

Using MakeCode Arcade



BRIDGE COURSE

TOPICS COVERED

95%

- What is a Traffic Light System?
- What is Coding?
- Searching for a Word in Dictionary
- MakeCode Arcade
- Types of Blocks Category
- Changing the Background
- Getting started with Block Coding
- Creating a Conversation Between Two Sprites in MakeCode Arcade
- What is a Bug?
- What are Variables?
- Using Logic Blocks
- Where Else do we See Applications of Coding?
- What is a Programming Language?
- Pseudocode
- Components of MakeCode Arcade Window
- Adding a Sprite
- Commonly used Blocks in MakeCode Arcade
- What is an Event?
- Using Math Blocks

In your day to day life, you must have come across the term 'coding'. Have you wondered what it means. In this chapter, you are going to learn what is coding and its usage in day to day life.

What is a Traffic Light System?

While going to school, you must have seen a traffic light.

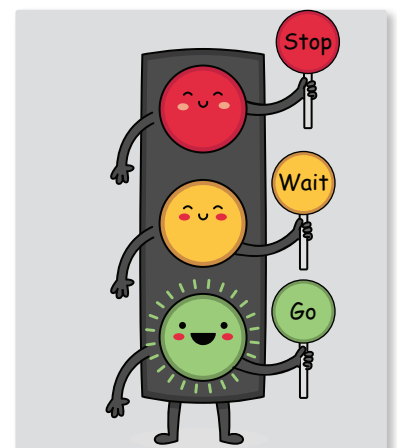
Traffic lights are devices that control traffic through the use of colours. These lights are placed at road intersections and pedestrian crossings for smooth traffic flow.

How do Traffic Lights Work?

In a traffic light, the lights cycle through green (allows traffic to proceed in the direction denoted), yellow (warns that the signal is about to change to red), and red (stops traffic from proceeding) at regular intervals of time. Traffic lights help to prevent accidents and congestion on the roads.

How do the Traffic Lights Change Automatically?

Traffic lights contain sensors which are programmed to calculate how many cars or pedestrians are present at a specific point.



At regular intervals of time, the code automatically changes the traffic signals to show colors (red, yellow, green).

Do You Know?

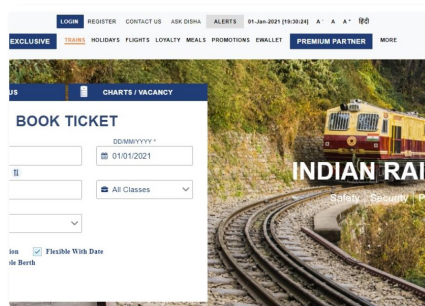
John Peake Knight invented the first traffic light, after that **William Potts** invented the modern three-lens traffic light and **Garrett Morgan** invented the first automatic traffic light.

Where Else do We See Applications of Coding?

These days, a lot of our daily lives rely on coding. We are somehow connected through coding. For example, smartphones, computers, videogames, car dashboards, etc., are all using some sort of code to perform their tasks.



Bar code scanner



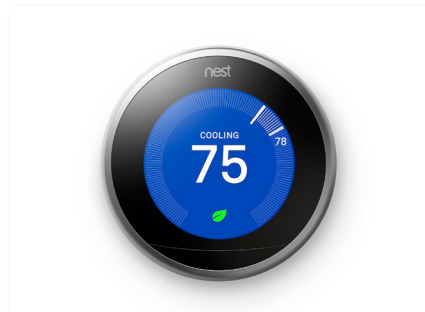
Booking tickets



Printer



Software



Thermostat

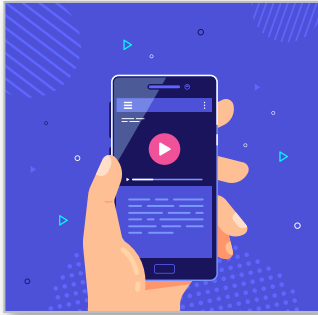


Video games

What is Coding?

Coding or programming is the process of creating codes to instruct a computer to perform a specific task. There can be more than one way to solve the problem; similarly, there are many ways to write code for the same task.





Let see an example:

When you play video on your smartphone. Your smartphone will act as a computer which needs to be instructed (in this instructions will be given by the application which plays the video) at each and every step on what to be done. The application playing the video provides this instruction via programming language.

Do You Know?

The first programmer or the first person to write our modern understanding of a program was Ada Lovelace.



What is a Programming Language? 🚀

Language is the primary means of communication for all human interactions. In the same way, to interact with computers, you need a language which the computer understands and is called a **programming language**.

To communicate with computer, programmers use computer language which is known as programming language. So, programming language can be defined as a set of instructions which are written in any specific computer language to perform a set of activities.

Programming languages have syntax. **Syntax** is the same as grammar in any language.

The **syntax** is a set of rules that we need to follow when we write a computer program. Every programming language has its own syntax. But programming languages will eventually be converted into a language which computer will understand.

Examples of programming languages are C, C++, C#, Python, PHP, Java, JavaScript, R, etc.



Do You Know?

Kautilya Katariya from the UK is a Guinness World Record holder to take up programming at the age of 6.





Give one word answer to the following questions:

1. Name the set of rules to be followed while writing a computer program. _____
2. Name the instructions which are given to a computer to perform a set of activities. _____

Searching for a Word in Dictionary

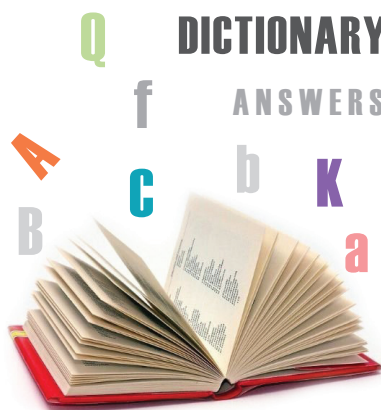
A dictionary is a book which contains words and phrases in an alphabetical order, with their meanings. Many times, while reading a book you may come across a word whose meaning you don't know. So how do you find out the meaning of the word?

You will find the meaning of the word by using a **dictionary**. But when you open a dictionary you see, there are many words in a dictionary. So how do you find that particular word in the dictionary?

Let's see an example of how to find the word 'Orange' in a dictionary:

1. Find the dictionary section with the first letter of the word, 'o'.
2. Within the list of words starting with the first letter 'o', find the section having the second letter of the word 'r'.
3. You need to do this process again with the third letter 'a', fourth letter 'n', fifth letter 'g' and sixth letter 'e', until you finally reach the word 'orange' in the dictionary and then find its meaning.

So, to find the meaning of a particular word in a dictionary, you need to follow a set of steps. Similarly, before writing a program for a given problem, it is important to define a set of steps which need to be followed to solve the problem successfully. This sequence of the set of steps is called an algorithm.



Do You Know?

The first computer game 'Spacewar' was created in 1961, by MIT programmer Steve Russell and his team.

Pseudocode

Pseudocode is used to describe the steps of an algorithm in a human-understandable language. It has no syntax and can be easily understood by a layman. So, pseudocode is an informal way of writing programs in which there is no need to think about semi-colons, and curly brackets.

Characteristics of Pseudocode

- It uses structured English statements.
- It can be reviewed/verified easily to see if it generates the desired outcome.
- You can focus on all possible scenarios. So, this helps you to understand the potential problems that might come up later.
- Writing pseudocode will help to write your code much easier.

Advantages of Pseudocode

- It is language-independent and can be used by most programmers to express the design in plain and natural language.
- Programmers do not have to think about syntax, they simply have to concentrate on logic.
- The focus is on the steps to solving a problem rather than how to use the computer language.

Example 1: Write pseudocode to calculate profit and loss.

Solution:

```
Read CostPrice
Read SellingPrice
IF (SellingPrice > CostPrice ) THEN
    Profit = SellingPrice - CostPrice
    PRINT Profit
ELSE
    Loss= CostPrice - SellingPrice
    PRINT Loss
```

Example 2: Write pseudocode to print 'Above average marks' if the average marks in the three subjects are greater than 60 and 'Below average marks' if the average marks are less than or equal to 60.

Solution:

```
Read Eng_Marks
Read Math_Marks
Read Sci_Marks
Sum= Eng_Marks+Math_Marks+Sci_Marks
Avg_Marks = Sum / 3
IF (Avg_Marks > 60) THEN
    PRINT "Above Average marks"
ELSE
    PRINT "Below Average marks"
```

Do You Know?

American computer scientist 'Margaret Hamilton,' wrote the computer code which helped to save the Apollo moon landing mission.



QUEST



Information Literacy
Technology Literacy

Answer the following questions:

1. Which book contains words and phrases in an alphabetical order, with their meanings? _____
2. What is used to describe the steps of an algorithm in a human-understandable language? _____

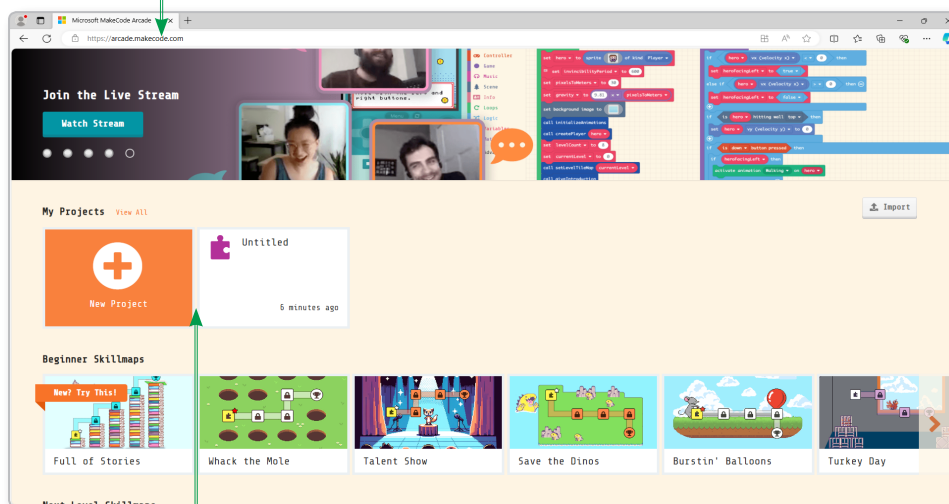
MakeCode Arcade

MakeCode Arcade is a free, open-source, online integrated development environment (IDE) for game production. Drag-and-drop block programming is used in this user-friendly coding editor for beginners.

Starting MakeCode Arcade

To start MakeCode Arcade, follow the steps given below:

- 1 Open web browser and type <https://arcade.makecode.com/> in the address bar and press the **enter** key.



