UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN


# "Desarrollo de un Videojuego: Sangre Sin Refugio"

TRABAJO FIN DE GRADO

Julio - 2025

AUTOR: Andrés Pérez Leonís

DIRECTOR/ES: Antonio Peñalver Benavent

# Contents

## Ilustraciones

## Tablas

# CAPÍTULO 1: INTRODUCCIÓN

## 1.1 El Sector de los Videojuegos

El sector de los videojuegos ha observado un crecimiento enorme a lo largo de las últimas décadas, estableciéndose como una de las industrias más rentables y dinámicas de su ámbito. El avance de la tecnología, la popularización del medio y la expansión de las redes sociales hicieron esto posible. Esta industria genera millones y millones de euros cada año, atrayendo una cantidad enorme de empresas y desarrolladores. Además, se estima que su valor global de mercado alcanzó los 250.000 millones de dólares [1] durante el año 2024, superando sectores como el cine y la música. [1][2]

Algunas de las compañías con más peso en la industria son Sony, Nintendo, Microsoft, Rockstar Games, Valve Corporation y Tencent [3]. Estas impulsan constantemente la industria tanto en hardware como en desarrollo de software. Asimismo, el desarrollo de los smartphones contribuyo especialmente a la expansión del mercado de los videojuegos, especialmente con la aparición de los juegos *casuales*[2]. No obstante, pese a ser un sector con un éxito sostenido, actualmente la industria pasa por una de sus etapas más complejas. En los últimos años han surgido una serie de problemáticas. Mencionaré algunos de los principales:

### 1.1.1 Escasez de Chips (2020-23)

También llamado la crisis de semiconductores. Tuvo un gran impacto en múltiples industrias informáticas (automotriz, telecomunicaciones, etc.), entre ellos, la de

[1] Moneda de curso legal en estados unidos
[2] Videojuegos de partidas de duración corta y con poca necesidad de concentración. Para jugadores que juegan unas pocas veces a la semana

los videojuegos. Esto entorpeció enormemente su capacidad de producción y distribución de hardware.

Los semiconductores son piezas esenciales para la fabricación de dispositivos electrónicos, como consolas, tarjetas gráficas, procesadores y mandos, entre otros. Esto provocó una escasez generalizada, que, junto con un aumento de la demanda, termino en un incremento de los precios.

Podemos encontrar múltiples causas. Por un lado, la pandemia del COVID-19 generó la detención de las cadenas de suministro global, debido a los cierres de las fábricas y una gran disminución de la producción. Por otro lado, la guerra comercial entre Estados Unidos y China, el cual generó restricciones al acceso de materias primas y componentes clave. A esto se le sumaron efectos climáticos, como sequías en Taiwán (uno de los mayores productores de chips) que afectaron a las fundiciones locales.

Puesto que la industria dependía enormemente de esta, poco diversificada, cadena de suministro, hizo que las empresas se replantearan sus estrategias de producción y distribución, ocasionando que empresas como Intel comenzaran la construcción de 2 fábricas en Ohio y Arizona [4].

### 1.1.2 Una subida de precios exagerada

Estos últimos años, la comunidad de jugadores se ha encontrado con una problemática monetaria: un aumento excesivo y generalizado de los precios en consolas, videojuegos y periféricos. Este fenómeno ha generado un gran debate entre los jugadores y desarrolladores, especialmente por ser la primera generación de consolas que ha experimentado un aumento de precio de venta tras su salida.

En los años recientes, la comunidad de jugadores ha observado un aumento significativo y constante en los costos de las consolas de videojuegos y de los juegos de lanzamiento.

En específico, la generación actual de consolas, que abarca modelos como PlayStation 5 (en sus distintas versiones), Xbox Series X/S y las versiones renovadas de Nintendo Switch, ha sido la primera en la historia de la industria en sufrir incrementos de precio tras su lanzamiento oficial, lo que generó múltiples críticas en la comunidad [5][6][7].

La siguiente tabla sintetiza las modificaciones en los precios de las consolas:

| **Consola** | **Precio Anterior (€)** | **Precio Actual (€)** | **Fecha de lanzamiento** |
|---|---|---|---|
| PS5 Edición Digital | 399.99 | 499.99 | Noviembre del 2020 |
| PS5 con lector de discos | 499.99 | 549.99 | Noviembre del 2020 |
| PS5 Slim | 550 | 550 | Noviembre del 2023 |
| PS5 Slim Digital | 450 | 450 | Noviembre del 2023 |
| PS5 Pro | 799 | 799 | Noviembre del 2024 (No lanzada) |
| Xbox Series X | 499.99 | 599.99 | Noviembre del 2020 |
| Xbox Series S | 299.99 | 359-399 | Noviembre del 2020 |
| Nintendo Switch | 329.99 | 299.99 | Marzo del 2017 |

*Tabla 1 - Comparación de precios de consolas*

Como se puede apreciar, los precios base de los primeros modelos de consolas (las más antiguas) han experimentado un aumento considerable, excepto para la Nintendo Switch, cuyo costo ha permanecido en el margen de la normalidad. Sin embargo, si comparamos dichas cifras con los precios de sus generaciones anteriores, encontramos un contraste mucho más marcado. Por ejemplo, la PlayStation 4 Slim fue lanzada con un precio base de 299€, lo cual representa una diferencia de 250€ respecto a su versión más actual la PS5 Slim, cuyo rendimiento es, en esencia, similar a sus hermanos generacionales (PS5 -> PS5 Slim, PS4 -> PS4 Slim) salvo por modificaciones físicas, principalmente con una reducción del tamaño de estos.

Estos aumentos también se ven reflejados en los precios de los videojuegos. Tradicionalmente, los títulos AAA[3] oscilaban entre los 40-60€, pero estos dos últimos 2 años se estableció una nueva media de precios entre los 70-80€ por título. Incluso, algunas empresas como Nintendo han fijado sus precios en los 90€ para sus ediciones físicas y 80€ para el digital. No obstante, la comunidad de jugadores teme que este aumento consolide el nuevo estándar, especialmente con la llegada del videojuego Grand Theft Auto VI, que podría fijar un precio de salida de 100€.

Las compañías desarrolladoras lo justifican mencionando el incremento en los costes de producción, derivados de:

- La implementación de gráficos más realistas
- Uso de motores de desarrollo más complejos
- Mayores tiempos de desarrollo
- Demanda de experiencias y duraderas por parte de los jugadores

[3] Los videojuegos se clasifican en 3 categorías:
- Triple A (AAA): Videojuegos realizados por grandes empresas: Presupuestos y equipos enormes
- Doble A (AA): Videojuegos realizados por medianas/pequeñas empresas. Presupuestos y equipos más modestos
- Indie (A): Videojuegos realizados por unas pocas personas con/sin presupuestos reducidos

Estos comentarios se reciben con cierto escepticismo, considerando estos incrementos respuestas a una un intento de maximizar beneficios que a una verdadera necesidad financiera. De hecho, hace unas pocas semanas la empresa Gearbox creadora de la saga *Borderlands* anunció que su último título tendrá un precio inicial de 80€ con el comentario: "Si eres un verdadero fan, encontrarás la forma de hacerlo posible." [8]. Esto desató numerosas amenazas por parte de la comunidad y la organización de un boicot. Luego de varios intentos desesperados y mal planeados por parte del CEO (Randy Pitchford), y un mayor rechazo de sus seguidores, declararon que el precio base del juego será 70€.

Es importante destacar que estos niveles de precios han estado presentes en la industria durante más tiempo, aunque con notables diferencias respecto a la situación presente. Como se mencionó previamente, los precios base han oscilado entre los 40-60€, sin embargo, se podían hallar ediciones especiales y coleccionistas con precios considerablemente más elevados y parecidos a los actuales.



### 1.1.3 Decadencia de la industria a nivel laboral

La industria del videojuego, a pesar de su crecimiento económico, enfrenta actualmente una profunda crisis estructural en el plano laboral.

#### 1.1.3.1 Despidos masivos y hostilidad empresarial

Práctica recurrente en las etapas finales de desarrollo de los proyectos. En muchos casos, las compañías deciden prescindir de perfiles técnicos y creativos con el fin de reducir costes luego de completar las etapas más duras. Tendencia que ha escalado a niveles totalmente alarmantes, ya que durante los primeros meses de 2024 se superaron el número de despidos registrados en todo 2023:

| Año | Despidos Totales |
|---|---|
| 2022 | 8500+ |
| 2023 | ~11250 |
| 2024 | 14600+ |
| 2025 (2 primeros meses) | 1200+ |

*Tabla 2 - Despidos en la industria del videojuego al año [9]*

Debo mencionar que estas cifras no incluyen las renuncias por parte de los empleados, como directivos, desarrolladores y diseñadores. Además, de la rotación constante de personal que provocan retrasos en los desarrollos, dañan la ya deteriorada estabilidad de los equipos, debido a la necesidad de familiarizarse con las dinámicas internas por parte de los nuevos integrantes [10].

En contraposición, corporaciones de gran envergadura como Microsoft, Embracer Group o Sony han intensificado la adquisición de estudios autónomos o de menor envergadura como parte de su estrategia de crecimiento, expansión y control de mercado. Aunque estas adquisiciones suelen venir acompañadas de promesas de libertad creativa para los estudios absorbidos, la realidad es que muchos de ellos terminan cerrando al poco tiempo si no generan los beneficios esperados [11].

También empresas como Microsoft se han visto involucradas en investigaciones y denuncias antimonopolio, tanto por parte del gobierno estadounidense y europeo. Un ejemplo de esto es la compra de Activision Blizard.

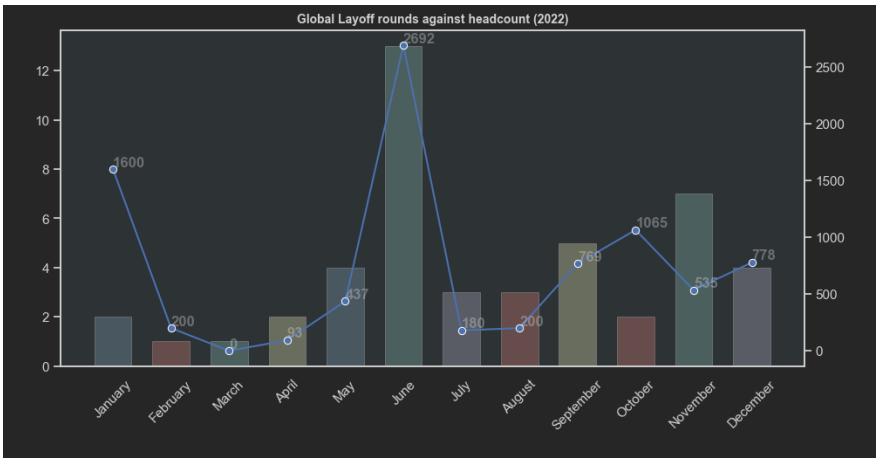


*Ilustración 1- Gráfico de despidos (2022) [12]*

Como es comprensible pensar los cierres de dichas empresas implican despidos masivos, incluso en ocasiones cuando los títulos lanzados al mercado han sido objetivamente un éxito o han generado beneficios, aunque sean inferiores a los proyectados o requeridos por parte de los inversores. Esta política ha sido altamente criticada por reducir la diversidad creativa del sector y concentrar el poder en unos pocos actores. El ejemplo más marcado de esto fue la empresa de *Hi-Fi RUSH*, Tango Gameworks [13].

#### 1.1.3.2 Cultura del Crunch

La llamada crunch culture, o cultura del trabajo excesivo, es una práctica arraigada en la industria desde los años 80, aunque en la última década ha comenzado a recibir una atención mucho mayor por parte de medios, trabajadores y legisladores. El término hace referencia a los periodos prolongados de jornadas laborales extremadamente largas, generalmente impuestas en las fases finales del desarrollo [14].

Uno de los casos más emblemáticos fue el denunciado en 2004 en un blog de LiveJournal, donde se exponían jornadas forzadas de hasta 90 horas semanales en Electronic Arts. Aunque no siempre ilegales, estas prácticas rozan o infringen normativas laborales según la jurisdicción [15]. Por ejemplo:

- En Estados Unidos, el crunch puede ser legal y no siempre requiere compensación por horas extra, dependiendo del estado.
- En países como España o Reino Unido, existen leyes que regulan de manera más estricta la jornada laboral y exigen remuneración adicional.

No obstante, incluso en estos países, incluso en estos países, la presión por mantener el empleo y la elevada competitividad del sector provocan que numerosos empleados dimitan oficialmente de estas salvaguardas, lo que permite a las compañías ejercer prácticas de abuso con total impunidad.

#### 1.1.3.3 Altísima competitividad

La industria del videojuego se caracteriza por una competitividad laboral y comercial extrema, que afecta tanto a los profesionales establecidos como a quienes buscan incorporarse al sector [16]. Dos factores son clave:

- El miedo a perder el empleo en un mercado con pocas vacantes y una gran cantidad de talento disponible.
- La saturación de la oferta de videojuegos, especialmente en plataformas como Steam, donde el volumen de lanzamientos diarios hace que destacar sea una tarea casi imposible para estudios pequeños.

Esta situación impacta negativamente en los desarrolladores independientes (*indies*), que con escasos recursos deben competir tanto con la industria AAA como con otros miles de creadores, siendo "una gota en un océano" de propuestas similares.

A su vez, la alta competitividad provoca que muchos profesionales acepten condiciones laborales inaceptables con tal de mantener su puesto o acceder a un empleo en el sector.

También persisten graves desigualdades de género y diversidad:

- Se estima que entre el 60% y el 75% del personal en áreas de diseño, arte y programación es masculino.
- En áreas técnicas, como ingeniería de software en EE. UU., el porcentaje masculino asciende al 95%–97%.
- Solo el 12% de los trabajadores se identifican como personas con discapacidad y un 25% pertenecen al colectivo LGBTIQ+, de los cuales apenas un 4% se identifican como no binarios o trans.

#### 1.1.3.4 Ciclos de desarrollo extensos

El desarrollo de videojuegos de gran presupuesto puede prolongarse durante 3 a 7 años, especialmente en proyectos AAA. En el caso de los estudios independientes, los ciclos suelen variar entre 6 meses y 2 años, aunque también pueden alargarse por falta de recursos.

Estos periodos extensos, combinados con plazos de entrega ajustados y escasa rotación de tareas, generan síntomas de agotamiento emocional y burnout creativo, lo cual contribuye a la fuga de talento y al abandono del sector por parte de muchos profesionales.

A pesar de lo anterior y con el objetivo de reducir costes y minimizar riesgos financieros, muchas empresas optan por estandarizar procesos creativos y reutilizar IPs ya consolidadas. Esta estrategia se traduce en:

- El lanzamiento anual de secuelas o versiones ligeramente modificadas de franquicias existentes.
- La reutilización de animaciones, estructuras de código y motores gráficos.
- La modificación superficial de títulos anteriores (historia y estética) presentados como productos nuevos.

Esta práctica, aunque rentable, ha sido duramente criticada por su impacto negativo en la innovación. Esta restringe la diversidad de propuestas y limita el desarrollo de conceptos originales.

#### 1.1.3.5 La muerte del videojuego físico

La digitalización nos ha proporcionado múltiples beneficios y comodidades en la vida cotidiana y laboral, además como todo tiene sus inconvenientes. Originalmente, la industria del videojuego ha tenido una completa distribución física de sus títulos. Sin embargo, con la aparición de Steam y Epic Games Launcher, la distribución de estos se ha digitalizado con el paso del tiempo [17].

El empresario ve que grandes beneficios en esto: reducción de costos por la nula necesidad, control directo del canal de venta, distribución global inmediata, etc. No obstante, si hablamos desde el lado de los consumidores la línea de los beneficios es algo más difusa. Se podría suponer que los costos de los juegos comercializados desde estas plataformas disminuirían debido a la reducción de gastos, pero no sucedió. En cambio, es importante señalar que mediante estas plataformas no estas adquiriendo juegos, sino licencias de uso revocables, por lo que pueden impedirte el acceso a ellos en cualquier instante. Aunque esto ya ha sucedido en numerosas plataformas, como con la Nintendo 3DS y el cierre de la Nintendo Eshop, el caso más sonado fue el de *Need for Speed 2015*, cuyos servidores cerraron y se eliminaron los accesos a este en todas las plataformas de venta. Este caso terminó en la corte dentro de la Unión Europea y se obligó a dicha empresa a permitir el acceso offline al juego [18].

### 1.1.4 Dificultad de inclusión

La representación de las minorías sociales dentro del contenido de videojuegos sigue suscitando numerosas polémicas dentro de la comunidad. Sin embargo, la inclusión de personajes de diferentes culturas y racializadas, miembros del colectivo LGBTQIA+, personas con discapacidad o incluso personas con un cuerpo no normativo, en numerosas ocasiones, ha sido objeto de críticas por parte de ciertos sectores del público. Estas reacciones se vinculan a un rechazo

a la llamada ideología woke [4]y a una supuesta inclusión forzada, acusando a los desarrolladores de anteponer la agenda política y social a la calidad narrativa y jugable. Un ejemplo marcado de esto es *The Last Of Us Parte 2* [18][19]. Aunque, el título es considerado por muchos uno de los mejores de su año, no estuvo libre de críticas. No solo por aspectos técnicos, como bugs[5], sino también por decisiones relacionadas con el trasfondo y estética de sus personajes: La protagonista, Ellie, es una mujer abiertamente lesbiana; la antagonista, Abby, tiene un cuerpo musculado poco habitual en representaciones femeninas en la industria.

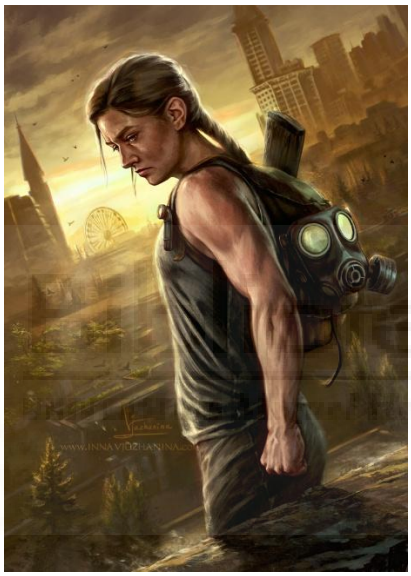

*Ilustración 2 – Abby [21]*

Como se puede imaginar, ambos aspectos generaron conflictos dentro de algunos jugadores, llegando a emitir comentarios ofensivos al cuestionar dicha apariencia

[4] Término referido a la conciencia sobre la injusticia racial y la discriminación. Actualmente, describe una postura política progresista enfatizando temas de equidad social, el feminismo, la justicia ambiental y los derechos de las personas LGBTIQ+. Usualmente usado de forma peyorativa.

[5] Termino referido a los errores o defectos en el código de un programa provocando un comportamiento incorrecto

y orientación sexual con afirmaciones como: "También me pregunto cómo Abby es tan musculosa, ¿encontró un alijo completo de esteroides?" [22].

Estos comentarios indican un desconocimiento sobre los cuerpos diversos y el entrenamiento físico, además de una visión estereotipada de los cuerpos femeninos. Por otro lado, revelan cómo algunas audiencias ven la inclusión como una imposición ideológica en vez de como una forma legítima de mejorar la narrativa y de representar a una gran parte de los jugadores.

A pesar de estas controversias, la industria continúa avanzando lentamente hacia una mayor representación, aunque en muchas ocasiones sea usado como marketing en lugar de como una necesidad global en respuesta a una realidad social.

### 1.1.5 Bastiones de la creatividad

En medio de las polémicas de falta de originalidad y la limitada creatividad en los títulos AAA, surgen lo que podríamos llamar "bastiones de la creatividad": los videojuegos independientes o "indie" (profesionales). Juegos desarrollados por pequeños equipos – incluso por una sola persona – y con presupuestos nulos o pequeños. Estos destacan por su gran libertad artística y por el amor que tienen gran parte de la comunidad hacia estos.

A pesar de la dificultad de financiación que estos desarrolladores enfrentan y la poca visibilidad en un mercado repleto de AAA, estos cuentan con una gran ventaja: la libertad creativa (tanto de jugabilidad como visual). Les permite explorar mecánicas y ambientaciones únicos en el medio, resultando más originales y frescos que los títulos producidos por las grandes compañías.

También cabe mencionar que sus precios son bastante más accesibles que los AAA. No han visto un aumento de precios en estos años y suelen situarse entre los 10 – 25 euros. Aunque podemos encontrar algunas excepciones tanto por encima como por debajo de dicho rango. Si bien algunos de los miembros de la

comunidad consideran que precios superiores a los 10-15 euros son elevados para un título de esta índole, esta visión es minoritaria frente al reconocimiento que estos juegos reciben por su valor tanto artístico como jugable.

No solo tenemos los videojuegos indie, también han surgido nuevos estudios como resultado de los despidos masivos mencionados anteriormente. Un ejemplo de estos estudios es Yellow Brick Games, fundada por el veterano de Bioware Mike Laidlaw, reconocido desarrollador responsable de sagas como *Dragon Age* y *Mass Effect* [23]. Sin embargo, estas empresas no solo surgen en respuesta de la inestabilidad laboral, sino también como una forma para recuperar la libertad creativa perdida dentro de las grandes corporaciones.

Dichos estudios se pueden concentrar en proyectos de menor tamaño y con una identidad mucho más marcada. Los títulos realizados por estos se les denomina AA. Algunos ejemplos destacados son: *Stray, Helldiver II* y *Hi-Fi Rush*.

