Universitat Oberta de Catalunya

uoc.edu

Codebreakers: A platform-action game as a Metroidvania base.

Jaime Martínez Villar
Máster en Diseño y Desarrollo de Videojuegos
Diseño de una demo jugable

Helio Tejedor Navarro
Joan Arnedo Moreno

Junio de 2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Codebreakers:* A platform-action game as a *Metroidvania* base.
Nombre del autor:	*Jaime Martínez Villar*
Nombre del consultor/a:	*Helio Tejedor Navarro*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega (mm/aaaa):	05/2020
Titulación:	*Máster en Diseño y desarrollo de videojuegos*
Área del Trabajo Final:	*Diseño de una demo jugable*
Idioma del trabajo:	*Castellano*
Palabras clave	*Demo, 2D, Plataformas*

Resumen del Trabajo (máximo 250 palabras):

Una pequeña demo para exponer varios de los conocimientos obtenidos durante los cursos enfocado en las mecánicas de un juego tradicional de plataformas y acción en 2D.

Exposición de comportamiento de diferentes enemigos y desarrollo de assets (básicos) para personaje y enemigos con animaciones 2D mediante huesos (*bones*).

Sirve de demostración del aprendizaje en el diseño, desarrollo y programación de videojuegos en motores gráficos gratuitos (*Unity*) sin contar con un trasfondo base de programador. Una puerta de entrada al mundo de la industria del videojuego.

Abstract (in English, 250 words or less):

A small demo to show the knowledge and skills obtained through these two years; focusing in basic mechanics of traditional platform and action 2D games.

Exposure of different enemies' behaviour and basic asset developing for player

character and enemies alike with 2D bone animations.

It works as a demonstration of the knowledge acquired at developing, designing and programming videogames in free game engines (*Unity*) lacking a programmer background. An entry door to the world of videogame developing industry.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

El proyecto nace cómo una idea durante las primeras asignaturas del máster (*Game Design*) y que se expandió durante el 3º Semestre con *Experiencia de usuario e interfaces; Media para videojuegos* y *Modding y creación de niveles.*

Cómo todo proyecto era un castillo en el aire y demasiado ambicioso para llevarlo a cabo en el momento; por lo que habría que decaparlo, y así es como empezó este trabajo; una vista de las mecánicas base y el desarrollo de niveles básicos; así como del interfaz completo.

Al final, nace como prueba a uno mismo para exponer todo lo que se ha aprendido y como cimiento de ese proyecto a futuro.

1.2 Objetivos del Trabajo

Al comienzo del proyecto se establecieron unos objetivos principales, a medida que avanzaba el proyecto han ido modificándose para adaptarse a las entregas; pero lo principal se ha mantenido invariable:

1.- Programación de mecánicas de un juego plataformas de acción clásico; centrándose en el entretenimiento del jugador.

2.- Diseño de niveles mediante *Tilemap* al estilo de juegos clásicos.

3.- Creación de *Assets* básicos (*placeholders*) para personajes y enemigos en 2D; mediante el uso de aplicaciones de edición de imagen (GIMP) y animaciones mediante huesos.

4.- Diseño de una pelea contra un jefe final (*boss fight*) con diferentes etapas y fases.

1.3 Enfoque y método seguido

Ya se ha establecido que el enfoque de este trabajo es exponer todo lo desarrollado en los dos últimos años y asentar una base mecánica de lo que sería un proyecto mayor.

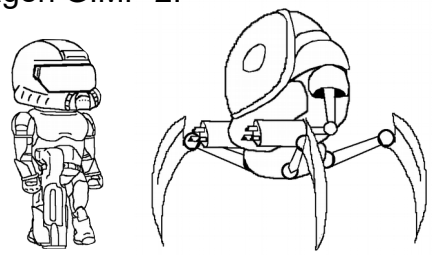
Dado que el proyecto a futuro que se plantea es algo más enfocado al género *Metroidvania* se explora en aquí sus mecánicas base: Niveles por habitaciones, diversos enemigos con distintos tipos de ataques y comportamientos; peleas contra jefes finales…

Aunque se pierde la visión de “mejora” y desarrollo del personaje característicos del género, se pensó que era mejor cementar los cimientos y preparar esta demo sobre la base del juego: Mecánicas y entretenimiento.

1.4 Planificación del Trabajo

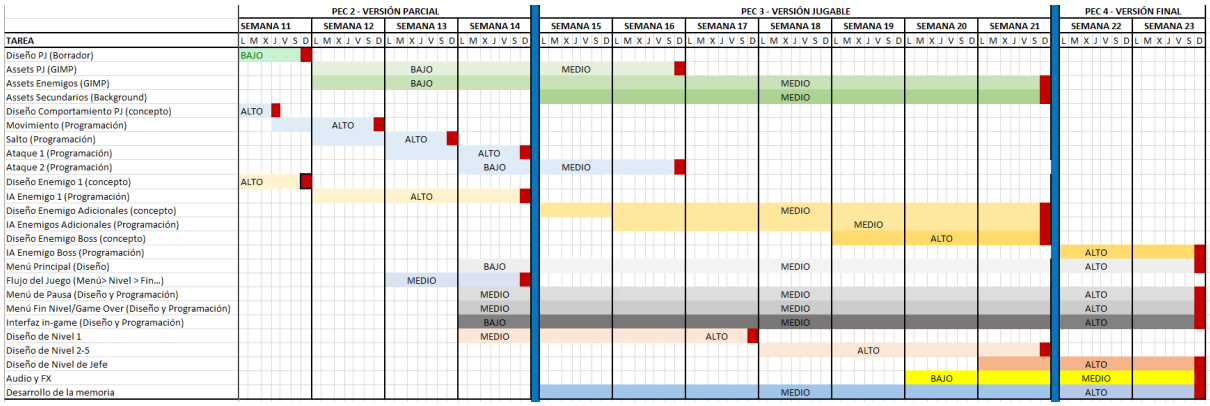
Para realizar el proyecto se ha elegido un motor gráfico gratuito y libre cómo *Unity*, tanto por su posibilidad de ser multiplataforma, como por ser uno de los motores con más información y documentación online.

Además, para realizar los *assets* básicos se ha utilizado el programa de edición de imagen GIMP 2.



Imágenes del personaje y un enemigo

Además, se planificó un pequeño diagrama de Gantt para planificar las tareas a realizar en las distintas entregas.



1- Gantt del Proyecto

Durante la *Versión Parcial* se buscaba tener el concepto y el desarrollo de los primeros *assets*: jugador y un enemigo; con los que poder probar y desarrollar las mecánicas, inteligencia artificial y animaciones base.

Para la *Versión Jugable* ya se buscaba tener el grueso del proyecto realizado; todo el diseño de niveles, y tener ya varios enemigos distintos entre sí que el jugador sepa identificar de un vistazo y saber cómo actuar en cada caso.

Lo único que dejamos para la *Versión Final* sería terminar la parte del *boss* final y terminar la parte artística, gráfica y audio, así como el testeo del juego ya completo.

1.5 Breve sumario de productos obtenidos

Con este proyecto hemos obtenido no sólo una demo jugable que servir de base para un futuro proyecto mayor sino, además, multitud de *assets* propios que puedan ser reutilizados en el futuro.

Sprites (*placeholders*) de enemigos y personajes, imágenes de interfaz, logo… Todo ello puede ser reutilizado en el futuro.

1.6 Breve descripción de los otros capítulos de la memoria

A continuación, se expondrá los contenidos de cada capítulo de la memoria.

- **Capítulo 2: Estado del arte.** Una pequeña descripción de la industria, concretamente del género que envuelve al proyecto.

- **Capítulo 3: Definición del juego.** Un pequeño documento sobre los detalles del juego: género y referencias a otros videojuegos del estilo; inspiraciones, controles e indicación de interacción juego-jugador. Descripción de la ambientación y de los personajes y elementos que componen el juego.
  Concept Art y bocetos del diseño de los elementos gráficos.

- **Capítulo 4: Diseño Técnico.** Justificación del *Game Engine* utilizado y herramientas, tanto internas como externas, utilizadas.
  Inventario de *Assets* y recursos y un pequeño esquema de la arquitectura del juego y de la Inteligencia Artificial.

- **Capítulo 5: Diseño de Niveles.** Justificación y criterios del desarrollo del diseño de los niveles.

- **Capítulo 6: Manual de Usuario.** Un breve documento sobre requerimientos técnicos e instrucciones de juego.

- **Capítulo 7: Conclusiones.** Conclusiones del proyecto; autoanálisis crítico de los objetivos y planificación y del estado actual y objetivos de futuro del proyecto.

- **Capítulo 8: Glosario.** Descripción de los términos y acrónimos utilizados dentro del documento.

2. Estado del arte

Para analizar el proyecto debemos enfocarnos en dos géneros; por una parte, lo que quiere llegar a ser, y por otra la que es.

Se trata de un videojuego del género *plataformas* con elementos de *acción* en 2 dimensiones; al más puro estilo de clásicos como *Metal Slug* o *Megaman,* acercándose a ese enfoque arcade y arraigando en la rama del entretenimiento.

Los videojuegos han ido evolucionando a lo largo de los años; cada vez centrándose más en la narrativa, el mensaje y la cinemática, haciendo de estos un arte variado en contenido y forma. [1]

En la actualidad, podemos encontrar numerosas obras que se enfocan en una u otra vertiente, y el género al que pertenece el videojuego suele ser indicativo de ello.

Las *aventuras gráficas* y, especialmente, los más recientes "*walking simulators*", tales como *What Remains of Edith Finch* o *Firewatch,* así como algunos juegos mucho más cinemáticos y de mayor producción como la saga *Metal Gear* de Hideo Kojima o los varios títulos de David Cage, se enfocan principalmente en la narrativa, esa historia detrás del videojuego y el mensaje que el autor quiere dejar en el jugador.

Aunque sus mecánicas puedan estar a la altura (o no) de su narrativa, está claramente enfocados en transmitir una historia o una moraleja. [2]

Por otro lado, los arcades fueron perdiendo fama durante los años 2000 y principios de la década; pero nunca desaparecieron; puesto que siempre tienen su lugar en el mercado: Los juegos como entretenimiento.

Todos los años encontramos un nuevo *Super Mario* (Nintendo) o *Call of Duty* (Activision); pero también en el mundo del desarrollo independiente o *indies* aparecieron numerosos juegos enfocados principalmente en su jugabilidad y en el entretenimiento tales como *Rogue Legacy*, *Dead Cells*, *Cuphead*…

Videojuegos que existen con la única meta del entretenimiento, la historia o marco del mismo es una mera excusa para adentrarnos en su mundo y jugar, olvidándonos de lo que nos rodea para pasar un buen rato sin más.

En este caso, *Codebreakers*, como juego de plataformas y acción, se parecería a estos últimos, fuertemente inspirado en el estilo de otros juegos del género cómo el reciente *Megaman 11* o, el anteriormente mencionado, *Cuphead.*

Por otro lado, el proyecto a futuro de *Codebreakers* entraría más en el género del *Metroidvania,* añadiendo elementos de desarrollo y mejoras del personaje; mapas interconectados y un guion que sirva de marco y contexto para todo lo ocurrido; asimilándose más a otros clásicos como los *Metroid* o *Castlevania* o los más recientes *Hollow Knight* y *Rogue Legacy.*

Ambos géneros son bien recibidos en multitud de plataformas: PC, consolas, consolas portátiles… y por ello un motor multiplataforma como Unity era idóneo para ello.

Sus controles suelen ser simples; pero es muy importante que sea fluido; especialmente para el movimiento. Lo más importante es entretener al jugador y no frustrarlos por errores en movimientos imprecisos.

Se basan en que son fácilmente accesibles, pues los *plataformas* son de los géneros más antiguos que existen dentro de los videojuegos; pero añadiendo el elemento de *acción* podemos añadir un *plus* de dificultad y variabilidad para los más veteranos. [3]

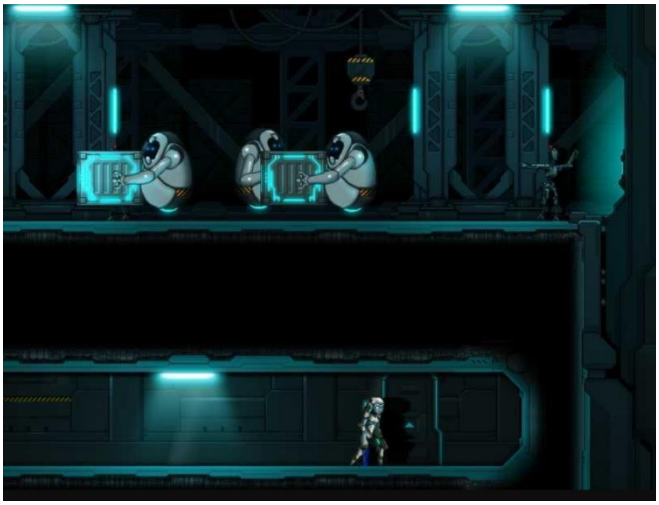
3. Definición del Juego

3.1 Subgénero y Referencias

Dentro de los juegos de Plataformas tenemos infinidad de ejemplos a los que recurrir; cómo el clásico *Super Mario*, o el afamado *Super Meat Boy*.

