Universidad San Jorge

Escuela de Arquitectura y Tecnología

Grado en Diseño y Desarrollo de Videojuegos

Proyecto Final

Diseño y desarrollo de una herramienta modular para la elaboración de cinemáticas en Unity adaptada a un sistema de diálogos

Autor del proyecto: Jon Imaz Dravasa

Director del proyecto: Enrique Martínez

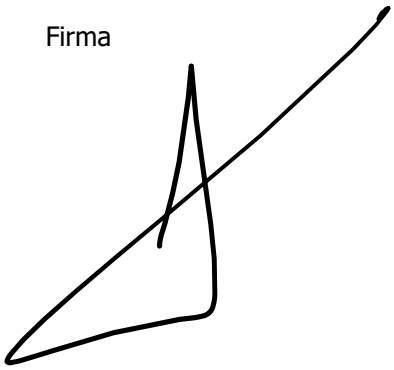
Lugar, 12 de septiembre de 2023

Este trabajo constituye parte de mi candidatura para la obtención del título de Graduado en Diseño y Desarrollo de Videojuegos por la Universidad San Jorge y no ha sido entregado previamente (o simultáneamente) para la obtención de cualquier otro título.

Este documento es el resultado de mi propio trabajo, excepto donde de otra manera esté indicado y referido.

Doy mi consentimiento para que se archive este trabajo en la biblioteca universitaria de Universidad San Jorge, donde se puede facilitar su consulta.

Firma



Fecha

12 de septiembre de 2023

Dedicatoria y Agradecimiento

Con la finalización de este proyecto de fin de grado, quiero tomar un momento para reconocer y agradecer a las personas que han sido fundamentales en este proceso. A lo largo de este proyecto, he aprendido mucho y ha sido una experiencia valiosa para muchos aspectos de mi vida.

En primer lugar, deseo extender mi más sincero agradecimiento a mi mentor, Enrique Martínez. Su conocimiento profundo y su compromiso incansable han sido una fuente inestimable de inspiración para mí. Gracias por dedicar tu tiempo, por compartir tus experiencias y por tu paciencia inagotable al orientarme a través de cada desafío.

También quisiera hacer una mención a Sergio Jimeno. Aunque no ha sido mi tutor directo, siempre ha estado disponible, dispuesto a ayudar y aclarar cualquier duda que tuviera, haciendo que mi experiencia durante este proyecto fuera aún más enriquecedora. Su generosidad en compartir su conocimiento y su actitud amigable han sido un apoyo adicional inestimable.

A mi familia, quiero decirles que son la razón de mis logros. Aunque la distancia nos haya separado físicamente en esta temporada, la resonancia de su apoyo y sacrificios me ha acompañado en cada paso. Este logro es tan mío como suyo, y es un testimonio de las bases que han establecido en mi vida.

También quiero dar las gracias a mis amigos, especialmente a Marta Simón y Germán Sánchez. Vosotros habéis estado a mi lado durante todo este proceso, y vuestro apoyo y ánimo han sido increíbles. Los buenos amigos hacen que los desafíos sean más fáciles de enfrentar.

Por último, una dedicatoria especial para mi novia, Hannah Pollet, mi compañera y confidente. En los momentos más difíciles, cuando la presión parecía insuperable, encontré consuelo y fuerza en tu apoyo y amor incondicional. Tu capacidad de comprenderme y tranquilizarme han sido un pilar en el que me he podido apoyar cuando lo he necesitado. No hay palabras que puedan expresar lo agradecido que estoy por tener a alguien tan maravillosa a mi lado.

En resumen, gracias a todos los que han contribuido de alguna manera a este proyecto. Vuestro apoyo ha sido invaluable, y estoy agradecido por cada uno de vosotros.

Índice de contenido

Índice de Figuras

Índice de Tablas

Resumen

Este proyecto, llevado a cabo en el motor Unity, se centró en la creación y adaptación de una herramienta específica para la gestión de cinemáticas. Si bien existen diversas herramientas en el mercado, el propósito principal no era superar lo existente, sino comprender profundamente las metodologías y técnicas involucradas en la creación de herramientas de desarrollo. ¿Y qué mejor manera de lograr este aprendizaje que inmerso en un entorno empresarial real y en el dinámico mundo del desarrollo de videojuegos?

El viaje de desarrollo abarcó desde familiarizarse con herramientas profesionales hasta la efectiva integración de la herramienta dentro de un proyecto real de videojuego. La herramienta resultante, si bien intuitiva y eficiente, sirvió como un medio para entender los desafíos, soluciones y compromisos inherentes al desarrollo de software para videojuegos.

Una fase crítica del proyecto fue la evaluación, donde se buscó *feedback* de profesionales del sector. Los resultados proporcionaron no solo validación, sino también áreas para futuras optimizaciones y mejoras.

En resumen, más allá de la creación de una herramienta, este proyecto ha sido una inmersión educativa sobre cómo se construyen y adaptan soluciones en el mundo real de desarrollo de videojuegos, y cómo se equilibran las necesidades técnicas con la visión y la funcionalidad.

Palabras clave: Desarrollo de videojuegos, Herramienta de cinemáticas, Unity, Software, Integración de videojuegos, Optimización de herramientas, Aprendizaje técnico, Entorno empresarial, Gestión de cinemáticas.

Abstract

This project, carried out in the Unity engine, focused on the creation and adaptation of a specific tool for the management of kinematics. Although there are several tools on the market, the main purpose was not to overcome the existing ones, but to deeply understand the methodologies and techniques involved in the creation of development tools, and what better way to achieve this learning than immersed in a real business environment and in the dynamic world of videogame development?

The development journey ranged from familiarization with professional tools to the effective integration of the tool into a real video game project. The resulting tool, while intuitive and efficient, served to understand the challenges, solutions and trade-offs inherent in video game software development.

A critical phase of the project was the evaluation, where feedback was sought from industry professionals. The results provided not only validation, but also areas for future optimization and improvement.

In summary, beyond the creation of a tool, this project has been an educational immersion into how solutions are built and adapted in the real world of game development, and how technical needs are balanced with vision and functionality.

Keywords: Game development, Cinematic tools, Unity, Software, Game integration, Tool optimization, technical learning, Business environment, Cinematic management.

1. Introducción

1.1. Descripción del Problema

En la industria del desarrollo de videojuegos, uno de los elementos más importantes de la narración inmersiva y efectiva son las cinemáticas (Contributor, 2022). Las cinemáticas son escenas renderizadas previamente o en tiempo real que ayudan a impulsar la trama, presentar a los personajes, crear el ambiente y mucho más. Sin embargo, la animación puede ser un proceso complejo y tedioso que involucra una combinación compleja de programación, arte, diseño y animación.

Además, los sistemas de diálogo son una parte integral de las narrativas de los videojuegos y deben integrarse con estas historias para garantizar una experiencia de juego fluida y coherente (Jedruszczak, 2016). Sin embargo, en muchos casos la creación de diálogos y su realización en el cine se realiza utilizando diferentes herramientas que no siempre están diseñadas para trabajar juntas. Este desajuste puede generar inconsistencias, errores y un desarrollo más lento y menos eficiente.

Para complicar aún más las cosas, muchas de las herramientas disponibles son complejas de usar y pueden requerir conocimientos avanzados de programación. Esto puede hacer que sean inaccesibles para los diseñadores de juegos y otros miembros del equipo de desarrollo que pueden no tener una sólida formación técnica.

Por lo tanto, el principal problema a abordar en este proyecto de fin de grado es la falta de una herramienta integrada y fácil de usar para la creación de cinemáticas y diálogos en Unity [1]. Esta herramienta debería permitir a los desarrolladores crear, implementar y modificar las cinemáticas y diálogos de manera eficiente, reduciendo el esfuerzo necesario para estas tareas, y permitiendo a los miembros del equipo con diferentes niveles de habilidad técnica contribuir de manera efectiva en el desarrollo del videojuego. Además, debería ser capaz de interactuar con Articy Draft [2], una de las herramientas más populares para la creación de diálogos en la industria del desarrollo de videojuegos.

1.2. Antecedentes y Justificación del Proyecto

El desarrollo de videojuegos es un campo interdisciplinario que requiere una amplia gama de habilidades y herramientas. Desde la programación y el diseño de niveles hasta la animación y el guion, hay muchos aspectos a considerar para crear una experiencia de juego cohesiva y atractiva. A medida que los videojuegos se vuelven más complejos y narrativos, aumenta la necesidad de herramientas eficientes para crear y administrar cinemáticas y diálogos (Balson, 2018).

La importancia de las cinemáticas y el diálogo en los videojuegos no puede subestimarse. Las cinemáticas agregan profundidad a la historia y crean momentos memorables que los jugadores recordarán mucho después de haber terminado el juego. Por otro lado, el diálogo ayuda a desarrollar los personajes y la trama, y brinda información valiosa que puede mejorar la experiencia de juego. Sin embargo, crear animaciones y diálogos puede ser un proceso complejo y que consuma mucho tiempo.

1 https://unity.com/es

2 https://www.articy.com/en/

En el pasado, los desarrolladores tenían que usar varias herramientas para crear escenas y diálogos, para luego tratar de integrarlos en el juego. Esto no solo es ineficiente, sino que también puede conducir a errores y falta de consistencia. Además, muchas de estas herramientas son difíciles de usar y requieren amplios conocimientos de programación, lo que limita su accesibilidad.

En este contexto, Entalto Studios, una empresa de desarrollo de videojuegos con un fuerte enfoque en la narrativa, reconoció la necesidad de una herramienta más integrada y accesible. Su objetivo es simplificar y acelerar el proceso de creación de cinemáticas y diálogos, proporcionando una solución que se integre a la perfección con el motor de videojuegos Unity y la herramienta de diálogos Articy Draft.

Este proyecto es importante y relevante por varias razones. Primero, ayudará a resolver los problemas existentes en el desarrollo de juegos al mejorar la eficiencia y la consistencia de la creación de cinemáticas y diálogos. En segundo lugar, el proyecto puede hacer que las herramientas de desarrollo de videojuegos sean más accesibles para un público más amplio, centrándose en la facilidad de uso. En última instancia, el proyecto contribuirá a la base de conocimientos existentes en el desarrollo de videojuegos y puede allanar el camino para una mayor investigación y desarrollo en esta área.

1.3. Objetivos del Proyecto

El principal propósito de este proyecto es el diseño y desarrollo de una herramienta modular y adaptable para la elaboración de cinemáticas y un sistema de diálogos en Unity, diseñada específicamente para su uso en el próximo videojuego oficial de Vampiro: La mascarada, un RPG isométrico centrado en la narrativa. Los objetivos del proyecto están diseñados para satisfacer las necesidades actuales de Entalto Studios y del videojuego en desarrollo. Estos objetivos se detallan a continuación:

1. **Diseño de una herramienta de creación de cinemáticas en Unity:** Este objetivo implica el diseño de una herramienta que permita la creación y gestión de cinemáticas de manera eficiente y efectiva en Unity. Esta herramienta deberá ser lo suficientemente flexible como para adaptarse a diferentes estilos de cinemáticas y a diferentes requerimientos del proyecto.
2. **Desarrollo de un sistema de diálogos integrado:** El sistema de diálogos deberá ser fácilmente integrable con las cinemáticas creadas en Unity y con Articy Draft. Este sistema también deberá ser intuitivo para permitir a los diseñadores de videojuegos crear y gestionar diálogos de manera eficiente.
3. **Integración de la herramienta con el videojuego en desarrollo:** Este objetivo implica implementar la herramienta desarrollado directamente en el videojuego actualmente en desarrollo. La herramienta deberá ser capaz de trabajar de manera efectiva con los sistemas existentes en el juego y satisfacer las necesidades específicas del proyecto.
4. **Desarrollo de una interfaz de usuario accesible y fácil de usar:** La herramienta deberá ser accesible y fácil de usar para un amplio rango de usuarios, incluyendo aquellos sin un fuerte trasfondo técnico. Esto implica diseñar una interfaz de usuario intuitiva y proporcionar la documentación adecuada.
5. **Validación y pruebas de la herramienta en un entorno real:** La herramienta desarrollada deberá ser probada y validada dentro del proyecto del videojuego en desarrollo para asegurar que cumple con los requerimientos y las expectativas del equipo de desarrollo.

Estos objetivos ayudarán a dirigir el desarrollo de la herramienta y asegurar que se alinean con las necesidades de la empresa y del videojuego en desarrollo. Además, la consecución de estos

objetivos permitirá al equipo de desarrollo del videojuego avanzar a un ritmo más rápido y eficiente, y permitirá que todos los departamentos puedan contribuir de manera más efectiva al desarrollo del videojuego en cuestión.

1.4. Metodología del Trabajo

La metodología Scrum ha sido elegida para desarrollar este proyecto. Scrum es una estructura de trabajo ágil que permite desarrollar e implementar proyectos en fases, enfocado en obtener el máximo valor en el menor tiempo posible. Aunque el método Scrum es ampliamente utilizado y efectivo, ha sido difícil de implementar por completo debido a la carga de trabajo. A pesar de estos desafíos, se ha tratado de utilizar los elementos más importantes de Scrum que fueran aplicables y valiosos en la gestión del proyecto.

Herramientas:

Para la organización y gestión de tareas, se ha utilizado Trello [3] como herramienta de administración de proyectos, aunque funcionó principalmente como *backlog*, tal y como se puede observar en la figura 1, y no fue la opción más adecuada para la administración del proyecto. Para la comunicación y coordinación se empleó Microsoft Teams [4] y Discord [5].

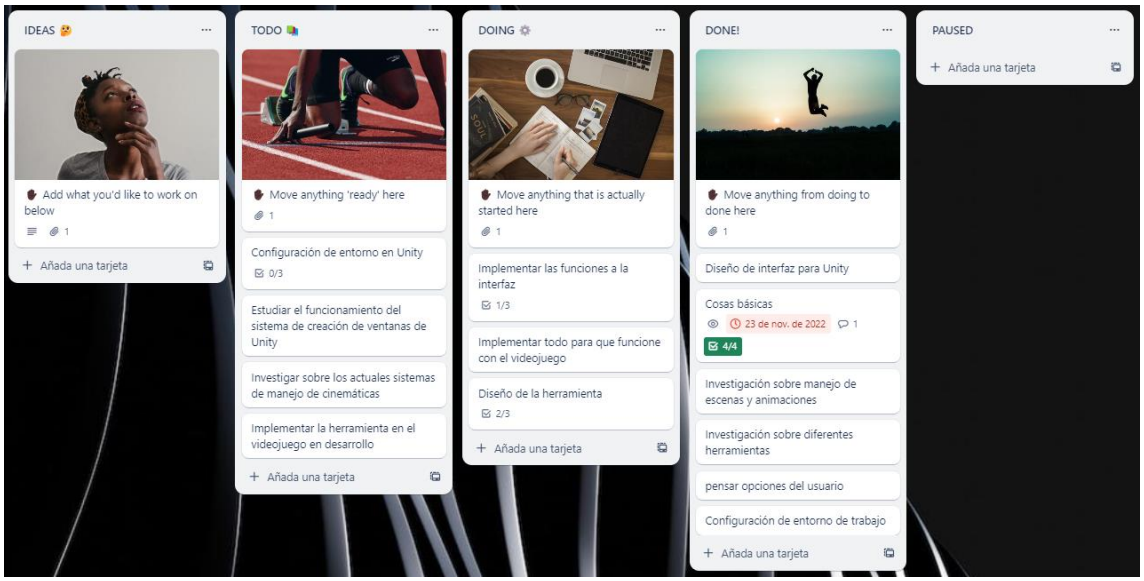


Figura 1: Gestión de tareas mediante Trello

FUENTE: ELABORACIÓN PROPIA

Reuniones:

Se realizaron reuniones regulares para discutir el progreso del proyecto y recibir retroalimentación. En este caso, el tutor actuó como cliente dentro de la metodología Scrum. A continuación, se muestra un resumen de las reuniones:

08/02/2023: Se discutió la selección de las herramientas y las características esenciales para el diseño de la herramienta de creación de cinemáticas y diálogos.

3 https://trello.com/es

4 https://www.microsoft.com/es-es/microsoft-teams/log-in

5 https://discord.com/

23/02/2023: Se presentó un esquema inicial del diseño de la herramienta y se recibió feedback constructivo del tutor.

10/03/2023: En esta reunión, se abordó el progreso en el desarrollo del sistema de diálogos. También se discutió la integración de este sistema con Unity.

06/04/2023: Se retomó el proyecto y se presentó un prototipo del sistema de diálogos integrado con las cinemáticas.

27/04/2023: En esta reunión, se discutió la importancia de la interfaz de usuario y cómo hacerla accesible y fácil de usar.

18/05/2023: Se revisó la primera versión de la interfaz de usuario. También se hizo hincapié en la necesidad de realizar ciertas mejoras.

05/06/2023: Se presentó la interfaz de usuario revisada. También se discutió el plan para las pruebas y la validación de la herramienta.

26/06/2023: Se revisó el proceso de pruebas y validación y se acordaron los próximos pasos para finalizar el proyecto.

17/07/2023: Se presentó una actualización del estado de las pruebas. Se identificaron algunos problemas menores y se planificaron los pasos para solucionarlos.

07/08/2023: En esta reunión, se presentaron los resultados finales de las pruebas y validación. Se discutió en plan para la finalización y entrega del proyecto.

02/09/2023: Se discutió la finalización del proyecto, se revisaron los resultados y se preparó el informe final para la entrega.

Problemas encontrados:

Durante el desarrollo del proyecto, se encontraron varios desafíos, como la carga de trabajo académica que condicionaba y dificultaba la implementación completa de la metodología Scrum, por lo que se han tenido que realizar muchas tomas de decisiones que han afectado al proyecto. Por ejemplo, ciertas características previamente propuestas tuvieron que ser priorizadas o pospuestas para ajustarse a las restricciones de tiempo, y la frecuencia de las reuniones Scrum se vio reducida, lo que a veces derivó en una comunicación menos fluida entre los miembros del equipo. Estos factores, aunque desafiantes, nos enseñaron a adaptarnos y a reajustar nuestras estrategias conforme avanzábamos en el proceso.

1.4.1. Fases de la Metodología

Planificación inicial y preparación

En esta etapa inicial, se han definido los objetivos y el alcance del proyecto.

- **Análisis de necesidades:** El primer paso fue comprender y definir las necesidades del videojuego en desarrollo. Esto implicó reuniones con los desarrolladores del juego y la revisión de documentación existente para identificar las funcionalidades que debería

tener la herramienta. Asimismo, se realizó un estudio del uso de Articy draft en el proyecto para asegurar una integración óptima.

- **Establecimiento de objetivos:** Basándose en el análisis de necesidades, se establecieron los objetivos del proyecto. Estos objetivos proporcionaron una visión clara de lo que se debía lograr y permitieron mantener el enfoque durante todo el proyecto.
- **Investigación de tecnologías:** Una vez se tuvieron claros los objetivos y necesidades del proyecto, se realizó una investigación de las tecnologías disponibles que podrían ser útiles para el desarrollo de la herramienta. Esto incluyó un estudio exhaustivo de Unity, Articy Draft y otros recursos relacionados.
- **Planificación de la arquitectura de la herramienta:** Se diseñó un boceto preliminar de la arquitectura de la herramienta. Este diseño proporcionó una guía para la implementación y ayudó a anticipar posibles problemas. El diseño de la arquitectura incluyó la definición de módulos, las interacciones entre estos y cómo se integrarían con el videojuego en desarrollo y Articy Draft.
- **Creación del backlog del producto:** Basándose en los objetivos establecidos y el diseño de la arquitectura, se creó el backlog del producto. Este incluiría todas las funcionalidades y tareas que se debían implementar durante el desarrollo de la herramienta.
- **Establecimiento del calendario del proyecto:** Por último, se estableció un calendario para el proyecto. Esto incluye la definición de las fechas de inicio y fin de cada sprint, así como la planificación de las reuniones de seguimiento y las revisiones del proyecto.

Sprints de desarrollo

El trabajo se dividió en varios sprints, que eran intervalos cortos y fijos de entre 2 semanas a 1 mes. Cada sprint funciona en un conjunto específico de características seleccionadas del backlog del producto. Al final de cada sprint, el proyecto se vería elevado, es decir, cada vez el proyecto iría teniendo mayores capacidades.

- Sprint 1: Definición de los requisitos y características del sistema. Se identificarán los requisitos clave y las características que la herramienta necesita tener. Se hará también una investigación y se consultará con los miembros del equipo del videojuego para asegurarse de que la herramienta será útil.
- Sprint 2: Se realizarán bocetos de diseño de la interfaz de usuario, se decidirá cómo se implementará cada característica de la herramienta.
- Sprint 3: Durante este sprint, se trabajará en la programación de la interfaz de usuario según lo decidido en el sprint anterior.
- Sprint 4: Se realizarán pruebas de la interfaz de usuario y se recopilarán comentarios para mejorarla. Esto incluirá pruebas de usabilidad y pruebas de interacción.
- Sprint 5: Se implementará el sistema de cinemáticas según las necesidades y requisitos definidor en el Sprint 1.
- Sprint 6: Se realizarán pruebas del sistema de cinemáticas y se recopilarán comentarios para su mejora.
- Sprint 7: Se trabajará en la integración de la herramienta con Articy Draft.
- Sprint 8: Se realizarán pruebas de la integración y se recopilarán comentarios para mejorarla.
- Sprint 9: Se dedicará este sprint a corregir cualquier error encontrado en los sprints anteriores y a hacer las mejoras necesarias.
- Sprint 10: Se preparará la versión final del sistema para su lanzamiento, se redactará la documentación y se presentará el proyecto.

Una vez definidos los sprints del proyecto, es el momento de representarlos en un diagrama de Gantt para ver los intervalos de tiempo de forma más visual.

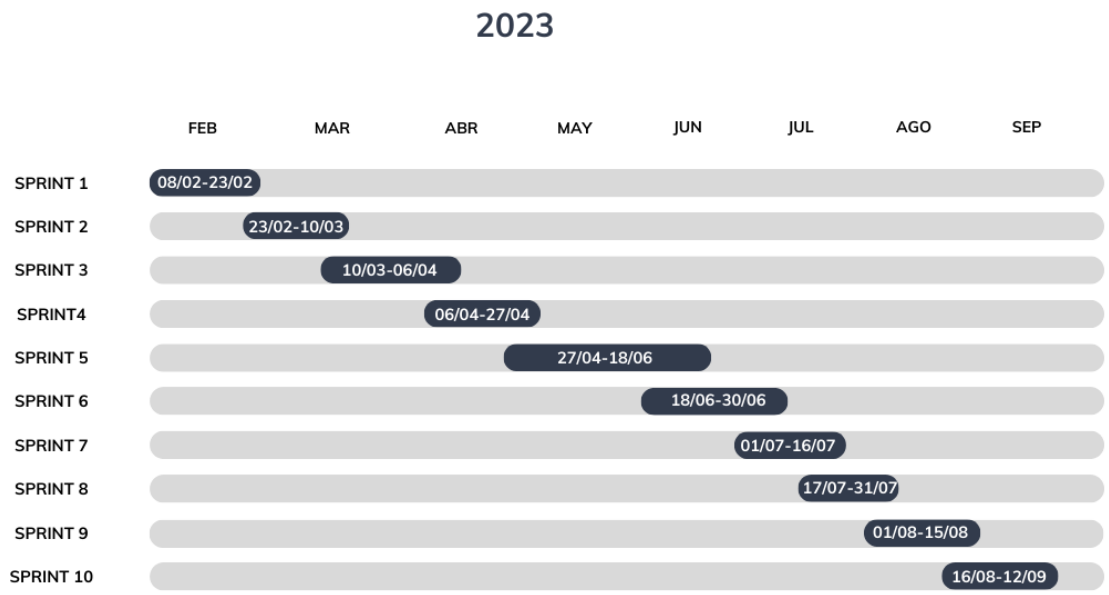


Figura 2: Diagrama de gantt de los sprints

FUENTE: ELABORACIÓN PROPIA

Revisiones y retroalimentación

Al finalizar cada sprint, se realiza una evaluación del trabajo realizado. Esto permite recibir comentarios y ajustar el progreso del proyecto según sea necesario. Si bien las sesiones de revisión suelen ser menos formales, estas fueron cruciales para garantizar que el proyecto estuviera en el camino correcto.

