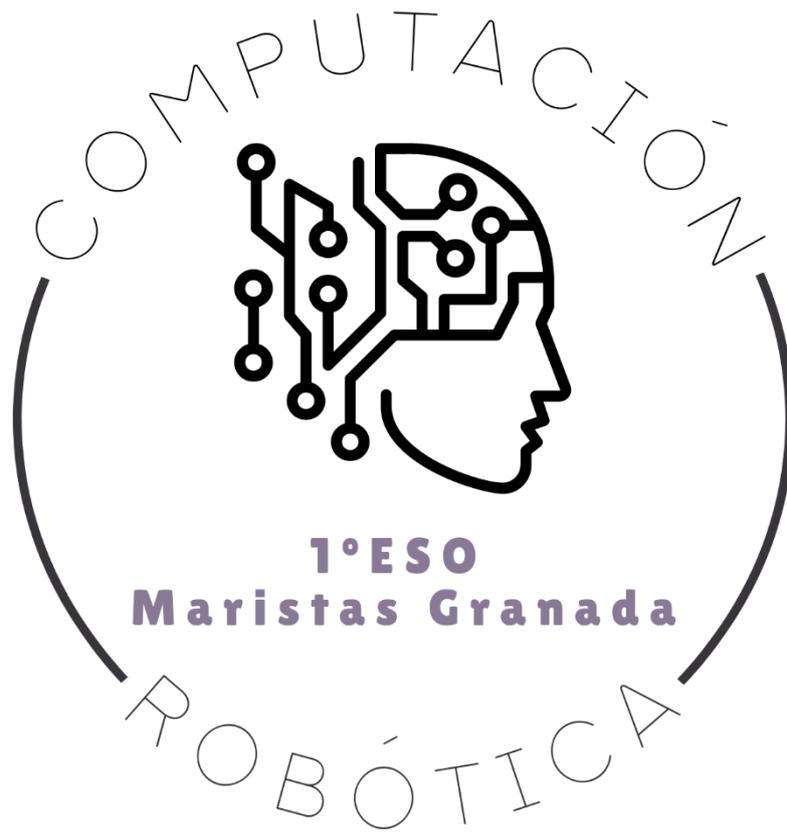


CURSO 2023-2024



RETO 16:
SEGUNDO VIDEOJUEGO
CON SCRATCH: SALTAR

COMPUTACIÓN Y ROBÓTICA 1ºESO

COLEGIO MARISTA LA INMACULADA
CALLE SÓCRATES, 8
18002 - GRANADA

Índice

0. Ubicación en la programación.....	2
1. Descripción del producto final a entregar.....	3
2. Aprendemos practicando	4
2.1 Videojuego paso a paso	4
3. Por si quieres seguir ampliando en casa	15

0. Ubicación en la programación

Título: Reto 16. Segundo videojuego con Scratch: Saltar

Evaluación: Tercera

Temporalidad: 1 sesión (2 horas consecutivas)

Índice de contenidos: ..

Breve resumen de la situación: ..

1. Descripción del producto final a entregar

En este tema

Los retos se califican según los siguientes criterios:

- Están completos

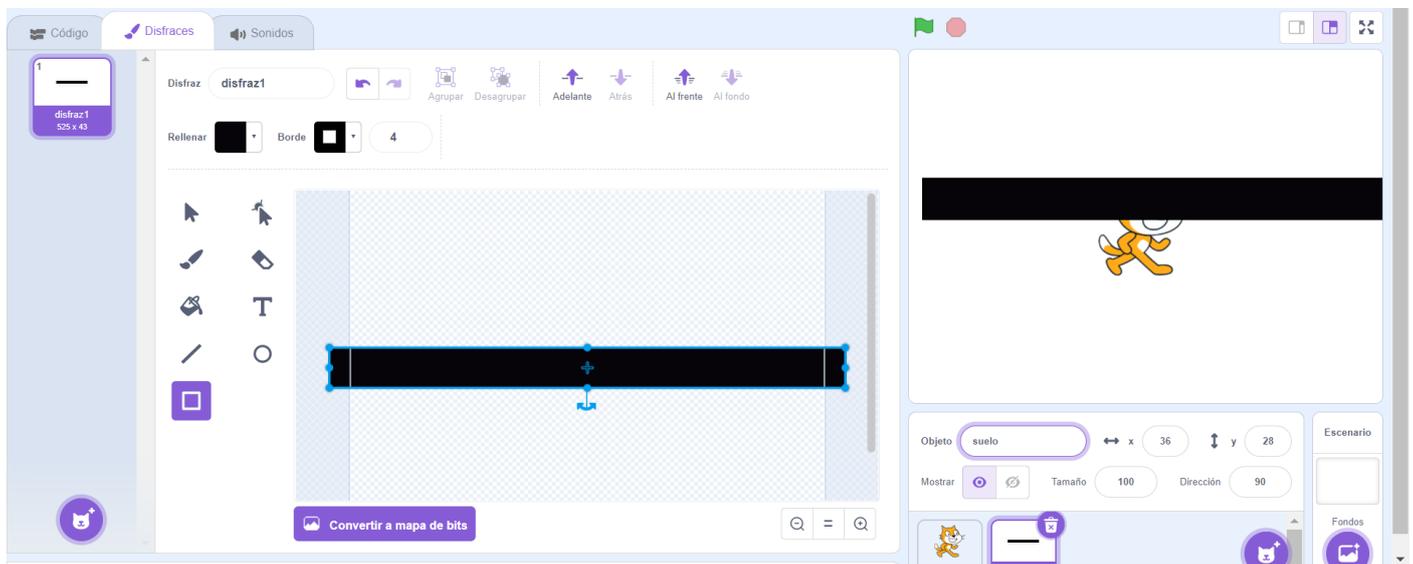
2. Aprendemos practicando

Vamos a simular el famoso “juego del dinosaurio” cuando el navegador Chrome trabaja sin conexión a internet:

