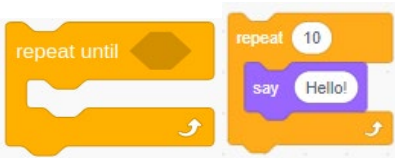
Some stack blocks have a C-shaped “mouth” into which you can insert other stack blocks.	

Table 6: Stack blocks description

1.4.2 Hat Blocks

Hat blocks have rounded tops. These blocks are placed at the tops of stacks. As Hat blocks are designed **to start a script**, they are shaped so no blocks can go on top of them. They wait for an event to happen, such as a certain key being pressed, to start a script and run the blocks underneath them. Notice that the text on Hat blocks always start with “when”. They will start running the code or instructions underneath when a certain event happens.

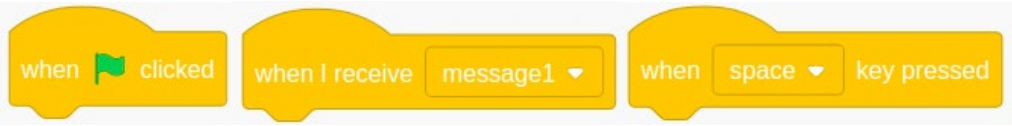


Figure 19: Examples of Hats blocks

1.4.3 Reporter Blocks (Reporters)

Reporter blocks are designed to fit in the input area of other blocks. Reporter blocks come in two shapes and fit only into “holes” of the same shape.

A reporter block is a block that **reports or returns a value**. This can be anything, from numbers to strings (text). Unlike a stack block, which changes something on the Stage, plays a sound, stops the script, or changes a variable, reporter blocks cannot be placed directly above or below another block. Instead, they are dropped into a number, text or drop-down input When Scratch runs a block with a reporter block inside, it will first run the reporter block to find the value of the input.

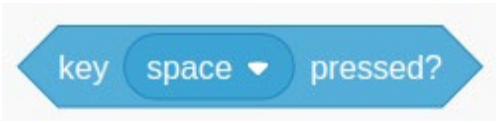
Description	Examples
Reporters with rounded ends: To quickly view the value of a reporter, simply click it in the editor and Scratch will display the value in a small bubble.	
Reporters with pointed ends: to report "Boolean" values (True or False) and fit inside blocks with pointed holes. A Boolean only checks whether a certain condition is True or False. In this example the Boolean will check if the "space key" is pressed.	

Table 7: Reporter blocks

Some reporter blocks have a check box next to them (figure 1.17). If you click the check box, a monitor appears on the stage, displaying the current value of the reporter. As the value of the reporter changes, the **monitor** updates automatically.

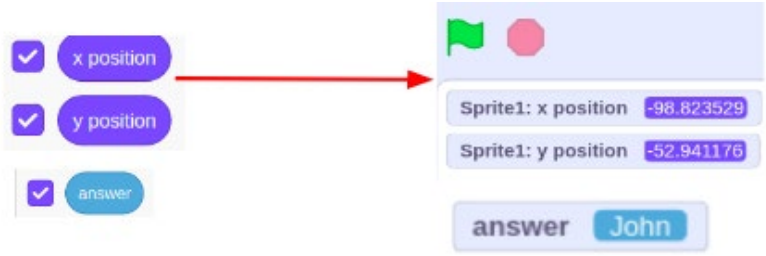


Figure 20: Blocks with checkbox are checked to appear on stage

A monitor can display the value of the reporter in several different formats. The following example monitors the value of someone's age:

Description	Example
A small readout with the name of the reporter	

A large readout without any name	
A slider that allows you to manipulate the value of the reporter (available only for variables)	

Table 8: Different display formats of values returned by reporter blocks

Double-click or right-click on the monitor to bring the slider. The slider format is available only for user-created variables. Right-click on the slider to adjust its minimum and maximum values.

1.4.4 Green Flag

Clicking the Green Flag starts the program you created. It will run all the scripts you have created in your Scratch project at the same time. In presentation mode, the green flag appears as a tiny icon  at the top-right corner of the screen. Pressing the **Enter key** has the same effect as clicking the Green Flag.

1.5 OPERATORS

The **Operators Block Palette** contains 18 blocks. All these blocks have one thing in common: they take one or more values and use them to return another value.

In programming, using values to produce other values is called **performing an operation**. The blocks that make operations possible are called “operators.” Operations are specific tasks that work with values to produce results.

A simple example of an operator is a mathematical calculation (e.g. 1+3=4) or making a combination of words to form a sentence (e.g. “Hello” + “World” = “Hello World”).

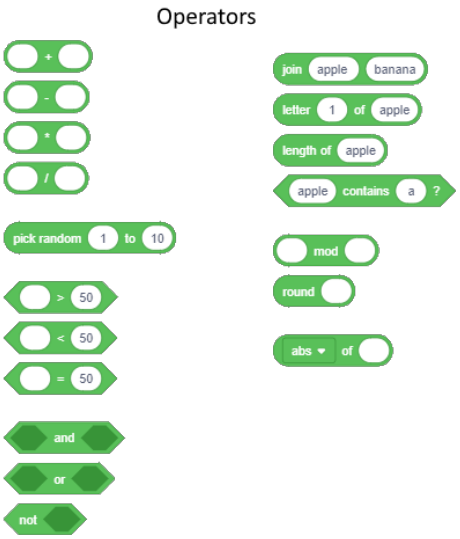


Figure 21: The operator blocks

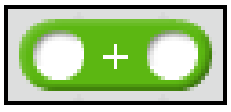
Operators are powerful blocks. Understanding and mastering them is key to do all sorts of useful and essential things in your projects. Some things they make possible include:

- Keeping track of the score or result
- Joining words to change what sprites say
- Making sprites move in a realistic way
- Solving math problems
- Checking answers of a quiz
- Randomizing the positions and movements of obstacles and monsters in a game

Doing Maths with Scratch

You don't need to be a math genius to write programs but knowing how to use the **math operator blocks** certainly makes you look like genius! This section shows you how to use each operator and gives you some tips, tricks, and examples that you can use in your own projects.

1.5.1 Addition



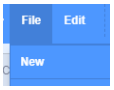
The addition block takes two numbers (or variables containing numbers) and returns the sum.

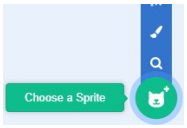
Activity: Addition

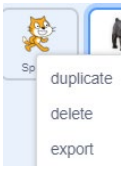
To try out the addition block, as well as the other math operators, you will ask the help of a puppy.

The following activity adds two values stored in two variables called **my Number** and **myNumber2** and the puppy tells the result.

Follow these steps:

1. Click File ⇒ **New** from the top menu to create a new project. 

2. Click the **Choose a Sprite** icon  in the **Sprite menu** to open the Sprite Library.


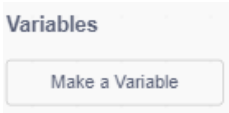
3. Locate the sprite named **Puppy** and click on it to add it to your project.
4. From the project, right-click the Cat and select delete to remove her from your project.


5. Drag a 

(when green flag clicked) block from the Events Block Palette to the Scripts Area.
6. Drag a 

(*forever)* block from the Control Block Palette and snap it to the 

block.
7. Create two new variables in the 

"Variables" Block Palette and then click on 

. The first should be called **my Number** and the second one **myNumber2**.

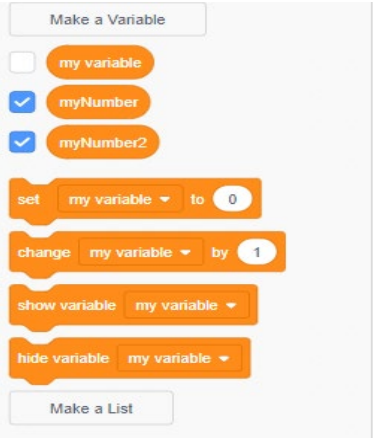
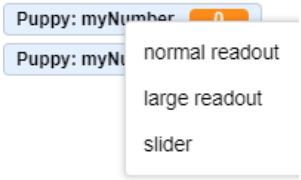


Figure 22: Creating two variables

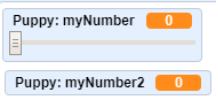
Your Variables Block Palette should now look like Figure 22.

8. Right-click on 

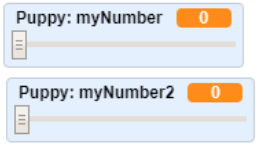
in the Stage.
    a. Several options for how to display the variable appear:

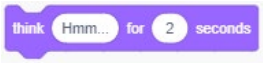


9. Select slider from the variable display menu.
    a. The variable display on the Stage turns into a slider.

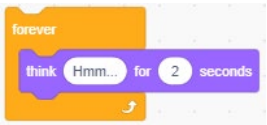


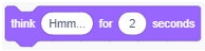
10. Right-click myNumber2 on the Stage and change it into a slider as well.



11. Drag a 

"think ()" block from the Looks Block Palette to the Scripts Area and snap it inside the "forever" block as shown below:



12. Drag a  "() + ()" block from the Operators Block Palette and place it into the  "think ()" block.

13. Drag the "my Number" variable from the Variables Block Palette into the first spot in the  block. Drag the "myNumber2" variable from the "Variables" Block Palette into the second spot in the  block.
    a. Your script should now look like Figure 23.

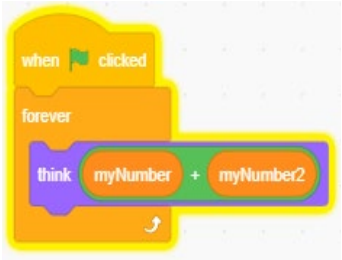


Figure 23: Script for addition

14. Change the title of your project in the space above the Stage to **Addition Puppy** and select File ⇒ Save from the top toolbar.

Click the Green Flag and then click and drag the sliders on your variables on the Stage. The puppy instantly does the math and tells you the answer (Figure 24).

Figure 24: The puppy doing addition

1.5.2 Subtraction

Subtraction is addition in reverse. The  or "() - ()" block in Scratch subtracts a number from another and returns the result.

Activity: Subtraction operator

You are going to subtract a value stored in "**myNumber2"** from the value stored in "**my Number"**, then the puppy tells the result. Remember that "my Number" and "myNumber2" were created in activity 1.1.

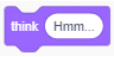
To create a Subtraction Puppy project, follow these steps:

1. Select File ⇒ File and then Save as a Copy from the top menu.

An exact copy of the project is created, with the title changed to Addition Puppy copy.

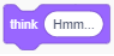
2. Change the title of the project to **Subtraction Puppy**.

3. Drag the 

block out of the 

block in the Scripts Area, as shown below:

4. Drag a 

block from the "Operators" Block Palette and drop it into the 

block.

5. Drag the "my Number" block from the 

block in the Scripts Area and place it into the first space in the 

block.

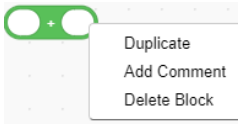
6. Drag the myNumber2 block from the 

block in the Scripts Area and place it into the second space in the 

block.

7. Right-click the empty 

block and select Delete.



Your script should now look like Figure 25.

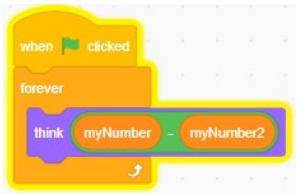


Figure 25: Script for subtraction

Test the Subtraction Puppy by clicking the Green Flag and dragging the variable sliders on the Stage. The puppy instantly executes the subtractions and shows the answer, as you can see below:

Figure 26: The Subtraction Puppy result

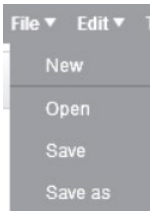
1.5.3 Multiplication

In Scratch, as well as in most other programming languages, the multiplication symbol is the * character. The Scratch  multiplication operator block is used to calculate multiplications.

Activity: Multiplication operator

You will multiply two values stored in two variables called **my Number** and **myNumber2,** and the puppy will show the result. Follow these steps to create the Multiplication Puppy program.

1. Select File ⇒ File and then Save as a Copy from the top toolbar.



2. Change the title of the project to **Multiplication Puppy**.

3. Replace the () - () block in the Scripts Area with the () * () block.

Your script should look like Figure 27.

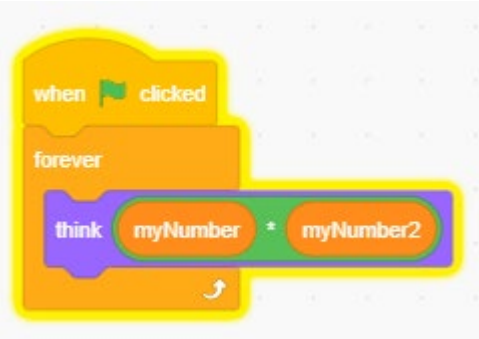
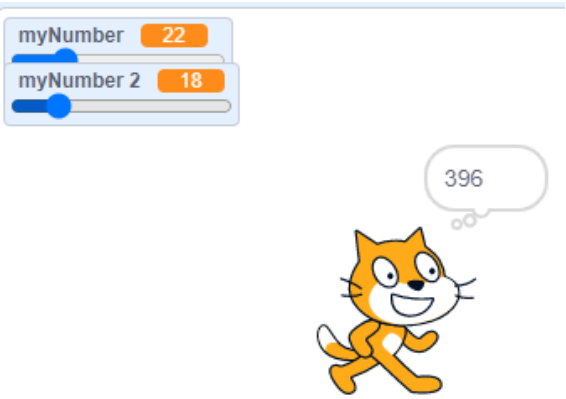


Figure 27: The Multiplication Puppy program



1.5.4 Division

If you are not completely impressed by how fast your puppy can do addition, subtraction, and multiplication, this next example will convince you! The Scratch division operator is shown here:

Activity: Division operator

This activity divides two numbers stored in the variables **my Number** and **myNumber2.** Next, the puppy shows the result. To create the Division Puppy program, follow these steps:

1. Create a copy of the Multiplication Puppy program by selecting File ⇒ File and then Save as a Copy from the top toolbar.
2. Change the title of the program to **Division Puppy** and then select File ⇒ Save.
3. Replace the  block in the Scripts Area with a  or "() / ()" block.
4. Click the Green Flag and then set both number variables to **0** by dragging the slider handles all the way to the left. The puppy starts thinking the characters NaN. This stands for Not a Number.

In math, we say that the result of dividing 0 by 0 is “no defined value” because there is no way to find an answer for it. Scratch simply says that it is not a number.

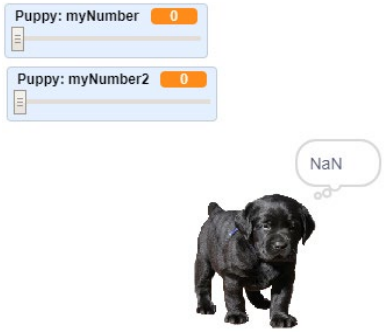


Figure 28: Puppy saying “Nan”

5. Set the value of the first variable to a number greater than 0 but leave the second number set to 0. The puppy starts to think the word Infinity, as shown in Figure 29.

NB: When you divide any number by zero, the result is always infinity. For example, if you have a pizza with 12 slices and no one to eat it, how many pieces will each person get and how long will it take to eat? If you have 20 blocks and you want to divide the blocks into groups of 0 blocks, how many groups can you make? There is no answer these questions, and no matter how many groups of blocks you have or how long you say it will take to eat the pizza, a larger number is also possible. So, Scratch and the Division Puppy say that the value is infinity.

6. Drag the slider for “my Number” all the way to the left to set it to “**0**” and drag the slider for “myNumber2” to any number other than “0”. The result is “0”. Can you figure out why? If you have no pizza, it does not matter how many people you want to give a slice of pizza to—they will all get no pizza.

Figure 29: Puppy saying “0”

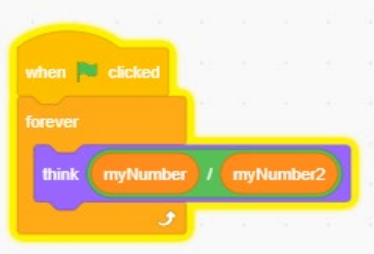


Figure 31: The Division Puppy program

Figure 30: Division Puppy saying "Infinity"

1.5.5 *Coding Logically- Boolean Operator Blocks*

The Boolean operator blocks each use a value, or multiple values, to decide whether something is true or false. Boolean operating is also known as Boolean logic. To quickly review, the **six Boolean blocks** in the Operators Block Palette are:

- () > (). The *greater than* block produces a value of true if the first number is bigger than the second.
- () < (). The *less than* block produces a value of true if the first number is smaller than the second.
- () = (). The *equal to* block produces a value of true if the values (which may be numbers or text) on both sides of the equal sign are the same.
- () and (). The *AND* block produces a value of true if the statements on both sides of the AND are true.
- () or (). The *OR* block produces a value of true if either statement to the right or to the left of OR is true.
- Not (). The *NOT* block produces a value of true if the result of the block inside of it is false.

Boolean blocks are used inside control blocks to choose between multiple paths or to decide whether to keep looping.

In the next activity, you will use Boolean operator blocks to experiment with creating true and false values.

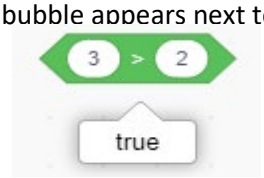
Activity: operations to produce true and false values.

The following activity shows how different operations produce true or false values.

1. Drag a "() > ()" block from the Operators Block Palette to the Scripts Area  .

2. Enter **3** into the left side of the () > () block and **2** into the right side. 

3. Double-click the (3) > (2) block in the Scripts Area. A talk bubble appears next to the block and tells you that the result is true, that 3 is greater than 2.



4. Right-click the () > () in the Scripts Area.

In addition to the normal right-click options, you also see three more: <, =, and > (Figure 32).

5. Select the less than (<) symbol from the right-click menu. The () > () block changes into a () < () block and leaves your values as they were.

6. Double-click the () < () block. A talk box appears displaying false, showing that 3 is not less than 2.

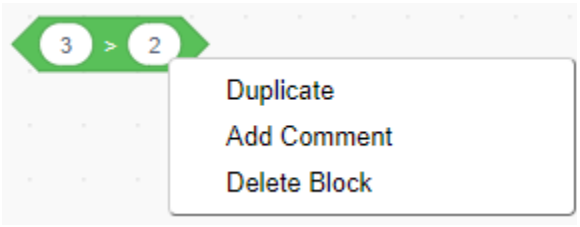
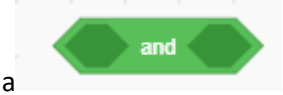


Figure 32: Finding out whether the number 3 is smaller than the number 2 (left) and additional right-click options (right)

Using the () < (), () = (), and () > () blocks by themselves is simple, but you can also combine them to code much more complicated logic. Follow these steps to see some of the things that Scratch can do with logic.

1. Drag a () and () block from the Operators Block Palette to the Scripts Area 

.

2. Snap your (3) < (2) block into the first space within the () and () block

.

3. Drag a () > () block from the Operators Block Palette to the Scripts Area and snap it into the right side of the () and () block.

4. Change the values in the () > () block to **99** and **1** 

.

5. Double-click the "() and ()" block. The result is false, as shown below, because only one of the two halves of the () and () block is true.

6. Drag a () or () block from the Operators Block Palette to the Scripts Area. 

7. Drag the (3) < (2) block from the left side of the () and () block to the left side of the () or () block.

8. Drag the (99) > (1) block from the right side of the () and () block to the right side of the () or () block. 

9. Double-click the () or () block. This block produces a true value because one of the sides of the block is true, as shown in Figure 1.35. 

1.5.6 Operations on Text

The text operators in Scratch can work with user-entered text, variables containing text, and just plain text to figure out how many characters are in it, find certain letters in the text, and join pieces of text together.

Combining Text with join () ()

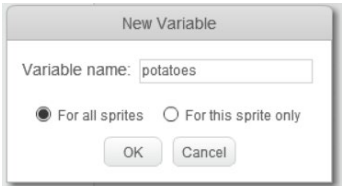
The join () () block lets you combine words, and even sentences, to create custom values to store in variables or to be used by sprites. A very common use for the join () () block is for mixing the values of variables into things that will be said by a sprite.

Activity: Cat counting potatoes

In this activity, you will let the cat count potatoes until a number stored in the variable called **potatoes** is reached. Follow the steps to perform this activity:

1. Click File ⇒ new in the top toolbar to create a new project.

2. In the Data Block Palette, create a new variable called **potatoes**.



3. Drag a" when green flag clicked" block from the Events Block Palette to the Scripts Area

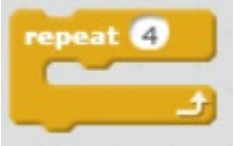
.

4. Drag the set () to () block from the Data Block Palette and snap it to the when green flag clicked block.

5. Make sure that the drop-down menu in the set () to () block is set to potatoes, and that the second space in the block contains a **0**.

6. Drag a repeat () block from the Control Block Palette and snap it to the bottom of the set (potatoes) to (0) block.

7. Change the value in the repeat () block to **4**.



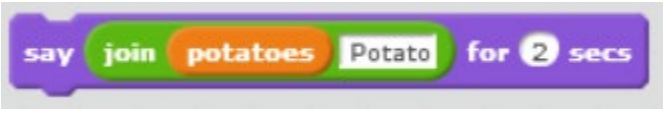
8. Drag a change () by () block from the Data Block Palette and snap it inside the repeat () block. The drop-down menu should be set to potatoes and the second space should be set to 1.

9. Drag a say () for () secs block from the Looks Block Palette and snap it to the bottom of the change (potatoes) by (1) block inside the repeat (4) block.

10. Drag a join () () block from the Operators Block Palette into the first space in the say () for () secs block.

11. Drag the potatoes variable from the Data Block Palette and place it inside the first space in the join () () block.

12. Type the word potato into the second space in the join () () block.



13. Click the Green Flag to start the counting.

The result on the Stage should look like Figure 33.

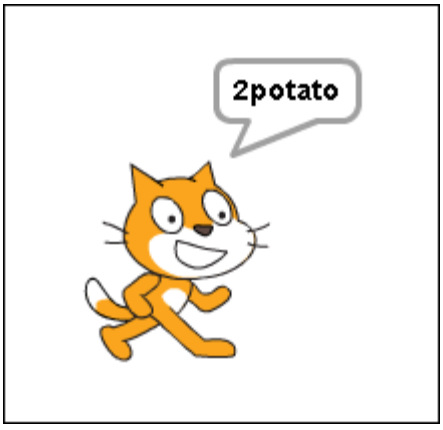


Figure 33: Joining a variable and text

You are almost there, but it is strange how Scratch the Cat does not have a space between the number and the word *potato*. It is important to remember that Scratch does not add spaces for you when you join words together.

To fix the problem, click your mouse just before the word potato in the second space in the join () () block and press the spacebar. Now click the Green Flag again, and it looks much better, as shown in Figure 34.

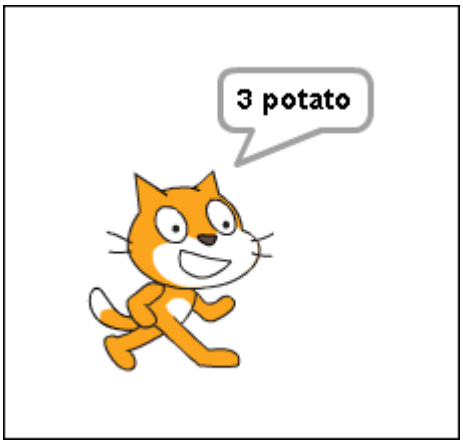
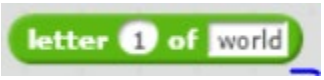


Figure 34: Do not forget to use spaces!

Finding Letters

The letter () of () block tells you what letter comes at a certain position in a word or text. In this activity, you will create a block which tells the first letter of a word.

1. Drag a letter () of () block from the Operators Block Palette to the Scripts Area.
2. All coding languages start counting from (0) so, you can choose which number to enter. For this, enter the number **1** into the first space and your name into the second space in the block.



3. Double-click the block.

You see a talk bubble with the first letter of your name in it (Figure 35).

Figure 35: Finding out the first letter in your name

When you combine the letter () of () block with the length of () block, you can do some nice things. Try it out.

Getting the length of a text

The length of () block tells you how many characters (including spaces and punctuation) there are in a piece of text. If you want to find the last letter of a person's name, you can combine the length of () block with the letter () of () block. Try it out:

1. Drag a letter () of () block to the Scripts Area.

2. Drag a length of () block to the Scripts Area and place it inside the first space in the letter () of () block.
3. Type your name into both the length of () block and the letter () of () block (Figure 36).
4. Double-click this block to find out the very last letter in whatever text you entered.

Figure 36: A block to get the last letter of a name

When Scratch runs this combined block, it first determines the result of the length of () block. It then uses this number to determine what letter is in that position. In the preceding example, the result of the length of () block is 5 and the result is the letter S because S is the fifth letter in the name Chris.

CHALLENGE

Can you figure out how to use a variable and **an ask () and wait block** to make the Cat ask you your name and then tell you what the last letter of your name is?

1.5.7 Understanding Other Operations

Scratch can go far beyond basic math and logic. The last three blocks in the Operators Block Palette perform math functions that are not basic operations but are quite common in Scratch programs.

() mod ()

The () mod () block divides the first number by the second number and then tells you what is left over (the remainder). For example, if you make a (7) mod (3) block, the result will be 1, because 7 can be divided by 3 twice, with 1 left over. Figure 37 shows the () mod () block in action.

Figure 37: Using the () mod () block

Round ()

The round () block rounds a number to its closest whole number. For example, the block in Figure 38 produces a value of 8 because 7.51 is closer to 8 than it is to 7.

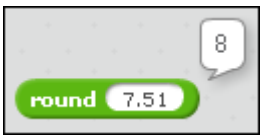


Figure 38: Rounding with Scratch () of ()

This last mathematical operator block is 14 blocks in one! The drop-down menu inside this block has 14 possible values, as shown in Figure 39.

Figure 39: The () of () block has many functions

Explaining how all these functions work and when you might use them is more than we have space for in this guide. If you are curious about them, you can read about them on the Scratch website at http://wiki.Scratch.mit.edu/wiki/()_of_(). Some of them are used in advanced mathematics, and they can help you create more natural movements, curves and lines on the Stage and solve complex problems.

Activity: Make a Math Practice Game

You will create a math practice game that will challenge you with math questions and keeps score of how well you did in answering ten questions. The activity will prompt you to choose between addition and multiplication, then it chooses two random numbers on which to do the operation.

We will start by making a sprite ask what operation you want to practice. We will start with addition and multiplication, but you can expand the program to include any type of operations!

1. Select File ⇒ New from the top toolbar to create a new project.

2. Drag a when green flag clicked block from the Events Block Palette to the Scripts Area.
3. Create a new variable in the Data Block Palette called **Score**.
4. Drag a set () to () block from the Data Block Palette and snap it to the bottom of the when green flag clicked block.
5. Make sure that the first value in the set () to () is **Score** and the second value is **0**.
6. Drag a say () for () secs block from the Looks Block Palette to the Scripts Area and snap it to the when green flag clicked block.
7. Drag another say () for () secs block to the Scripts Area and snap it to the bottom of the previous one.
8. Drag an ask () and wait block from the Sensing Block Palette and snap it to the bottom of the script.
9. Change the value of the first say () for () secs block to **Hello!**
10. Change the value of the second say () for () secs block to **Let us do some math!**
11. Change the question in the ask () and wait block to **Do you want to practice addition or multiplication (enter + or x)?**

Your script should now look like Figure 40.

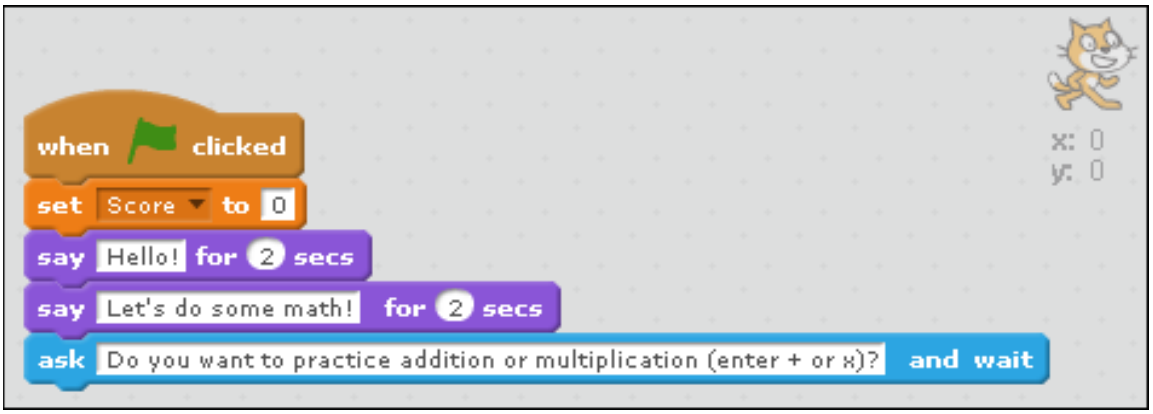


Figure 40: Getting things started in the math practice program

1.5.8 Programming with Different Paths

The next thing you do is to create three different paths the program can take depending on the user's answer to the first question (**nested condition**):

1. If the user enters +, the program takes the user to the addition quiz.
2. If the user enters x, the program takes the user to the multiplication quiz.
3. If the user enters anything other than x or +, Scratch the Cat says that she does not understand.

Activity: Creating Nested Conditions

To program these three paths, follow these steps:

1. Drag an if () then, else block from the Control Block Palette to the Scripts Area and attach it to the ask () and wait block.
2. Drag a () = () block from the Operators Block Palette and place it in the hexagram shape in the if () then, else block.
3. Drag the answer block from the Sensing Block Palette and place it into the left side of the () = () block.
4. Enter the **+** symbol into the right side of the () = () block.
5. Drag an if () then, else block into the else part of the first if () then, else block, as shown in Figure 1.44.
6. Drag a () = () block into the new if () then, else block.
7. Drag an answer block into the left side of the () = () block.
8. Enter a lowercase ***x*** into the right side of the () = () block.
9. Drag a say () for () secs block from the Looks Block Palette and drop it into the else section of the nested if () then, else block.
10. Change the value in the say () for () secs block to **I did not understand your answer.**
11. Your script should now look like Figure 41.

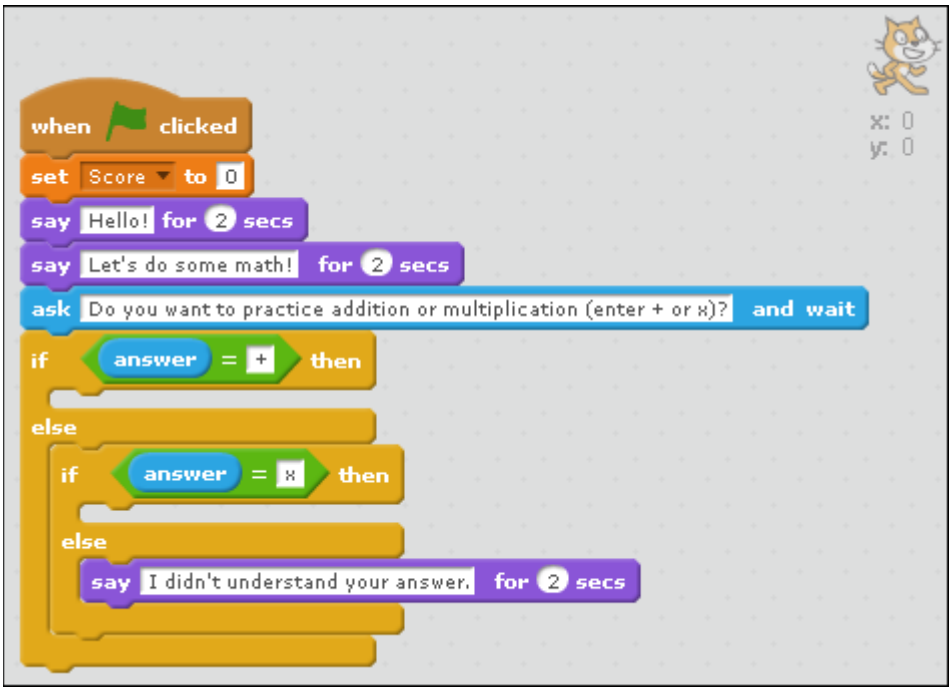


Figure 41: Creating a nested condition

Now you have created the three main branches of the program! Next, you will build the addition quiz.

Activity: Making an Addition Quiz

Follow these steps to create an addition Quiz.

1. Drag a repeat () block from the Control Block Palette and drop it into the empty space in the first if () then, else block.
2. Create two new variables in the Data Block Palette: **Number1** and **Number2**.
3. Drag a set () to () block from the Data Block Palette and snap it inside the repeat () block.
4. Set the first value in the set () to () block to **Number1**.
5. Drag a pick random () to () block from the Operators Block Palette and snap it into the second space in the set () to () block.
6. Change the second value in the pick random () to () block to **100**.
7. Duplicate the set () to () block you just created and snap it to the bottom of the first one.
8. Change the first value in this new set () to () block to **Number2**.
9. Drag an ask () and wait block to the Scripts Area and snap it to the bottom of the set (Number2) to (pick random (0) to (100)) block.
10. Drag a join () () block into the space in the Ask () and wait block.
11. Drag another join () () block into the second space in the join () () block.
12. Drag one more join () () block into the second space of the last join () () block.
13. Make your Ask () and wait block look like Figure 1.46.
14. Drag an If () then, else block to the Scripts Area and snap it to the bottom of the Ask () and wait block.
15. Drag a () = () block into the space in the If () then, else block.
16. Drag an answer block into the first part of the () = () block.
17. Drag a () + () block into the second part of the () = () block.
18. Snap the Number1 variable into the first space in the () + () and snap the Number2 variable into the second space in the () + () block.
19. Drag a say () for () secs block inside the space after then, in this if () then, else block.
20. Change the value of this say () for () secs block to **Correct!**
21. Drag a change () by () block from the Data Block Palette and snap it to the bottom of the say () for () secs block.
22. Change the first value in the change () by () block to **Score** and change the second value to **1**.
23. Drag a say () for () secs block to the Scripts Area and snap it into the else part of the If () then, else block.
24. Change the value of this say () for () secs block to **No, that's not correct.**

Figure 42: An addition question

CHALLENGE

Can you create a message that displays after the repeat (10) loop to tell the users how many answers they got correct?

The finished addition quiz, including the answer to the challenge question is shown in Figure 43.

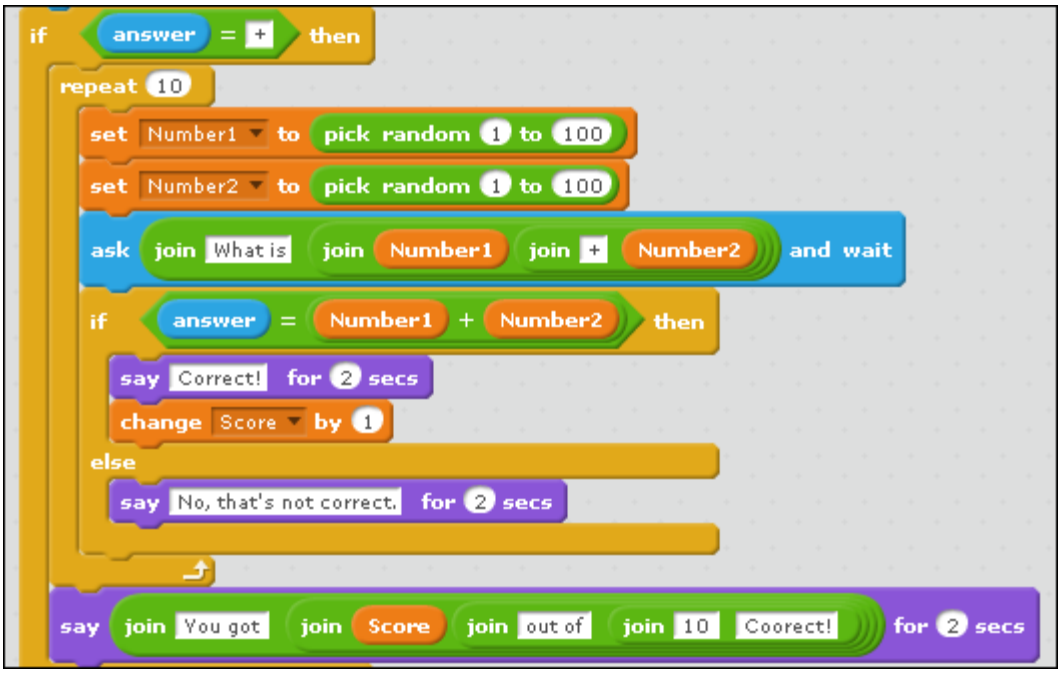


Figure 43: The final addition quiz

Activity: Making a Multiplication Quiz

The multiplication quiz is very similar to the addition quiz. The only difference lies in the messages that are displayed and in the values. Set the range for the multiplication quiz from 0 to 12, so that it is easier for people taking the quiz to do the math in their heads.

Can you create this part of the program yourself? Here are the main steps:

1. Drag the repeat () block 

out of the script for a moment, and then duplicate it and place the duplicate into the space after if (answer = x) then.

2. Put the original repeat () block back where it was.

3. Modify the contents of the new repeat () block, as well as the say () for () block that follows it and gives the score, to match Figure 1.48.