- 2 Click on **New Project** button.

3 Type the name for your project.

Create a Project 🥳🥳🥳

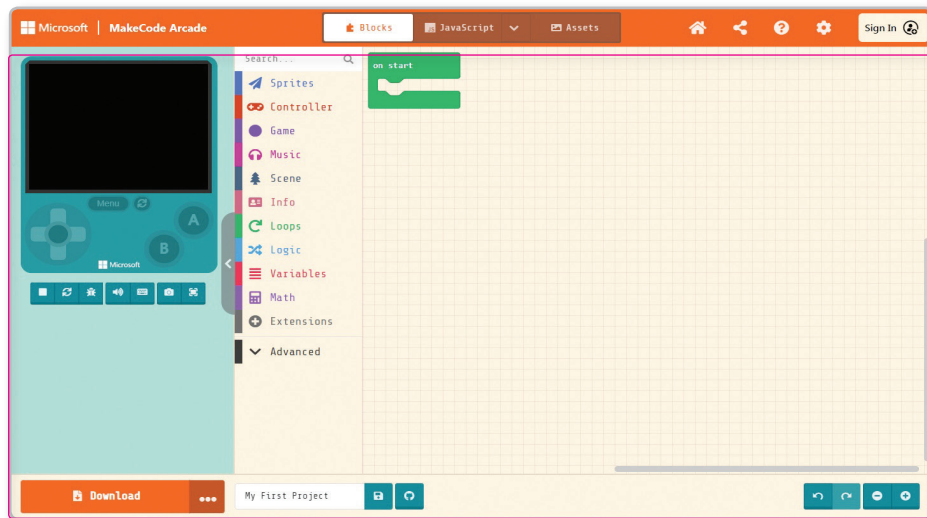
Give your project a name.

> Code options

Create ✓

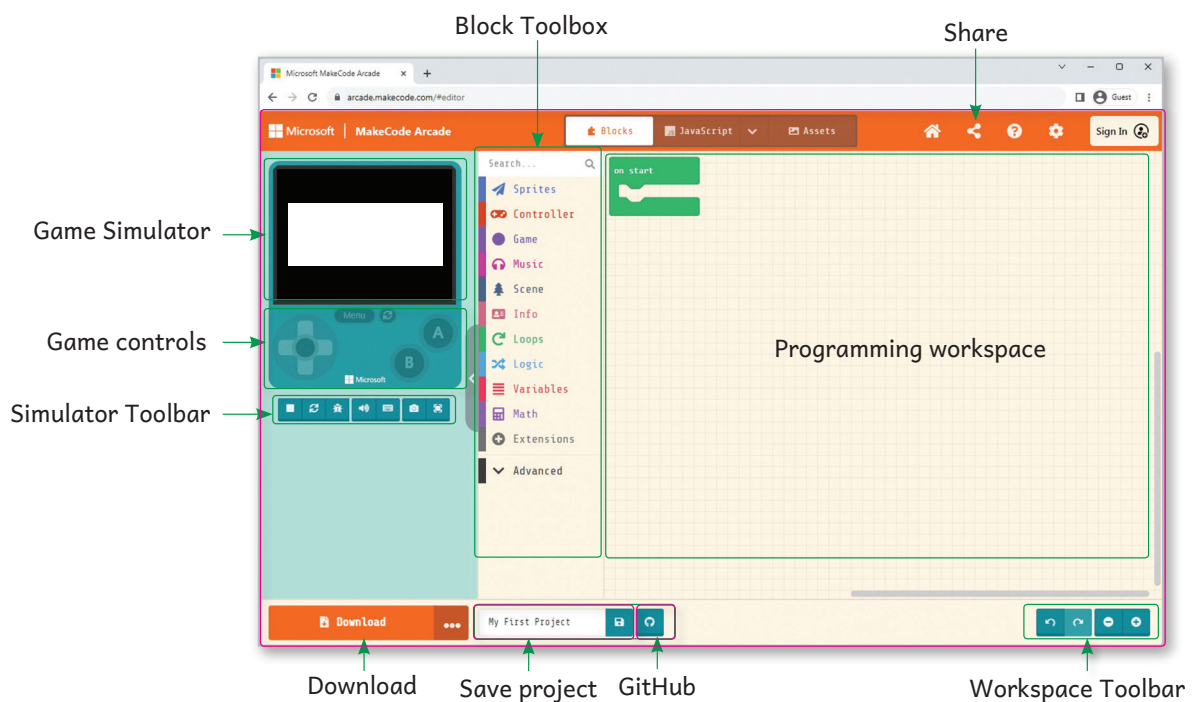
4 Click on the **Create** button.

MakeCode Arcade editor window will appear on the screen.



Components of MakeCode Arcade Window

Let us now learn about the components of MakeCode Arcade.

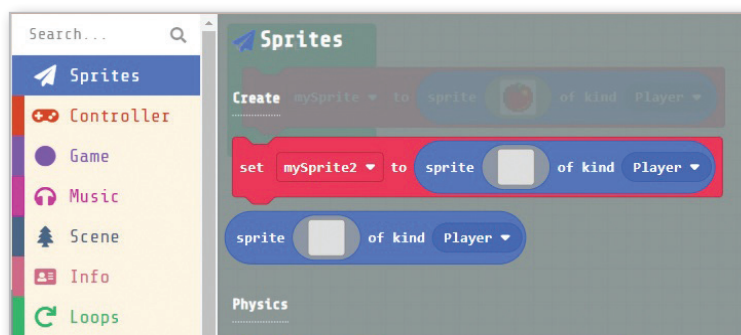


- **Block Toolbox:** The Toolbox is your source for all the necessary code elements to create your game. These elements are neatly categorised in drawers based on their functions and methods.
- **Programming workspace:** The workspace is your creative canvas, where you construct your game. You can grab blocks from the toolbox and place them where you need them. Blocks can be copied, moved, or customised once they're in the workspace.
- **Game Simulator:** Experience your game in action within a fully operational game board simulator. This feature lets you run, test, and debug your game's code.
- **Game Controls:** Take control of your game by moving your player character or activating various in-game functions through the use of buttons. You can also access the menu and restart your game from here.
- **Simulator Toolbar:** The Simulator Toolbar equips you with essential controls for your game simulation. You can run the game, pause it, restart it, enable debugging, and manage the game's display settings.
- **Save Project:** Secure your game project by saving it to a file with a name of your choice. This ensures that your progress is preserved.
- **Workspace Toolbar:** The workspace toolbar provides tools to undo or redo changes made to your code. It also allows you to zoom in or out to adjust your view of the code within the workspace.
- **Share:** Collaborate and showcase your game by sharing it in the cloud. You'll receive a shareable link to distribute to others.
- **Download:** When your game is ready for deployment, use this option to download your code to the arcade hardware for it to be played on a physical device.
- **GitHub:** Make a gaming project repository on GitHub.

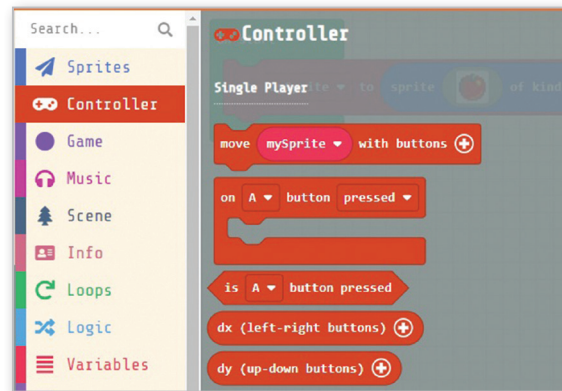
Types of Blocks Category

Beginners can program using block-based programming, which allows users to drag and drop code pieces into an editor. To program our games in MakeCode Arcade, we can use a variety of code blocks category, which are listed below:

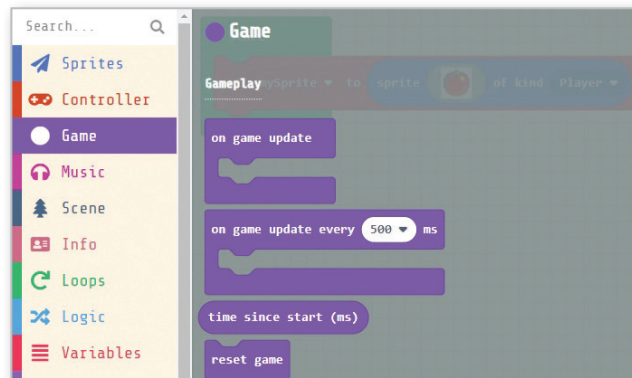
- **Sprites Blocks:** These blocks allow you to create and define sprites, which represent objects within your game. Sprites can be anything from players and enemies to food items and projectiles.



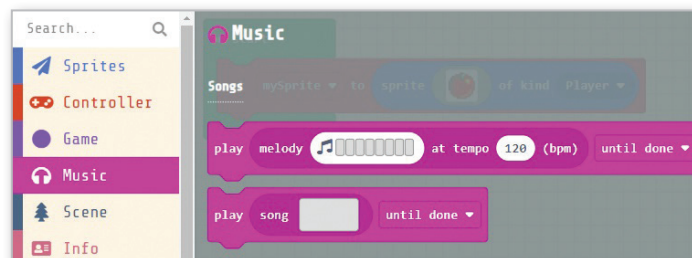
- **Controller Blocks:** These blocks allow you to specify which buttons on the controller trigger specific actions. For example, you can assign button A to shoot arrows in your game.



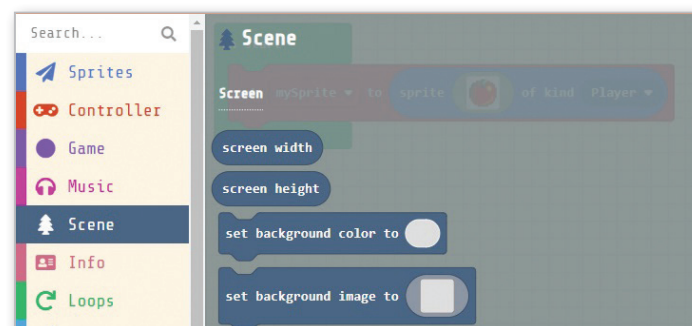
- **Game Blocks:** These blocks allow us to control the game's timeline and define win-or-lose conditions. They govern the overall flow and rules of your game.



