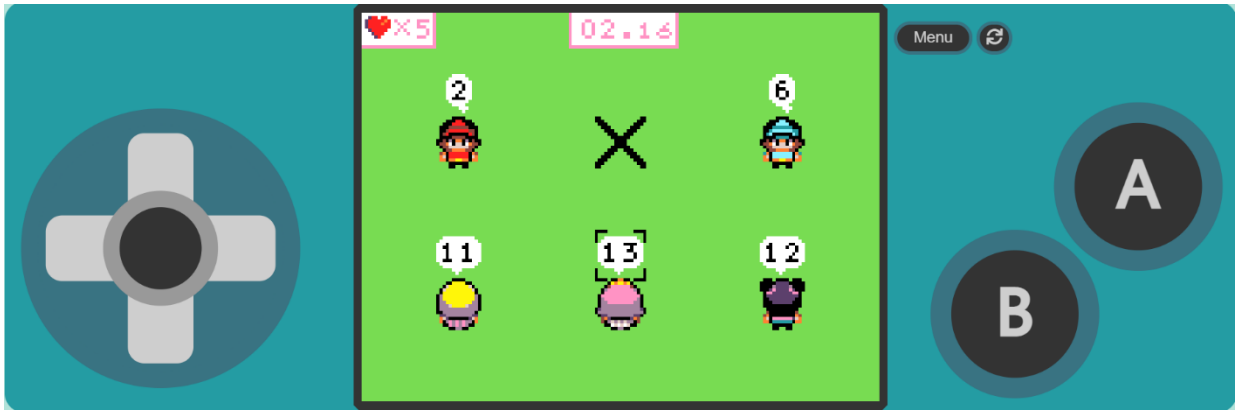
SECOND ARRAYS EXERCISE



DESCRIPTION

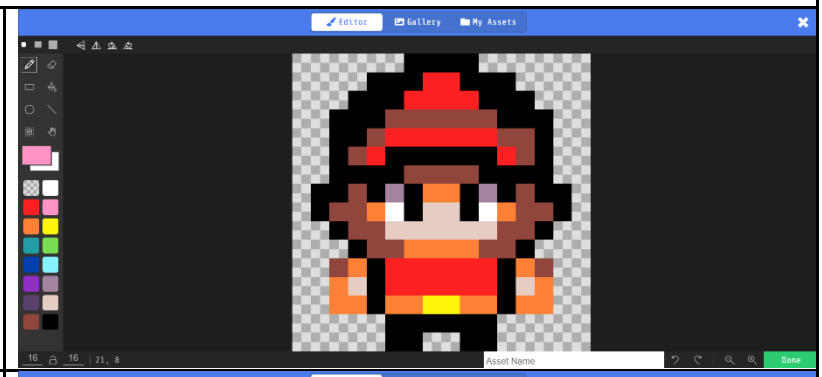
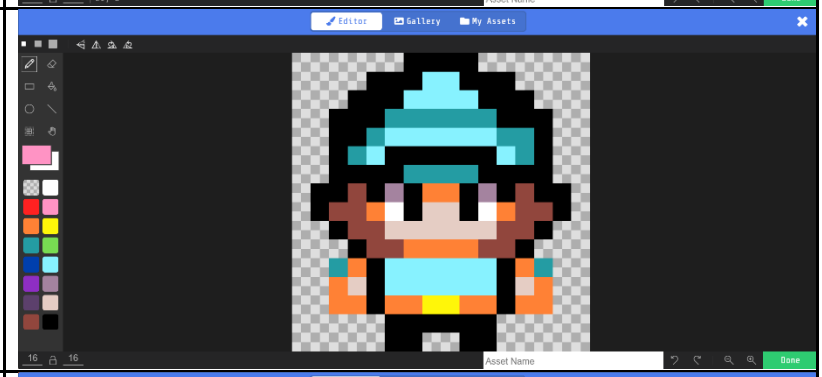
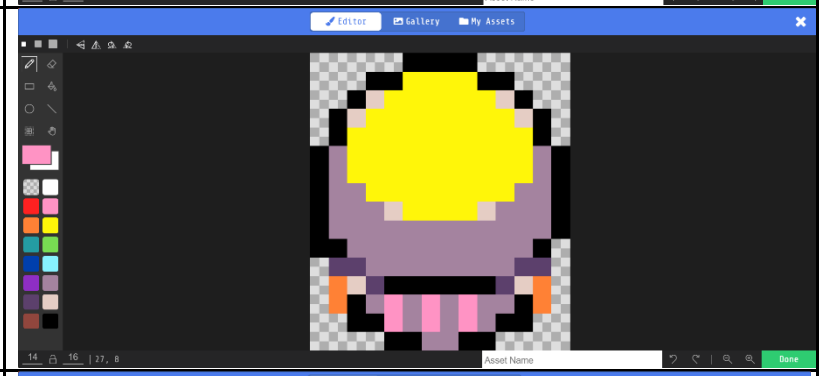
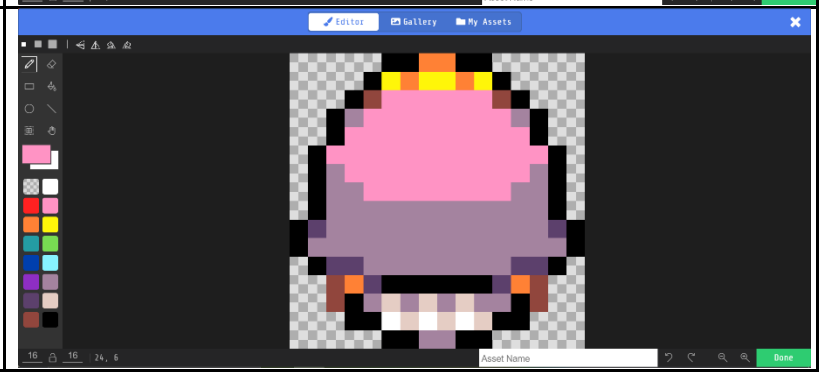
En este ejercicio crearemos un videojuego educativo de multiplicaciones.

