UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Fuseworks: videojuego de sigilo en tercera persona usando Unreal Engine 4

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Daniel Reig Martínez

Tutor: Ramón Pascual Mollá Vayá

Curso: 2019-2020

Resumen

Este TFG consiste en el desarrollo de un videojuego de sigilo en tercera persona utilizando el motor gráfico *Unreal Engine v4*. Se emplea el sistema de programación visual propio del motor (*Blueprints*) para programar el comportamiento de juego, así como los *assets Behaviour Trees* y *Blackboard* para la gestión de la *IA*. El proyecto se realiza desde cero y funciona en Windows.

El videojuego tiene temática *sci-fi*. Está ambientado en la ciudad de Avico que, gobernada por el grupo Kao-Fouque planea un ataque con androides a escala mundial. Detenerlos será el trabajo del agente Wraith.

Palabras clave: unreal engine, videojuego, windows, ia, sigilo, blueprints, Fuseworks

Abstract

This TFG consists in the development of a third person stealth game using the game engine Unreal v4. The engine's visual programming system (Blueprints) is being used to code the behaviour of the game as well as the assets Behaviour Trees and BlackBoard to manage AI. The project is being developed from scratch and it works on Windows.

The game has a sci-fi setting. It is set in the city of Avico that while ruled by the Kao-Fouque group plans a droid attack on a global scale. Stopping them will be agent Wraith's job.

Keywords: unreal engine, game, windows, ai, stealth, blueprints, Fuseworks

Índice

Índice de figuras

Índice de Diagramas

1. Introducción

El documento describe el desarrollo de un videojuego creado como parte del Trabajo de Fin de Grado (TFG) del grado de ingeniería informática.

El juego se ha desarrollado en el motor *Unreal Engine 4*, usando además varias de sus tecnologías como *Blueprints* y *Behaviour Trees*, que agilizan el proceso de desarrollo a la hora de controlar el comportamiento del videojuego. Además, tanto los modelos como las animaciones de estos se han adquirido a través del *store* propio del motor.

Se pretende demostrar los conocimientos adquiridos durante el grado de ingeniería informática, más concretamente de la rama de ingeniería del software, por lo que también se estudiarán las distintas características y mecánicas propias del género del sigilo y su consiguiente implementación, a la hora de realizar el análisis y posterior diseño de los requisitos del producto.

La intención no es la de crear un juego completo, el objetivo es el de implementar algunas de las características más comunes del género y otras propias con la intención de realizar el prototipo de un posible juego que se podría desarrollar con las herramientas escogidas.

1.1 Motivación

Tras haber trabajado en varios videojuegos sencillos, tanto durante el transcurso de la carrera como en mi tiempo libre, decidí que el desarrollo de un juego como TFG sería lo adecuado, además como motivación adicional me propuse el uso de *Unreal Engine* como motor gráfico, uno de los motores más importantes en la industria y con el que no había realizado ningún proyecto de este alcance.

Con la disposición de herramientas de desarrollo como *Unity* o *Unreal Engine* [15], el desarrollo de videojuegos está al alcance de todos y no solo al de grandes equipos de desarrolladores. Sin embargo, no por ser más accesibles se tratan de proyectos sencillos, necesitan de conocimientos de Ingeniería del Software, como el análisis de requisitos y soluciones, el diseño e implementación de dichas soluciones y la realización de pruebas para comprobar el correcto funcionamiento de la aplicación, y es por lo que considero que este proyecto es el adecuado para este tipo de trabajo, permitiendo demostrar los conocimientos adquiridos en el grado.

1.2 Objetivos

Este TFG tiene como objetivo principal el desarrollo de un juego de sigilo en tercera persona usando el motor gráfico *Unreal Engine*, de forma que adquiera experiencia en el uso de este tipo de herramientas, así como en el proceso general de creación de un videojuego.

Estudiar e implementar las características y mecánicas comunes en los juegos del género del sigilo, teniendo en cuenta las decisiones que otros desarrolladores de la industria del videojuego han tomado y adaptando estas soluciones a las necesidades y restricciones del proyecto.

Correcto funcionamiento de todos los elementos del juego, es decir que todo desde los botones del menú hasta el comportamiento de los enemigos funcionen de la forma esperada sin ninguna anomalía, cumpliendo los requisitos especificados en su diseño.

La intención no es desarrollar un producto comercial sino mostrar algunas de las posibilidades que ofrece esta tecnología a la hora de crear un juego de este género.

1.3 Metodología

La metodología elegida para el desarrollo del proyecto es la *Scrum* [1], adaptada al trabajo individual, que ayudará a definir las funcionalidades que se implementarán en el juego desde una fase temprana, además de permitir una mayor autoorganización en el desarrollo del proyecto.

Siguiendo esta metodología, al inicio del proyecto se divide el juego en una lista de funcionalidades (*Backlog*), que se agrupan en "*Sprints*", periodos de tiempo de desarrollo, tras los que se evalúa el progreso realizado y se deciden los objetivos del siguiente teniendo en cuenta los progresos realizados en el anterior.

Esta metodología ofrece una visión general de las necesidades y funcionalidades deseadas del proyecto teniendo en cuenta el progreso real y permitiendo cambios en el juego según sean necesarios.

1.4 Estructura del documento

En este apartado se presenta la estructura del documento describiendo el contenido de los siguientes capítulos.

Como primer apartado se discute el estado del arte, donde se presenta es estado actual de la industria de los videojuegos y de los juegos similares al que se desea desarrollar, discutiendo tanto la historia como las mecánicas comunes de los mismos.

A continuación, encontramos con el análisis del problema donde se proponen los requisitos que el proyecto deberá de cumplir y se estudiaran las soluciones a los requisitos propuestos.

El tercer punto es planificación, su función es la de mostrar la planificación temporal de las tareas necesarias para completar el proyecto, siguiendo la metodología de desarrollo escogida.

El siguiente punto será el de diseño, en él se desarrollan las ideas propuestas en el análisis de problemas y se consolidan como soluciones implementables en el desarrollo. Adicionalmente, en este apartado encontramos otros comentarios como justificaciones sobre las tecnologías escogidas y el diagrama de flujo de la aplicación.

En el apartado de desarrollo de la solución, se discuten los problemas encontrados en la implementación de la solución y los métodos utilizados para cumplir con las mecánicas propuestas y diseñadas en los apartados anteriores.

Por último, encontramos apartados dedicados a las conclusiones y trabajos futuros, donde se reflexiona sobre el proyecto, los objetivos iniciales y el desarrollo general del trabajo.

Además, se incluyen anexos y referencias que se han utilizado en el desarrollo del trabajo, así como un glosario con términos poco conocidos o específicos del desarrollo de videojuegos y un documento que detalla elementos más relacionados con el diseño de videojuegos.

2. Estado del arte

Esta sección consistirá en un breve estudio de la situación del mercado actual de los videojuegos, así como del género del sigilo, con el objetivo de identificar las características que hacen que un juego pertenezca al mismo, además de los problemas o inconvenientes que tienen otros videojuegos de sigilo.

2.1 Mercado actual de los videojuegos

La industria del videojuego forma parte de la más amplia industria del entretenimiento, y como tal ha experimentado un gran crecimiento en la última década. Esto es debido a que hoy en día ya no solo están restringidos a los PC y las consolas tradicionales, sino que también se pueden encontrar en los teléfonos móviles y redes sociales.

Este aumento de los canales de distribución es el principal causante del crecimiento de la industria, dando a los creadores acceso a una audiencia más heterogénea compuesta por personas de todas la edades y géneros, muchas de las cuales no acceden a los videojuegos por las vías más tradicionales (PC y consolas).

A continuación, se muestra un estudio detallando la situación actual por dispositivo, así como el crecimiento por año que experimentan cada uno de estos segmentos [2].

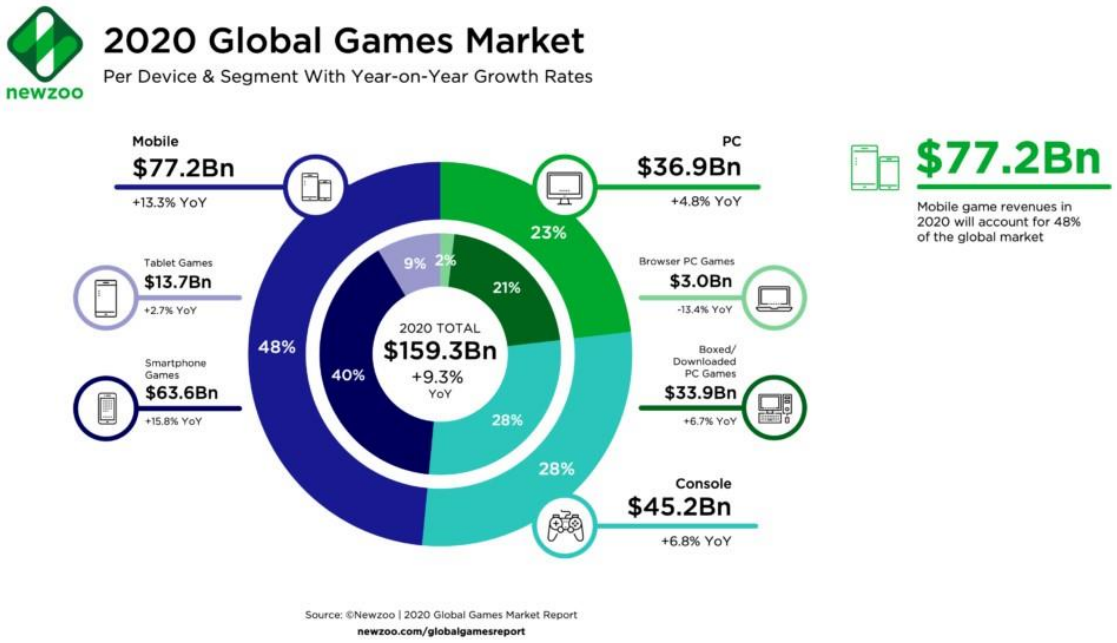


Figura 1: Situación actual del mercado

Como se puede observar en la gráfica el tipo dispositivo predominante a la hora de consumir videojuegos es el móvil, que reúne tanto tabletas como teléfonos inteligentes y que conforma un 48% del dinero de la industria, además, se espera un crecimiento mayor en este tipo de dispositivos que del resto.

Sin embargo, de los sectores más tradicionales de la industria, como pueden ser el ordenador personal o las consolas, también se espera que continúen su crecimiento y tan solo los juegos en el navegador pierden cuota de mercado, probablemente debido a que gran parte de su público se ha trasladado al mercado móvil.

Pese a formar parte de la misma industria, se pueden distinguir dos mercados diferenciados:

El mercado móvil, con una mayoría de juegos *free-to-play* que obtienen sus ingresos de las micro transacciones y la publicidad, hace que la barrera de entrada sea muy baja debido a no tener que pagar para jugar y al poder hacerlo en dispositivos que tiene la mayoría de la gente. Este tipo de juegos además suelen tener un tiempo y equipo de desarrollo menor y su tiempo de vida es mayor a los de la industria tradicional. La baja barrera de entrada y los menores costes de desarrollo permiten maximizar los ingresos de las empresas desarrolladoras y es por esto por lo que vemos una mayor cuota de mercado comparada con otros dispositivos. [3] El mayor tiempo de vida está relacionado con uno de los aspectos más valorados a la hora de desarrollar juegos móviles, la recepción que es vital a la hora de medir el potencial de los juegos móviles.

El mercado más tradicional, que consiste en ordenadores personales y consolas, obtiene gran parte de sus ingresos de la venta de sus juegos, cuyo precio suele oscilar entre 60 y 70 euros, y en muchos casos esta sería la única fuente de ingresos del videojuego, los cuales tienen costes de desarrollo mayores que los del mercado móvil. Como consecuencia, las desarrolladoras de este tipo de videojuegos han empezado a emplear mecanismos propios del mercado móvil, como las micro transacciones y la ampliación del tiempo de vida para poder seguir obteniendo beneficios una vez lanzado el juego.

Teniendo en cuenta estos datos y tipo de juego que se quiere desarrollar, la plataforma idónea es el PC, tanto por la cuota actual de mercado y su tendencia de crecimiento como por las facilidades a la hora de desarrollar que ofrece esta plataforma.

2.2 Videojuego de sigilo[1]

El género del sigilo es un subgénero de los videojuegos de acción en los que las mecánicas y sistemas de este invitan al jugador a evitar o superar a sus enemigos mediante el uso del sigilo, evitando así el enfrentamiento directo. Este género suele combinarse con elementos de otros géneros como *Shooters*, plataformas, o *RPGs*.

El origen de estas mecánicas se remonta a juegos como *Pac-man (1980)* [4] que pese a su naturaleza arcade ya cumple con la definición de un juego del género, para completar el nivel y avanzar, el jugador trata de evitar a los enemigos.

Sin embargo, muchos atribuyen a *Castle Wolfenstein (1981)* el título de primer juego de sigilo, ambientado en la segunda guerra mundial, consiste en atravesar una serie de niveles, obtener los secretos (objetivo) y huir del nivel. La escasez de munición y la

[1] Los detalles de los juegos que se mencionan en este apartado pueden encontrarse en el anexo “Juegos del Género del Sigilo” página 78.

dificultad de los enemigos invita al uso del sigilo, estableciendo así las primeras características del género [5].

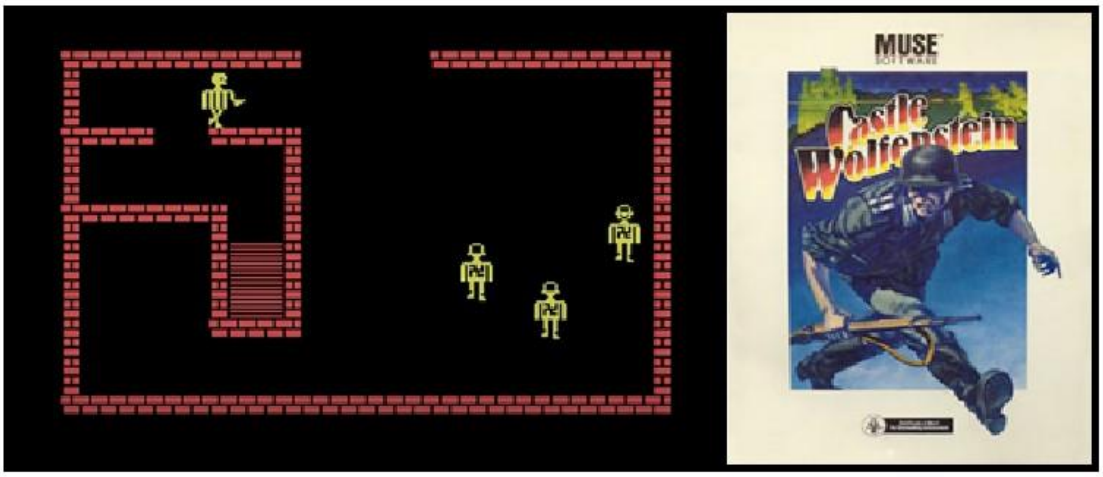


Figura 2: Castle Wolfenstein

Será en 1998 cuando el género encuentra éxito con el gran público con juegos como *Metal Gear Solid*, *Tenchu: Stealth Assasins* y *Thief: The Dark Project*. Una de las razones por las que el género gana popularidad se puede achacar a el salto a los gráficos 3D que se dio en esta misma época y de los que este tipo de juegos se beneficia enormemente gracias a la ambientación e inmersión que estos aportan.

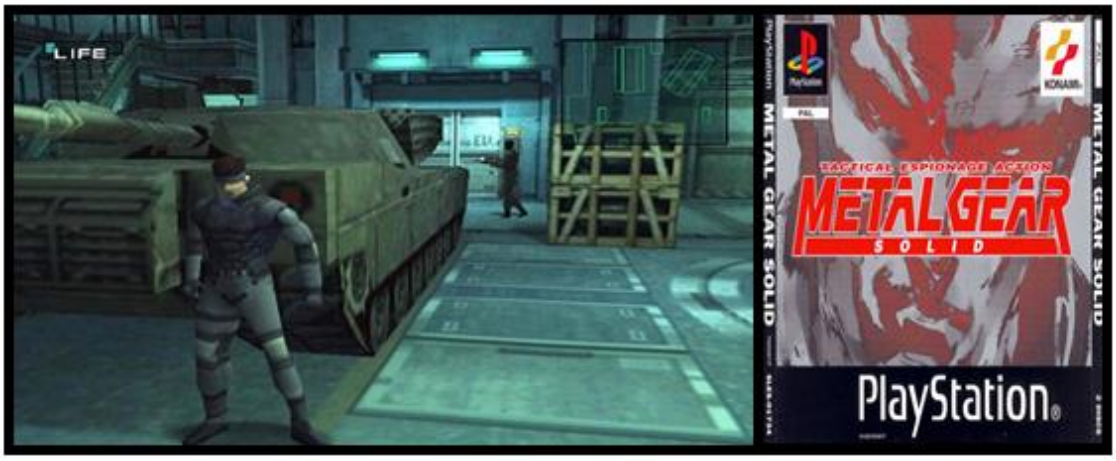


Figura 3: Metal Gear Solid

El género continuó evolucionando, introduciendo diferentes mecánicas que añaden complejidad y aspectos únicos a cada uno de los juegos, ejemplos de estas son el sistema de visibilidad basada en la iluminación (*Thief* o *Splinter Cell*), el sistema de camuflaje (*Metal Gear Solid 3*) o elementos sociales como la capacidad de ocultarse entre multitudes de gente (*Assassin's Creed*).

Como se ha mencionado anteriormente, el género del sigilo es especialmente idóneo a la hora de ser combinado con otros géneros ya que es una forma relativamente sencilla de añadir variedad y potencialmente diferenciarse de la competencia, así como reforzar la narrativa y la ambientación del juego. Algunos de los géneros con los que se suele

combinar son: *Survival Horror* (*Alien Isolation (2014)*), *Shooters* (*Far Cry 3(2012)*) o plataformas (*Sly Cooper (2002)*).

Figura 4: Sly Cooper

En los últimos diez años los juegos el género del sigilo ha madurado y aunque por sí mismo no es tan popular como una vez fue, la profundidad de sus mecánicas se ha incorporado en un gran número de juegos modernos. A continuación, se destacan algunos de los diez mejores juegos de la década dentro de este género según su puntuación en *Metacritic* [6]*:*

- *Batman: Arkham City (2011):* juego protagonizado por *Batman,* es una secuela al *Batman: Arkham Asylum* de 2009. Combina aspectos del sigilo con un mundo abierto y un combate de estilo *Beat 'em up*.
- *Metal Gear Solid V: The Phantom Pain (2015):* la última entrega de la saga clásica de *Metal Gear Solid*. En él se apuesta por introducir mecánicas de mundo abierto encima de una serie de mecánicas y sistemas muy potentes que marcan un nuevo hito en el género y del videojuego sistémico en general.
- *Mark Of The Ninja (2012):* el único juego indie que encontramos entre los diez mejores según *Metacritic*. Es un juego 2D en el que controlamos a un ninja y hacemos uso del sistema de luces y sombras paro ocultarnos de los enemigos y completar el nivel.
- *Tom Clancy's Splinter Cell: Conviction (2010):* es el juego de sigilo más lineal que encontramos en esta lista, se apoya en los componentes narrativos para mantener la atención del jugador, pero mantiene muchos de los *gadgets* característicos de la saga, animando al jugador a buscar alternativas al enfrentamiento directo.

2.3 Crítica al estado del arte

El problema al que se enfrentan los juegos del género actualmente es la transformación del sigilo en una táctica, en lugar del pilar principal sobre el que se construye el juego, se ha integrado en juegos donde la jugabilidad invita a un comportamiento más basado en la acción directa que va en contra de los conceptos que definen los juegos de sigilo más puros al aumentar la variedad y la ambición de

sus mecánicas[2]. Esta combinación con elementos de otros géneros fuerza a los fans a pasarse a otros videojuegos de distintos géneros, que no les ofrecen la experiencia de sigilo que buscan.

Tan solo con volver a mirar la lista de los mejores juegos de sigilo según Metacritic [6], se puede apreciar una falta de lanzamientos en los últimos tres años y unas ambientaciones en su mayoría contemporáneas o ligeramente futuristas, lo que deja espacio para innovar en este aspecto y ofrecer a los jugadores del género nuevas experiencias.

2.4 Propuesta

Tras haber analizado el estado en que se encuentran otros juegos del mismo género y los problemas presentes en el género, se lleva propone la creación de este videojuego con el objetivo de paliar los dos problemas que encontramos en el mercado actual, la falta de variedad temática y la escasez de lanzamientos de calidad en los últimos años.

Por lo tanto, este juego, de título "*Fuseworks",* tiene la intención de plantear una ambientación basada en la ciencia-ficción más futurista que no podemos encontrar entre los juegos más destacados de la última década, aplicando las mecánicas comunes de los juegos de este género de forma que los seguidores de este tipo de juego encuentren la experiencia que estaban buscando y añadiendo algunas mecánicas propias que lo diferencien del resto.

[2] Referencia a un documento de opinión sobre el descenso de la popularidad del género - https://venturebeat.com/2010/01/10/the-stealth-handbook-for-the-seventh-generation/

3. Análisis del problema

Tras el estudio sobre el estado del arte, llega el momento de realizar un análisis del problema, que contará de tres partes y tras el cual tendremos una idea clara de la solución a implementar.

En primer lugar, se realizará un análisis de requisitos, en el que se enumeraran una serie de objetivos que el proyecto tendrá que cumplir para ser considerado un éxito, y la forma en que se comprobará el cumplimiento de dichos objetivos.

A continuación, procedemos a examinar una serie de soluciones propuestas para cumplir cada uno de estos requisitos, repasando sus ventajas e inconvenientes.

Por último, se propone una solución final, argumentando las decisiones tomadas en función a las características y restricciones de este proyecto.