Por último, he de mencionar que los juegos no se limitan únicamente al entretener, sino que puede abarcar muchísimos más ámbitos. Estos últimos años se han construido diferentes proyectos sobre la salud mental y la medicina. Un caso destacado es *Endeavor RX* es un videojuego, desarrollado por Alivi Interactive, primer videojuego aprobado por la FDA para el tratamiento del TDAH infantil [24]. Otro gran ejemplo es el proyecto *The Mind Guardian*, utilizado para la detección temprana del alzhéimer [25].

## 1.2 Descripción del proyecto

Este Trabajo de Fin de Grado tiene como objetivo el desarrollo completo de un videojuego original, construido desde cero, y pasando por todas las fases de creación: desde la concepción de la idea, el diseño de mecánicas, hasta su eventual publicación.

El videojuego se desarrolla en una ambientación oscura: el jugador encarna a un vampiro que debe alimentarse de los habitantes de una aldea, intentando, entre otras cosas, pasar desapercibido y no ser descubierto por los demás aldeanos o guardias. La jugabilidad combina elementos de sigilo, estrategia y toma de decisiones, con un sistema de sospechas y detección para promover una experiencia más enriquecedora.

Se sitúa en la ciudad de Brujas, Bélgica, durante la Edad Media durante la época del gótico arquitectónico (siglo XII). Luego, a nivel visual, se optó por un estilo 2D simplificado.

## 1.3 Justificación del proyecto

### 1.3.1 Desarrollo personal y amor al medio

Desde mi infancia, los videojuegos han sido un componente esencial en mi vida. Mi hermano me obsequió mi primera consola, una Game Boy. Desde aquella consola inició una fuerte conexión con este medio. Aunque siempre se me ha considerado una persona extrovertida y sociable, los videojuegos pasaron a ser un espacio seguro, un refugio donde descansar tras las horas del colegio y donde podía explorar libremente cientos de mundos.

Gracias a mi hermano y al trabajo de mis padres, en mi casa siempre tuvimos oportunidad de disfrutar de muchas consolas, como la PlayStation 2, y recuerdo este con especial cariño ya que me pasaba numerosas tardes viendo como jugaba mi hermano al *Spider-Man 2*, esperando mi turno para poder jugar. Con el tiempo, se adquirieron otras consolas como la GameCube, la PlayStation 3 y 4, lo cual siguió alimentando mi curiosidad y pasión por lo videojuegos. No obstante, la Nintendo 2DS fue la primera consola que me hizo completamente única, con la que dedicaba horas y horas explorando sus universos, solucionando sus retos y descubriendo sus secretos. En cierto modo, los videojuegos satisfacían mi necesidad de exploración, ya que, durante algunas de las salidas con mis padres

a lugares con acceso al bosque, me escaqueaba para explorar aquella zona – aunque más tarde valiera alguna reprimenda.

Mi vínculo con los videojuegos cambió al momento de entrar a esta carrera universitaria, donde aprendí los fundamentos de la programación y comprendí el funcionamiento de los sistemas de software, entre ellos los videojuegos, desde dentro. Fue entonces cuando me pregunté: "¿Por qué no hacer un videojuego yo mismo?". A pesar de que en varias ocasiones he intentado realizar este proyecto, al llegar el momento de hacer el Trabajo de Fin de Grado decidí dar el paso.

Aunque este proyecto no representa todavía el juego que algún día aspiro a desarrollar, si conforma el primer paso real hacia ese objetivo. Mi propósito final es crear un videojuego que sea capaz de evocar en otros jugadores aquellos sentimientos que yo tuve durante mi infancia: un espacio de consuelo y exploración. Un juego que pueda acompañar y ayudar, tal y como estos hicieron conmigo en su momento.

Por otro lado, mi intención con este proyecto es sobre todo conocer como es el proceso de desarrollar un videojuego desde 0: desde la concepción de la idea, la elaboración del concept art, hasta su publicación y posible comercialización. Además de conocer las diferentes herramientas que se usan en el día a día en un proyecto de esta índole.

### 1.3.2 Venta del producto y aporte al mundo

Otro aspecto relevante para el desarrollo de este videojuego es su posible publicación y venta al público. Dado que este proyecto no es muy grande, su precio final se situaría entre los 3 y 5 euros. No obstante, para aquellos desarrolladores que están iniciándose dentro de esta industria, cuyo desarrollo no haya requerido una gran inversión económica y un desarrollo muy prolongado (+6 meses, por ejemplo), se recomienda una distribución gratuita como parte de una estrategia para darse a conocer.

La publicación de un videojuego sin un coste permite alcanzar a un público mucho mayor y aumenta la visibilidad del desarrollador. Esta estrategia si se realiza correctamente y con un poco de suerte, se puede llegar a construir una base de jugadores y obtener feedback [6]para futuros desarrollos.

Además del aspecto comercial, este proyecto intenta ofrecer un experiencia gratificante y divertida al usuario, mientras busca la manera de superar los niveles y problemas que puede encontrarse.

Al ser un producto tecnológico implica usar numerosas herramientas para su desarrollo, mencionaré las más importantes:

- Motor de videojuegos: Unreal Engine, Unity y Godot
- Software de gráficos (entornos, personajes, etc.): Krita, Inskape, Procreate, Blender, GIMP y Maya
- Software de sonido: Audacity y Reaper
- Software de Testing: VSXCode, Unity Test Runner, Unreal Engine Debugger

[6] Retroalimentación de un proceso. Usualmente información sobre errores.

# CAPÍTULO 2: ANTECEDENTES Y ESTADO DE LA CUESTIÓN

## 2.1 Patrones de diseño

Un patrón de diseño es una estrategia empleada para organizar el código con el fin de hacer más legible, mantenible y escalable. Durante el desarrollo de este proyecto, he investigado e implementado diversos patrones que me han permitida dar una mejor estructura a las funcionalidades del videojuego y reducir la complejidad de las relaciones entre los distintos objetos [26].

Uno de los patrones más significativos que se han implementado ha sido el patrón State, que posibilita cambiar la conducta de un objeto basándose en un estado interno. Además, este patrón tiene una fuerte conexión con la idea de la Máquina de Estados Finitos. En este contexto, cada estado representa una funcionalidad del personaje. Por ejemplo, si el jugador no está realizando ninguna acción, el personaje se encontrará en el estado "Parado", o más bien el estado por defecto del objeto. Si, por el contrario, el jugador decide atacar a un NPC, el personaje cambiará al estado "Atacando". Es importante resaltar que un objeto solo puede encontrar en un estado a la vez, y las transiciones entre estos estados deben estar claramente señalados, en función de eventos y reglas establecidas.

Otro patrón que resultó especialmente útil es Mediator. Su propósito es reducir las dependencias directas entre objetos y evitar estructuras de comunicación caóticas. En situaciones donde 2 objetos necesitan comunicarse frecuentemente y aplican numerosos métodos para la comunicación, este patrón permite centralizar las comunicaciones a través de una instancia mediadora, la cual mantiene las referencias necesarias. Evitando de esta manera que los objetos se comuniquen de forma directa y mantengan referencias mutuas, favoreciendo el desacoplamiento.

En mi caso, la implementación de este último patrón se ha complementado con el uso del patrón Singleton, que garantiza una instancia única de un objeto, el cual es accesible de manera global. Además, se combinado con una técnica de optimización llamada Object Pooling, que permite la reutilización de objetos en lugar de instanciar nuevos. El resultado fue la creación de un Mediator Manager, encargado de gestionar las peticiones de comunicación, comprobando que ambos objetos estén preparados para el intercambio de información y asignándoles un mediador.

## 2.2 Técnicas de optimización

Como puede intuirse en cualquier desarrollo de software, la optimización es un aspecto crucial para garantizar un correcto funcionamiento. Las técnicas de optimización son métodos diseñados para usar de forma más eficiente los recursos del sistema, como puede ser la memoria y el procesamiento.

En el caso concreto de este proyecto, se ha implementado una técnica conocida como Object Pooling. Consiste en la reutilización de objetos ya instanciados en lugar de crear y destruir objetos, lo cual es un proceso bastante costoso. Esta resulta útil en entornos donde se necesitan múltiples instancias del mismo tipo objeto durante un determinado número de ciclos. Durante el desarrollo de este videojuego se utilizó esta técnica para optimizar el uso de los objetos relacionados con el patrón de diseño Mediator. Su implementación se basa en 2 componentes esenciales:

- Object Pool Manager: Es responsable de administrar la colección de objetos intermedios registrados previamente. Antes de crear uno nuevo comprueba si existe algún mediador disponible dentro del "pool". Si hay uno libre, lo reutiliza y asigna; de lo contrario, instanciará uno nuevo.

- Objecto Instanciable: Es el objeto que se reutiliza constantemente. Es este caso son los objetos mediadores encargados de facilitar las comunicaciones entre NPCs.

## 2.3 Herramientas de desarrollo

Una de las piezas clave en el desarrollo de cualquier videojuego es el motor de videojuegos. Estos son frameworks [7]que proporcionan unas herramientas para facilitar el desarrollo de estos productos. Su funcionalidad radica en ofrecer al desarrollador y al videojuego de:

- Sistema de renderizado gráfico, el cual puede ser 2D y/o 3D
- Simulación de físicas
- Gestión de animaciones, scripts y sonidos
- Conectividad de red
- Gestión de memoria y recursos

Cada motor cuenta con su metodología enfoque de trabajo distinto. Los más destacados son Unity, Godot, Unreal Engine y GameMaker, todos ellos con interfaces gráficas. Sin embargo, podemos encontrar alternativas como Pygame, que carece de interfaz, pero es utilizado gracias a su sencillez y flexibilidad.

Además de los motores, es muy habitual utilizar herramientas para el diseño y creación de assets[8]. Algunas de las más comunes son:

- Blender y Autodesk Maya: para la creación y animación de modelos 3D [28][29].

[7] Es el conjunto de herramientas y componentes que facilitan el desarrollo de aplicaciones y sistemas

[8] Son los recursos de un videojuego: sonidos, fondos, iconos, personajes, etc. Pudiendo ser tanto 2D como 3D

- Procreate: Aplicación de ilustración empleado para la creación de escenarios, personajes y fondos en 2D, incluso para animaciones en Stop Motion (animación cuadro por cuadro) [30]
- Inskape: Es un software de diseño vectorial, usado para elementos de la interfaz, aunque puede ser usado para la creación de Sprite y personajes [31]
- Substance Painter: Especializado en la creación de materiales y texturas realistas en modelos 3D [32]

Debo mencionar que no siempre es necesario crear todos los elementos visuales. Existen plataformas como Itchio o OpenGameArt donde artistas comparten sus recursos bajo licencia de uso libre. No obstante, solo se recomienda el uso de assets gratuitos en elementos secundarios, como piedras, arboles, etc.

## 2.4 Historia de los vampiros

El origen del vampiro, tal como lo conocemos hoy, es el resultado de una evolución cultural extensa, arraigada en el folclore europeo del XVIII. Aunque pueden encontrarse figuras similares al vampiro en diversas culturas alrededor del mundo, es en los Balcanes, Europa, donde se consolida su forma actual [33].

Durante este periodo, en zonas como Grecia, Bulgaria, Serbia y Hungría, comenzaron a circular leyendas de criaturas nocturnas que salían de sus tumbas para alimentarse de los vivos. Podemos encontrar diversos nombres en función de la región: Vrukolakas, Varkolak, Wampir, Upir o Upior. Representaban entidades grotescas, con el abdomen hinchado por la sangre que consumían, con la piel descamada y la boca manchada de rojo. Su método de ataque más habitual era sentarse sobre el pecho de sus víctimas con el objetivo de asfixiarlas, o chuparles la sangre directamente del corazón [34][35].

*Ilustración 3 – Upir*

Estos vampiros solían ser el resultado de muertes no natural o mal tratadas. Personas que se suicidaban, que cometían malos actos o que no habían recibido un entierro (cristiano) adecuado eran los mayores candidatos para regresar de la muerte. Para prevenirlo realizaban rituales que incluían: cortarles la cabeza, clavarles una estaca en el corazón o quemarlos.

Otro factor que influyó en la aparición de estas entidades fue la medicina poco desarrollada de la época. Enfermedades como la tuberculosis afectaban a las familias: debilitando el cuerpo, provocando tos con sangre, palidez y perdida peso. Síntomas atribuidos a los vampiros. También existen teorías que relacionan los vampiros con la enfermedad de la porfiria, que provocaba sensibilidad a la luz, palidez y problemas en la piel, dando la imagen de la entidad de la noche.

La criatura fue evolucionando hasta transformarse, en el siglo XIX, en la figura refinada y simbólica con el auge del romanticismo. Movimiento cultural que favoreció la exploración de lo irracional, la muerte, lo sobrenatural y lo exótico. Grandes obras de la época tenemos:

- El vampiro (1819) de John William Polodori donde se presenta uno de los primeros vampiros aristocráticos, seductores e inmortales que conocemos.
- Carmilla (1872) de Sheridan Le Fanu, con fuerte inspiración en la figura histórica de Elizabeth Báthory.
- Drácula (1897) de Bram Stoker, posiblemente la representación más icónica, basada parcialmente en la leyenda de Vlad Tepes, el Empalador

## 2.5 Referencias

Gracias a mi experiencia en la industria del videojuego, al plantear el desarrollo de este proyecto tenía una idea clara de los referentes que quería tomar para los diferentes apartados del juego.

En relación con la movilidad y control del personaje, tome como referencia principal *Hollow Knight*, desarrollado en 2017 por Team Cherry. El desplazamiento es ágil y dinámico, que encajaba con la experiencia que deseaba ofrecer. Aunque en este proyecto el diseño de niveles no es tan amplio como el del título mencionado, sí buscaba replicar esa fluidez.

A nivel artístico decidí combinar 2 influencias:

- Beholder, desarrollado por Warm Lamp Games en 2016, por su estilo y estética simple y con personajes diferenciados, pero sin mucho detalle con el fin de obtener una atmósfera oscura y misteriosa.
- Valiant Heart: The Great War, desarrollado por Ubisoft Montpellier en 2014. Lo seleccioné debido a sus fondos y su paleta desaturada.

Por último, decidí ambientarlo en Brujas, Bélgica. Gracias a su marcada arquitectura gótica, calles estrechas, canales y edificios ofrece un entrono que encaja de perfectamente con el estilo visual planteado. Además de reforzar una atmosfera oscura.

# CAPÍTULO 3: OBJETIVOS Y LÍMITES DEL PROYECTO

## 3.1 Objetivos

### 3.1.1 Objetivos generales

Como he comentado dentro del apartado anterior, uno de mis principales objetivos era conocer en profundidad el proceso de creación de un videojuego. Incluyendo las etapas iniciales de concepción de ideas y diseño visual, hasta un desarrollo técnico y posible publicación. Además de ponerme a prueba para gestionar un proyecto de principio, incluyendo su planificación, organización, documentación y cumplimiento de plazos. De este modo podía evaluar mis capacidades y mejorarlas, encontrando estrategias para el control del estrés y una correcta forma de concentrarme en el proyecto.

Otro de mis objetivos era tener un contacto real con la comunidad de desarrollados de videojuegos, en especial con los independientes. Aunque no he llegado a contactar con figuras destacadas de la comunidad como Alva Majo y Guinxu, sí he podido participar y comunicarme con otros desarrolladores en foros, como Reddit, y comunidades de Discord [36]. Donde encontré gente dispuesta ayudarse entre sí: compartir dudas, obtener respuestas y conocer a otros dentro del ámbito con más experiencia que tú.

### 3.1.2 Objetivos principales

El desarrollo de este proyecto se llevará a cabo con utilizando el motor de desarrollo Godot, apoyado de los lenguajes de programación C# y GDScript. Inicialmente Godot trabaja por defecto con GDScript, pero ofrece compatibilidad

con C# mediante el uso de la API [9]de .NET 6, usando de esta manera las herramientas y estructuras propias del entorno [37].

Tal como se ha mencionado con anterioridad, el objetivo principal de este trabajo es experimentar y pasar por todas las fases de desarrollo de un videojuego, los cuales se dividen en 3 [38]:

- Preproducción:
  Esta etapa inicial busca documentarse sobre las herramientas disponibles, así como la definición de todos los aspectos fundamentales de un videojuego: concepto, público objetivo, vías de monetización, presupuesto, análisis de competencia, fechas de límite, estilo artístico, entre otros.
  También mencionar el GDD, considerado la "biblia" del proyecto. Este documento recoge todo lo relacionado con el videojuego, desde la historia y los personajes, hasta sus mecánicas y dirección artística. Al finalizarlo, este no debe ser modificado, y se recorre a este a la hora de tomar decisiones importantes durante el desarrollo [39].
  Para este proyecto, se realizó una investigación sobre la historia de los vampiros, su origen y zonas donde eran más frecuentes sus apariciones. Asimismo, me documenté sobre la vestimenta y el contexto histórico de la época, recopilé referentes tanto visuales como de jugabilidad, los cuales me ayudaron a definir la estética y el diseño del proyecto. En base a esto, construí el GDD.

- Producción: Esta fase corresponde al desarrollo puro del videojuego. Se crean modelos, sprites, entornos y se implementa el código de las funcionalidades del juego. Es común, durante esta fase, lanzar pequeñas campañas de marketing para medir la aceptación del concepto entre el público general.

[9] Conjunto de reglas que permite que diferentes sistemas se comuniquen entre sí.

En mi proyecto, no se ha realizado ninguna campaña de marketing, ya que he decidido centrar mis esfuerzos en la implementación de los sistemas y mecánicas del juego, como puede ser el control de los personajes, diseño de escenarios y la aplicación de patrones de diseño con el fin de mejorar la mantenibilidad y organización del código.

- Postproducción: En esta última fase, aún sin iniciar en mi proyecto, se realiza la publicación y la mayor parte de las campañas que dan a conocer el producto. Sin embargo, para este proyecto se prevé su publicación con el objetivo de que los usuarios puedan jugarlo y recibir feedback por parte de estos.

Cabe destacar que, en un entorno profesional, estas fases no suelen desarrollarse de manera secuencial, ya que algunas tareas pueden solaparse en función de las tareas realizadas. Sin embargo, al tratarse de un proyecto individual, se ha seguido un enfoque algo más lineal y estructurado al habitual.

Durante la etapa de preproducción, se definieron una serie de funcionalidades/sistemas a implementar antes de entregar el proyecto:

- Máquina de estados: Aplicación del patrón de diseño State, que permite organizar el comportamiento de los objetos en función de un estado.
- Sistema de Movimiento: Gestión del movimiento tanto del personaje principal (el vampiro) como de los NPCs
- Sistema de ataque: Implementación de la mordida del vampiro, así como la respuesta de los guardias al detectar a este
- Sistema de detección y ocultamiento de cadáveres: Permite al vampiro recoger y ocultar los cuerpos de las víctimas, y a los NPCs descubrirlos si dentro de estos escondites.
- Sistema de sospechas: Probablemente la funcionalidad más compleja del proyecto, construido para que los NPCs generen sospechas sobre los posibles culpables y, si es necesario, avisar a los guardias.

### 3.1.3 Objetivos secundarios

A pesar de que los objetivos secundarios no están directamente relacionados con el proyecto, sí que aportan a su desarrollo:

- Aprendizaje de nuevas herramientas: Familiarizarse con herramientas como Godot y Procreate, los cuales son fundamentales para el desarrollo de videojuegos y el arte digital.
- Aplicación de metodologías ágiles: Al ser un proyecto de software, decidí adentrarme en las metodologías como Kanban para gestionar el proyecto.
- Implementación de patrones de diseño: Como la carrera universitaria se centra en otros temas nunca se profundiza en la aplicación de patrones de diseño, por lo que este proyecto fue un método de entrada para explorar un uso real y para apoyarme en la construcción de un mejor código.

## 3.2 Límites del proyecto

Durante la fase de concepción del proyecto, se valoraron múltiples ideas que, si bien resultaban muy interesantes, finalmente se descartaron en la etapa de preproducción. Por razones como limitación de tiempo, complejidad técnica y viabilidad al realizar de forma individual.

Una de las ideas descartadas fue la implementación de un sistema de enfermedades. El objetivo de esta mecánica era forzar al jugador a considerar cuidadosamente dónde esconder los cadáveres de sus víctimas. Por ejemplo, si se lanzara en el pozo de la aldea – donde se va a abastecerse de agua – este podría descomponerse y causar una epidemia entre los habitantes. Esto afectaría al jugador por 2 motivos: los aldeanos morirían y el jugador no tendría como alimentarse, o incluso al morder a alguien infectado, el vampiro vería gravemente reducida movilidad. Aunque esta propuesta daba mayor profundidad a la jugabilidad, fue descartada debido a la carga de trabajo adicional que conllevaba.

Otra idea considerada fue introducir en un entorno online local. En este modo, los jugadores competirían en un mismo entorno, tratando de eliminar el mayor número de habitantes, a la vez que intentan sabotear a los demás. Por ejemplo, colocando cadáveres cerca de otros jugadores. Sin embargo, es una mecánica muy compleja para un primer proyecto de estas características, además de no estar alineada con la visión original del juego.