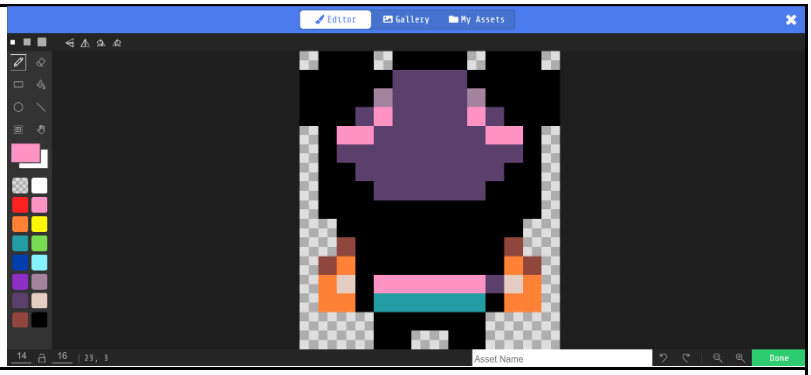
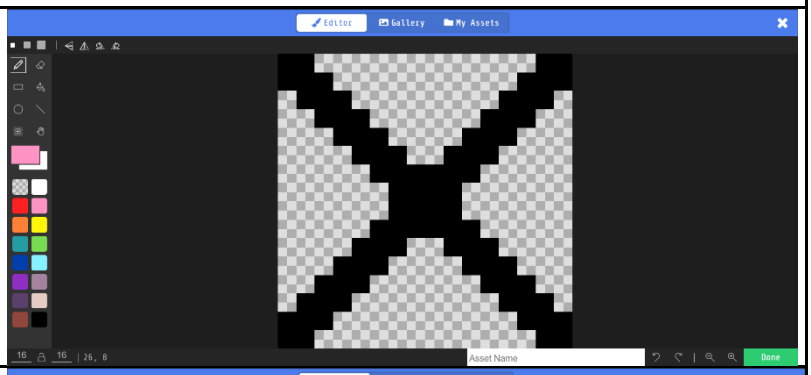
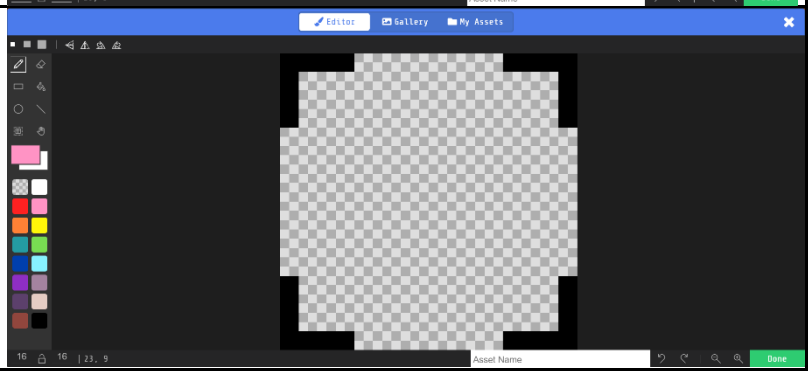
Para ello accederemos a **MakeCode Arcade** y realizaremos las operaciones necesarias.

