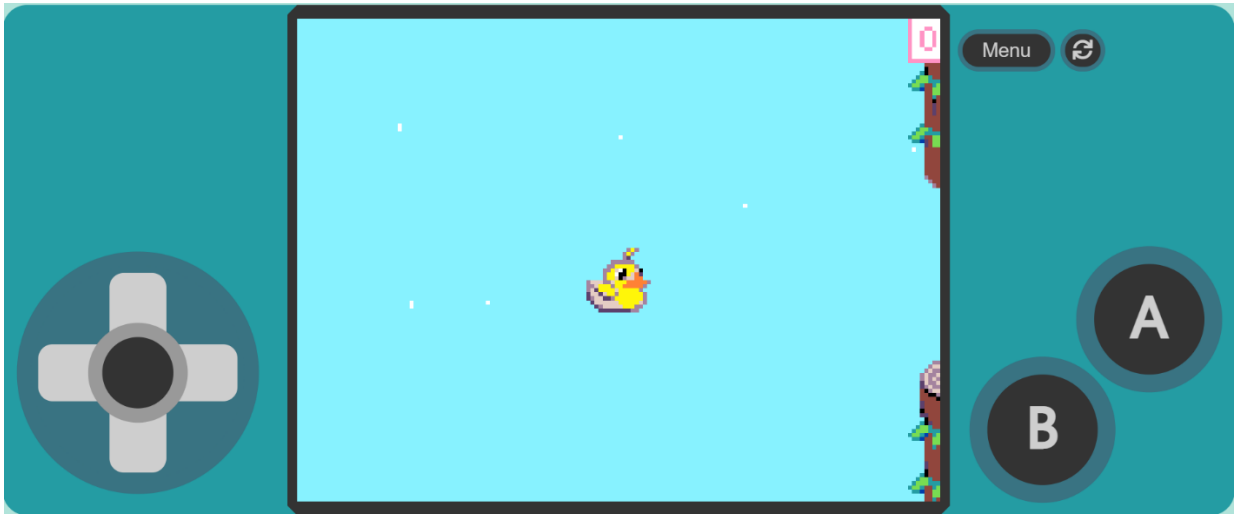
Tercer Ejercicio Motion



Descripción

En este proyecto realizaremos un juego basado en el conocidísimo “Flappy Bird”.

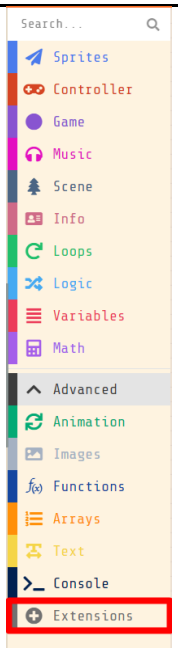
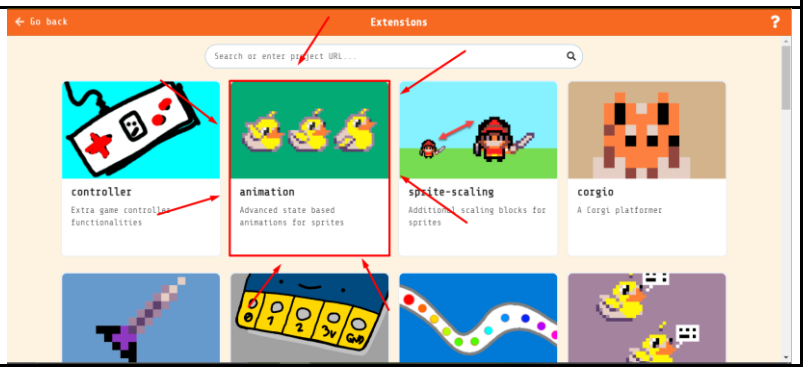
El objetivo principal del videojuego es sortear obstáculos tanto superiores como inferiores pulsando la tecla de salto.

