

PMDM14.- Desarrollo de un videojuego.

Caso práctico

Juan está decidiendo su futuro profesional a corto plazo. Para ello ha decidido probar usando uno de estos motores de videojuegos sobre los que tanto puede leerse en Internet. En la empresa donde ha realizado la entrevista le han hablado de GDevelop; al parecer es uno de los motores usados para juegos sencillos. El primer paso a seguir por parte de Juan será instalar el motor en su ordenador para intentar construir un juego por sí mismo y entender cómo sería su trabajo en caso de aceptar la oferta.



Mikhail ([Pexels](#))

Leyendo sobre el motor GDevelop descubre que es sencillo de aprender y que cuenta con varios elementos interesantes como la capacidad de uso de la Inteligencia Artificial para algunos eventos como el cálculo de caminos hacia objetos en la escena del videojuego.

Además, se trata de un motor gratuito y que ofrece un montón de posibilidades para introducirse en este mundo de creación de juegos. Otra de las ventajas es que es multiplataforma, es decir desde su página web existen instaladores/ejecutables para Windows, Linux y MacOSX.



[Ministerio de Educación y Formación Profesional.](#) (Dominio público)

Materiales formativos de FP Online propiedad del Ministerio de Educación y Formación Profesional.

[Aviso Legal](#)

1.- La herramienta de desarrollo GDevelop. Introducción.

Caso práctico

Se trata de un motor de videojuegos de código abierto para dispositivos móviles y ordenadores en géneros como plataformas, puzzles, estrategia, disparos y más.

Cuenta con un sistema de eventos con los que es posible determinar la lógica de tu juego: podrás añadir comportamientos predeterminados a tus objetos de juego, o expandir y crear nuevos comportamientos que actúen de manera intuitiva.

Incluye sprites con múltiples animaciones, emisor de partículas, máscaras de colisión personalizadas y generador de patrones entre otros recursos. También es posible expandir el motor de juego mediante JavaScript e incluso incluir anuncios para obtener una fuente de ingresos por medio de tu proyecto.

El primer paso es la instalación del motor y el segundo paso la creación de un nuevo proyecto.



[GDevelop en Wikipedia](#) (Dominio público)

Para llevar a cabo nuestro videojuego, vamos a utilizar una potente, pero a la vez sencilla herramienta de código abierto llamada GDevelop.

GDevelop permitirá crear juegos HTML5 para ser jugados en cualquier navegador web o mediante aplicaciones. Según el propio desarrollador, los juegos son exportables a las plataformas iOS, Android, Steam, Facebook Gaming, Itch.io, Newgrounds, the Microsoft Store y otras sin necesidad de tener que usar, o siquiera conocer un lenguaje de programación específico. Dispone de una organización y distribución de componentes bastante intuitivo y donde la potencia principal de la herramienta reside en un logrado y completo sistema de gestión y control de eventos.

Versiónes anteriores del motor ofrecían juegos para entornos Escritorio Windows y Linux en formato C++, pero esta característica está obsoleta.

La [arquitectura de GDevelop](#) , según la fuente indicada en el enlace, viene distribuida en una serie de directorios cuya finalidad está descrita en esta lista:

- ✓ Core: La biblioteca principal de GDevelop, que contiene herramientas comunes para implementar el IDE y trabajar con los juegos de GDevelop.
- ✓ GDJS: El motor de juego, escrito en TypeScript y que usa PixiJS (WebGL) para crear juegos todos los juegos GDevelop.
- ✓ GDevelop.js: Enlaces de Core, GDJS y Extensiones a JavaScript (con WebAssembly), utilizados por el IDE.

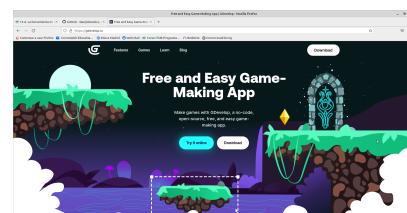
- ✓ newIDE: El nuevo editor del juego, escrito en JavaScript con React, Electron y PixiJS.
- ✓ Extensions: Extensiones para el motor del juego, provee objetos, comportamientos, eventos y otras características.

1.1.- Instalación de la herramienta de desarrollo GDevelop.

Podemos obtener GDevelop para instalarlo en nuestro ordenador de través de la página: <https://gdevelop.io/download>  Se accede a la página mostrada en la figura 1.

Si clicamos sobre el enlace Download, se abrirá la página mostrada en la figura 2, con enlaces a la instalación sobre los sistemas operativos más utilizados, e incluso es posible crear juegos sin ni siquiera instalar la herramienta, utilizando directamente el navegador web.

Figura 1. Página GDevelop.



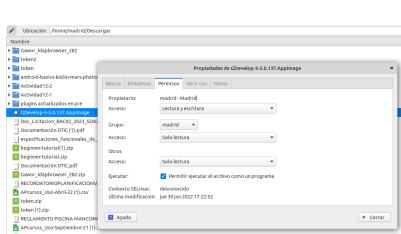
GDevelop.io (Todos los derechos reservados)

Figura 2. Clic en Download.



GDevelop.io (Todos los derechos reservados)

Se deja a elección del alumno la versión a instalar, aunque se recomienda la más reciente. Puede haber cambios con respecto a lo que aquí se muestre, pero en todo caso, será bastante similar. En este caso el juego se ha desarrollado con la versión 5.5.0.137, en un sistema operativos MAX 11.5 (Madrid Linux), una distribución basada en Ubuntu 20.04. En caso de tener un sistema operativo Windows el proceso de instalación es el mismo que con cualquier otro tipo de aplicación Windows.



Ministerio de Educación (Uso educativo nc)

En el caso de Linux, la aplicación se distribuye en forma de AppImage. Una vez descargada y descomprimida es necesario dotar al fichero de imagen (.AppImage) de permisos de ejecución y para ejecutarlo basta con ejecutar en un terminal el nombre del paquete desde la carpeta donde se encuentra descomprimido. En este caso desde la carpeta /home/madrid/Descargas ejecutaremos ./GDevelop-5.5.0.138.AppImage

1.2.- Tutoriales y procedimiento de elaboración de nuestro juego.

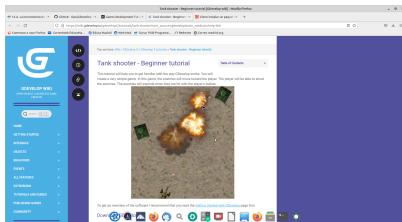


[GDevelop.io](https://gdevelop.io/academy) (Todos los derechos reservados)

Volviendo a la página web principal de GDevelop, podemos observar que en la parte superior existe un botón Learn que habilita el acceso a la zona de tutoriales para crear unos juegos de temáticas diferentes y que con un esfuerzo relativamente pequeño nos permitirán crear en poco tiempo unos juegos, ambos de tipo 2D, con resultados bastante satisfactorios:

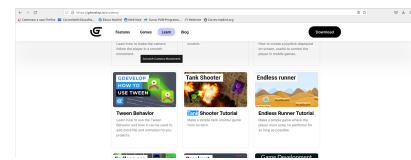
<https://gdevelop.io/academy> 

Si nos desplazamos hacia abajo en la página, encontraremos un tutorial sobre la creación del juego "Tank Shooter Game", el enlace nos lleva a un manual que explica cómo crear un juego donde un cañón orientable por el jugador y que dispara proyectiles de manera controlada también por el jugador, se defiende del avance dirigido hacia éste, de múltiples tanques que tratan de colisionar contra dicho cañón.



[GDevelop.io](https://gdevelop.io/academy) (Todos los derechos reservados)

Si abrimos el tutorial accederemos a las instrucciones y a una serie de recursos como son sprites de los personajes, imágenes estáticas, sonidos, fuentes para textos, etc. Te recomendamos que uses este manual para los casos en que no encuentres una solución a cómo hacer los distintos puntos de la creación del juego.



