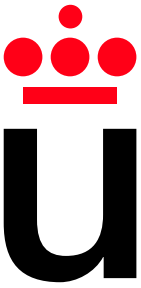

Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

Grado en Desarrollo y Diseño de videojuegos

2024-2025

Trabajo Fin de Grado

DESARROLLO DE UN SISTEMA DE COMBATE POR TURNOS INMERSIVO A TRAVÉS DE UN SISTEMA DE UTILIDAD EN VIDEOJUEGOS

Autor: Violeta Trujillo Almeida
Tutor: Nicolás Rodríguez Uribe

Agradecimientos

En el ambito academico quiero darle las gracias a la universidad y a todos los profesores que me han ido acompañando a lo largo de este viaje, especialmente mi tutor Nicolas Rodr guez Uribe, por su gran profesionalidad y apoyo como docente, tanto a lo largo de la carrera, como en este proyecto.

Quiero tambien mostrar mi agradecimiento a mis compañeras de grado de videojuegos del campus de Quintana **Marta Rodríguez Castillo** y **Elena López-Negrete Burón** por haberme cedido parte de su arte y haberme ayudado para poder mostrar visualmente el potencial de lo desarrollado. Otro agradecimiento muy especial a mi compañero **Antón Rodríguez Seselle** alumno del Doble Grado de Desarrollo y Diseño de videojuegos e Ingenier a de Computadores por haberme cedido su gran capacidad para la composicion musical en la Demo del combate de este videojuego. Tambien especial mencion a la fuente gratuita de assets para videojuegos [1]

Quisiera expresar mi mas sincero agradecimiento especialmente a mis padres, porque se han esforzado enormemente para que yo pudiese estudiar una carrera universitaria.

Por ultimo darle las gracias a mi hermano mayor, Ricardo Trujillo Almeida porque es la persona que mas me ha apoyado en absolutamente todo a lo largo de mi vida, se encontrase o no en condiciones para hacerlo. Hoy y para siempre mil millones de gracias.

Índice de contenidos

Índice de figuras

1
Introducción

El objetivo de este proyecto, es el desarrollo de un videojuego de combate por turnos, que posea el mundo, reglas, e **inmersión más realista y sumergida** posible. Para ello, uno de los mas relevantes, es que el jugador sienta que la IA de dicho juego, sea lo mas realista posible. Se busca que su comportamiento y relaciones con la misma, sean lo mas cre bles posible, y para ello, es necesario que la IA proporcione al jugador una experiencia rica y disfrutable.

Para este trabajo, se realiza un desarrollo desde cero de un sistema de utilidad aplicado a un videojuego de combate por turnos. De esta manera, se consigue un combate rico y experimental que mejora la experiencia del usuario. Esto se debe a que se obtiene un combate con mayor complejidad, mas realista e inmersivo.

Esto no solo se consigue con el desarrollo de un sistema de utilidad, sino que ademas, tratando las acciones, factores de decision, factores de fusion y sus respectivos resultados con elementos que el propio jugador pueda identi car y sentirse familiarizados con ellos (Gritos/discusiones, hablar/dialogar, llorar, ansiedad, etc...). De esta manera, los elementos principales en los que se divide este sistema de utilidad son:

1. Acciones: gritar, hablar y llorar.
2. Factores de decision: ansiedad y puntos de integracion social (de la IA y del jugador).
3. Factores de fusion: fusionar estos factores de decision.
4. Resultados: dependiendo del estado del enemigo y el jugador, realizar una accion u otra sera determinante para el equilibrio del sistema de utilidad.

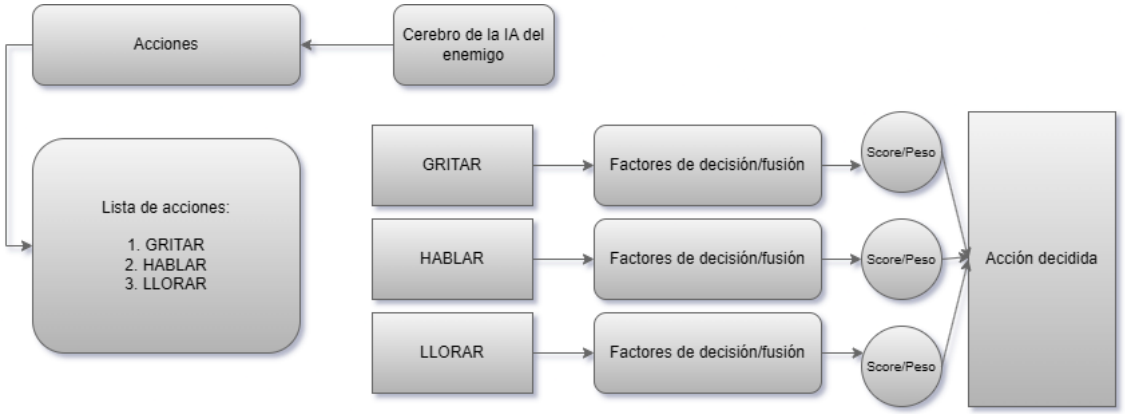

Figura 1.1: Esquema de Acciones

De esta manera, se consigue desarrollar un combate **RPG** (role-playing game), en el cual, los enemigos que el jugador tiene que desa ar para superar combates poseen una IA realizada integramente con sistema de utilidad cuyo esquema aparece en la Figura 1.1.

1.1. Contexto y alcance

Como se ha explicado previamente, el desarrollo de un sistema de utilidad con estas caracter sticas inmersivas en este tipo de juego (RPG), se dedica, no solo la busqueda de una IA que ofrezca comportamientos mas cercanos a la realidad, en cuanto combate se re ere, como se ha implementado recientemente en juegos como *Killzone 2*. [2]

Tambien, se busca asociar este combate a comportamientos y sentimientos que el jugador pueda relacionar directamente con actividades o hechos que le ocurran en su vida cotidiana. En un combate, el jugador puede empatizar mas con que grites, o discutas con una persona, a que le ataques sin motivo. Este tipo de sentimientos, acciones y factores de fusion se asemejan mas al funcionamiento de clasico videojuego: *The Sims* [3]. De esta manera, se obtiene un tipo de combate mas realista e inmersivo.

1.2. Estructura del TFG

Estructura de la memoria nal asociada a este TFG es la siguiente:

1. Introduccion.

2. Objetivos (incluyendo descripcion del problema, estudio de alternativas y metodolog a empleada).

3. Diseño y creacion del entorno para el desarrollo de la IA (diseño del mundo, escenarios, personajes y combate).

4. Desarrollo e implementacion tecnica

5. Conclusiones (incluyendo los logros principales alcanzados y posibles trabajos futuros).

6. Bibliograf a.

2

Objetivos

El objetivo de este proyecto es un sistema de combate para mejorar la experiencia del jugador, y que ademas, sea *cre ble, realista e inmersiva*. El resto del cap tulo consta de:

1. Diseño de un entorno. En la Seccion 2.1 se encuentran los subobjetivos
2. Estudio de alternativas. En la Seccion 2.2 se exploran las herramientas que se acercan, pero no lo su ciente, al objetivo de este proyecto.
3. Metodolog a. En la Seccion 2.3 se detalla la metodolog a seguida en este proyecto.

2.1. Diseño de un entorno

Antes de comenzar el diseño e implementacion del **Sistema de Utilidad** (**SU**), es necesario desarrollar un entorno y sistema de combate adecuados para la meta que se quiere alcanzar. *Worldbuilding, Game Design y Level Design*. Estos terminos son muy amplios, y abarca, el estudio y creacion de un mundo, su sistema y diseño, desde sus personajes, hasta sus reglas y ambientacion, para la implementacion de este **SU** son imprescindible los siguientes aspectos:

1. Coherencia interna: que combatir tenga un sentido.

2. Ambientacion y consistencia detallada: que el combate sea una manera de introducirnos profundamente en el juego.

3. Sistema de reglas: el sistema de combate e interaccion con el entorno.

Para conseguir todo esto, se disena un mundo en que se dan una serie de reglas, vivencias y situaciones concretas que son contadas a traves de una serie de personajes. Esas experiencias son contadas mediante el punto de vista de nuestra protagonista.

2.2. Estudio de alternativas

Para pasar al desarrollo adecuado de un **SU** aplicado al combate de un videojuego RPG por turnos, se ha de estudiar a traves de la meta que se quiere alcanzar, cuales son los tipos de herramientas que pueden aplicarse a IAs mas comunes aplicadas en videojuegos, y sus tipos de diseno, y cual ser a la mas adecuada para este caso. Las herramientas para disenar IAs mas aplicadas en videojuegos [4] en las ultimas decadas en las siguientes secciones:

2.2.1. Máquina de estados finita (FSM)

Máquina de estados finita (FSM): Las maquinas de estado nitas han sido el **metodo de procesos de control y toma de decisiones** mas conocido aplicado a la Inteligencia Arti cial de videojuegos hasta mediados de la decada de los 2000 [4]. Esto se debe, principalmente, a su facilidad de diseno e implementacion. Se tratan de procesos de control de estado muy sencillos, que carecen de exibilidad, sin embargo, esta sencillez en su diseno, es lo que permite, un uso funcional pero muy basico.

Las maquinas de estado nitas tienen tres componentes principales:

1. **Estados**: almacena la informacion de un determinado estado de la FSM.

2. **Transiciones**: son las condiciones que se deben de dar para que la FSM pase de un estado a otro.

3. **Acciones**: la secuencia de acciones que se deben de realizar en cada uno de los estados.

Tamb en cabe destacar que existen dos tipos de **FSM**, son las siguientes:

1. **Moore**: Su salida unicamente depende del estado actual, y esta sincronizada directamente con el estado.

2. **Mealy**: Su principal diferencia respecto a las maquinas moore son, que su salida no solo depende del estado actual, sino tambien de la entrada proporcionada. Ademas esta maquina no esta sincronizada con el estado, sino, a traves de un pulso de reloj.

A pesar de estas pequeñas diferencias, su funcionamiento y diseño es similar, as como sus componentes. Una breve tabla explicativa sobre sus diferencias es la siguiente Tabla 2.2.1:

Caracter sticas	Mealy	Moore
Salida	Estado y entrada actual	Estado actual
Sincronizacion	Salida con reloj	Salida con estado

Aqu se puede observar un breve esquema de ejemplo del funcionamiento de una **FSM** con dos variables que establecen las posibles condiciones que se pueden dar. Se ha elaborado un esquema, donde una IA, diseñada con la herramienta de FSM, de un videojuego posee tres estados con sus respectivas acciones, y dos variables que seran las que estableceran las condiciones para que los estados transicionen entre s .

