Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial, de estudios de nivel superior, según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Maestría en Informática Aplicada

Metodología para el desarrollo independiente de videojuegos

TRABAJO RECEPCIONAL para obtener el **GRADO** de
MAESTRO EN INFORMÁTICA APLICADA

Presenta: **ALFONSO DE ALBA FERNÁNDEZ**

Asesora **NORMA ISABEL VILLANUEVA PAREDES**

Tlaquepaque, Jalisco, Abril de 2025.

Dedicatoria

Este trabajo se lo dedico a mi familia, mis padres: Alfonso y Ana Graciela del Carmen, por su ejemplo y los valores que me han dado.

A mi hermana Ana Graciela, por entendimiento y cariño.

A mi tutora, la maestra Norma Isabel Villanueva Paredes, por su paciencia durante este año y medio.

Abstracto

Con el objetivo de que los desarrolladores con recursos escasos puedan crear videojuegos de manera independiente, este documento ofrece una metodología específica para la industria mexicana. Al reconocer el auge mundial de la industria de los videojuegos y lo que representa para desarrolladores jóvenes, particularmente en naciones donde esta industria todavía no ha alcanzado su consolidación. Se examinan los desafíos específicos del desarrollo independiente, por ejemplo la ausencia de respaldo técnico y financiero, y se reconocen las ventajas competitivas de los desarrolladores locales, tales como la capacidad de adaptación, la disposición a asumir riesgos y las nuevas herramientas que permiten crear productos competitivos en el mercado.

La metodología propuesta está basada en el análisis comparativo entre dos enfoques: el diseño enfocado en el juego y la metodología de la tétrada. El método abarca desde la definición de visión y alcance, hasta el reconocimiento de habilidades y roles. Se enfatiza la importancia de la iteración, la validación mediante prototipos, la retroalimentación del usuario y tener un control estricto sobre el proyecto basándose en gastos y metas cumplidas.

Finalmente, se demuestra la viabilidad técnica y económica de la propuesta a través de un caso práctico, y se concluye que la metodología es suficiente para sentar las bases del desarrollo independiente de videojuegos en México.

Palabras clave

1. Videojuegos
2. Metodología
3. Prototipado
4. Emprendimiento
5. Unity

Índice

Índice de tablas

Índice de figuras

Resumen

Este documento propone y describe una metodología para el desarrollo independiente de videojuegos. El término “independiente”, surgió orgánicamente en la prensa especializada, para referirse a los pequeños desarrolladores independientes, de grandes estudios y publicadores. Lo cual; genera una excelente oportunidad de negocio para los jóvenes desarrolladores, en el campo del entretenimiento.

Para darle validez a esta metodología, se han estudiado otros procedimientos, usados en proyectos de forma exitosa. Cabe aclarar, que estos procedimientos fueron utilizados en productos no independientes, por lo que la contribución de esta nueva metodología, consiste en usar técnicas de la forma más económica y eficiente posible, usando los limitados recursos de un solo desarrollador multidisciplinario o un equipo pequeño (no mayor a 5 personas).

Esta metodología, contiene las técnicas para determinar, si un proyecto es viable económicamente; evaluar las herramientas más apropiadas para el proyecto, basándose en viabilidad técnica, económica y experiencia del desarrollador.

También se ha explorado la tétrada, como una de las metodologías y filosofías de diseño a nivel narrativo, técnico, estético y mecánico, junto con una aproximación centrada en juegos de forma iterativa.

Se ha concluido; que la metodología propuesta es apropiada para su objetivo establecido, sin embargo, dadas las muchas variaciones que pueden ocurrir, durante el desarrollo de un proyecto de desarrollo de software, siempre se recomienda mayores estudios para su refinamiento.

1. Introducción

Hoy en día, hay un mercado importante en el entretenimiento electrónico, es una industria millonaria, que desde hace tiempo ha superado a otras industrias destacadas del entretenimiento, tales como lo fueron: la música y las películas. Esto incluye desarrolladores independientes, es decir; desarrolladores que no tienen el apoyo de una compañía o publicista.

Independientemente de este crecimiento económico, en los últimos años no hay presencia de la industria en México y el desarrollo de videojuegos independientes, a pesar de su popularidad tiene una presencia nula.

La metodología aquí propuesta, se ha desarrollado para ayudar a determinar, cuál sería el producto viable, para un desarrollador o grupos pequeños de desarrolladores emergentes, junto con las herramientas necesarias, para llevar el producto a su conclusión. Permitirá a los jóvenes escoger las herramientas que sean más eficientes y llevar un control del proyecto, de manera formal, en vez de un ad hoc como seria usual en una situación, donde hay desarrolladores con poca o mínima experiencia.

Se espera que con la ayuda de esta metodología propuesta, haya más desarrolladores independientes que permitan a la industria, crecer y generar una entrada financiera personal y comunitaria, así como generar un trabajo remunerado, que les permita alcanzar un mejor nivel de vida.

2. Planteamiento del problema u oportunidad de negocio

Hoy en día la industria del entretenimiento, refiriendo a música, películas, libros, comics y videojuegos, este último el más relevante para este documento, es uno de los mercados más grandes en el mundo. Esta sobrepasa unas ganancias de más de 200 billones de dólares de forma global y se espera que continúe creciendo (Wijman, 2020). Sin embargo, en México, no hay ninguna casa de desarrollo de videojuegos, a pesar de que hay herramientas que permiten a desarrolladores independientes crear juegos de forma relativamente fácil y quienes son conocidos coloquialmente como “indie” (derivado de la frase inglesa “independent developer”), y se han logrado colocar como innovadores dentro del mercado.

Si bien, hay evidencia para determinar que un desarrollador independiente, puede implementar exitosamente un producto. En el 2011 Kerbal Space Program (Wikipedia, 2022) fue desarrollado por “Squad”, una empresa de mercadotecnia donde uno de sus empleados (Felipe Falanghe) tuvo la iniciativa de sugerir el desarrollo de un videojuego, como un proyecto de la empresa. En el 2013, este juego debuto en “Steam”, logrando mantenerse como uno de los 5 juegos mejor vendidos y obteniendo aclamación critica profesional y múltiples premios internacionales.

El desarrollo independiente, es como cualquier otro proyecto de emprendimiento, en el sentido que tiene los mismos problemas generales que todo emprendimiento debe afrontar, como son: la incertidumbre y la toma de riesgos; adicionalmente en este proyecto, se implican conocimientos técnicos especializados: (tecnologías de *renderizamiento* de gráficos, diseño de interfaces gráficas, codificación, etc.).

Particularmente en México, como nación en vías de desarrollo, el problema que existe, es que no cuenta en este momento, con una cantidad suficiente de profesionales que tengan el conocimiento técnico y la capacidad financiera para afrontar estos retos. La mayoría de los buenos

desarrolladores (Xantomila, 2020) en el país, eventualmente emigran a otros países (la fuga de cerebros), llevándose con ellos tanto su talento como el incentivo capital. Si bien estos factores son importantes, en México se tienen ciertas ventajas entre las cuales podemos destacar:

a) Una buena base para el emprendimiento
b) Desarrolladores jóvenes, interesados en entrar a la industria de videojuegos, que en su comienzo no tienen tantas necesidades económicas y pueden trabajar con un mínimo de recursos.
c) Desarrolladores nacionales, tienen la capacidad de adaptar nuevas tecnologías a sorprendente velocidad y a tomar mayores riesgos, que son demasiado inciertos para que sus competidores internacionales exploren y aprovechen.

Según la Asociación Mexicana de Videojuegos: (AMEXVID) (Videojuegos, 2017), en su reporte de la industria del 2017, Guadalajara y la Ciudad de México fueron los primeros clústeres tecnológicos de empresas relacionadas con los videojuegos.

Hoy en día, también hay varias organizaciones ofreciendo soporte, tales como: Altered Ventures, Fonca, Epic Megra Grants, The inde Fund, Conacyt, etc. Además, fuera de las organizaciones de apoyo a los emprendimientos, existen publicadores, capaces de brindar ayuda para sacar adelante el proyecto a nivel internacional, pero ninguno en este momento a nivel nacional (Videojuegos, 2020).

Finalmente, también es necesario mencionar, que hoy en día hay un método para fondear el juego de forma pública, sin tener que recurrir a las inversiones o casa de valores. En este caso, hablamos de las plataformas de "crowdfunding", tales como, Kickstarer, Fig, Play Business, Patreon, en las cuales, se promete al público una recompensa a cambio de su inversión. La recompensa puede incluir parte de los activos de la empresa, o el producto en sí mismo, una vez que éste sea terminado.

Dicho esto, actualmente hay un mercado para videojuegos independientes, creciendo todos los días, con productos que han probado ser exitosos, y más importante aún, herramientas que permiten a desarrolladores individuales o estudios pequeños desarrollar un juego de principio a fin.

3. Objetivos

3.1 Generales

Proponer una metodología, que permita establecer las bases para el desarrollo independiente de videojuegos, en un mercado en el cual, se requieren características de innovación y competitividad.

3.2 Particulares

1) Investigar y comparar, al menos dos de las metodologías existentes en el mercado para el desarrollo de videojuegos.
2) Explorar y determinar, las herramientas más apropiadas para este tipo de desarrollos.

3) Realizar el prototipo, usando la metodología propuesta, las herramientas y métricas sugeridas (Estimar el esfuerzo requerido en unidades de tiempo).

4.Justificación

Demostrar que es posible realizar un proyecto de desarrollo de videojuego mexicano, siguiendo una apuesta metodológica, que permita establecer los lineamientos que se deben seguir, en el desarrollo independiente de videojuegos, específicamente.

Si bien; las otras metodologías pueden ser usadas, nuestra hipótesis es que los desarrolladores independientes, usualmente desarrolladores en solitario o equipos pequeños, se verían aún más beneficiados, siguiendo está metodología, dirigida específicamente a sus necesidades, tomando en cuenta los problemas y retos únicos del desarrollo independiente, con relación al desarrollo formal por empresas más grandes.

5.Marco teórico

5.1Estado del arte

5.1.1 Áreas de conocimiento relacionadas.

El desarrollo de un videojuego requiere conocimientos en diferentes áreas:

- Desarrollo de software
- Diseño artístico
- Diseño narrativo
- UX/UI
- Mecánicas del juego
- Funciones del juego

Por otra parte, también es indispensable la administración, para dar un buen uso a los recursos disponibles y finalmente el “marketing” para asegurar que los productos llegan al mercado y clientes deseados.

En el trabajo de investigación se encontraron 2 metodologías: la metodología de la tétrada y la metodología de diseño centrado en el juego.