[GDevelop.io](https://gdevelop.io/academy) (Todos los derechos reservados)

2.- Videojuego de tanques.

Caso práctico



Mikhail (Pexels)

Juan comienza a desarrollar el videojuego siguiendo uno de los tutoriales que ha encontrado en la web del motor de videojuegos que va a utilizar. Descubre que es un nuevo mundo con multitud de opciones, alternativas y sobre todo una componente gráfica muy importante. Es tremadamente sencillo crear un juego si se dispone de los elementos gráficos creados de antemano y, sin tirar ni una sola línea de código, el motor se encarga de transformar los eventos, las acciones y los objetos en código JavaScript que ejecutará el dispositivo que ejecute finalmente el juego.

El trabajo con este tipo de motores le ha abierto un nuevo camino para poder desarrollar juegos de una forma muy rápida, lo cual es un valor añadido para su currículum y para el conjunto de herramientas que controla y con la que puede ofrecer más alternativas a las empresas que le contratan.

Definitivamente, Juan ha decidido su futuro a corto plazo. Aunque el mundo de los videojuegos es muy tentador, la ausencia de necesidad de escribir código, al menos en estos motores sencillos, hace que se decante por seguir apostando por el desarrollo de aplicaciones Android, donde la importancia de los conocimientos técnicos de informática y de programación es vital. En este campo, Juan se siente competente y puede marcar la diferencia frente a otros profesionales que no han estudiado el ciclo de grado superior de Desarrollo de Aplicaciones Multiplataforma, así que por ahora continuará su aprendizaje trabajando junto a sus compañeros en su actual empresa.

El próximo reto de Juan es... ¡aprender Kotlin! Pero el tuyo, por ahora, diseñar este videojuego de tanques.

Autoevaluación

Elije las respuestas adecuadas sobre GDevelop...

- Los juegos resultantes están en HTML5.

- La versión actual de GDevelop ofrece juegos en formato C++ para juegos nativos de Windows y Linux.

- El IDE está desarrollado con React, Electron y PixiJS.

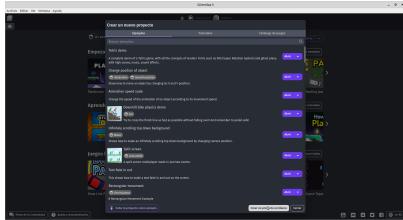
- El motor está escrito con TypeScript.

[Mostrar retroalimentación](#)

Solución

1. Correcto
2. Incorrecto
3. Correcto
4. Correcto

2.1.- Creación del proyecto.



Elaboración propia a partir de GDevelop ([MIT License](#))

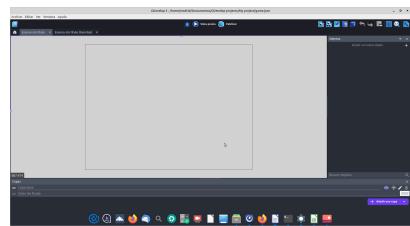
En los siguientes apartados vamos a ver cómo crear un videojuego. Lo primero que tenemos que hacer es entrar en la web [gdevelop.io](#) y descargar la aplicación. Una vez que la tenemos instalada y la hemos arrancado, debemos seleccionar la opción de Archivo > Crear un nuevo proyecto. En este momento podremos descargar alguno de los proyectos existentes (opción muy interesante para aprender), o crear uno desde cero.

A continuación seleccionamos un nombre para nuestro proyecto y una ruta donde se almacenarán todos los ficheros del juego.

Nota importante: antes de seguir, se recuerda al alumno la conveniencia de ir guardando todos los cambios que se vayan haciendo en el proyecto. No se guardan automáticamente los cambios, por lo que no salvarlos sistemáticamente puede jugarnos una mala pasada.

Se creará entonces una escena y quedará preparada para la incorporación de componentes del juego con las siguientes áreas principales:

- ✓ Una escena está compuesta por varias capas, como puedes ver en la parte inferior.
- ✓ En la parte derecha está la zona de creación de objetos.
- ✓ En la pestaña de la derecha, "Escena sin título (Eventos)" es donde podremos programar el comportamiento de nuestro juego.



Elaboración propia a partir de GDevelop ([MIT License](#))

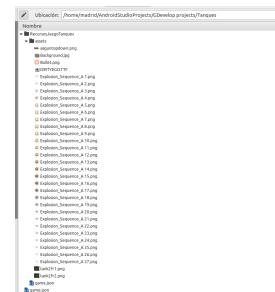
2.2.- Incorporación de recursos.

A continuación incorporaremos los recursos del juego. Concretamente podrán descargarse del siguiente enlace:

<https://wiki.compilgamer.net/lib/exe/fetch.php/gdevelop5/tutorials/beginner-tutorial.zip>



Tendremos que descomprimir dicho fichero e incorporar el contenido en la misma carpeta del proyecto. En nuestro caso hemos renombrado la carpeta de recursos a "RecursosJuegoTanques".



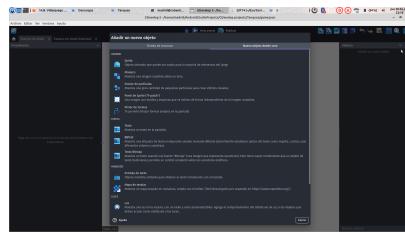
Elaboración propia a partir de GDevelop
[\(MIT License\)](#)

2.3.- Creación del objeto cañón e incorporación a la escena.

A continuación incorporaremos el cañón con el cual nos defenderemos del avance y amenaza de los tanques. Consistirá en un objeto cuya imagen se encuentra entre los recursos proporcionados. Para ello sobre la zona derecha de la pantalla, en la zona de Objetos clicaremos en "+" para insertar un nuevo objeto, apareciendo la ventana desde la que elegiremos el tipo de objeto "Sprite" (figura 1), donde indicaremos el nombre "Canon" (figura 2).

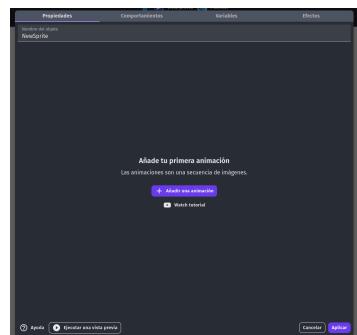
Sobre dicha ventana tendremos que agregar la imagen del cañón. Para ello, pulsamos "Añadir una animación" y en la siguiente ventana pulsaremos al botón Añadir y seleccionaremos la imagen del cañón marrón de la carpeta de los elementos descargados en el apartado anterior, quedando como se muestra en la figura 3.

Figura 1. Insertar objeto.



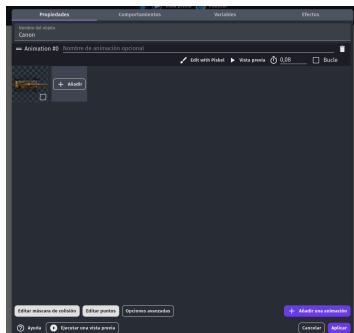
Elaboración propia a partir de GDevelop ([MIT License](#))

Figura 2. Objeto sprite "Canon"



Elaboración propia a partir de GDevelop ([MIT License](#))

Figura 3. Añadir animación.

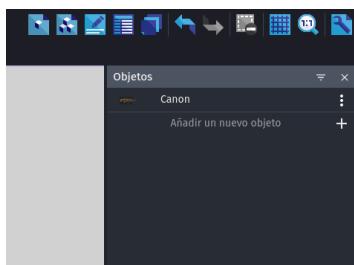


Elaboración propia a partir de GDevelop ([MIT License](#))

Al cerrar la ventana en la parte derecha habrá quedado creado el objeto "Canon" tal y como se muestra en la figura 4 y podemos arrastrarlo hasta la escena principal para ubicarlo, donde podemos cambiarlo de posición o redimensionarlo (figura 5).