2- *Super Meat Boy Forever* (Izquierda), *Super Mario Bros. U Deluxe* (Arriba).

Sin embargo, para nuestro caso añadiremos un poco más de acción y menos precisión mecánica (*Super Meat Boy*), asemejándolo a lo que sería el estilo de un *Metroidvania* como *Dead Cells* o *Ghost 1.0*, si nos fijamos en el ámbito nacional.



3- *Dead Cells* (Arriba), *Ghost 1.0* (Derecha)

3.2 Tipo de Interacción Juego-Jugador

Nuestras mecánicas base serán el movimiento y la acción. Al tratarse de un juego en 2D podremos movernos hacia delante y atrás (derecha e izquierda) así como hacia arriba en forma de salto.

Para la acción dispondremos de dos acciones; un golpe cuerpo a cuerpo y un ataque a distancia. Será importante facilitar al jugador el atacar mientras se mueve rápidamente por el nivel.

3.3 Historia, Ambientación y/o Trama

En su concepto original nos encontraríamos en un futuro dominado por la avaricia de las grandes compañías, *Nova Enterprise* nace como desarrolladora de la gran revolución robótica. Como muchas de esas grandes corporaciones, *Nova Enterprise* oculta multitud de secretos; entre ellos su nueva y potente **Inteligencia Artificial**.

Durante el inicio del juego, el tutorial, haríamos un par de sesiones de niveles simples para explicar los controles. Es aquí donde vamos a enfocar nuestra ambientación para la implementación de nuestro juego; es decir, se ambientará dentro de una simulación donde el jugador deberá probar distintos “prototipos” de androides; cada uno con unas características distintas, proporcionando variedad en los puzles y retos que se le presentan al jugador.

Todo este marco no se presenta en el juego, dado que se ha decidido prescindir de ningún tipo de guion o presentación para dejar una demo que vaya directo a la acción.

3.4 Definición de los Personajes / Elementos

Para esta iteración de la idea, contaremos principalmente con nuestro personaje principal:

Nova: Una, supuesta, Inteligencia Artificial desarrollada por *Nova Enterprise* para ser usada como herramienta de testeo en el campo de pruebas de simulación.

Nova Enterprise: Será la que nos ponga a prueba; nos encontraremos en su mundo de "niveles" simulados que deberemos superar.

Los *Administradores*: Serán los distintos *jefes finales* (*bosses*) a los que nos enfrentaremos; representan creaciones controladas por expertos de Nova Enterprise.

Zachary Lee: Creador de la de *Terra-Nova*, y programador jefe de *Nova Enterprise.* Será el último *Administrador* y nuestro *jefe final.* (NO PRESENTE EN LA DEMO)

3.5 Interacción entre los Actores del Juego

Cómo hemos definido anteriormente; Nova será nuestro avatar en el juego, mientras que las creaciones de Nova Enterprise y los distintos *Administradores* serán los enemigos que nos encontraremos en los distintos niveles del juego.

Además, la propia Nova Enterprise sirve de marco y ambientación para los distintos niveles. Desde ciudadelas simuladas, a salas de servidores, laboratorios de robótica y demás. Claramente con una fuerte temática futurista.

3.6 Objetivos Planteados al Jugador

El juego será una sucesión de niveles, cada vez más difíciles, donde deberemos alcanzar la meta de este.
En muchos casos, entre nuestro objetivo y nosotros habrá enemigos u otros peligros de los que deberemos hacernos cargo.

Algunos niveles especiales serán los niveles de *jefes*, Los Administradores. Para estos niveles se diseñará una pequeña batalla al estilo de las de juegos como el excelentísimo *metroidvania Hollow Knight.*

4- Pelea contra el *Falso Caballero* (*False Knight*), en *Hollow Knight.*

3.7 Concept Art

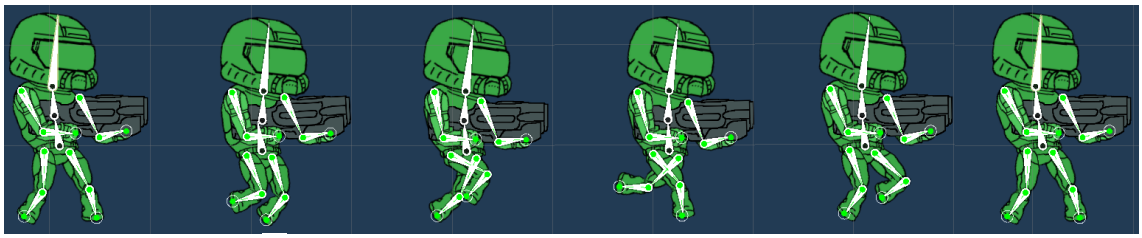
Para el estilo artístico del juego nos basaremos mucho en *Assets* futuristas; puede que quede con un concepto similar al estilo de *Ghost 1.0.*

Para realizar las animaciones pertinentes recurriremos a animación mediante huesos o *bone animation* muy similar a la que se puede ver en *Scribblenauts,* donde cada parte del cuerpo de los personajes está definida por un *sprite* asociados a un esqueleto que mueve los *sprites* del mismo modo que se haría en animación 3D (*Skinning*) [4]

5- Personajes en *Scribblenauts*

De ese modo se facilita mucho el proceso de animación; puesto que de otro modo tendríamos que hacer una sucesión de *sprites* completos del

personaje para cada animación y cada personaje; una tarea titánica para llevarla a cabo en tan solo unos meses.



6- Animación de movimiento mediante *bone animation* del personaje.

4. Diseño técnico

4.1 Evaluación de *Game Engines* y Kits de desarrollo

Ya hemos explicado a grandes rasgos porque elegimos *Unity* como motor gráfico para nuestro proyecto; y vamos a exponer los detalles ahora.

Para nuestro criterio, el motor a utilizar debía cumplir con tres requisitos indispensables:

1- Debe ser libre y gratuito

2- Flexible y sencillo (*user-friendly*), con documentación extensa a la que poder recurrir.

3- Debe tener funcionalidad para juegos 2D

Hay que destacar que mi *background* cómo programador es escaso; dado que ni siquiera provengo de una rama de informática; sino de electrónica y automatización, por lo que un motor que requiera conocimientos profundos queda descartado; así como crear un motor propio; pues el tiempo para realizar el prototipo es escaso.

Durante los últimos años hay varios motores de fácil acceso y con grandes funcionalidades. Para el caso que nos atañe recurriremos a los dos más accesibles de los últimos años: Unity y Unreal.

7- Ejemplo de juego 2D en *Unity*

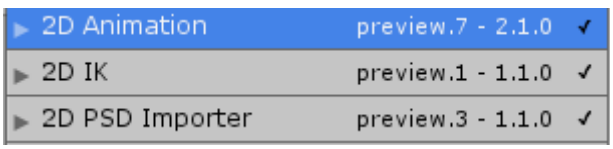
Ambos son gratuitos; flexibles y tienen soporte para juegos 2D; sin embargo, y tras dos años habiendo estado trabajando con Unity en el máster de la Universidad Oberta de Catalunya (UOC), utilizaremos este mismo al estar ya acostumbrado a sus opciones, funcionalidades y programación.

4.2 Herramientas empleadas

Dentro de *Unity* se han utilizado varias herramientas concretas para ciertas partes del desarrollo; además de usar algunas herramientas externas para desarrollar los *assets* necesarios.

Para las animaciones, hemos recurrido a uno de los paquetes en desarrollo de *Unity:*

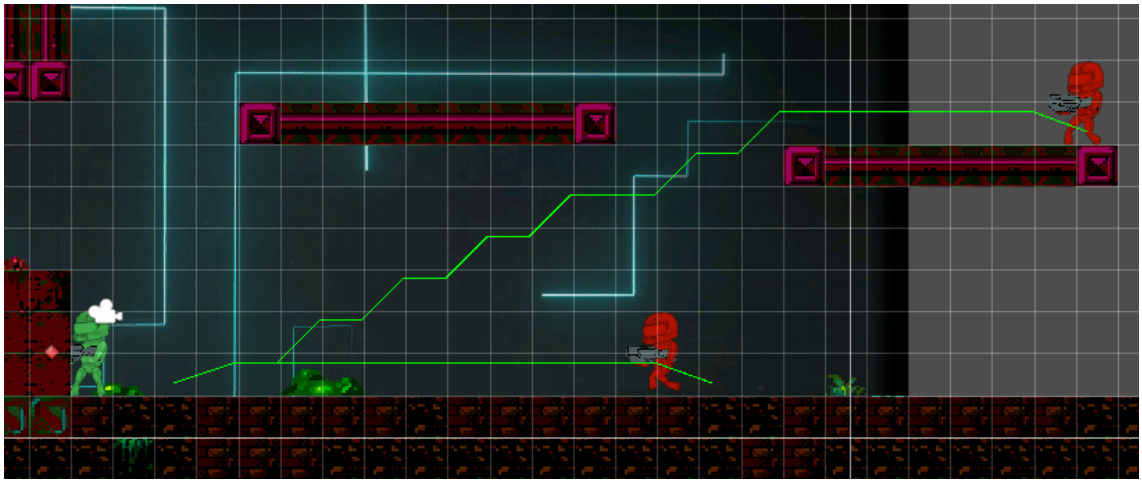
- **2D Animation** (*preview.7 – 2.1.0*): Para las herramientas necesarias para animación con esqueletos utilizando *Sprites*.
- **2D IK** (*preview.1 – 1.1.0*): Herramientas necesarias para preparar los personajes 2D para funcionar con *inverse kinematics* (*IK*).
- **2D PSD Importer** (*preview.3 – 1.1.0*): Una pequeña herramienta para importar archivos de imagen *Adobe Photoshop* en formato .psd para crear personajes multi-*sprite* y animaciones mediante el paquete *2D Animation.*



8- Paquetes de recursos de *Unity* para animaciones 2D.

Para la Inteligencia Artificial se ha añadido un paquete de la *AssetStore* para controlar el sistema de *pathfinding* en 2D: ***The A* Pathfinding Project*** de Aron Granberg (http://www.arongranberg.com).

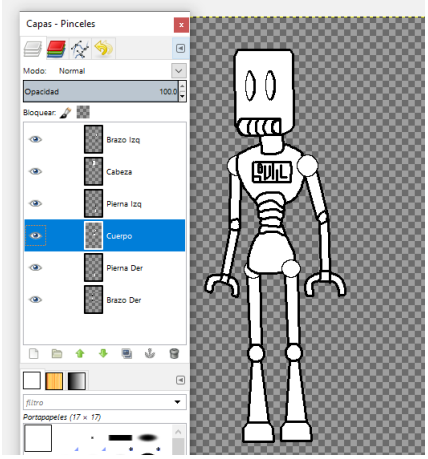
Mediante su algoritmo se puede definir el movimiento, destino y camino a seguir de los enemigos según la *IA* programada utilizando como base el algoritmo A* de búsqueda de caminos. Así, podemos establecer que el enemigo persiga a nuestro personaje por el nivel.



9- Representación del camino que la IA desea tomar para llegar hasta el personaje.

De este modo, no debemos preocuparnos en exceso del *pathfinding* enemigo y centrarnos en el comportamiento de estos, como veremos más adelante.

Para finalizar, se ha utilizado el programa de edición de imagen **GIMP 2** para realizar nuestros propios *sprites* del personaje y enemigos, que importaremos a *Unity* mediante el *2D PSD Importer* para realizar los esqueletos y asociarlos a los *sprites;* además de alguno *assets* gráficos para la interfaz.



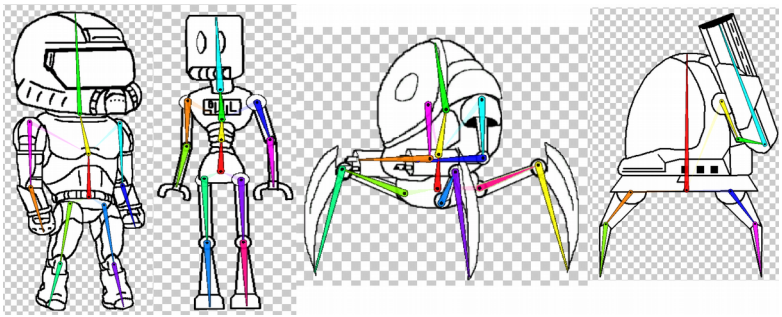
10- Separando en capas cada parte del cuerpo, aparecerán como múltiples *sprites* en *Unity*.

4.3 Inventario de *Assets* y recursos.

Dentro de los *assets* utilizados encontramos tanto unos propios; generados específicamente para este proyecto; mientras que otros son procedentes de la *AssetStore*.

PERSONAJES Y ENEMIGOS

La mayoría de estos *sprites* han sido realizados mediante un programa de edición de imagen para servir como *placeholder* de lo que serían en un futuro; se han hecho con formas simples y sin color (posteriormente se le dio un tono rojo a los enemigos y uno verde para el personaje.