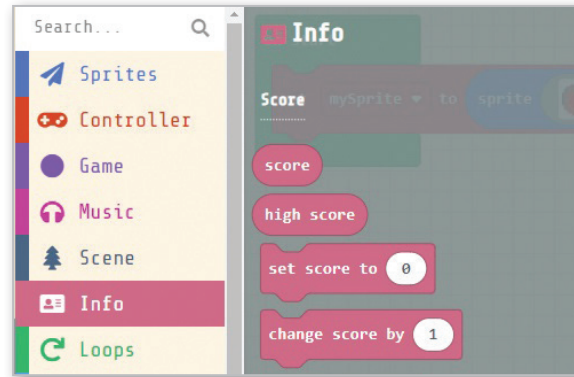
- **Music Blocks:** These blocks allow you to enhance your game's experience by incorporating music and sound effects.



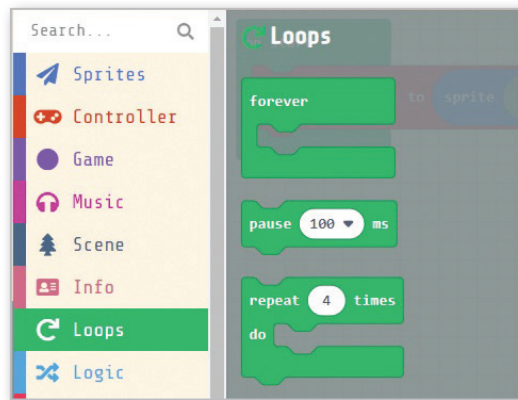
- **Scene Blocks:** These blocks help you to manage the game's background and screen dimensions, allowing you to customise the visual aspects of your game world.



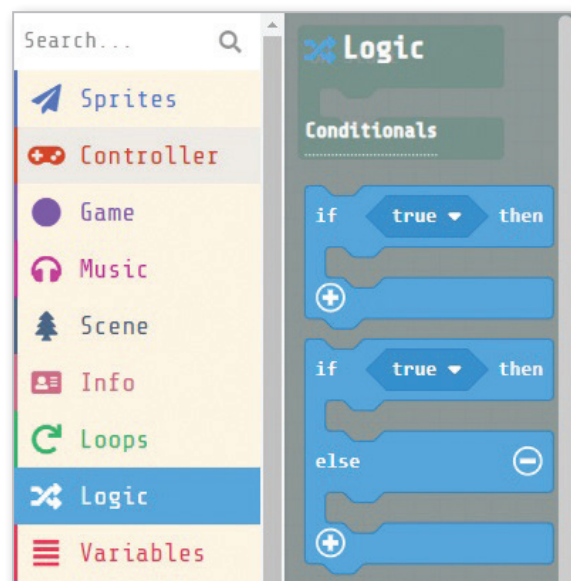
- **Info Blocks:** These blocks provide control over essential game elements such as score, lives, and the game clock.



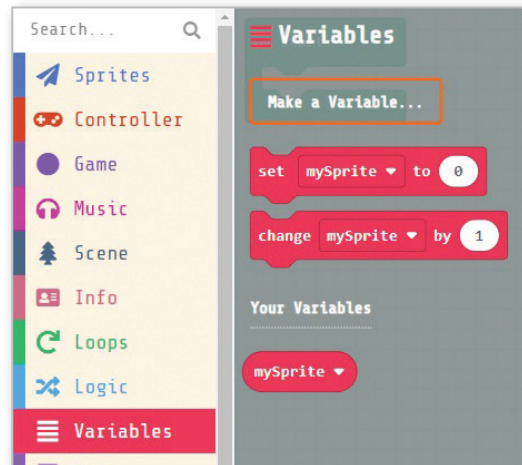
- **Loops Blocks:** These blocks allow you to repeatedly execute specific code blocks under different conditions, enabling you to create repetitive game logic.



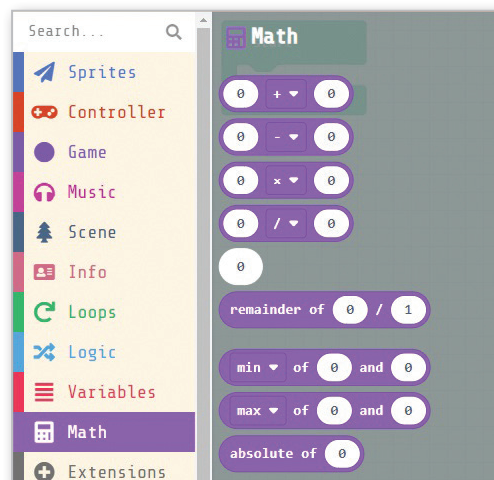
- **Logic Blocks:** These blocks are used to implement conditional logic to control when specific code blocks execute based on certain conditions or criteria.



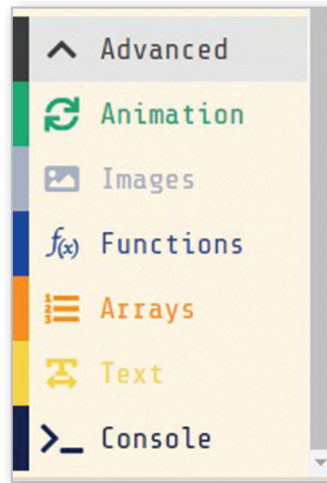
- **Variable Blocks:** These blocks allow you to create variables to store data, such as sprite velocity, allowing you to manage and manipulate in-game values.



- **Math Blocks:** These blocks allow us to perform various mathematical operations that can be used to modify variables and perform calculations within your game.



- **Advanced Blocks:** This block comprises a range of advanced functionalities, including:
- **Animation Blocks:** Create and edit sprite animations.
- **Images Blocks:** Design and edit graphics for both sprites and backgrounds.
- **Function Blocks:** Define custom functions that can be called within your programme.
- **Array Blocks:** Create lists of data for managing and storing information.
- **Text Blocks:** Generate strings, which can be used for creating character dialogue or speech bubbles.
- **Console Blocks:** Aid in debugging and troubleshooting errors in your program.
- **Extension Blocks:** Access additional block types, such as animations and other specialised functionalities, to expand your programming capabilities.



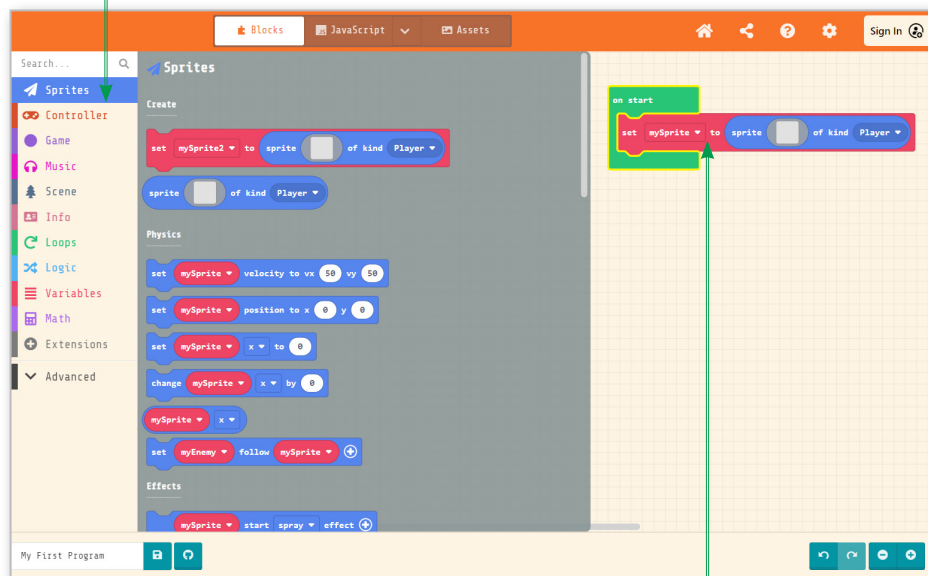
Adding A Sprite

When you open MakeCode Arcade, there is no default sprite present. So, firstly, we need to add a sprite by choosing from two options, i.e., creating a new sprite using the image editor or using a built-in sprite from Gallery. Let us learn about both options.

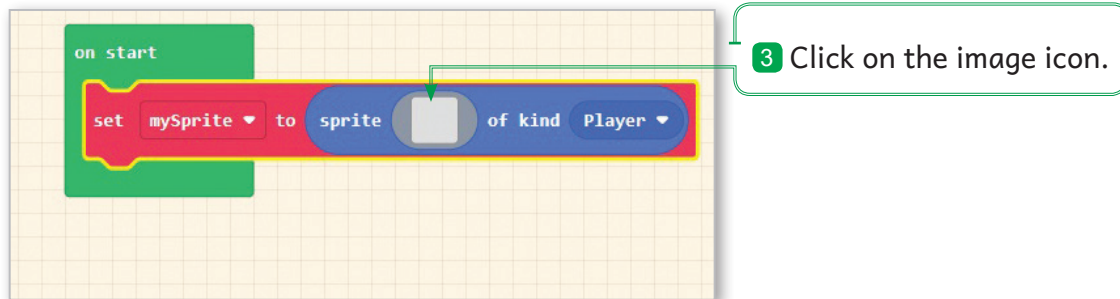
Creating a sprite using the image editor

To create a sprite using the image editor, follow the given steps:

1 Click on the **Sprites** blocks category.



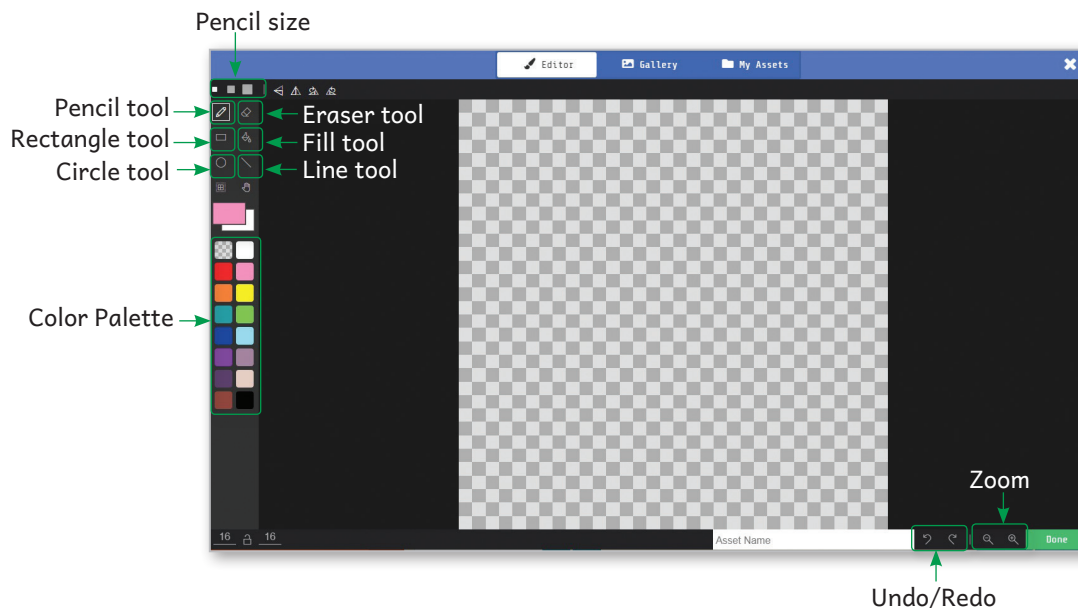
2 Drag the **set to sprite of kind** block and drop it inside the **on start** block.