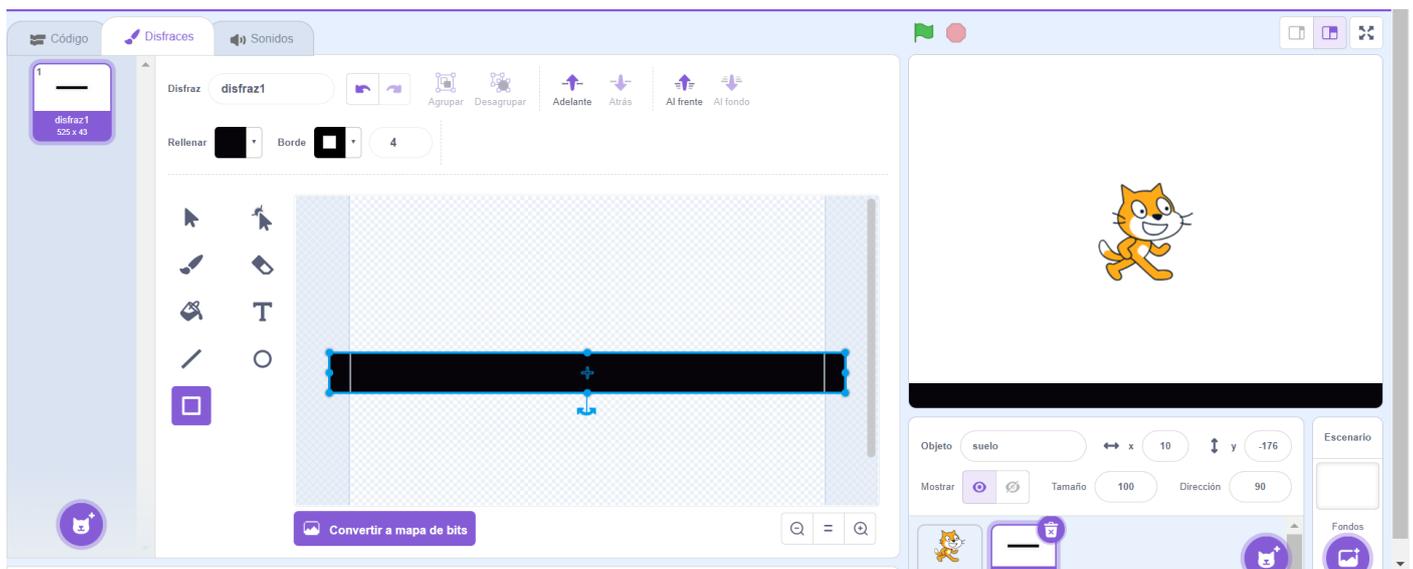
<https://dinorunner.com/es>

2.1 Videojuego paso a paso

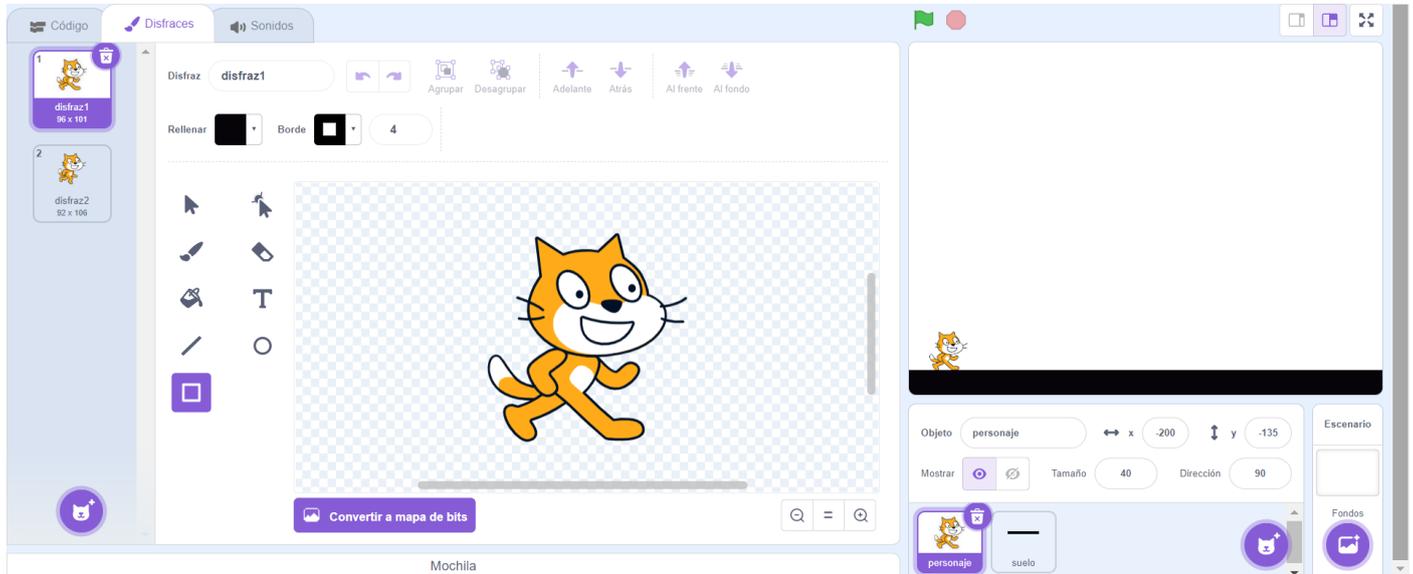
Creamos el suelo. Un objeto formado por un rectángulo negro sobre el que correrá nuestro personaje.



IMPORTANTE. En la zona de edición, ajusta el centro del rectángulo con el centro del objeto. Así la programación del objeto estará siempre vinculada al centro del rectángulo.



Ajustamos el tamaño del personaje principal a 40. Elegimos nuestro gato, por tener dos disfraces que dan sensación de estar el gato en movimiento. Situamos al gato sobre el suelo.

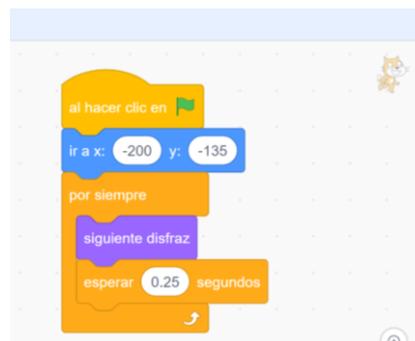


En estas imágenes, he situado al gato en las posiciones $x=-200$, $y=-135$. Estos valores los puede modificar cada creador, según la posición de su suelo. En estos apuntes asumiremos siempre las posiciones (x,y) indicadas.

De esta forma, en la zona de programación, al pulsar bandera verde para comenzar el juego, siempre colocaremos al personaje en esa posición de partida.

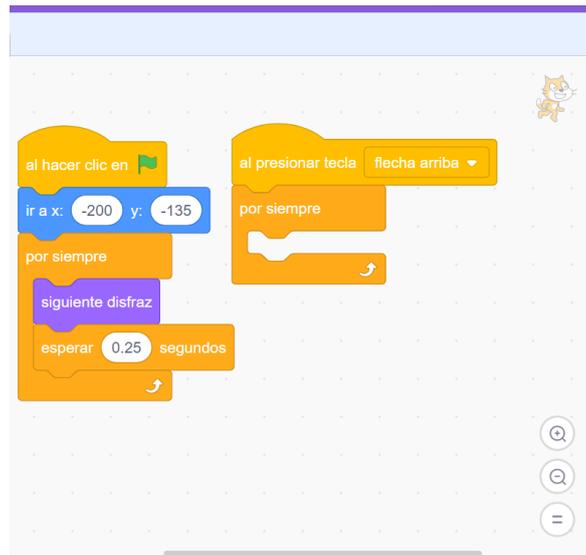


Además, en un bloque "por siempre", alternaremos sus dos disfraces cada 0.25 segundos para imitar la sensación de estar corriendo.