11- *sprites* y esqueletos de los enemigos y el personaje.

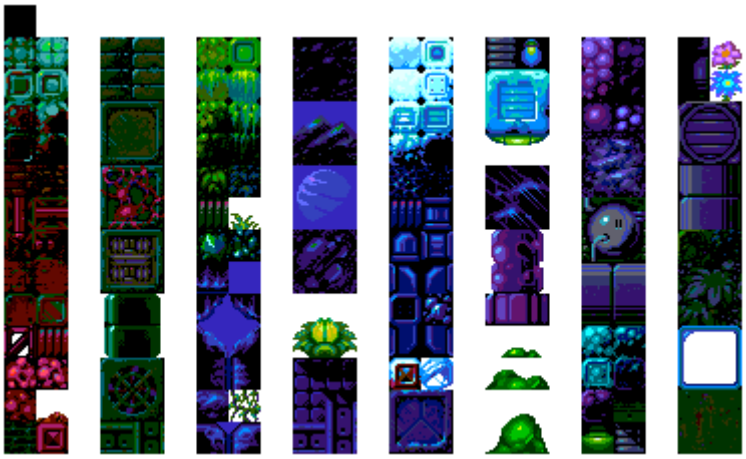
Para desarrollarlos se pensó en un diseño simple estilo droide de combate; muy similares a los que se pueden encontrar en la trilogía de las precuelas de *Star Wars* y otros medios del estilo.

Además, para el personaje principal quería darse un aspecto más robusto y humanizado sin dejar de ser estilo “robot” con una armadura más pesada. Para ello se hizo un diseño similar al de los “marines espaciales” de *Warhammer 40K* o del *Doom Slayer* en *DOOM.*

ESCENARIO Y NIVELES (TILEMAP)

El escenario y los niveles del juego se han realizado mediante *Tilemap*, usando una capa como escenario y plataformas y otro de *background* (sin *colliders*).

Para ello se ha utilizado una *Tile Palette* gratuita de estilo *Metroidvania* de KarlisZabers (itch.io).



12- Paleta completa de estilo Metroidvania de KarlisZabers. (Metroidvania_tileset by KarlisZabers)

itch.io es un *Marketplace* y blogs para desarrolladores independientes con un modelo de compra "*Pay what you want*" (Paga lo que quieras), donde cualquiera puede subir sus proyectos y creaciones para compartir con otros desarrolladores.

EFECTOS Y ESCENOGRAFÍA

Como hemos indicado, hay distintos tipos de enemigo; que tendrán distintos tipos de ataques.

Tanto para el jugador como para los enemigos más comunes se ha utilizado una especie de "bala láser".

Al igual que las *tiles* para el escenario, estos *sprites* de efecto láser han sido obtenidos a través de un *Marketplace*; en este caso, OpenGameArt.org.

13- *Sprite* múltiple de los estílos de láser por Rawdanitsu

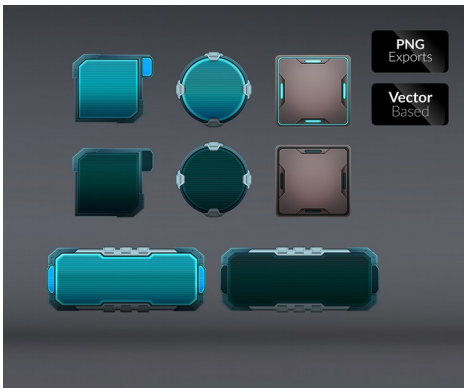
MENÚ E INTERFACES

Para el interfaz del juego se ha adoptado una estrategia mixta; utilizar *assets* ya creados para algunos menús y botones; y generar algunos nuevos siguiendo lo aprendido en *Experiencia de usuario e interfaces.*

Para la barra de vida se ha utilizado algunos de los *assets* pertenecientes al paquete *SpaceDepotUI* del *GUI Megapack (AssetStore)* del equipo de artistas REXARD (*AssetStore*).

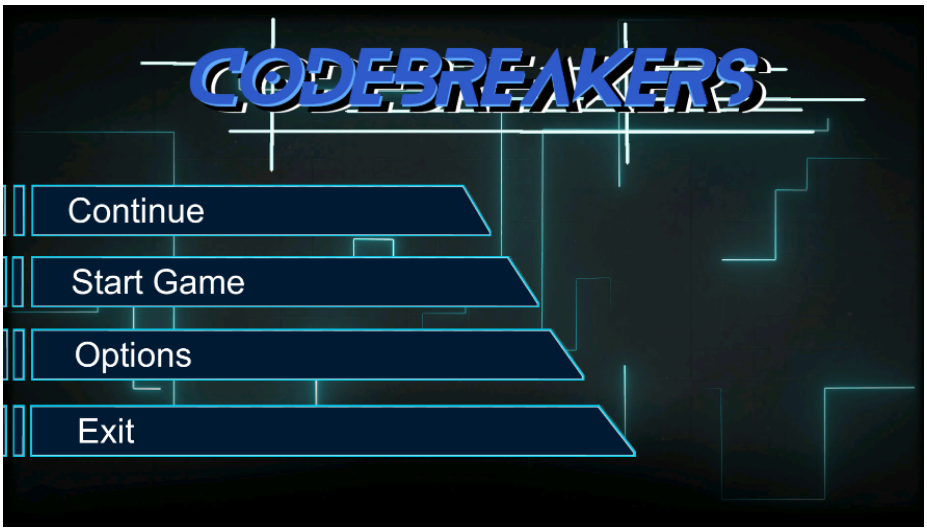
14- Barra de salud

Pero a parte de la *AssetStore* se han obtenido algunos de los *assets* de otros *Marketplace* de *assets* gráficos para videojuegos; como GameArtPartners.com.



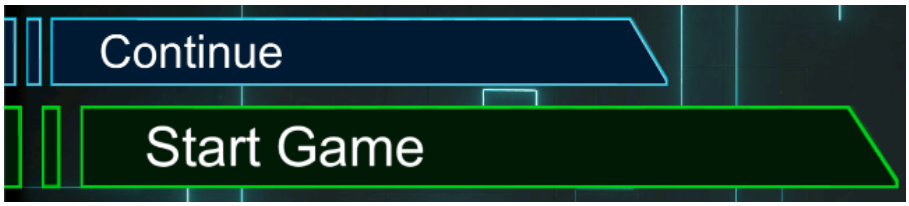
15- *Tech Buttons GUI Collection* *(Gameartpartners.com)*

Pero algunos de los utilizados para la interfaz también han sido desarrollada propia para el juego; cómo es el caso del logo del título y los botones del menú principal.



16- Menú principal de *Codebreakers.* El título y los botones se hicieron para este proyecto.

Se realizaron con el objetivo de darle un toque de animación y movimiento para destacar la selección; que las tres “partes” que forman la imagen se expandiesen para dar sensación de “navegar” entre menús.



17- Animación de botones del menú

AUDIO

Para el audio también hemos recurrido a *Assets* ajenos gratuitos; la mayoría de ellos de la *Asset Store* de *Unity.*

Entre ellos destacamos:

Futuristic Weapons Sounds, por Raffaele

Free Sound Effects Pack, por Olivier Girardot

Ambos packs fueron utilizados para suplir los efectos de sonido básicos; que sirven para dar una idea de cómo quedaría el producto final; pues habría que añadir variabilidad de efectos y otros.

Los efectos de sonido utilizados son:

- Disparos (Balas láser / Cohetes…)
- Ataques Cuerpo a Cuerpo
- Impacto (Daño recibido)
- Efectos de alguna puerta o la explosión final del *boss.*

Pero otros provienen de la web FreeSound.org, una famosa web colaborativa de multitud de efectos, *simples* y grabaciones gratuitos de diversos creadores; bajo la licencia Creative Commons.

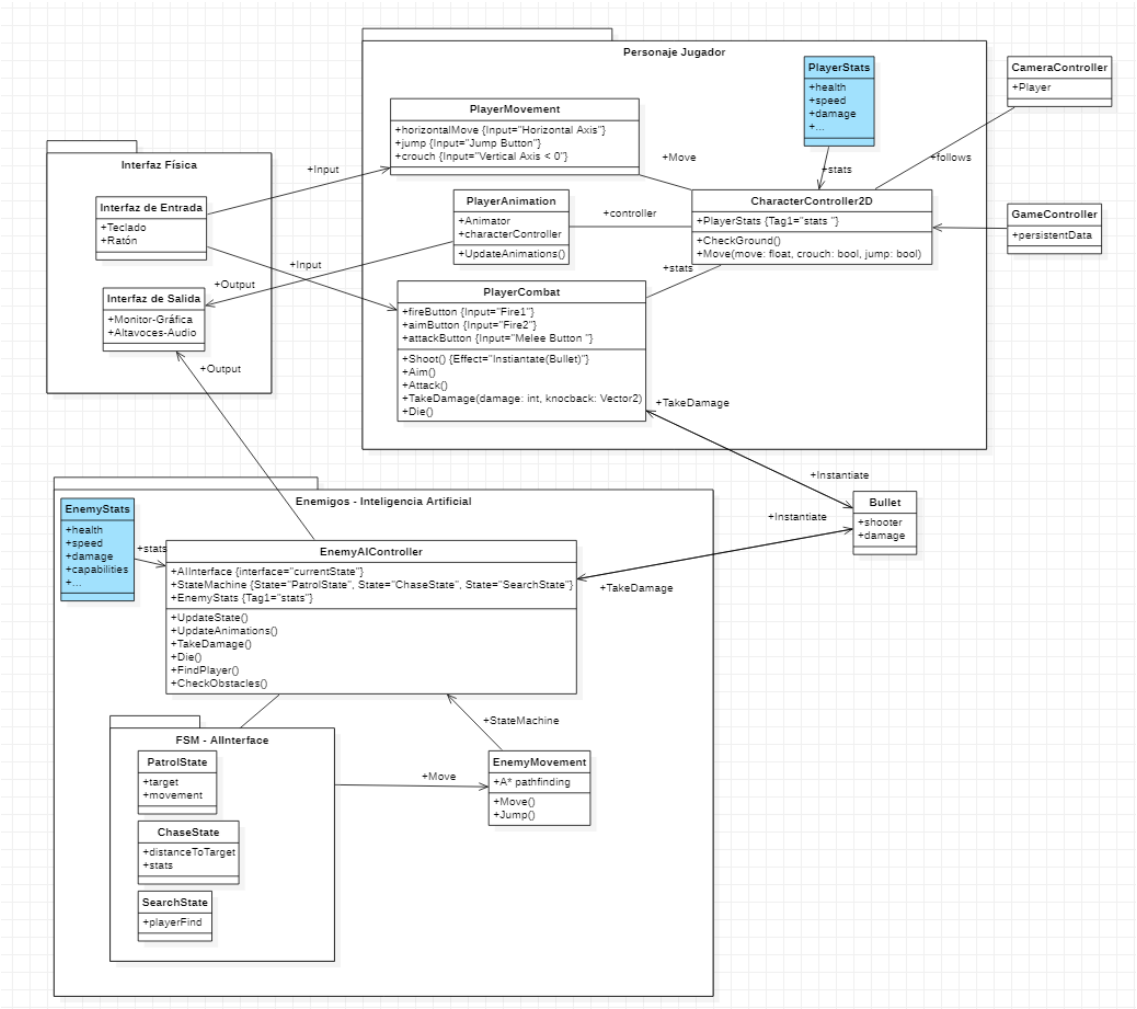
Game Intro (Space | Futuristic), por Tyops

Futuristic Soundscape, por 13NHarri

Boss Battle Loop #3, por Sirkoto51

Estas son piezas de audio para reproducir de fondo; tanto en los menús cómo en los niveles del juego.

4.4 Esquema de arquitectura del juego



18- Arquitectura de los componentes del ciclo de juego entre sí y entre el interfaz físico

En la imagen superior tenemos a grandes rasgos el diagrama de componentes de lo que sería el ciclo principal del juego.