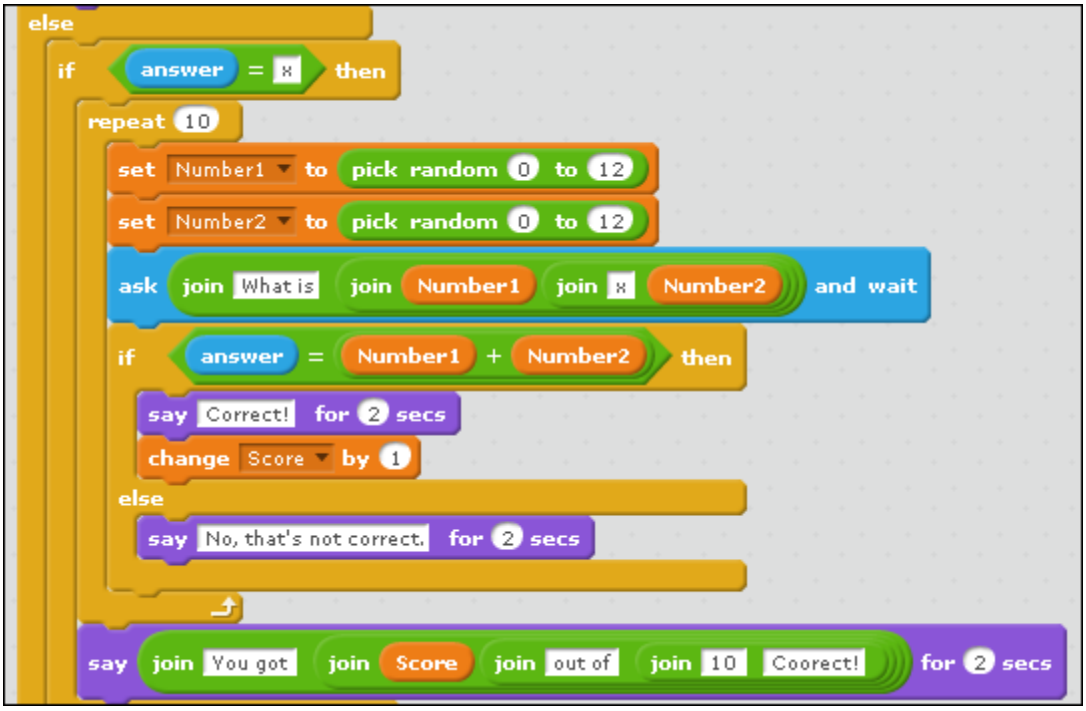


Figure 44: The finished multiplication quiz

When you're done, the whole program should look like Figure 45.

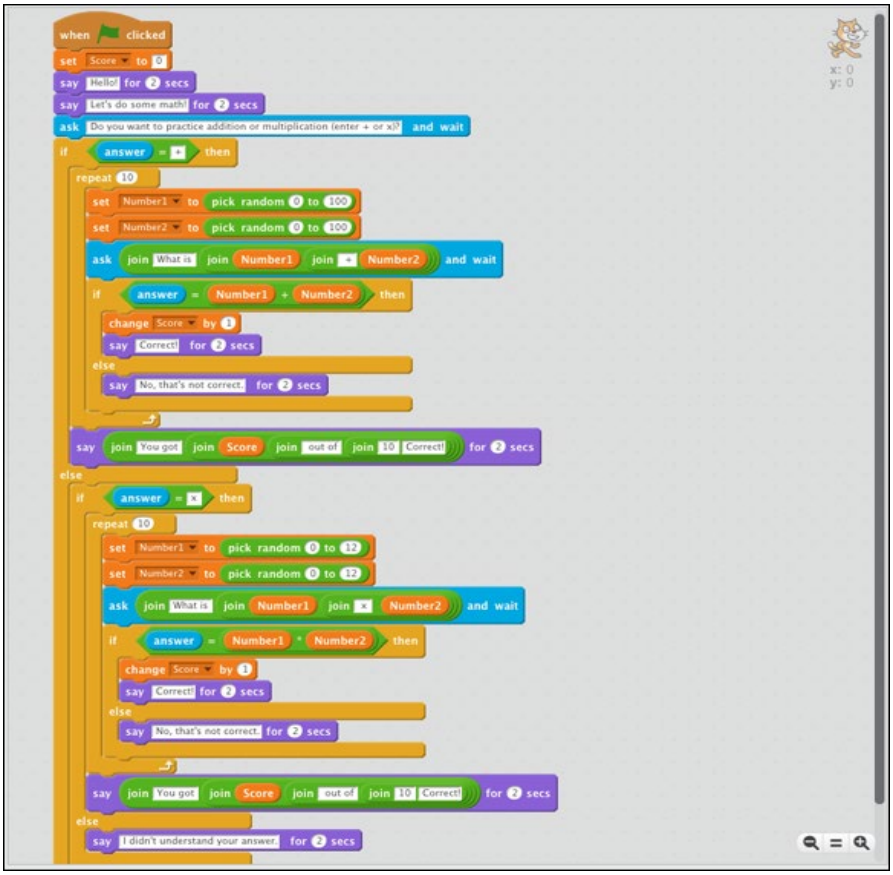


Figure 45:The final Math and Multiplication Quiz Project

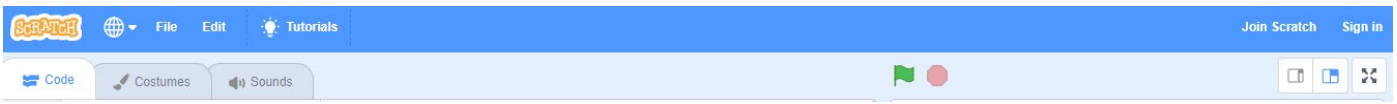
1.6 SEQUENTIAL EXECUTION OF INSTRUCTIONS

1.6.1 INTRODUCTION

A Sprite is a flat image representing a character or object in your program. So far, sprite characters have stayed still. In this section, you will learn how to move around a Sprite and let it talk.

1.6.2 A TALKING SPRITE

By placing blocks in the Script area, the Sprite will tell us his/her name. First go to the *Code section* located at the top left of the Scratch home page.

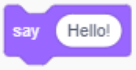


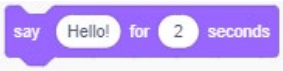
Go to the ***event*** folder. 

Place the following block in the Script area:

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

1. Go to the **Looks** section located in the Folders panel. 

2. Select the blocks twice that Say Hello for two seconds. 

and 

3. Type in the text *Hello!* And, *my name is Kitty* or your choice of name before placing both in the Scripts area.

Figure 46: Blocks to make the sprite talk

It would be nice to give the Sprite a name.

To do so, click on the 

button at the bottom left of the cat sprite icon and type in the text ***Kitty*** or your name of choice in the box.

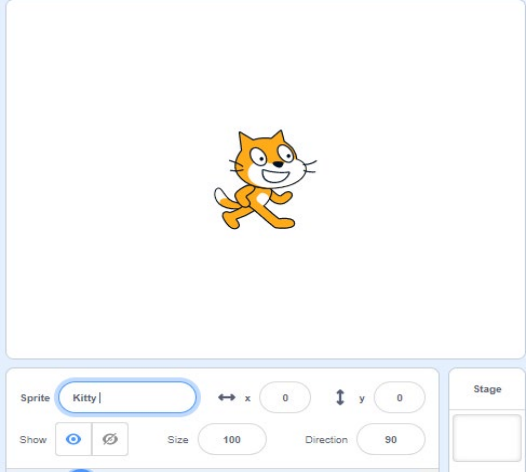


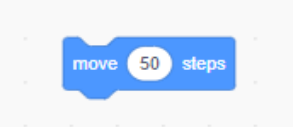
Figure 47: Changing the name of a sprite

Click on the ***Green*** *Flag* at the top right-hand side of the computer screen and see what happens.

1.6.3 A MOVING SPRITE

Now we want to get the sprite moving.

Go to the ***Motion*** folder and select the *Move* block. Change the number of steps to 50.



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

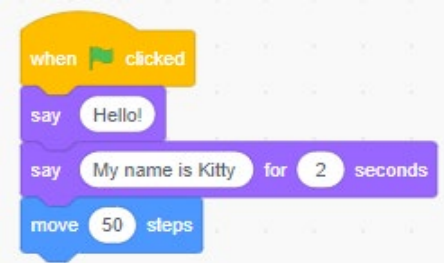
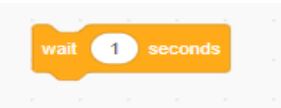


Figure 48: Adding motion to the sprite

To increase movement of the Sprite and to give the impression of walking, 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 our cat back to the center when (s)he reaches the edges of the stage!

The screen is divided into **X (horizontal)** and **Y (vertical) coordinates.** the center of the screen being (X) 0 (Y) 0 and the numbers being positive or negative depending on their positioning.

The coordinates for the centre of the screen are X(0) and Y(0). A positive X value means that an object will move to the right, a negative X value brings an object to the left. A positive Y value makes an object go up. A negative Y value makes an object go down Figure 49.

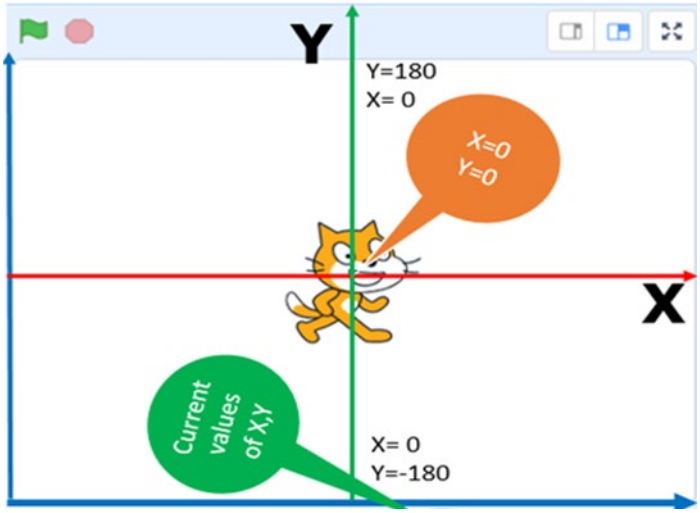


Figure 49: A sprite at coordinates (0,0)

Move the sprite around the screen and observe as the values of the X and Y coordinates change just below the sprite icon at the bottom right side of the Script area.

This function allows the user to position different sprites at different locations. Hence, we can place a script 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 we select the Green Flag.

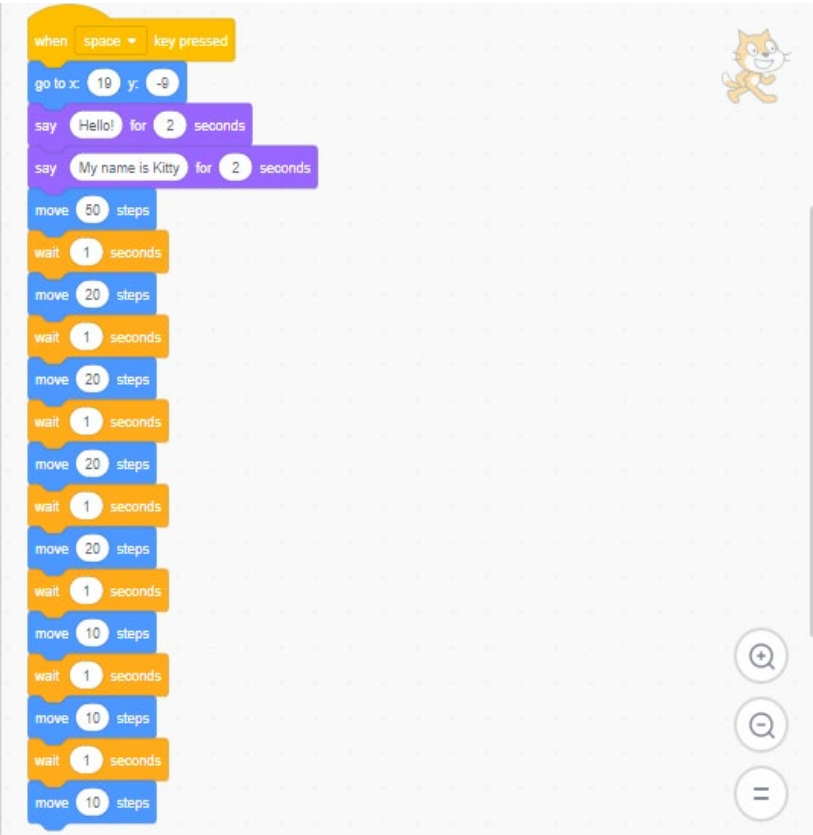
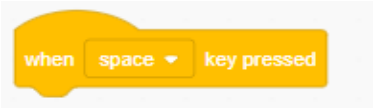


Figure 50: Script with additional blocks to make the sprite move

Test the effectiveness of this new 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 event folder. For instance, when the Space Bar or Arrow Keys are hit.

Replace 

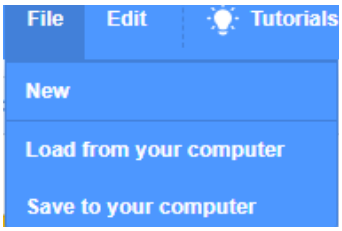
with 

(From the *Events* folder) in the Script.

Now *click the Space Bar* on the computer keyboard to start the program.

Saving the project

Go to the File folder



Select ‘Save to your computer’.

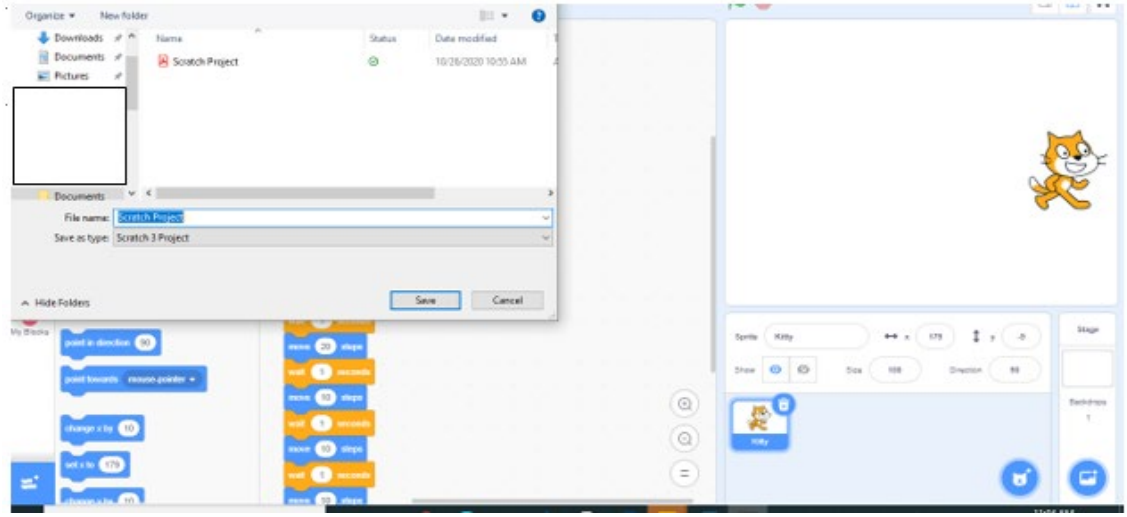


Figure 51: Saving your project

Type in *Moving cat* or your own name for your cat icon in the section labelled “*Filename”.*
Then choose where you want to store it, such as *Desktop* or *My Projects* followed by selecting the save icon.

1.7 PARALLELISM

1.7.1 PARALLELISM WITH ONE SPRITE

Parallelism means getting two or more things happening at the same time. For example, what changes when you split a simple sequence into two programs executing in parallel?

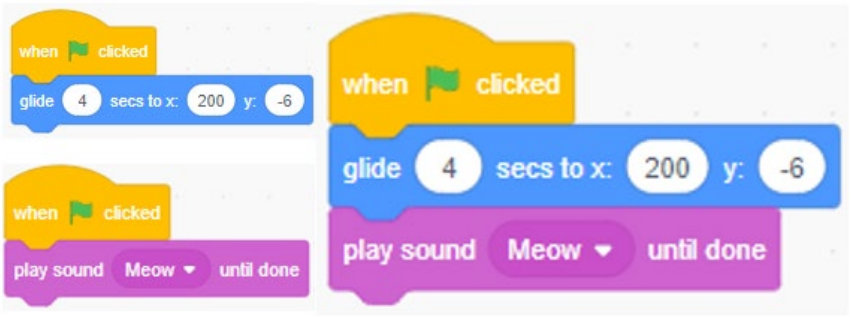


Figure 52: Parallelism – Two Sprites

Follow the steps:

- Add another sprite to your stage
- Create some code to make your second sprite do something
- Make sure you execute the code for the second sprite using the same control as the first sprite

- You can keep adding sprites and putting code into each one, as long as they have the same control to start their activities the code executes in parallel.

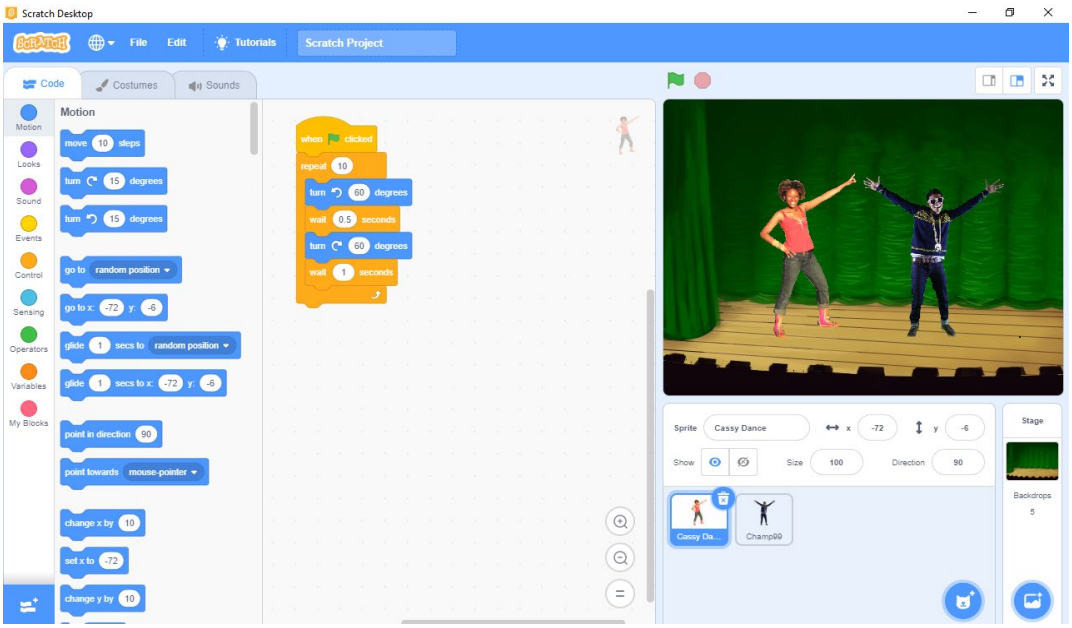


Figure 53: Example of parallelism: Two sprites on the same stage

1.7.2 PARALLELISM WITH TWO SPRITES

Make a copy of the blocks of code to use it with the second sprite. Go to the upper block 

and then right click. You will get a dropdown list of items. Select “Duplicate”.

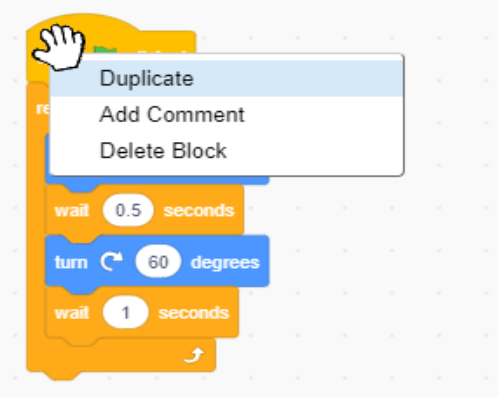


Figure 54: Using duplicate to create a copy of blocks of code

When you select **Duplicate** you will get a new group of blocks, which is the same as the old set of blocks.

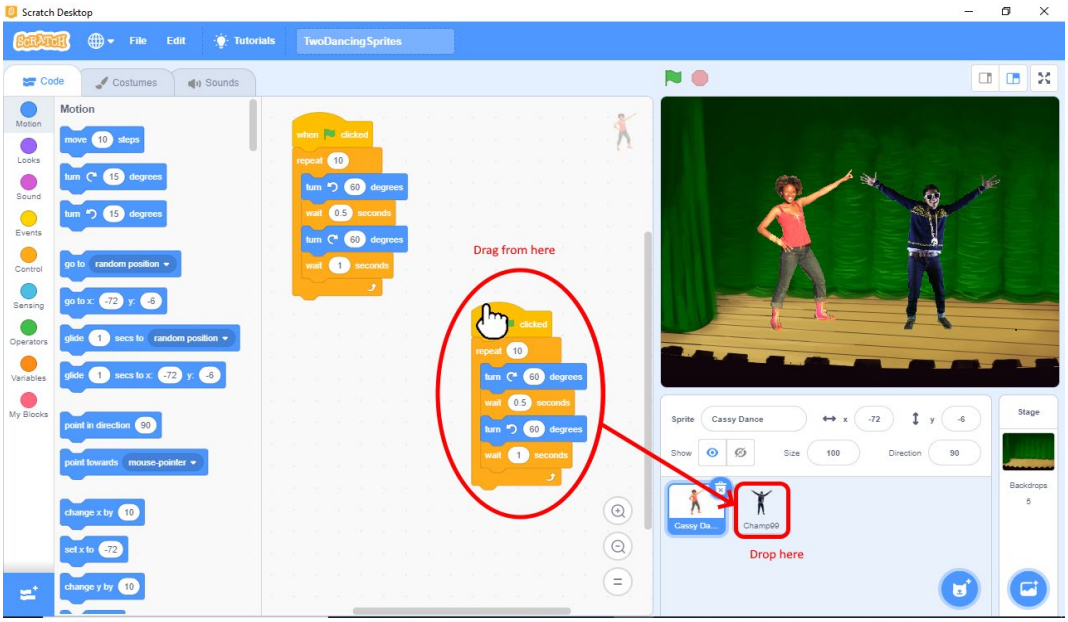


Figure 55: Drag to duplicate the blocks of code to the second sprite

When you drag the duplicate to the second sprite, you are giving it the same instructions and code. You can see that all sprites are doing the same thing at the same time. However, you can also change some

of the instructions or values in the duplicated block. Parallelism is when two sets of instructions **run at the same time**. However, these instructions can be the same or different.

END OF MODULE 1 ACTIVITY

Activity: Conversation with a guest.

Perform the following instructions for a short conversation with a user.

1. Ask a user to enter his name whenever the sprite is clicked
2. Store the name in the **answer** block and **my name** variable and display both on the stage.
3. Great the person using the name entered for example.
   *Good morning **Mary.***
4. Create a variable **adult age** and set it to 18.
5. Ask the person his/her age and store the value in **my age** variable and display it on stage.
6. If the entered age is greater than 18, give this message
   My name, you are an adult at **my age**.
   Ex. Mary, you are an adult at 21.
7. If the entered age is less than 18, give this message
   My name, you are young at **my age**.
   Ex. Mary, you are young at 12.
   After completing the exercise, please use this link to compare your answers.

MODULE 2

MOTION AND DIRECTION IN AN XY COORDINATES SYSTEM

OVERVIEW

This module will equip you with the skills to control motion and direction of objects in the XY Coordinate system. You will identify the boundaries of X Axis and Y Axis. The module will also help you to design and interact with custom stages.

LEARNING OUTCOMES

By the end of this module, you will be able to:

1. Understand the XY coordinates system and the stage size.
2. Customize the stage.
3. Control motion and direction in the XY coordinate system.
4. Debug a program that controls motion and direction using XY coordinates.

2.1 INTRODUCTION TO STAGE SIZE

The **stage** is where you see your stories, games, and animations come to life. Sprites move and interact with one another on the stage. The stage is 480 units wide and 360 units tall. It is divided into an x-y grid.

The screen is divided into X (horizontal) and Y (vertical) coordinates based on the centre of the stage which is at x-y coordinate (0, 0) and the numbers being **positive or negative depending on their position.**

This means the lower left is at (-240, -180), the upper left is at (-240, 180), the upper right is at (240, 180), and the lower right at (240, -180). To find out x-y positions on the stage, move the cursor around and look at the x-y display just below the stage on the right (Figure 56).

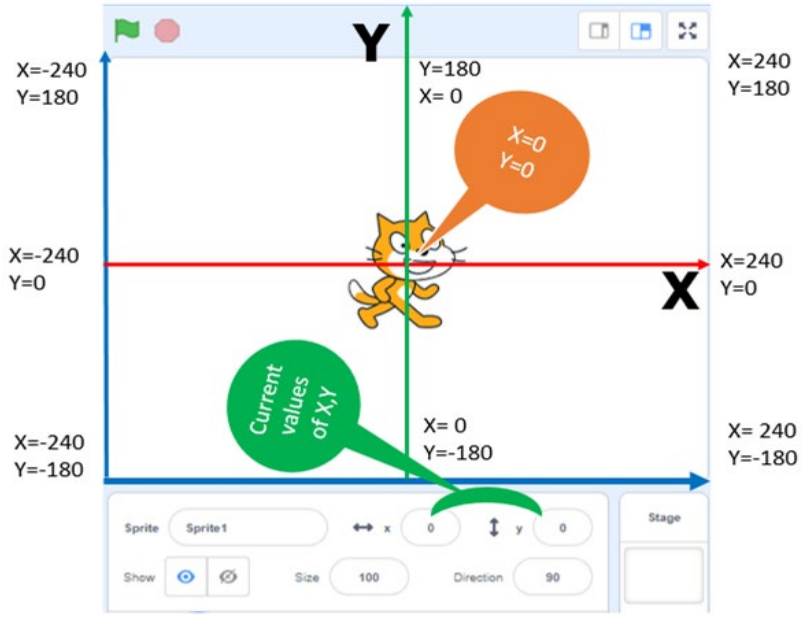


Figure 56: XY Coordinate system with (x,y) values at different positions

2.2 GEOMETRY AND PIXELS

A computer screen or a picture is made up of tiny dots known as **pixels.** X and Y coordinates are respectively the horizontal (X) and vertical (Y) addresses of any pixel or addressable point on a computer display screen.

In Scratch, one unit of X or Y, or one step is equal to one pixel. 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.

To familiarize yourself with **XY gridlines**, follow these steps:

Step 1: Click on the button “Choose a Backdrop” as shown on the figure below.

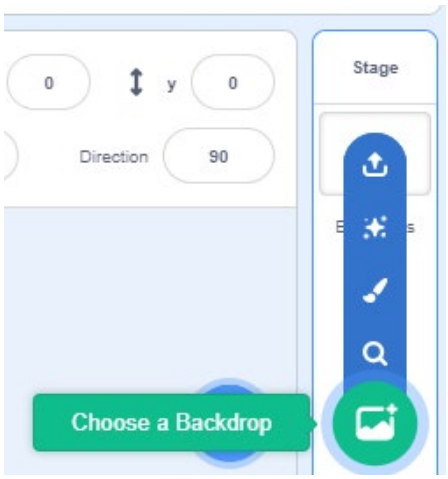


Figure 57: Choosing a backdrop

Step 2: In the search box of “Choose a Backdrop” window type “xy-” without quotes to filter all the backdrops starting with “xy-”. Three backdrops will be listed. Choose the one named “Xy-grid” by clicking on it.

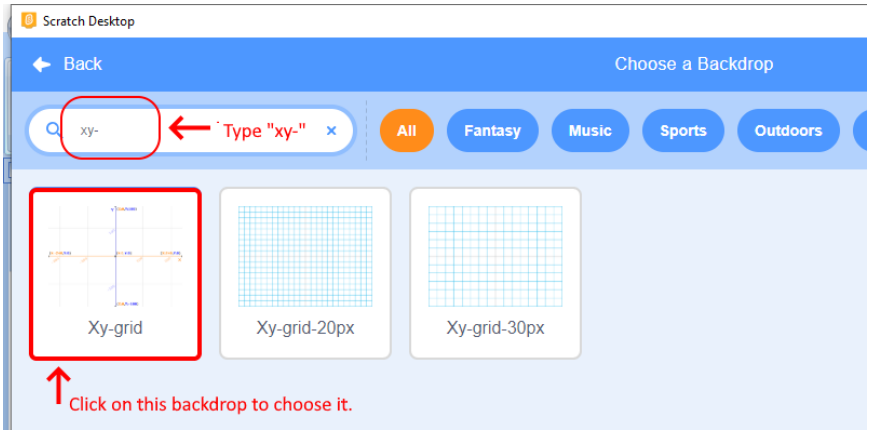


Figure 58: Choosing a new backdrop

You will get the stage of XY-grid with a sprite in the centre.

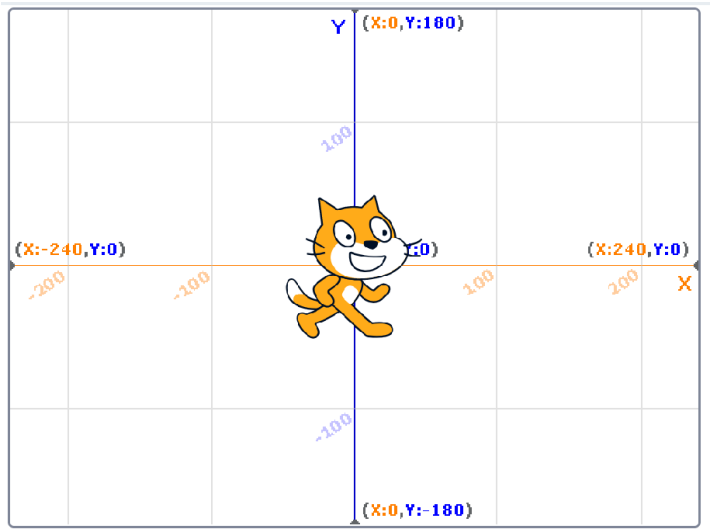


Figure 59: XY-grid backdrop

Step 3: To observe well the XY-grid stage you can hide the sprite as shown below.

Figure 60: Hiding the sprite

Step 4: You can go back to the collection of backdrops and choose the **xy-grid-30px** (Figure 61).

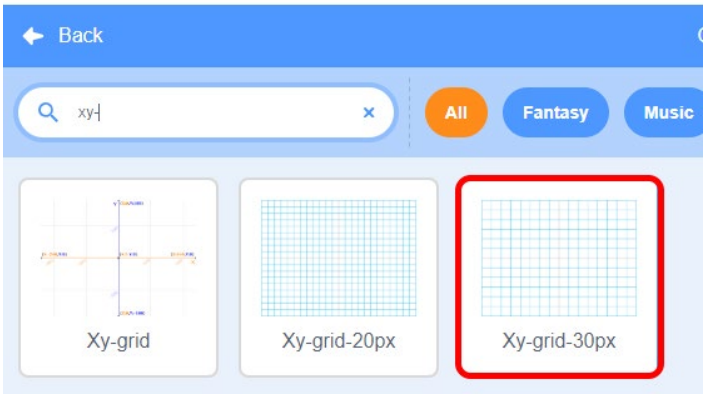


Figure 61: Choosing a new XY-grid-30px

The result is an XY-grid that consists of small squares of 30px by 30px.

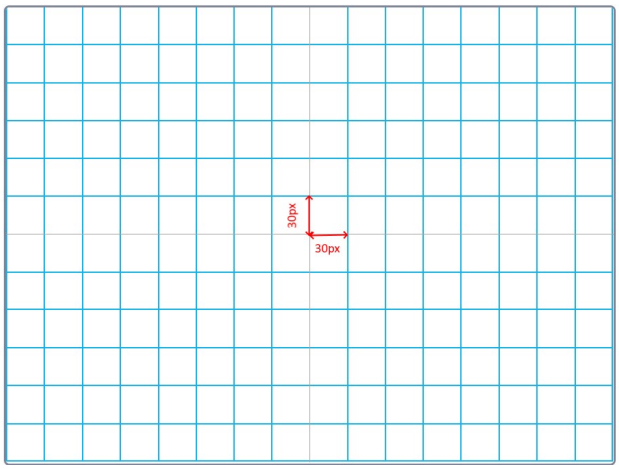


Figure 62: XY-grid with interval of 30 pixels

2.3 MOVEMENT ALONG THE X AXIS AND Y AXIS

Activity

Step 1: Go to the Backdrop section and import Bedroom1.

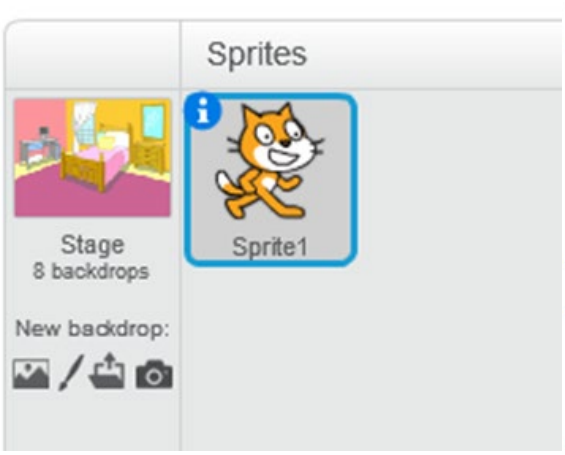


Figure 63: Bedroom back in stage

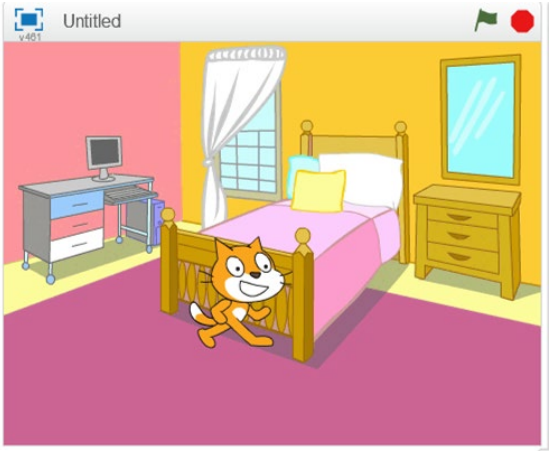


Figure 64: New Backdrop section

Step 2: Move the sprite to the top right-hand corner of the stage.

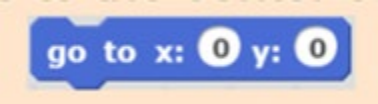
The X and Y coordinates for the sprite will appear under the faded sprite icon located at the top right-hand side of the script area.

2.3.1 Go to XY Block

The "GO TO XY" block allows the sprite to jump to the indicated XY position from its initial position.

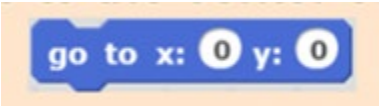
Examples:

Make the sprite go to the centre of the stage use the following block and coordinates:



Make the sprite go to the lower left of the stage, use the following coordinates:

Activity

1. Move this block from the Motion tab into the Script area and position it directly under the first (When Green Flag clicked) block in the set of programme instructions.
2. 

3. Fill in the X & Y coordinates with (-200,100)
4. Click the Green Flag icon to start the programme
5. Add event "When clicked"

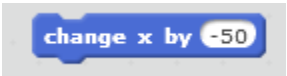
Script:

2.3.2 "Change X By" and "Change Y By" Block

The "change x by" block

The "change x by" block changes the value of X. Positive numbers move X to the right and negative numbers move X to the left.

The above will move x position to 10 pixels or units to the right.



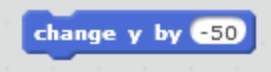
The block above will move the X position 50 pixels to the left.

The "change y by" block

The "change y by" block changes the current value of Y. Positive numbers move Y to the top and negative numbers move Y to the bottom.

The following will move Y position 10 pixels to the top

The following will move Y position 150 pixels to the bottom

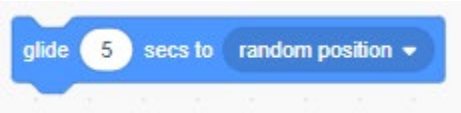


2.3.3 Set X and Set Y Blocks

Differently to change x and change y. The SET X and SET Y will initialize or reset to X and Y to the values provided.

2.3.4 The "Glide to x-y" BLOCK

The glide block is a motion block that allows a sprite to move steadily from one place to another. With glide blocks, you can move your character or sprite from places to new positions on the stage as you like. You can set the speed and the direction of the sprite movement using the glide blocks.



Activity: **Moving a sprite using motion block**

Step1: Place your sprite at any position and go to the glide block in the motion folder.

Step2: To glide, set a location where you want your sprite to move to, drag the sprite to that new location and drag out another glide block and put it just under the first block.

Step3: To set a starting point, place the character where you want to start, and drag out a "**Go to**" block and put it on top. Finally, add a green flag on top to start the script

Follow this link to get a hint.

Activity: moving a sprite using XY position

1. Choose a sprite, and use the glide blocks, make it move from (x:-117, y: 2) to (x: 122, y: 9) then to (x:-122, y:-70) and back to (x:-117, y: 2). Note that each move should have 3 seconds. Save your work. Follow this link to get a hint.

2. Choose the Ballerina Sprite as shown below and place it on the stage wherever you want. Make the sprite jump up from the original position and back to its original position using the Glide blocks. Save your work.

Figure 65: Sprite to move using XY glider

Use this link to get a hint.

2.3.5 The Motion Blocks

You can use a set of blocks from the motion palette to turn, move and glide your sprite on the same stage. The "Turn" block allows the sprite to turn clockwise or anticlockwise at any specified degrees. The "Move" block allows the sprite to move forward from place to another in terms of the number of specified steps. A step is equal to a one-pixel length. The Default Value is 10 and it can be replaced by any number.

Follow the instructions below and observe what happens at every stage.

Figure 66: Use of Turn blocks

2.3.6 The Rotation Block

The **rotation style** is a motion block in Scratch which changes the rotation of the sprite. It allows 3 types which are (1) rotation from left to right, (2) rotation all around and (3) no rotation. The figure below shows blocks that you can use to set your rotation style. This block is called “Set rotation style”.

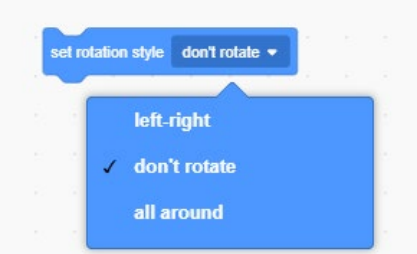


Figure 67: The Set Rotation Style block

Example

Rotation of a sprite in the opposite direction by using the rotation style block

Step 1: set the event

Step 2: put the rotation block

Step 3: Add a return function to allow it to rotate. The **return function** needs to have a value of more than 90 degrees.