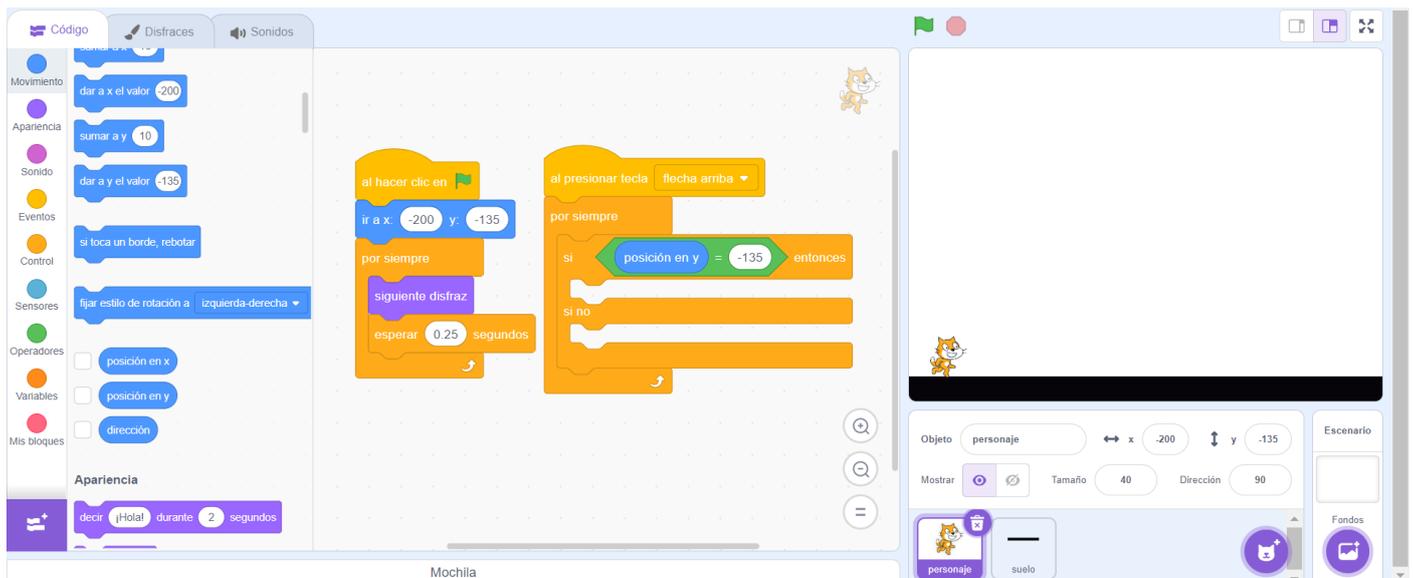


Pasemos al salto. Cuando pulsemos la flecha hacia arriba, el personaje debe saltar. Y debe hacerlo de una manera realista: al principio del salto, avanza muy distancia. Y al final del salto, recorre menos distancia debido a la gravedad que frena el impulso del salto, hasta detener al personaje.

¿Cómo conseguirlo? En primer lugar creamos el evento “al presionar tecla flecha arriba”, junto a otro bloque “por siempre”.



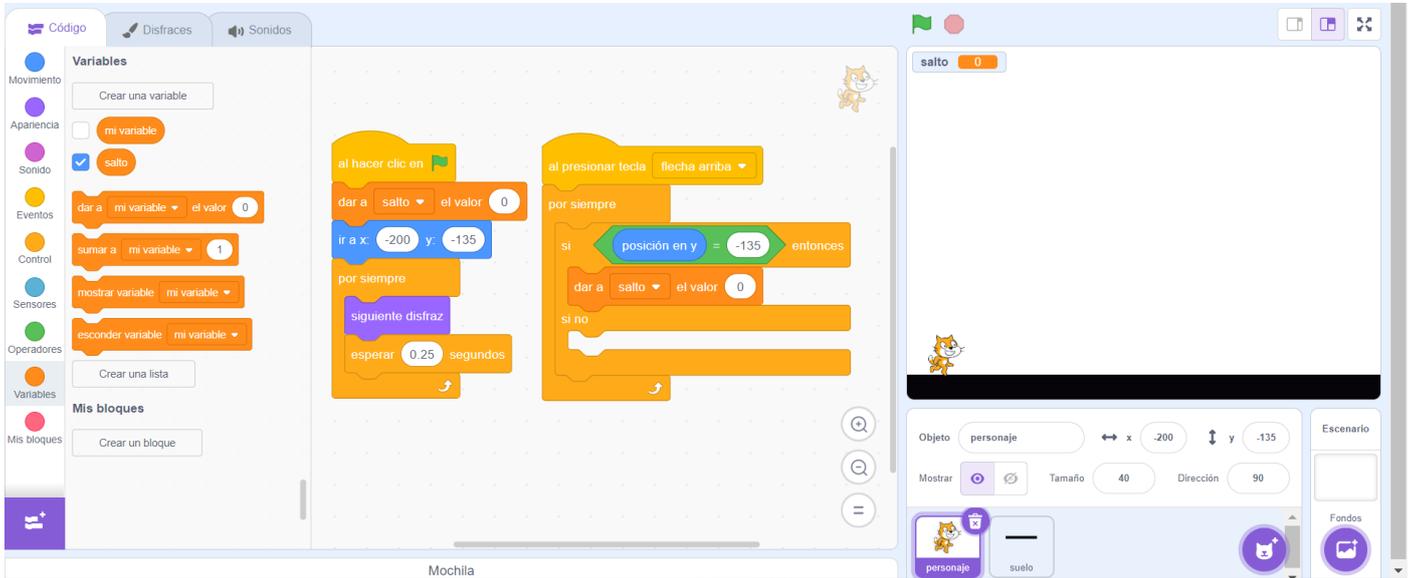
Con un bloque condicional “si... si no” vamos a detectar si el personaje está en la posición inicial sobre el suelo. El bloque “posición en y” nos informa de la coordenada vertical del objeto.



