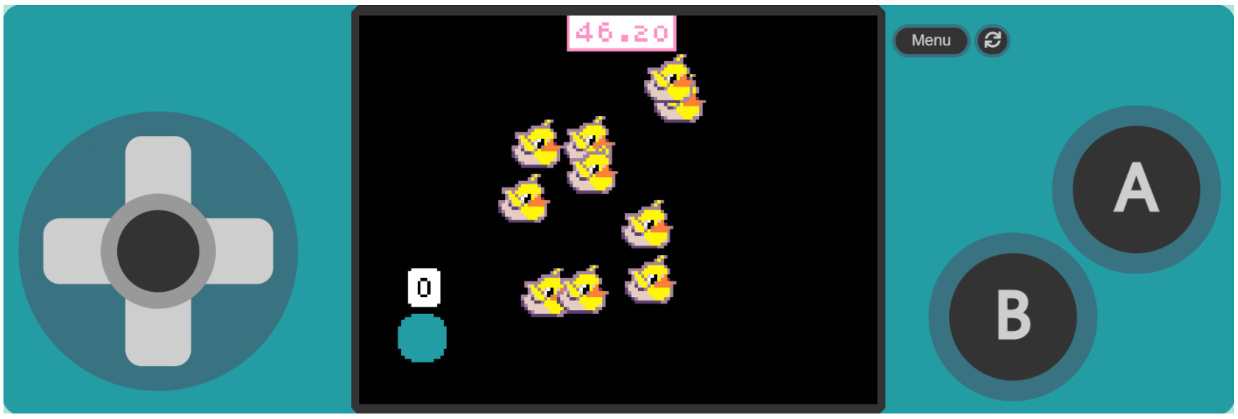
THIRD LOGIC TASK



DESCRIPTION

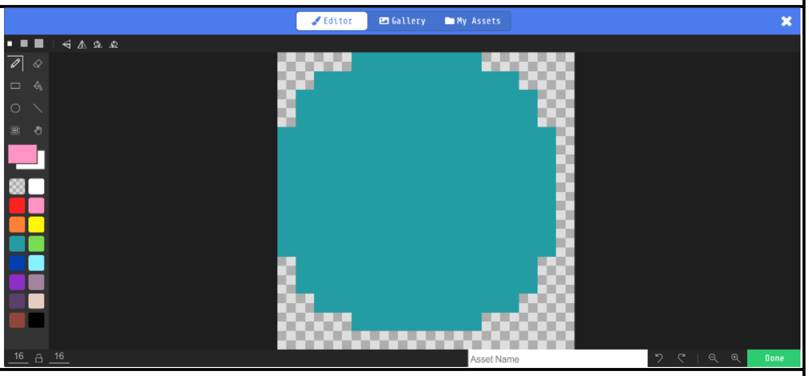
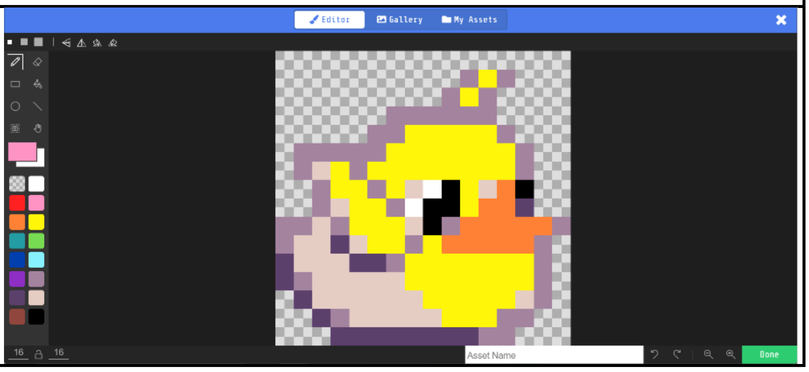
To create a Duck Hunt-style game where elements randomly appear on the screen and the player has to catch them with a ball.

