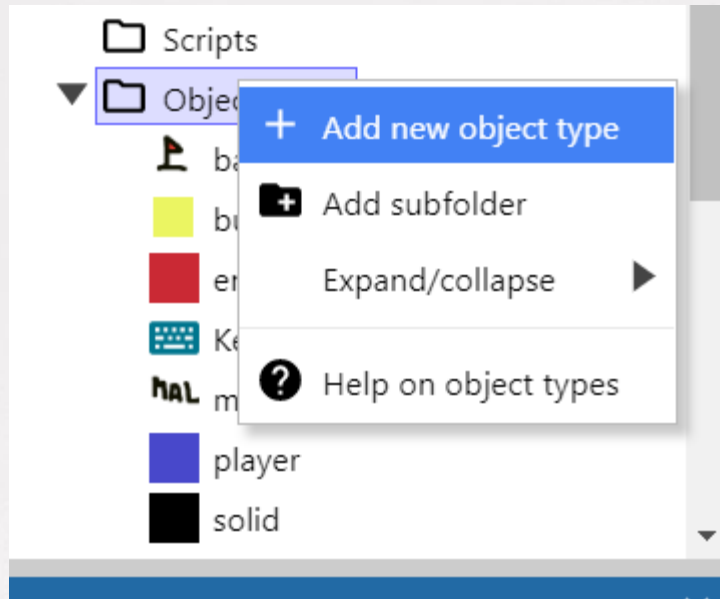


# El prototipo

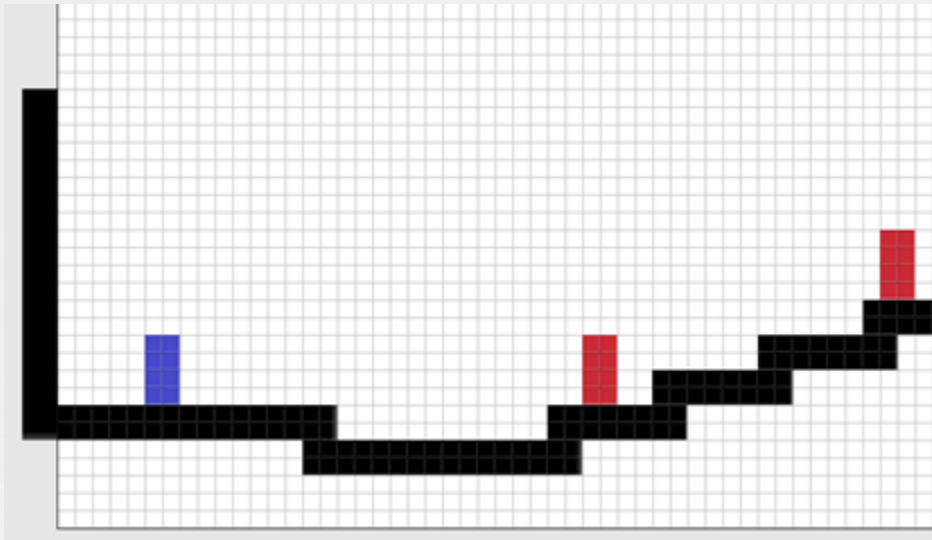
Empezamos planificando el proyecto, pensando los elementos que necesitaremos (mecánicas, “assets”, etc) y cómo vamos a organizar el proyecto. Un videojuego es una producción compleja y larga, el objetivo del curso es hacer algo en unas 2 horas, un tiempo extremadamente limitado. Es por eso que vamos a hacer un juego muy simple, un “plataformas” donde podemos disparar, hay enemigos que nos persiguen y tenemos que alcanzar una bandera.

Nos centraremos en los apartados técnicos de cómo producir el juego y que funcione, este curso deja de lado tareas como el diseño de niveles o balanceado (conceptos esenciales para que nuestro juego sea divertido).



Hacemos click derecho en “Object types” (dentro de la ventana de proyecto) y añadimos tres “sprites” (imágenes 2D). Este será el objeto que más usaremos.

Llamaremos a uno “player”, a otro “enemy” y a otro “solid”. Es importante dar nombres claros y mantener nuestro proyecto ordenado. Debemos evitar caracteres como la “ñ” o los acentos.