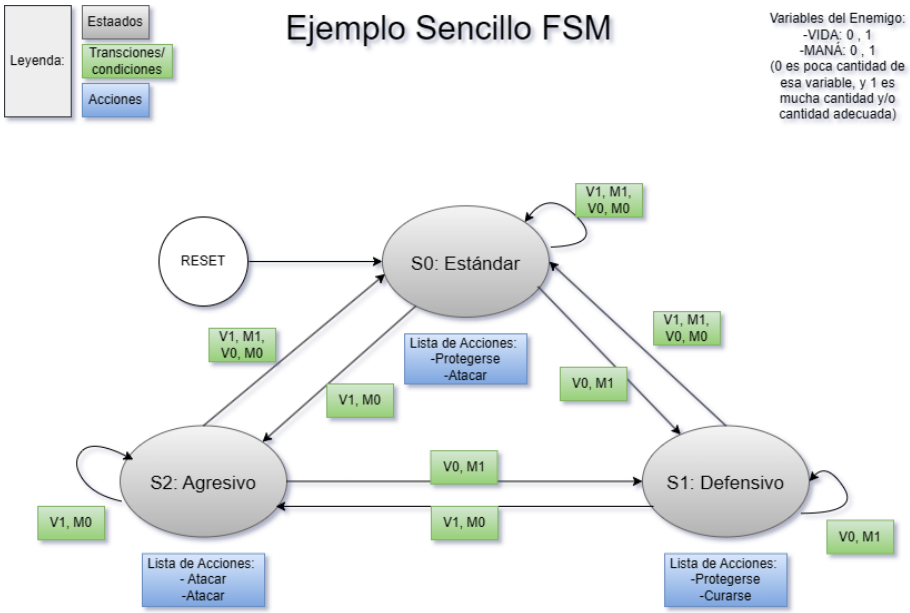

Figura 2.1: Esquema Ejemplo FSM

En la Figura 2.1 se pueden observar los distintos estados de la FSM, sus acciones y las condiciones que se tienen que dar para transicionar de estados. Estos vienen dados por un resultado binario, preestablecido por el diseñador. Por estas razones, se puede ver que la maquina de estados nita. Es simple y sencilla de diseñar, pero extremadamente simple y poco exible, ademas esta sujeto a unas condiciones muy espec cas, que dan como resultado patrones de comportamiento repetitivos. Por ello, se descarto su uso para este proyecto, ya que no aportar a, credibilidad realismo e inmersion al jugador.

2.2.2. Beheaviour Trees (BT)

Al igual que las FSM, los arboles de comportamiento son un sistema usado para controlar el comportamiento de personajes, y es muy similar a las FSM, pero con ligeras diferencias de funcionamiento y estructura: [4]

- No se centran en estados, se centran en comportamientos.
- Parten de un nodo ra z que va bifurcandose en comportamientos mas simples, es decir, al contrario que las FSM, se da la existencia de un **nodo padre**.
- La estructura y funcionamiento de sus nodos es distinta, tanto en el resultado que pueden devolver los nodos, como los diferentes tipos de nodo (**en FSM, todos los nodos funcionaban igual, solo cambiaba el resultado devuelto**).

La estructura y funcionamiento de estos nodos, son:

1. **Posibles resultados devueltos:**
   a) En ejecucion (comportamiento en curso)
   b) Exito (comportamiento se pudo completar)
   c) Fracaso (comportamiento no se pudo completar)
2. **Tipos de nodos:**
   a) **Secuencia**: requiere que todos los comportamientos secundarios (hijos directos) tengan exito. Si alguno falla, toda la secuencia falla (en logica, la secuencia es similar a un AND) [4].
   b) **Selector de probabilidad:** elige comportamientos secundarios en base a probabilidades establecidas por el diseñador. Si el comportamiento secundario elegido tiene exito, el selector lo tendra (en logica, la secuencia es similar a un **OR**) [4].

c) **Selector de prioridad:** prueba los comportamientos secundarios en un orden espec co. Si uno tiene exito, el selector lo tendra (en logica, la secuencia es similar a un **OR**) [4].

d) **Decorador:** Agrega complejidad a un comportamiento secundario individual. Aqui existen muchos tipos, loops, delays, inverter, etc... (Esto se suele dar para comportamientos mas complejos y/o espec cos que se salgan de la norma del esquema inicialmente disenado).

Para poder entender estos elementos, tambien se va a explicar con la Figura 2.2 el porque, a pesar de ser un poco mas complejo que una maquina de estados nita (FSM), no es su ciente para alcanzar el objetivo estudiado.

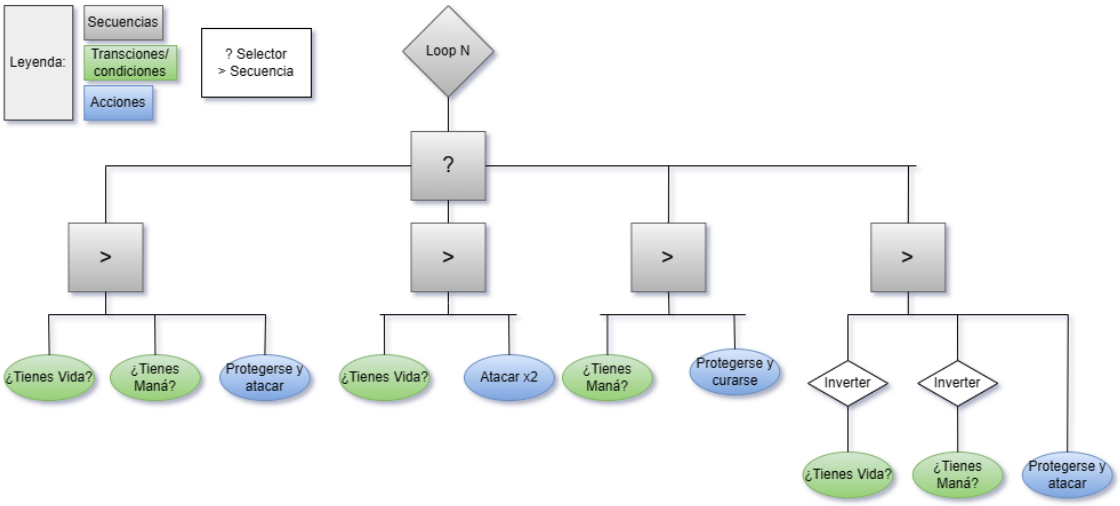

Figura 2.2: Esquema Ejemplo BH

A pesar de que los arboles de comportamientos sean mucho mas claros visualmente y mucho mas exibles que las maquinas FSM, sigue siendo un comportamiento restrictivo y que puede escalar de tal manera que las condiciones que se tengan que establecer para el correcto funcionamiento de la IA sean demasiado extensas para un **BH**. Si queremos un mayor realismo, y evitar comportamientos repititivos. Las acciones de esta IA se ejecutan por un orden de prioridad, no hay una seleccion de la mejor accion del arbol, se ejecuta la que antes se cumpla y esto evita un comportamiento realista, por lo tanto, tampoco es su ciente. Por ello, se descarta su uso para este proyecto.

2.2.3. Sistemas de utilidad (US)

Los sistemas de utilidad se trata de un metodo cada vez mas popular para la creacion de comportamientos. Esto se debe a que elimina la modularidad y poca

exbilidad de los previamente explicados **FSM** y **BH**. Se trata de un enfoque de IA que usa una herramienta basada en utilidades, es decir, mediante una serie de asignaciones de valor (pesos) que se le asigna a una determinada accion. Se decide cual es la mejor decision que puede tomar la IA en ese preciso instante. Este tipo de sistema esta basado en las redes neuronales [4]. A pesar de la clara mejora en exibilidad, se trata de un enfoque mucho mas tedioso y complejo de diseñar e implementar, ya que es difuso, y hay que tener en cuenta muchos aspectos de la IA. Los sistemas de utilidad tienen los siguientes componentes cruciales para su desarrollo:

1. **Acciones**: se trata del conjunto de acciones posibles que puede realizar la IA

2. **Factores de decisión**: las condiciones que asignaran los pesos a determinadas acciones determinando que importancia tiene cada accion, en segun que situacion, y porque.

3. **Factores de fusión:** cuando se unen las condiciones de varios factores de decision, en un solo factor, para asignar un determinado peso.

4. **Pesos, o suma de los mismos**: el peso asignado a cada una de las acciones. La accion con mayor peso sera la elegida. Esto se dara tras haber tenido en cuenta los factores de decision.

5. **Mejor acción decidida**: La accion con mayor peso que sera ejecutada por la IA.

En la Figura 2.3 para entender esta serie de elementos explicados, ser a el siguiente:

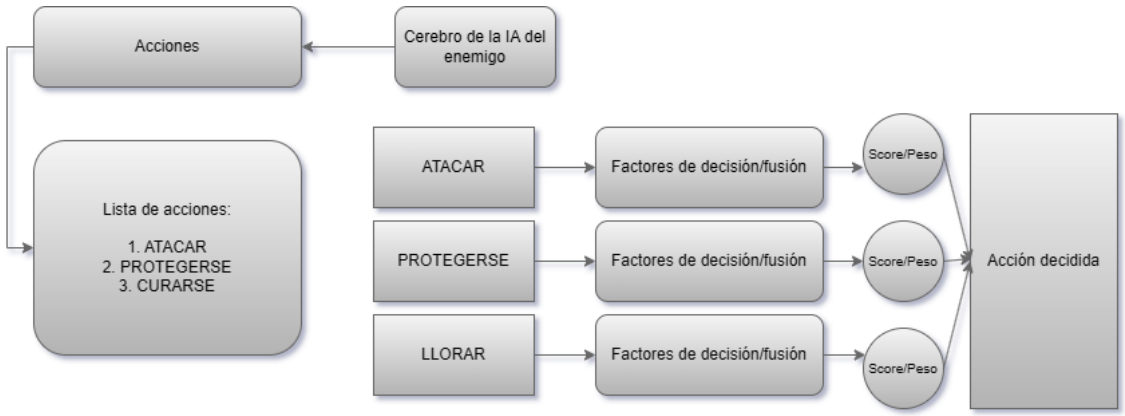

Figura 2.3: Esquema Ejemplo US

En videojuegos existen muchas mas formas de tratar una IA (esto puede englobar muchos terminos, desde herramientas como **FSM** y **BH**, ciertos tipos de algoritmos, y hasta redes neuronales, que es en lo que esta basado el uso de **SU** en videojuegos) y desarrollarla (generacion procedural, aleatoriedad, machine learning, algoritmos de busqueda y plani cacion, etc...). Sin embargo, muchas de estas herramientas no son utiles para el desarrollo de este tipo de Inteligencia Arti cial, debido a que cuando se busca un diseño de un sistema que funcione en base a una serie de reglas, como es el caso de un sistema de combates por turnos, lo mas comun, es guiarse por estas tres herramientas previamente mencionadas. Por todo eso, y por el objetivo que se persigue (algo que sea lo mas cercano al realismo posible), se ha decidido optar por el **Sistema de utilidad** [4]