Pruebas y validación

En esta etapa se realizaron pruebas para verificar la efectividad y precisión de la herramienta creada. Esto incluye probar la integración con el videojuego en desarrollo y verificar el correcto funcionamiento del sistema.

Documentación y entrega

Finalmente, se registraron los resultados, problemas y las soluciones implementadas. La documentación es crucial para garantizar que el conocimiento obtenido del proyecto esté disponible para trabajos y posibles desarrollos futuros. La entrega final incluyó tanto el código fuente como la documentación detallada del proyecto.

1.4.2. Tecnologías Aplicadas

Para llevar a cabo este proyecto, se han utilizado una serie de tecnologías que han jugado un papel vital en el desarrollo de la herramienta. Estas tecnologías se detallan a continuación:

Unity: Esta plataforma ha sido elegida el motor principal para desarrollar la herramienta de cinemáticas. Unity se destaca como uno de los motores de videojuegos más populares y ampliamente utilizados (Singh, 2022). Aparte de su popularidad, Unity posee varias características y funcionalidades que resultaron vitales para nuestro proyecto, tales como:

- Motor de físicas avanzado: Unity ofrece un sistema de físicas altamente detallado y preciso, lo que permite crear interacciones realistas dentro del juego.

- Interfaz de usuario personalizable: Con Unity, se puede diseñar y adaptar la interfaz de usuario a las necesidades específicas, proporcionando una experiencia de usuario más intuitiva.

Articy Draft: Articy Draft se utilizó para gestionar los diálogos del videojuego. Esta herramienta permite a los escritores y diseñadores de juegos crear y organizar diálogos de manera eficiente. La herramienta desarrollada en este proyecto se integró con Articy Draft para permitir una fácil transferencia de diálogos al motor de juego.

C#: Este es el lenguaje de programación utilizado para desarrollar la lógica y las funcionalidades de la herramienta. C# es compatible con Unity y permite a los desarrolladores escribir códigos de alto rendimiento que pueden ser fácilmente integrados en el motor de juegos.

2. Estado del Arte

2.1. Cinemáticas

Las cinemáticas en videojuegos se refieren a las secuencias de animación que cuentan una historia o establecen el contexto para una parte del juego. Suelen ser secuencias predefinidas que ayudan a desarrollar el argumento, presentar personajes, o ambientar una escena, y generalmente no implican la participación del jugador (Cutscene (Concept) - Giant Bomb, 2021).

Existen varios tipos de cinemáticas en videojuegos:

- **Cinemáticas Pre-renderizadas:** Estas son cinemáticas que han sido completamente renderizadas y animadas de antemano. Son similares a las películas en miniatura y a menudo ofrecen la más alta calidad de gráficos y animación. Estas cinemáticas son ideales para escenas importantes o dramáticas, pero debido a que están pre-renderizadas, no permiten la interactividad durante la secuencia (Pre-Rendered Graphics, n.d.).
- **Cinemáticas en tiempo real:** Estas cinemáticas utilizan el motor gráfico del juego en tiempo real para renderizar la secuencia. Esto significa que la calidad visual de la secuencia está limitada por las capacidades del motor del juego, pero a cambio, las cinemáticas en tiempo real pueden ser interactivas y pueden reflejar las decisiones y acciones previas del jugador en el juego (Wikipedia contributors, 2023).
- **Cinemáticas Interactivas:** Una cinemática interactiva es una secuencia predefinida en la que los jugadores tienen alguna forma de interacción o control. A menudo, este tipo de cinemáticas permiten al jugador tomar decisiones que pueden afectar el resultado de la secuencia o incluso el desarrollo posterior del juego.
- **Secuencias de Juego Guiado:** Es un tipo de cinemática en el que la jugabilidad se fusiona con elementos de la narrativa cinematográfica. Estas secuencias son diseñadas para crear una experiencia más inmersiva y emocionantemente atractiva. Aunque permiten al jugador mantener el control sobre el personaje, suelen limitar ciertas acciones o guían al jugador en una dirección determinada para avanzar en la trama. Se diferencian de las cinemáticas interactivas en que, mientras las cinemáticas interactivas otorgan al jugador decisiones que pueden afectar el desenlace o el curso de la historia, las secuencias de juego guiado tienen un resultado predeterminado y su propósito principal es mantener la continuidad narrativa.

2.2. Herramientas de Cinemáticas para Videojuegos

Las cinemáticas y secuencias interactivas han emergido como elementos cruciales en la creación de videojuegos modernos, realzando la experiencia del jugador al proporcionar mayor inmersión y una narrativa más rica (Jenkins, 2021). A continuación, se muestra una descripción general de las herramientas más populares en el campo, examinando sus capacidades, limitaciones y usos típicos.

Unity Timelines [6]: Esta es una herramienta nativa de Unity, que permite a los desarrolladores diseñar cinemáticas y secuencias de tiempo complejas. Usando su interfaz de usuario visual, le permite manipular objetos y eventos a lo largo de una línea de tiempo. Aunque es potente y está fuertemente integrado en Unity, requiere mucho conocimiento técnico para usarlo y sus capacidades pueden ser excesivas para pequeños desarrollos (Pluralsight and Unity Technologies Partner to Equip Game Developers With Skills to Create Games in Unity, 2017).

6 https://docs.unity3d.com/Packages/com.unity.timeline@1.2/manual/index.html

Playmaker [7]: Es un complemento de Unity que presenta la capacidad de programar visualmente utilizando máquinas de estado finito. Esta no es una herramienta de cinemáticas, aunque se puede utilizar para crear secuencias interactivas. Su enfoque de programación visual es amigable para los no programadores, pero puede estar limitado para secuencias de comandos complejas o interacciones avanzadas (French, 2022).

Por ejemplo, si estás creando una cinemática interactiva que tiene múltiples ramificaciones y condiciones, puede ser complicado manejar estas interacciones en Playmaker. Podría convertirse en un laberinto de estados y transiciones que puede ser difícil de rastrear y mantener.

Articy Draft: Esta es una herramienta de gestión de contenido y diseño narrativo que se puede utilizar para crear secuencias de eventos y diálogos complejos y ramificados. Articy Draft tiene una excelente organización de contenido con Unity, pero se enfoca en la narrativa en lugar de las escenas en sí (Features List, n.d.).

Además de estas herramientas, también se consideró relevante **Unreal Engine's Sequencer [8]**. Aunque está fuera del contexto de Unity, la herramienta de cinemáticas de Unreal Engine [9] tiene capacidades impresionantes, permitiendo a los desarrolladores crear cinemáticas de alta calidad casi al nivel de las producciones cinematográficas. Además, Sequencer tiene un poderoso sistema de animación que permite a los desarrolladores manipular personajes y objetos en gran detalle (Controlling Anim Instances With Sequencer, n.d.). Puedes, por ejemplo, sincronizar perfectamente el movimiento de los personajes con el diálogo, los gestos y las acciones en la escena, creando una animación fluida y realista. Esto lo convierte en una referencia en la industria para la creación de cinemáticas, sin embargo, su curva de aprendizaje es notablemente más pronunciada.

Cada una de estas herramientas tiene sus propias ventajas y desventajas, y qué herramienta elegir depende en gran medida de las necesidades específicas del proyecto. En el caso de nuestro proyecto, buscamos una solución que sea fácil de usar, pero que también tenga la flexibilidad para adaptarse a las necesidades de un videojuego en desarrollo.

2.3. Uso y Aplicaciones en Unity y Articy Draft

Unity y Articy Draft son herramientas versátiles que pueden ser aplicadas de muchas formas en el desarrollo de videojuegos (Thielmann, 2017). A continuación, se detalla su uso y aplicaciones en el contexto de este proyecto.

2.3.1. Unity

Unity es un motor de videojuegos multiplataforma altamente flexible y potente que permite a los desarrolladores crear y distribuir juegos en una variedad de plataformas, incluyendo Windows, Mac, iOS, Android y muchas más. En este proyecto, se utilizará Unity en una serie de formas clave.

- **Renderizado y Animación:** Unity proporciona un conjunto de herramientas avanzadas para la creación de escenas y animaciones de alta calidad. Estas capacidades se utilizarán

7 https://assetstore.unity.com/packages/tools/visual-scripting/playmaker-368

8 https://docs.unrealengine.com/4.26/en-US/AnimatingObjects/Sequencer/Overview/

9 https://www.unrealengine.com/es-ES

para dar vida a las cinemáticas y eventos del juego (Advanced Animation — CC/iC Unity Tools 1.3.0 Documentation, 2021).

- **Programación y Scripting:** Unity utiliza C# como su principal lenguaje de programación, lo que permite una amplia gama de funcionalidades de scripting. Se utilizarán scripts para controlar el comportamiento del juego, incluyendo la lógica de las cinemáticas y la interacción con el usuario.
- **Unity Timelines:** Como se mencionó anteriormente, Unity Timelines será una herramienta clave en la creación de cinemáticas de este proyecto. Se utilizará para sincronizar eventos y controlar la progresión de las cinemáticas.

2.3.2. Articy Draft

Articy Draft es una herramienta especializada en la creación y gestión de contenido narrativo en videojuegos (Shylenok, 2019). En este proyecto, Articy Draft será utilizado en las siguientes formas:

- **Creación de la narrativa:** Articy Draft permite a los desarrolladores crear narrativas complejas con múltiples ramas. Esta funcionalidad será esencial para diseñar la historia y los eventos del juego.
- **Gestión de contenidos:** Además de su potente herramienta de narrativa, Articy Draft también proporciona capacidades robustas de gestión de contenidos. Se utilizará para organizar y gestionar todos los elementos narrativos del juego, desde los diálogos hasta las descripciones de los personajes y las ubicaciones.
- **Integración con Unity:** A través de su plugin de Unity, Articy draft permite importar directamente su contenido al motor de juego. Esta funcionalidad será crucial para integrar la narrativa con las cinemáticas y eventos programados en Unity.

2.4. Desarrollo de Interfaces de usuario en Unity

Diseñar y desarrollar una interfaz de usuario (UI) eficaz es una parte importante del desarrollo de videojuegos. Un buen diseño de interfaz de usuario mejora la accesibilidad, hacer que la herramienta sea más fácil de entender para los usuarios y, en última instancia, mejorar la experiencia de usuario (Djatmiko, 2020). El desarrollo de la interfaz de usuario de Unity se gestiona mediante el sistema de UI integrado, que proporciona las herramientas necesarias para crear interfaces de usuario atractivas e interactivas.

En este proyecto, el desarrollo de la interfaz de usuario se centrará en tres áreas principales, las cuales serán explicadas en mayor profundidad en el apartado [5.3 Interfaz de Usuario y Usabilidad]:

2.4.1. Diseño de UI

El diseño de la interfaz de usuario incluirá la creación de un esquema visual coherente y atractivo para la herramienta de cinemáticas. El objetivo es crear una interfaz intuitiva y fácil de usar manteniendo las funcionalidades de esta lo más accesibles posibles. Esto implicará la selección de fuentes, colores y estilo gráficos adecuados, así como la disposición de los elementos de la UI de una manera que promueva una experiencia de usuario fluida (Aziz, 2023).

2.4.2. Programación de UI

La programación de UI se refiere a la lógica detrás de la creación de interfaces de usuario. En Unity, esto se realiza principalmente mediante el uso de scripts de C#. Los scripts controlan el comportamiento de la interfaz de usuario, por ejemplo, cómo se mueven y cambian los elementos de la interfaz en respuesta a las acciones del usuario (French, 2022).

2.4.3. Pruebas de UI

Una vez que la interfaz de usuario está diseñada e implementada, es importante probarla para asegurarse de que funciona y es fácil de usar (Degen, 2011). Las pruebas de interfaz de usuario pueden incluir verificar que los botones y los menús funcionen según lo previsto, que la interfaz sea intuitiva y fácil de navegar, y que no haya problemas de usabilidad que puedan molestar a los usuarios que lo vayan a utilizar.

2.4.4. Mejora y Evolución

Como parte del proceso de desarrollo ágil, el diseño y la implementación de la UI deberán estar sujetos a un proceso constante de iteración y mejora. Esto significa que el diseño inicial de la UI probablemente cambiará a medida que se reciba *feedback* de los usuarios y se identifiquen oportunidades de mejora (Ferreira, 2007).

2.5. Gestión de Datos y Animaciones en Videojuegos

Trabajar con datos y animaciones en videojuegos es un componente clave para crear una experiencia de juego inmersiva (Mizuguchi, 2001). Esta sección detalla cómo integrar y administrar estos dos aspectos en el desarrollo de videojuegos, en referencia al contexto de nuestra herramienta de cinemáticas para Unity.

2.5.1. Gestión de datos

El procesamiento de datos en los videojuegos se refiere a cómo se recopilan, almacenan, procesan y utilizan los datos en los juegos. En el contexto de una herramienta de cinemáticas, esto puede incluir varios tipos de datos, incluidos los detalles de los personajes, la configuración del entorno, los estados de la historia, los elementos de la interfaz de usuario y más (Stone, 2019).

Almacenamiento y Recuperación de Datos

Para nuestro propósito, es muy importante comprender cómo Unity administra el almacenamiento de datos. Unity permite guardar y cargar datos utilizando una variedad de métodos (Talbert, 2021), como la serialización de objetos, PlayerPrefs, bases de datos y archivos de texto, los cuales serán explicados en mayor profundidad en la sección [4.5 Herramientas de Gestión de Datos]. Elegir el enfoque adecuado depende en gran medida de las necesidades específicas del proyecto y de la complejidad de los datos que se van a gestionar.

En nuestro caso, se tendrán que tratar con datos relacionados con escenas de cinemáticas, que pueden contener una gran cantidad de elementos y variables. Por lo tanto, se debe pensar detenidamente sobre cómo estructurar y almacenar estos datos de una manera que facilite su recuperación y procesamiento.

Manipulación de Datos

Una vez que se guardan los datos, deben recuperarse y procesarse según lo requiera la herramienta de cinemáticas. Las estructuras de datos juegan un papel crucial aquí. Estos pueden ser tan simples como variables primitivas de Unity (int, float, string, etc.) o tan complejas como listas y diccionarios personalizados para cada situación en concreto.

En el contexto de la herramienta de cinemáticas, es posible que se necesiten procesar datos que representen escenas, interacciones de personajes, eventos del juego, etc. Por lo tanto, es

fundamental seleccionar y diseñar estructuras de datos que puedan manejar eficazmente esta complejidad.

2.5.2. Gestión de Animaciones

La gestión de animaciones en los videojuegos es igualmente importante (Hancock, 2002) y puede referirse a cómo se crean, importan, controlan y modifican las animaciones durante el juego. Unity proporciona un poderoso conjunto de herramientas para crear y administrar animaciones, y comprender cómo funcionan y cómo usarlas es esencial para el desarrollo de nuestra herramienta de cinemáticas.

Creación e Importación de Animaciones

Las animaciones del juego se pueden crear directamente en Unity usando el sistema de animación de Unity o se pueden importar desde un software de animación externo como Blender [10] o Maya [11] (Jiovaine, 2018). En ambos casos, las animaciones generalmente se guardan como "clips" de animación que se pueden controlar y manipular mediante scripts y el programa de estado de animación de Unity.

Para la herramienta de cinemáticas, se tendrá que trabajar con una amplia variedad de animaciones que representen el rango de posibles acciones y reacciones de los personajes. Esto podría incluir todo, desde simples animaciones de caminar y correr, hasta más complejas animaciones de interacción y conversación.

Control de Animaciones

Una vez que se tienen las animaciones en Unity, estas pueden ser controladas mediante el Animator Controller (Kristiel, 2016), que funciona como una máquina de estados para las animaciones tal y como se puede observar en la siguiente figura.

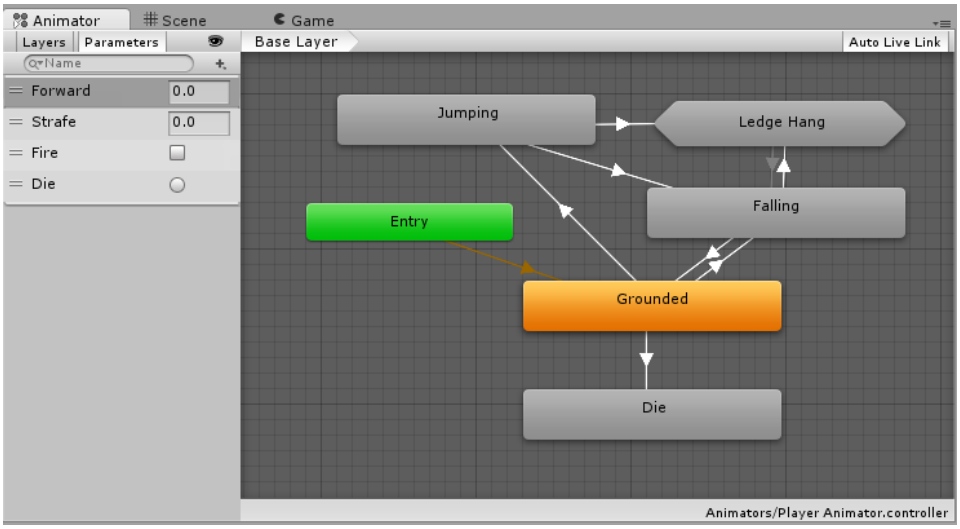


Figura 3: Ejemplo de Animator Controller en Unity

Fuente: Externa [12]

10 https://www.blender.org/

11 https://www.autodesk.es/products/maya/overview?term=1-YEAR&tab=subscription

12 Fuente de Figura 3: (Technologies, 2023)

En este sistema, cada estado representa un clip de animación o una "mezcla" de varios clips, y las transiciones entre los estados se controlan mediante parámetros que pueden ser ajustados en un tiempo de ejecución de scripts.

Esto significa que, para llevar a cabo la herramienta, se tendrá que diseñar un sistema de control de animaciones que permita activar y cambiar las animaciones de los personajes en respuesta a los eventos y condiciones de la historia.

2.6. Tecnologías de Almacenamiento y Manipulación de Datos para Videojuegos

La capacidad de almacenar y manipular datos es esencial para el desarrollo de juegos (Smith, 2017). Al crear herramientas de cinemáticas para Unity, se debe prestar mucha atención a las distintas técnicas existentes, para poder así aplicar estas de forma efectiva y lograr un resultado óptimo.

2.6.1. Almacenamiento de Datos

El almacenamiento de datos en el desarrollo de juegos se puede lograr utilizando una variedad de métodos, cada uno con sus propias ventajas y desventajas. La elección de la tecnología de almacenamiento adecuada a menudo depende el tipo y la cantidad de datos que se van a procesar, así como de las necesidades específicas del juego.

Bases de Datos Relacionales y No Relacionales

Las bases de datos son una opción común para el almacenamiento de datos en los videojuegos, especialmente cuando se trata de grandes cantidades de datos o se necesita acceder a la información de manera rápida y eficiente (Ziebarth, 2022). Una base de datos puede ser relacional, como MySQL [13] o PostgreSQL [14], o no relacional, como MongoDB [15] o Firebase [16]. Las bases de datos relacionales son adecuadas para trabajar con datos estructurados (Khan, 2017), mientras que las no relacionales son más flexibles y se adaptan mejor a los datos no estructurados (Kaur, 2013).

13 https://www.mysql.com/

14 https://www.postgresql.org/

15 https://www.mongodb.com/es

16 https://firebase.google.com/?hl=es

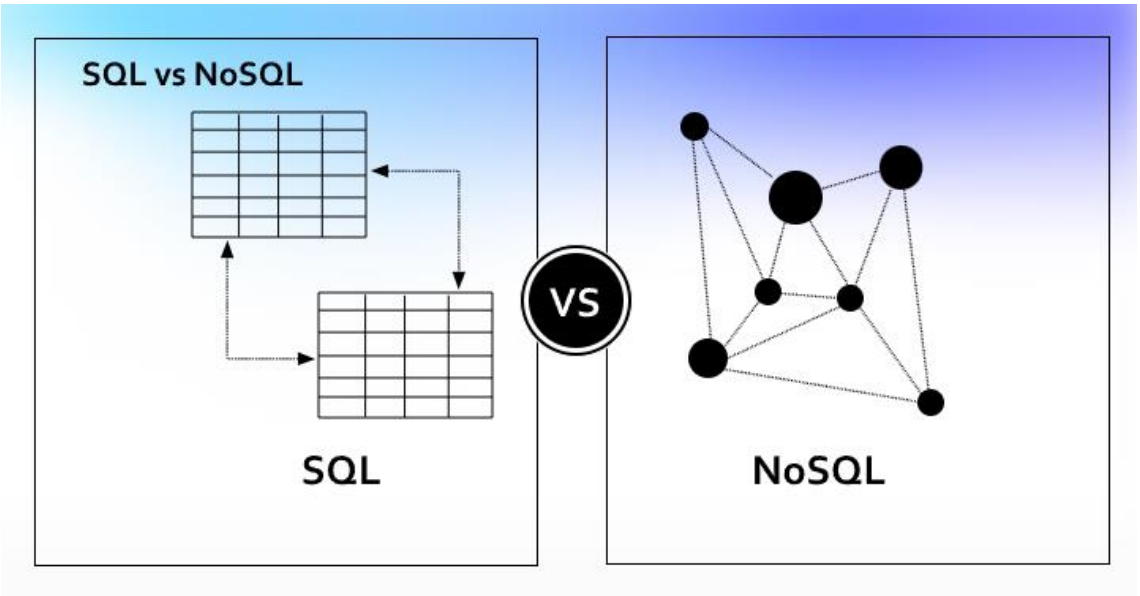


Figura 4: SQL vs NoSQL

FUENTE: EXTERNA [17]

Almacenamiento en la Nube

El almacenamiento en la nube es otra opción viable, especialmente para los juegos que requieren la sincronización de datos en múltiples dispositivos o plataformas. Servicios como Google Cloud Storage [18], Amazon S3 [19] y Azure Blob Storage [20] proporcionan soluciones escalables y seguras para almacenar datos en la nube.

2.6.2. Manipulación de Datos

Además del almacenamiento, la manipulación de datos es un aspecto importante del desarrollo de videojuegos (Ronzio, 2022). Los juegos a menudo requieren que los datos se modifiquen, filtren y procesen dinámicamente y en tiempo real.

Lenguajes de Programación y Scripts

Los lenguajes de programación y los scripts son las principales herramientas para la manipulación de datos en los videojuegos (Motiso, 2022). C# se usa comúnmente para secuencias de comandos en Unity y proporciona amplias capacidades de manipulación de datos, como *arrays*, listas, diccionarios y otras estructuras de datos.

APIs y Bibliotecas

Además, muchas API y bibliotecas proporcionan funciones y métodos adicionales para procesar datos (¿Qué Es Una API Y Cómo Funciona?, 2023). Por ejemplo, Unity tiene su propia API de manipulación de datos y puede usar bibliotecas externas como JSON.NET para serializar y deserializar datos en formato JSON (Benny, 2022).

17 Fuente de Figura 4: (Kasam, 2022)

18 https://cloud.google.com/storage?hl=es

19 https://aws.amazon.com/es/s3/

20 https://azure.microsoft.com/es-es/products/storage/blobs

Tecnologías de Interfaz de Usuario

La interfaz de usuario (UI) también juega un papel de la manipulación de datos. Los elementos de la UI, como formularios y campos de entrada, se pueden utilizar para recopilar y validar datos de usuario. En Unity, esto se logra a través del marco de la interfaz de usuario, que le permite crear interfaces de usuario interactivas.

2.7. Tendencias Actuales en Herramientas de Desarrollo para Videojuegos

Las tendencias en el ámbito del desarrollo de videojuegos reflejan un campo que está en constante evolución, en el que se abordan desafíos técnicos y creativos cada vez mayores (Howarth, 2023). En el caso específico de las herramientas de gestión de cinemáticas, existen varias tendencias emergentes que están remodelando el campo.