3.1 Análisis de requisitos

Interfaz de usuario

La interfaz de la aplicación estará compuesta de menús y HUD (Head-Up Display), ambos tipos de interfaz deberán ser fácilmente legibles de forma que muestren al jugador la información necesaria en cada momento

La interfaz de usuario es uno de los principales puntos de interacción con el jugador. Su objetivo es el de ofrecer la información necesaria para que el jugador interactúe con el juego de manera intuitiva, guiándole de forma que entienda lo que el juego requiere de él o donde está la información que está buscando.

[9] En el diseño de interfaces para videojuegos hay que tener en cuenta cuatro principios:

1. Entorno: tener en cuenta el sistema en que se jugará, ya que cada uno de ellos tiene sus limitaciones (mando, ratón, tamaño de pantalla, distancia a la pantalla, resolución, ...).
2. Contenido: dependiendo del diseño del juego, se mostrará más o menos información al jugador.
3. Diseño visual: la interfaz ha de seguir un mismo estilo de arte de forma que no se confunda al jugador.
4. Arquitectura de la información: organizar los elementos de la interfaz dependiendo de su importancia para el jugador, con el objetivo de que sea fácilmente localizable en cada momento.

Para que el diseño y la implementación de la interfaz pueda considerarse un éxito, estos cuatro principios han de ser respetados y ser coherentes con el juego que se desarrolla, lo que significa un estudio de las posibles soluciones que se presentan a este requisito y que se discutirán en el siguiente apartado, análisis de soluciones.

Antes de concluir, es importante analizar las necesidades que el usuario puede tener en relación con sus interacciones con la aplicación. Se distingue entre las relaciones entre el usuario y los menús del juego y en puntos posteriores se analizará su relación con la aplicación en términos jugables.

El análisis se ilustrará por medio de diagramas de casos de uso siguiendo el estándar UML:

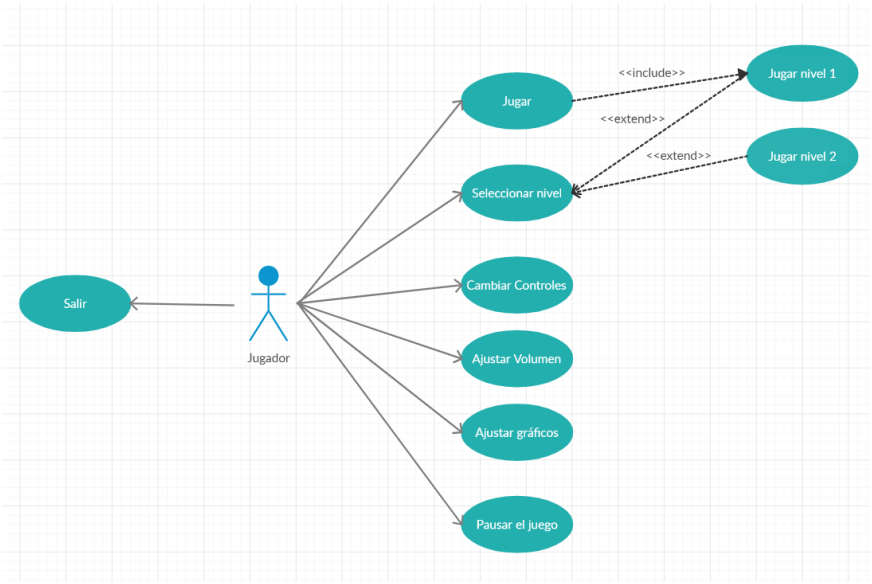


Diagrama 1: Casos de uso menús

Todos los casos de uso presentes en el diagrama han de ser contemplados por la aplicación y ejecutables desde los menús, algunas de estas acciones se realizarán mediante el uso de botones, *sliders*, y otros elementos de la interfaz y otras mediante pulsaciones de botones en el momento que se ejecuta el juego en sí, pero siempre en relación con la interacción con los menús.

Inteligencia artificial

Al ser un juego de un solo jugador, se hace especialmente importante ya que será la única forma de ofrecer un reto al jugador. La IA indicará al jugador en todo momento el estado en el que se encuentra de forma que el jugador entienda lo que "piensa" el enemigo y pueda planear estrategias acordes a ello.

Por ello, hemos de diseñar los modelos de percepción de los enemigos de forma que la interacción con el jugador sea satisfactoria, para cumplir este objetivo seguiremos tres principios [11] :

- La IA tiene que ser justa: lo que esto significa es que ha de sentirse justa, y es "sentirse" la palabra clave aquí ya que, aunque técnicamente se cumpla con todas las reglas establecidas por el sistema, si el jugador cree que por ejemplo no debería haber sido detectado en una situación determinada, podría enfadarse y dejar de jugar, y es por esto que debemos crear sistemas fáciles de comprender, que den al jugador la sensación de estar en control de lo que sucede, pudiendo planear o formar estrategias a partir de dichos sistemas.

- La IA ha de proveer 'feedback': los enemigos comunicarán lo que están pensando en la medida de los posible, permitiendo al jugador formular estrategias alrededor de estas interacciones. Este 'feedback' se puede proveer tanto de forma orgánica, por ejemplo, el enemigo dice lo que piensa, como a través de sistemas externos, un ejemplo podría ser el HUD mostrando iconos o barras de estado.
- La IA ha de demostrar inteligencia: los sistemas no han de llevar al jugador a pensar que la IA es "estúpida", esto tampoco significa necesariamente que tenga que ser inteligente, lo que significa es que todos los sistemas han de parecer posibles, el enemigo no ha ido a por el jugador porque sí, ha ido porque le ha oído o le ha visto como sucedería en la vida real.

Para poder considerar que el funcionamiento de la inteligencia artificial se adapta a las necesidades de un juego de sigilo como es "Fuseworks", hay que asegurar el cumplimiento de estos tres principios de forma que se asegure que la experiencia del jugador es la óptima.

Aspecto audiovisual

El proyecto se plantea con tres requisitos en el apartado audiovisual que tendrán que ser respetados:

El primero es la fidelidad del juego, es decir el aspecto visual del juego, que tendrá una apariencia lo más realista posible, siempre respetando los otros dos requisitos, por ello, se necesitarán assets de gran calidad que sumerjan al jugador en la experiencia de juego.

Como segundo requisito se plantea alcanzar una tasa de 60 imágenes por segundo garantizando la máxima fluidez posible en cada momento, este aspecto puede afectar al requisito de fidelidad anterior, pero será necesario para proveer al jugador de una fluidez consistente que mejore su experiencia con el juego.

Por último, los aspectos relacionados con la obtención de los assets y herramientas relacionadas con el aspecto visual del juego deberán ser conseguidos sin ningún coste adicional.

Adicionalmente, los niveles y assets que se encuentren en ellos deberán respetar la temática sci-fi del juego asegurando que la inmersión del jugador no se vea interrumpida en ningún momento y ayudando a que los niveles presenten la consistencia que se espera de este tipo de proyecto.

Mecánicas del personaje principal

Las mecánicas del personaje representan las distintas acciones que puede realizar el jugador a través del personaje principal. Estas acciones permitirán al jugador interactuar con el mundo del juego y dar solución a los problemas que este le presente, por ello se deberá de disponer de un conjunto de mecánicas variadas que animen al jugador a experimentar distintas formas de completar los niveles.

Estas mecánicas no deberán presentar fallos, y sus intenciones se comunicarán al jugador por medio de mensajes en la interfaz.

3.2 Análisis de las soluciones

En este apartado se analizarán más detenidamente los requisitos enumerados en el apartado anterior, además de las posibles soluciones que se pueden desarrollar para cumplir dichos requisitos.

1. Interfaz de usuario

Existen diversas soluciones a la hora de desarrollar la interfaz del videojuego y según qué relación tengan los elementos de la interfaz con el mundo del juego, se pueden agrupar en distintas categorías [10]:

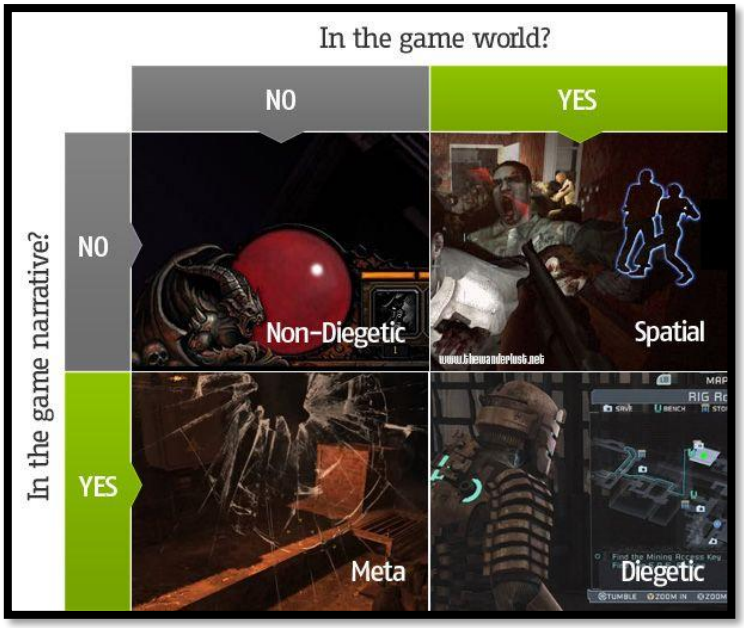


Figura 5: Tipos de interfaz

- Diegéticas: existen dentro del mundo del juego y pueden ser vistas y escuchadas por los personajes.
- No diegéticas: los elementos de la interfaz de muestran fuera del juego y solo son visibles por el jugador.
- Espacial: la interfaz cuenta con elementos que se representan en el espacio 3D del juego, puede existir en el mundo del juego o no pero su información está dirigida al jugador.
- Meta: forman parte de la historia del juego, pero no forman parte del espacio de este y tienen la intención de sumergir al jugador dentro de la experiencia del juego.

Estas categorías no se suelen aplicar por separado, sino que se combinan para ofrecer al usuario la mejor experiencia posible haciendo un balance entre las funciones y ventajas que ofrecen unas y otras.

Teniendo en cuenta los aspectos presentados, el diseño de la interfaz de "Fuseworks", contendrá elementos que pertenecen a las categorías que no influyen a la narrativa del juego debido a que los que sí lo hacen presentan un desafío en cuanto a tiempo de diseño, desarrollo y justificación narrativa que sería difícilmente afrontable con las restricciones que un proyecto como este supone y restarían tiempo de desarrollo a otras tareas

consideradas de mayor prioridad para el trabajo y el juego. Por lo tanto, en el juego se integrarán elementos de interfaz no-diegéticos y espaciales:

- Los elementos no-diegéticos compondrán la gran mayoría de la interfaz y estarán diseñados para ofrecer la mejor experiencia posible al usuario en cuanto que garantizaran que la información de la que requiere es fácilmente accesible. Algunos elementos de esta categoría serán el menú principal, de opciones, de pausa y elementos de la interfaz como la barra de vida o el contador de balas.
- Entre los elementos espaciales que se implementarán encontramos el medidor de sospecha y el indicador de objetivo que se situaran en el espacio 3D del juego para indicar la posición exacta de los elementos a los que representan.

2. Apartado audiovisual

Como solución a los requisitos de fidelidad, fluidez y precio, se pueden plantear diversas alternativas que pasan desde la fabricación de los distintos componentes por cuenta propia hasta la obtención de estos de terceros.

Uso de softwares de modelado

Respetando la intención de usar tan solo software de uso gratuito se plantean diversas tecnologías a la hora de modelar los *assets* 3D que podrían utilizarse en el proyecto, entre ellas destacan *Blender* [3]y *Daz Studio*[4].

El uso de este tipo de software proporciona una serie de ventajas y libertades a la hora de construir assets para un videojuego, pudiendo realizar completamente las ambiciones que el diseñador plantea sin estar limitado al trabajo ya realizado por otros.

Sin embargo, también supone una tarea de aprendizaje adicional además del tiempo que se dedicaría a la creación de cada uno de los assets necesarios para construir los niveles que conforman el juego. Adicionalmente todos los modelos que requiriesen de alguna animación tendrían un tiempo de fabricación adicional.

Obtención de assets de terceros

Otro medio por el cual sería posible conseguir los assets necesarios para construir el juego es por medio de terceros. Entre otros lugares donde podemos encontrar este tipo de contenido destacamos:

- *Unreal Engine Marketplace*[5]: es el store oficial del motor gráfico Unreal Engine y en el podemos encontrar artículos como sistemas de gameplay completos. niveles, modelos 3D y animaciones que ayudan a los desarrolladores a agilizar el proceso de desarrollo. Además, al tratarse del store propio de Unreal Engine cuenta con una integración mayor con el motor que otras plataformas similares.

[3] Blender: software de modelado 3D - https://www.blender.org/
[4] Daz Studio: software de modelado 3D - https://www.daz3d.com/
[5] Unreal Engine Marketplace: store de assets – https://www.unrealengine.com/marketplace/en-US/store

- *Sketchfab*[6] : plataforma en la que creadores comparten, venden y compran modelos 3D, contiene una gran cantidad de assets, pero muchos de ellos no están preparados para ser utilizados en juegos.
- *Cubebrush*[7]: similar al anterior consiste en una plataforma en la que se comparten modelos 3d, a diferencia de *Sketchfab*, en este sí que contamos con modelos ya preparados para su utilización en el desarrollo de juegos.

La decisión que se ha tomado con relación a la obtención de modelos 3D para este juego es la de usar los assets creados por terceros ya que las restricciones de tiempo que marca el desarrollo de este trabajo y la falta de experiencia en el uso de estas herramientas por parte del desarrollador hacen imposible que se modelen estos assets por cuenta propia. De entre las plataformas de distribución presentadas, se eligió el *Unreal Engine Marketplace* por las ventajas de integración con el motor de Unreal Engine que se utilizará en el proyecto y por presentar los assets que más se ajustan a la ambientación del juego.

De los modelos y assets disponibles en el store de Unreal Engine, se han seleccionado los siguientes para el desarrollo del trabajo debido a su relación con la temática del juego y la calidad con la que están construidos de forma que se cumpla las restricciones de fidelidad y fluidez requeridos:

- Assets pertenecientes a *Paragon*[8]: estos assets pertenecían a un juego de la compañía Unreal y fueron publicados al cerrar los servidores del juego por la propia compañía. Gracias a que pertenecían a un juego con un gran presupuesto, estos assets cuentan con una gran fidelidad visual tanto en los modelados como en las animaciones que vienen incluidas.
    - Paragon Wraith: con un aspecto futurista y un arma ya modelada, el modelo de Wraith es el modelo que junto con sus animaciones compondrá el personaje principal, ya que se adapta perfectamente a las necesidades que se plantean en el diseño del personaje.

Figura 6: Asset personaje principal

[6] Sketchfab: store de modelos 3D - https://sketchfab.com/
[7] Cubebrush: store de modelos 3D - https://cubebrush.co/
[8] Videojuego desarrollado por Epic Games – https://en.wikipedia.org/wiki/Paragon_(video_game)

    - Paragon Drongo: es el modelo que se usara para el enemigo, sencillamente por su aspecto visual y por las animaciones que ofrece, además al formar parte del mismo juego que el personaje anterior tienen un estilo de arte similar por lo que se mantiene la identidad visual del juego.
- Assets para los niveles: para los niveles se han localizado tres packs de contenido de ambientación *Sci-fi* que ayudaran a construir los niveles:
    - *Soul City*: formaba parte de una demo técnica del motor, habrá que adaptar el mapa que presenta para adaptarse a las necesidades de los niveles del juego.
    - *Modular Scifi Season 2 Starter Bundle*: paquete de modelos y efectos que agilizaran la construcción de los niveles mientras se mantiene la estética y temática principal.

Figura 7: Assets nivel

    - Scifi Hallway: otro paquete de modelados de gran calidad que servirán para terminar de completar la apariencia de los niveles.

Otro aspecto audiovisual a tener en cuenta, son las animaciones que vienen con una serie de condiciones debido a que [7] el videojuego es un medio pensado para ser interactivo, como consecuencia y especialmente en juegos en tercera persona, donde tienes el control total del movimiento del personaje y la cámara, las animaciones de los personajes tienen que ser legibles desde todos los ángulos, comunicando la intención y el impacto de cada acción. Es por esto por lo que, aunque los assets seleccionados ya cuentan con animaciones, será necesario adaptarlas a las necesidades del juego que estamos desarrollando, estas adaptaciones se discutirán en el apartado de desarrollo.

De forma similar el audio y otros assets que pueden conformar un nivel, deberán obedecer estos mismos principios para asegurar la mejor experiencia posible para el jugador y garantizar que se encuentra inmerso en el mundo y la narrativa del juego. Este tipo de recursos también será adquirido de terceros, utilizándose software de edición para adaptarlos a las necesidades del juego.

3. Inteligencia Artificial

En un juego de un solo jugador como Fuseworks, la inteligencia artificial (IA) cobra especial importancia ya que será la que se encargue de ofrecer un desafío al jugador.

Siguiendo los principios presentados en el análisis de requisitos, se definen cuatro modelos de percepción, en los que vamos a suponer que el enemigo o dueño de la IA intenta emular los sentidos propios de una persona, modelando estos sistemas a partir de los sentidos humanos:

1. Visual

En este modelo intentamos modelar el sentido de la vista, de forma que la IA pueda detectar y reconocer los objetivos que está viendo o que debería ver.

La vista del ser humano ve con claridad lo que está en el centro y va perdiendo precisión hacia los lados y en la distancia, además de tener una visión periférica de casi 180º, y aunque existen muchas posibilidades a la hora de definir este comportamiento, vamos a definir dos ejemplos generales acerca de cómo se modela este sentido en los videojuegos:

- Cono de visión

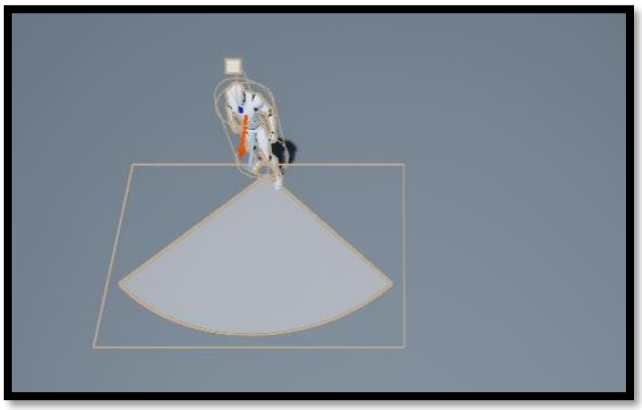


Figura 8: Cono de visión

Este es el modelo clásico que utilizan la mayoría de los videojuegos, define un área en forma de cono que se amplía en la distancia, es un sistema que funciona bien por su sencillez y es fácil de entender por el jugador [12].

Sin embargo, este modelo no es muy realista ya que no modela las distintas áreas en las que el ser humano ve con más claridad, aportando demasiada precisión a distancia y no la suficiente visión periférica. Soluciones a estos problemas suelen ser ampliar el ángulo inicial aportando más visión periférica, además de restringir la distancia de visión con lo que conseguimos una visión más equilibrada y justa pero todavía poco realista.

- Compuestas

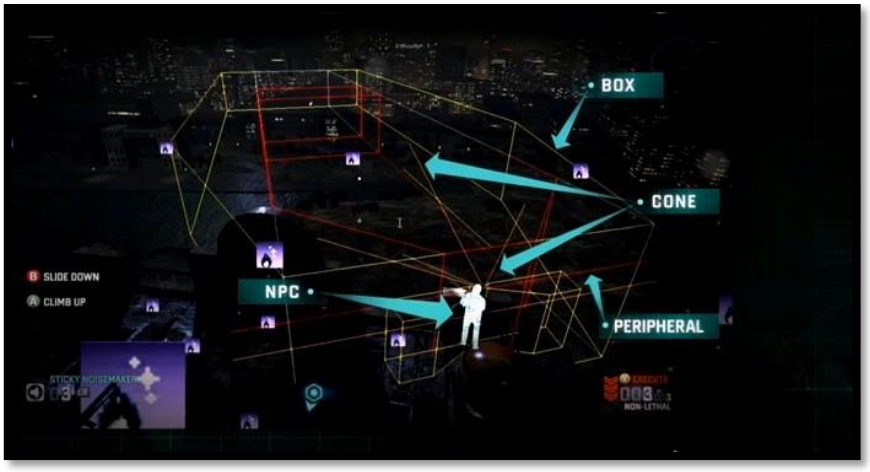


Figura 9: Modelo de visión compuesto

Muchos juegos utilizan distintas áreas de visión con las que representar más fielmente la visión de una persona, el que vemos en la imagen es el modelo usado en "Splinter Cell: Blacklist"[9] [11], está compuesto por zonas con forma de ataúd y es el que usaremos para explicar en qué consisten este tipo de modelos.

El modelo de la imagen está compuesto por dos zonas principales, una exterior y una interior que reflejan la visión periférica y el foco central de la visión humana, además también se han definido dos áreas auxiliares, un cono de visión a corta distancia en el que vera al jugador con facilidad y otra zona en la parte posterior del *npc* que modela el "sexto sentido". En tiempo de ejecución, estas áreas se comprueban en orden y se utiliza el tiempo de detección de la zona más inclusiva.

Cabe destacar, que las distintas zonas aportan una mayor flexibilidad al equipo de diseño permitiendo ajustar con mayor precisión tiempos y rangos de detección de una forma que no es posible con soluciones más sencillas.

El problema con este tipo de modelos es la complejidad, debido a la cantidad de zonas disponibles, además del tiempo que se tarda en programar en comparación con la visión en forma de cono, se añade el tiempo en que tardarían los diseñadores en encontrar el balance adecuado de tiempo de detección en cada una de las zonas y más si el juego ha de tener distintos niveles de dificultad.