“Solid” será el suelo donde se apoyen los personajes.  
“Player” será el protagonista y “enemy” los enemigos. Le damos a cada uno un color diferente para distinguirlos y los colocamos en el “layout”.

Properties

Layout

Name

Layout 1

Event sheet

Event sheet 1

Size

▶ 854 x 480

Unbounded scrolling

☐

Effects

Add / edit

[Effects](#)

Editor

Margins

▶ 1000 x 1000

Show grid

☒

Snap to grid

☒

Grid size

▶ 16 x 16

Show Collision Polygons

☐

Show Translucent Inactive Layers

☐

Project properties

[View](#)










More information

[Help](#)

Si hacemos click en nuestro “layout” podemos activar “Show grid” y “Snap to grid”. Si ponemos 16x16 en “Grid size” nos ayudará a colocar los elementos de forma más organizada en el layout.

Behaviors	
Platform	
Max speed	330
Acceleration	1500
Deceleration	1500
Jump strength	500
Gravity	1500
Max fall speed	1000
Double-jump	<input type="checkbox"/>
Jump sustain	0
Default controls	<input type="checkbox"/>
Enabled	<input checked="" type="checkbox"/>
Add / edit	<a href="#">Behaviors</a>

Tras añadir el “platform behavior” a nuestro objeto “player”, podemos hacer click sobre él y desactivar la casilla de “Default controls” en el inspector. Esto hará que ya no se controle con las flechas de dirección.

 Keyboard	<b>D</b> is down	 player	Simulate  Platform pressing Right	Add...
 Keyboard	<b>A</b> is down	 player	Simulate  Platform pressing Left	Add...
 Keyboard	<b>W</b> is down	 player	Simulate  Platform pressing Jump	Add...

Añadimos el objeto “keyboard” a nuestro proyecto y vamos a la “event sheet” para añadir tres eventos que nos permitan controlar a nuestro personaje con las teclas W, A y D.

Properties

Object type properties

Name

bullet

Global

☐

Plugin

Sprite

Common

Position

▶ 320, -16

Size

▶ 16 x 16

Angle

0°

Opacity

100%

Color

Layer

fondo ▼

Z elevation

0

Z index

0 of 1

UID

11

Instance variables

Add / edit

[Instance variables](#)

Behaviors

Bullet

Speed

400

Acceleration

0

Gravity

0

Bounce off solids

☐

Set angle

☒

Step

☐

Enabled

☒

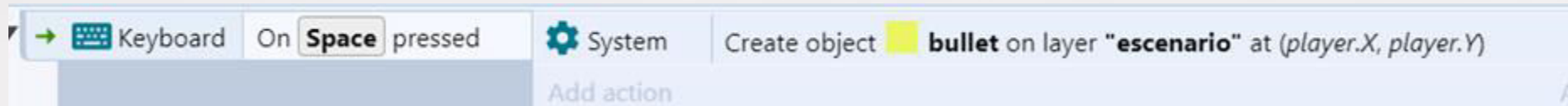
Add / edit

[Behaviors](#)

Menu





Añadimos un nuevo “sprite” al proyecto y lo llamamos “bullet”. Le añadimos el “behavior bullet” para que se comporte como una bala y lo dejamos en nuestro “layout”.





Hacemos un evento que, al presionar la tecla espacio, cree una bala sobre el jugador. Si le damos al botón de “play” para probar el juego, veremos que no podemos disparar en las dos direcciones.