5.1.2 Metodologías.

5.1.2.1 Metodología de la tétrada.

Tétrada. La tétrada en capas: es una herramienta que ayuda a entender y a crear varios aspectos de un juego. Ésta, ayuda a analizar juegos de forma completa, como se puede apreciar en la siguiente imagen (Gibson, 2015).

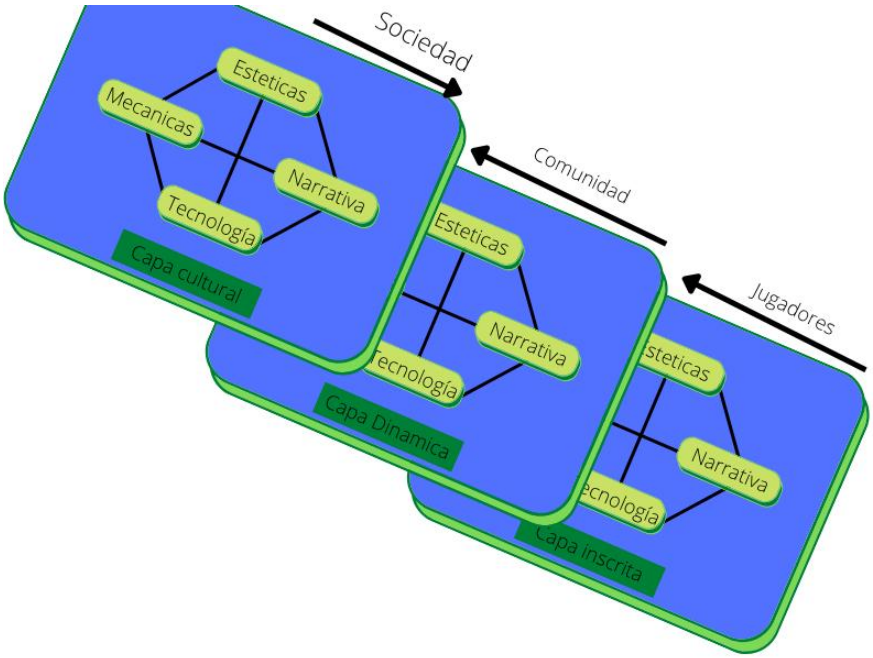

Figura 5.1.2.1 - Metodología de la tétrada en capas.

Capa Inscrita.

La primera capa, contiene todos los diagramas, ideas conceptuales, reglas, objetivos, límites y recursos. Esta capa incluye 4 componentes y cada uno se describe a continuación.

1. Mecánicas inscritas. Definen: Cómo los jugadores interactúan dentro del juego. Incluyen: Objetivos, reglas, recursos, límites y espacios.

- Objetivos: ¿Qué están tratando de lograr los jugadores?
  a. Inmediatez de objetivos
  b. Importancia de objetivos (principal vs opcional)
- Relación con el jugador: ¿Cómo, los objetivos de los jugadores se entrelazan y esto causa que los jugadores compitan o combatan?
  a. Único jugador vs juego,
  b. Múltiples individuos vs juego,
  c. Juego cooperativo,
  d. Jugadores vs jugadores,
  e. Competición multilateral,
  f. Competición unilateral,
  g. Competición por equipos

- Reglas: Indicaciones que especifican y limitan las acciones de los jugadores. Por ejemplo; cuántas vidas tiene un jugador, cuántas municiones tiene una pistola o cuántos objetos de curación tiene un personaje en específico
- Recursos: Los recursos y valores que son relevantes dentro de los campos del juego.
- Límites: Definir las fronteras del juego
- Espacios. Definen la forma del espacio del juego y las posibilidades para la interacción dentro del espacio y el tiempo, las reglas y otros aspectos que apliquen al juego. Se debe tomar en cuenta al diseñar los espacios:
    a. Flujo: Se debe permitir al jugador moverse fácilmente y si restringe el movimiento debe haber una buena razón para estas limitantes.
    b. Marcas de tierra: Puntos de referencia para que el jugador se pueda orientar
    c. Experiencias: El espacio debe tener eventos o actividades que mantengan al usuario entretenido

2. Estéticas inscritas. Describe, cómo es que el juego se ve y suena, incluyendo: Visión, sonido y metas estéticas.

- Visión
    a. Fidelidad de la experiencia visual
    b. Todo lo que los jugadores ven, debe ser relevante para el jugador
- Sonido
    a. Efectos de sonido: Alertan al jugador, muestran información simple
    b. Música: Crea el ambiente, en el cual el jugador se sumerge
    c. Dialogo: Entrega información compleja
- Metas estéticas:
    a. La estética, hace un gran trabajo, al poner el ambiente emocional de un juego
    b. Información: El jugador puede ser entrenado para comprender alguna información estética, que tenga un significado específico, por ejemplo, una barra de vida verde. También se puede tomar ventaja de los colores, que tienen un significado psicológico (rojo es peligro, amarillo es cuidado, azul es pacífico, etc.)

3. Narrativa inscrita. Incluye toda la historia prescrita y personajes generados en el juego: Premisa, Marco narrativo, Personajes y Trama.

- Premisa: La base sobre la que la historia emerge
- Marco narrativo: Expande sobre la premisa para proveer un mundo detallado, en el cual la narrativa tiene lugar
- Personajes: Las historias son sobre personajes y los personajes se componen de trasfondos y uno o más objetivos

- Trama: Secuencia de eventos que tiene lugar en la narrativa

Diferencias entre narrativa linear e interactiva:

- Trama vs libre albedrío: El autor no tiene control total sobre el progreso de la historia debido a la agencia del jugador
- Posibilidades limitadas: Estas son parte de toda narrativa lineal
- Permitir a los jugadores escoger misiones alternativas lineares: Existe la narrativa principal y narrativas alternas, que el jugador puede explorar simultáneamente
- Anticipar muchas cosas que pueden suceder y algunas que definitivamente van a suceder
- Desarrollar personajes menores, que no juegan un papel importante, si bien un personaje menor no es el foco de la narrativa, su existencia ayuda a enfocar los temas en los que se quieren enfocar.

Dramas inscritos:

- Evocar emoción: Enfocar la historia en manipular las emociones del jugador por efecto dramático
- Motivación y justificación: El drama, es usado para que el jugador tome ciertas acciones o justifique sus acciones
- Progresión y recompensa: Muchos juegos usan escenas cortas y otras narrativas inscritas, para ayudar al jugador a conocer la historia
- Mecanismos de refuerzo: Consiste en el uso de mecánicas, como refuerzo para dar información, que permita al usuario interactuar con los sistemas del juego.

4. Tecnología inscrita. Este elemento cubre toda la tecnología que hace funcionar el juego.

- Aleatorización: Es la forma más común de tecnología en juegos de papel. Sean cartas o dados, introducen un elemento de "*impredecibilidad"* al juego.
- Cuidar el estado: Puede ser cualquiera, como mantener el conteo de puntos en el juego
- Progresión: Se monitorea por tablas y gráficos, esto incluye información como la progresión de habilidades de los jugadores.

Capa Dinámica.

Estos elementos que se describirán a continuación, existen cuando el juego está siendo jugado. En esta capa, se describe la experiencia del jugador mientras se encuentra jugando y cómo los jugadores interactúan entre ellos. Por su naturaleza los desarrolladores tienen control limitado total sobre esta capa.

1. Mecánicas Dinámicas: Cómo es que los jugadores se involucran con las limitantes del sistema.

2. Estéticas Dinámicas: En general animaciones, ocurren cuando el jugador interactúa con el juego y cómo se representan las cosas.

3. Narrativa Dinámica: Historias que emergen durante el sistema basado en el juego y las reglas.

4. Tecnología Dinámica: El comportamiento de la tecnología, qué retroalimentación recibe el jugador y el cambio en el estatus del juego.

Capa Cultural.

Describe el juego más allá del acto de jugar. La comunidad de jugadores sobre la que el juego se mueve, responsabilidad social y productos derivados al respecto del juego. De igual manera que la capa anterior, los desarrolladores tienen un control mínimo sobre esta capa.

1. Mecánicas Culturales: Se representa como Mods (coloquialismo para modificaciones) del juego.

2. Estéticas Culturales: Se habla de arte hecho por los fanáticos y seguidores del producto. Remix, (una pieza del medio usualmente música al que se le agrega o remueve un aspecto de su estado original), cosplay (del inglés “costume play” es una actividad en la que los individiuos se disfrazan de personajes del producto en cuestión), etc.

3. Narrativas Culturales: Abarca los aspectos narrativos que los fans expanden sobre el juego, por ejemplo, creando historias literarias en el mismo marco narrativo (fan-fiction), creando personajes visualmente similares a los del juego, siguiendo el mismo estilo artístico (fancharacters), etc.

4. Tecnologías Culturales: Cubre el uso de la tecnología de juegos para propósitos independientes de los juegos, como por ejemplo los gráficos en tiempo real empezaron a usarse en la previsualización médica.

5.1.2.2 Metodología de diseño centrado en el juego

Se enfoca en involucrar al jugador en el proceso de diseño de principio a fin. Entre más pronto sea posible introducir a los jugadores al proyecto, mejor. Estos son los pasos básicos de la metodología en términos de desarrollo:

i) Definir las metas del jugador. No es la funcionalidad del juego, sino la descripción de las situaciones que se esperan que el jugador tenga. Definiciones estrictamente de alto nivel, por ejemplo: “Se espera que los jugadores se sientan interesados en las decisiones estratégicas que tomarían”.

ii) Probar y *prototipar* ideas. Idealmente cualquier diseño que se tenga debe ser probado tan pronto como sea posible, utilizando prototípicos físicos o incluso actuado.

iii) Iterar. Se tomarán los resultados obtenidos, se evaluarán buscando encontrar fallos y mejorar las funcionalidades del juego hasta que la experiencia del jugador sea la deseada.