Definimos tres grandes actores durante el ciclo de juego: Los enemigos, compuestos por una Máquina de Estados y parámetros de estadísticas; el personaje controlable; definido por las estadísticas y los inputs procedentes del interfaz físico; y el Jugador, representado como interfaz físico de entrada (Teclado y Ratón) y de salida (Monitor y Audio). [5] [6]

4.5 Inteligencia Artificial

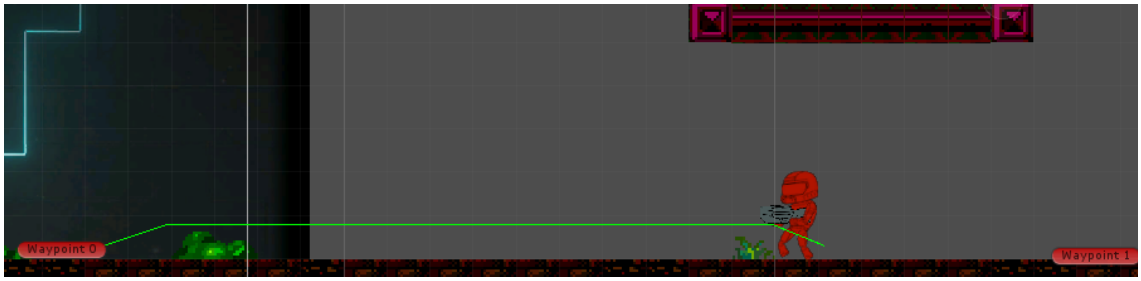
Para la inteligencia artificial hemos utilizado una pequeña FSM (*Finite State Marchine*) para gestionar el comportamiento de cada enemigo; junto a *Scriptable Objects* para definir los distintos parámetros que definen dicho comportamiento, de ese modo, con unos pocos *scripts* podemos definir los comportamientos de varios enemigos. [7]

Cada *scriptable object* (el *script EnemyStats*), contiene distinta información de cómo la IA debe actuar, así como estadísticas numéricas de los enemigos, tales como:

- Puntos de Vida
- Velocidad de Movimiento (Andando/Corriendo)
- Capacidad para: Moverse, Saltar, Volar, Atacar cuerpo a cuerpo, Atacar a distancia…
- Estadísticas sobre los ataques (daño, rango, velocidad, *knockback*…)

19- Diferentes *stats* definen diferentes comportamientos.

La máquina de estados sirve como base en la que "enchufaremos" cada bloque de estadísticas y así regularemos el comportamiento [8]; de ese modo; tenemos tres estados: ***Patrol*** (Patrulla), es el estado estándar, en el que el enemigo está a la espera, dependiendo de las estadísticas definidas, el enemigo se mantendrá estático o se moverá siguiendo unos puntos de control *waypoints* definidos.



20- Estado *Patrol*, el enemigo se mueve de un *waypoint* a otro.

Una vez el jugador entra dentro de la visión del enemigo; definido por una distancia y un ángulo de visión; pasaría al estado ***Chase*** (Persecución).

21- Funcionamiento de la visión de la IA.

En el estado de Persecución comprueba varias condiciones para saber qué acción realizar:

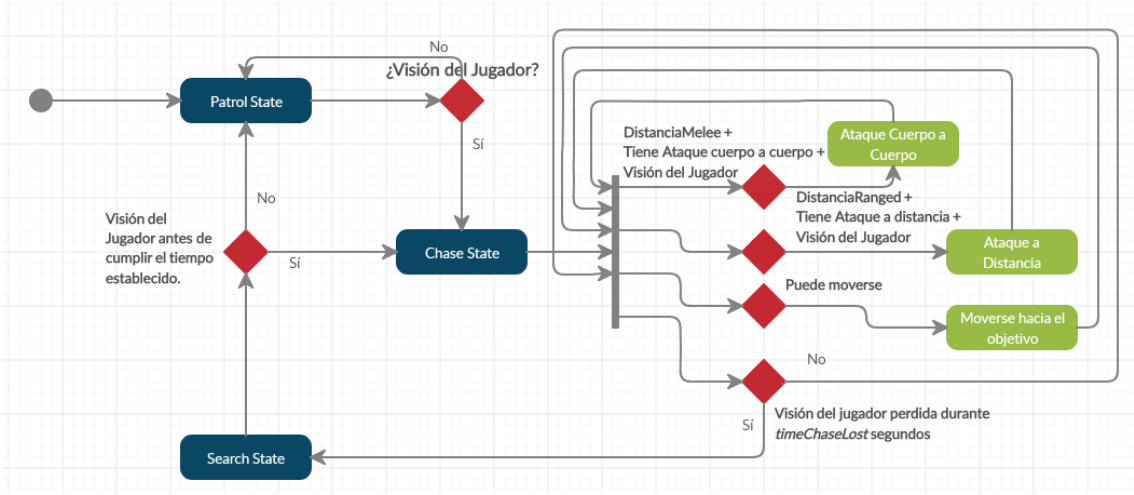
- Si está dentro de rango; y puede disparar (tiene ataque a distancia, el *cooldown* de ataque ha pasado, etc.), procede a **Disparar.**
  Este disparo está definido dentro del *EnemyStat* y puede ser de tres tipos: **Bullet**, proyectil recto, pequeño y rápido pero visible, **Raycast** un disparo tipo "láser", que se dispara a gran velocidad, se tarda en cargar; y **Launcher**, dispara un cohete que se desplaza lentamente apuntado a la posición del jugador; gran daño, pero fácilmente esquivable y puede explotarse con ataques propios.

- Si está dentro de rango; y puede atacar (tiene ataque cuerpo a cuerpo, y el *cooldown* ha pasado), ataca con un ataque de corta distancia y que empuja al jugador hacia atrás.

- Si no está en rango, pero puede moverse, el enemigo se desplaza hacia la última posición conocida del jugador; que se va actualizando mientras el enemigo tenga visión del jugador. Si tiene la capacidad de **saltar** evitará obstáculos, saltando por encima de ellos. para encontrarlo.

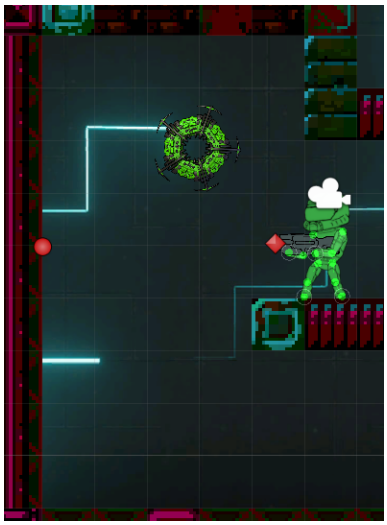
- Si ha pasado un tiempo desde que el enemigo dejó tener visión del jugador pasa al estado ***Search*** (Búsqueda).

En este último estado, ***Search***, el enemigo simplemente mira alrededor buscando al jugador; si, tras un tiempo no lo encuentra, vuelve al estado ***Patrol***, mientras que, si establece visión con él, volverá a la persecución ***Chase.***



22- Esquema de la FSM de los enemigos.

A parte de los enemigos definidos; existe otro tipo de peligro o enemigo, ajeno al comportamiento por Máquina de Estados; dado que se trata más de un obstáculo que de un enemigo "inteligente".

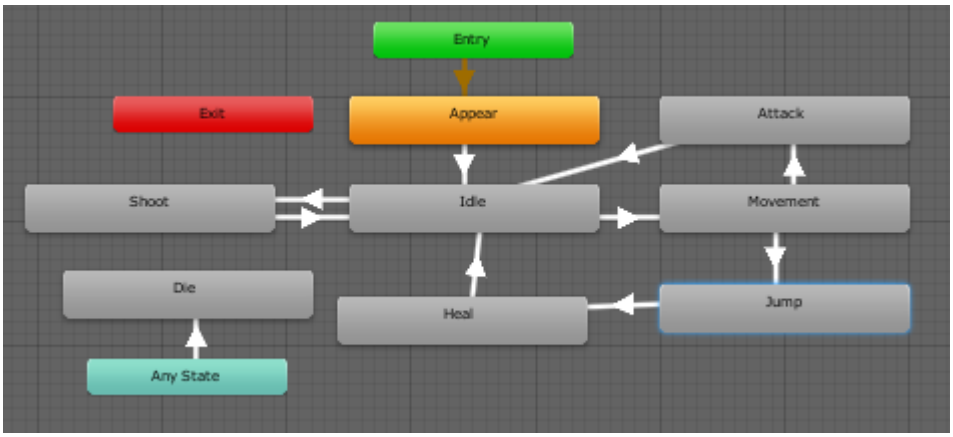


Este enemigo tan sólo se mueve de arriba abajo y cambia de dirección una vez golpee contra el suelo/techo.

23- Enemigo "Obstáculo".

Para el *boss* final; sin embargo, se ha seguido otra filosofía. No podía implementarse como un enemigo más, dado que se requería acciones más concretas; así que, o hacíamos otra FSM propia, con su interfaz y sus estados; o reutilizábamos una *State Machine* propia de *Unity:* El *Animator.*

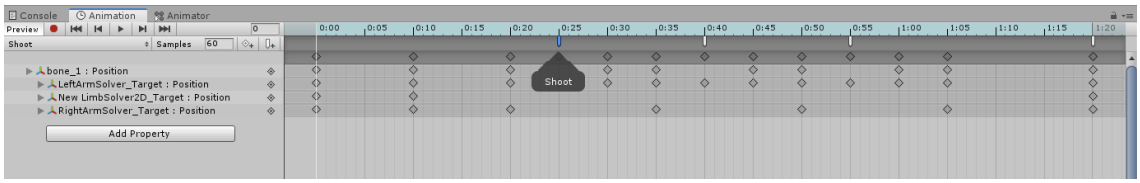
Dado que nuestro jefe va a contar con un *Animator* para poder controlar las animaciones, podemos reutilizar el árbol de animaciones cómo FSM para el comportamiento del *boss*; puesto que al fin y al cabo, el *Animator* de *Unity* es, efectivamente, una FSM.



24- El *Animator* de Unity no deja de ser una *Finite State Machine* en sí mismo.

Para ello, dispondremos de un *script* principal, que controlará las funciones generales del *boss* y estarán públicas para que el *Animator* pueda acceder a ellas (como veremos más adelante); y *scripts* de estados dentro del *Animator,* o como se llaman concretamente en *Unity*, StateMachineBehaviour.

Así, podemos disponer de toda la información y un control exacto de la comunicación Animación-Lógica; por ejemplo, uno de los ataques del jefe final es una ráfaga de tres disparos, para realizarla de la manera habitual habría que llamar tres veces a una instanciación de la bala y cuadrarla con los tiempos de la animación; pero, si damos acceso a esa instanciación al *Animator*, podemos modificar las animaciones para que utilicen eventos.



25- Animación *Shoot* del jefe final.

En la imagen superior, podemos ver como la animación *Shoot* del jefe final, llama a la función homònima, pública, dentro del *BossBehaviour* (nuestro *script* principal).

De ese modo, podemos tener medido a la perfección el punto concreto donde queremos instanciar la bala.

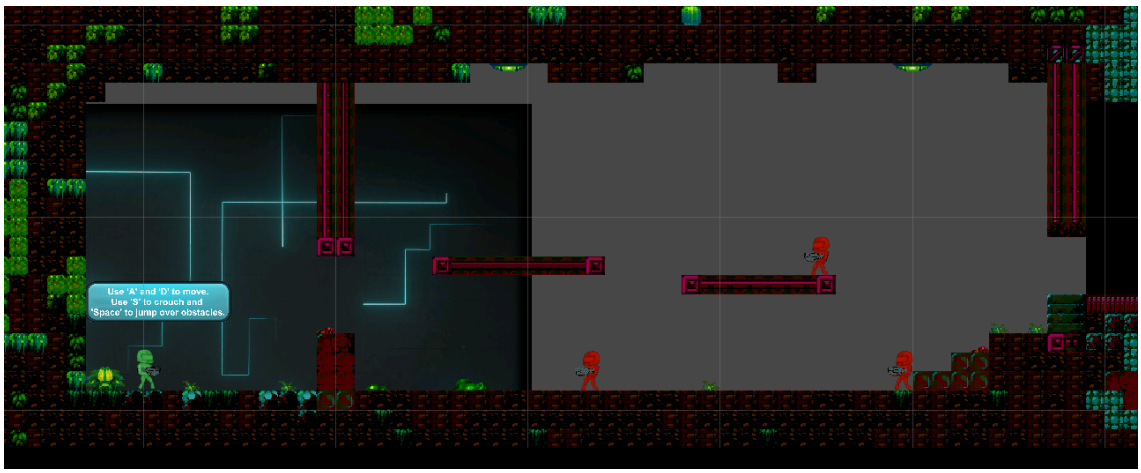
5. Diseño de niveles

26- Nivel completo en una imagen

Como podemos observar en la imagen de arriba, el nivel actualmente está formado por 4 subniveles; a los que faltaría añadir 2 más: La zona de descanso previa a la pelea final y el combate contra el jefe (*boss*).

Para analizarlo correctamente debemos dividir los niveles en sus respectivos subniveles. La nomenclatura de cada escena vendrá dada por “Nivel del Juego” – “Subnivel dentro del Nivel”.