GOALS

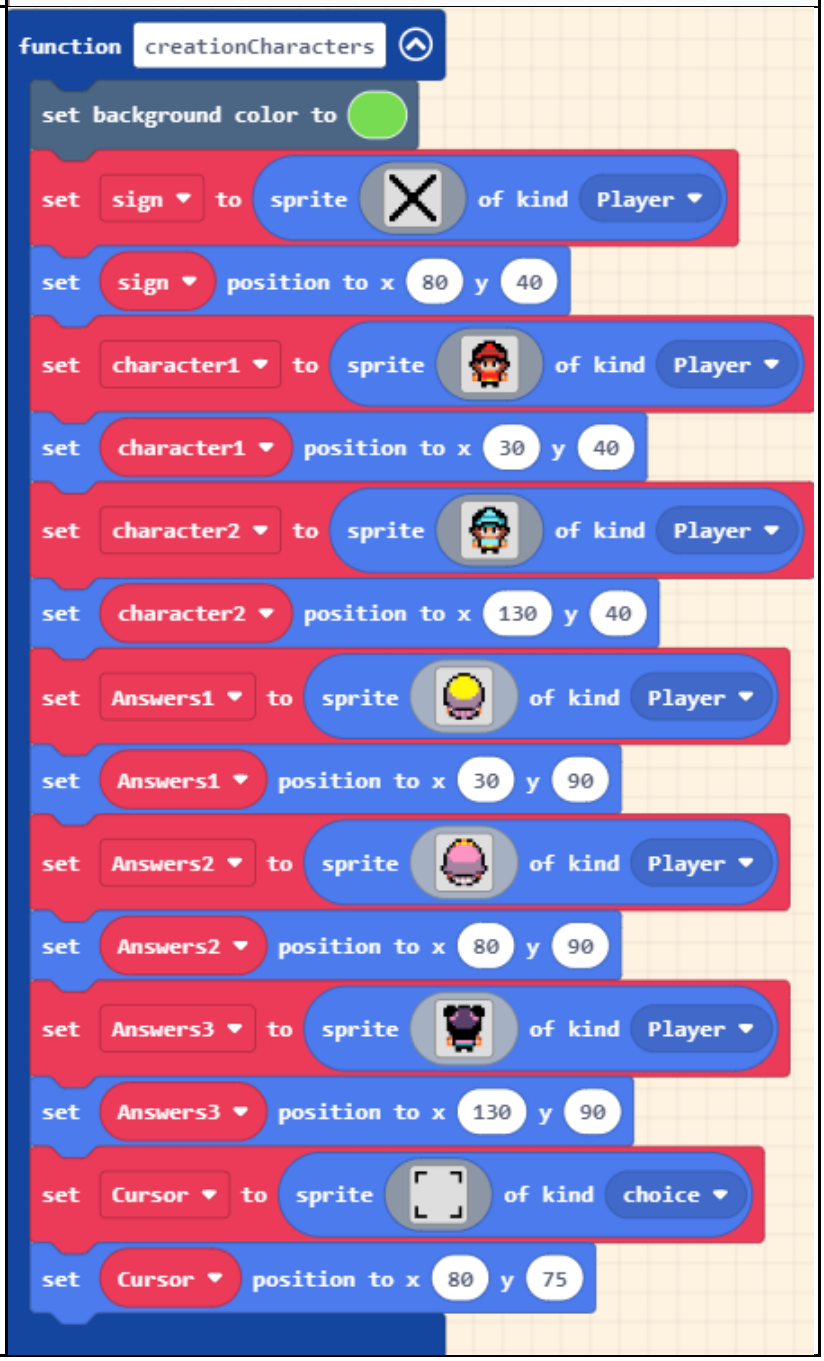
- Working with arrays using game control in MakeCode Arcade.
- Working with and understanding variables in MakeCode Arcade.
- Assigning a position to each game element.
- Using mathematical operations to solve problems.
- Converting numerical values to strings.
- Increasing the difficulty.

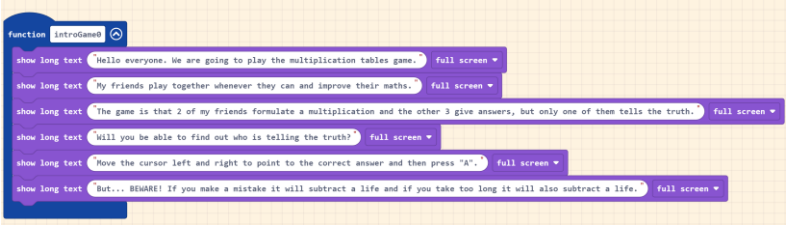
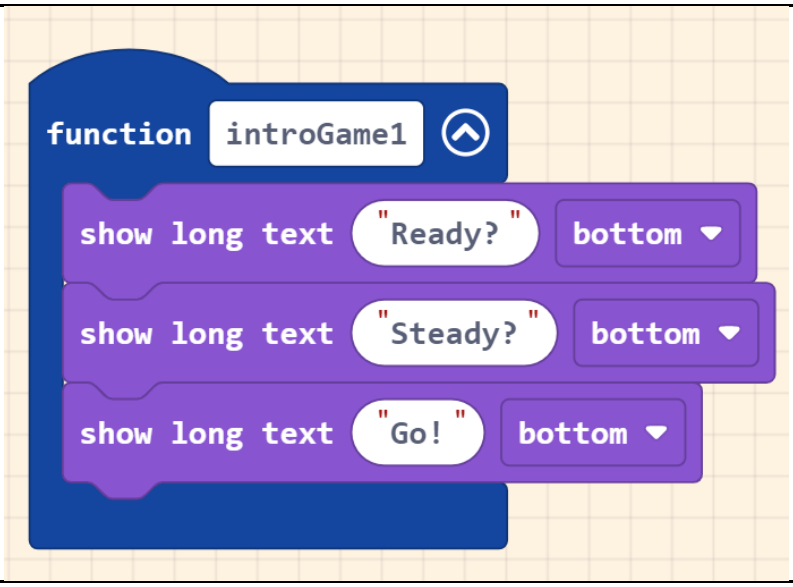
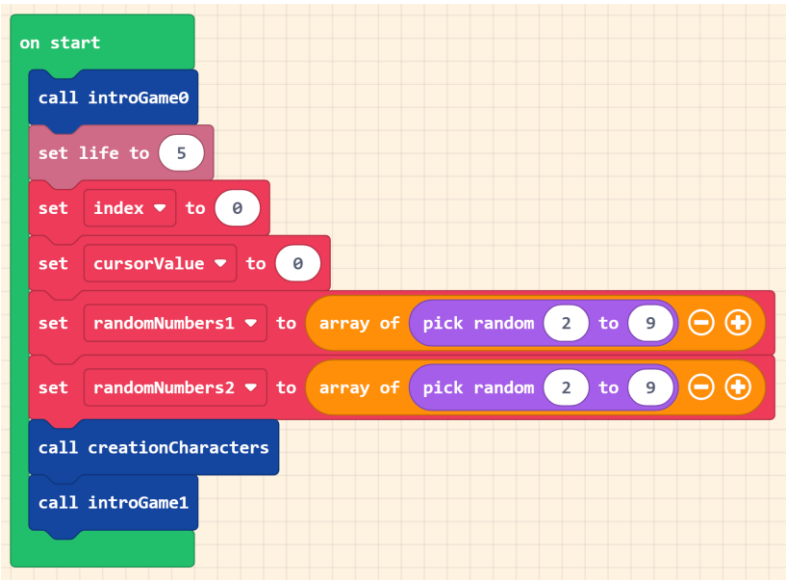
Programación del juego

