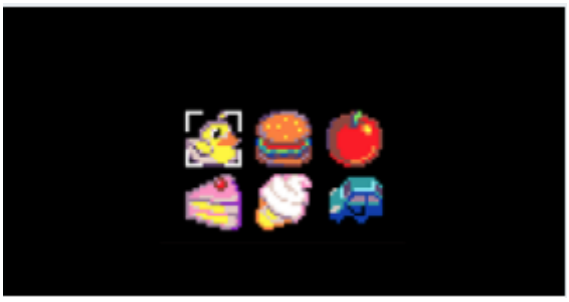
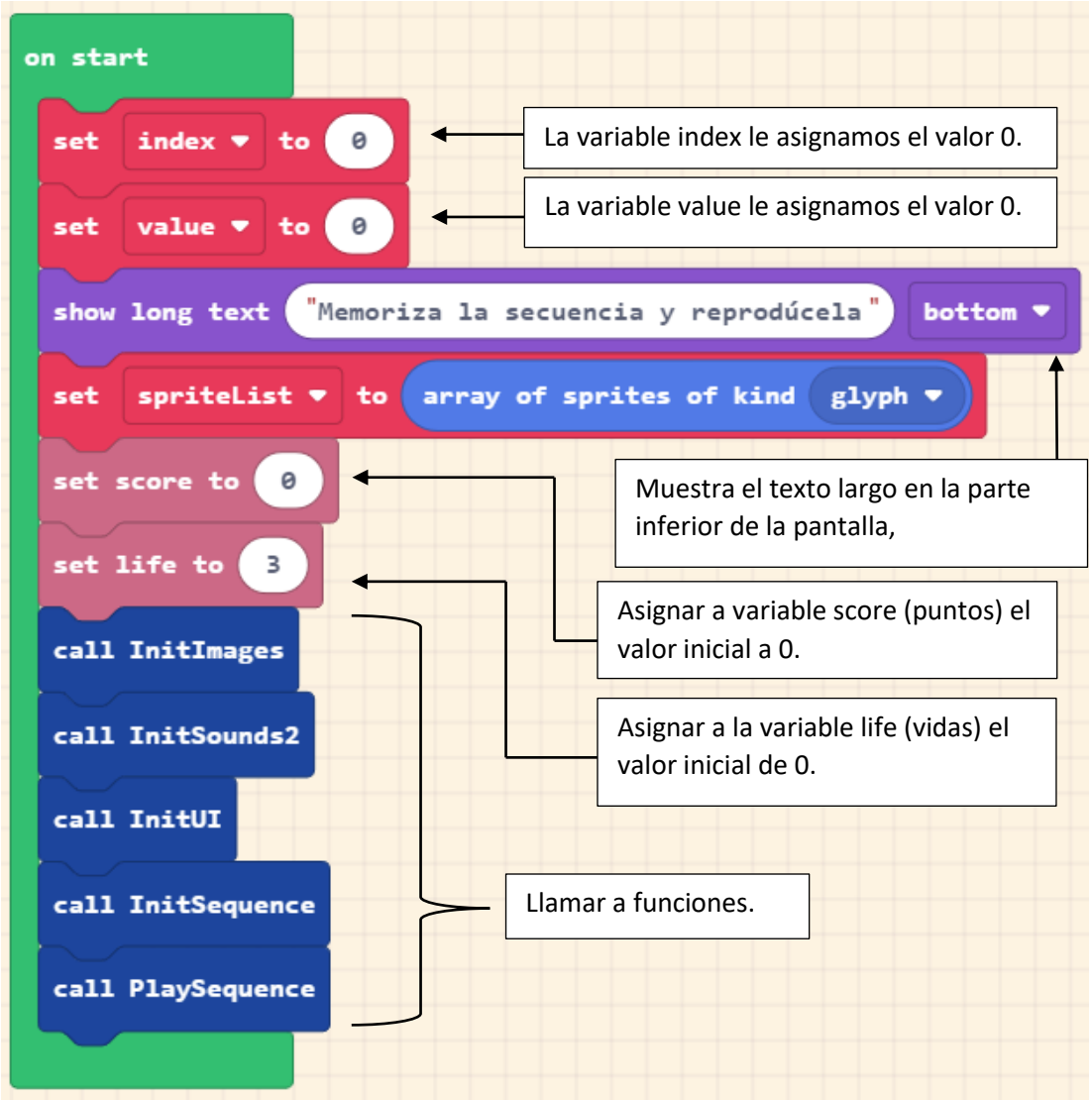
Memory