2.3. Metodología:

Por ultimo, antes de empezar a exponer todo el desarrollo del diseño, es necesario plantear que se ha seguido para poder llevarlo a cabo, es decir, la metodolog a usada. Lo esencial de la metodolog a es destacar que se ha basado en el Mani esto Agil [5]. El motivo de esto es, que en este tipo de proyectos relacionados con videojuegos, el desarrollador ha de tener la capacidad para volver atras de manera exible, y recti car a tiempo. Para ello, un planteamiento de metodolog as mas tradicionales es inviable, por ello, se aplican las bases del Mani esto Agil 2.4, basadas principalmente en el esquema de visualizacion de tareas:

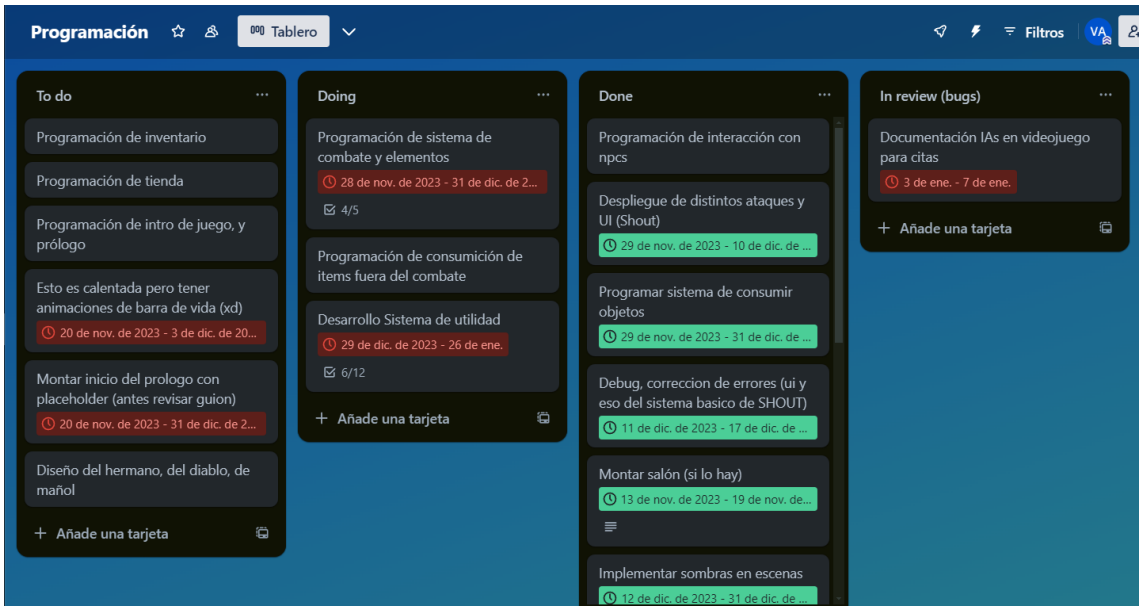

Figura 2.4: Esquema agil

Hay muchos tipos de metodolog as agiles, concretamente la usada para este

proyecto es la *Kanban*, por su facilidad iterativa mediante y su mejor organizacion para grupos reducidos y/o un solo individuo. La estructura seguida para la organizacion del desarrollo de este proyecto es [6]:

1. **Visualización de todas las tareas:** se usa una determinada herramienta para exponer todas y cada una de las tareas y su estado.
2. **Por hacer:** son todas y cada una de las tareas que aun no se han empezado
3. **En proceso:** son todas y cada una de las tareas que estan en proceso
4. **Terminadas:** son todas y cada una de las tareas que estan nalizadas
5. **En revisión:** son todas y cada una de las tareas que estan cerca de estar nalizadas, pero quedan una serie de problemas por solucionar.
6. **Priorización según importancia y urgencia, y seguimiento del tiempo:** se puede organizar todas las tareas del proyecto con la libertad de priorizarla segun la urgencia que se considere, y gestionar su fecha de nalizacion como se ve en la Figura 2.4.

Una vez expuesta la estructura de la metodolog a agil, existen una serie de herramientas para que se pueda dar un seguimiento adecuado del objetivo perseguido en **Kanban**, como ya se ha explicado, esta estructura se re eja en la Figura 2.4.

Para poder entender la estructura de visualizacion de tareas de la metodolog a *Kanban*, un ejemplo de esta tra dos al desarrollo de este proyecto ser a:

- **Requisitos:**
  - World-Building.
  - Diseño de juego y sistemas.
  - Diseño de enemigos y balanceo.
  - Diseño de niveles.
  - Diseño de mapas.
  - Diseño del funcionamiento del Sistema de Utilidad del enemigo.
- **Visualizaciones de tareas:**
  - Tarea 1: diseño de gestion de turnos del combate.
  - Tarea 2: diseño de posibles ataques del jugador.
  - Tarea 3: diseño de posibles ataques del enemigo.
  - Tarea 4: diseño de uso de posibles objetos por partel del jugador (inventario).

- Tarea 5: diseno de gestion de estad sticas del jugador.
- Tarea 6: diseno de gestion de las estad sticas del enemigo.

Se ha de remarcar, como se ve re ejado en la Figura 2.4, que la herramienta utilizada para aplicar la metodolog a *Kanban* es Trello, debido a su sencillo y visual diseno, ademas de la familiarizacion previa con esta herramienta.

3

Desarrollo y diseño del entorno

En este apartado se va a explicar paso a paso el procedimiento que se ha llevado a cabo para hacer un diseño y desarrollo del entorno adecuado. El objetivo es implementar una IA que tenga un sentido util y practico para el videojuego.

Antes de entrar en detalle en cada uno de los aspectos necesarios para desarrollar un buen entorno, se va a realizar una breve presentacion de cada uno de ellos mostrando una breve de nicion.

- **World-building**: es un proceso clave que se ha de realizar siempre que se trabaja sobre cualquier videojuego, por pequeño que sea. No en todos es necesario hacerlo de forma detallada, ya que se trata del proceso de crear un entorno adecuado y cticio, que sea coherente para una historia narrativa, y no todos los videojuegos requieren un entorno narrativo detallado (genero del videojuego, gameplay, etc...). Se corresponde con la Seccion 3.1.

- **Diseño de juego y sistemas** (*Game Design*): una vez contemplado el contexto narrativo del videojuego, es necesario pasar al diseño de mas elementos que son importantes para terminar de cerrar los aspectos previos a la implementacion tecnica. El diseño de juego o *game design* es crucial no solo para el disfrute del jugador, sino que a pesar de tener un contexto, no sirve de nada si no es aplicable a las mecanicas y dinamicas [7] del juego. Se corresponde con la Seccion 3.2.

- **Diseño de enemigos y balanceo**: el diseño de personajes y enemigos contemplado para este proyecto, abarca principalmente el apartado tecnico

de los mismos, es decir, que tipos de enemigos existen y que impacto tienen estos para los objetivos planteados. Se corresponde con la Seccion 3.3.

- **Diseño de niveles** (*Level Design*): el diseno de niveles es la parte crucial, que plantea el camino que llevara el jugador a lo largo de su aventura y los objetivos principales y secundarios, que ira alcanzando el jugador, con el objetivo de ir avanzando a traves de los escenarios. Se corresponde con la Seccion 3.4.

- **Diseño de mapas**: el diseno de mapas trata esa parte visual del diseno de niveles, que posee la arquitectura visual de todos y cada uno de los escenarios por los que tiene que pasar el jugador para ir alcanzando sus objetivos. Se corresponde con la Seccion 3.5.

- **Diseño del funcionamiento del Sistema de Utilidad del enemigo:** Por ultimo, se procedera a explicar el diseno del Sistema de Utilidad del enemigo, tras haber explicado todo el diseno previo necesario. Se corresponde con la Seccion 3.6.

Ahora que se han expuesto las distintas etapas del desarrollo del videojuego previas para poder implementar un **SU**, se puede pasar a hacer hincapie en el proceso expl cito de cada una de ellas. Una vez comprendida la metodolog a usada, se puede proceder a explicar el desarrollo del diseno.

3.1. World-building

Esta implementacion tiene como n principal, conseguir una IA lo mas realista e inmersiva para el jugador durante los combates. No solamente basta con un buen apartado tecnico, es imprescindible que todos los aspectos del combate sean familiares y cercanos para el jugador para una experiencia unica. Estos aspectos, re riendonos en un combate RPG por turnos clasico, se pueden resumir en los siguientes:

- **Estadísticas del jugador y del enemigo**: Estos elementos dentro de un juego RPG de combates por turnos suelen ser:

  1. Vida
  2. Poder de ataque
  3. Poder de defensa
  4. Mas elementos anadidos como: (mana, experiencia, magia, etc...) estos pueden variar dependiendo del juego.

- **Posibles acciones del jugador y del enemigo**: aqu entran otros tipos de variables, que son las acciones a realizar por cada turno. Aqu tambien pueden variar mucho dependiendo de cada juego, pero siempre hay unas bases solidas marcadas:

  1. Accion de atacar
  2. Accion de defender
  3. Accion de curarse
  4. Mas elementos anadidos

- **Objetos usables durante el juego**: se implementa para poder usar objetos activables dentro de un combate en un videojuego de este tipo. Se suelen usar elementos, nombres y aspectos del jugador que dentro del world-building previamente desarrollado sean familiares y facilemente reconocibles para el jugador, ya sean usados por el mismo, o por un enemigo.

Por ejemplo, si se construye un mundo y un tipo de combate con tematica medieval, no tiene sentido implementar objetos de uso comun o cotidiano en personas del *S. XXI.* Por tanto, todas las mecanicas de un juego facilmente reconocibles y familiares para el jugador.

El objetivo del *World Building* en este proyecto es implementar una IA de combate que se acerque al jugador lo maximo posible, que sea lo mas inmersiva posible, tanto en elementos tecnicos, contextuales y narrativos [8].

Antes de entrar en materia, y destacando lo que se busca con el World-Building en este caso, se desarrolla los siguientes elementos partiendo de la siguiente base narrativa:

La historia comienza en una pequena ciudad rural en la que el jugador se metera en el papel de una joven adolescente llamada Elisa de unos 16 anos, en la que a lo largo de su pequena aventura en la que sale de su casa, en el que algun d a llamo hogar, se pasara paseando por su ciudad con sus amigos y enfrentando todo tipo de con ictos sociales, que se iran narrando mediante criticas sociales sumergidas en capas de iron a sat rica. En resumen, Elisa se encuentra en una situacion emocional complicada en la que se le ha olvidado realmente quien es y hacia donde va, sumergida en un bucle vicioso de ansiedad y querer sentirse integrada en la sociedad. Estos dos elementos seran cruciales para desarrollar el world-building por completo [8].