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

Objetivos de programación y diseño

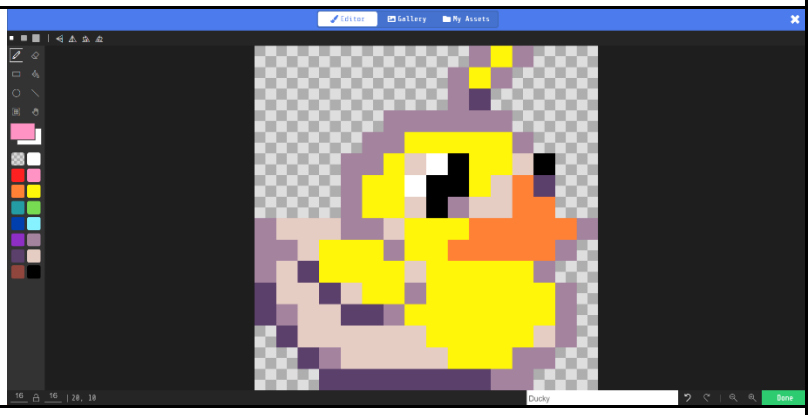
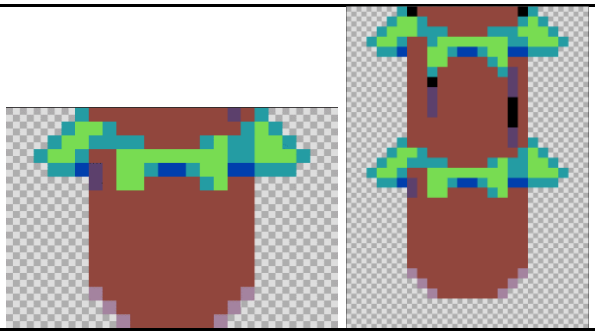
- Crear una simulación de ir en el aire.
- Crear un Sprite de nuestro personaje principal: “Ducky”. Que podamos controlar su salto.
- Crear una animación para nuestro personaje.
- Crear Sprites de los distintos obstáculos.
- Crear la mecánica de aparición de dichos obstáculos.
- Crear la mecánica de obtención de puntos.

Antes de empezar

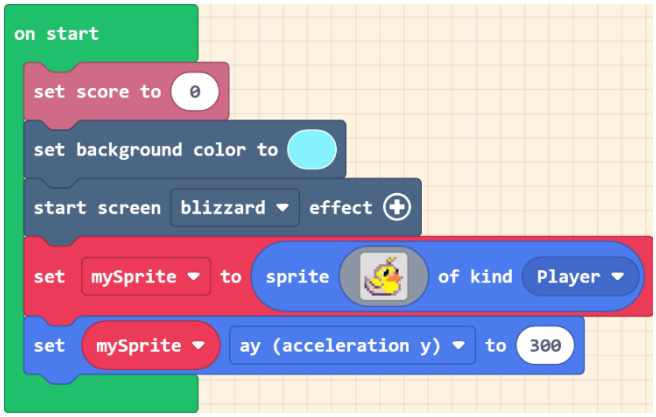
INSTALACIÓN DE EXTENSIONES NECESARIAS	
Para instalar una extensión, hacemos clic abajo del todo en **Extensions**.	
Una vez en las **Extensions** seleccionaremos “animation”.	

Programación del juego

Aquí os dejamos el enlace con parte de la programación y Assets hechos: https://makecode.com/_bk8E7fcfJPh3 **También podéis encontrarlos en "Gallery"**
CREACIÓN DE ASSETS
CREACIÓN SPRITE PRINCIPAL

Te recomendamos utilizar una matriz de 16x16 px para el Sprite de "Ducky".	
CREACIÓN SPRITES OBSTÁCULOS	
Crearemos los Sprites de los obstáculos de arriba obstacleUp0, obstacleUp1, obstacleUp2 y obstacleUp3 en una matriz de 24x16, 24x32, 24x48 y 24x56 px respectivamente.	

	
Crearemos los Sprites de los obstáculos de abajo obstacleDown0, obstacleDown1, obstacleDown2 y obstacleDown3 en una matriz de 24x24, 24x40, 24x56 y 24x64 px respectivamente.	