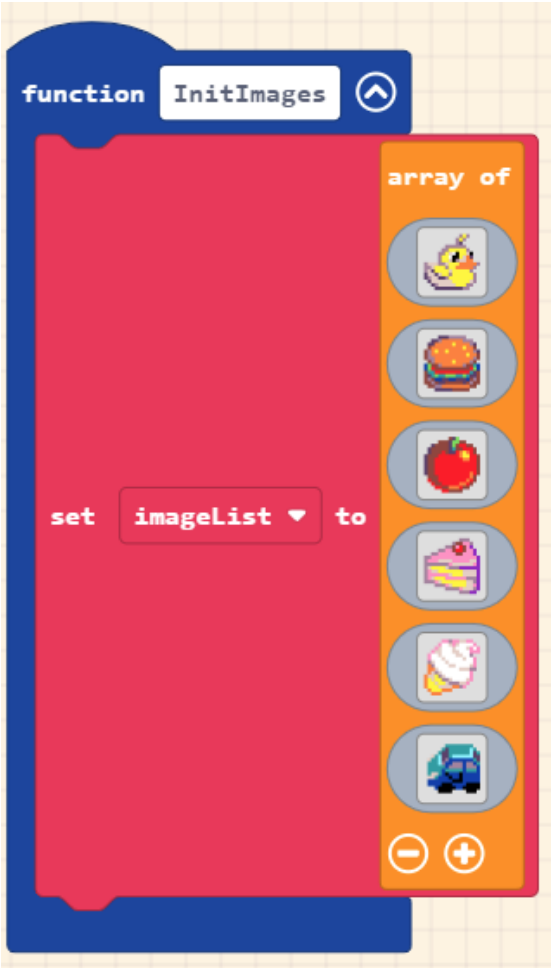


Cuando empieza



InitImages: Almacena en la array imageList las 6 imágenes que mostrará el juego.
InitSounds2: almacena una serie de 6 números que servirá para reproducir sonido.
InitUI: Muestra los 6 elementos del juego en la parte inferior con su respectivo cursor para poderlas seleccionar.

Función InitImages.



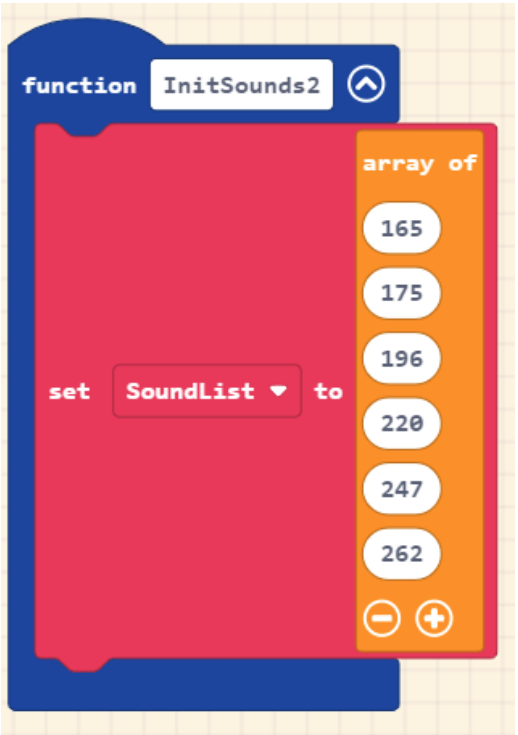
InitSequence: realiza un bucle que se repite tres veces y llama tres veces a la función AddToSequence, esta a su vez almacena al final de la array CodeSequence un valor aleatorio de 0 hasta 5.

PlaySequence: Muestra la nueva serie de objetos a memorizar durante 600 ms.

Definimos una array llamada imageList que almacenará 6 imágenes.