Otro aspecto para tener en cuenta es que, aun siendo más realista, este sistema puede no casar bien con las mecánicas del juego lo que llevará al jugador a creer que no es justo y obligando a los programadores y diseñadores a crear casos específicos para las diferentes situaciones en que se sienta injusto, lo que requiere muchos test y tiempo.

[9] Juego desarrollado por Ubisoft - https://www.ubisoft.com/es-es/game/splinter-cell-blacklist/

2. Auditivo

En cuanto al sentido del oído la solución generalmente pasa por envolver al npc en una “esfera” cuyo volumen delimita la distancia de escucha de la inteligencia artificial. [12]

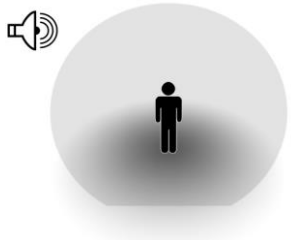


Figura 10: Esfera auditiva

Por supuesto, en esta solución se podrían generar varias esferas para limitar distintas áreas donde se escucharía con diferente claridad.

Algunos problemas que presenta esta aproximación, es el determinar que se debería oír y que no, por ejemplo, ¿se puede escuchar a través de una pared?, técnicamente si está dentro de la esfera sí, sin embargo, esto puede llevar a confusión al jugador o ser poco realista. Estas complejidades vienen dadas por la forma en que tiene el motor de simular la propagación del sonido en el entorno y será tarea de programadores y diseñadores el definir el comportamiento deseado en función a las necesidades del juego.

3. Ambiental

En este caso lo que intentamos emular no es un sentido, es la inteligencia o el razonamiento acerca del entorno que tiene la IA, por ejemplo, se acordará de que una puerta estaba cerrada y reaccionará al verla abierta [11].

Este tipo de inteligencia es muy amplia y se puede extender tanto como se desee, pero lo más importante es que la IA comunique lo que ha visto (“esta puerta estaba cerrada”) al jugador mediante sonido o alguna indicación en la interfaz, garantizando que el jugador entiende lo que ha pasado y quede claro que la IA es capaz de reconocer esa clase de situaciones haciéndola parecer más inteligente.

4. Social

Similar a la ambiental, esta consiste en la relación que tiene la inteligencia artificial con otras IAs de forma que es capaz de conversar con ellas y organizarse o reaccionar al jugador de manera conjunta. De nuevo, además del comportamiento en sí, es de suma importancia el comunicar estos comportamientos al jugador para que sea consciente de las capacidades de la inteligencia artificial [11].

Para concluir este apartado de la inteligencia artificial de los enemigos, se referencia al apartado de diseño de los enemigos para extraer cuales de estas soluciones se adaptan mejor a los comportamientos que este presentará.

- Como es necesario que sean capaces de localizar al jugador, se requiere la presencia del modelo visual, y de entre las alternativas presentadas, el cono de

visión presenta las características necesarias para el funcionamiento de este comportamiento, sin suponer el coste de desarrollo de otras soluciones.
- Para representar el comportamiento de investigación de ruido, se necesita la implementación del modelo auditivo, que se resolverá con la solución general que presentan la mayoría de los juegos, la esfera de escucha cuyo tamaño se definirá en el desarrollo dependiendo del tamaño del nivel en el que se encuentre el enemigo.

4. Mecánicas del personaje principal

El usuario tomará control de un personaje a través del cual interactuara con el mundo del juego, el cómo interactúa viene dado por las mecánicas/acciones que este personaje principal puede realizar [13].

Al tratarse de un juego de sigilo en tercera persona, podemos analizar dos de las características más comunes en este tipo de juegos:

1. Movimiento

Dependiendo de las intenciones del desarrollador se dota de más o menos libertad de movimiento al personaje principal, esto cambia la relación que se tiene con el entorno en el que se encuentra y puede ensalzar o restringir la forma en que se navegan estos niveles y las posibilidades de resolver los enfrentamientos.

Generalmente, estas restricciones de movimiento tienen que ver con la capacidad de saltar del personaje, existiendo tres sistemas principales:

- Salto libre: el personaje puede saltar libremente cuando y donde quiera, este el sistema que más libertad otorga, pero puede llevar a bugs o "glitches" no deseados.
- Sistema de cobertura: el juego tiene un sistema de coberturas que permite voltear obstáculos, es un sistema que aporta mucho control al equipo de desarrollo, que puede decidir los lugares a donde quieren que llegue el jugador, sin embargo, es difícil de implementar.
- Sin saltos: no se permite saltar al jugador, restringiendo el diseño del nivel.

De acuerdo con el documento de diseño del juego, la única opción posible será la del salto libre, y debido a que no se dispone del tiempo necesario para pulir los posibles fallos que este tipo de libertad puede causar, se utilizarán zonas de colisión invisibles para impedir que el jugador acceda a zonas del nivel a las que no debería.

2. Gadgets

Es un apartado amplio en el que cada juego innova de alguna forma, pero la mayoría de ellos se pueden agrupar en tres grandes grupos:

- Interactuar con el entorno: este tipo de gadgets se usan para afectar al nivel de alguna forma, ya sea para moverse a través de él o resolver puzles (por ejemplo, activando o desactivando interruptores a distancia). El uso de estos se limita a establecer una relación entre el nivel y el jugador, siendo complementario de las mecánicas de movimiento discutidas anteriormente.

- Interactuar con los enemigos: se trata de gadget u objetos que interactúan con la inteligencia artificial de los enemigos, van desde distracciones a camuflajes y ayudan al jugador a resolver los encuentros con enemigos de distintas formas, aumentando la inmersión del jugador y fomentando la creatividad.
- Recoger información: consisten en reunir información del entorno y suelen ser algún tipo de cámara, escáner o sensor que permiten localizar enemigos y objetivos, ayudando al jugador a formular estrategias sobre como afrontará el nivel.

Los gadgets que finalmente se implementaran en el juego, vienen definidos en el anexo de diseño (página 59), y se limitaran a interacciones con el entorno y con los enemigos debido a que el diseño de los niveles que se plantean en el juego no plantea situaciones en las que la recogida de información por parte de un gadget sea necesaria.

3.3 Solución propuesta

Partiendo de las soluciones del apartado anterior, se ha decidido optar por una solución que cumple con todos los requisitos propuestos y se adapta a las restricciones que supone este tipo de proyecto.

A continuación, se analiza punto por punto la solución propuesta a los distintos requisitos:

1. Interfaz de usuario

Siguiendo los principios establecidos en el análisis de soluciones, la interfaz de Fuseworks será diseñada para funcionar sin problemas tanto en PC como en consolas (1.), ya que no será necesario mostrar demasiada información al jugador y sus menús se recorrerán fácilmente tanto con mando como con ratón. Además, se ha optado por una interfaz no diegética, es decir que no existe en el mundo del juego y solo es visible para el jugador, también tendrá elementos espaciales situados en el espacio 3D del juego, como podría ser el indicador de sospecha de los enemigos o los objetivos que el jugador ha de alcanzar.

2. Audiovisual

Debido a las restricciones del trabajo, la mayoría de los assets se obtendrán de terceros ya que la calidad de estos será superior que si se construyesen desde cero y ahorraran tiempo de desarrollo que podrá ser empleado en otros aspectos del proyecto.

3. Inteligencia Artificial

Se cumplirá con los principios establecidos en el análisis de soluciones, creando sistemas sencillos que permitan al jugador planear alrededor de los mismos y entender la situación en cada momento. En cuanto al *feedback* que la inteligencia artificial proveerá al jugador, ser realizará a partir de la interfaz con un medidor de alarma sobre el enemigo además de líneas de voz sencillas que comunicarán al jugador los cambios y las intenciones de la IA.

En cuanto a los modelos de percepción (visual, auditivo, ambiental y social), se ha decidido implementar tres de ellos ya que el modelo ambiental necesita de más tiempo

de desarrollo para funcionar de manera óptima y sus beneficios solo estarían bien empleados en un proyecto de mayor alcance. Por lo tanto, se proponen soluciones a los tres modelos que se implementarán en este proyecto:

- Visual: la solución que se propone a la forma de simular la visión humana es la del cono de visión debido a sus ventajas a la hora de ser entendida por el jugador y su sencillez para ser ajustada dependiendo del diseño requerido.
- Auditivo: se envolverá a los enemigos en una esfera que delimitará la distancia de escucha de la IA.
- Social: este modelo presenta muchas opciones, pero se ha decidido implementar una en que las IAs se mantengan en contacto (en forma de conversación) unas con otras de forma que si pierden el contacto entre ellas entraran en estado de alarma. Estos comportamientos se comunicarán al jugador a través de la interfaz.

4. Mecánicas del personaje principal

En lo que respecta a las mecánicas con las que contará el personaje principal, se ha decidido otorgarle un movimiento totalmente libre de forma que atraviese los obstáculos del nivel con facilidad.

Además, el personaje dispondrá de distintos gadgets que le permitirán interactuar con el nivel, de los tres tipos de gadgets discutidos de ha decidido desarrollar los de interacción con el entorno y los enemigos debido a que son más fáciles de aplicar en el diseño de los niveles.

4. Planificación

En el siguiente apartado se discute la planificación seguida en el desarrollo del proyecto. Para ello siguiendo la metodología Scrum escogida para desarrollar el trabajo, dividiremos el proyecto en cuatro *sprints*. Para ayudar a visualizar la planificación y desarrollo temporal de los *sprints* un Diagrama de Gantt para después explicar en qué consistiría cada uno de ellos y qué objetivos pretenden cumplir.

Cabe destacar que al estar trabajando con una metodología ágil los objetivos de los diferentes sprints pueden variar en función al desarrollo del sprint anterior o a los conocimientos adquiridos durante el mismo. Esta flexibilidad a la hora de marcar los objetivos es especialmente importante en un proyecto como este en el que muchas de las tecnologías a utilizar deberán ser aprendidas antes del desarrollo de los componentes de la aplicación.

4.1 Planificación de los Sprints

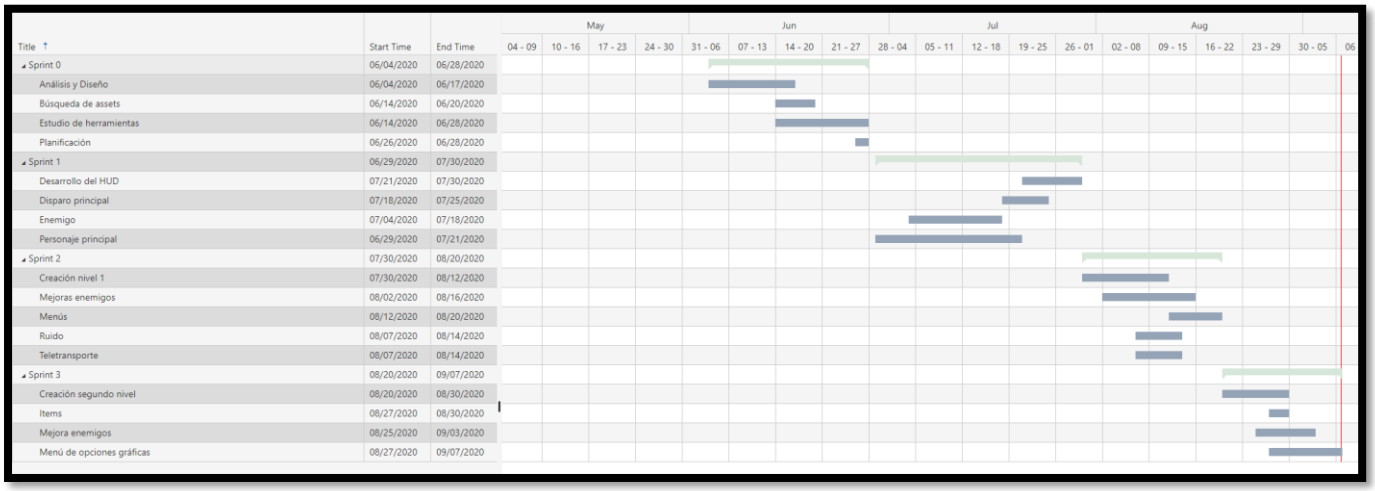


Diagrama 2: Diagrama Gantt Planificación

El proyecto se planeó con la intención de tener una duración de aproximadamente tres meses, en los que se trabajaría de media unas 23 horas a la semana lo que significa entre tres y cuatro horas al día. Los segmentos en los que se ha dividido el trabajo consisten en aprendizaje, diseño, implementación y redacción de la memoria, además de una reflexión sobre los objetivos conseguidos en el sprint acabado y las necesidades de cara al siguiente.

A continuación, pasamos a describir los objetivos de cada sprint, estos objetivos serán una estimación con las intenciones iniciales para cada uno, que estarán abiertas a cambios según avance el desarrollo.

Sprint 0

Antes de iniciar el desarrollo del juego, es importante realizar una serie de tareas que facilitarán el trabajo de los siguientes *sprints*, estas tareas de preparación consisten en un estudio del estado del género y análisis del problema que supone el proyecto. El

tiempo de este sprint se dedica también a la búsqueda de *assets*, modelos 3D y efectos que se utilizaran en el desarrollo.

Tras el análisis realizado, el proyecto se encontrará en condiciones de ser correctamente planificado y se definirán los objetivos que se pretenden alcanzar en los *sprints* siguientes.

Es también en este sprint donde se dedica la mayor parte de tiempo de aprendizaje de las herramientas a utilizar. Se estiman unos 24 días dedicados a este sprint de los cuales se dedican al menos 15 días a familiarizarse con las herramientas.

Sprint 1

Las intenciones de este *sprint* son las de implementar las mecánicas básicas del juego, lo que se ha estimado en 30 días de trabajo. Estas mecánicas básicas incluyen las tareas que se describen a continuación:

- Programación del movimiento del personaje principal: esto incluye la adaptación de las animaciones de las que disponía el personaje ya importado para adaptarlo a las necesidades del juego y la programación con relación a la respuesta que se da a los inputs del jugador.
- Mecánica de disparo: de la misma forma que con el movimiento se han tenido que adaptar las animaciones disponibles para que se adapten a las necesidades del diseño. Esta tarea también ha conllevado el estudio de cómo se implementan los proyectiles en el motor gráfico y la implementación en sí misma de estos.
- Desarrollo de la cámara principal: ha sido necesario modificar el funcionamiento de la cámara por defecto disponible para los juegos en tercera persona para que se adapte a los cambios al apuntar o cambiar el tamaño del personaje.
- Desarrollo de los enemigos: en este *sprint* se comienza a desarrollar el comportamiento de los enemigos aportándoles el sentido de la vista y la visión, además de la habilidad de perseguir al jugador. Ha sido necesario estudiar el comportamiento de los Behaviour Trees, un modelo matemático que define el plan de ejecución de la inteligencia artificial y más concretamente la implementación que encontramos en Unreal Engine 4.
- Variables del personaje principal: estas son contadores internos que regulan el comportamiento del personaje, más en concreto, un sistema para contar las balas disponibles y la vida en una escala del 0-100.
- HUD del personaje principal: diseño e implementación de los elementos que componen el HUD del personaje principal, estos son el punto de mira, el contador de balas y el medidor de vida.
- HUD de los enemigos: representación visual por medio de la interfaz del estado en el que se encuentra el enemigo.

Sprint 2

Con las mecánicas establecidas, en este sprint se construirá el primer nivel, y se desarrollarán más elementos de interfaz, iterando los establecidos en el anterior y añadiendo nuevos elementos en forma de menús, además de mejoras en el comportamiento de los enemigos.

- Creación del nivel: uso de los assets importados para construir el primer nivel.
- Integración con los elementos ya desarrollados: ajuste tanto de los componentes ya programados como de la estructura del nivel para que esta no cause problemas al resto.
- Mecánica de teletransporte: creación del proyectil de teletransporte y de las plataformas de teletransporte que se distribuirán a lo largo del nivel. También se implementarán una serie de animaciones y efectos que comuniquen que el teletransporte está teniendo lugar.
- Objetivo y salida: creación de la tarjeta de objetivo y la salida del nivel, además del indicador que comunica estos objetivos al jugador a través del HUD.
- Mecánica de ruido: proyectil de ruido que será una de las armas del personaje principal y su integración con la inteligencia artificial de los enemigos.
- Menú principal: desarrollo del menú principal desde el que el jugador accede a los distintos menús y al juego en sí.
- Menú de opciones: submenú accesible desde el principal y que despliega acceso a todas las opciones disponibles.
- Menú de controles: menú desde el que el usuario será capaz de cambiar el mapeo de los inputs que utilice, tanto teclado como mando.
- Menú de pausa: desarrollo de la capacidad de pausar la ejecución del juego y mostrar un menú de pausa desde el que acceder a las opciones, volver al menú principal o continuar la partida.
- Mejora enemiga: añade al enemigo la capacidad de disparar proyectiles al jugador, aprovecha los conocimientos adquiridos en el desarrollo de los proyectiles del personaje principal para adaptarlos al enemigo. También requerirá enlazar el impacto de estos proyectiles con la vida del personaje desarrollada anteriormente.

Sprint 3

En el último sprint se construirá el segundo nivel, y se corregirán los posibles fallos de integración que tengan las mecánicas con los niveles, además de implementar un menú de ajustes gráficos que permitirá adaptar el rendimiento del juego a distintos ordenadores. En este sprint es donde se realiza la mayor parte de la redacción de la memoria.

- Creación del segundo nivel: uso de los assets importados para construir el segundo nivel.
- Menú de opciones gráficas: menú accesible desde las opciones del juego que permite cambios en la calidad del juego con la intención de preservar la fluidez en sistemas menos capacitados.
- Menú de sonido: menú accesible desde las opciones de juego que permite ajustar el volumen general, de la música y de los efectos del juego.

- Comportamiento social de los enemigos: desarrollo del modelo social que permita que los enemigos parezcan interactuar entre sí y los indicadores por HUD que indican al jugador que esta interacción está sucediendo.
- Sonidos: añadir sonidos que comuniquen las acciones vistas en pantalla, tanto de los cambios de estado de los enemigos como de los disparos, efectos y mecánicas del personaje principal.
- Varios ítems: desarrollo de ítems que complementan el diseño de los niveles, como las cajas de balas o el ítem que desbloquea nuevas armas.

Reflexión sobre la planificación

Tras haber terminado el desarrollo del juego, se puede concluir que los objetivos que se querían alcanzar al final del proyecto eran acertados ya que todos ellos se han podido alcanzar, sin embargo, el progreso de los *sprints* se ha encontrado con complicaciones que han requerido reajustar los objetivos de estos.

En el primer *sprint* se encontraron problemas desarrollando los proyectiles, debido a que no impactaban correctamente ni con el escenario ni con los enemigos, este problema conllevo una tarea de investigación adicional con la que no se contaba en un principio que empujo tareas relacionadas con el comportamiento y los elementos de interfaz de los enemigos al segundo sprint.

En el segundo sprint, debido a la falta de experiencia del desarrollador en la creación de entornos 3D, se interpretó incorrectamente el tiempo de creación del primer nivel, y debido a que el tiempo consumido en la creación del nivel no estaba ofreciendo los resultados esperados, se decidió adaptar el mapa en el que venían los assets a las necesidades del proyecto de forma que se preservasen las intenciones de diseño del nivel. Por otro lado, las dificultades encontradas en el primer sprint y lo aprendido de ellas facilitaron la tarea de implementar las mecánicas de ruido y teletransporte, que se implementaron en menor tiempo del esperado inicialmente.

Por último, en el último segmento del desarrollo, se valoraron más correctamente las capacidades de creación de niveles desde el inicio, por lo que no se encontraron tantos problemas en este respecto. Sin embargo, el diseño del nivel creó ciertas necesidades que debían ser atendidas para asegurar que la experiencia del usuario final fuera la óptima, un ejemplo es la creación de un “disparador” invisible que activase el comportamiento de los enemigos poco antes de llegar el jugador para que estos encuentros muestren al jugador las intenciones del desarrollador sin depender del tiempo que haya tardado el jugador en llegar allí.

4.2 Presupuesto

En el momento en que se decide trabajar en un proyecto de estas características uno de los aspectos más importantes para tener en cuenta es el coste que tendrá desarrollarlo, por tanto, antes de empezar con el desarrollo es necesario estimar los gastos que conllevará.

La duración estimada del proyecto es de tres meses aproximadamente, durante este tiempo es necesario mantener al desarrollador que trabajará en él, el sueldo medio para un puesto junior oscila entre 17.000 y 21.000 euros brutos anuales [12], lo que significa

unos 1.500 euros al mes, por lo que en nuestro caso son 4.500 euros destinados a pagar al trabajador.

A el coste del trabajador habría que añadirle el coste de las herramientas utilizadas, pero en este caso todas las herramientas serán de uso libre por lo que no generaran ningún gasto. Teniendo en cuenta que el ordenador a utilizar requiere de una potencia suficiente para trabajar cómodamente en el entorno de desarrollo de Unreal se estima que el precio de sus componentes será de 800 euros.

Se va a requerir de un espacio para trabajar, con lo cual se deberá tener sumar el coste de alquiler y luz de un establecimiento, en este caso al tratarse del trabajo de una sola persona trabajando desde casa por lo que se asumirá el alquiler de esta como gasto, lo que significa 500 euros al mes más el gasto de luz de 70,95 euros cada dos meses, entre todo supone un gasto de unos 1600 euros al mes dedicados a cubrir los gastos del puesto.

Por último, los costes por ser autónomo en 2020 [16] suponen un 30.3% de la base de cotización, que se establece en 1.500 euros obtenemos una cuota de 454 euros mensuales en concepto de cuota de autónomo. Por lo tanto, el presupuesto total requerido para el desarrollo de este proyecto es de unos 8.300 euros.

5. Diseño

5.1 Tecnología utilizada

Una de las decisiones más importantes a la hora de desarrollar un proyecto de estas características, es la elección del motor gráfico, ya que determinará las funcionalidades que podremos implementar y la forma en que trabajaremos. Debido a que es un juego en 3D, las tres alternativas más populares son Unity 3D[10], Unreal Engine 4[11] y Godot[12] [14], en este apartado discutiremos las características de los tres, así como cual se ha decidido utilizar.