In order to do that we go to **MakeCode Arcade** and do the following operations

Goals

- Work with game logic using controls in MakeCode Arcade.
- Work with and understanding variables in MakeCode Arcade.
- Assign an angle based on the duration of a key press.

Game programming

ASSETS CREATION	
PLAYER SPRITE CREATION	
We recommend using a 16x16 pixel grid for the player sprite.	
ADITIONAL SPRITE CREATION	
We recommend using a 16x16 pixel grid for the duck sprite.	

MAIN PROGRAMMING

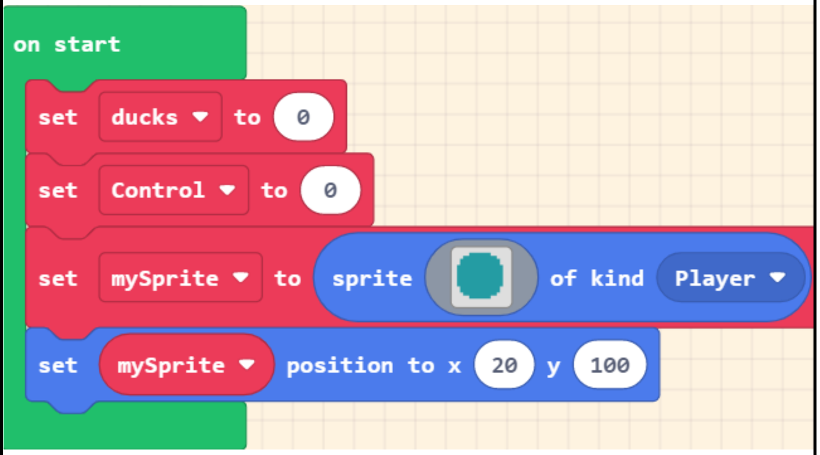
ON START GAME CREATION

In the "on start" event, we will place the variables (previously created) "ducks" and "control" and assign them a value of 0.

We will also include a sprite called "player" and assign it as a Player, giving it the position x=20 and y=100.

The variable "control" will indicate the speed at which the ball is launched.

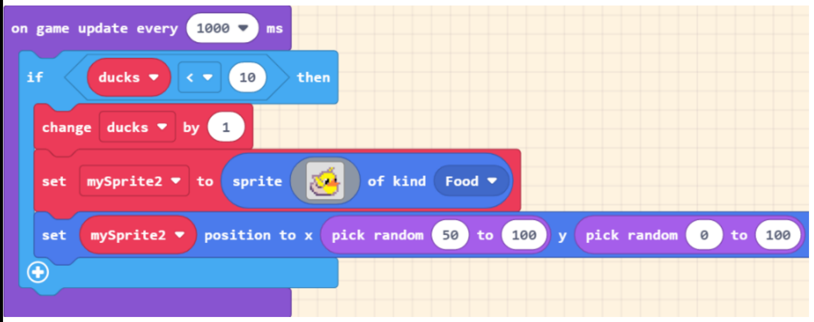
And the variable "ducks" will indicate the number of ducks on the screen.



```
on start
    set ducks to 0
    set Control to 0
    set mySprite to sprite of kind Player
    set mySprite position to x 20 y 100
```