That is how you can set the rotation in an opposite direction or to face any angle.

Figure 68: Rotation of the Cat

Another option to rotate a sprite is by clicking on the sprite and selecting *directions*, from there, you can rotate to any angle that you want.

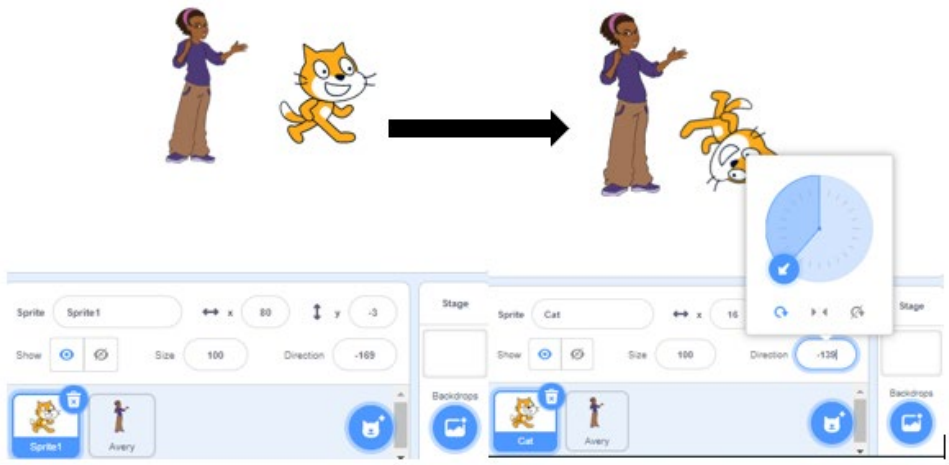


Figure 69: Rotation of the Cat sprite with different angle

The keyboard can also be used to direct and control your sprite in Scratch. This is done by motion blocks and event blocks. You can set coordinates to move a sprite to a given position and you can also use keys to set the direction of any element.

Figure 70: Use keys to set the direction of any element

When you are controlling the sprite using the left and right arrows, the sprite is moving along the X-axis. To move the sprite upwards and downwards, you use the up and down arrows to move along the Y axis. Please look at Figure 70 for an example. Conditions can be set to any key.

There is another option that would allow us to rotate a sprite and animate it using costumes. This is done by creating different costumes and giving each costume its own behaviours. After setting up those behaviours, give conditions must be seton how each event is going to follow on each other.

Example

To make the sprite rotate to different positions using change of costumes.

I have created 2 costumes for this sprite. I want it to turn up and down whenever I click to different keys. I have created 2 costumes and when I click on Space, the first costume will show and when I click on Green Flag another costume will show.

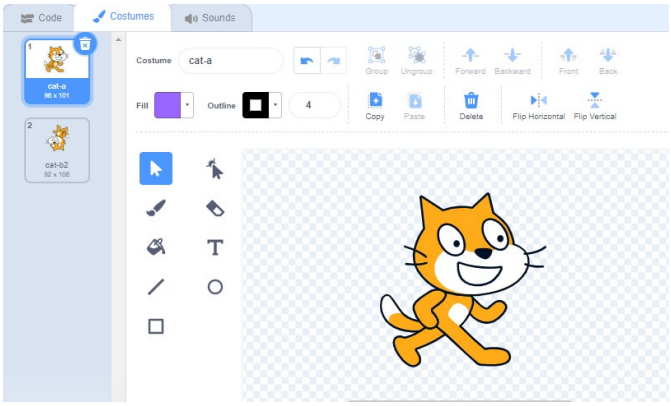


Figure 71: Two costumes for the cat sprite

Step 1: You start with setting the event.

Step 2: Add “Forever” loop and an “IF” condition.

Step 3: Add a key that is going to be clicked to trigger the event.

Step 4: Set what you want to happen when that key is clicked.

If you followed the steps correctly, it should look like this:

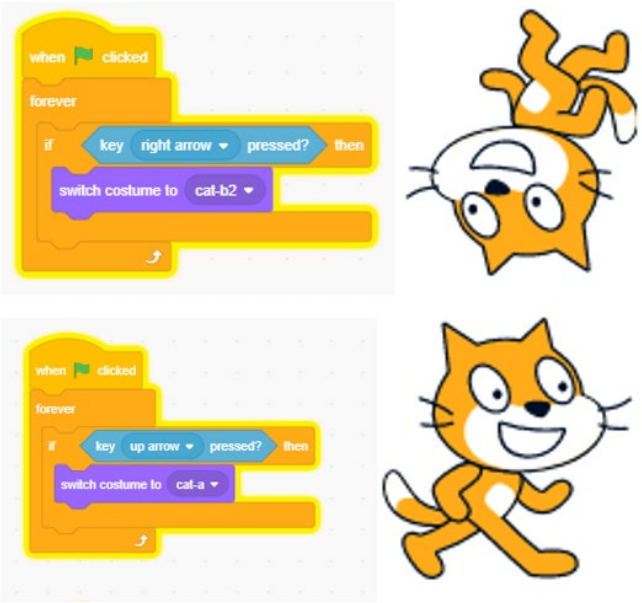


Figure 72: Cat rotation using change of costumes

2.3.7 The Reporter Block

Reporter blocks help in directing a sprite. A value can be defined into them Reporter blocks must be placed into a number, text, or drop-down input. When Scratch runs a block that include a reporter block, it will first run the reporter block to find its value.

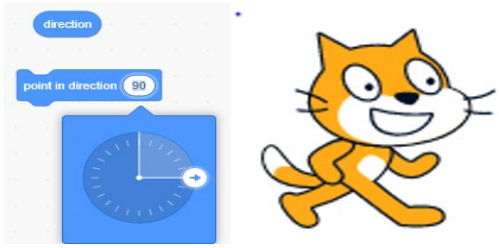


Figure 73: Sprite position before setting degrees

In Figure 74, the sprite is set to rotate on an angle of 135 degrees. To do that, you first insert the direction block and then define the degree you want the sprite to rotate.

Figure 74: Rotating sprite at an angle of 135 degrees

END OF MODULE 2 ACTIVITY

Using XY coordinates and the motion control palette, animate the locomotion of the following animals: Cheetah, penguin, duck, horse, donkey, gorilla, sheep, and cow.

Have a look at the example of the locomotion of the frog, man, bear, snake.

Use the following instruction to move each animal.

1. Press **b** to move the bear
2. Press **f** to move the frog
3. Press **m** to move the man
4. Press **s** to move the snake

Locomotion link: https://Scratch.mit.edu/projects/449984077/

MODULE 3

CREATING STORIES AND ANIMATIONS IN SCRATCH PART I

OVERVIEW

In this module you will learn about storytelling using computer design activities. You will work together and build on the creative work of each other. You will get familiar with concepts introduced in module 1 regarding the use of many sprite in the same scene, namely events management, parallelism, concurrency and iteration. The module begins by code conversation, and then combines the characters' conversation and their animation within the backdrops in a larger story project. You will learn how to work with different sprites, animate them in different backdrops and manage the timing of different events.

LEARNING OUTCOMES

By the end of this module, you will be able to:

- use computational concepts in storytelling (parallelism, events management, and iteration).
- work with different sprites and manage the timing.
- animate sprites and manage different backdrops in the same project.
- debug the program that contains different blocks for a story.

3.1 INTRODUCTION

In this module we will use different sprites to create a story. In module 1 we have seen how to select a sprite to use in Scratch project and how you can create your own sprite. Revisit Module 1 if you have difficulties in adding sprites, modifying their costumes, or creating your own sprites.

3.2 WORKING WITH DIFFERENT SPRITES

The cat is the default character when starting a new project, but you can remove it and add any other character or sprite that matches better your story. In the following section we will see how to combine interesting images and sounds to make an interactive collage about your story.

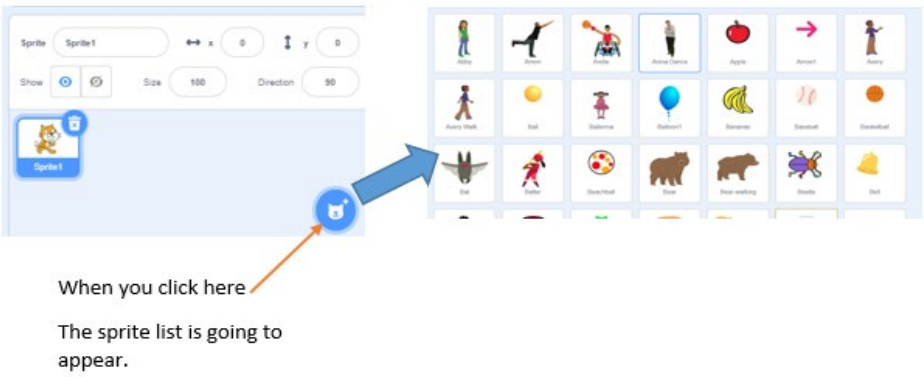


Figure 75: How to add multiple sprites

Example

In this example we will move different animal sprites on the stage:

Step 1: Create a new project and call it “jungle”.

Step 2: Click on the Backdrop icon

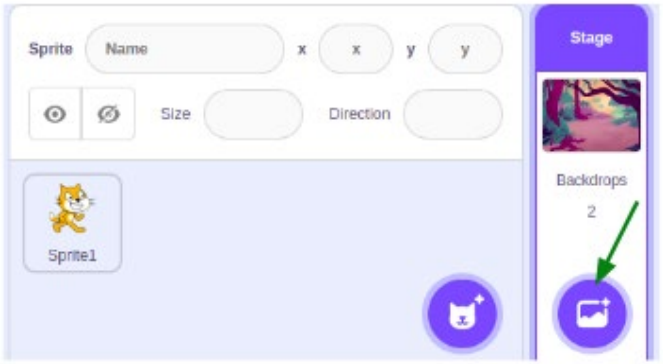


Figure 76: Backdrop icon location

Step 3: Select the jungle Backdrop from the Outdoors collection.

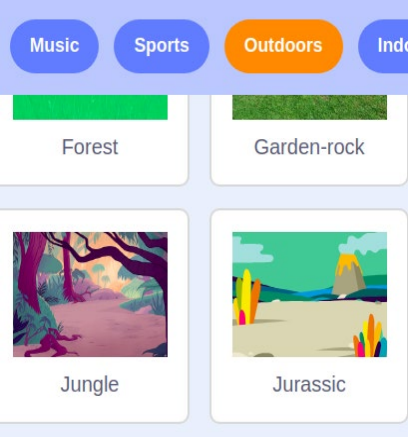


Figure 77: Backdrops

Now we have the Stage with the cat in the Jungle.

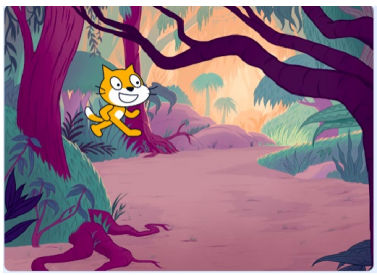


Figure 78: Cat in the Jungle

Step4: Duplicate the cat three times.

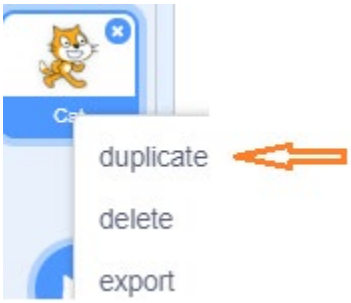


Figure 79: Duplicating a sprite

Step 5: Add the sprites named “Elephant” and ”butterfly1”.

Figure 80: Animal sprites

Step 6: We have different sprites on the stage.

Figure 81: Different sprites on the stage

Figure 82: Different sprites icon under the stage

3.3 ANIMATING DIFFERENT SPRITES

Our next action is to make the sprites we have on the stage interactive. We want them to respond to clicks, key presses and more! Using the previous example, move to the next steps:

Step 7: Apply the following group of blocks. These blocks will make the cat move "forever" when the space key is pressed.

Figure 83: Scripts to move the sprite in the jungle

Step 8: Using the concept of multithreading from Module 1, drag and drop the same blocks to the other sprites.

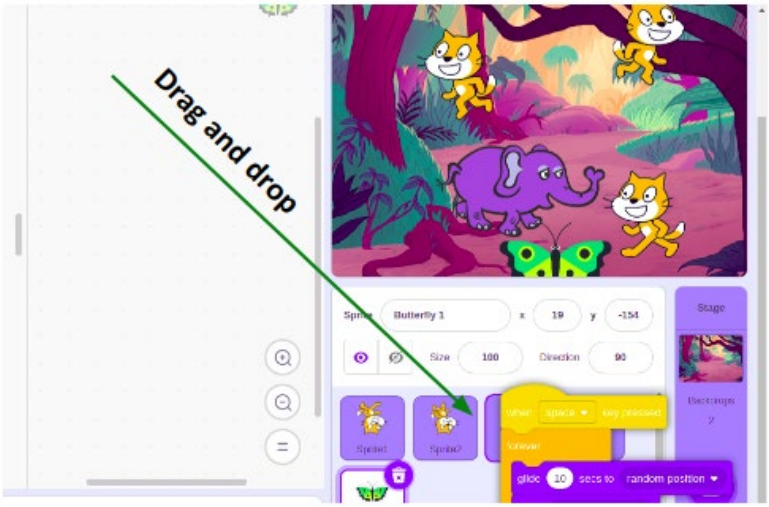


Figure 84: Dragging scripts to another sprite

Step 9: Press the **space key** to move different sprites in the jungle. An example is found under this link: **https://Scratch.mit.edu/projects/439108501/**

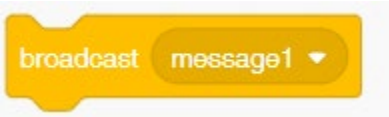
Step 10: Try changing some values (such as the glide time) for the other animals and see how this impacts your animation.

3.4 COMMUNICATION BETWEEN SPRITES

In this lesson, you will gain experience in synchronizing interaction between sprites. To enable two sprites to communicate, we will use **Sensing and Broadcast commands**. Broadcasts send information in other blocks of code. In addition, you will be more familiar with the use of events and parallelism gained in module 1.

3.4.1 Broadcast command

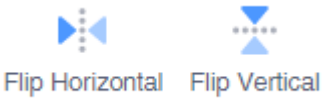
The command which makes two sprites communicate with each other is "Broadcast". Broadcast is used to send a message:



The response of the recipient of the message is triggered by the block: "When I Receive":

In the following exercise, we will make two sprites have a simple chat, by using the blocks Broadcast and When I Receive.

1.Add a bear and a beetle from the sprites overview. If you want to change their position, click on



Costume tab, and select according to the position you want.

2. Add the following code to the Bear sprite:

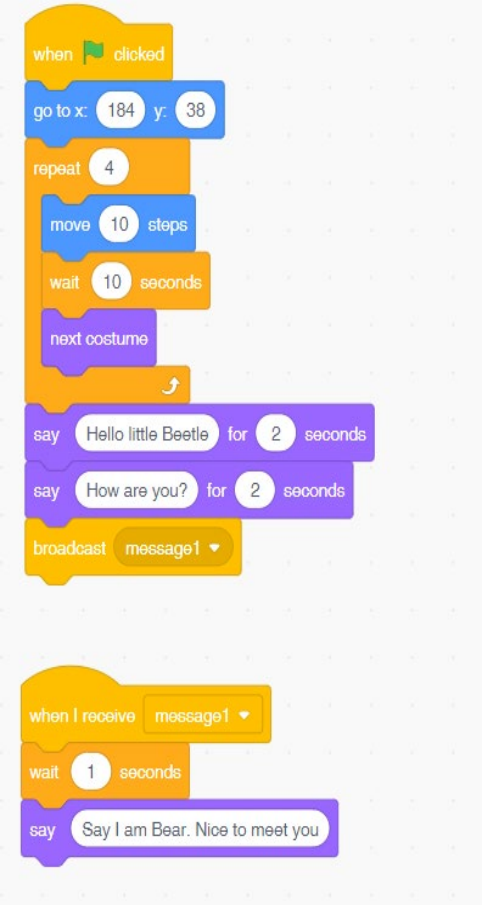


Figure 85: The script for the bear

3. Add the following code to the Beetle sprite:

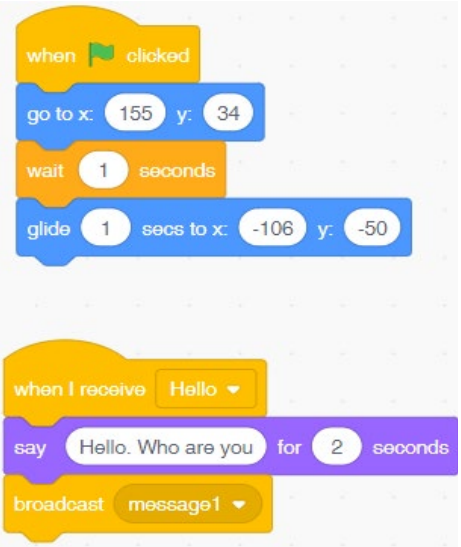


Figure 86: The script for Beetle sprite

4. Run the program. You should get the following result:

Figure 87: Bear broadcast message to Beatle and Beatle responds

3.5 STORY CREATION

This part focuses on helping you to develop storytelling skills through a variety of activities, providing opportunities to work together and build on each other's work.

Building on initial experiences, the activities in this unit are designed to help you develop fluency in the computational concepts of events and parallelism and the computational practices of experimenting, iterating, reusing and remixing. Each activity is designed to help you build storytelling projects by discovering new blocks and methods for programming interactions between sprites and backdrops.

3.5.1 Creating Blocks for Different Characters

Scratch allows you to create your own blocks. You can do this by using the Make a Block feature. In the following project you will learn to create your own blocks that define two behaviours for two different characters.

Activity: Creating custom blocks

Step 1: Add two Sprites to your project, either by selecting them from the sprite library or by uploading them.

3. 14

Figure 88: Jumping Sprite

Step 2: Click on the Make a Block button in the My Blocks category to create and name your block.

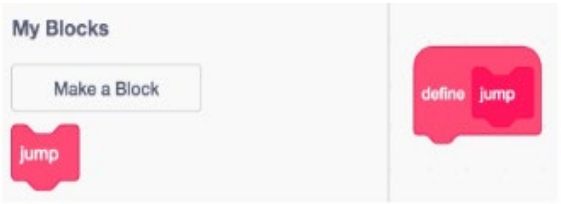


Figure 89: Jump block creation

Step 3: Add blocks under the *Define* block to control what your custom block will do.

Figure 90: Jump block definition

Step 4: Experiment with using your block to program your characters' behaviour.

Figure 91: Use of the Jump block

Step 5: Repeat

3.5.2 Conversations

What are different ways to coordinate interactions between sprites?

In this activity, you will explore different options of making a sprite have a conversation.

The remix function is added to Scratch so that people can experiment with other people's projects. For example, you can start from someone else's project and you find something to improve, instead of creating your own story, you could just add what you want, and remix! Keep in mind that the purpose of remixing is to add/change something, not just copying someone else's work!

Activity: Remix the Penguin Story

1. Visit the Penguin Joke starter project in the link below to observe how a conversation can be animated using wait blocks.

2. Use the **remix function** and redesign the Penguin Joke project to coordinate the conversation using the *broadcast*, *broadcast and wait*, and *when I receive* blocks. (http://Scratch.mit.edu/projects/10015800)

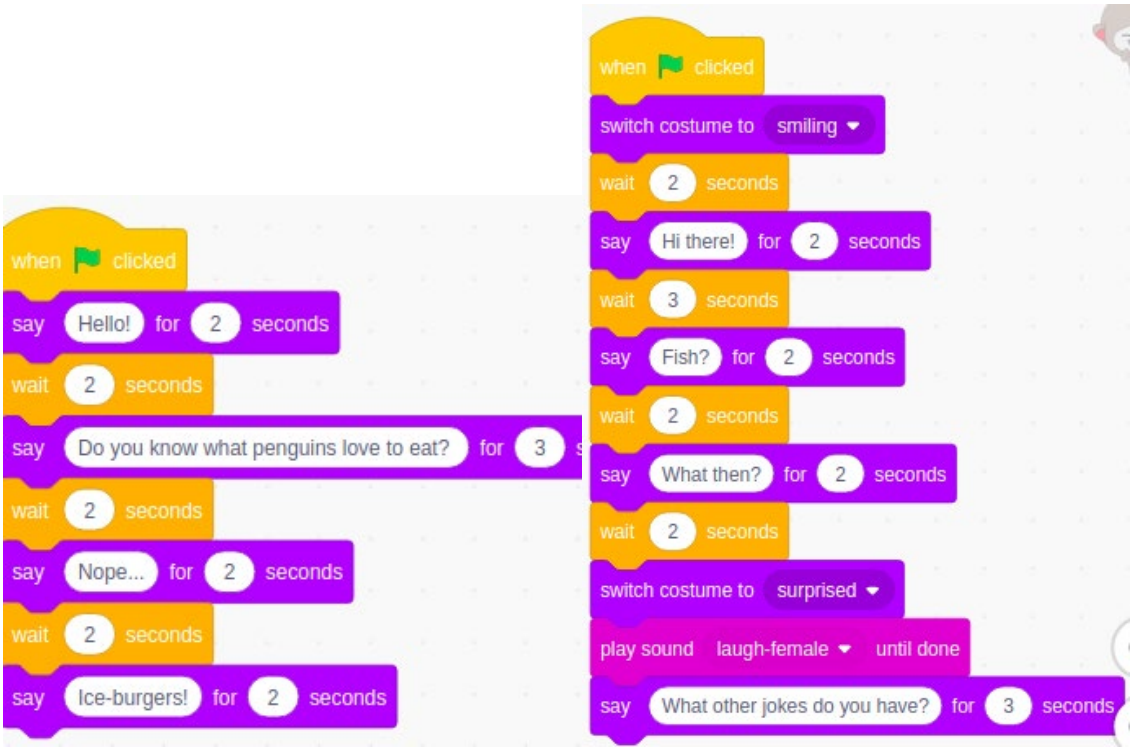


Figure 92: Penguin and Puppy blocks

3. Rename each message from Penguin and broadcast the message name using *broadcast and wait* block from the event palette.
4. Instead of using wait for x number of seconds, enable the puppy to respond when it receives the message from the penguin using "*when I receive"*.

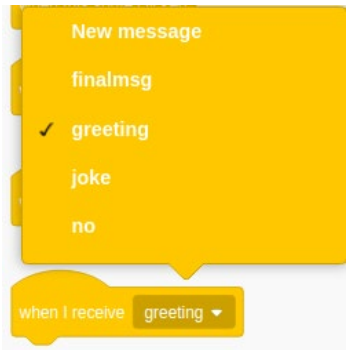


Figure 93: Using the "When I receive block"

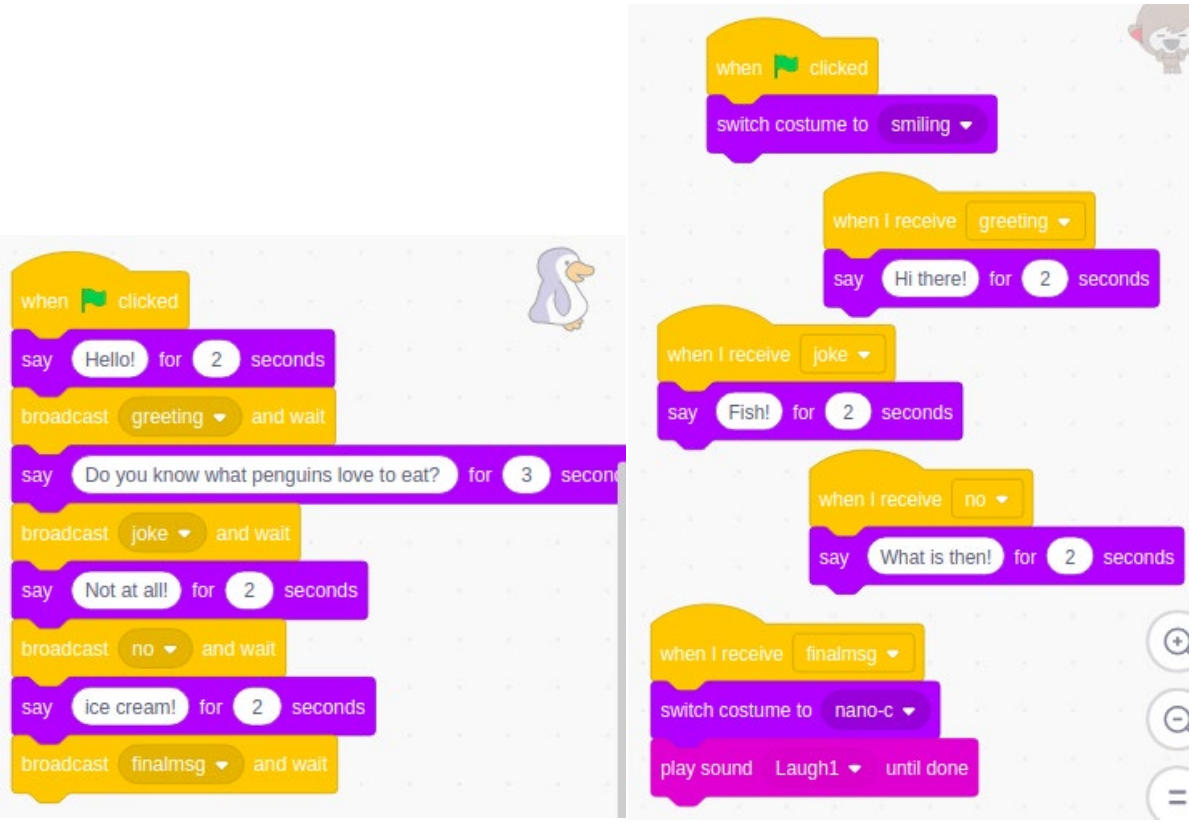


Figure 94: Penguin and Puppy conversation after using remix functions

Have a look at this example*:* *https://Scratch.mit.edu/projects/449725031/*

3.5.3 Switching between Backdrops

In this activity, you will create a project that experiments with backdrops, like a story with multiple scenes or a slideshow. The activity shows how different backdrops are switched while the sprite is on stage.

Follow this guide:

1. Add multiple backdrops to your project
2. Add a sprite from the sprite list
3. Experiment with blocks from the Looks and Events categories to initiate switching backdrops.
4. Add scripts to the stage and sprites to coordinate what happens when the backdrop changes in your project!

Figure 95: Add backdrops

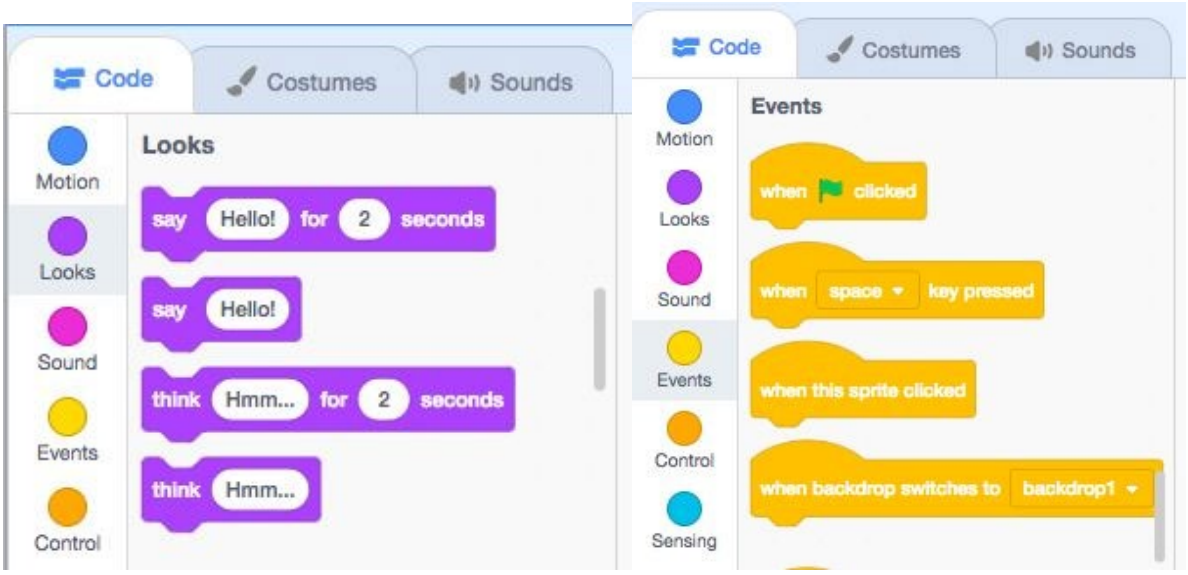


Figure 96: Use Looks and events backdrops

Figure 97: code for switching backdrops

3.5.4 Event Timing

The Timer block is a sensing block that reports the amount of time since the project was loaded or the timer was last reset. This block is almost always used with the Reset Timer block. The timer must be reset at the beginning of a project for the Timer block to hold the right value.

This block can be displayed as a Stage monitor, though it will only display intervals of 0.1. This value can be made more precise by making a script that constantly sets a variable to the timer.

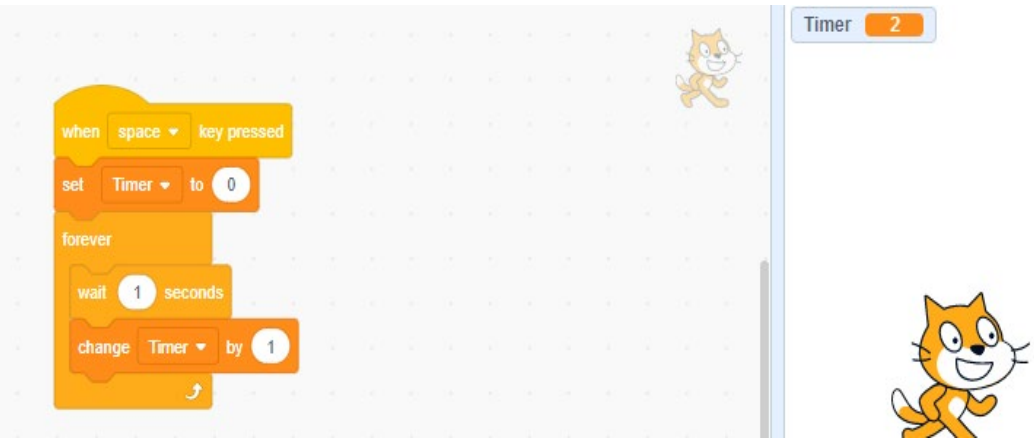


Figure 98: Event Timing

In this example, we are showing how you can set a timer to measure how much time a certain activity or a game took to be completed. We had to set an event first and we said that when you click the space key, the timer will be set to 0 and then wait for a second to start timing forever.

If you want to set a time each event should last, you need to set the timer and give it the number of seconds you want the event or the game to last.

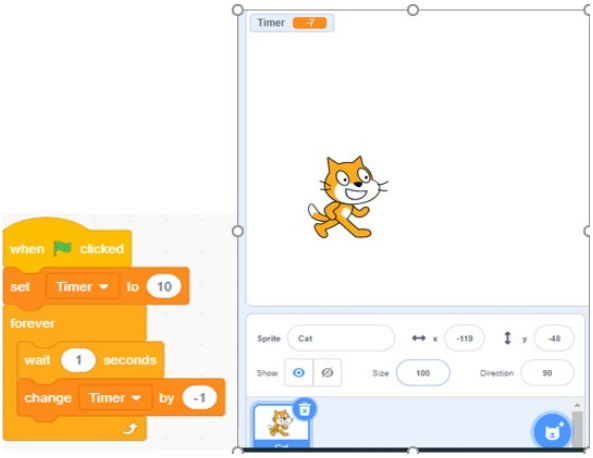


Figure 99: Timing reminder setting blocks

When you click on the green flag, the timer is going to be set to 10 seconds and then wait one second and start deducting one second from the set time.

You can also set a reminder of the time and stop the game all the event.

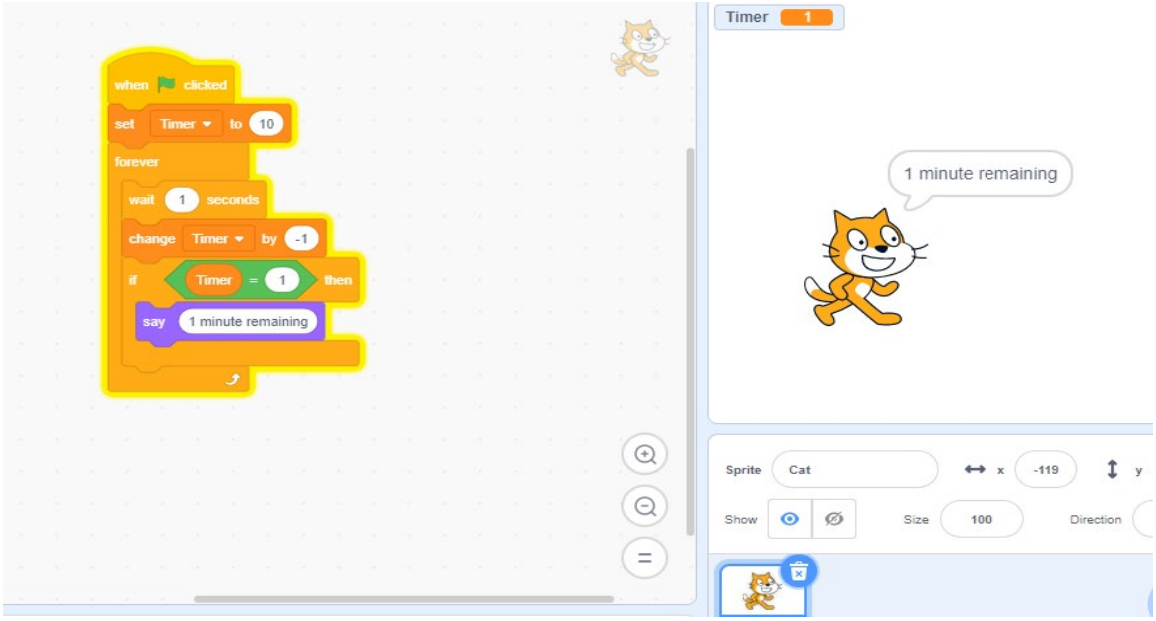


Figure 100: Timing reminder

We have added a reminder that when the time value reaches 1 minute, the cat is going to tell the player that there is only one-minute remaining. Other actions are also possible, such as stopping the game. The important thing is to know That the timer can trigger actions on a certain value.

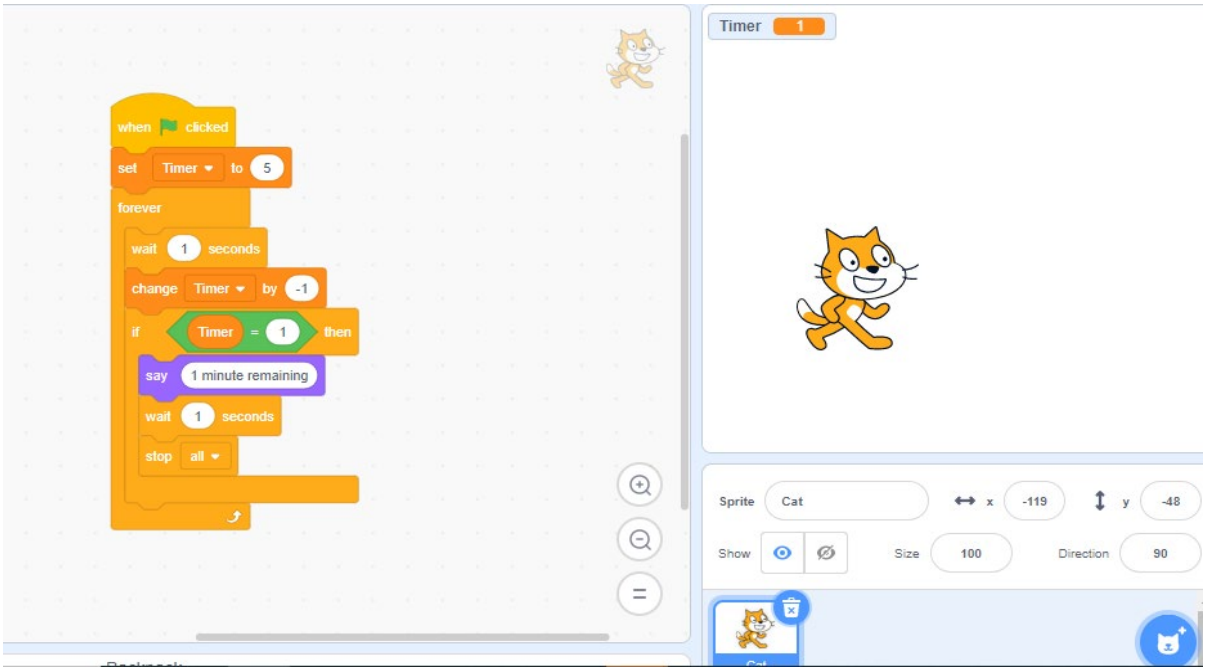


Figure 101: A scenario of stopping all activities instead of waiting

Look at Figure 101. We have added another controller that is going to stop everything when the timer reaches to 1 and wait for 1 second.

END OF MODULE ACTIVITY: MAKING A COVID 19 STORY

Design a story related to the Covid-19 pandemic. The objective of this project is to motivate your learners to respect the hygiene regulations. The conversation below is an example.

Step 1: Student Mary greets student Peter. Nice to see you in a face mask.

Step 2: Mary asks Peter how he passed the covid19 Lockdown.

Step 3: They walk in school and the backdrop changes.

Step 4: Mary replies: “It was not good because I should have finished my S6”.

Step 5: Mary makes apology with “sorry”.

Step 6: Peter also asks Mary: What about you?