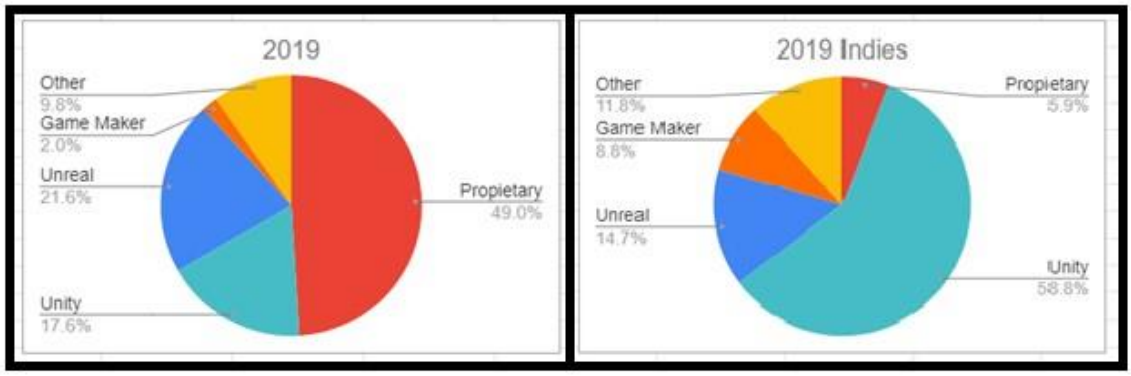


Figura 11: Tecnologías utilizadas en la industria

Unity es el motor más utilizado en el desarrollo de juegos independientes, con un 58,8% de los juegos siendo desarrollados con este motor, Unity cuenta con una enorme comunidad de desarrolladores y documentación disponible. Algunas de las características de este motor son su capacidad de desarrollar juegos tanto en 2D como en 3D, utiliza C# como lenguaje de programación. En cuanto al precio, el uso de esta herramienta es gratuito en su versión personal, pero si se desea usar en una empresa, los ingresos anuales de la misma determinarán cuanto hemos de pagar por la licencia.

Como se puede ver en las gráficas, Unreal Engine es el segundo motor más utilizado por los desarrolladores indies y el motor no propietario más utilizado en la industria. Está centrado en el desarrollo 3D, pero no solo de videojuegos, también está presente en otras industrias como la arquitectura, el automovilismo y el cine. A la hora de programar Unreal ofrece la opción de utilizar C++ o Blueprints, un lenguaje propio de programación visual. El coste de Unreal está basado en un sistema de royalties cuyo precio depende de los ingresos del juego.

Por último, Godot es un motor *open source* o de código abierto, y es este su aspecto más atractivo, puede ser modificado y compilado para ajustarse a las necesidades del proyecto. No dispone de una comunidad tan grande como la de los anteriores y el número de plataformas para las que puede exportar los juegos es limitada, pero

[10] Unity 3D: https://unity.com/es
[11] Unreal Engine 4: https://www.unrealengine.com/en-US/
[12] Godot Engine: https://godotengine.org/

características como el número de lenguajes de programación soportados (GDScript[13], Visual Scripting, C# y C++) y la propia licencia de código abierto, lo hacen una opción para tener en cuenta.

De entre estas opciones, se ha escogido trabajar con Unreal Engine 4 debido a su popularidad en la industria, el sistema de programación visual Blueprints que permitirá un prototipado rápido, ayudando a centrarnos en los problemas de más alto nivel, y la disposición de assets como Behaviour Trees ya integrados en la herramienta que facilitarán la tarea de programar la inteligencia artificial necesaria.

Otras herramientas utilizadas durante el desarrollo serán Gimp 2[14], un programa de edición de imágenes digitales similar a Photoshop[15], pero de uso libre, para crear los elementos visuales de la interfaz y Audacity[16], una aplicación de edición de audio, para adaptar el sonido y la música del juego. Este tipo de software no ha sido estudiado durante la carrera por lo que será necesario dedicar algún tiempo a aprender a utilizarlos correctamente.

No se ha utilizado ninguna herramienta de modelado/animación 3D ya que los assets utilizados han sido obtenidos a través del store propio del motor (Unreal Engine Marketplace).

5.2 Diagrama de flujo

A continuación, se muestra un diagrama representando la forma en la que se interactuará y navegará dentro de la aplicación.

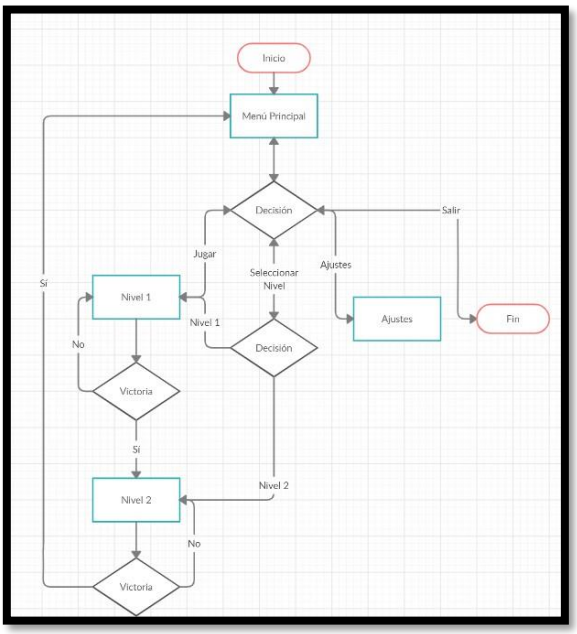


Diagrama 3: Diagrama de flujo

[13] GDScript: lenguaje propio de Godot - https://gdscript.com/
[14] Gimp 2: herramienta de edición y retoque de imágenes - https://gimp.es/
[15] Photoshop: editor de fotografías desarrollado por Adobe https://www.adobe.com/es/products/photoshop.html
[16] Audacity: editor de audio - https://www.audacityteam.org/

En el diseño de la navegación se ha seguido el modelo tradicional que emplean los videojuegos a la hora de definir la navegación de sus menús, lo que significa tomar el menú principal como eje central desde el que se tomarán las decisiones y al que se podrá regresar en cualquier momento.

5.3 Diseño del sistema

Este apartado está dedicado a especificar en qué partes se ha dividido del proyecto a la hora de ser desarrollado y la relación que tienen esas partes entre sí. Los apartados están acompañados con un diagrama de clases que no representa el sistema en su plenitud, sino que ilustran las relaciones entre los componentes.

Personaje principal

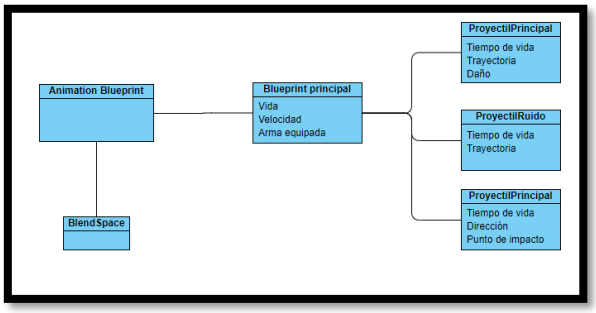


Diagrama 4: Diagrama clases personaje principal

La funcionalidad del personaje principal se implantará en el sistema con las siguientes componentes y las relaciones que se forman entre ellos:

- Animaciones
  Blueprint que controla las animaciones del personaje principal y las transiciones entre ellas dependiendo de variables como la velocidad del personaje o la dirección en la que está mirando. Estos cambios se realizan utilizando una máquina de estados.
    - BlendSpace: será un objeto adicional al blueprint de animaciones que se encarga de que las transiciones entre las distintas animaciones sean fluidas.
- Blueprint principal
  Integra el blueprint de animaciones anterior con el comportamiento del personaje que está definido aquí en su totalidad. Desde este objeto se controlan todas las mecánicas a las que puede acceder el personaje (que se pueden encontrar en el anexo de diseño del juego(página 59)).
  En este objeto también encontramos funciones que interactúan con otros objetos como los proyectiles, decidiendo la dirección en la que avanza.

Proyectiles

Contamos con cuatro clases de proyectiles que serán creados por sus respectivos dueños, que decidirán la dirección en la que avanzan. Por su parte, cada proyectil cuenta con un tiempo de vida distinto, con el objetivo de no saturar la escena, un daño que realizará al objeto con el que impacte y unos comportamientos que deberá mostrar en el impacto. Los cuatro tipos de proyectil son: “DisparoPrincipal”, “DisparoDeRuido”,

“DisparoDeTeletransporte” y “DisparoEnemigo”, y sus características están definidas en el anexo de diseño del juego (página 57).

Algunos de estos proyectiles interactuarán con otros elementos del sistema como el enemigo en caso de ruido o las plataformas de teletransporte en el caso del “DisparoDeTeletransporte”.

Enemigo

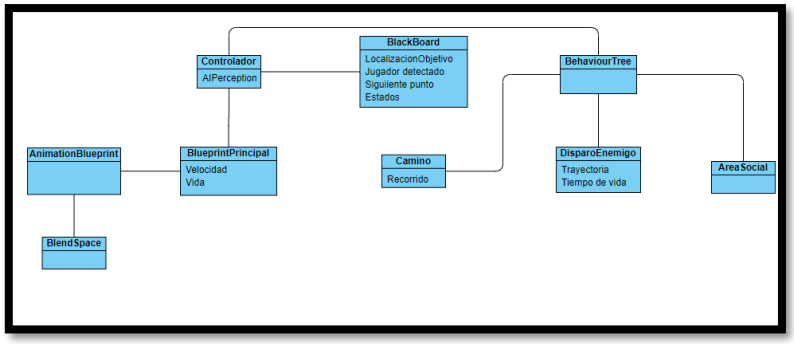


Diagrama 5: Diagrama de clases enemigo

- Animaciones
  De forma similar al encontrado en el personaje principal, existe un Blueprint que se encarga de gestionar las transiciones entre las animaciones del enemigo
    - BlendSpace: también cuenta con un objeto que regula fluidez de las transiciones del personaje.
- Blueprint principal: a diferencia del encontrado en el personaje principal, este blueprint no controla todo el comportamiento del enemigo, solo controla el nivel de vida del personaje y la integración de este con las animaciones definidas.
- Behaviour Tree: elemento que controla las decisiones que tomará la IA, incorporará el funcionamiento de una máquina de estados definida en el siguiente apartado de diseño de la IA. A su vez este elemento está compuesto por blueprints de tamaño reducido denominados tareas, que como conjunto se encargan de controlar las acciones del enemigo.
    - Blackboard: elemento en el que se almacenan las variables que se utilizarán tanto en el controlador como en el Behaviour Tree para entre otras cosas, gestionar el acceso a las ramas.
- Controlador de la IA: se integrará tanto con el Behaviour Tree como con el blueprint principal y controlará los aspectos relacionados con los estímulos que recibirá por parte de su componente *AIPerception* y cuyos detalles se discuten en el apartado de desarrollo. El motivo de separar el controlador del blueprint principal es el de lograr la mayor compatibilidad posible en caso de añadir nuevos enemigos en el futuro.
- Camino: componente diseñado específicamente para delimitar la ruta que seguirá el enemigo en su patrulla. Consiste en un array de vectores que se sitúan en el espacio 3D del juego y marcan los puntos que recorrerá el enemigo.
- Área social: elemento creado con la intención de limitar una zona invisible en la que los enemigos activan su comportamiento social e interactúan con otro

enemigo. El componente consiste en un área de colisión invisible que activa este comportamiento en los enemigos al entrar en ella.

Menús y HUD

El diseño de los menús se basa en el flujo de navegación de la aplicación definido en el apartado anterior y los elementos presentes en los mismos se encuentran en el anexo de diseño del juego página 65.

En cuanto al diseño más técnico de los componentes que formarán el menú y el HUD, el comportamiento estará definido en Widgets, una subclase de Blueprint que se sobrepone a la vista del juego.

Otros elementos

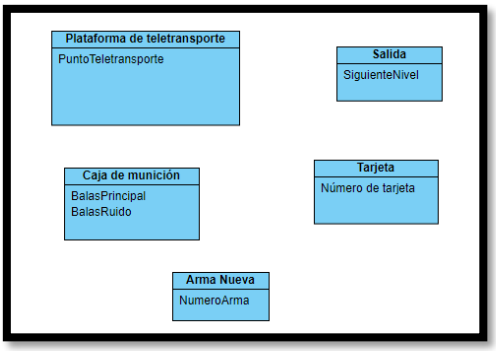


Diagrama 6: Diagrama de clases ítems adicionales

En este apartado se encuentran ítems independientes que no conforman ningún sistema pero que si interactúan con los otros. El comportamiento de estos está definido en Blueprints independientes y se encargan de tareas menores como la transición entre niveles o el cambio en variables de otras entidades como la munición del jugador. Para más información sobre el diseño de estos ítems consultar el anexo de diseño, página 62.

5.4 Diseño de la IA

En el diseño de los enemigos se ha decidido que la única forma en que se detectará al jugador será con el sentido de la vista, utilizando el resto de los sentidos como estímulos para guiar a la IA. Esta detección no será inmediata, sino que será necesario mantener contacto durante un periodo de tiempo determinado. Además, se indicará al jugador el estado de alerta en que está la IA mediante un medidor en el HUD del mismo.

Figura 12: Diseño HUD enemigo

El comportamiento del enemigo viene determinado por una máquina de estados que cuenta con siete estados, que se muestran en el siguiente diagrama y cuyas transiciones pueden ser tanto unidireccionales (→) como bidireccionales (--):

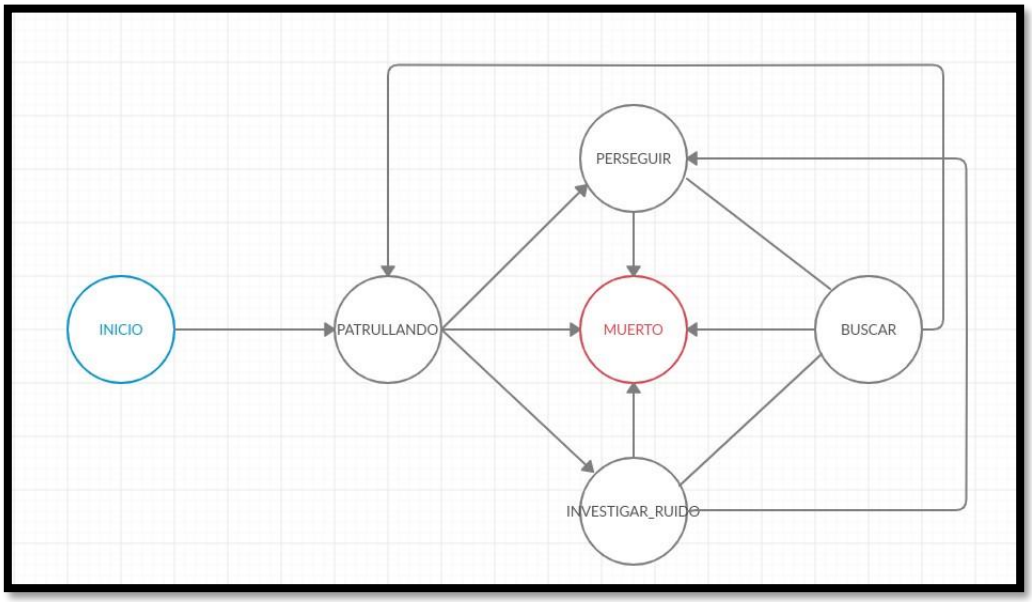


Diagrama 7: Diagrama IA Enemigo

1. Inicio: el estado inicial de la IA, en este estado se preparan todos los componentes necesarios para su funcionamiento, estableciendo la ruta de patrulla que seguirá e inicializando los elementos de interfaz necesarios. Desde este estado se avanza al estado “Patrullando”.
2. Patrullando: dependiendo de la ruta establecida y las opciones escogidas, este estado tiene diferentes comportamientos.
    a. Recorre un camino dado y al llegar al final se detiene.

   b. Recorre el camino en bucle, dirigiéndose al punto inicial al acabar el camino.
   c. Si no tiene camino asignado se quedará quieto vigilando una zona.

   En cualquier caso, al recibir los estímulos adecuados, desde este estado se puede hacer transiciones a “Perseguir”, “Investigar_Ruido” y “Muerto”.

3. Perseguir: cuando el enemigo tiene contacto visual con el personaje principal durante el tiempo suficiente entrará en este estado. En este estado tratará de perseguir al jugador hasta eliminarlo o perder contacto visual con el mismo, cuando lo segundo suceda entrará en el estado “Buscar”.
4. Investigar ruido: cuando el jugador genere un ruido, se entrará en este estado, en él la IA se dirigirá al lugar donde se ha producido el ruido y entrará en el estado “Buscar”.
5. Buscar: dependiendo del estado de donde se venga, el enemigo buscará al jugador por la última zona donde lo vio/escucho. Una vez finalizada la búsqueda, si no ha detectado al jugador, volverá al estado “Patrullando”.
6. Muerto: se puede llegar desde cualquier otro estado, cuando se pierda toda la vida. El enemigo caerá al suelo y no podrá acceder a ningún otro estado.

6. Desarrollo de la solución

En este apartado se describe el desarrollo de la solución propuesta partiendo de los diseños planteados en el apartado anterior, y con el objetivo de cumplir los requisitos establecidos durante el análisis. No se profundizará mucho en la implementación, sino que se centrará en discutir algunos de los problemas encontrados en el desarrollo de la solución propuesta y como se han resuelto.

El apartado se divide en bloques dedicados al personaje principal (Wraith), a la inteligencia artificial, a los menús de ajustes gráficos y al desarrollo de algunos de los ítems adicionales ya que son los pilares más importantes del desarrollo y contienen las soluciones más interesantes.

6.1 Personaje principal

Wraith es el personaje principal de Fuseworks y el que controlará el jugador, cuenta con una serie de mecánicas que a las que accede a través de sus herramientas/armas y cuyo comportamiento gira entorno a disparar distintos proyectiles.

Por lo tanto, en este apartado discutiremos la forma en que se calcula la trayectoria de estos proyectiles además de las diferentes técnicas que se han utilizado para crear los distintos proyectiles.

1. Determinar el objetivo del proyectil

Debido a que tres de los cinco movimientos principales del personaje dependen de que los proyectiles que se disparan sean precisos, es de vital importancia garantizar que siguen una trayectoria acorde a las intenciones del jugador.

Al ser un juego 3D en tercera persona, la cámara no está perfectamente alineada con la dirección en que apunta el personaje, por ello es necesario realizar una serie de procesos para determinar la dirección del proyectil.

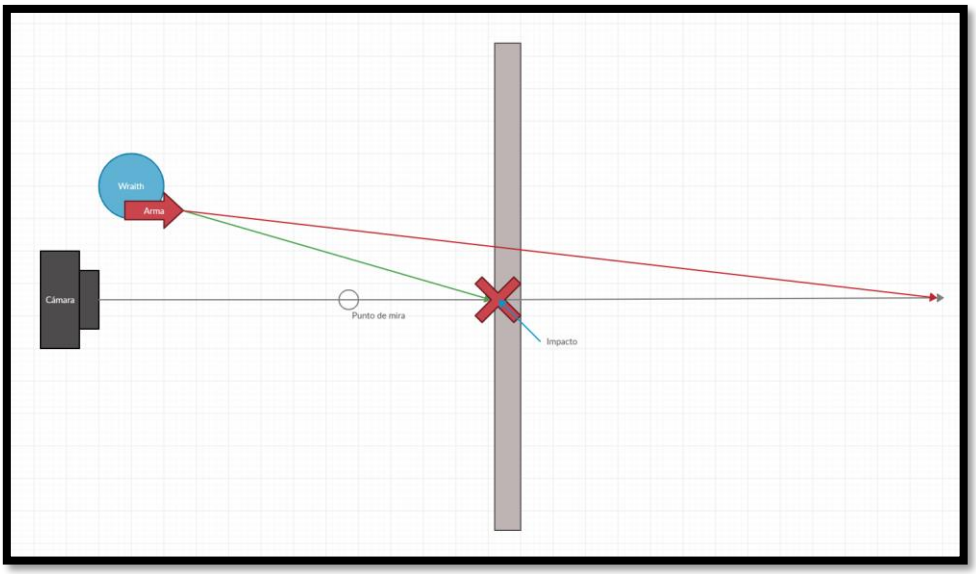


Figura 13: Cálculo trayectoria proyectil

Como puede verse en la imagen se castea una línea partiendo de la cámara desde la cual se recogerá el impacto con el primer objeto con el que colisione, entonces se combinará la posición del impacto y del arma para obtener la rotación necesaria para que el proyectil alcance su objetivo (línea verde). En caso de no impactar con nada, se utilizará la rotación de la cámara para trazar la trayectoria (línea roja).

Esta clase de implementación es posible gracias al método "*LineTraceByChannel*" que permite proyectar una traza de colisión desde la que se obtiene el primer objeto golpeado.

2. Proyectiles

En Unreal Engine 4 existen dos formas de implementar proyectiles y dependiendo del comportamiento que se le quiera dar al proyectil se utilizará una o la otra:

- "*ProjectileMovement*": es el método más sencillo a la hora de crear un proyectil y consiste en añadir el componente "*ProjectileMovement*" a una clase "Blueprint" que hereda de "Actor" y configurar este nuevo componente con las características de velocidad y comportamiento deseadas. El inconveniente de esta implementación es su incompatibilidad con el uso de físicas de modo que su uso está limitado a objetos como balas, que tienen trayectorias más sencillas que objetos arrojadizos o de menor velocidad cuyas físicas hay que simular más fielmente.
- Uso de físicas: la alternativa al componente anterior es utilizar el objeto tal y como se ha creado y añadirle un impulso o fuerza en la dirección en la que se decida. De esta forma se mantienen las físicas por defecto de los objetos "Actor" consiguiendo trayectorias más realistas.