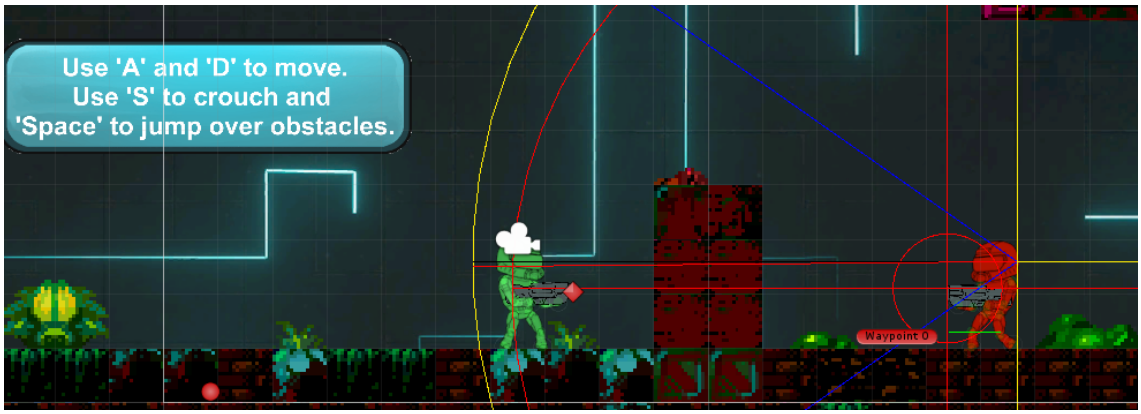
5.1 Nivel 1-1: Fase de Tutorial



27- Escena "Level 1-1"

Aquí tenemos un nivel tipo tutorial; cortito, con un solo tipo de enemigo, pero dos comportamientos y con indicaciones para ayudarnos a adaptarnos a los controles.

Empezaremos en una “zona segura”, aquí una ventana indicará los controles de movimiento y el obstáculo en frente prevendrá que el enemigo patrullando pueda vernos.

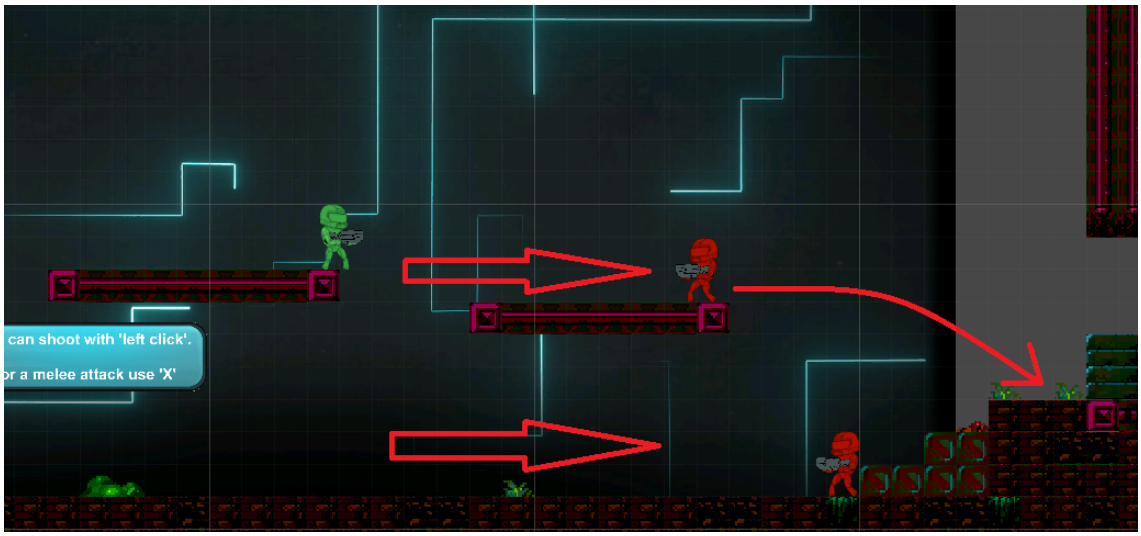


28- El primer enemigo que encontramos patrulla entre dos puntos, pero no podrá vernos al principio.

Tan pronto saltemos por encima del obstáculo aparecerá un mensaje indicándonos los controles del combate; pues estamos a punto de empezar a ello.

En este nivel de presentación sólo el enemigo que patrulla nos perseguirá, los otros dos se mantendrán en el sitio, de ese modo podemos separarlos para ir derrotándolos de uno en uno.

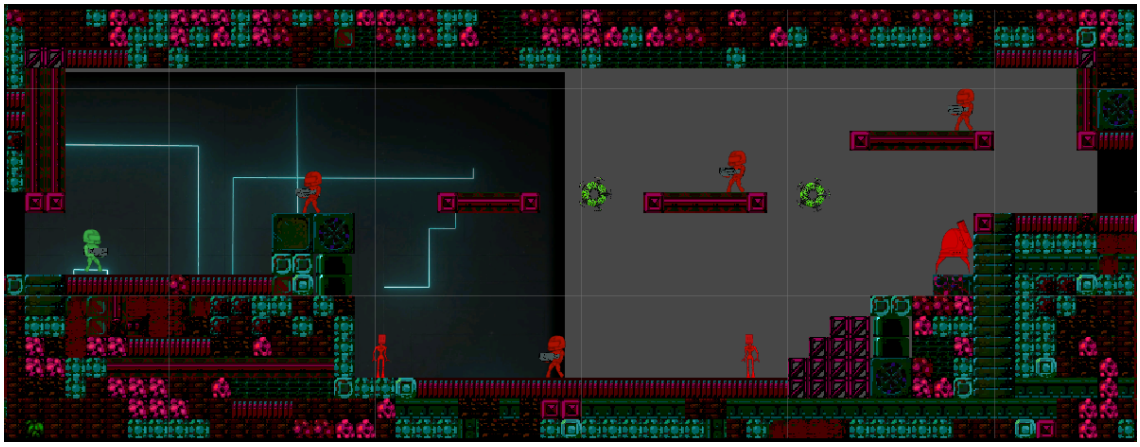
Para derrotar a ambos podemos ir por arriba o por abajo; incluso no es necesario eliminarlos para avanzar, en muchos niveles nuestro objetivo será pasar a la siguiente fase. Sólo unos pocos niveles nos exigirán derrotar a todos los enemigos de la sala para poder avanzar.



29- Aunque en este caso no cambia mucho, habrá momentos que podamos elegir más de una ruta para avanzar.

El concepto de este nivel es más que nada el familiarizarse con los controles y los enemigos más comunes.

5.2 Nivel 1-2: Fase “Abierta”



30- Escena "Level 1-2"

En este nivel se busca introducir nuevos tipos de enemigos y la capacidad de analizar las distintas rutas para avanzar.

Cómo podemos ver de un simple vistazo, tenemos dos rutas; una por arriba, en la que dependemos del “plataformeo” y el esquivar a los enemigos saltando; mientras que la de abajo es un plano, pero fuertemente protegido.

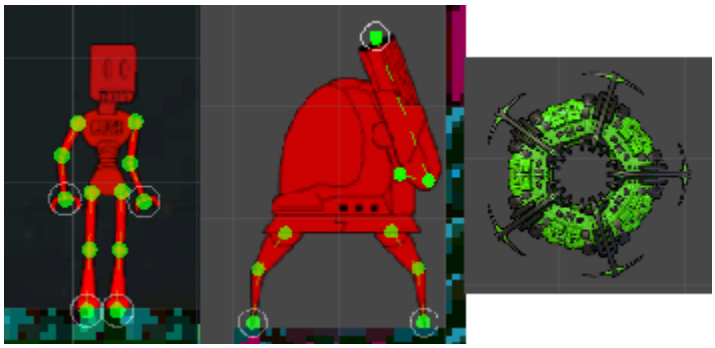
Aquí pues, el jugador puede elegir la ruta que más se adapte a su estilo de juego, evitar combate directo contra muchos enemigos, valiéndose de la capacidad de salto del personaje o enfrentarse directamente al peligro.

Además, en este nivel aparecen tres enemigos nuevos:

1- El robot *melee*; este enemigo es rápido, pero solo dispone de ataque cuerpo a cuerpo, con lo que podemos eliminarlo desde distancias seguras.

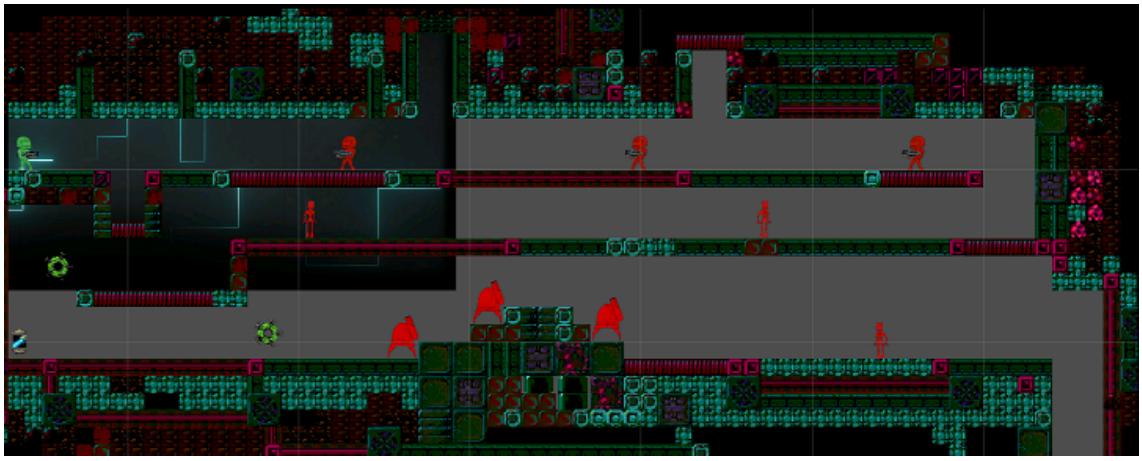
2- El robot *rocket*; este enemigo no se mueve y dispara lentamente; pero sus proyectiles son misiles dirigidos directamente a la posición del jugador. Estos misiles pueden ser derribados con nuestros disparos, o podemos esquivarlos simplemente; pero si nos golpea recibiremos una gran cantidad de daño

3- El robot *mine;* un enemigo simple que sirve simplemente de obstáculo. No dispone de una *IA* como el resto de enemigos basados en *FSM*, sino que simplemente se mueve obstaculizando al personaje y haciendo daño al contacto.



31- De izquierda a derecha: Robot *melee*, robot *rocket*, robot *mine.*

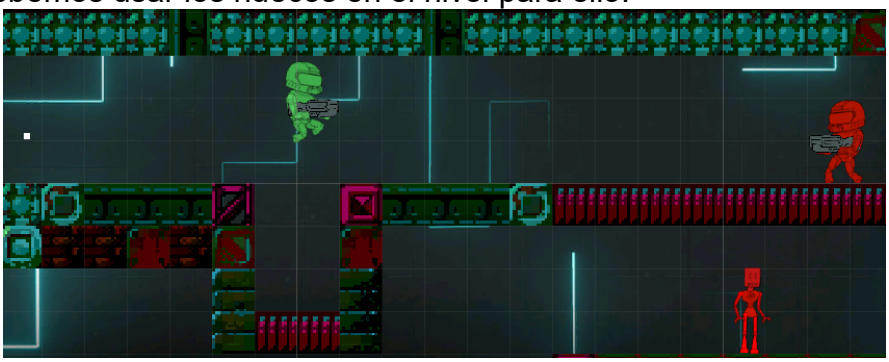
5.3 Nivel 1-3: Fase Lineal



32- Escena "Level 1-3"

Aquí tenemos un nivel de resistencia. Ahora que el jugador sabe como combatir a estos enemigos, deberá ir eliminándolos a medida que avanza por este nivel lineal.
El objetivo es que el jugador aprenda a reaccionar rápido.

La primera parte del nivel consta de tres enemigos a distancia; no podemos saltar ni agacharnos para evitar sus disparos; por lo que debemos usar los huecos en el nivel para ello.



33- Primer enemigo del Nivel 1-3

Para el primer enemigo debemos colocarnos en el hueco que hay en el suelo y calcular el tiempo entre los disparos para poder acertarle. Esto sirve al jugador como pausa y para que vea que no siempre es mejor ir directo a la acción; que a veces hay que pararse y pensar.

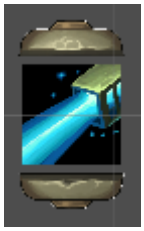
34- Segundo enemigo del Nivel 1-3

Para el segundo y tercer enemigo es el mismo funcionamiento que el primero; sólo que, en vez de meternos en el hueco del suelo y calcular el tiempo de los disparos, debemos disparar y luego saltar en el hueco del techo para poder esquivar los disparos enemigos.

La siguiente parte del nivel es totalmente plana, con dos enemigos que sólo atacan cuerpo a cuerpo, pero son muy rápidos. Es un cambio con respecto a la primera parte que debíamos ir mucho más pausados.

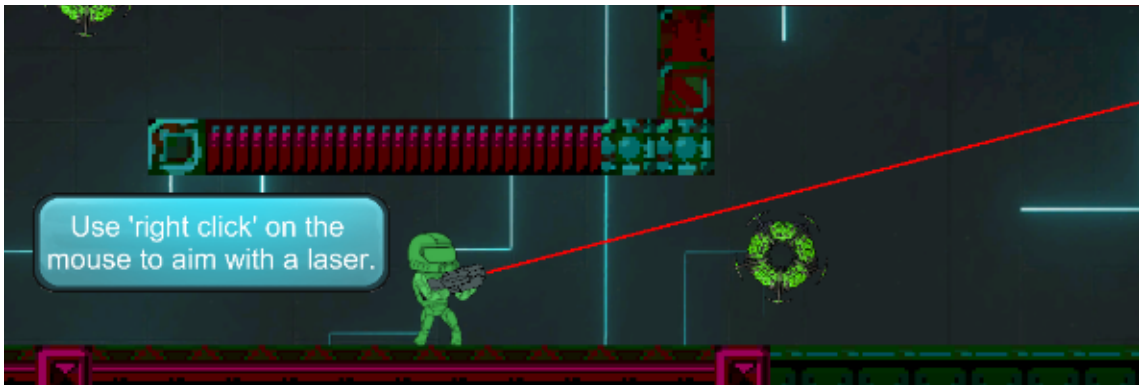
Después nos encontraremos un enemigo obstáculo, debemos analizar su patrón de movimiento para pasar sin ser dañados.