2.7.1. Integración de Software de Diseño de Cinemáticas

Una de las tendencias más influyentes es la incorporación de herramientas de diseño cinemáticas en el motor de juegos directamente. Este enfoque resuelve varias ineficiencias que antes prevalecían en el proceso de desarrollo. En el pasado, las cinemáticas se creaban como entidades separadas, utilizando software de animación y edición de video especializado. Luego, estos proyectos se importan al motor de juegos y se programan para ejecutarse en momentos específicos (Söderhäll, 2022).

La desventaja de este enfoque es que cualquier cambio de animación requiere personalizar el software de animación y luego volver a importarlo al motor del juego. Esto a menudo conduce a un tedioso proceso de prueba y error para obtener el resultado deseado (Nelson, 2019). Al integrar el software de diseño de cinemáticas directamente en el motor de juegos, los desarrolladores pueden diseñar, probar y ajustar cinemáticas en tiempo real dentro del videojuego (Thorne, 2022). Unity, por ejemplo, integra esta funcionalidad de Líneas de Tiempo y *Playable Graphs*, proporcionando a los desarrolladores una interfaz para el diseño de cinemáticas.

2.7.2. Mayor Interactividad en las Cinemáticas

En la intersección de la narración y el diseño de juegos, la interactividad de la trama está ganando popularidad. Tradicionalmente, las cinemáticas eran secuencias de video estáticas, insertadas en puntos específicos en el juego para avanzar en la narrativa o proporcionar contexto a los jugadores. Sin embargo, con el tiempo, este enfoque ha evolucionado hacia historias más interactivas, donde las acciones y decisiones del jugador pueden afectar el resultado de la historia e incluso la trama general del juego. Esta interactividad aumenta la inmersión y proporciona a los jugadores más libertad de acción en el mundo del juego (Nonlinear Storytelling, 2009).

2.7.3. Uso de la Inteligencia Artificial para Crear Cinemáticas

La inteligencia artificial (IA) se está convirtiendo en una poderosa herramienta en el diseño y la producción de cinemáticas. Aunque la IA aún se encuentra en sus primeras etapas, tiene el potencial de cambiar la forma en que se crean las cinemáticas. Una aplicación prometedora de la IA en este campo es la generación automática de animaciones en base a determinados parámetros o reglas. Este enfoque es especialmente útil para juegos con mundos grandes y sistemas complejos, donde la creación manual sería una tarea abrumadora (Liao, 2023).

2.7.4. Herramientas Específicas para Narrativa

Finalmente, existe una creciente adopción de herramientas específicamente diseñadas para la creación de narrativas en los videojuegos. Los videojuegos como medio tienen una capacidad

única para contar historias de formas que ningún otro medio puede hacerlo. Herramientas como Articy Draft reflejan el reconocimiento de esta realidad, proporcionando a los desarrolladores una plataforma para diseñar y administrar la narrativa de juegos de manera detallada y estructurada. Con una interfaz intuitiva y una amplia funcionalidad, esta, junto a otras herramientas similares pueden crear narrativas complejas y no lineales esenciales para la inmersión del jugador.

2.8. Casos de Uso de Herramientas de Cinemáticas y Diálogos

2.8.1. "The Witcher 3: Wild Hunt"

"The Witcher 3: Wild Hunt" es un ejemplo de la utilización efectiva de las cinemáticas y los diálogos. Las cinemáticas del juego, creadas con una gran atención al detalle y la calidad cinematográfica, han sido elogiadas por su narrativa y caracterización (Vulkk, 2017). Las herramientas de cinemáticas y diálogos permitieron a los desarrolladores de CD Projekt Red crear personajes complejos y una narrativa envolvente que se desarrolla a lo largo de las numerosas misiones y tareas repartidas a lo largo del juego.

Figura 5: The Witcher 3: Wild Hunt

FUENTE: EXTERNA [21]

El juego fue desarrollado con el motor de juego REDengine 3, una herramienta propietaria de CD Projekt Red. Este es conocido por su capacidad para manejar mundos abiertos vastos y detallados, y tiene una poderosa funcionalidad de cinemáticas que permitió a los desarrolladores crear inmersivas cinemáticas y los diálogos interactivos por los que el juego es tan conocido (Krzyścin, 2019).

En particular, las herramientas de cinemáticas de REDengine 3 permitieron a los desarrolladores controlar con precisión las expresiones faciales y los movimientos de los personajes, lo que resultó en personajes visualmente ricos y emocionalmente resonantes. Las cinemáticas también se integraron perfectamente con las escenas de diálogo interactivo, lo que permitió a los jugadores tomar decisiones que afectaban la trama dentro de las propias cinemáticas.

21 Fuente de Figura 5: (Paur, 2019)

2.8.2. "Red Dead Redemption 2"

"Red Dead redemption 2" de Rockstar Games, hace uso del motor de juego RAGE (Rockstar Advanced Game Engine). Este motor es una herramienta propietaria de Rockstar, diseñada para manejar el alto nivel de detalle y la escala de los mundos abiertos por los que la empresa es famosa (Ampoloquio, 2023).

Figura 6: Red Dead Redemption 2

FUENTE: EXTERNA [22]

Las sofisticadas herramientas de cinemáticas de RAGE permitieron a los desarrolladores de Rockstar crear secuencias de eventos visuales impresionantes y emocionantes impactantes (GameForest, 2019). Estas cinemáticas junto con una dirección de arte detallada y una iluminación precisa ayudaron a establecer el tono y el contexto de la trama ambientada en el lejano Oeste.

Además, el sistema de diálogos interactivo del videojuego se beneficia del uso del motor RAGE. Este sistema, que permite a los jugadores elegir respuestas durante las conversaciones, logra una sensación de inmersión y de control en la historia, añadiendo profundidad y complejidad a las relaciones entre los personajes. Todo esto se realiza mediante la sincronización precisa de las animaciones de los personajes con los diálogos y los eventos del juego, lo que resulta en interacciones creíbles y convincentes entre los personajes (Skrebels, 2018).

2.8.3. "The Last of Us"

"The Last of Us" de Naughty Dog es otro ejemplo impresionante del uso de cinemáticas y diálogos para ofrecer una narrativa profunda y conmovedora. Las cinemáticas de juego están cuidadosamente diseñadas para transmitir la historia de los personajes principales y explorar temas complejos y emocionales. Todo esto, con la ayuda de su motor de juegos propio, el Naughty Dog Engine, el cual ha sido perfeccionado a lo largo de varias generaciones de juegos para proporcionar un nivel impresionante de detalle y realismo (Dyer, 2013).

22 Fuente de Figura 6: (Suderman, 2018)

Figura 7: The last of us

FUENTE: EXTERNA [23]

Además de la función de las cinemáticas, el juego emplea diálogos para desarrollar la relación entre los personajes y proporcionar a los jugadores una comprensión más profunda de sus luchas y motivaciones. Las herramientas de cinemáticas y diálogos permitieron a los desarrolladores crear un juego que ha sido aclamado por su narrativa fuerte y emotiva (Nicolaou, 2019).

23 Fuente de la figura 7: (Shaver, 2020)

3. Estudio Económico

El estudio económico es una parte esencial de cualquier proyecto, y esto no es diferente en el desarrollo de videojuegos. Aquí se explorará el mercado actual para las herramientas de cinemáticas y diálogos, identificando las oportunidades existentes y las posibles vías de monetización.

3.1. Descripción del Mercado Actual

La industria de los videojuegos ha experimentado un crecimiento exponencial en las últimas décadas y sigue en aumento. Según el informe de 2023 de Newzoo (Newzoo, 2023), la industria de los videojuegos generó más de 180.000 millones de dólares en ingresos en el año 2022, superando industrias como el cine y la música. Se espera que esta tendencia de crecimiento continúe en los próximos años.

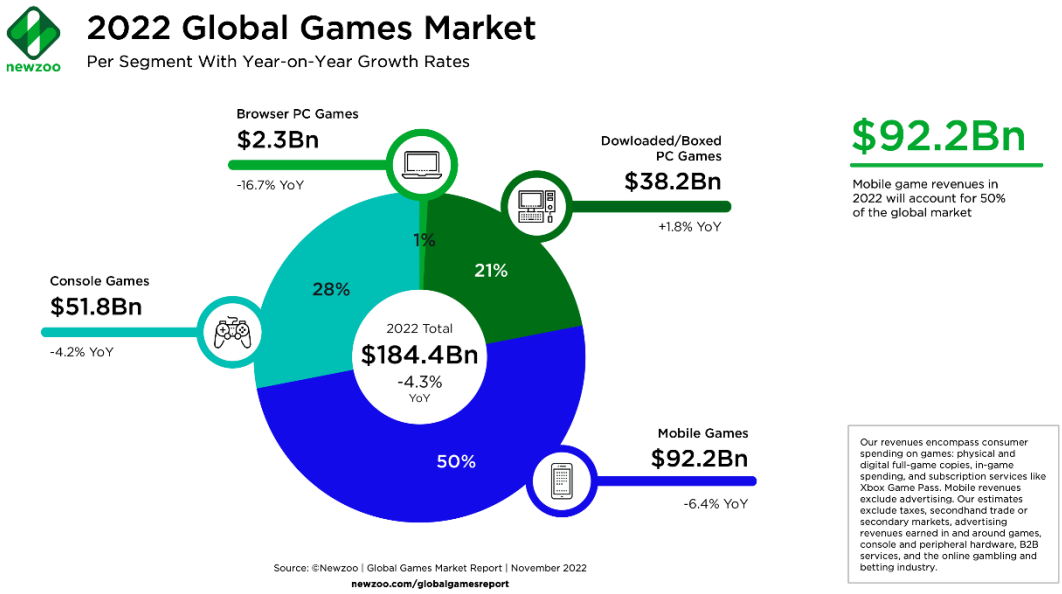


Figura 8: Gráfico del mercado de Videojuegos en 2022

FUENTE: EXTERNA [24]

El mercado de las herramientas de desarrollo de juegos es un segmento que está experimentando un crecimiento significativo donde en el año 2030 la cantidad estimada que los videojuegos generarán será de aproximadamente 583.000 millones de dólares (Turner, 2023). Los desarrolladores buscan constantemente soluciones para crear experiencias más envolventes e inmersivas, y las herramientas de cinemáticas y diálogos juegan un papel clave.

Herramientas como Unity Timeline, Playmaker y Articy Draft son cada vez más populares, lo que simplifica el proceso de creación de escenas y diálogos. Estas herramientas permiten a los desarrolladores centrarse en la creatividad en lugar de los aspectos técnicos. A medida que los videojuegos se vuelven más narrativos y basados en historias, es probable que crezca la necesidad de estas herramientas (Soto, 2023).

24 Fuente de Figura 8: (Newzoo, 2023)

3.2. Perspectivas de Crecimiento del Mercado y Posicionamiento en el Mercado

Adicionalmente, se ha identificado un incremento en la demanda de soluciones personalizadas y orientadas a géneros específicos. Por ejemplo, los desarrolladores que se dedican a la creación de videojuegos de rol podrían requerir herramientas de diálogo más sofisticadas para gestionar sistemas complejos de toma de decisiones. En contraste, aquellos que desarrollan videojuegos de aventura podrían centrar su atención en herramientas que faciliten la construcción de historias, con el objetivo de garantizar una transición suave y fluida entre la jugabilidad y las secuencias narrativas (Sliding Scale of Gameplay and Story Integration, n.d.).

A pesar del crecimiento y las exigencias de este mercado, todavía hay espacio para la innovación y la mejora. Las herramientas existentes a menudo requieren una larga curva de aprendizaje y pueden ser costosas, especialmente para desarrolladores independientes o estudios pequeños (Best Game Development Software in 2023, 2023). Además, la integración de diferentes herramientas suele ser difícil, donde las nuevas herramientas de gestión de cinemáticas y diálogos pueden encontrar su lugar y tener éxito en el mercado.

Como se ha mencionado anteriormente, se espera que este mercado experimente un crecimiento anual significativo en los próximos años. Las principales fuerzas impulsoras detrás de este crecimiento incluyen el auge de la realidad virtual y aumentada, y una creciente comunidad de desarrolladores independientes que buscan herramientas accesibles y potentes (Gaming Trends 2023 - Top 10 Trends That Will Rule Industry, 2021).

Posicionarse en este mercado dinámico requiere una comprensión profunda de las necesidades cambiantes y las tendencias emergentes. La innovación será clave, y cualquier herramienta nueva debe mostrar cómo mejora el proceso de creación de cinemáticas y diálogos, ahorrando tiempo, reduciendo la complejidad o ampliando las posibilidades creativas.

El mercado también es cada vez más consciente de la importancia de la accesibilidad y la inclusión, lo que significa que las herramientas que son fáciles de usar para una amplia gama de creadores, incluso aquellos sin experiencia en programación, pueden obtener una ventaja competitiva.

En términos de posicionamiento en el mercado, es fundamental identificar y comunicar un valor único y diferenciado. Esto puede incluir la especialización en ciertos géneros o estilos, la integración con ciertas plataformas, estructuras de precios o combinaciones de funciones que no se encuentran en las herramientas existentes.

3.3. Demanda y Oferta en el mercado

3.3.1. Demanda

La demanda del mercado de herramientas de gestión de cinemáticas y diálogos es impulsada principalmente por el crecimiento sostenido de la industria de los videojuegos. A medida que aumenta la producción de juegos que requieren narrativas complejas, tramas y diálogos interactivos, también aumenta la necesidad de herramientas eficientes y profesionales. Los desarrolladores de videojuegos están buscando soluciones que les permitan trabajar de manera más rápida y eficiente, al mismo tiempo que proporcionan control creativo y flexibilidad (Looper, 2023).

Además, la democratización de la tecnología y el auge de los desarrolladores independientes ha creado una gran demanda de herramientas asequibles y fáciles de usar. La comunidad *indie* está

buscando soluciones que puedan manejar proyectos de todos los tamaños sin comprometer la calidad.

La demanda también se extiende a las instituciones educativas que ofrecen cursos de diseño y desarrollo de videojuegos, ya que buscan herramientas que permitan a los estudiantes experimentar con la creación de cinemáticas en un entorno práctico.

Es importante también considerar el Gartnet Hype Cycle (2022 Gartner® Hype CycleTM for Digital Identity, 2022), que es una representación gráfica de la adopción y madurez de tecnologías específicas.

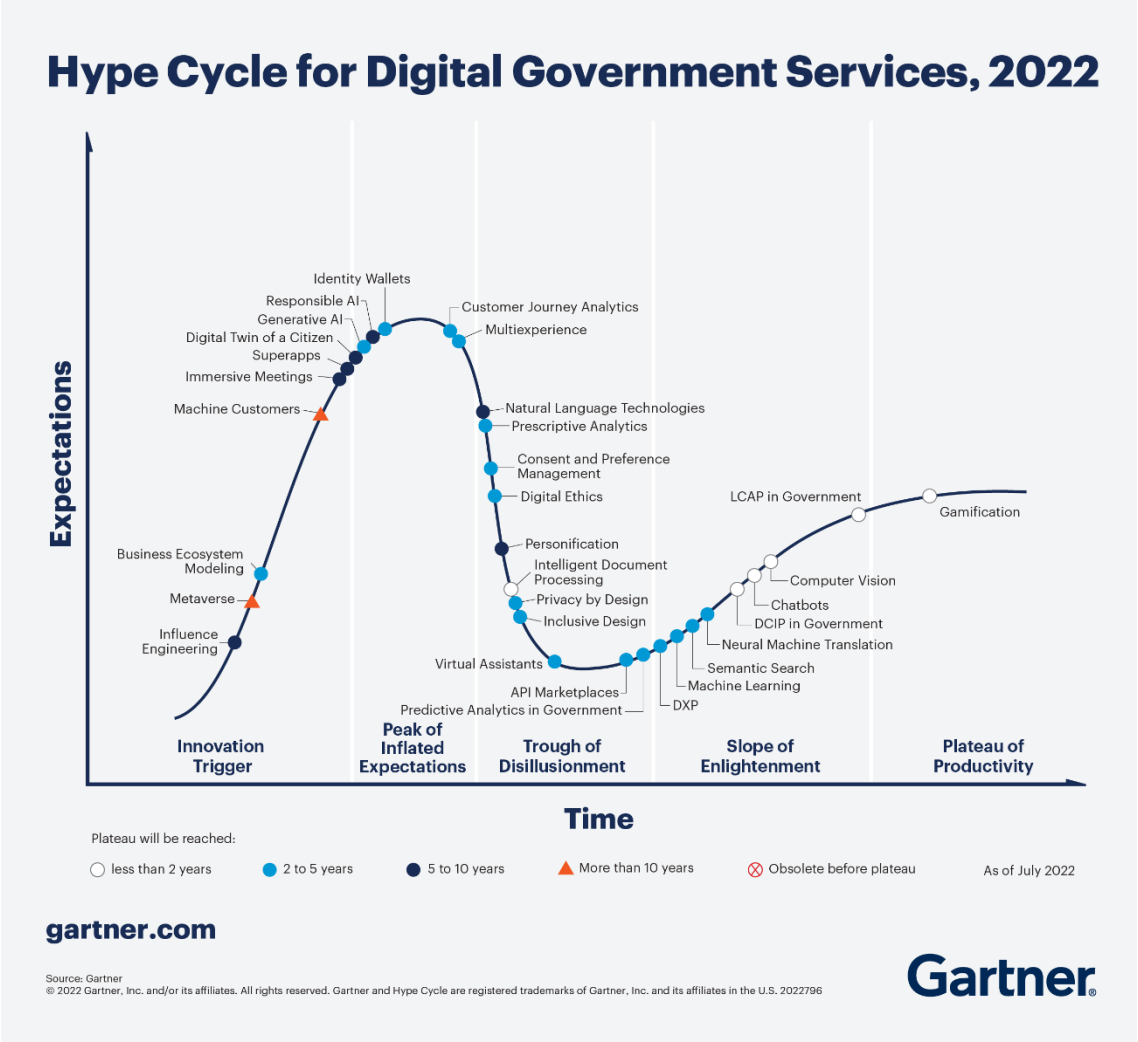


Figura 9: Gartnet Hype Cycle

ELABORACIÓN: EXTERNA [25]

El Gartner Hype Cycle se compone de cinco fases:

25 Fuente de figura 9: (What's New in Digital Government From the 2022 Gartner Hype Cycle, 2022)

1. **Innovación y Disparo de Expectativas:** Cuando una nueva tecnología emerge, genera altas expectativas y entusiasmo, pero aún no está probada o ampliamente adoptada.
2. **Pico de Expectativas Infladas:** El interés y las expectativas alcanzan su punto máximo, pero a menudo se basan en casos de éxito aislados y no en adopción generalizada. Aquí es donde algunos proyectos fallan por no llegar a cumplir con las expectativas.
3. **Abismo de Desilusión:** La tecnología no cumple con las expectativas iniciales, y el interés comienza a disminuir.
4. **Pendiente de la Iluminación:** Las empresas y desarrolladores comienzan a entender mejor cómo aplicar la tecnología de manera efectiva. Las mejoras y avances comienzan a mostrar su verdadero potencial.
5. **Meseta de Productividad:** La tecnología se vuelve más ampliamente adoptada y su valor se extiende y acepta de manera más clara.

Desarrollo de Videojuegos

Actualmente el desarrollo de videojuegos se encuentra en la Meseta de Productividad del Gartner Hype Cycle. Esta fase indica que el desarrollo de videojuegos es una industria madura, con procesos y tecnologías bien establecidas. La amplia aceptación cultural de los videojuegos, combinada con una base de usuarios global y un mercado en constante crecimiento reafirma su posición. (What's New in Digital Government From the 2022 Gartner Hype Cycle, 2022)

Creación de Cinemáticas en Videojuegos

La creación de cinemáticas, por otro lado, está avanzando desde la Pendiente de la Iluminación hacia la Meseta de Productividad. Esto se debe a que las cinemáticas se han vuelto fundamentales para la narrativa en muchos juegos, y las herramientas y técnicas para crearlas han evolucionado enormemente (Wibbu, 2018). Las transiciones fluidas entre el *gameplay* y cinemáticas, junto a la creciente calidad cinematográfica, evidencian cómo se han integrado eficazmente en la experiencia de juego. La razón principal de su posición en la Pendiente es el constante desarrollo y adopción de tecnologías como la captura de movimiento y técnicas de renderizado en tiempo real, que están llevando las cinemáticas a niveles previamente reservados para la producción de películas.

3.3.2. Oferta

Por el lado de la oferta, hay varias empresas y desarrolladores que ofrecen diversas herramientas en esta área proporcionado por los competidores directos, los cuales están siendo analizados próximamente en este documento en la sección [3.4.1 Competidores Directos]. Desde gigantes de la industria que ofrecen *suites* de desarrollo completas hasta empresas más pequeñas que se especializan en soluciones de terceros.

La gama incluye herramientas tanto de código abierto como propietarias, variando en términos de complejidad, características y precio. Algunos se enfocan en una experiencia rápida y fácil para principiantes, mientras que otros ofrecen funciones avanzadas para profesionales experimentados.

Aunque existen múltiples opciones, elegir la adecuada puede representar un desafío. La vasta gama de herramientas disponibles a veces puede resultar abrumadora para los desarrolladores, complicando la selección de la que mejor se ajuste a sus necesidades. Además, el rápido avance tecnológico y las cambiantes demandas de los usuarios implican que estas herramientas necesiten actualizaciones y adaptaciones frecuentes.

3.4. Análisis Competitivo

El análisis competitivo en el mercado de las herramientas de gestión de cinemáticas requiere una comprensión profunda de los competidores tanto directos como indirectos. Este conocimiento es crucial para identificar oportunidades, amenazas y desarrollar estrategias que permitan una posición competitiva en el mercado.

3.4.1. Competidores Directos

Los competidores directos en el mercado representan un grupo que ofrece productos o servicios similares y que atienden a las mismas necesidades y segmento de clientes (Dinu, 2018). Por ejemplo, Unity ofrece Unity Timelines, una poderosa herramienta para crear cinemáticas y secuencias, mientras que Articy Draft admite la gestión detallada de diálogos y el diseño narrativo con su producto. Epic Games [26] proporciona funcionalidades de cinemáticas mediante Sequencer en su Unreal Engine, y Playmaker ofrece una solución visual para la creación de lógica de un videojuego, incluyendo las cinemáticas. La competencia directa es feroz, con cada competidor tratando de superar al otro en términos de funcionalidad, facilidad de uso, innovación y precio. La elección de tecnologías y características puede determinar el éxito en este campo competitivo.

3.4.2. Competidores Indirectos

En cuanto a los competidores indirectos, estos no ofrecen exactamente las mismas soluciones, pero sus productos o servicios pueden satisfacer las mismas necesidades o resolver problemas similares en la industria de los videojuegos (Indeed Editorial Team, 2023). Esto incluye software de edición de video como Adobe After Effects [27], que puede ser usado para crear cinemáticas, aunque no estén específicamente diseñadas para ser aplicadas en videojuegos. También hay herramientas de diseño gráfico como Blender que pueden ser utilizadas para diseñar animaciones y cinemáticas. Además, las herramientas de escritura de guion como Final Draft [28], aunque enfocadas en la escritura de guiones de cine y televisión, pueden ser adaptadas para la creación de diálogos en videojuegos. Estos competidores indirectos representan alternativas a las soluciones especializadas en la gestión de cinemáticas y diálogos y pueden influir en las decisiones de compra de los clientes potenciales.

Este proyecto tiene como objetivo desarrollar una herramienta de cinemáticas que se adapte a los requisitos específicos expuestos anteriormente, en lugar de competir en el mercado actual. La ventaja competitiva de este proyecto no está en el entorno o mercado competitivo, sino en su flexibilidad y capacidad para abordar las necesidades y desafíos específicos expuestos en este documento.

Analizar la competencia es crucial para identificar tendencias en el uso de esta tecnología y tener una idea de las características y funciones que más se valoran. Esto proporciona una comprensión más completa de los factores que deben tenerse en cuenta al diseñar el sistema de control de acceso.