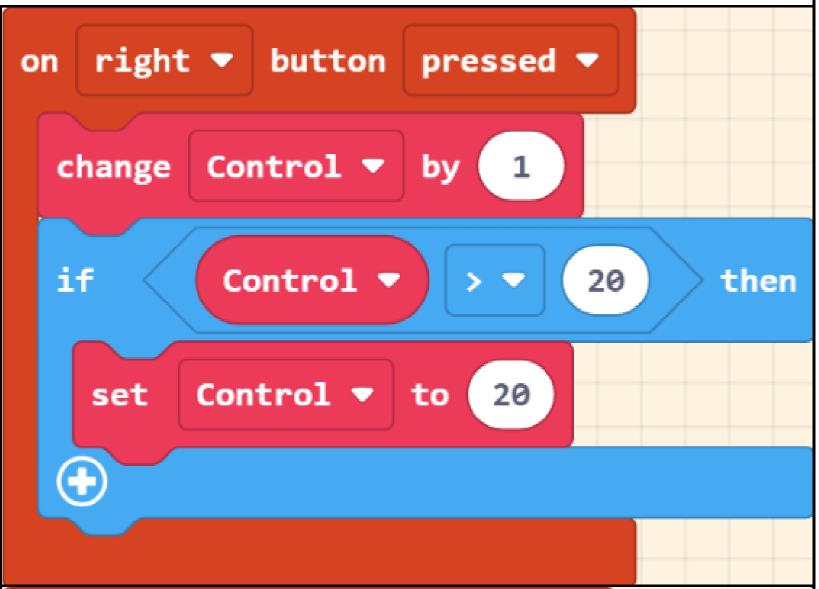
DUCKS SPAWN CREATION

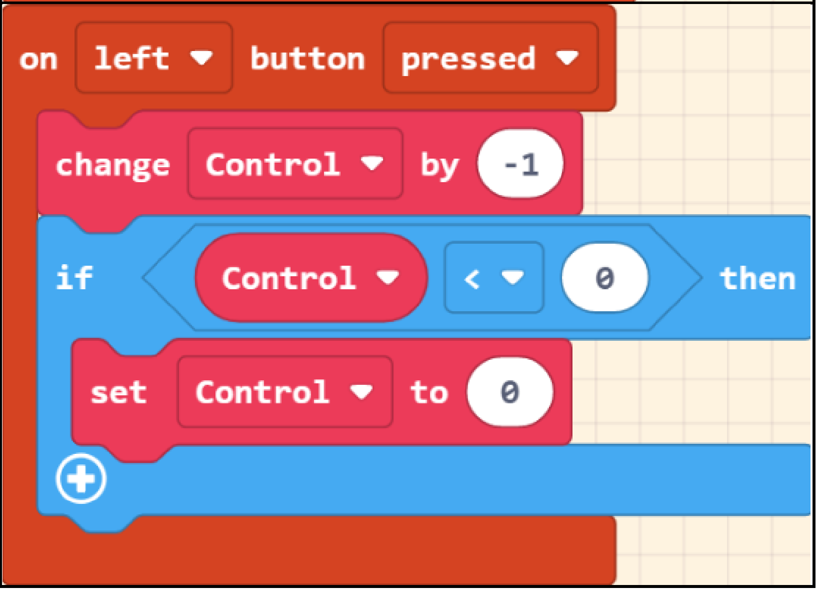
In an "on game update every" event, we will add a conditional "if" statement with the condition "if ducks is less than 10". Inside this conditional statement, we will include the following:
Change the value of "ducks" by 1.
Set "mySprite2" to a sprite of kind Food (using the sprite "duck").
Finally, we instruct it to place the sprite randomly on the screen.



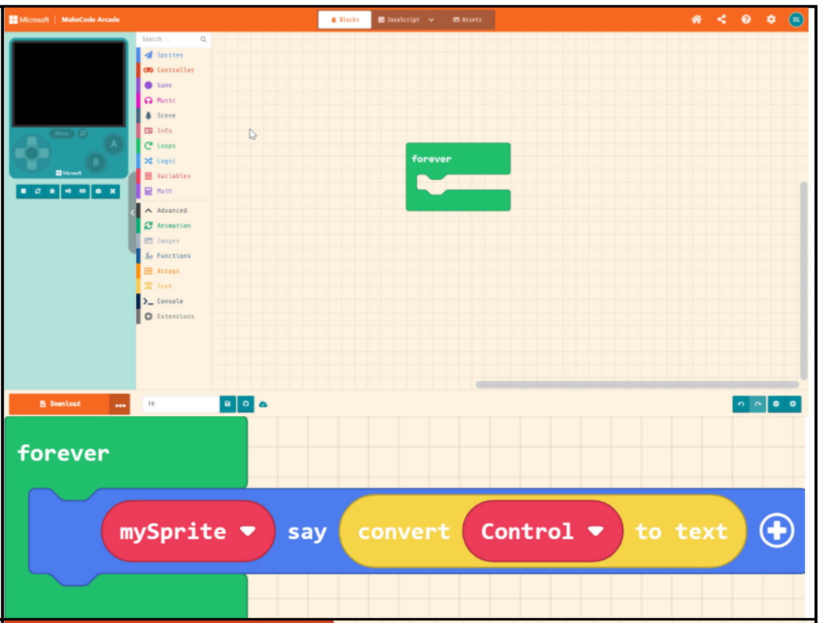
```
on game update every 1000 ms
    if ducks < 10 then
        change ducks by 1
        set mySprite2 to sprite of kind Food
        set mySprite2 position to x pick random 50 to 100 y pick random 0 to 100
```