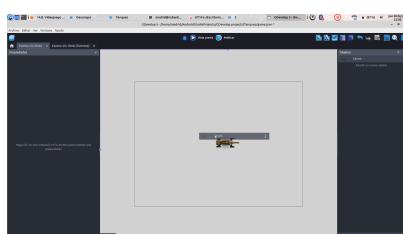
También es posible girarlo haciendo clic sobre el punto de la imagen (figura 6).

Figura 4. Objeto "Canon"



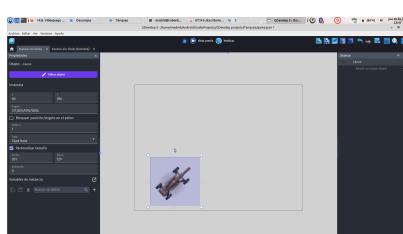
Elaboración propia a partir de GDevelop ([MIT License](#))

Figura 5. Reposicionar o redimensionar.



Elaboración propia a partir de GDevelop ([MIT License](#))

Figura 6. Girar.

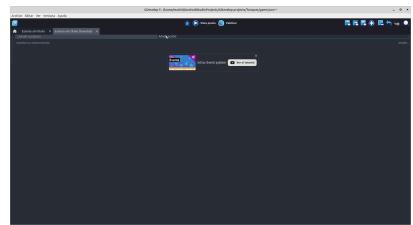


Elaboración propia a partir de GDevelop ([MIT License](#))

2.4.- Incorporando los primeros eventos.

Ahora es momento de incorporar los primeros eventos. **En nuestro caso, haremos que el cañón gire orientándose hacia donde se sitúe, en la escena del juego, el cursor de ratón.**

Para ello tenemos varias opciones. La más rápida es pulsando sobre la pestaña superior "Escena sin título (Eventos)". Aquí pulsamos sobre "Insertar un nuevo Evento" quedando como se muestra en la imagen.

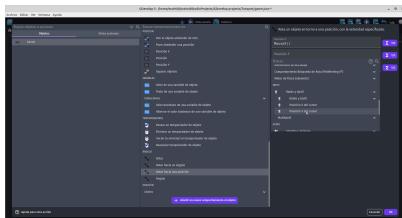


Elaboración propia a partir de GDevelop ([MIT License](#))

Ahora agregaremos una acción (pulsamos sobre el texto "Añadir una acción" en la parte derecha), apareciendo la ventana donde seleccionamos el objeto "Canon" y nos desplazamos en la parte derecha hacia abajo hasta encontrar "Rotar hacia una posición" en el grupo ÁNGULO donde podemos especificar los siguientes valores que indican que el objeto girará hacia la posición del ratón:

- ✓ Posición X: MouseX()
- ✓ Posición Y: MouseY()
- ✓ Velocidad Angular: 0 (inmediata)

NOTA: Podemos escribir directamente observando que el IDE muestra el color rojo en caso de que el valor introducido no sea correcto sintácticamente o pulsar también el botón morado "E 123" y seleccionar en INPUT los eventos del ratón "Posición X del cursor" e Y. Observa que los valores y las variables son sensibles a mayúsculas.



Elaboración propia a partir de GDevelop ([MIT License](#))

Al dar a Ok, ya se quedará en primer lugar en el desplegable de acciones.

Como para dicha acción no hemos especificado condición (sin condiciones), el cañón refrescará su posición aproximadamente 60 veces por segundo, de forma que prácticamente de manera inmediata, cada vez que mueva el cursor, el cañón se moverá apuntando al cursor.

Ahora sería el momento de comprobar que efectivamente el cañón obedece al evento de giro descrito. Para ello probaremos lo que hace hasta el momento mi aplicación pulsando el botón superior de Vista Previa para comprobar que el cañón gira cuando movemos el ratón.

2.5.- Más eventos del cañón. Disparos.

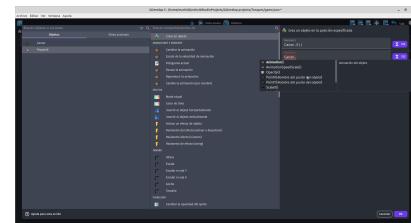
En este punto del desarrollo, haremos que nuestro **cañón dispare balas o proyectiles al pulsar el botón principal del ratón**. Para ello, lo primero será incorporar el objeto característico o representativo de un proyectil, siguiendo los mismos pasos que cuando creamos el Cañón (objeto "Canon"), es decir, sobre la zona derecha de la pantalla, en la zona de Objetos clicaremos en "+" para insertar un nuevo objeto, apareciendo la ventana desde la que elegiremos el tipo de objeto "Sprite", donde indicaremos el nombre "Proyectil".

A continuación, sobre dicha ventana tendremos que agregar la imagen del proyectil (bullet.png). Para ello, pulsamos "Añadir una animación" y, en la siguiente ventana, pulsaremos al botón Añadir y seleccionaremos la imagen del proyectil  de la carpeta de los elementos descargados anteriormente.

El siguiente paso consistirá en crear un nuevo evento desde la pestaña superior derecha (escena sin título "Eventos") y en este segundo evento creado pulsamos en la parte derecha, encima del título "Añadir Acción". Después seleccionamos en la izquierda el tipo de objeto que queremos crear: proyectil. En la derecha indicamos el comportamiento que queremos, en este caso seleccionamos Crear un Objeto. Cuando lo hacemos a la derecha se abren las propiedades de la creación, en este caso necesitamos decirle dónde deseamos (en qué coordenadas de la escena) que sea creado el proyectil.



Elaboración propia a partir de GDevelop ([MIT License](#))

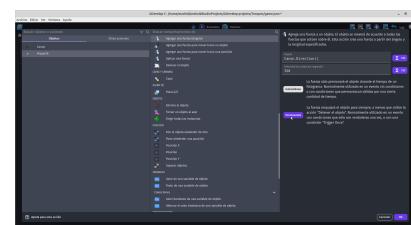


Elaboración propia a partir de GDevelop ([MIT License](#))

Observa que la posición donde queremos que el proyectil sea creado es variable, depende de las coordenadas del cañón, que puede girar. Para indicar esto usamos las funciones Canon.X() y Canon.Y() que devuelven la posición del cañón.

Esto sólo situará el proyectil en dicha posición (además ni siquiera será en la boca del cañón). Ahora añadiremos otra acción, debajo de esta acción ya creada, pero dentro de este mismo segundo evento.

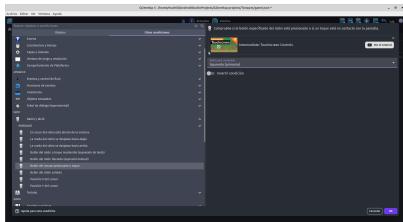
Con esta nueva acción trataremos de dotar de movimiento al proyectil con la misma dirección que tiene el cursor en el momento de ser disparado el proyectil. Para ello seleccionaremos el tipo de objeto proyectil en la parte izquierda, en el medio seleccionamos el grupo MOVIMIENTO USANDO FUERZAS --> Agregar una fuerza (ángulo) y en la parte derecha indicamos estas propiedades:



Elaboración propia a partir de GDevelop ([MIT License](#))

- ✓ Ángulo: Canon.Direction() que indica que aplicará la misma dirección al proyectil que ángulo esté girado el cañón.
- ✓ Velocidad: 350 (píxeles por segundo).
- ✓ Permanente: Marcado.