Finalmente, la propuesta más innovadora y compleja consistía en incorporar una inteligencia artificial generativa (IA) para que permitiera interactuar con los aldeanos. Permitiría al usuario manipular a los NPCs e incitarlos a desconfiar y confrontar a otros NPCs. No obstante, esta idea presentó numerosas dificultades. Por un lado, la integración y consumo de una IA generativa requiere de un pago y conexiones con APIs externas, posiblemente gratuitas, con grandes limitaciones, cuyos recursos asignados se agotarían rápidamente durante la fase de pruebas. Por otro lado, existen soluciones locales con modelos offline como DeepSeek, pero presentan un alto consumo de recursos que afectarían al rendimiento del juego, dificultando el acceso al título de gran parte de los usuarios. Además, el uso de IA generativa en tiempo real requiere de una implementación técnica, tanto por parte del desarrollador como del usuario, que excede el alcance del proyecto.

# CAPÍTULO 4: HIPÓTESIS DE TRABAJO

## 4.1 Motor de videojuegos

Como se ha comentado anteriormente, la elección del motor de desarrollo es una de las decisiones más relevantes del proyecto, ya que condiciona el flujo de trabajo, la estructura del código, el lenguaje a utilizar y la eficiencia del desarrollo. Al analizar las diferentes opciones, se optó por utilizar Godot, un motor de desarrollo de videojuegos de código abierto, con una comunidad altamente activa, sobre todo para desarrollo de videojuegos indie, y en constante evolución.

Una de las características más destacables de Godot es su paradigma de trabajo, que difiere de los modelos utilizados por motores como Unity y Unreal Engine. Godot introduce un sistema basado en escenas y nodos, donde cada nodo tiene una responsabilidad única, fomentando una arquitectura modular, legible y escalable. Dicho enfoque esta más alineado con los principios del diseño de software, como la separación de responsabilidades y el principio de composición sobre herencia.

En este sistema, cada nodo solo puede tener asignado 1 script, lo que obliga a estructurar de forma más ordenada la funcionalidad de cada componente. A su vez, una escena puede contener múltiples nodos de manera jerárquica para la construcción de estructuras complejas y reutilizables. Además, cualquier escena puede comportarse como un nodo dentro de otra escena, permitiendo una organización más flexible.


*Ilustración 4- Logo Godot Engine [40]*

## 4.2 Lenguajes de programación

Los lenguajes de programación están determinados en gran parte por herramientas como Godot. Este permite 2 lenguajes: GDScript nativo de la plataforma, y C#, incorporado debido a una colaboración con Microsoft quien financio su integración [41].

GDScript es un lenguaje interpretado y no tipado, con una sintaxis muy similar a Python. Especialmente diseñado para Godot ofreciendo una curva de aprendizaje más liviana que otros lenguajes [42].

Por otro lado, C# es un lenguaje compilado, fuertemente tipado y con una sintaxis estricta. Ampliamente utilizado en este entorno, gracias a su uso dentro de Unity, el cual forma parte del ecosistema de .Net de Microsoft.

A pesar de existir ciertas limitaciones asociadas al no ser el lenguaje principal del motor y tener un soporte comunitario menor, finalmente, me decanté por C# como lenguaje principal, debido a un mayor rendimiento al ser un lenguaje compilado, también ser un sistema altamente tipado, evitando problemas de tipo en tiempo de ejecución y aumentando la robustez. Además, sus sintaxis la considero más clara y estructurada que su contraparte, lo cual mejora la legibilidad del proyecto.

*Ilustración 5 - Logo C# [43]*

## 4.3 Herramientas CASE

Las herramientas CASE (Computer-Aided Software Engineering) son aplicaciones diseñadas para automatizar y facilitar tareas a lo largo del ciclo de vida del desarrollo de software. Editores de código, compiladores, sistemas de control de versiones y herramientas de modelado entran dentro de esta categoría [44].

### 4.3.1 Sistema de control de versiones

Una herramienta muy importante en un proyecto de desarrollo son los sistemas de control de versiones. Estos permiten gestionar los cambios en los archivos de un proyecto a lo largo de su desarrollo [45].

Un sistema de control de versiones es una aplicación que registra las modificaciones realizadas en los archivos fuente de un proyecto. Usualmente es usado por programadores para guardar los cambios dentro del código, sin embargo, puede usarse para el proceso de creación de animaciones, fondos, escenarios, etc. Permite visualizar el historial de cambios, comparar versiones, restauración de versiones y facilita el trabajo cooperativo.

Aunque existen alternativas como Subversion o Mercurial, me decanté por Git. Sistema altamente usado en la industria del desarrollo, gratuito, de código abierto y con una gran comunidad detrás. Una de las ventajas que me atrajo, fue su fácil integración con plataformas como GitLab y GitHub, este último usado durante el

proyecto, que permite alojar repositorios[10] remotos y coordinar tareas entre los miembros del equipo.

*Ilustración 6 - Logo de Git [46]*

### 4.3.2 Editor de textos

A pesar de que Godot cuenta con un editor de texto integrado, este está principalmente diseñado para su uso con GDScript. Por lo que se recomienda el uso de un editor externo para trabajar con C# de manera más cómoda y eficiente.

El editor más utilizado es Visual Studio Code (VSCode), un editor de código gratuito y de código abierto utilizado en múltiples ámbitos del desarrollo de software. Además del soporte activo de la comunidad, la empresa Microsoft también contribuye a su desarrollo y mantenimiento [47].

Para integrar correctamente VSCode con Godot, es necesario realizar algunas configuraciones:

- Dentro del motor Godot, se debe activar el uso de un editor externo dentro de su configuración
- En VSCode, es recomendable instalar algunas extensiones para facilitar el desarrollo con C#. Entre las más necesarias tenemos:
    - C# Dev Kit: Proporciona la integración con la API de .NET
    - C# Tools for Godot: Permite funcionalidades de depuración y utilidades de trabajo con el motor

[10] Lugar donde se guarda algo

Cabe mencionar que existen otras extensiones como Godot Docs for C# y Godot Snippets for C#, que permiten acceder a la documentación desde el propio editor y activar el autocompletado de las funciones específicas del motor, respectivamente.


*Ilustración 7 - Logo de Visual Studio Code [48]*

### 4.3.3 Herramientas de Diagramación y documentación

Durante el ciclo de vida de este proyecto se han utilizado principalmente 2 herramientas para la creación de diagramas y la gestión de documentación: Obsidian y Lucidchart.

Lucidchart ha sido empleada para la elaboración de diagramas de flujo con el fin de ayudar a estructurar el desarrollo del videojuego. Esta tiene un interfaz amigable e intuitiva, permitiendo la creación de diagramas de forma rápida [49].


*Ilustración 8 - Logo de Lucidchart [50]*

Por otro lado, Obsidian ha jugado un papel más central en todo el proceso. Además de utilizarse para creación de diagramas de comportamiento y de jerarquía de clases, ha servido como repositorio de información y documentación [51]. Obsidian recoge elementos como:

- Información sobre las referencias e inspiraciones
- La investigación mitológica vampírica y de elementos narrativos
- El GDD completo
- La planificación de tareas

Obsidian permite mantener una documentación interconectada y navegable. Además, mediante extensiones para permitir el uso de Canvas y tablas de Kanban, permitió una planificación visual más sencilla y agradable. Aunque también se utilizó GanttProject.

GanttProject es una herramienta de código abierto y gratuita diseñada para la gestión y planificación de proyectos. Se basa en la creación de diagramas de Gantt, los cuales permiten dividir el proyecto en tareas individuales, establecerles una duración, definir relaciones de dependencia y la asignación de diferentes miembros del equipo [52]. Además de permitir un seguimiento del progreso de cada actividad y la generación de un diagrama PERT[11].

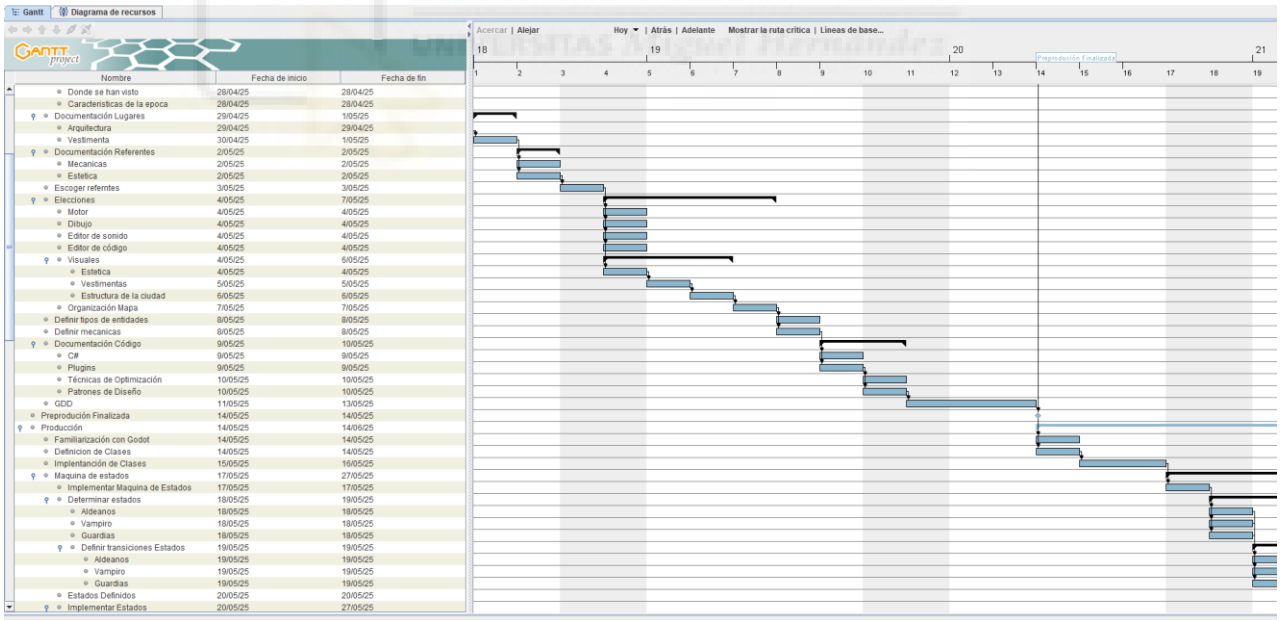


*Ilustración 9 - Aplicación GanttProject*

[11] Program Evaluation and Review Technique ó Técnica de Evaluación y Revisión de Programas es una técnica usada en la gestión de proyectos, para analizar y representar visualmente las tareas de un proyecto

En el diagrama mostrado podemos observar la planificación de este proyecto, el cual como hemos comentado está dividido en 3 fases principales:

| Nombre | Tiempo estimado |
|---|---|
| Preproducción | 20 |
| Producción | 32 |
| Postproducción | 8 |

*Tabla 3 - Estimación de tiempo para fases del proyecto*

Por otro lado, los assets son aquellos recursos utilizados en el desarrollo de videojuegos, tales como sonidos, imágenes, animaciones, personajes y demás elementos visuales. Para la creación y recopilación de estos recursos se han utilizados 2 herramientas principales, así como webs de distribución de assets gratuitos.

En cuanto a la edición y tratamiento de sonido, he utilizado Audacity, programa gratuito y de código abierto, que permite cortar, mejorar y adaptar pistas de audio.

Para la creación de fondos, personajes y animaciones 2D, la herramienta de pago usada fue Procreate, debido a su versatilidad para la ilustración digital y la gran comunidad de artistas que usan esta aplicación a diario.

Además de crear los recursos de forma manual, también he recurrido a plataformas como Itch.io, OpenGameArte y Freesound para obtener assets de forma gratuita, tratando siempre de mantener una coherencia con los recursos creados.

# CAPÍTULO 5: METODOLOGÍA Y RESULTADOS

## 5.1 Planificación del proyecto

Para este proyecto, se aplicó un ciclo de vida ágil. Dentro de este enfoque se pueden encontrar metodologías como Kanban, ya mencionada anteriormente, Scrum y Extreme Programming (XP). La idea principal de este tipo de ciclo es dividir el trabajo en pequeñas iteraciones llamadas sprints (ciclos), con el fin de realizar entregas frecuentes, revisiones continuas y la posibilidad de facilitar los posibles ajustes el desarrollo [53][54].

Aunque el desarrollo ágil está pensado principalmente para el trabajo en equipo, al tratarte de un proyecto individual, no ha sido difícil adaptarlo. Podemos encontrar 2 importantes ventajas: facilita el trabajo en equipo y permite adaptarse fácilmente al feedback recibido por testers y jugadores.

Estas metodologías son muy usadas dentro de la industria del videojuego, especialmente en empresas medianas y grandes. Para estructurar la planificación y visualizar sus componentes se emplearon varias herramientas:

- Obisidian con 2 plugin para Kanban y otro para Canvas, para mapas mentales y esquemas visuales
- Lucidchart para la creación de diagramas de flujo para representar funcionalidades, algoritmos complejos o interacciones
- GanttProject, mencionado anteriormente

Con estas herramientas se crearon algunos diagramas de componentes, clases y objetos, de secuencia y estados para tener una organización del proyecto.

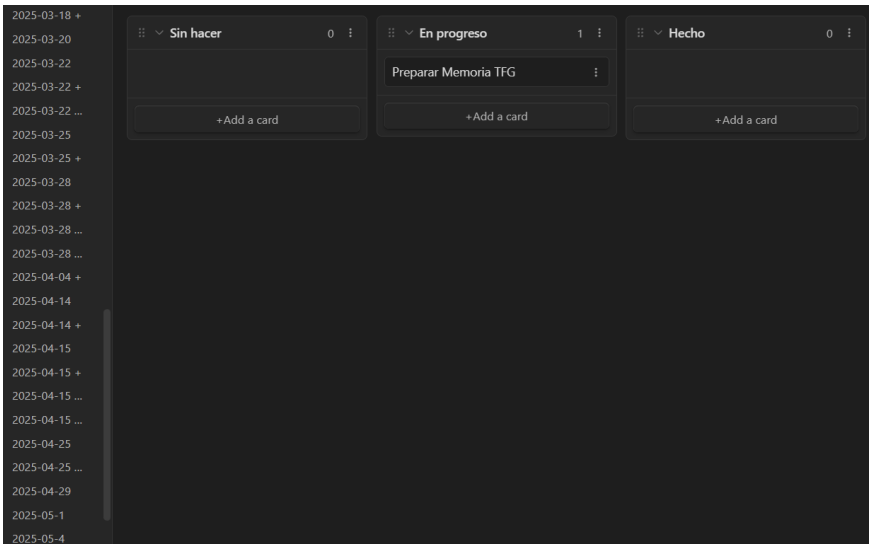


*Ilustración 10 - Sistema de Kanban utilizado en Obsidian*

## 5.2 Captura de requisitos

Dentro de este apartado se mostrarán los diagramas realizados durante el proyecto, los casos de uso, los requerimientos de hardware del videojuego, así como se hablará del público objetivo que se quiere atraer [55].

### 5.2.1 Casos de uso

Los casos de uso son la descripción detallada de tareas y funcionalidades que un usuario puede realizar en una aplicación o programa, con información como los autores, los pasos que se realizan, pre-postcondiciones para realizarlos, etc.

| | |
|---|---|
| **C.U.1** | Iniciar Partida |
| **Actores** | Jugador |
| **Descripción** | Comenzar a Jugar |
| **Dependencias** | |

| | |
|---|---|
| **Precondición** | Ejecutar el juego |
| **Flujo estándar** | 1. Inicia el menú del juego<br>2. Pulsa el botón de "Nueva partida"<br>3. Sistema carga la escena principal |
| **Flujo alternativo** | |
| **Postcondición** | |
| **Excepciones** | El sistema no tiene los recursos necesarios para iniciar el juego o partida. |
| **Comentarios** | |

*Tabla 4 - Caso de Uso: Iniciar Partida*



| | |
|---|---|
| **C.U.2** | Salir del juego |
| **Actores** | Cerrar el juego |
| **Descripción** | Jugador |
| **Dependencias** | |
| **Precondición** | Abrir el juego<br><br>Alt. → C.U: 1 |
| **Flujo estándar** | 1. Inicia el menú del juego<br>1. Pulsa el botón "Salir"<br>2. Se muestra una advertencia de que los datos no guardados se perderán<br>3. Jugador pulsa que está de acuerdo |

| | |
|---|---|
| | 4. Sistema cierra programa |
| **Flujo alternativo** | |
| **Postcondición** | |
| **Excepciones** | Jugador pulsa en que no está de acuerdo durante la advertencia, por lo que no sucede nada |
| **Comentarios** | "Alt." se refiere a una precondición alternativa |

*Tabla 5 - Caso de Uso: Salir del Juego*

| | |
|---|---|
| **C.U.3** | Volver al menú principal |
| **Actores** | Jugador |
| **Descripción** | Retornar a la pantalla inicial del videojuego |
| **Dependencias** | |
| **Precondición** | C.U: 1 |
| **Flujo estándar** | 1. Inicia el menú del juego<br>2. Pulsa el botón "Volver al menú principal"<br>3. Se muestra una advertencia de que los datos no guardados se perderán<br>4. Jugador pulsa que está de acuerdo<br>5. Videojuego carga la pantalla inicial |
| **Postcondición** | |
| **Excepciones** | Jugador pulsa en que no está de acuerdo durante la advertencia, por lo que no sucede nada |

| | |
|---|---|
| **Comentarios** | |

*Tabla 6 - Caso de Uso: Volver al menú principal*

| | |
|---|---|
| **C.U.4** | Movimiento Izquierda-Derecha |
| **Actores** | Jugador |
| **Descripción** | Jugador hace que el vampiro se desplace por el mapa |
| **Dependencias** | |
| **Precondición** | C.U: 1 |
| **Flujo estándar** | 1. Jugador pulsa tecla "A" o "D"<br>2. Vampiro se mueve en alguna de esas direcciones |
| **Flujo alternativo** | 1. Jugador pulsa tecla "←" o "→"<br>2. Vampiro se mueve en alguna de esas direcciones |
| **Postcondición** | |
| **Excepciones** | |
| **Comentarios** | Las teclas A/← y D/→ indican la dirección del movimiento. A para la izquierda y D para la derecha. |

*Tabla 7 - Caso de Uso: Movimiento Izquierda-Derecha*

| | |
|---|---|
| **C.U.5** | Matar Aldeano |
| **Actores** | Jugador |
| **Descripción** | Jugador hace que el vampiro ataque a un aldeano |

| | |
|---|---|
| **Dependencias** | |
| **Precondición** | C.U: 1<br><br>Tener en rango a un Aldeano |
| **Flujo estándar** | 1. Pulsa tecla F<br>2. Vampiro ataca Aldeano<br>3. Aldeano muere<br>4. Se inicia cooldown de ataque |
| **Flujo alternativo** | |
| **Postcondición** | |
| **Excepciones** | Tiene activado el cooldown de ataque activado |
| **Comentarios** | Cooldown o enfriamiento es una restricción de tiempo para la ejecución de una acción |

*Tabla 8 - Caso de Uso: Matar Aldeano*

| | |
|---|---|
| **C.U.6** | Arrastrar cadáver |
| **Actores** | Jugador |
| **Descripción** | Mover de lugar el cadáver de un aldeano |
| **Dependencias** | C.U:4 |
| **Precondición** | C.U: 5<br><br>Estar lo suficientemente cerca del cadáver |
| **Flujo estándar** | 1. Pulsa la tecla E<br>2. Jugador mueve al vampiro |

| | |
|---|---|
| | 3. El cadáver se mueve con él |
| **Flujo alternativo** | |
| **Postcondición** | No debe suceder C.U:10 |
| **Excepciones** | Sucede C.U: 10 |
| **Comentarios** | |

*Tabla 9 - Caso de Uso: Arrastrar cadáver*

| | |
|---|---|
| **C.U.7** | Esconder Cadáver Aldeano |
| **Actores** | Jugador |
| **Descripción** | Luego de arrastrar el cadáver hasta un lugar específico, como puede ser el matorral, esconde dentro de este |
| **Dependencias** | |
| **Precondición** | C.U:6<br><br>Esta dentro del rango de un lugar para esconder cadáveres |
| **Flujo estándar** | 1. Pulsa la tecla E<br>2. El cadáver se asigna al lugar<br>3. El cadáver desaparece |
| **Flujo alternativo** | |
| **Postcondición** | No debo suceder C.U: 10 |
| **Excepciones** | Sucede C.U: 10 |

| | |
|---|---|
| **Comentarios** | |

*Tabla 10 - Caso de Uso: Esconder Cadáver*

| | |
|---|---|
| **C.U.8** | Modificar Ajustes |
| **Actores** | Jugador |
| **Descripción** | Cambiar elementos como el sonido, la resolución, la dificultad, etc. |
| **Dependencias** | |
| **Precondición** | |
| **Flujo estándar** | 1. Pulsa tecla Esc<br>2. Pulsa el botón "Ajustes"<br>3. Cambia el/los ajuste(s) que necesita<br>4. Pulsa F para aplicar ajustes<br>5. Sistema recopila los ajustes modificados |
| **Flujo alternativo** | 1. Pulsa tecla Esc<br>2. Pulsa el botón "Ajustes"<br>3. Cambia el/los ajuste(s) que necesita<br>4. Pulsa botón "Aplicar" para aplicar ajustes<br>5. Sistema recopila los ajustes modificados |
| **Postcondición** | |
| **Excepciones** | Se pulsa la tecla Esc o el botón "Cancelar" |

| | |
|---|---|
| **Comentarios** | Dentro de la ventana de ajustes se puede pulsar de nuevo el botón Esc para salir de dicha ventana cancelando todas las modificaciones en el proceso. |