ASSETS CREATION	
MAIN CHARACTER SPRITES CREATION	
We recommend using a 16x16 pixel grid for the character1 Sprite.	
We recommend using a 16x16 pixel grid for the character2 Sprite.	
We recommend using a 16x16 pixel grid for the Answers1 Sprite.	
We recommend using a 16x16 pixel grid for the Answers2 Sprite.	

We recommend using a 16x16 pixel grid for the Answers3 Sprite.	
ADDITIONAL SPRITES CREATION	
We recommend using a 16x16 pixel grid for the sign Sprite.	
We recommend using a 16x16 pixel grid for the Cursor Sprite.	

MAIN PROGRAMMING
CREATE ON START
CREATE FUNCTION creationCharacters

Let's start by creating a function to set up the game's environment and elements. This function will be responsible for initializing the game, creating the necessary sprites, and setting up the initial state.	
Here we will set the color of our background to green and also place all our player sprites in their positions on the stage. Finally, we will add the cursor sprite and assign it the "choice" type.	

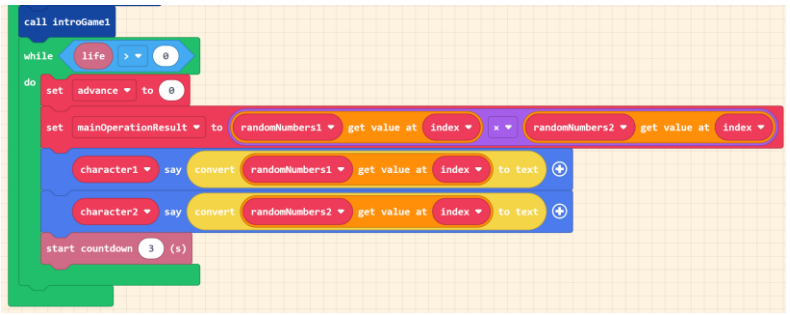
CREATE FUNCTION introGAME0	
We start by displaying a message on the screen to provide the player with information on how the game works.	
CREATE FUNCTION introGAME1	
Mostramos más mensajes de inicio de la partida	
Presentation of Elements and Initial Variable Declaration Programming	
In the **on start** block, we begin by setting variables and activating functions to display a game introduction. Both **introGame** functions show messages to the player, and **creationCharacters** creates and positions all the sprites in the game. The **index** variable will be used to look up certain positions in the arrays later on. The **cursorValue** variable will control the cursor. The **randomNumbers** variables will hold random values that the player needs to guess the result of multiplying both variables.	

PROGRAMMING LOOP TO RESET SCREEN FOR EACH ATTEMPT

We will create a loop that runs as long as the player has remaining lives. Inside the loop, we will set certain variables, display a message for the current operation using characters, and start a countdown.

The **advance** variable will be used to check if we have selected a result.

The **mainOperationResult** variable will hold the correct result for the operation that the player needs to guess.



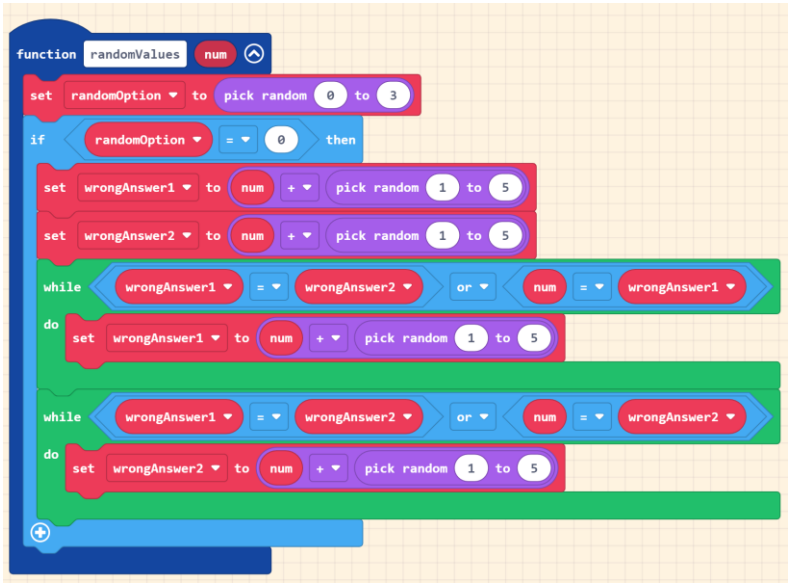
CREATE FUNCTION RANDOMVALUES

We will create the **randomValues** function with a numeric parameter. This function will generate variations in the answers, adding a certain level of randomness.