Teniendo en cuenta las características de ambos métodos, se ha aplicado el componente "*ProjectileMovement*" a los proyectiles principales y de hacer ruido y el uso de físicas a el proyectil de teletransporte, que es un proyectil más pesado y necesita físicas para obtener el comportamiento esperado.

Figura 14: Proyectil

Tanto el proyectil de disparo principal como el de ruido, utilizan el componente "*ProjectileMovement*" para simular su comportamiento, además de esto en los tres tipos de proyectiles, se utiliza una esfera que contiene una capsula de colisión, que es la encargada de establecer la relación de este componente con los demás (en cuanto a físicas y colisiones se refiere) y un sistema de partículas para agregar los distintos efectos a los proyectiles.

6.2 Mecánica de teletransporte

La mecánica de teletransporte está ligada a tres elementos del sistema: el proyectil de teletransporte, el personaje principal y la plataforma de teletransporte. Como se ha implementado el diseño de esta mecánica se describe a continuación:

Figura 15: Código proyectil teletransporte

Al impactar el proyectil con algún objeto del mapa se comprueba si este es una plataforma de teletransporte, en caso de serlo se activará el evento de teletransporte definido en el Blueprint del personaje principal (indicándole el objetivo al que se transportará), se genera un efecto de impacto y se destruye el proyectil.

El evento de teletransporte activará una serie de animaciones y efectos que se ejecutaran consecutivamente hasta transportar al jugador al punto determinado por la plataforma. La forma en que se han tenido que situar los efectos ha tenido que ser estudiada ya que es necesario que sigan los movimientos del personaje, por ello se han tenido que añadir sockets (puntos de referencia) al esqueleto del modelo del personaje principal, lo que garantiza que el efecto se adhiera a los movimientos del personaje.

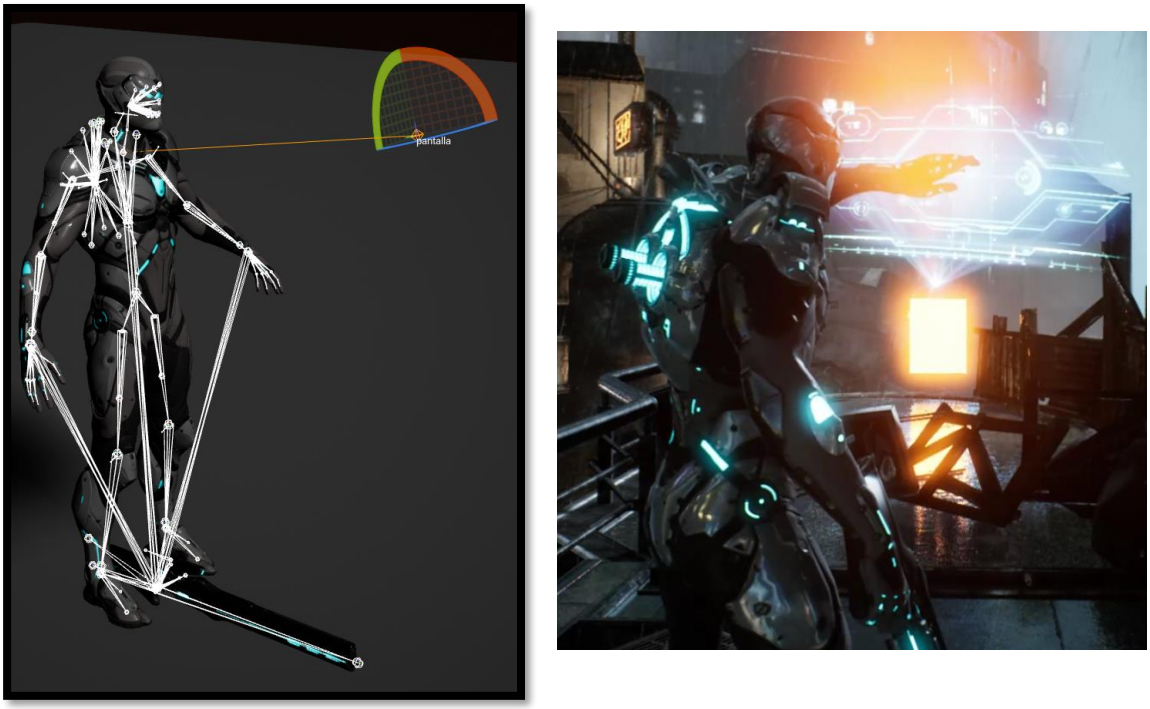


Figura 16: Socket de la pantalla

Como se ve en la imagen, el efecto de la pantalla se genera sobre el socket llamado "pantalla" de forma que siga al personaje tanto si rota sobre sí mismo como si se mueve.

6.3 Inteligencia Artificial

El obstáculo más importante que encontrará el jugador serán los enemigos que patrullan los niveles en su busca, los comportamientos de estos enemigos están controlados por su inteligencia artificial cuyos estados se han discutido en el apartado de diseño.

Para implementar la IA se ha hecho uso del asset "Behaviour Trees" en combinación con el "AI Perception System" y "Blackboard" disponibles en Unreal Engine y que ayudan a dotar a la inteligencia artificial la capacidad de toma de decisiones en base a lo que sucede en el nivel. A continuación, se describen las implementaciones realizadas sobre dichos componentes.

En primer lugar, un cambio que se ha realizado con respecto al desarrollo habitual en blueprints en este proyecto, es que el comportamiento descrito en los mismos se ha extraído a un componente "*controller*" en lugar de encontrarse directamente en el componente en el que se va a aplicar, de esta forma logramos que el comportamiento aquí descrito pueda usarse en enemigos o IAs que podrían implementarse en el futuro con mayor facilidad. Es en este "Controller" donde encontramos el componente "AIPerception" que se encarga de recoger los estímulos visuales y auditivos, que posteriormente se analizarán.

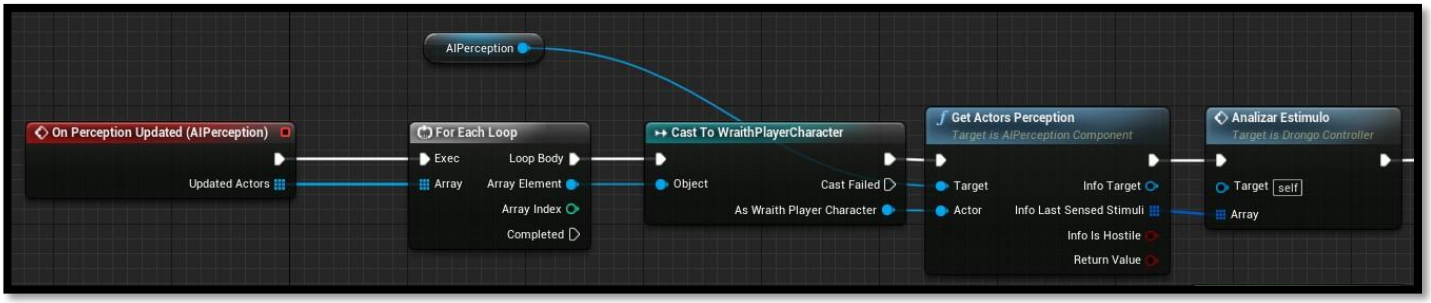


Figura 17: Análisis de estímulos

Como puede verse en la imagen, los estímulos recogidos por este componente se reciben en forma de un array que habrá que recorrer, recoger los que tienen que ver con el personaje principal ("*WraithPlayerCharacter*") y analizar con una función propia en la que se diferenciará entre estímulos auditivos y visuales y se responderá acorde.

En cualquier caso, la respuesta pasará por cambiar los valores de varias variables en la "Blackboard", un asset que ayuda a comunicar el controlador con el "Behaviour Tree" de forma eficiente. Entre las variables utilizadas cabe destacar "Status" que marca el estado en que se encuentra la IA y que se utilizará para marcar la rama a seguir en el "*BehaviourTree*" y "*TargetLocation*" que se utilizará para indicar la posición a la que acudirá el enemigo.

- En el caso de la escucha, se cambiará el valor de las variables “TargetLocation” y “Status” al lugar en el que se ha producido el ruido e “InvestigandoRuido” respectivamente.

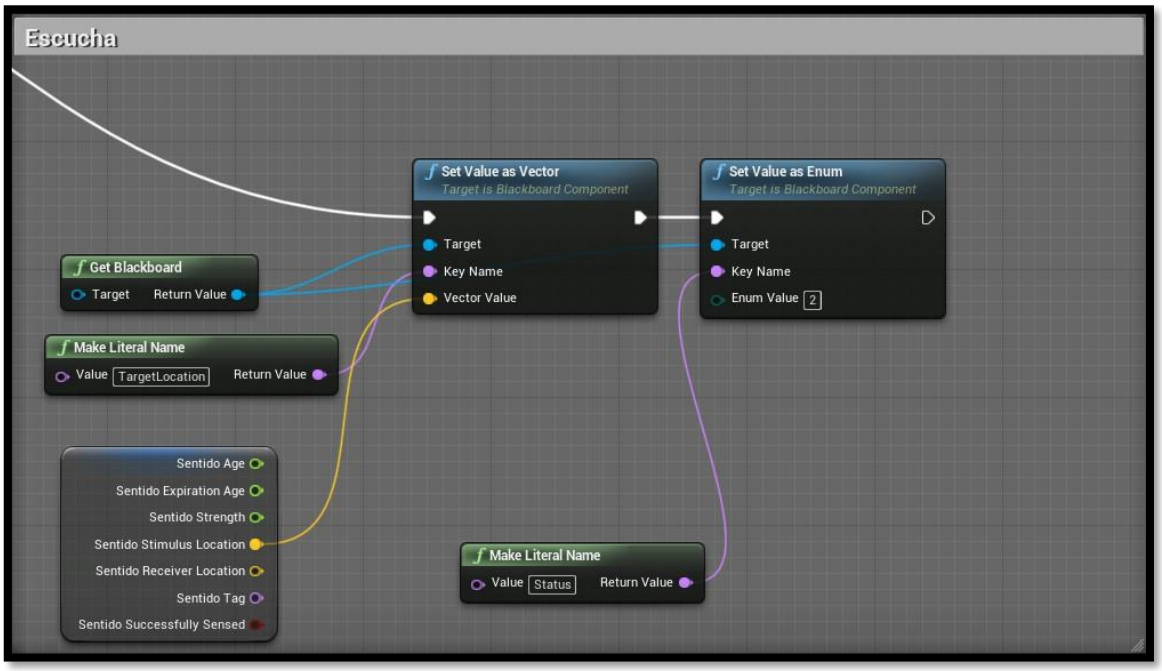


Figura 18: Respuesta escucha

- La visión tiene un comportamiento distinto y sus consecuencias no son inmediatas, sino que se rellena un contador en función a cuánto tiempo lleva viendo al jugador, y cuando este llega al máximo, se establece la variable “TargetLocation” como la localización actual del jugador y se cambia “Status” a “Persiguiendo”.

A continuación, se describe el siguiente componente que compone la inteligencia artificial y el principal responsable de que esta pueda tomar decisiones, “BehaviourTree”, pero antes de entrar en la implementación se explicará su funcionamiento de manera superficial de forma que pueda ser entendido, ya que su construcción es distinta a la de “Blueprints” y puede resultar confuso:

El árbol está formado por una raíz que almacena los datos del “Blackboard”, compuestos, tareas y ramas:

- Las tareas son “Blueprints” que describen el comportamiento o código a ejecutar. Estas tareas acaban con una función “*Finish Execute*” en la que se indica si la tarea ha finalizado con éxito.
- Los compuestos pueden ser selectores o secuencias y se encargan de ejecutar varias tareas. La diferencia entre ellos es que una secuencia ejecuta todas las tareas y un selector ejecutará una y solo si falla ejecutará la siguiente. Estos compuestos pueden agruparse unos dentro de otros.
- Las ramas unen todos los demás componentes y están ordenadas por prioridad de forma que las que se encuentran más a la izquierda se ejecutarán primero.

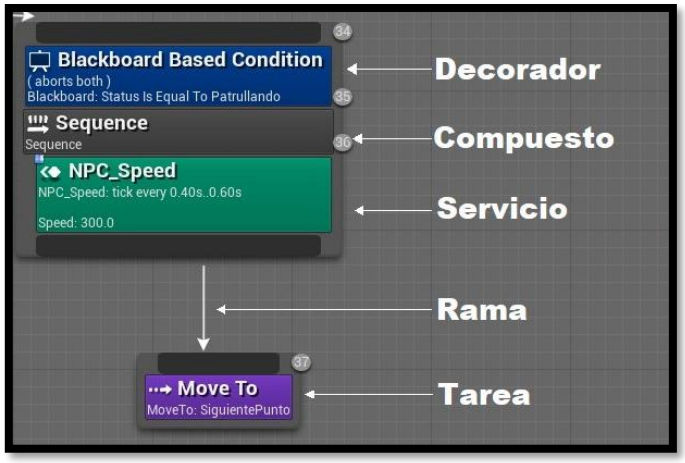


Figura 19: Componentes BT

Además, tanto los compuestos como las tareas pueden usar decoradores que comprueban ciertas pautas para dar acceso al nodo y servicios que son tareas sencillas que se ejecutarán con el nodo.

Una vez entendidos los conceptos básicos de los "Behaviour Trees" podemos pasar a discutir su implementación en este proyecto. Para no extendernos demasiado en este apartado, se describirá el funcionamiento de una de las ramas, la rama del estado "Persiguiendo".

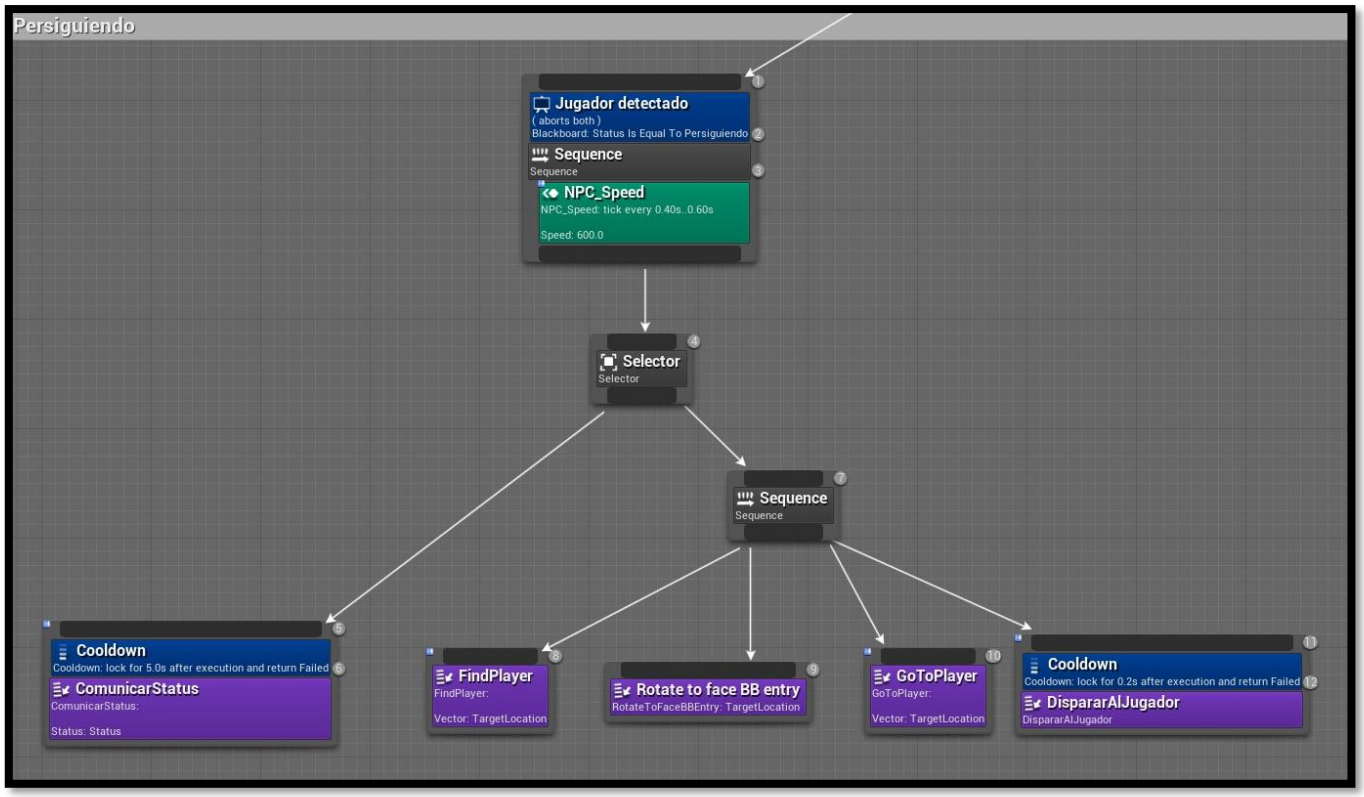


Figura 20: Rama "Persiguiendo" BT

Se entra en esta rama cuando el "Status" es "Persiguiendo", se aumenta la velocidad del enemigo con el servicio "NPC_Speed", el selector lleva a ejecutar la tarea "ComunicarStatus". En esta tarea, se reproduce un audio para que el jugador sepa que

ha sido detectado y el tener un decorador "*Cooldown*" evita que se reproduzca más de una vez seguida, haciendo que el selector use la siguiente rama. La segunda rama del selector es una secuencia:

1. "FindPlayer", actualiza el valor de "TargetLocation" a la posición actual del jugador.
2. "Rotate to fase BB entry", reorienta al enemigo a la posición "TargetLocation".
3. "GoToPlayer", se dirige a la posición "TargetLocation".
4. "DispararAlJugador, dispara proyectiles al jugador y cuenta con un decorador "Cooldown" que regula la ratio de disparo".

Si en algún momento el valor de "Status" cambia, se abortarán todas las tareas que se estaban realizando en esta rama pasando a la rama correspondiente.

6.3.1 Social

Otra de las características de la IA propuestas en el análisis del problema, es la implementación de modelos de percepción que permitan a la inteligencia artificial recoger datos de su entorno, dos de estos eran el modelo visual y el auditivo y para resolverlos se ha utilizado el "AI Perception System" como se ha discutido anteriormente.

Como tercer modelo se proponía un sistema social que permitiese que dos IAs interactuasen entre sí, para implementarlo se ha modificado una rama del "Behaviour Tree" además de creado un objeto que ayudará a establecer la conexión entre dos IAs.

Como primer paso se ha añadido dos variables, una al controlador haciendo referencia a la otra IA y otra en la "Blackboard" que no es más que un booleano que indica si se encuentra en un estado social. La primera variable ha de ser instanciada antes de comenzar la ejecución seleccionando la entidad con la que se quiere conectar en menú de detalles del editor, mientras que el booleano social cambiará según entre en contacto con el siguiente objeto.

El objeto creado para activar el comportamiento social de la IA es "AreaSocial" un objeto propio que forma un área invisible, cuando alguna de las IAs entra en esta área invisible queda a la espera de la otra IA, y si no llega entra en un estado de sospecha en el que detectará al jugador el doble de rápido.

6.4 Creación niveles

Nivel 1

Debido a la falta de experiencia del desarrollador en la creación de entornos 3D, el nivel se ha creado partiendo del mapa importado (*Soul City*) y se ha adaptado a las necesidades del proyecto creando nuevas calles y zonas por las que circular.

También se ha realizado un proceso de limpieza de los obstáculos que impedían el funcionamiento de las mecánicas del juego y se ha poblado el nivel con enemigos e ítems como requería el diseño del nivel.

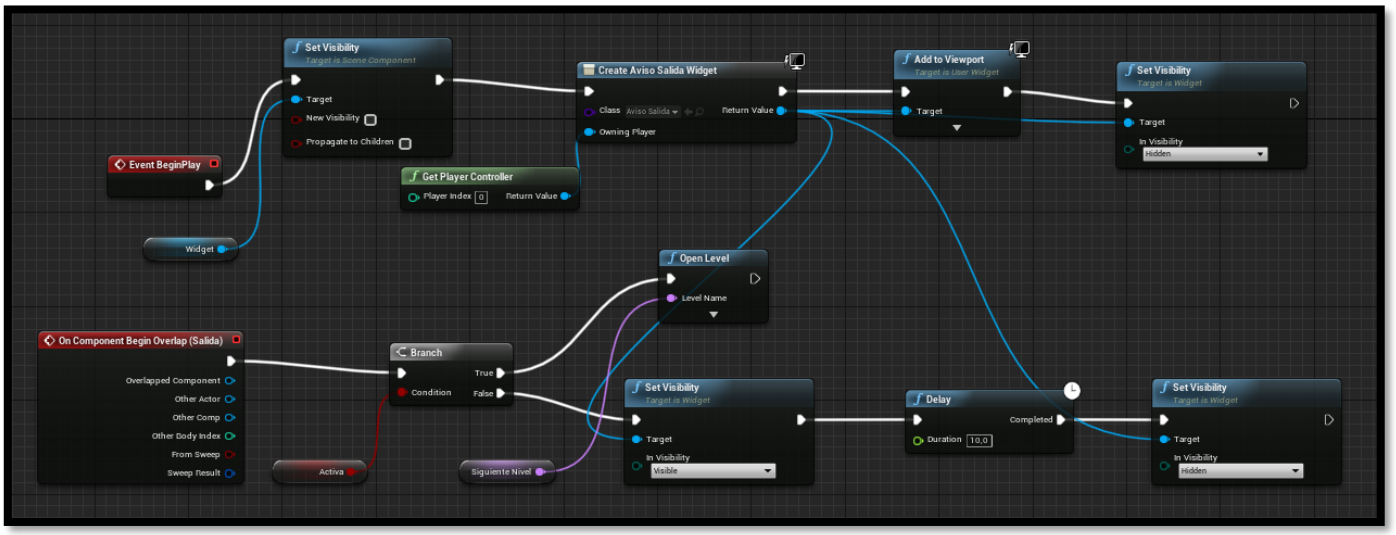
Nivel 2

En el segundo nivel se han usado assets de aspecto más industrial (*Modular Scifi Season 2 Starter Bundle*) lo que ha permitido que sea el desarrollador quien construya este nivel. Para ello, además del posicionamiento de los assets en sí se han tenido que aplicar dos soluciones para que las mecánicas se integrasen correctamente en este nivel.