*Tabla 11 - Caso de Uso: Modificar Ajustes*

| | |
|---|---|
| **C.U.9** | Vampiro elimina a todos los aldeanos |
| **Actores** | Jugador |
| **Descripción** | Vampiro mata a cada uno de los aldeanos de la aldea |
| **Dependencias** | C.U.1, 5<br><br>Alt. C.U: 21, 5 |
| **Precondición** | |
| **Flujo estándar** | 1. Jugador mata al último aldeano del pueblo<br>2. Sistema detecta que ha eliminado a todos los aldeanos<br>3. Sistema muestra un mensaje de que terminó la partida y ganó<br>4. Sistema muestra estadísticas de la partida |
| **Flujo alternativo** | |
| **Postcondición** | |
| **Excepciones** | |
| **Comentarios** | La información de las estadísticas sería:<br><br>• Aldeanos muertos |

| | |
|---|---|
| | • La medía de sospecha que global hacia este con la que finalizó<br>• El tiempo que tardó en finalizar |

*Tabla 12- Caso de Uso: Vampiro elimina a todos los aldeanos*

| | |
|---|---|
| **C.U.10** | Vampiro es detectado |
| **Actores** | Jugador, NPC |
| **Descripción** | Cuando los NPCs saben que el jugador es un Vampiro |
| **Dependencias** | C.U: 5 |
| **Precondición** | Que no suceda C.U. 9 |
| **Flujo estándar** | 1. Aldeano encuentra cadáver<br>2. La última entidad que vio es el vampiro<br>3. Aumenta la sospecha hacía el vampiro<br>4. Supera el umbral de sospecha<br>5. Aldeano avisa a los guardias<br>6. Guardias atacan vampiro<br>7. Vampiro muere<br>8. Jugador pierde |
| **Flujo alternativo** | |
| **Postcondición** | |
| **Excepciones** | |
| **Comentarios** | |

*Tabla 13 - Caso de Uso: Vampiro es detectado*

| | |
|---|---|
| **C.U.11** | Lanzar Cadáver al Rio |
| **Actores** | Jugador |
| **Descripción** | Luego de arrastrar el cadáver hacía un puente, podemos lanzarlo desde este, para que se lo lleve la corriente |
| **Dependencias** | |
| **Precondición** | C.U: 6 |
| **Flujo estándar** | 1. Jugador arrastra cadáver<br>2. Se hacer a un puente<br>3. Lanza el cadáver desde el puente<br>4. Cadáver es llevado por la corriente |
| **Flujo alternativo** | |
| **Postcondición** | |
| **Excepciones** | |
| **Comentarios** | Este caso es de un solo uso, ya que después de un tiempo de lanzar el cadáver, guardias se postarán en a los lados del puente |

*Tabla 14 - Caso de Uso: Lanzar Cadáver al rio*

| | |
|---|---|
| **C.U.12** | Vampiro Es Descubierto Alimentándose |
| **Actores** | Jugador |
| **Descripción** | Durante el caso de uso 5, los aldeanos siguen moviéndose por el mapa y pueden encontrarte |

| | |
|---|---|
| | alimentándote del NPC, lo cual hará que este se asuste y llame a los guardias |
| **Dependencias** | C.U: 5 |
| **Precondición** | |
| **Flujo estándar** | 1. Jugador ataca aldeano 1<br>2. Aldeano 2 encuentra al vampiro matando<br>3. Aldeano 2 grita por ayuda<br>4. Sucede C.U: 10 |
| **Flujo alternativo** | |
| **Postcondición** | C.U:10 |
| **Excepciones** | Sucede C.U: 9 |
| **Comentarios** | Si da la casualidad de que el vampiro mata al último aldeano delante de un guardia, el guardia no lo detectaría a tiempo y ganaría la partida |

*Tabla 15 - Caso de Uso: Vampiro es descubierto alimentándose*

| | |
|---|---|
| **C.U.13** | Aldeano encuentra Cadáver |
| **Actores** | Aldeano |
| **Descripción** | Uno de los NPCs que se mueven por el mapa, encuentra el cadáver de otro |
| **Dependencias** | Alt. C.U: 7 |
| **Precondición** | C.U: 5 |
| **Flujo estándar** | 1. Aldeano camina por el mapa |

| | |
|---|---|
| | 2. Aldeano encuentra cadáver en el suelo<br>3. Aldeano grita por ayuda (Guardias)<br>4. Aldeano recopila información sobre el último Aldeano que ha visto<br>5. Aumenta la sospecha de este |
| **Flujo alternativo** | 1. Aldeano camina por el mapa<br>2. Encuentra un lugar donde se esconden cadáveres<br>3. Aldeano revisa el lugar<br>4. Aldeano detecta el cadáver<br>5. C.U: 15<br>6. Aldeano recopila información sobre la última entidad que ha visto<br>7. Aumenta la sospecha de este |
| **Postcondición** | |
| **Excepciones** | |
| **Comentarios** | Dentro del videojuego podemos encontrar 2 ocasiones en las que el Aldeano se puede encontrar un cadáver, el flujo estándar se refiere a cuando se lo encuentra en medio del camino, y el alternativo se refiere al momento de buscar en un escondite determinado para esconder esos cadáveres. Aunque este último se le añadió "una ruleta" para que no siempre busque. |

*Tabla 16 - Caso de Uso: Aldeano encuentra cadáver*

| | |
|---|---|
| **C.U.14** | Aldeano acusa a otro Aldeano |
| **Actores** | NPCs |

| | |
|---|---|
| **Descripción** | Luego de varias muertes y varias suposiciones por parte de los aldeanos, estos pueden aumentar sus sospechas hacia otros, haciendo que este les diga a los guardias que él es el asesino |
| **Dependencias** | C.U: 13 |
| **Precondición** | Debe de haber sucedido C.U: 5 en varias ocasiones |
| **Flujo estándar** | 1. Aldeano encuentra cadáver<br>2. Aumenta sospecha<br>3. Nivel de sospecha supera umbral de sospecha<br>4. Aldeano decide avisar a los guardias |
| **Flujo alternativo** | 1. Aldeano habla con otro aldeano<br>2. Se comparten sus sospechas<br>3. Aumenta las sospechas sobre esos aldeanos<br>4. Alguno supera el umbral de sospecha<br>5. Aldeano decide avisar a los guardias |
| **Postcondición** | |
| **Excepciones** | Alt. → El Aldeano no se influencia de otros |
| **Comentarios** | Dentro de la clase Aldeano, encontramos entre otras cosas, un atributo "Personalidad". Este se usa para darle más vida a los NPCs haciendo que estos puedan o no compartir sus sospechas o aumentarlas o no en función de lo que les dicen otros |

*Tabla 17 - Caso de Uso: Aldeano acusa a otro Aldeano*

| | |
|---|---|
| **C.U.15** | Aldeano Alerta Guardia |
| **Actores** | Aldeano, Guardia |
| **Descripción** | Aldeano luego de encontrar un cadáver llama al guardia para que se lo lleven |
| **Dependencias** | C.U: 5 |
| **Precondición** | C.U: 13 o 10 |
| **Flujo estándar** | 1. Aldeano busca los guardias más cercanos<br>2. Aldeano comunica el suceso<br>3. Guardias se dirigen a la posición |
| **Flujo alternativo** | |
| **Postcondición** | |
| **Excepciones** | |
| **Comentarios** | Hay 2 casos en los que el aldeano puede avisar a un guardia de un suceso:<br>• C.U: 13<br>• C.U: 5<br>• C.U: 14 |

*Tabla 18 - Caso de Uso: Aldeano alerta a Guardia*

| | |
|---|---|
| **C.U.16** | Sistema Actualiza Sospechas |
| **Actores** | Aldeano |

| | |
|---|---|
| **Descripción** | Aumenta o disminuye la cantidad de sospecha hacia una entidad, ya sea Aldeano o Vampiro |
| **Dependencias** | |
| **Precondición** | C.U: 13 o Aldeano habla con otro (solo si es influenciable) |
| **Flujo estándar** | 1. Aldeano piensa en quien es el último que ha pasado cerca de la zona (el último que ha visto)<br>2. Aumenta la sospecha hacia este |
| **Flujo alternativo** | 1. Aldeano escucha las sospechas de otro<br>2. Aldeano aumenta sospecha de estos<br>3. Si sospecha igual que él, reduce la sospecha hasta este |
| **Postcondición** | |
| **Excepciones** | Aldeano no es fácilmente influenciable por otros aldeanos y sus sospechas |
| **Comentarios** | |

*Tabla 19 - Caso de Uso: Sistema actualiza sospechas*

| | |
|---|---|
| **C.U.17** | Vampiro deja un cadáver para generar sospechas entre aldeanos |
| **Actores** | Jugador |

| | |
|---|---|
| **Descripción** | Al matar un aldeano, el vampiro podría dejarlo en medio de la calle o arrastrarlo hasta un punto estratégico para manipular a los aldeanos |
| **Dependencias** | C.U: 5, 6 |
| **Precondición** | |
| **Flujo estándar** | 1. C.U: 5<br>2. Jugador se aleja de la escena<br>3. Jugador se esconde |
| **Flujo alternativo** | 1. C.U: 6<br>2. Jugador se aleja de la escena<br>3. Jugador se esconde |
| **Postcondición** | C.U: 13 |
| **Excepciones** | Sucede C.U.12 o C.U.9 |
| **Comentarios** | |

*Tabla 20 - Caso de Uso: Vampiro deja un cadáver para generar sospechas entre aldeanos*

| | |
|---|---|
| **C.U.18** | Guardia Ataca Vampiro |
| **Actores** | Guardia, Jugador |
| **Descripción** | Luego de ser detectado, los guardias se dirigen hacia el vampiro para atacarlo y matarlo |
| **Dependencias** | |
| **Precondición** | C.U: 10 |

| | |
|---|---|
| **Flujo estándar** | 1. Guardia se acerca al vampiro<br>2. Guardia ataca vampiro<br>3. Guarda mata vampiro<br>4. Vampiro muere<br>5. Jugador pierde la partida<br>6. Se muestra una pantalla con las estadísticas de la partida |
| **Flujo alternativo** | |
| **Postcondición** | |
| **Excepciones** | |
| **Comentarios** | La información de las estadísticas sería:<br>• Aldeanos muertos<br>• La medía de sospecha que global hacia este con la que finalizó<br>• El tiempo que tardó en finalizar |

*Tabla 21 - Caso de Uso: Guardia Ataca Vampiro*

| | |
|---|---|
| **C.U.19** | Consultar información Aldeano |
| **Actores** | Jugador, Aldeano |
| **Descripción** | Mostrar la información sobre las sospechas de un NPC |
| **Dependencias** | C.U. 1 |
| **Precondición** | |

| | |
|---|---|
| **Flujo estándar** | 1. Jugador pulsa sobre un aldeano<br>2. Se muestra una tabla con información de este |
| **Flujo alternativo** | |
| **Postcondición** | |
| **Excepciones** | |
| **Comentarios** | La información mostrada dentro de esta ventana será:<br>• Las 3 mayores sospechas de ese aldeano<br>• Su personalidad |

*Tabla 22 - Caso de Uso: Consultar información aldeano*

| | |
|---|---|
| **C.U.20** | Guardar Partida |
| **Actores** | Jugador |
| **Descripción** | Guardar la información actual de la partida |
| **Dependencias** | C.U. 1 |
| **Precondición** | |
| **Flujo estándar** | 1. Pulsa la tecla Esc<br>2. Se muestra el menú de opciones<br>3. Se pulsa en el botón de guardar partida<br>4. Sistema guarda la información mínima necesaria |
| **Flujo alternativo** | |
| **Postcondición** | |

| | |
|---|---|
| **Excepciones** | Fallo en el guardado de información |
| **Comentarios** | El guardar información busca retener la información mínima necesaria para que, al momento de cargar la partida, el jugador piense que la partida sigue donde lo dejó. Entre los datos que se guardarán tenemos:<br>• Colocando los aldeanos vivos<br>• Posiciones de guardias<br>• Sospechas de esos aldeanos |

*Tabla 23 - Caso de Uso: Guardar partida*

| | |
|---|---|
| **C.U.21** | Cargar Partida |
| **Actores** | Jugador |
| **Descripción** | Cargar la partida guardada desde el menú de inicial del videojuego |
| **Dependencias** | |
| **Precondición** | C.U. 20 |
| **Flujo estándar** | 1. Pulsa el botón "Cagar Partida"<br>2. Sistema recoge toda la información de la partida guardada<br>3. Inicia la partida con dicha información |
| **Flujo alternativo** | |
| **Postcondición** | |
| **Excepciones** | No hay partida guardada |

| | |
|---|---|
| **Comentarios** | Con la información guardada se cargan los datos dentro del sistema:<br>• Colocando los aldeanos vivos<br>• Posiciones de guardias<br>• Sospechas de esos aldeanos |

*Tabla 24 - Caso de Uso: Cargar partida*

| | |
|---|---|
| **C.U.22** | Vampiro ataca Guardia |
| **Actores** | Jugador |
| **Descripción** | Vampiro intenta matar a un guardia |
| **Dependencias** | C.U. 1 |
| **Precondición** | |
| **Flujo estándar** | 1. Guardia se protege<br>2. Guardia ataca<br>3. Vampiro muere<br>4. Jugador pierde la partida<br>5. Se muestra pantalla con las estadísticas de la partida |
| **Flujo alternativo** | |
| **Postcondición** | |
| **Excepciones** | |
| **Comentarios** | |

*Tabla 25 - Caso de Uso: Vampiro ataca Guardia*

| | |
|---|---|
| **C.U.23** | Aumento de Guardias en Aldea |
| **Actores** | Guardias |
| **Descripción** | El número de guardias de la aldea se ve incrementado |
| **Dependencias** | |
| **Precondición** | C.U.13 (más de 2 veces)<br><br>Alt. C.U.11 |
| **Flujo estándar** | 1. Aparecen nuevos guardias por el mapa |
| **Flujo alternativo** | 1. Aparecen nuevos guardias alrededor del puente |
| **Postcondición** | |
| **Excepciones** | No encuentran cadáveres |
| **Comentarios** | El puente es para dar a entender que se encontró el cadáver que se lanzó al rio |

*Tabla 26 - Caso de Uso: Aumento de guardias en aldea*

| | |
|---|---|
| **C.U.24** | Consultar Información Global |
| **Actores** | Jugador |
| **Descripción** | Mostrar información general de la partida |
| **Dependencias** | C.U.1 |
| **Precondición** | |

| | |
|---|---|
| **Flujo estándar** | 1. Pulsa la tecla Esc<br>2. Se muestra la pantalla de opciones con una tabla de información de la partida |
| **Flujo alternativo** | |
| **Postcondición** | |
| **Excepciones** | |
| **Comentarios** | La información que se mostraría es:<br>• Los aldeanos asesinados<br>• Los días que llevas<br>• Los aldeanos vivos |

*Tabla 27 - Caso de Uso: Consultar información global*

| | |
|---|---|
| **C.U.25** | Dejar de arrastrar un cadáver |
| **Actores** | Jugador |
| **Descripción** | Dejar en el suelo el cadáver que se estaba arrastrando |
| **Dependencias** | C.U.6 |
| **Precondición** | |
| **Flujo estándar** | 1. Pulsa la tecla E<br>2. Vampiro comienza a moverse libremente<br>3. Cadáver queda parado en el lugar |
| **Flujo alternativo** | |
| **Postcondición** | |

| | |
|---|---|
| **Excepciones** | |
| **Comentarios** | |

*Tabla 28 - Caso de Uso: Dejar de arrastrar un cadáver*

| | |
|---|---|
| **C.U.26** | Reiniciar Partida |
| **Actores** | Jugador |
| **Descripción** | Volver a comenzar una partida nueva |
| **Dependencias** | C.U.1 |
| **Precondición** | |
| **Flujo estándar** | 1. Se pulsa la tecla Esc<br>2. Se muestra el menú de opciones<br>3. Se pulsa el botón "Reiniciar"<br>4. Sistema carga la partida desde 0 |
| **Flujo alternativo** | |
| **Postcondición** | |
| **Excepciones** | Vuelve a pulsar Esc |
| **Comentarios** | |

*Tabla 29 - Caso de Uso: Reiniciar partida*

| | |
|---|---|
| **C.U.27** | Cerrar Consultar información Aldeano |
| **Actores** | Jugador |

| | |
|---|---|
| **Descripción** | Cerrar la ventana de información sobre el aldeano |
| **Dependencias** | |
| **Precondición** | C.U. 19 |
| **Flujo estándar** | 1. Jugador pulsa en cualquier lugar fuera del aldeano<br>2. La tabla de información se cierra |
| **Flujo alternativo** | |
| **Postcondición** | |
| **Excepciones** | No se está mostrando ninguna tabla de información |
| **Comentarios** | |

*Tabla 30 - Caso de Uso: Cerrar consultar información aldeano*



| | |
|---|---|
| **C.U.28** | Salir del menú de opciones |
| **Actores** | Jugador |
| **Descripción** | Cerrar el menú de opciones |
| **Dependencias** | |
| **Precondición** | Se pulsa Esc |
| **Flujo estándar** | 1. Se pulsa Esc<br>2. Menú de opciones desaparece |
| **Flujo alternativo** | |

| | |
|---|---|
| **Postcondición** | |
| **Excepciones** | |
| **Comentarios** | |

*Tabla 31 - Caso de Uso:  Salir del menú de opciones*

### 5.2.2 Diagrama de clases y objetos

Los diagramas de clases y de objetos son una representación gráfica de la estructura interna de un sistema. Muestra: clases, atributos, métodos y relaciones entre ellas [56].

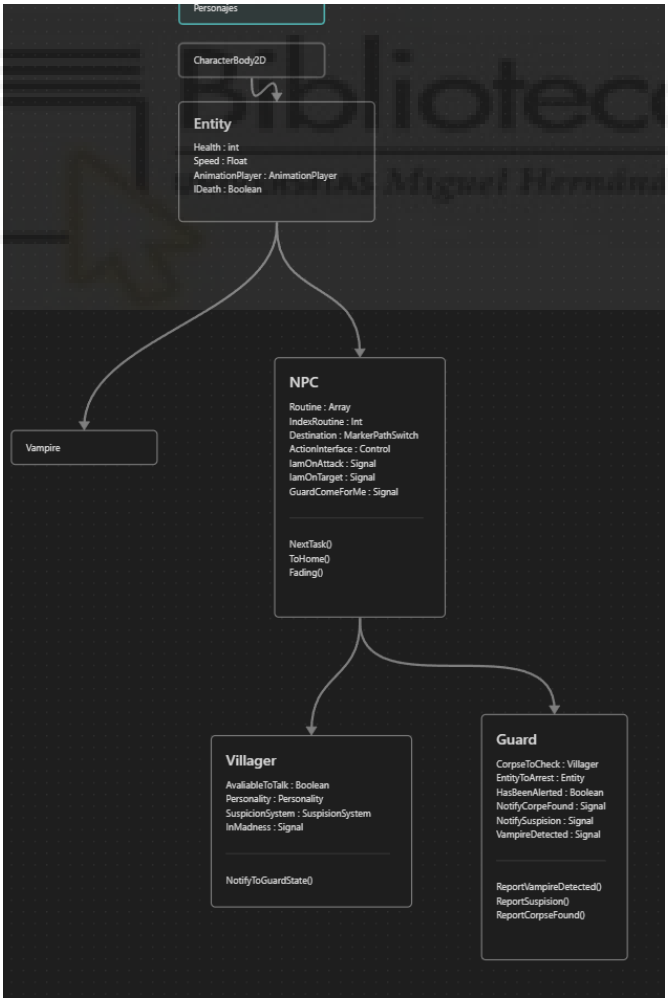


*Ilustración 11 - Herencia de los personajes del videojuego*

En la imagen se representa la jerarquía de herencia entre distintas clases de personajes del juego. La clase raíz de todas ellas es CharacterBody2D, una clase nativa de Godot, la cual se suele utilizar para definir personajes y NPCs integrándolos en el sistema de físicas del motor 2D.

Además, entre los atributos encontramos una clase personalizada llamada MarkerPathSwitch. Tiene 2 funciones esenciales:

- Marcar puntos relevantes del mapa
- Funcionar como interconectares entre los caminos prefijados del mapa

Por último, se destaca otro tipo de atributo, Signal, otra clase nativa de Godot. Esta permite la comunicación entre nodos mediante un patrón de eventos con el fin de facilitar una arquitectura desacoplada entre nodos.