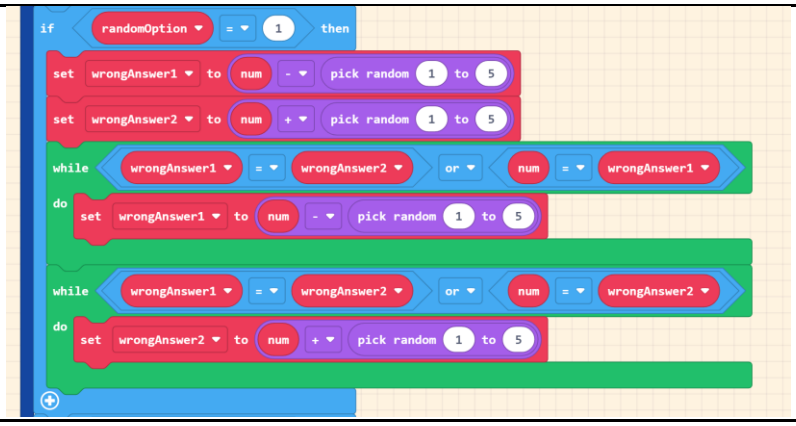
We will start by generating a random value for **randomOption**. The number of possible values for this variable will determine the patterns for the incorrect results.

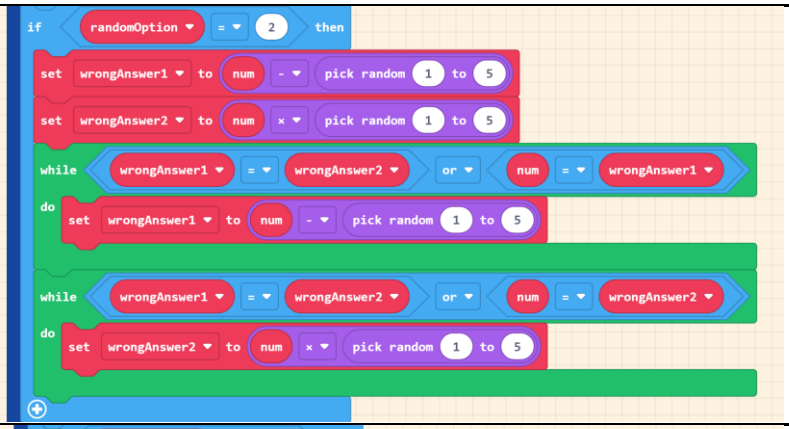
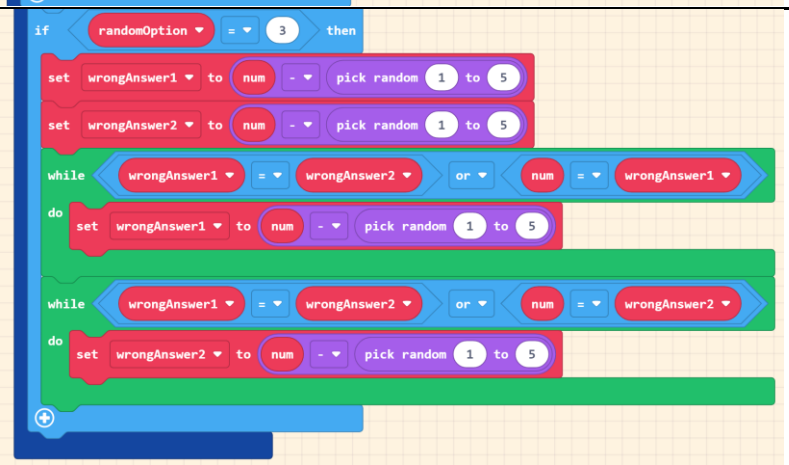
Starting with **randomOption** equal to 0, we will assign values to the **wrongAnswer** variables. These values will be the parameter value plus a random number between 1 and 5.

Next, we will use while loops to ensure that the options for the player to choose from do not have any duplicate numbers.



Here's an updated version where we duplicate the set of blocks under the **randomOption = 0** condition and change the values to create another possible result. In this case, we will subtract from **wrongAnswer1**.