Una vez se han tenido en cuenta los distintos aspectos del *World-Building*, y su contexto narrativo, se puede explicar el proceso de desarrollo del mismo sabiendo

a ciencia cierta lo que se busca, las bases del *world-building*, son las siguientes [8]:

- **Gobierno, sociedad, cultura y religión**: Estos elementos son los mas reconocibles e importantes del world-building. A pesar de ser elementos distintos entre s , tienen una gran correlacion entre ellos, porque se tratan de aspectos que giran en torno a como se comportan los personajes. Esto es debido a que son el re ejo del comportamiento del mundo y su sociedad.

  En primer lugar, su **cultura**. Es la base principal y centralizada del resto de aspectos, ya que la sociedad, gobierno y religion, se rigen en base a la cultura existente de dicho mundo, y que ha sido heredada por una serie de motivos historicos y contextuales que pueden explicarse a lo largo del desarrollo narrativo.

  La cultura existente en esta historia sigue corrientes comunes a la cultura occidental, su forma de vida, ra ces y forma de pensar son extremadamente similares a como se conocen en occidente, a pesar de nunca dejar clara la ubicacion exacta de la protagonista, es importante ampliar el margen de publico objetivo para que muchos tipos de personas se puedan sentir identi cadas con las variopintos pensamientos, de una simple adolescente en occidente. Respecto a la cultura popular la narrativa contada esta llena de referencias, comparaciones y explicaciones de la historia en base a este tipo de conceptos conocidos por una gran parte de la poblacion, as es, aun mas directo, el acercamiento al jugador.

  En segundo lugar, se da la **religión**. Puede ir muy a la par de la cultura, porque al igual que pasa en nuestra sociedad, la situacion social y gubernamental actual depende de la cultura y religion que se ha desarrollado a lo largo de los siglos. Los principios inculcados de la religion y, las costumbres de las personas que habitan en el mundo, se puede a rmar, que la situacion de la sociedad es consecuencia directa del desarrollo de la cultura y religion en dicho lugar a lo largo del tiempo. *La religion existente en esta historia, bebe y procede de Estados que actualemnte son laicos. A pesar de esto, la religion en estos lugares, por el desarrollo a lo largo de los tiempos y la inculcacion generacional de la misma, para muchas personas sigue estando presente. Cada persona es libre de seguir la fe la que considere oportuna, pero es innegable la importancia e impacto de la religion catolica en Occidente. Como se ha explicado anteriormente, la religion va a la par con la cultura en muchas ocasiones, pero no tiene porque ser una norma.*

  En tercer lugar, se da la **sociedad**. Es consecuencia de su cultura, el como se

comportan las personas habitantes de este mundo, como se relacionan entre s , hasta donde han llegado los derechos individuales y globales de los seres de este mundo. Vienen arraigadas de su cultura, y tambien de la religion. Como se ha explicado anteriormente, la cultura y la religion son conceptos que en segun que tipo de mundos se desarrolle (si tiene un enfoque laico o mas religioso), la religion tendra mas o menos impacto dependiendo del enfoque.

Es complejo hablar de algo tan extenso como la sociedad y todo lo que engloba, no solo de la que rodea al jugador en su d a a d a, sino re ejar adecuadamente esa sociedad, esa cercan a que el pueda identi car, llevarla a un mundo cticio sin que sea algo copiado, a pesar de que se quiera expresar una cotidianidad y familiaridad en todos los aspectos del entorno. Sigue siendo una historia cticia, con re ejos historias, comportamientos y narrativas max exagerados de los que pueden llegar a ser en el mundo real. No se puede contar algo de la misma forma en la que ocurre en el mundo real, esa es la clave para un buen re ejo, no identico de un elemento de world-building. En el caso tan extenso de la sociedad, aun que las historias, sociedades y narrativas contadas en los libros, videojuegos y pel culas, el oyente o espectador siempre trata de buscar una similitud al mundo real, traerlo a algo que pueda entender. Explicado de forma algo burda; llevarlo a su propio terreno. Aunque se cuente una historia de una joven llamada Elisa, el jugador instintivamente buscara aspectos en los que identi carse, y la sociedad, los comportamientos humanos, las re exiones y temas que siempre han asustado, divertido y hecho re exionar al ser humano, son conceptos necesarios, para poder representar una sociedad no solo interesante, sino cre ble, que el jugador pueda vivir y entender lo que esta ocurriendo. Eso es lo que se tiene que conseguir tras partir de un pequeno acercamiento con la cultura y la religion, la sociedad es el tercer gran pilar.

Por ultimo, se da el **gobierno**. Es el resultado nal de la union y desarrollo de estos tres ultimos conceptos. El gobierno y sus distintas leyes establecidas son un re ejo de lo que ha sido y es: la cultura, la sociedad y religion a lo largo de los ultimos tiempos en ese lugar, independientemente de que tenga mas o menos impactos de estas tres.

Por ultimo, todo estos aspectos previamente explicados, sucumben a todas y cada una de las leyes establecidas en el mundo que se ha creado, ya sea la existencia de un solo gobierno, de varios y como se relacionan entre ellos.

- **Geografía**: de forma simple, esto se re ere el sitio en el que se encuentra el jugador. No tiene porque conocerse la ubicacion exacta, pero si las carac-

ter sticas de su entorno (un sitio rural, un sitio muy poblado, una ciudad futurista, los diferentes tipos de escenarios, desiertos, bosques, lagos, r os, playas, etc...).

Respecto a la geograf a, y el alcance de la importancia que esta tiene en el desarrollo del world-building para esta historia, con que el jugador sepa identi car que la protagonista vive en una ciudad rural, pequeña y familiar. Ser a mas que su ciente para que el jugador se vea absorbido en su atmosfera. Algo muy similar a lo que ocurre en el reconocido videojuego: *Undertale* [9].

- **Cosmología y/o mundo físico**: como su propio nombre indica, estos conceptos englobal todas aquellas reglas, principios y estamentos f sicos que rigen el curso de planeta o sistema en el que se desarrollan los hechos. Todas estas normas van desde el planeta o sistema planetario existente en la narrativa que se desarrolla, hasta pequeños detalles como la fauna, ora, y en normales generales, seres existentes en dicho lugar.

  Este aspecto es el mas simple de todos para la historia, no por ello es menos importante, ya que es necesario que el jugador identi que rapidamente que el planeta y leyes f sicas que existen en el mundo que esta viviendo, son similares a las que esta acostumbradas, aun que con leves diferencias que buscan desencajar al jugador (desde personajes pintorescos, animales que hablan, animales con formas humanoides, etc...).

- **Género**: El genero de la historia contada del videojuego, estos generos son muy similares a los cinematogra cos:

  1. Accion
  2. Aventura
  3. Catastrofe
  4. Ciencia Ficcion
  5. Comedia
  6. Drama
  7. Fantas a
  8. Suspense
  9. Terror

  Como se ha dejado entre ver a lo largo de las distintas explicaciones, acerca del mundo desarrollado, y su world-building, el jugador identi ca rapidamente generos como la comedia, la accion, la aventura y el drama en la historia de la joven Elisa.

  Existen muchos mas tipos de generos, pero, estos son los mas comunes [8].

- **Lenguaje y comunicación:** Tambien es necesario de nir cual es lenguaje, la forma de comunicarse que tienen los personajes que se desarrollan y comunican entre s , sea ya existente o inventado.
- **Educación:** Pueden ir desde educacion formal de conocimientos del mundo y la sociedad, como pasa en nuestra vida cotidiana, hasta un mundo que se rige por normas de la magia, y por lo tanto su educacion esta altamente ligada a ella.
- **Economía y comercio:** estos conceptos engloban como se relacionan los pueblos, naciones o aldeas que existen en el mundo desarrollado, y el tipo de econom a usada (algun tipo de moneda, trueque, etc...).

  Por ultimo, el lenguaje, comunicacion, educacion y econom a del world-building, a pesar de ser aspectos bastante diferenciables entre s , para ser entendidos por el jugador, y facilmente identi cables se podra observar a lo largo del juego, a traves de ejemplos f sicos (moneda usable en el juego, tiendas para comprar diferentes objetos, no puedes robarlos, porque va en contra de las normas, etc...). Respecto al lenguaje y comunicacion desde el temprano comienzo del juego, el jugador puede identi car rapidamente la forma natural de comunicacion y lenguaje a la que se esta acostumbrado, mediante la exposicion de diferentes dialogos y monologos de los propios personajes no jugables, del juego. Por otra parte, la educacion es claramente identi cable a lo largo del juego, al ver diferentes instituciones educativas y niveles de formacion de los distintos personajes, que no solo puede averigar el jugador observandolas, sino, tambien hablando con distintos personajes.

  De esta manera, queda explicada la relevancia del desarrollo de un buen world-building, para sumergir al jugador en una historia con tres pilares fundamentables: **Creíble**, **Realista** e **Inmersiva**

3.2. Diseño de juego y sistemas

Una vez se consigue un contexto narrativo, se pueden nombrar el motivo y objetivo del sistema, se puede empezar a diseñarlo, antes de su implementacion tecnica, de las partes mas extensas y complejas del desarrollo de un videojuego. Teniendo este aspecto en cuenta, hay que tener en consideracion que hay una serie de aspectos que incluyen un videojuego, y en base a esos aspectos, se puede hacer un breve desglose del que partir, para ir enumerando los elementos mas importantes de un juego respecto al *Game Design* existentes [7]:

1. **Actividad**

2. **Reglas**
3. **Conflicto**
4. **Metas**
5. **Toma de decisiones**
6. **Libertad**
7. **Sistema**

Gracias a estos elementos, y focalizando en el genero y estilo del videojuego en cuestion (*RPG*, **Por turnos**), se pueden identi car facilmente cuales son todos los elementos formales de un videojuego, y como se ven re ejado en el gameplay [7]:

1. **Jugador (Uno o varios):** Este es el elemento mas clasico en cualquier sistema de juego, que viene arraigado desde los clasicos juegos de mesa. Numero de jugadores, si es individual o cooperativo, si interctuan los jugadores entre ellos, y de que manera, si el multijugador es local u online, etc...

   Se ve claramente por el tipo de inquietudes que giran alrededor del jugador, que son elementos que no tienen origen en el videojuego como ya se ha indicado. Por lo tanto, es sencillo y facil de gestionar este aspecto del *Game Design*, tan solo teniendo en cuenta los aspectos anteriores.