CREACIÓN SPRITE DE ANIMACIÓN	
Con nuestro Sprite de Ducky, haremos los distintos Sprites que compondrán las animaciones.	
PROGRAMACIÓN PRINCIPAL	
CREACIÓN INICIO DEL JUEGO	
Para comenzar, crearemos un bloque on start y colocaremos en su interior set score to lo establecemos en 0, set background color to que pondremos el color celeste, start screen blizzard effect, lo utilizaremos para hacer efecto de ventisca. Sprite de Ducky y por último le estableceremos una aceleración vertical de 300 para que caiga el personaje.	

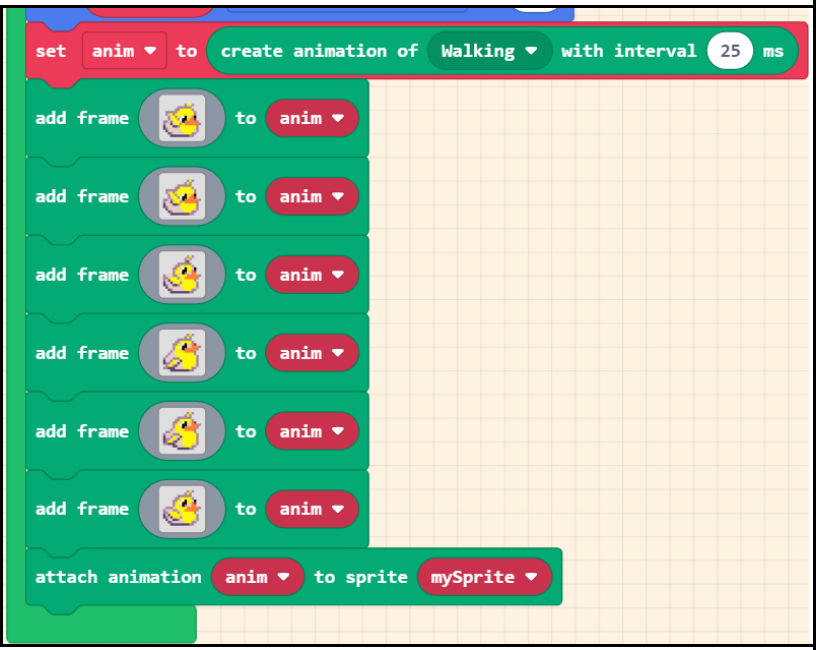
CREACIÓN ANIMACIÓN DEL PERSONAJE

Acto seguido, le haremos la animación a nuestro personaje.

Para ello usaremos el bloque de código set anim to créate animation of walkin with interval 25 ms. Que lo encontramos en Animation.

Luego le añadiremos los distintos frames que hicimos antes con add frame (Sprite de animación) to anim.

Por último, attach animation anim to sprite mySprite. Para asignarlo a nuestro personaje.