La primera de ellas ha sido el uso de "Volúmenes Bloqueantes", un componente que se encuentra disponible por defecto en Unreal Engine y que ayuda a que los proyectiles generen los impactos deseados para su funcionamiento y bloquea el acceso del jugador a determinadas zonas.

Y la segunda ha sido la creación de un "Disparador" que no es más que una zona invisible que activa a los enemigos, cuando el jugador entra en ella de manera que no se depende del tiempo que tarde el jugador en llegar a un lugar para ver el comportamiento de los enemigos que se ha diseñado para este nivel.

Por último, un elemento que tienen en común ambos niveles es la disposición de una salida, que comprobará si el jugador tiene permiso o no para acceder al siguiente nivel.



Al comenzar la partida se crea el widget de interfaz necesario para mostrar un mensaje al jugador en caso de no tener acceso, y las comprobaciones sobre si la salida se encuentra activa se realizan al entrar en la zona delimitada por la misma, en caso de tener acceso, se carga el nivel indicado por la variable segundo nivel y en caso negativo, se muestra el mensaje pertinente.

6.5 Desarrollo de los menús

Los menús se han creado de forma similar al resto de los elementos de la interfaz del juego, es decir, mediante widgets (subclase de Blueprints) que se añaden a la vista del juego. El menú principal se crea al iniciar el juego y este es el que crea el resto de los menús, al pulsar los botones que acceden a ellos.

A continuación, se describe el desarrollo de dos de los menús con los que cuenta la aplicación:

Desarrollo de los ajustes de controles

Como parte del menú de ajustes de la aplicación, se requería el desarrollo de un menú en el que se pudiese alterar el mapeo de los controles por defecto del juego. Para ello se ha empleado el componente "Input Key Selector" disponible en los widgets de Unreal, que, tras pulsarlo, espera a recibir el siguiente input y lo registra.

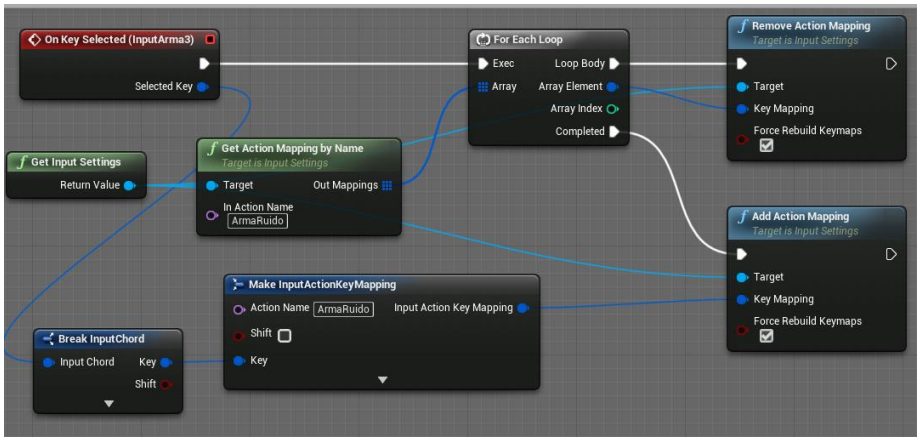


Figura 21: Registro de inputs

Tras recibir el input, se eliminan los posibles mapeos anteriores y se crean los nuevos utilizando los "*Action Mapping*" de Unreal, que permite separar el comportamiento de los inputs de las acciones que realizan, permitiendo este tipo de cambios en el control sin afectar al código del juego.

Desarrollo de las opciones gráficas

En el desarrollo de un videojuego, hay que tener en cuenta que no todos los usuarios tendrán una máquina la suficientemente potente para ejecutar el juego de forma fluida, por este motivo, será necesario ofrecer diversas opciones gráficas que se adapten a la potencia de cada dispositivo y garanticen al jugador una experiencia satisfactoria.

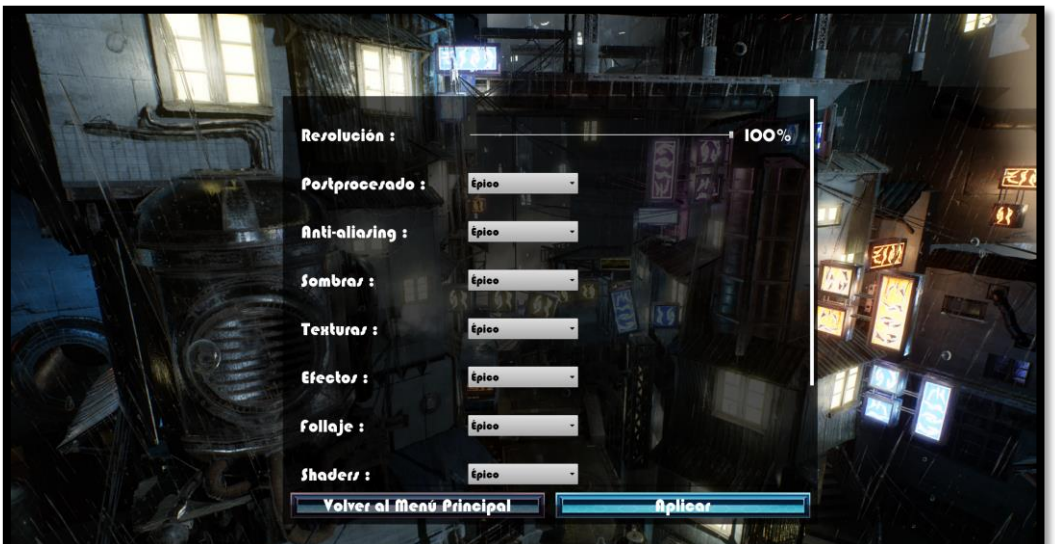


Figura 22: Menú de opciones gráficas

El acceso a estas opciones gráficas se realiza a través del menú de ajustes gráficos que encontramos en el menú principal.

En este menú encontramos tres tipos generales de opciones:

- *Slider*: se utiliza un componte de tipo slider para controlar la escala de resolución del juego, esta escala controla el porcentaje de resolución en que se renderiza el juego, pero mantiene la resolución de la pantalla manteniendo los elementos de la interfaz a la resolución nativa del monitor.
- Selectores: se han implementado muchas opciones gráficas por medio de selectores que permiten escoger una calidad entre Baja y Cinemática y que ajustan en gran medida la forma en que el juego se ve gráficamente y el rendimiento que tiene.
- *Checks*: por último, encontramos dos *checks* booleanos que deciden si aplicar o no ciertas tecnologías como la niebla volumétrica o la sincronización vertical.

Todas estas opciones, a excepción de la niebla volumétrica hacen uso del objeto "*Game User Settings*" que automatiza la aplicación y persistencia de los diversos perfiles gráficos de los *assets* del juego.

Rendimiento

Para concluir el apartado de las opciones gráficas, vamos a analizar las consecuencias que ha tenido la implementación de estas opciones en el rendimiento del juego para comprobar que el tiempo empleado en ellas está justificado y permitirá que el juego se adaptar a dispositivos con distintas potencias.

Figura 23: Rendimiento

En la imagen mostramos el rendimiento del juego utilizando distintas opciones gráficas (en orden, épico, medio y bajo), utilizando el software *MSI Afterburner* y *RiviaTunnerStatistics*[17] para medirlo. El sistema utilizado cuenta con un procesador i5-7500 y una tarjeta gráfica gtx 1060.

En la primera imagen vemos un rendimiento de 44 imágenes por segundo, y podemos observar que la utilización de la gráfica es del 98% lo que significa que este es el factor que limita nuestra fluidez. Gracias a que las opciones gráficas afectan principalmente al rendimiento de la tarjeta, podemos ajustar la calidad para reducir la carga. En el caso de la segunda imagen observamos que ha aumentado la fluidez al elegir las opciones de calidad medias llegando a las 60 imágenes por segundo, lo que supone una mejora del 36%. Para terminar las mediciones de rendimiento se han elegido todas las opciones en bajo y se han obtenido 129 imágenes por segundo, lo que significa aumento del 193% en el rendimiento.

[17] Softwares de información sobre el estado del hardware:
MSI Afterburner y RiviaTunnerStatistics - https://es.msi.com/page/afterburner

7. Conclusiones

Llegamos al final del documento y es hora de reflexionar sobre el trabajo realizado y analizar el cumplimiento de los objetivos establecidos al inicio del proyecto.

Los objetivos que se plantearon al inicio del proyecto han sido cumplidos, el estudio e implementación de las mecánicas comunes del género ha llevado a la creación de un prototipo en el que la experiencia del usuario tiene la prioridad y se anima al jugador a experimentar con las mecánicas de juego. Adicionalmente, se ha asegurado el correcto funcionamiento de todos los elementos del juego, desde los menús a las IAs, garantizando que muestran el comportamiento esperado en todas las circunstancias siguiendo las intenciones establecidas en su diseño.

Relación con las asignaturas cursadas

El proyecto ha permitido aplicar los conocimientos adquiridos durante el grado a un trabajo real y de mayor escala que los desarrollados hasta el momento. De entre los conocimientos que han ayudado en el desarrollo cabe destacar aquellos adquiridos durante los dos últimos años en los que he cursado la rama de ingeniería del software que con asignaturas como “Análisis y Especificación de Requisitos” (AER) y “Proyecto de Ingeniería de Software” (PIN) que me han ayudado a definir la forma que tomaría el proyecto en cuanto a requisitos y objetivos, además de la metodología y la manera en que ha evolucionado el trabajo durante el tiempo. Además, asignaturas como diseño de software (DDS) y mantenimiento y evolución del software (MES) han influenciado la toma de decisiones en la implementación de las distintas características del juego de modo que fuesen desarrolladas con la intención de ser fácilmente mejoradas y mantenidas en el futuro.

Conclusión personal

Pasando a temas más ajenos al juego en sí, la realización de ese trabajo me ha ayudado a adquirir una mayor experiencia en el desarrollo de videojuegos aumentando mi comprensión de los procesos tienen lugar durante su creación. Asimismo, el proyecto ha servido para ampliar mis conocimientos en Unreal Engine 4 permitiéndome aprender a utilizar algunos de sus assets más sustanciales como son “*Behaviour Trees*” o el “*AI Perception System*”. En una nota más personal, realizar un proyecto de esta magnitud me ha permitido apreciar las ventajas que un sistema de programación visual como “Blueprints” puede aportar a la hora de mejorar la legibilidad y mantenibilidad del código.

También he encontrado dificultades en el transcurso del proyecto, sobre todo en temas más lejanos del desarrollo del software como son la música, los sonidos y los modelos y animaciones del juego que, aun siendo imprescindibles para el juego, escapan a los conocimientos adquiridos durante la carrera y han tenido que ser adquiridos de terceros, pese a ello, han sido difíciles de seleccionar e integrar debido a las restricciones que marca el motor gráfico y por supuesto la temática sci-fi del juego. Por ello, ha habido que buscar recursos específicos tanto en el store propio del motor como en webs de terceros, y en los casos en los que ha resultado imposible encontrarlos en el formato

correcto, se han utilizado herramientas como *Audacity* y *GIMP* para adecuarlos a las necesidades del proyecto.

En lo relativo a la redacción de la memoria, al igual que el proyecto en sí, es la primera vez que he afrontado un trabajo de esta longitud y complejidad, además debido a la situación actual la relación con el tutor no ha podido ser la adecuada. Tan solo he podido hablar una vez con él, al final del proyecto, por lo que no ha podido responder mis dudas y algunos de los fallos que destacó no han podido ser corregidos en el tiempo que quedaba hasta la entrega del trabajo.

8. Trabajos Futuros

Al haberse diseñado como un prototipo, este proyecto ha sido pensado para ser ampliable en el futuro. Estas ampliaciones o mejoras dependerán de la dirección en la que se quiera llevar el juego, podrían ser tanto nuevo contenido o nuevas mecánicas y sistemas, a continuación, se enumeran distintas mejoras que se podrían aplicar:

- Añadir niveles: una de las posibles mejoras podría ser la creación de nuevos niveles que utilicen las mecánicas ya establecidas y aumenten la complejidad de los desafíos que se presentan al jugador. Es una mejora necesaria ya que el contenido, en cuanto a niveles, del prototipo no presenta la duración que se esperaría de un producto comercial de este tipo.
- Narrativa: una historia aportaría al juego la narrativa necesaria para mantener al jugador comprometido con los objetivos del juego y deseoso de avanzar y completar los niveles que se le presentan. Es por esto por lo que los juegos de este género suelen presentar narrativas que mejoran la inmersión del jugador en su mundo y por lo que aportaría grandes beneficios al proyecto si se desease ampliar.
- Mecánicas: al mismo tiempo que se amplían los niveles podría ser interesante añadir nuevas mecánicas que aumenten las posibilidades para diseñar niveles y desafíos. Gracias a la forma en que se ha diseñado e implementado el juego, la tarea de crear nuevas mecánicas será relativa mente sencilla, pudiendo diseñarlas en forma de armas que pueda utilizar el personaje principal.
- Enemigos: al añadir más contenido al juego en forma de niveles, es posible que el jugador pueda llegar a aburrirse de los enemigos que los habitan, por eso será necesario crear nuevos enemigos con distintas apariencias y comportamientos. La creación de estos enemigos también podrá ser cumplida con relativa facilidad gracias a que los componentes de los enemigos actuales pueden ser reutilizados por las nuevas entidades.
- Inteligencia artificial: además de nuevos enemigos la inteligencia artificial de los actuales podría ser mejorada añadiendo nuevos comportamientos y puliendo los actuales para que se adapten mejor al diseño de los niveles. Asimismo, uno de los modelos de percepción presentados (el modelo ambiental) no ha sido incluido en el desarrollo del juego y sería interesante añadirlo a la inteligencia artificial actual para terminar de completarla.

9. Referencias

9.1 Bibliografía

[1] Scum.org. *WHAT IS SCRUM?*
Consultado en: https://www.scrum.org/resources/what-is-scrum

[2] Newzoo. *Newzoo Global Games Market Report 2019 | Light Version*. 2019
Consultado en: https://newzoo.com/insights/trend-reports/newzoo-global-games-market-report-2019-light-version/

[3] GameAnalytics. *Mobile Gaming Industry Analysis for 2018*
Consultado en: https://gameanalytics.com/resources/item/mobile-gaming-industry-analysis-2018

[4] Gamasutra.com. *The History and Meaning Behind the 'Stealth Genre'*. 06/10/2011
Consultado en: https://www.gamasutra.com/blogs/MuhammadAlkaisy/20110610/7764/The_history_and_meaning_behind_the_Stealth_genre.php

[5] Venturebeat.com. *A Brief History of Stealth Games*. 10-10-2010.
Consultado en: https://venturebeat.com/2010/10/10/a-brief-history-of-stealth-games/

[6] Gamerant.com. *The 10 Best Stealth Games Ever Made (According to Metacritic)*. 24-09-2019
Consultado en: https://gamerant.com/best-stealth-games-metacritic/

[7] Pluralsight.com. *How Animation for Games is Different from Animation for Movies*. 14/04/2014.
Consultado en: https://www.pluralsight.com/blog/film-games/how-animation-for-games-is-different-from-animation-for-movies

[8] Hackaboss.com. *Salario de programador en España 2018/2019*
Consultado en: https://hackaboss.com/blog/salario-programador-espana-2018-2019/

[9] Womeningameses.com. *¿Qué es la interfaz de usuario para videojuegos?* 13-05-2018.
Consultado en: https://womeningameses.com/2018/05/13/que-es-la-interfaz-de-usuario-para-videojuegos/

[10] DigitalSte. *Video Game Interface Types*. 3-10-2019
Consultado en: https://www.digitalste.com/post/video-game-interface-types

[11] GDC. *Modeling AI Perception and Awareness in Splinter Cell: Blacklist*. 8-02-2018.
Consultado en: https://www.gdcvault.com/play/1020436/Modeling-AI-Perception-and-Awareness

[12] Packtpub.com. *AI for Unity game developers: How to emulate real-world senses in your NPC agent behavior*. 6-06-2018.

Consultado en: https://hub.packtpub.com/ai-unity-game-developers-emulate-real-world-senses/

[13] Gamestudies.com. *Defining Game Mechanics*.
Consultado en: http://gamestudies.org/0802/articles/sicart

[14] OpenWebinars.net. *Ventajas y diferencias entre Unity, Unreal Engine y Godot*. 12-06-2019.
Consultado en: https://openwebinars.net/blog/ventajas-diferencias-unity-unreal-engine-godot/

[15] Xataca.com. *Los indies tenían razón: Unity y los motores de terceros le han ganado la partida a los motores propios a la hora de crear juegos*. 21-02-2020.
Consultado en: https://www.xataka.com/videojuegos/indies-tenian-razon-unity-motores-terceros-le-han-ganado-partida-a-motores-propios-a-hora-crear-juegos

[16] Infoautonomos.com. *Seguridad social de los autónomos*. 30/04/2020
Consultado en: https://www.infoautonomos.com/seguridad-social/cuota-de-autonomos-cuanto-se-paga/

9.2 Glosario

- **Asset:** representación de un objeto que puede ser utilizado en un juego o proyecto. Puede tratarse de modelos 3D, archivos de audio, imágenes o código. (Documentación)
- **Behaviour Tree:** los "arboles de comportamiento" son modelos matemáticos que describen comportamiento de un plan de ejecución, describiendo cambios de ejecución en un conjunto de tareas, formando así tareas complejas a partir de tareas sencillas. En el contexto de este proyecto, "Behaviour Tree" se refiere a la implementación de estos conceptos que tiene Unreal Engine 4.
(Wikipedia) (Documentación)
- **Blueprints:** lenguaje de programación visual utilizado en Unreal Engine 4, se trata de un lenguaje orientado a objetos (OO) que define las clases u objetos que utilizará el motor, es por ello por lo que los objetos en el entorno de desarrollo UE4 también pueden ser llamados "Blueprint".
(Documentación)
- **Feedback:** forma en que el sistema devuelve información al usuario como consecuencia de una acción que este ha realizado, en el contexto de los videojuegos, el sistema informará al jugador de los cambios que suceden en el mismo.
- **Free-to-play:** tipo de videojuego que permite a los jugadores el acceso a su contenido forma gratuita.
- **Gadget:** dispositivo con un propósito y función específica, generalmente de tamaño reducido y con tecnología novedosa.
(Wikipedia)
- **HUD (Head-Up Display):** información que se muestra en todo momento en pantalla durante la partida.

- **Juego sistémico:** juego donde los elementos y sistemas que lo forman pueden interactuar entre sí.
  (Moneda Roja)
- **Mecánica:** cualquier acción realizada por el jugador que afecta al estado del juego. Las mecánicas de juego describen lo que el jugador puede hacer, como lo hace y las reglas que gobiernan esas acciones.
  (Tokioschool)
- **Plataformas:** género de videojuegos caracterizados por tener que atravesar una serie de obstáculos generalmente mediante saltos para completarlo. (Wikipedia)
- **RPG (Role-Playing Game):** juego de rol en el que uno o más jugadores desempeñan un determinado papel. En el género de videojuegos, los RPGs suelen tener elementos comunes como árboles de habilidades y toma de decisiones, permitiendo al jugador definir su personaje y afectar a la historia del juego.
  (Wikipedia)
- **Shooter:** los juegos de disparos engloban un amplio número de subgéneros, que tienen la característica de controlar un personaje con la capacidad de disparar proyectiles para acabar con sus enemigos.
  (Wikipedia)
- **Store:** tienda virtual, en el contexto de este trabajo se refiere al *UE4 Marketplace* como lugar en que se han adquirido los distintos assets que componen el proyecto.
  (UE4 Marketplace)
- **Survival Horror:** horror de supervivencia, es un género de videojuegos inspirado en la ficción de terror que se centra en la supervivencia del personaje.
  (Wikipedia)

Anexo: Diseño del juego

A continuación, se muestra el GDD, el documento de diseño del juego a partir del cual se ha ido desarrollando el proyecto, en él se muestra el comportamiento de los diferentes elementos del juego, así como preguntas generales sobre el mismo y otros aspectos de interés a la hora de desarrollar un videojuego.

1 Preguntas frecuentes

¿Sobre qué trata el juego?

Consiste en un juego de sigilo en tercera persona para un solo jugador en el que el objetivo es infiltrarse a través de los niveles, evitando ser detectado por los enemigos mientras se completan los diferentes objetivos necesarios para finalizar el nivel.

¿Cuál es la temática?

El juego tiene una temática sci-fi, tiene lugar en la ciudad de Avico, una ciudad futura en la que el grupo llamado Kao-Fouque planea un ataque con androides para hacerse con el control del mundo.

¿Cuál es el papel del jugador?

El jugador tomará el control de Wraith, un agente especial al que le han encargado infiltrarse en dicha ciudad y evitar que el grupo Kao-Fouque lance el ataque.

¿Qué novedades aporta al género?

El juego aportará novedades al género a partir de sus mecánicas, que permitirán al jugador resolver sus encuentros con enemigos y puzles de diversas formas aportando variedad al juego más allá del acabar con los enemigos de manera directa.

2 Aspectos técnicos

Motor gráfico

Se utilizará el motor gráfico Unreal Engine en su versión 4, ya que es el entorno ideal para desarrollar un juego de estas características. Este motor gráfico cuenta una serie de assets como Behaviour Trees que ayudarán a gestionar la inteligencia artificial, además cuenta con un lenguaje de programación visual propio Blueprints que ayudará en la tarea de implementar las distintas mecánicas del juego.