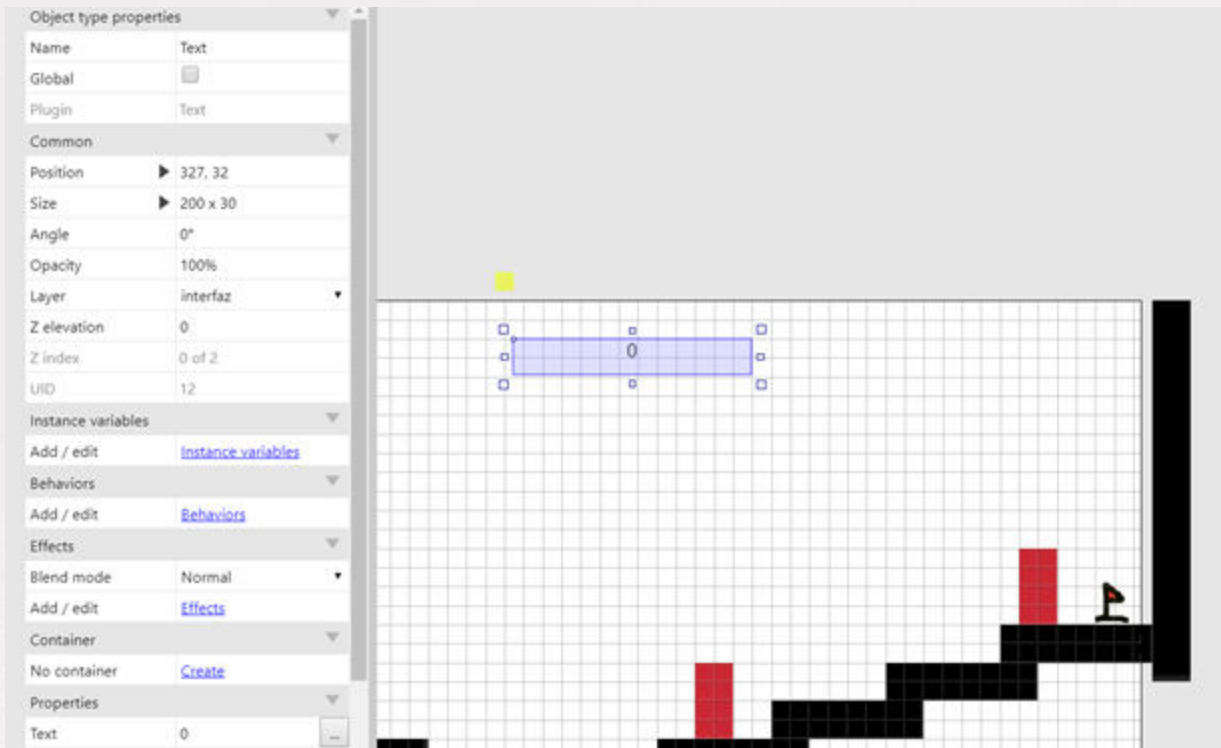


→  bullet	On collision with  <b>enemy</b>	 enemy	Destroy
		 bullet	Destroy
		Add action <span>Add..</span>	

Añadimos un evento que hace que nuestra bala destruya a cualquier enemigo con el que colisione.

bullet	X > LayoutWidth	bullet	Destroy
bullet	OR X < 0	Add action	Add...

También añadimos un evento que destruye las balas cuando salen de los márgenes de nuestro “layout”. Esto evita que se vayan acumulando con el tiempo hasta que causen problemas de rendimiento.



Podemos añadir también un objeto de texto al proyecto y arrastrarlo a nuestro layout para que muestre en pantalla cuantos enemigos hemos destruido.

→ bullet	On collision with enemy	player	Add 1 to puntuacion
		enemy	Destroy
		bullet	Destroy
		Text	Set text to <i>player.puntuacion</i>

Para que el texto vaya cambiando en pantalla faltaría añadir dos nuevas acciones a este evento que habíamos añadido antes.

Creamos una nueva variable en “player” (esta vez de tipo numérico) y hacemos que cada vez que una bala colisione con un enemigo se aumente en 1 el valor de esta nueva variable y que el objeto texto cambie para reflejar la variable (a la que referenciamos como *player.puntuacion*).

Add action

Add...

## ▼ Controlar al personaje

Keyboard

D is down

player

Simulate  Platform pressing Right

player

Set **mover\_derecha** to *True*

Add action

Add...

Keyboard

A is down

player

Simulate  Platform pressing Left

player

Set **mover\_derecha** to *False*

Add action

Add...

Keyboard

W is down

player

Simulate  Platform pressing Jump

Add action

Add...

Add event to 'Controlar al personaje'

Add to 'Controlar al personaje'...


## ▼ IA Enemigo

System

For each













enemy

enemy

Simulate  Platform pressing Left

Es recomendable crear grupos dentro de la “event sheet” (se crean haciendo click derecho sobre esta interfaz) para mantener nuestros eventos ordenados.