LAUNCHING THE BALL MECHANIC	

To give movement to the ball so that it can catch or hunt the ducks that appear, we will instruct it to increment the variable "control" by 1 each time the right arrow key is pressed. We will also add a condition using an "if" statement to ensure that the maximum value of "control" is 20.	 `on right button pressed`
	`change Control by 1`
	`if Control > 20 then`
	`set Control to 20`

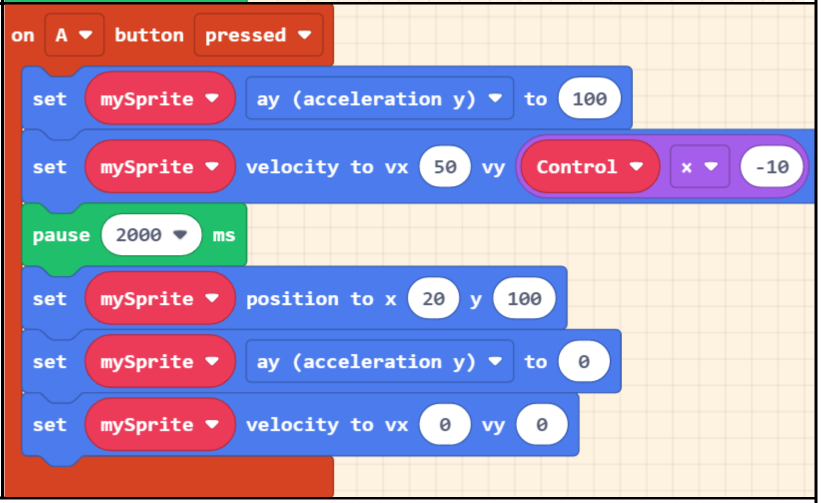
When the left arrow key is pressed, we will decrease the variable "control" by 1 until it reaches a minimum of 0.	 `on left button pressed`
	`change Control by -1`
	`if Control < 0 then`
	`set Control to 0`

To determine the amount of increase or decrease in the control variable, we use a forever loop and input the following: mySprite say convert Control to text
With this, we convert the numbers of the variable into text format and display them on the screen.



```
forever
    mySprite ▾ say convert Control ▾ to text ⊕
```

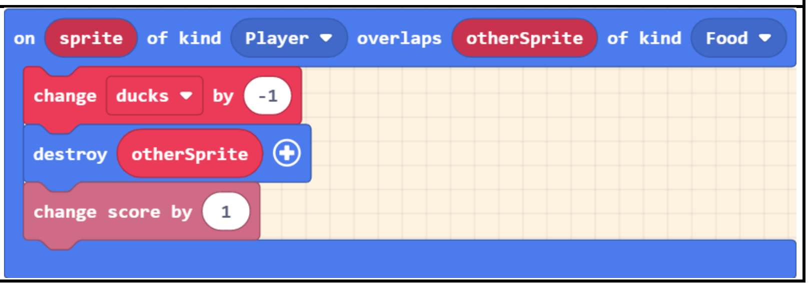
Finally, when pressing A button, we give the ball an acceleration in the y-direction of 100 and the velocity in the y-direction that we had accumulated in the control variable. Then, we wait for 2 seconds and set the original position, acceleration, and velocity.