   En este proyecto, al ser un videojuego RPG por turnos, es de un solo jugador real, jugandose asi de forma individual, con un tipo de combate uno contra uno contra un enemigo que tiene disenada su inteligencia arti cial a traves de un **SU**, *Head to Head*, y uno contra varios, *One-against-to-many* [7].

   En de nitiva, su estilo de gestion de jugadores sigue la l nea de los clasicos **JRPGS** (Japanese role-playing game), como la famosa saga de videojuegos *EarthBound* [10], o *Pokemon* [11].

2. **Metas:** los objetivos del jugador a lo largo del videojuego, son los pasos que tiene que perseguir el jugador para completar y/o avanzar del juego, y completar esos pasos van proporcionando al jugador recompensas que le permiten seguir avanzando, de esta manera, retroalimentando al jugador.

   Dependiendo de cada juego puede ser completamente distinto, desde los clasicos juegos de mesa:

   - **Trivial**: contestar preguntas (*pasos/metas*), y con ellas se obtienen quesos (*recompensas*).

- **Civilization VI**: Controlar territorio poco a poco, es el *objetivo* del jugador, y ampliar su territorio a mayor capacidad, y mayor cantidad de recursos, esta sera su *recompensa*
- **Sims**: el analisis de objetivos y recompensas que tiene un videojuego se puede volver mas ambiguo, dependiendo del estilo de juego y de sus propias reglas, un claro ejemplo son *Los Sims*, *El simulador de vida* [7].

  Al n y al cabo los simuladores son complejos de analizar, no tiene porque haber objetivos jos, ni recompensas inamovibles. Los simuladores nacen de tratar de llevar al jugador la experiencia mas realista posible y en este caso, esta dinamica objetivo/recompensa, se queda mas dispersa.

  A pesar de esto, la mejor forma de de nir a Los Sims es acercandose a su signi cado de simulacion de vida, en el que tu objetivo principal es construir, obtener cada vez mas posesiones, objetos, relaciones, llevar al personaje al maximo nivel posible de equipables, recursos y objetos de todo tipo. Y esa propia satisfaccion y sensacion de avance es lo que consigue recompensar, y a la par, enganchar al jugador.
- Acercandose mas al proyecto trabajado, otro ejemplo de analisis de objetivos y recompensas son los *JRPG*, el jugador tiene uno o varios personajes bajo su control, en los que sus objetivos principales son avanzar en su aventura, y a su vez vencer los distintos enemigos que se va encontrando, y la recompensa es ir ampliando las estad sticas del jugador, y posibilidades en todos sus aspectos, y muchos mas dependiendo de cada juego. Como objetos, vida y ataques.
  En este proyecto, los objetivos y recompensas son muy similares a estos propiamente dichos, solo que las recompensas var an un poco. En este proyecto estas estad sticas mejoras que se van obteniendo son:
  a) **SIP** (Puntos de Integracion Social, similar a la barra vida).
  b) **Ansiedad** (Similar a una barra de **debuff** y/o desventaja).
  c) **Objetos**.
  d) **Gritos** (Ataques).
  e) **Hablar** (Capacidad comunicativa del jugador para envaucarlo y/o proprocionar desventajas al enemigo).
  f) **Llorar** (Curacion)

3. **Reglas (mecánicas)**: en este apartado se busca identi car las mecanicas del videojuego expuesto, pero primero, es necesario explicar que es una mecanica.

Son todas y cada una de las reglas creadas por un disenador o creador unico del juego, que delimita al jugador de alguna u otra manera. Por ejemplo, si en las mecanicas del juego se contempla que el jugador pueda saltar, puede hacerlo [7]. Un ejemplo clasico y sencillo para entenderlo ser a:

- **Super Mario Bros:**
  a) **Perspectiva 2D:** La perspectiva del jugador
  b) **Desplazamiento lateral y salto:** izquierda, derecha y salto hacia arriba. El jugador solo puede moverse en dichas direcciones.
  c) **Muerte del jugador delimitada por contacto con enemigos o caídas al vacío.**
  d) **Existencia de consumición de diferentes *power-ups* que otorgan habilidades.**

En base a estas reglas, el jugador tendra que elaborar una serie de estrategias (*dinamicas, de las que se hablara mas adelante*), para superar cada nivel *(objetivo)*. En este proyecto, las diferentes reglas vienen delimitadas principalmente por dos situaciones distintas:

a) **El jugador fuera del combate**: Esta es la mas sencilla pues solo dispone de reglas basicas de **movimiento e interacción con su entorno.**
b) **El jugador dentro del combate:** Aqu el jugador se ve delimitado por las siguientes normas:
  1) El jugador puede gritar (**ataques**).
  2) El jugador puede hablar (**engañar al enemigo, añadirle ansiedad**).
  3) El jugador puede llorar (**curaciones**).
  4) **Condición de victoria**: si los **SIP** del enemigo bajan a 0, el jugador ha ganado el combate.
  5) **Condición de derrota**: si los **SIP** del jugador bajan a 0, el jugador ha perdido el combate.

4. **Reacción del jugador ante las mecánicas (dinámicas):** lo que sigue a las mecanicas, son las dinamicas. Son las estrategias que tiene que pensar y elaborar el jugador para poder conseguir sus objetivos.

   Al tratarse de una serie de estrategias, pero no reglas jas y estrictas, cada ejemplo puede ser valido si esas estrategias se elaboran pensando en cumplir esos objetivos para seguir avanzando. Cabe recordar que estan limitadas por las *mecanicas* previamente establecidas.

   Sabiendo las mecanicas del **Super Mario**, el jugador, puede elaborar una serie de posibles estrategias para superar los niveles:

- Saltar o moverse, para seguir avanzando en el nivel. De esta manera se puede desde esquivar los enemigos para completar el nivel, o derrotarlos directamente. Tambien se puede conseguir ciertos *power-ups*, para aumentar las posibilidades de victoria. Al tener mas fuerza o vida, el jugador podra avanzar entre los enemigos con mayor facilidad. Todo lo que sea usar estas reglas de nidas por el juego, y sacar el maximo bene cio para avanzar en el juego, son las llamadas *dinamicas*.

En este proyecto, siguiendo las reglas establecidas por el mismo, y por otros muchos clasicos JRPG, son dinamicas establecidas para garantizar la supervivencia del jugador en cada uno de los combates. Esto abarca multiples *dinamicas*:

- Si tu objetivo no es derrotar al enemigo lo antes posible, sino reducirlo para poder derrotarlo mas facilmente, las opciones mas logicas que ha de realizar el jugador son hablar con el enemigo, y llorar para mantener el maximo de vida posible. *Hablar y llorar* son mecanicas internas del videojuego presentado, que corresponden a asignar desventajas que permitan al jugador derrotarlo de forma mas segura. Esto no siempre es una estrategia aceptable, enemigos que se han de derrotar lo antes posible, por su baja *SIP* o su gran potencia en sus ataques, entre otras cosas.
- Otra posibilidad es derrotar al enemigo lo mas rapido posible, es el ejemplo claro en que el enemigo tiene que realizar la accion de *gritar*, sin descuidar sus puntos de *SIP*, para no ser derrotado.

En de nitiva, cualquier secuencia de gritos, hablar o llorar, gestionado de tal manera que se conserven en una cantidad superior a 0, y se pueda derrotar satisfactoriamente al enemigo, son las *dinamicas* apropiadas.

5. **Percepción final del jugador (estéticas)**: en de nitiva, en el esquema, mecanicas, dinamicas y esteticas. Las esteticas son la parte mas ambigua y dispersa de este esquema. Son aquellas sensaciones e impresiones que recibe el jugador tras percibir la experiencia de juego: *mecanicas-dinamicas*.

   Por lo tanto dependen mucho de cada jugador. La sensacion buscada en el jugador principalmente es la inmersion, credibilidad y realismo en la experiencia de juego.

6. **Recursos y su gestión:** todos los aspectos que rodean a los terminos como **recursos** y **gestión**, son aquellos que estan bajo el control del jugador [7]. Dependiendo del tipo y estilo de juego, var an mucho:

   - Juegos de estrategia o control, (**Civilization VI**): Territorios, moneda disponible, numero de tropas, etc...

- Juegos de simulacion, (**Sims**): cantidad en todas las diferentes estad sticas, numero de propiedades, de objetos equipables o no, etc...
- Juegos JRPG, por turnos, trayendolo a este caso concreto, ser an:
  a) Cantidad de habilidades jugables por turnos (**Gritos, hablar, y lloros**).
  b) Cantidad de objetos equipables.
  c) Cantidad de objetos usables durante un turno.
  d) Estad sticas de SIP y Ansiedad.
  e) Moneda usable del juego.

7. **Estado de juego**: este es un aspecto, que a lo largo de la evolucion de los videojuegos, puede a llegar a perder el signi cado, ya que procede de juegos de mesa tan clasicos como el ajedrez. Por ejemplo, haciendo referencia a este ultimo, se re ere a todos los posibles estados que tiene un tablon de ajedrez, y en videojuegos, ser an todos los multiples estados en los que se puede encontrar el jugador [7], como se puede observar en la Figura 3.1

Figura 3.1: Estados de juego

Esto con el desarrollo y avances de las nuevas tecnolog as que han venido de la mano de los videojuegos, ha alcanzado un nivel de estados demasiado altos como para de nirlos de forma sencilla. Por ejemplo, cuantos estados de juego tiene un tipo de juego como *rogue-like*, si cada partida es distinta. Aunque a d a de hoy de nir el estado del videojuego puede llegar a ser complejo se puede reducir estos estados a elementos mas sencillos del juego:

- Cantidad de vida actual, en todos sus aspectos posibles.

- Cantidad de habilidades aprendidas actualmente.
- Cantidad de objetos equipables actualmente.
- Cantidad de dinero equipable actualmente.

Y esto es extrapolable a cualquier tipo de videojuego, en lo que en este caso respecta:

a) Cantidad de SIP actual, del jugador y del enemigo.

b) Cantidad de Ansiedad actual, del jugador y del enemigo.

c) Cantidad de objetos equipables.

d) Cantidad de habilidades posibles que usar en el estado actual.

8. **Orden y secuencialidad de los hechos:** este apartado dictamina, la secuencia de acciones que puede tomar durante el juego, tanto el numero de acciones que puede tomar simultaneamente, como el orden de ellas.

   En videojuegos en que el orden esta mucho mas ordenado y secuenciado es facil de exponer. Por ejemplo, en clasicos **JRPG** por turnos, y en general, RPGs, estan muy separados la diferenciacion entre las acciones que puede hacer cada jugador a lo largo de un turno (desde juegos de cartas, hasta juegos de combate RPG, etc...).