Step 7: Mary replies “It was not too boring to me. Parents gave me a computer to learn online”.

Step 8: Peter replies “That’s good”.

Step 9: Mary asks Peter “Haven’t you recorded anything good during lockdown”?

Step 10: Peter replies “Washing hands several times and wearing a face mask is good for hygiene”.

Steps 11: Mary replies “Okay. Ooh you remind me to wash my hands and put my face mask.”

Step 12: Mary replies “Let us go together to wash our hands”.

Working example: https://Scratch.mit.edu/projects/449741583/

MODULE 4

CREATING STORIES AND ANIMATIONS IN SCRATCH PART II

OVERVIEW

In this module, you will learn about creating stories in Scratch animations. We will discuss the interchange between different sprite costumes, incorporating time and motion using sensing blocks, control blocks and broadcast blocks. You will also learn how to import, create, and record sounds to use in your Scratch projects.

LEARNING OUTCOMES

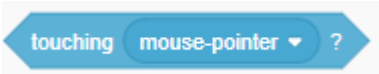
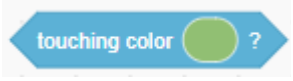
By the end of this module, you will be able to:

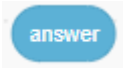
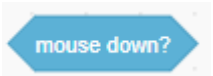
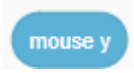
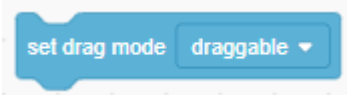
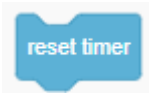
- understand the benefits of sensing blocks and control blocks while designing an animation.
- understand the importance of broadcasting while animating.
- explore computational creation within the genre of stories by designing collaborative narratives.
- Create animated stories in Scratch.

4.1 SENSING BLOCKS

Sensing blocks are light blue in colour and are used to detect different factors of a project. Sensing blocks are one of the eight categories of Scratch blocks. Sensing block is used to sense when a certain event is happening and display any other instruction. For example, it would make a sprite say "You're touching the edge" when the sprite touches the edge.

There are different elements of sensing blocks and each element has its own functionality. The table below shows…

Sensing Block	What it does
	The block checks if its sprite is touching the mouse-pointer, edge, or another sprite. If the sprite is touching the selected object, the block returns true; if it is not, it returns false.
	The block checks whether its sprite is touching a specified color. If it is, the block returns "true".

	The block checks whether a color on its sprite is touching another color. If it is, the block returns "true".
	The block reports the Euclidean distance, in pixels, between it and the mouse-pointer or a specified sprite's costume centre.
	The block will make an input box (with the specified text above it) show at the bottom of the screen. Scratchers can input text into it and submit it, and the input is then stored in the Answer block. The Answer block automatically updates to most recent input.
	The block holds the most recent text inputted with the Ask () and Wait block. When nothing has been inputted yet, the value will hold nothing.
	The block checks to see if the specified key is pressed. If the key is being pressed, the block returns "true"; if it is not, it returns "false".
	The block checks if the computer mouse's primary button is activated (being clicked).
	The block holds (reports) the mouse-pointer's current Mouse X.
	The block holds the mouse-pointer's current Mouse Y.
	The block allows the setting on which element that can be dragged or not dragged.
	The block reports how loud the noise is that a microphone receives, on a scale of 0 to 100. To use this block, a microphone must be used, and so a message will appear on the screen, asking for permission to use the microphone. If you deny it, the block will report a loudness of 0 or -1.
	The block starts at 0 when Scratch is launched and increases gradually; every second it will have increased by 1.
	The block sets the timer's value back to 0.0. When this block is present, the project typically utilizes the Timer output block;

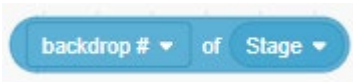
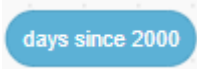
	usually the timer must be reset at the beginning of a project for the Timer block to hold the right value.
	The block will report a specified value of the specified sprite or the Stage.
	It reports either the current local year, month, date, day of the week, hour, minutes, or seconds, depending on the argument. The block gets the data based on the user's computer's clock and sets at a 24-hour clock.
	It reports the number of days (and fractions of a day) since 00:00:00 1 January 2000 (UTC).
	Display the account holder's username

Table 9: Sensing blocks

The sensing blocks allow user inputs and can make the animation change or adjust itself according to that user input. The block senses when the user has entered data. User input can be the click of the mouse, using the arrow keys on the keyboard to make a sprite move around the stage, or when a user types an answer into a question box. The sensing blocks allow the program to sense when a user input happens and how to respond to that input.

In this module we will look at a few sensing blocks in more detail:

4.1.1 Touching block

The Touching block is a combination of a sensing block and a Boolean block. The touching? Block checks if its sprite is touching the mouse-pointer, edge, or another sprite (a reporter block which returns the sprite's name, usually a variable can be used). If the sprite is touching the selected object, the block returns *true*; if it is not, it returns *false*.

The following picture shows the script of how a sprite can report back (saying that it touched an edge) when the sensing statement is true (touching an edge).

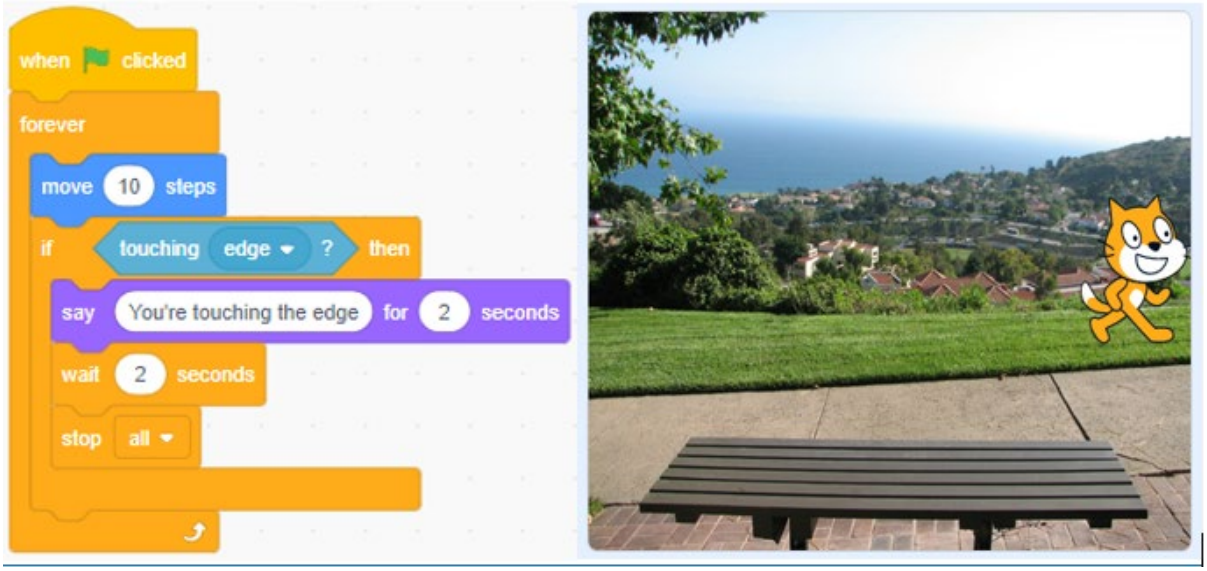


Figure 102: Sprite touching the edge

With a sensing block, you can build a script that makes a sprite respond when it is touching another sprite, the mouse position/ pointer? Or the edge of the stage and then report to anything you added on the script.

Example

The example below shows what a sprite can do when touching a mouse pointer:

- Write the Scratch programme as seen in the figure.
- Put the mouse pointer in the direction of the sprite.
- Click on space to start the script and observe what happens when it touches the mouse pointer.

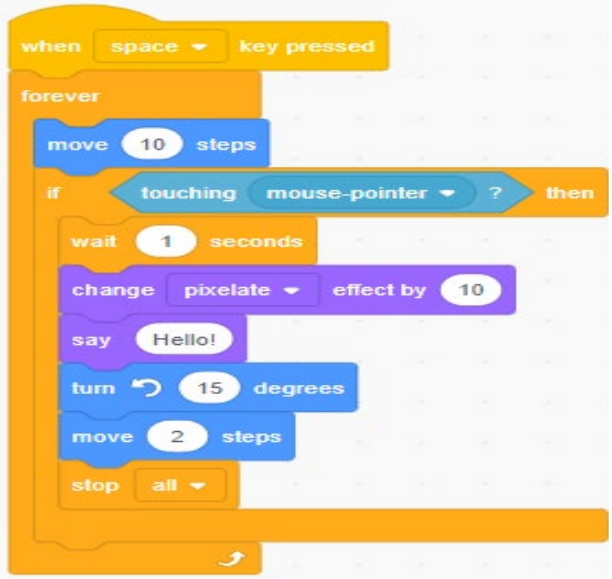
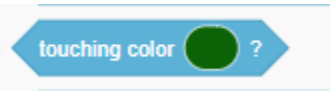


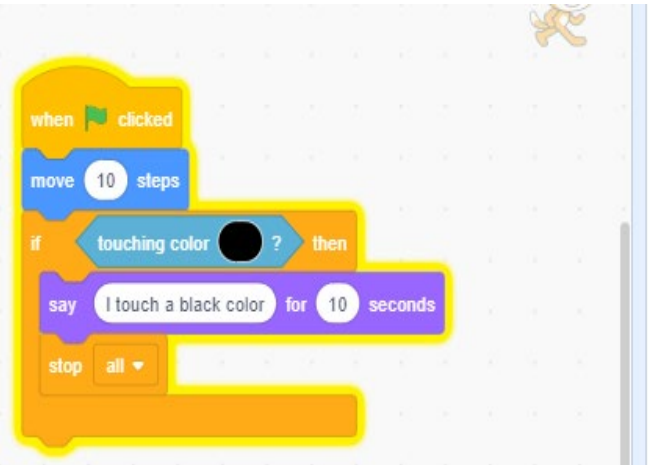
Figure 103: Sprite touching a mouse pointer

The sprite can also sense a colour. The block checks whether its sprite is touching an object with a specified colour and responds by using the following statement.



.

Example: this script is showing how a sprite responds when it touches a black object



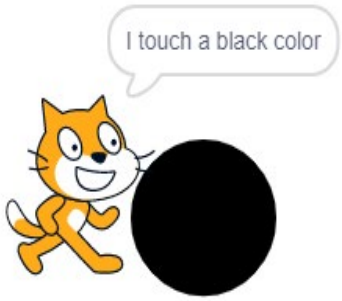
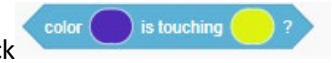


Figure 104: Script in which sprite responds when it touches a black object

The colour of a sprite can also sense another colour by using this block 

. This block checks whether the first input, a colour on its sprite, is touching another colour. If it is, the block reports true, and if not, it reports false. Both colours need to be the same for the sprite to return true!

The example below shows what a sprite can do when touching a particular colour. If the script touches the colour red it will go to position x:5, y: -31

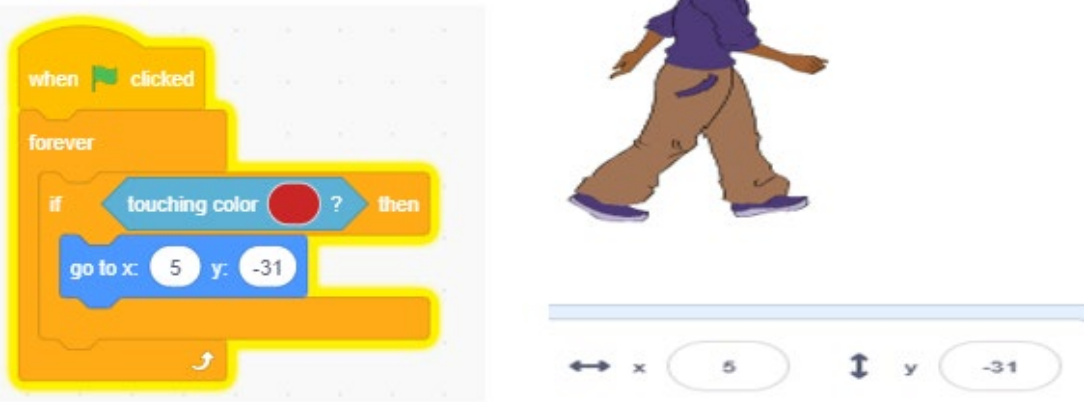
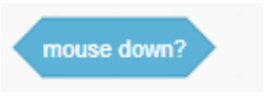


Figure 105: Script and results when a sprite touches the 5X and -31Y

4.1.2 Mouse down block

The **mouse down block** checks if the mouse is being pressed. It is a useful block as a substitute for click detection. It is frequently used in one sprite one script projects. If the project requires clicking, this block can be used to serve as a replacement for the "when clicked" block, as the hat block cannot be used in the middle of a script.

A common use for the mouse down block is sensing.



This sensing block will respond when the right mouse button is clicked. This means that you can have a sprite respond to user interaction. This is another example of **event driven programming.** Event driven programming is a programming model in which the flow of the program is determined by events such as user actions, sensor outputs, or messages from other programs. The script will execute when the mouse button is clicked.

Example

In this example the sprite will say “you touched me” and turn 15 degrees every time you click on the sprite.

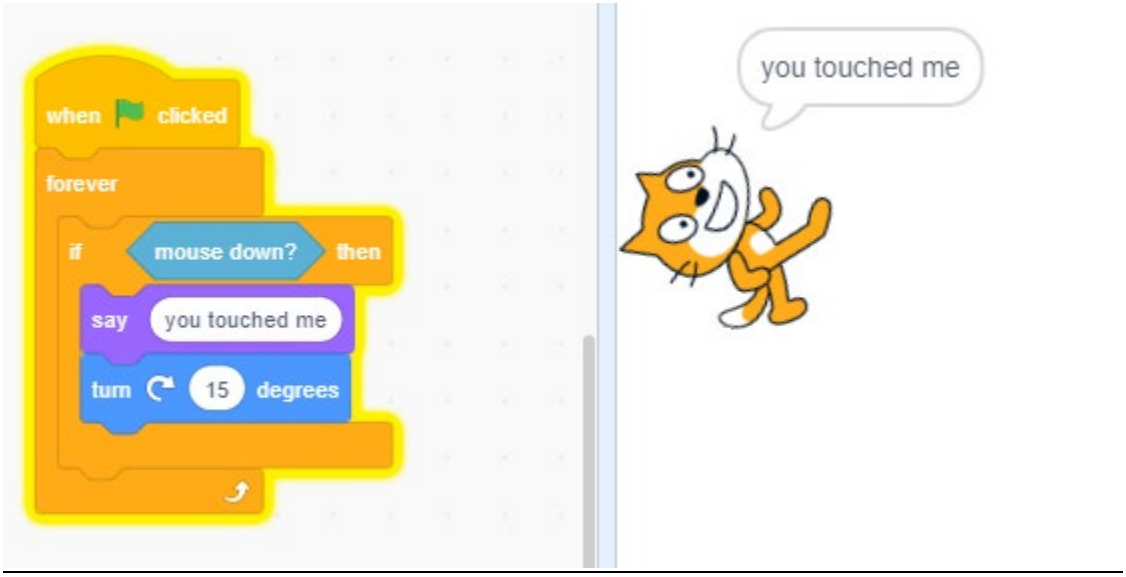


Figure 106: Script where the Sprite says “you touched me”

4.1.3 Follow the mouse pointer

With the “sensing” block, you can use the pointer to control the movement of the sprite. The code on the left will continually have the sprite go to the location of the mouse x and mouse y co-ordinates. We use a “**forever loop”** to make sure that the sprite continually follows the mouse pointer.

Figure 107: Sprite follows the mouse pointer (cursor)

4.1.4 Key is pressed

This block can be used to detect when the user presses a specific key. It can be any letter, a space, arrow…. or any other key on the keyboard. 

4.1.5 When 2 sprites collide

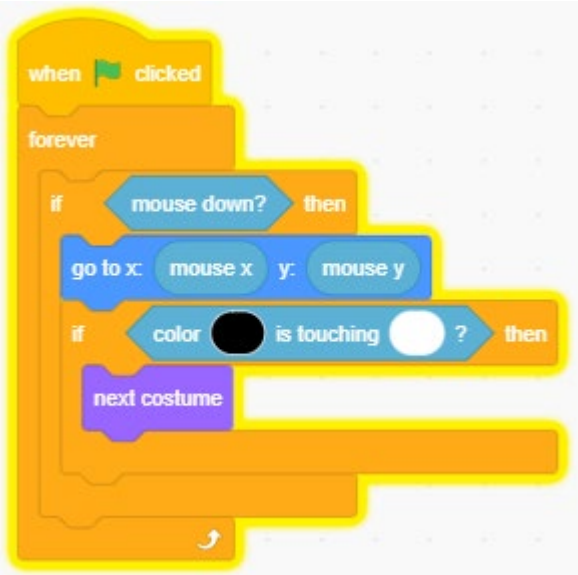


Figure 108: Sprite changing colour when touches another sprite

With this code, a sprite will move from one point to another and will change colour when a black sprite is touching the white sprite. One sprite will be black, and another will be white, so when they meet, they will change costumes.

4.1.6 Asking a question

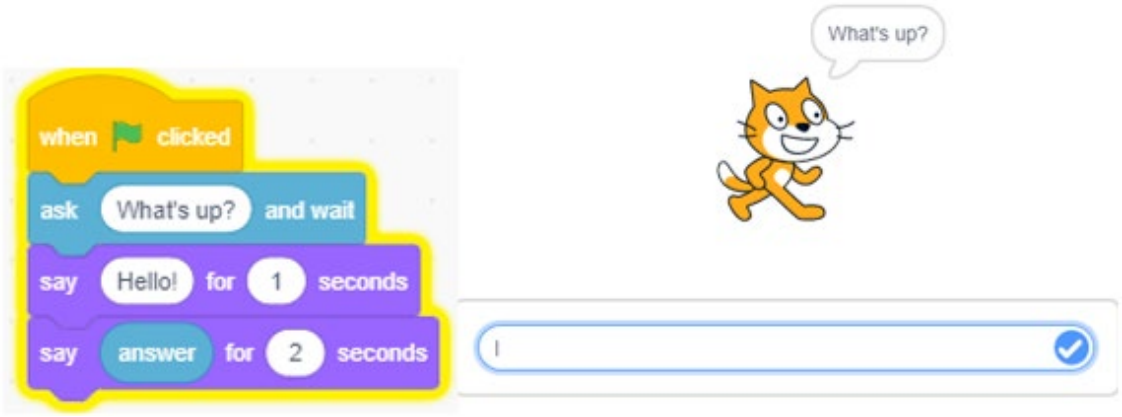


Figure 109: Asking questions

This block allows you to ask a question and stores keyboard input in the answer block. The question appears in a voice balloon on the screen. The program waits as the user types in a response, presses the enter key or clicks the check mark.

Figure 110: Asking and responding to questions

Users can type the answer to a question. The example above shows the user answering a question.

Example:

The Sensing folder contains blocks of code that allow sprites to interact with each other. Select a sprite whose code allows it to cross the path of another sprite. Add the following code to this sprite:

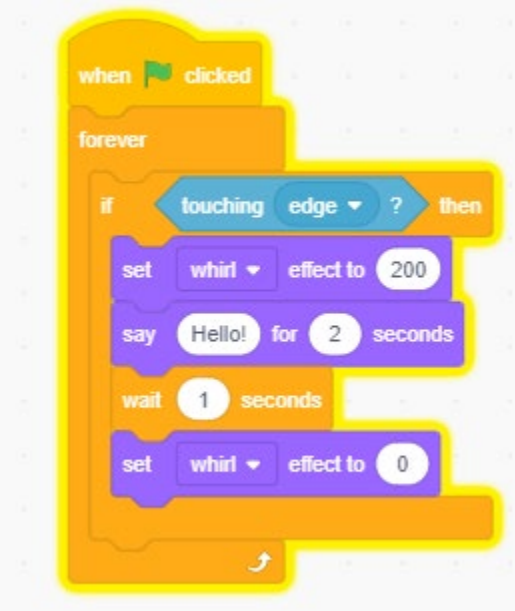


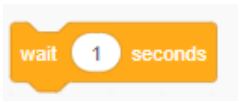
Figure 111: Interaction between sprites

Please note that 'Sprite5', as referred to above, is only used as an example and could be any other sprite of your choice. 'Whirl', the 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 to the block of code, after waiting 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 only one special effect in this command block located within the Looks folder as shown below: experiment with these options and in changing the number in the set whirl effect to __ An Ambient Aquarium Sound. Finally, select suitable music to match the mood of slow-moving fish.

4.2 CONTROL BLOCKS

Control blocks are color-coded gold and are used to control scripts. It has different blocks with different functions. Most of the blocks are shaped like the letter C and therefore, they are known as "C" blocks. Few of the control blocks which are not C-shaped are known as "stack" blocks.

4.2.1 Wait

The 

or "**Wait () Seconds"** block is a Control block and a Stack block. The block pauses the execution of a script for the specified number of seconds. The number after "wait" can also

be a decimal number. This block is used whenever a sprite must wait for the specified time before the next action takes place.

Example

The example below instructs the sprite to say “Hello!” when the green flag is clicked and, after five seconds, to say “How are you?”

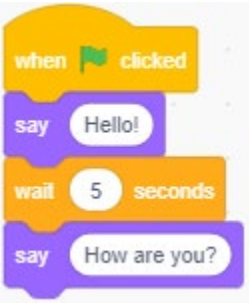


Figure 112: Instructions to say “Hello!"

4.2.2 Repeat

The **Repeat** () block is a Control block and a C block. Blocks which are put inside this block will repeat **(loop)** a given or specified number of times, before allowing the execution of the script to continue. Below is a figure of a “Repeat ()” block.

Example

The example below instructs the sprite to say “Ha” five times when the green flag is clicked and then to say: “I am happy to see you again!”

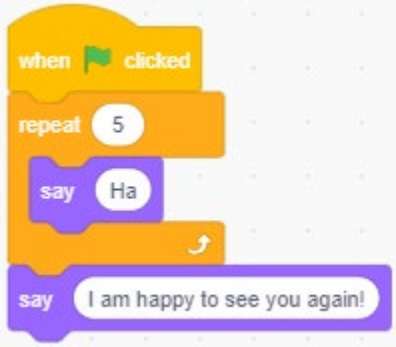


Figure 113: Instructions to say "Ha"

4.2.3 Wait until

The block or "**wait until** ()" block is a control block and a stack block. The block pauses the execution of the script **until** the specified condition is performed.

Example

The example below instructs the sprite to say "Hello!" when the green flag is clicked and to wait until you press the space bar key on your keyboard before saying "How are you?"

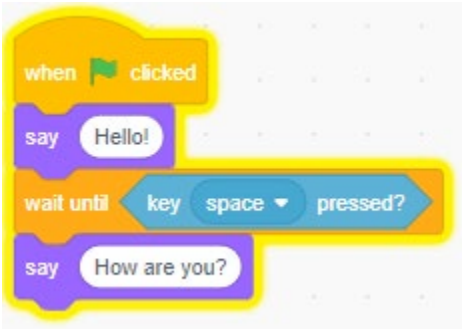


Figure 114: Instructions to say "Hello!" when green flag is clicked

4.2.4 "Repeat until ()" block

The block or **"Repeat Until ()"** block is a Control block and a C block. Blocks inside this block will repeat until the specified statement or condition is performed. After what, the code beneath the C block (if any) will execute. This loop is similar to a "while" loop used in other programming languages.

For example, in the following example, the code in the loop will run, repeatedly, as long as a variable (i) is less than 10:

```
while (i < 10) {

text += "The number is " + i;

 i++;
}
```

The above codes do the same thing as the **"Repeat Until ()"** block in Scratch. The above code will allow the function to continue as long as the number is less than 10.

4.2.5 The "If" statement block

The "IF statement" block is one of the main blocks in the control block. They are color-coded yellow and are used to execute a section of code only when a condition specified is true. It is often combined with repetition (repeat commands).

Example

The example below instructs the sprite to move 10 steps when the green flag is clicked and to say "You're touching the edge" when the sprite reaches the edge and then the loop stops.

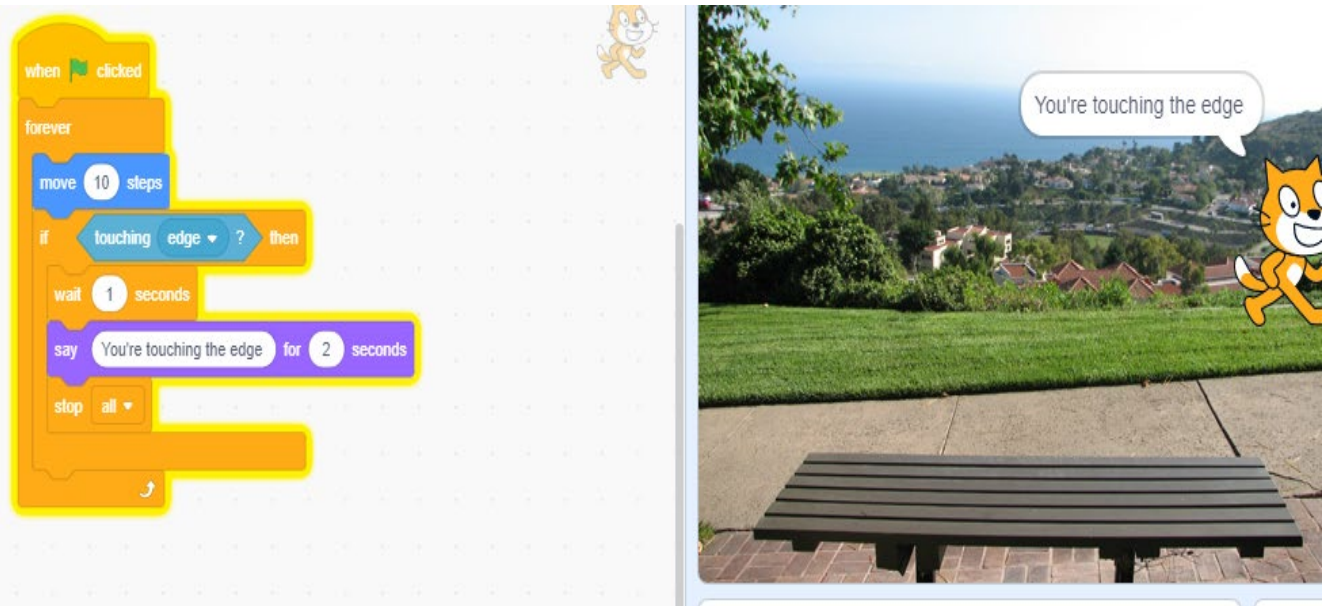


Figure 115: Code using if statement block

An “If statement” block works when you want to execute a section of code only when a specified condition is fulfilled. The “if statement” blocks comes in two types: the “If statement” block and the “If-else statement” block.

Sometimes you wish to execute one section of code when a condition is true, and a different section of code when the condition is false. The code following the “if” is executed when the condition is true, while the code following the “else” is executed when the condition is false. This is called a “two-way branch”.

Example:

If, for example, the sprite does not reach the edge and you want the sprite to say “You did not touch the edge”, we can use the “If-Else” block as in the figure below.

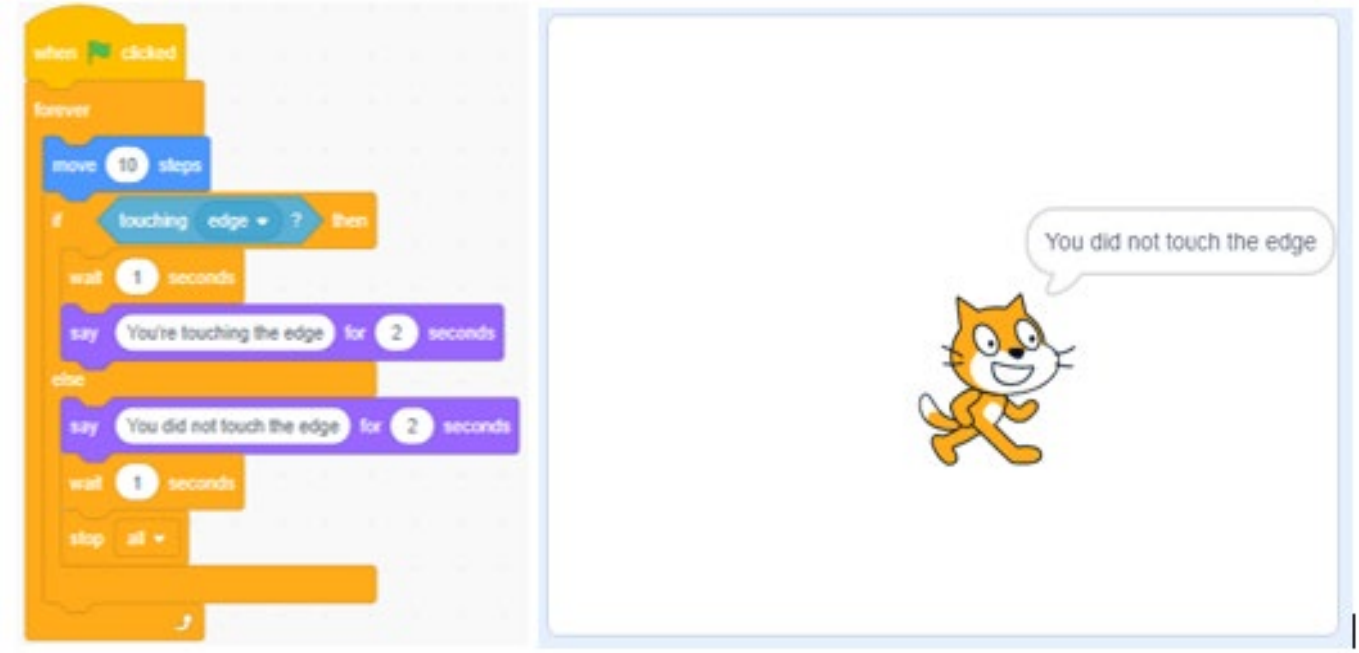


Figure 116: Using If-Else control blocks

The example above instructs the sprite to move 10 steps when the green flag is clicked. If after the 10 steps the sprite has not reach the edge, then the sprite will say “You did not touch the edge” and will move 10 more steps. If after this again it has not touched the edge, it will say again “You did not touch the edge”. When it reaches the edge, then it will say “You’re touching the edge” and will stop.

4.2.6 Forever loop

The block or the “forever” block is a block with an infinite loop that repeats itself forever. Blocks held inside this block instruct the sprite to perform a specified action and will not stop unless it is instructed to stop using the “Stop all” block.

Example

The example below instructs the sprite to move forever until it moves beyond the edge of the stage.

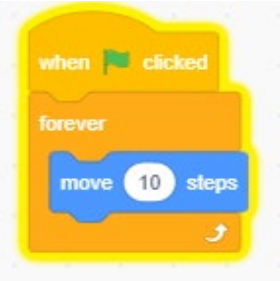
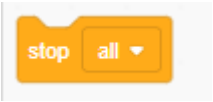


Figure 117: Forever Loop

Due to this infinite loop, the block has no bump. This means it cannot be connected to another block at the bottom. Having a bump would be pointless, as the blocks below would never be activated.

4.2.7 “Stop” block



The block or “stop” block will stop everything from happening. It is always the last block to stop all other functions and processes. It is usually used to stop a loop mainly actions in a “forever” block like in the example below.

Figure 118: Stop all function

4.2.8 "When I start as a Clone" block

Cloning is a feature that allows a sprite to create a copy of itself while the project is running. Each clone has the same costumes, sounds, scripts, and variables as the original but is otherwise independent. Cloning is commonly used when a project has many similar sprites doing similar things. Because clones are created by the project rather than the user, cloning prevents the user from needing to make the same changes to each of many sprites. There are three blocks related to cloning, all of which are found in the control palette:

The "**When I Start as a Clone"** block is a control block and a Hat block. Remember that a Hat Block is a block that starts a script when a specific event occurs. All hat blocks are either Control blocks, Events blocks, or More Blocks.

It activates in a **clone** when it gets created. You use the block to Initialize clones and give them behaviour.

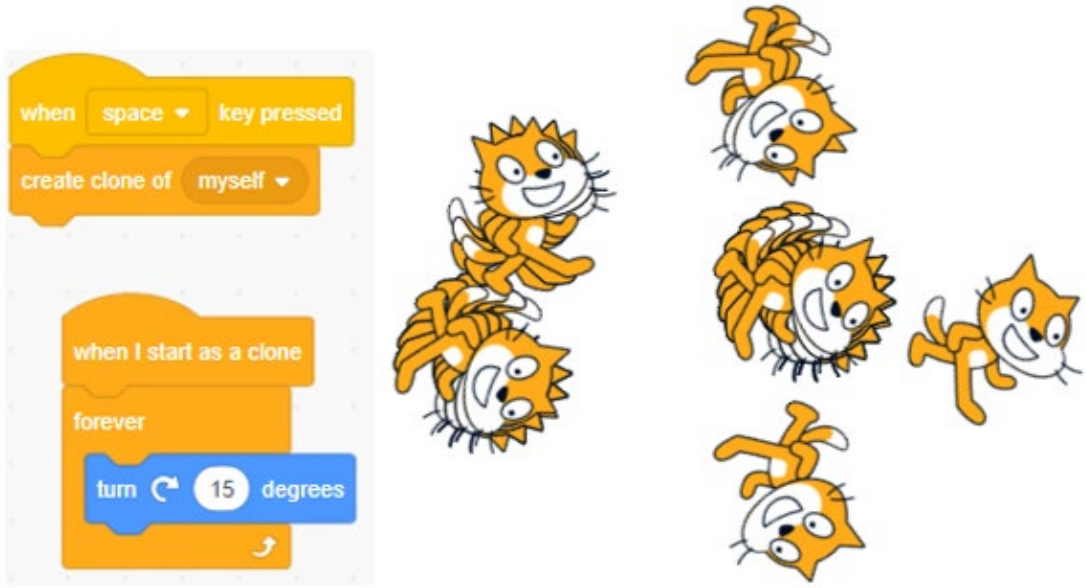
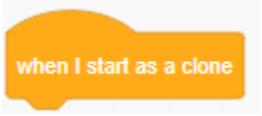


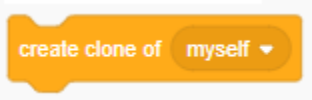
Figure 119: Cloned sprites

When space key is pressed, the sprite is going to turn 15 degrees forever and create clone. While holding the space key, you can move each cloned sprite and it will have the same behaviour as the first sprite.



4.2.9 Create clone of () Block

The Create Clone of () block is a control block and a stack block. It creates a clone of the sprite in the argument. It can also clone the sprite it is running in, creating clones of clones, recursively.



4.2.10 Delete this clone

The **Delete This Clone** block is a **control** block and a Cap block. It deletes the **clone** it runs in. This block is the only way, besides clicking the Green Flag or Stop Sign, to **delete clones**. When a **clone** is created, you can use this block to **delete** it.

4.3 BROADCAST BLOCKS

Broadcasting is the process of making one sprite move to another sprite or send a message. For example, you can make one sprite tell another to do something. Broadcast blocks are used in games and animations. Broadcasts are sent with the blocks Broadcast () and Broadcast () and Wait and are received by the hat block **When I Receive ()**.

4.3.1 Broadcast () Block

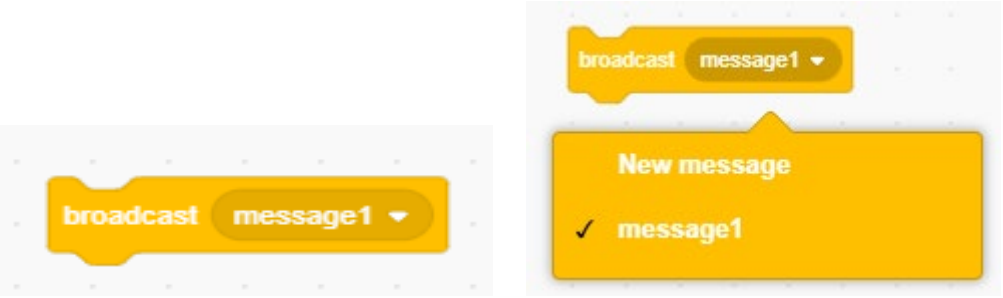


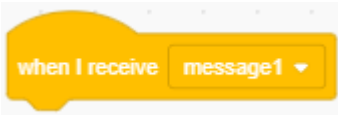
Figure 120: Broadcast () block

This block broadcasts the specified message and has no further effect. You can add a message to be broadcasted by clicking on the white down arrow and select “New message” (Figure 120).

4.3.2 Broadcast () and Wait Block

This block broadcasts the specified message and blocks its scripts until all scripts under a “When I receive message 1” block have been executed.

4.3.3 When I Receive () Block



This block will stay inactive until it receives the specified broadcast. Once it has been received, the script is executed. It can be started more than once.

Example: Broadcasting a message in Scratch

- Add two sprites.
- One sprite broadcasts a message to another, and another has to wait for a specific time.
- After that time the other sprite will reply to the message.