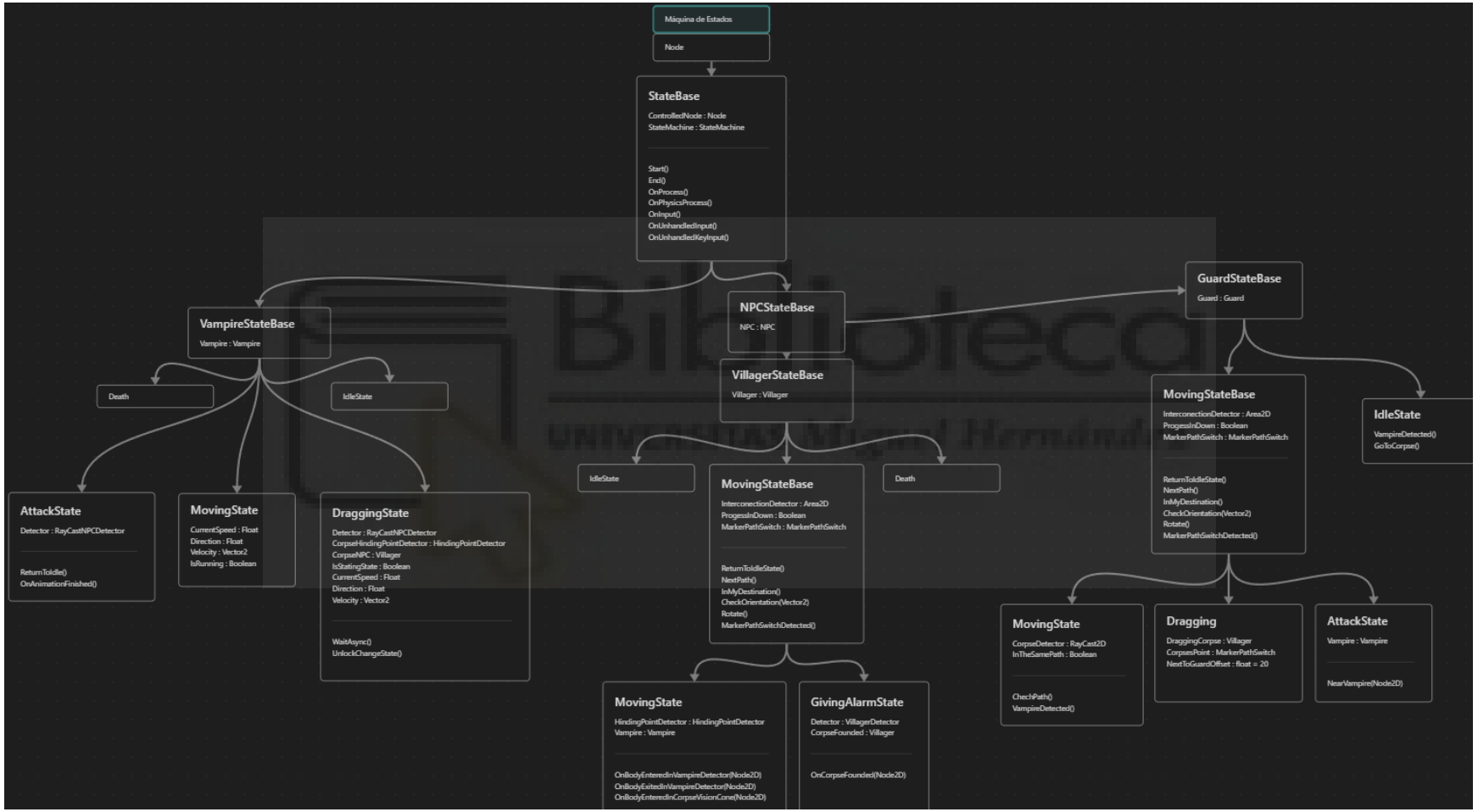


*Ilustración 12- Diagrama de clases de los estados de los personajes*

En el diagrama anterior podemos observar las jerarquías de herencia entre las diferentes clases que representan los estados de los personajes del videojuego. Esta organización se basa en el patrón de diseño conocido como Máquina de Estados, mencionado anteriormente.

Como clase común o raíz encontramos StateBase, una clase abstracta de la que heredan todos los estados del juego. Define los métodos fundamentales del ecosistema de Godot y de la propia definición de máquina de estados. Tenemos:

- Start → Método que se realizará siempre que se inicie ese estado en el objeto
- End → Método que se realizará siempre que se finalice ese estado en el objeto
- OnProcess → Método que se realizará en cada fotograma que dicho estado este activo
- OnPhysicProcess → Método que se realizará durante el procesamiento de las físicas del motor
- OnInput – OnUnhandledInput – OnUnhandledKeyInput → Métodos que se ejecutarán cuando el usuario realice un Input, como puede ser pulsar una tecla o pulsar el ratón.

Además de referencias a la máquina de estados y al nodo que está ejecutando ese estado.

Al igual que en el diagrama de herencia de los personajes que hemos mostrado, de StateBase derivan 3 tipos principales de clases y objetos:

- VampireStateBase → Para los estados del vampiro, siendo el jugador
- VillagerStateBase → Para los estados de los aldeanos, siendo parte de los NPCs
- GuardStateBase → Para los estados de los guardias, también parte de los NPCs

### 5.2.3 Diagrama de estados

Los diagramas de estado muestran una representación de los diferentes estados por los que puede pasar un sistema u objeto, además de sus transiciones y eventos que los activan [57].

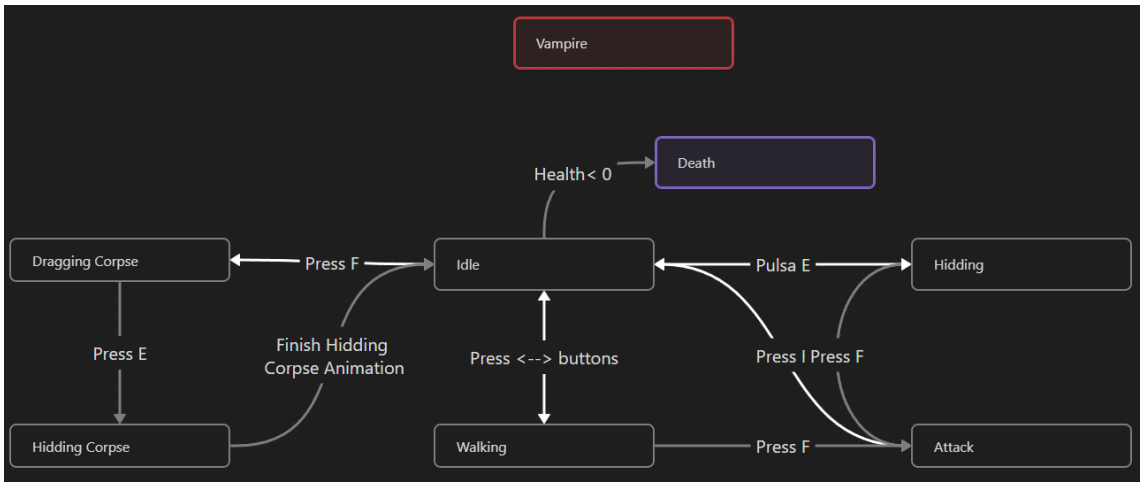


*Ilustración 13 - Diagrama de estados del vampiro*

La imagen muestra la estructura del sistema de estados finitos implementado para el vampiro, el personaje principal del juego. Como hemos comentado anteriormente para todas las entidades se aplica la norma de que solo se puede estar en un único estado. Y consta realmente de 6 estados principales: Parado (Idle), En movimiento (Walking), Escondido (Hiding), Atacando (Attack), Arrastrando (Dragging). Realmente Dragging podríamos dividirlos en 2 como se muestra en el diagrama, ya que al estar cerca de un punto donde se puede esconder el cadáver esta comprueba si estas en rango de uno de estos puestos y si lo estas se esconden este pulsando la tecla pertinente.

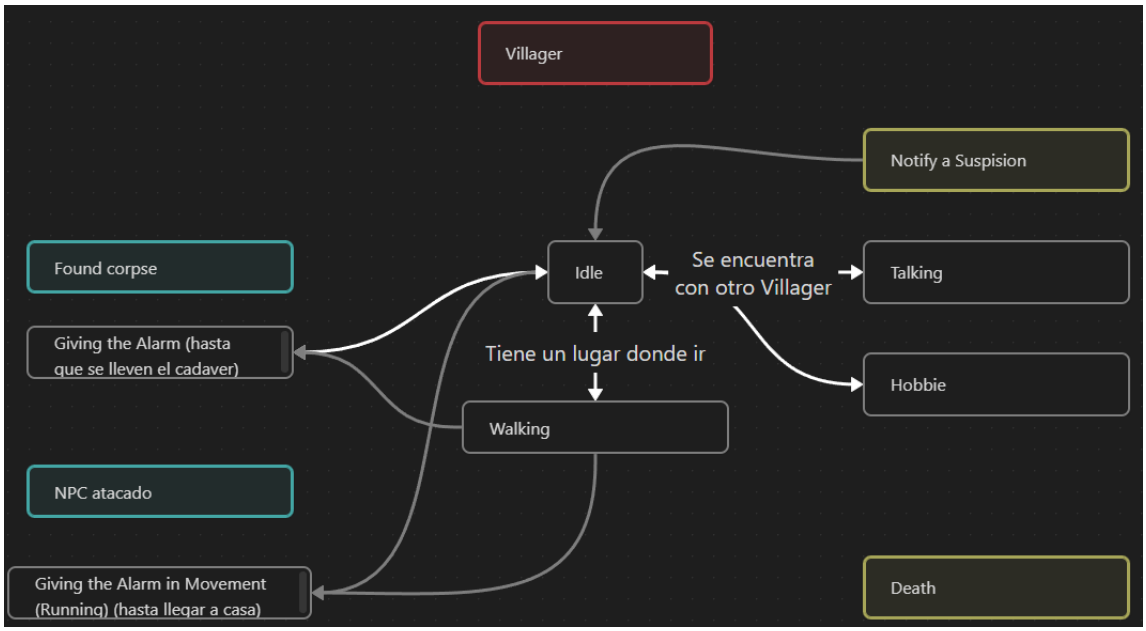


*Ilustración 14- Diagrama de estados Villager*

En la imagen se representa la máquina de estados aplicada a los aldeanos. Podemos observar 2 tipos de estados, unos principales: Hobbie que simplemente cuando llegue a un punto del mapa determinado iniciará, Parado (Idle), En movimiento (Walking) y los dos estados para dar la alarma, ya sea porque ha visto el vampiro está atacando a otro aldeano, Giving the Alarm In Movement, o porque encontró un cadáver, Giving the Alarm, y 2 especiales: Death y Notify a Suspision, los cuales están marcados en amarillo. Este color indica que los estados pueden iniciar desde cualquiera de los otros estados, ya que se activan por eventos del juego. Un posible ejemplo de es que el aldeano ha superado el umbral permitido de sospecha por lo que de manera automática se emite una señal (Signal en Godot) lo que provoca el cambio de estado.

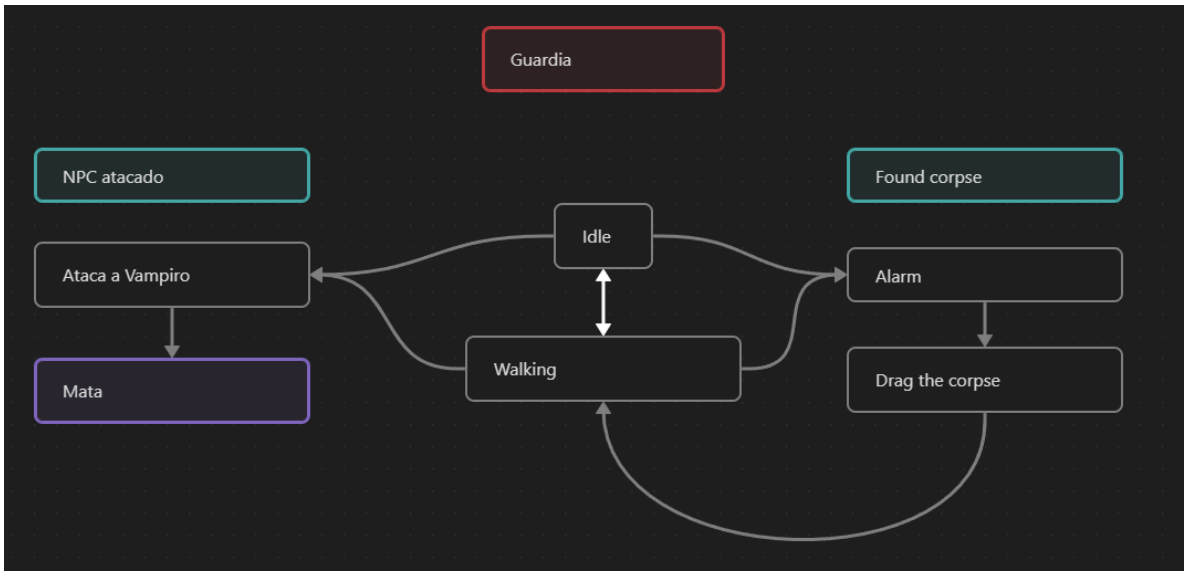


*Ilustración 15 - Diagrama de estados del Guardia*

En esta imagen podemos observar el diagrama de estados de los guardias. Se han estructurado en 3 sectores:

- Centro: Idle y Walking
- Derecha: Alarm y Drag the corpse, los cuales se inician cuando algún Aldeano a encontrado un cadáver. El primero para cuando alguien les avisa y el segundo para quitar el cadáver del camino
- Izquierda: Attack, sucede cuando el vampiro es detectado y se dirigen atacarle, acabando en la muerte de este

Cabe recalcar que "Mata" es una consecuencia del estado "Ataca a Vampiro", al igual que el estado Death del diagrama del vampiro. Estos suceden para dar paso a un evento.

### 5.2.4 Diagrama de secuencias

Los diagramas de secuencia son una herramienta para representar la interacción entre 2 o más elementos de un sistema, es decir, se centran en líneas de vida, procesos e intercambio de mensajes [58].

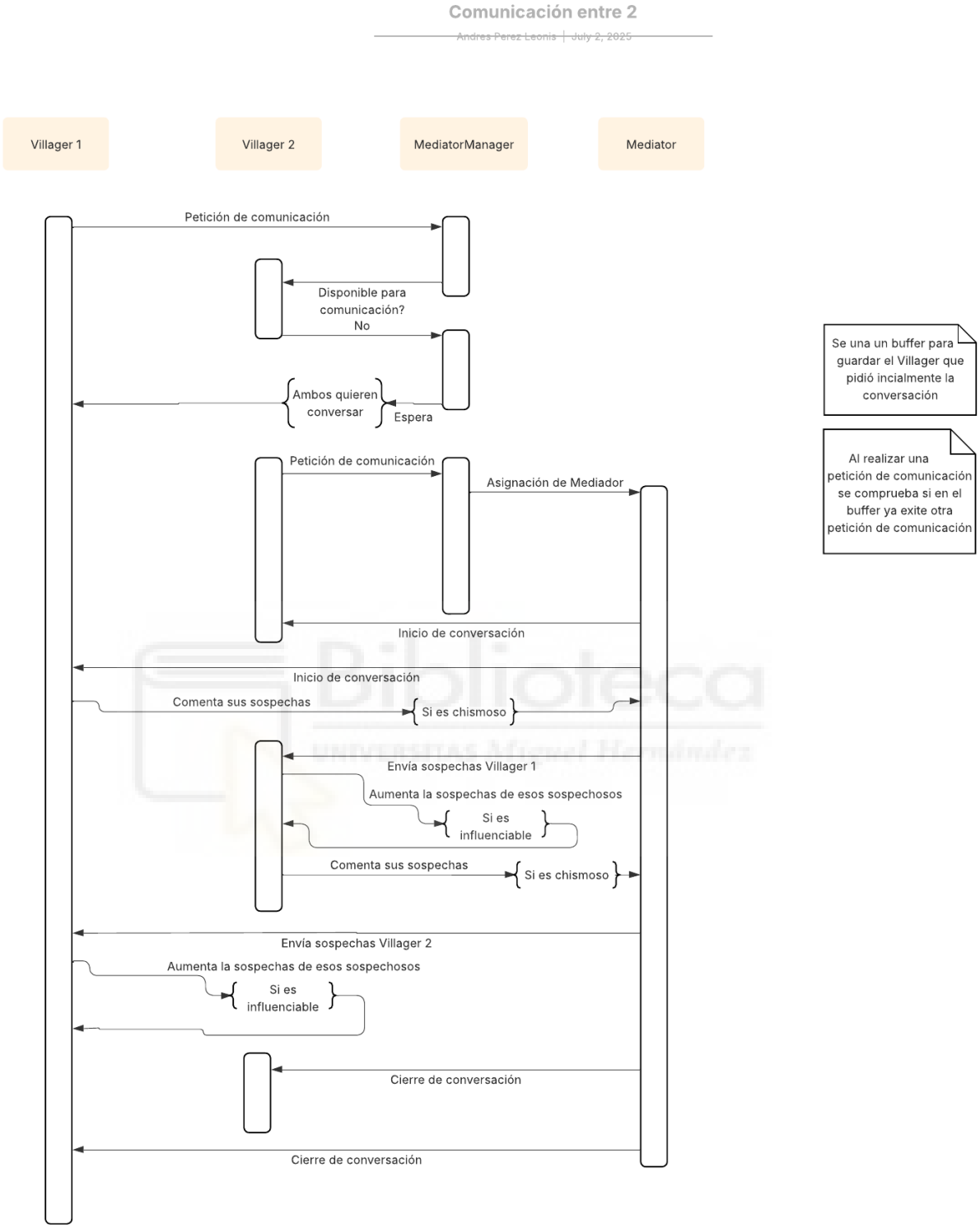


*Ilustración 16 - Diagrama de secuencia: Conversación entre Aldeanos*

En el diagrama se secuencia se detalla el proceso completo de comunicación entre dos aldeanos durante una partida, incluyendo algunos de los pasos lógicos y la intervención del patrón de diseño "Mediator" para facilitar y controlar la interacción. Además de permitir el desacoplamiento entre estas entidades.

El proceso inicia cuando uno de los aldeanos solicita iniciar una conversación con otro. Esta petición es enviada al MediatorManager, objeto común donde se aplica la técnica de optimización "Object Pooling". Antes de realizar la petición de comunicación Villager 1 ha pasado al estado "Talking", sin embargo, al ser un proceso secuencial y que la petición es enviada antes de que Villager 2 pase a este mismo estado, se aplicó un buffer de peticiones. El MediatorManager al recibir la primera petición comprueba que el Villager 2 esté en ese estado, si no lo está guarda la información de ambos Villagers hasta que reciba la petición de comunicación de este último, a modo de confirmación. Esto provoca que se inicie todo el proceso de conversación del diagrama, asignando un mediador para ello.

Una vez establecido el canal de comunicación, se inicia el intercambio de información. Aquí entre en juego los atributos de personalidad, donde cada aldeano puede presentar características diferentes, como puede ser que este sea chismoso y tienda por compartir sus sospechas o influenciable que provoca que acepte las sospechas de otros. Además de reducir las sospechas sobre este solo si piensa igual que él.

Este sistema me permite centralizar las comunicaciones en el MediatorManager, con el fin de evitar conflictos y colisiones en la comunicación, asimismo el uso del buffer temporal garantiza que ninguna petición se pierda o pierda coherencia si entra otra. Y, por último, el uso de atributos de personalidad me permite dar un distintivo al videojuego.

### 5.2.5 Diagrama de algoritmos

Los diagramas de algoritmos son diagramas de flujo que representan los pasos a realizar para resolver una tarea [59].

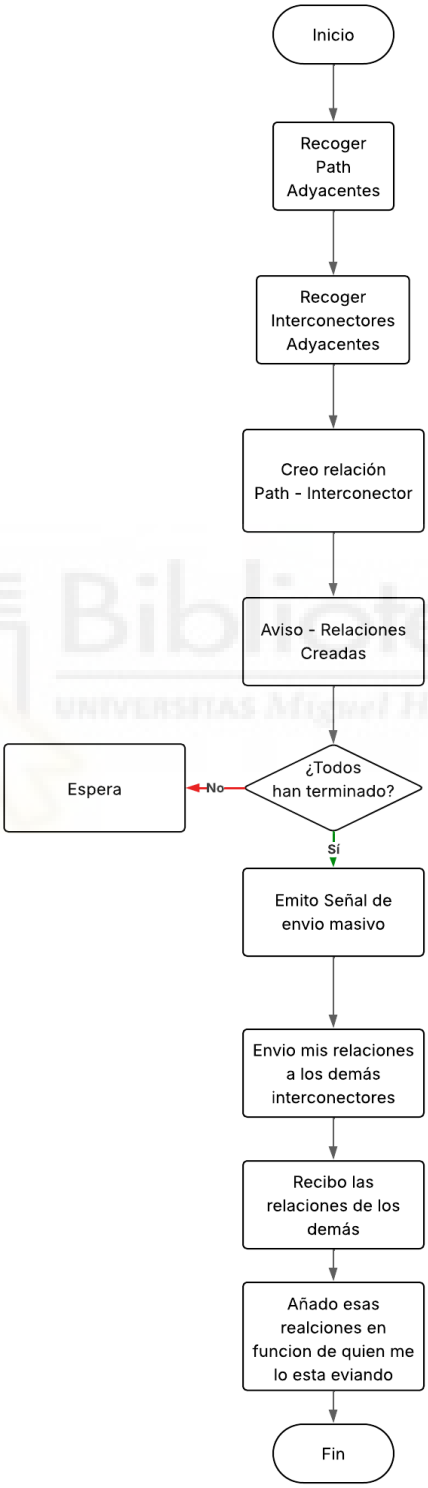


*Ilustración 17 - Diagrama de algoritmo: Creación de tabla de encaminamiento*

El algoritmo representado en el diagrama se encarga de construir las tablas de encaminamiento usadas por una parte del sistema de movimiento de los NPCs. La base de esta lógica se recoge del funcionamiento de routers y switches de las redes reales, donde cada nodo interconector conoce los destinos para dirigir el tráfico en función de la información recibida de sus vecinos.

Durante la fase de preproducción se tomó la decisión de utilizar los nodos Path2D para definir los caminos por los que se moverían los NPCs, en lugar de utilizar los nodos NavigationRegion2D y técnicas de Pathfinding que calcula las rutas en tiempo de ejecución.