26 https://www.epicgames.com/site/es-ES/home

27 https://www.adobe.com/es/products/aftereffects.html

28 https://www.finaldraft.com/

3.5. Oportunidades y Desafíos en el Mercado

3.5.1. Oportunidades

El mercado de herramientas de gestión de cinemáticas en videojuegos está en constante expansión, lo que proporciona algunas oportunidades únicas. La creciente complejidad de los videojuegos y la necesidad de historias más ricas y atractivas proporciona un terreno fértil para el desarrollo de nuevas herramientas y técnicas. La integración con plataformas populares como Unity y Unreal Engine abre la puerta a una amplia comunidad de desarrolladores, los cuales ofrecen oportunidades para innovar y expandirse a nuevos mercados (Archaeogaming, 2017).

3.5.2. Desafíos

Sin embargo, hay algunas cuestiones importantes a considerar. La competencia con grandes actores establecidos y la rápida evolución de la tecnología pueden hacer que sea difícil mantenerse al día con las últimas tendencias y estándares. La necesidad de cumplir con las expectativas de los consumidores en cuanto a calidad, rendimiento y facilidad de uso crea desafíos constantes para el desarrollo y la mejora de productos similares. Además, la regulación y las consideraciones éticas en torno al manejo de datos y la propiedad intelectual pueden presentar obstáculos legales y operativos, los cuales están explicados en el apartado [Regulaciones y Consideraciones Éticas]

3.5.3. Enfoque Considerando Oportunidades y Desafíos

Navegar por este panorama complejo y aprovechar las oportunidades y abordar los desafíos requiere un enfoque estratégico y flexible. Esto incluye invertir en investigación y desarrollo para mantenerse a la vanguardia de la tecnología y cumplir con las expectativas de los consumidores. También es importante formar alianzas y colaboraciones con otras empresas y plataformas para expandir el alcance del proyecto y fortalecer su posición en el mercado. El cumplimiento constate y la adhesión a las mejores prácticas éticas garantizarán que se opere dentro de la ley y mantenga una buena reputación (Adiguzel, 2020).

3.6. Nivel de Madurez Tecnológica (TRL)

3.6.1. Definición de TRL

Para poder identificar el estado de desarrollo que se ha realizado durante el TFG, he optado por el Nivel de Madurez Tecnológica (Technology Readiness Level, TRL por sus siglas en inglés). TRL se entiende como un sistema de medida que se utiliza para evaluar la madurez de una tecnología específica. La NASA fue la primera en crear el TRL, que desde entonces se ha utilizado como estándar para evaluar el progreso de la tecnología desde la investigación básica hasta la implementación comercial (A Model Based on the Technology Readiness Level (TRL) Scale to Measure the Maturity Level of Research Projects That Can Become Spinoffs in Higher Education Institutions, 2021).

Los TRLs se dividen en 9 niveles, donde el nivel 1 se refiere a la investigación básica y el nivel 9 se aplica a la tecnología probada en condiciones reales y preparada para la implementación comercial. A continuación, se proporciona una breve descripción de cada nivel (Tzinis, 2021):

- **Investigación básica (TRL 1):**
  En esta etapa inicial, se estudian los fundamentos de la tecnología en consideración. Se trata de un estudio teórico destinado a determinar las propiedades y posibles aplicaciones. Esta investigación es de naturaleza exploratoria y tiene como objetivo avanzar en el conocimiento, sin una aplicación específica en mente.
- **Concepto de tecnología (TRL 2):**
  En este nivel, la atención se centra en la práctica de los principios descritos en TRL 1. Se llevan a cabo estudios analíticos y de simulación para explorar el potencial de la

tecnología. Todavía está en la etapa de concepto, pero ha quedado claro cómo se puede desarrollar y utilizar en aplicaciones reales.

- **Prueba experimental de concepto (TRL 3):**

  Aquí el concepto de tecnología se traslada al laboratorio. Se realizan experimentos preliminares para verificar que el concepto funciona como se espera en un entorno controlado. Esto generalmente incluye la creación de prototipos y pruebas a pequeña escala para demostrar la viabilidad del proyecto en cuestión.

- **Validación de la tecnología en laboratorio (TRL 4):**

  La tecnología se desarrolla y se somete a fondo en el laboratorio. Esto incluye la optimización de prototipos y la evaluación de su rendimiento en diversas condiciones controladas. Los resultados de estas pruebas se utilizan para refinar el diseño del prototipo.

- **Validación de la tecnología en entorno relevante (TRL 5):**

  Los prototipos se prueban en entornos que se parecen mucho a las aplicaciones de la vida real. Esto permite determinar cómo funciona la tecnología en situaciones más realistas y posiblemente imprevistas. Este paso es importante para refinar y mejorar la tecnología antes de pasar a las pruebas en condiciones más extremas.

- **Demostración de la tecnología en entorno relevante (TRL 6):**

  En este nivel, la tecnología está representada en un entorno que refleja fielmente su uso final. Esto proporciona una estimación más confiable de cómo funcionará la tecnología en el mundo real y permite identificar y abordar cualquier problema restante antes de pasar a las etapas finales de desarrollo.

- **Demostración de la tecnología en entorno operacional (TRL 7):**

  En el TRL 7, esta tecnología se prueba en el entorno operativo real en el que se espera que opere. Esto significa exponer la tecnología a todas las condiciones cambiantes e imprevistas del entorno real. Esta es una prueba importante que se acerca a un producto casi finalizado.

- **Sistema completo y cualificado (TRL 8):**

  En esta etapa, la tecnología se integra con tolos los demás componentes necesarios para formar un sistema completo. Se realizan una serie de pruebas de calidad para garantizar que el sistema cumple con todos los requisitos funcionales y de seguridad necesarios para su correcta implementación.

- **Sistema probado y aprobado para uso comercial (TRL 9):**

  En este último paso, la tecnología ha superado todas las etapas de desarrollo y pruebas, por lo que está lista para su implementación comercial. Esto significa que ha demostrado un rendimiento confiable y constante en condiciones reales, por lo que cumple con los estándares y regulaciones de la industria.

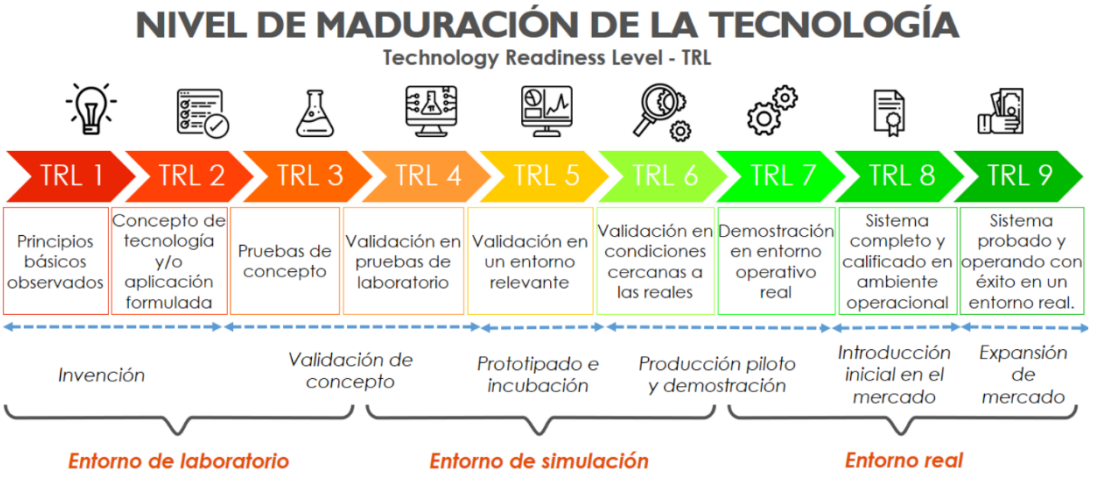


Figura 10: Nivel de Maduración de la Tecnología (TRL)

FUENTE: EXTERNA [29]

3.6.2. Nivel TRL actual y plan para alcanzar TRL 9

Por el momento, se estima que el nivel TRL 3-4 está asociado con el proyecto. En esta etapa, la herramienta ha sido desarrollada con éxito, y se ha probado la tecnología demostrando su funcionamiento, aunque, esto ha sido dentro de un entorno controlado de laboratorio. A este nivel, se puede afirmar que la herramienta funciona correctamente. Sin embargo, no se ha probado en un entorno relevante, por lo que aún queda trabajo por hacer antes de poder clasificar el proyecto a un nivel superior.

Para poder alcanzar el nivel TRL 9, se deben superar varios hitos. Inicialmente, la tecnología debe mostrarse en un entorno aplicable, lo que podría implicar pruebas en un videojuego real finalizado. El siguiente paso es exhibir la tecnología en una situación operativa tangible y luego vincularla con otros sistemas, lo que da como resultado un sistema que lo abarca todo. Por último, el sistema debería ser probado y aprobado para uso comercial.

Los obstáculos que deben superarse son sustanciales y requieren esfuerzos coordinados para desarrollar, probar, validar y cumplir con las reglamentaciones pertinentes. Sin embargo, el reconocimiento de estos problemas también permite un enfoque directo para mejorar la sofisticación del sistema.

3.7. Análisis Económico

Una comprensión clara del coste total asociado y el potencial retorno de la inversión es crucial para que los proyectos de desarrollo tecnológico tomen consideraciones económicas. El análisis económico incluirá costes de desarrollo, costes operativos y posibles ingresos en caso de comercializar el sistema.

3.7.1. Coste del Desarrollo

El desarrollo de la herramienta de gestión de cinemáticas implica diversos costes asociados a distintas etapas del proceso. Esto incluye los costes iniciales de investigación y diseño, la adquisición de hardware y software, el pago a desarrolladores, diseñadores y otros profesionales

29 Fuente de Figura 10: (Innova - Ciencia Asturias, 2022)

involucrados, así como los costes de pruebas y validación (Talent.com, n.d.). La inversión en tecnologías y licencias actualizadas como las de Articy Draft y Unity Pro, y el cumplimiento de los estándares de calidad también contribuyen a los gastos totales. En conjunto, el coste del desarrollo es una consideración clave y requiere una planificación cuidadosa y un presupuesto bien estructurado para garantizar que el proyecto se mantenga en el camino correcto.

Tabla 1: Coste del desarrollo

Elemento	**Coste (€)**	**Horas de Desarrollo**	**Perfil del Miembro del Equipo**	**Sueldo medio por hora (€)**
Hardware (Incluyendo ordenadores)	1800	-	-	-
Software (Licencias, herramientas de desarrollo)	1877 + 115 = 1992	-	-	-
Desarrollo				
Investigación y Diseño	-	50	Programador	10.77
	-	30	Diseñador UX/UI	15.38
Implementación del prototipo	-	80	Programador Junior	10.77
	-	40	Programador Senior	17.69
Pruebas y Validación	-	20	Ingeniero de pruebas	16.67
	-	15	Programador Junior	10.77
Documentación	-	100	Redactor técnico	12.31
Total	1000	335	-	-

El coste total aproximado del desarrollo de este proyecto sería calculado de la siguiente manera:

$$Coste\ total = 1992 + (50 \cdot 10.77) + (30 \cdot 15.38) + (80 \cdot 10.77) + (40 \cdot 17.69) + (20 \cdot 16.67) + (15 \cdot 10.77) + (100 \cdot 12.31) = 6287.05€$$

3.7.2. Costes Operativos

Los gastos de mantenimiento de *hardware* y el *software*, el suministro y la actualización o adaptación del sistema para satisfacer las necesidades cambiantes están incluidos en los costes operativos del sistema. Estos costes incluyen el mantenimiento de equipos, costes de mano de obra, etc. Estos costes son continuos y deben tenerse en cuenta en el análisis económico.

3.7.3. Potenciales Ingresos

Si bien este proyecto no se está desarrollando con la intención de comercialización, es importante considerar los ingresos potenciales que podrían resultar dicho desarrollo. El precio de venta de la herramienta está sujeto a varios factores como los costes de producción, la demanda del mercado y los precios de la competencia. Sin embargo, una estimación inicial podría basarse en el coste de desarrollo y operación del sistema, con un margen adicional para beneficios.

3.8. Regulaciones y Consideraciones Éticas

En el proceso de desarrollo e implementación de la herramienta, es esencial tener en cuenta una serie de regulaciones y consideraciones éticas.

3.8.1. Regulaciones

Derechos de Propiedad Intelectual

Todas las tecnologías, algoritmos y contenidos desarrollados deben cumplir con las leyes de derechos de propiedad intelectual aplicables. Esto incluye la obtención de las licencias adecuadas para cualquier *software* de terceros y el aseguramiento de que no se estén infringiendo patentes o derechos de autor existentes.

Privacidad y Protección de Datos

La herramienta debe cumplir con todas las regulaciones pertinentes relacionadas con la privacidad y protección de datos, especialmente si se recopila y almacena información del usuario. Esto puede incluir cumplir con el Reglamento de Protección de Datos (GDPR) en Europa y leyes similares en otros territorios (General Data Protection Regulation (GDPR) – Official Legal Text, 2022).

Clasificación y Contenido

La clasificación y contenido son esenciales dependiendo del tipo de videojuego y de la audiencia a la que esté destinado. Es vital estar al tanto de las regulaciones relacionadas con la clasificación y el contenido (Dogruel, 2013). Aunque la herramienta es simplemente una plataforma de desarrollo, es responsabilidad de quienes la usan asegurarse de que el contenido producido con ella se ajuste a las normativas y no viole las leyes locales ni sea inapropiado para determinadas edades.

3.8.2. Consideraciones Éticas

Transparencia y Consentimiento

Aunque muchas herramientas en el ámbito del desarrollo de videojuegos pueden requerir el acceso a información personal o sensible del usuario, es esencial señalar que la herramienta propuesta en este trabajo no solicita ni almacena datos personales de los usuarios. Sin embargo, es siempre una buena práctica, en cualquier desarrollo, asegurarse de proporcionar transparencia y obtener el consentimiento explícito en caso de que se maneje información delicada.

Accesibilidad

Si bien es fundamental diseñar la herramienta teniendo en cuenta la accesibilidad para asegurar su uso por una diversidad de usuarios, es razonable asumir que, al tratarse de una herramienta especializada en desarrollo, los usuarios tendrán cierto nivel de habilidad técnica. No obstante, esto no debe eximirnos de la responsabilidad de hacerla lo más intuitiva y accesible posible, garantizando que pueda ser aprovechada al máximo dentro de su público objetivo.

Impacto Social y Cultural

Aunque la herramienta en sí es neutral, es crucial que los desarrolladores sean conscientes del impacto social y cultural de las cinemáticas que crean utilizando esta plataforma. Es responsabilidad de quienes la usan asegurarse de que el contenido producido sea respetuoso con diversas culturas y no perpetúe estereotipos negativos o contenido ofensivo. La herramienta, al igual que cualquier otra herramienta, solo es tan respetuosa o irrespetuosa como las manos y las intenciones de quien la maneja.

Sostenibilidad

La sostenibilidad en el desarrollo y la operación de la herramienta también es una consideración ética importante. Esto puede incluir la eficiencia en el uso de recursos y la minimización del impacto ambiental.

4. Análisis y Selección de Herramientas

4.1. Introducción al Análisis de Herramientas

Crear un videojuego exitoso requiere de una cuidadosa selección de herramientas que satisfagan las necesidades técnicas y creativas del proyecto. A medida que la industria de los videojuegos ha crecido y madurado, también lo han hecho las herramientas disponibles para los desarrolladores. La variedad de opciones, aunque beneficiosa en muchos sentidos, puede resultar abrumadora a la hora de decidir qué herramientas son las mejores para un proyecto en particular.

El propósito de este capítulo es analizar, comparar y seleccionar las herramientas para su uso en proyectos centrados en el desarrollo de juegos, creación de diálogos, animación, cinemáticas y procesamiento de datos. La selección adecuada de estas herramientas es fundamental para el éxito del proyecto, ya que afectará la eficiencia del desarrollo, la calidad final y la satisfacción del equipo.

El análisis se basará en un conjunto de criterios predefinidos que reflejen las necesidades y objetivos del proyecto. Estos criterios incluyen funcionalidad, facilidad de uso, integración con otras herramientas, coste, soporte y comunidad de desarrolladores, entre otros. Además, se considerará el contexto actual del mercado y las tendencias en la industria del desarrollo de videojuegos.

A través de este análisis exhaustivo, presente proporcionar una base sólida para las decisiones de selección de herramientas y garantizar que se seleccione la herramienta más adecuada para lograr los objetivos y la visión del proyecto.

4.2. Herramientas de Desarrollo de Videojuegos

En la actualidad, el desarrollo de videojuegos es una tarea compleja que involucra múltiples aspectos, desde la programación y el diseño hasta la creación de gráficos y sonido. La selección de las herramientas adecuadas para cada uno de estos aspectos es vital para garantizar un proceso de desarrollo eficiente y exitoso.

4.2.1. Motores de Juegos

Los motores de juego son una parte fundamental en el desarrollo de videojuegos, ya que proporcionan la base sobre la que se construyen todos los elementos del juego. A continuación, se describen algunos de los motores más populares:

- **Unity:** Uno de los motores más populares y versátiles, ofrece una amplia gama de funcionalidades y es conocido por su facilidad de uso y su capacidad para exportar a múltiples plataformas. Este motor de juegos está analizado más en profundidad en el apartado [2.3.1 Unity].
- **Unreal Engine:** Conocido por su capacidad de crear gráficos de alta calidad, es una opción potente para desarrolladores que buscan control y una personalización profundos.
- **Godot [30]:** Un motor de código abierto que ha ganado popularidad por su enfoque flexible y comunidad activa.

4.2.2. Herramientas de Diseño y Modelado

Las herramientas de diseño y modelado son esenciales para crear los activos visuales que conforman el mundo del juego. Algunas de las herramientas destacadas incluyen:

30 https://godotengine.org/

- **Blender:** Una poderosa herramienta de modelado 3D de código abierto que ofrece una amplia gama de funcionalidades.
- **Maya:** Utilizado en muchas producciones profesionales, ofrece una gama completa de herramientas para modelado, animación y renderizado.
- **Adobe Photoshop [31]:** A menudo utilizado para el diseño de texturas y elementos 2D, convirtiéndose en una herramienta estándar en la industria.

4.2.3. Herramientas de Programación

La programación es el núcleo de cualquier videojuego, y la elección del lenguaje y las herramientas de programación es una decisión clave en el desarrollo.

- **Visual Studio:** Un entorno de desarrollo integrado (IDE) ampliamente utilizado que ofrece una amplia gama de funcionalidades y soporte para varios lenguajes de programación.
- **MonoDevelop:** A menudo utilizado junto con Unity, es un IDE de código abierto que soporta varios lenguajes.
- **Sublime Text:** Un editor de texto popular por su velocidad y flexibilidad, utilizado por muchos desarrolladores.

4.3. Herramientas de creación de Diálogos

La narración es un componente esencial en muchos videojuegos, y la creación de diálogos efectivos juega un papel clave en la construcción de una historia inmersiva y atractiva. A continuación, se presentan algunas de las herramientas más utilizadas en la industria para la creación y gestión de diálogos.

4.3.1. Articy draft

Articy draft es una solución integral para la escritura y diseño de contenido interactivo. Ofrece una variedad de características, incluyendo la capacidad de crear estructuras de diálogo complejas, definir variables y condiciones, y exportar directamente a varios motores de juegos. Su enfoque visual y flexible lo convierte en una opción popular entre los equipos de desarrolladores de todos los tamaños. Esta herramienta está explicada más en profundidad en el apartado [2.3.2 Articy Draft].

4.3.2. Twine

Twine [32] es una herramienta de código abierto que permite a los escritores diseñar historias interactivas y no lineales. Su interfaz gráfica facilita la visualización y la construcción de rutas complejas en la narración. Es especialmente útil en los juegos centrados en la narrativa, donde los jugadores tienen varias opciones y caminos para explorar.

4.3.3. Ink by Inkle

Ink by Inkle [33] es un lenguaje de escritura y una herramienta para escribir diálogos interactivos. Es ligero y fácil de aprender, y es especialmente útil para escritores que quieran centrarse en la narrativa sin tener que preocuparse demasiado por la programación. Ha sido utilizado en varios juegos aclamados por la crítica, como "*Heaven's Vault*" y "*80 Days*".

31 https://www.adobe.com/es/products/photoshop.html

32 https://twinery.org/

33 https://www.inklestudios.com/ink/

4.3.4. ChatMapper

Es una herramienta diseñada específicamente para la creación y prueba de diálogos complejos en juegos. ChatMapper [34] permite a los escritores y diseñadores visualizar y editar estructuras de diálogo, asignar emociones y animaciones, y simular conversaciones para probar cómo fluyen.

4.3.5. Yarn Spinner

Yarn Spinner [35] es una herramienta de código abierto que facilita la escritura de diálogos en una forma que es fácil de leer y escribir. Es una opción flexible que se puede utilizar con una variedad de motores y ofrece una sintaxis simple y clara.

4.4. Herramientas de Animación y Cinemáticas

Las animaciones y cinemáticas son elementos cruciales en la producción de videojuegos modernos, ya que dan vida a los personajes, la narrativa, y el mundo del juego. A continuación, se describen algunas de las principales herramientas empleadas en la creación de animaciones y cinemáticas para videojuegos.

4.4.1. Maya

Autodesk Maya es una de las herramientas de animación 3D más poderosas y versátiles en la industria. Utilizada tanto en la animación cinematográfica como en la de videojuegos, ofrece un amplio conjunto de características para modelar, esculpir, texturizar y animar personajes y escenarios.

4.4.2. Unity Timeline

Unity Timeline es una herramienta dentro del motor Unity que permite a los desarrolladores diseñar y controlar secuencias de animación y cinemáticas. Ofrece una interfaz visual intuitiva, lo que facilita la coordinación de múltiples pistas y elementos.

4.4.3. Blender

Es una solución de código abierto para la creación de contenido 3D, incluyendo animaciones. Es muy apreciada por su flexibilidad y capacidad de adaptación, y su naturaleza de código abierto significa que cuenta con una comunidad activa que contribuye con complementos y mejoras.

4.4.4. Adobe After Effects

Es una herramienta empleada para la postproducción de animaciones y efectos visuales. Permite la composición y edición de elementos visuales en un flujo de trabajo no lineal, y es la comúnmente utilizada para agregar efectos y refinar animaciones en videojuegos.

4.4.5. Anima2D

Es un complemento de animación 2D para Unity, utilizado para crear animaciones de personajes y escenas en 2D. Anima2D [36] ofrece herramientas como *rigging* (articulación), deformación y control de huesos, lo que permite una animación fluida y natural.

34 https://www.chatmapper.com/

35 https://yarnspinner.dev/

36 https://unity.com/es/features/2danimation

4.5. Herramientas de Gestión de Datos

4.5.1. Bases de datos SQL

Las bases de datos SQL, como MySQL [37] y PostgreSQL [38], son ampliamente utilizadas para almacenar información estructurada. Permiten consultas eficientes y ofrecen una gran cantidad de características para garantizar la integridad y seguridad de los datos.

4.5.2. Bases de datos NoSQL

Las bases de datos NoSQL, como MongoDB [39] y Redis [40], proporcionan soluciones de almacenamiento más flexibles y escalables. Son especialmente útiles para manejar grandes cantidades de datos y estructuras que no se ajustan fácilmente a los sistemas tradicionales.

4.5.3. Unity PlayerPrefs

Unity ofrece una clase de utilidad que permite a los desarrolladores almacenar y recuperar datos de manera sencilla. Aunque no es adecuada para grandes volúmenes de datos, es útil guardar preferencias y progresos del jugador.

4.5.4. JSON y XML

Estos son formatos de intercambio de datos comúnmente utilizados en el desarrollo de videojuegos. Permiten una representación estructurada de los datos y son compatibles con una amplia variedad de lenguajes y plataformas.

4.5.5. Hojas de cálculo

Las hojas de cálculo, como Excel [41], son a menudo utilizadas en el diseño y la gestión de datos durante la fase de desarrollo. Pueden servir como una herramienta accesible para definir y manipular datos, como estadísticas de personajes y configuraciones de escenarios.