Una vez llegado a este punto recompensamos al jugador con una pequeña habilidad extra para el personaje; el apuntado láser.



35- *Power-up* del apuntado láser.

Al recoger el *power-up* aparecerá un mensaje explicándonos cómo funciona el apuntado láser. Desde este momento podremos utilizarlo cuando queramos.

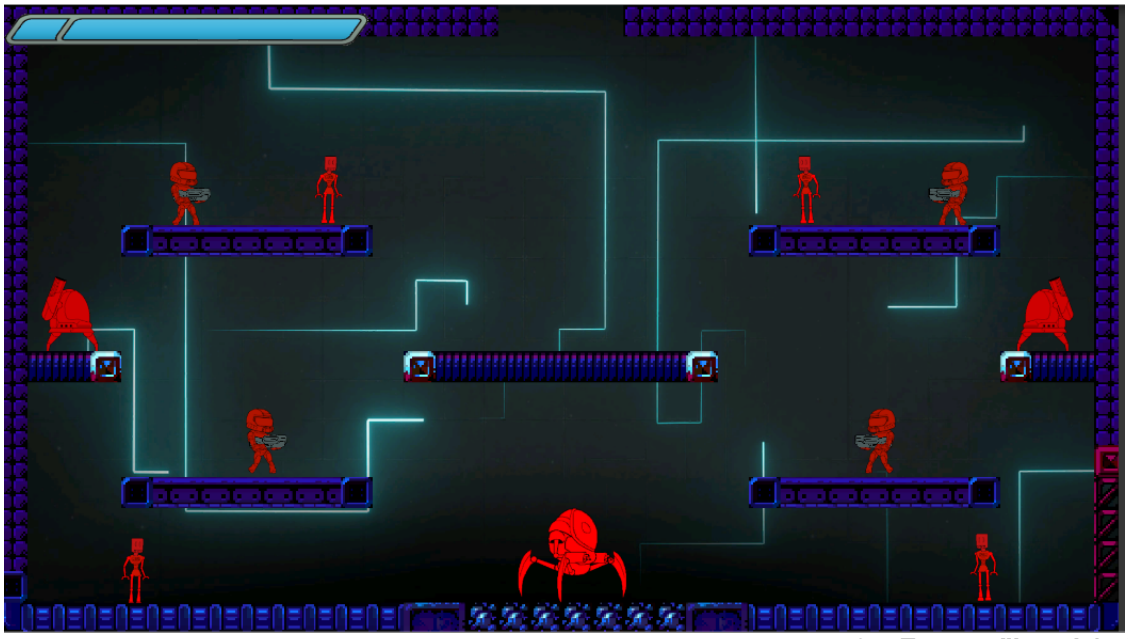


36- Nueva mecánica para el jugador: El apuntado láser.

Ahora sólo queda avanzar hasta el final del nivel, contra tres enemigos con cohetes teledirigidos. Por suerte, gracias a nuestra nueva habilidad, destruir los cohetes y a los enemigos es mucho más sencillo.

Por último, llegamos al final donde un enemigo cuerpo a cuerpo intentará sorprendernos, y nos dejaremos caer para avanzar y llegar a la fase de desafío.

5.4 Nivel 1-4: Fase de Desafío.



37- Escena "Level 1-4"

Este nivel es distinto al resto; los enemigos aparecerán en cuanto el personaje haya caído a la plataforma central, y no hay manera de escapar.

Esto es un nivel de desafío; un nivel en el que debemos derrotar a todos los enemigos para poder avanzar. Deberemos usar todo lo aprendido y nuestra nueva habilidad para poder salir airosos.

Para dar esa sensación, la paleta de colores del nivel es diferente al resto; y no sólo eso, sino que, en este caso, la cámara es estática; es decir, en vez de seguir al jugador por el nivel, abarca por completo lo que ocupa el nivel y no se desplaza.

Además, para complicar más este pequeño desafío al jugador, se introduce un nuevo enemigo, mucho más fuerte que los anteriores.

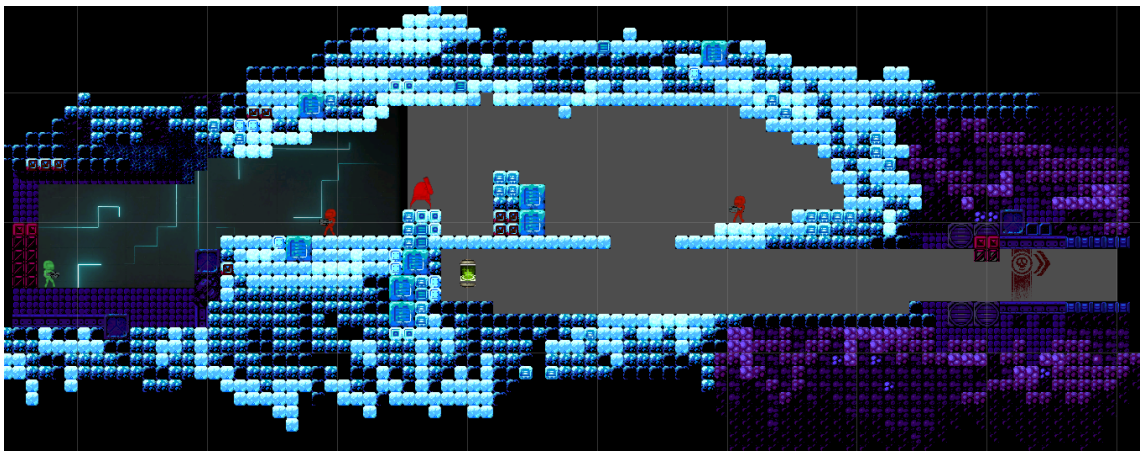
Aquí el jugador deberá pensar la estrategia a seguir rápidamente; lo óptimo sería ir eliminando a los enemigos uno a uno; empezando por los enemigos rápido cuerpo a cuerpo que nos saltarán a la plataforma central.

Después de ellos, los robots *rocket* debe ser derrotados, pues su largo alcance pueden ser un peligro mientras busquemos eliminar al resto.

Para seguir, eliminaremos a los dos enemigos de las plataformas superiores, dejando para el final las plataformas inferiores y el suelo.

Obviamente, lo idóneo es quitarse del medio a todos los enemigos sin que se lleguen a agrupar demasiado y enfrentarse tranquilamente al nuevo enemigo.

5.5 Nivel 1-5: Fase de Reposo y Recuperación

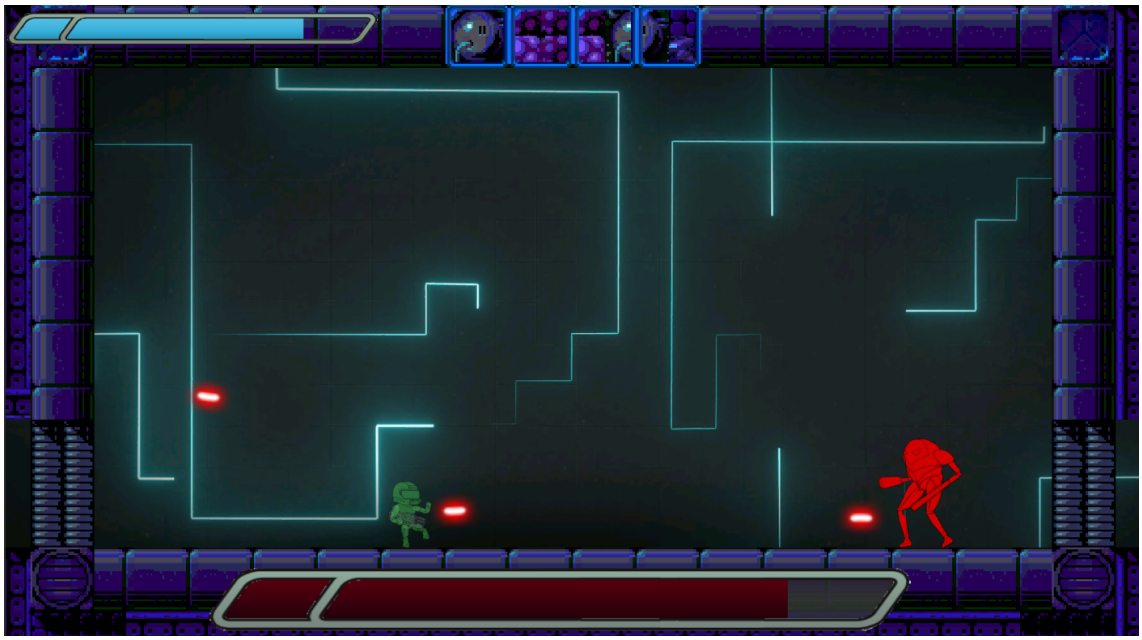


38- Escena "Level 1-5"

Después del desafío anterior, se deja al jugador reposar con un nivel sencillo, que simplemente destaca por presentarnos un objeto de sanación, algo que nos será tremendamente útil dado que a continuación, viene la fase final.

Este objeto nos recuperará el 50% de la vida máxima; de ese modo, podemos llegar sin muchos problemas a la fase final.

5.6 Nivel 1-6: *Final Boss*



39- Escena "Level 1-6"

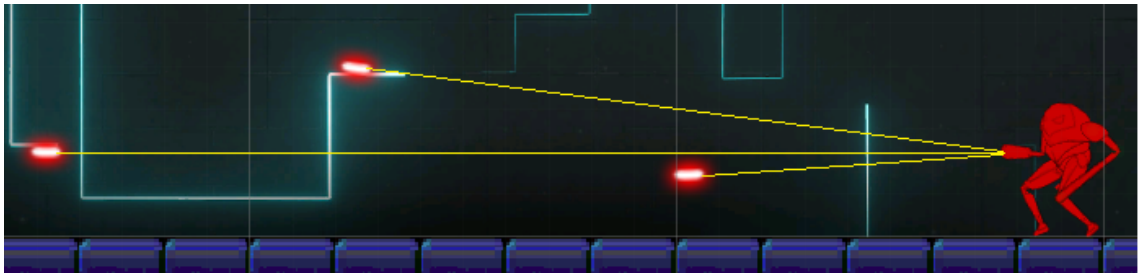
Al fin llegamos a la fase final; donde debemos enfrentarnos al enemigo más complejo hasta el momento.

Aquí nos encontraremos de nuevo en una habitación cerrada con cámara fija; y sólo un enemigo, pero con una gran cantidad de vida que debemos ir bajando poco a poco con nuestros ataques.

Se trata de un combate de resistencia y de aprender los patrones de ataques; que en principio son dos:

- Ataque a distancia
- Ataque cuerpo a cuerpo

Ataque a Distancia:
Primer ataque de nuestro enemigo. Con este ataque, el *boss* lanza una rápida ráfaga de balas en varias direcciones.



40- Ataque a distancia del *Boss*

La clave aquí es el orden de los disparos; el primero va recto, por lo que podemos saltarlo o agacharnos para evitarlo.

El segundo va alto; por lo que lo esquivaremos sin movernos; pero, esto nos enseña que el primer disparo es mejor esquivarlo por abajo, puesto que si saltamos el primer disparo nos arriesgamos a que nos dé el segundo.

El tercer disparo va por abajo, por lo que debemos saltar para esquivarlo; aunque si estamos muy cerca del jefe podemos seguir esquivándolo manteniéndonos agachados; aunque nos exponemos a otros ataques del enemigo.