Para llevar a cabo este proceso iterativo se pueden usar los siguientes métodos:

A. Lluvia de ideas
    a. Establecer las metas de la experiencia del jugador.
    b. Inventar los conceptos y mecánicas que pueden conseguir la meta de la experiencia del jugador.
    c. Limitar la lista a las mejores 3.
    d. Escribir una descripción corta de estas ideas (documento de concepto).
B. Prototipo físico
    a. Crear un prototipo usando pluma y papel.
    b. Demostrar la mecánica del juego que logre las metas de la experiencia de juego.
C. Presentación (opcional)
    a. Se hace para buscar fondos de financiamiento.
    b. Deberá incluir una demostración del arte.
    c. Se tomará la retroalimentación sobre las fuentes de fondos monetarios y trabajar para satisfacer sus requerimientos.
D. Prototipo software
    a. Se podrá empezar a crear un prototipo funcional para probar la *jugabilidad*.
    b. Continuar con el proceso de prototipado hasta que se alcancen las metas establecidas por los jugadores.
E. Documentación de diseño
    a. Mientras se sigan prototipando y trabajando en la *jugabilidad*, se deben de tomar notas para el producto final.
    b. Usar este conocimiento para crear el documento de diseño que detalle las funciones del juego.
F. Producción
    a. Trabajar con todos los miembros del equipo para asegurar que todos los aspectos del diseño son realizables y correctamente descritos en el documento de diseño.
    b. En cuanto se llegue a la versión final del documento de diseño; se debe mover a producción.

c. Es importante no perder de vista los procesos pilo céntricos durante producción, pruebas, *jugabilidad*, personajes.

G. Pruebas

a. En realidad, esta fase debe ocurrir al mismo tiempo que producción
b. Además de las pruebas normales de funcionalidad y contenido se deben agregar pruebas de *jugabilidad*.

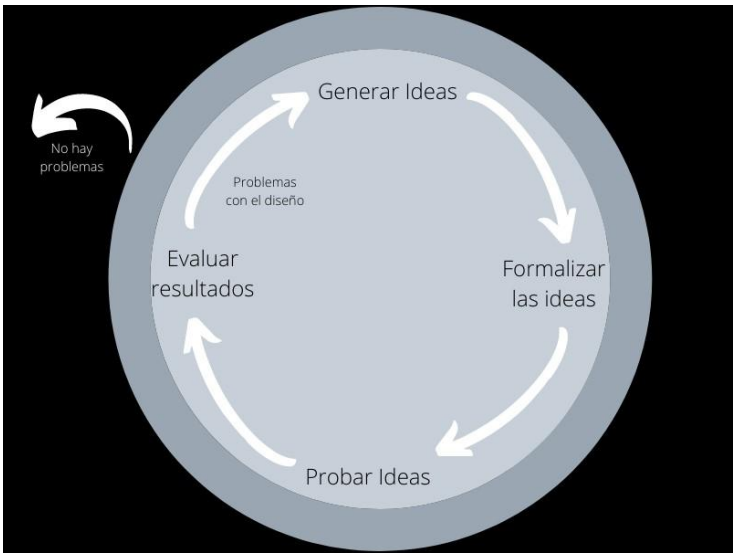

Figura 5.1.2.2 - Modelo de la innovación en la metodología centrada al juego

Diseñar para innovar, esto es importante, especialmente en el mercado del entretenimiento si no se quiere ser demasiado repetitivo; a riesgo de aburrir a los consumidores, por este motivo, hay que diseñar pensando en innovar, sin tomar demasiados riesgos en términos de tiempo y dinero. En términos de videojuegos por innovación entiéndase:

- Diseñar juegos con mecánicas de juego originales.
- Apelar a los nuevos jugadores
- Intentar resolver problemas difíciles en el diseño de juegos:
    - La integración de historia y *jugabilidad*,
    - Empatía por los personajes en el juego,
    - Crear un juego emocionalmente rico,
    - Descubrir las relaciones entre juegos y aprendizaje.

Tipos de prototipado

Hay varios tipos de prototipo, entre los cuales se destacan: físicos, visuales, de video, por software, etc.

Como punto de partida, no se esperan crear prototipos para generar ideas, sino para validar buenas ideas y refutar malas ideas.

Prototipos físicos.

Son los tipos de prototipados más fáciles de construir, por los diseñadores de juegos. Estos son, típicamente creados usando papeles, cartón y objetos caseros con dibujos hechos a mano.

Los beneficios de los prototipos físicos son muchos. Nos permite enfocarnos en la *jugabilidad* en vez de la tecnología, lo cual impide que los diseñadores estén atados a su código. El prototipado físico permite iterar más rápidamente, e iniciar la participación de los miembros menos técnicos en procesos de alto nivel (Fullerton, 2008).

> Si bien, *prototipar* un juego de mesa es relativamente sencillo, también es posible prototipar un juego lleno de acción. Por ejemplo, para un juego de disparos en primera persona, un prototipo físico, puede ayudar a entender los problemas tácticos y estratégicos sobre en balance de armas y control territorial; por otro lado, no puede ayudarnos a entender el proceso fluido de correr, apuntar y disparar en un ambiente 3D.

Por lo tanto, si bien un prototipo físico, siempre es valioso como una fuente de información, también se debe de tomar en cuenta; que la información no siempre puede estar completa.

Prototipos digitales

Como los prototipos físicos, los prototipos digitales son hechos únicamente con los elementos necesarios para que sean funcionales, enfocándose únicamente en las preguntas que se han de investigar las partes de diseño que requieren mayor claridad.

Se pueden clasificar los prototipos digitales en 4 tipos (Fullerton, 2008):

1. Tipo mecánicas de juego. Si ya se ha creado un prototipo físico, se puede estar un paso adelante en esta área del diseño, sin embargo; hay situaciones en las cuales una pregunta de *jugabilidad* no puede ser respondida por un prototipo físico.
   No se debe intentar integrar, todas las preguntas del juego en un solo prototipo o al menos no, en la primera interacción, siempre se debe empezar con la *jugabilidad* central.

2. Tipo estéticas. Las estéticas, son los elementos visuales y dramáticos del juego, que se han ignorado en los otros tipos de prototipo. Sin embargo; aquí es donde se rompe esta regla, añadiendo un poco del diseño visual y sonido al prototipo, para ayudar a articular las mecánicas del juego.

3. Tipo kinestésicas. Son el “sentimiento” del juego, como se sienten los controles, que tan responsiva es la interfaz, etc. Para probar kinestésicas, es necesario tener un prototipo digital, es importante tener en cuenta los controles que se tienen disponibles; para poder diseñar con ellos en mente.

4. Tipo tecnológico. Los prototipos tecnológicos, son modelos de todo lo que el software debe tener; para que el juego funcione de forma técnica. Se incluyen prototipos de capacidades gráficas para el juego, sistemas de inteligencia artificial, simulación física y cualquier otro problema específico del juego.

Es diferente del prototipo usado en ingeniería de software; en que es una oportunidad de intentar nuevas ideas, de forma rápida y sucia, los productos aquí mostrados no deben de ser integrados en el producto final. Por esto; también se recomienda trabajar en un lenguaje diferente del cual se usará en el producto final.

5.1.3 Análisis comparativo de metodologías de desarrollo de videojuegos

En el trabajo de investigación, se encontraron estudios relevantes sobre las metodologías de videojuegos, siendo las más recientes las presentadas anteriormente, primero: la tétrada elemental que es propuesta por Jesse Schell, un profesor en tecnología del entretenimiento, para la Universidad Carnegie Mellon, que ha sido ampliamente estudiada y citada por otros desarrolladores y profesores ((Gibson, 2015), (Cleveland Institute of Art, 2015), (Lu, 2013)). Esta tétrada es importante, para el desarrollo independiente, porque permite dar una estructura y un enfoque a la visión del juego, en secciones interrelacionadas, lo que determina sobre cuáles es necesario enfocarse y administrar los recursos del desarrollo.

Por otro lado, se ha adaptado la metodología de diseño centrada en el juego, en particular la fase de prototipo. Esto se debe a que, a pesar de que no es tan conocida como la tétrada elemental, se adapta bien, al ciclo de vida del prototipado lo que ayuda a determinar el producto mínimo viable para el sector de mercado que se está apuntando.

Para compararlas se usarán las siguientes características.

- Cantidad de personas. El número mínimo de personas, dentro del equipo que se requiere para poder ejecutar la metodología.
- Tiempo invertido: El tiempo que se debe invertir en la metodología durante la fase de diseño.
- Conocimiento especializado: El conocimiento mínimo necesario y suficiente para producir un diseño, que cumpla con los requerimientos esperados.
- Mecanismos de retroalimentación: Se definen los mecanismos de retroalimentación que permitan probar el diseño.

Características	**Metodología de diseño de la tétrada**	**Metodología de diseño centrado en el juego**
Cantidad de personas.	Mínima 1 persona. Es posible ejecutar la metodología con 1 persona o más.	Debido a la alta cantidad de prototipos requeridos y validación, se requiere un equipo mínimo 3. Una persona que genere el prototipo y 2 (o más) que lo validen.

Tiempo invertido.	Dada que la visión es definida al inicio de la metodología, el tiempo invertido en el diseño es menor.	La metodología, requiere diversas iteraciones y cada iteración requiere tiempo. Para cada iteración se deben estimar tiempos para: • Diseño de prototipo. • Pruebas.
		• Retroalimentación.
Conocimiento especializado.	Se requiere tener una visión muy especializada en el campo. Dado que no hay mecanismos de retroalimentación, en comparación de la otra metodología y por ende un conocimiento más profundo del campo.	No se requiere tener un conocimiento profundo del campo. Si bien, siempre es conveniente tenerlo, no es requerido.
Mecanismos de retroalimentación.	En la metodología de la tétrada, no hay mecanismos de retroalimentación.	Está metodología, propone varios prototipos como mecanismos de prueba y retroalimentación. Cada iteración debe tener un tiempo determinado.

Tabla 5.1.3 - Tabla comparativa de las metodologías investigadas

Como resultado, se observa que la tétrada es superior para los desarrolladores con menos recursos (tiempo y recursos humanos, específicamente), pero demanda mayor conocimiento del campo. Contrariamente a la metodología centrada en juegos, la cual exige menor conocimiento especializado, a cambio de mayores recursos. Tal y como es presentado por (Fullerton, 2008).

5.1.4 Análisis de las herramientas de desarrollo

En la evaluación de motores de juego, se han establecido 5 características a tomar en cuenta, a la hora de elegirlos y/o comprarlos:

- Monetización: ¿Qué tan cara es la herramienta y bajo qué términos financieros se permite publicar?, un juego desarrollado con estas herramientas.
- Alcance de plataformas: Este punto es muy importante, por qué determina ¿en que hardware puede funcionar el juego? y por tanto, el alcance de mercado del producto.
- Activos de soporte: Importante para desarrolladores independientes, que no pueden hacer todo el trabajo por sí mismos. Si la plataforma cuenta con una tienda de los activos, se puede

subcontratar a un desarrollador creativo o bien crear un juego usando activos genéricos, que liberan tiempo de desarrollo.

- Lenguajes: Definitivamente es muy importante como puente de entrada, a la herramienta para los desarrolladores. Determina: si se requiere invertir más tiempo aprendiendo un nuevo lenguaje de programación.
- Documentación de soporte: Determinar si la herramienta cuenta con la documentación para una fácil adaptación y consulta, durante el proceso de desarrollo si se requiere.