Notes

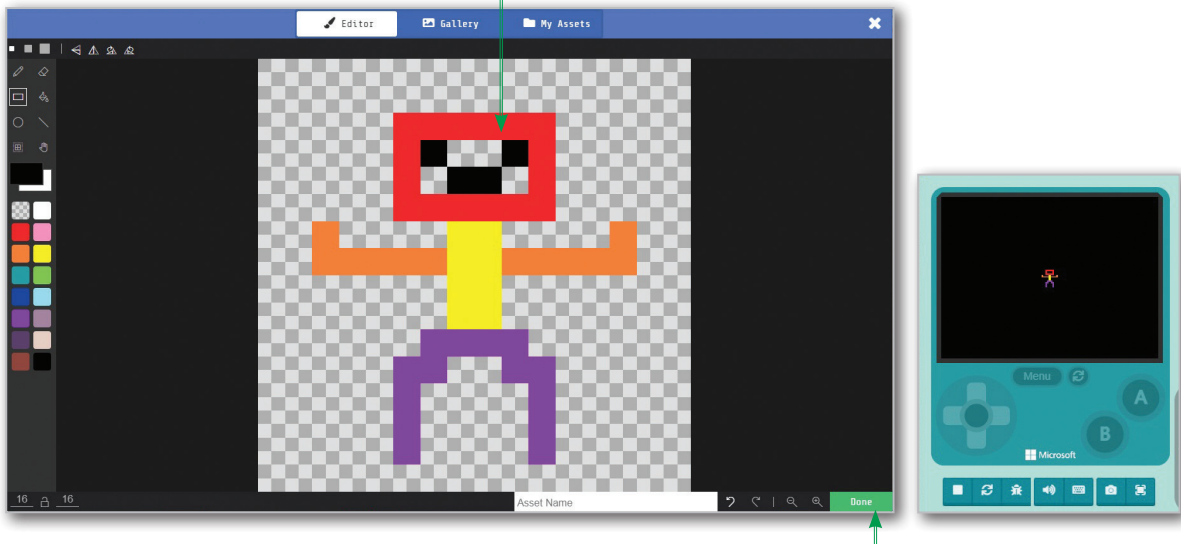
Clicking on image icon allow you to draw an image for your own sprite in image editor.

The image editor window will open. Let us understand some basic components of image editor window.



- **Pencil size:** Adjusts the thickness of the pencil tool for drawing.
- **Pencil tool:** Used for freehand drawing with adjustable stroke size.
- **Rectangle tool:** Creates rectangular shapes with specified dimensions.
- **Circle tool:** Draws circular shapes with adjustable radius.
- **Eraser tool:** Removes or erases parts of the image or drawing.
- **Fill tool:** Fills a selected area or shape with a solid color or pattern.
- **Line tool:** Draws straight lines between two points.
- **Color Palette:** Provides a range of colors to choose from for drawing and filling.
- **Undo/Redo:** Reverses or re-applies recent changes made to the image.
- **Zoom:** Adjusts the view size of the image for detailed editing or a broader view.

4 Create your own sprite using various tools.

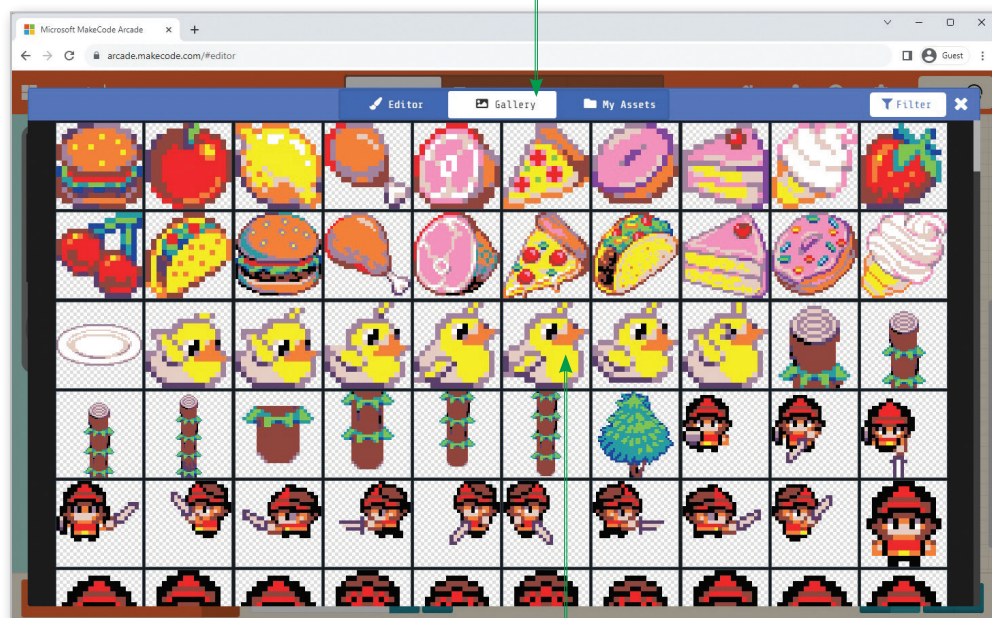


5 Click on the **Done** button. You will see the created sprite in **Game Simulator** window.

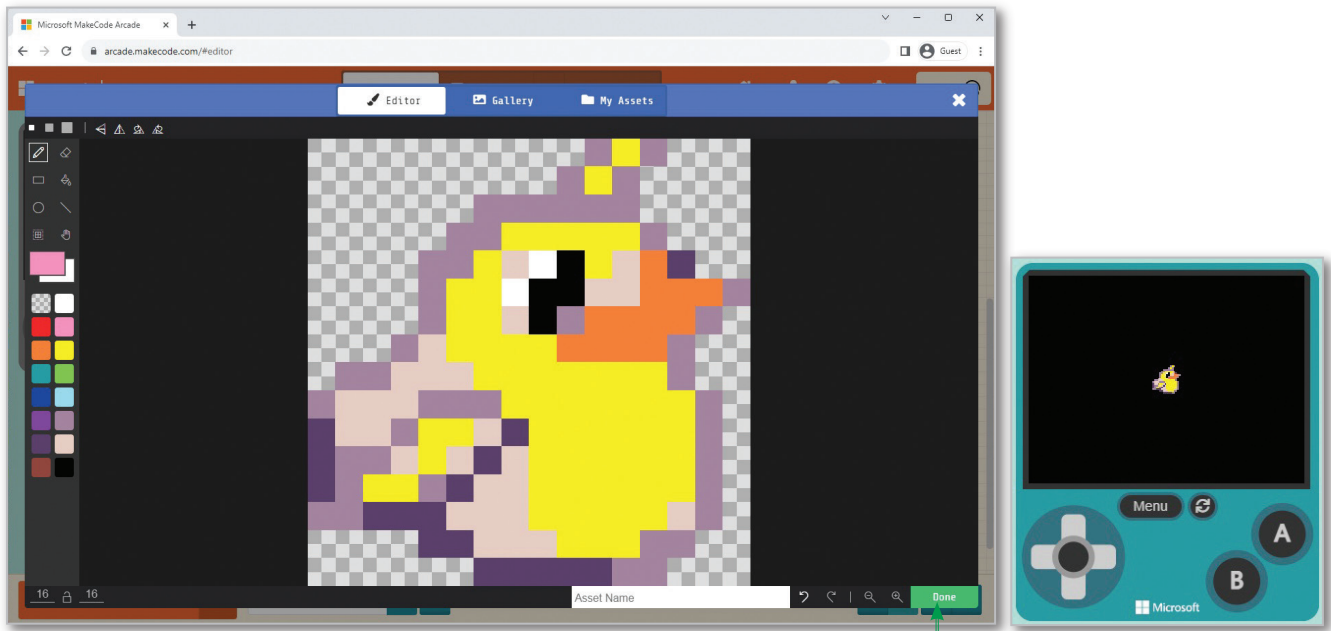
Using built-in sprites from the gallery

To use built-in sprites from the gallery, follow the given steps:

1 Select **Gallery** after clicking on the image icon.



2 Select desired sprite to add it to your workspace.

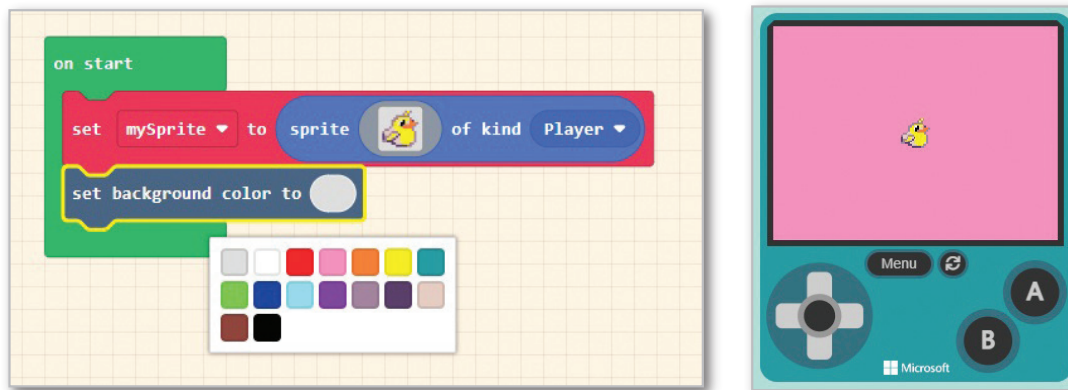


3 Click on the **Done** button. You will see the selected sprite in Game Simulator window.

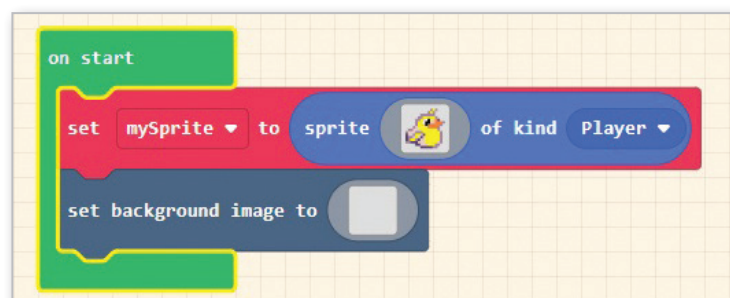
Changing the Background 🚀

To change the background, we have two blocks in Scene blocks in MakeCode Arcade which are as follows:

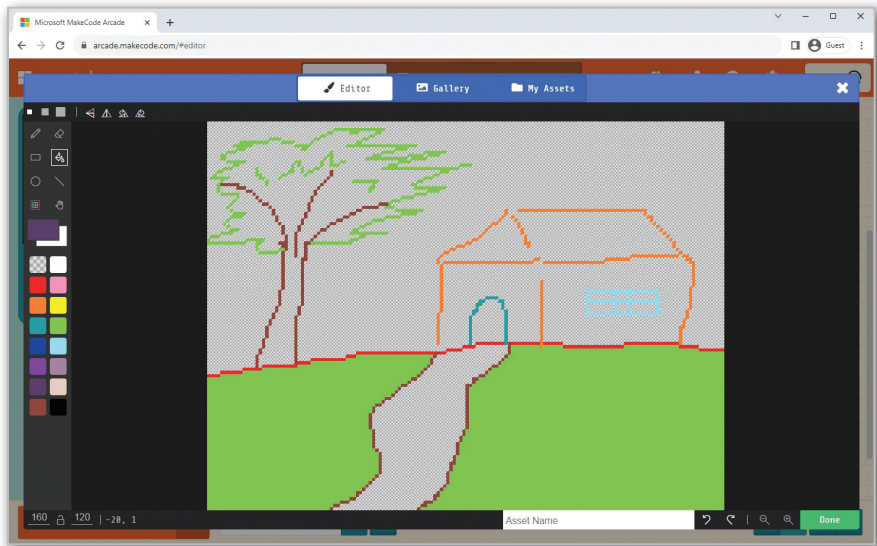
1. **set background color to:** This block allows you to specify and change the background color of the screen in your game with 16 predefined colors.



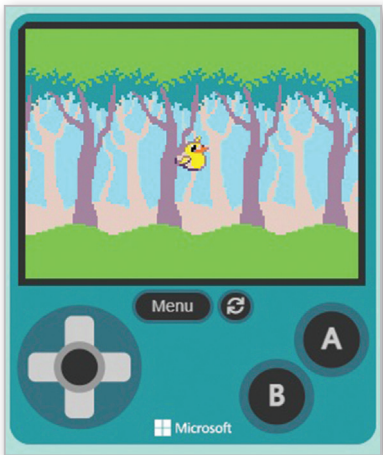
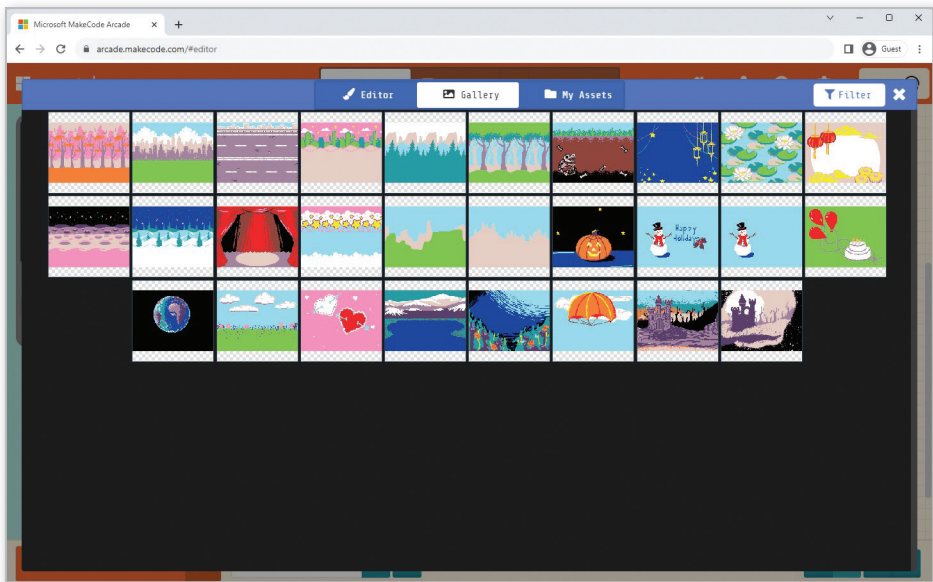
2. **set background image to:** This block allows you to select a built-in background or draw an image as the background in the Image editor.



The process to create a background using the image editor is same as creating a sprite.

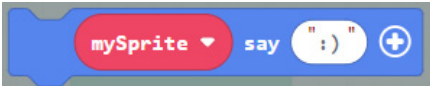
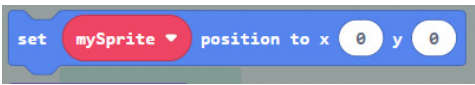


Or you can choose the readymade background from the gallery.



Commonly used Blocks in MakeCode Arcade

Block	Block Name	Description
	splash	The splash block in MakeCode displays a message on the screen for a specified duration
	pause	The pause block in MakeCode temporarily stops program execution for a specified duration in milliseconds, allowing for timed delays between actions.

	say	The say block in MakeCode shows a message on the screen for a brief moment, typically used for quick text displays.
	set position	The set position block in MakeCode places a sprite or element at specific X and Y coordinates on the screen.

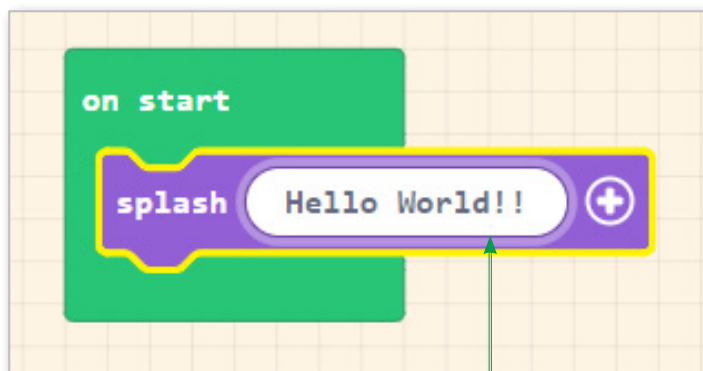
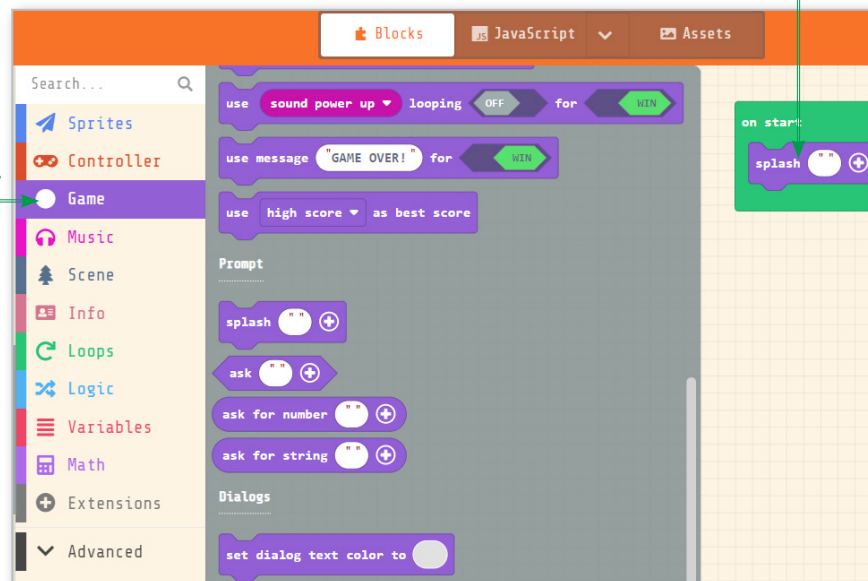
Getting started with Block Coding

Block-based coding uses drag and drop approach. It's a programming activity to develop computational thinking. You can use coding instruction 'blocks' to perform a variety of tasks.

To display a message 'Hello World', create a New project in MakeCode Arcade and follow the given steps:

2 Drag the **splash** block and drop it inside the **on start** block.

1 Click on the **Game** blocks category



3 Type **Hello Worlds!!** in the **splash** block. The output will display in the Game **Simulator** window.



Well done, you have created your first block code program!

Creating a Conversation Between Two Sprites in MakeCode Arcade

To create a simple interaction between two sprites in MakeCode Arcade, Follow the given steps:

1 Click on the **Sprites** blocks category.



2 Drag the **set to sprite of kind** block from **Sprites** block category and drop it inside the **on start** block.

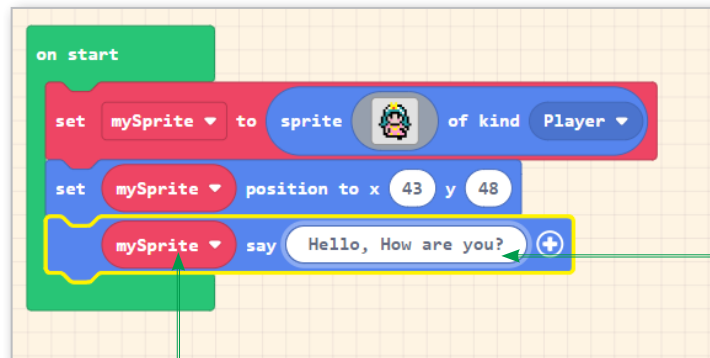


3 Click on the **image icon** and select the sprite you want to add from **Gallery**.

5 Change the position of **mySprite** x axis from 0 to 43 and y axis from 0 to 48.



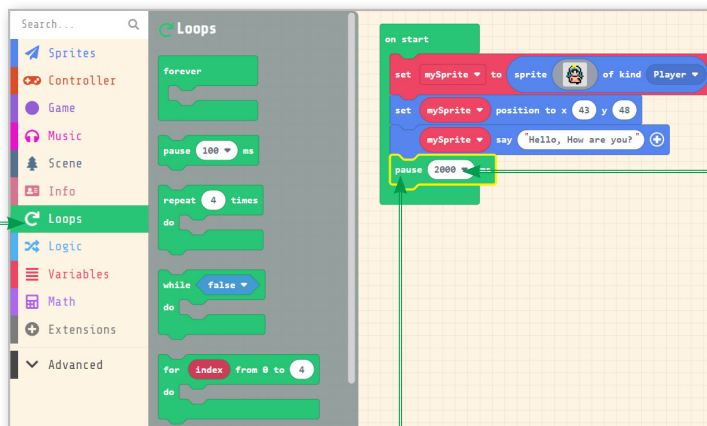
4 Drag the **set position** block from **Sprites** block category and drop it below the **set to sprite of kind** block.