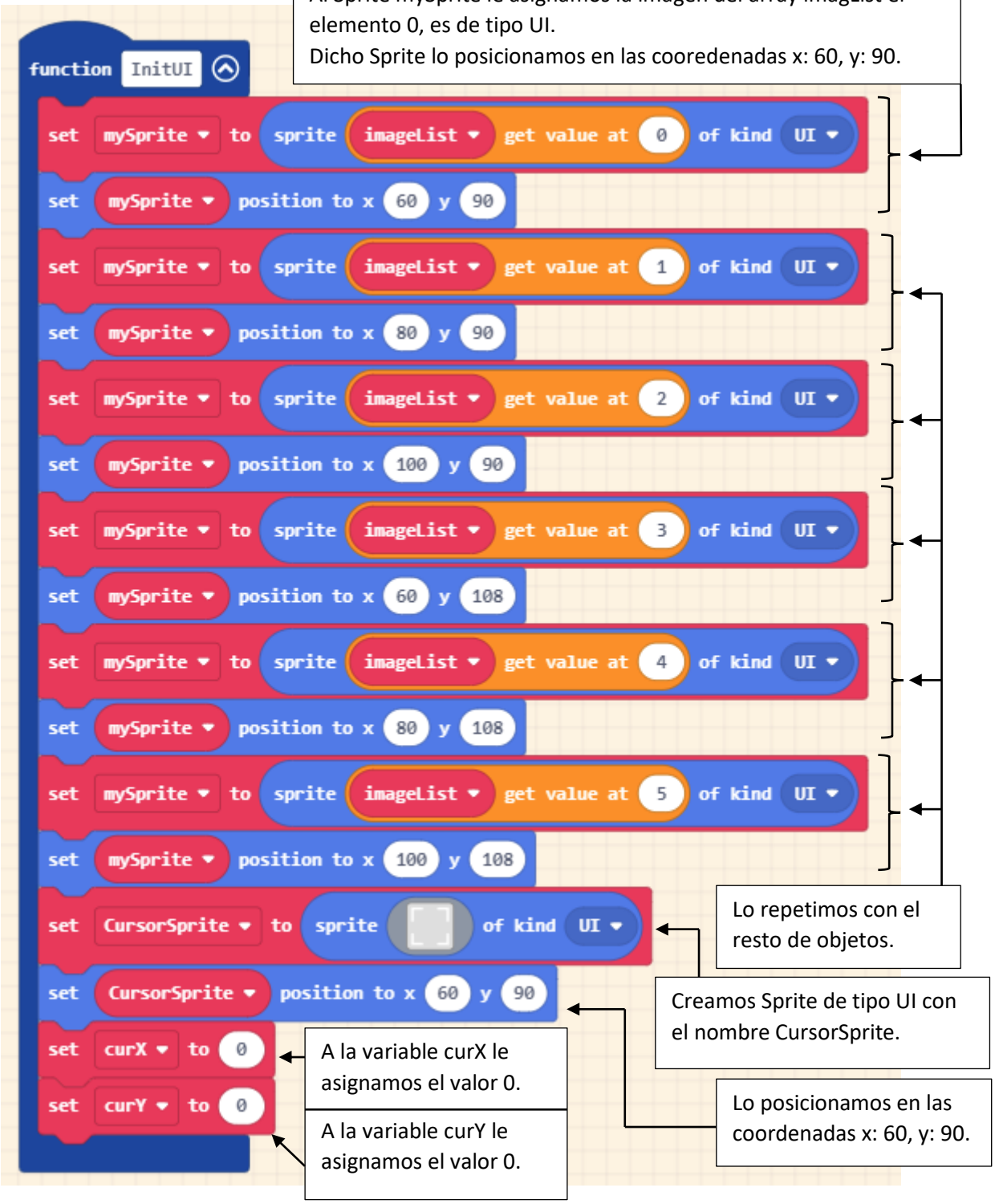
Todas están disponibles en la galería.

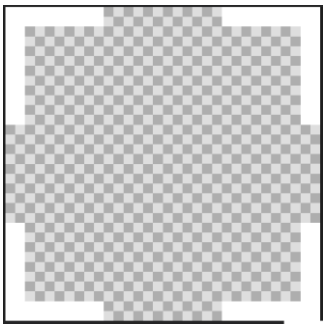
Función InitSound2



Definimos una array llamada SoundList que almacenará 6 valores numéricos que utilizaremos a posteriores para reproducir sonidos.

