HIT AND RUN

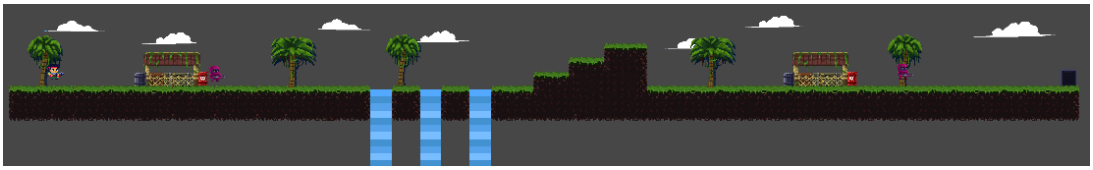
Autor/es:JORGE RODRÍGUEZ FOLGUERA

Indice

1 INTRODUCCIÓN

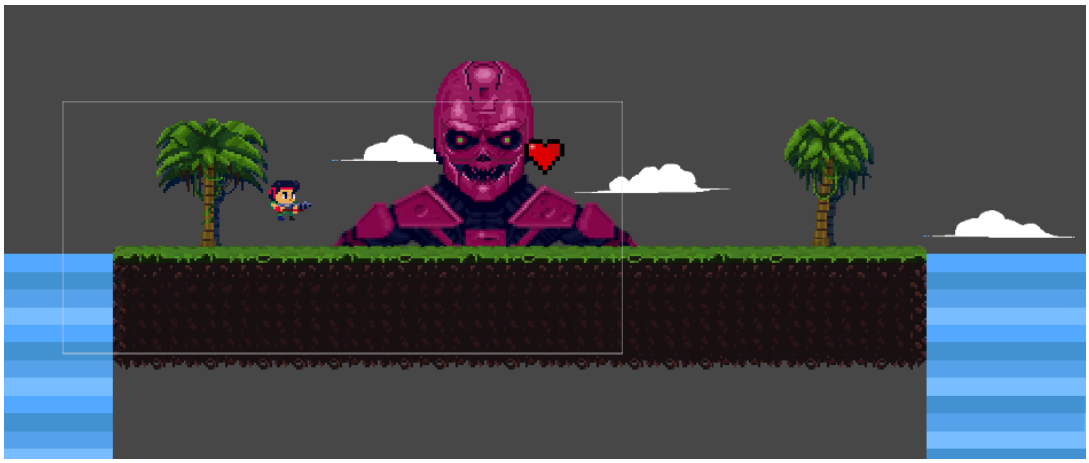
“Hit And Run” es un videojuego 2D, con una visual retro, es decir, de videojuegos antiguos, con secciones de plataformas y obstáculos que evitar y con zonas de disparos donde eliminar los enemigos usando las tres armas que posees. Este juego esta dividido en tres mapas diferentes:

El primer nivel es una zona de practicas, con un par de enemigos para poder poner en practica la mecánica de disparar y las diferentes armas que posees, y un segmento de plataformas donde practicar como saltar.



El segundo nivel se basa en una plataforma móvil, la cual realiza un recorrido con diversos obstáculos por medio los cuales te quitarán un porcentaje de vida si colisionas con alguno de estos, siendo el movimiento del personaje la mecánica a destacar para este nivel.

El tercer mapa es una batalla contra un jefe donde deberás atacarle mientras esquivas sus disparos, donde el saber moverte y el saber atacar son de gran importancia para poder pasarte el nivel.



Este videojuego ha sido creado usando Unity Hub para crear las escenas, las animaciones y cada uno de los objetos que se usan en él. Cada uno de los scripts han sido programados usando Visual Studio 2022 usando el lenguaje C# y todo ello guardado en un repositorio

remoto de git, siendo GitHub, lo cual me ha protegido de perdidas enormes ante fallos de ordenador. También para editar algunos sprites usé la página Pixlr, la cual es una pagina de edición de imágenes.

2 ORIGEN Y CONTEXTUALIZACIÓN DEL PROYECTO

Este proyecto ha salido debido a la intriga que he tenido siempre de como los videojuegos que tanto me han gustado desde pequeño han sido creados. El proyecto de crear yo mi propio juego comenzá a rondar mi cabeza desde que empecé a obtener las bases de programación al entrar a este grado, solo que no lo acabe llevando acabo hasta hoy en día ya que no poseía el conocimiento suficiente para comenzar a programarlo y el completo desconocimiento de este mundo, junto a una falta de motivación, la cual fue dada cuando pensé en hacerlo como proyecto final de curso con un compañero, aunque debido ha ciertos problemas acabe haciéndolo yo en solitario. Cunado plateé en hacerlo para el final de grado aún no sabía como lo llevaría a cabo, el estilo del juego que iba a hacer, ni de donde iba a sacar todo el apartado visual y sonoro, pero según iba informándome y poniéndome con ello mas claras eran mis ideas sobre este.

3 OBJETIVOS DEL PROYECTO

El objetivo principal de la creación de este juego era introducirme en el mundo del desarrollo de estos, ya que como mencioné anteriormente, siempre he tenido una gran curiosidad de como eran hechos los juegos que he disfrutado desde pequeño.

4 TAREAS

4.1 Tarea 1: Obtención de los sprites y la música

Lo primero de todo antes de poder hacer nada en el editor de videojuegos es conseguir los sprites, imágenes que conforman todo el apartado visual del juego ya sean las animaciones como el mapa, y la música y algunos efectos de sonido. Todos estos fueron descargados de internet de forma gratuita con licencia educativa y uso personal, pero no comercial.