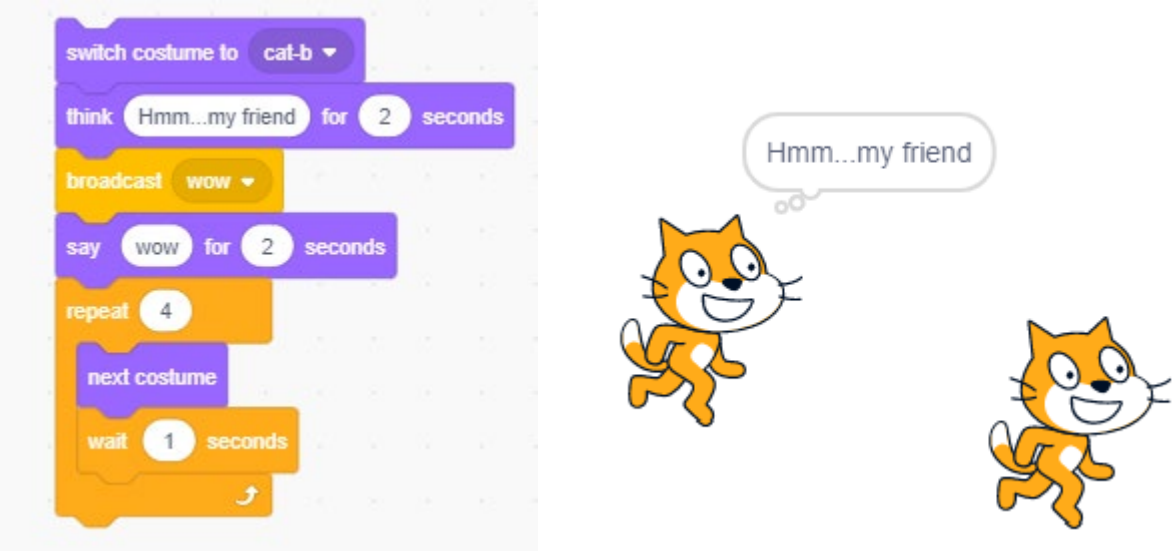


Figure 121: Broadcasting messages between sprites

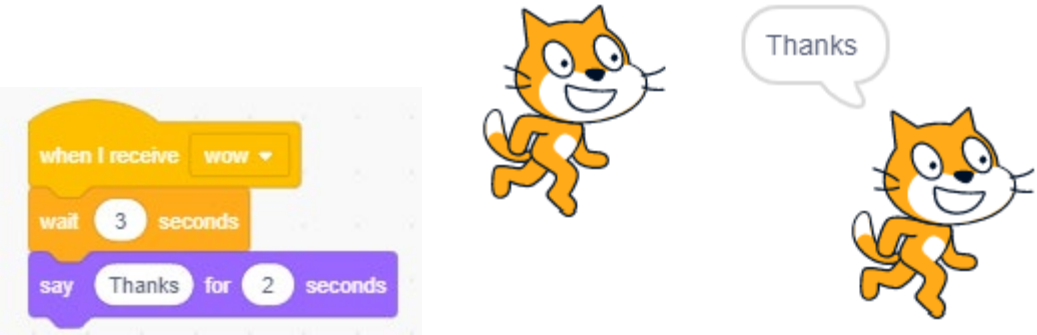


Figure 122: Sprite received message and replies

Activity on Broadcasting

Let us do the activity below to practice broadcasting.

- Step1: Choose two sprites of ladies. Give them both a name, for example Anne and Abby.
- Step2: Prepare the following settings: Anne is walking on the beach and she saw Abby. She called her "Abby". Abby waited for 3 seconds and replied, "Yes Anne, talk to me".
- Step3: Click on Anne sprite and write down the instructions below.

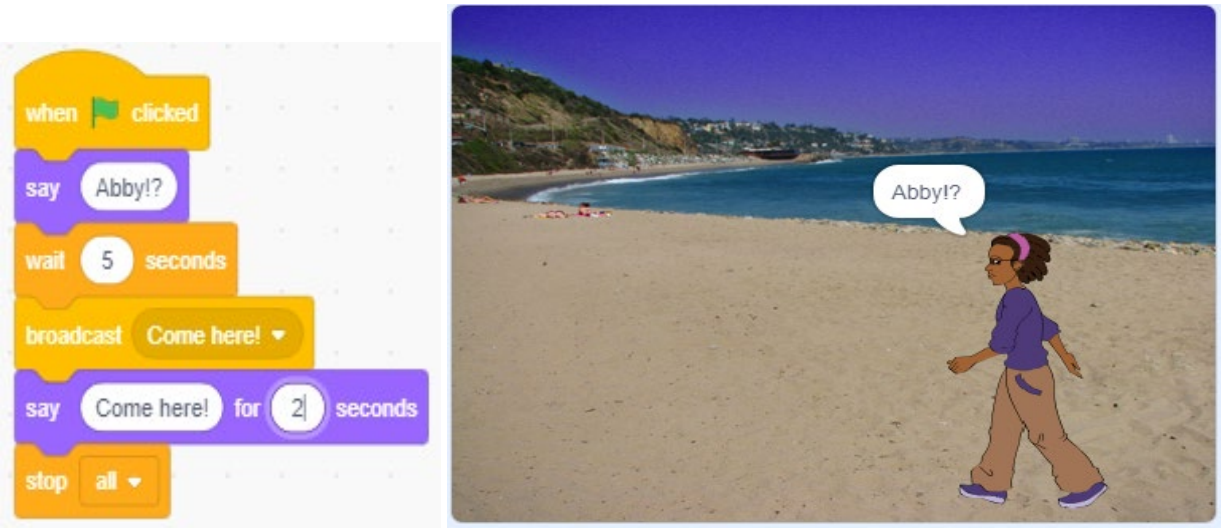


Figure 123: Calling Abby

- Step 4: Then click on Abby sprite and write down the following instructions:

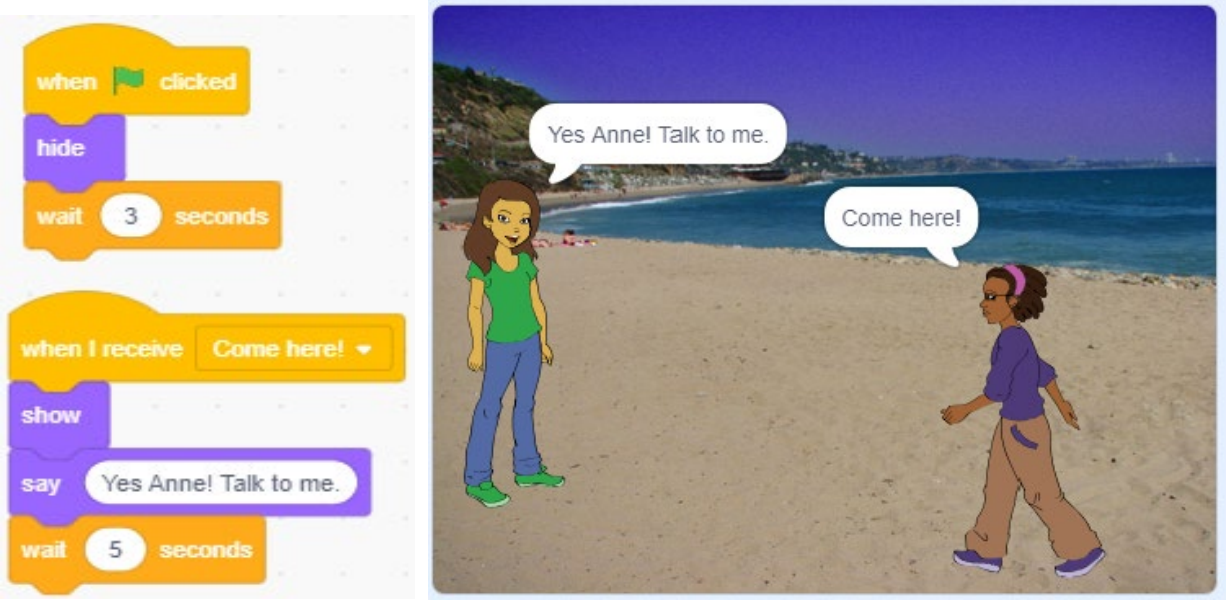


Figure 124: Abby responding and Anne giving instructions

When the green flag is clicked, the following will happen:

- Anne calls Abby. After 5 seconds Anne sends a message “Come here!”
- Abby is hidden but when she receives the message from Anne calling her, she shows up and says: “Yes Anne, talk to me!”

This example shows us how two sprites can talk to each other. With broadcast blocks, we can create a story with more than one sprite.

Activity

- Choose 2 sprites of your choice and a background reflecting two people meeting in an outside place.
- They are talking on how to develop a music project together.
- Write down their dialogue.
- Prepare to share your results.

4.4 FEEDBACK GROUPS

When you write a project, it is useful to get feedback from other Scratch users. It can be done online in the Scratch community or in your club. Let’s see how you can share your projects in Scratch.

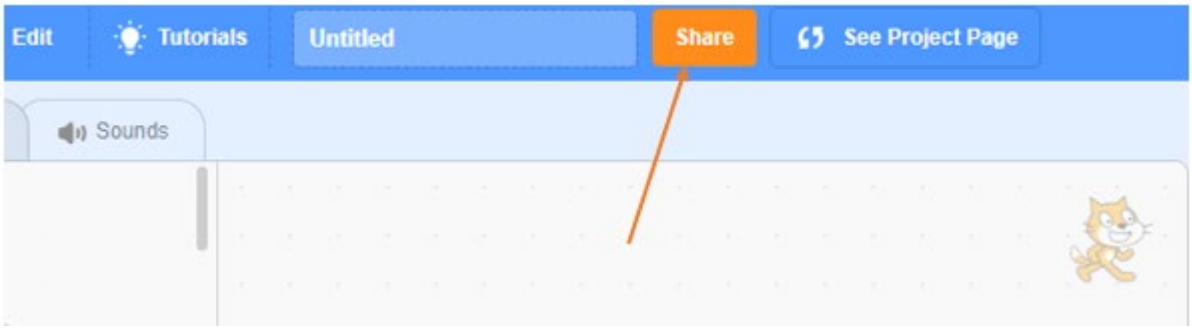


Figure 125: Share buttons

On the top header of the platform, there is a button called share. By clicking this button, the platform will share your project. You can edit the name of the project and give clear instructions of how to use your project.

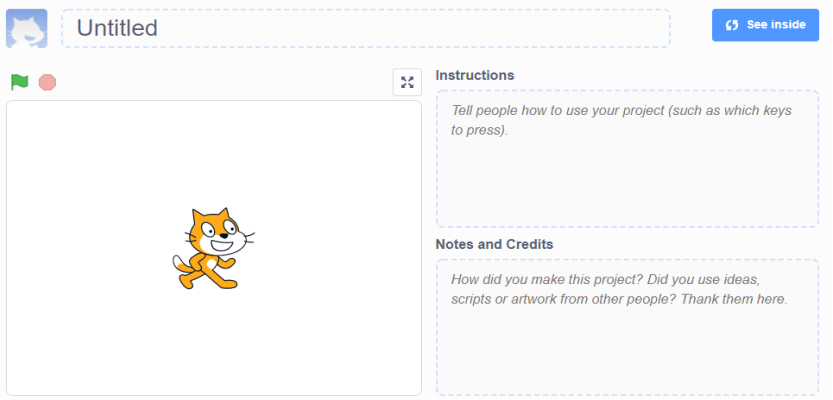


Figure 126: Editing the name of the project and giving instructions

By sharing your project, you invite people from the Scratch community to give you feedback and comments. You can also reply to the feedback and comments.

4.5 SOUND BLOCK

You may add sounds for animation purposes.

Figure 127: Adding sound to animation

Or you may create your own sound through recording. For recording you just click where you have a default sound “Meow “after that the below image will show then you press on the record then after the start button and it starts recording.

Figure 128: Choosing a Meow sound

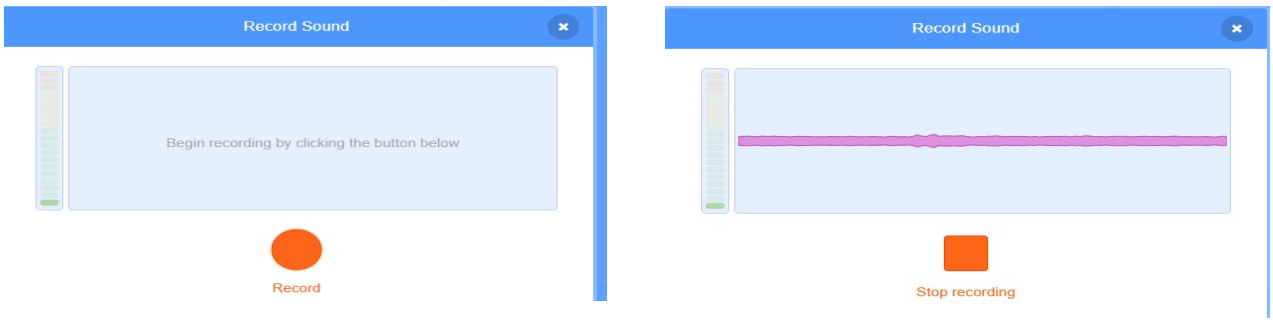


Figure 129: Recording a sound

Later you will press the button to stop recording as below then when you click on save it will save it where you have “meow” and import it into your workspace by selecting it.

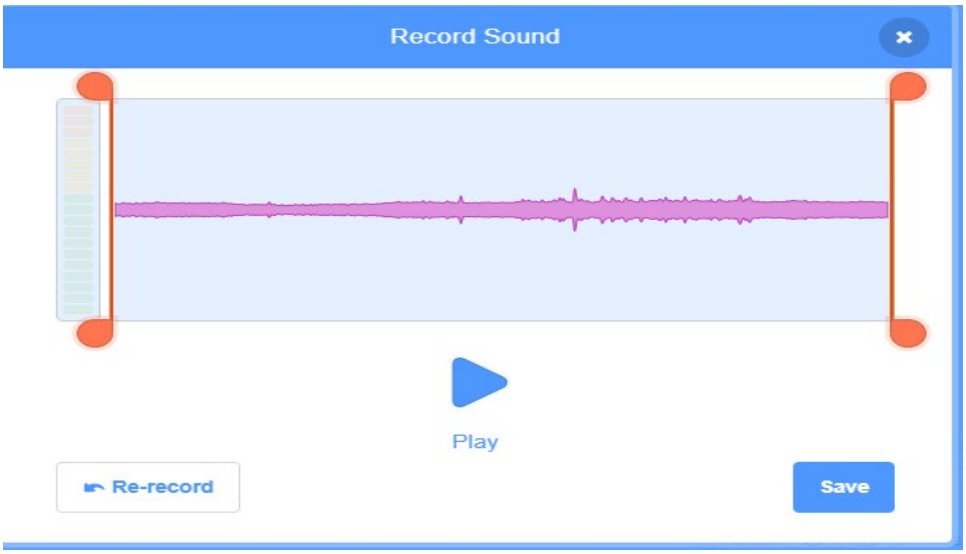


Figure 130: Playing the recorded sound

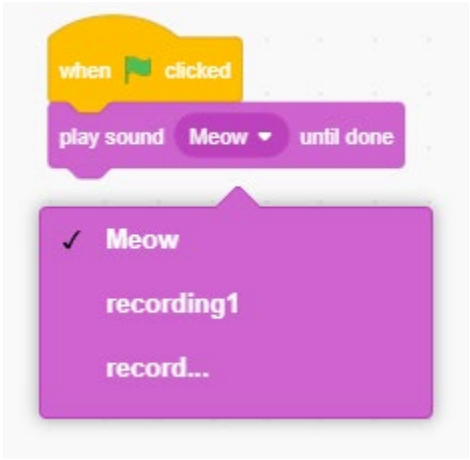


Figure 131: Inserting the recorded sound

END OF MODULE 4 ACTIVITY

Create the following story, applying what you have learnt in this module. It is a discussion between Kamana and Kalisa. Use Scratch to create that discussion.

- Kalisa: Kamana, where are you?
- Kamana: I' m in Kayonza centre (round about)
- Kalisa: Aahhh Kamana would you join me for a talk?
- Kamana: It's Ok Mr Kalisa, where can I find you?

- Kalisa: Kamana, assume that the road heading to Kigali is X –axis and the road heading to Kibungo is Y- axis, find me at point (-50, -50) coordinate at Cyeru cell nearby Kayonza district Headquarter
- Kalisa: You mean I' m at (0, 0) in coordinate system (Haaaa, Haaaa) it is now easy to find your location.
- Kamana: Well, come find me at (-50, -50) then
- Kalisa: Thank you Kamana for this orientation

If you are done with the exercise, use this link to compare your work.

MODULE 5

POLYGONS AND FLOWERS

OVERVIEW

In this module, you will learn about the function of a pen in creating games and stories in Scratch animations. You will learn to create 2D shapes and images by combining lines and angles. You will also use repeat controls to draw shapes and images.

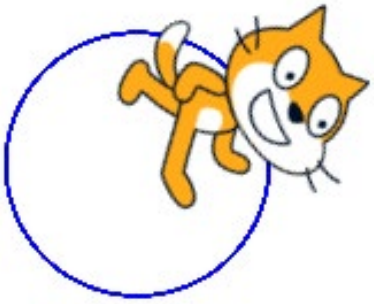
LEARNING OUTCOMES

At the end of this module, you will be able to:

- create 2D shapes.
- create images.
- draw images.

5.1 PEN BLOCKS

The **pen block** allows you to draw images in Scratch. Just like in our daily lives, we can use a pen to draw anything. The pen tool draws lines behind the centre of a sprite as it moves.



• While drawing in Scratch, the sprite always carries a pen. Everywhere the sprite moves, it leaves a line behind. If you want to draw a shape, move the sprite in that shape and the line will be drawn.

• A fun way to introduce this concept is to ask a learner to become a sprite and go to the whiteboard (or blackboard) blindfolded with a marker pen (or chalk), ready to draw; but always waiting for instruction from other learners around her/him who direct her/him where to move her/his hand. In this way the children are giving and following specific instructions such as “move the marker 10 centimetres up, five centimetres forward. Turn 90 degrees to the left.”

The pen block is not a default block in Scratch. You must add it as an **extension block**. Follow these steps to add the pen block:

Step 1: Click on the add extension button in the bottom left-hand corner.

Step 2: Choose pen.

Step 3: The pen section will appear at the bottom of the block’s menu.

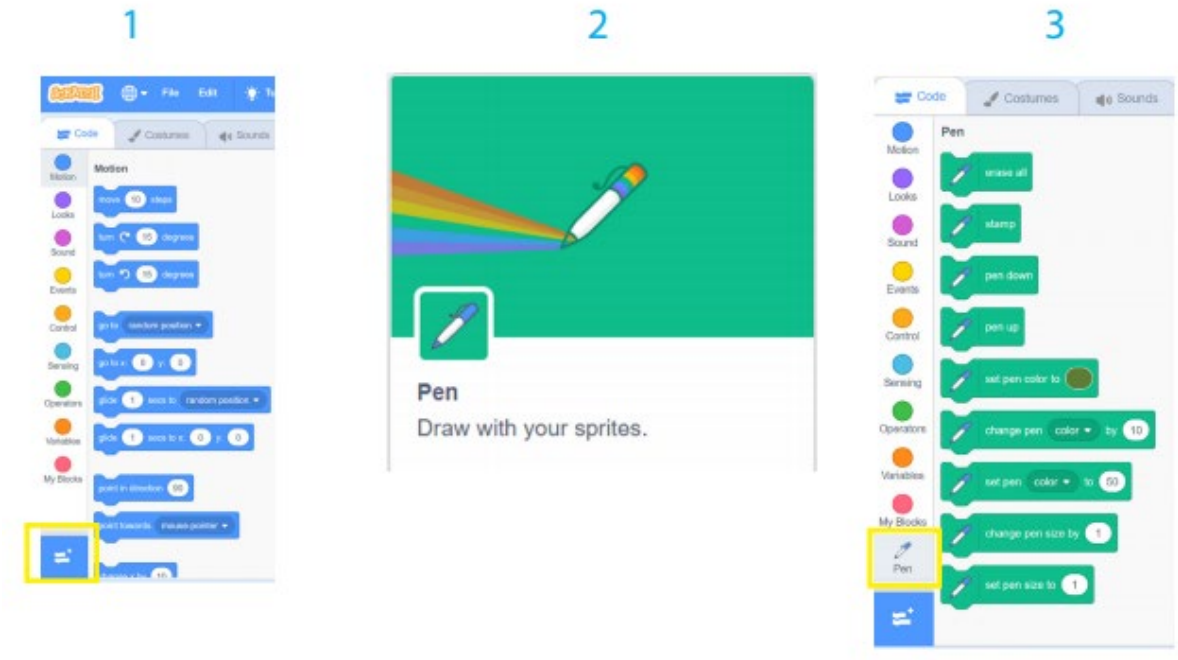


Figure 132: Adding a pen

5.1.1 Drawing shapes

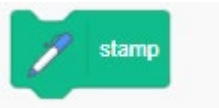
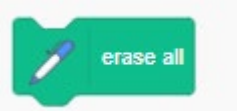
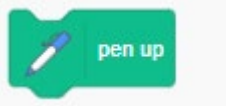
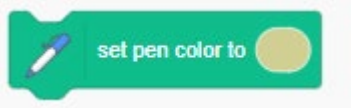
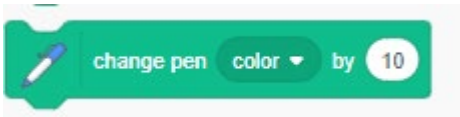
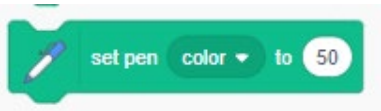
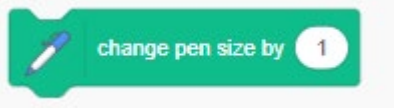
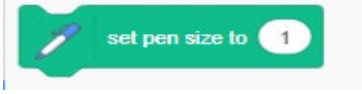
Pen block	Description
	When used in a script, the sprite will produce a bitmap image of itself which is stamped onto the stage. Bitmap is a digital image composed of a matrix of dots. Like other Pen blocks, the Stamp block will not draw over sprites.
	The block removes all marks made by the pen or stamping. It is the only pen block that the Stage can use.
	The block will make its sprite continuously pen a trail wherever it moves, until the Pen Up block is used. The colour, size, and transparency of the trail can be changed with other blocks.
	If a sprite is currently using the pen because of the Pen Down block, the block will cause the sprite to stop drawing a trail. (Otherwise, it has no effect.)
	The block sets the pen's colour to the given colour, which can be selected by clicking on the input.
	This block sets the pen's colour, saturation, brightness, and transparency. The first value can be selected from "colour" (default), "saturation", "brightness", and "transparency".
	The block sets the pen's colour at the certain value.
	The block changes the specified value by the number input.
	The block sets the pen's size to the given number. The pen draws a trail of circles. The diameter of the circle, in pixels, is equal to the pen size.

Table 10: Pen block

Figure 133: Drawing a line

In the example above, we were testing if the pen really works. Since we are sure that it works, we can now go ahead and explore the other options that we have.

5.1.2 Hide and Show

You can hide the sprite just to give a clear view of what you are drawing. When a sprite is hidden, you will only see the drawings but not the sprite drawing. This will give the viewer a clear view and make the drawing better. To hide the sprite, add a hide block from Looks to the start of the program and it will disappear.

Figure 134: Hiding a sprite

When you want the sprite to appear again, there is an option that allows you to make the sprite visible. To bring back the sprite, go to the Looks block and add show. This option is going to make the sprite appear again.

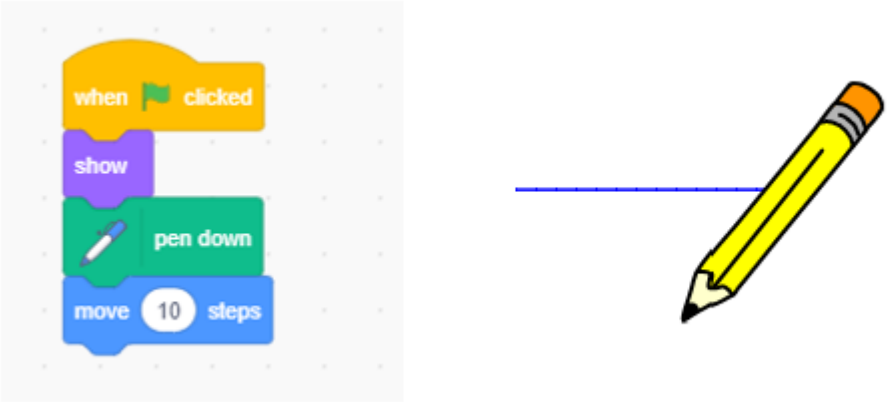


Figure 135: Showing the sprite

5.1.3 Changing colour

When you are drawing, blue is the default colour. You can change the colour of the pen to make your drawing look good. Drag a set pen colour to block into your sprite panel and snap it in above the *pen down* block.

Figure 136: Changing colour

5.1.4 Erasing

When you are drawing, some of the things you draw will not go automatically. You must clear them to give space to the new drawings. Add an “erase all” block from the Pen section to the start of your code to clear all the drawing you drew before. It is better to add the “erase all” function before just after the click and the hiding. This will help to erase everything before the sprite starts drawing.

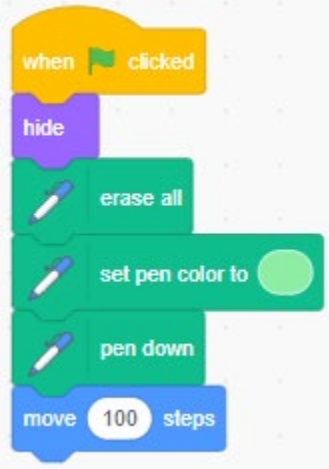


Figure 137: Erasing

5.2 DRAWING ANGLES

Even though drawing seems to be easy and direct, you can make it even more amazing. You can draw with angles to make a clear shape. For example, let us draw a shape with an angle of 90 degrees.

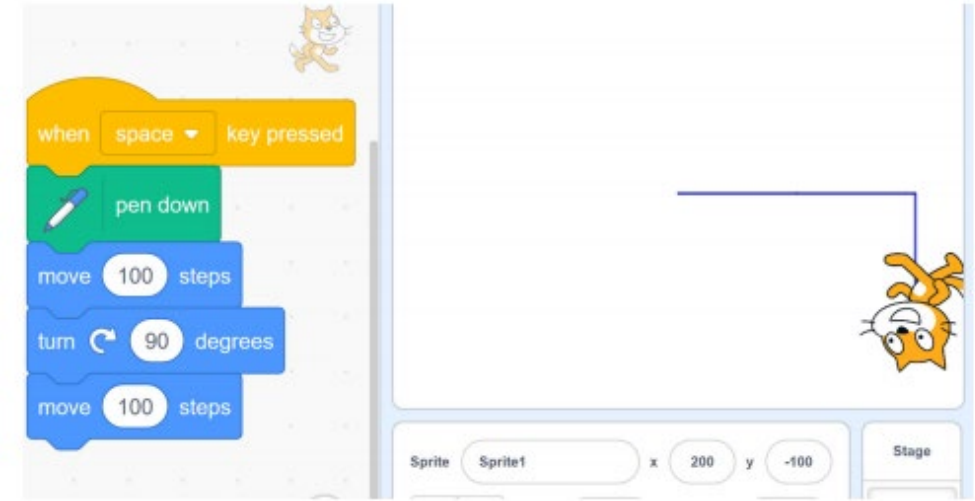


Figure 138: Drawing an angle of 90 degrees

For a full rotation, the angles must add up to 360 degrees. These degrees are divided between the numbers of corners a shape has.

Therefore:

- Square /rectangle = 4 x 90
- Triangle = 3 x 120
- Pentagon = 5 x 72

- Hexagon = 6 x 60

5.2.1 Drawing a square

Without loops, you can draw a square by drawing a certain length of line and then turns right 90 degrees, again and again until you complete the square. You will surely have your square shape, but you will have a lengthy thread of blocks. For a complex shape you will feel confused with your own code. You need something shorter that does the same job.

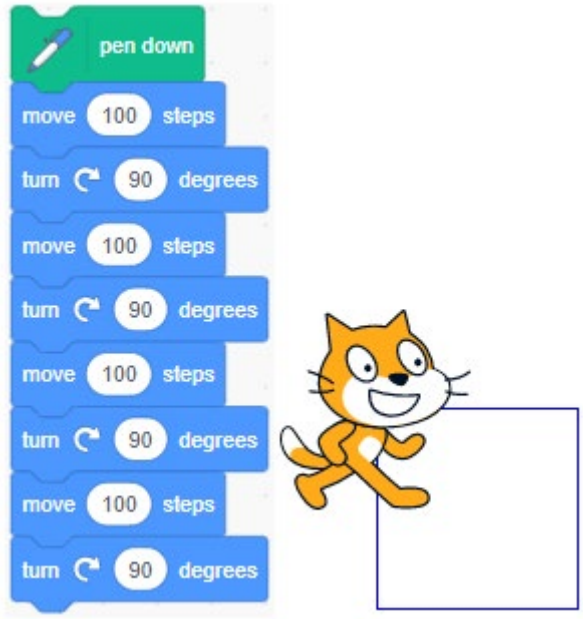


Figure 139: Drawing a square

You can draw the same square with fewer blocks. All you need to do is using a type of loop called repeat until, which you find in the Control section. Repeats allow you to draw shapes more quickly. They also enable you to create interesting patterns. To draw a square, you can just draw one side, turn, and repeat it 4 times.

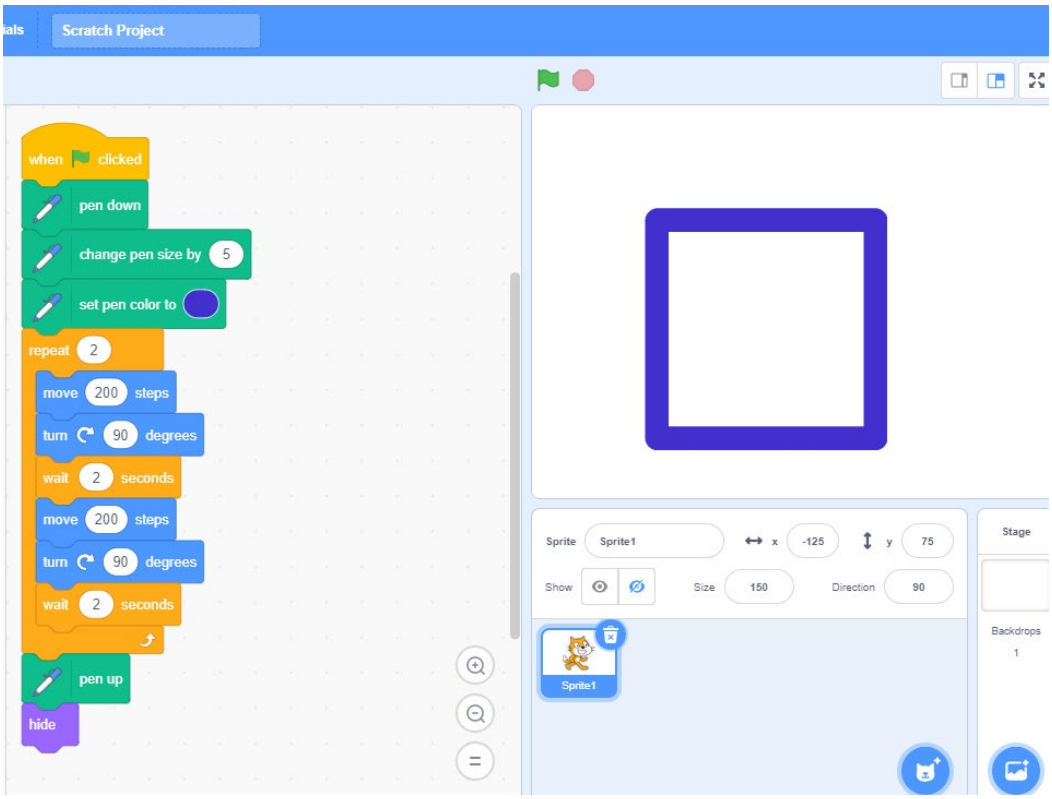


Figure 140: Drawing Square using a repeat *loop*

5.2.2 Drawing a circle

If you start out at any point on the circle and travel 360 degrees, you will end up in the same place and facing the same way as when you started. Try rebuilding the program as indicated in the screenshot below.

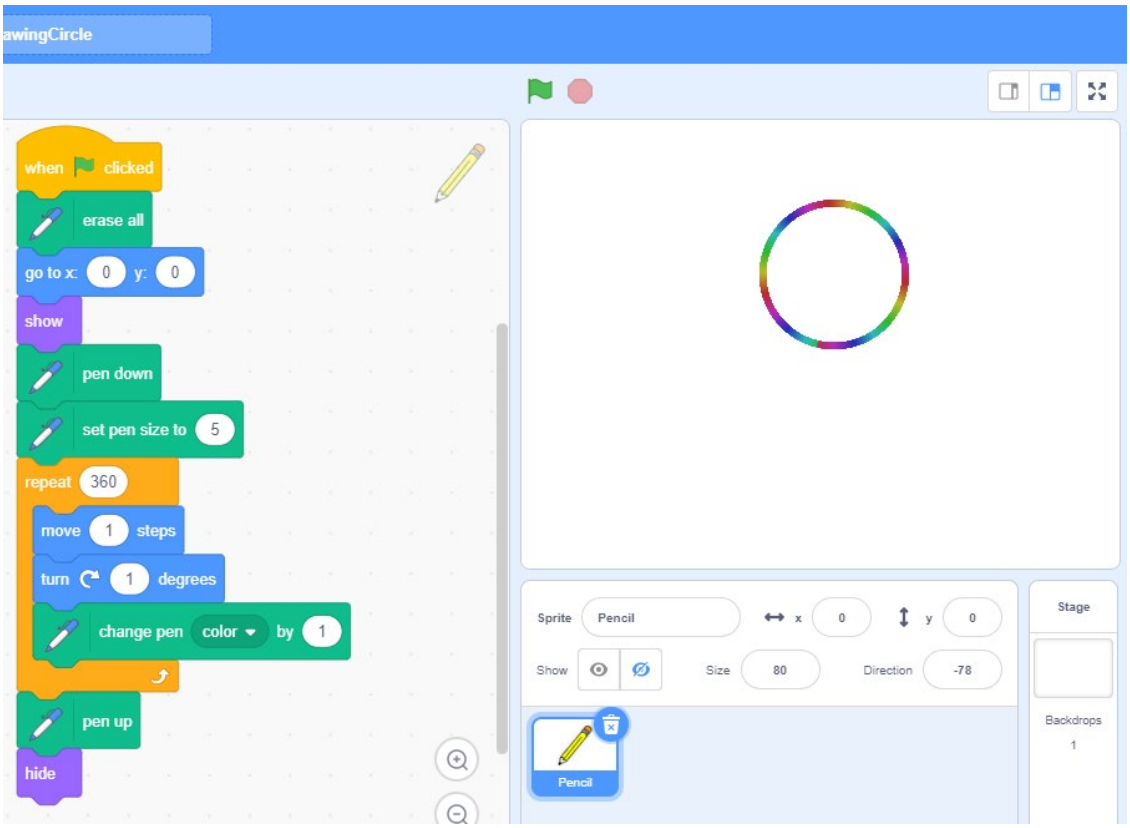


Figure 141: Drawing a circle

5.2.3 Drawing a Hexagon

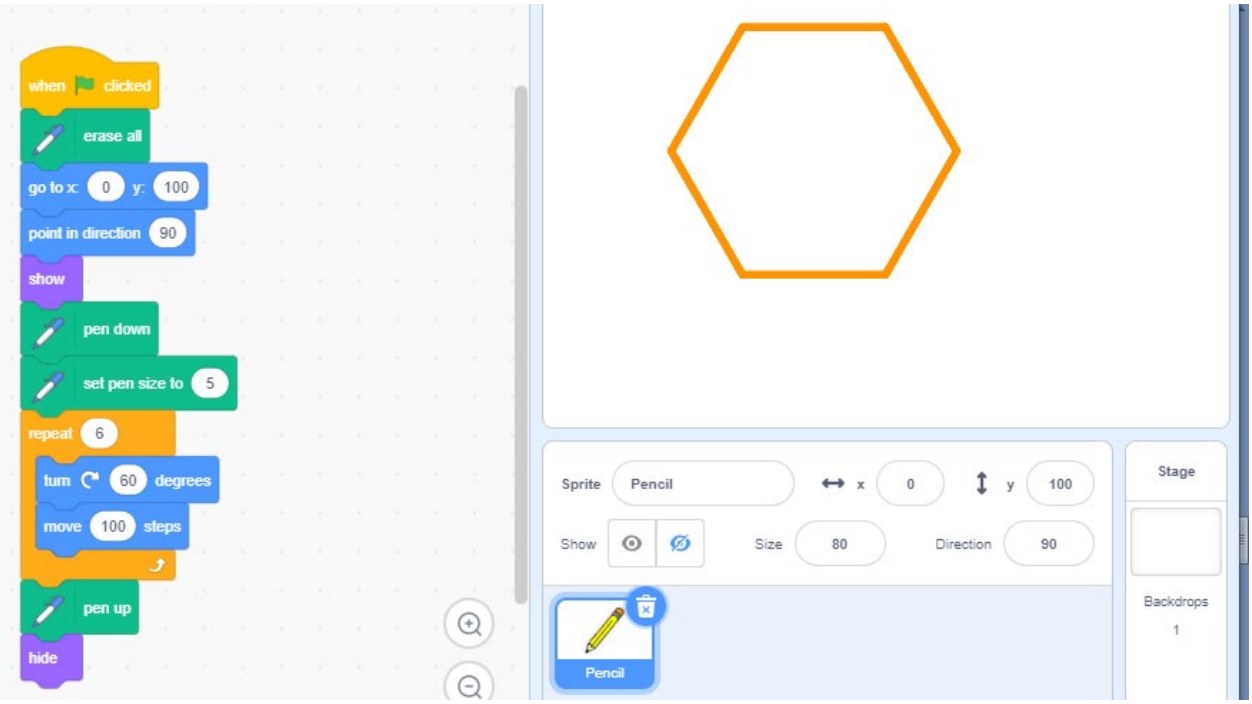


Figure 142: Drawing a hexagon

5.2.4 Drawing an Octagon

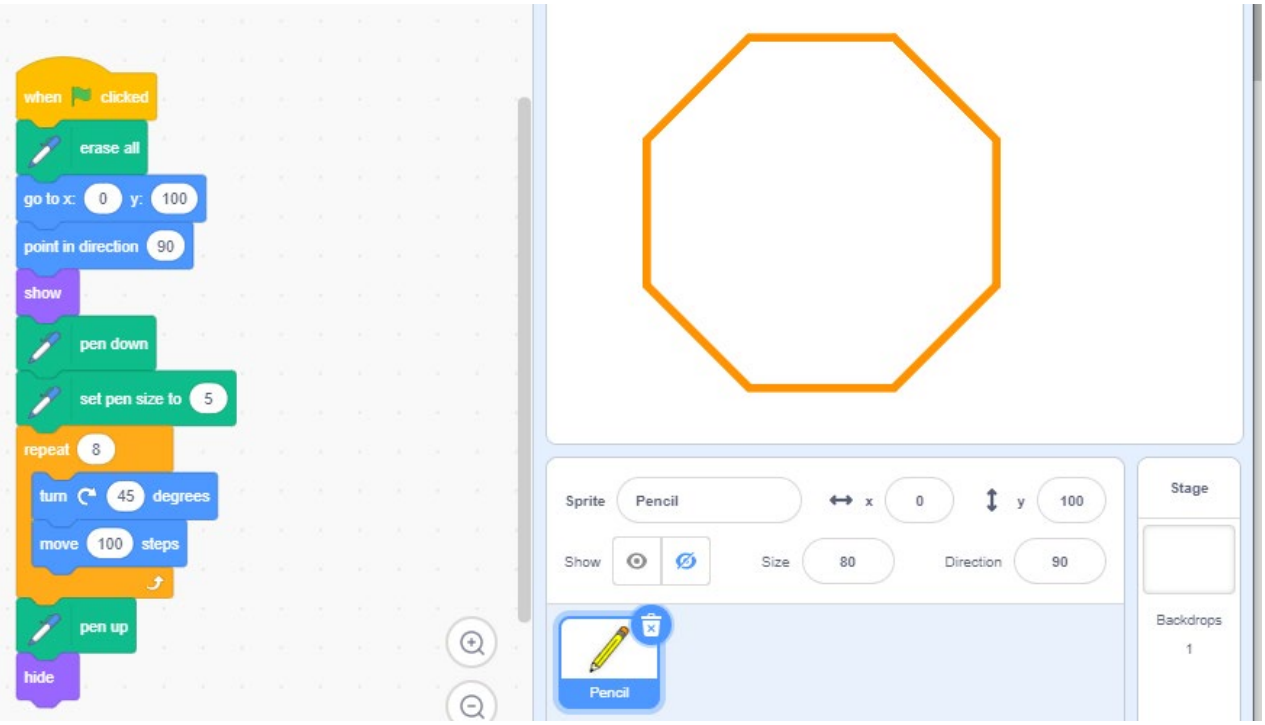


Figure 143: drawing an octagon

5.2.5 Drawing a Flower

There are many ways to draw a flower, depending on your creativity and Scratch skills. Let us show one of the flowers you can draw. In the Inner Loop we want to change the colour of the pen ink after every 15 steps. The inner loop will repeat for 5 times.