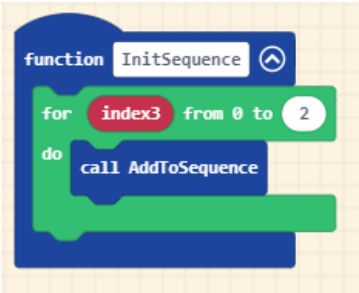
Función InitUI





Con una dimensiones de 16 x 16

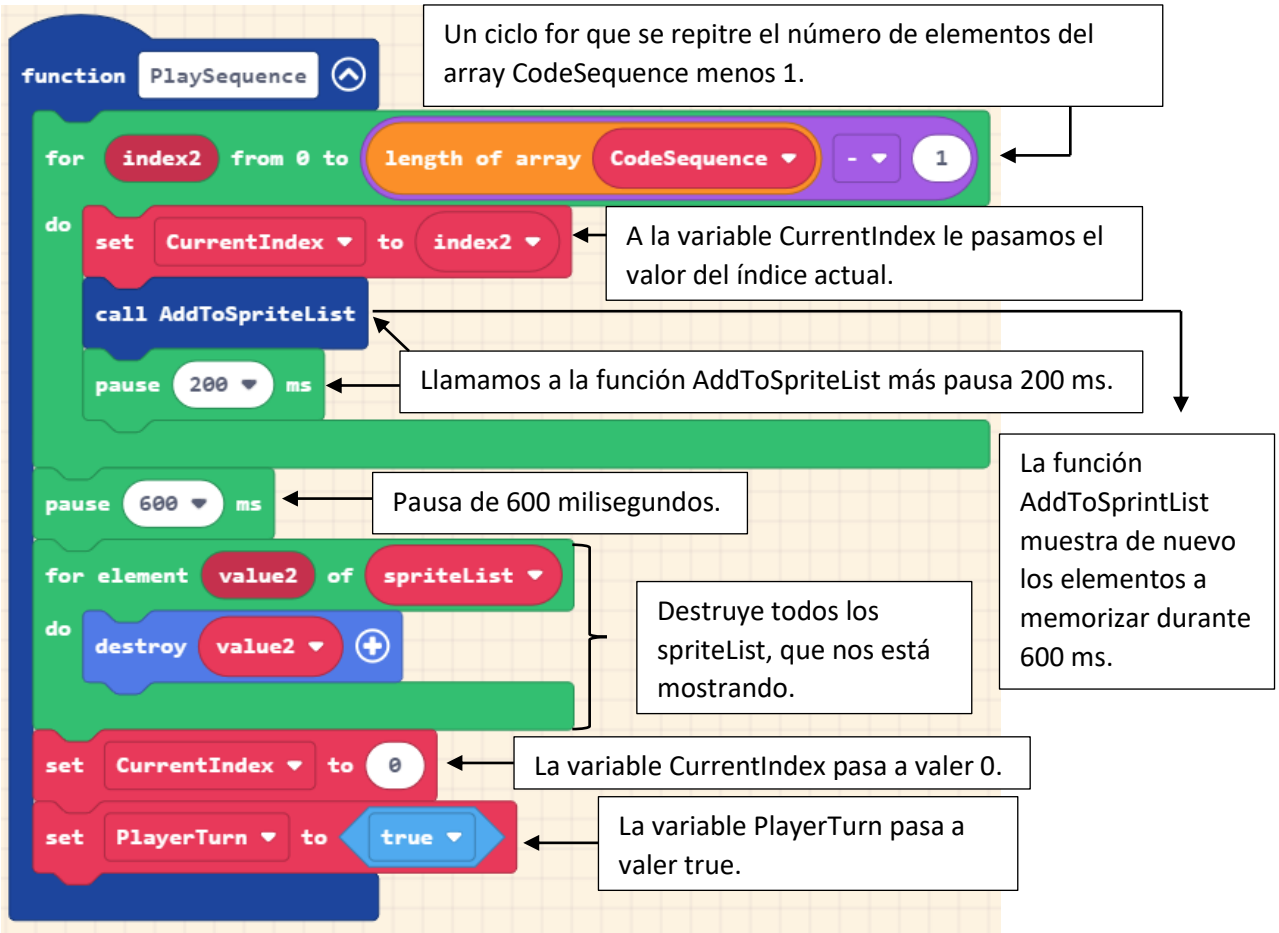
La función InitSequence



Realizamos un bucle que se repite 3 veces, por este motivo llama tres veces a la unción AddToSequence.