4.2 Tarea 2: Creacion de los niveles

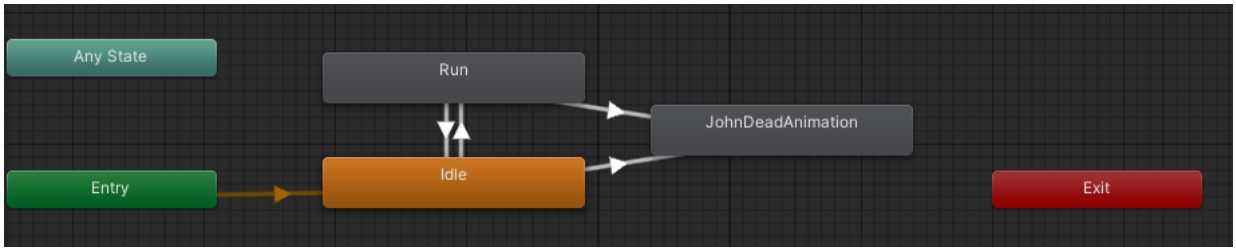
Lo primero de todo era crear un primer mapa donde poder probar todas las mecánicas que se implementen y posteriormente modificarlo hasta crear el primer nivel. Para ello cree una paleta con los sprites que contenían la textura del suelo y la de la cascada. Luego use un tilemap, un GameObject con un componente que permite dibujar tiles o celdas en él, y lo rellené con las imágenes de la paleta creada anteriormente creando así como se vería el suelo. Después, le di las colisiones para evitar que el personaje y los enemigos caigan al vacío. Una vez con el suelo creado, añadimos decoración como algunos arboles, barriles, cajas, etc. Y para decorar el cielo hice un script que crea uno de los tres modelos de nubes según va pasando el tiempo. Guardo los tres modelos en un array, el punto de aparición el cual se localiza a 2 unidades de la cámara y el tiempo de espera, el cual es 20 segundos. Mientras el nivel se esté ejecutando, el punto de aparición se moverá manteniéndose siempre a dos unidades de distancia de la cámara y se comprobara si han pasado 20 segundos de la ultima aparición y en caso afirmativo generará un numero del 0 al 2 y el numero resultante sera el id de la nuve que se generará.

Para evitar que el cielo se llene de nuves y darle mas vida al mundo, las nubes poseen un script el cual hae que se muevan de derecha a izquierda con una velocidad de 0.2 unidades.

4.3 Tarea 3: Creación del personaje principal

4.3.1 Crear las animaciones y añadir el personaje a la escena

Primero crearemos las animaciones, para ello deberemos separar los sprites de cada animación añadiéndole la propiedad múltiple y seleccionando las diferentes posiciones que tendrá en cada acción, ya que por cada acción vienen las posiciones en la misma imagen. Después, seleccionaremos las diferentes posiciones que tendra el personaje para una animación y la crearemos, luego indicaremos que la animacion de correr y la de estar quieto se ejecuten en bulcle. Por ultimo, crearemos un animator controler, el cual se encargará de intercalar las animaciones según se requiera, y le daremos las condiciones que se han de seguir para que se cambie de una animación a otra, desmarcando que cambie de animación cuando acabe la que se está ejecutando y lo haga cuando el script se lo indique. Con todo esto creado, añadiremos una imagen del personaje y le añadiremos el animator controler junto a ciertas propiedades como la colisión, para la cual usaremos un “Capsule Collider 2D” y que se vea afectado por la gravedad con un “Rigidbody 2D”.

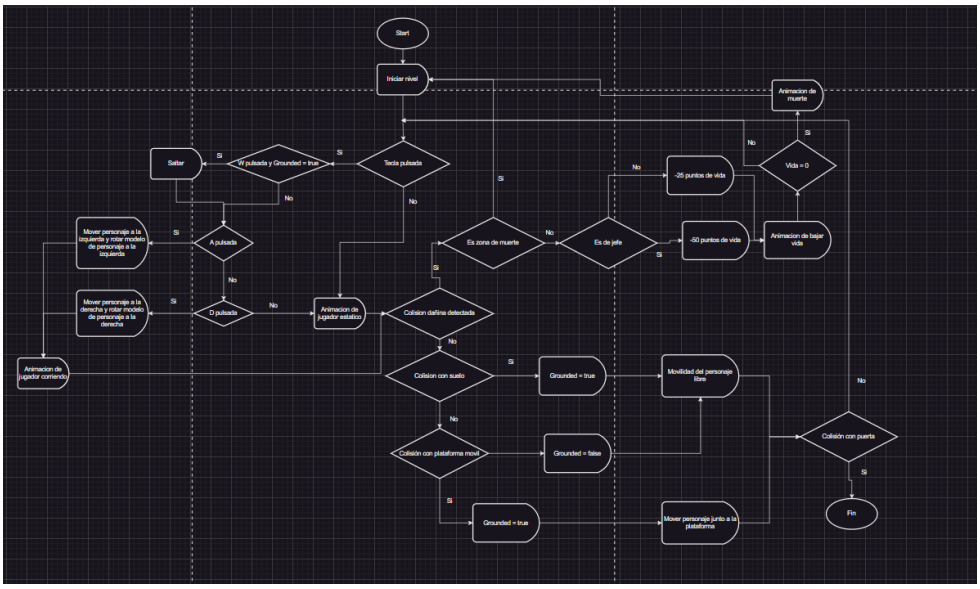


4.3.2 Añadir la movilidad, la vida y el paso de niveles

Crearemos un script donde configuremos una de sus mecánicas principales, el movimiento, junto a su vida y su interacción con el entorno.