4.6. Integración de Herramientas

La integración de herramientas es un aspecto crítico en el desarrollo de videojuegos modernos. Se refiere a la coordinación y el ensamblaje de diversas herramientas y tecnologías para trabajar de manera cohesiva dentro de un único flujo de trabajo. A continuación, se detallan los elementos clave de la integración de herramientas en el contexto del desarrollo de videojuegos.

4.6.1. Interoperabilidad

Es la capacidad de diferentes herramientas y sistemas para trabajar juntos, incluso si provienen de diferentes proveedores. Esto significa que las herramientas deben ser capaces de compartir datos y funcionalidades de una manera coherente. Por ejemplo, una herramienta de modelado 3D debe ser capaz de exportar modelos en un formato que pueda ser importado y utilizado por el motor de juegos (Chapurlat, 2012).

37 https://www.mysql.com/

38 https://www.postgresql.org/

39 https://www.mongodb.com/es

40 https://redis.io/

41 https://www.microsoft.com/es-es/microsoft-365/excel

4.6.2. Compatibilidad

La compatibilidad se refiere a la capacidad de las herramientas para funcionar conjuntamente sin conflictos. Esto puede implicar consideraciones como la compatibilidad entre versiones, la compatibilidad con diferentes sistemas operativos, y la compatibilidad con diferentes plataformas de hardware (Ghosh, 2013).

4.6.3. Automatización y Flujos de Trabajo

La automatización en la integración de herramientas ayuda a simplificar y agilizar los procesos de desarrollo. Esto puede incluir la automatización de la compilación, las pruebas, la implementación y otras tareas repetitivas. La configuración de flujos de trabajo efectivos asegura que las distintas etapas del desarrollo estén bien coordinadas y se realicen de manera eficiente (Martinez, 2022).

4.6.4. Comunicación entre Herramientas

Una integración exitosa requiere una comunicación efectiva entre diferentes herramientas. Esto puede incluir transferencias de datos, llamadas a funciones y coordinaciones de tareas. Las API (Interfaz de Programación de Aplicaciones) desempeñan un papel vital en esta comunicación, permitiendo que diferentes herramientas interactúen entre sí (Johl & Johl, 2023).

4.6.5. Estándares y Protocolos

La adhesión a estándares y protocolos comunes es vital para asegurar una integración sin problemas. Esto puede incluir estándares de formato de archivo, protocolos de comunicación y convenciones de codificación (Pratt, 2023).

4.6.6. Evaluación y Selección

La evaluación y selección de herramientas deben ser llevabas a cabo con cuidado. Esto incluye la consideración de factores como la funcionalidad, el rendimiento, la fiabilidad, el coste y el soporte de la comunidad.

4.6.7. Documentación y Soporte

La documentación adecuada y el soporte continuo son fundamentales para una integración exitosa. Esto incluye la documentación técnica, las guías del usuario, y el acceso a soporte técnico cuando sea necesario.

4.7. Análisis Comparativo y Selección Final

Las herramientas mencionadas anteriormente, cada una con sus propias fortalezas y limitaciones, se analizarán con mayor profundidad, exponiendo las ventajas y desventajas de las más relevantes a la hora de tener que decidir cual utilizar al implementar la herramienta de gestión de cinemáticas para el motor de juegos Unity:

1. Unity Timelines
    a. Ventajas: Esa es una herramienta poderosa que proporciona un alto nivel de control sobre las cinemáticas y eventos del juego. Permite la edición de pistas y la sincronización con animaciones y sonido. Además, su integración nativa con Unity facilita la implementación directa en el motor de juego.
    b. Desventajas: Puede ser excesiva para desarrollos más pequeños y requiere un conocimiento técnico considerable. Además, es más una herramienta de programación visual que una herramienta de diseño narrativo.
2. Playmaker

   a. Ventajas: Es amigable para los no programadores y es adecuada para crear secuencias interactivas de juego, con un sistema visual basado en máquinas de estado que permite crear lógicas de juego sin escribir código.
   b. Desventajas: Aunque es poderoso, puede resultar limitado para secuencias de cinemáticas más complejas y no está específicamente diseñado para la narración de historias.
3. Articy Draft
   a. Ventajas: Es fuerte en la organización de contenido y en la creación de narrativas ramificadas. Ofrece una gran variedad de opciones para diseñar y estructurar el flujo del juego.
   b. Desventajas: Su enfoque está más en la narrativa que en las cinemáticas. La integración con Unity puede ser compleja y puede requerir trabajo adicional para asegurar la compatibilidad.

Después de una cuidadosa evaluación, se ha decidido utilizar una combinación de Unity Timelines y Articy Draft. Unity Timelines ofrece un control granular sobre las cinemáticas y eventos, mientras que Articy draft permitirá manejar eficientemente las complejidades de la narrativa ramificada.

Tabla 2: Comparativa de las herramientas

	Unity Timelines	**Playmaker**	**Articy Draft**
Enfoque	Cinemáticas y eventos	Interactividad del juego	Narrativa ramificada
Integración con Unity	10 (Integración Nativa)	9 (Muy alta integración)	6 (Integración compatible, pero requiere configuración)
Facilidad de uso	5 (Requiere conocimientos técnicos)	8 (Amigable para no programadores)	6 (Requiere tiempo para aprender a usar adecuadamente)
Flexibilidad	9 (Altamente configurable y adaptable)	6 (Buena para tareas específicas, pero limitada en otras)	9 (Adaptable a múltiples narrativas y estructuras)
Adecuado para	Desarrollos más grandes, secuencias de cinemáticas detalladas	Juegos interactivos, desarrollos más pequeños	Organización de contenido, estructuración de la narrativa

La tabla presenta una comparación detallada entre tres herramientas prominentes en el ámbito de las cinemáticas y diálogos en videojuegos: Unity Timelines, Playmaker y Articy Draft. A cada una de estas herramientas se le ha asignado un valor entre 1 y 10 para que fuera posible puntuarlos de forma eficiente en cada uno de los apartados presentes en la tabla.

5. Análisis, Diseño e Implementación de la Herramienta

5.1. Introducción

En este capítulo, nos enfocaremos en el diseño e implementación del sistema de gestión de cinemáticas, el cual será el encargado de asegurar la funcionalidad de la herramienta. Abordaremos el análisis del sistema, como la robustez, estética y usabilidad. El objetivo es demostrar una herramienta que sea capaz confiable en términos de eficiencia a la hora de crear nuevas cinemáticas en cualquier videojuego y al mismo tiempo accesible y fácil de usar, sobre todo, por los desarrolladores que vayan a utilizarla para futuros desarrollos.

5.2. Análisis del Sistema

En la fase inicial, es imprescindible comprender el contexto en el que el sistema operará y las necesidades que debe satisfacer. La herramienta tiene como objetivo principal mejorar el desarrollo de videojuegos facilitando la creación de nuevas cinemáticas.

5.2.1. Requisitos funcionales

En cuanto a los requisitos funcionales de la herramienta, estos se centran en las capacidades específicas y las operaciones que debe realizar. Es fundamental que permita a los desarrolladores establecer y configurar eventos específicos dentro del juego. Estos eventos abarcan desde disparadores y condiciones previas hasta las acciones resultantes que ocurren una vez que se cumple un evento. Además, la creación y edición de cinemáticas es otro pilar central de la herramienta. Debe proporcionar una interfaz dedicada para esto, proporcionando control total sobre elementos como animaciones y diálogos de las escenas.

5.2.2. Requisitos no Funcionales

Los requisitos no funcionales, por otro lado, tratan sobre las características generales y la calidad de la herramienta más que sobre sus funciones específicas. La usabilidad se presenta como una prioridad; es esencial que la herramienta sea intuitiva, con una curva de aprendizaje que no sea abrumadora para los usuarios. En términos de rendimiento, se espera que la herramienta ofrezca tiempos de carga rápidos y una operación sin contratiempos, evitando latencias indeseadas durante la edición o previsualización.

Dada la naturaleza siempre cambiante de la industria, la escalabilidad de la herramienta es crucial, con una arquitectura diseñada para adaptarse y expandirse según las necesidades del proyecto. La compatibilidad es otro factor para considerar; la herramienta debe funcionar en diferentes sistemas operativos y hardware, y lograr una integración armoniosa con otras herramientas. La estabilidad garantiza una experiencia sin frustraciones, minimizando errores o fallos. Y, por supuesto, una documentación completa es indispensable para ayudar a los desarrolladores en su trabajo diario, proporcionando guías, ejemplos y soluciones a problemas comunes.

5.2.3. Interacción entre Usuarios y el Sistema

La interacción entre los usuarios y la herramienta es una piedra angular en el diseño de cualquier sistema. Para comprender y diseñar adecuadamente estas interacciones, es crucial identificar y entender a los diferentes actores involucrados y las operaciones que levarán a cabo.

Actores del Sistema

1. Diseñador de Cinemáticas: Este profesional se especializa en la creación y diseño de secuencias cinemáticas y diálogos en el juego, integrando todo en un flujo coherente dentro del juego.

2. Diseñador de Narrativa: Su enfoque se centra en la historia y los diálogos del juego. Aunque puede solaparse con el trabajo del diseñador de cinemáticas en algunos aspectos, este actor interactúa con la herramienta principalmente para moldear la narrativa y asegurarse de que los eventos y cinemáticas en el juego se alineen fielmente con la trama diseñada.
3. Testers: Estos profesionales tienen la tarea de probar la herramienta y el juego en su conjunto, garantizando que todo funcione conforme a las expectativas y detectando posibles errores o incongruencias.

Cada uno de estos actores interactuará con el sistema de maneras específicas y para diferentes propósitos. Por ejemplo, mientras que el desarrollador de videojuegos podría estar más interesado en la configuración técnica de las cinemáticas, el diseñador de narrativa podría centrarse en cómo se presentan y entrelazan las historias.

Diagrama de Casos de Uso:

A continuación, se presenta un diagrama de casos de uso que ilustra estas interacciones. Este diagrama visualmente representa las diferentes formas en que cada actor interactúa con el sistema y las operaciones o "casos de uso" que pueden realizar.

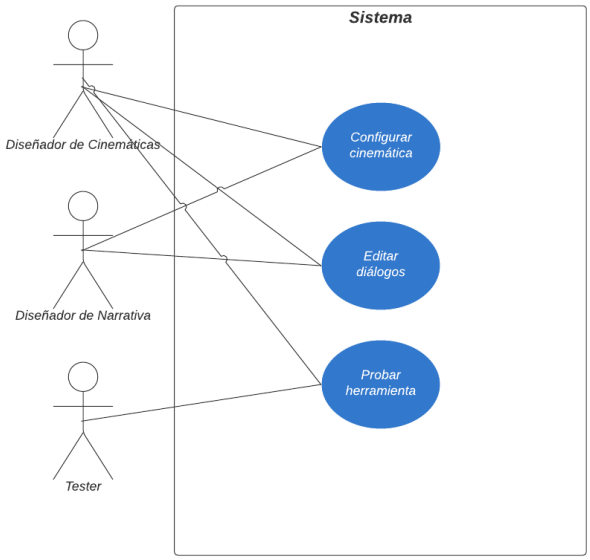


Figura 11: Diagrama casos de uso

FUENTE: ELABORACIÓN PROPIA

Al analizar este diagrama, es evidente que hay áreas de superposición y colaboración entre los actores. Estas áreas son críticas, ya que resaltan puntos en los que la herramienta debe ser especialmente robusta y fácil de usar, permitiendo una transición fluida entre diferentes partes del proceso de desarrollo.

5.3. Diseño de la Herramienta

Antes de sumergirse en la estructura y operatividad de nuestra herramienta, es fundamental clarificar cómo las cinemáticas son procesadas y cómo se ha estructurado la herramienta para orquestarlas de forma óptima.

En el núcleo del sistema se encuentra un script denominado "Cinematic Manager". Esta pieza clave tiene la responsabilidad de recopilar y administrar todas las cinemáticas, juntamente con sus eventos asociados, garantizando una gestión coherente y efectiva a lo largo de toda la experiencia del juego.

Es esencial diferenciar entre el "Cinematic Manager" y la herramienta visual que se ha implementado para configurar los eventos. Esta herramienta visual no sólo simplifica el proceso de edición, sino que también proporciona una representación gráfica intuitiva de los eventos, haciendo que la configuración y revisión sean más accesibles y claras para los desarrolladores.

5.3.1. Herramienta Visual para Configurar los Eventos

Cada cinemática se podría considerar como un conjunto de eventos, y ese es el motivo por el que estos eventos tienen que ser diseñados de forma eficiente y correcta. Esta es la razón por la que se ha implementado la herramienta visual por nodos.

Esta herramienta visual permite diseñar y editar los eventos asociados a cada cinemática en forma de árbol, es decir, cada nodo dentro de un árbol representará un evento y el árbol en sí mismo representará en conjunto todos los eventos de una cinemática.

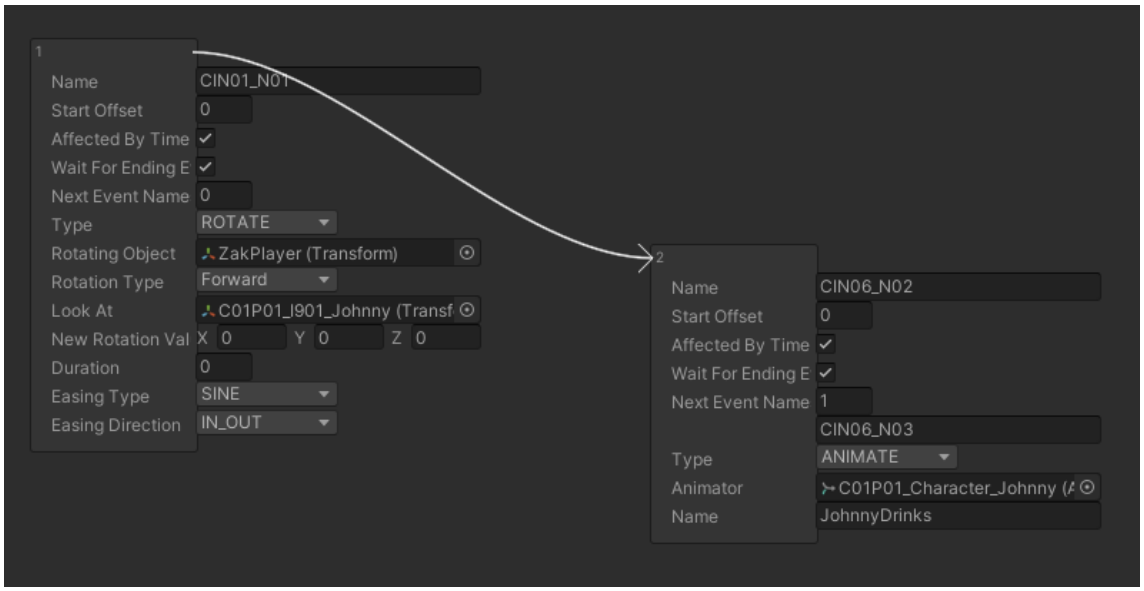


Figura 12: Ejemplo del editor de eventos

FUENTE: ELABORACIÓN PROPIA

Como se puede observar en la figura anterior, tenemos un número de nodos conectados entre sí, los cuales representan los eventos que completan la cinemática que representa en conjunto el árbol entero.

La principal intención de diseñar este editor de forma visual es facilitar la visualización de lo que está pasando dentro de cada cinemática de forma fácil y sin tener que pensar demasiado. De esta forma, la tarea de editar los eventos no es algo tan difícil y puedes mantener un orden de forma más sencilla.

Esto simplificará el proceso de solucionar errores o agilizará el proceso de edición de los eventos en un futuro cuando ciertos cambios se requieran en algún punto en específico, ya que de otra forma, manejar una cantidad tan grande de datos podría incluso llegar a ser sofocante para los desarrolladores, haciendo de una tarea simple algo inhumano.

Como he mencionado anteriormente, tenemos dos tipos de objetos en esta herramienta, los árboles que representan los conjuntos de eventos (TechTree) y los nodos, los cuales representan los eventos en sí (TechNode) como se puede observar en la siguiente figura.

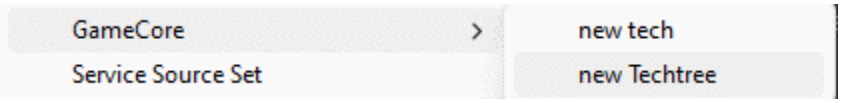


Ilustración 13: Techtree y TechNodes

FUENTE: ELABORACIÓN PROPIA

Estos son directamente configurables desde el inspector de Unity, al tratarse de un tipo de objetos llamados ScriptableObjects, estos pueden ser fácilmente editados y utilizados por cualquier otro código que queramos utilizar en el programa.

En nuestro caso, al estar tratando con eventos, cada nodo contendrá las cosas necesarias para su correcto funcionamiento dentro del juego. Uno de los parámetros más importantes entre las configuraciones es el tipo de evento del que se trata, ya que dependiendo del tipo de evento las configuraciones que aparecerán en pantalla serán diferentes como se puede observar en la siguiente figura.

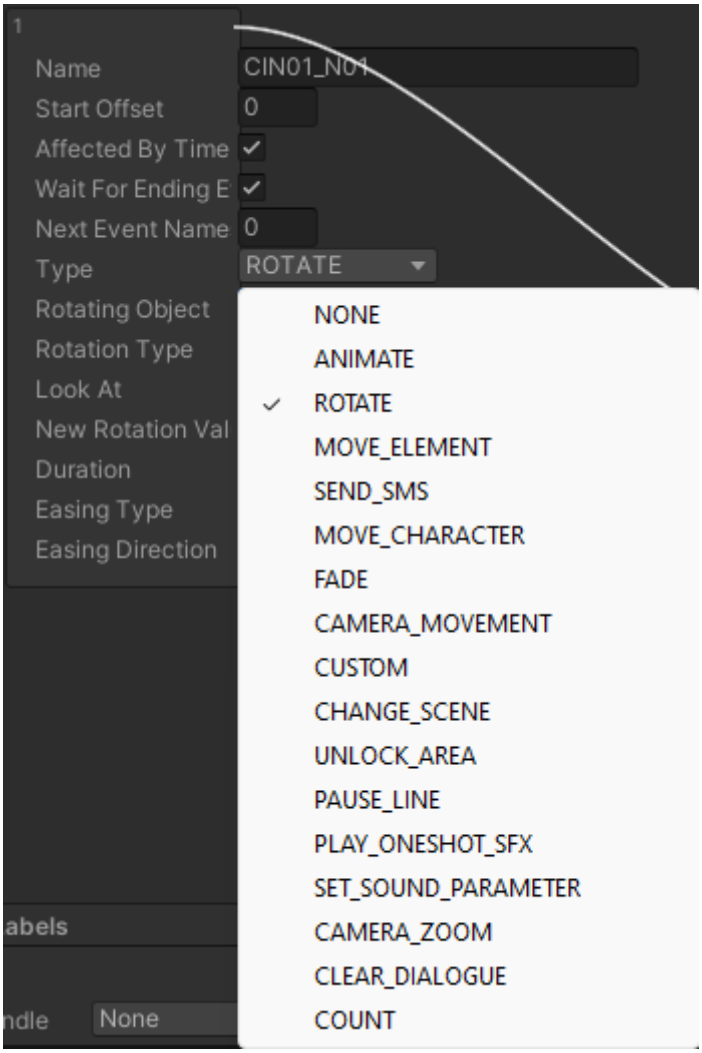


Ilustración 14: Tipos de eventos

FUENTE: ELABORACIÓN PROPIA

Cada uno de los tipos de eventos tendrá su configuración y estará diseñado para que solo tengas que introducir los valores necesarios para su funcionamiento.

Una vez terminada la configuración del árbol, cuando queramos utilizarlo en algún lado, todos los nodos serán almacenados en forma de lista, lo cual facilitará su lectura y hará que sea mucho más fácil de interpretarlos para los desarrolladores.

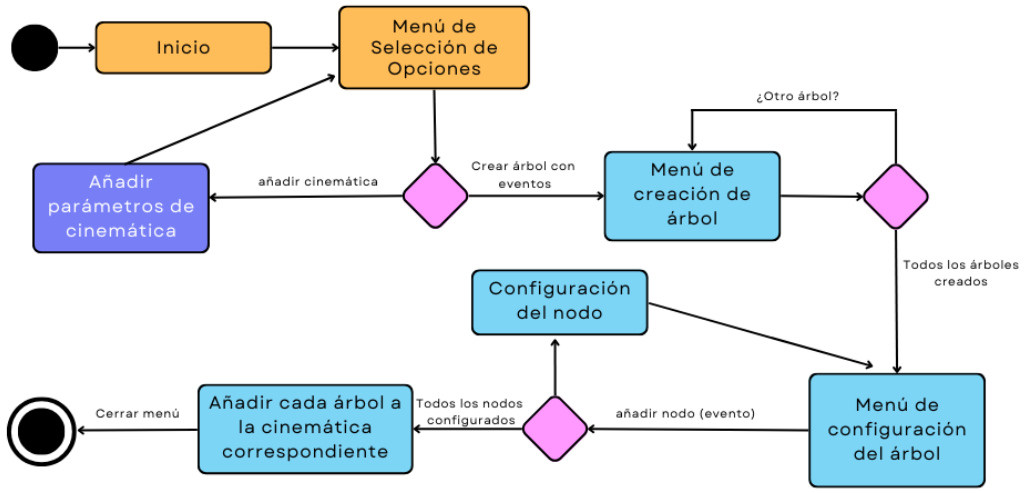


Figura 15: Diagrama de flujo UML

FUENTE: ELABORACIÓN PROPIA

5.3.2. Manejo de Cinemáticas en el Juego

Una vez explicada la herramienta visual, es importante hablar sobre la herramienta que gestiona las diferentes cinemáticas del juego, cambia entre ellas, y se encarga de que todo funcione de forma correcta.

Para ello, tenemos un script llamado "Cinematic Manager", el cual tendrá almacenados todas las cinemáticas con sus respectivos árboles conteniendo los eventos, contendrá la conversación de Articy Draft que se quiera mostrar, los cuales serán explicados en el apartado [5.5 Almacenamiento y Gestión de Datos].

Este script será el encargado de almacenar todas las cinemáticas que se ejecutarán en una misma escena e irá controlando cómo y cuándo se irán ejecutando los diferentes eventos configurados anteriormente dependiendo de los valores que hayamos asignado a los eventos o incluso al script mismamente.

5.4. Almacenamiento y Gestión de Datos

La organización adecuada de los datos es esencial para nuestra herramienta de gestión de cinemáticas. Esta organización no solo permite que la herramienta se adapte y crezca según las necesidades, sino que también facilita su uso. Un diseño claro y bien estructurado significa que tanto los nuevos usuarios como los experimentados pueden trabajar con eficacia.