Inteligencia Artificial

En Fuseworks, la inteligencia artificial será la encargada de enfrentarse al jugador, convirtiéndose así en el principal desafío que encontrará en los distintos niveles. Por ello, esta IA se diseñará con la intención de interaccionar con el jugador de forma satisfactoria de manera que este no sienta que la IA no es inteligente ni por el contrario que hace trampa.

Niveles

Fuseworks es un juego de temática sci-fi, ambientado en la ciudad de Avico, así pues, los niveles de este representarán esta ambientación teniendo lugar en las calles de la ciudad y en complejos de temática sci-fi de forma que el jugador se sienta parte de este mundo.

En cuanto al diseño de estos, se tendrá como objetivo representar una dificultad en aumento, a medida que el jugador se sienta más cómodo con las mecánicas y los retos presentados se irán introduciendo nuevas mecánicas y desafíos para mantener al jugador interesado.

3 El mundo de Fuseworks

El juego está ambientado en un futuro lejano en el que la humanidad ha desarrollado su tecnología hasta el punto de que la robótica y la nano tecnología se encuentran a la orden del día. En este mundo, ante el auge del uso de tecnologías militares por grupos criminales se ha formado la Agencia, con el objetivo de detenerlos.

Wraith es un agente de la agencia y el protagonista de la historia del juego, que tiene lugar en Avico una ciudad plagada de criminales del grupo Kao-Fouque, uno de los más peligrosos.

4 Diseño de personajes

Diseño personaje principal: Wraith

El personaje principal (Wraith) será el protagonista del juego y a través de él, el jugador interactuará con el mundo del videojuego. En el diseño de Wraith se han tenido en cuenta las principales mecánicas que se suelen tener los personajes jugables del género del sigilo además de haberse añadido otras para diferenciarlo y aportar variedad al juego.

Figura 24: Personaje principal

1. Movimiento

El movimiento se ha diseñado con la intención de proveer al jugador de la libertad necesaria para atravesar el nivel de manera satisfactoria, esto incluye la capacidad de moverse en cualquier dirección, además de saltar para sortear algunos obstáculos.

También se ha otorgado la capacidad de encoger su tamaño lo que permite ocultarse de la línea de visión de los enemigos y atravesar huecos y conductos para acceder a algunas partes del nivel, sin embargo, el encogerse reducirá la velocidad del personaje además de impedir que pueda apuntar con el arma.

2. Herramientas/Armas

El jugador dispone de tres tipos de armas o herramientas que se irán desbloqueando durante el transcurso del juego. Estas armas le ayudarán a completar los niveles, resolver puzles o acabar con el enemigo.

- Arma principal

Consiste en un arma que dispara proyectiles, su munición será limitada y se utilizará para eliminar a los enemigos, necesitando dos disparos para ello. Además de la limitada munición, cada disparo hará ruido lo que podría atraer la atención del resto de enemigos, con esto se invita al jugador a intentar sortear a los enemigos de otras formas. La cadencia de disparo de estos proyectiles es baja de manera que se fomenta el ser preciso y conservar la munición.

- Arma de ruido

Esta arma consiste en proyectiles similares a los del arma anterior, pero no serán letales y el ruido lo generan en el lugar de impacto, permitiendo al jugador atraer a los enemigos a un lugar determinado. El arma de ruido será útil para sortear a los enemigos sin ser visto y resolver puzles.

- Arma de teletransporte

Por último, el arma de teletransporte dispara proyectiles más lentos y pesados que los anteriores, la función de esta arma es transportar al jugador a un punto del mapa determinado por la plataforma a la que golpea dicho proyectil. Resultará útil para resolver puzles y acceder a partes del mapa inaccesibles de otro modo.

La mecánica del teletransporte conlleva las siguientes acciones:

1. El proyectil impacta con una plataforma de teletransporte
2. El personaje comienza una animación que comunicará que está a punto de teletransportarse
3. El personaje realiza un gesto final a la vez que se producen unos efectos audiovisuales y desaparece
4. El personaje aparece en el punto de destino

3. Otros

Además de las armas anteriormente mencionadas, el personaje principal cuenta con dos movimientos adicionales para aumentar las posibles soluciones a los problemas que encuentre, estos son:

- Golpear

Una alternativa a usar el arma principal, si el jugador golpea a un enemigo que todavía no lo ha localizado, lo eliminará al instante. Este golpe tiene un rango mucho menor a el disparo principal, pero no gasta munición ni hace ningún ruido.

- Silbar

El personaje principal podrá atraer a los enemigos a un punto determinado en el mapa silbando, este movimiento no gasta munición como el disparo de ruido, pero será más fácil que el enemigo localice al jugador debido a que acudirá a la posición en la que se ha producido el silbido.

Diseño personaje enemigo: Drongo

Enemigo del juego, estará controlado por una inteligencia artificial y patrullará los niveles dificultando la progresión del jugador.

Figura 25: Enemigo

Salud: de 0 a 10, serán necesarios dos disparos del protagonista para acabar con él

Comportamiento: el comportamiento base de este enemigo será el de patrullar los niveles en busca del jugador, una vez lo ha localizado, pasará a un estado de persecución en el que disparará al jugador tratando de acabar con él. A esto se le añaden comportamientos de investigación de ruido y de búsqueda, el primero consiste en investigar la procedencia del ruido acudiendo al lugar donde se ha producido, y el segundo es un estado en el que se entra al finalizar la persecución o la investigación y que consiste en investigar la zona donde se ha producido el estímulo, acudiendo a varios puntos de esta zona para luego volver a patrullas.

A estos comportamientos se le añade el comportamiento social por el cual el enemigo interactuará con otros al entrar en una zona determinada del nivel y si no logra establecer esta comunicación se le aplicará un modificador de alerta.

Tiempo de detección: el tiempo que se tarda en detectar al jugador una vez se ha establecido contacto visual es de 2 segundos, 1 si se está aplicando el estado de alerta.

5 Diseño de los controles

A la hora de controlar el personaje se presentan dos alternativas de uso por defecto según el dispositivo de entrada que se esté utilizando, se distinguen:

1. Teclado y ratón

   Entrada por defecto que se usará en la mayoría de las ocasiones al tratarse de un juego enfocado al PC, a continuación, se muestran dos esquemas de control:

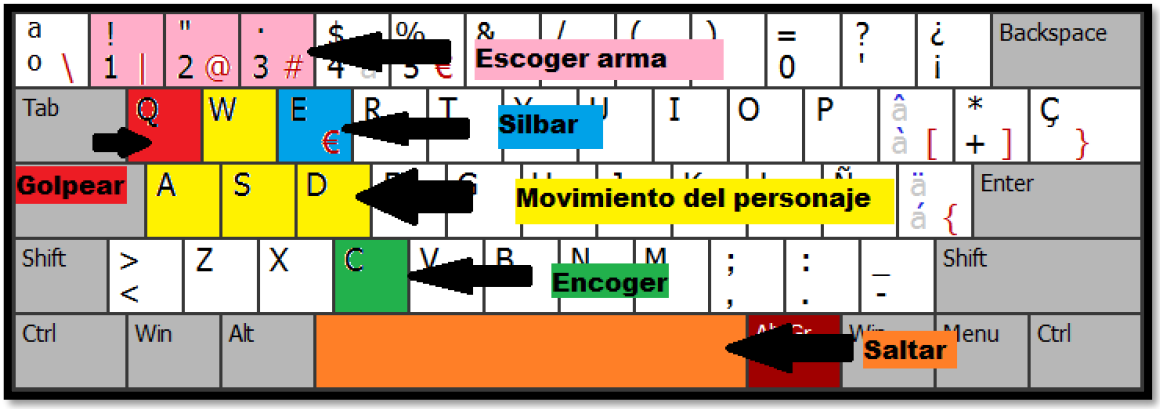


Figura 26: Diseño controles teclado

La imagen muestra los inputs que se podrán realizar por defecto con el teclado y su disposición en el mismo.

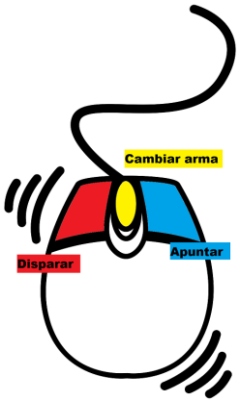


Figura 27: Diseño controles ratón

Complementando estos movimientos está el ratón que se encarga de mover la cámara con siguiendo los movimientos de los ejes vertical y horizontal del ratón y las acciones de

disparar y apuntar controladas con los botones izquierdo y derecho. También se puede cambiar de arma utilizando la rueda del ratón como alternativa al uso del teclado.

2. Mando

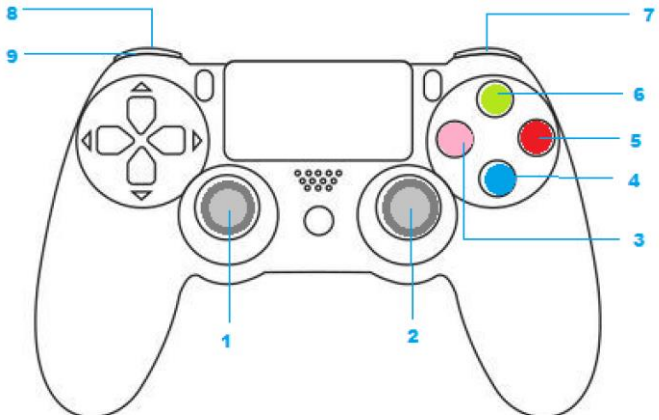


Figura 28: Diseño controles mando

Los controles cambian cuando se trata de un mando y se pierde la capacidad de escoger un arma especifica teniendo que rotar por la lista con la pulsación de una tecla. Los números del esquema se corresponden con:

1. Movimiento del personaje
2. Movimiento de la cámara
3. Golpear
4. Saltar
5. Encoger
6. Cambiar de arma
7. (Gatillo derecho) Disparar
8. (Gatillo izquierdo) Apuntar
9. Silbar

6 Diseño de los ítems

Se han diseñado ítems u objetos adicionales que complementan los diseños anteriores y que servirán como objetivos en el nivel o para ayudar al jugador a completarlos.

1. Tarjeta

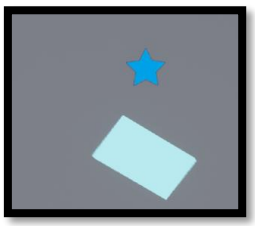


Figura 29: Diseño ítem tarjeta

Un pequeño objeto que el jugador tendrá que recoger para completar el nivel, normalmente estará custodiada por enemigos o será necesario resolver algún puzle para

obtenerla. Rotará en el aire rotando sobre sí misma y contará con un elemento que se representará en el HUD del jugador indicando donde se encuentra.

La tarjeta tiene un número asignado, que corresponderá con el número de la salida lo que permite comprobar si la tarjeta obtenida da acceso a la salida en la que se encuentra el jugador.

2. Plataforma de teletransporte

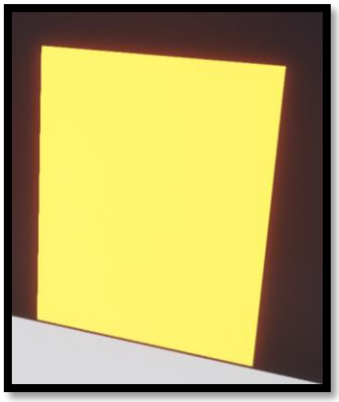


Figura 30: Diseño ítem plataforma de teletransporte

Complementaria al arma de teletransporte del personaje principal, definirá los puntos a los que se puede teletransportar el jugador, será necesario golpearlos con el proyectil de dicha arma para activarlos. Brilla con un color anaranjado que ayuda al jugador a identificarla como plataforma de teletransporte.

3. Caja de munición

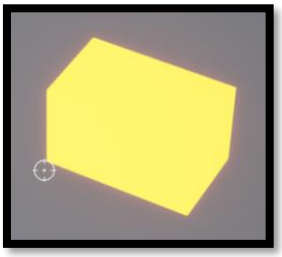


Figura 31: Diseño ítem caja de munición

Se encuentran distribuidas por el mapa y servirán para reponer la munición del jugador, no es necesario recogerlas para completar el nivel, pero ayudarán al jugador en caso de necesitarlo.

- Recarga: 3 balas del arma principal y 5 del arma de ruido.

4. Arma nueva

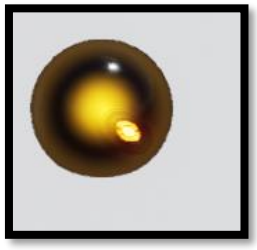


Figura 32: Diseño ítem arma nueva

Esfera flotante que sirve para que, al recogerla, el jugador desbloquee otra arma que le ayude a completar el nivel.

5. Salida
   Situadas en algún extremo del nivel, se trata de áreas invisibles que servirán para moverse al siguiente nivel una vez completado el actual. Cuentan con un número que indica la tarjeta necesaria que lo abre y muestra un mensaje si la tarjeta que se tiene no es la correcta. Además, de forma similar a la tarjeta, tendrán un elemento que se mostrara en la interfaz del jugador, indicándole que este es su objetivo actual.

7 Diseño de niveles

El juego contará con dos niveles diferenciados, de dificultad creciente, que servirán para mostrar las posibilidades de las mecánicas definidas para el juego, además de suponer un desafío para el jugador.

1. Nivel 1

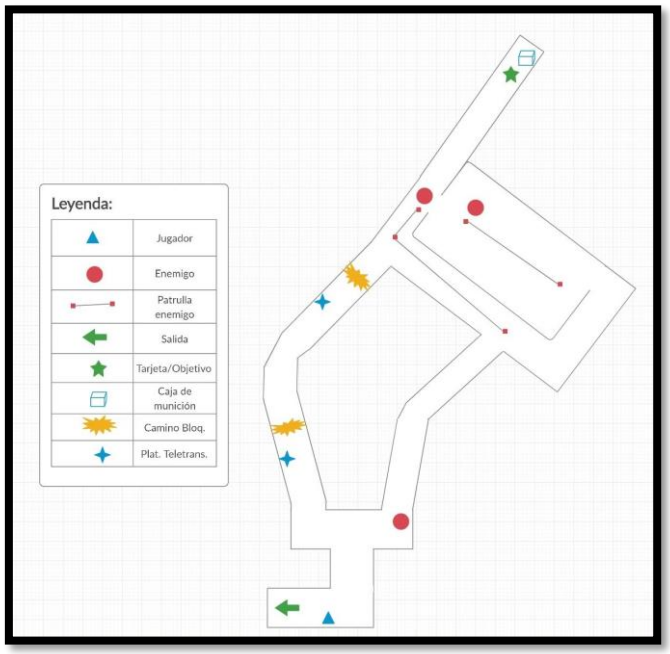


Figura 33: Diseño nivel 1

El primer nivel consiste en una introducción a las mecánicas del juego [19], el jugador empezará tan solo con el arma principal y al conseguir el objetivo desbloqueará el arma de teletransporte, lo que le permitirá llegar a la salida por una ruta segura.

2. Nivel 2

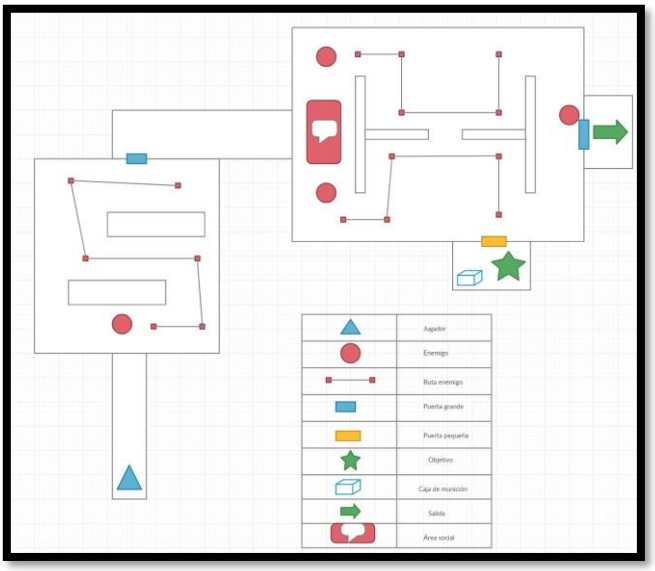


Figura 34: Diseño nivel 2

En este segundo nivel, se incrementa la dificultad, se introduce la mecánica del arma de ruido, además de mostrar el comportamiento social de los enemigos. A diferencia del nivel anterior, en este no se regresa al principio del nivel para completarlo, sino que al conseguir el objetivo se continuará hasta la meta.

8 Interfaz

De acuerdo con la solución al problema propuesta en el apartado 2.3, el diseño de la interfaz pretende ser funcional tanto en pc como en consola y será una interfaz no diegética. Además, elementos como los indicadores de sospecha de los enemigos o los objetivos del jugador, estarán situados en el espacio 3D del juego.

Con el propósito explicar las decisiones tomadas en el diseño de la interfaz, esta se ha dividido en los siguientes apartados:

8.1 Menús

Este apartado comprende tanto el menú de inicio como el menú de pausa además de los submenús que ofrecen y la navegación entre ellos, el primero será accesible al iniciar la aplicación, y al segundo se podrá acceder en cualquier momento al pausar el juego.

Menú principal

Figura 35: Menú principal

Al menú principal llegaremos al iniciar la aplicación y presenta el título del juego, el nombre del desarrollador y cuatro opciones:

- Jugar: permite empezar a jugar el primer nivel del juego.
- Seleccionar nivel: lleva al menú de selección de niveles.
- Ajustes: muestra el menú de ajustes.
- Salir: muestra una pantalla donde se ha de confirmar si se desea salir.

Menú de selección de niveles

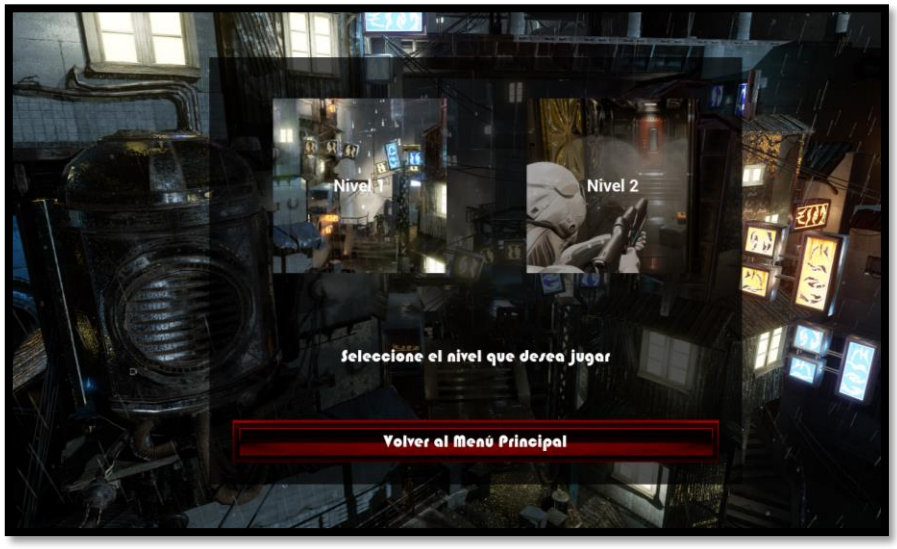


Figura 36: Menú selección de nivel

El nivel de selección de niveles permite acceder al nivel del juego que se desee y accedemos a él a través de su botón correspondiente en el menú principal, ofrece tres opciones:

- Nivel 1: pone en marcha el juego comenzando por el primer nivel.

- Nivel 2: pone en marcha el juego comenzando por el segundo nivel.
- Volver al menú principal: regresa al menú anterior, es decir el principal.

Menú de ajustes

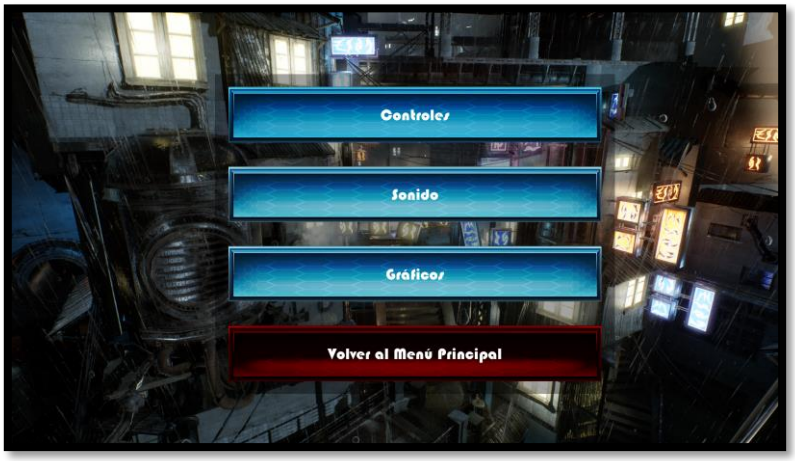


Figura 37: Menú ajustes

Accesible a través tanto del menú principal como por el de pausa, este menú presenta diversas opciones:

- Controles: accede al menú de ajustes de controles.
- Sonido: accede al menú de ajustes de sonido.
- Gráficos: accede al menú de ajustes gráficos.
- Volver: regresa al menú principal o al de pausa según desde donde se acceda a este menú.

Ajustes de controles

Figura 38: Menú controles

Menú permite cambiar las acciones asignadas a los distintos inputs, presenta dos columnas de selectores que el pulsarlos quedan a la espera de un nuevo input al que se le asignará la acción.

- Volver al menú de ajustes: regresa al menú anterior.

Menú de sonido

Figura 39: Menú sonido

Segunda opción del menú de ajustes permite cambiar el volumen de sonido de diferentes canales a través de *sliders* con valores de 0 al 100:

- General: ajusta el volumen general del juego.
- Música: regula el volumen de la música del juego.
- Efectos: regula el volumen de los efectos, disparos, explosiones, etc.