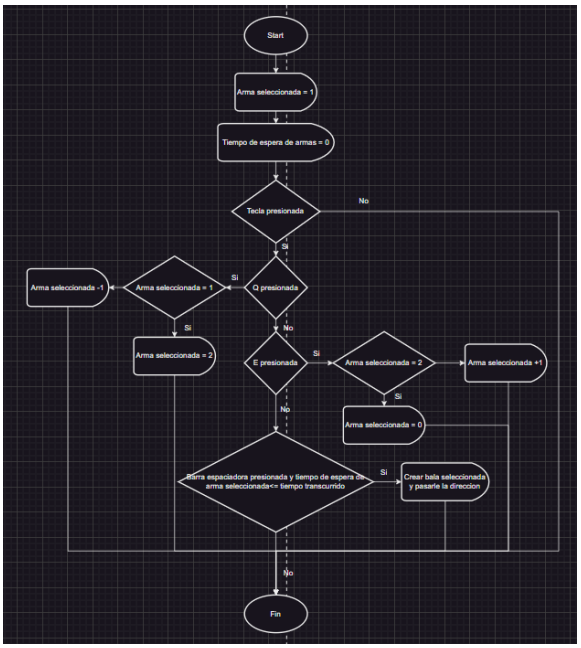
Lo primero es crear Para ello, crearemos un script, lo primero es guardar la vida que tendrá al iniciar cada nivel, la cual sera de 100 puntos, la fuerza del salto, su movimiento horizontal, si esta tocando el suelo, el Animator que este posee, la plataforma móvil si existe en la escena y si esta sobre esta o no, el indicador de vida, el sonido que emitirá al ser golpeado y el nombre del siguiente nivel. Mientras la escena se esté ejecutando si se detecta que se pulsa la A cambiara la velocidad horizontal a -1 moviéndose hacia la izquierda, si se detecta que se a pulsado la D la esta cambiara a ser 1 moviéndose hacia la derecha, y si no se pulsa ninguna de esas la velocidad horizontal será 0 haciendo que permanezca quieto, si la velocidad horizontal es < 0 se cambiará la escala horizontal del personaje a -1 haciendo que este mire a la izquierda, si esta es 1 cambiara la escala a 1 haciendo que mire ha la derecha, luego se le pasará al animator la velocidad horizontal, haciendo que cambie a la animacion de caminar cuando esta es diferente a 0 y que pase a la de estatico cuando es 0. Si detecta que se ha pulsado la tecla W y estas en el suelo , “Grounded = true”, realizara un salto con la fuerza de salto que tiene guardada. Si estas en la plataforma mobil, InMobilePlatform = true, modificara la velocidad horizontal del componente Rigidbody del personaje, haciendo que este se mueva con la plataforma y no se caiga. Cuando el personaje colisiona contra algo, comprobara que es con lo que colisiona. Si detecta el suelo, guardara que que está sobre este cambiando la variable Grounded a true; si lo que detecta que es una bala de un enemigo, este acudira al metodo Dañado pasándole los puntos de vida que pierde, si es la de un minion perderá 25 puntos y si es la del jefe perderá 50, donde le quitaran los puntos de vida, emitira el sonido de quejarse, le pasara al animator del indicador de vida cuanto le queda para que este cambie de animacion acorde con lo que recive y

comprobará si su vida es 0 o menor, y en caso afirmativo se lo indicara al animator del personaje para que este cambie a la animacionde muerte y luego se ejecute el método Die el cual reiniciara el nivel actual; si es la plataforma móvil guardara como que esta sobre esta y también que está sobre suelo, InMobilePlatform = true y Grounded = true; y si es la puerta cargara la siguiente escena cuyo nombre estaba guardado. Cuando detecta que deja de colisionar con algo, comprobara si deja de colisionar con el suelo cambiando Grounded a false impidiendo que este salte hasta que vuelva a entrar en contacto, o si es con la plataforma movil cambiando InMobilePlatform a false, haciendo que este se deje de mover junto a esta.



4.3.3 Añadirle la funcionalida de disparar y el cambio de arma

Ahora lo que haremos es añadirle el disparo. El personaje tiene tres tipos de armas, así que lo primero sera indicarle el numero del arma que tendrá en uso al iniciar cada nivel la cual sera 1 e inicializaremos el tiempo transcurrido desde el anterior disparo, que al iniciar el nivel sera 0 y las diferente balas. Mientras la escena este en ejecución, si detecta que que se ha pulsado la Q cambiara al arma anterior restándole 1 al numero de arma seleccionada, en caso de que el arma seleccionada sea 0 se cambiara a 2 y emitirá un sonido que indica que has cambiado el arma. Si lo que detecta es que se pulso la E en cambiara a la siguiente arma, sumando 1 al numero de arma seleccionada, en caso de que el arma seleccionada sea 2 se cambiara a 0 y emitirá el sonido nuevamente. Si detecta que se ha pulsado la barra espaciadora, comprobara que arma tienes seleccionada y si ha transcurrido el tiempo de espera, en caso afirmativo ejecutara el metodo ShootDamage, donde guardara la direccion en la que quieres disparar según donde este mirando tu personaje y creara la bala correspondiente al arma seleccionada y le dará la dirección.

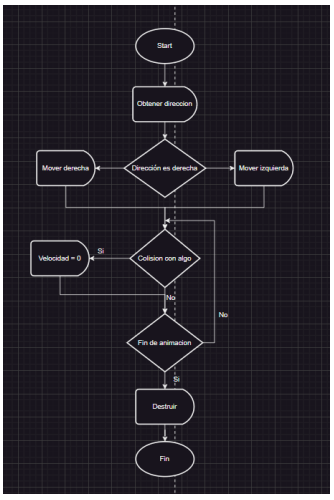


4.4 Tarea 4: Creacion de los proyectiles

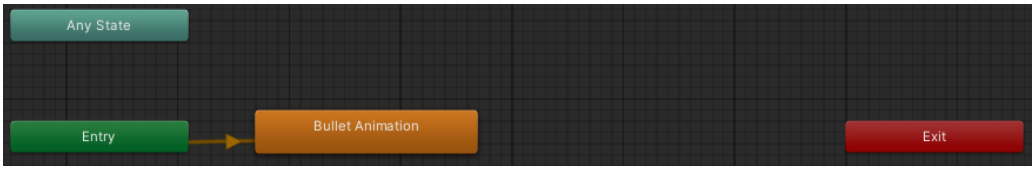
Como los proyectiles son todos prácticamente iguales, dividí al tarea en tres partes, debido a que dos de ellos tienen pequeñas variaciones en su código. Pero antes de todo, como en los sprites descargados solo existen dos balas, la del jefe y la del jugador y los enemigos, lo primero que haremos sera modificar los colores, para cada tipo de proyectil, según el arma que el jugador tenga seleccionada, las de color amarillo serán para la “Standar Weapon” del jugador y para los minions, la azul sera para la “MachineGun” y la roja sera para la “Omega Gun”.