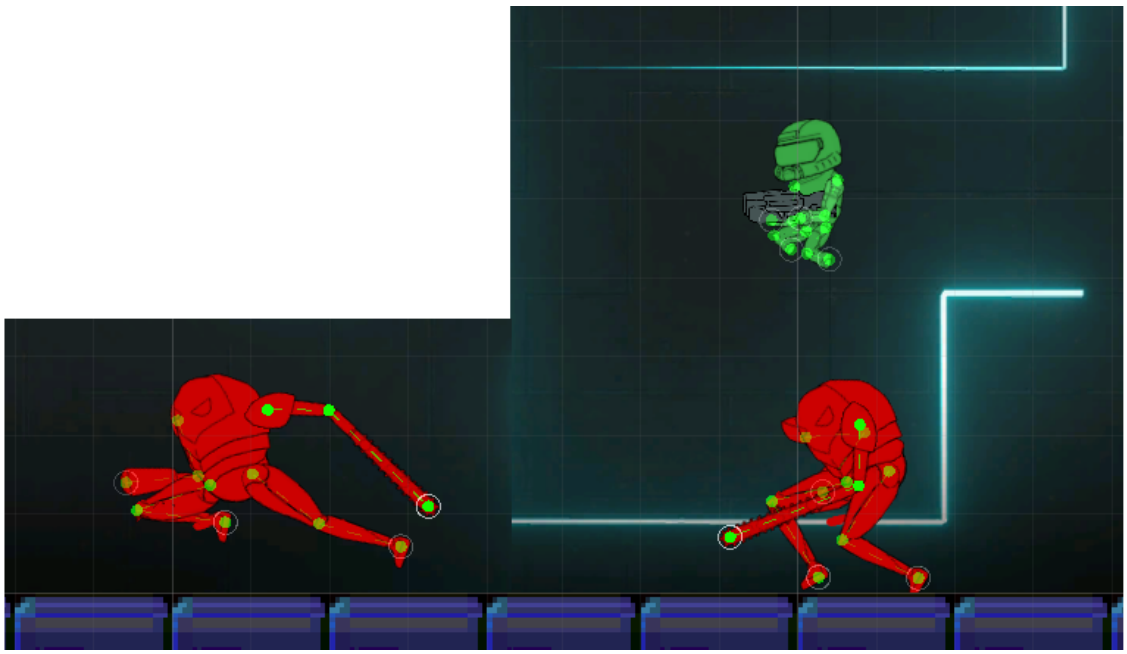
El movimiento óptimo sería esquivar el primero agachado, esperar que pase el segundo y saltar el tercero mientras disparamos al enemigo para mitigar su barra de vida.

Ataque Cuerpo a Cuerpo

En este segundo ataque, el enemigo se abalanza sobre nosotros a gran velocidad; y una vez llegue a nuestro lado atacará.

El *boss* no parará hasta encontrarnos cerca de nosotros, por lo que arrinconarnos en una pared es contraproducente.

Podemos intentar saltar por encima del enemigo; pero nos encontraremos que, tras atacar, el enemigo vuelve a la posición en la que estaba; por lo que nos golpeará a la vuelta.



41- Ataque cuerpo a cuerpo del jefe final.

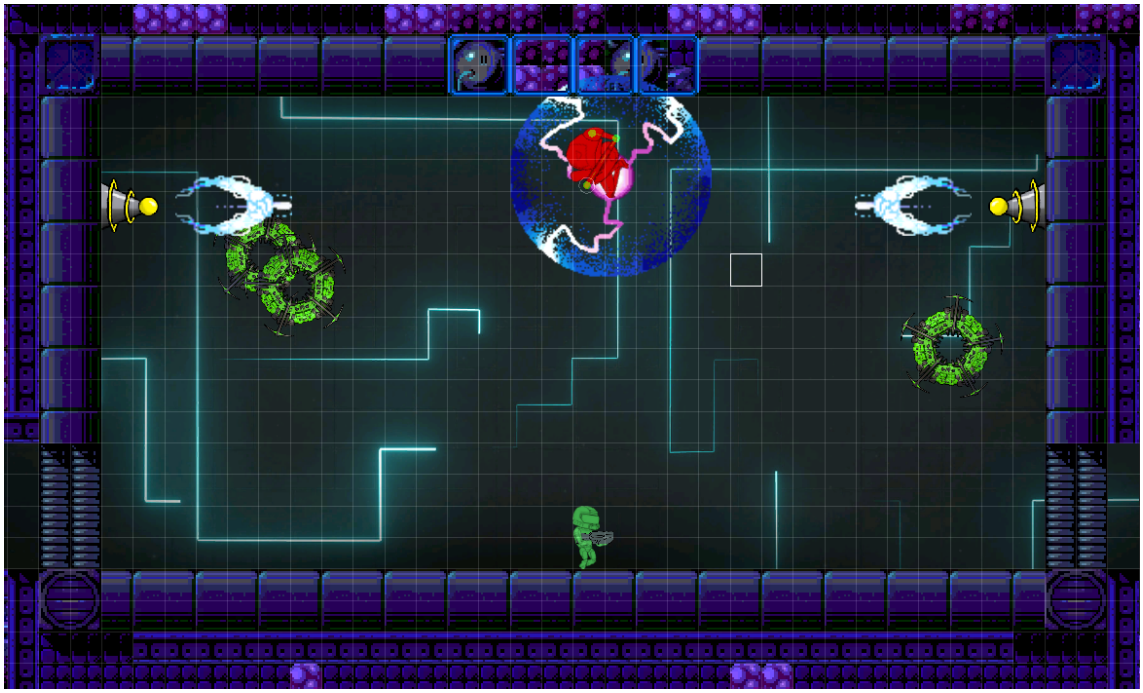
Con ello, sólo nos queda una opción, y es una prueba de nervios, pues debemos dejar que el enemigo se acerque y saltar hacia atrás en el momento justo para evitar el daño.

Si lo conseguimos, podemos castigar al enemigo disparando sin parar mientras regresa a su posición.

Ataque en Furia

Cuando el jefe final pierda la mitad de la vida entrará en fase de Furia (o *Rage*); este estado comenzará con un nuevo ataque; en el que el enemigo se acerca al centro del nivel y se alza, protegiéndose con una burbuja que lo hace invulnerable.

Durante este estado, no podremos hacerle daño, y, además, nos arrojará enemigos tipo “obstáculo” que rebotan por las paredes. Estos enemigos desaparecerán al golpearnos o cuando pase un tiempo determinado; y no habrá más de 3 en pantalla, para no atosigar al jugador demasiado.



42- Ataque en Furia

Para avanzar, debemos utilizar nuestra habilidad de apuntar para derribar las dos antenas que proporcionan esa barrera de protección al enemigo.

Son débiles, pero debemos hacerlo mientras esquivamos los ataques del enemigo.

Tras sacarlo de ese estado de estasis, este nuevo ataque entrará dentro de la rotación normal del enemigo.

Una vez consigamos derrotar al Jefe, la puerta del fondo se abrirá y podremos finalizar la demo.

6. Manual de usuario

Actualmente, el juego está preparado para ser ejecutado en PC con teclado y ratón; pero se busca poder implementar un control con mando de consola en la *Versión Final*, de cara a poder ser importado a plataformas de consola.

En cuanto al funcionamiento del juego; se ha buscado unos controles bastante estándar para que no sea un problema de entrada a los jugadores.

Al ser un juego en 2D los controles de movimiento son los estándares; A y D controlan el desplazamiento mientras que S sería la tecla para agacharse.

Para los ataques tenemos una acción cuerpo a cuerpo y otra a distancia; aquí; utilizaremos la tecla X para usar el comando de ataque cuerpo a cuerpo mientras que usaremos el ratón para el control del arma a distancia:

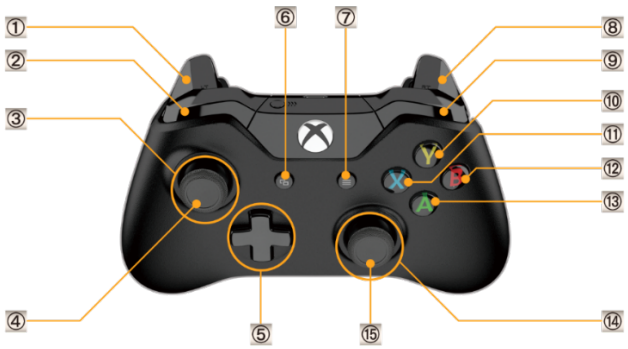
- Clic Derecho: Disparar Arma
- Clic Izquierdo: Preparar Arma (Apuntar con láser – sólo si está disponible)
- Movimiento del Ratón: Apuntar Arma.

Por tanto; los controles en teclado y ratón quedarían tal como sigue:

Mapeado en teclado y ratón

Botón	Acción	Botón	Acción
A	Movimiento Izquierda	**D**	Movimiento Derecha
S	Agacharse	**Espacio**	Saltar
Q	*Dash* Izquierda (*)	**E**	*Dash* Derecha (*)
X	Ataque Cuerpo a cuerpo	**W**	Usar Herramienta (*)
Esc	Menú de Pausa		
Ratón Der.	Preparar Arma a Distancia (Si está disponible)	**Ratón Izq.**	Disparar Arma a Distancia

Mientras que, los controles en mando de consola serían los siguientes:



Mapeado en mando Xbox One

Botón	Acción	Botón	Acción
1	Preparar Arma a Distancia (Si está disponible)	**9**	Esquiva Derecha (*)
2	Esquiva Izquierda (*)	**10**	Interactuar
3	Movimiento	**11**	Usar Arma Cuerpo a Cuerpo
4	-	**12**	Usar Herramienta (*)
5	Movimiento (Alt)	**13**	Saltar
6	-	**14**	Apuntar Arma a Distancia
7	Menú de Pausa	**15**	-
8	Disparar Arma a Distancia		

(*)- Las mecánicas o controles no están programados/testeados

7. Conclusiones

Con todo lo expuesto, llegamos al final de este proyecto; un camino en el que aprendemos no sólo a programar y diseñar un pequeño proyecto en el ámbito de los videojuegos, sino también a lidiar con planificaciones, fechas límite y, en este año 2020, también a lidiar con ciertos problemas externos, ajenos a nuestra influencia.

Está claro que una de las partes a la hora de diseñar y desarrollar nuestro propio videojuego es la planificación y tener una visión inicial, completa, del proyecto; la primera parte de todo proyecto, ya sea un videojuego, cualquier tipo de software o cualquier otro tipo de trabajo, proyecto o estudio de ingeniería. Y es que tener claro unos objetivos y lo que queremos conseguir es vital para que el proyecto llegue a buen puerto, y es algo que, objetivamente, se ha fallado.

Cómo ya se ha definido, el proyecto nació como una idea mucho más grande; pero que era imposible de llevar a cabo en este breve periodo de tiempo, por lo que se decidió reducir la visión que se tenía de este, centrándose únicamente en las mecánicas.

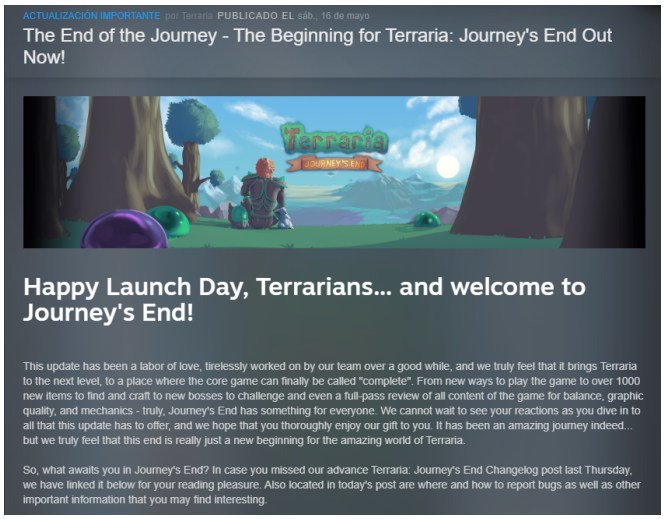
Aun teniendo claro eso, se abarcó más de lo que se podía, especialmente con la situación global que ha afectado al proyecto indirectamente.

Desarrollar un videojuego, cómo cualquier otro proyecto de *software* es un proceso continuo en evolución que podemos dividir en fases, con diferentes objetivos a cumplir, sólo que contiene más disciplinas diferentes: Programación, diseño, interfaces, arte, audio, estudio entretenimiento y balanceo… Claramente un proyecto multidisciplinar y una tarea titánica para una sola persona si no se administra correctamente.

Está claro porque muchos desarrolladores independientes acaban dejando sus ahorros y salud en sus proyectos, jugando una lotería en la que su juego puede triunfar o hundirse; el mercado es competitivo, y las ideas de estos desarrolladores pueden hundirse antes incluso de llegar a tener el proyecto en fase beta. [9]

Incluso las grandes compañías no están exentas a todo esto; sus trabajadores no sufrirán tanto el riesgo económico, puesto que funcionan como cualquier otra gran empresa; pero no es desconocido en la industria el *crunch*, el trabajar en horarios que pueden llegar a duplicar las horas normales y ponen en riesgo la salud, mental y física, de los trabajadores; incluso este *crunch* a veces viene por parte de los propios trabajadores (especialmente en los estudios pequeños), que creen en su proyecto y lo dan todo por él; pero sin una buena planificación el proyecto puede desvariar sin parar hasta que se quede sin presupuesto o sin capacidad humana de seguir adelante. [10]

Ningún videojuego es perfecto; y menos en la visión de sus diseñadores y programadores; siempre se te ocurre algo nuevo que meter, algo nuevo que diseñar; un cambio aquí, otro allá…
Es por ello que muchos videojuegos se actualizan sin parar con nuevo contenido; además en muchos casos gratuito, especialmente en el caso de los *indies* que llegan a triunfar, cómo es el caso de *Terraria,* que el pasado sábado 16 de mayo de 2020 sacó una nueva actualización: *Journey's End* tras más de tres años desde la última actualización; porque para muchos es más que un trabajo diario, es su obra, su creación; y es algo bonito, pero peligroso. [11]



43- *Post* de Steam sobre la actualización 1.4 (*Journey's End*) de *Terraria.*

Pero hablando del proyecto propiamente; algunos de los objetivos cambiaron entre partes, pues se sobreestimó el tiempo disponible y la envergadura del proyecto; además, parte de las mecánicas se decidieron cambiar a mitad del desarrollo para adaptarlo mejor a lo que se buscaba; cambios pequeños, pero que afectaron al tiempo disponible. Por ello, he de decir que no todos los objetivos se han alcanzado; hay varias cosas que aún me gustaría incluir pero que desgraciadamente no creo que vean la luz en la versión final, objetivos secundarios en su mayoría, complementos.