Sin embargo, esta decisión trajo una limitación importante: los Path2D no permiten la creación de interconexiones dentro de estos y el intercambio de NPCs entre los distintos Path2D. Es decir, cuando los NPCs llegaban al final de ese camino no podían saltar al siguiente para continuar.

Para darle solución a este problema se desarrolló un sistema de intrconectores, los cuales actúan como una especie de pseudo switches. Estos nodos son colocados al final de los caminos y son los responsables de decidir, durante la partida, el camino por donde continuarán caminando los NPCs.

El algoritmo consta de 5 tareas o fases:

1. Recopilación de caminos e interconectores adyacentes
2. Creación de relaciones: Dentro de un Diccionario se crean una relación "Siguiente Camino" - "Destino"
3. Sincronización global: Un pseudo semáforo, para que todos los interconectores terminen de crear sus tablas iniciales antes de comunicárselas a los demás
4. Intercambio de información entre interconectores
5. Construcción de tabla final

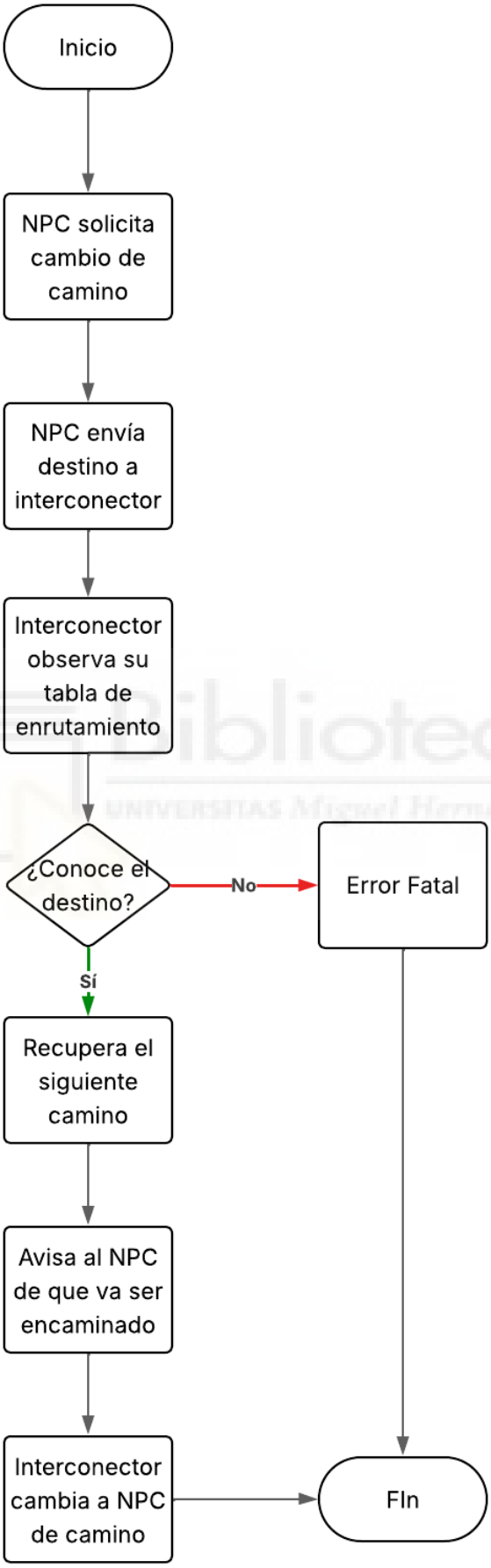

*Ilustración 18 - Diagrama de algoritmo: Intercambio de camino*

Como resultado de la decisión mencionada anteriormente y la solución aplicada, se necesita un algoritmo de intercambio de caminos como el representado en el diagrama anterior. Este consta de 3 fases:

1. Solicitud de intercambio: Donde el NPC le pregunta al interconector por donde debe continuar, mandándole su destino

2. Búsqueda del siguiente camino

3. Intercambio de camino

### 5.2.6 Público objetivo y requerimientos de hardware

En este apartado se abordan dos elementos esenciales dentro de un proyecto de videojuegos: público objetico, los usuarios a los que va dirigido el juego, y las necesidades de hardware que necesita dicho programa para ejecutarse de manera correcta.

El público objetivo de este proyecto se sitúa dentro en un rango de edad comprendido entre los 16 y 35 años. Decisión basada en 2 claves:

- El tipo de experiencia de juego: El título se diseñó para ser rápido, sencillo y accesible, especialmente atractivo para un público joven-adulto, lo que se convierte en un producto ideal para jugadores que buscan partidas breves, por ejemplo, mientras esperan que otro juego cargue.
- La clasificación por edades: De acuerdo con las categorías PEGI[12], este juego entra dentro de la categoría PEGI 16, debido a su contenido violento y presencia de sangre.

Los requerimientos de hardware son un aspecto muy complejo dentro del desarrollo de videojuegos, ya que no existe una metodología exacta precisa y estandarizada para determinar con exactitud cuales son las especificaciones mínimas necesarias para ejecutar un título sin problemas [60].

[12] Pan European Game Information. Clasificación de edad mínima recomendada para el jugador.

En la práctica, los desarrolladores recurren a 2 métodos para aproximarse a estos: Pruebas directas en diferentes dispositivos, con diferentes configuraciones de hardware, y el usar el propio ordenador de desarrollo como referencia.

Aunque existen otras alternativas, como el uso de máquinas virtuales para emular diferentes asignaciones de recursos, esas dos primeras son las más comunes. Es decir, en muchos casos, dicha estimación viene de una "adivinación informada".

En mi caso, debido a que el juego fue desarrollado en Godot Engine, podemos encontrar dentro de su documentación oficial una orientación general sobre los requisitos mínimos y recomendados de un proyecto simple realizado en el motor, ya sea 2D o 3D. Teniendo en cuenta la pequeña escala del proyecto, con mecánicas simples y sin mucha carga gráfica, asumo que dichos requerimientos son adecuados para este proyecto:

Requerimientos mínimos:

| | |
|---|---|
| **CPU** | • **Windows - Linux:** x86_32 CPU with SSE2 instructions, any x86_64 CPU, ARMv8 CPU<br>• **macOS:** x86_64 or ARM CPU (Apple Silicon) |
| **GPU** | Integrated graphics with full OpenGL 3.3 support or Direct3D 11 support (Windows). |
| **RAM** | 2 GB |
| **Almacenamiento** | 1GB |
| **Sistema operativo** | Windows 7 |

*Tabla 32 - Requerimientos mínimos*

Requerimientos recomendados:

| | |
|---|---|
| **CPU** | • **Windows:** x86_64 CPU with SSE4.2 instructions, with 4 physical cores or more, ARMv8 CPU<br>• **macOS:** x86_64 or ARM CPU (Apple Silicon)<br>• **Linux:** x86_32 CPU with SSE2 instructions, x86_64 CPU, ARMv7 or ARMv8 CPU |
| **GPU** | Dedicated graphics with full OpenGL 4.6 support |
| **RAM** | 4 GB |
| **Almacenamiento** | 150 MB (used for the executable, project files and cache) |
| **Sistema operativo** | Windows 10, macOS 10.15 (Forward+/Mobile, Vulkan), macOS 13.0 (Forward+/Mobile, Metal), Linux distribution released after 2020 |

*Tabla 33 - Requisitos Recomendados*

## 5.3 Implementación

En este apartado se presentan tres elementos clave del proceso visual del videojuego: algunos de los bocetos más representativos del desarrollo, sus

diseños finales y la visión general del aspecto gráfico. Además de algunas SpriteSheets de los personajes.

Durante las primeras fases del proyecto se realizaron numerosos bocetos para definir cosas como el entorno, la distribución de la ciudad y sus puntos clave. Sin embargo, uno de los procesos que ha conllevado más iteraciones fue el diseño de los NPCs y del personaje principal, el vampiro. La intención fue mantener un estilo simple pero funcional, con personajes distinguibles entre sí, siguiendo las referencias visuales del videojuego *Beholder*.

Durante todo el proceso se buscó que la interfaz siguiera el mismo hilo que los personajes, para que no hubiera un contraste en la atmosfera de este. Se buscó que esta fuera intuitiva, para no distraer al jugador. Además de buscar mejorar la experiencia del jugador sin necesidad de gráficos excesivamente complejos.

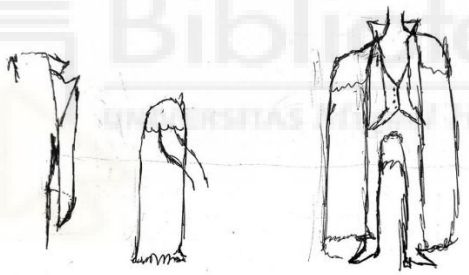

*Ilustración 19 - Boceto Vestimenta Vampiro 1*

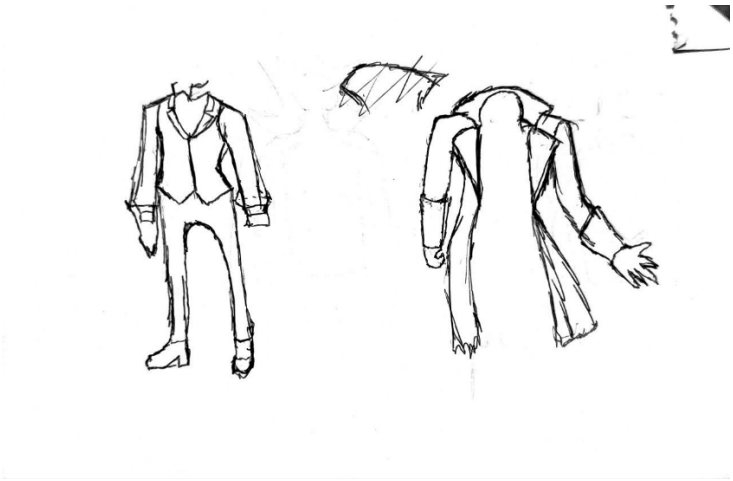

*Ilustración 20 - Boceto Vestimenta Vampiro 2*

*Ilustración 21 - Boceto Presentación Vampiro*

En las dos primeras ilustraciones puede apreciarse el concepto inicial del personaje principal. Se optó por una vestimenta que evocara una idea similar a las altas esferas de la época, especialmente por el lado de la nobleza, reflejando el lujo y su alto estatus. Dando la idea de una figura distinguida y poderosa.

Sin embargo, en fases más avanzadas del desarrollo visual, se decidió dar un giro al diseño del personaje, orientado hacia una apariencia más cercana a un cazador nocturno. Este nuevo estilo incorpora referencias a los murciélagos, símbolo

habitual de estas entidades, sin renunciar a los elementos de la elegancia y distinción de noble.

*Ilustración 22 – Vampiro*

Tras todo el proceso de diseño y de múltiples bocetos conceptuales, se llegó finalmente al estilo visual que representa al vampiro dentro del juego, como se observa en la ilustración. Como teníamos intención y hemos mencionado con anterioridad, el personaje presenta un enfoque estilizado y minimalista con el objetivo de reforzar la silueta y la lectura visual del personaje. Se buscó transmitir desde el primer vistazo una combinación de elegancia, misterio y autoridad, priorizando una postura erguida, con un porte sofisticado. Las formas del vestuario, es especial el torso y los hombros tratan de evocar un aire de nobleza y confianza. El rostro sin rasgo busca universalizar el personaje y centrar la atención del usuario en la silueta del personaje.

Uno de los elementos que más cambió dentro de este diseño es la capa. En algunos de los bocetos podíamos ver una capa extravagante y prominente, sin embargo, en esta versión final, este recurso se integró visualmente en forma de

chaleco o abrigo junto con otra capa más relajada de color rojo. Este diseño no solo busca una fuerte identidad visual, sino también facilitar la animación, evitar detalles excesivos, centrándose en formas claras y colores sólidos, garantizar el destaque del personaje dentro del entorno.

Aunque el vampiro tiene varias animaciones que reflejan diferentes acciones uno de los primeros que se verán al empezar una partida es la del vampiro cuando está parado.

*Ilustración 23 - SpriteSheet Vampiro Parado*

Aunque sea el personaje se vea algo pequeño, en la imagen anterior corresponde a la animación del vampiro parado, es decir, cuando el jugador no está realizando ninguna acción con él. Esta animación, aunque sutil, juega un papel fundamental en dotar de vida al personaje y evitar que parezca estático mientras se permanece inactivo.

La animación consiste en un movimiento vertical del torso, simulando el ritmo de la respiración. Esta pequeña animación transmite la idea de que el vampiro está vivo y en vigilancia. A pesar de su sencillez, el efecto resulta muy efectivo para reforzar la presencia del personaje.

Además, este tipo de animación tiene beneficios tanto estéticos como funcionales. Estéticamente, mantiene al jugador inmerso en el juego al evitar cortes abruptos entre las acciones y estados pasivos. Desde el punto de vista técnico, se trata de una animación ligera, la cual no afecta al rendimiento.

*Ilustración 24 - Vampiro Parado*

El proceso de diseño de los NPCs siguió una metodología diferente a la empleada para el vampiro. En lugar de partir directamente de bocetos conceptuales de los diferentes estilos de vampiro que podemos encontrar, el primer paso fue realizar una investigación sobre el estilo de vida de los aldeanos. Se analizaron aspectos como la vestimenta típica, las ocupaciones más comunes, las herramientas y objetos usados, e incluso su comportamiento social. Esta aproximación permitió construir una base sólida sobre cómo desarrollar estos personajes.

*Ilustración 25 - Primer boceto aldeanos*

Uno de los principales referentes fueron diversas punturas y grabados de la época, los cuales ofrecieron una representación del día a día de estos habitantes. A partir de estas fuentes, se realizó un primer boceto general de cómo eran estas personas, presentando especial atención a los elementos más identificativos: sombreros, capas, faldas largas, delantales y peinados.

Posteriormente, se trabajó es estilizar estas figuras para adaptarlas al estilo gráfico general del juego. El objetivo era conseguir personajes que, a pesar de su simplicidad visual, fueran reconocibles a simple vista. Por ello, se extrajeron las características más relevantes para sintetizarlas en siluetas fácilmente distinguibles.

Se recopiló referencias de diferentes perfiles sociales que podrían encontrarse dentro de una aldea medieval: enfermos, discapacitados, comerciantes y agricultores. La mayoría de los aldeanos vestían túnicas sencillas y capas, principalmente en tonos marrones, amarillentos, verdosos o azulados, colores que respondían a los tintes naturales de la época y su posición económica. En contraste, los más desfavorecidos, como pobres o enfermos, llevaban prendas

deterioradas e incompletas. Luego de varias iteraciones para simplificar y caracterizar a los personajes, llegamos a este último boceto:

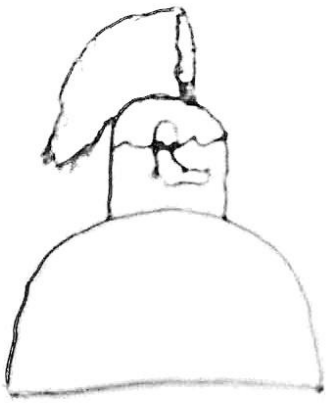

*Ilustración 26 - Boceto final aldeana*

Como podemos observar el personaje acabo con un trazo simple y esquemático, en el que podemos diferenciar diferentes elementos: un gorro largo bastante característico en las mujeres trabajadoras de la Europa de aquella época. Este tipo de gorro, junto con la figura abultada, viene dada por un ropaje típico de la época en el que la persona se envolvía en varias capas o túnicas, propias del clima y de las condiciones de vida rurales. Finalmente fue mínimamente modificada para su diseño visual final.

*Ilustración 27 – Diseños finales aldeanos*

El diseño del entorno y el mapa se encontró entre los elementos de desarrollo más ágiles del proyecto, gracias al uso de la ciudad de Brujas (Bélgica), como referencia.

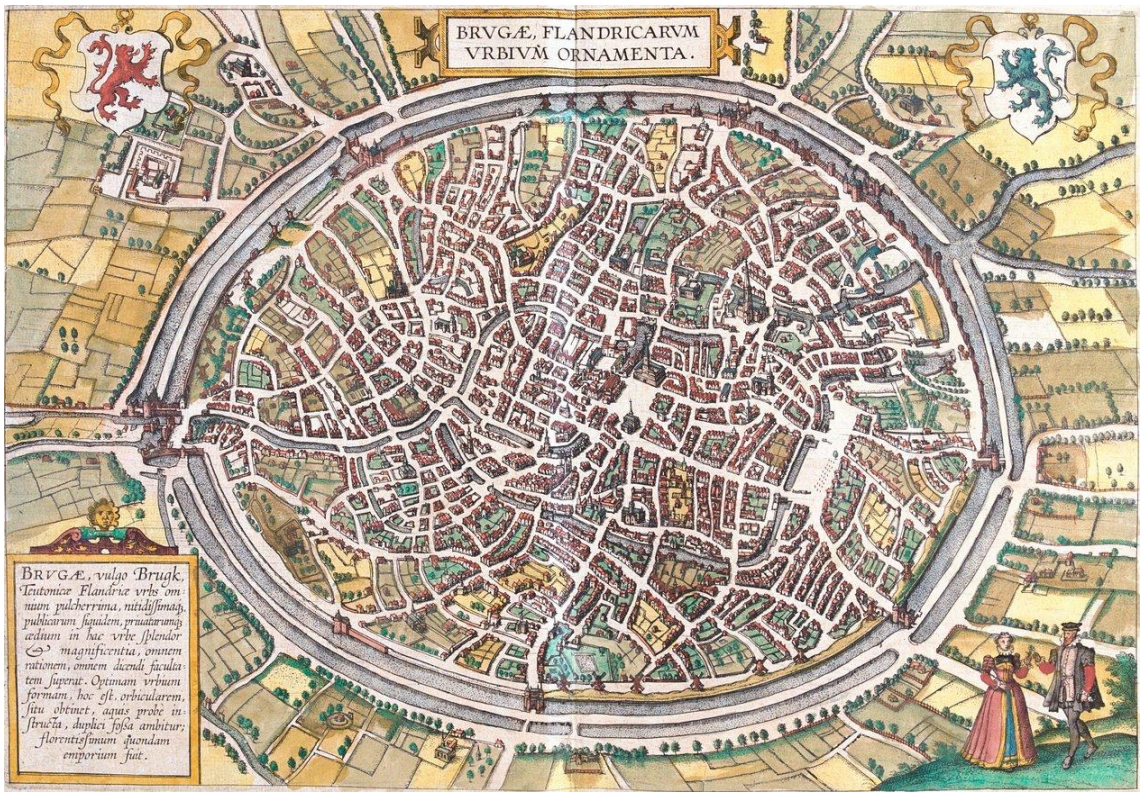


*Ilustración 28 - Mapa de Brujas*


*Ilustración 29 - Boceto Aldea*

En esta ilustración pueden observarse varios elementos del desarrollo del entorno del juego. Se incluyen los bocetos conceptuales de distintos puntos del mapa, asó como objetos característicos como la picota, carretas o el puente sobre el río. También se aprecia una lista preliminar de edificios y barios que se consideraron para ser integrados dentro del diseño final.

Algunos de los estos espacios, como el "barrio de trabajo", fueron finalmente fusionados con otras áreas, como el mercado. Otros elementos, como los interiores de casas, mostrado en el boceto como "edificio de artesano", fueron descartados en fases posteriores al determinar que no aportaban una experiencia jugable lo suficientemente relevante para justificar su implementación.

*Ilustración 30 - Boceto del entorno*

En la ilustración presenta un entorno lúgubre, misterioso y envuelto en una oscuridad propia de las horas de la noche. Este diseño busca materializar la atmosfera central que guía la concepción del mapa y sus escenarios. La

ambientación oscura refleja un periodo histórico: aquel en el que los aldeanos, seguían un patrón de sueño bifásico[13].

*Ilustración 31 – Edificios*

En la imagen se muestran algunos de los diseños finales de los edificios realizados para este proyecto con estética gótica estilizada. Estas edificaciones presentan detalles góticos en sus ventanas, balcones y arcos. Se utilizaron formas simples y colores oscuros para mantener la coherencia visual, especialmente con los personajes y el ambiente nocturno del juego.

Finalmente, tras un largo proceso de trabajo y múltiples iteraciones, el proyecto comienza a mostrar resultados concretos. En las siguientes ilustraciones se presenta el menú principal del juego, en el cual se puede ver al personaje principal contemplando la ciudad donde se desarrollará la historia.

Aunque en la imagen estática no se aprecie, el vampiro cuenta con una animación de respiración que le otorga vida mientras el jugador permanece en la pantalla de inicio. Al pulsar el botón de "Iniciar partida", se ejecuta una animación especial del personaje, la cual oscurece progresivamente toda la pantalla y da paso a la escena de carga, generando una transición fluida y coherente con la atmósfera del juego.