4.4.1 Subtarea 1: Creacion de los proyectiles de los minions enemigos y de la “Standar Weapon” y de la “MachineGun”

Tanto el proyectil de los minions enemigos como de la “Standar Weapon” y de la “MachineGun” del personaje funcionan igual, crearemos un script que nos sirvan para los tres indicados anteriormente, el cual hará que cuando el proyectil sea creado se moverá en una dirección la cual fue especificada por el personaje o el enemigo cuando este lo crea, también haremos este se frene cuando colisione contra algo y que cuando su animación acabe se destruya.

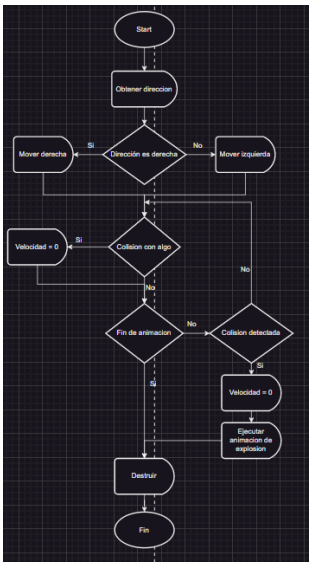


Luego, crearemos las animaciones tal y como creamos las del jugador, separamos los sprites que usaremos añadiéndole la propiedad múltiple y seleccionando las diferentes formas que tendrán con el paso del tiempo, las seleccionamos y creamos una animación por cada proyectil y les añadiremos que ejecute el metodo “Destroy” en el segundo final de la animación, para que una vez que acabe la animación el proyectil se destruya. Posteriormente, crearemos un animator controler por cada proyectil, el cual se encargará de ejecutar la animación creada cuando vayan apareciendo. Con todo esto creado, añadiremos una imagen de cada una de las tres balas y le añadiremos el animator controler correspondiente junto a las colisiones usando un “Circle Collider 2D” y ajustando el tamaño de la colisión al de la bala y les añadimos el script. Por ultimo, guardaremos el objeto resultante como un objeto prefabricado para que el personaje y los enemigos puedan usarlas.

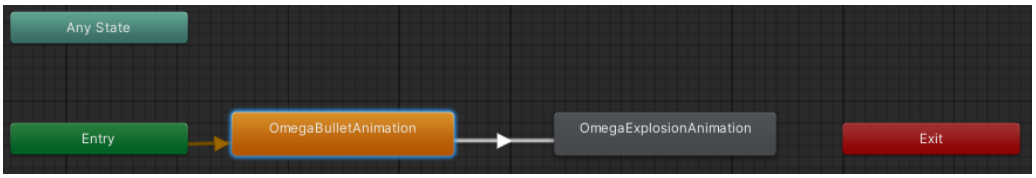


4.4.2 Subtarea 2: Creacion de los proyectiles de la “Omega Gun”

Para este proyectil crearemos un script diferente, ya que a pesar de que el funcionamiento es algo parecido, al tener una animación extra que debe ser iniciada cuando este colisione contra algo, no podemos usar el script de las anteriores. Este, al igual que el anterior, hará que cuando el proyectil sea creado se moverá en una dirección la cual fue especificada por el personaje cuando este la crea y que cuando la animación inicial acabe sin que este haya colisionado con nada este se destruya. Pero si detecta que alcanzó algo ejecutara una animación de explosión en ese mismo instante y cuando esta acabe se destruya.

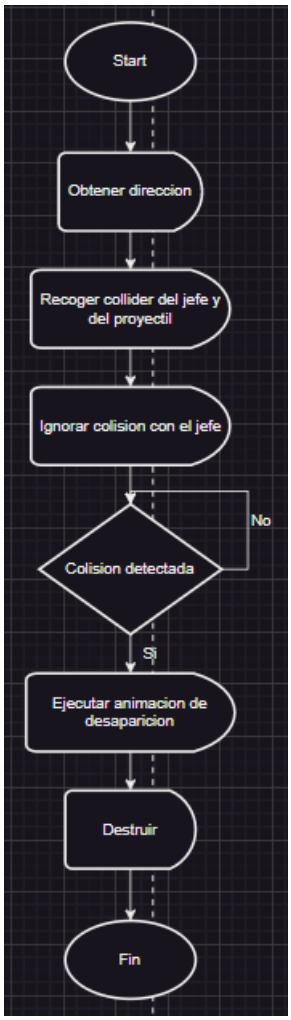


Una vez hecho el script, al igual que con las anteriores balas, crearemos la animación principal y la de la explosión final, y le añadiremos en el ultimo segundo de la animación principal que la bala sea destruida. Luego crearemos un animator controler y le añadiremos ambas animaciones, desmarcando que la transición de una animación a otra se haga cuando la principal se acabe, si no cuando el script le indique que debe cambiar, pudiendo cambiar entre animaciones sin que obligatoriamente tenga que acabar. Con todo esto creado, añadiremos una imagen de la bala y le añadiremos el animator controler junto a la colisión usando un “Circle Collider 2D” y ajustando el tamaño de la colisión al de la bala y les añadimos el script. Por ultimo, guardaremos el objeto resultante como un objeto prefabricado para que el personaje pueda usarla.