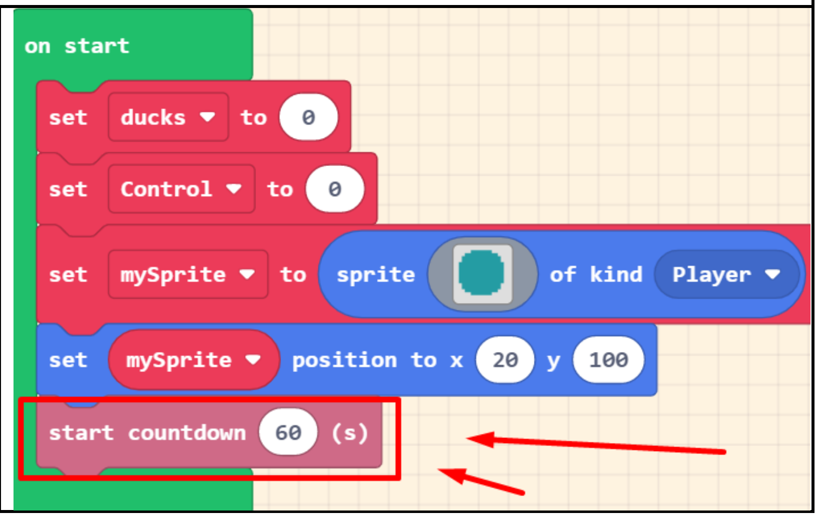
```
on A ▾ button pressed ▾
    set mySprite ▾ ay (acceleration y) ▾ to 100
    set mySprite ▾ velocity to vx 50 vy Control ▾ x ▾ -10
    pause 2000 ▾ ms
    set mySprite ▾ position to x 20 y 100
    set mySprite ▾ ay (acceleration y) ▾ to 0
    set mySprite ▾ velocity to vx 0 vy 0
```

SCORING POINTS MECHANIC

Now, to enable the ball to collect the ducks, we will give the following instructions when the Player sprite collides with a Food sprite:
Subtract 1 from the variable ducks.
Destroy the duck sprite.
Add 1 to the score.



```
on sprite of kind Player ▾ overlaps otherSprite of kind Food ▾
    change ducks ▾ by -1
    destroy otherSprite ⊕
    change score by 1
```

GAME OVER MECHANIC	
To prevent the game from becoming endless and add an incentive for improvement, we will introduce a start countdown in the "on start" event and set it to 60. This way, when the time runs out, the final score will be displayed, and players will be challenged to beat it.	

```
on start
    set ducks to 0
    set Control to 0
    set mySprite to sprite of kind Player
    set mySprite position to x 20 y 100
    start countdown 60 (s)
```

With this programming, the ducks will appear randomly on the screen, and we will have to use the ball to collect as many as we can within one minute. We have learned how to increase the number of elements on the screen and also how to give our throw more value to make it reach higher.

Now, it's your turn to customize and add content. Here's the description you provided for inspiration:

https://makecode.com/_3kg7webwHats

Glossary

Loops: A sequence of code that is executed repeatedly.

Conditionals: A sequence of instructions that are executed based on the value of a condition.

Comparison operators: Operators that compare one value to another and are used within a condition.

Variables: It is a space associated with an identifier, where a value can be stored and modified.

Event: Executes a sequence of instructions when an external event occurs in the system.

If: A conditional statement that executes a sequence of instructions or skips them based on the result of a logical operation.

physic: In video games, physics refers to the behaviour of different elements within an environment. They often simulate real-world physics.

Acceleration: The change in velocity per unit of time.

Velocity: A physical quantity that relates position to the rate of change of time.

Vectors: A line segment in a space representing a physical or mathematical quantity.

Life cycle: The duration of an element in a program from its creation to its destruction.

Aleatory: The generation of numbers that have an equal probability of occurring.

Score: The total points obtained by a player through certain interactions.

Countdown: A set amount of time that counts down and triggers something to happen, such as ending the game.

Game Over: The game has ended. It usually displays scores and asks if you want to play another game.