## ▼ IA Enemigo

 System  player  enemy	For each  enemy	 enemy	Simulate  Platform pressing Left	
	X < enemy.X	Add action		Add...
	Is <b>perseguir</b>			
 System  player  enemy	For each  enemy	 enemy	Simulate  Platform pressing Right	
	X > enemy.X	Add action		Add...
	Is <b>perseguir</b>			

Para añadir una inteligencia artificial a los enemigos copiamos los eventos de movimiento del jugador pero, en lugar de tener la condición de pulsar una tecla concreta, los sustituimos por comparaciones de la coordenada X del “player” y el “enemy”. Así el enemigo se moverá en la dirección en la que esté el jugador.

Es importante añadir un “for each enemy” a cada evento para garantizar que hace la comprobación para cada entidad del objeto “enemy”.









## ▼ IA Enemigo

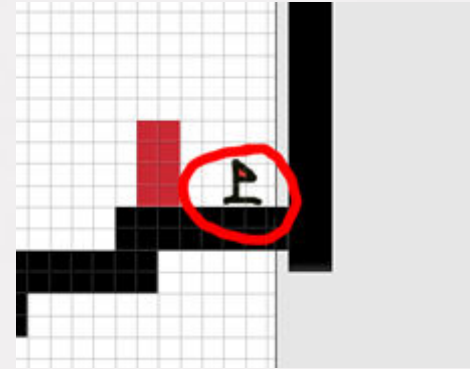
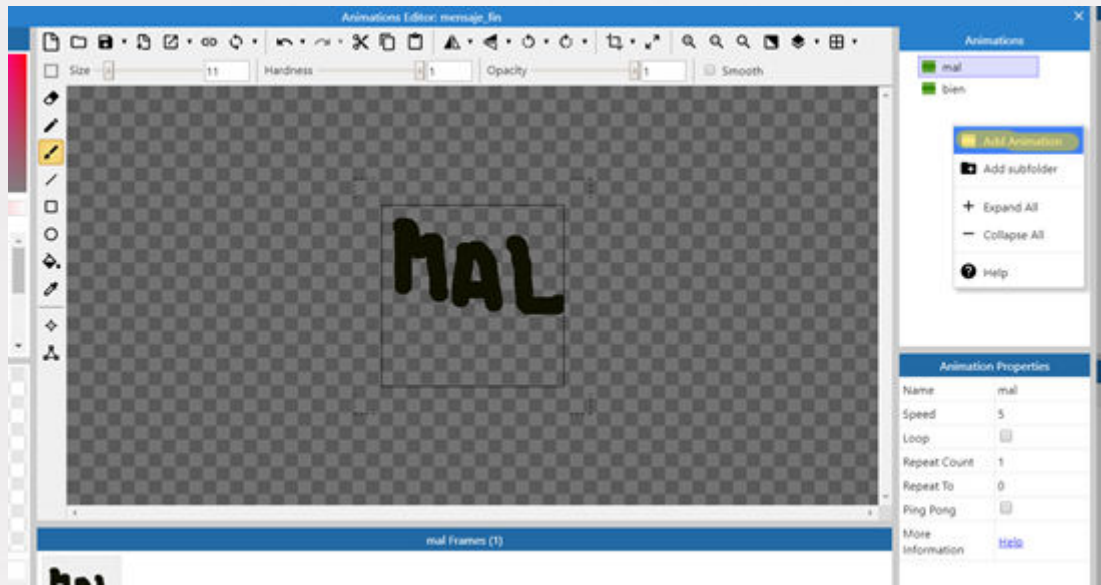
 System  player  enemy	For each  enemy	 enemy	Simulate  Platform pressing Left
	$X < \text{enemy.X}$	Add action	Add...
	Is <b>perseguir</b>		
 System  player  enemy	For each  enemy	 enemy	Simulate  Platform pressing Right
	$X > \text{enemy.X}$	Add action	Add...
	Is <b>perseguir</b>		
 System   System	For each  enemy	 enemy	Set <b>perseguir</b> to True
	distance(player.X, player.Y, enemy.X, enemy.Y) < 300	Add action	Add...

Para que los enemigos no persigan al jugador de entrada, vamos a añadir una nueva condición que hará que sólo lo persigan cuando se acerque a ellos.