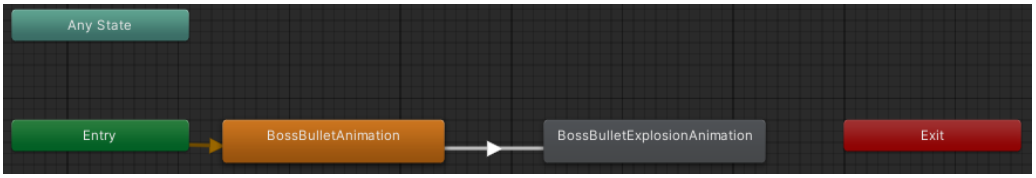


4.4.3 Subtarea 3: Creacion de los proyectiles del jefe

Por ultimo crearemos las balas del jefe, primero crearemos el script, el cual hara que esta se mueva en una dirección dada nada mas ser creada, cuya dirección a diferencia de los anteriores proyectiles, ya que esta dirección sera el vector resultante de unir el centro de la cabeza del jefe, donde este es generado, y como punto final la posición del personaje cuando esta es disparada, y se indicara que ignore las colisiones del jefe, para evitar que esta se quede encerrada en este, y cuando el proyectil detecte una colisión ejecutara la animacion de desaparicion y una vez que esta acabe el proyectil sea destruido, haciendo que este proyectil solo desaparezca cuando choque contra algo, teniendo un rango de ataque infinito.



Con el script acabado, crearemos la animación inicial esta vez usando uno de los sprites de la desaparición de la bala, concretamente el primero, y luego creamos la animación de desaparición usando todos ellos, y añadiremos que ejecute el metodo “Destroy” cuando esta acabe. Crearemos el animator controler como en anteriores casos y le añadimos las dos animaciones creadas y hacemos que cambie la animacion cuando el script lo indique y no cuando acabe la animacion inicial, ya que esta carece de duracion al estar formada por una sola imagen. Por ultimo como siempre, añadimos una bala a la escena y añadiremos las colisiones usando un “Circle Collider 2D” y ajustando el tamaño de la colisión al de la bala, el script y el animator controler y lo guardamos como un objeto prefabricado para que el jefe pueda usarla.

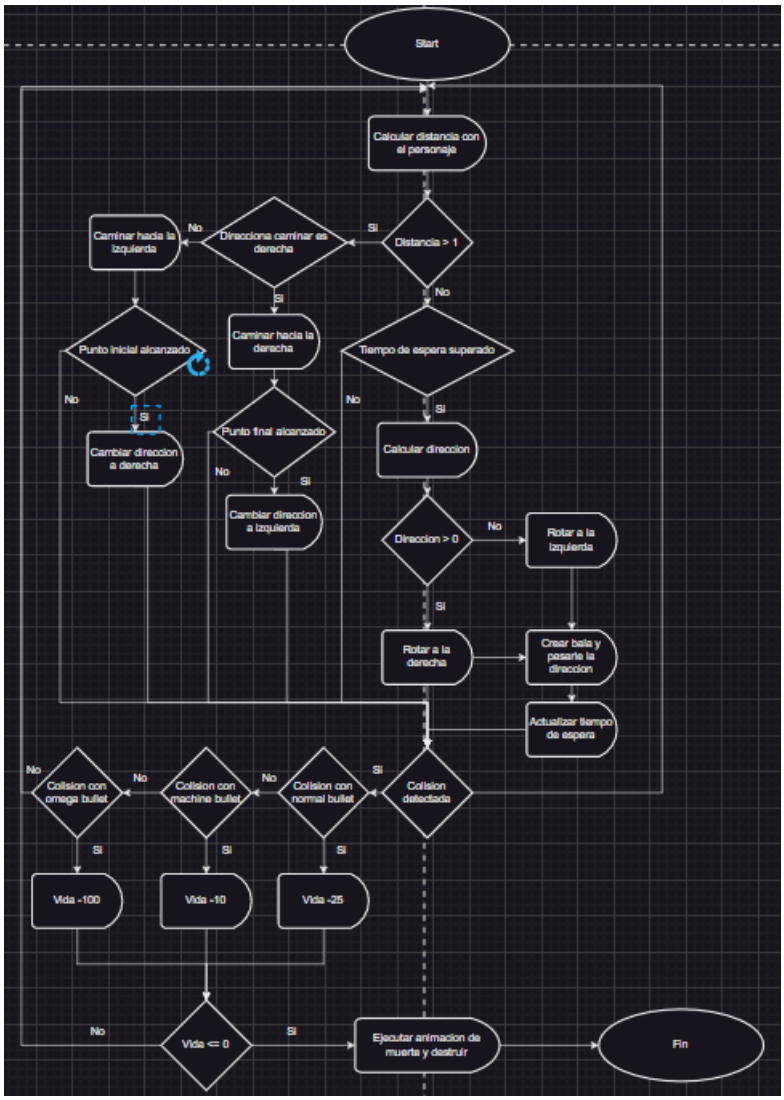


Una vez que tenemos todas la balas guardadas como objetos prefabricados las eliminamos de la escena.