   En de nitiva, cualquier videojuego que pueda plantear turnos, suele funcionar de esta manera. En el caso de este proyecto, un turno del jugador, puede realizar:

   a) Realizar un maximo de una accion ofensiva por turno, es decir, cualquier accion que contemple: gritar, hablar o llorar. Una vez el jugador ha realizado esa accion, se pasara al turno del jugador.

   b) En cuanto al consumo de objetos, no son in nitos (como las acciones), pero si puede usar cuantos quiera y disponga el jugador a lo largo de un turno. Dicho turno solo nalizara al realizar una accion ofensiva, nunca despues del consumo de un objeto.

9. **Interacción del jugador:** Por otro lado, el concepto de *interaccion del jugador*, es bastante amplio y muy diferenciable en varias posibles vertientes. Antes se resum a en **interacción de los jugadores con otros jugadores**, pero a d a de hoy podemos diferenciar entre:

   a) **Interacción jugador-jugador**: Para este proyecto, no es relevante, ya que es un apartado que de ne como interactuan los jugadores en entornos en los que exista al menos mas de un jugador.

b) **Interacción jugador-NPC:** este apartado cobra mas importancia, porque en juegos de combate RPG por turnos, es muy importante la interaccion existente entre un jugador y un *NPC*, o cualquier tipo de IA en general, no solo en el combate, sino a lo largo de toda la aventura, por ejemplo:

1) **Interacción con un NPC, fuera del combate**: A lo largo de este de tipo de juego, es muy comun que el jugador pueda interaccionar y conversar con muchos personajes variados. El objetivo de esto, a parte de motivos narrativos, son posibles pistas y ayudas que proporcionan estos personajes al jugador.
2) **Interacción con un NPC, dentro del combate**: Por otro lado, esta la interaccion con el jugador que puede tener con el enemigo que pelea contra el.

c) **Interacción jugador-entorno**: Este tercer y utlimo apartado, quizas es el mas ambiguo, porque puede englobar cualquier tipo de interaccion que tiene el jugador con su entorno.

Una vez se tiene planteada y diseñada la narrativa, el diseño del juego y sus sistemas, se puede pasar al diseño de personajes, que podr a considerarse una parte mas profunda del diseño, que puede ir tambien de mano con la narrativa, pero el orden de ejecucion ser a:

Narrativa-Game Design-Characters Design-Level and Maps Design

3.3. Diseño de enemigos y balanceo

El diseño de enemigos no solamente abarca el diseño visual de los personajes, que son elementos art sticos en los que no se va a profundizar a lo largo de este proyecto, sino a nivel tecnico, que tipos de enemigos hay, y que implicacion tienen en el combate.

De cierta manera, el diseño de enemigos es parte del *Game Design*, pero como abarca muchos aspectos, se ha decidido hacer una pequeña separacion de las explicaciones previas del funcionamiento del combate para una mejor comprension del mismo. Tambien es necesario hacer esta separacion para hacer hincapie en el *Game Balance* [7] del juego, en el que es necesario que exista varios tipos de enemigos con diferenciaciones y funcionamientos diferenciables para crear un sistema equilibrado en la relacion *jugador-enemigo*.

Los aspectos a tener en cuenta para añadir variedad y diferenciacion en los enemigos en un videojuego son:

1. **Apariencia distinta:** Como se ha hablado, la parte visual es importante, pero no es su ciente para añadir variedad al comportamiento y jugabilidad de un determinado enemigo.

2. **Patrones y estilos:** A lo largo de la historia, en los videojuegos en los que han existido enemigos con los que el jugador tiene que enfrentarse, estos han tenido una serie de reacciones jas ante las circustancias que giran alrededor de su entorno. Lo que comunmente es comonocido como *patrones* [12], pero que realmente siempre han sido IAs diseñadas mediante *FSM*. A pesar de que esta tecnolog a sigue siendo usada, para un comportamiento mas enriquecido (como se ha hablado en apartados anteriores), han ido apareciendo mejores formas de representar el comportamiento de una IA, de forma mas realista, como el propio *Sistema de utilidad* que se ha diseñado y desarrollado para este proyecto.

   Asi que, para el videojuego desarrollado, como tal la IA de combate, no tiene un patron especi co establecido, sino que, gracias al desarrollo del sistema de utilidad, el enemigo decidira cual es la mejor accion que puede realizar durante ese turno teniendo en cuenta una serie de variables, que en este caso son:

   a) SIP

   b) Ansiedad

   c) Rol del enemigo

   Como el SIP y la ansiedad, ya han sido explicadas, se va a pasar a explicar que son los Roles del enemigo, que es exactamente.

3. **Roles (JRPG):** Los roles de los enemigos en cualquier juegos de combate y/o de rol en general, especialmente en los JRPG, siempre han existido una serie de roles que han cumplido, para as crear un sistema equilibrado y variado, con el n de que el juego sea mas divertido y diferente para el jugador, y tenga que ir adaptando su estrategia en cada tipo de combate, existen tipos muy variados y espec cos, personajes ofensivos, defensivos, sanadores, hibridos, etc... En este proyecto se ha seguido las bases originales de muchos videojuegos, pero adaptadas al mismo, para seguir de cerca el objetivo previamente planteado; **creíble, realista y cercano.** Estos roles son:

   a) **Altruista**: Posee un rol mas curativo, priorizando el bienestar antes que atacar, es decir, priorizara **llorar**.

   b) **Egocéntrico**: Posee un rol mas ofensivo, priorizara sobrevivir y eliminar su contricante, antes que otras opciones, es decir prioridad la accion de **gritar**.

c) **Empático**: Posee un rol mas defensivo, priorizara dejar en desventaja a su enemigo para poder abarcar mejor el combate, es decir priorizara la accion de **hablar**.

3.4. Diseño de niveles

Tras explicar el funcionamiento del diseño de juego y su sistema, se procede a explicar el diseño de niveles y mapas. Es la parte nal para poder tener el diseño del entorno de nido, sobre el que implementar el desarrollo tecnico del sistema del combate, su IA, y por lo tanto el realismo y credibilidad que se busca a la hora de llegar a una inmersion plena en el videojuego para el jugador.

En de nitiva, es necesario establecer el diseño del sistema, las mecanicas y las dinamicas, para poder establecer niveles en el juego, ya que estos, tienen que cumplir esas reglas previamente diseñadas. No tiene sentido pensar en *Level Design*, sin un *Game Design* de nido.

El *Level Design*, es la **creación del entorno de esas reglas del juego, escenarios, lugares, misiones, etc...**, por las que el jugador ha de pasar. El paso por estos lugares o misiones se ven sujetos a un objetivo general. Cualquier diseño de juego de una parte del gameplay, va sujeta a cumplir un objetivo principal, y todos esos niveles, lugares o misiones que ha de superar, tienen un determinado objetivo de existencia, que ofrece al jugador una recompensa, que acercara al jugador cada vez mas a superar ese objetivo principal y nalmente, que ofrece el juego [13]. Como esto es algo abstracto de entender, se va a exponer un caso de ejemplo de un famoso videojuego y su diseño de niveles, *Pokemon*. Se va a separar el diseño de Pokemon en tres pequeños apartados para poder entenderlo:

1. **Game Design:** para que existan una serie de objetivos en un videojuego tiene que existir previamente una parte crucial del *Game Design*. Las mecanicas en el caso de Pokemon son:

   a) **Condición de derrota:** si los pokemons que tiene disponible el entrenador se debilitan, el jugador habra perdido el combate.

   b) **Condición de victoria:** Si el jugador logra debilitar a todos los enemigos del entrenador contricante, habra ganado el combate.

   c) **Reglas principales del combate**: El jugador y el enemigo solo podran realizar una accion por turno.

   En base a estas reglas, se puede concebir una serie de objetivos:

2. Objetivo principal: En Pokemon lo principal, es lograr superar al entrenador más fuerte de la región, es decir, vencer al líder de la Liga Pokemon.

3. Pequeños objetivos: Dado el objetivo final, el jugador irá completando pequeños objetivos, que le ofrecerán recompensas. Estas recompensas podrán ser obtenidas superando los gimnasios pokemon a lo largo de la aventura. Por lo general, para poder enfrentar al líder de la liga pokemon, el jugador se verá obligado a superar los ocho gimnasios Pokemon reconocidos en la región. De esta manera, cada gimnasio superado es ese pequeño objetivo que superar y con su respectiva recompensa.

De esta manera se ha construido un diseño de niveles completo, por el que el jugador podrá ir avanzando uno por uno, a lo largo de todo el juego. Con este ejemplo práctico, ahora se puede desglosar de forma más sencilla el caso para este proyecto:

1. Game Design (mecánicas): Previamente explicadas.

2. Objetivo principal: es a priori narrativo, pero en seguida el jugador se verá sorprendido por las mecánicas previamente explicadas antes de salir, de tal manera que los acontecimientos que podrá experimentar el jugador de primera mano serán:

   a) Prólogo previo y contexto narrativo
   b) Encontronazo con un personaje, de tal manera que se usa como excusa para presentar las mecánicas
   c) Salida y propuesta del objetivo final, que viene de la mano con la salida y avance del jugador para seguir con la aventura, protagonizada por Elisa.

3. Pequeños objetivos: Elisa, tendrá que ir recorriendo su ciudad natal para llegar a cada rincón donde se encuentran, todos y cada uno de los enemigos a los que necesita ver. Cada uno es especialista en uno de los sentimientos que necesita expresar. Los amigos de Elisa la conocen mejor que nadie, por lo tanto cada uno de ellos sabrá llevar su:

   a) Ira
   b) Ansiedad
   c) Tristeza
   d) Miedo
   e) Euforia
   f) Asco

Llegar a cada uno de los amigos de Elisa, y superar estas emociones dispersas e incomprendidas, serán los pequeños objetivos del jugador, que a su vez conllevarán una serie de pequeñas recompensas (objetos, gritos, habla, etc...).

3.5. Diseño de mapas

Planteados en su totalidad el GameDesign, y Level Design, falta por último el diseño de mapas. Se trata del aporte visual y gestión mapeada del diseño de niveles, ya que estos objetivos tienen que ir de la mano de la creación de pequeños lugares y escenarios, como se trata de una parte más visual, se plantea con un breve esquema, que es el siguiente:

1. Escenario 1, Casa de Elisa: se trata del escenario inicial del videojuego, donde se presentará a al jugador su cometido, y las mecánicas principales del juego

2. Escenario 2, Barrio de Elisa: En este punto el jugador sabrá desenvolverse en las mecánicas del juego, pero a penas habrá tenido oportunidad de desarrollar dinámicas de forma adecuada, este escenario sirve al jugador como puente a dominar el juego.

3. Escenario 3, Barrio céntrico de la ciudad Natal de Elisa: se trata del escenario final, en este punto del videojuego el jugador haber completado multiples obstáculos y saber de una forma más adecuada como desenvolverse en él.