Esta función añade al final un valor aleatorio desde 0 hasta 5 a la array CodeSequence.

La función PlaySequence



Un ciclo for que se repitre el número de elementos del array CodeSequence menos 1.

A la variable CurrentIndex le pasamos el valor del índice actual.

Llamamos a la función AddToSpriteList más pausa 200 ms.

La función AddToSprintList muestra de nuevo los elementos a memorizar durante 600 ms.

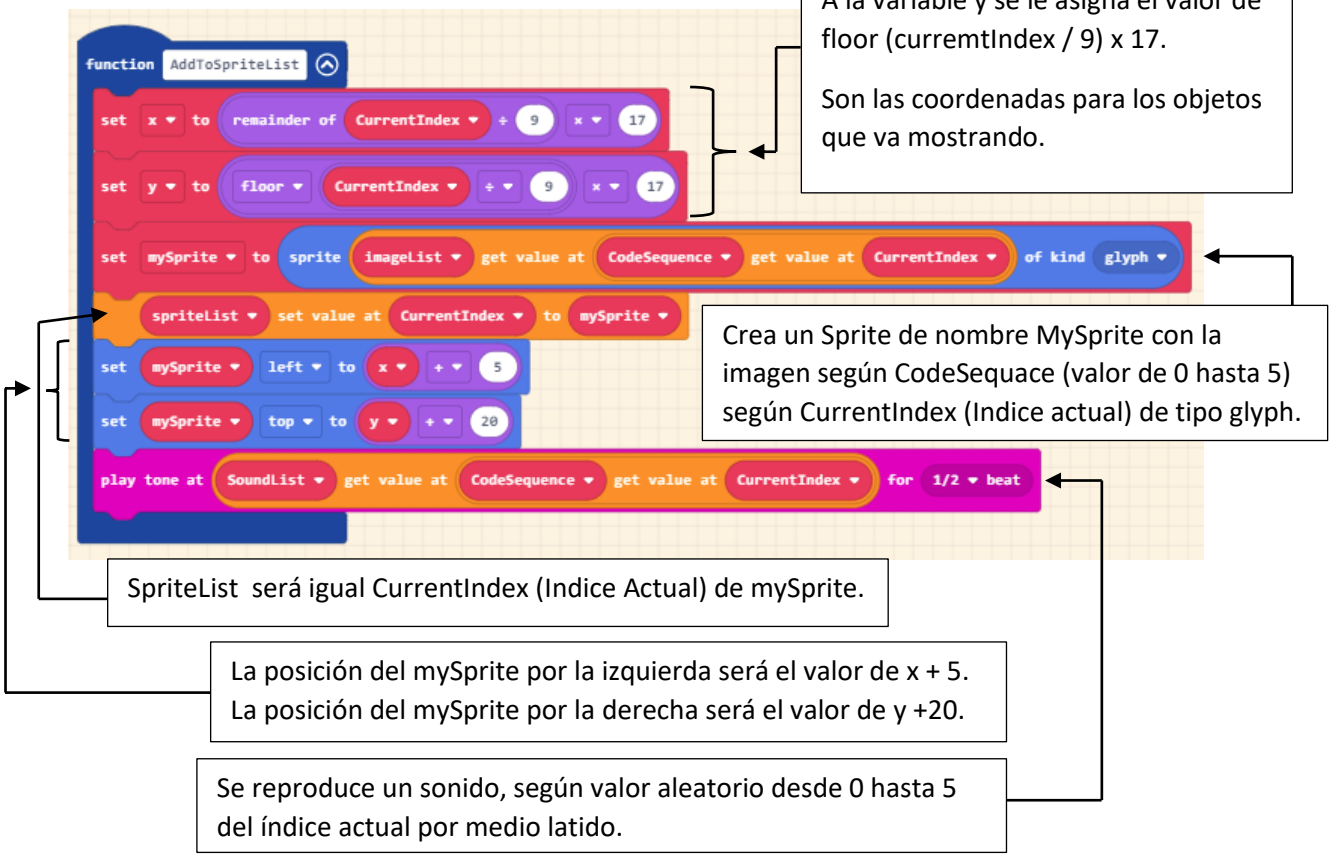
Pausa de 600 milisegundos.

Destruye todos los spriteList, que nos está mostrando.