5.1.5 Descripción de las herramientas de desarrollo.

En términos de importancia de las herramientas, la elección del motor de juego (o game engine) que se propone utilizar, es fundamental y a la vez increíblemente difícil de realizar. Primero se debe definir un motor de juego, como la parte central, para que el programa de juego se ejecute adecuadamente, dado su importancia, como una parte central del proyecto, es importante que todas nuestras decisiones consecuentes, sean compatibles con el motor de juego.

1.- Motor de juego. Colección de herramientas dirigidas específicamente para el diseño de videojuegos.

2.- IDE. Ambiente de desarrollo y codificación del juego.

3.- Servicios de red. Esta herramienta es opcional, pero si el juego tiene una funcionalidad en línea, se deben definir las herramientas que le permitirán operar de tal manera.

4.- Generador de gráficos de juego. Una herramienta, que genere los gráficos utilizados para representar los objetos dentro del juego, usando la estética escogida previamente.

5.- Generador de música y sonido. Una herramienta que genere la música y sonidos utilizados dentro del juego, usando la estética escogida previamente.

6. Estrategia metodológica.

6.1 Sustento teórico acerca de la metodología seleccionada

En esta sección: se describirá la metodología propuesta. En términos generales podemos descomponer el tema de 2 secciones:

- Preparación para el desarrollo: Durante esta fase se buscará definir la visión del proyecto. Se requiere definir el alcance, ¿lo que se quiere lograr?, ¿las herramientas y habilidades que serán requeridas para llevar este proyecto a su término?. Durante el plan de desarrollo, se definirán los tiempos que se esperan.
- Diseño, desarrollo implementación y pruebas: Durante esta fase; se deben definir las especificaciones y requerimientos. Si bien, ya se ha definido lo que se quiere hacer durante la fase anterior, ahora; se deben definir las especificaciones y diseñar la solución. Esta es la fase durante la cual se codificará el prototipo, el cual pasará por pruebas (esto se hará por

varias iteraciones), dependiendo de lo que el tiempo y los recursos permitan, una vez que se tiene un prototipo candidato a liberar, el prototipo debe ser validado.

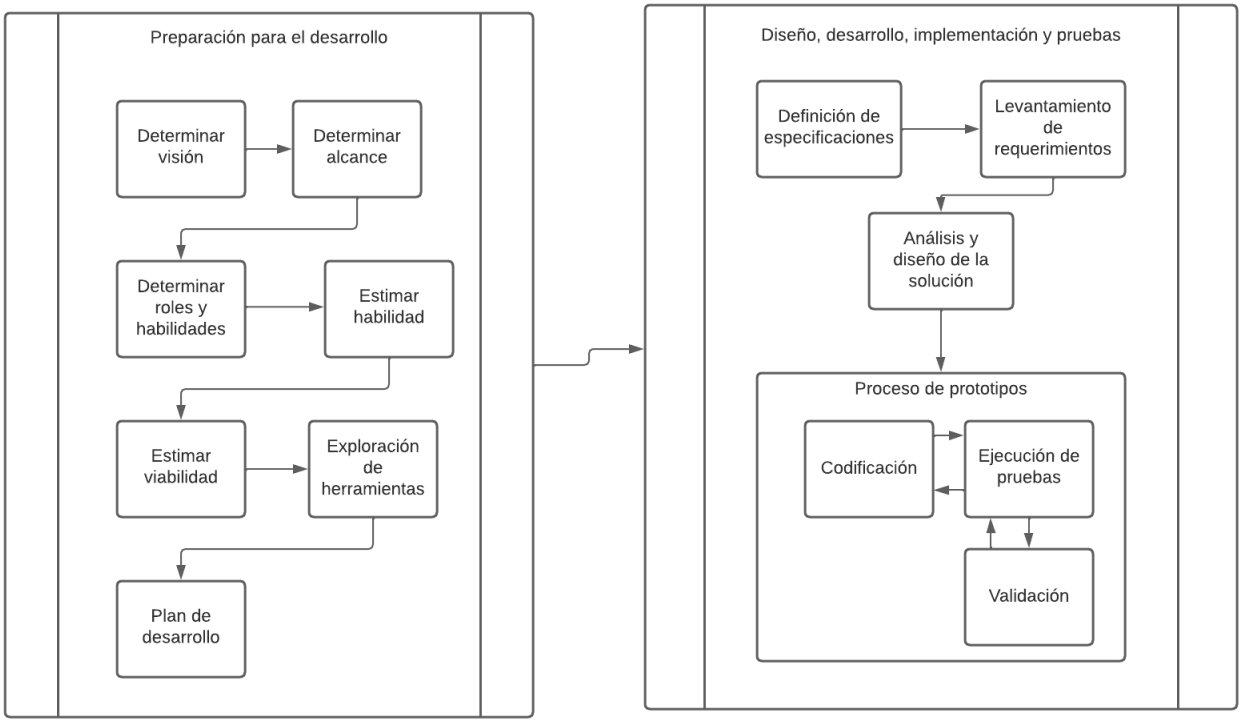

Figura 6.1 - Esquema metodológico para el desarrollo de videojuegos.

6.2 Descripción de la metodología a utilizar.

6.2.1 Preparación para el desarrollo.

6.2.1.1 Determinar visión

Para este punto, la herramienta de la tétrada, sirve muy bien para una visión general de lo que se quiere crear.

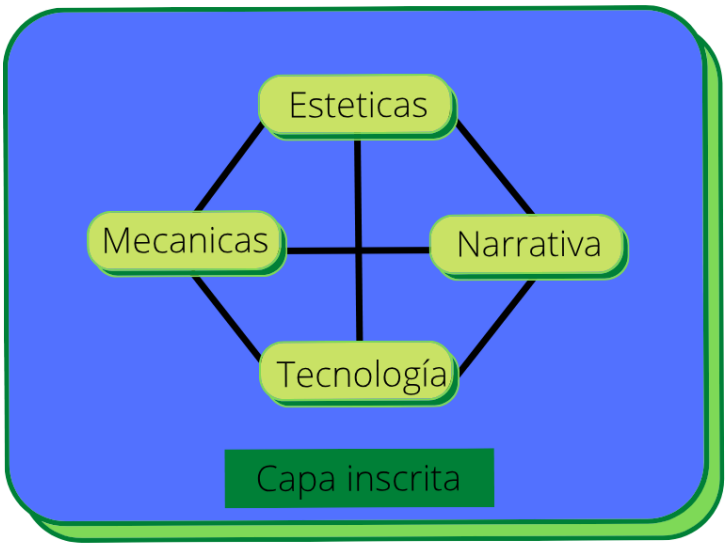

Figura 6.2.1.1 - Capa inscrita de la tétrada

La creación del videojuego tomará como base los 4 aspectos de la tétrada:

Estéticas. Cómo se verá el juego.

Mecánicas. Definir las mecánicas con las que el jugador va a interactuar.

Narrativa. La historia dentro del juego.

Tecnología. Las herramientas tecnológicas para usar.

6.2.1.2 Determinar alcance.

En este paso se definirá el objetivo y las metas que se desean alcanzar, así como las restricciones.

El alcance del proyecto debe cumplir los siguientes criterios:

- Específico.
- Sin contradicciones
- Medible.
- Realista.
- Factible.
- Determinado en el tiempo.

Y se definirán las siguientes tareas como mínimo.

- Breve descripción del proyecto.
- Descripción del objetivo a lograr.
- Entregables. Objetivos específicos del proyecto.
- Características de los entregables. Resultados esperados.

- Criterios de aceptación. Condiciones que indican que los productos o servicios están listos para ser entregados.
- Priorizar entregables: Críticos, opcionales y agregados a futuro.
- Restricciones. Elementos que limiten el alcance (económicos, sociales, temporales).
- Exclusiones. Resultados que el plan no va a lograr o lo que no se va a producir.
- Coste del proyecto si aplica. Determinar el presupuesto del juego.

6.2.1.3 Determinar roles y habilidades

Dada la naturaleza de la metodología, que considera que el equipo sea pequeño y de recursos limitados, es de esperar que una persona represente uno o más roles. Para proyectos más grandes podría haber personas especializadas en cada rol.

- Nombre de rol: Titulo del rol que se debe cumplir. Debe de describir brevemente lo que se espera del rol.
- Brecha de habilidad por rol: Habilidades que serán requeridas, pero que la persona encargada para el rol no es todavía proficiente.
- Actividades por rol: Las actividades que se espera que la persona cubriendo el rol sea capaz de ejecutar.

6.2.1.4 Estimar habilidad.

Nombre de rol	**Brecha de habilidad por rol**	**Actividades por rol.**
Programador.		
Diseñador gráfico.		
Compositor.		
Diseñador UI.		
Diseñador del juego.		
Escritor.		
Ingeniero QA.		
Administrador de proyecto.		

Tabla 6.2.1.4 - habilidades requeridas y actividades por rol

6.2.1.5 Estimar viabilidad

Dependiendo de la brecha de habilidad y los roles es posible que se determine que el proyecto no es viable. Para determinar, si el proyecto es viable, se debe hacer un análisis del esfuerzo respecto al coste. Estimar, ¿Cuánto tiempo de desarrollo se requiere?, contra ¿Cuántas copias del producto se requerirían vender? para determinar si el proyecto generará o no, un retorno de inversión (ROI).

- Período: Ciclo financiero relevante para el cálculo requerido.
- Flujo: Circulación de efectivo para la organización. En términos generales entrada y salida de dinero en un periodo específico.

- Valor Presente: Es el valor presente durante el periodo del flujo.
- Valor Presente Acumulado: Es el valor obtenido hasta ese periodo, en particular incluyendo inversión y gastos de períodos anteriores.
- Inversión: Es el desembolso financiero que se requiere para arrancar el proyecto.
- Gastos: Coste o pérdida de carácter financiero, requerido para mantener la operatividad de la organización.
- Cantidad de productos que se espera vender: Cantidad de productos que se esperan vender para generar un retorno de inversión.
- Cantidad de dinero que se ha ganado: Cantidad de dinero proyectado por nuestras ventas.
- Costos: Son necesarios para obtener ingresos y permiten a la organización mantenerse activa.
- Retorno de la inversión: Valor que nos permitirá indicar la rentabilidad del proyecto.
- Coste del dinero actual: Incluye todo coste de dinero que sea relevante (i.e. inflación).