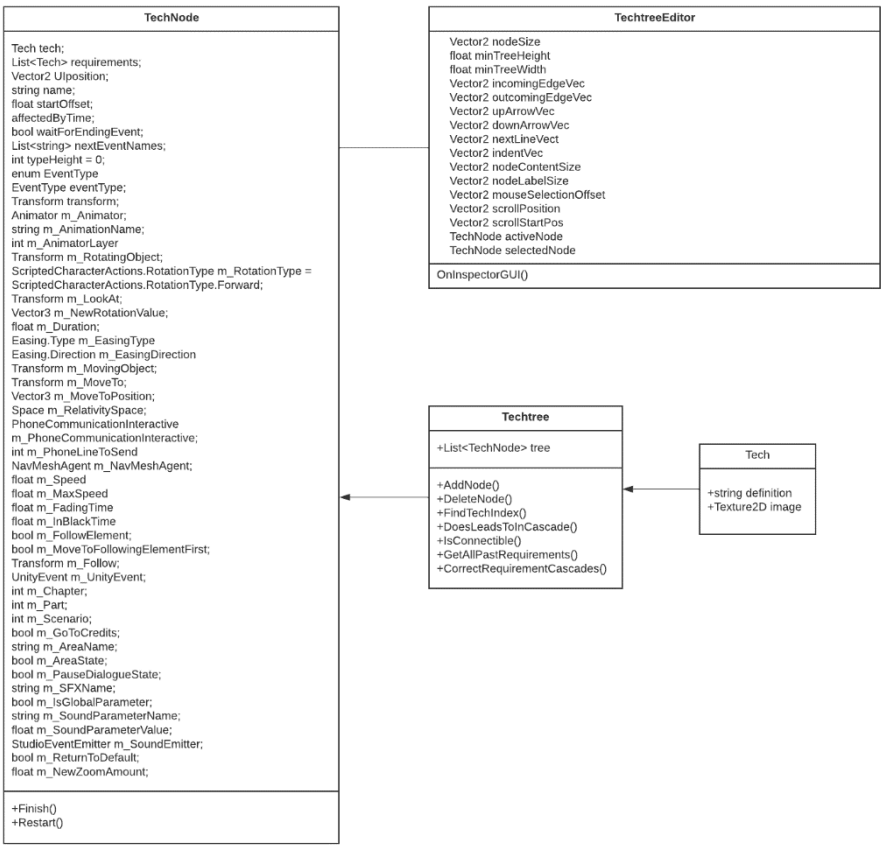


Figura 16: Diagrama de clases UML de herramienta visual

FUENTE: ELABORACIÓN PROPIA

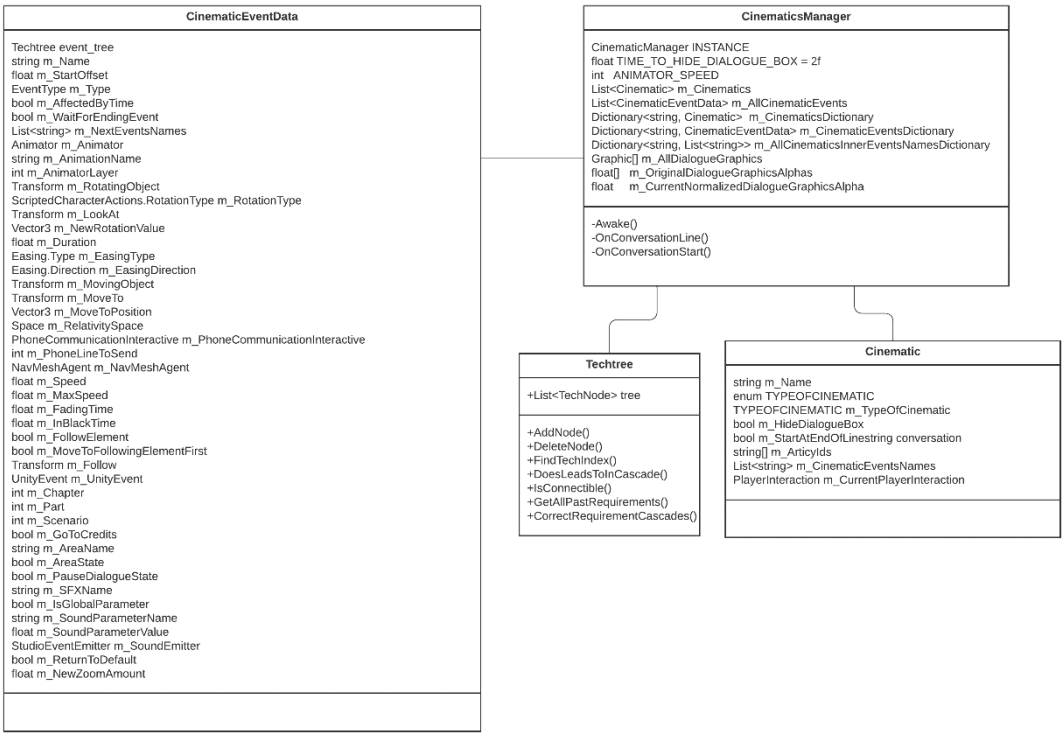


Figura 17:Diagrama de clases UML de herramienta de gestión de cinemáticas

FUENTE: ELABORACIÓN PROPIA

Un diagrama de clases es fundamental porque muestra cómo se relacionan las distintas partes del sistema entre sí. Ayuda a entender rápidamente la estructura del software y es una herramienta valiosa para planificar y revisar el diseño. En pocas palabras, actúa como una guía visual, asegurando que todos en el equipo comprendan cómo funciona el sistema.

5.5. Implementación de la Herramienta

Como se ha mencionado anteriormente, se han de hacer distinciones entre la herramienta visual para la configuración de cinemáticas y entre la herramienta que gestionará todo el contenido almacenado en estas.

Es por ello que, primero se empezará por la implementación de la herramienta visual y después se mencionarán las funcionalidades del script.

5.5.1. Herramienta Visual para Configurar los Eventos

Lo primero fue crear un script para poder configurar cómo quería representar los árboles en la herramienta, donde se declaran las clases "TechNode" y "Techtree".

La clase TechNode almacena todas las variables necesarias para que un evento funcione y sea funcional, es decir, almacena todas las variables que un nodo necesita para poder ser representado en el inspector de Unity. Algunos ejemplos de estas variables son la posición en la que se quiere representar, algunos parámetros de los eventos y una lista de los nombres de los siguientes eventos como se puede apreciar en la figura presente.

```
[System.Serializable]
25 referencias
public class TechNode
{
    public Tech tech;
    public List<Tech> requirements;
    public Vector2 UIposition;
    public string name;
    public float startOffset;
    public bool affectedByTime;
    public bool waitForEndingEvent;
    public List<string> nextEventNames;

    public int typeHeight = 0;

    19 referencias
    public enum EventType...

    public EventType eventType;
```

Figura 18: Clase TechNode

FUENTE: ELABORACIÓN PROPIA

Además de las variables mencionadas anteriormente, cada evento tiene una variable de tipo enumeración donde hay ciertas opciones a elegir, las cuales indicarán el tipo de evento que estamos configurando:

- **NONE**: No hay tipo de evento por lo que solamente almacenaríamos las variables mencionadas anteriormente en este nodo.
- **ANIMATE**: Requerirá un objeto de tipo Animator y un nombre asociado para poder aplicar este evento.

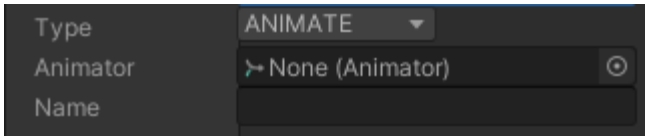


Figura 19: Evento ANIMATE

FUENTE: ELABORACIÓN PROPIA

- **ROTATE**: Requerirá varios objetos de tipo Transform para definir los objetos a los que se le aplicará la rotación, además de otras variables para la configuración de las rotaciones.

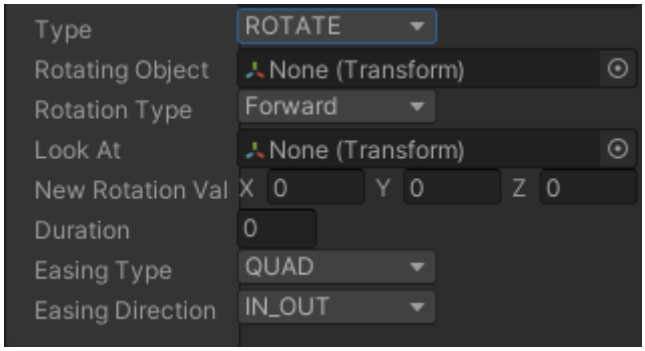


Figura 20: Evento ROTATE

FUENTE: ELABORACIÓN PROPIA

- **MOVE_ELEMENT**: Este será el evento que se usará para mover un elemento de un punto a otro, es por eso que se requiere el objeto a mover además de ciertas configuraciones para ajustar el destino al que este será movido.

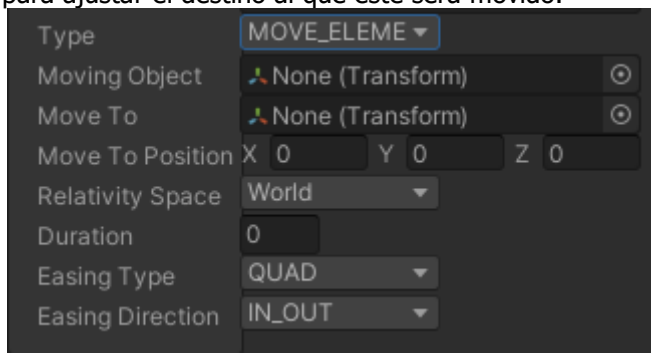


Figura 21: Evento MOVE_ELEMENT

FUENTE: ELABORACIÓN PROPIA

- **SEND_SMS**: Este será el evento utilizado para simular enviar un SMS.

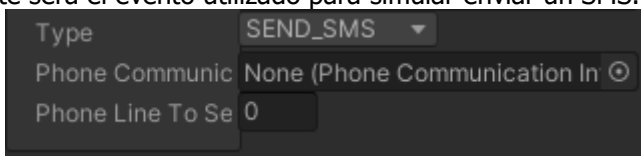


Figura 22: Evento: SEND_SMS

FUENTE: ELABORACIÓN PROPIA

- **MOVE_CHARACTER_TO_POINT**: Este será el evento usado para mover los personajes del juego de un punto a otro, es por eso que requerirá del objeto a desplazar, un Mav Mesh Agent para ayudar al movimiento y de ciertos parámetros para definir el destino.

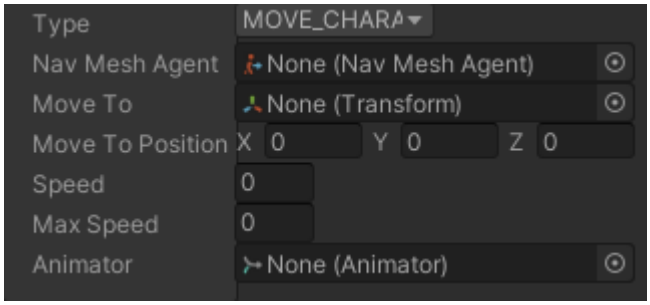


Figura 23: Evento MOVE_CHARACTER_TO_POINT

FUENTE: ELABORACIÓN PROPIA

- **FADE**: Este es el evento utilizado para aplicar un desvanecimiento, es por eso que solamente se necesitan establecer los tiempos.

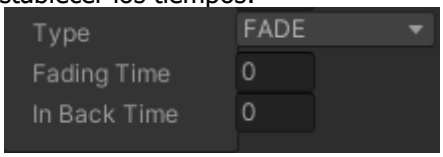


Figura 24: Evento FADE

FUENTE: ELABORACIÓN PROPIA

- **CAMERA_MOVEMENT**: Este evento se utilizará para desplazar la cámara, es por ello que se requiere de un objetivo al que apuntar con la cámara, y ciertos parámetros de rotación además de añadir un booleano para definir si la cámara debe o no seguir al objetivo.

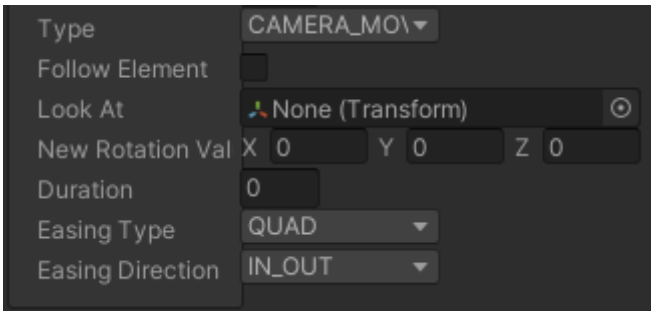


Figura 25: Evento CAMERA_MOVEMENT

FUENTE: ELABORACIÓN PROPIA

- **CUSTOM**: Este es el tipo de evento más versátil, ya que en él puedes configurar cualquier evento que quieras dentro de Unity. Esto es posible mediante los llamados Unity Events ya integrados dentro del motor de Unity.

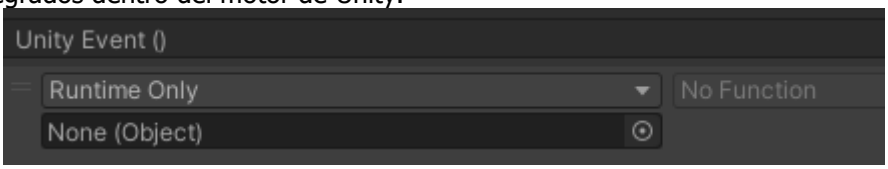


Figura 26: Evento CUSTOM

FUENTE: ELABORACIÓN PROPIA

- **CHANGE_SCENE**: En este evento definiremos un cambio de escena mediante la definición del capítulo, apartado y escenario correspondientes.

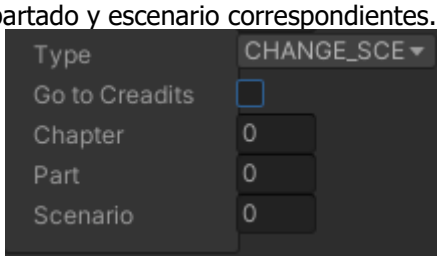


Figura 27: Evento CHANGE_SCENE

FUENTE: ELABORACIÓN PROPIA

- **UNLOCK_AREA**: Dentro de todas las áreas definidas en el juego, con este evento se desbloquea el área que queramos simplemente introduciendo su nombre en el campo correspondiente.

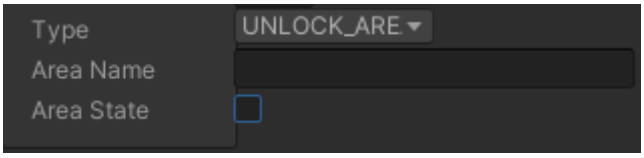


Figura 28: Evento UNLOCK_AREA

FUENTE: ELABORACIÓN PROPIA

- **PAUSE_LINE**: Con este evento simplemente conseguimos pausar el sistema de diálogos actual.

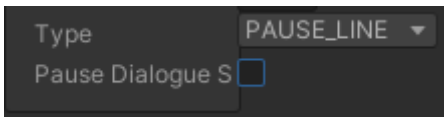


Figura 29: Evento PAUSE_LINE

FUENTE: ELABORACIÓN PROPIA

- **PLAY_ONESHOT_SFX**: Con este evento reproducimos un sonido cualquiera solamente almacenando su nombre en el bloque correspondiente.

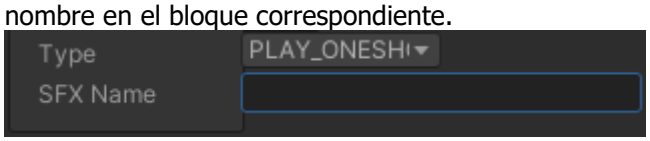


Figura 30: Evento PLAY_ONESHOT_SFX

FUENTE: ELABORACIÓN PROPIA

- **SET_SOUND_PARAMETER**: Con este evento es posible hacer que un cierto sonido se reproduzca desde algún objeto en particular.

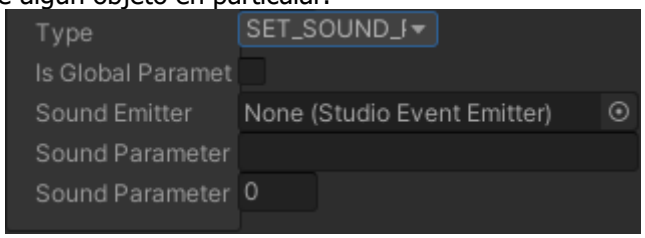


Figura 31: Evento SET_SOUND_PARAMETER

FUENTE: ELABORACIÓN PROPIA

- **CAMERA_ZOOM**: este evento es para cambiar los valores actuales de zoom establecimos en la cámara.

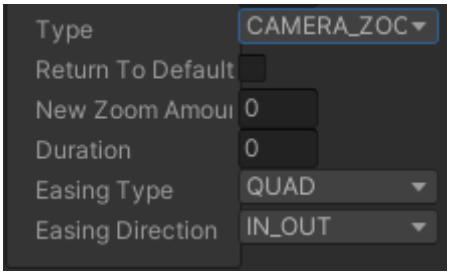


Figura 32: Evento CAMERA_ZOOM

FUENTE: ELABORACIÓN PROPIA

Una vez analizados todos los tipos de eventos que hay dentro de cada evento o nodo de nuestro árbol, es hora de analizar la implementación del árbol o cinemática que va a ser editable desde el inspector.

Lo primero de todo es establecer una lista de nodos asociados a cada árbol, los cuales serán los eventos asociados y que serán representados en cada uno de los árboles. A su vez, una función

para añadir y eliminarlos, para así tener una mayor libertad a la hora de editar los árboles como al desarrollador más conveniente le parezca.

Antes de eliminar uno de los nodos, es importante asegurarse que este existe en el árbol actual, al igual que antes de añadir uno de los nodos a la lista que contiene todos los nodos primero hay que asegurarse que no esté actualmente dentro de la lista. Para ello, se ha creado una función que, recibiendo un nodo por parámetro, este mirará en la lista de nodos para ver si este existe o no en ella.

```
public int FindTechIndex(Tech tech)
{
    for (int i = 0; i < tree.Count; i++)
    {
        if (tree[i].tech == tech) return i;
    }
    return -1;
}
```

Figura 33: Función para encontrar nodos

FUENTE: ELABORACIÓN PROPIA

Una vez que tenemos lo necesario para representar los nodos, es el momento de establecer las funciones necesarias para que estas puedan ser unidas. Para ello, es importante comprobar si el nodo en cuestión puede ser conectado de alguna manera, y en el caso de ser así, devolver un booleano positivo para indicar que dos nodos pueden ser unidos.

```
public bool DoesLeadsToInCascade(int query, int subject)
{
    foreach (Tech t in tree[query].requirements)
    {
        if (t == tree[subject].tech) return true;
        if (DoesLeadsToInCascade(FindTechIndex(t), subject)) return true;
    }
    return false;
}

1 referencia
public bool IsConnectible(int incomingNodeIdx, int outgoingNideIdx)
{
    if (incomingNodeIdx == outgoingNideIdx) return false;
    return !(DoesLeadsToInCascade(incomingNodeIdx, outgoingNideIdx) || DoesLeadsToInCascade(outgoingNideIdx, incomingNodeIdx));
}
```

Figura 34: Funciones para comprobar conexiones

FUENTE: ELABORACIÓN PROPIA

Una vez tenemos la clase que representa los árboles ha sido definida con todas las funciones mencionadas anteriormente, es el momento de crear el script llamado "TechtreeEditor" el cual será el encargado de representar todos los elementos visuales en el inspector de manera correcta. Cabe mencionar que este script debe estar almacenado dentro de una carpeta llamada "Editor" para decirle a Unity que no añada este script a la build, lo que indica que es un script utilizado solamente para el desarrollo.

Se definirá la clase TechtreeEditor, la cual heredará de la clase Editor de Unity. Además de contener en ella todas las variables necesarias para una correcta organización para todo lo relacionado con el apartado visual de la herramienta:

- Variables para el posicionamiento

- Variables para el scrolling y permitir movernos por el editor
- Variables para indicar el nodo actual seleccionado

Ahora es el momento para definir la única función de esta clase, la cual será llamada "OnInspectorGUI". Esta función le indica a Unity que estamos tratando de modificar su propia función, por lo que estaremos sobrescribiendo una función propia del motor.

Primero, es importante tener la referencia del árbol activo, es decir, el que tenga el desarrollador seleccionado en el inspector y quiera modificar. Esto es posible mediante una variable que viene del editor de Unity llamada "target", el cual indica el árbol activo en nuestro caso.

```
Techtree targetTree = (Techtree)target; //Get the associated Thechtree
```

Figura 35: Obtener referencia al árbol activado

FUENTE: ELABORACIÓN PROPIA

También es importante gestionar los inputs del ratón, ya que, al añadir un nuevo nodo, para mover los nodos actuales o para simplemente donde está posicionado el ratón es necesario saber la posición del mismo, saber si se está pulsando algún botón, y gestionar las consecuencias que ello conlleva.

Ahora es el momento de empezar con la representación del árbol en el inspector. Para ello, se define un área de scroll, el cual será el área de trabajo que utilizaremos para representar todos los elementos visuales necesarios para la herramienta.

Ahora es el momento de definir los elementos de cada evento en una posición respecto a nuestro ratón. Es por eso, que primero de todo vamos a representar un rectángulo que actuará como base para poder posicionar todos los demás elementos dentro del inspector de manera ordenada y coherente mediante funciones definidas en Unity como funciones de "EditorGUI".

Para cada variable que se quiera mostrar en el inspector, un nuevo rectángulo tiene que ser creado para almacenar el nombre de la variable y posteriormente, se posicionará el campo correspondiente acorde al tipo de dato al que corresponda la variable en cuestión, por ejemplo, un texto.

```
EditorGUI.LabelField(new Rect(targetTree.tree[nodeIc
targetTree.tree[nodeIdx].name = EditorGUI.TextField(
```

Figura 36: Representación de un texto con EditorGUI

FUENTE: ELABORACIÓN PROPIA

Una vez representadas todas las variables teniendo en cuenta el tipo del evento, para mostrar unas variables u otras, es el momento de representar las conexiones entre los nodos que hemos configurado anteriormente.

Para ello, recorreremos la lista de requerimientos de cada uno de los nodos, y si el nodo en cuestión existe en el árbol actual, se representará una curva de Bezier desde el nodo actual hasta el requerido. También, se ha añadido una flecha al final de la curva para representar de una

forma más clara la dirección de la conexión, por lo que se representan dos líneas diagonales al extremo de la curva de Bezier, es decir, en el nodo objetivo.

```
//Draw conenction curve
Handles.DrawBezier(targetTree.tree[nodeIdx].UIposition - scrollPosition + outcomingEdgeVec,
    targetTree.tree[reqIdx].UIposition - scrollPosition + incomingEdgeVec,
    targetTree.tree[nodeIdx].UIposition - scrollPosition + outcomingEdgeVec + Vector2.left * 100,
    targetTree.tree[reqIdx].UIposition - scrollPosition + incomingEdgeVec + Vector2.right * 100,
    Color.white,
    null,
    3f
    );

//Draw arrow
Handles.DrawLine(targetTree.tree[nodeIdx].UIposition - scrollPosition + outcomingEdgeVec, targetTr
Handles.DrawLine(targetTree.tree[nodeIdx].UIposition - scrollPosition + outcomingEdgeVec, targetTr
```

Figura 37: Conexiones entre nodos

FUENTE: ELABORACIÓN PROPIA

Ahora es el momento de manejar y controlar todos los eventos del ratón, puesto que la intención es la de poder mover los nodos, añadir conexiones entre nodos y eliminarlos como al desarrollador más cómodo le resulte.

Se comprueba la posición del ratón para ver si este se encuentra dentro de algún nodo, si esto ocurre, se comprueba si se pulsa algún botón del ratón para proceder a controlar las posibles acciones que el usuario puede llegar a hacer:

- Botón izquierdo: Mover la posición de nodo.
- Botón derecho: Seleccionar el nodo para establecer una conexión con otro de los nodos, uniéndolos por una flecha.
- Botón central: Desplazarte por el editor.

5.5.2. Script Para el Manejo de Cinemáticas

En este script se tiene almacenado todo lo relacionado con las cinemáticas. Primero de todo nos encontramos ante la clase Cinematic, la cual contiene el nombre, el tipo de cinemática, la conversación que se va a ejecutar y los eventos que contiene cada una de las cinemáticas.

```
public class Cinematic
{

    public string       m_Name;
    3 referencias
    public enum TYPEOFCINEMATIC { NEEDED, FLAVOUR };
    public TYPEOFCINEMATIC m_TypeOfCinematic = TYPEOFCINEMATIC.FLAVOUR;
    public bool         m_HideDialogueBox = false;
    public bool         m_StartAtEndOfLine = false;
    [ConversationPopup(true)]
    public string conversation = string.Empty;
    public string[] m_ArticyIds;
    public List<string> m_CinematicEventsNames;

    [HideInInspector] public PlayerInteraction m_CurrentPlayerInteraction;
}
```

Ilustración 38: Clase Cinematic

FUENTE: ELABORACIÓN PROPIA

Por otro lado, nos encontramos la clase CinematicEventData, la cual contiene todos los datos de los eventos que no son configurados en la herramienta visual. Desde el árbol con los eventos, sus características y todas las variables necesarias para que funcionen los eventos correctamente en el entorno del videojuego.

```
[Tooltip("Techtree of the event")]
public Techtree event_tree;
[Tooltip("Works as ID")]
public string m_Name;
[Tooltip("Time to wait before starting this event when it is called")]
public float m_StartOffset;
[Tooltip("Type of Event")]
public EventType m_Type = EventType.CUSTOM;
[Tooltip("Is this Event affected by time or works in pause, etc.?")]
public bool m_AffectedByTime = true;
[Tooltip("Should wait for this Event to finish before calling the next one and notifying as finished?")]
public bool m_WaitForEndingEvent = true;
[Tooltip("The next Event/s to be played")]
public List<string> m_NextEventsNames = new List<string>();

private bool m_IsFinished = false;
```

Fuente 39: Clase CinematicEventData

FUENTE: ELABORACIÓN PROPIA

Lo primero que se ejecuta de este script es la función Start() de Unity, donde se van a agrupar todos los eventos que hay y se meten en un diccionario para tenerlo todo más ordenado y más fácil para ser accedido posteriormente. Para ello, se han implementado funciones para recorrerse la lista de siguientes eventos que contiene cada uno de los eventos dentro de la cinemática seleccionada.