La variable CurrentIndex pasa a valer 0.

La variable PlayerTurn pasa a valer true.

AddToSprinteList



Remaider: Este es un operador adicional para la división. Puede averiguar cuánto queda si un número no se divide en el otro número de manera uniforme.

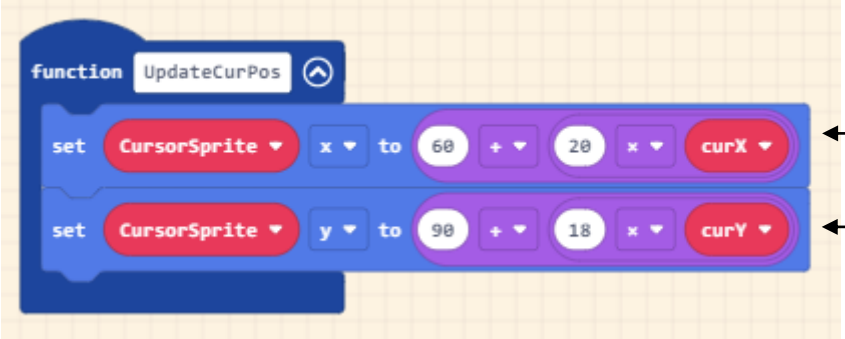
Sabemos que 4/2 es igual a 2 por lo que se divide en 4 uniformemente. Pero 5/2 es igual a 2 pero con un resto de 1. Entonces la operación de resto%2 es igual a 1, del número que queda de una operación de división.

Floor: Para hacer que un número cambie al siguiente número entero más bajo (entero), obtenga el valor mínimo del número. El valor mínimo de 8,76 es 8, ya que es el siguiente número entero más bajo. Para el número negativo de 6,17, su piso es -7 ya que es el siguiente número entero más bajo.

Función AddSequence

La variable CodeSequence se le asigna un valor aleatorio desde 0 hasta 5

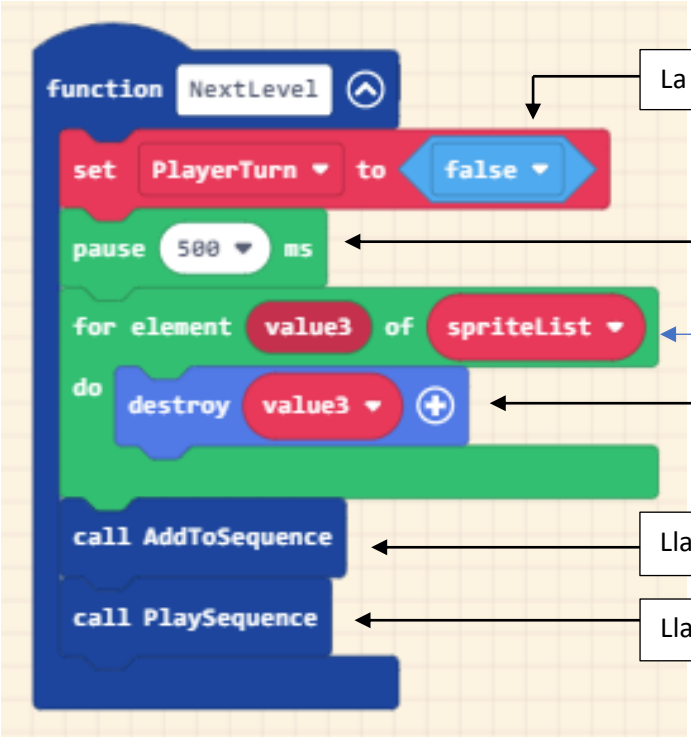
Función UpdateCurPos



A la variable CursorSprite de x se le asigna el valor de (60+20x(valor de crux).

A la variable CursorSprite de y se le asigna el valor de (90+18x(valor de curY)

Función NextLevel



La variable PlayerTurn pasa a valer false.

Una pausa de medio segundo.

Un bucle que se repetirá según el número de elementos de spriteList.

Los destruye a todos los elementos.