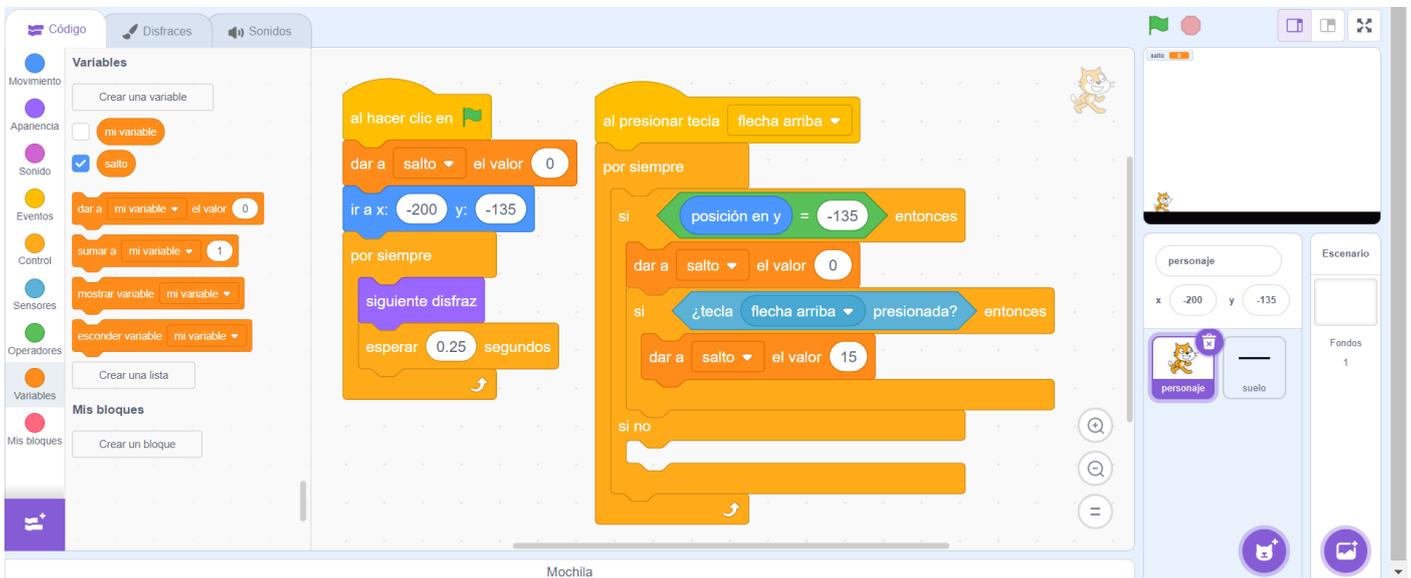
Creamos una variable “salto” para almacenar el valor del impulso vertical del personaje.

Iniciamos el valor de “salto” a 0.

Si el personaje se encuentra en la posición vertical $y=-135$, también forzamos que “salto” sea igual a 0.



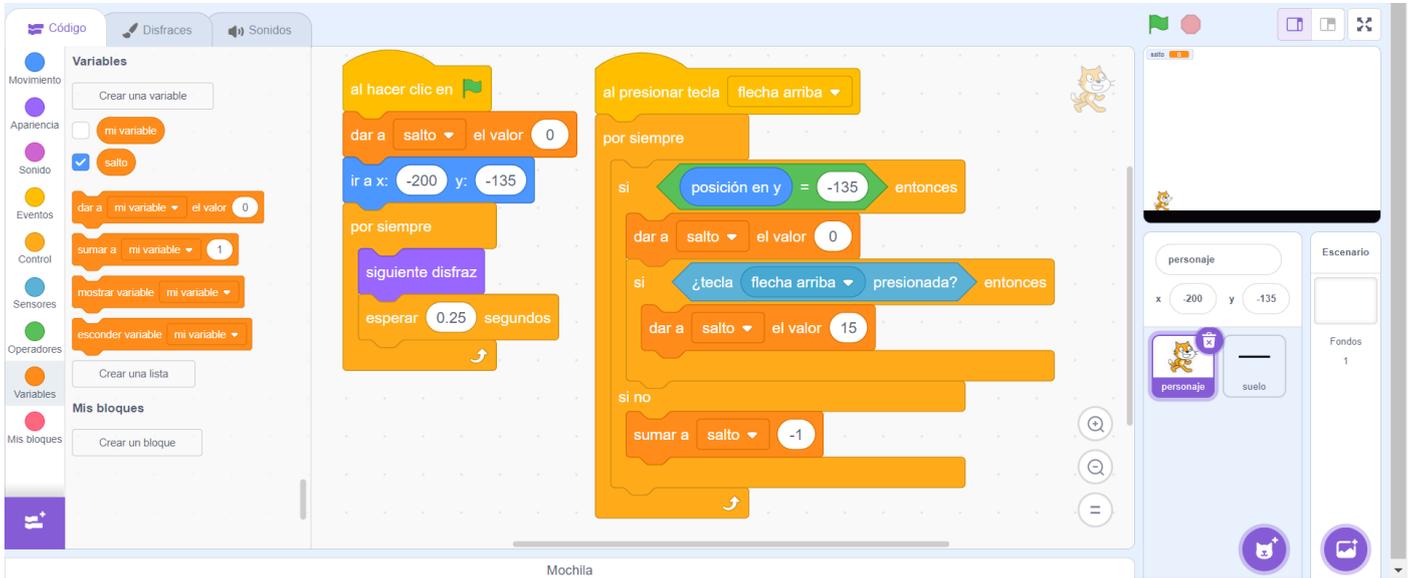
Si hemos detectado que el personaje está sobre el suelo, en la posición vertical de partida, y pulsamos la flecha “hacia arriba”, debemos impulsarlo con una velocidad inicial. Por ejemplo, salto = 15.



Si el personaje no está sobre el suelo, significa que está por el aire saltando.

Para simular el efecto de la gravedad, debemos restar alguna cantidad fija al impulso inicial marcado por la variable salto. Por ejemplo, restar 1 a la variables salto en el bloque “si no”.

Cuando mayor sea el valor inicial de salto, más alto llegará el personaje. Y cuando más cantidad restemos en el bloque “si no”; antes regresará al suelo.

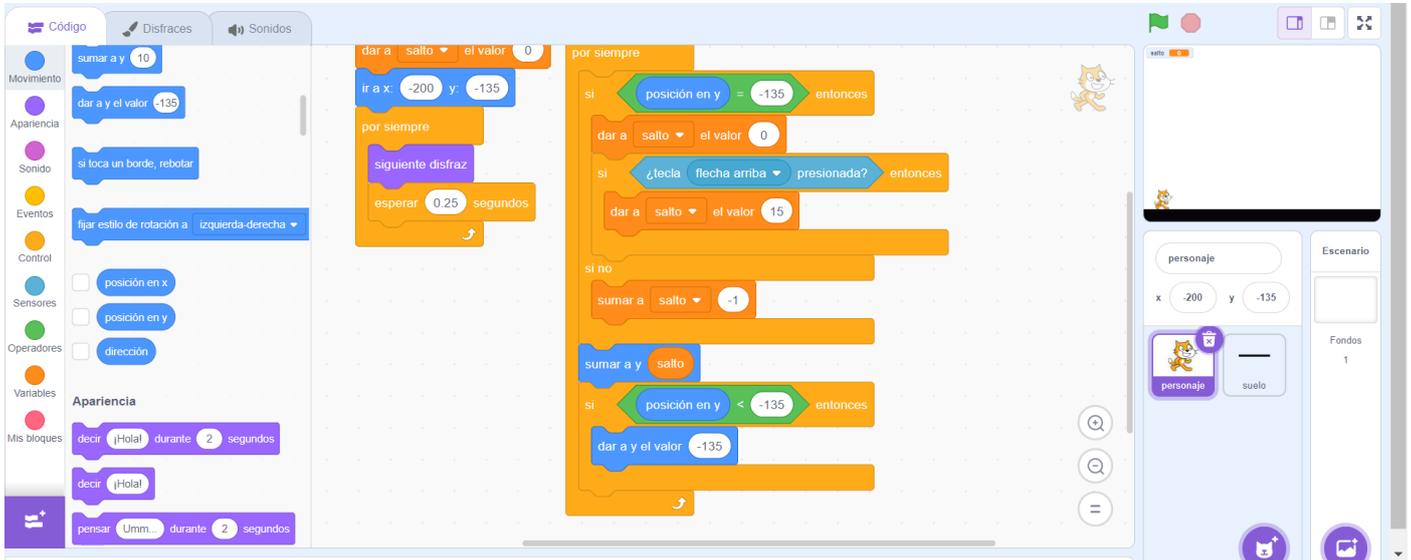


Nuestro personaje no se mueve porque solo hemos modificado el valor de la variable salto. Necesitamos que esta variable afecte a la coordenada vertical del personaje. Usamos el bloque “sumar a y”. Y con esto, nuestro personaje salta de una forma bastante realista.