Ahora debemos asociar a estas dos acciones de este segundo evento una condición en la parte izquierda para que se desencadenen. En nuestro caso, haremos que al clicar sobre el botón izquierdo del ratón se realicen dichas acciones de creación del proyectil y su movimiento dirigido. Para ello, sobre este segundo evento añadiremos una condición ("Añadir



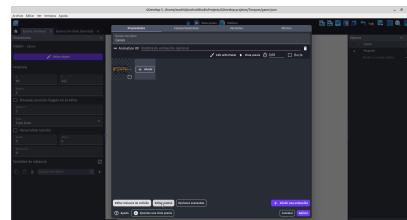
Elaboración propia a partir de GDevelop ([MIT License](#))

condición" en la parte izquierda). Elegiremos "Ratón y táctil" > "Botón del ratón presionado o toque" > y en el campo "Botón para comprobar" dejamos "Izquierda (Primario)", o bien lo seleccionaremos a través del navegador de botones a su derecha.

Ya podremos probar el resultado, viendo que efectivamente dispara, pero con el inconveniente de que los proyectiles no salen de la boca del cañón, ya que hemos indicado la coordenada del cañón en general (todo el objeto) pero no de la boca del cañón.

Para solucionarlo debemos cambiar el punto donde se crea el proyectil, pasando a indicar que ese punto es la boca del cañón. Para indicarlo, vamos a crear un punto con coordenadas en el objeto "Canon" que llamaremos "Boca" y que cuyas coordenadas usaremos después.

Para ello, seleccionamos arriba a la izquierda la pestaña "Escena sin título". Esto despliega a la derecha la lista de los objetos creados hasta ahora ("Canon" y "Proyectil"). Hacemos doble clic el objeto "Canon" y clicamos el botón "Editar Puntos" en la parte inferior.



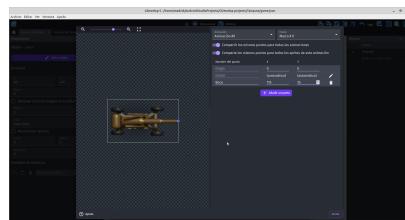
Elaboración propia a partir de GDevelop ([MIT License](#))

Ahora añadimos un punto en la derecha y especificamos las coordenadas adecuadas, que pueden variar dependiendo de si has cambiado el tamaño del cañón. Observa que el punto debe quedar justo en la boca del cañón, como se muestra en la figura 1, y observa que le hemos dado el nombre al punto de "Boca".

Toca reescribir las coordenadas de creación del proyectil para que, en lugar de crearse en el punto "Canon.X()" x "Canon.Y()", se creen en los puntos "Canon.PointX("Boca")" x "Canon.PointY("Boca")", tal y como muestra la figura 2.

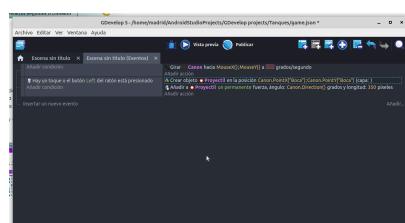
De esta forma el proyectil saldrá de la boca del cañón como se aprecia en la figura 3.

Figura 1



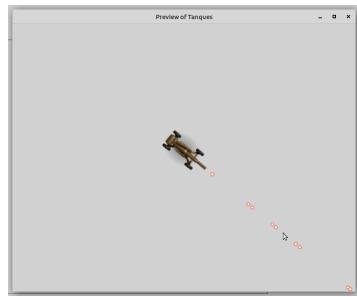
Elaboración propia a partir de GDevelop ([MIT License](#))

Figura 2



Elaboración propia a partir de GDevelop ([MIT License](#))

Figura 3

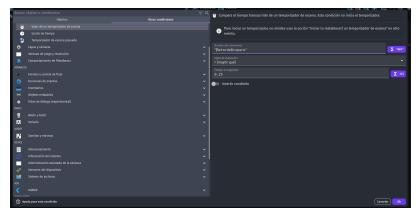


Elaboración propia a partir de GDevelop ([MIT License](#))

Además, si queremos, podemos incorporar un pequeño retardo desde que clico el botón izquierdo del ratón por primera vez y el proyectil se dispara. Quizá en este juego este comportamiento no sea necesario, pero en otro tipo de juego puede que sí. Veamos cómo hacerlo.

Se tratará de incorporar una condición basada en un temporizador, una especie de alarma para que hasta que no suceda no permita crear los proyectiles. Esto retardará la creación de los disparos.

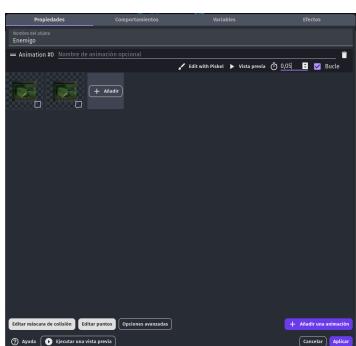
Para ello, sobre este segundo evento, agregaremos una segunda condición yendo a Cronómetros y tiempo --> Valor de un temporizador de un temporizador de escena. Aparecerán tres campos que rellenaremos: por un lado, el nombre que daremos a este temporizador: "RetardoDisparo"; por otro lado, el tiempo medido en segundos: 0.25; y por otro lado el signo de evaluación: ">". Esto significa que hasta que no pasen 0.25 segundos no se podrá hacer un siguiente disparo.



Elaboración propia a partir de GDevelop ([MIT License](#))

Si ahora lo probamos, veremos que aún no se produce el retardo. Se debe a que, como puedes ver en la imagen anterior, es necesario iniciar el temporizador, algo así como activar la alarma. Veremos cómo hacerlo en próximos apartados.

2.6.- Creación de los objetos enemigos. Tanques como imágenes animadas.



Elaboración propia a partir de GDevelop ([MIT License](#))

Ahora trataremos de **crear los enemigos del cañón**, que básicamente serán múltiples tanques que irán apareciendo de manera aleatoria en posición (solamente exigiremos que aparezcan por la parte de arriba) cada cierto tiempo (puede ser a intervalos fijos o aleatoriamente en el tiempo) y **que se acercarán a nuestro cañón**. Para modelizar el comportamiento de cada uno de estos tanques, crearemos el objeto de tipo "Enemigo" de modo similar al Cañón, pero la diferencia será que agregaremos dos imágenes con idea de crear una imagen animada provocada por la alternancia sucesiva de esas dos imágenes. En la animación dichas imágenes se intercambiarán cada 0.05 segundos, de modo que dará sensación de que las cadenas del tanque enemigo se mueven.

Una vez incorporadas las imágenes pulsamos sobre la caja de texto, donde está el icono del cronómetro, el tiempo que dura cada imagen antes de cargar la otra, en este caso 0.05 segundos.

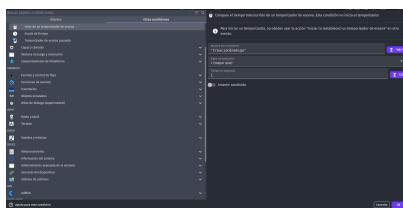
Además, como queremos que las imágenes se alternen y se vuelvan a realizar continuamente en forma de bucle, marcaremos la casilla "Bucle".

2.7.- Fijar dirección de los tanques. Inteligencia artificial.

En este apartado buscamos crear objetos de tanques enemigos constantemente (en intervalos de 1 segundo) y que se dirijan hacia el cañón con ánimo de chocarse contra él.

Para ello añadiremos en el Editor de eventos un tercer evento. En principio, la dirección que deben tomar los tanques debe ser directa hacia al cañón.

En nuestra primera versión, el cañón no se traslada, solo rota moviendo el ratón. Esto significa que los tanques únicamente tendrán que calcular su dirección una vez, cuando inician tras su instanciación el movimiento. Pero si el cañón tuviese incorporado movimiento de traslación, los tanques debieran corregir su dirección para dirigirse a la nueva posición del cañón. Es decir, dispone de ciertas funciones propias de la inteligencia artificial que permiten en todo momento recalcular su ruta atendiendo a factores externos. El método que usaremos de desplazamiento de los tanques tendrá en cuenta esta posibilidad y, al final del proyecto, cuando incorporemos traslación, comprobaremos que este fenómeno está perfectamente contemplado.