[13] *Los patrones de sueño bifásico comunes en la Europa preindustrial interrumpían su descanso entre la 1 y las 3 de la madrugada para realizar actividades cotidianas antes de retomar el sueño hasta el amanecer.*

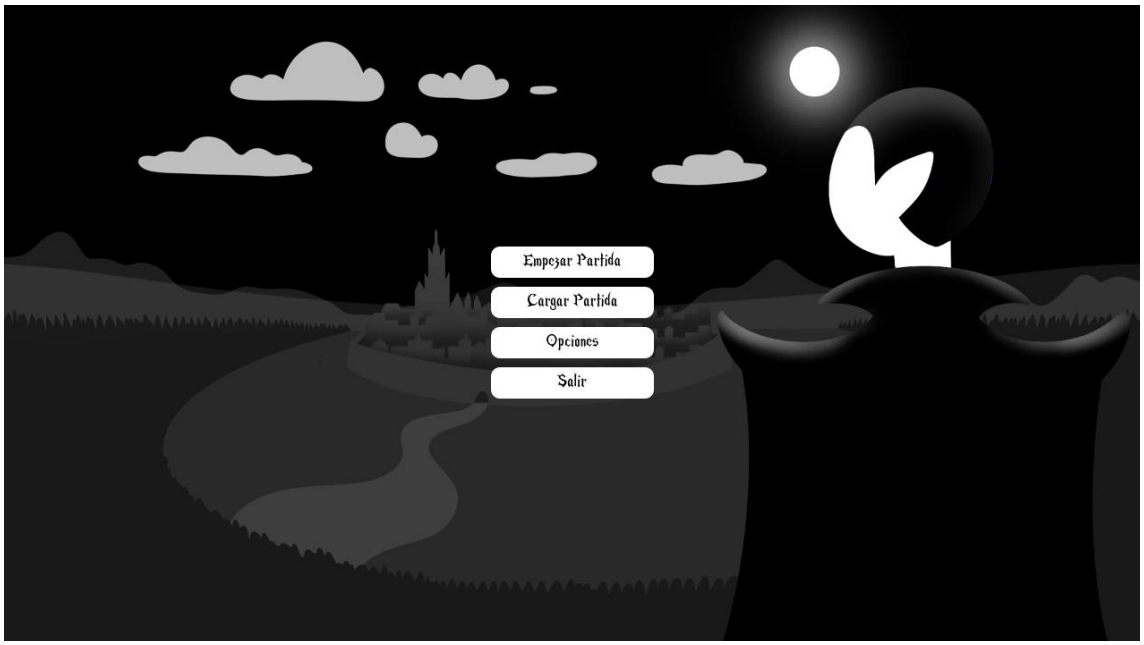


*Ilustración 32 - Imagen del menú principal*

En lo que respecta a la interfaz de usuario (UI) se optó por un diseño totalmente limitado, en coherencia con que el jugador solo tiene una vida única, y si es detectado, morirá de forma casi instantánea e irreversible.

Dado que no existe un sistema de salud o recursos relevantes que gestionar, se ha prescindido de elementos tradicionales como barras de vida o indicadores de estado. En su lugar, la interfaz se reduce a 2 elementos principales:

- Un temporizador circular con un símbolo de mordisco, que indica el cooldown tras atacar a un aldeano y así darle un ciclo de alimentación estratégico.
- Un cuadro de información contextual que aparece al pulsar sobre los NPCs, mostrando sus 3 mayores sospechas y su tipo de personalidad.

El proyecto incorpora 3 tipos de menús, que tratan de ser funcionales y mantener la estética oscura y misteriosa del juego:

- Menú principal, que contiene opciones básicas como: Inicio partida, Cargar partida, Ajustes y Salir del juego
- Menú de pausa, permite reanudar partida, Volver al menú principal y abrir los ajustes. Cuando este aparece se oscurece el fondo para diferenciarlo más con el entorno del juego.
- Menú de ajustes, para personalizar la configuración gráfica como la resolución y el modo de pantalla (completa, ventana, …) y el volumen.

Dentro del desarrollo del proyecto, hay 3 aspectos que debo mencionar, que determinaron el correcto funcionamiento y organización del código. En primer lugar, tenemos la implementación de la máquina de estados. Como se ha mencionado y explicado anteriormente, este patrón permite gestionar de manera eficiente los distintos comportamientos del personaje mediante transiciones entre los estados. Su implementación en Godot presentó ciertos desafíos técnicos:

- Integración con la jerarquía de nodos: Dado que en Godot cualquier script que deba ser ejecutado debe estar asociado a un nodo dentro de la escena, se optó por utilizar nodos del tipo "Node" – los más básicos del motor – para albergar los scripts de cada estado y de la propia máquina de estados. Estos nodos no requieren una representación visual ni espacial en el juego, pero permiten que los scripts se ejecuten.

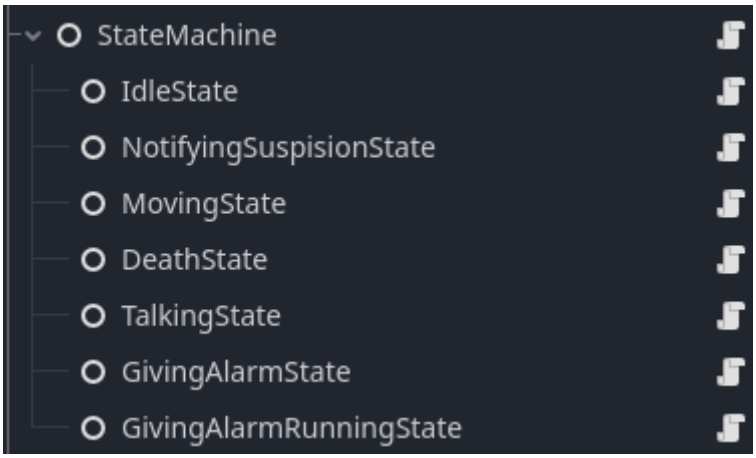


*Ilustración 33 - Máquina de estados implementada en Godot*

La máquina de estados fue diseñada para ser genérica, lo que significa que puede ser utilizada con otros objetos del juego sin modificaciones estructurales. Esto se logró aprovechando que el nodo "Node" es la clase padre de todos objetos de Godot, permitiendo una adaptación a distintos contextos. La máquina actúa de nodo raíz, añadiendo cada estado individual como hijo directo de la máquina, simplificando la gestión de estados y facilitando la depuración, acceso y escalabilidad del sistema.

```
public partial class StateMachine : Node
{
    /***
     * The initial state to be set when the scene is ready.
     */
    [Export]
    private StateBase _defaultState;

    /***
     * The currently active state.
     */
    private StateBase _currentState;

    /***
     * The node being controlled by this state machine
     */
    private Node _controlledNode;

    /***
     * Called when the node enters the scene tree.
```

```
     * Uses deferred call to ensure all nodes are initialized before
setting the state.
     */
    public override void _Ready()
    {
        CallDeferred("_readyCalledDefered");
    }

    /***
     * Deferred method called after the node is ready.
     * Initializes the current state to the default state and starts it.
     */
    private void _readyCalledDefered()
    {
        _currentState = _defaultState;
        _startState();
    }

    /***
     * Configures and starts the current state.
     */
    private void _startState()
    {
        _currentState.ControlledNode = _currentState;
        _currentState.StateMachine = this;
        _currentState.Start();
    }

    /***
     * Changes the current state to the specified state.
     * Ensures the previous state is properly ended before switching.
     * @param newState The name of the new state node.
     */
    public void ChangeState(string newState)
    {
        if (_currentState != null)
            _currentState.End();
        if(_currentState.Name == newState) return;
        GD.Print("Soy " + Owner.Name + " Estado cambiado a: " + newState
+ " desde " + _currentState.Name);
        _currentState = (StateBase)GetNode<Node>(newState);
        _startState();
    }

    /***
     * Gets the current active state.
     */
    public StateBase CurrentState
    {
```

```
        get => _currentState;
    }

    #region Automatic Execution Methods

    /***
     * Called every frame. Passes execution to the current state's
processing method.
     */
    public override void _Process(double delta)
    {
        _currentState?.OnProcess(delta);
    }

    /***
     * Called during the physics step. Passes execution to the current
state's physics processing method.
     */
    public override void _PhysicsProcess(double delta)
    {
        _currentState?.OnPhysicsProcess(delta);
    }

    /***
     * Handles input events and delegates them to the current state.
     */
    public override void _Input(InputEvent @event)
    {
        _currentState?.OnInput(@event);
    }

    /***
     * Handles unhandled input events and delegates them to the current
state.
     */
    public override void _UnhandledInput(InputEvent @event)
    {
        _currentState?.OnUnhandledInput(@event);
    }

    /***
     * Handles unhandled key input events and delegates them to the
current state.
     */
    public override void _UnhandledKeyInput(InputEvent @event)
    {
        _currentState?.OnUnhandledKeyInput(@event);
    }
```

```
    #endregion
}
```

El fragmento de código presentado corresponde a la implementación de una máquina de estados en Godot, diseñada para gestionar los comportamientos de los personajes y entidades del juego. Podemos observar la clase StateMachine, la cual hereda de Node y actúa como núcleo de este sistema. Sus elementos más relevantes son:

- Atributos
    - _defaultState: Estado inicial del objeto
    - _currentState: Estado activo
    - _controlledNode: Referencia al nodo controlado
- Métodos:
    - _Ready y _readyCalledDefered: Primer método que se ejecuta al cargar el objeto en la escena. CalledDefered, se usa para evitar errores asegurando que todos los nodos están inicializados antes de asignar el estado inicial.
    - ChangeState: Método que se usa cuando se quiere cambiar el estado de un objeto
- Sección de ejecución de eventos y métodos
    - Referido a la sección de código de los métodos _Process, _PhysicsProcess, _Input, etc.
    - Son métodos del propio ecosistema de Godot
    - Se usa para ejecutar su equivalente dentro del estado en ejecución

En segundo lugar, nos encontramos con el sistema de sospechas que ha introducido dentro del juego. Los aspectos más relevantes de este son:

El uso de 2 tipos de estructuras de datos para referencias los mismos elementos:

```
    private List<EntitySuspechData> _suspechData = new();
    private Dictionary<Entity, EntitySuspechData> _dicSuspechData;
```

Esta decisión viene dada de las reiteradas ocasiones en las que se accede a los datos de estas estructuras. La lista se usa como una lista ordenada de forma continua, sin embargo, como las comunicaciones entre los aldeanos varían en las sospechas que se envían se debería recorrer toda la lista por cada sospecha que hayamos recibido por parte del otro aldeano, lo cual puede conllevar un gasto de recursos importante durante la ejecución de estos métodos. Para apaliar esto se utiliza el diccionario, el cual proporciona un acceso rápido (O(1)) a los datos y evita las búsquedas a la lista, que relaciona una entidad con una clase propia llamada "EntitySuspechData". La combinación de estas estructuras optimiza el rendimiento, ya que el diccionario acelera las operaciones individuales, mientras que la lista mantiene un orden jerárquico para las decisiones globales del sistema. Dicha clase recoge 3 atributos:

```
    private Entity _entity;
    private int _amountOfSuspision;
    private bool _thinkIsVampire;
```

- _entity: La entidad de la cual se sospecha
- _amountOfSuspision: La cantidad de sospecha [0,100]
- _thinkIsVampire: Un booleano para determinar si el NPC piensa que dicha entidad es un vampiro

Cabe mencionar que Entity es una clase propia de la cual heredan todos los personajes del juego (Aldeano, Vampiro y Guardia).

Luego de recibir las sospechas se ejecuta el método AnalizeSuspision, el cual gestiona las sospechas recibidas de otros aldeanos y actualiza las estructuras de datos, teniendo en cuenta en el transcurso la personalidad del aldeano. Primero comprueba si es fácilmente influenciable, si no lo es, aumenta los niveles de sospecha de aquellos aldeanos comentados y si el emisor sospecha de nosotros se aumentará la sospecha de este también. En el caso de que el emisor sospeche

de los mismos aldeanos que el receptor, se simula un efecto de confianza y se reduce los niveles de sospecha hacia el emisor. Luego de realizar estos cambios se reordena la lista.  Durante el incremento de las sospechas se gestionan las denuncias a los guardias, si el nivel de sospecha supera ciertos umbrales de sospecha el NPC podrá avisar al guarda para su arresto.

```
public void AnalizeSupisions(List<EntitySuspechData> entities, Entity whoSaidMeThat)
    {
        //Comprobamos las sospechas del otro NPC
        if (!_myVillager.Personality.EasyInfluenced) return;

        bool heThinkItsMe = false;

        //Dictionary<Entity, EntitySuspechData> suspechData = _suspechData.ToDictionary(s =>
s.Entity);
        foreach (EntitySuspechData entityData in entities)
        {
            heThinkItsMe = (entityData.Entity == _myVillager);
            IncreaseSuspision(entityData.Entity);
        }

        if (heThinkItsMe) IncreaseSuspision(whoSaidMeThat);

        HashSet<Entity> hashEntity = new
HashSet<Entity>(EntityInSuspech(entities.Count).Select(s => s.Entity));
        int matches = entities.Count(e => hashEntity.Contains(e.Entity));

        _dicSuspechData[whoSaidMeThat].AmountOfSuspicion -= _reductionSuspisionAmount * matches;

        _suspechData.Sort((a, b) => b.AmountOfSuspicion.CompareTo(a.AmountOfSuspicion));
    }
```

```
public void IncreaseSuspision(Entity entity)
    {

        if (!_dicSuspechData.TryGetValue(entity, out var data))
        {
            GD.PrintErr($"[IncreaseSuspicion] Error: No se encontró la
entidad '{entity.Name}' en el diccionario.");
            return;
        }
```

```
        data.AmountOfSuspicion += _increaseSuspisionAmount;

        if (data.AmountOfSuspicion >= 100)
        {
            TellingToGuard();
        }
        else if (data.AmountOfSuspicion > 90 && _rnd.NextDouble() > 0.5)
        {
            TellingToGuard();
        }
    }
```

Y, por último, el sistema de carriles simula el comportamiento de una red de switches y routers, permitiendo a los NPCs navegar por el mapa. Este sistema se compone de puntos de interconexión (*Markers*) que gestionan las rutas disponibles y dirigen a los NPCs hacia sus destinos. Estos pasan por 2 etapas:

- Análisis de adyacentes: En esta fase, cada punto de interconexión configura sus rutas iniciales basándose en sus conexiones adyacentes. Este proceso se realiza dentro del método _Ready, usando atributos exportables para definir manualmente las rutas conectadas desde el editor de Godot.

  Cada Marker se suscribe a una señal global para coordinar el envío de rutas, recoge sus puntos adyacentes y se añaden a un diccionario interno. Una vez terminan todos se emite la señal mencionada.

```
public override void _Ready()
    {
        // Connect the MarkerSignal with the method that it must do
        MarkerSignals.MarkerToCommunicate += SendRoutes;

        // Only the first Marker initialize the TotalMarkers variable
        if(TotalMarkers == 0) TotalMarkers =
GetTree().GetNodeCountInGroup(NameGroups.InterconnectionRoute);


        // Build the initial table resend

        MarkerPathSwitch[] markersOfOther;
        for(int i = 0; i < _markersAdjacent.Length; i++) {
            AddRoute(_markersAdjacent[i], _pathInterconectedByMe[i]);
            markersOfOther = _markersAdjacent[i].MarkersAdjacent;
```

```
            for(int j = 0; j < markersOfOther.Length; j++) {
                AddRoute(markersOfOther[j], _pathInterconectedByMe[i]);
            }
        }

        // Tell everyone that it finished
        MarkersFinished++;
        // If everyone finished, Emit the signal to communicate his
inital table
        if(MarkersFinished == TotalMarkers) {
            MarkerSignals.EmitCommunicateMarkerSignal();
        }

    }
```

- Envío de diccionarios: Los Markers intercambian sus diccionarios de rutas para crear la tabla de redireccionamiento completa. Esto permite que cada punto de interconexión conozca las rutas óptimas hacia cualquier destino en el mapa.

```
  private void SendRoutes() {
        foreach(MarkerPathSwitch marker2D in _redirectionDictionary.Keys)
{
            if(marker2D.GetDictionarySize() == TotalMarkers - 1)
continue;
            marker2D.ReciveRoutes(_redirectionDictionary, this);
        }
    }

    public void ReciveRoutes(Dictionary<MarkerPathSwitch, Path2D>
directions, MarkerPathSwitch markerObject) {

        int index = 0;
        for(int i = 0; i < _markersAdjacent.Length; i++) {
            if(_markersAdjacent[i] == markerObject) index = i;
        }
        foreach(MarkerPathSwitch marker in directions.Keys) {
            AddRoute(marker, _pathInterconectedByMe[index]);
        }
        //PrintRoutes();
    }
```

Una vez configurada la tabla de redireccionamiento, los Markers esperan a que los NPCs lleguen a sus ubicaciones. Cuando un NPC alcanza un Marker, este determina la siguiente ruta basándose en el destino del NPC. Este analiza que camino debe asignarle al NPC y lo hace, avisando al NPC de que ha sido cambiado de camino, enviándole información del camino para que este sepa hacia donde mirar. A no ser que este punto de interconexión sea el propio destino, que en su lugar avisa al NPC de que ha llegado.

```
public void ChangeTheNPCRoute(NpcPathFollow npcPathFollow,
MarkerPathSwitch destination) {
        npcPathFollow.LastPassMarker = this;
        if (destination == this)
        {
            npcPathFollow.EmitInMyDestinationSignal();
            return;
        }
        Path2D nextPath = _redirectionDictionary[destination];
        npcPathFollow.EmitOnChangePathSignal(nextPath.ToGlobal(nextPath.C
urve.GetPointPosition(nextPath.Curve.PointCount - 1)));
        npcPathFollow.GetParent<Path2D>().RemoveChild(npcPathFollow);
        nextPath.AddChild(npcPathFollow);
    }
```

## 5.4 Pruebas

El proceso de pruebas es fundamental para garantizar la calidad del videojuego y asegurar que todas las funcionalidades implementadas funciones correctamente. Durante el desarrollo del proyecto, se llevaron a cabo diferentes tipos de pruebas para validar el comportamiento del sistema, identificando errores:

- Pruebas unitarias y Debugging: Godot no tiene una herramienta integrada para realizar pruebas unitarias, y aunque existen herramientas por parte de la comunidad como UnitTest, también llamada gdUnit4, se imposibilitó su uso debido a que esta herramienta solo permite el uso de GDScript y actualmente no es compatible con C#. En su lugar se realizó los test de

manera manual y mostrando los valores y puntos importantes por la consola del editor, es decir, se realizó un debugging con la ayuda de VSCode y plugins para permitirlo.

- Pruebas de integración: Algo que se debe tener en cuenta durante el desarrollo de un videojuego es que debido a que se permite la exportación sobre diferentes sistemas, como Android, IOS, Windows o Linux, aquellas funcionalidades que funcionan perfectamente en el editor, pueden no funcionar luego de realizar la exportación del proyecto. Por lo que se al finalizar ciertos sprints, como puede ser el sistema de carriles de los NPCs, se realizaba una exportación de prueba para comprobar que todo funcione.

# CAPÍTULO 6: CONCLUSIONES Y DESARROLLOS FUTUROS

## 6.1 Conclusiones

Este Trabajo de Fin de Grado ha representado un desafío técnico y creativo, el cual me ha permitido aplicar gran parte de los conocimientos adquiridos durante la carrera, orientándolos al desarrollo de un videojuego desde su concepción hasta su implementación final. A lo largo del proyecto, se han abordado todos los aspectos clave del sector de los videojuegos, como las polémicas actuales de la industria, así como las oportunidades que ofrecen los juegos independientes para explorar nuevas narrativas y mecánicas innovadoras.

Puedo afirmar que el desarrollo del videojuego, ambientado en la Brujas medieval, ha cumplido con todos los objetivos planteados en el tercer capítulo de este documento:

- Aprendizaje técnico: Se ha profundizado en el uso de Godot Engine, patrones de diseño como State y Mediator, y técnicas de optimización como Object Pooling.
- Metodología ágil: La planificación con herramientas como Obsidian, Lucidchart y GanttProject ha demostrado ser eficaz para gestionar un proyecto individual.
- Resultado jugable: Se ha logrado una experiencia funcional con mecánicas de sigilo, toma de decisiones y un sistema de sospechas dinámico.

Sin embargo, el proyecto también ha dejado claras áreas de mejora, como la fluidez en los controles del vampiro o la expansión de contenido (más NPCs, mejora del contenido audiovisual). Estos aspectos, junto con posibles mejoras visuales y de inteligencia artificial, se plantean como futuros desarrollos.

En lo personal, este TFG ha reforzado mi pasión por el desarrollo de videojuegos, confirmando que el equilibrio entre libertad creativa y disciplina técnica es esencial para crear experiencias memorables. La publicación del juego en plataformas como Itch.io o Steam podría ser el siguiente paso, no solo como un producto terminado, sino como el inicio de un camino profesional en la industria.

Finalmente, este trabajo refleja cómo los videojuegos, más allá de su faceta de entretenimiento, son un medio poderoso para contar historias, explorar mecánicas innovadoras y, en mi caso, materializar un sueño de infancia: crear mundos que, como aquellos que me inspiraron, puedan ser refugio y aventura para otros jugadores.