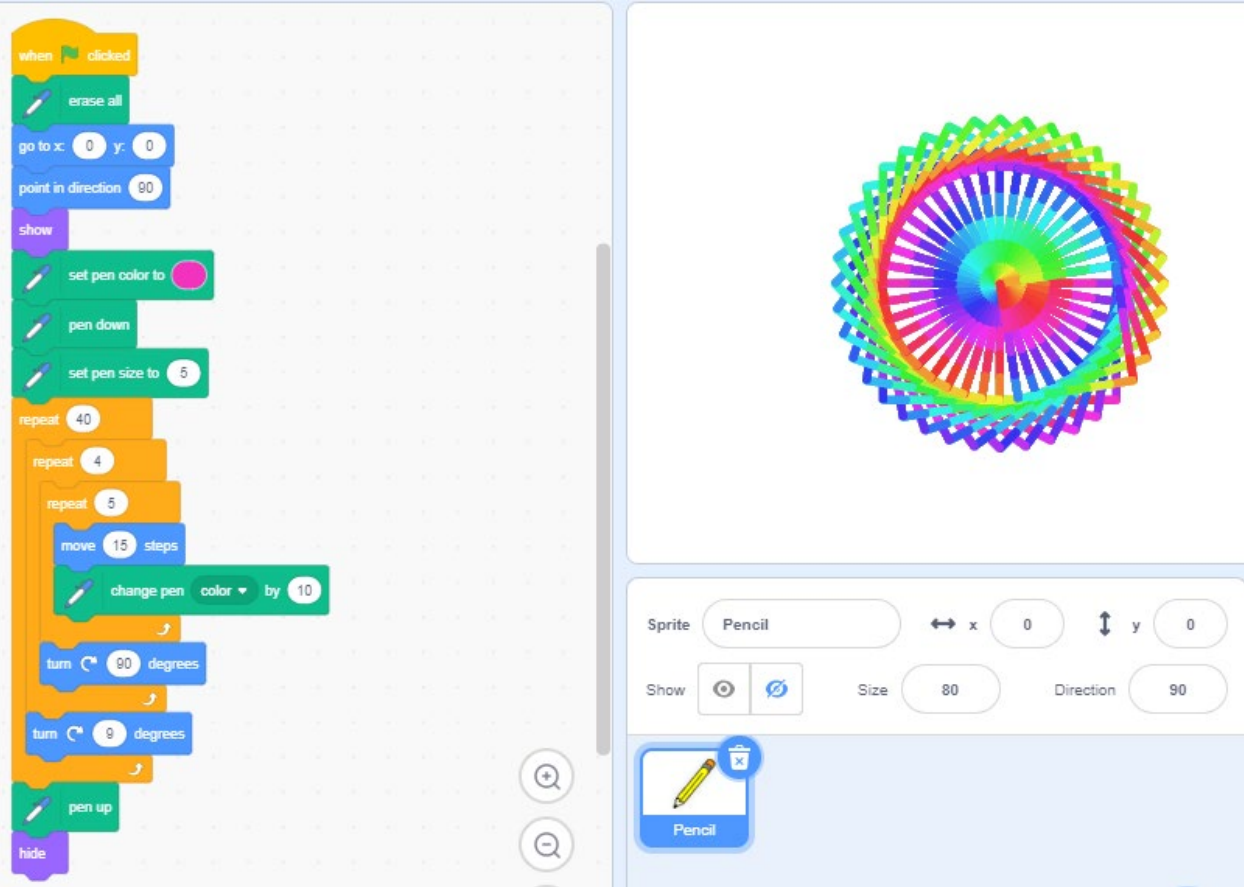


Figure 144: Drawing a flower

With this **repeat until loop**, the sprite will repeat until a certain condition is met. The pen always starts drawing in the direction of the middle, because the first Motion block that runs after the pen down block is go to x: 0 y: 0. so the pen will draw a line as it moves to the centre of the Stage.

The pen does not stop at the edge of the Stage, because you have not yet told the *repeat until* loop what condition it is checking. This means the condition can never be met, so the loop will run forever. To stop the sprite, you must use a sensing block.

Time to fix your *repeat until* loop so that it stops when you want it to. You are looking to figure out if the (invisible) sprite is touching the edge of the Stage, so you need a Sensing block — in this case, the touching block. We will discuss Sensing blocks in more detail in Module 6.

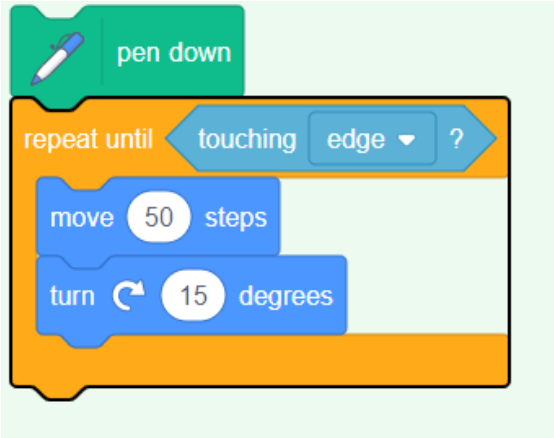


Figure 145: Sensing block stops the repeat until loop

By adding the *touching* block into the repeat until loop and select edge. Then the loop will run until the (invisible) sprite touches the edge of the Stage.

With the repeated drawing, we can create unusual patterns that suit our thinking and design.

5.2.6 Similarity between "drawing a shape using Scratch" and "printing a shape using C++"

Shape drawing is not only done in Scratch. It's also done in other programming languages. Let's see one of the programs that can print out a triangle in C++ programming language. Then we will show another program made in Scratch that draws the triangle too.

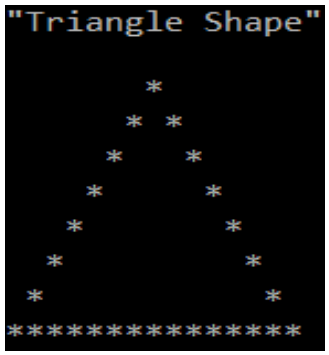
You will notice that a triangle drawn in Scratch is seamless shape made up of straight lines while a triangle printed out in C++ is made up of asterisks.

We can easily see that Scratch can be chosen over C++ if you want to quickly program a multimedia scenario.

C++ program that prints out a triangle shape

```
//C++ program to print triangle
#include<iostream>
Using namespace STD;
int main (){
 Cout<<"\"Triangle Shape\"\n\n";
 int z=1;

 For (int i=0; i<7; i++) {for (int j=7; j>i; j--) {   cout<<" "; // displaying space her   }
```



```
  Cout<<"*";  // displaying asterisk here
  if (i!=0){
   for (int k=1; k<=z; k++){
     Cout<<" ";
   }
   Cout<<"*";
   z+=2;
  }
  Cout<<end; // endl is for new line
 }

 For (int i=0; i<=z+1; i++) {
  Cout<<"*";
 }
Cout<<endl;
return 0;
}
```

Scratch code that draws a triangle shape

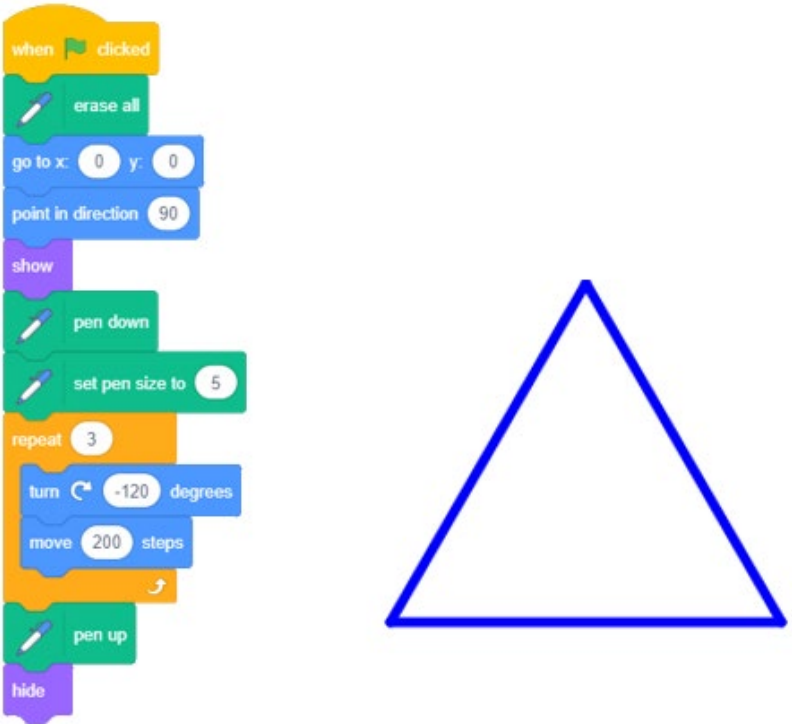


Figure 146: Building a triangle in Scratch

END OF MODULE 5 ACTIVITY

The following figures show the sprite paths in the stage. Write scripts to implement the paths when clicked.

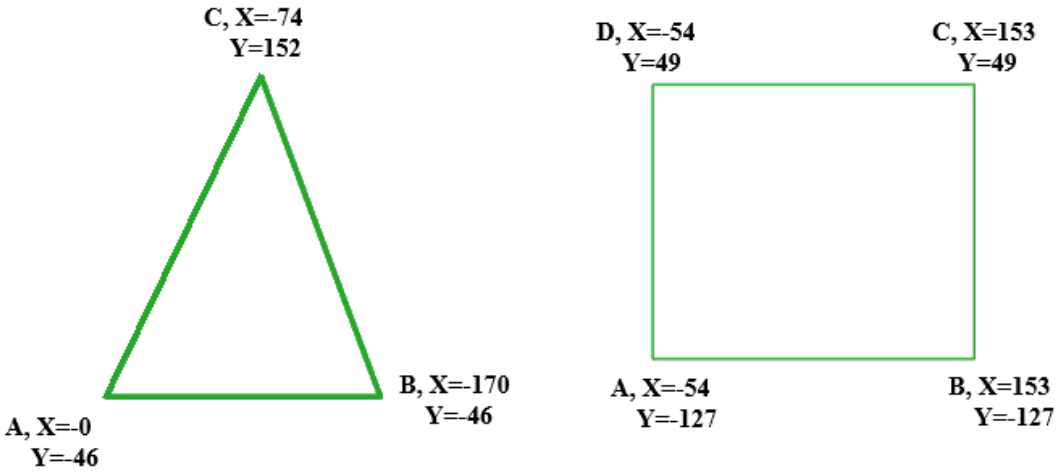


Figure 147: Path of sprites

When you are done with the excise, use this link to compare the answers.

MODULE 6

SCRATCH GAMES (1)

OVERVIEW

In this module, you will learn about designing games in Scratch. You will learn how to create variables and Lists of data to store information and characteristics of sprites. These are key skills that you need in the creation of games. Lastly, you will learn how to use sensing blocks to detect a certain movement or event to trigger an action.

LEARNING OUTCOMES

By the end of this module, you will know how to:

- apply the computational concepts of conditionals, operators, and data (variables and lists).
- apply the computational practices of iterating, testing, debugging, and reusing, abstracting, and modularizing by building game projects.
- Identify and use common game mechanics.

6.1 DESIGN PROCESS

The Design Process is an approach for breaking down a large project into manageable sections. The process defines each step to tackle each project and reminds us to sketch all our ideas throughout the process. This process is used to design and develop a game in Scratch. We need to break the project into small parts to get a better understanding of each section and ease the implementation process. In the process of designing the game, there are 4 things we need to ask ourselves.

1. **What is the objective of the game?** The goal of the game needs to be well defined.
2. **What is the operation process of the game?** How will users play the game we are designing (i.e. Will they use keyboard arrow keys to move? Will they press the spacebar to jump or shoot? Will they need to move the mouse around? Etc.)
3. **What are the challenges to overcome?** (i.e. getting rid of bad guys, a timer counting down, running out of fuel, life meter, etc.)
4. **What to let the user know s/he won or lost?** (i.e. “Congratulations, You Won!” message on the screen, “Sorry, You Lost” message screen, celebration dance, etc.)

6.2 INTRODUCTION TO VARIABLES AND DATA

6.2.1 Variables

In computer science, variables are a named location where information can be stored. A variable is a container that can hold one piece of information at a time, like a word or a number. Being able to hold this value allows us to refer to it and manipulate it at different places in a program.

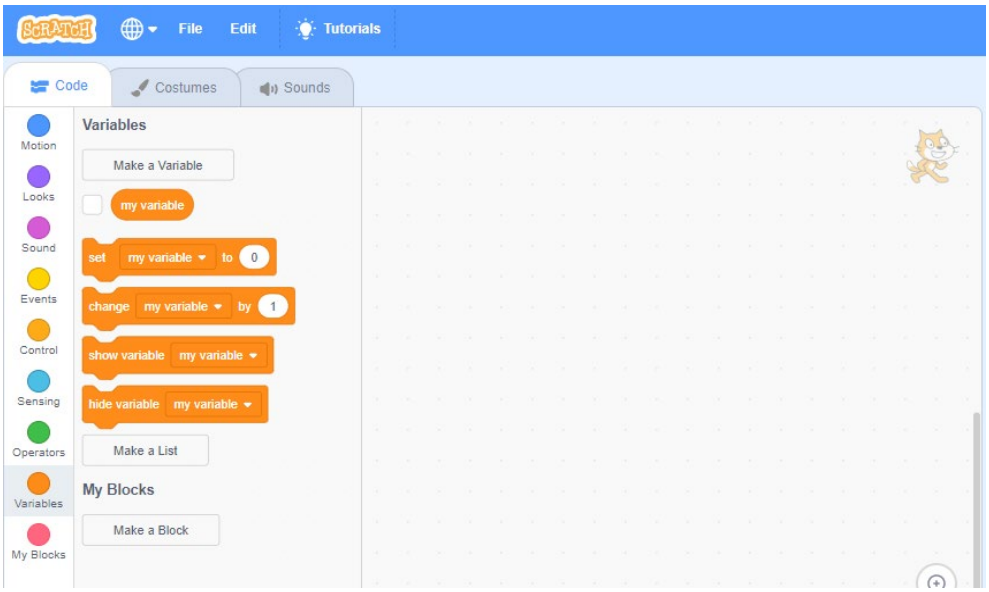


Figure 148: The interface of the Variables block

Scratch has provided an easy way to create variables. A set of blocks is provided to manipulate variables that you created. In Scratch and many other programming languages, there are two ways to use a variable. You can store information in it, or you can read what is already there.

The Variables block is divided into **2 sections**: the **stack block** and the **reporter's block**. Initially the reporter's block has 1 item while the stack block has 4 items (Figure 148).

Figure 149: Reporter and Stack block

Before using variables in a Scratch project, let us first create our own variables.

6.2.2 Creating a Variable

We start by defining the information we want the variable to hold. That will come from defining what you want to use the variable for and how you are going to use it. For example, we want to add a time limit to the game, by creating a variable that will start at a given number and then count down by 1 every second. When the number gets to zero, the time is up!

- **What?** We want to create a timer.
- **For what?** To set a time limit to the game where the time will start at a given number and decrement (decrease by 1) every second until it gets to zero.

Guided activity: Creating a variable

After defining this, we can start to create a new variable.

1. Select the Stage and go to the Variables blocks. Click on the Make a Variable button to display the New Variable dialog box.

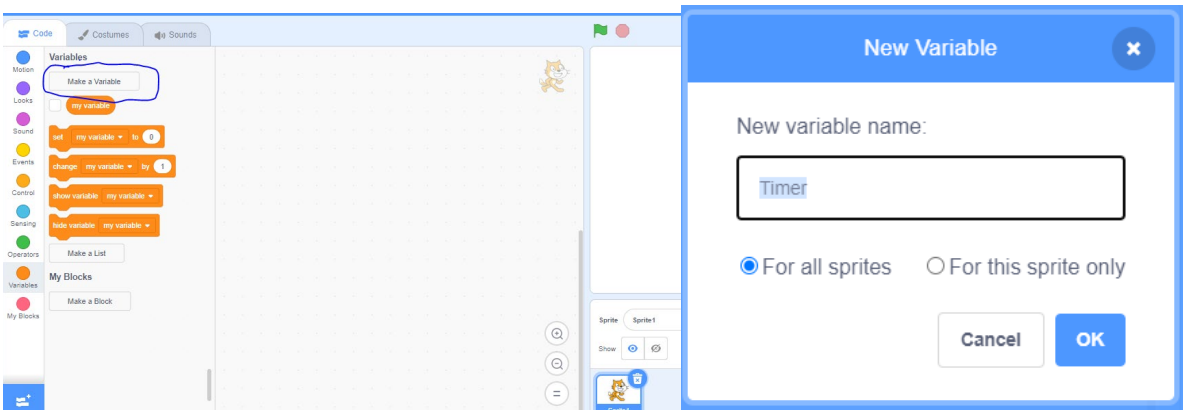


Figure 150: Creating a variable

2. Type in the name of your variable, "Timer" in this example.
3. Choose if the variable will be for all sprites or for the sprite you are creating only. That introduces the difference between **global** and **local** variables. So, you have come to the crossroads: do you choose for all sprites or do you choose for your sprite only?

The choice you make (for all sprites or for this sprite only) will affect the use of the variable in the whole project.

If you select for all sprites, your variables become **global**, meaning that it can be changed or accessed from any sprite in your project, regardless on which sprite it was created. On the other hand, if you choose for this sprite only, your variable will become **local**. A local variable is one that can only be changed or accessed from the sprite on which it was created.

For example, the internet is global, and anything saved online can be accessed using any computer in the world! However, if you save something to the hard drive of your computer, you will not be able to access it on a different device as it was saved locally.

4. Click the OK button.

5. Your variable will now be displayed on the stage, and variable blocks will appear that you can use to manipulate it.

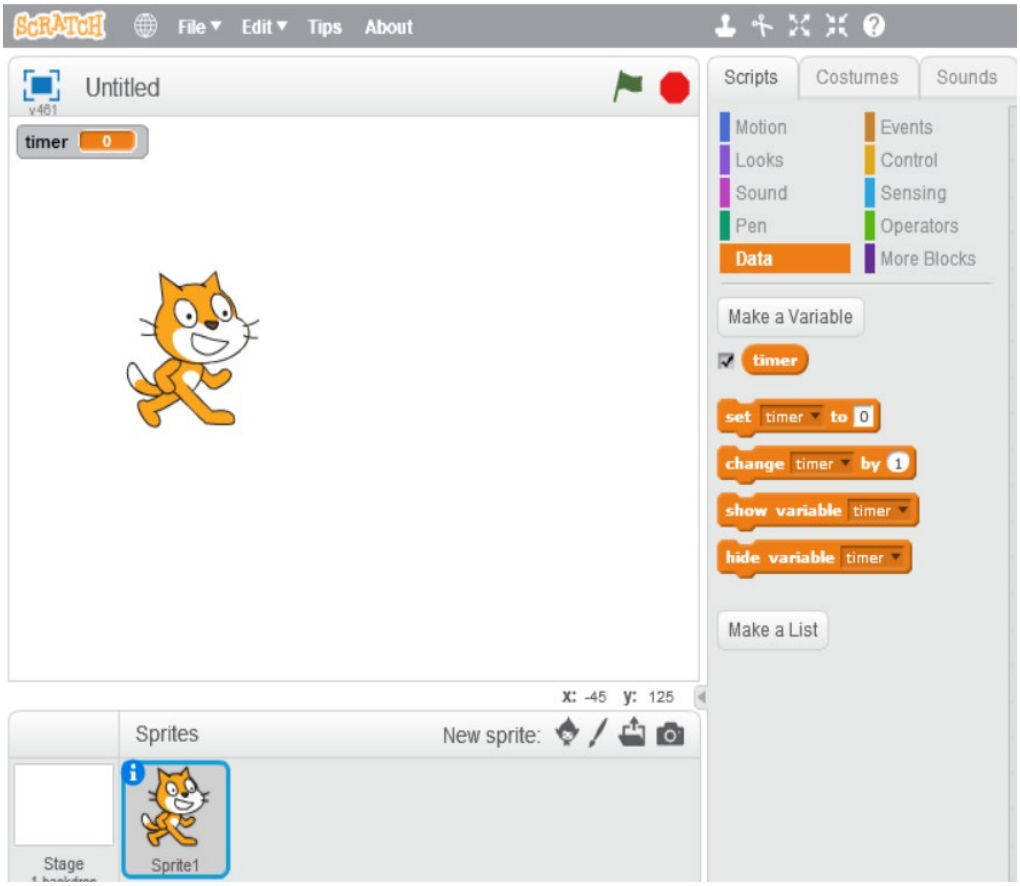


Figure 151: Displaying a variable on the stage

6.2.3 Using Created Variables

To explore how to use variables in Scratch, we are going to make a program that generates two random numbers. The Scratch Cat will then report what these two randomly generated numbers are and tell us if they are the same.

Guided activity: Using a variable

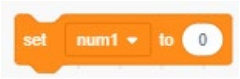
Let's create variables ***num1*** and ***num2.***

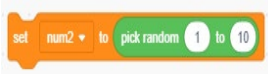
Figure 152: Creating the variables num1 and num2

Let's set variables ***num1*** and ***num2*** to be random numbers between 0 and 10 every time the flag button is clicked.

To do so, we need to follow these steps:

- **Step 1:** From Events, pick up the block 

"when flag is clicked"
- **Step 2:** From Variables, pick up the block 

"set variable to 0"
- **Step 3:** From Operators, pick the block "pick random from 1 to 10" and let it replace the value 0 in the block created in step 2 

as shown in Figure 152.

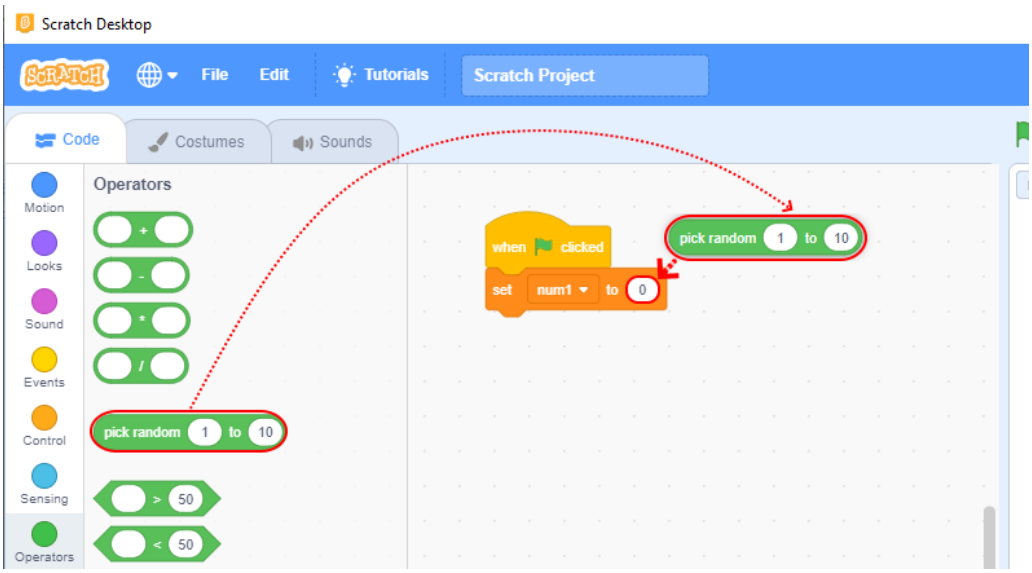


Figure 153: Picking a random number

- **Step 4:** Duplicate the block "set num1 to 'pick random from to 10'" created on the step 3.

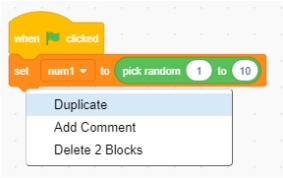


Figure 154: Duplicating a block

- **Step 5:** In the block created up to the step 4, replace num1 by num2.

- **Step 6:** On the stage the output of ***num1*** variable is overlapped with the output of ***num2*** variable. Now drag and drop the output of num2 variable from above the output of num1 variable. That is illustrated in the figure below.

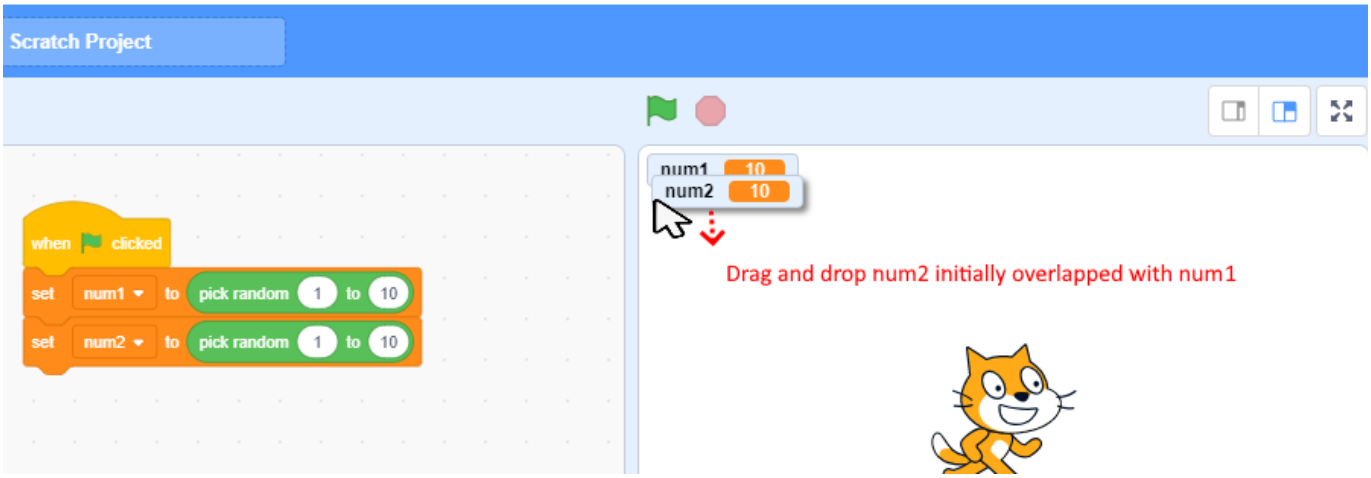


Figure 155: Num1 and num2 are given a random value

We can see in our code that we have created two variables: “num1”, and “num2”. Both variables have been set to pick a random number between 1 and 10. Therefore, when the green flag is clicked, each of both variables will be given a random number between 1 and 10 to store!

Next, we need Scratch Cat to tell us what these two numbers are. We can do this using the following code (Figure 156).

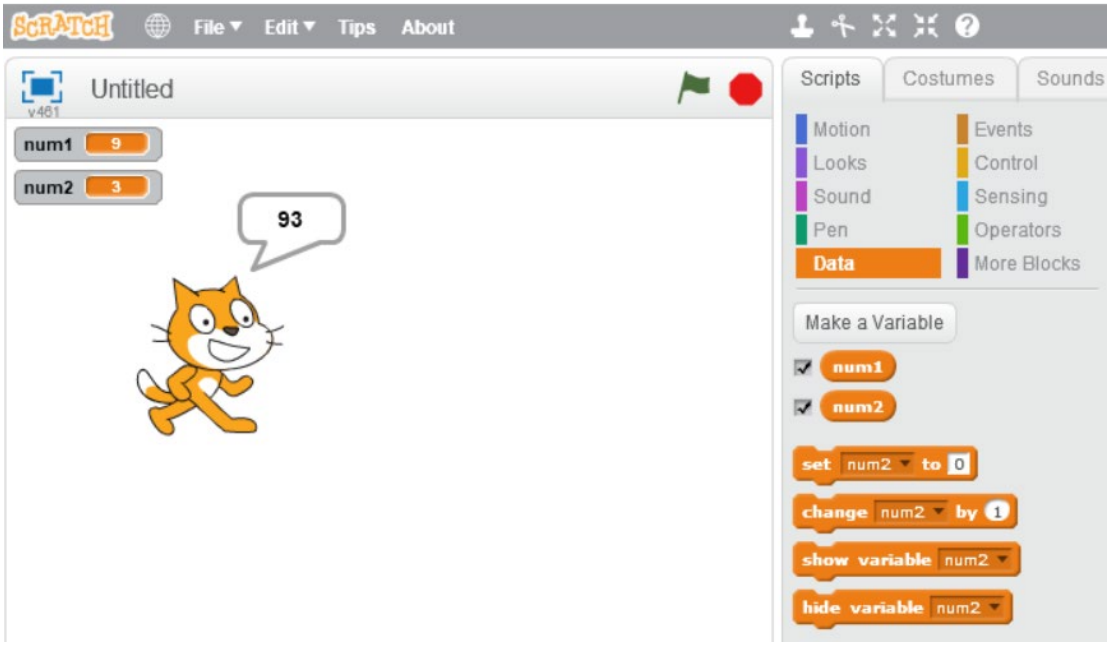


Figure 156: Cat saying the two random numbers

When a green flag is clicked, the cat will display the two random numbers. From what the cat displays, you can confuse the numbers and think that it is 93 instead of 9 and 31. With Scratch we will be able to correct that error.

To correct that, we will create some more code. We are going to join the string “Var1:” with its value. After that, we will have to join “Var2:” with its value as well. The cat is going to combine both values and display it at once.

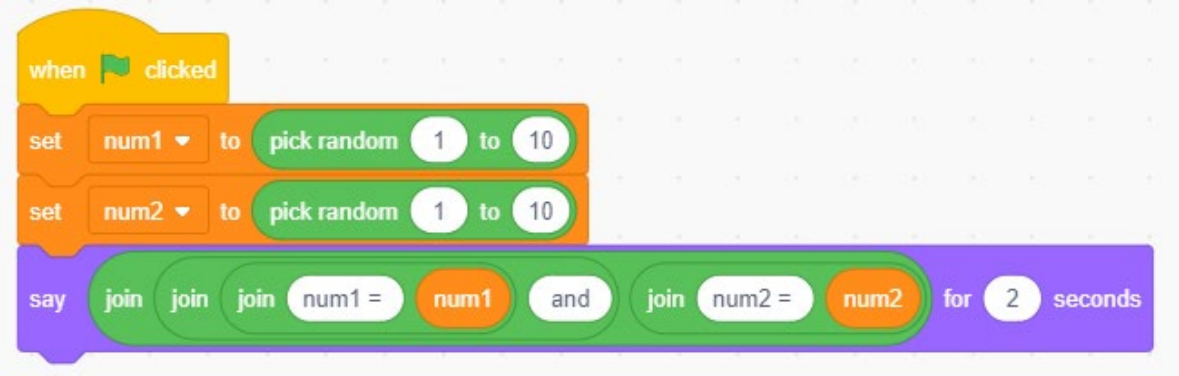


Figure 157: Joining Var1 and Var2 with its values

The output of the above program is shown above.

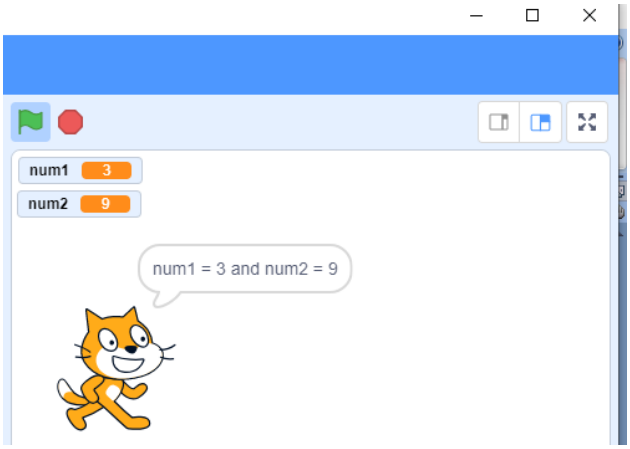


Figure 158: Output after joining Var1 and Var2

We show the values of both variables and then display them as Var1:2 Var2:10. It does not put some space in between but at least we can see some difference.

From the above examples, we have been using the global variable. We have stored some values and conditions to var 1 and var 2. We can use the values over and over. However, there is another option that we can use without storing this into variables.

Figure 159: Joining Var1 and Va2 without storing their values

This is a value returned by the function “pick random” and can only be used once. This means that you will have to build new codes if you need to use the same values. These two pieces of code produce the same result — but in the second example, the random numbers that are generated are not saved. So, yes, they exist in that one instance of the program, but if you want to use these same numbers at some other point in the program, you can’t — because you didn’t save their information in a variable!

So, by using the second example, it would be difficult if for example, we were to extend the program by getting Scratch Cat to compare the numbers and see if they are the same instead of stating their values. What if we want Scratch Cat to report it when we find a match? What if we want to keep track of the number of times the two random numbers match? It is obvious that we need variables to store values in Scratch to achieve these!

6.2.4 Data

While defining variables, we noticed that variables are used to store information for use in programs. Scratch can store only numeric values in a variable which can be dropped into any program block space. From that definition, data is unprocessed information. That is the data that can be transformed into information to be used in any program.

So far, we have seen the initial variable blocks that appear when you open the Scratch and click on variable blocks. After these initial variables, we can create a list that holds a set of variables.

To name the list, you click on “Make a List” in variable blocks and then a text area will show, and you can put the name of the list there.

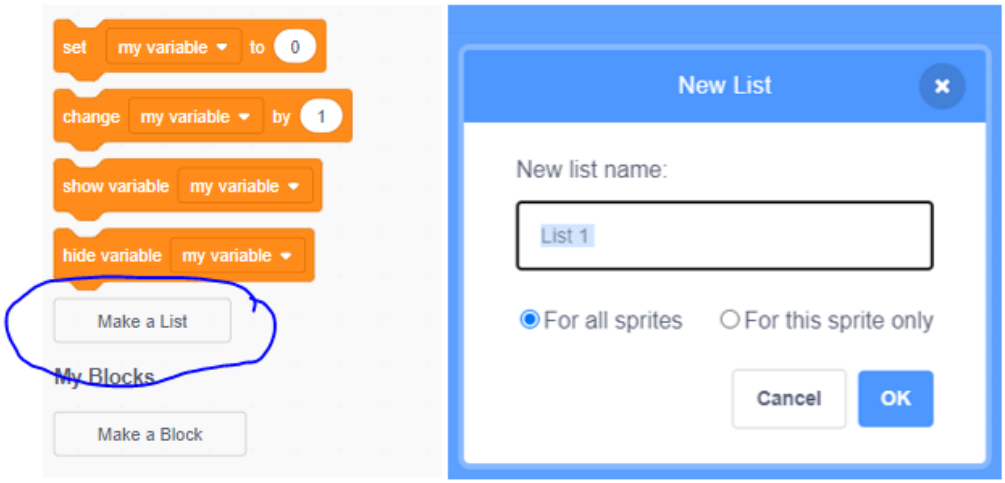


Figure 160: Adding a list

The list has 7 Stack blocks, 4 Reporter blocks and 1 Boolean block.