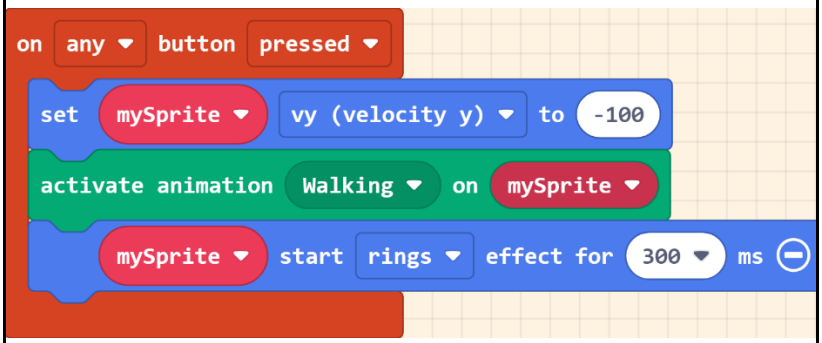
CREACIÓN MECÁNICA DE SALTO Y PERDER AL TOCAR EL FONDO

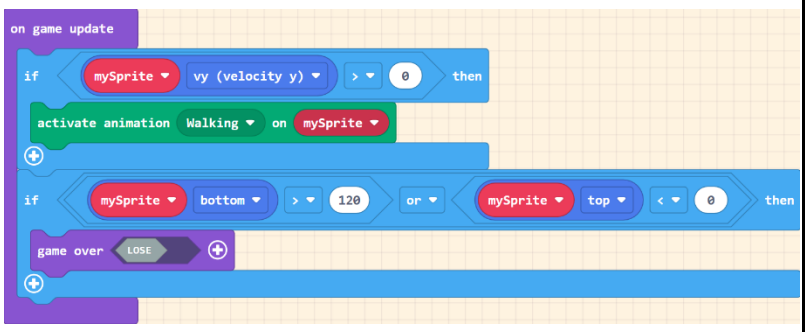
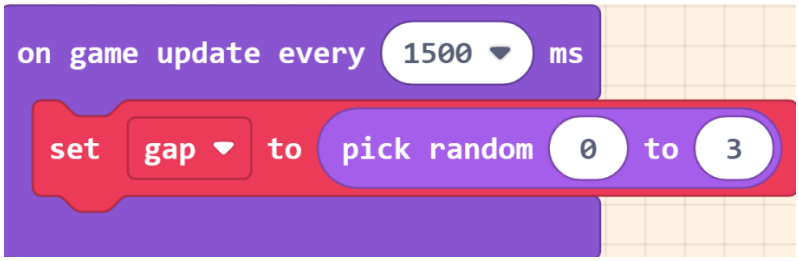
En Controller, usaremos el bloque on any button pressed.

Le añadiremos set mySprite vy (velocity y) to -100. Con esto hacemos que cuando pulsemos cualquier botón Ducky salte.

activate animation Walking on mySprite, para que se ejecute la animación creada antes.

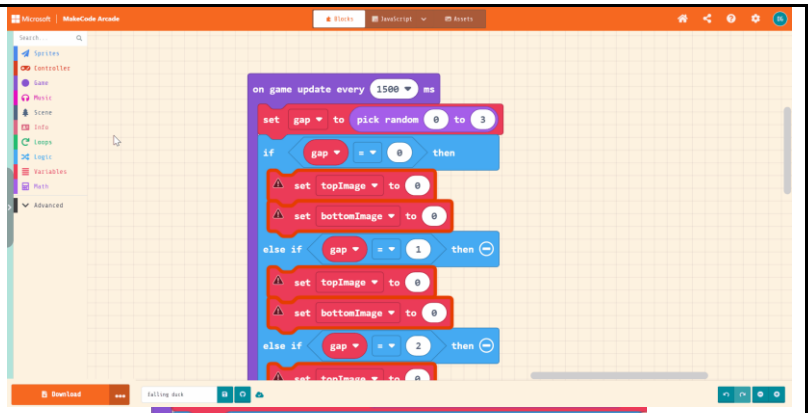
mySprite start rings effect for 300 ms. Esto añade un efecto visual al pulsar el botón y da sensación de impulso.