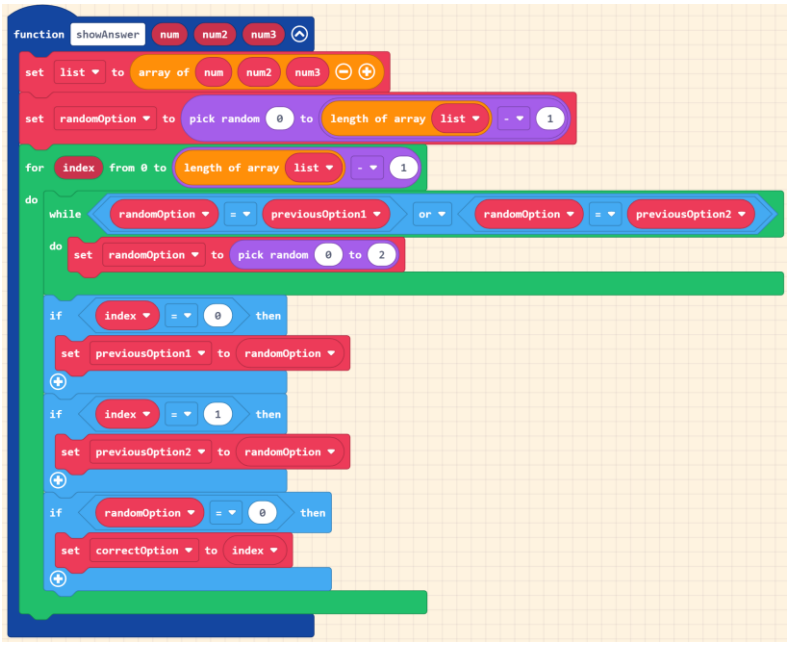
Here's an updated version where we modify the previous group of blocks and use multiplication for **wrongAnswer2**.	
In the last variation of possible incorrect results, we will use subtraction for both **wrongAnswer1** and **wrongAnswer2**.	

CREATE FUNCTION SHOWANSWER

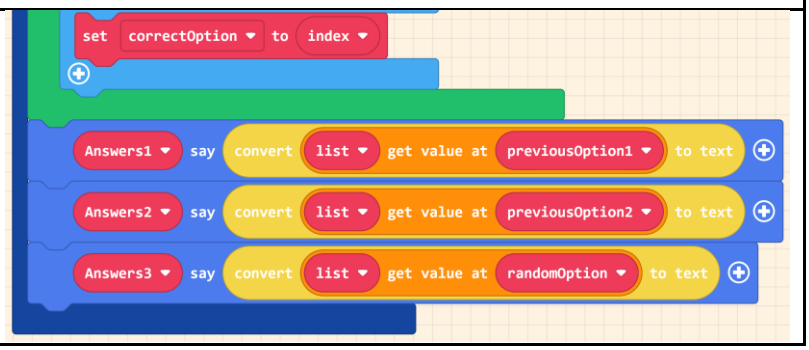
We will create a function with 3 numerical parameters called **shortAnswer**. The parameters will contain the results to be chosen by the player. Within this function, we will place the correct result in different positions, filling the other spaces with incorrect results.

We will create an array with the different values entered in the parameters. Then, we will create a variable that has a random value between 0 and the size of the **list** array minus one. In this case, that operation results in 2.

We will use a loop to iterate through all the spaces of the **list** array. We will assign the positions to the variables, which will be displayed later.

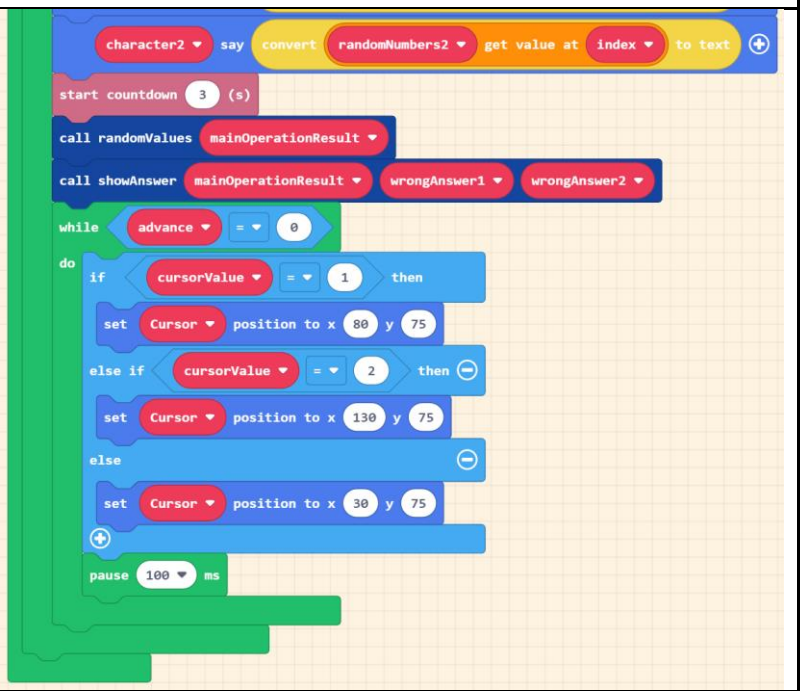


To conclude the function, we will program certain characters to say the options that the player can choose from.