4.5 Tarea 5: Creación de los enemigos

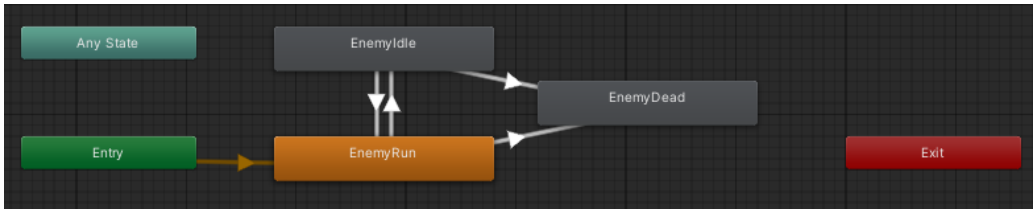
4.5.1 Creacion del script

Crearemos el script de los enemigos para indicarles su comportamiento y gestionar su vitalidad. Al iniciar el nivel, guardará al personaje, su vida que sera 100, su animator, su bala, su posición inicial y su posición final, siendo esta una unidad superior a la de inicio, hacia que dirección se está moviendo y el tiempo que debe esperar entre disparos. Mientras el mapa se este ejecutando, guardara a que distancia se encuentra del personaje y si esta a mas de una unidad de el cambiara a la animación de carrera y comprobara hacia donde se estaba moviendo, si estaba caminando hacia la derecha cambiara su escala horizontal a 1 mirando a la derecha y se desplazara hasta que su posición sea igual o mayor a la posición final y cambiara de sentido, si se estaba moviendo hacia la izquierda cambiara su escala horizontal a -1 mirando a la izquierda hasta que su posicion sea igual o menor a la posición inicial. Si la distancia con el personaje es menor o igual a la unida y el tiempo de espera ha pasado, calculara el vector que indica la dirección a la que esta, si el vector en el eje horizontal es menor que 0 rotara y mirará a la izquierda, si es mayor que 0 rotará mirando a la derecha y ejecutara el método Shoot, el cual btiene la direccion del disparo y creara la bala que tiene guardad y le pasará la dirección. Si detecta una colisión comprobara con que colisionó, si fue con un proyectil de la “Standar Weapon” este perderá 25 puntos de vida, si fue con uno de la “MachineGun” este perderá 10 puntos y si fue con uno de la “Omega Gun” perderá 100, para luego comprobar si tiene 0 o menos puntos de vida, en caso afirmativo ejecutara la animación de muerte y cuando esta acabe, se ejecutara el metodo Muerte que lo destruye.



4.5.2 Creacion de las animaciones

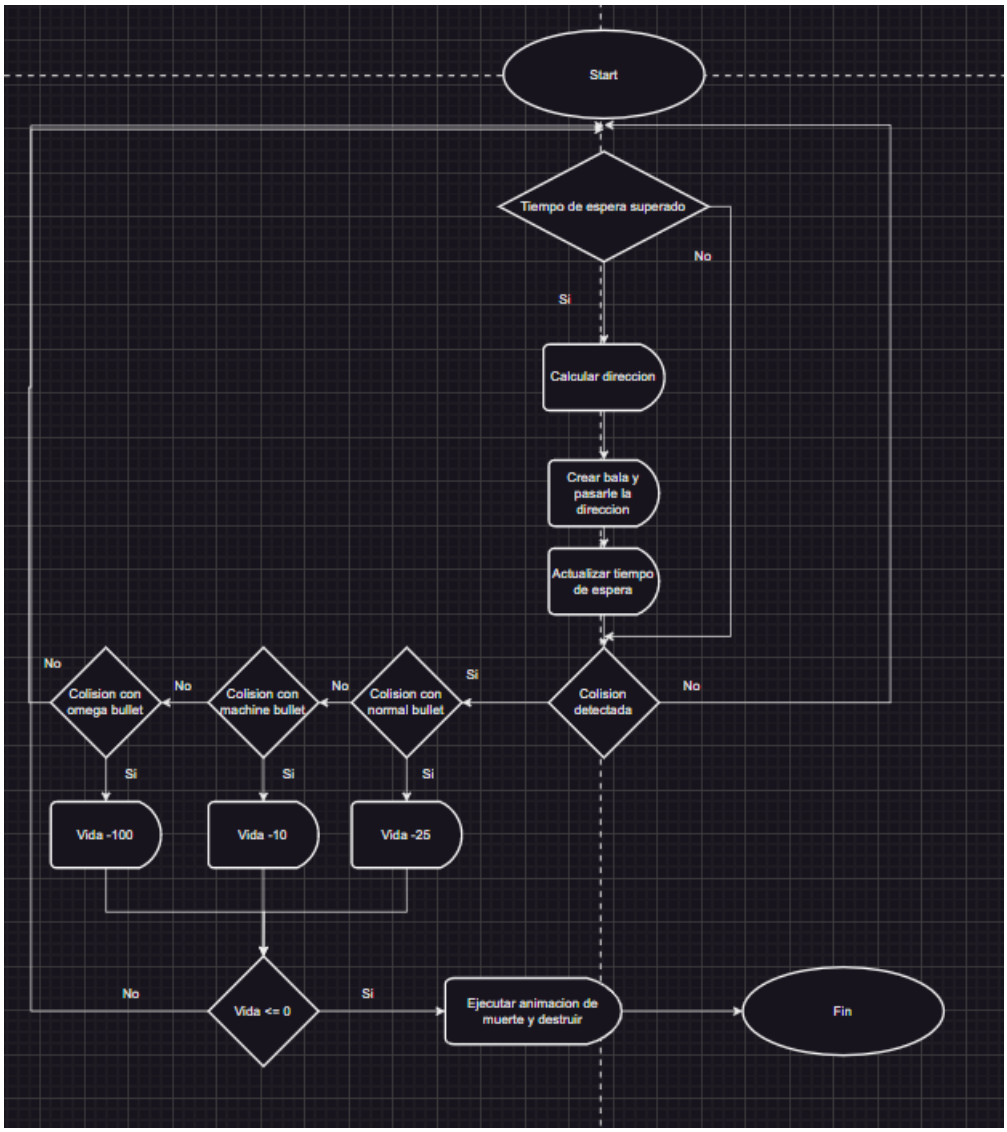
Ahora con el script hecho crearemos las animaciones, como en casos anteriores, separamos las diferentes posturas de que conforman las animaciones dandoles el atributo multiple a las imagenes que las contienen y con todas seleccionas crearemos la animación de estar estatico, la de carrera y la de muerte, añadiendola que en el ultimo segundo ejecute el metodo Muerte, que lo destruirá, y pondremos que la de estatico y la de carrera se ejecuten en bucle. Una vez tenemos todas las animaciones creamos el animator controler y añadimos las animaciones, indicaremos que para que cambie a ejecutar la animación de carrera el script debera indicarle que está en movimiento y que cuando se pare el script se le indique al animator controler y ejecute la animacion de estático y que cuando su vida baje a 0 el script le indicara que ejecute la animación de muerte, habiendo deshabilitado el cambio entre animaciones por tiempo, pudiendo cambiar entre estas sin importar que no hayan acabado. Por ultimo le damos las colisiones con un “Capsule Collider 2D”, fisicas con el “Rigidbody 2D” y le añadimos el animator controler, para luego guardarlo como prefabricado.