Ahora, es el momento de establecer todas las funciones que van a hacer que el funcionamiento de las cinemáticas sea posible. La principal función de este script es la que se encarga de iniciar las cinemáticas, la cual es llamada "StartCinematic". Esta será llamada indicando el nombre de la cinemática que quieras ejecutar, y la función se encargará de reiniciar la cinemática y cargarla

de la forma correcta. Por último, tras verificar que no haya corrutinas activas, se recorrerá el listado de eventos que esta cinemática contiene, iniciando una corrutina con cada una de ellas.

```
StartCoroutine(CallWhenCinematicFinished(cinematicName, callWhenFinished));

foreach (string eventName in currentCinematic.m_CinematicEventsNames)
{
    StartCoroutine(CinematicEvent(m_CinematicEventsDictionary[eventName]));
}
```

Ilustración 40: Iniciar los eventos de una cinemática

FUENTE: ELABORACIÓN PROPIA

Otra de las funciones más importantes, y que será llamado cada vez que un evento termine, será la función llamada "EndCinematicEvent()". Esta función será la encargada de verificar si el evento tiene asignado algún evento siguiente para así poder empezar una corrutina con cada uno de ellos y así ejecutarlos.

```
private void EndCinematicEvent(CinematicEventData data)
{
    foreach (string eventName in data.m_NextEventsNames)
    {
        if (eventName.Length > 0)
        {
            StartCoroutine(CinematicEvent(m_CinematicEventsDictionary[eventName]));
        }
    }
}
```

Figura 41: Función EndCinematicEvent

FUENTE: ELABORACIÓN PROPIA

A continuación, comenzaremos por crear una función por cada tipo de evento que tengamos, haciendo que, cada tipo de evento tenga su propia conducta diferente al resto.

- AnimateEvent()
  Esta función coge los parámetros establecidos a la hora de configurar el evento en cuestión y ejecuta la animación asignada. Después, verifica si la opción de esperar a que el evento termine está activo y cuando la animación en cuestión termina, el evento se da por terminado llamando a la función "EndCinematicEvent()" explicada anteriormente.

```
data.m_Animator.CrossFade(data.m_AnimationName, 0.0f, data.m_AnimatorLayer);
yield return null;
AnimatorClipInfo[] allInfo  = data.m_Animator.GetCurrentAnimatorClipInfo(data.m_AnimatorLayer);
float              duration = allInfo.Length == 0 ? 0 : allInfo.Last().clip.length;

if (data.m_WaitForEndingEvent)
{
    float currentTime = 0.0f;
    while (currentTime < duration)
    {
        currentTime += data.m_AffectedByTime ? Time.deltaTime : Time.unscaledDeltaTime;
        yield return null;
    }
}

data.Finish();
EndCinematicEvent(data);
```

Figura 42: Función AnimateEvent

FUENTE: ELABORACIÓN PROPIA

- RotateEvent()
  En esta función, se obtendrán los valores establecidos en el evento y dependiendo del tipo de rotación, se aplicará al transform que se le pasa por parámetros. Tras aplicar la rotación, se verifica si el evento ha terminado para poder darla por terminada y llamar a la función "EndCinematicEvent()".

```
private IEnumerator RotateEvent(CinematicEventData data)
{
    Vector3 newRotValue;
    if (data.m_RotationType == ScriptedCharacterActions.RotationType.Forward)
    {
        newRotValue = (data.m_LookAt == null ? data.m_NewRotationValue : data.m_LookAt.position) -
                      data.m_RotatingObject.position;
        newRotValue.y = 0;
        newRotValue.Normalize();
    }
    else
    {
        newRotValue = data.m_NewRotationValue;
    }

    ScriptedCharacterActions.Rotate(data.m_RotationType, data.m_RotatingObject, newRotValue, data.m_Duration,
                                    data.m_EasingType, data.m_EasingDirection);

    if (data.m_WaitForEndingEvent)
    {
        while (ScriptedCharacterActions.IsRotating(data.m_RotatingObject))
        {
            yield return null;
        }
    }

    data.Finish();
    EndCinematicEvent(data);
}
```

Ilustración 43: Función RotateEvent

FUENTE: ELABORACIÓN PROPIA

- MoveElementEvent()
  En esta función, se utiliza la función definida para mover el transform definido de un lugar a otro. Después se verifica que el evento haya terminado y se da por terminado llamando a la función "EndCinematicEvent()".

```
private IEnumerator MoveElementEvent(CinematicEventData data)
{
    Vector3 dest = data.m_MoveTo == null ? data.m_MoveToPosition : data.m_MoveTo.position;

    ScriptedCharacterActions.Move(data.m_MovingObject, dest, data.m_Duration, data.m_RelativitySpace, data.m_EasingType, data.m_EasingDirection);

    if (data.m_WaitForEndingEvent)
    {
        while (ScriptedCharacterActions.IsMoving(data.m_MovingObject))
        {
            yield return null;
        }
    }

    data.Finish();
    EndCinematicEvent(data);
}
```

Figura 44: Función MoveElementEvent()

FUENTE: ELABORACIÓN PROPIA

- SendPhoneMessageEvent()
  Con esta función simplemente se llamará a una función para que añada esa línea a los diálogos actuales y después se terminará el evento.

```
private void SendPhoneMessageEvent(CinematicEventData data)
{
    data.m_PhoneCommunicationInteractive.SendLine(data.m_PhoneLineToSend);
    data.Finish();
    EndCinematicEvent(data);
}
```

Figura 45: Función SendPhoneMessageEvent

FUENTE: ELABORACIÓN PROPIA

- MoveCharacterEvent()
  Primero se calcula el path desde el transform hasta el punto al que se quiere ir mediante el NavMesh, y se verifica que este es posible. En caso de serlo, se compara la distancia entre el objetivo y el cuerpo en movimiento hasta que la distancia sea menos a un valor mínimo. Entonces es cuando se da por terminado el evento.

```
IEnumerator AnimateMovement()
{
    const float minDistance = .25f;
    while (data.m_NavMeshAgent.isActiveAndEnabled &&
           data.m_NavMeshAgent.remainingDistance > minDistance)
    {
        if (data.m_NavMeshAgent.isStopped)
        {
            data.m_NavMeshAgent.isStopped = false;
        }
        else
        {
            data.m_Animator.SetFloat(ANIMATOR_SPEED,
                                     data.m_NavMeshAgent.velocity.magnitude / data.m_MaxSpeed);
        }

        yield return null;
    }

    data.m_Animator.SetFloat(ANIMATOR_SPEED, 0);
}
```

Figura 46: Función AnimateMovement

FUENTE: ELABORACIÓN PROPIA

- FadeEvent()
  Esta función coge los tiempos establecidos para hacer el desvanecimiento, espera en ese estado hasta que el tiempo de duración es alcanzado y después vuelve a deshacer el desvanecimiento de la misma forma antes de dar por terminado el evento.

```
Fade.FadeOut(data.m_FadingTime);
float currentTime = 0.0f;
while (currentTime < data.m_FadingTime)
{
    currentTime += data.m_AffectedByTime ? Time.deltaTime : Time.unscaledDeltaTime;
    yield return null;
}

currentTime = 0.0f;
while (currentTime < data.m_InBlackTime)
{
    currentTime += data.m_AffectedByTime ? Time.deltaTime : Time.unscaledDeltaTime;
    yield return null;
}

Fade.FadeIn(data.m_FadingTime);
currentTime = 0.0f;
while (currentTime < data.m_FadingTime)
{
    currentTime += data.m_AffectedByTime ? Time.deltaTime : Time.unscaledDeltaTime;
    yield return null;
}
```

Figura 47: Función FadeEvent

FUENTE: ELABORACIÓN PROPIA

- CameraMovementEvent()
  Primero se comprueba si la opción de seguimiento está activada. En caso de no estarlo, la cámara simplemente mirará al objetivo establecido y dará por terminado el evento.

  Por otro lado, si la opción de seguimiento sí está activada, la cámara se girará para mirar al objetivo mientras este lo sigue posicionándose a una cierta distancia de este y dando por terminado el evento cuando transcurra el tiempo establecido.

- CustomEvent()
  Esta función permite crear eventos personalizados del tipo Unity Event. Con ello, invocará los eventos creados desde el inspector y dará por terminado el evento una vez este termine.

```
data.m_UnityEvent.AddListener(data.Finish);
data.m_UnityEvent.Invoke();
yield return null;
```

Figura 48: Invocar Unity Events

FUENTE: ELABORACIÓN PROPIA

- UnlockArea()
  Esta función simplemente llama a una función pasándole como parámetro el nombre de un área y cambiará el booleano que indica si esta está activada o no para que esté activada.

- PauseDialogue()

Esta función llamará a una función del DialogueManager que evitará que los diálogos sigan avanzando o los volverá a activar, dependiendo del parámetro establecido a la hora de configurar el evento.

```
if (data.m_PauseDialogueState)
{
    DialogueManagerSettings.PauseDialogue();
}
else
{
    DialogueManagerSettings.ResumeDialogue();
}
```

Figura 49: Función PauseDialogue

FUENTE: ELABORACIÓN PROPIA

- ChangeSceneEvent()
  Se verifica si la opción de ir a los créditos está activado. Si no lo está, cogerá los valores establecidos para el capítulo, parte y escenario y cambiará de escena mediante el SceneManager.

```
StartCoroutine(SceneManagement.ChangeSceneCoroutine(data.m_Chapter, data.m_Part, data.m_Scenario));
```

Figura 50: Cambio de escena

FUENTE: ELABORACIÓN PROPIA

- PlayOneShotSFX()
  Mediante el nombre del sonido introducido por parámetro, se ejecutará el sonido seleccionado y dará por terminado el evento en cuestión.

- ChangeEmitterParameter()
  Mediante esta función, se le asignará un Studio Event Emitter a un sonido seleccionado y se establecerán los nuevos parámetros del sonido en cuestión.

- ChangeGlobalParameter()
  Con esta función simplemente hará que el sonido en cuestión suene de manera global y no tenga ninguna fuente de sonido específico.

- CameraZoomEvent()
  Esta función, cambiará el valor actual del zoom de la cámara mediante el CameraController, al cual pasándole los parámetros del nuevo zoom será suficiente para que este se encargue de cambiar los parámetros de la cámara

- ClearDialogueEvent()
  Esta función se encargará de hacer que todos los diálogos se limpien, es decir, se vaciarán todas las listas que contengan diálogos en ese preciso momento y se eliminará su contenido.

6. Evaluación

6.1. Evaluación de Rendimiento y Funcionalidad

Realizar pruebas es un paso esencial en cualquier proceso de desarrollo, especialmente cuando se trata de herramientas destinadas a ser utilizadas por profesionales del sector. Estas pruebas no solo garantizan que el software funcione como se esperaba, sino que también ofrecen una valiosa retroalimentación sobre cómo se puede mejorar la herramienta para satisfacer mejor las necesidades de los usuarios.

Con el objetivo de obtener una perspectiva honesta y directa de quienes podrían beneficiarse de nuestra herramienta, hemos creado un cuestionario dirigido a desarrolladores. Esta encuesta se diseñó para capturar sus impresiones, comentarios y sugerencias luego de haber probado la herramienta.

6.1.1. Justificación de las Preguntas del Cuestionario para la Herramienta

La elaboración de cuestionarios requiere un diseño cuidadoso para asegurarse de que las preguntas proporcionen información valiosa y relevante para los objetivos del proyecto. A continuación, se detalla la justificación de las preguntas formuladas en el cuestionario de retroalimentación para la herramienta de gestión de cinemáticas:

1. **Experiencia en el desarrollo de videojuegos:** Es fundamental entender la experiencia previa de los participantes en el desarrollo de videojuegos. Esta pregunta nos permite clasificar las respuestas basándonos en el nivel de experiencia del usuario, lo cual podría influir en su percepción y expectativas de la herramienta. Por ejemplo, un desarrollador veterano podría tener diferentes criterios y expectativas en comparación con alguien que acaba de entrar en el sector.
2. **Plataformas de desarrollo:** El desarrollo de videojuegos varía según la plataforma, y cada una tiene sus propios desafíos y requisitos. Al entender en qué plataformas desarrollan con más frecuencia, podemos obtener información sobre la versatilidad y aplicabilidad de la herramienta en diferentes contextos de desarrollo.
3. **Usabilidad e Interfaz de Usuario:** La facilidad de uso es uno de los factores más críticos en la adopción y satisfacción del usuario de cualquier herramienta de *software*. Esta pregunta nos da una idea directa sobre la calidad de la experiencia del usuario con nuestra herramienta y si la interfaz es lo suficientemente intuitiva.
4. **Funcionalidad y eficiencia:** La principal razón para desarrollar esta herramienta es facilitar y mejorar el proceso de creación y gestión de cinemáticas en el desarrollo de videojuegos. Con esta pregunta, buscamos determinar si estamos logrando este objetivo.
5. **Estabilidad y problemas:** Es esencial identificar rápidamente los errores o problemas que los usuarios puedan encontrar. Esta pregunta no solo nos permite identificar fallos técnicos, sino también posibles áreas de mejora.
6. **Sugerencias y comentarios:** Siempre es útil tener un espacio abierto para que los usuarios compartan sus ideas y sugerencias. Esta sección proporciona un espacio para que los usuarios se expresen libremente y aporten ideas que quizás no hayamos considerado.

6.1.2. Cuestionario de Retroalimentación para la Herramienta

1. ¿Cuánto tiempo llevas trabajando en el desarrollo de videojuegos?
    a. Menos de 1 año
    b. De 1 a 3 años
    c. De 3 a 5 años
    d. Más de 5 años

2. ¿En qué plataformas sueles desarrollar?
    a. PC
    b. Consolas
    c. Móviles
    d. VR
    e. Otros (Especifique)
3. ¿Cómo calificarías la interfaz de usuario de la herramienta?
    a. Muy intuitiva (5)
    b. Intuitiva (4)
    c. Neutral (3)
    d. Confusa (2)
    e. Muy confusa (1)
4. ¿La herramienta facilitó la creación y gestión de cinemáticas en tu juego?
    a. Totalmente de acuerdo
    b. De acuerdo
    c. Neutral
    d. En desacuerdo
    e. Totalmente en desacuerdo
5. ¿Te enfrentaste a algún problema o bug mientras usabas la herramienta?
    a. Sí (Por favor, especifica)
    b. No
6. ¿Qué características añadirías o cambiarías en la herramienta?

Comentarios adicionales

6.2. Feedback y Validación con Usuarios

Tras la distribución y recolección de los cuestionarios, se analizaron las respuestas de los 15 participantes para obtener una comprensión clara del desempeño y la percepción de nuestra herramienta. A continuación, presentamos las conclusiones principales:

1. **Experiencia en el desarrollo de videojuegos:** La mayoría de los encuestados tienen más de 3 años de experiencia en el desarrollo de videojuegos. Esto nos indica que nuestra herramienta fue evaluada en su mayoría por profesionales con un conocimiento sólido en el ámbito, lo que añade peso y relevancia a su *feedback*.

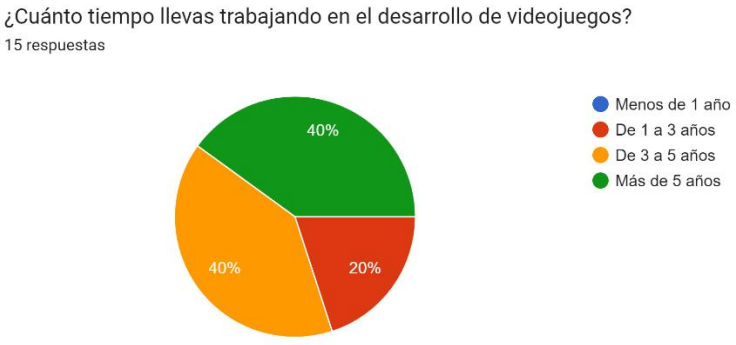


Figura 51: Pregunta 1 cuestionario

FUENTE: ELABORACIÓN PROPIA

2. **Plataformas de desarrollo:** Aproximadamente el 60% de los encuestados desarrolla principalmente para PC, seguido por un 26% para móviles, y el restante 15% está en las consolas. Esto refleja que nuestra herramienta ha sido probada en una variedad de contextos de desarrollo.

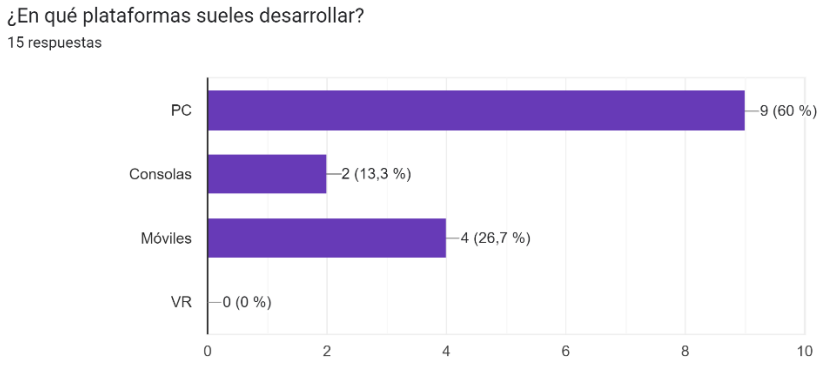


Figura 52: Pregunta 2 cuestionario

FUENTE: ELABORACIÓN PROPIA

3. **Usabilidad e Interfaz de Usuario:** El 86% de los participantes calificó la interfaz de usuario de la herramienta como "Muy intuitiva" o "Intuitiva". Sin embargo, algunos comentaron que ciertas funciones podrían ser más accesibles o que la herramienta podría beneficiarse de tutoriales adicionales para usuarios menos expertos.

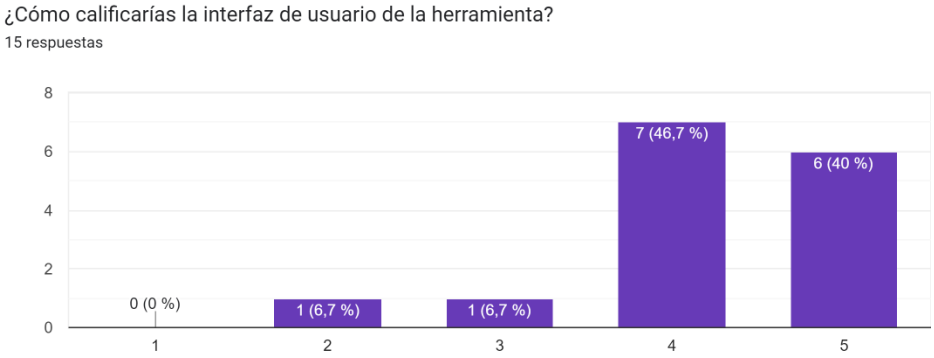


Figura 53: Pregunta 3 cuestionario

FUENTE: ELABORACIÓN PROPIA

4. **Funcionalidad y eficiencia:** El 86% de los encuestados estuvo "Totalmente de acuerdo" o "De acuerdo" en que la herramienta facilitó la creación y gestión de cinemáticas en sus proyectos. Esto es un testimonio positivo de la eficacia de la herramienta. A pesar de ello, algunos usuarios sugirieron características adicionales que podrían enriquecer aún más la experiencia.

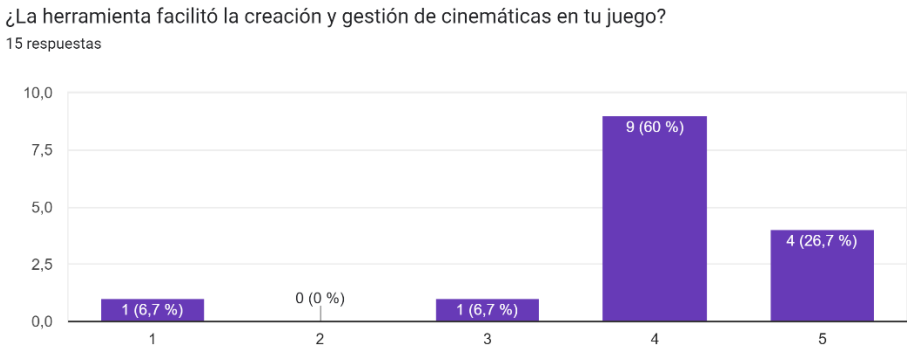


Figura 54: Pregunta 4 cuestionario

FUENTE: ELABORACIÓN PROPIA

5. **Estabilidad y problemas:** Un 20% de los usuarios reportó haber encontrado problemas o bugs al usar la herramienta. Las descripciones de estos problemas varían, pero nos proporcionan una base sólida sobre la cual trabajar y mejorar en futuras actualizaciones.

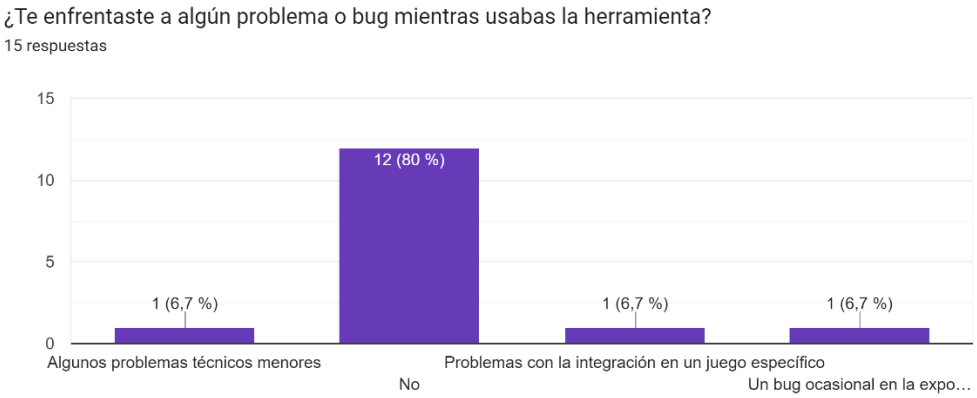


Figura 55: Pregunta 5 cuestionario

FUENTE: ELABORACIÓN PROPIA

6. **Sugerencias y comentarios:** Varios encuestados ofrecieron ideas valiosas sobre cómo mejorar la herramienta. Algunas de las sugerencias más comunes incluyeron la incorporación de plantillas predefinidas para ciertos tipos de cinemáticas y la posibilidad de integración con otras herramientas populares de desarrollo.