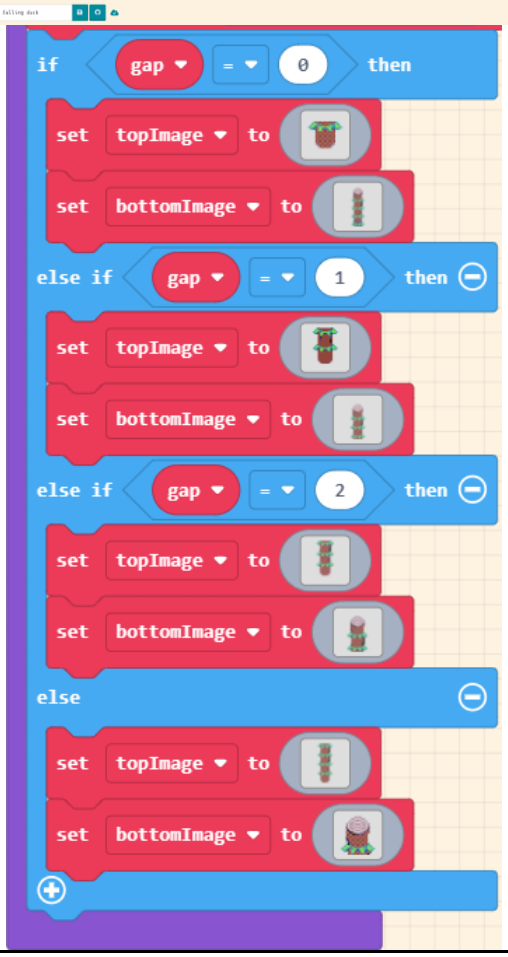


En on game update vamos a realizar varias comprobaciones. La primera será que, si nuestro personaje está cayendo, realice una animación. Para esto usaremos un if mySprite vy (velocity y) > 0 then y le colocamos activate animation Walking on mySprite. La otra comprobación será la de saber si nuestro personaje toca el borde de la pantalla para perder. Usaremos otro if mySprite bottom > 120 or mySprite top < 0 then y le colocamos un game over LOSE.	
CREACIÓN MECÁNICA APARICIÓN DE OBSTACULOS	
Para empezar, estableceremos una variable gap. Servirá para establecer que elementos salen en pantallas.	
En on game update 1500 ms vamos a realizar varias comprobaciones en función al número de nuestra variable. Primero haremos que nuestra variable gap tenga un número aleatorio entre 0 y 3.	

A continuación, haremos esas comprobaciones.

En un if, comprobaremos el valor de gap y en función a su valor hará una cosa u otra y esto se consigue con if, else if y else, tal y como se muestra en la imagen.



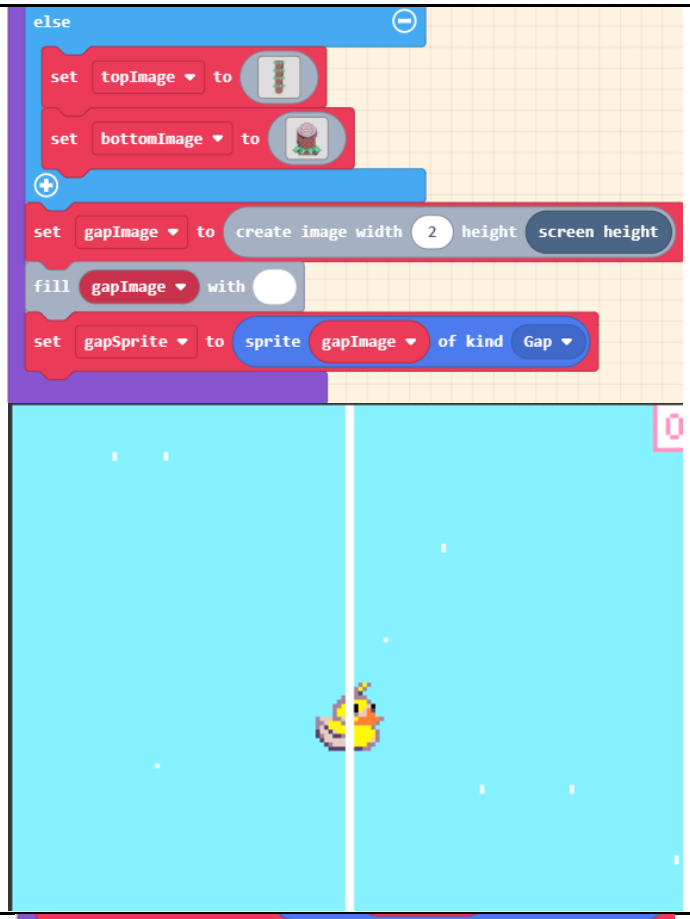


Para la aparición de los obstáculos, colocaremos una serie de instrucciones. Los colocamos justo a continuación de lo anterior.

set gapImage to (créate image width 2 height screen height). Aquí le pedimos al juego que nos cree una línea de 2 pixeles de ancho y de alto como el alto de la pantalla.