3.6. Diseño del funcionamiento del Sistema de Utilidad del enemigo

Una vez explicado, como funcionan los turnos durante el combate, que puede hacer el jugador y de que manera, y que puede hacer el enemigo, se ha de explicar, de que manera está diseñada la IA del enemigo, diseñada a través de SU. Un pequeño vistazo a las posibles acciones, y la importancia de ellas dependiendo de ciertas variables, se explica con la siguiente Figura 3.2:

Ahora que se tiene conocimiento de todas las variables que tienen importancia en el resultado de la acción decidida en el SU (Ansiedad, roles, SIP). Se va a proceder a explicar, el contenido técnico de este sistema. Factores de decisión:

Figura 3.2: Esquema US

1. Rol: el tipo de rol del enemigo determina sumar, más a o menos peso a los factores de fusión, todos ellos se hacen por una función constante

$$y = a \tag{3.1}$$

donde y es el resultado, a es la asignación de la constante asignada por el diseñador.

2. Ansiedad : la cantidad de la ansiedad determina, sumar, más o menos peso a los factores de fusión, en este caso no solamente hay una función ja para los diferentes factores, sino que es variable, dependiendo del factor de fusión. Para el factor de fusión ganas de llorar la función es la siguiente:

$$y = x^e(\text{creciente}) \tag{3.2}$$

donde y es el resultado de la función, y x^e es la función asignada por el diseñador, que en este caso es creciente, es decir, que el resultado de la y crecerá de forma directamente proporcional a la x, de tal manera, que cuanto más alta sea la x (ansiedad), más alto será el peso asociado a ganas de llorar .

Para el factor de fusión ganas de gritar la función establecida es:

$$y = a^x(\text{decreciente}) \tag{3.3}$$

De tal manera, que a es una constante de numero relativamente bajo, pre-establecida pro el diseñador, y x la ansiedad, de esta manera, cuanto más alta sea la ansiedad, menos será el peso asociado a las ganas de gritar y viceversa.

3. SIP: la cantidad de SIP determina, sumar, más o menos peso a los factores de fusión, en este caso como en la ansiedad, se dan dos funciones distintas, para generar un peso a dos factores de fusión distintos. Para el factor de fusión ganas de gritar la función es:

$$y = x(\text{lineal}) \tag{3.4}$$

Es la más sencilla de entender. Junto a las constantes, simplemente la cantidad de SIP (normalizada), será el peso asociado a las ganas de gritar. Por otro lado, la función asociada a las ganas de hablar es:

$$y = a^x(\text{creciente})(\text{exponencial}) \tag{3.5}$$

Es similar a la función creciente explicada previamente, pero en este caso está diseñada de tal manera que el resultado de x a (cantidad de SIP), dará como resultado un crecimiento exponencial en y a (peso asociado a ganas de hablar), siendo a, una constante previamente preestablecida por

el diseñador.

Tras analizar los factores de decisión, sus respectivas funciones matemáticas que dan como el resultado diferentes pesos asociados a distintos factores de fusión. La acción final decidida por el SU se obtendrá sumando los pesos de los diferentes factores de fusión, y el factor con un peso asociado más alto será la acción elegida, por ejemplo:

$$sumTotal(ganas\ de\ gritar) = 0,4 \tag{3.6}$$

$$sumTotal(ganas\ de\ hablar) = 0,6 \tag{3.7}$$

$$sumTotal(ganas\ de\ hablar) = 0,9 \tag{3.8}$$

La acción elegida por el SU será hablar.

4

Desarrollo e implementacion tecnica

Una vez nalizados todos los aspectos de diseno previo, que garantizan un entorno rico y completo, donde implementar los aspectos tecnicos del videojuego, se puede dar paso a la explicacion tecnica, la cual en este videojuego, tiene los siguientes apartados, para su completo entendimiento:

1. Desarrollo e implementacion basica: En la Seccion 4.1 se explica todos los aspectos de implementacion tecnica basica, que necesita el videojuego previamente, antes de pasar a aspectos mas avanzados, estos terminos basicos son:

   a) Movilidad del jugador

   b) Interaccion con el entorno

   c) Gestion de recursos

2. Desarrollo e implementacion de escenarios: En la Seccion 4.2 se explica aquellos aspectos relacionados con la implementacion tecnica de escenarios, ciertas como por ejemplo; como avanzar entre escenarios, interacciones entre ellos, elementos importanntes, etc...

3. Desarrollo e implementacion de sistema de combate: En la Seccion 4.3 se explicara todas las partes del codigo en el que actua el sistema de combate, como se relacionan los scripts entre si para su correcto funcionamiento, el sistema de turnos, etc...

4. Desarrollo e implementacion del Sistema de Utilidad: Por ultimo, se explica en la Seccion 4.4 la parte mas importante de la parte tecnica, el funcionamiento de la IA del combate, y como se ha planteado para seguir el diseño previamente explicado.

4.1. Desarrollo e implementacion basica

Empezando por lo que es la implementacion mas basica del proyecto, se va a explicar en primer lugar, como se mueve el jugador (respecto al tipo de movimiento diseñado previamente explicado en apartados anteriores).

Como interactua con su entorno (de que manera puede interactuar, con todos los aspectos del entorno que el videojuego permite interaccion directa con el jugador, como se explico en apartados anteriores).

Por ultimo como se gestionan los recursos del jugador (el uso del inventario del jugador, y el consumo de sus objetos). Se va a explicar la movilidad del jugador, la interaccion que tiene con el entorno, y la gestion de sus recursos en la Figura 4.1. El desarrollo e implementacion tecnica, del movimiento del jugador en todo su entorno y la interaccion con este, contempla los siguientes scripts en la Figura 4.1:

Una descripcion del funcionamiento de este sistema, sera el siguiente:

1. movementPlayer: este script es el mas importante de este sistema de movimiento e interaccion del jugador, porque de forma directa o indirecta, esta comunicandose con el resto de scripts .

   Por un lado, gestiona el movimiento basico del jugador que permite su movimiento libremente por los escenarios, tambien gestiona la interaccion con la UI (fuera del combate, HUDSystemNotBattle), es decir, la reaccion que tiene el juego mediante la interfaz de todas esas interacciones que realiza el jugador con su entorno fuera del combate.

   Por otro lado, se comunica con el script de inventory para la gestion de uso de objetos y creacion de objetos.

   Por ultimo, se comunica con GameManager , que es un script que se dedica principalmente a la gestion de las estadsticas numericas del jugador (SIP Maximo, SIP actual, ansiedad maxima, ansiedad actual, etc...).

2. GameManager: es el script encargado de gestionar las estadsticas del jugador, para poder comunicarlas de esta manera movementPlayer y

Figura 4.1: Esquema Scripts de Interacción básica

que estescript se lo comunique aHUDSystemNotBattle para tener una representacion visual del estado del jugador.

3. Inventory: el script del inventario sirve como intermediario entre el jugador (movementPlayer), y los objetos que consumeitemClass), de tal manera que los objetos se asignan o se borran del inventario a traves de itemClass , que son los objetos en s para que el jugador pueda comunicarse con el inventario que sera a traves de este, con el que el jugador consumira los objetos que quiera.

4. HUDSystemNotBattle: este script gestiona la interfaz de usuario del jugador mientras no este en el combate, de tal manera que re eja mediante una interfaz los cambios que ocurren en el inventario, y en el propio jugador.

5. itemClass: estescript contiene una serie de de niciones que tienen todos los objetos (ventajas que proporcionan), que se comunican dorventory para poder crearlos o borrarlos.

6. cameraFollow2D: Porultimo, este script , gestiona la camara del juego, para que siga al jugador.

4.2. Desarrollo e implementacion de escenarios

En este apartado se procede a explicar la implementacion tecnica de los escenarios, es decir, su gestion mediante escenas de Unity y como el jugador puede moverse entre ellas. Para ello el script del jugador que se comunica con los escenarios, y objetos del mismo tambien esmovementPlayer , pero se ha decidido separarlo del anterior esquema para una mayor comprension.

En la Figura 4.1 se ayuda a entender esta implementacion de transiciones y escenarios es:

Figura 4.2: Esquema Scripts de transicion entre escenas

Como se observa en la figura, la transición del jugador entre las diferentes escenas se gestiona a través de la comunicación bidireccional del movementPlayer (script previamente explicado) y el SceneManager (Clase proporcionada por UnityEngine) [14].

1. movementPlayer: en la parte de transiciones entre escenas, se encarga de gestionar todos y cada uno de los GameObjects que funcionan como trigger, para poder transicionar entre las distintas escenas del juego, al darse una interacción directa entre esos triggers y el jugador, podría cambiar de escena.

2. SceneManager: tiene una serie de métodos encargados de identificar escenas en activas actualmente, métodos que se encargan de pasar de una escena a otra, etc...

En resumen movementPlayer gestiona los triggers y objetos de la escena activa con los que el jugador necesita interactuar actualmente para cambiar de una escena a otra, y SceneManager gestiona las transiciones entre escenas.

4.3. Desarrollo e implementación de sistema de combate

En este apartado se procede a explicar el desarrollo e implementación técnica del funcionamiento del sistema de combate, de que manera se gestionan los turnos entre jugador-enemigo y que acciones pueden realizar cada uno de ellos, esto se puede visualizar en la siguiente Figura 4.3. Este sistema es el encargado de la gestión de turnos en el sistema de combate implementado, un breve desglose del funcionamiento de los distintos scripts:

1. BattleManager: es el intermediario entre el resto de scripts para la gestión de turnos, se ha desarrollado una serie de STATES para definir los respectivos estados que puede tener la batalla, y para asignar permiso a ejecutar una determinada acción al jugador o enemigo dependiendo de ese estado, los distintos estados definidos son:

   a) START: define el inicio de la batalla.
   b) PLAYERTURN: define el turno del jugador.
   c) ENEMYTURN: define el turno del enemigo.
   d) WIN: define la victoria del jugador, y por lo tanto el fin del combate.
   e) LOST: define la derrota del jugador, y por lo tanto, el fin del combate.

Figura 4.3: Esquema Scripts Sistema de combate

Por otro lado, EnemyBattle y playerBattle, son los scripts que definen las posibles acciones a realizar del enemigo (y su sistema de utilidad, que se explicará posteriormente), y del jugador respectivamente.

Por último, HUDSystemBattle, refleja a la interfaz del combate; diferentes estadísticas del jugador y el enemigo, posibles acciones que ocurren durante el combate, etc...

2. playerBattle: es el encargado de definir todas las acciones posibles que tiene el jugador: gritar, hablar y llorar. También se definen la potencia de esas acciones y las estadísticas del jugador (HP y Ansiedad).