7 Type the **Hello, How are you?** message in the **say** block.

6 Drag the **say** block from **Sprites** block category and drop it below the **set position** block.

8 Click on the **Loops** blocks category.



10 Set the duration to 2000 milliseconds (2 second).

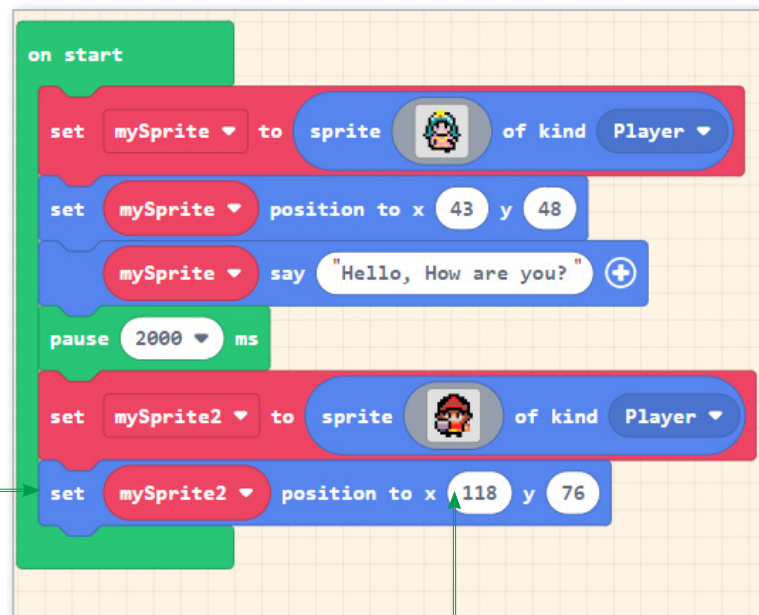
6 Drag the **pause** block and drop it below the **say** block.

11 Drag the **set to sprite of kind** block from **Sprites** block category and drop it below the **pause** block.



12 Click on the **image icon** and select the another sprite you want to add from **Gallery**.

13 Drag the **set position** block from **Sprites** block category and drop it below the **set to sprite of kind** block.



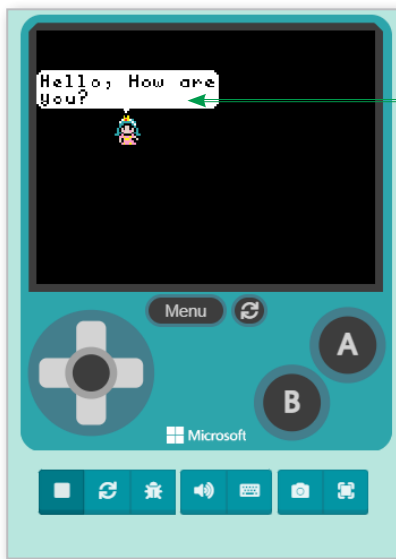
14 Change the position of **mySprite2** x axis from 0 to 118 and y axis from 0 to 76.

15 Drag the **say** block from **Sprites** block category and drop it below the **set position** block.



16 Type the **Hey, I am Fine!** message in the **say** block.

18 After 2 seconds the output of **mySprite2** will display in the **Game Simulator** window.



17 The output of **mySprite** will display in the **Game Simulator** window.



What is a Bug? 🐛

A bug is an unexpected problem in your program. You follow a defined sequence to write a program, from which you expect to return a specific output. Any change in the expected and actual output of the program is said to be the result of a bug.

So, we can say that bug is a general term which is used to describe any unexpected problem with hardware or software.

Example: Suppose you are going out for a family picnic. You are ready to go, but you notice that the tyre of the car is puncture, due to which your picnic will get delayed for few hours or you need to postpone it for some other day. As first you need to repair the tyre. So, this puncture in the tyre of the car can be termed as a ‘bug’ in programming.

What is an Event? 🚀

Event is an action which has happened. You can consider event as a generalization of things on which the program responds.

In programming, an event is an action which occurs as a result of the user or another source, such as a mouse click.

Examples of the events are:

- Clicking and loading the web page on web browser
- Creation of file in a file system
- Webcam or microphone receiving sensory input
- Incoming network traffic
- Typing on keyboard
- Timer
- In mobile apps, the events that happen are the results of the user doing something (on a click of button the orientation of the phone screen changes)

What are Event Handlers?

Event handler is a block of code which get executed when the event occurs and it is associated with the event.

Examples of Event Handlers:

- In Makecode Arcade, you display message by using ‘splash’ block . ‘splash’ block will get execute when you click on ‘start the simulator’ button. In this ‘start the simulator’ button is an event handler.
- When you click “send” for a text message, it sends the message and makes a sound.
- When you purchase an app from an app store, the phone asks for a password.
- When you click an icon for an app, the app opens. In the previous class, you learned about the introduction to coding and MakeCode Arcade. In this class, you are going to learn more about MakeCode Arcade.

What are Variables? 🚀

You have seen your mother naming boxes in the kitchen. That box has a little storage capacity and your mother gave it a name such as salt, sugar, tea, etc. Similarly, a variable is a name given to a location in the computer’s memory to store values or data. A variable has a name (for reference), a type (which defines the kind of data it can store and its size), and a value (the actual data).



Naming Variables

Each variable in a program must have a unique name; you cannot use the same name for more than one variable. The name of a variable serves as an identifier.

Rules for Naming a Variable

Some rules for naming a variable are as follows:

- Name your variables in such a way that it describes their purpose.
- Variable names must begin with a letter or an underscore.
- No special characters or spaces are allowed in variable names.
- Uppercase and lowercase characters are distinct.
- Keywords (reserved words) cannot be used as a variable name. A keyword is a predefined word



QUEST



Information Literacy
Technology Literacy

Give one word answer to the following questions.

1. What is an action that triggers a response in programming? _____
2. What is the term for a block of code that executes when a specific event occurs? _____

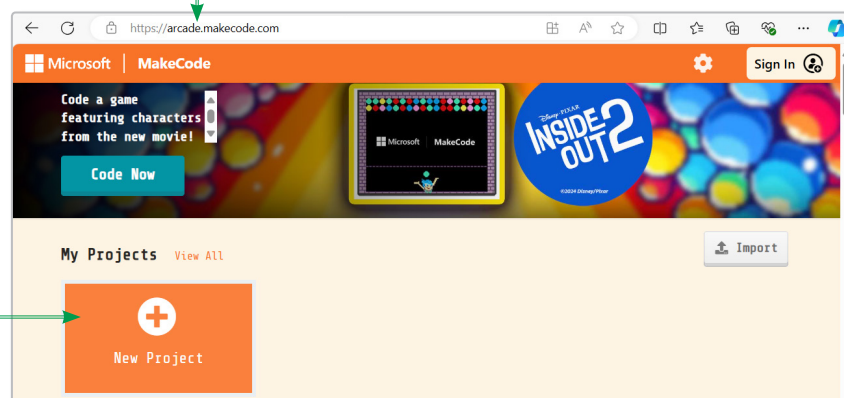
in any programming language.

Creating Variable

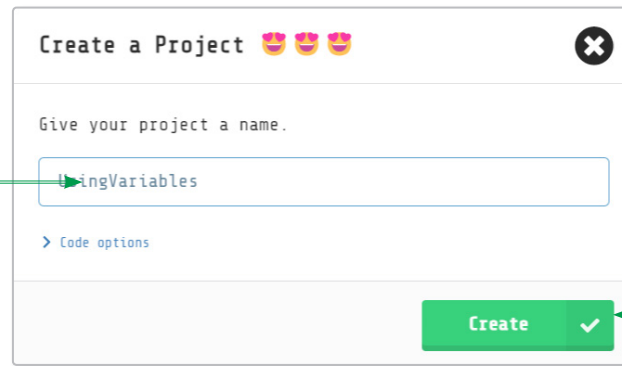
In MakeCode Arcade, we can create a variable using the **Variables** block category. Once a variable is created, you need to assign a value to it. This process of assigning a value to a variable is called **initialisation**. To create and initialise a variable in MakeCode Arcade, follow these steps:

1 Visit <https://arcade.makecode.com/>.

2 Click on the **New Project** button. The **Create a Project** dialog box appears.



3 Type the name of your project.



Create a Project 🥳🥳🥳

Give your project a name.

UsingVariables

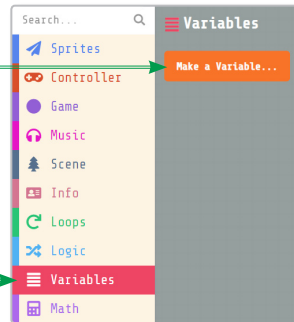
> Code options

Create ✓

4 Click on the Create button.

6 Click on the Make a Variable button.

5 Click on the Variables blocks category.



Search...

Variables

Make a Variable...

Sprites

Controller

Game

Music

Scene

Info

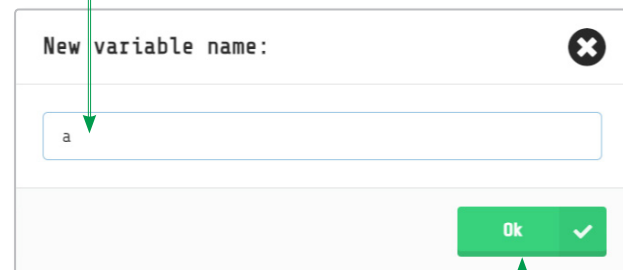
Loops

Logic

Variables

Math

7 Type a name for the variable.



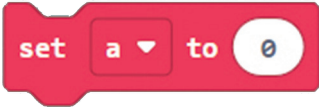
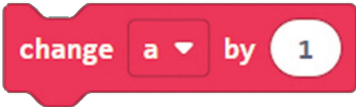
New variable name:

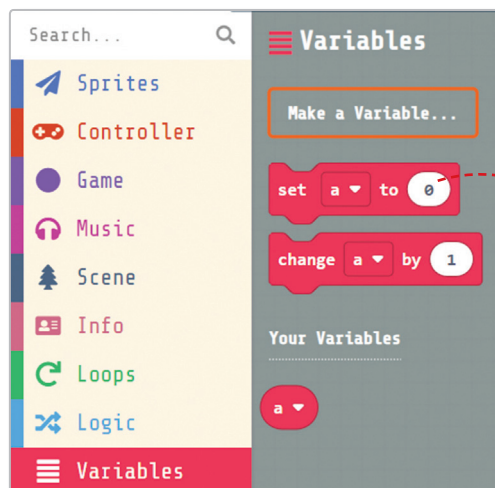
a

Ok ✓

8 Click on the Ok button.

The variable is created with the specified name (in our case it is a) and two new blocks:

Block	Name	Description
	set to	Assign a value to a variable.
	change by	Increase or decrease the value of a variable by a specified amount.