Nuestro código para saltar tiene un error sutil. Puede ocurrir, por el tiempo de procesamiento del ordenador al ejecutar los bloques de programación, que nuestro gato no vuelva a la posición de partida $y=-135$ tras terminar su salto. Acabando un poco por debajo de la línea del suelo.

Para solventar este error, con un nuevo condicional, detectamos si la posición vertical es inferior al valor $y=-135$ y situamos al personaje en la posición deseada con un bloque “dar a y el valor”.

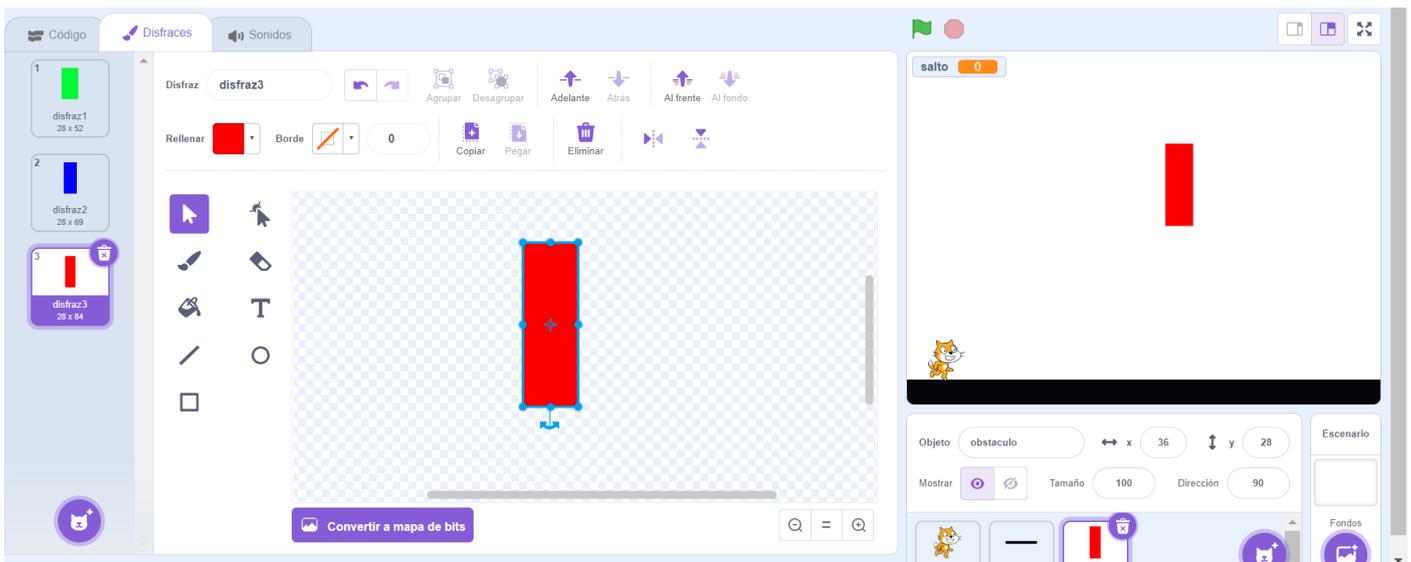


Pasemos a programar los obstáculos a saltar.

Creamos un nuevo objeto llamado “obstaculo”. Recuerda que, por norma general, no ponemos tildes a los nombre de objetos ni de variables.

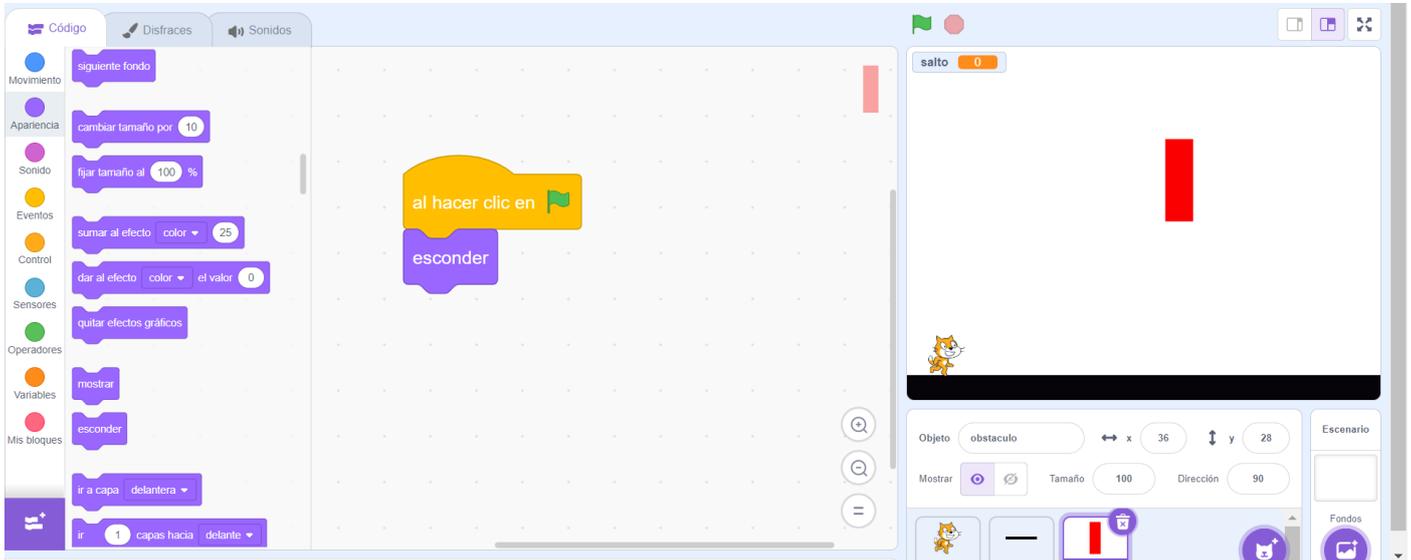
Este objeto va a tener tres disfraces: tres rectángulos de distinta altura, que haremos aparecer de manera aleatoria durante el juego. Recuerda hacer coincidir el centro de cada rectángulo con el centro del objeto.

Por ejemplo, el rectángulo verde es más bajo que el azul. Y el rectángulo azul es más bajo que el rojo.



Al pulsar bandera verde, debemos esconder el objeto “obstaculo”.