3. EnemyBattle: es el encargado de definir todas las acciones posibles que tiene el enemigo: gritar, hablar y llorar. También se definen la potencia de esas acciones y las estadísticas del enemigo (HP y Ansiedad). Por supuesto a través de todas estas posibles acciones se ven gestionadas por el Sistema de utilidad implementado, que será explicado más adelante en un esquema diferente para facilitar su comprensión.

4. HUDSystemBattle: es el encargado de reflejar visualmente todo lo que ocurre en el combate, básicamente la UI (User Interface) del combate.

4.4. Desarrollo e implementación del Sistema de Utilidad

En este apartado se va a proceder a explicar el desarrollo e implementación técnica del sistema utilidad desarrollado para la IA del enemigo, sirviendo como apoyo la Figura 3.2, se va a hacer un desglose de la Figura 4.4 haciendo hincapié en el código:

Antes de pasar a la explicación de cada una de las entidades que se ven en la Figura 3.2, es necesario hacer una diferenciación de cada uno de los que hay, y la función que desempeñan, estas son:

1. Scripts: en este caso solo hay uno, EnemyBattle es el encargado (entre otras cosas), de definir una serie de métodos del sistema de utilidad encargados de calculo de pesos, decidir la siguiente acción y ejecutarla.

2. Objetos asociados a scripts: como su nombre indica, son todos aquellos los objetos asociados y defenidos en el propio script.

3. Scriptable Object: al contrario que los scripts comunes (definidos con MonoBehaviour [15]), los scriptable objects [16] son scripts que para

Figura 4.4: Esquema Scripts Sistema de Utilidad

su correcto funcionamiento no dependen de estar asociados a ningún GameObject , sino que son persistentes a lo largo de todo el proyecto. ActionsEnemy y DecisionFusionFactorsEnemy son aquellos Scriptable Objects que gestionan el calculo de los pesos del factor de decisión, y la elección de una acción determinada, respectivamente.

4. Factores de fusión: como se ha visto a lo largo de la explicación de este proyecto, de nen la cantidad de peso asociada a una determinada acción a través varias variables.

5. Acciones: las posibles acciones que pueden ser ejecutadas por el Sistema de Utilidad.

Una vez se tienen claras las posibilidades del esquema, y los signi cados de cada una de las entidades, se puede proceder a realizar un desglose de la secuencia de ejecución típica que se da en este Sistema de Utilidad, se dan 12 pasos como se observa en la Figura 4.4:

1. La ejecución comienza en EnemyBattle como es lógico, ya que es el script que gestiona todos los métodos relacionados con el Sistema de Utilidad. El primer paso será el comienzo de la ejecución del método de WeightAction , que comenzará cuando sea el turno del enemigo.

2. La ejecución del WeightAction , que es a su vez quien hace la llamada al resto de Scriptable Objects , que darán como resultado todos y cada uno

de los pesos finales asociados a cada acción. En WeightAction se contemplarán todas las acciones posibles con ActionsEnemy, y a su vez también se contemplarán todos los factores de fusión con DecisionFusionFactorsEnemy, para terminar finalmente con el calculo de los diferentes pesos asociados a cada factor de fusión (RolFactorToCry, etc...).

3. Una vez se tienen conocimiento del resultado de los pesos, se vuelve al inicio de la ejecución, donde EnemyBattle podrá hacer una llamada a DecideBestAction, que es el método encargado de comparar todos los pesos calculados, una vez comparados se podrá realizar una llamada al método ExecuteNextActionAI, que simplemente ejecutará la acción finalmente decidida: Gritar, hablar o llorar.

4.5. Código Github

Este breve apartado solamente pretende comunicar que todos los scripts de código desarrollados a lo largo del proyecto están públicos en el siguiente enlace: UtilitySystemTFG-RPG-Unity

5

Conclusiones y trabajos futuros

Acerca de las posibles reflexiones que se van a plantear en la Sección 5.1 y en la Sección 5.2, se hace especial hincapié en como se ha conseguido implementar de forma exitosa una gran credibilidad, realismo e inmersión para el jugador. Además se consideran nuevas herramientas y tecnologías para trabajos futuros.

5.1. Conclusiones

Se ha utilizado un enfoque basado en Sistemas de Utilidad debido a que a día de hoy es lo que se encuentra actualmente en el estado del arte. Trabajando en ello, se puede comprobar que la implementación, así como el uso de las tecnologías, es fundamental en esta propuesta. El uso de estas tecnologías son esenciales debido a que el principal objetivo de este proyecto es desarrollar un entorno de videojuego que sea capaz de ser lo más realista, inmersivo y creíble para el jugador. Esto se consigue a través del diseño de juego(World-Building, mecánicas, dinámicas y estéticas) , y la implementación técnica de un Sistema de Utilidad en los

enemigos. Además, aunque el desarrollo técnico es primordial, hay una serie de aspectos a tener en cuenta para llegar más lejos en este objetivo:

- Por un lado, además de ser un proyecto con un gran desarrollo técnico detrás, se ha mantenido la importancia en el diseño del entorno, para captar la atención del jugador y así conseguir de una forma más completa el objetivo final. Además de la parte técnica, se ha decidido darle importancia a más aspectos del proyecto, debido a las múltiples disciplinas que abarca un videojuego (arte, programación, diseño, música, etc...).

- Por último, al combinar un diseño capaz de sumergir al jugador en el gameplay del videojuego, y una implementación técnica lo suficientemente enriquecida se consigue esta credibilidad, realismo e inmersión al jugador.

5.2. Trabajos futuros

A lo largo del recorrido de este proyecto, y una vez expuestas las Conclusiones en la Sección 5.1, se ha visto que es posible seguir desarrollando trabajos futuros a partir de este proyecto. Dadas las circunstancias en las que se encuentran este tipo de tecnologías (continuas apariciones de nuevas tecnologías similares, más desarrollo y profundación en herramientas ya existentes, etc...), se puede retomar este proyecto en trabajos futuros a través de múltiples herramientas, no solo para compararlas unas con otras, sino también para mejorar el trabajo ya realizado. Las posibles líneas de trabajos futuros son:

1. Sistemas de planificación automáticas. (Especialmente usados en robótica) [17]

2. Sistemas multiagente [18].

3. Redes Neuronales Convolucionales (CNN): basadas principalmente en el análisis de datos (Reconocimiento de elementos visuales)

4. Redes Neuronales Recurrentes (RNN): usados principalmente para procesamiento de datos y reconocimiento de patrones en tiempo real (algo realmente útil para evitar la repetición de secuencias en el jugador, se consigue mayor realismo aún con la aplicación de esta herramienta en el desarrollo técnico del enemigo).

5. Redes Neuronales Profundas (DNN): similares a las recurrentes, pero en vez de procesar datos y patrones para evitarlos, lo procesan para aprender y realizarlos ellos, con esta herramienta de se obtiene la capacidad de que el enemigo pueda elaborar estrategias y patrones de ataque.

Más allá de los Sistemas Multiagente y los Sistemas de Planificación Automática ya mencionados, está el desarrollo de **Redes Neuronales**. Su aplicación no se tiene porque limitar al sistema de combate, sino que puede abarcar diversos aspectos del juego. Por ejemplo, en los JRPG, las redes neuronales podrán entrenar las conversaciones de los NPCs para crear interacciones más naturales y complejas.

El aprendizaje automático [19], pilar fundamental de las redes neuronales, es crucial para la credibilidad, el realismo y la inmersión en los videojuegos, de ahí el posible enfoque para trabajos futuros. Este permite que los elementos del juego se adapten y aprendan del comportamiento del jugador, creando experiencias únicas y dinámicas. Un ejemplo de su aplicación es el ChatGPT [20], una red neuronal de gran tamaño.

Bibliografía

[1] Kenney, \Kenney assets free." [Online]. Available: https://kenney.nl/assets

[2] J. Rasmussen, \Are behavior trees a thing of the past?'Introducing the Next Generation of AI: How Utility AIs Are Replacing Behaviour Trees. , pp. 279{299, 2016. [Online]. Available: https://www.gamedeveloper.com/programming/are-behavior-trees-a-thing-of-the-past-

[3] G. M. Toolkit, \The genius ai behind the sims," 2023. [Online]. Available: https://gmtk.substack.com/p/the-genius-ai-behind-the-sims

[4] I. MILLINGTON and J. FUNGE, AI For Games, Third Edition . CRC Press; 3nd edicon, 2019, vol. 3.

[5] K. B. M. B. A. van Bennekum Alistair Cockburn Ward Cunningham Martin Fowler James Grenning Jim Highsmith Andrew Hunt Ron Je ries Jon Kern Brian Marick Robert C. Martin Steve Mellor Ken Schwaber Je Sutherland Dave Thomas, Mani esto Agil , 2002-2016. [Online]. Available: https://agilemanifesto.org/iso/es/manifesto.html

[6] Asana, \Metodologa kanban," 2024. [Online]. Available: https://asana.com/es/resources/waterfall-agile-kanban-scrum

[7] B. R. Ian Schreiber, Game Balance CRC Pr I Llc, 2021, vol. 1.

[8] D. E. Cubero, \Que es el worldbuilding," Guionista, 2017. [Online]. Available: https://cursosdeguion.com/40-worldbuilding-creacion-mundos/

[9] T. Fox, \Undertale," 2015.

[10] A. H. Laboratory, \Earthbound," 1989-2022.

[11] G. Freak, \Pokemon," 1996-2024.

[12] C. M. Freitas, \Diseno de enemigos,"Disenador de videojeugos y prograamador2023. [Online]. Available: https://gamedesignla.com/enemigos-en-videojuegos/

[13] R. Kremers, Level Design: Concept, Theory, and Practice A K Peters/CRC Press, 2009, vol. 1.

[14] UnityEngine, \Scripting api scene manager." [Online]. Available: https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.html

[15] Unity, \Monobehaviour." [Online]. Available: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

[16] ||, \Scriptable objects." [Online]. Available: https://docs.unity3d.com/Manual/class-ScriptableObject.html

[17] R. Martn, \Planicacion automatica," 2016. [Online]. Available: https://blogswa.sel.inf.uc3m.es/?p=1400

[18] S. Gonzalez Jimenez, \Sistemas multiagente," 2018. [Online]. Available: https://docta.ucm.es/entities/publication/b1adbc61-3b12-4a3e-a666-fc32081be47f

[19] J. Gonzalez, \Inteligencia arti cial o aprendizaje automatico," 2018. [Online]. Available: https://culturacion.com/inteligencia-arti cial-o-aprendizaje-automatico-cual-es-la-diferencia/

[20] S. Gehles, \Gpt-3 { >como funciona?" 2022. [Online]. Available: https://neuro ash.com/es/gpt-3-que-es-lo-que-pasa/