4.6 Creacion del jefe

4.6.1 Crear el script

El script del jefe al iniciar el nivel, guardara el personaje, la vida, que sera de 200 y su animator. Mientras el mapa se está ejecutando, se comprobara si ha pasado el tiempo de espera, en caso afirmativo ejecutara el método Shoot, el cual calculara la dirección que sera el vector resultante de unir el centro de la cabeza del jefe, donde este es generado, y como punto final la posición del personaje cuando esta es disparada y generará un proyectil al que le pasará la dirección. Si detecta una colisión comprobara con que colisionó, si fue con un proyectil de la “Standar Weapon” este perderá 15 puntos de vida, si fue con uno de la “MachineGun” este perderá 7 puntos y si fue con uno de la “Omega Gun” perderá 30, para luego comprobar si tiene 0 o menos puntos de vida, en caso afirmativo ejecutara la animación de muerte y cuando esta acabe, se ejecutara el metodo Die que lo destruye.



4.6.2 Crear las animaciones

Para el jefe solo crearemos dos animaciones, como hemos hecho anteriormente, separamos las diferentes posturas de que conforman las animaciones dandoles el atributo multiple a las imagenes que las contienen y con todas seleccionas crearemos la animación de estar estático, la cual haremos que se ejecute en bucle, y la de muerte, añadiéndola que en el ultimo segundo ejecute el método Die. Luego creamos el animator controler, añadimos las dos animaciones y hacemos que pase de la estática a la de muerte solo cuando el script le diga que su vida bajo a 0.

	Duración (horas)	**Fecha inicio**	**Fecha fin**	**Responsable**
Tarea 1	2	25/03/2023	25/03/2023	Jorge Rodríguez
Tarea 2	12	26/03/2023	03/04/2023	Jorge Rodríguez
Tarea 3	36	05/04/2023	29/04/2023	Jorge Rodríguez
Tarea 3.1	4	05/04/2023	05/04/2023	Jorge Rodríguez

Tarea 3.2	17	06/04/2023	16/04/2023	Jorge Rodríguez
Tarea 3.3	15	22/04/2023	29/04/2023	Jorge Rodríguez
Tarea 4	6	01/05/2023	02/05/2023	Jorge Rodríguez
Tarea 4.1	3	01/05/2023	01/05/2023	Jorge Rodríguez
Tarea 4.2	1	02/05/2023	02/05/2023	Jorge Rodríguez
Tarea 4.3	2	02/05/2023	02/05/2023	Jorge Rodríguez
Tarea 5	18	06/05/2023	20/05/2023	Jorge Rodríguez
Tarea 5.1	16	06/05/2023	15/05/2023	Jorge Rodríguez
Tarea 5.2	2	20/05/2023	20/05/2023	Jorge Rodríguez
Tarea 6	12	21/05/2023	28/05/2023	Jorge Rodríguez
Tarea 6.1	10	21/05/2023	27/05/2023	Jorge Rodríguez
Tarea 6.2	2	28/05/2023	28/05/2023	Jorge Rodríguez

5 RECURSOS HUMANOS

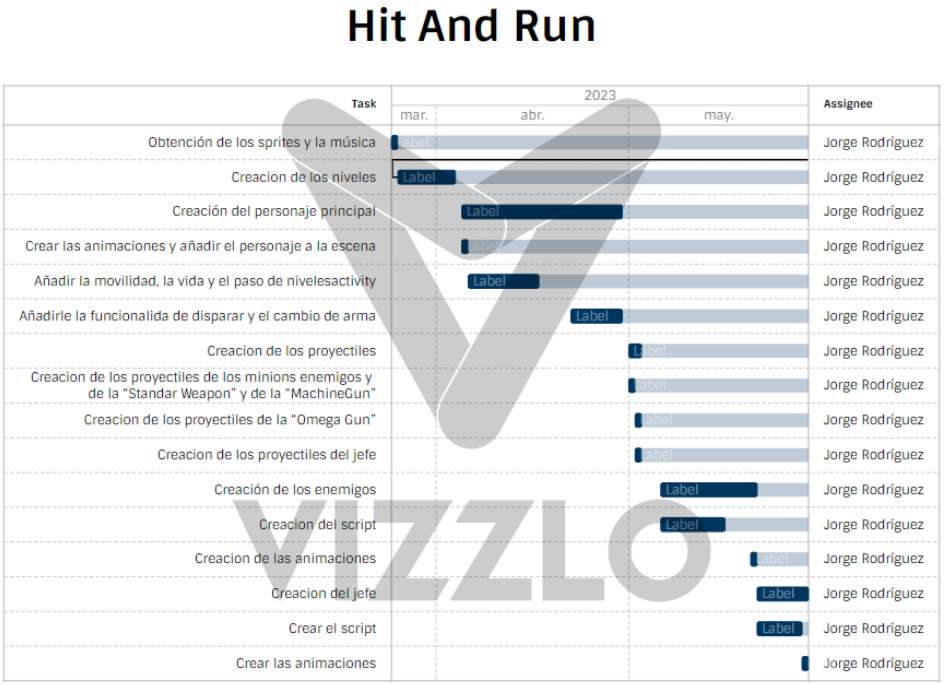
Este proyecto lo he realizado en solitario encargándome de todo y siendo el responsable de todo.