Periodo	**Flujo**	**Valor presente**	**Valor presente acumulado**	**Inversión**	**Gastos**	**Ventas**		**Costos**
						Cantidad de productos que se han vendido	Cantidad de dinero que se ha ganado	Costos que ocurren durante el proyecto

Tabla 6.2.1.5 - Herramienta de estimación financiera

6.2.1.6 Exploración de herramientas.

Este paso, conlleva un trabajo exploratorio, respecto a las herramientas que se requieran usar. Una vez; que se tengan todas las herramientas seleccionadas, lo conveniente sería establecer una jerarquía y agrupación de herramientas de acuerdo con las siguientes categorías:

1. Motor de juego.
2. Entorno de desarrollo integrado (IDE).
3. Servicios de red (opcional).
4. Gráficos de juego.
5. Generador de música y sonido.

6.2.1.6.1 Consideraciones para la selección de las herramientas basado en un esquema de ponderación.

Al seleccionar una herramienta en general, se deben tomar en consideración los siguientes parámetros en orden de importancia:

1. Cumplir objetivo. La herramienta correcta para el trabajo correcto. La herramienta debe ser capaz de llevar a cabo lo esperado, para no comprometer la visión inicial. Este es un parámetro binario (0 o 1), 0 si no cumple, 1 si cumple.

2. Experiencia de los desarrolladores con la herramienta. Se debe preguntar a los desarrolladores cuáles herramientas les permitirán trabajar de forma más eficiente. Habrá que darle valores a este parámetro en 3 niveles: sin experiencia, con experiencia, maestría respectivamente (0, 0.5, 1).

3. Costo comparativo de la herramienta. Se debe tomar en cuenta, el costo de la herramienta dentro de la misma categoría y si tendrá que ser comprada o rentada. Se podrá dar un valor de 0 a N en estas herramientas: fuera de presupuesto es 0, gratuita 1 y el resto ponderadas en una escala de valores de 0.1, 0.2 hasta 0.9.

4. Compatibilidad entre herramientas. El producto de las herramientas, debe ser compatible con las demás herramientas, que se usarán en otras categorías. Similar al punto anterior este parámetro debe ser evaluado en una escala de 0 a 1, donde 0 es no compatible, 1 compatible con todas las otras herramientas y el resto con valores mayores a 0 y menores a 1, según el grado de compatibilidad entre ellas. Si solo hay una herramienta entonces el valor es siempre 1.

5. Soporte. En este punto; se deberá considerar si la herramienta cuenta con el soporte necesario. Esto puede incluir, venta de extensiones que faciliten el trabajo, tutoriales, foros de soporte, actualizaciones para la última versión, etc. (0, 0.1, 0.2, ..1) El valor 0 representa sin soporte, el valor 1 soporte suficiente y los valores intermedios soporte regular.

6. Resultado. Sumatoria de todos los demás puntos que ya se han contemplado.

Nombre de la herramienta	**Cumplir objetivo**	**Experiencia de los desarrolladores con la herramienta**	**Costo comparativo de la herramienta**	**Compatibilidad entre herramientas**	**Soporte**	**Resultado**

Tabla 6.2.1.6 - Comparativa de herramientas de desarrollo

Al final de la evaluación la mejor herramienta tendrá el mayor puntaje en el rubro de resultado y es la que deberá usarse para el proyecto.

6.2.1.7 Explicar acerca de cómo se realiza el plan de desarrollo.

- Exploración de herramientas.
    - Se tomará en cuenta la visión y el alcance para escoger la mejor herramienta para el desarrollo.
    - La elección del motor de videojuego, es la elección primaria y es la columna vertebral del desarrollo.
    - Las herramientas secundarias, que permitirán generar los recursos que sean requeridos (modelos, sonido, música, etc.)
- Preparación de entorno.
    - Instalación.
    - Configuración.
    - Conexión con terceros.
- Selección y obtención recursos.
    - Desarrollar los recursos dentro del equipo de desarrollo, incluyéndolo como una actividad más dentro del desarrollo.
    - En caso de no desarrollar los recursos dentro del equipo de desarrollo, entonces se requiere determinar; cómo se conseguirán los recursos: Se comprarán: los activos en una tienda abierta al público en general o se comprarán los activos a un desarrollador independiente del proyecto.

- Definir el plan de actividades calendarizadas.
    - Establecer tareas, hitos y fechas.
- Creación de plan de pruebas.
    - Pruebas funcionales.
    - Pruebas de rendimiento.
    - Pruebas unitarias.
    - Pruebas de integración.
    - Pruebas de escalabilidad.

6.2.2 Diseño, desarrollo, implementación y pruebas

6.2.2.1 Definición de especificaciones

Durante esta fase, se debe especificar el cómo se llevará a cabo el videojuego y para ello es importante partir de la visión establecida en la tétrada, lo cual implica definir:

- Qué estéticas se aplicarán a los personajes y la interfaz de usuario.
- Qué funciones se implementarán para el jugador.
- Qué estructura tendrá la narrativa.
- Qué herramientas tecnológicas se usarán.

6.2.2.2 Levantamiento de Requerimientos

Existen varias técnicas para hacer el levantamiento de requerimientos, tales como: encuestas al público en general o entrevistas a grupos de enfoque, por segmento de mercado identificado, entrevistas o investigaciones de observación.

6.2.2.2.1 Diseño de herramientas de diagnóstico

Este enfoque corresponde a la segmentación de mercado (Research Optimus, s.f.) y a la investigación de mercado primaria. Actualmente, las herramientas disponibles para este propósito son las que se describen a continuación:

- Encuestas: Son usadas ampliamente en la investigación de mercados y son excelentes herramientas porque no requieren mucha inversión.
- Grupo de enfoque: Si bien; un grupo de enfoque es altamente eficiente, se necesita un grupo estadísticamente válido, para poder obtener información relevante, esto lo hace un pobre candidato, para un equipo de desarrollo independiente.
- Entrevista: Es similar a la encuesta, que podría dar datos con mayor profundidad, pero debido a que requiere atención personalizada, puede ser difícil que el grupo de desarrollo independiente, pueda llevarlo a cabo dadas las limitaciones observadas.
- Investigación observación: La investigación de observaciones se refiere: a la obtención y el análisis de datos para obtener conclusiones válidas. Es un buen candidato para la obtención de herramientas, siempre y cuando se pueda encontrar una fuente válida de datos.

Es importante hacer un diagnóstico, para saber sobre ¿qué población? nuestro producto tiene mayor interés y se busca tener una alta cantidad de participantes, aunque la información no sea muy profunda. Por esta razón, se estima que un cuestionario es el instrumento adecuado para obtener la información necesaria.

6.2.2.2.2 Justificación de reactivos.

A continuación: se presentan la información que esperamos obtener por nuestra encuesta. Es importante tener los reactivos en mente al elaborar nuestras preguntas, para que la información sea de calidad y clara para el proyecto.

Número	Información de Interés
1	Hay que determinar: si la persona entrevistada considera los videojuegos su forma principal de entretenimiento como parte de su perfil.
2	Se busca determinar: los gustos del cliente para saber si estos influencian sus respuestas.
3	Se requiere conocer la información, sobre ¿como deciden si consumir un producto o no?, para determinar la estrategia de mercadeo de la organización para vender el producto.
4	Se debe determinar: si el cliente tiene interés en el producto solo por una breve descripción de sus funciones.
5	Determinar: cuáles de las funciones son más importantes para el cliente y por tanto, debemos enfocar nuestros recursos.

6	Conocer el mercado y si hay otros productos más notables, que sean similares a lo estamos desarrollando.

Tabla 6.2.2.2 – Justificación de reactivos de la herramienta de diagnóstico.

6.2.2.3 Análisis y diseño de la solución

Una vez determinado lo que se debe hacer, es planear la siguiente iteración de desarrollo, es decir analizar los requerimientos para dar paso al diseño de la solución. Esto desde el punto de vista del videojuego debe incluir:

- La implementación de las funciones mecánicas
- El diseño estético (música, personajes, interfaz de usuario, etc.)
- El diseño del guion narrativo y su transmisión a los usuarios
- La arquitectura tecnológica con las herramientas integradas.

6.2.2.4 Verificación y ajustes

Antes de liberar el producto; se debe realizar un proceso de verificación. Se recomienda hacer una sesión de "*play testing*", con un grupo tan grande como sea posible y tomar la retroalimentación al respecto. Dependiendo de ¿si hay suficiente tiempo y recursos?, se deberá integrar la retroalimentación.

6.2.2.5 Reglas de uso

Se le enseñará al jugador a usar las funcionalidades como:

- Participar en el juego:
  - Reglas del juego.
  - Condiciones de victoria.
  - Condiciones de derrota.
  - Mecánicas.
  - Como navegar la UI.
- Guardar avances. El jugador debe de ser capaz: de guardar el estado del juego en un archivo, que sea legible por el juego en sesiones futuras.
- Cargar avances. El jugador debe ser capaz: de recuperar el estado del juego guardado en una sesión previa.

6.2.3 Proceso de prototipos

6.2.3.1 Codificación

Las buenas prácticas en programación, siempre deben estar presentes a la hora de codificar cualquier solución, por lo que se sugieren algunas convenciones de codificación (Microsoft Corporation, 2021):

Convenciones de nombres:

- Pascal casing: se usa esta convención al nombrar una clase, un método o estructura
  - Cuando se nombra una interfaz se agrega una “I”, esto claramente indica que es una interfaz.
- Camel casing: cuando se nombran campos privados o internos, se comienza con el prefijo “_”.
  - Cuando se trabaja con un campo “static” que sea privado o interno se usa el prefijo “s_” y para hilos estáticos se usa “t_”.

- Convenciones de disposiciones: Un buen formato de la disposición del código enfatiza la estructura del código para hacerlo más legible.
  - Escribir una sola declaración por línea.
  - Si las líneas continuas no son indentadas, indentarlas usando “tab”
  - Usar paréntesis para hacer las cláusulas en una expresión obvia.
- Convenciones de comentarios.
  - Escribir cada comentario en una línea separada, no al final de una línea de código.
  - Empezar cada comentario con una mayúscula.
  - Acabar el comentario con un punto.
  - Insertar un espacio entre el delimitador de comentarios (//) y el texto.
  - No crear bloques de asteriscos alrededor de los comentarios.

6.2.3.2 Ejecución de pruebas

- **Pruebas unitarias.** Las pruebas unitarias, son una forma de comprobar que un fragmento de código funciona correctamente. Cada función que tenga una salida, debe ser probada y debe asegurarse que la salida funciona y se obtienen los resultados esperados.
- **Pruebas de rendimiento**. Es una prueba que se realiza sobre la aplicación, con una carga de trabajo determinada. Se debe mantener un tiempo de respuesta: menor a 2 segundos bajo la carga más pesada de renderización.
- **Pruebas de integración.** Son las pruebas que se realizan sobre la aplicación, para asegurarse que todas las interfaces del juego funcionan correctamente. Todos los módulos que se han desarrollado durante esta iteración, deben de funcionar correctamente entre ellos para poder entregar el resultado esperado.
- **Pruebas funcionales.** Son las pruebas de caja negra (es decir sin conocimiento del código), que se realizan para asegurarse que las funciones individuales son correctas.