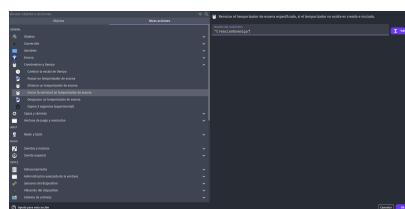
Elaboración propia a partir de GDevelop ([MIT License](#))

En primer lugar, deben crearse tanques que se dirigirán hacia el cañón. Esto lo haremos, por ejemplo, cada segundo. Para ello iremos al editor de eventos y crearemos un nuevo evento, el tercero. Sobre dicho evento añadiremos una condición (haciendo clic en "Añadir condición" a la izquierda). Seleccionamos la condición "Valor de temporizador de escena" que configuraremos de esta forma, llamando a este temporizador "CreacionEnemigo" (debe ir entre comillas).

Además, observa que, al crear el temporizador, el propio interfaz advierte que es necesario iniciar/resestar el temporizador desde un evento. Es decir, hay que añadir una acción para iniciar e ir reseteando el contador, de forma que cada vez que pase un segundo, se lleva a cabo la acción, paramos el tiempo y volvemos a activar la cuenta para que en el próximo segundo volvamos a hacer la acción repetidamente.

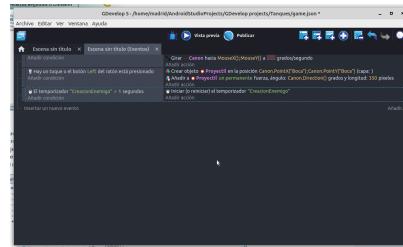
Para ello damos a "Añadir acción" en la derecha y seleccionamos en el grupo de "Cronometro y tiempo" > "Iniciar (o reiniciar) un temporizador de escena". Ahora configuramos el Nombre del cronómetro, el mismo que indicamos en el evento anteriormente, es decir, "CreacionEnemigo" (figura 1). Damos a Ok quedando como muestra la figura 2.

Figura 1



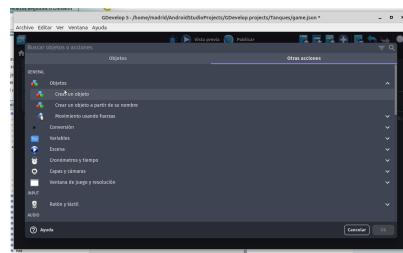
Elaboración propia a partir de GDevelop ([MIT License](#))

Figura 2



Elaboración propia a partir de GDevelop ([MIT License](#))

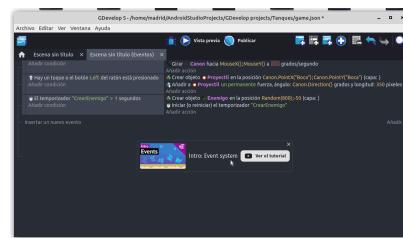
Además, ahora añadiremos otra acción para crear un objeto "Enemigo". Para ello, daremos de nuevo, para este tercer evento, en la columna de acciones, "Añadir acción" > Pestaña "Otras acciones" > Objetos > "Crear un objeto", escribiendo como primer parámetro (Objeto a crear): "Enemigo", como segundo parámetro (Posición X, del objeto a crear): Random(800), de forma que el objeto se creará con un valor aleatorio en la componente X, que cubrirá más o menos el ancho de la ventana del juego, mientras que el tercer parámetro (Posición Y, del objeto a crear): -50, de forma que los tanques se crearán fuera de la ventana de juego (por la parte superior, teniendo en cuenta que el valor 0 del eje Y corresponde justo al marco horizontal superior, para que así el jugador no tenga la sensación de que los tanques de repente "aparecen").



Elaboración propia a partir de GDevelop ([MIT License](#))



Elaboración propia a partir de GDevelop ([MIT License](#))



Elaboración propia a partir de GDevelop ([MIT License](#))

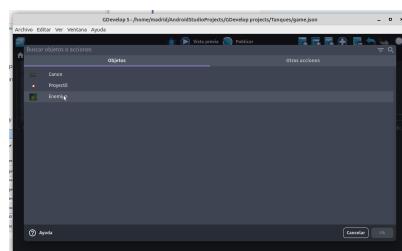
Esto hará que los tanques se creen, sin embargo, no tienen atribuido todavía movimiento. Para ello debemos crear un nuevo evento (cuarto) pero sin ninguna condición, ya que los tanques siempre se moverán.

Ahora sobre este cuarto evento, añadiremos una acción (figura 3): "Añadir acción", marcamos el objeto "Enemigo" (esto aplicará sobre todas las instancias de esta clase de objetos, todos los enemigos), seleccionamos "Agregar una fuerza para mover hacia un objeto", y rellenaremos los campos de la forma:

- ✓ Objetivo: Canon (ya que los enemigos se moverán hacia donde esté el cañón).
- ✓ Velocidad (en pixeles por segundo): 1 (puede que sea un bug del motor en el momento de hacer esta documentación, pero con 150px que es lo esperadamente adecuado va demasiado deprisa).
- ✓ Permanente: Marcado.

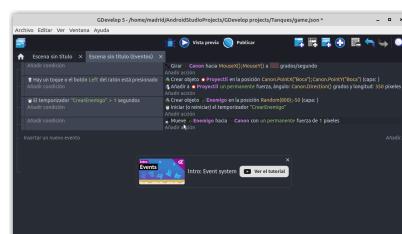
Dando OK el resultado quedaría tal y como muestra la figura 4.

Figura 3.



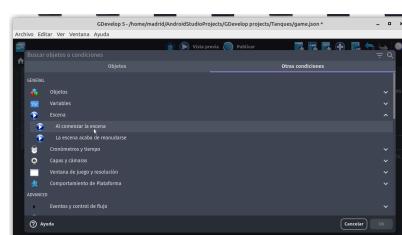
Elaboración propia a partir de GDevelop ([MIT License](#))

Figura 4.

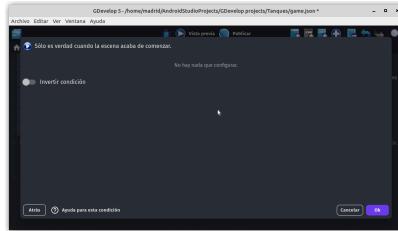


Elaboración propia a partir de GDevelop ([MIT License](#))

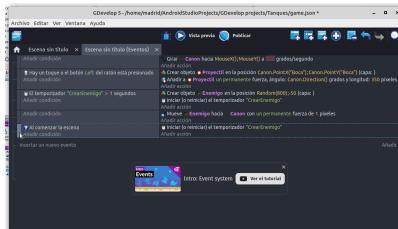
Para que los enemigos sean creados es necesario arrancar el temporizador que vaya generando eventos cada segundo. La idea es arrancarlo al empezar el juego y cada vez que se detecte que ha pasado un segundo, crear al enemigo y volver a reiniciar el contador. Esto es justo lo que hacemos creando el siguiente evento que, aunque al inicio estará en la parte de abajo, luego lo arrastraremos hacia la parte de arriba, para que quede más claro que se lanza al inicio de cada partida.