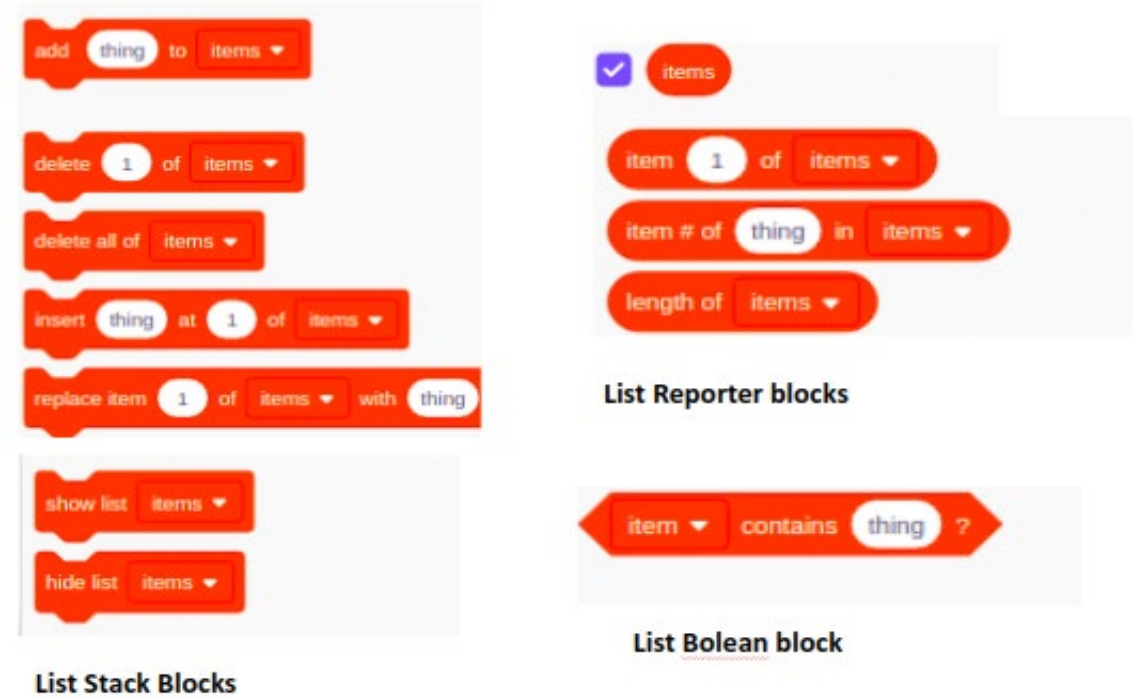


Figure 161: The list has 7 Stack blocks, 4 Reporter blocks and 1 Boolean block

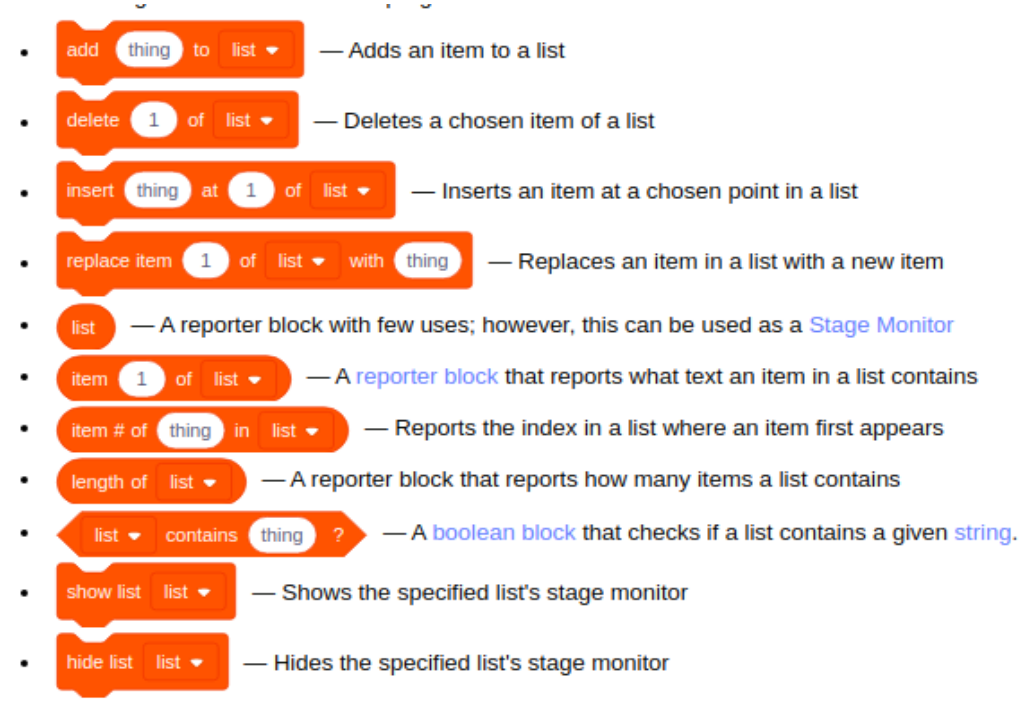
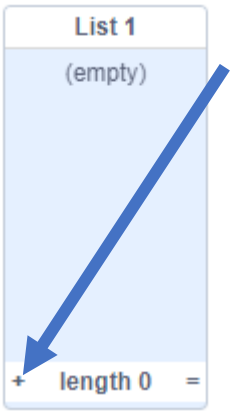


Figure 162: Explanation of list blocks

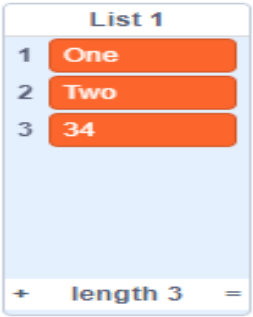
After creating that list, you must give it values. There are 2 ways to give values: to create those values within Scratch or to import values from your computer.

6.2.5 Creating Values of Lists

After creating a list, an option will appear on the stage.



On the down left side, there is a + symbol that you click to add any value that you want. It can be both numerical and nonnumerical. It just must be making sense in your own narrative.



After clicking on the + sign, a new space will appear to insert a new value. You can continue the process up to your limit.

Figure 163: Adding a value to a list

This is what you can do when you are creating values in Scratch. There is a second option that allows you to import data from your computer. Scratch reads the “tab delimited” format, which implies that you must convert your data to that format.

Here are the guidelines of how you can convert the spreadsheet into **“tab delimited”** format.

1. Copy and paste the data into a Notepad file. Open the notepad file and copy from the file you have pasted the data there. Afterwards, you will need to save and import to the Scratch.

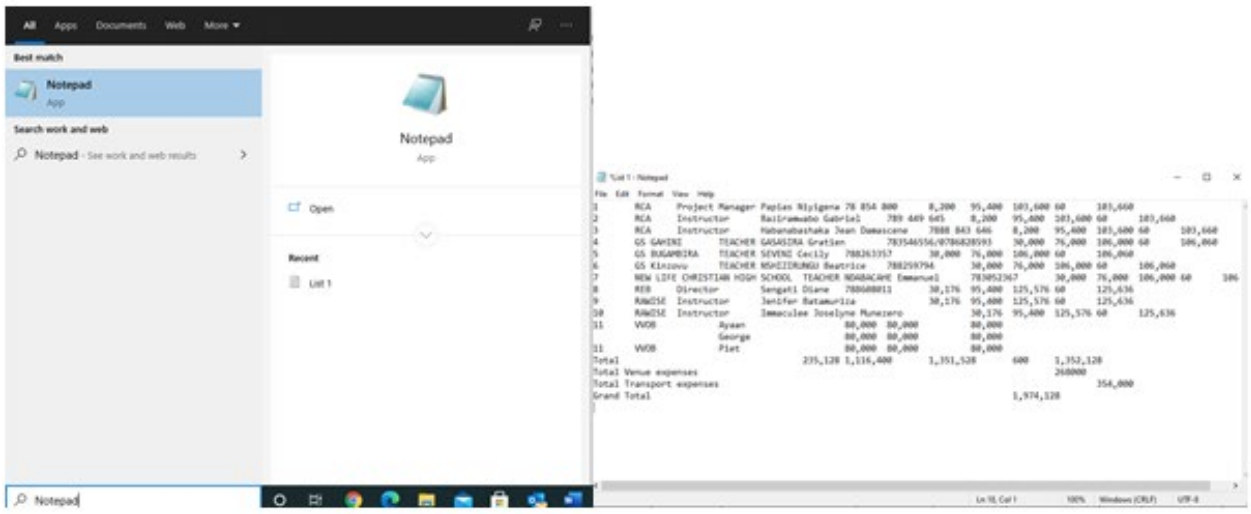


Figure 164: Pasting data into a Notepad file

If you are using a windows computer, go down to the search button and search for Notepad. Once appears, open the file.

Copy all the data from either the spreadsheet, word document or any other source and paste into the file. Afterwards, click on save and then upload to Scratch.

2. The second option is to convert the file into “tab delimited” format.
    a. Right click on the file
    b. Click on “Save as”
    c. Choose a location where to save the file
    d. Choose the file format to be “tab delimited” and then click OK

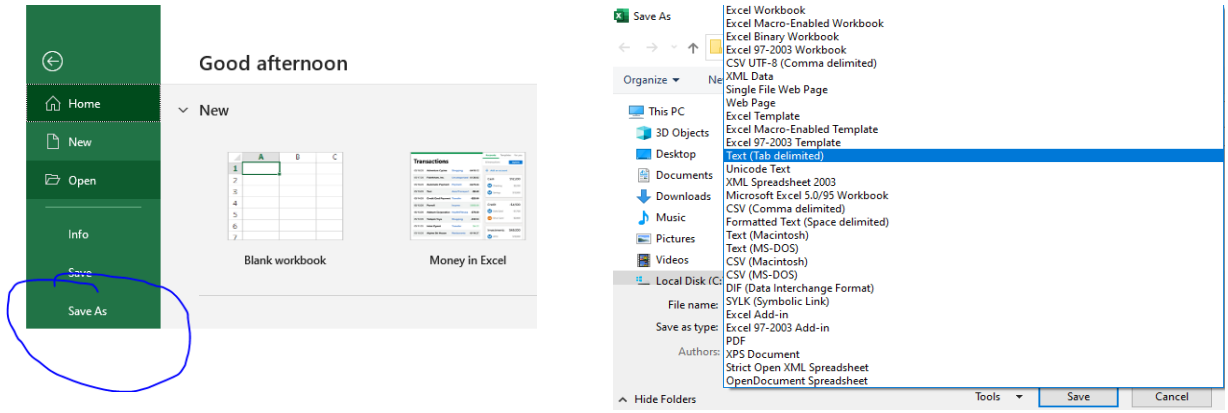


Figure 165: Convert a file into "tab delimited" format

After converting the file into “tab delimited” format, upload to the Scratch choose which column that holds the data you want to use.

Data is stored into variables to be given as values for any set condition. To understand how data is used, you need to start with variables. They are just ways to name and store data.

For example, if you had a game and wanted to keep a score, you would create a variable called score. You should always initialise variables.

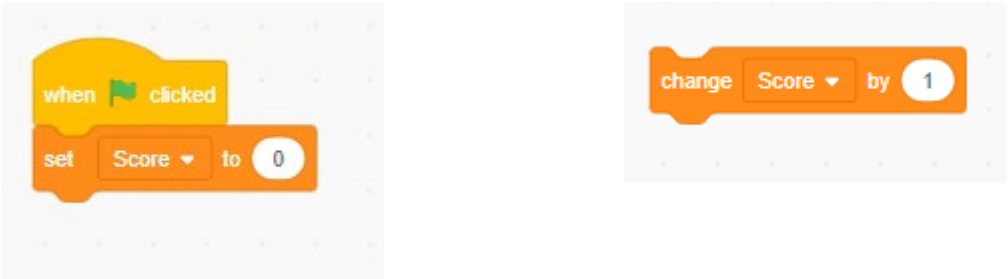


Figure 166: Setting the score to 0

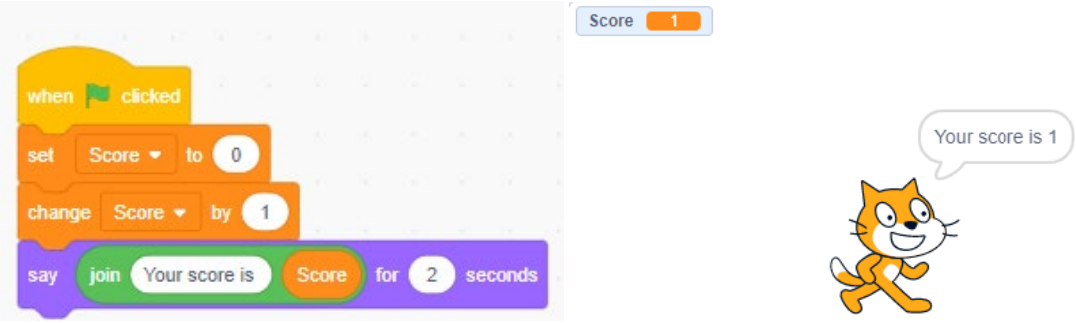


Figure 167: The Sprite reading the score for 2 seconds

Variables don't have to store just numbers they can also store letters and words. To make it a word or letters, you only need to replace the score with any letter or word that you want.

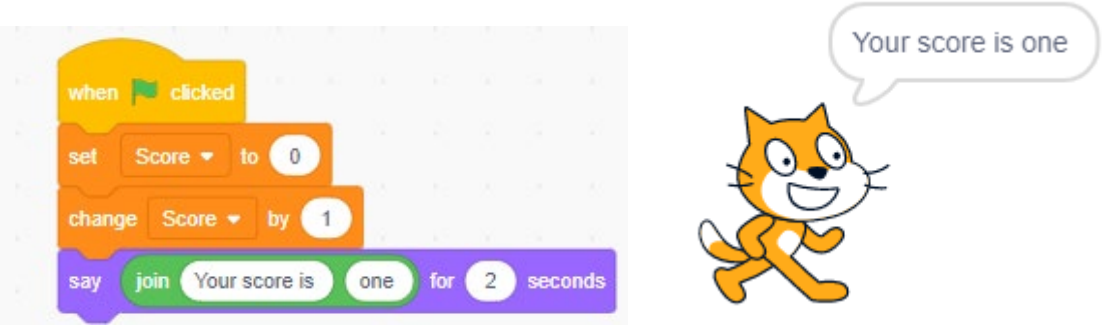
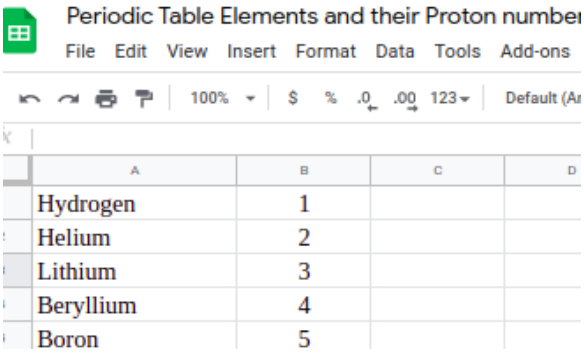


Figure 168: The Sprite showing the score

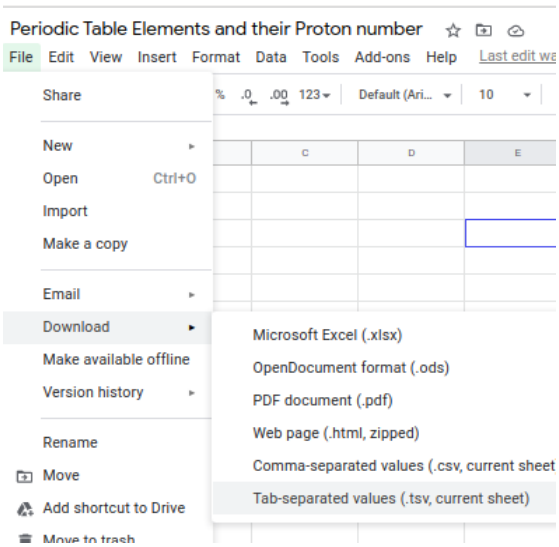
Literally, this is how you can use data and variables and it is more applicable in creating games and stories. We have seen how to use variables and data in creating stories in the previous modules. In this module, we are going to look at how we can use variables and data in building games.

Guided activity: Creating a list with values

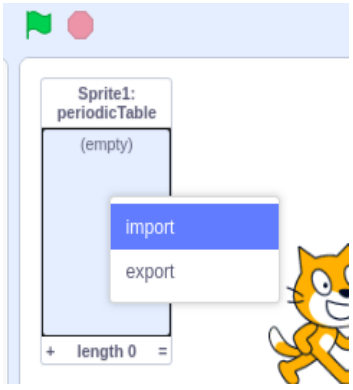
1. Create a list and call it Periodic Table.
2. Create an excel sheet with Periodic Table elements and their Proton Values (the number of protons in their core) as the second column.



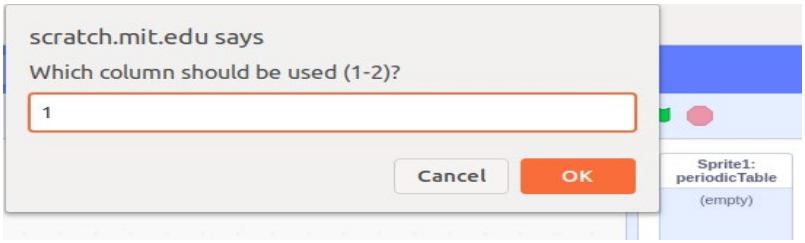
3. Save the sheet as Tab separated values document.



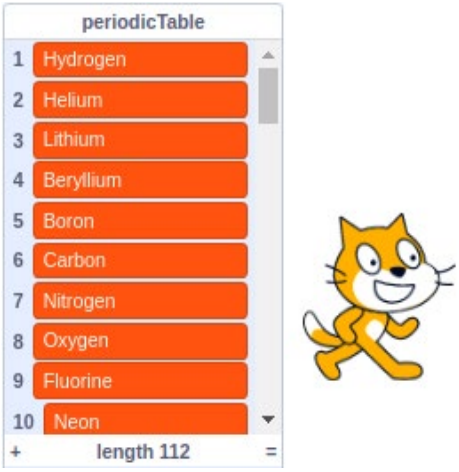
4. Right click on the list then select import.



5. Select the Tab separated values file and select the column to use and press ok.



6. The list is imported as shown below.



6.3 GAME DESIGN

In game designing, the process is the same as the design process with a focus on games. For game design we refer to this Game design Template.

6.3.1 Creating Characters

In this section, we are going to learn how to create a character that suits your story. A character is a person or any other entity acting in a game or a story. A character is a main component of a game. In most cases, the story is describing the character, what the character does, likes.... Each game has its own characters, and they should work exactly as the rules or scripts suggest. In Scratch, there are many ways to create a character.

6.3.2 Inserting a Sprite

Let us briefly recap how you can add a sprite on the stage.

1. Go to the right side of your screen and click on the small icon just below the stage.
2. Choose a sprite.

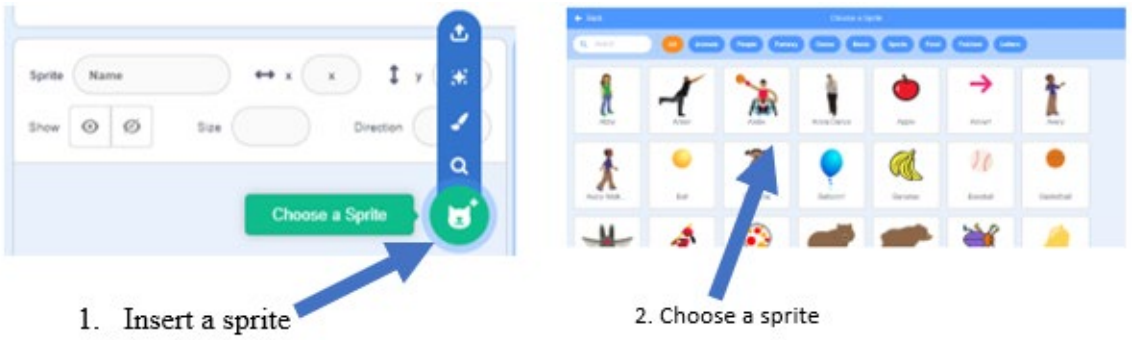


Figure 169: Inserting a sprite

3. Change the costume and manipulate the sprite as you want. Refer to module 1 to understand how you can change the costume.

6.3.3 Creating a Sprite

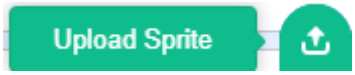
There are two ways to create new sprite:

1. Click 

to paint a new costume in the Paint Editor. You can also take a picture.

Click 

to take photos from a webcam (built into or connected to your computer). Each time you click the button (or press the spacebar), it takes a photo.

2. Click 

to import an image file from your hard disk.

After inserting or creating a sprite, you will use that sprite to create a game and a story. In the game plan, you have described what each sprite is going to do. That is a good time to see if the sprite you described is already on the stage. In the process of instructing a sprite to do anything, you will need to use the sensing block to do that.

6.3.4 Inserting Touch Detection

The Touching block is a sensing block and a “true or false” block. The block checks if its sprite is touching the mouse-pointer, edge, or another sprite (a reporter block which returns the sprite's name, usually a variable can be used). If the sprite is touching the selected object, the block returns true; if it is not, it returns *false*. We are not going to provide more details about the touching block since we covered it in module 4. We are going to focus on how we use the touching block in creating games.

- We are going to create a game which uses variables to calculate the lives and score.
- In the exercise we are going to use the sensing to effect change in a game.

We will use 2 sprites. One will be called “Cat” and the second will be called “Mouse”.

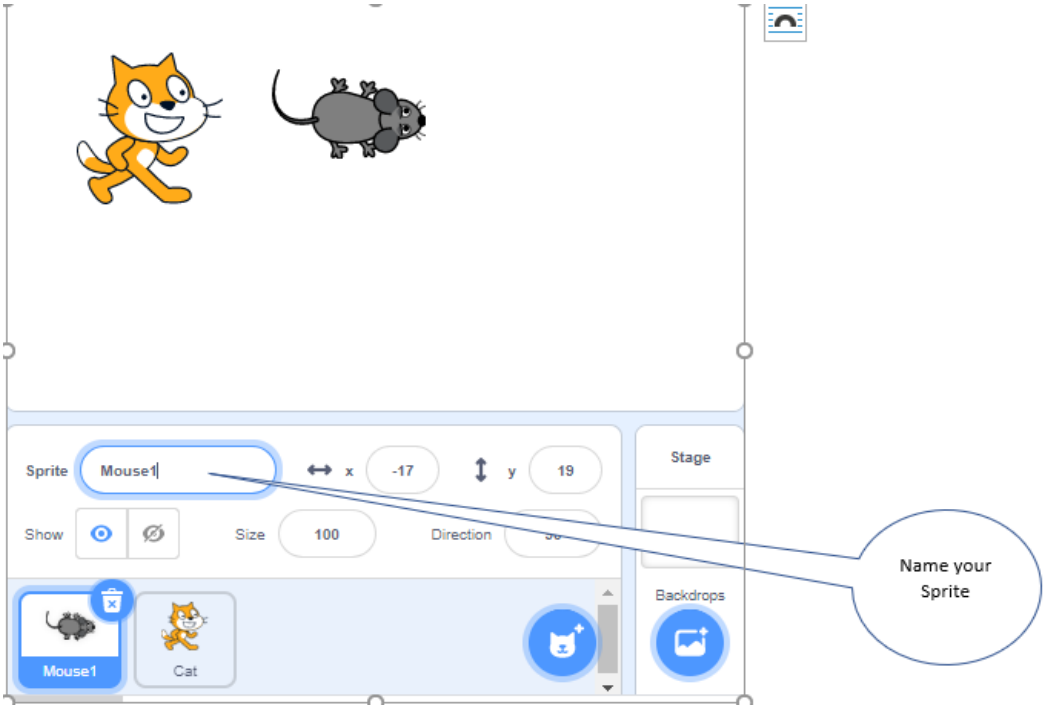


Figure 170: Naming a sprite

We want the cat to chase the mouse and when it touches the mouse, the cat will say “Thank you God for this food” and the mouse will say “I am dead”. This will be possible when we use the “forever” loop

and the “if” condition

.

Here is how your code will look like.

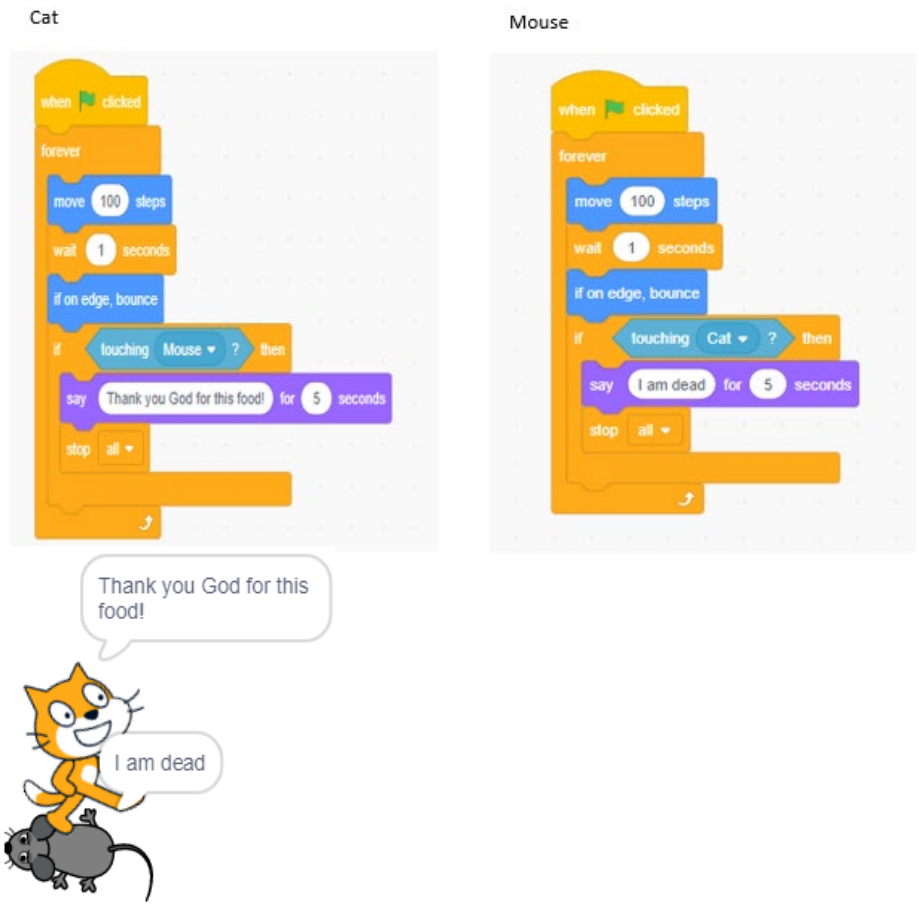


Figure 171: Code to let the sprites chase each other on the stage

With this code, both sprites will run 100 steps and bounce when they touch the edge. When the cat catches the mouse, the mouse will say “I am dead” and the cat will say, “Thank you God for this food!” Then the game stops. Sensing gives a true or false response. We can then create scripts which do different things if the answer is true or false.

6.4 ADDING A SCORE AND TIMER TO THE GAME

To add the score and the timer, you must understand the need for pauses between actions within loops. Then you will have to understand the timing of each action and the use of the sensing block to trigger the next action. We are going to use an example of a game to get more understanding of how each component works in the game creation.

We are going to use the same example as we used in the above section.

The cat will chase the mouse and when the cat catches the mouse, the game is going to record the score and give a meow sound and say some “Thank you God for this food”. The game will continue recording the number of times the cat will catch the mouse for 10 seconds. During the game, the timer will be set and count to 10 seconds and the game will stop and check the score. The winner is the one with the highest score at the end of the game

Cat

Mouse

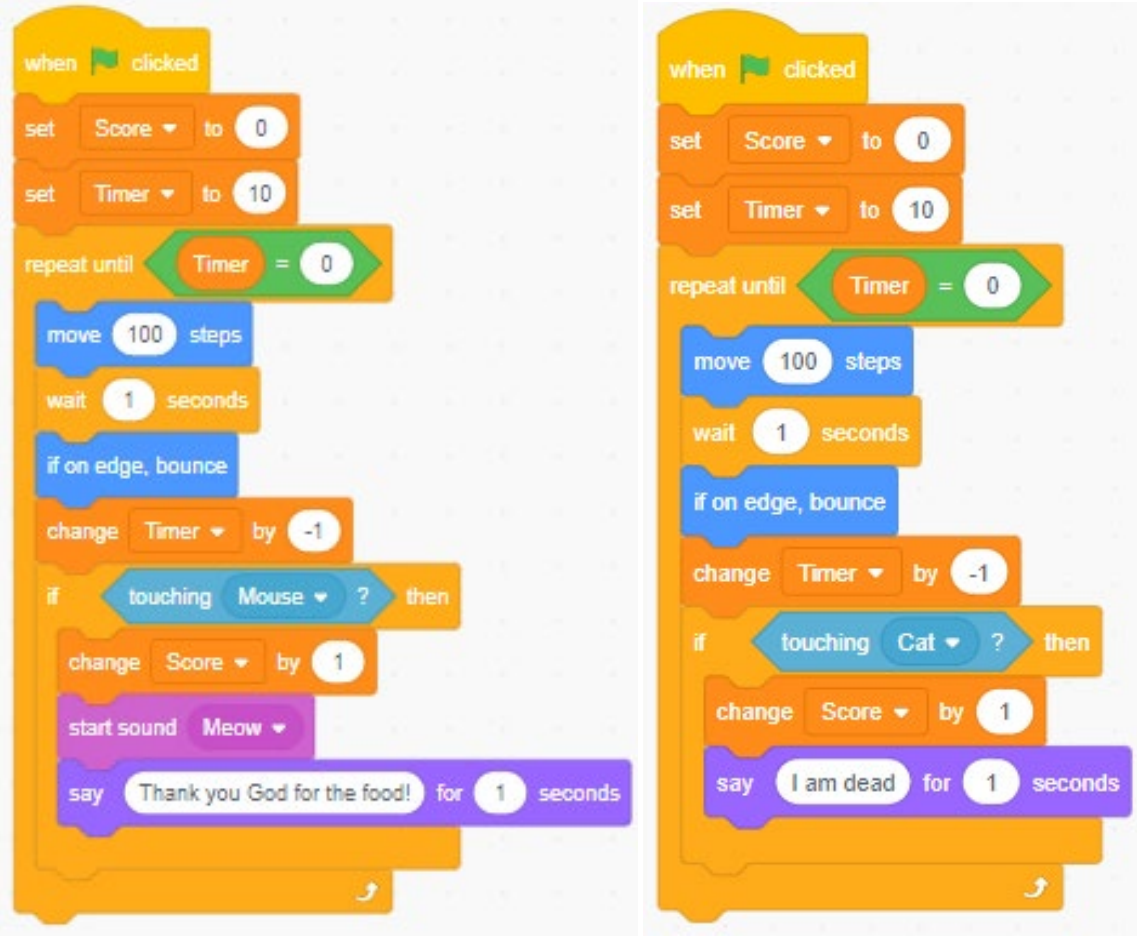


Figure 172: Setting a timer

We have created 2 variables. The score variable that is going to record the number of times the cat is going to catch the mouse. And then the timer that is going to contains the specific time in seconds of the duration of the game.

At the beginning of the game, we set the timer to 10 seconds, and then the cat and the mouse are going to run in the scene for 10 seconds and the game will record how many times the mouse was caught. The time will be removed (-1) 1 on the set time every after one second.

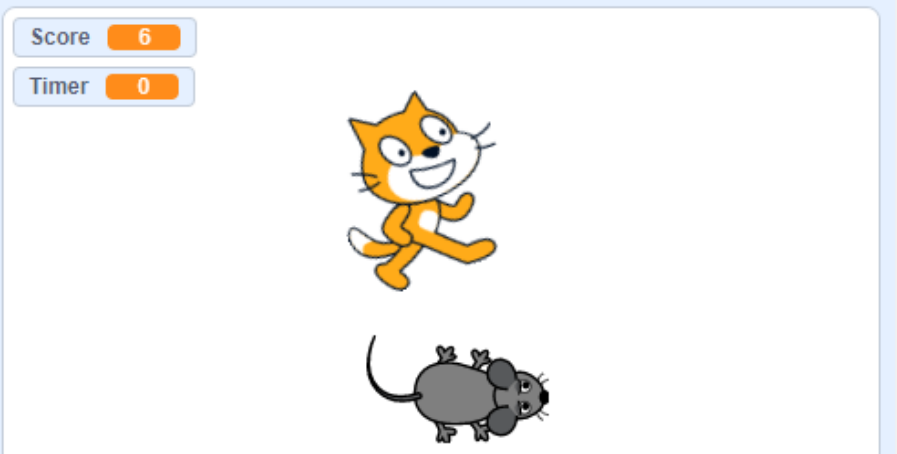


Figure 173: The mouse was caught 6 times in 10 seconds

This means that the mouse was caught 6 times in 10 seconds. That is how you create a variable and use it in the game creation. There are a lot of options created depending on the nature of the game.

END OF MODULE 6 ACTIVITY

In this module we have learned about the process of designing a game. The first step is to design the game. We are going to use following questions to design that game.

Ask & Imagine

- What is your project about?
- How will your project work?
- What are the rules?
- What are the goals?
- How do you start?
- How do you end or win?

How will the Stage look like?

Which Sprites will you use in your game?

What will the player need to do? Describe what happens when it happens, how to win, etc.

Background music

Sprite	Sound effect	To be recorded
1 **2**		

Visit this link to get more game examples.

MODULE 7

SCRATCH GAMES (2)

OVERVIEW

This module is the continuation of module 6. In this module, you will build on the game design process. You will learn how to manipulate data and variables with random numbers to create a game logic. In addition to designing a game, you will study how to increment and decrement variables to trigger and report the game results. Lastly, after equipping yourself with these skills, you will practice what you have learnt by participating in a hackathon.

LEARNING OUTCOMES

By the end of this module, you will know how to:

- apply the computational concepts of conditionals, operators, and data (variables and lists).
- apply the computational practices of experimenting and iterating, testing, and debugging, reusing, and remixing, and abstracting and modularizing by building and extending a self-directed maze, pong, or scrolling game project.
- use common game mechanics.
- create a game which uses variables and sensing to effect changes.

7.1 GAME DESIGN PROCESS (2)

In module 6, we have looked at the game design process. We will go into more detail of designing a game and focus on the 3 first stages (Ask, Imagine and Plan) of the design process. The figure below

shows the stages of the game design process.

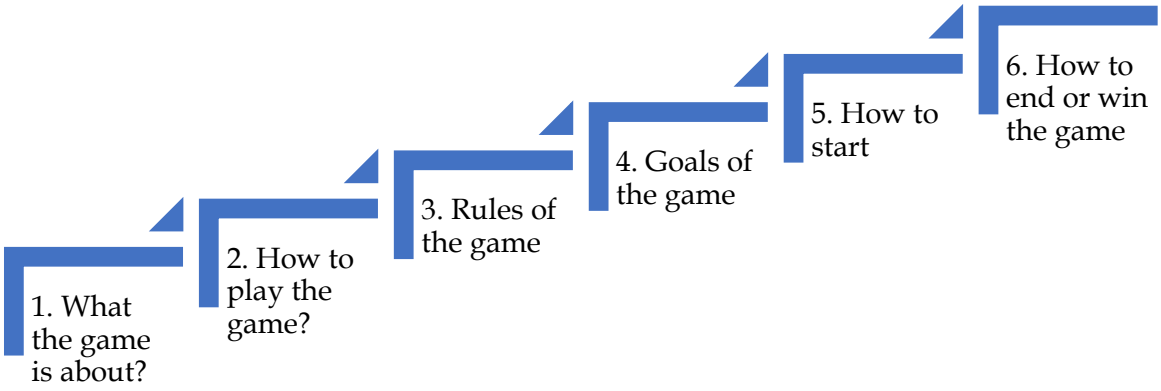


Figure 174: Steps of the game design process

The first thing to do in the game design is to understand what your game is about. That is where you look at the general overview of your game and describe the game from the name to all components of that game. For example, you are designing a chess game and therefore you are calling it Chess.

After you have decided what your game is about, you must describe how the game will work. If your game is chess, you are going to describe how one person will play the game until he wins the game. That is describing how your game will work.

In the next step, you will set the rules for your game. The rules will guide players of your game in what they can do and what they are not allowed to do. The rules will also help in determining when you win the game.

Next you will define the goals of the game. You have to describe what you expect from people to win the game.

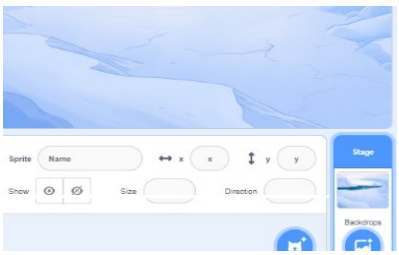
In all games, there is always a button to click to start the game or any other action that triggers the game to start. The next step of the game design process is to decide on what will trigger the game to start and describe that so that players understand how to start the game.

The last action of the game is to think of how someone will lose or win the game. After that, you will need to clearly describe all those steps that will make you lose or win the game.

7.2 TECHNICAL GAME PLAN

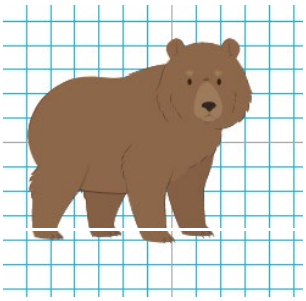
In Scratch there are some important elements that you will use in the game design. They include, sprite, stage, costumes, block plates, to name but few. In the plan, you must think about which elements to use and how to use them. That is the technical plan.

7.2.1 How the stage will look like



In the previous modules we have looked at how to create and add a stage. But it is not all stages that would work in any story of a game. Each game needs a matching stage. A stage adds a meaning to the game and story. So, always to choose a stage that matches your story or game.

7.2.2 Which Sprites to use



A sprite is a character in the story or an actor in the game. For each story or game, you will choose or create a sprite based on its role, characteristics and other factors.