## 6.2 Desarrollos futuros

Dentro de la industria del videojuego, especialmente en el ámbito indie, es común encontrarse con el término "Feature Creep". Este concepto hace referencia a la tendencia de algunos desarrolladores de introducir constantemente nuevo contenido a su videojuego, en muchas ocasiones sin una planificación clara y más allá de su alcance original. Lo que puede llevar a que le producto nunca llegue a su versión final. Un ejemplo extremadamente claro es *Yandere Simulator*, título que actualmente lleva en desarrollo algo más de 11 años. [61]

Menciono esto porque, aunque tengo en mente múltiple ideas para expandir el juego, considero esencial marcar unos límites claros respecto a las funcionalidades que se añadirán en un futuro, para asegurar la viabilidad del proyecto:

- Mejorar el movimiento del vampiro, ya que actualmente se siente tosco y puede transmitir una sensación de falta de peso o fluidez en su control
- Incluir emboscadas, permitiendo que el vampiro ataque desde lugares elevados o desde escondites

- Añadir la posibilidad de escapar de la aldea si es descubierto, permitiendo al jugador huir usando escondites o disfraces
- Mejoras visuales, como la adición de partículas o animaciones adicionales

## 6.3 Publicación

Como en todo videojuego, existe la posibilidad de que llegue a publicarse, ya sea de forma gratuita o mediante una compensación económica. Para la publicación de juegos existen varias alternativas:

- Itch.io: Plataforma popular dentro del ámbito indie, donde se pueden encontrar títulos de todo tipo y escala. La publicación de juegos dentro de esta web es gratuita, aunque la plataforma retiene una comisión sobre cada venta. Además de videojuegos, también permite la comercialización de assets y otros recursos relacionados con el desarrollo.

*Ilustración 34 - Logo de Itch.io [62]*

- Steam: La plataforma de distribución digital de videojuegos por excelencia. Aunque contiene juegos independientes, el proceso de publicación no es gratuito, ya que requiere una tarifa única por título publicado. Además, Valve, la empresa detrás de la plataforma, se queda con un 30% de los ingresos generados por cada venta. Aunque, su enorme base de usuarios la convierte en muy buena opción.


*Ilustración 35 - Logo de Steam [63]*

- Epic Game Store: Comperidora directa de Steam, esta plataforma se caracteriza por tener comisiones más bajas para los desarrolladores, quedándose un 12% de las ventas. Sin embargo, tiene políticas más estrictas de publicación, dificultando la entrada de los desarrolladores independientes. Además, su base de usuarios es menor que la de Steam.


*Ilustración 36 - Logo de Epic Games [64]*

## 6.4 Nuevos conocimientos

Durante el desarrollo de este proyecto he adquirido una gran variedad de conocimientos, tanto técnicos como de gestión de proyectos y diseño de videojuegos. En primer lugar, uno de los aprendizajes más significativos ha sido el dominio del motor Godot, que plantea un paradigma de trabajo muy diferente al de otros motores comerciales. Su sistema de escenas y nodos me obligó a entender nuevas formas de organizar la lógica del juego y a repensar la arquitectura interna del proyecto. También he aprendido a trabajar con su sistema de señales y eventos, que resultó esencial para comunicar diferentes elementos del juego de forma desacoplada.

Otro aspecto importante ha sido la profundización en C# como lenguaje de programación dentro del entorno de Godot. Elegí este lenguaje por su rendimiento y claridad sintáctica, y durante el proceso he podido reforzar conocimientos sobre programación orientada a objetos, herencia, encapsulamiento y uso de estructuras de datos adaptadas a videojuegos.

Desde el punto de vista del diseño, he aprendido a estructurar diagramas de estado, de herencia y de flujo que han resultado fundamentales para entender el comportamiento de los personajes (como el vampiro o los aldeanos) y sus transiciones lógicas. También he experimentado con sistemas de navegación personalizados, desarrollando una solución propia inspirada en conceptos de redes y enrutamiento debido a las limitaciones de Path2D en Godot, lo cual me permitió crear un sistema de caminos con puntos de interconexión pseudo-inteligentes.

Además, he conocido y aplicado conceptos metodológicos relevantes en el desarrollo de videojuegos como el uso de ciclos de vida ágiles (Kanban en este caso) y herramientas como Obsidian, LucidChart o GanttProject, que me han ayudado a planificar y organizar el trabajo de forma eficiente, incluso siendo un único desarrollador. Esto también me ha permitido entender cómo adaptar metodologías de trabajo en equipo a un entorno unipersonal.

Finalmente, desde un punto de vista más general, he tomado conciencia de riesgos comunes en el desarrollo indie, como el feature creep, y de la importancia de establecer límites claros para evitar que un proyecto se alargue innecesariamente. También he explorado posibles vías de publicación, valorando sus ventajas y desventajas según el tipo de producto y su estado de madurez.

# CAPÍTULO 7: BIBLIOGRAFÍA

[1] Statista. *Industria mundial del videojuego.* https://es.statista.com/temas/9150/industria-mundial-del-videojuego/#editorsPicks

[2] Statista. *Evolución de la industria del videojuego (2017-2027).* https://es.statista.com/estadisticas/598622/valor-de-mercado-del-videojuego-en-el-mundo/

[3] Europapress. *Tencet, Microsoft y Sony encabezan las listas de las 25 empresas con más ingresos de la industria*. https://www.europapress.es/portaltic/empresas/noticia-tencent-microsoft-sony-encabezan-lista-25-empresas-mas-ingresos-industria-videojuego-20151105135945.html

[4] Wikipedia. *Escasez de Microchips.* https://es.wikipedia.org/wiki/Escasez_global_de_chips_(2020-2023)#:~:text=Autom%C3%B3viles%2C%20tarjetas%20gr%C3%A1ficas%2C%20consolas%20de%20videojuegos%2C%20computadoras,Estados%20Unidos%20y%20varios%20incidentes%20clim%C3%A1ticos%20severos

[5] Xakata. *Microsoft sube con fuerza el precio de las consolas Xbox y sus mandos en todo el mundo: así quedan en Europa y EE. UU.* https://www.xataka.com/videojuegos/microsoft-sube-drasticamente-precio-consolas-xbox-sus-mandos-todo-mundo-asi-quedan-europa-eeuu

[6] Euronews. *Sony sube los precios de PlayStation 5 un 25% en Reino Unido, Europa y Australia*. https://es.euronews.com/business/2025/04/14/sony-sube-los-precios-de-playstation-5-un-25-en-reino-unido-europa-y-australia

[7] Infobae. *PlayStation 5 vuelve a subir de precio: este es el nuevo costo para España y Europa*. https://www.infobae.com/tecno/2025/04/14/playstation-5-vuelve-a-subir-de-precio-este-es-el-nuevo-costo-para-espana-y-europa/

[8] Randy Pitchford [@DuvalMagic]. (2024, fecha). *CEO de Gearbox habla sobre el precio de Borderlands 4 [Tweet]. Twitter*. https://es.ign.com/borderlands-4/216533/news/el-ceo-de-gearbox-habla-sobre-el-precio-de-borderlands-4-encontraras-la-manera-de-conseguirlo-si-ere

[9] Game Industry Layoffs. (2024). *Gráficos y tablas de despidos en la industria (2022-2024).* https://publish.obsidian.md/vg-layoffs/Archive/2022

[10] Wikipedia. *Despidos en la industria del videojuego.* https://en.wikipedia.org/wiki/2022%E2%80%932025_video_game_industry_layoffs

[11] HobbyConsolas. *Tango Gameworks y Hi-Fi Rush han sido comprados y "resucitados" por Krafton, el editor de PUBG, quien ya piensa en secuela y nuevos juegos.* https://www.hobbyconsolas.com/noticias/krafton-compra-tango-gameworks-hi-fi-rush-ser-resucitados-1399783

[12] Documento de Obsidian público. *Gráfico de Despidos [Imagen].* https://publish.obsidian.md/vg-layoffs/Archive/2022

[13] 3DJuegos. *El jefe de Xbox Game Studios desvela el motivo del cierre de Tango Gameworks. Microsoft no sólo miraba Hi-Fi RUSH, sino también el futuro del estudio.* https://www.3djuegos.com/juegos/hi-fi-rush/noticias/jefe-xbox-game-studios-desvela-motivo-cierre-tango-gameworks-microsoft-no-solo-miraba-hi-fi-rush-sino-tambien-futuro-estudio

[14] OrgulloGamers. *¿Qué es la cultura del crunch en los videojuegos?* https://orgullogamers.com/videojuegos/que-es-la-cultura-del-crunch-en-los-videojuegos/

[15] LiveJournal. *EA: The Human Story.* https://ea-spouse.livejournal.com/274.html

[16] Repositorio Comillas. *La industria del videojuego; la competencia de sus empresas en el ciberespacio, su evolución histórica y su posible futuro.* https://repositorio.comillas.edu/xmlui/handle/11531/69628

[17] Reddit. *El formato físico está MURIENDO.* https://www.reddit.com/r/VideojuegosMX/comments/1kis6r5/el_formato_fisico_esta_muriendo/

[18] Baitibay. *¿La MUERTE del formato FÍSICO? [Video].* https://www.youtube.com/watch?v=LbKyRire6ik

[19] Wikipedia. *Que es Woke.* https://es.wikipedia.org/wiki/Woke

[20] Agencia Presentes. *Que es Woke.* https://agenciapresentes.org/2025/01/23/el-verdadero-sentido-de-ser-woke-y-su-relacion-con-america-latina/

[21] Inna-Vjuzhanina. 2021. *Dibujo de Abby [Imagen].* https://www.deviantart.com/inna-vjuzhanina/art/Abby-867991650

[22] LIMU3MU. 2024. *¿Por qué The Last of Us Parte 2 es woke?* https://www.reddit.com/r/TheLastOfUs2/comments/1ca1oqv/why_is_lastofuspart2_woke_spoiler_alert/?tl=es-es

[23] Reddit. *Yellow Brick Games.* https://www.reddit.com/r/pcgaming/comments/jzi7xj/yellow_brick_games_debuts_with_dragon_age_creator/

[24] Endeavor Rx. *The only doctor prescribed video game treatment for kids with ADHD.* https://www.endeavorrx.com/

[25] Samsung. *The Mind Guardian.* https://www.samsung.com/es/tecnologiaconproposito/accesibilidad-bienestar/themindguardian/

[26] Refatoring. *Patrones de diseño.* https://refactoring.guru/es/design-patterns/catalog

[27] Profile. *Patrones de diseño en los videojuegos: Object Pooling.* https://profile.es/blog/patrones-de-diseno-object-pooling/

[28] Blender. *Web oficial de Blender.* https://www.blender.org/

[29] AutoDesk. *Web official de Maya.* https://www.autodesk.com/es/products/maya/overview

[30] Procreate. *Art is for everyone.* https://procreate.com/

[31] Inkscape. *Web oficial de Ikscape.* https://inkscape.app/

[32] Adobe. *Web official de Subtance Painter.* https://www.adobe.com/es/products/substance3d/apps/painter.html

[33] Wikipedia. *Historia del vampiro.* https://es.wikipedia.org/wiki/Vampiro

[34] Cursiosamente. *¿Cómo evolucionaron los VAMPIROS?* [Video] https://www.youtube.com/watch?v=5g9N8y298L8&pp=ygUUaGlzdG9yaWEgZGVsIHZhbXBpcm8%3D

[35] Estlea Naïad. *El Origen del Mito del Vampiro* [Video]. https://www.youtube.com/watch?v=le_VWuGIQC8&pp=ygUUaGlzdG9yaWEgZGVsIHZhbXBpcm8%3D

[36] Discord. *Grupo de Godot en Discord.* https://discord.com/invite/zH7NUgz

[37] Godot Engine. *Tu motor de juegos gratuito y de código Abierto.* https://godotengine.org/es/

[38] The Good Gamer.

*Desarrollo de un videojuego: en qué etapas se divide.* https://thegoodgamer.es/desarrollo-de-un-videojuego-en-que-etapas-se-divide/

[39] Wikipedia. *Documento de diseño de videojuegos.* https://es.wikipedia.org/wiki/Documento_de_dise%C3%B1o_de_videojuegos

[40] Wikimedia Commons. *Logo de Godot* [Imagen]. https://commons.wikimedia.org/wiki/File:Godot_logo.svg

[41] Microsoft. *Documentación de C#.* https://learn.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/

[42] Godot. *Documentación Oficial de Godot.* https://docs.godotengine.org/es/4.x/index.html

[43] Wikimedia Commons. *Logo de C#* [Imagen]. https://commons.wikimedia.org/wiki/File:C-Sharp_Logo.svg

[44] Wikipedia. *Herramienta CASE*. https://es.wikipedia.org/wiki/Herramienta_CASE

[45] Git. *Web official de Git.* https://git-scm.com/

[46] Wikimedia Commons. *Logo de Git* [Imagen]. https://commons.wikimedia.org/wiki/File:Git-logo.svg

[47] Visual Studio Code. *Web oficial de Visual Studio Code.* https://code.visualstudio.com/

[48] Windows-Tiles. 2021. *Logo de Visual Studio Code* [Imagen]. https://www.deviantart.com/windows-tiles/art/WindowsTiles-Visual-Studio-Code-Medium-900341585

[49] Lucidchart. *Web oficial de Lucidchart.* https://www.lucidchart.com/pages/es

[50] Wikimedia Commons. *Logo de Lucidchart* [Imagen]. https://commons.wikimedia.org/wiki/File:Lucidchart-logo.svg

[51] Obsidian. *Sharpen your thinking*. https://obsidian.md/

[52] Gantt Project. *Web oficial de Gantt Project.* https://ganttproject.biz/

[53] Schwaber, Ken. Sutherland, Jeffrey Victor. *"Software in 30 days [electronic resource]: how Agile managers beat the odds, delight their customers, and leave competitors in the dust /".* https://dama.umh.es/discovery/fulldisplay?docid=alma991001234175106331&context=U&vid=34CVA_UMH:VU1&lang=es

[54] Hoboken, N.J.: John Wiley & Sons, Inc., 2012. Álvarez García, Alonso. Heras del Dedo, Rafael de las / Lasa Gómez, Carmen. *"Manual imprescindible de métodos Ágiles y Scrum".* https://dama.umh.es/discovery/fulldisplay?docid=alma991000466069706331&context=U&vid=34CVA_UMH:VU1&lang=es

[55] Wikipedia. *Casos de Uso.* https://es.wikipedia.org/wiki/Caso_de_uso

[56] Lucidchart. *Diagrama de Clases.* https://www.lucidchart.com/pages/es/tutorial-de-diagrama-de-clases-uml

[57] Miro. *Diagrama de estados*. https://miro.com/es/diagrama/que-es-diagrama-maquina-estados-uml/

[58] Lucidchart. *Diagrama de Secuencia*. https://www.lucidchart.com/pages/es/diagrama-de-secuencia

[59] Lucidchart. *Que es un diagrama de flujo.* https://www.lucidchart.com/pages/es/que-es-un-diagrama-de-flujo

[60] Reddit. *Obtención de los requisitos de los videojuegos.* https://www.reddit.com/r/GameDevelopment/comments/1juj5cv/how_to_get_system_requirements/?tl=es-es

[61] Reddit. *¿Será el desarrollo de Yandere Simulator más largo que el de Duke Nukem Forever?*

https://www.reddit.com/r/Osana/comments/1lly7zp/will_yandere_simulator_development_be_longer_then/?tl=es-419

[62] Icon-Icon. *Logo de Itch.io* [Imagen]. https://icon-icons.com/es/icono/comez%C3%B3n-io/198115

[63] Flickr. *Logo de Steam* [Imagen]. https://www.flickr.com/photos/121483302@N02/14606200823

[64] Icon-Icon. *Logo de Epic Games* [Imagen]. https://icon-icons.com/id/icon/sosial-media-epic-permainan-bermain-logo/228467

# CAPÍTULO 8: ANEXOS

## 8.1 GDD

### 1. Resumen ejecutivo / visión general

**Nombre del juego:** No Harvor Blood - Sangre Sin Refugio

**Género:**
Sigilo, Estrategia, Aventura narrativa.

**Plataforma(s):**
PC.

**Público objetivo:**
Jugadores mayores de 16 años. Aficionados a temáticas de vampiros, sigilo y toma de decisiones. Fans de Dishonored, Thief o Vampyr.

**Pitch corto:**
Encarnas a un vampiro que debe alimentarse de los aldeanos sin ser descubierto, escondiendo cadáveres y confundiendo a los sospechosos, en una ciudad medieval donde cada paso en falso puede delatarte.

**Referencias e inspiración:**

- *Dishonored*, *Thief*, *Vampyr* (jugabilidad e inspiración temática)
- *Hollow Knight* (movimiento)
- *Beat Cop* (estructura de mapa)
- *Beholder*, *Valiant Hearts* (estética visual)

## 2. Historia y ambientación

**Sinopsis del juego:**
El jugador controla a un vampiro recién despertado que debe alimentarse en secreto de los aldeanos de la ciudad de Brujas, Bélgica, durante la Edad Media. Al hacerlo, deja cadáveres que deben ser ocultados para evitar ser descubierto.

**Descripción del mundo o universo:**
Ciudad gótica medieval inspirada en Brujas, Bélgica, con calles estrechas, canales, arquitectura gótica y ambiente sombrío.

**Cronología:**
Edad Media, durante el auge del estilo gótico arquitectónico (siglo XII).

**Estilo narrativo:**
Narración ambiental, minimalista. Historia contada a través de acciones, reacciones de NPCs y eventos del mundo.

**Escenarios y localizaciones:**

- Brujas (ciudad medieval)
- Pozos, montones de paja, ríos, cofres, baños públicos, corrales de cerdos (escondites)
- Casas, herrería, plaza central, iglesia, puente

## 3. Jugabilidad

**Mecánicas básicas:**

- Movimiento lateral, correr, escalar, saltar
- Ataque rápido al NPC más cercano
- Arrastrar cadáveres
- Esconder cuerpos en diferentes lugares
- Esconderse para evitar ser descubierto
- Interacciones sociales indirectas (manipulación de sospechas)

**Objetivos del jugador:**
Sobrevivir, alimentarse, ocultar tus actos, manipular la aldea sin ser descubierto.

**Fases del juego:**

- Introducción/tutorial
- Fase media con aumento de sospechas y guardias
- Final: o eliminas a todos, o eres descubierto



**IA de enemigos y aliados:**

- NPCs siguen rutinas programadas
- Pueden dar la alarma si descubren cuerpos
- Generan sospechas y se las comunican
- Guardias responden a alertas y atacan si detectan al vampiro

## 4. Personajes

**Protagonista(s):**
Vampiro solitario, sin nombre, despierto tras siglos. Hambre intensa lo lleva a actuar.

**Enemigos:**
Guardias de la ciudad: armados, patrullan y atacan al ser alertados.

**NPCs:**
Aldeanos con rutinas (ir a trabajar, conversar, patrullar, etc.), características de personalidad e influenciabilidad.

**Atributos y habilidades:**

- Velocidad
- Sigilo
- Capacidad para arrastrar o esconder cadáveres
- Capacidad de generar sospechas estratégicamente

## 5. Controles

**Esquema de control (teclado y ratón):**

- A/D: movimiento lateral
- W: escalar
- Barra espaciadora: saltar
- F: atacar
- E: interactuar (arrastrar, esconder, etc.)
- Esc: menú de opciones
- Click: ver información de NPC

**Accesibilidad y personalización:**
Se aplican medidas del Libro Blanco de Accesibilidad para Desarrolladores. No se mencionan personalizaciones específicas.

## 6. Interfaz de usuario (UI/UX)

**HUD:**

- Vida (si se implementa)
- Nivel de sospecha global
- Contador de cadáveres ocultos
- Iconos de interacción (cuando se puede esconder un cadáver, por ejemplo)

**Menús:**

- Inicio (nueva partida, cargar partida, salir)
- Pausa (guardar, cargar, reiniciar)
- Opciones (volumen, resolución, controles)

**Diálogos / notificaciones:**

- Alertas de descubrimiento
- Mensajes contextuales (cadáver escondido, guardias alertados, etc.)

## 7. Arte y estética

**Estilo visual:**
2D, simplificado, inspirado en *Beholder* y *Valiant Hearts*.

**Paleta de colores:**
Desaturada, tonos oscuros, marrones, grises y rojos apagados.

**Referencias visuales:**

- *Beholder* (estilo de personajes y ambiente)
- *Valiant Hearts* (fondos y tono general)

**Diseño de personajes y entornos:**

- Personajes con siluetas definidas, sin muchos detalles
- Ambientes góticos, medievales, cargados de atmósfera

---

## 8. Audio

**Música:**
Ambiental, oscura y tensa. Temas variados según el nivel de sospecha o eventos (ataques, descubrimientos).

**Efectos de sonido:**
Pasos, gritos, arrastre de cuerpos, alarma de los aldeanos, ataques del vampiro, sonido del agua en el río.

## 9. Tecnología

**Motor de desarrollo:**
Godot Engine

**Lenguaje(s) de programación:**
C# , GDScript

**Plugins o herramientas externas:**

- Visual Studio Code
- Obsidian
- Lucidchart
- Procreate
- Audacity
- GitHub

## 10. Producción

**Cronograma general:**

- Preproducción: hasta 31 de marzo
- Producción: implementación de sistemas, prototipo funcional
- Postproducción: publicación y feedback

**Metodología:**
Ágil, uso de Kanban

**Roles del equipo:**
Proyecto individual – todo a cargo del desarrollador.

**Recursos necesarios:**

- PC de desarrollo
- Herramientas de software mencionadas
- Assets visuales/sonoros (propios o libres)



## 11. Marketing y distribución

**Público objetivo:**
Jugadores mayores de 16 años, fans del sigilo, la narrativa oscura y los vampiros.

**Competencia directa:**
*Dishonored*, *Thief*, *Vampyr*

**Plataformas de publicación:**
Steam, Itch.io

**Modelo de negocio:**
Inicialmente pago único (3-5€) o distribución gratuita para visibilidad

# 8.2 Diagrama de Gantt