- **Pruebas de escalabilidad.** Son pruebas no funcionales, que determinan la capacidad del sistema para servir a un número determinado de usuarios. Básicamente: comprueba el rendimiento de una aplicación en diferentes cargas de trabajo.

6.2.3.4 Validación

La validación, es diferente a la verificación, una validación no puede ser realizada por los desarrolladores. La verificación debe llevarse a cabo por un sector de usuarios finales.

Se debe presentar el prototipo a los usuarios, para obtener retroalimentación sobre el mismo. Los usuarios deben estar informados del alcance del prototipo. Estos comentarios de retroalimentación, se deben de tomar en cuenta para la siguiente iteración del juego o si es la última iteración contemplada, entonces: evaluar si es suficiente para generar un producto mínimo viable.

7 Desarrollo del proyecto

7.1 Propuesta socio-técnica

Al implementar esta metodología, se espera que el desarrollador sea capaz de determinar ¿qué herramientas funcionan mejor para su proyecto?, el alcance logrado y definir así el alcance final del proyecto que se busca.

También permitiría generar: nuevas empresas conforme los desarrolladores independientes van creciendo, en sus propios negocios, aumentando el prestigio internacional del país; como un *colectivo* tecnológico y exportador de productos de la cultura mexicana.

Desde el punto de vista técnico, se tendrá una metodología enfocada a los pequeños desarrolladores, lo cual les representa un beneficio directo, donde podrían adquirir experiencia y conocimiento valioso en otras áreas. Para los creadores de herramientas, analizar esta metodología, les podría ayudar a entender las necesidades de sus clientes.

7.2 Implementación de la propuesta

La metodología propuesta, está basada principalmente en la tétrada de juegos, como se ha explicado anteriormente.

Se ha escogido este método principalmente, debido a las limitaciones del proyecto en términos de recursos. A diferencia de la otra metodología presentada (Proceso de diseño centrado en el juego), donde se espera contar con el tiempo y equipo suficientes, para realizar diversas iteraciones en un solo producto, la tétrada nos permite: crear una visión del juego y establecer metas de avance.

Cabe aclarar; que no se abandonaran los prototipos del todo, pero en vez de crear una infinidad de prototipos para validar todas las ideas, confiaremos, en que el diseñador tenga la experiencia y el conocimiento para poder diseñar el producto desde cero.

La tétrada incluye 3 capas: inscrita, cultural y dinámica. En este caso; solo se centra en la capa inscrita, porque es sobre la que se tiene control directo. No se tiene un control directo sobre la capa dinámica y la capa cultural, por lo que es imposible crear un diseño coherente sobre tal.

7.2.1 Visión

En la visión metodológica, se recomienda planear sobre las capas inscritas.

Se debe determinar el tipo de producto a proponer y ¿cuál sería el enfoque que se le dará?, usando las Mecánicas, Estéticas, Tecnología y Narrativa. Para este proyecto se adoptan y describen en los siguientes puntos.

Mecánicas:

El ciclo de juego está determinado por un mapa cuadriculado, el jugador tendrá un número determinado de unidades o personajes, que podrá mover de forma limitada dentro del tablero.

El jugador puede:

- Mover las unidades asignadas como las del jugador dentro del mapa
- Usar las habilidades de dichas unidades
- Pasar el turno sin mover una unidad
- Salvar el juego y continuar más tarde
- Determinar el volumen de la música y sonidos
- Aumentar las características de sus unidades

El programa puede:

- Mover las unidades asignadas como las del enemigo dentro del mapa
- Usar las habilidades de dichas unidades
- Pasar el turno sin mover una unidad

Estéticas:

- El juego seguirá una estética de 2 dimensiones
- Los personajes usarán una estética de arte en pixel
- En estilo de chip tune (música de 8 bits)

Tecnología:

- Para la generación de activos gráficos se ha determinado a usar la herramienta Aseprite
- Para correr el juego se usará como columna vertebral, el motor de juegos Unity
- Para la generación de activos de música y sonido se usará (LMMS)

Narrativa:

La ficción del juego, estará centrada en un mundo de fantasía medieval, solo habrá humanos como criaturas civilizadas en este mundo (a diferencia de, por ejemplo, Tolkien, donde los humanos

habitan el mundo con elfos, enanos, hobbits y orcos). Sin embargo, habrá otras criaturas salvajes (osos, leones, águilas) y criaturas míticas (dragones, minotauros, etc.), pero ninguna de ellas, tendrá una civilización de la cual hablar.

En términos técnicos, la narrativa no es importante, solo existe para crear una guía para nuestra estética y una expectativa para nuestros usuarios.

7.2.2 Alcance

Al completar el juego se espera tener al menos:

- 1 mapa estratégico. Candidato para mover a la segunda iteración
- 3 mapas tácticos (donde se desarrolla la acción del juego)
- 3 tipos de unidades para el jugador
- 1 enemigo
- 1 menú de inicio
- 1 menú de configuración

7.2.3 Determinar roles

1. Programador.
2. Diseñador gráfico.
3. Compositor.
4. Diseñador UI.
5. Diseñador del juego.
6. Escritor.
7. Ingeniero QA.
8. Administrador de proyecto.

7.2.4 Estimar habilidades por rol

Nombre de rol	**Brecha de habilidad**	**Actividades para desarrollar**
Programador	Usar Unity en 2d Arquitectura Software	Implementar algoritmo A* Diseñar el software y sus funciones
Diseñador grafico	Dibujar Sprite 2d	Aprender manejo de Sprite Dibujar personajes
Compositor	Aprender teoría musical. Aprender a usar el software (LMMS)	Crear las piezas musicales

Diseñador UI	Diseñar la interfaz de usuario	Diseñar interfaz de usuario Implementar diseño de interfaz
Diseñador del juego.	NA	Diseñar mecánicas del juego.
Ingeniero QA	Determinar pruebas de contenido y funcionalidad para cada función.	Ejecutar pruebas de contenido. Ejecutar pruebas de funcionalidad.
Escritor	Aprender teoría narrativa.	Escribir la narrativa
Administrador de Proyecto	Aprender a determinar la visión de un proyecto	Determinar objetivos. Métricas de desempeño (Función sobre Tiempo)

Tabla 7.2.4 - Estimación de habilidad

7.2.5 Estimar viabilidad

7.2.5.1 Viabilidad técnica de la propuesta

La propuesta es completamente viable en términos técnicos, otros proyectos similares ya se han producido anteriormente con estas herramientas y diseño gráfico (itch.io, 2022)

7.2.5.2 Viabilidad económica

Se empieza por el análisis económico del proyecto. Los costos proyectados son en base al tiempo que el desarrollador le dedicará al proyecto y cada período se define como un mes. La inversión inicial empieza en $185, por la herramienta Aseprite para empezar a generar los valores gráficos.

No se consideran ventas o entrada de capital todavía, pero es un inicio para estimar el coste de inicio del proyecto y una vez que se comience a vender, será posible repetir el proceso de estimación para determinar el coste de cualquier proyecto similar y determinar ¿cuándo será redituable un proyecto?

Período	**Flujo**	**PV**	**Acumulado PV**	**Inversión**	**Gastos**	**Ventas**		**Costos**
						Cantidad de productos vendidos	Entrada de capital por periodo	Costo del tiempo que se le dará al proyecto
1	($185.00)	($183.87)	($183.87)	$ (185.00)	0	0	$ -	
2	($1,500.00)	($1,481.77)	($1,665.64)		0	0	$ -	$ (1,500.00)
3	($1,500.00)	($1,472.74)	($3,138.38)		0	0	$ -	$ (1,500.00)
4	($1,500.00)	($1,463.76)	($4,602.13)		0	0	$ -	$ (1,500.00)
5	($1,500.00)	($1,454.83)	($6,056.97)		0	0	$ -	$ (1,500.00)

6	($1,500.00)	($1,445.97)	($7,502.93)		0	0	$ -	$ (1,500.00)
7	($1,500.00)	($1,437.15)	($8,940.08)		0	0	$ -	$ (1,500.00)
8	($1,500.00)	($1,428.39)	($10,368.47)		0	0	$ -	$ (1,500.00)
9	($1,500.00)	($1,419.68)	($11,788.16)		0	0	$ -	$ (1,500.00)
10	($1,500.00)	($1,411.03)	($13,199.19)		0	0	$ -	$ (1,500.00)
11	($1,500.00)	($1,402.43)	($14,601.61)		0	0	$ -	$ (1,500.00)
12	($1,500.00)	($1,393.88)	($15,995.49)		0	0	$ -	$ (1,500.00)
13	($1,500.00)	($1,385.38)	($17,380.87)		0	0	$ -	$ (1,500.00)
14	($1,500.00)	($1,376.94)	($18,757.81)		0	0	$ -	$ (1,500.00)
15	($1,500.00)	($1,368.54)	($20,126.35)		0	0	$ -	$ (1,500.00)

Tabla 7.2.5.2 - Muestra de los resultados de la encuesta de diagnóstico

Dadas las respuestas encontradas en la herramienta de diagnóstico, anteriormente, con un 76% de respuesta positiva para el proyecto, podemos decir que se tiene un grupo estadísticamente válido, que permite afirmar que hay un interés por parte del público general en este tipo de productos.

El diagnóstico, es dirigido al público en general usando un cuestionario como se especifica en el Anexo 1.

Hubo 68 resultados. De los cuales: solo 23 dieron como respuesta que eran entusiastas de los videojuegos. Es decir; la mayoría de los entrevistados no eran entusiastas de los videojuegos o no consideraban videojuegos como su pasatiempo principal.

52 personas interesadas en el proyecto, a pesar de que la vasta mayoría de los entrevistados no eran entusiastas de los videojuegos. Y 16 personas tenían una opinión negativa o ambivalente conforme al proyecto. Esto indica que de los entrevistados al menos un 76% mostraron un interés en el proyecto.