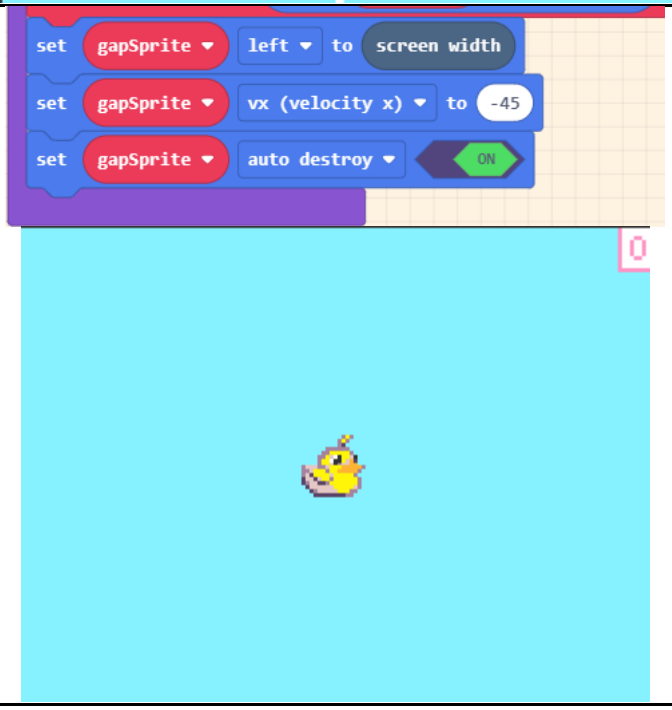
fill gapImage with (color blanco). Con esto le decimos que la línea sea de color blanco.

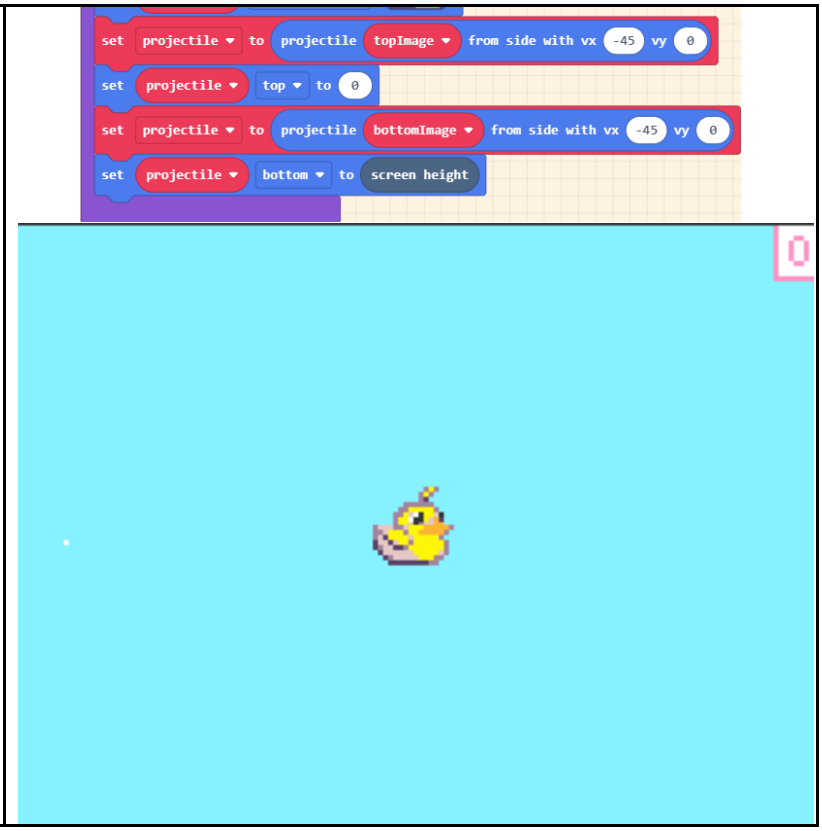
set gapSprite to (sprite gapImage of kind Gap). Aquí declaramos el dibujo anterior como Gap. Será el encargado de darnos los puntos posteriormente.