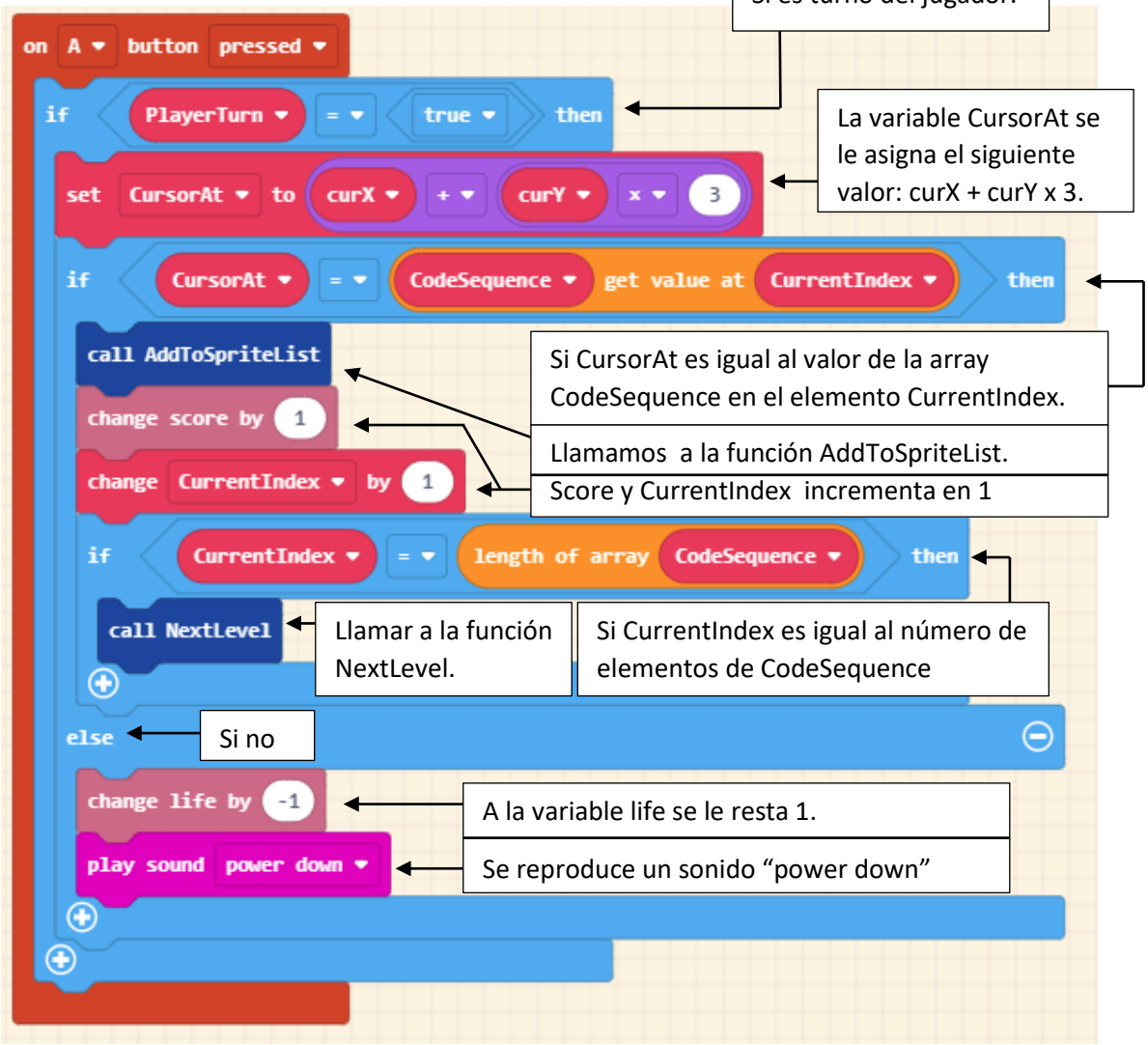
Llamamos a la función AddSequence.

Llamamos a la función PlaySequence.

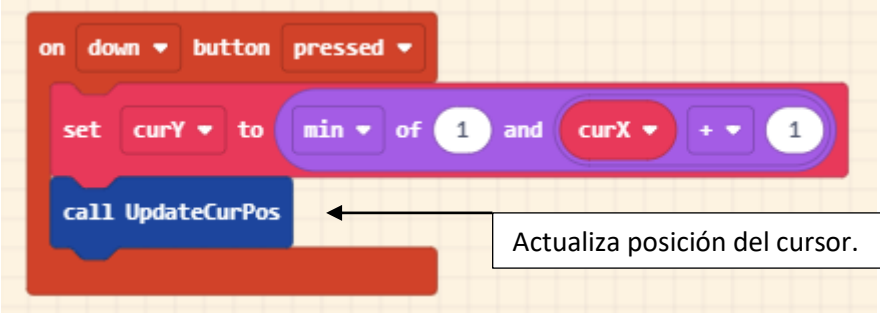
Cuando se crea un Sprite de tipo UI

Sprite en modo fantasma.