Elaboración propia a partir de GDevelop ([MIT License](#))



Elaboración propia a partir de GDevelop ([MIT License](#))



Elaboración propia a partir de GDevelop ([MIT License](#))

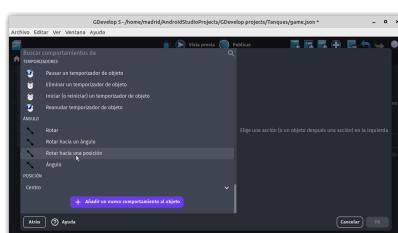


Elaboración propia a partir de GDevelop ([MIT License](#))

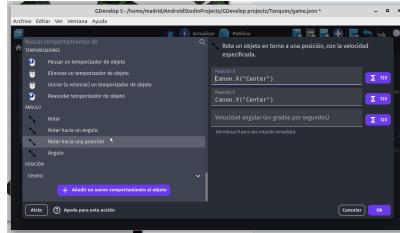
Si hacemos una prueba mediante la vista previa, vemos que los tanques se crean e incluso se aproximan al cañón según lo programado, pero llama la atención que los tanques enemigos no tienen la orientación de manera adecuada, es decir, la boca de su cañón, no está apuntando al cañón del jugador, provocando un efecto bastante irreal.

Para solucionar esto, añadiremos al (ahora quinto) evento una segunda acción detrás de la acción de "Mueve Enemigo...". Para ello pulsamos en "Añadir acción", seleccionamos el objeto "Enemigo", buscamos en el grupo "ÁNGULO" > "Rotar hacia una posición", y rellenaremos los parámetros de la forma:

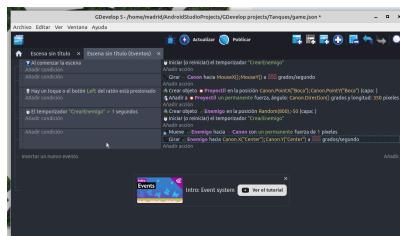
- ✓ Posición X: Canon.X("Center") (se referirá a cada tanque)
- ✓ Posición Y: Canon.Y("Center") (se referirá a cada tanque)
- ✓ Velocidad angular (en grados por segundo) (0 para una rotación inmediata): 0



Elaboración propia a partir de GDevelop ([MIT License](#))

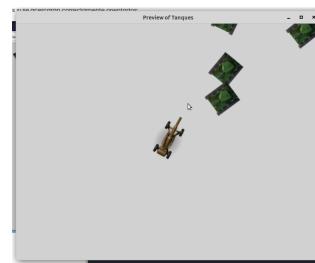


Elaboración propia a partir de GDevelop ([MIT License](#))



Elaboración propia a partir de GDevelop ([MIT License](#))

Si ahora probamos de nuevo, vemos que ahora los tanques sí se acercarán correctamente orientados:



Elaboración propia a partir de
GDevelop ([MIT License](#))

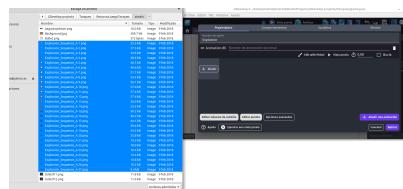
2.8.- Más animaciones. Explosiones.

A continuación trataremos de añadir explosiones para que tras el impacto de un proyectil procedente de cañón sobre el tanque, se muestre una explosión. Para ello añadiremos un nuevo Sprite al cual llamaremos "Explosion".

Para ello, vamos al Editor de objetos del panel derecho superior y hacemos clic sobre "Añadir un nuevo objeto", seleccionamos Sprite y en la pantalla de Propiedades del objeto, indicamos el nombre: "Explosion" y a continuación pulsamos el botón "+ Añadir" desde donde podemos seleccionar y cargar todas las imágenes que tenemos en nuestra carpeta de recursos cuyo nombre son Explosion_X (figura 1).

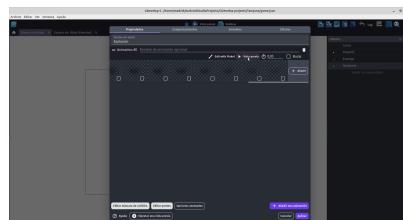
Además, para especificar el tiempo entre cada una de esas imágenes, especificaremos el valor 0,03 (será medido en segundos) en el ícono del cronómetro en la parte superior, quedando como muestra la figura 2.

Figura 1



Elaboración propia a partir de GDevelop ([MIT License](#))

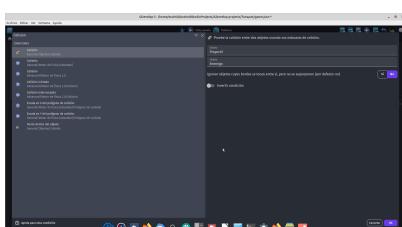
Figura 2



Elaboración propia a partir de GDevelop ([MIT License](#))

Nota: Los nombres que traen los ficheros originales, hacen que por ordenarse alfabéticamente, se cambie el orden de la secuencia de imágenes. Por ello conviene cambiar nombre de forma que el "1" pase a ser "01" y así hasta el "9" por "09".

Podemos comprobar la secuencia sin salir de esta ventana pulsando sobre "Vista previa" junto al lugar dónde has especificado el tiempo entre las imágenes de la secuencia.

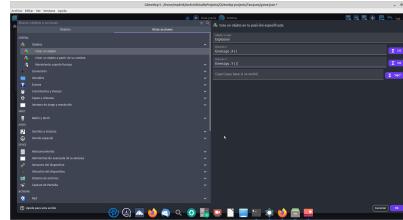


Elaboración propia a partir de GDevelop ([MIT License](#))

Tras esto, el objeto de la explosión (Explosion) estará listo. Cerraremos la ventana de edición del Sprite y nos iremos a la pestaña de "Escena sin título (Eventos)" para crear un nuevo evento, al que añadiremos una condición basada en una colisión. Fíjate que en la parte superior izquierda de la ventana puedes buscar el término por el que se pueden filtrar las condiciones disponibles, en este caso escribimos "Colisión" y configuraremos el evento para que se lance cuando colisione un objeto "Proyectil" con un objeto "Enemigo".

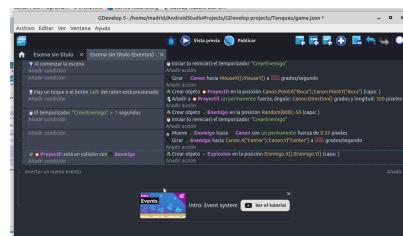
El siguiente paso es añadir la Acción en la parte derecha de la pantalla, que consiste en crear un objeto de tipo explosión, de forma que el evento queda programado para que cada vez que un proyectil impacte o colisione con un enemigo se cree una explosión. Configuramos las coordenadas dónde se crea el objeto "Explosion" con las coordenadas ocupadas por el Enemigo tal y como se muestra en la figura 3, quedando el evento configurado como se muestra en la figura 4.

Figura 3



Elaboración propia a partir de GDevelop ([MIT License](#))

Figura 4

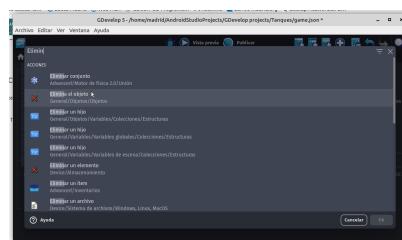


Elaboración propia a partir de GDevelop ([MIT License](#))

2.9.- Efecto final de explosión (I). Desaparición del tanque.

Además, haremos que los tanques enemigos, tras ser impactados y desencadenarse la explosión, desaparezcan. Del mismo modo haremos desaparecer el proyectil que ha impactado con el enemigo, para que no continúe su camino, sino que simulemos que ha provocado la explosión y desaparece junto al tanque eliminado.