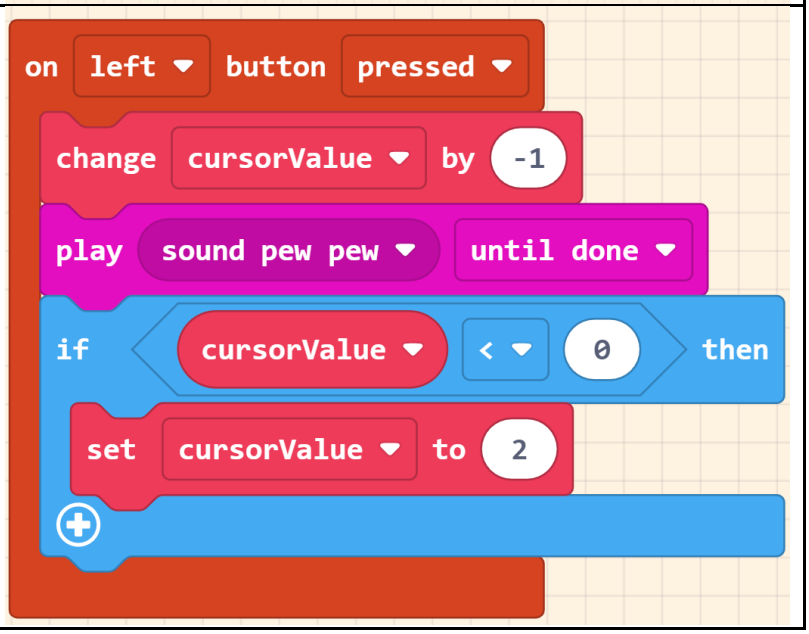


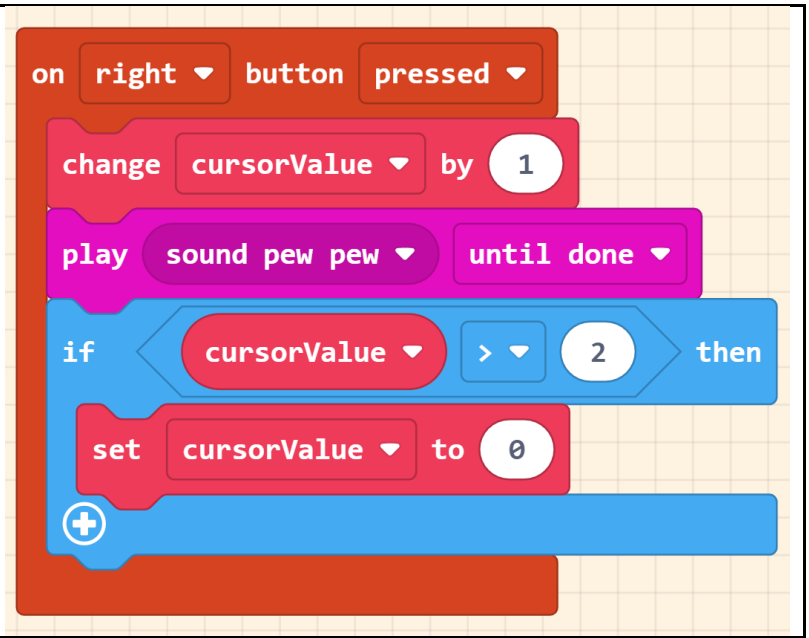
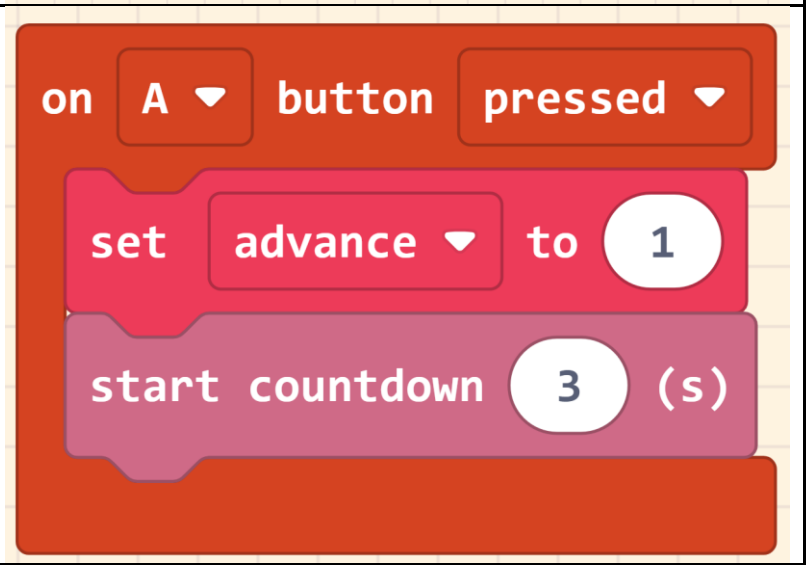
CREATING MECHANICS FOR CURSOR MOVEMENT AND RESULT SELECTION

We will continue with the "on start" block where we left off. We will add the "randomValues" and "showAnswer" functions, passing in the appropriate variables as parameters.
Then, we will use a loop to make the game continuously change the position of the cursor Sprite, with a small pause in between each movement to allow the player to see it.