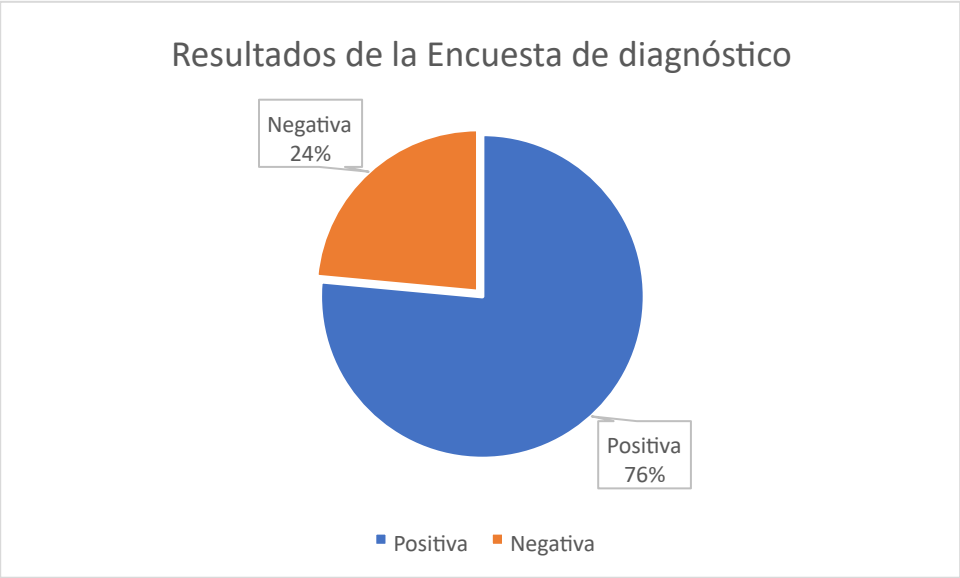

Figura 7.2.5.2 – Resultados de la encuesta de diagnóstico

Esto nos indica que podemos *marquetear* el juego al público en general y no necesariamente a un público especializado.

7.2.6 Exploración de herramientas

7.2.6.1 Análisis para la herramienta de generación de activos de gráficos

Aseprite. Es una herramienta de arte en píxel para Windows, MacOS y Linux. (Aseprite, s.f.). Tiene un coste.

MSPaint. Es una herramienta de edición de imágenes disponible para Windows. (Wikipedia, s.f.). Es completamente gratuito.

Photoshop. Es una herramienta de edición de imágenes usada para crear gráficos de alta calidad. (Adobe, s.f.). Tiene un coste.

Nombre	**Cumplir objetivo.**	**Experiencia de los desarrolladores con la herramienta**	**Costo comparativo de la herramienta**	**Compatibilidad entre herramientas**	**Soporte**	**Total**
Aseprite	1	0,5	0,5	1	1	4
MSPaint	1	0,5	1	1	0,5	4
Photoshop	1	0	0	1	0	2

Tabla 7.2.6.1 - Comparativa de herramientas de generación de activos gráficos

7.2.6.2 Análisis del motor de juego

En el mercado hay varios motores de juego, sin embargo, la literatura recomienda 3 por encima (Toftedahl, Which are the most commonly used Game Engines?, 2019) de los demás como estándar de industria, Unity y Unreal, son los motores de juego más usados en el mercado profesional y Unity es el motor de juegos más usado en juegos independientes (itch.io, s.f.).

También se comparan estos 2 titanes con Godot, el cual, si bien es un motor de juego más pequeño, ha recibido aclamaciones y está apuntando a crecer en el futuro. (gamefromscratch, 2019)

A continuación, se explican las características de cada motor de juegos. Y al final una tabla comparativa para justificar la elección final de este proyecto, que ha sido la herramienta Unity.

El factor decisivo ha sido la popularidad de la herramienta, y por lo tanto mayor cantidad de documentación de soporte no oficial, además de libros y tutoriales, la gran cantidad de plataformas que soporta y la flexibilidad de su sistema de monetización para desarrolladores independientes.

7.2.6.2.1 Unity

- Tiene límite en su programa gratuito a recibir $100,000 en fondo (Lily, 2021).
- Es gratuito.
- Escribe una vez, despliega donde sea (OS X, PC, Web, Linux, iOs, Android, BlackBerry, Windows Phone, Windows Store, etc).
- Soporte continuo.
- Varias características (detección de colisión, simulación física, encontrar caminos, sistemas de partículas, shaders, ciclo de juego) (gibson, 2015, págs. 237-238).
- Diseñado con simpleza en mente.
- Compatible con C#.
- Gran comunidad de desarrolladores con bibliografías, tutoriales y materiales de soporte (Borromeo, 2020, pág. 798).

7.2.6.2.2 Godot

- Godot puede crear juegos en 2D o 3D (Felicia, 2020, págs. 30-33).
- Godot exporta juegos a una amplia gama de plataformas, incluyendo móviles, Android iOS, Windows MacOS o Linus.
- Godot usa C# o GDScript.
- Godot se encuentra en la versión 3.0 lo que ha hecho que se agreguen funciones con versiones diferentes (Bradfield, 2018, pág. 300):
- Godot 1 (2014): light mapping, shaders y navegación
- Godot 2 (2016): Instanciación de escenas, herencia y mejoras al corregir.
- Godot 3 (2018): Soporte a C#, visual scripting, soporte a VR y soporte para ensamblaje red
- Código abierto.
- Renderizar en 2D y 3D.
- Simulador físico.
- Soporte de plataforma.

- Ambiente de desarrollo común. Al usar la misma interfaz unificadas para hacer juegos múltiples no se tiene que reaprender un nuevo flujo de trabajo al inicio de un proyecto.
- Hay herramientas que ayudan en el uso de imágenes, sonidos, animaciones, corrección y más.

7.2.6.2.3 Unreal

- Unreal pide 5% de las ganancias de tu producto después del primer millón (unrealengine, s.f.).
- Unreal usa C++.
- Puede renderar en 3D (Hammad Fozi, 2020).
- Tiene templados para diferentes tipos de juego (Cordone, 2019):
    1. Blanco.
    2. Primera persona.
    3. Volar.
    4. Realidad aumentada.
    5. Rompecabezas.
    6. Rolling.
    7. 2D.
    8. Tercera persona.
    9. Desde arriba.
    10. Vehículo.
    11. Realidad virtual.

7.2.6.4 Análisis para la herramienta IDE

Atributo del IDE	**Godot**	**Unity**	**Unreal**
Monetización	Gratis	Tiene límite en su programa gratuito a recibir $100,000 en fondo	Unreal pide 5% de las ganancias de tu producto después del primer millón
Alcance de plataformas	Tiene acceso a plataformas convencionales	Tiene acceso a casi todas las plataformas actuales.	Mobile Android, Linux, Web, Playstation 4 y Switch, VR
Activo de soporte	Tiene acceso de una tienda de activo.	Tiene acceso de una tienda de activo.	Tiene acceso de una tienda de activo

Lenguajes	GDScript o C#, C++ y Visual Scripting.	C# y Javascript. También tiene visual scripting (no requiere código).	Unreal usa C++, También es posible usar blueprints (no requiere código).
Documentación de soporte	Buena documentación, pero hay pocos tutoriales	Muchos tutoriales y buena documentación de recursos.	Varios tutoriales, aunque no tantos como Unity, buena documentación

Tabla 7.2.6.4.1 - Comparativa de motores de juego con base a sus características

Nombre	**Cumplir objetivo.**	**Experiencia de los desarrolladores con la herramienta.**	**Costo comparativo de la herramienta.**	**Compatibilidad entre herramientas.**	**Soporte.**	**Total**
Unity	1	0.5	1	1	1	4.5
Unreal	1	0	0.2	1	1	3.2
Godot	1	0	0.5	1	1	3.5

Tabla 7.2.6.4.2 - Comparativa de motores de juego con base a su puntaje

Visual Studio. Es un entorno de desarrollo integrado (IDE) para Windows y macOS que en su versión “Code” y es compatible con Unity (unity, 2019). Además, es compatible con HTML, CSS, JavaScript, JSON, Python y C# (visualstudio, s.f.). Esta herramienta no requiere un costo adicional y es completamente gratuito (visualstudio, s.f.).

Eclipse. Es un IDE para Windows desarrollado por la fundación Eclipse, sin embargo, no es compatible con Unity (unity, 2019). Este IDE permite escribir en C, C++, JavaScript, PHP (Eclipse, 2022). La herramienta es completamente gratuita. (Eclipse, 2022).

JetBrains. Es un entorno de desarrollo integrado para Windows, compatible con Unity (unity, 2019). Jetbrains puede ser usado para codificar en C#, Visual Basic, C++, F$, Javascript, CoffeScript, JSon, Markdown, HTML, Xpath, XML, Node y otros. (jetbrains, 2021). JetBrains no es gratuito, y requiere un costo anual durante a cuando se escribe este documento. (jetbrains, s.f.)

Nombre	**Cumplir objetivo**	**Experiencia de los desarrolladores con la herramienta**	**Costo comparativo de la herramienta**	**Compatibilidad entre herramientas**	**Soporte**	**Total**
Visual Studio	1	0.5	1	1	1	4.5
Eclipse	1	1	1	0	1	4
JetBrains	1	0	0	1	1	3

Tabla 7.2.6.4.3 - *Comparativa de IDE en base al puntaje*

7.2.6.5 Análisis para la herramienta de generación de activos de música y sonido

LMMS. Es una herramienta de creación musical en línea de código abierto. Forma parte de una familia de softwares conocidos como Digital Audio Workstation (DAWs). LMSS funciona con archivos FLAC, WAV y Ogg (Lily, 2021). Unity es compatible con archivos “.wav”, “.aif”, “.mp3” y “.ogg”. (Unity, s.f.).

BandLab. Es una herramienta de creación musical, funciona en cualquier browser web, Android y iOS. Bandlab soporta archivos MIDI, MP3, WAV, ACC, M4A y OGG (help.bandlab.com, 2022). Unity es compatible con archivos “.wav”, “.aif”, “.mp3” y “.ogg”. (Unity, s.f.).

FL Studio. Es una herramienta de creación musical, funciona en cualquier browser macOS, OS X. FL Studio usa archivos con el formato dmptrn, dwp, finstaller, flkey, flm, flp, fpr, fsc, fst, sf2, vst, wv. Unity es compatible con archivos “.wav”, “.aif”, “.mp3” y “.ogg”. (Unity, s.f.).

Nombre	Cumplir objetivo	Experiencia de los desarrolladores con la herramienta	Costo comparativo de la herramienta	Compatibilidad entre herramienta	Soporte	Resultado
LMMS	1	0	1	1	0.5	3.5
BandLab	1	0	1	1	1	4
FL Studio	1	0	1	0	0	2.5

Tabla 7.2.6.5 - Comparativa de herramientas de generación de música y sonido

7.2.7 Plan de Desarrollo

Para realizar un plan de desarrollo: se debe tomar un listado de las funciones a desarrollar, subdividida en la más pequeña unidad posible, determinar el esfuerzo medido en el tiempo que se le da a cada característica y el tiempo que durará el *sprint*.