6 RECURSOS MATERIALES

Para poder realizar este proyecto he necesitado unos assets, donde venían los sprites y los efectos de sonido, y todos ellos han sido obtenidos de manera gratuita en una pagina de internet donde podías descargarlos, aunque algunos sprites como algunos proyectiles y el indicador de vida han sido creados por mí con una página de edición grafico totalmente gratuita. También he requerido del uso de un repositorio el cual era totalmente gratuito, aunque tienen otras dos opciones con ciertos beneficios las cuales son de pago, la versión teams la cual cuesta 3.67$ y 19.25$, las cuales podrian ser útiles en el caso de que este proyecto siga creciendo.

Para la creación del videojuego he usado UnityHub, el cual es de uso totalmente gratuito, aunque en la tienda de assets que hay, existen algunos que pueden costar algo. Y apra programar los diversos script he usado Visual Studio 2022 Comunity la cual es gratuita, aunque esta es recomendada para uso individual, para las empresas estan la versión Profesional que cuesta 45$ y la versión Enterprise que cuesta 250$, las cuales poseen mas ventajas que la version que yo uso.

7 CRONOGRAMA



8 PRESUPUESTO

	Recursos humanos	Recursos materiales	Total
Tarea 1	0€	0€	0€
Tarea 2	0€	0€	0€
Tarea 3	0€	0€	0€
Subtarea 3.1	0€	0€	0€
Subtarea 3.2	0€	0€	0€
Subtarea 3.3	0€	0€	0€
Tarea 4	0€	0€	0€
Subtarea 4.1	0€	0€	0€
Subtarea 4.2	0€	0€	0€
Subtarea 4.3	0€	0€	0€
Tarea 4	0€	0€	0€
Subtarea 4.1	0€	0€	0€
Subtarea 4.2	0€	0€	0€
Tarea 5	0€	0€	0€
Subtarea 5.1	0€	0€	0€
Subtarea 5.2	0€	0€	0€

Total:	0€	0€	0€

9 ANEXOS

Este estilo de juegos se ven cada vez menos en ordenadores y videoconsolas, pero son muy jugados actualmente en moviles, debido al bajo consumo de estos, los cuales hacen que sean mas ligeros y llevaderos para un dispositivo de potencial limitado como son los telefonos en comparación con el resto. Adaptar este videojuego para poder ser usado en un teléfono movil no seria muy complejo, ya que se pueden aprovechar muchas cosas del original, como por ejemplo:

- **Lógica del juego**: Puedes reutilizar gran parte de la lógica del juego existente. Esto incluye la mecánica de juego, la lógica de colisiones, el progreso del jugador, la inteligencia artificial, entre otros. La mayoría de la lógica subyacente del juego puede ser independiente de la plataforma.
- **Arte y diseño**: Puedes reutilizar los elementos de arte y diseño del juego, como sprites, texturas, animaciones y efectos visuales. Asegúrate de optimizar los recursos gráficos para adaptarse a las capacidades y restricciones de los dispositivos móviles.
- **Sonidos y música**: Si tienes elementos de sonido y música en tu juego, puedes reutilizarlos en la versión para dispositivos móviles. Sin embargo, ten en cuenta que los dispositivos móviles pueden tener altavoces y características de sonido diferentes, por lo que es posible que debas ajustarlos y optimizarlos según sea necesario.
- **Niveles y escenarios**: Si tu juego tiene diferentes niveles o escenarios, puedes reutilizarlos en la versión para dispositivos móviles. Sin embargo, ten en cuenta que los tamaños de pantalla y las relaciones de aspecto pueden ser diferentes en dispositivos móviles, por lo que es posible que necesites ajustar la disposición y el diseño de los niveles para una experiencia óptima en dispositivos móviles.

Aunque también se deberán hacer algunos cambios como serían:

- **Controles táctiles**: En lugar de utilizar controles de teclado o ratón, debes implementar controles táctiles para que los jugadores puedan interactuar con el juego en dispositivos móviles. Puedes usar elementos como botones virtuales, joysticks virtuales u otros gestos táctiles, dependiendo de las necesidades del juego.
- **Resolución y relación de aspecto**: Los dispositivos móviles tienen diferentes resoluciones y relaciones de aspecto en comparación con las computadoras de escritorio. Asegúrate de ajustar los elementos del juego y la interfaz de usuario para adaptarse a diferentes tamaños de pantalla y proporciones. Puedes utilizar anclajes y sistemas de diseño adaptativo para lograr una visualización adecuada en diferentes dispositivos.
- **Rendimiento y optimización**: Los dispositivos móviles pueden tener limitaciones de rendimiento en comparación con las computadoras de escritorio. Es importante optimizar el juego para asegurarte de que se ejecute sin problemas en dispositivos móviles. Considera reducir la cantidad de polígonos, limitar los efectos visuales complejos y optimizar el uso de recursos, como texturas y materiales.
- **Interfaz de usuario**: Es posible que debas ajustar la interfaz de usuario para que sea más amigable para pantallas táctiles. Asegúrate de que los botones y elementos de la interfaz de usuario sean lo suficientemente grandes y espaciados para que los jugadores puedan tocarlos fácilmente con los dedos.

Otro de los problemas podría ser la disminución de feedback al usuario, ya que cuando juegas en una videoconsola o en un ordenador, notas cuando haces algo no solo porque lo ves en la pantalla sino porque también notas el botón siendo presionado, en contra posicion con los juegos de movil ya que para interactuar retriegas tus dedos por la pantalla, por eso habría que añadirle mas respuesta que te indique que realizastes una accion.

10 BIBLIOGRAFÍA

Assets: https://didigameboy.itch.io/jambo-jungle-free-sprites-asset-pack

Repositorio GitHub: https://github.com/Jorge-Rodriguez-Folguera/HitAndRun.git

Pixlr: https://pixlr.com/es/e/