Para ello incorporaremos una segunda y una tercera *Acción* de tipo "Elimina el objeto" para este mismo evento como se muestra en las siguientes imágenes:



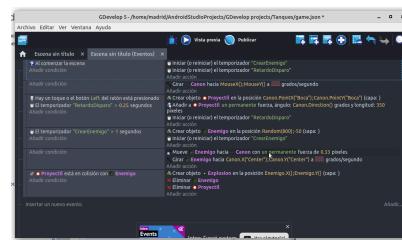
Elaboración propia a partir de GDevelop (MIT License)



Elaboración propia a partir de GDevelop ([MIT License](#))

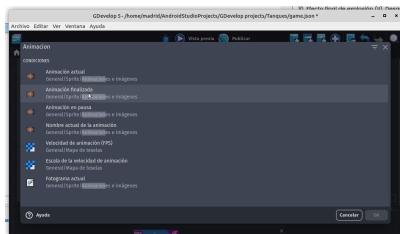
Si previsualizamos, veremos que ya funciona, provocándose las explosiones y desapareciendo los tanques impactados y los proyectiles.

NOTA: Observa que puedes añadir una acción más al inicio de la escena para inicializar el Temporizador de retardo de disparo. Esta acción es opcional y provoca que haya que esperar un cuarto de segundo entre un dispara y otro para dotar a la escena de un poco más de dinamismo.



Elaboración propia a partir de GDevelop ([MIT License](#))

2.10.- Efecto final de explosión (II). Desaparición de estela.



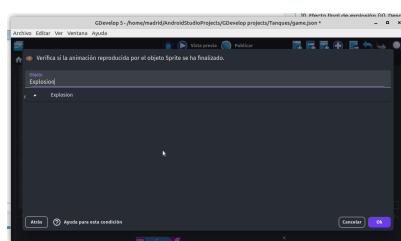
Elaboración propia a partir de GDevelop ([MIT License](#))

Ahora, tras jugar un rato, vemos que la estela o humo de cada explosión queda de manera permanente en la pantalla, dejando toda la escena negra, es decir, no desaparece, simplemente porque el objeto de la explosión realmente no se ha eliminado, sino que simplemente se ha quedado en su último estado (humo negro). Dependiendo de nuestro criterio, podemos decidir que dicha estela finalmente desaparezca, es decir, como si el humo pasado cierto tiempo se "dispara" quedando el fondo "limpio" de nuevo.

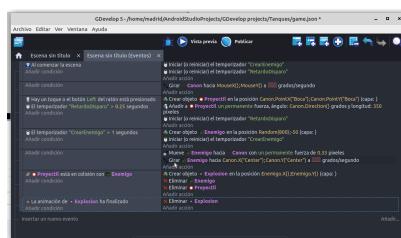
Además, hay que tener en cuenta que al no desaparecer o finalizar dichos objetos, penalizará, si se juega durante un cierto tiempo, el rendimiento del juego, por mantenerse gran cantidad de objetos en la escena.

Para resolverlo haciendo que desaparezcan, debemos añadir un nuevo evento y sobre él una nueva condición de "Animación Finalizada".

Para crear la Acción de "Eliminar objeto" Explosión. Se realiza del mismo modo que en el apartado anterior, quedando de la siguiente forma:



Elaboración propia a partir de GDevelop ([MIT License](#))



Elaboración propia a partir de GDevelop ([MIT License](#))

Autoevaluación

Para poder repetir una acción cada X tiempo en GDevelop...

- Añadiremos una condición que compare el estado de un temporizador con una cantidad de tiempo.

- Añadiremos una acción que compare el estado de un temporizador con una cantidad de tiempo.

- Es necesario iniciar / reiniciar el temporizador mediante una condición.

- Es necesario iniciar / reiniciar el temporizador mediante una acción.

[Mostrar retroalimentación](#)

Solución

1. Correcto
2. Incorrecto
3. Incorrecto
4. Correcto

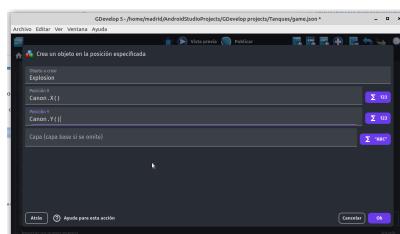
2.11.- Más colisiones. Tanque contra cañón: final del juego.

Ahora además de las colisiones de los proyectiles con los tanques, habrá que tener en cuenta las colisiones de los tanques con el propio cañón. Vamos a programar que **si se produce una colisión de un "Enemigo" con el objeto Cañón ("Canon") lanzaremos un objeto de tipo explosión** (por sencillez utilizaremos el mismo). Además, consideraremos que en ese momento el juego debe finalizar, lanzándose en el siguiente apartado el clásico mensaje de GameOver.

En primer lugar añadiremos el evento y sobre él agregaremos una condición de Colisión, donde rellenaremos los campos "Objeto" con *Enemigo* y de nuevo "Objeto" con el *Canon*. Además, habrá que crear para dicho evento dos acciones:

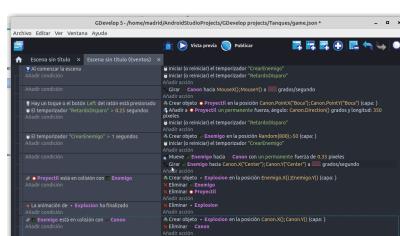
- ✓ La primera consistirá en crear la explosión con una acción de tipo "Crear un objeto en la posición especificada", haciendo donde esté situado el cañón configurando los campos "Posición X" como *Canon.X()* y "Posición Y" como *Canon.Y()* (figura 1).
- ✓ La segunda consistirá en eliminar el objeto *Canon*. Para ello usaremos una acción de "Eliminar el objeto" seleccionando *Canon* quedando los eventos tal y como se muestra en la figura 2.

Figura 1



Elaboración propia a partir de GDevelop ([MIT License](#))

Figura 2



Elaboración propia a partir de GDevelop ([MIT License](#))

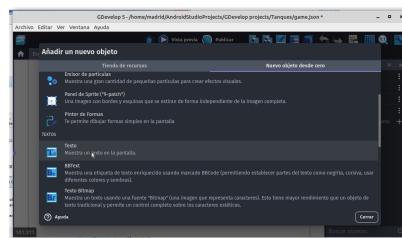
2.12.- Incorporación de texto GameOver.

Vamos ahora a incorporar el objeto que contiene el **texto GameOver para mostrarlo cuando se haya producido el impacto con el Cañón**. Para ello creamos un nuevo objeto, seleccionando el tipo > Texto (figura 1).

Damos a Ok y aparecerá una ventana donde podemos dar nombre al objeto de texto, concretamente se llamará "GameOver" y donde podemos modificar su texto, cambiando el tamaño a 48 y el color de la fuente a rojo (figura 2).

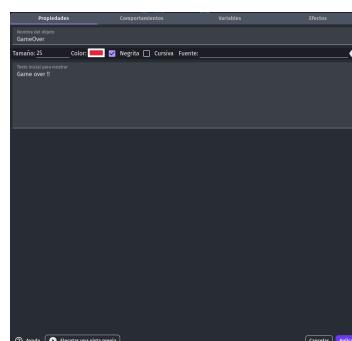
A continuación, arrastraremos con el clic izquierdo, el objeto GameOver desde el Editor de objetos hasta la escena. De esta forma el texto quedará superpuesto al fondo y al propio cañón (figura 3).