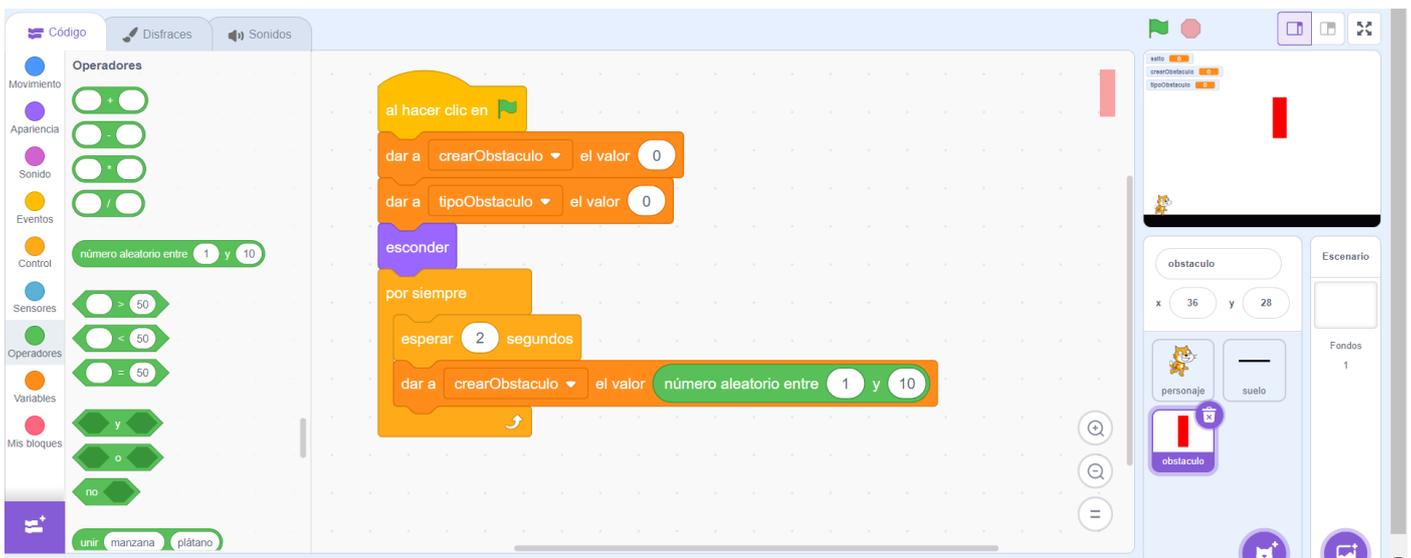
Y posteriormente, de manera aleatoria, iremos mostrando uno de los tres rectángulos cada vez que creamos un clon del objeto.



En un bloque “por siempre”, cada 2 segundos, creamos un objeto “obstaculo”. Podemos modificar este tiempo al gusto del creador.

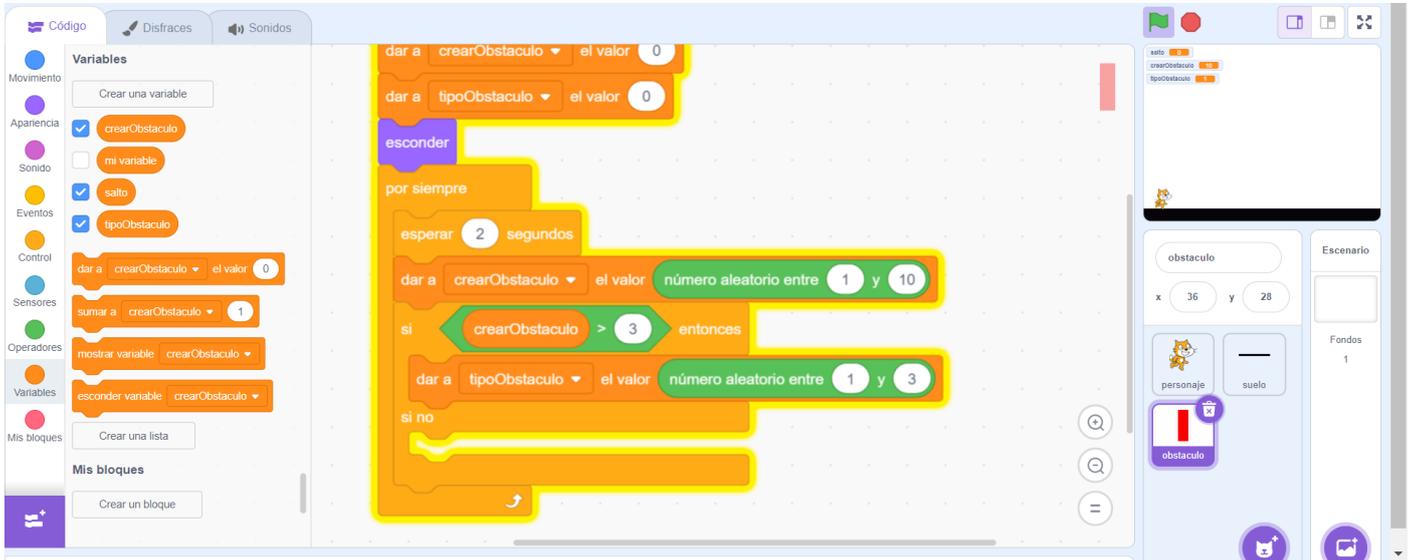
Creamos una variable “crearObstaculo” para saber si se muestra el rectángulo o no con ayuda de un bloque “número aleatorio”.

Y creamos una variable “tipoObstaculo” para saber si mostramos el rectángulo verde, azul o rojo, con ayuda de otro bloque “número aleatorio”.



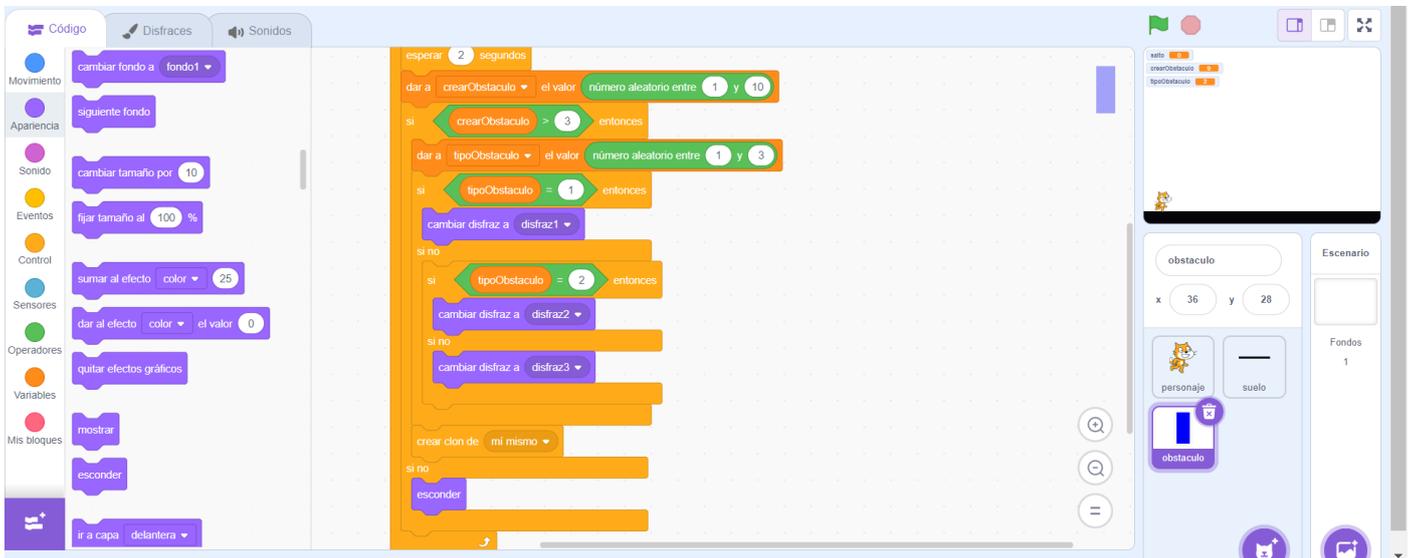
Por ejemplo: si “crearObstaculo” es mayor que 3, creamos el obstáculo. Esta probabilidad la podemos modificar al gusto del creador del programa.