Ajustes gráficos

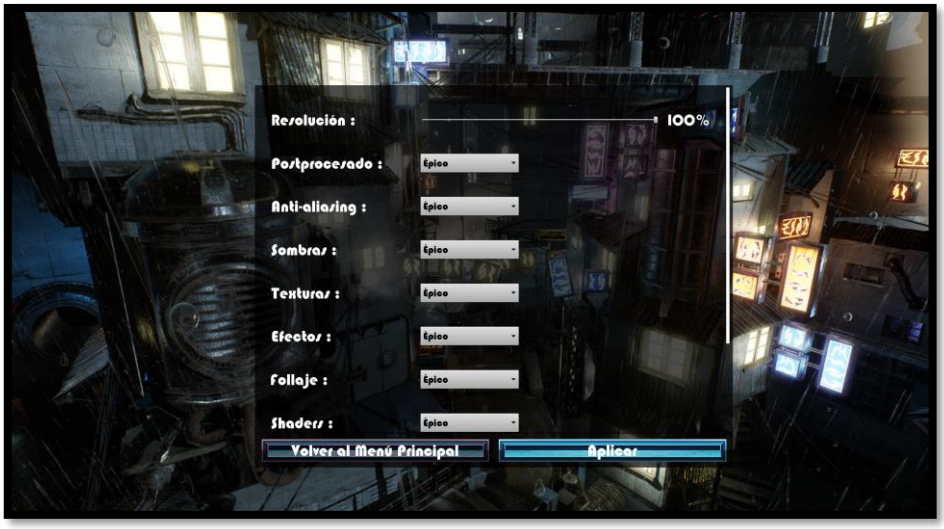


Figura 40: Menú Gráficos

Tercera opción del menú de ajustes es aquí donde el jugador puede adaptar los gráficos del juego a la potencia de su ordenador contando con las opciones:

- Escala de resolución: slider con una escala del 10-100% que controla en que porcentaje de resolución con respecto a la resolución de la pantalla se renderiza el juego.

- Opciones con selector: permiten seleccionar una calidad entre 4 (Baja, Medio, Alta, Épico y Cinemático)
- Opciones con casilla de verificación: hay dos casillas de verificación que controlan la inclusión o no de niebla volumétrica y el uso de sincronización vertical.

Además de dos botones:

- Volver al menú de ajustes: regresa al menú anterior.
- Aplicar: aplica los cambios en las opciones graficas que lo requieran.

Menú de pausa

Figura 41: Menú pausa

Se accede a este menú pulsando la tecla “P” o a la que se haya asignado en el menú de ajustes de controles. Pausa la acción del juego y ofrece las siguientes opciones:

- Continuar: continua con la ejecución del juego.
- Ajustes: accede al menú de ajustes.
- Salir: regresa al menú principal.

8.2 HUD (Head-Up Display)

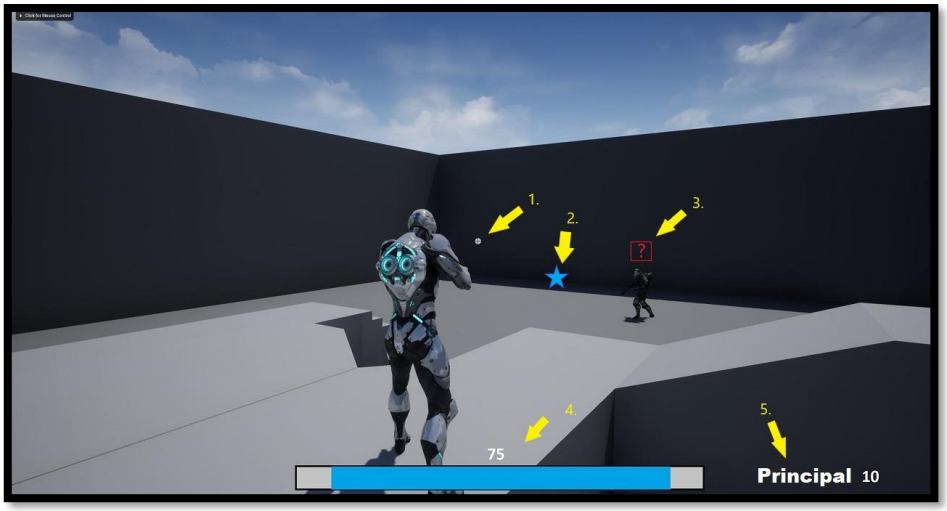


Figura 42: Diseño HUD jugador

El HUD contiene los elementos necesarios para guiar al jugador a lo largo del nivel, en el encontramos información relativa al siguiente objetivo a alcanzar, la salud de nuestro personaje o el estado en que se encuentran los enemigos. Siguiendo la numeración de la imagen, encontramos los siguientes elementos:

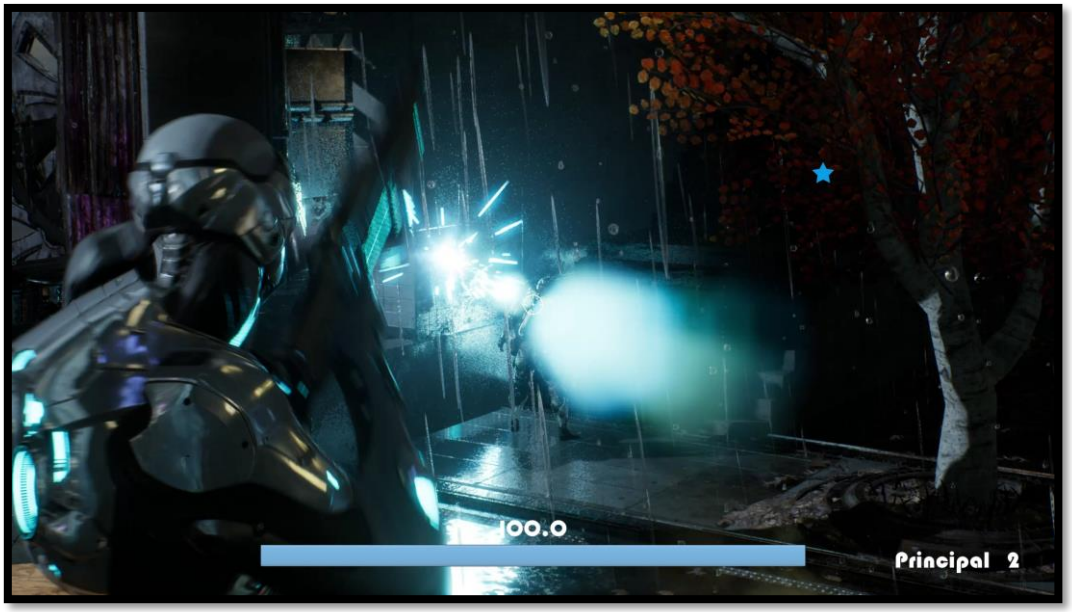
1. Punto de mira, indica la dirección en que se dispara el proyectil.
   Se ubicará en el centro de la pantalla y solo aparecerá cuando el jugador este realizando la acción de apuntar, de manera que no obstruya a la visibilidad durante los momentos en los que no es necesaria. De la misma forma, tendrá un tamaño reducido permitiendo que el jugador tenga mayor visibilidad del objetivo al que va a disparar. Por este motivo el porcentaje de espacio de pantalla que ocupará será del 0.3%.
2. Indicador de objetivo, está situado en el espacio 3D del juego y es visible a través de las paredes lo que ayuda a guiar al jugador hasta el siguiente objetivo.
   Será visible en todo momento a través de las paredes de forma que el jugador sepa en todo momento la posición de su próximo objetivo. El porcentaje de pantalla que ocupará dependerá de la distancia a la que se encuentre debido a su naturaleza espacial. Este objetivo se sitúa encima de ítems como la tarjeta o la salida del nivel descritos en el anexo sobre el diseño del juego.
3. Indicador de sospecha, comunica el estado de alerta en que se encuentra el enemigo.
   Este situado sobre el espacio 3D de forma similar al indicador de objetivo, pero se encuentra encima del enemigo. Indica la progresión de tiempo que ve al jugador hasta que entra en el estado de persecución y el descenso de esta alerta cuando ha perdido al jugador de vista.
4. Barra de salud, muestra la cantidad de vida que le queda al personaje principal.
   Se sitúa en la parte inferior central de la pantalla para impedir bloquear información del nivel, mientras se mantiene fácilmente accesible para consultar su estado, ocupa casi el 80% de la longitud horizontal del inferior de la pantalla dejando clara su importancia al jugador y un 12% de la vertical no obstruyendo la visión del jugador.

5. Arma seleccionada, dependiendo del arma que se haya equipado, mostrará su nombre y la cantidad de balas disponibles.
   Este elemento de la interfaz se encuentra situado en la parte inferior derecha de la pantalla en el espacio dejado por la barra de salud y cuenta con dos componentes diferenciados. El primero indica el arma que se está utilizando actualmente en forma de texto y el segundo consiste en un contador de las balas disponibles en esta arma.

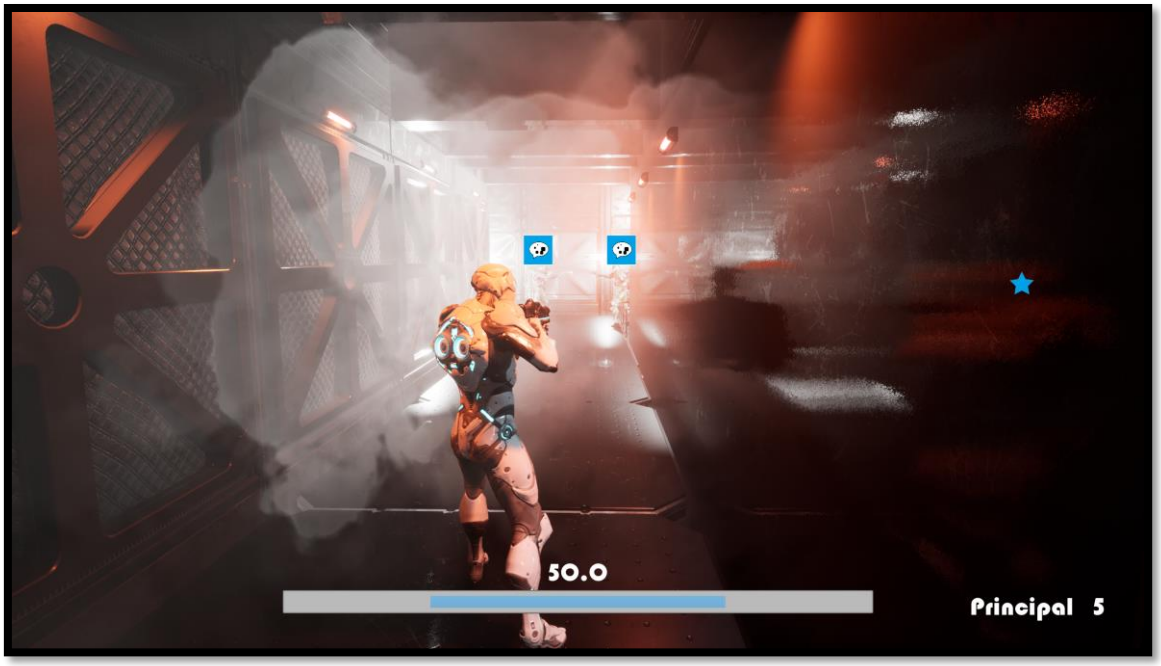
Anexo: Recursos utilizados

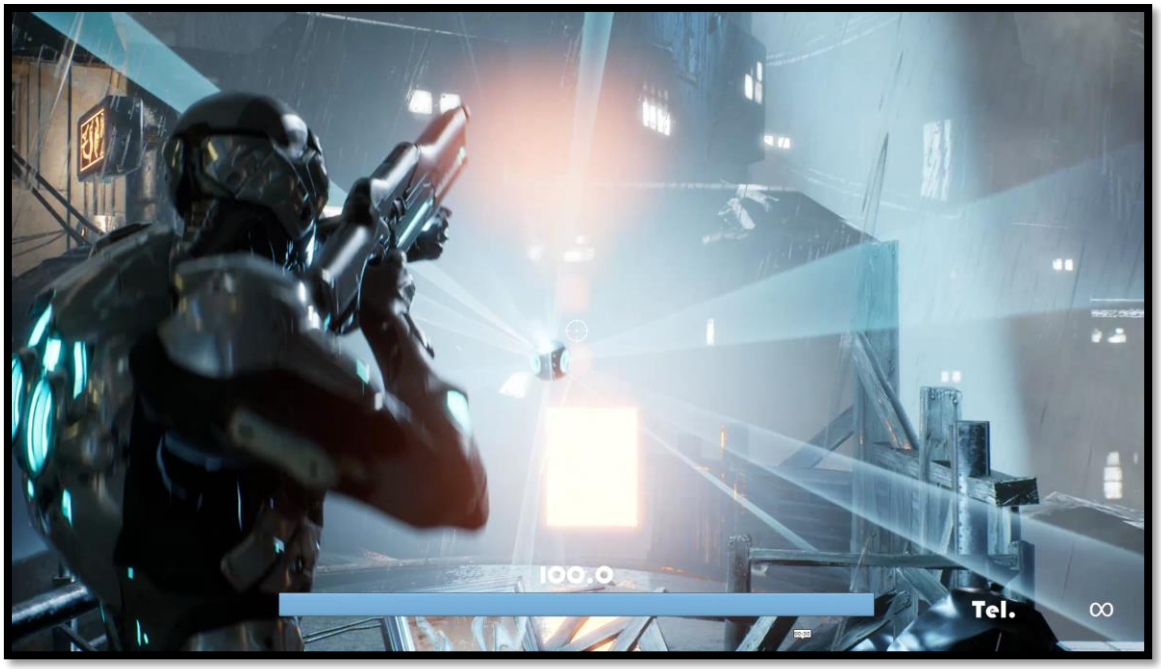
- Soul City – Epic Games – Jan 23,2018 https://www.unrealengine.com/marketplace/en-US/product/soul-city
- Modular Scifi Season 2 Starter Bundle - Jonathon Frederick - Feb 5, 2018- https://www.unrealengine.com/marketplace/en-US/product/modular-scifi-season-2-starter-bundle
- Paragon: Drongo - Epic Games – Sep 4, 2018 https://www.unrealengine.com/marketplace/en-US/product/paragon-drongo
- Paragon: Wraith – Epic Games – Sep 4, 2018 https://www.unrealengine.com/marketplace/en-US/product/paragon-wraith
- Sonidos adquiridos en: zapsplat.com - https://www.zapsplat.com/
- Música: Orion – CO.AG Music – https://www.youtube.com/watch?v=wrJoitsyVR8
- Fuente de texto: Blippo – https://sp.maisfontes.com/blippo
- Mirilla del arma: Jing.fm – https://www.jing.fm/sclip/crosshair/

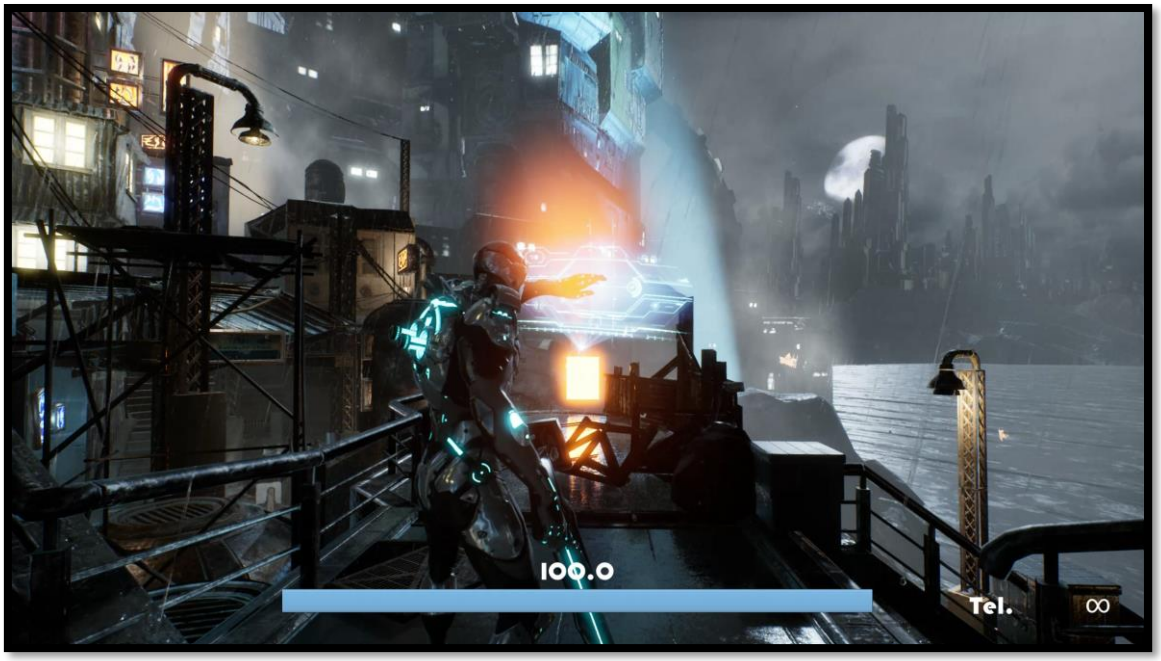
Anexo: Aspecto final del juego




100.0
9


50.0
Principal 5


100.0
Tel.
∞


100.0
Tel.
∞

Anexo: Juegos del Género del Sigilo

Pac-man

- Fecha de lanzamiento: 21-05-1980
- Compañía: Namco
- Página del juego: https://pacman.com/en/
- Características: El protagonista del videojuego Pac-Man es un círculo amarillo al que le falta un sector, por lo que parece tener boca. Aparece en laberintos donde debe comer puntos pequeños. El objetivo del personaje es comer todos los puntos de la pantalla, momento en el que se pasa al siguiente nivel o pantalla. Sin embargo, cuatro fantasmas, recorren el laberinto para intentar capturar a Pac-Man. (Wikipedia)

Castle Wolfestein

- Fecha de lanzamiento: septiembre 1981
- Compañía: Muse Software
- Secuelas: Wolfestein II: The New Colossus, Return to Castle Wolfenstein
- Características: juego de disparos en tercera persona, en el que se controla a un prisionero aliado con la misión de escapar de un calabozo fuertemente custodiado y destruir los planes de los nazis. (Wikipedia)

Metal Gear Solid

- Fecha de lanzamiento: 03-09-1998
- Compañía: Konami
- Secuela: Metal Gear Solid 2: Sons of Liberty
- Características: el jugador debe manejar al protagonista, Solid Snake, a través de las áreas del juego sin ser detectado por los enemigos. Para evitar ser detectado, el jugador puede realizar técnicas que hacen uso de las capacidades tanto de Solid Snake, como del medio ambiente, tales como gatear debajo de

objetos, usar cajas como escondite, agacharse o esconderse alrededor de las paredes, y hacer algún ruido para distraer a los enemigos. (Wikipedia)

Metal Gear Solid 3: Snake Eater

- Fecha de lanzamiento: 17/09/2004
- Compañia: Konami
- Secuela: Metal Gear Solid 4: Guns of the Patriots
- Características: La mecánica de juego de Snake Eater es similar a la de juegos anteriores de la serie. Snake, controlado por el jugador, debe atravesar espacios y lugares plagados de enemigos sin ser detectado

Metal Gear Solid 5: The Phantom Pain

- Fecha de lanzamiento: 1-09-2015
- Compañia: Konami
- Página del juego: https://www.konami.com/mg/mgs5/
- Características: El videojuego incorpora un sistema de mundo abierto, en el cual el jugador tiene la posibilidad de realizar misiones de forma no-lineal, con un total de 50 misiones principales y 157 secundarias.

Tenchu: Stealth Assassins

- Fecha de lanzamiento: 26/02/1998
- Distribuidora: Activision
- Secuela: Tenchu 2: Birth of the Stealth Assassins
- Características: El jugador ejerce el papel de un ninja, a lo largo de 10 pantallas, donde en cada una de ellas deberá cumplir una misión. (Wikipedia)

Thief: The Dark Project

- Fecha de lanzamiento: 30/11/1998
- Compañia: Eidos Interactive
- Secuela: Thief Gold, expansión de este
- Características: Thief fue el primer juego de sigilo en utilizar la luz y el sonido como la mecánica de juego, y el primero que cuenta con una perspectiva en primera persona. (Wikipedia)

Tom Clancy's Splinter Cell

- Fecha de lanzamiento: 17-11-2002
- Compañia: Ubisoft
- Página del juego: Splinter Cell
- Secuela: Tom Clancy's Splinter Cell: Pandora Tomorrow

Tom Clancy's Splinter Cell Conviction

- Fecha de lanzamiento: 13/04/2010
- Compañia: Ubisoft
- Página del juego: Tom Clancy's Splinter Cell Conviction
- Secuela: Tom Clancy's Splinter Cell Blacklist

Assassin's Creed

- Fecha de lanzamiento: 13/09/2007
- Compañia: Ubisoft
- Secuela: Assassin's Creed II
- Página del juego: Assassins Creed

Alien Isolation

- Fecha de lanzamiento: 06/10/2014
- Compañia: The Creative Assembly
- Página del juego: Alien Isolation

Far Cry 3

- Fecha de lanzamiento: 29/09/2012
- Compañia: Ubisoft
- Secuela: Far Cry 4
- Página del juego: Far Cry 3

Sly Cooper

- Fecha de lanzamiento: 23/09/2002
- Compañia: Sucker Punch Productions
- Secuela: Sly 2: Ladrones de guante blanco
- Página del juego: Sly Cooper

Batman: Arkham City

- Fecha de lanzamiento: 18/10/2011
- Compañia: Rocksteady
- Secuela: *Batman: Arkham Knight*
- Página del juego: Batman: Arkham City

Mark of the ninja

- Fecha de lanzamiento: 7/09/2012
- Compañia: Klei Entertainment
- Página del juego: Mark of the ninja