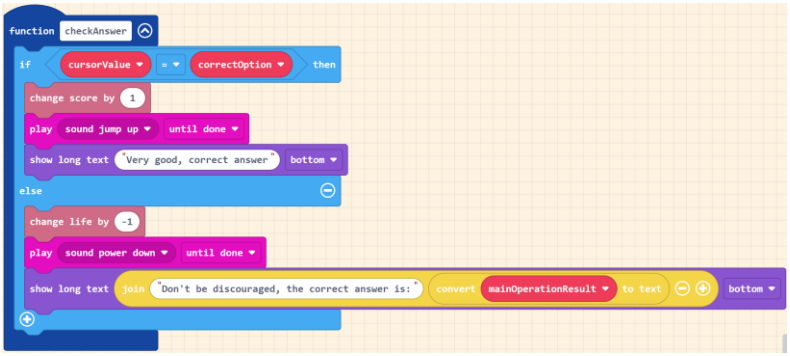
We will program the left button to decrease the value of **cursorValue**, making the Cursor Sprite move to the left. We will also add a sound effect to provide feedback to the player. Finally, we will check if the cursor is in the leftmost position, and if so, we will move it to the rightmost position.



For the right button, we will follow a similar approach as the left button, but with some modifications. We will increase the value of **cursorValue** when the right button is pressed, causing the Cursor Sprite to move to the right. We will also add a sound effect for feedback. Additionally, we will check if the cursor is in the rightmost position, and if so, we will move it to the leftmost position.	
For the A button, we will set the value of **advance** to 1, indicating that the player has made a selection. This will trigger the result evaluation. We will also reset the countdown timer to its initial state.	

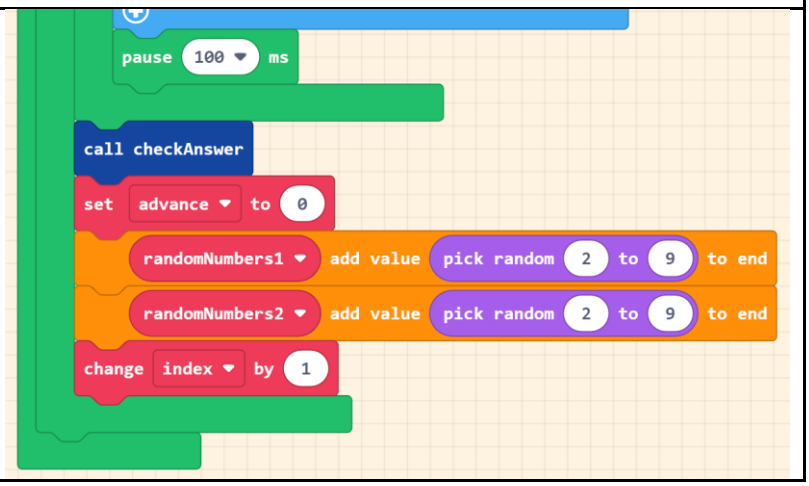
CREATE FUNCTION CHECKANSWER

We will create the **checkAnswer** function where we will check if the position of the cursor matches the correct answer option. If the correct option is chosen, we will increase the score, play a sound to indicate a correct result, and display a message congratulating the player.
In case of an incorrect answer, the programming will be similar, but with messages indicating that the answer is incorrect and the player loses a life instead of gaining points.



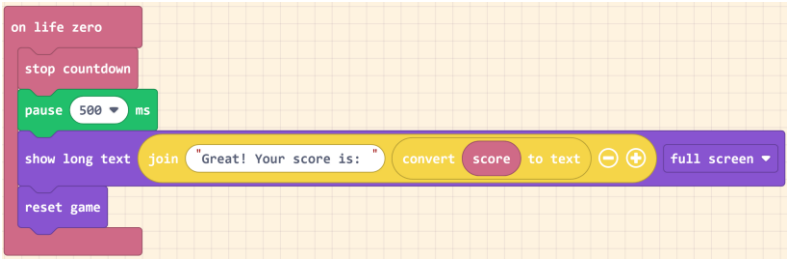
NEW OPERATION MECHANICS

We will go back to the end of the **on start** block. We will call the **checkAnswer** function. Since we have exited the while loop where **advance = 0**, we will reset this value to enter the loop again. We will add random values to **randomNumbers** to start the next round, and increase the value of **index** by 1 to progress in the corresponding arrays.

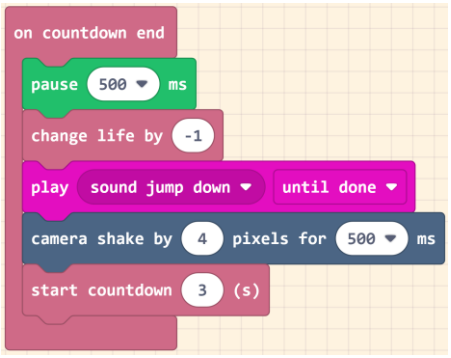


CREATING GAME OVER MECHANICS

To wrap things up, we will add an **on life zero** block to the workspace. We will stop the countdown to prevent any issues when the game is lost. We will include a half-second pause to prevent any further interactions in the game. We will display a text message and the final score of the game. Additionally, we will reset the game to play another attempt.



Finally, we will program what happens when the countdown ends. We will add a pause to allow the following programming to be appreciated without any issues. We will subtract one life. We will play a sound effect indicating that we have failed. We will shake the camera to create a sense of stress. We will reset the countdown for the next operation.



With this programming, our player will have to choose the correct result of the multiplication presented in the game. The position of the correct option will vary, and the patterns of incorrect results appearing will also change. We need to stay sharp to quickly identify the right choice since we only have 3 seconds to select the correct solution.

Now, it's your turn to customize and add content. Here's ours to inspire you a bit:

https://makecode.com/_VpfYb7gs1b2s