Si hemos creado el obstáculo, deberemos elegir con un nuevo número aleatorio el tipo de obstáculo (verde sería el disfraz1, azul el disfraz2, y rojo sería el disfraz3).



Con nuevos bloques condicionales, seleccionamos un disfraz del objeto “obstáculo”. Y al final creamos un clon de ese objeto.

No olvidar esconder el objeto, al final de todo el bloque de programación, para evitar que quede visible en caso de que la probabilidad del número aleatorio indique que el objeto no puede mostrarse en pantalla.



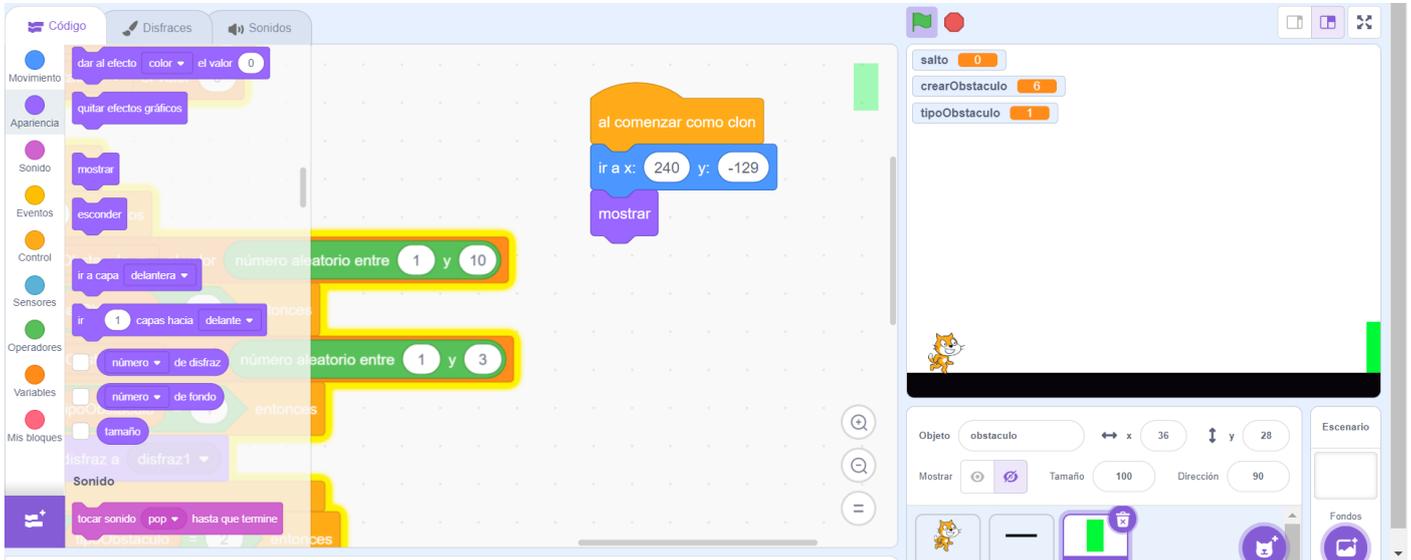
¿Qué hace el clon de objeto “obstaculo”?

Aparecer por la derecha, a la altura del suelo, y avanzar hacia el personaje.

La posición (x,y) del obstáculo le elegimos libremente, para situar a los rectángulos sobre el suelo.

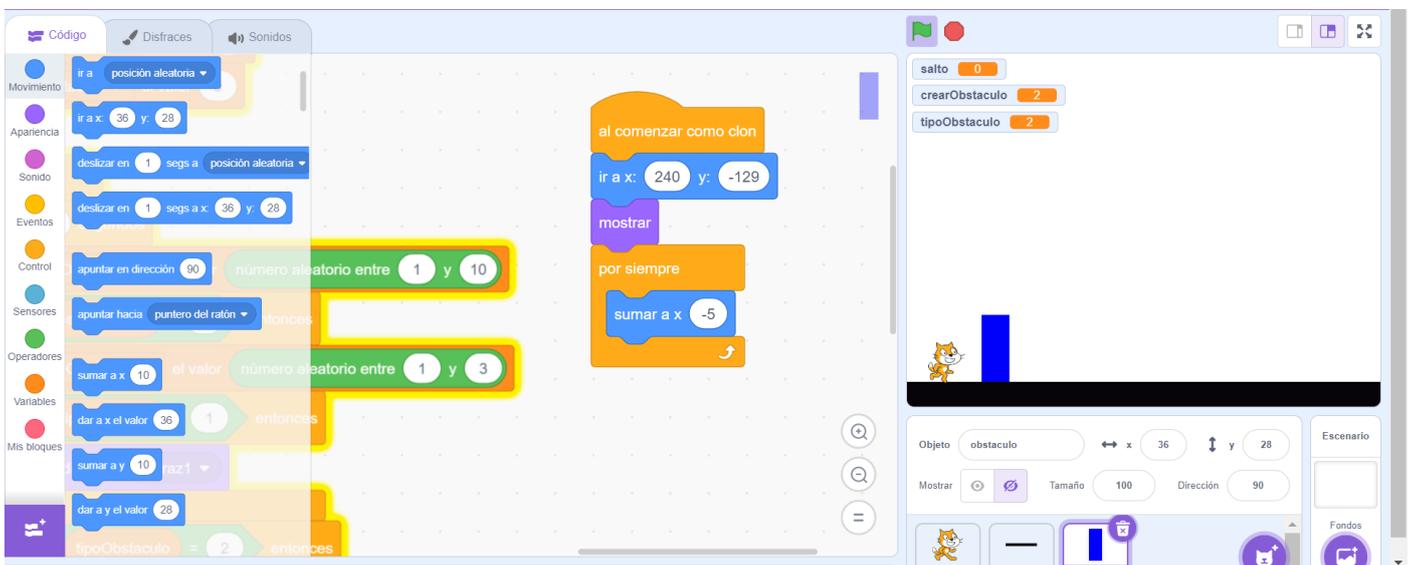
En estos apuntes hemos elegido $x=240$, $y=-139$.