Cuando se presiona el botón A



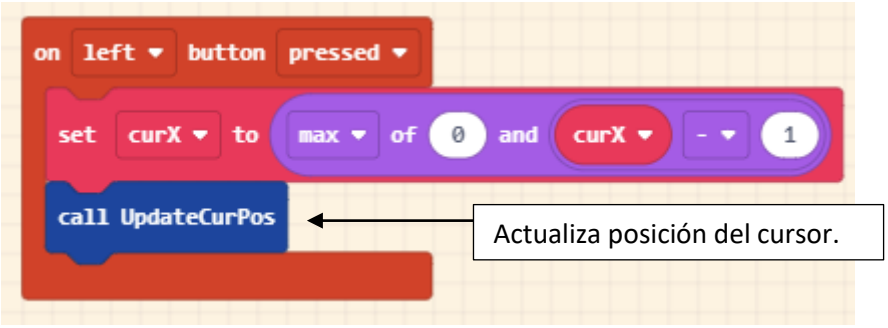
Cuando se presiona el boto de abajo.



La variable curY obtiene el valor mínimo de 1 y curX + 1.

Llama a la función UpdateCurPos.

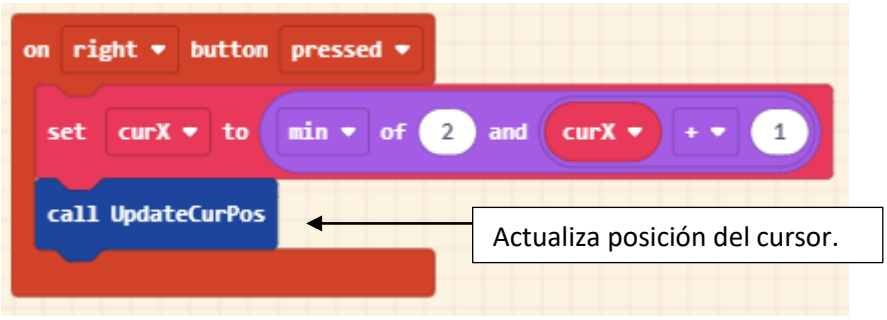
Cuando se presiona el botón Izquierda



La variable curX obtiene el máximo de 0 y curX – 1.

Llama a la función UpdateCurPos.

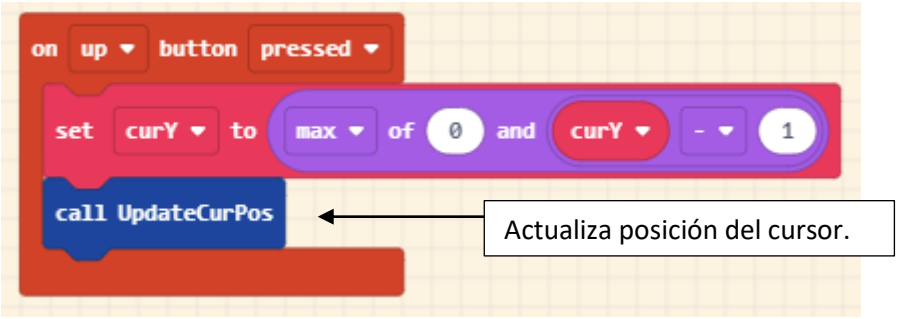
Cuando se presiona el botón derecha.



La variable curX obtiene el mínimo de 0 y curX + 1.

Llama a la función UpdateCurPos.

Cuando se presiona el botón arriba



La variable curY obtiene el máximo de 0 y curY – 1.

Llama a la función UpdateCurPos.

Mínimo y máximo de dos valores: Puede obtener el menor o el mayor de dos números con las funciones min() y max().
El mínimo de 2 y 9: Math.min(2,9) es igual a 2.
El máximo de 3 y 9: Math.max(3,9) es igual a 9.