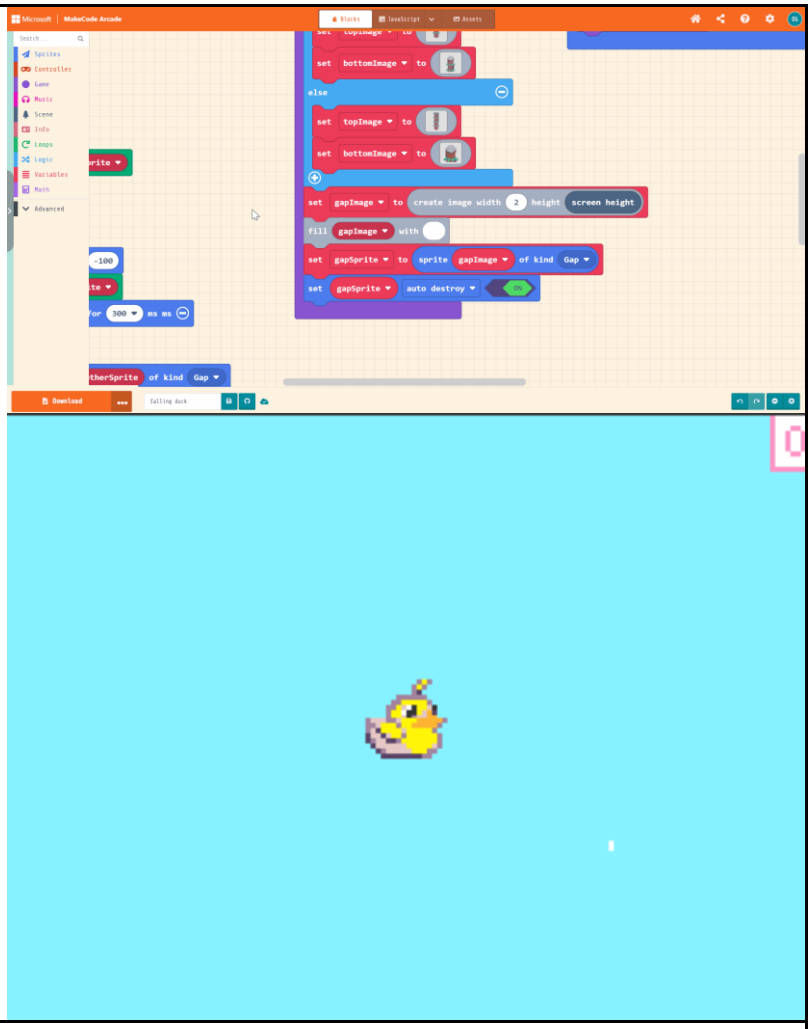
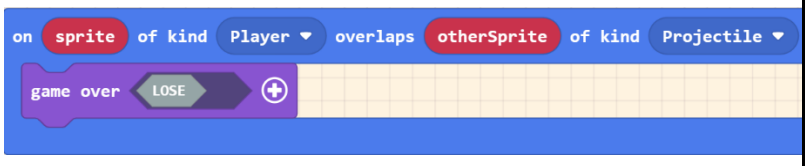


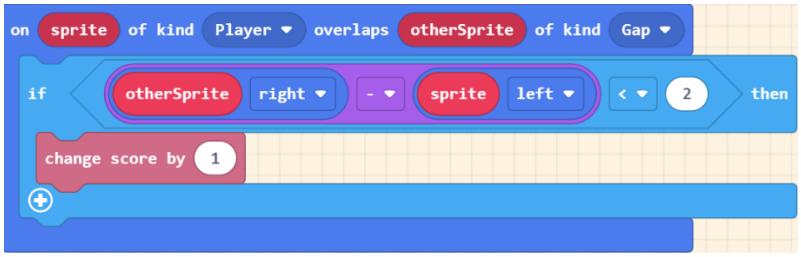
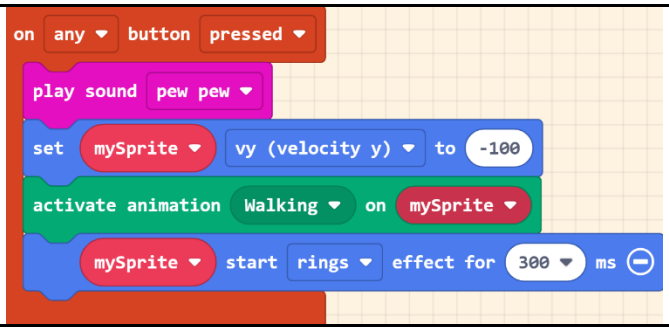
Justo a continuación, le decimos al juego que nos coloque esa barra justo a la izquierda de nuestro borde derecho de la pantalla set gapSprite left to screen wisth.