No olvidemos el bloque “mostrar” si hemos iniciado el clon.



Un nuevo bloque por siempre va a controlar el movimiento de los rectángulos.

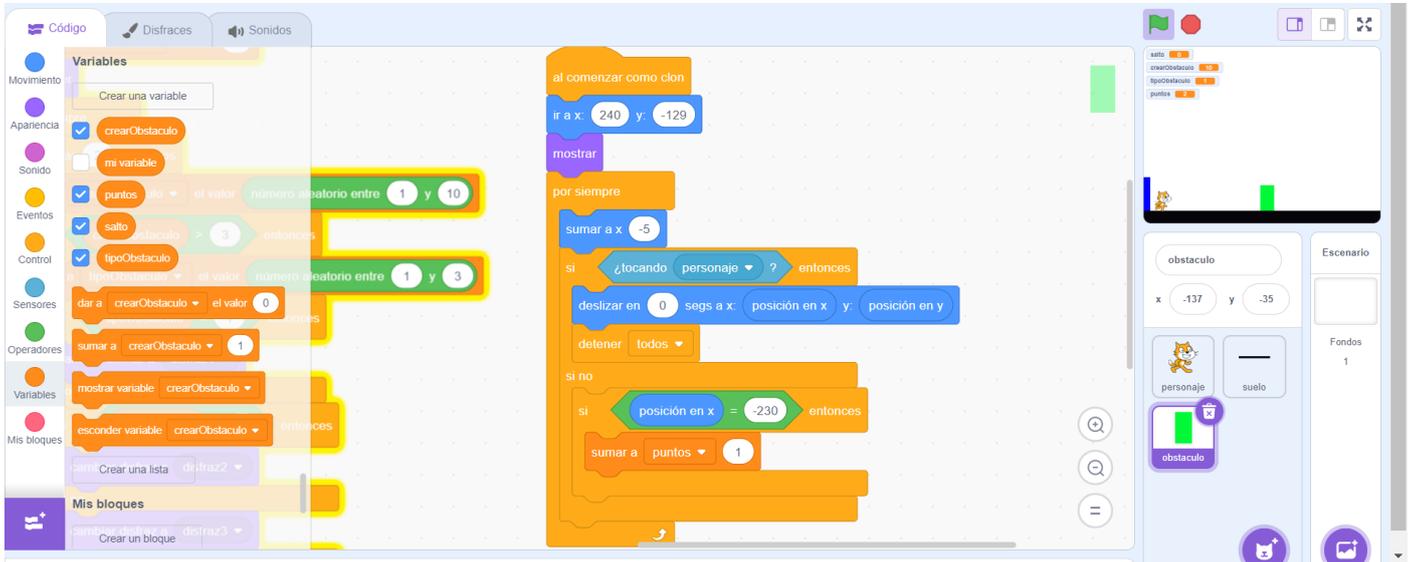
Con el bloque “sumar a x” podemos ajustar la velocidad a la que se desplazan los rectángulos.



Ya hemos conseguido que los rectángulos, aleatoriamente cada 2 segundos y aleatoriamente entre sus tres disfraces, aparezcan por la derecha y se desplacen por la izquierda.

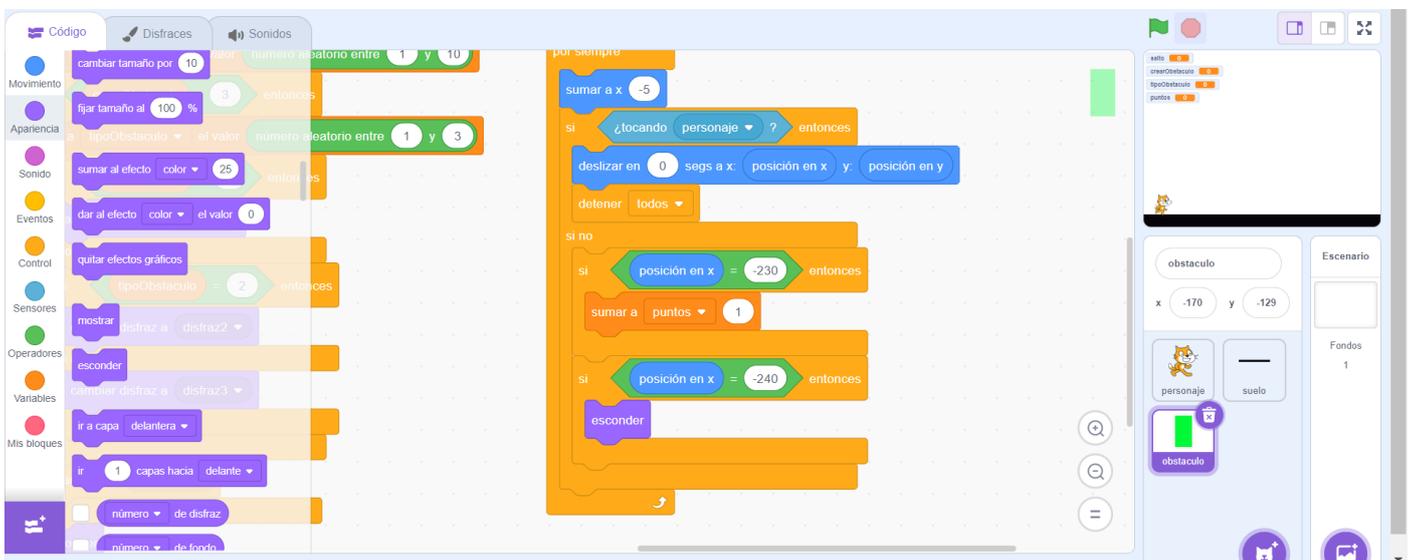
Ahora debemos conseguir, en primer lugar, si el obstáculo toca al personaje.

Para ello, usamos un condicional “si... si no”. Y empleamos el bloque de sensores “¿tocando personaje?”.



Finalmente, fíjate que los obstáculos quedan almacenados por detrás del personaje.

Si llegan al final, deben desaparecer. Por lo que podemos detectar su posición para ocultarlos.



3. Por si quieres seguir ampliando en casa

Podemos mejorar nuestro videojuego de salto con las siguientes opciones (de menor a mayor dificultad):

1. Introduce una música que acompañe al salto del personaje.
2. Introduce una melodía de “game over”, cuando el personaje impacte con un obstáculo.
3. Muestra un fondo de “game over “ cuando el juego finalice.
4. Crea otro tipo de obstáculos, jugando ahora con la anchura en vez de con la altura.
5. Introduce objetos voladores que, si son cazados por el personaje en su salto, generan 5 puntos extras en la puntuación.

Es posible que, para los pasos 4 y 5, debas ajustar los parámetros de salto y gravedad del código de programación.

¡Ánimo!