Search...

Variables

Make a Variable...

set a to 0

change a by 1

Your Variables

a

Sprites

Controller

Game

Music

Scene

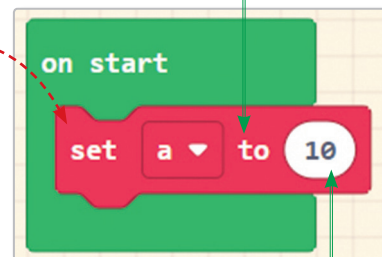
Info

Loops

Logic

Variables

9 Drag the set to block and drop it inside the on start block.

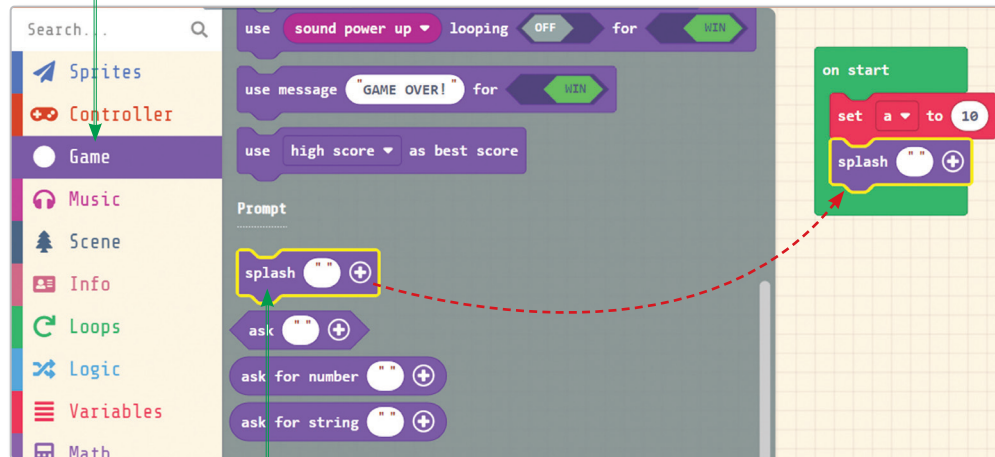


on start

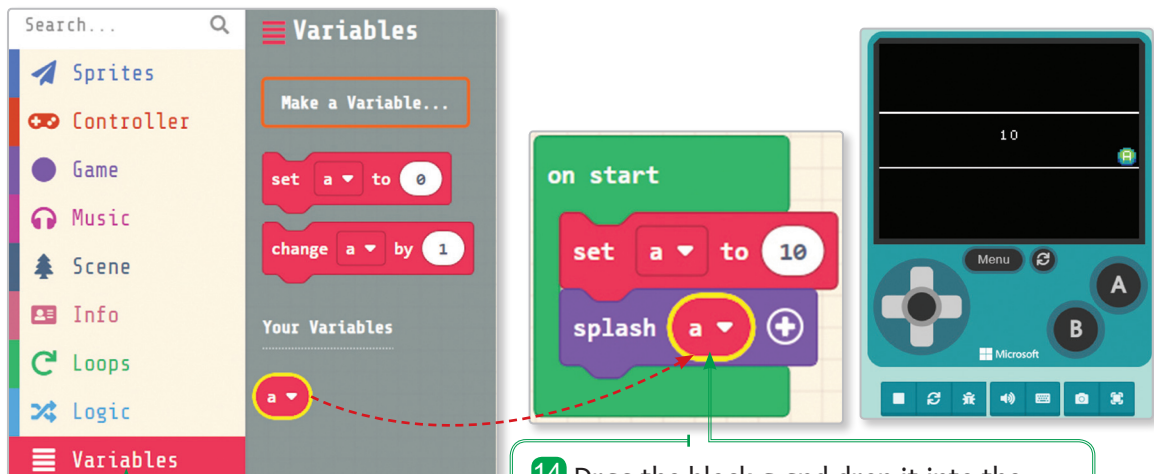
set a to 10

10 Change the value from 0 to 10. The variable a is initialised.

11 Click on the **Game** category of blocks.



12 Drag the **splash** block and drop it below the **set to** block.



13 Click on the **Variables** blocks category.

14 Drag the block **a** and drop it into the message area of the **splash** block. The value of the variable is displayed on the output.

Similarly, we can create many variables.

Using Math Blocks

In MakeCode Arcade, Math blocks are used to perform various mathematical operations and computations in your games and projects, like addition, subtraction, multiplication and division. The following table show different **Math** blocks:

Block	Name	Description
	addition	Return the sum of the two numbers.
	subtraction	Return the difference of the two numbers.

Block	Name	Description
	multiplication	Return the product of the two numbers.
	division	Return the quotient of the two numbers.
	remainder of	Return the remainder after performing the division operation.
	min	Return the minimum between two numbers.
	max	Return the maximum between two numbers.
	absolute of	Return the absolute value of a number.
	square root	Return the square root of a number.
	round	Round of a number to the nearest whole number.
	pick random	Generate random numbers within a specified range.

Let us create a simple project to perform the sum of two numbers. Follow the given steps to do so:

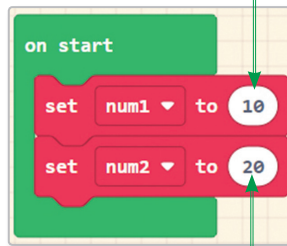
2 Drag the **set to** block and drop it inside the **on start** block.

3 Click on the down arrow of the **set to** block.

4 Select the **num1** variable.

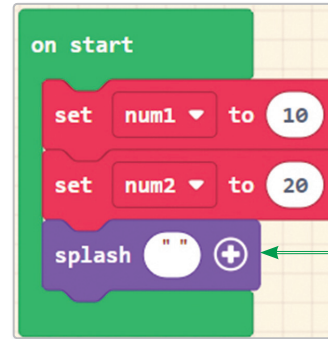
1 Create two variables named **num1** and **num2**.

5 Type **10** in the value box of the **set to** block for the **num1** variable.



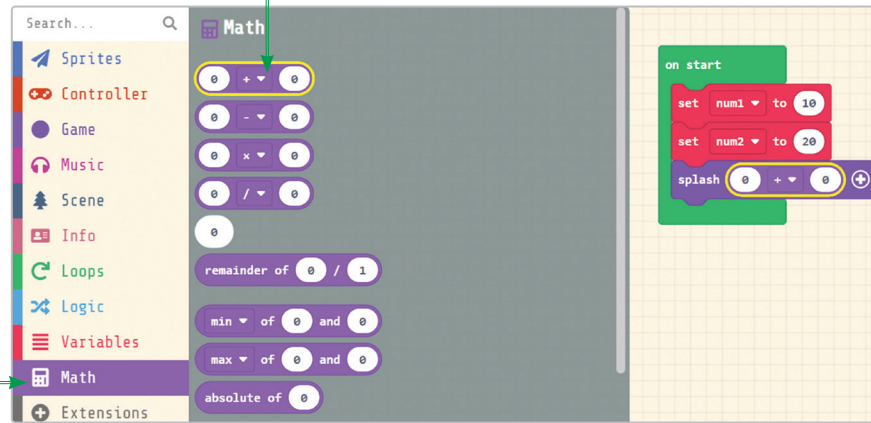
6 Drag the **set to** block again and drop it below the earlier **set to** block.

7 Type **20** in the value box of the **set to** block for the **num2** variable.



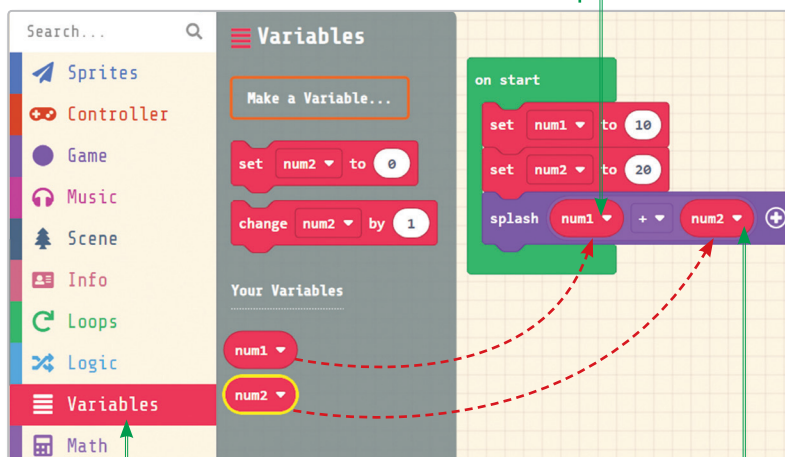
8 Drag the **splash** block from the **Game** category and drop it below the second **set to** block.

10 Drag the **addition** block and drop it into the message area of the **splash** block.



9 Click on the **Math** blocks category.

12 Drag the **num1** block and drop it into the left value box of the **addition** block.



11 Click on the **Variables** blocks category.

13 Drag the **num2** block and drop it into the right value box of the **addition** block. The output appears on the **Game** simulator.


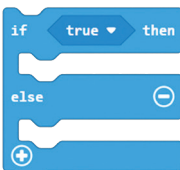


Similarly, we can use other Math blocks.




Using Logic Blocks

In MakeCode Arcade, logic blocks are used to handle decision-making and control flow in your projects. We can access these blocks by clicking on the Logic blocks category. The Logic blocks are divided into three types:

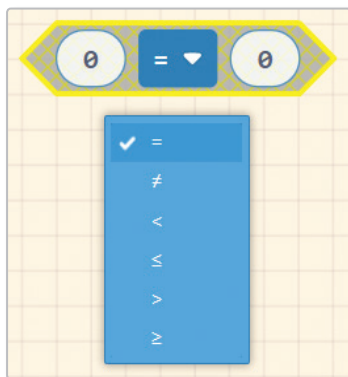
- **Conditionals:** These blocks, also known as decision-making blocks, are fundamental constructs in programming that allow a program to make decisions and execute different blocks of code based on certain conditions. There are two conditional blocks in MakeCode Arcade, which are as follows:

Block	Name	Description
	if-then	Allow us to execute code only if a certain condition is true.
	if-then-else	Allow us to execute one block of code if a condition is true, and a different block of code if the condition is false.