Para ello creamos una nueva variable booleana para los enemigos (perseguir) que pondremos en true cuando el jugador se acerque a ellos. Los enemigos no se moverá si esta variable no está activa.

 enemy	 Platform has wall to left	 enemy	Simulate  Platform pressing Jump
 enemy	<b>OR</b>  Platform has wall to right	Add action <span>Add..</span>	

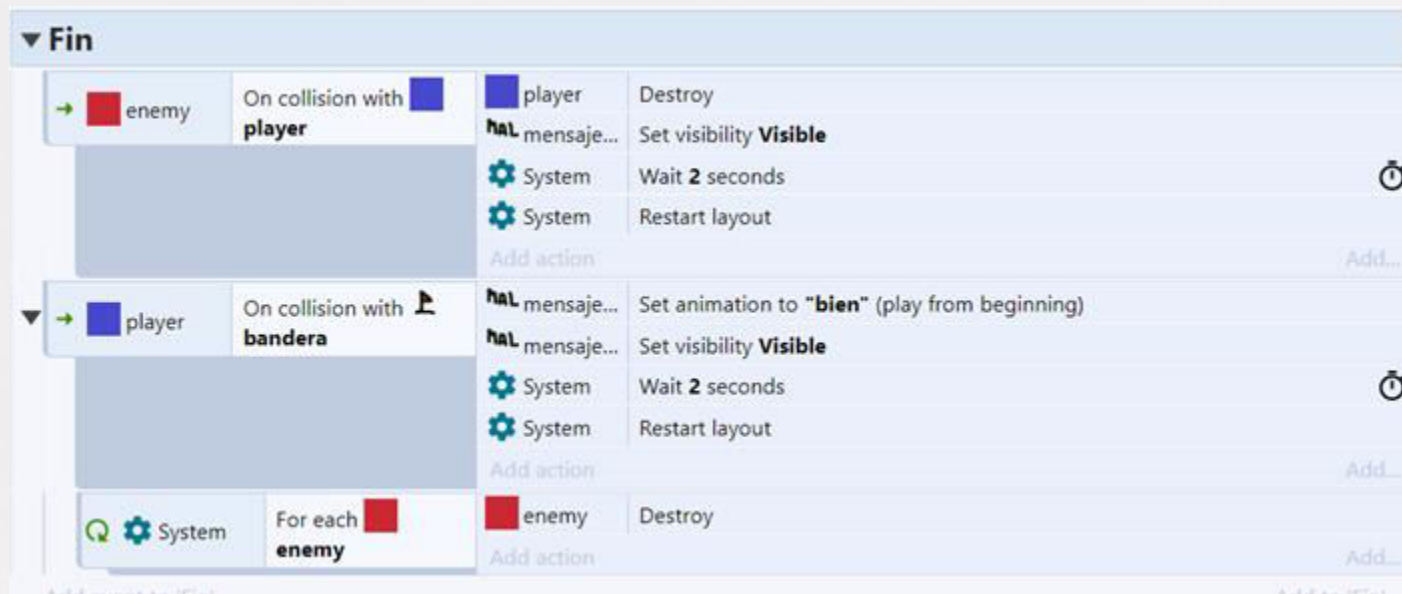
Añadimos un nuevo evento para que los enemigos salten los obstáculos cercanos y no se queden atascados.



Los niveles pueden terminar de dos formas: llegando a una meta o cuando nos pilla un enemigo.

Para la primera condición añadiremos un nuevo objeto “sprite” al proyecto (una bandera) y la colocaremos en el “layout”.

Necesitaremos crear otro nuevo “sprite” que mostrará una imagen en pantalla cuando se termina el nivel. Lo arrastramos al centro del “layout” y desactivamos la condición de “initially visible” en el inspector. Además hacemos doble click en este “sprite” para abrir el editor de imágenes y luego hacemos click derecho en animaciones para añadir una nueva (teniendo así una imagen para cuando completamos el nivel y otra para cuando nos pilla un enemigo).



Añadiremos dos nuevos eventos:

- Uno que hará aparecer en pantalla el último "sprite" que hemos añadido, esperará 2 segundos y reiniciará el nivel.
- Otro que, si tocamos la bandera, hará aparecer ese mismo "sprite" pero le pondrá la animación que tengamos para cuando el nivel se ha completado de forma correcta. Luego esperará 2 segundos y reiniciará el nivel.