Figura 1



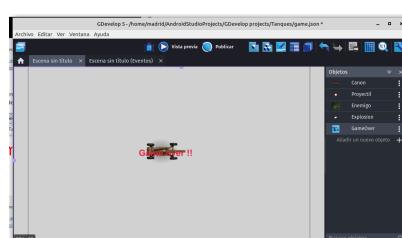
Elaboración propia a partir de GDevelop ([MIT License](#))

Figura 2



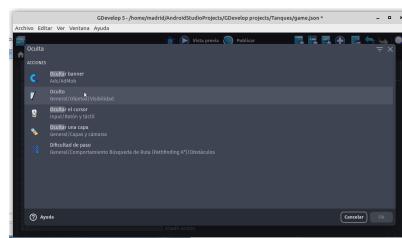
Elaboración propia a partir de GDevelop ([MIT License](#))

Figura 3

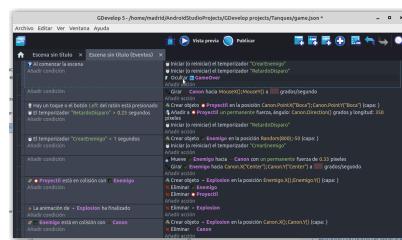


Elaboración propia a partir de GDevelop ([MIT License](#))

Con esto, el texto quedará siempre visible. Lo que haremos será ponerlo oculto en el evento inicial, "Al comenzar la escena". Para ello añadimos una acción de tipo "Oculto", quedando:



Elaboración propia a partir de GDevelop ([MIT License](#))



Elaboración propia a partir de GDevelop ([MIT License](#))

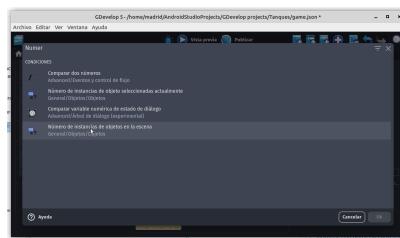
2.13.- Evento de control de fin de partida.

El momento de poner visible el texto de final de partida será en el momento en el que tras la colisión entre un "Enemigo" y el "Canon", este último haya desaparecido de la escena. Simplemente bastará con contabilizar cuantos cañones hay para poder determinar si ha de finalizar o no (habrá un cañón mientras el jugador no haya perdido, y 0 en el momento que el jugador pierda debido a la colisión de un tanque con el cañón).

Para llevarlo a cabo, añadiremos un nuevo evento y sobre él añadiremos una *condición* de tipo "Número de instancias de objetos en la escena" (figura 1).

Establecemos la configuración de la condición para que se lleve a cabo la *acción* asociada al evento en base a que el número de elementos "Canon" sea "igual a" 0, es decir, haya desaparecido por que haya colisionado con un "Enemigo" (figura 2).

Figura 1



Elaboración propia a partir de GDevelop ([MIT License](#))

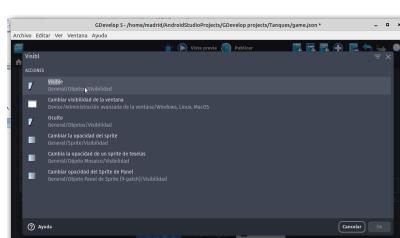
Figura 2



Elaboración propia a partir de GDevelop ([MIT License](#))

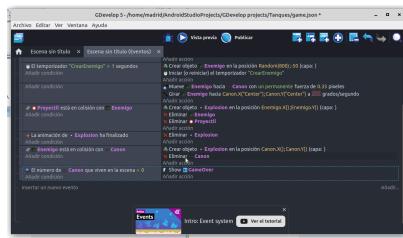
La *Acción* asociada no será otra que hacer visible el texto que había sido ocultado al inicio de la partida (figura 3). Quedando de esta forma configurada como se muestra en la figura 4.

Figura 3



Elaboración propia a partir de GDevelop ([MIT License](#))

Figura 4



Elaboración propia a partir de GDevelop ([MIT License](#))

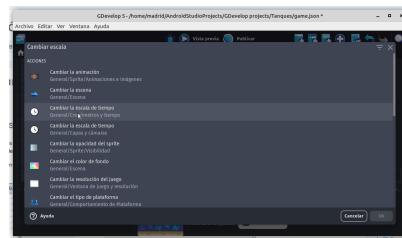
Nota: Si se desea puede añadirse un retardo de entre 0.5 y 1 segundo antes de mostrar el texto de Fin de Partida, del mismo modo a cómo se ha realizado en otros eventos del juego.

2.14.- Bloqueo del juego tras GameOver.

Por último, un efecto también necesario es finalizar el juego una vez que se ha producido el *GameOver*. Esto se debe principalmente al indeseable efecto que provoca el que sigan generándose tanques a pesar de haber finalizado el juego. Para ello, sobre el último evento, el que muestra el texto del Final de Partida, agregaremos una nueva *acción*. Esta debe ser de tipo "Cambiar la escala de tiempo de la escena" (figura 1).

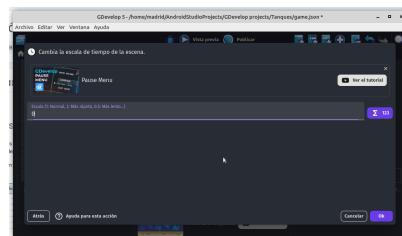
En el campo "Escala (1: Normal, 2: Más rápido, 0,5: Más lento...)" pondremos 0, con lo cual, el juego se parará (figura 2).

Figura 1



Elaboración propia a partir de GDevelop ([MIT License](#))

Figura 2



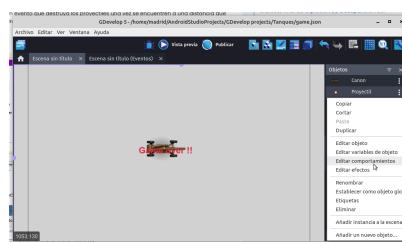
Elaboración propia a partir de GDevelop ([MIT License](#))

2.15.- Optimización de rendimiento. Finalización de hilos innecesarios.

Para optimizar el rendimiento del juego, debemos tratar de finalizar todos los hilos que se generen y que ya no tengan interés, porque son del todo innecesarios y una vez fuera de la vista del usuario lo único que hacen es sobrecargar el juego y penalizar su rendimiento. Concretamente, cuando el cañón dispara cada proyectil, si este sale fuera de los límites de la ventana seguirá recorriendo su trayecto de manera indefinida, aunque esté fuera de los límites de la ventana. Pero incluso con los proyectiles que impactan sobre un tanque sucede igual, ya que en lo que hemos programado hasta ahora dichos proyectiles, a pesar de haber provocado una explosión, siguen recorriendo su trayectoria, pudiendo darse incluso la circunstancia de que un mismo proyectil puede explosionar sobre dos o más tanques si estos se encuentran en la misma trayectoria. Lo cierto es que esto no sería muy real. Por ello vamos a tratar de "poner fin" o imponer una finalización a los proyectiles. En primer lugar, vamos a hacer que estos finalicen si salen de la ventana que ve el usuario. Para ello podríamos incorporar un evento que destruya los proyectiles una vez se encuentren a una distancia que consideremos adecuada, pero en lugar de eso, vamos a utilizar otra técnica que consistirá en incorporar un comportamiento sobre un objeto. Concretamente incorporaremos un comportamiento sobre un el objeto Proyectil.

Sobre el objeto Proyectil, en la parte derecha, damos clic derecho --> Editar comportamientos (figura 1). Aparecerá una ventana sobre la cual marcaremos el ítem "Destruir cuando sale fuera de pantalla" (figura 2).

Figura 1



Elaboración propia a partir de GDevelop ([MIT License](#))

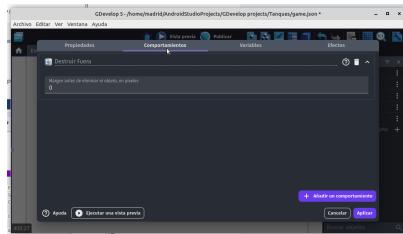
Figura 2



Elaboración propia a partir de GDevelop ([MIT License](#))

Dejamos el valor 0 (figura 3), quedando finalmente este evento tal y como se muestra en la figura 4.

Figura 3



Elaboración propia a partir de GDevelop ([MIT License](#))

Figura 4



Elaboración propia a partir de GDevelop ([MIT License](#))

Con esto ya tendríamos nuestro videojuego de tanques finalizado.