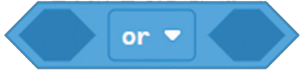

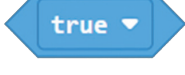
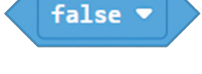
- **Comparison:** These blocks help you compare values. By default, MakeCode Arcade shows only three comparison blocks, which are as follows:

Block	Name	Description
	equal	Return true if both the numbers are equal.
	Less than	Return true if the first number is less than the second number.
	equal (for text)	Return true if both the text values are equal.

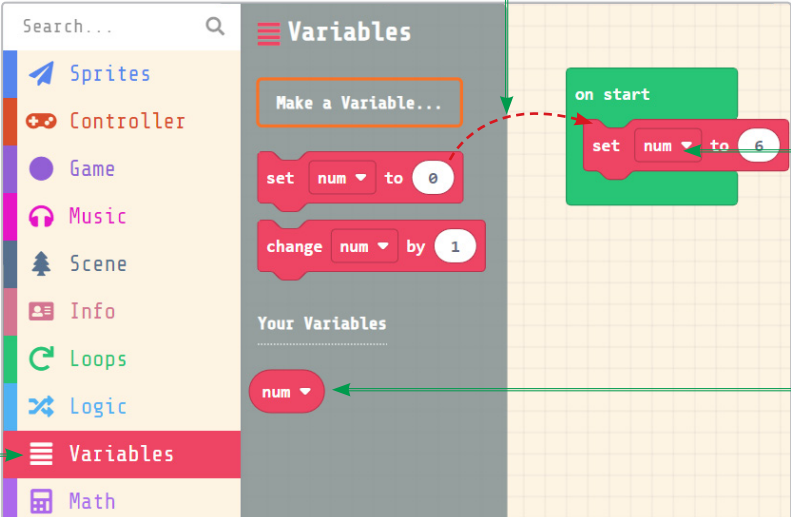
However, when we click on the down arrow of these blocks, we will get more comparison options like not equal to, greater than, greater than or equal to, and less than or equal to.



- **Boolean:** These blocks are used to combine or invert conditions. These blocks represent true/false values and are often used in conditions. They can be used to set or test boolean variables.

Block	Name	Description
	logical and	Return true if both the conditions are true.
	logical or	Return true if any of the conditions is true.
	logical not	Return true if the condition is false and vice-versa.
	boolean true	Set the value as true.
	boolean false	Set the value as false.

Let us create a project to display a message if a number is positive.

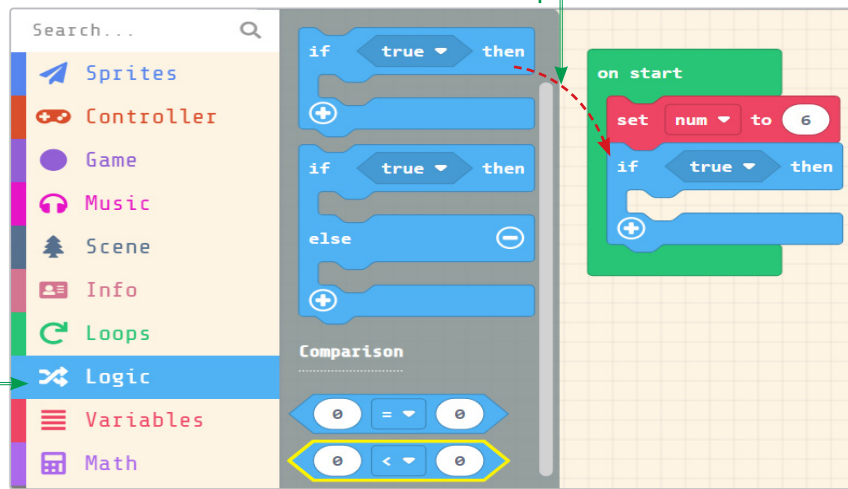


The screenshot shows the MakeCode Arcade interface. On the left, the 'Variables' category is selected in the block palette. In the center, the 'Variables' panel shows a variable named 'num' with a value of 0. On the right, the 'on start' block contains a 'set num to 6' block. Four numbered callouts provide instructions: 1. Click on the Variables blocks category. 2. Create a variable named num. 3. Drag the set to block and drop it inside the on start block. 4. Type 6 in the value box of the set to block to set the value of the num variable to 6.

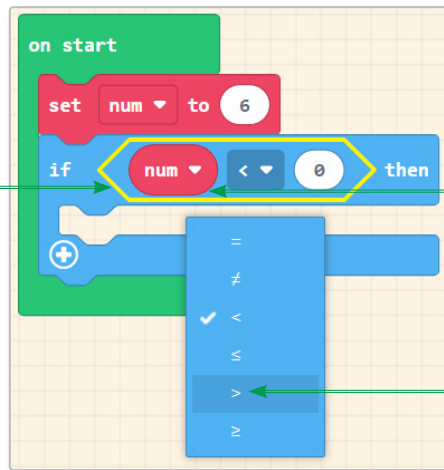
- 1 Click on the **Variables** blocks category.
- 2 Create a variable named **num**.
- 3 Drag the **set to** block and drop it inside the **on start** block.
- 4 Type **6** in the value box of the **set to** block to set the value of the **num** variable to **6**.

6 Drag the **if-then** block and drop it below the **set to** block inside the **on start** block.

5 Click on the **Logic** blocks category.

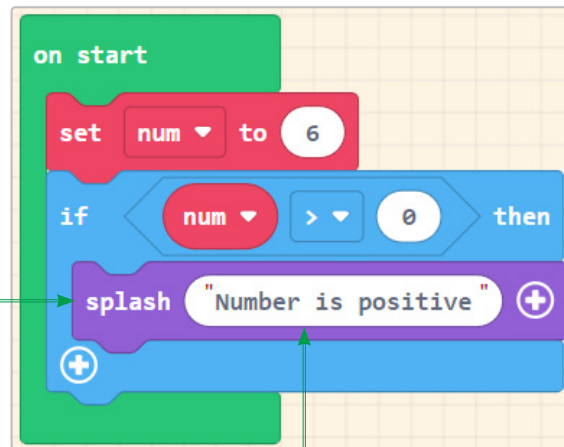


7 Drag the **less than** block from the **Logic** blocks category and drop it into the condition area.



8 Drag the **num** variable from the **Variables** blocks category and drop it into the **first value** box of the **less than** block.

9 Click on the down arrow of the **less than** block and select the **>** sign.



10 Drag the **splash** block from the **Game** blocks category and drop it inside the **if** block.

11 Type **Number is positive** in the message area of the **splash** block.

The output appears on the Game simulator.



If we change the value of the variable num to -6 in the set to block, no output will be shown because the condition becomes false.

Let us create another project to check whether a number is even or odd using the if-then-else block. Follow the given steps to do so:

A screenshot of the MakeCode Arcade interface. The left sidebar shows the "Variables" category selected. The main workspace shows the "on start" block with a "set number to 5" block inside it. The "Variables" panel on the right shows a variable named "number" with a value of 0. Four numbered callouts provide instructions: 1. Click on the Variables blocks category. 2. Create a variable named number. 3. Drag the set to block and drop it inside the on start block. 4. Type 5 in the value box of the set to block to set the value of the number variable to 5.

1 Click on the **Variables** blocks category.

2 Create a variable named **number**.

3 Drag the **set to** block and drop it inside the **on start** block.

4 Type **5** in the value box of the **set to** block to set the value of the **number** variable to 5.

5 Click on the **Logic** blocks category.

6 Drag the **if-then-else** block and drop it below the **set to** block inside the **on start** block.

7 Drag the **equal** block from the **Logic** blocks category and drop it into the condition area.

8 Drag the **remainder of** block from the **Math** blocks category and drop it into the first value box of the **equal** block.

9 Drag the **number** block from the **Variables** blocks category and drop it into the first value box of the **remainder of** block.

10 Type **2** in the second value box of the **remainder of** block.

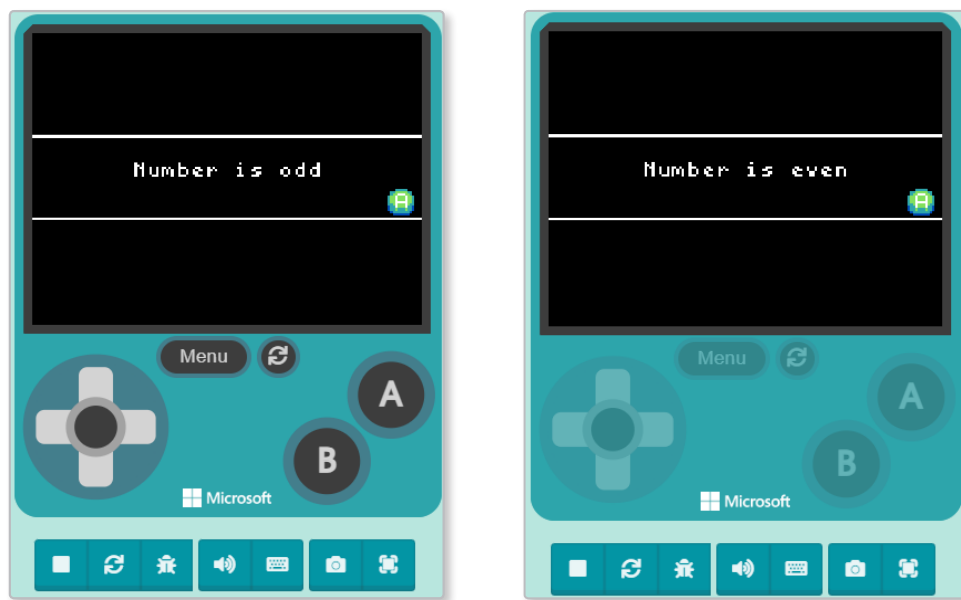
11 Drag the **splash** block from the **Game** blocks category and drop it inside the **if** block.

13 Drag the **splash** block from the **Game** blocks category and drop it inside the **else** block.

12 Type **Number is even** in the message area of the **splash** block.

14 Type **Number is odd** in the message area of the **splash** block.

The output appears on the Game simulator. Change the value of the number variable in the set to block to see more output.



QUEST



Critical Thinking
Information Literacy

Multiply two variables named `n1` and `n2` and store the result in a third variable named `result` in MakeCode Arcade. Display values of all the variables using splash block.

Do You Know?

You can test your games on physical devices like the Adafruit PyBadge, PyGamer, and even on the BBC micro.