7.2.3 Game Flow

After you have identified the sprite and the stage, you need to determine what the player will do. Which buttons to click and how to start the game. Most people use when you click the green flag, but you can choose other options. You describe what will happen and when it happens from the beginning of the game to the end. In the game flow, you also mention how to win your game.

7.2.4 Sound Effects/Music

In some games, there are sounds played in the background and each sound affects the game in one way or another. There are cases where each sprite produces a sound. In this section of planning, it is better to assign each sprite a sound and if possible, record or upload what you already have. In that category, you can provide each sprite with an associated sound effect.

7.3 LOGICAL OPERATORS

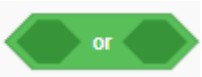
In module 1, we have looked at operators and how each operator works. Now we will look at how logic operators can help you in building exciting games.

7.3.1 Using Logical operators in game design

With logical operators, you can combine two or more relational operators to produce a single true/false result. Relational operators compare the relationship between two values or variables (great, less than, equal) and decide if it is true or false. They are joined by logical operators. Using these blocks allows you to further refine your comparison of values.

There are 3 logical operators:

The result is true only if both expressions are true



The result is true if one of the two or both expressions are true

The result is true if the expression is false

7.3.2 Periodic Table game

The Periodic Table is the organized array of all chemical elements in order of increasing atomic number. The number of each element corresponds to the number of protons in its nucleus.

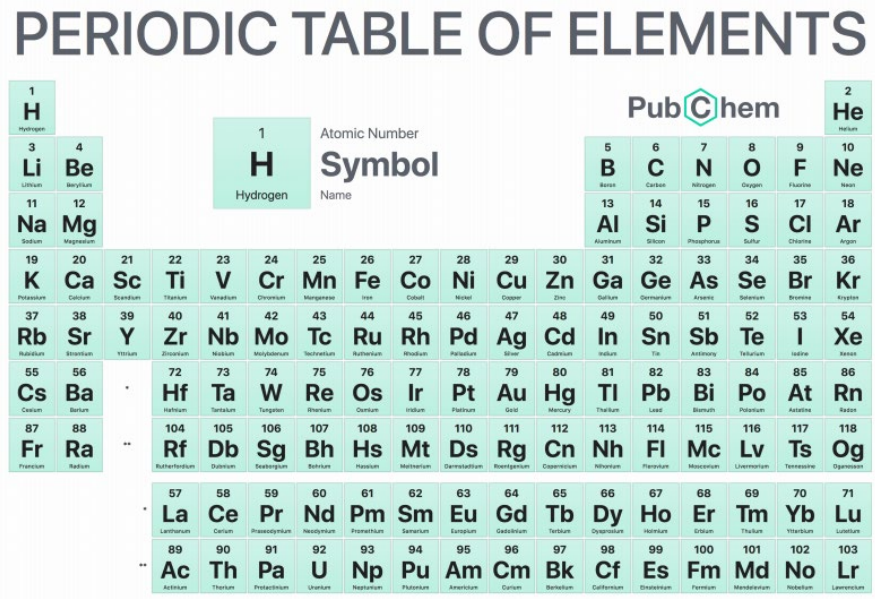
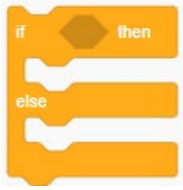
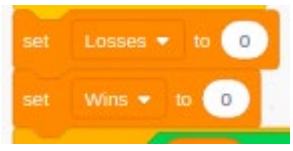
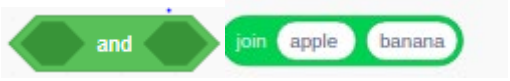


Figure 175: Periodic Table of Elements

- **About the project?**
  This game is going to help students master periodic table elements and their corresponding proton number.
- **How will your project work?**
  Students will get a random element and they will have to enter its proton number.
- **What are the rules?**
  Fill in the Proton number/atomic number that corresponds to the selected element.
- **What are the goals?**
  The goal is to see if you master the periodic Table elements and their atomic number.
- **How do you start?**
  To start the game, you need to press the Letter **"P"** on the keyboard.
- **How do you end or win?**
  If the answer is correct you get one point, otherwise you lose one point, and five mistakes will mean the end of the game.

Instructions

To build this game, you will use different blocks:

Blocks	
	Use the Press **"p"** event block to start the game.
	Use a control block (if then, else) to set conditions to fulfil and if not something else happens. It's like using "True" or "False"
	The first step is to create your variables and give them different values.
	We will use logic operators and Operator reporter blocks together with the controls
	List blocks

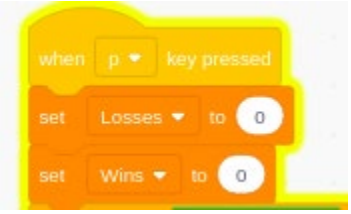
	The Looks block will give the sprite something to say or help in displaying any value attributed to the sprite.

Figure 176: Blocks to use in building the Periodic Table game

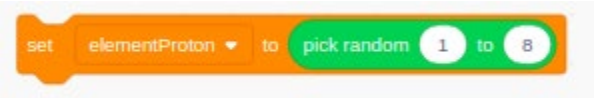
The Stage and Sprite:

Step 0: Import the Penguin sprite and make the backdrop shown above. Add an instruction to the backdrop to start the game as indicated above.

Step1: The first step in creating this game, is to create variables. We have to create the first 2 variables called Wins and Losses. Wins and Losses variables are set to zero and they will be changing whenever the score changes.



Step 2: Create the third variable element Proton and set it to a random Number between 1 and 112.



Step 3: Make a list of Periodic table elements, see Guided activity 6.3: Creating a list with values.

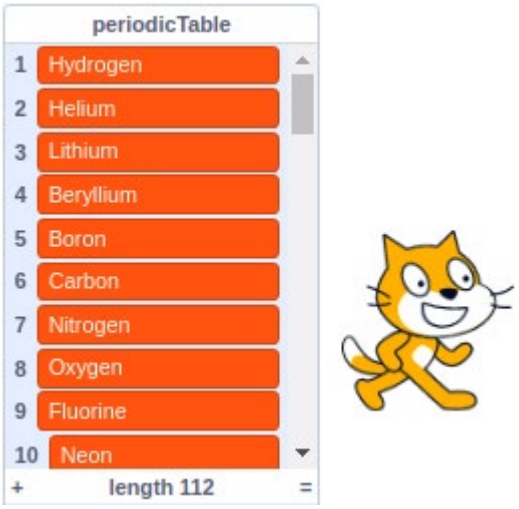


Figure 177: Creating a list of chemical elements

Step 4: Use a join block to ask a Question of elements related to the selected random atomic number in the list.

Figure 178: Get an element using a random atomic number

Step 5: To check the answer is equal to the atomic Number randomly selected; if it is correct and provides feedback; when the answer is correct, the Wins will change by and when the answer is wrong the Losses will change by one as well.

To record the answer, we are going to use the if else control block to give a congratulations message when the answer is correct and a Fail message when the answer is wrong. Sprite costume may also change as well.

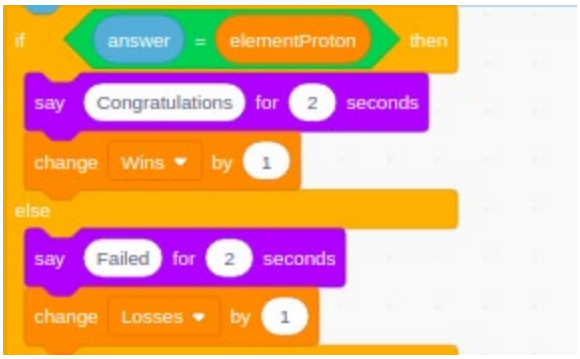


Figure 179: Checking answer and increment wins and fails.

Step 6: Set the loop with condition. Here below the game will continue until five mistakes are made.

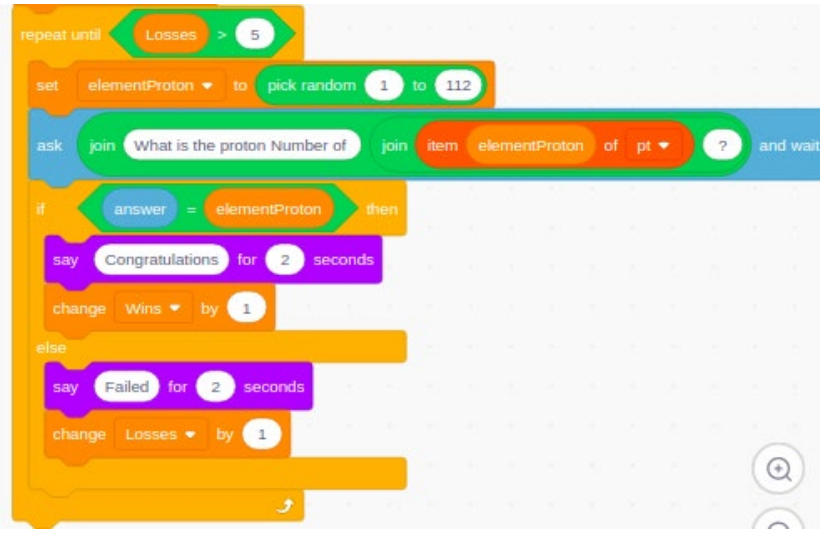


Figure 180: Looping until five mistakes are reached

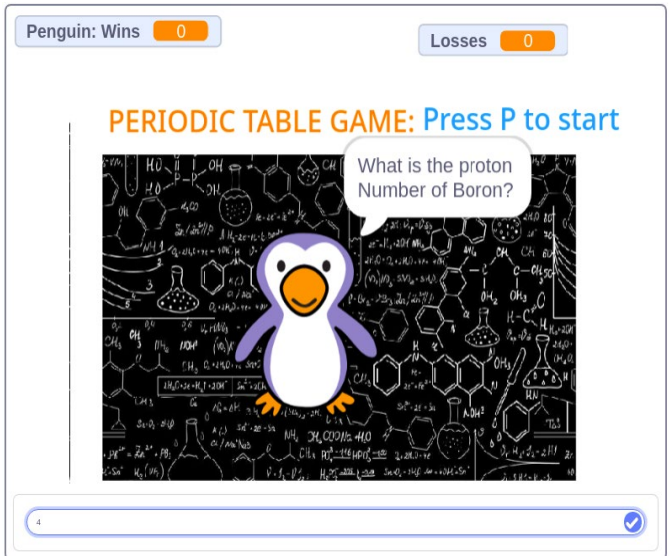


Figure 181: Game preview

URL: https://Scratch.mit.edu/projects/463160707/

SCRATCH 2050 HACKATHON GUIDE

A hackathon is an event where programmers get together for a short period of time to collaborate on a project. There are hackathons for all kinds of levels, topics and programming languages. Here is an example: https://younggates.com/hackathon/index.html.

1. **HOW TO REGISTER**

   A google form will be shared to Scratch coding clubs in Kayonza through respective schools by contacting headteacher and STEM teacher's representative.

2. **WHAT THE HACKATHON IS ABOUT**

   The hackathon theme will be chosen from STEM subject in secondary level. Below are some examples.

   Examples of Themes

   1. Animate a vehicle navigating from one location to another with the shortest path (Math, ICT).
   2. Animation for Chemical bonds-mixtures (Chemistry)
   3. Animation for circulatory system (Biology)
   4. Electricity to show how electrons move in each circuit diagram (Physics)
   5. Plotting a Quadratic equation to the XY Coordinates (Mathematics)

3. REQUIREMENTS

This hackathon is perfect for students Senior one to Senior six students in Kayonza District with some coding experience in their clubs. The hackathon will be done in Coding clubs in schools and a selection will be made at school, sector and district level.

4. EXPECTED OUTCOMES OF THE HACKATHON

Outstanding animations, games and stories will be developed and shared online all secondary schools in Kayonza District and beyond.

5. RULES

1. Basic Rules TEAM SIZE: One to three students can form a team.
2. Each team can choose only one topic.
3. You cannot submit a team project. Adults (instructors, parents, etc) can help students debug but most of the work must be done by the students in Secondary schools of Kayonza District.
4. If copyrighted materials are used such as photos, music, text, video, and other mediums, the author must be acknowledged and referenced.

5. Each team should have at least one boy and one girl.

6. HOW TO WIN THE COMPETITION

The winner is awarded marks based on the following:

1. *30% Innovation in STEM.*
2. *40% Scratch usage and creativity.*
3. *30% Presentation and QA.*

7. WHO IS ALLOWED TO COMPETE?

Students in Secondary schools of Kayonza District.

MODULE 8

PEDAGOGICAL INSIGHTS

INTRODUCTION

Coding clubs are communities of students that meet regularly after school to learn and exchange knowledge on Scratch. This module will help you facilitate a Coding Club for your learners and inspire the next generation to get excited about computing and coding through Scratch.

OBJECTIVES

The objectives of the clubs are to help students:

1. Develop the concept of computational creation through Scratch.
2. Use different concepts and features of Scratch.
3. Develop games, stories, and animations with Scratch.

4. Develop communication, critical thinking, problem solving, collaboration, and creativity skills.
5. Develop an interest in ICT and coding.
6. Understand the link between coding and the world of work.

8.1 PREPARING TO START YOUR CODING CLUB

Infrastructure

Scratch can be used both online and offline. Therefore, there needs to be some infrastructure in place before your club can start.

Computers

Students need access to computers to be able to use Scratch. For schools that have computer labs, that is a blessing but also for those that have no labs they can arrange how students can have computers to use. Since there will be 3 clubs per school, it is better to organise clubs on different days and reduce the size of a club just to ensure proper utilization of the computers and other facilities. In case where computers are still few, students can share a computer.

Internet

After getting computers, students will also need internet in the process of downloading and installing the software. Afterwards, they will still need internet to utilize online resources and get more understanding of the Scratch platform. Access to the internet will help them to interact with other people in the Scratch community. Once Scratch has been installed on a computer, students can access the program and design projects without access to the internet.

Projector

With computers, participants can work individually or in pairs. Having a projector to show examples and demonstrate how to get started will be very useful.

8.1.1 Preparations

You also need to make some practical preparation before starting with the clubs. First, you need to announce the launch of the coding clubs in your school. This announcement should trigger the interest of students to register and participate in the program. Here is a guidance on how to communicate about the clubs:

- Stress that the coding club is not a lesson. The purpose is to learn together about coding and work on fun coding projects.
- Stress that coding skills will help your learners in finding a job. Many interesting jobs require ICT skills and coding competences.
- Everyone is welcome to join the club. Learners do not need to have advanced ICT skills. They just need to be interested in learning to code.

- Make sure to encourage girls to participate in the clubs. At least half the members of your coding club should be girls.
- Clearly communicate which learners (grades, specialisations) can join the coding club.

Logistics

Here are some tips for the logistics of the club, but you could make some changes based on the situation of your school. Discuss with your head teacher, deputy head teacher and colleagues about the best way to organize the coding clubs.

- Club sessions are best organized weekly and last about 2 hours.
- Participants will get a certificate of completion from the school at the end of the term or coding club cycle.

Learning Goals

Concerning the learning goals:

- The club will have 12 weeks, with 2 hours of club meeting a week, that's 24 hours of club sessions. The time will not be enough to transform participants into professional developers. Those who are interested **can invest more time to continue learning**.
- The time will not be enough, but they will be equipped with **good foundational skills** in Scratch.
- Participants will be equipped with a basic understanding of **how programming works**.
- If they work hard, within a year from the start of the clubs, they can be very good at Scratch and programming.

8.1.2 Student Registration

After announcing the coding clubs, teachers should register students. The number of students per club will be determined by the availability of computers. Every semester, a club will start with new members so that all learners who are interested will have the opportunity to take part in the club. 3 clubs per school will be set up, each facilitated by a STEM or ICT teacher.

Every term, a new coding club cycle will start. Students who cannot join the first cycle, can join the second or third cycle. There are different ways to select students for a club. You can select them on a "first come, first serve" basis. An alternative is to give all learners a deadline for expressing their interest in joining a club and select members by lottery. Those who are not selected, can then join the next cycle.

8.1.3 Informing Your Colleagues and School Management

Before the start of the learning trajectory, there will be an informative session where all head teachers, sector education inspectors and all people involved in the education will be invited. In this session, they will be informed about the project and discuss the practicalities of how this project is going to be implemented. Afterwards, 3 teachers from each school will be trained. Before the launch of the clubs, the trained teachers will be required to raise awareness of the project and its benefits. Head teachers will be required to inform parents about the project and help in encouraging students to join the clubs during the launch.

8.1.4 Motivating Students to Join the Coding Club

You are encouraged to use Scratch in your STEM and ICT lessons. However, for deeper learning, learners need to join a coding club. In a coding club, students have many opportunities to use Scratch at different ages and grade levels. In order to learn how to express their ideas with code, students need to learn more than the basic grammar and vocabulary of coding. They need time and space to experiment with making different types of projects, such as interactive stories, games and animations. By exploring ways to combine their own images, words and sounds into online projects, they expand their ability to give voice to their ideas.

While launching the coding clubs, begin your speech like this:

*"Hello everyone, I am **(teacher's name)***

I participated in the Scratch trainings and I would like to share what I have learnt with you. Technology is growing so fast in Rwanda and in the rest of the world. As matter of fact, it is being integrated in all sectors. The education sector is also trying to integrate technology in teaching and learning.

We have launched the project called Scratch 2050. The aim of the Scratch2050 project is to equip 135 ICT and STEM teachers of 45 secondary schools in Kayonza district with the competences needed to initiate and facilitate after school Scratch 2050 coding clubs for secondary school learners and to integrate Scratch into ICT and STEM lesson plans. I have participated in the training and I want to start a coding club.

Coding clubs are communities of students that meet regularly after school to learn and exchange knowledge of Scratch. We are now on the stage of launching clubs. I would like you to take this opportunity to join this club and shape your future. We want you to have experience with technology when you finish your high school. That will prepare your life after high school and pave a good way to your careers.

Together with my colleagues, we will facilitate these clubs and we encourage you to join."

8.2 THE FIRST CODING CLUB SESSION

8.2.1 Practicalities

Give each member the opportunity to introduce themselves. Ask them to talk about their expectation from the club. Discuss with members of the club on the days and the time of the club sessions. There should be coordination between clubs since many schools will have more than 1 club.

After agreeing on the time and the days, the teacher should have the club choose their representative who will be facilitating the communication between club members and teachers. The representative will help in organising club activities.

Afterwards, you should help members set their codes of conduct. The coding club is different from a class, and members are responsible for the management of their club, the activities and what they want to achieve. As a teacher, you can provide ideas and guide them, but the format of a club enables more student-centred learning. Ideally, members arrive at a club session and start with a brief brainstorming, followed by collaborative coding on projects, and concluded with a sharing moment.

8.2.2 Setting Targets

Let members set targets at the beginning of the club cycle and write them down. At the end of the cycle, discuss with them whether they have achieved their targets. There are many Scratch coding clubs all around the world that can inspire you and your learners to set ambitious targets.

https://scratch.mit.edu/studios/3476938/

8.2.3 Setting Guidelines for Your Coding Club

It is best that you let club members discuss and set their guidelines. In this way, members are more likely to respect them. Let club members make a poster with the guidelines and hang it in the classroom where they meet, so they see the guidelines each session and can easily refer to them.

Possible guidelines are:

- Be punctual (do not be late, do not get absent)
- Do not misplace lab items
- Do not enter the lab without permission
- Complete assignments on time
- Only use internet for activities related to learning (Scratch)

To agree on the guidelines of the club, here are some techniques to use:

- Put students in two groups
- Allow them to brainstorm what they think would guide them

- Have one group present and the other complement

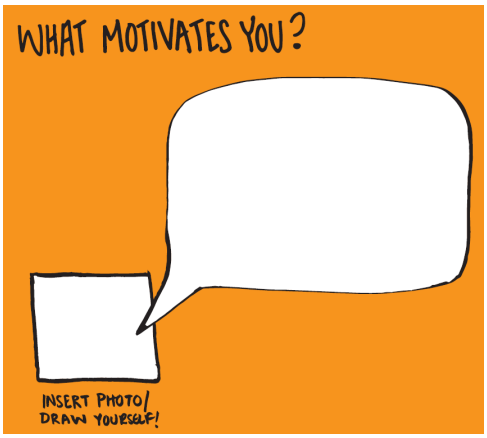
8.2.4 Introducing Your Coding Club

At the start teachers should make sure to remind students about these principles:

- Programming is not taught by teachers on blackboards. Programmers teach themselves.
- Programming is not learned by reading theory, but writing code/ moving blocks and understanding why it works or not.
- Programming is a collaborative effort. So is learning it.

You may show an intro video or a script like this one: https://scratch.mit.edu/projects/124863501/ to show to your students what they will be able to do at the end of the term.

Activity: Ask members to take a few minutes and write down what they expect from this club? After a few minutes, ask a few learners to share their ideas.



8.2.5 Setting Up a Scratch Account

During the first session of your club, members should sign up for their own Scratch account at Scratch.mit.edu, or you can set up student accounts if you have a Teacher Account. It takes one day to get your account approved, so try it a day before and when it is approved you can use it to create students accounts. Otherwise, let them create their own accounts. Make sure that you have installed Scratch beforehand on each computer.

8.2.6 Review of Club Organization

After a few club sessions, revisit the list and discuss if anything needs to be changed. The Next Sessions. Prepare the agenda (introduction, timing for the activities, show and tell, ending the session). The

session will take place once a week (a day will be determined), after school, for two hours. Preview the pedagogical guide and prepare materials to share with club members.

8.2.7 Session Structure

A session of a coding club is not the same as a regular lesson. A coding club is an after-school activity where learners come together to learn about coding, under the guidance of a teacher facilitator. Teacher and learners decide together what they want to do in the club.

An agenda for a session should look different from a lesson plan. A structure of a 1-hour coding session could look like in Figure 181.

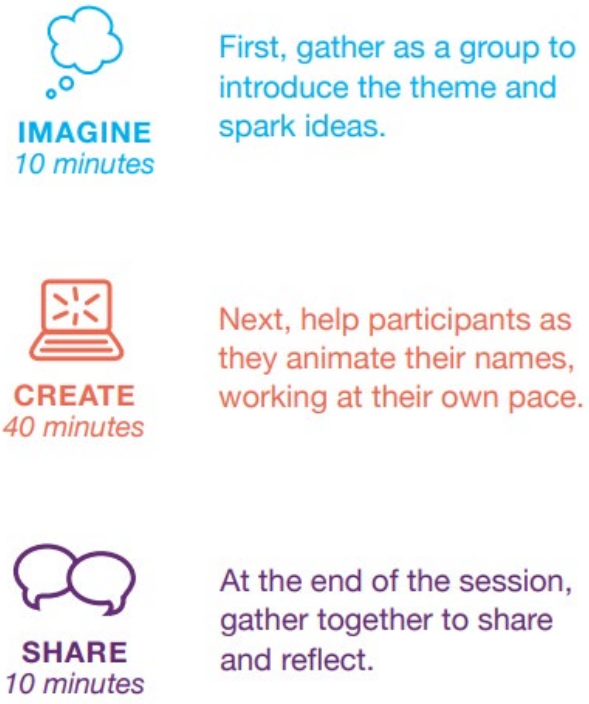


Figure 182: Structure of a 1-hour coding session

(https://resources.Scratch.mit.edu/www/guides/en/EducatorGuidesAll.pdf)

For example, an **activity** that helps learners to get familiar with Scratch is to let them **create an animated version of their name**.

1. **Imagine**:
    - Demonstrate the first few steps of the tutorial so participants can see how to get started.
    - Introduce the theme and let learners share some ideas
2. **Create**: Support participants as they create projects at their own pace:
    - Ask learners a few questions to get started;

- Provide resources;
- Suggest ideas to get started;
- Offer ideas for more things to try out
- Support collaboration:
    - When someone gets stuck, connect them to another participant who can help.
    - See a great idea? Ask the creator to share with others.
- Encourage experimentation:
    - Help participants feel comfortable trying different combinations of blocks and seeing what happens.
    - To understand their thought process, you can ask questions like: what are you working on now? Or what are you thinking of trying next?

3. **Share:** at the end of the session, gather your learners to share and reflect
    - Have participants share their project with their neighbours.
    - Share some questions they can discuss, such as:
        - What do you like the best about the project you made?
        - What did you learn?
        - What was the hardest part?
        - What would you like to improve still?

Help members to add their projects to a **shared studio in Scratch**. Give them a link to the studio. Then they can click 'Add Projects' at the bottom of the page. Ask for volunteers to show their project to the group.

- Award a certificate of participation after every term.
- Organise Intra-school and inter-school competitions.
- Take part in competitions at sector level, district level and online hackathons.

At the end, tell students what they will learn in the next session and remind them when and where the session will take place. You may assign some tasks they can take home (a project, research, discovery…)

8.2.8 Answering Questions

The main point of teaching is to help students discover answers on their own. This will help students to get more understanding of the concept and build their skills of researching and problem solving.

Here are different techniques you can use:

- Answering by giving examples. You give an example that would describe a scenario or a story that would help a student discover the answer to the question.
- Answering by asking question leading to answer. This is a technique that you can use when answer is something simple students can get easily.
- Answering by recommending further research and discovery. This is where you ask students to research and they share ideas later.

- Start a discussion in groups from where an answer can be found
- Point to online/local recourses

MODULE 9

GENDER AND INCLUSIVENESS IN STEM EDUCATION

INTRODUCTION

Activity: Discuss the following entry question in pairs:

- What do you do in your teaching to make sure that all learners can learn to the best of their abilities?

Rwanda has achieved gender parity in net and gross enrolment at pre-primary, primary, and secondary levels (USAID, 2018). In fact, girls' enrolment is higher than for boys at primary and secondary levels. However, dropout rates for both boys and girls, as well as disabled learners, remain a challenge. Boys younger than 13 are more likely to repeat and drop out than girls; and at age 14, the dropout rate for girls surpasses that of boys (MINEDUC, 2018).

In national examination results, boys outperformed girls in almost all districts at P6 and S3 levels during the period 2008-2014 (MINEDUC, 2018). This indicates that girls face more challenges inside and outside schools than boys. To eliminate all the obstacles which lead to disparity in education, the Ministry of Education included gender and inclusive education as crosscutting issues in the Competence Based Curriculum framework (Rwanda Education Board, 2015).

LEARNING OUTCOMES

By the end of this unit, you will be able to:

- Explain the meaning of gender, inclusive education and related concepts;
- Explain the difference between inclusive education and special needs education;
- Relate gender and inclusive education to classroom teaching and learning processes;
- Apply a gender responsive pedagogy in the classroom;
- Reflect on how to apply gender concepts to teaching and learning STEM;
- Design learning activities that will equally interest and engage girls and boys in STEM;

- Make learning of STEM enjoyable for both girls and boys;
- Address gender stereotypes in STEM instruction;
- Appreciate that boys and girls have equal abilities to achieve proficiency in STEM;
- Commit to working towards equity in their school.

9.1 WHAT IS GENDER?

Many people wrongly think that gender means "women's issues". In reality, gender refers to socially determined roles and relations between males and females. Gender is different from sex. Sex refers to purely biological differences between men and women. Gender roles, on the other hand, are created and sustained by the society, which assigns different responsibilities to men and women, e.g., cooking for women and decision-making for men.

Gender roles can therefore be changed and vary over time and from community to community. These gender roles are consciously or unconsciously carried into the classroom dynamics by both teachers and learners. In children's textbooks, for example, women are often represented as cleaners, caregivers and nurses, and men are drivers, doctors, and leaders. These images reinforce gender roles.

In a famous study, thousands of children were asked to draw a scientist (Chambers, 1983). Of the almost 5,000 drawings, just 28 depicted a female scientist, and all of those were drawn by girls. Not a single boy drew a woman. In follow-up studies, the number of learners drawing a woman as a scientist has increased, but is still far from 50% (Miller et al., 2015). Even girls, as they grow up, draw scientists as men. At age 6, girls draw 70 percent of scientists as women, but this proportion flips around ages 10 to 11 and by 16, they draw around 75 percent of scientists as men (Yong, 2018). **Lower secondary school is a critical period** in which they're learning this gender-biased information about what is a scientist (Miller et al., 2015) .

Figure 183: Example of a female scientist, drawn by learners (Yong, 2018)

Activity

Can you give examples of how cultural norms and practices have an impact on girls' and boys´ participation in STEM lessons in your school? Has there been any evolution over the years?

Areas where consistent gender differences have emerged are children's beliefs about their abilities in STEM, beliefs and attitudes of other stakeholders (school leaders, teachers, parents, community leaders) in abilities of boys and girls, learning outcomes in STEM, children's interest in STEM and their perceptions of the importance of STEM for their future.

Researchers have found that girls often have **less confidence in their abilities for STEM and ICT subjects** than males do and that girls show less interest in STEM or ICT careers. Girls tend to **underestimate their abilities in STEM subjects**. This is a problem because research shows that children's beliefs about their abilities are crucial to determining their interest and performance in different subjects and the career choices they make (Beilock et al., 2010).

This gender difference contrasts with research that males and females generally show similar abilities in STEM, as measured by test scores. There is no such thing as a "female" or "male" brain.

9.2 KEY TERMS

When discussing gender, we use various terms. In this section, we clarify their meaning.

- **Gender discrimination**: Denying opportunities and rights or giving preferential treatment to individuals on the basis of their sex.

- **Gender equality**: The elimination of all forms of discrimination based on gender so that girls and women, boys and men have equal opportunities and benefits (OECD, 2015).

- **Gender equity**: Fairness in the way boys and girls, women and men are treated. In the provision of education, it refers to ensuring that girls and boys have equal access to enrolment and other educational opportunities (Subrahmanian, 2005).

- **Gender stereotype**: The constant presentation, such as in the media, conversations, jokes or books, of women and men occupying social roles according to a traditional gender role or division of labour (OECD, 2015). However, avoiding gender stereotyping does mean the denial or minimisation of differences between males and females.

- **Gender sensitive**: The ability to perceive gender issues. It is the beginning of gender awareness (UNICEF, 2017). The opposite of gender sensitivity is gender blindness. This is an attitude to ignore gender issues, claiming that they don't exist. For example, a gender-blind teacher may see no problem with boys taking all leadership roles in the class.

- **Gender-based violence** refers to acts of violence inflicted on women because of their gender and sexuality. It includes physical violence in the form of corporal punishment, psychological violence such as verbal abuse, and sexual violence ranging from unwanted sexual talk and indecent touch to rape.

The figure below shows the difference between equality and equity. Equality means treating all learners in the same way. Equity is about giving all learners the support they need to achieve the learning outcomes. This means that some learners will need more or different support than others (see Figure 183). In the third image, all three children can see the game without any supports or accommodations because the systemic barriers have been removed. The cause of the inequity was removed.

Figure 184: Equality versus Equity (Save the Children, Mureke Dusome project, 2017)

9.3 GENDER RESPONSIVE PEDAGOGY

Gender responsive pedagogy refers to **teaching and learning processes that pay attention to the specific learning needs of girls and boys** (Mlama, 2005). Gender responsive pedagogy calls for teachers to take an integrated gender approach in the processes of lesson planning, teaching, classroom management and evaluation.

Gender responsive pedagogy includes **gender neutral language use by the teacher**. Inappropriate language use can transmit negative messages and inhibit learning. A boy or girl whose teacher constantly tells them "you are stupid", will come to believe this to be true. Language can also reinforce gender differences and inequalities and in the classroom often reflects male dominance and reduces females to an inferior position. By contrast, a teacher can enhance students' performance by using encouraging, inclusive language in the classroom.

9.4 MAKING STEM AND ICT LESSONS GENDER RESPONSIVE

9.4.1 Introduction

In this section, we discuss strategies that teachers can use to promote the involvement and learning of girls in STEM and ICT lessons and sessions of your coding club.

What can you do to encourage girls in learning STEM? Often, boys tend to dominate learning processes to maintain their superiority in the presence of girls. Therefore, you need to consider the specific gender needs of girls and boys in planning your lessons and club sessions.

Think about how all students can participate in learning activities. Ensure that there is equal participation in activities such as presenting projects, conversations and practical activities. In group activities, ensure that both girls and boys are given leadership positions and roles. Make sure that learning materials and computers are distributed equitably to both girls and boys, especially in cases of shortages.

9.4.2 Making STEM lessons gender equitable:

1) Classroom arrangement

Consider the typical classroom arrangement – desks arranged in rows facing the teacher. Often, such an arrangement reinforces traditional gender patterns. Since girls are not brought up to speak out, when they sit at the back of the class, they are less likely to participate unless the teacher makes a special effort to involve them. Remember the distinction between equality and equity. Being gender responsive does not means treating all learners equally but making sure that all learners have equal opportunities to learn. A different arrangement such as breaking the class into smaller groups may encourage girls to participate more.

2) Teach students that all students have abilities to learn

To change girls' beliefs about their abilities, teachers should understand and communicate to students that abilities in STEM, ICT and coding —like all abilities—can be improved through consistent effort and learning (Dweck, 2006, 2015). To help girls resist negative reactions to the difficulty of STEM and ICT, it

is important to stress for them to learn that their abilities can improve with continuous effort and engagement.

3) Expose girls to female role models

Researchers have found that negative stereotypes can affect performance and have called this phenomenon “stereotype threat.”

Studies show that stereotype threat can lead young adolescent girls and women to choose unchallenging problems to solve, lower their performance expectations and devalue mathematics as a career choice. In addition, we suggest that teachers invite women or elder students who can serve as role models in STEM to be guest speakers or tutors. These role models should communicate that becoming good at mathematics and science takes hard work and that self-doubts are a normal part of the process of becoming expert in any field.

Examples of **role models for STEM in Africa**:

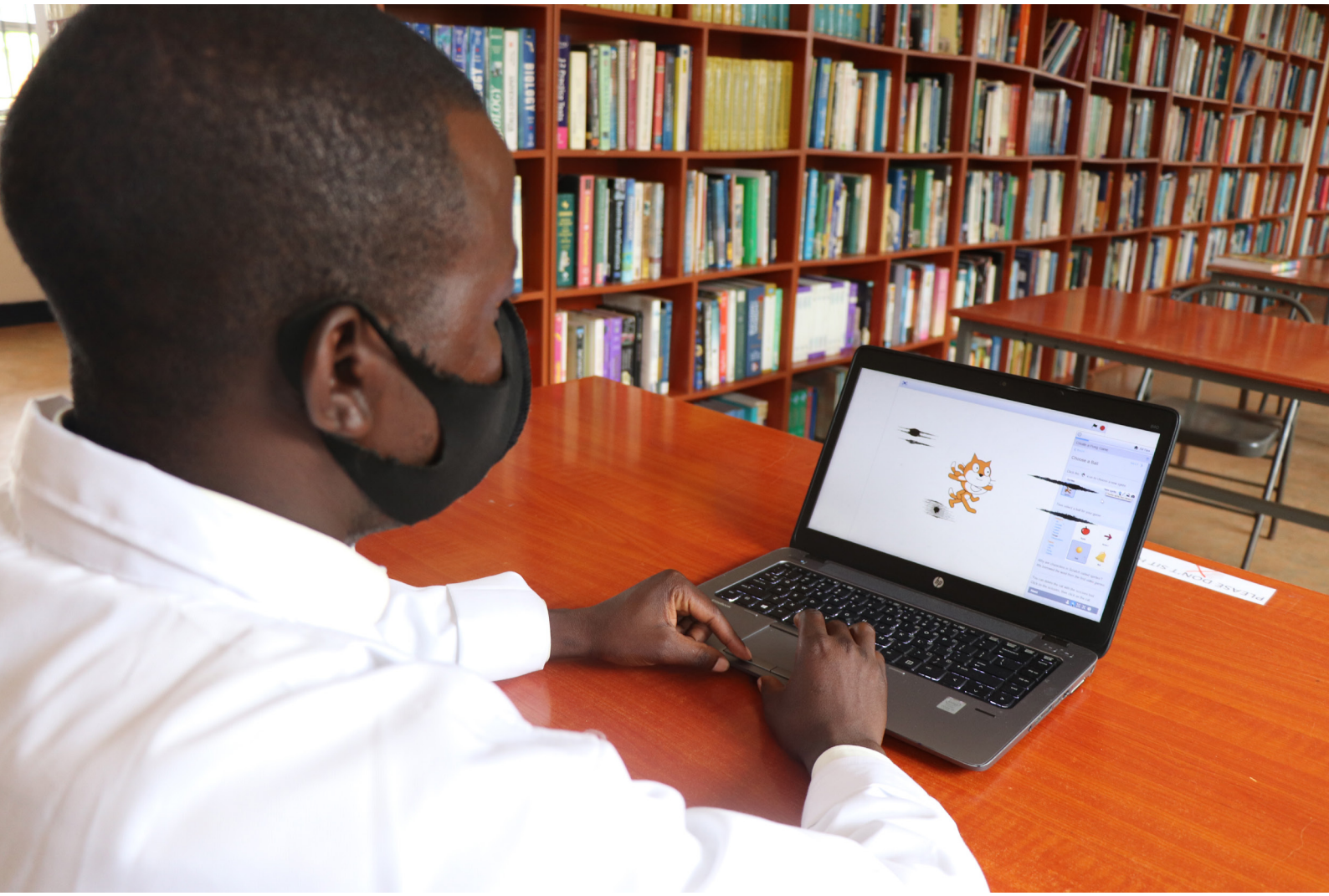
Apps and Girls

Apps & Girls is a Tanzanian registered social enterprise that was founded in July 2013 by Carolyne Ekyarisiima. It seeks to bridge the tech gender gap by providing quality coding training (web programming, mobile app development game development and robotics) and entrepreneurship skills to girls in secondary schools via coding clubs and other initiatives such as mentorships and scholarships. So far, they have created 25 coding clubs in Tanzania and they have trained 269 teachers and 2656 girls. They want to train 1 million girls before 2025.

Link website: http://www.appsandgirls.com/

Link YouTube: https://www.youtube.com/watch?v=yNNrVqUvkjg

Activity: Discuss the techniques above to make STEM lessons more equitable. Which ones do you already apply in your class? How? Which ones have you not yet applied? Why not?



Belgium
partner in development

Flanders
State of the Art

CC BY NC SA