Pero hay que tener en cuenta una situación inusual, un factor que ha afectado globalmente a todas las industrias y también, en consecuencia, al proyecto.

El 31 de diciembre de 2019, la Comisión Municipal de Salud y Sanidad de Wuhan (provincia de Hubei, China) informó sobre un agrupamiento de casos de neumonía de origen desconocido; que posteriormente fue nombrado por la OMS como COVID-19 (enfermedad infecciosa por coronavirus -19).

El 14 de marzo de 2020, el gobierno de España decreta el estado de alarma para hacer frente a la expansión del coronavirus. Muchos negocios cerraron debido a este decreto; aunque, afortunadamente, no afectó directamente a nuestro Máster de Diseño y Desarrollo de Videojuegos al tratarse de un curso completamente online. [12]

Sin embargo, yo, como la mayoría de los alumnos de este Máster, lo llevamos a cabo además de nuestro trabajo diario; por lo que el COVID-19 sí afectaría a nuestro proyecto indirectamente.

En mi caso concreto, mi trabajo es de técnico de automatización en una Industria Lechera, por lo que, durante las primeras semanas de confinamiento, la adaptación del trabajo y la carga de este hizo que se careciese de tiempo para continuar el proyecto; por ello, hubo que revisar los objetivos y reducir el tamaño del proyecto.

Con todo, ha servido para aprender a gestionar el proyecto en situaciones negativas que no dependen de nosotros, aprender a reubicar objetivos, cambiar el foco del juego y cambiar la planificación para adaptarse a la situación.
Además, tal y como se lleva indicando durante todo el documento, esto es una base; un esbozo de un proyecto a futuro que ha ido evolucionando. Todo lo aprendido, especialmente en el ámbito de la gestión de recursos y tiempo será aplicado en el futuro.

Está claro que el desarrollo de un videojuego es un proceso largo, continuo y en evolución; donde diferentes disciplinas deben ser consideradas; y aprender a gestionar los recursos es clave: ¿Aumentar el equipo para aliviar la carga de trabajo a cambio del coste de un

salario? ¿Desarrollar elementos propios o utilizar elementos de otros desarrolladores y artistas? ¿Buscar un proyecto pequeño, con riesgo capital pequeño o un proyecto ambicioso, una visión de nuestra obra?

Todo depende de los recursos disponibles y de nuestra intención; pero una cosa está clara; esta industria está llena de soñadores, es una industria muy vocacional, muchos de los trabajadores, desarrolladores, programadores, artistas… acaban abandonando la industria a una temprana edad; y es que, al fin y al cabo, es un trabajo. Un trabajo duro en muchos casos, y que muchas empresas aprovechan ese sueño, esa vocación para estirar lo máximo posible a sus trabajadores. Trabajando en una empresa para otra industria, esto también lo he visto; pero parece que en el ámbito de los videojuegos se ve más comúnmente, como si fuera algo intrínseco a ello; y es algo peligroso y que espero que algún día cambie.

8. Glosario

Los siguientes términos y acrónimos se visualizan en orden alfabético.

- ***Asset(s).***
  - Representación de cualquier objeto que puede ser utilizado en el juego o proyecto. Puede provenir de cualquier archivo ajeno al motor del juego (Unity) como un modelo 3D, una imagen, un archivo de audio; o cualquiera de los tipos de archivos propios del motor del juego.

- ***Boss Fight.***
  - Nivel de juego contra un enemigo importante en la historia o mecánicamente exigente y complejo. Sirven como clímax de una sección particular del juego, como al final de un nivel o fase; o defendiendo un objetivo específico.

- ***Bone/Skeletal Animation.***
  - Técnica de animación por ordenador en la que un personaje (u otro objeto articulado) es representado en dos partes: Una representación superficial, donde el personaje está dibujado (llamado *mesh* o *skin*) y un conjunto jerárquico de partes interconectadas (denominada *bones*, que colectivamente forman el esqueleto, o *skeleton* o *rig*)

- ***Collider* (Unity).**
  - Son áreas que envuelven al *GameObject* utilizadas para detectar colisiones entre diferentes objetos del juego.

Está formado de una determinada **forma** y un **material físico**.

- ***Concept Art.***
  - Ilustraciones cuyo objetivo es dar una representación visual de un diseño, idea y/o estado de ánimo para su uso en películas, videojuegos, animación o cómic antes de que se realice en el producto final.

- ***Cooldown.***
  - Enfriamiento; tiempo que necesita una acción en un videojuego para volverse a ejecutar. Suele verse en habilidades o armas en las que se debe esperar un determinado tiempo antes de que dicha habilidad vuelva a estar activa o que el arma vuelva a disparar.

- ***Crunch.***
  - Periodos en los que los desarrolladores trabajan por encima de las horas estipuladas, normalmente para llegar a tiempo a una fecha acordada.

- ***Dash.***
  - Esquiva, desliz. En videojuegos es un movimiento rápido utilizado para ganar velocidad, avanzar más rápido, y, en algunos casos, para obtener invulnerabilidad momentánea.

- ***FSM* (*Finite State Machine*).**
  - Es un modelo matemático de computación; una máquina abstracta que puede estar en cada momento en un estado exacto de un número finito de estados. La *FSM* puede cambia de uno a otro estado en respuesta a diferentes señales (*inputs* o entradas); el cambio de un estado a otro se llama transición.
  - Una *FSM* es definida por una lista de sus estados, su estado inicial, y los *inputs* que activan cada transición.

- ***Game Engine* (Motor Gráfico o Motor de Juego).**
  - Es un ambiente de desarrollo de software diseñado para desarrollar videojuegos. La funcionalidad principal del motor incluye un motor de renderizado (*renderer*) para gráficos 2D y/o 3D, un motor de físicas o detecciones de colisión, sonido, *scripting,* animación, inteligencia artificial, *networking*, *streaming*, gestión de memoria, *threading*, soporte de localización…

- ***Indie.***

    - Videojuegos independientes. No existe una definición exacta, pero todos comparte rasgos comunes: Son desarrollados por pocas personas, equipos o compañías pequeñas e independientes. Por lo general disponen de poco presupuesto y no tienen respaldo financiero por publicadores.
    - Al ser independientes, los desarrolladores no tienen limitaciones creativas y no requieren de aprobación editorial.

- ***Inverse Kinematics* (*IK*).**
    - En animación por ordenador y robótica, *Inverse Kinematics*, o Cinemática Inversa, es un proceso matemático para calcular los parámetros de las articulaciones necesarios para colocar el final de la cadena cinemática, como el manipulador de un robot o la mano de un personaje, en una posición y orientación dada relativa al comienzo de la cadena.

- ***Knockback.***
    - Del inglés, golpear hacia atrás. Es el efecto de un golpe que empuja al personaje afectado hacia atrás, generalmente dejándolo también aturdido (*stunned*).

- ***Marketplace.***
    - Sitios web que reúnen una gran cantidad de compradores y vendedores de distintas partes del mundo, cumpliendo de intermediario.

- ***Metroidvania.***
    - Subgénero de videojuegos de acción-aventura basado en un concepto de plataformas no lineal. Sus bases lo asentaron las sagas de Metroid y Castlevania; de ahí el nombre de dicho subgénero.

- ***Pathfinding.***
    - Una de las áreas de estudio más amplias de los videojuegos y que se basa en algoritmos para encontrar un camino concreto desde un punto a otro según el mapeado. En la mayor parte de los casos busca el camino más corto entre la posición actual y el objetivo.
    - Los algoritmos más conocidos son el de Dijkstra y el algoritmo A* (A estrella o A *star*).

- ***Placeholder.***
    - Versión muy básica, icono o señal que sirve de indicador de un elemento que forma parte del videojuego (objeto, personaje, evento, etc.), pero que todavía no ha sido desarrollado o diseñado.

- ***Plataformas* (Género de Videojuegos).**
    - Género de los videojuegos que se caracterizan por tener que caminar, correr, saltar o escalar sobre una serie de plataformas y acantilados, con enemigos, mientras se recogen objetos para poder completar el juego.
    - Este tipo de videojuegos suelen usar vistas de desplazamiento horizontal. Es un género muy popular de los videojuegos, surgido a comienzo de la década de 1980 y que sigue manteniendo bastante popularidad en la actualidad.

- ***Scriptable Objects.***
    - Funcionan como contenedores de datos que se pueden usar para almacenar grandes cantidades de datos, independientemente de las instancias de la clase.
    - Uno de los usos más comunes para los *Scriptable Objects* es para reducir el uso de memoria del proyecto, evitando copiar los valores de dicho objeto.
- ***Skinning.***
    - Técnica de animación donde un modelo es animado usando un esqueleto o *rig,* al que finalmente se le aplica una piel o *skin* que sirve como textura del modelo. (Ver *bone/skeletal animation* más arriba).

- ***Sprite.***
    - Inicialmente, un *sprite* era una imagen de mapa de bits que era gestionada por un hardware especializado, independiente de la CPU de la máquina, usado generalmente para los protagonistas de los videojuegos u otros personajes, ya que permitía gestionarlos de manera independiente a los fondos.
    - Con el paso del tiempo el término *sprite* acabó por hacer referencia a cualquier imagen de mapa de bits presente en pantalla, generalmente refiriéndose a los personajes del juego.
    - En programación de videojuegos es la máscara o imagen de un objeto que tiene la capacidad de colisión.

- ***Tilemap.***
    - Componente de *Unity*. Es un sistema que almacena y gestiona *Tile Assets* para crear niveles en 2D. Transfiere la información de las *Tiles* (Baldosas) colocadas en él a otros componentes relacionados como el *Tilemap Renderer* y el *Tilemap Collider 2D.*

- ***Tile Palette.***

    - Conjunto de *sprites* que conforman una paleta para ser usada sobre un *tilemap.*

- ***Walking Simulator.***
    - Género de los videojuegos que rechazan las mecánicas clásicas para apostar por una narrativa intensa y unos escenarios desbordantes de personalidad.
    - Suelen ser de escasa duración, pero capaces de dejar una profunda huella por su inmersiva historia.

- ***Waypoints.***
    - Conjunto de puntos que definen un camino a seguir por una Inteligencia Artificial; de forma que para pasar al siguiente punto debe haber alcanzado previamente el punto definido anteriormente.

9. Bibliografía

[1] The Literary Platform. "*The evolution of video game narrative."* [Fecha de consulta: 18/05/2020]

[2] ORTEGA, José L. (2017). IGN. "*Los mejores Walking Simulators. Un género que se abre paso en el mundo del videojuego".* [Fecha de consulta: 22/04/2020]

[3] Wikipedia. "*Videojuego de plataformas*". [Fecha de consulta: 23/04/2020]

[4] SORIANO, Marc. (2011). alumni.cs.ucr.edu/ "*Skeletal Animation*". [Fecha de consulta: 03/05/2020]

[5] GARCÍA PEÑALVO, Francisco José; GARCÍA HOLGADO, Alicia; VÁZQUEZ INGELMO, Andrea. Ingeniería de software I (*Tema 6: Flujos de Trabajo del Proceso Unificado*). Universidad de Salamanca, Departamento de Informática y Automática.

[6] (2017) Wordpress: Planeación de Sistemas de Información. "*Diagrama de Componentes Empresarial*" [Fecha de consulta: 20/05/2020]

[7] (2018) Youtube - Brackeys. "*SCRIPTABLE OBJECTS in Unity*". [Fecha de consulta: 05/04/2020]

[8] learn.unity. "*Pluggable AI With Scriptable Objects*". [Fecha de consulta: 06/04/2020]

[9] REICHERT, Kate. (2012). Gamasutra.com "*Top 5 Problems Face By Indie Game Developers*". [Fecha de consulta: 20/04/2020]

[10] SCHREIER, Jason. (2016). Kotaku.com "*The Horrible World of Video Game Crunch*". [Fecha de consulta: 19/05/2020]

[11] YIN-POOLE, Wesley. (2020). Eurogamer.net "*9 years after launch, Terraria is finally complete. Journey's End*". [Fecha de consulta: 22/05/2020]

[12] LaMoncloa.gob.es. "*El Gobierno decreta el estado de alarma para hacer frente a la expansión del coronavirus COVID-19*". [Fecha de consulta: 21/05/2020]