Que tenga una velocidad horizotal de -45 para que vaya a la izquierda y que se destruya al llegar al final de la pantalla.



Ahora le diremos que donde hay barras blancas, coloque nuestros Sprites de los obstáculos declarándolo como Projectile, colocándolos arriba y debajo de la pantalla y dándole una velocidad horizontal de -45 para que vayan hacia la izquierda.	

Por último, le indicaremos al juego que las barras blancas estén invisibles para darle mejor visual al juego.	
CREACIÓN MECÁNICA DE COLISIÓN CONTRA EL OBSTACULO	
Estableceremos la forma de perder en este juego al chocar nuestro personaje contra un obstáculo. Es muy sencillo, ya que lo hemos hecho en otros ejercicios. on sprite of kind Player overlaps otherSprite of kind Projectile y dentro colocamos game over LOSE.	

CREACIÓN MECÁNICA DE OPTENCIÓN DE PUNTOS	
No todo es perder en este juego, también merecemos ganar puntos y para ello haremos lo siguiente. Como anteriormente on sprite of kind Player overlaps otherSprite of kind Gap. Introduciremos un if en el cual le diremos que si nuestro personaje ha superado la barra, que sumemos un punto.	
ÚLTIMOS DETALLES	
Para darle un poco más de recursos al juego, le podéis añadir un efecto de sonido al pulsar cualquier tecla, tal y como se muestra en la imagen.	

Gracias a esta programación hemos hecho un “Flappy Bird”, en el cual, hemos aprendido a realizar obstáculos, colocarlos y darles velocidad. También hemos aprendido a realizar que cuando pasemos el obstáculo ganemos puntos. Es tu turno de personalizarlo y añadirle contenido. Aquí te dejamos el nuestro para que te inspires un poco: https://makecode.com/_37wRA4du7WgX

Glosario

Jugador: Un participante en un juego

Física: En videojuegos la física se refiere al comportamiento que tiene los distintos elementos dentro de un entorno. Suelen simular a las físicas del mundo real.

Aceleración: Es la variación de velocidad por unidad de tiempo.

Velocidad: Es una magnitud física que relaciona la posición con el incremento de tiempo.

Escenario: Espacio donde se desarrolla el videojuego.

Ciclo de Vida: Duración de un elemento de un programa desde su creación hasta su destrucción.

Aleatoriedad: La generación de números que tienen la misma probabilidad de generarse.

Puntuación: Puntos totales que obtiene un jugador al realizar ciertas interacciones.

Game Over: La partida se ha terminado. Se suele mostrar puntuaciones y te pregunta si quieres jugar otra partida.

Animación: La animación de un sprite es la sensación de movimiento que tiene por los diferentes frames que lo componen.

Imagen: Elemento que se visualiza en una pantalla y muestra algo (un paisaje, personas, etc)

Música: Combinación de sonidos y silencios que componen un ritmo.

Efecto: Algo que se aplica sobre el escenario, objeto, personaje y más elementos para transmitir realismo o una sensación dentro del juego.

Evento: Ejecuta una secuencia de instrucciones cuando ocurre un suceso externo al sistema.

If: Sentencia condicional que, según el resultado de una operación lógica, ejecuta una secuencia de instrucciones o se omite.