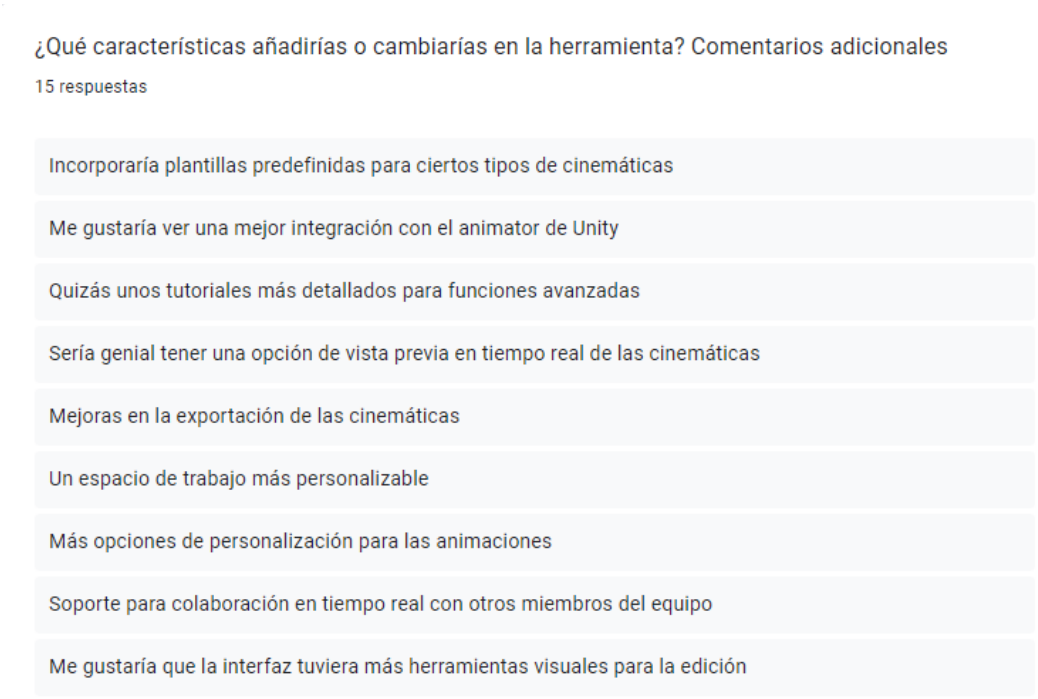


Figura 56: Pregunta 6 cuestionario

FUENTE: ELABORACIÓN PROPIA

7. Conclusiones

7.1. Retos y Lecciones Aprendidas

Durante el proceso de desarrollo de este proyecto, nos encontramos con retos que, si bien en momentos parecieron insuperables, se convirtieron en oportunidades de crecimiento y aprendizaje. Cada desafío nos proporcionó una nueva perspectiva sobre la industria del desarrollo de videojuegos y nos permitió pulir nuestra herramienta de manera que mejor satisfaga las necesidades de nuestros usuarios.

Uno de los principales retos fue la **complejidad técnica** que implicaba el diseño e implementación de una herramienta que se ajustara a las exigencias específicas de los desarrolladores de videojuegos. A menudo nos encontrábamos en la difícil posición de tener que reconsiderar ciertas funcionalidades o características debido a limitaciones técnicas que no habíamos anticipado. Esta situación nos enseñó la importancia de la adaptabilidad y de contar con un equipo técnico robusto y bien informado.

El ***feedback* diverso** de los usuarios, aunque esencial, también presentó un desafío. La variedad de opiniones y experiencias nos puso en una posición donde tuvimos que discernir y priorizar las características en función de su impacto y viabilidad. Esta experiencia reafirmó la importancia de la comunicación constante y efectiva con los usuarios y la capacidad de sintetizar y actuar según sus opiniones.

En cuanto a las **lecciones aprendidas**, la importancia de la iteración fue una que destacó por encima de todas. Nos dimos cuenta de que raramente las cosas salen perfectas en el primer intento. El éxito, más que ser un destino, es un proceso de constante aprendizaje, adaptación y mejora. Esta experiencia nos enseñó a valorar el *feedback*, ser flexibles y, en ocasiones, decidir entre profundidad y alcance. A veces, es preferible hacer menos, pero hacerlo excepcionalmente bien.

7.2. Resultados Finales y Logros del Proyecto

Con el cierre del proyecto, es vital revisar los objetivos que inicialmente se plantearon y la medida en la que fueron alcanzados. Cada objetivo actúa como un hito que orienta el esfuerzo y define el parámetro de éxito.

Respecto al primer objetivo de familiarizarme con las herramientas propias del desarrollo profesional de videojuegos en Unity, puedo afirmar que fue un éxito rotundo. No sólo se logró un entendimiento profundo de las herramientas estándar de Unity, sino que también se investigaron y experimentaron con extensiones y complementos avanzados que potenciaron la funcionalidad y versatilidad del trabajo.

El segundo objetivo, centrado en el desarrollo e implementación de un sistema innovador de cinemáticas, se presentó como un reto apasionante. Tras un detenido análisis y diseño, se concibió una herramienta que no solo cumple con los estándares contemporáneos en cuanto a la calidad y fluidez de las cinemáticas, sino que también se adapta a las necesidades cambiantes del videojuego con el que se trabajó. El *feedback* que se obtuvo de los profesionales que examinaron la herramienta fue una clara evidencia de este logro.

Finalmente, en lo que concierne al tercer objetivo de integrar el sistema de cinemáticas en un videojuego profesional en desarrollo, la herramienta se insertó de manera fluida y efectiva.

Mediante pruebas iterativas y un constante perfeccionamiento, se aseguró que la herramienta se adaptara y operara a la perfección dentro del entorno del juego.

7.3. Trabajos Futuros

El cierre de un proyecto marca un punto de referencia importante, pero no necesariamente el final del camino. A menudo abre nuevas posibilidades y horizontes para explorar. A lo largo del desarrollo de la herramienta se identificaron áreas potenciales para expansión, refinamiento y mejora. El *feedback* recopilado, así como las reflexiones surgidas del proceso, proporcionaron información valiosa sobre aspectos que pueden ser abordados y mejorados en futuras investigaciones o implementaciones.

7.3.1. Limitaciones y Posibles Mejoras

Un desafío identificado en el desarrollo fue encontrar un equilibrio entre la versatilidad y la facilidad de uso de la herramienta. A pesar de que fue diseñada para ser intuitiva, existe margen para mejorar la interfaz, permitiendo una adaptabilidad aún mayor a desarrolladores con diferentes niveles de experiencia en el ámbito de videojuegos.

Aunque la herramienta fue específicamente diseñada con una perspectiva centrada en Unity, se considera que expandirla para ser compatible con otros motores de juego sería una mejora considerable. Esto no solo podría ampliar el rango de usuarios, sino también incrementar su relevancia y utilidad en la industria del desarrollo de videojuegos.

Una mejora potencial sería facilitar la implementación de la herramienta en diversos juegos, especialmente aquellos que ya están en fases avanzadas de desarrollo. Hacer que el proceso de integración sea más fluido y menos laborioso podría ser de gran valor para muchos equipos de desarrollo.

Adicionalmente, el *feedback* recopilado señala la posibilidad de incorporar funcionalidades que no fueron contempladas inicialmente. Estas podrían abarcar características avanzadas, como la integración de inteligencia artificial para la adaptación dinámica de cinemáticas en función de las decisiones del jugador o eventos en tiempo real.

8. Bibliografía

¿Qué es una API y cómo funciona? (2023, January 20). Retrieved July 16, 2023, from https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

2022 Gartner® Hype CycleTM for Digital Identity. (2022, August 22). Callsign. https://www.callsign.com/knowledge-insights/gartner-hype-cycle-for-digital-identity

A model based on the Technology Readiness Level (TRL) scale to measure the maturity level of research projects that can become spinoffs in Higher Education Institutions. (2021, September 29). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9619667

Adiguzel, Z. (2020). Strategic Approach to Business Intelligence and Its Impacts on Organizational Performance. https://www.semanticscholar.org/paper/Strategic-Approach-to-Business-Intelligence-and-Its-Adiguzel/c68c33635e6eaec156246182b63b19c5557c9b4b

Advanced Animation — CC/iC Unity Tools 1.3.0 documentation. (2021). Retrieved July 26, 2023, from https://soupday.github.io/cc_unity_tools/adv-usage.html

Ampoloquio, R. (2023, February 15). Grand Theft Auto 6 might be using updated Red Dead Redemption 2 engine. Xfire. https://www.xfire.com/grand-theft-auto-6-updated-red-dead-redemption-2-engine/

Archaeogaming. (2017, September 27). The Archaeology of Video Game Complexity. Archaeogaming. https://archaeogaming.com/2017/09/27/the-archaeology-of-video-game-complexity/

Aziz, K. A. (2023, July 20). Smooth Navigation, Captivating Experience: A UI/UX Design Guide for Effective Layout and Navigation. Medium. https://bootcamp.uxdesign.cc/smooth-navigation-captivating-experience-a-ui-ux-design-guide-for-effective-layout-and-navigation-d7e20386f73a

Balson, F. (2018). The past, present and future of the video game cinematic: a look at the evolution of the video game cinematic and its impact on the industry. https://www.semanticscholar.org/paper/The-past%2C-present-and-future-of-the-video-game-a-at-Balson-Aitken/82c08d0f537d0d621d0b8ecfe312a8b5f6708409

Benny. (2022, January 4). How to fetch data from API in Unity - Bina Nusantara IT Division - Medium. Medium. https://medium.com/bina-nusantara-it-division/how-to-fetch-data-from-api-in-unity-99e58820b2d4

Best Game Development Software in 2023. (2023). https://www.nuclino.com/solutions/game-development-software

Chapurlat, V. (2012). System interoperability: definition and proposition of interface model in MBSE Context. https://www.semanticscholar.org/paper/System-interoperability%3A-definition-and-proposition-Chapurlat-Daclin/fcfce1e120a7b0846300a6d942f9ceff23987423

Contributor, A. (2022, December 12). Game Dev in Movies: Video game Technology meets Cinematography. Academy of Art University. Retrieved July 17, 2023, from https://blog.academyart.edu/game-development-meets-filmmaking-cinematography-in-video-games/

Controlling Anim Instances with Sequencer. (n.d.). Unreal Engine. https://docs.unrealengine.com/4.26/en-US/AnimatingObjects/Sequencer/HowTo/ControlAnimInstances/

Cutscene (Concept) - Giant Bomb. (2021, October 10). Giant Bomb. Retrieved June 7, 2023, from https://www.giantbomb.com/cutscene/3015-22/

Degen, H. (2011). UX Best Practices How to Achieve More Impact with User Experience. https://www.semanticscholar.org/paper/UX-Best-Practices-How-to-Achieve-More-Impact-with-Degen-Yuan/c9f7089fd6bac5b65464737e80a1c5d7fa90dc9a

Dinu, A. (2018). International Market Entry Strategies. https://www.semanticscholar.org/paper/International-Market-Entry-Strategies-Dinu/c02b1f1be46ac177fb4ebe147cb99e74e9144a41

Djatmiko, J. (2020). UI design development for informative mobile game about light pollution. https://www.semanticscholar.org/paper/UI-design-development-for-informative-mobile-game-Djatmiko-Pinasthika/77e89f305d188124d5a9453d2f63c5816ba07458

Dogruel, L. (2013). Video game rating systems in the US and Europe. https://www.semanticscholar.org/paper/Video-game-rating-systems-in-the-US-and-Europe-Dogruel-Joeckel/07dfa2a75ca35e2d0ae812c5d6573cd34eca36ca

Dyer, M. (2013, May 31). Naughty Dog Explains Its Next-Gen Game Engine - IGN. IGN. https://www.ign.com/articles/2013/05/31/naughty-dog-explains-its-next-gen-game-engine

Features List. (n.d.). Articy. https://www.articy.com/en/articydraft/feature-list/

Ferreira, J. (2007). Agile Development Iterations and UI Design. https://www.semanticscholar.org/paper/Agile-Development-Iterations-and-UI-Design-Ferreira-Noble/503696f22d5a72bcd6f353d64e042f17f60d5c1e

French, J. (2022). PlayMaker vs Unity Visual Scripting (Bolt). Game Dev Beginner. https://gamedevbeginner.com/playmaker-vs-unity-visual-scripting-bolt/

French, J. (2022). Interfaces in Unity (how and when to use them). Game Dev Beginner. https://gamedevbeginner.com/interfaces-in-unity/

GameForest. (2019, March 16). Evolution of RAGE Engine Games 2006-2018 [Video]. YouTube. https://www.youtube.com/watch?v=fruSKeMKNY0

Gaming Trends 2023 - Top 10 Trends that Will Rule Industry. (2021, October 4). EDIIIE. Retrieved July 23, 2023, from https://www.ediiie.com/blog/gaming-trends/

General Data Protection Regulation (GDPR) – Official Legal Text. (2022, September 27). General Data Protection Regulation (GDPR). https://gdpr-info.eu/

Ghosh, A. (2013, August 21). Compatibility of Hardware and Software. The Customize Windows. https://thecustomizewindows.com/2013/08/compatibility-of-hardware-and-software/

Hancock, H. (2002, April 2). Better Game Design Through Cutscenes. Game Developer. https://www.gamedeveloper.com/design/better-game-design-through-cutscenes

Howarth, J. (2023, June 13). 7 Huge Gaming Industry Trends 2023-2026. Exploding Topics. https://explodingtopics.com/blog/gaming-trends

Indeed Editorial Team. (2023). Indirect vs. Direct Competitor: What's the Difference? Indeed.com. https://www.indeed.com/career-advice/career-development/indirect-competitor-vs-direct

Innova - Ciencia Asturias. (2022). Ciencia Asturias. Retrieved May 16, 2023, from https://ciencia.asturias.es/innovaci%C3%B3n

Jedruszczak, B. (2016). Identity Creation and World-Building through discourse in video game narratives. https://www.semanticscholar.org/paper/Identity-Creation-and-World-Building-Through-in-Jedruszczak/04bb0aab5447b9d74ba5a7452f4bb6df380a3c11

Jenkins, B. (2021, September 17). How Video Is Failing To Enhance And Promote Games. Forbes. https://www.forbes.com/sites/forbestechcouncil/2021/09/17/how-video-is-failing-to-enhance-and-promote-games/?sh=24a1b0966b7b

Jiovaine. (2018, May 4). How to import Maya animations into Unity - Jiovaine - Medium. Medium. https://medium.com/@jiovaine/how-to-import-maya-animations-into-unity-7cb0450efd98

Johl, J., & Johl, J. (2023, January 12). What are APIs and how do APIs work? MuleSoft Blog. https://blogs.mulesoft.com/learn-apis/api-led-connectivity/what-are-apis-how-do-apis-work/

Kasam, M. (2022). SQL vs NoSQL - ¿Cómo decidir cuál y cuando utilizar? Nisum. https://www.nisum.com/es/nisum-knows/sql-vs-nosql-how-you-decide-which-to-use-and-when

Khan, W. (2017). Predictive Performance Comparison Analysis of Relational & NoSQL Graph Databases. https://www.semanticscholar.org/paper/Predictive-Performance-Comparison-Analysis-of-%26-Khan-Ahmed/325bbe2c92072b16645a5cded3b9eda3cb97660b

Kaur, H. (2013). A Review Of Non Relational Databases, Their Types, Advantages And Disadvantages. https://www.semanticscholar.org/paper/A-Review-Of-Non-Relational-Databases%2C-Their-Types%2C-Kaur-Kaur/1118508a5757817d0b0ad1334691cbd0cbb2a966

Kristiel. (2016, December 28). Unity Tutorial: How to Use Animator Controllers and Triggers - Studica Blog. Studica Blog. https://blog.studica.com/unity-tutorial-animator-controllers

Krzyścin, K. (2019). The Witcher 3 Interview: A Deeper Look into RedEngine 3. 80.lv. https://80.lv/articles/the-witcher-3-interview-a-deeper-look-into-redengine-3/

Liao, S. (2023, May 22). Blizzard Trains Image Generator as A.I. Pervades Video Game Design. The New York Times. https://www.nytimes.com/2023/05/22/arts/blizzard-diffusion-ai-video-games.html

Looper, W. (2023). Benefits of using game development tools in the gaming industry. www.linkedin.com. https://www.linkedin.com/pulse/benefits-using-game-development-tools-gaming-industry-work-looper/

Martinez, K. (2022). Workflow automation: What it is, why it matters, and how you can use it. zapier.com. https://zapier.com/blog/workflow-automation/

Mizuguchi, M. (2001). Data driven motion transitions for interactive games. https://www.semanticscholar.org/paper/Data-driven-motion-transitions-for-interactive-Mizuguchi-Buchanan/f94da5331afe03bddb41473a382dd4244fe85ceb

Motiso, D. (2022). What Is a Scripting Language? (With Types and Advantages). Indeed.com. https://www.indeed.com/career-advice/career-development/what-is-scripting-language

Nelson, X., Jr. (2019). How developers create cinematics. Pcgamer. https://www.pcgamer.com/how-developers-create-cinematics/

Newzoo. (2023, August 5). The Games Market in 2022: The Year in Numbers | Newzoo. https://newzoo.com/resources/blog/the-games-market-in-2022-the-year-in-numbers

Nicolaou, K. (2019, December 19). The Last of Us: Storytelling Methods Across Mediums - UBVIA. UBVIA. https://ubvia.com/the-last-of-us-storytelling-methods-across-mediums/

Nonlinear Storytelling. (2009, July 30). Game Design Concepts. https://gamedesignconcepts.wordpress.com/2009/07/30/level-10-nonlinear-storytelling/

Paur, J. (2019). Glorious Opening Cinematic for THE WITCHER 3: WILD HUNT — GeekTyrant. GeekTyrant. https://geektyrant.com/news/glorious-opening-cinematic-for-the-witcher-3-wild-hunt

Pluralsight and Unity Technologies Partner to Equip Game Developers with Skills to Create Games in Unity. (2017, November 2). Pluralsight. Retrieved June 26, 2023, from https://www.pluralsight.com/newsroom/press-releases/pluralsight_and_unity_technologies_partner_to_equip_game_developers_with_skills_to_create_games_in_unity

Pratt, M. K. (2023). Top 12 most commonly used IoT protocols and standards. IoT Agenda. https://www.techtarget.com/iotagenda/tip/Top-12-most-commonly-used-IoT-protocols-and-standards

Ronzio, R. (2022). How Does Data Power Modern Video Games? Pure Storage Blog. https://blog.purestorage.com/perspectives/how-does-data-power-modern-video-games/

Pre-Rendered graphics. (n.d.). TV Tropes. Retrieved June 13, 2023, from https://tvtropes.org/pmwiki/pmwiki.php/Main/PreRenderedGraphics

Shaver, M. (2020). Can you skip cutscenes in The Last of Us Part 2? AllGamers. https://ag.hyperxgaming.com/article/10383/can-you-skip-cutscenes-in-the-last-of-us-part-2

Shylenok, P. (2019, April 2). Effective Game Design: How We Use Articy Draft to Organize Game Content. Game Developer. https://www.gamedeveloper.com/design/effective-game-design-how-we-use-articy-draft-to-organize-game-content

Singh, S. (2022). Game Development using Unity Game Engine. https://www.semanticscholar.org/paper/Game-Development-using-Unity-Game-Engine-Singh-Kaur/b7160c7bd84c32668ea78d64394d35238685380a

Skrebels, J. (2018). Red Dead Redemption 2's Dialogue System Turns It Into a Comedy - IGN. IGN. https://www.ign.com/articles/2018/09/20/red-dead-redemption-2s-dialogue-system-turns-it-into-a-comedy

Sliding Scale of Gameplay and Story Integration. (n.d.). TV Tropes. https://tvtropes.org/pmwiki/pmwiki.php/Main/SlidingScaleOfGameplayAndStoryIntegration

Smith, T. (2017). How Important Is the Database in Game Development? dzone.com. https://dzone.com/articles/how-important-is-data-in-game-development

Söderhäll, A. (2022, January 25). Tracing the past, present, and future of game cinematics. GamesIndustry.biz. https://www.gamesindustry.biz/tracing-the-past-present-and-future-of-game-cinematics

Soto, J. (2023). Cómo trabajar el guion de un videojuego. Caja De Letras. https://cajadeletras.es/como-se-trabaja-el-guion-de-un-videojuego/

Stone, C. (2019, January 7). The evolution of video games as a storytelling medium, and the role of narrative in modern games. Game Developer. https://www.gamedeveloper.com/design/the-evolution-of-video-games-as-a-storytelling-medium-and-the-role-of-narrative-in-modern-games

Suderman, P. (2018, November 23). Opinion | Red Dead Redemption 2 Is True Art. The New York Times. https://www.nytimes.com/2018/11/23/opinion/sunday/red-dead-redemption-2-fallout-76-video-games.html

Talbert, L. (2021). Saving Game Data with Unity. Simple Talk. https://www.red-gate.com/simple-talk/development/dotnet-development/saving-game-data-with-unity/

Talent.com. (n.d.). Búsqueda de empleo en Talent.com | Encuentra vacantes disponibles cerca de ti. https://es.talent.com/es/

Technologies, U. (2023, July 21). Unity - Manual: Animator Controller. Retrieved May 11, 2023, from https://docs.unity3d.com/Manual/class-AnimatorController.html

Thielmann, P. T. (2017, February 15). Game Content Creation Tool „articy:draft" Adds Unity Integration with New Version 3.0. Newswire. https://www.newswire.com/news/game-content-creation-tool-articy-draft-adds-unity-integration-with-18798349

Thorne, B. (2022). How to design video games cinematics you won't want to skip. Creative Bloq. https://www.creativebloq.com/advice/how-to-design-video-games-cinematics-you-wont-want-to-skip

Turner, A. (2023, July 3). Video Game Industry Revenue & Market Share (Aug 2023). BankMyCell. https://www.bankmycell.com/blog/video-game-industry-revenue

Tzinis, I. (2021). Technology Readiness Level. NASA. https://www.nasa.gov/directorates/heo/scan/engineering/technology/technology_readiness_level

Vulkk. (2017). Get Behind the Scenes of the Cinematic Dialogues in The Witcher 3. VULKK.com. https://vulkk.com/2017/08/09/get-behind-scenes-cinematic-dialogues-witcher-3/

What's new in digital government from the 2022 Gartner hype cycle. (2022). Gartner. https://www.gartner.com/en/articles/what-s-new-in-digital-government-from-the-2022-gartner-hype-cycle

Wibbu. (2018, February 14). The Importance Of Narrative In Video Games - Edtech and Language Learning - Medium. Medium. https://medium.com/edtech-in-language-learning/the-importance-of-narrative-in-video-games-4a879fb01fe8

Wikipedia contributors. (2023). Cutscene. Wikipedia. https://en.wikipedia.org/wiki/Cutscene

Ziebarth, K. (2022). Using Databases to Improve the Efficiency of Computations for Sub-Optimal Pathfinding in Video Games. https://www.semanticscholar.org/paper/Using-Databases-to-Improve-the-Efficiency-of-for-in-Ziebarth/74f3948e9ed4c43794a85391ef18a294eb23935f

ANEXO I – Propuesta del Proyecto

Nombre alumno:	Jon Imaz Dravasa
Titulación:	Diseño y Desarrollo de Videojuegos
Curso académico:	2022-2023

1. TÍTULO DEL PROYECTO

Diseño y desarrollo de una herramienta modular para la elaboración de cinemáticas en Unity adaptada a un sistema de diálogos.

2. DESCRIPCIÓN Y JUSTIFICACIÓN DEL TEMA A TRATAR

En el mundo del desarrollo de videojuegos, el desarrollo e implementación de herramientas que agilicen labores mecánicas durante el desarrollo está más que a la orden del día.

El correcto desarrollo de estas es una compleja labor que combina Diseño, Interacción Humano-Máquina y Programación y permite que todos los departamentos puedan avanzar a un ritmo muy superior. Debido a esto, los desarrolladores de herramientas ocupan puestos de relevancia tantos en pequeñas empresas como en grandes triples A.

La herramienta que se plantea desarrollar en este trabajo de Fin de Grado consiste en un entorno visual para la elaboración de cinemáticas modulares y adaptadas a un sistema de diálogos, en un juego 3D.

3. OBJETIVOS DEL PROYECTO

- Familiarizarse con herramientas propias del desarrollo profesional de videojuegos en Unity.
- Desarrollo e implementación de un sistema nuevo de cinemáticas.
- Integración del mismo dentro de un videojuego profesional en desarrollo.

4. METODOLOGÍA

La metodología se fijará en las primeras reuniones con el tutor

5. PLANIFICACIÓN DE TAREAS

La planificación se especificará en las primeras reuniones con el tutor

6. OBSERVACIONES ADICIONALES

Este proyecto de Fin de Grado se enmarca en el ámbito industrial; el alumno trabajará estrechamente con la empresa Entalto Games SL.