Para este proyecto se tomó un tiempo de 4 meses, para hacer un avance importante.

Figura 7.2.7 - Instanciación de la tabla del plan de desarrollo del primer ciclo de desarrollo

Para la segunda ronda de implementación, se ha redefinido el alcance del proyecto tal y como está definido en el documento de diseño 1.1. Se tomo como tiempo de desarrollo otros 4 meses.

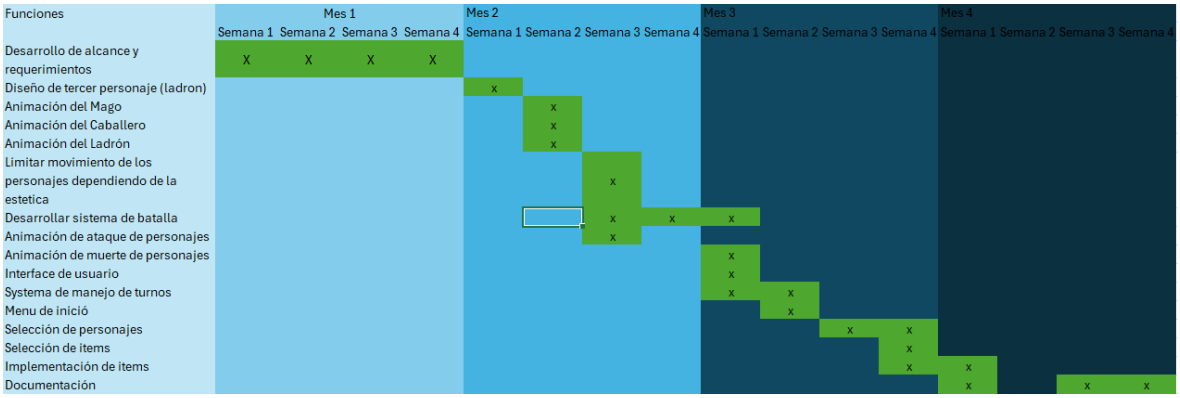

Figura 7.2.8 - Instanciación de la tabla del plan de desarrollo del segundo ciclo de desarrollo

7.3 Diseño, desarrollo, implementación y pruebas

7.3.1 Definición de especificaciones

Mapa táctico:

a. Mecánicas. Estética. Debe tener celdas vacías dentro para simular obstáculos.
b. Estética. Debe tener un punto de vista isométrico.
c. Estética. Debe estar implementado usando un diseño en 2d.
    i. Se usará Aseprite para diseñar las celdas individuales.
    ii. Se usará la herramienta interna de Unity, GridBrush, para crear el mapa usando las celdas.
d. Mecánicas. Debe ser 8 x 10 celdas.

Mago personaje:

a. Estética. Debe estar diseñado usando pixel art. El personaje debe tener las características de un mago (sombrero, báculo y toga).

a. Debe diseñarse el personaje usando 4 puntos de vista. Mirando al frente a la derecha, mirando al frente a la izquierda, mirando atrás a la izquierda, mirando atrás a la derecha.
b. Funcionalidad. Debe de poder moverse entre 2 puntos.

a. No puede moverse en diagonal, solo debe poder moverse perpendicular y lateralmente.
b. Debe moverse tomando la ruta más corta posible.
    i. Se codificará un algoritmo A*, usando el lenguaje C#.
c. Debe moverse usando la ruta indicada.
d. Debe moverse usando una animación, a un paso de “1.0f”.
e. No puede moverse fuera del tablero.

7.3.2 Levantamiento de requerimientos

Usando el método de observación, se ha identificado los siguientes requerimientos para el proyecto. Se llegaron a esta conclusión al observar otros juegos, dentro del mismo género que hemos determinado. No se debe confundir el método de observación, para determinar requerimientos técnicos del proyecto, con el cuestionario que nos sirve para determinar la viabilidad económica del proyecto.

Un mapa táctico. El mapa debe tener una cuadrícula, que pueda simular los caminos de un personaje y los obstáculos sobre que debe superar.

Un personaje. Debe ser diseñado usando píxel art, poder moverse sobre la cuadricula.

7.3.3 Análisis y diseño de la solución

Durante esta iteración del prototipo debemos incluir las siguientes mecánicas:

- Movimiento del personaje. Específicamente se usará el Algoritmo A* con el código descrito en el Anexo 3.
- GridManager es la clase que administra el movimiento del personaje dentro del tablero del juego. Usando el método “Update” que es llamado en cada “frame” del juego.
- Cada instanciación de los personajes tiene acceso a un GridManager.

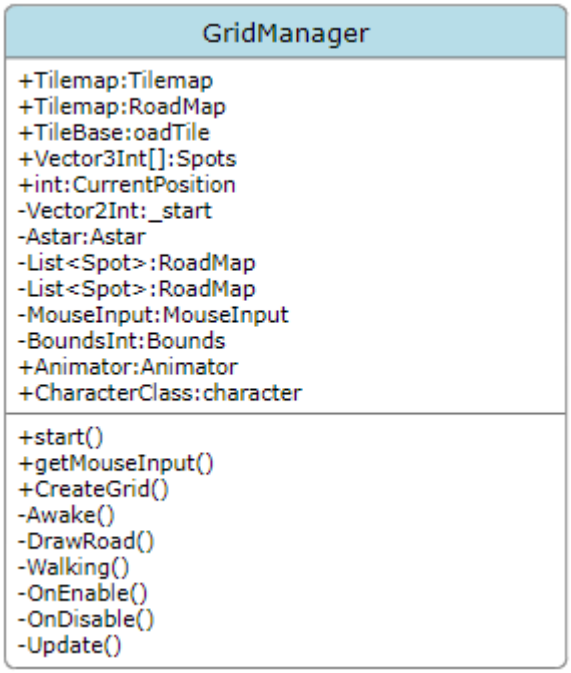

*Figura 7.3.3.1 – Diagrama UML de la clase “GridManager” a implementar el algoritmo A**

El método Update de la clase, determina si un personaje es seleccionado y si es seleccionado: entonces a donde se va a mover.

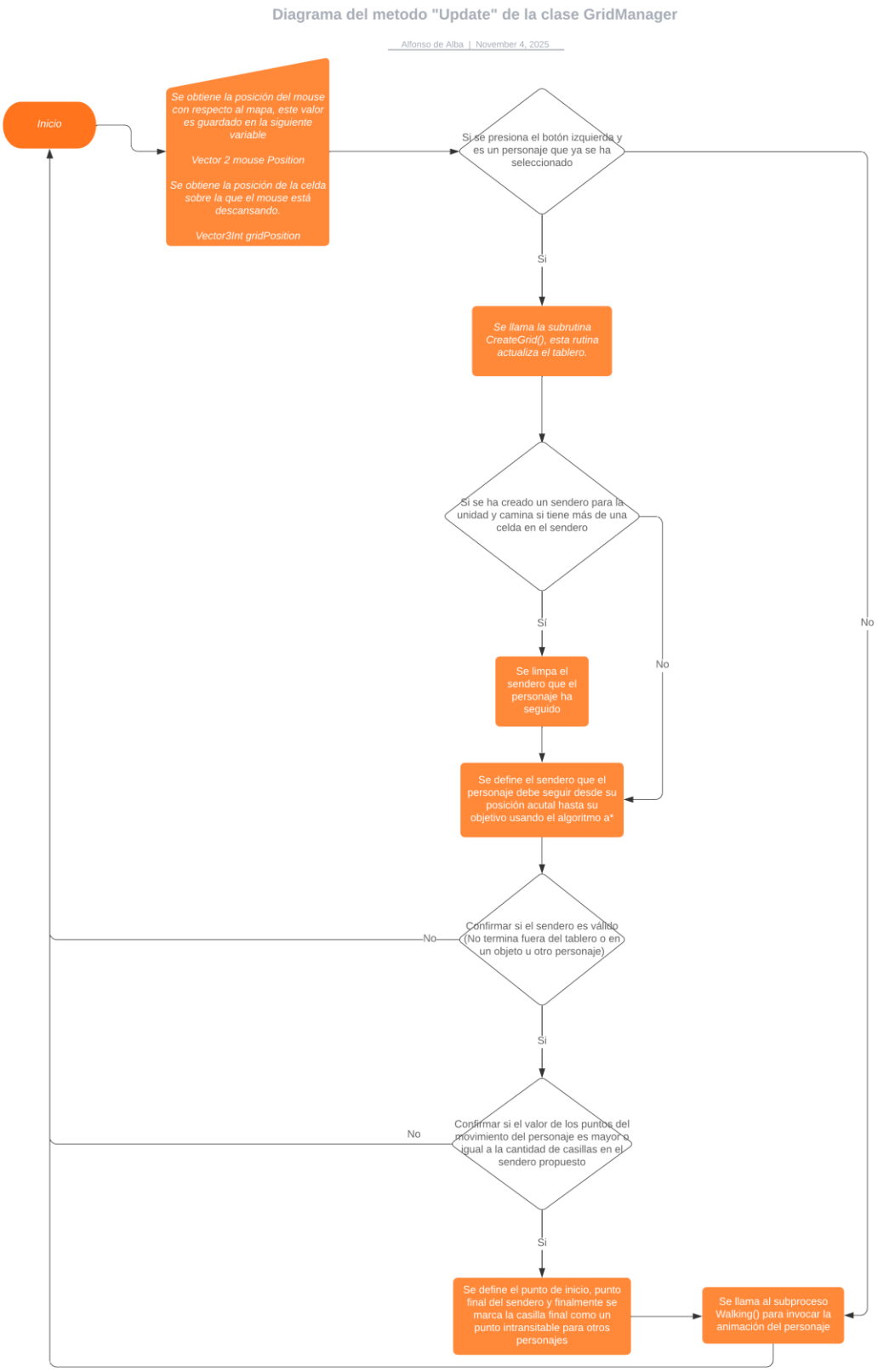

Figura 7.3.3.2 – Método "Update" de la clase GridManager

Con este método: se busca controlar el movimiento del personaje. Se toma la posición actual del personaje y la casilla objetivo y se aplica el algoritmo A* para encontrar el camino más corto entre ambas.

Para que el sendero sea aceptado dentro del juego, debe cumplir con que el sendero sea usable por el personaje y el personaje tenga, suficientes puntos de movimiento para viajar por el sendero; de otra forma es rechazado y el personaje no se mueve.

Siendo este un método “update” es llamado cada actualización de la pantalla.

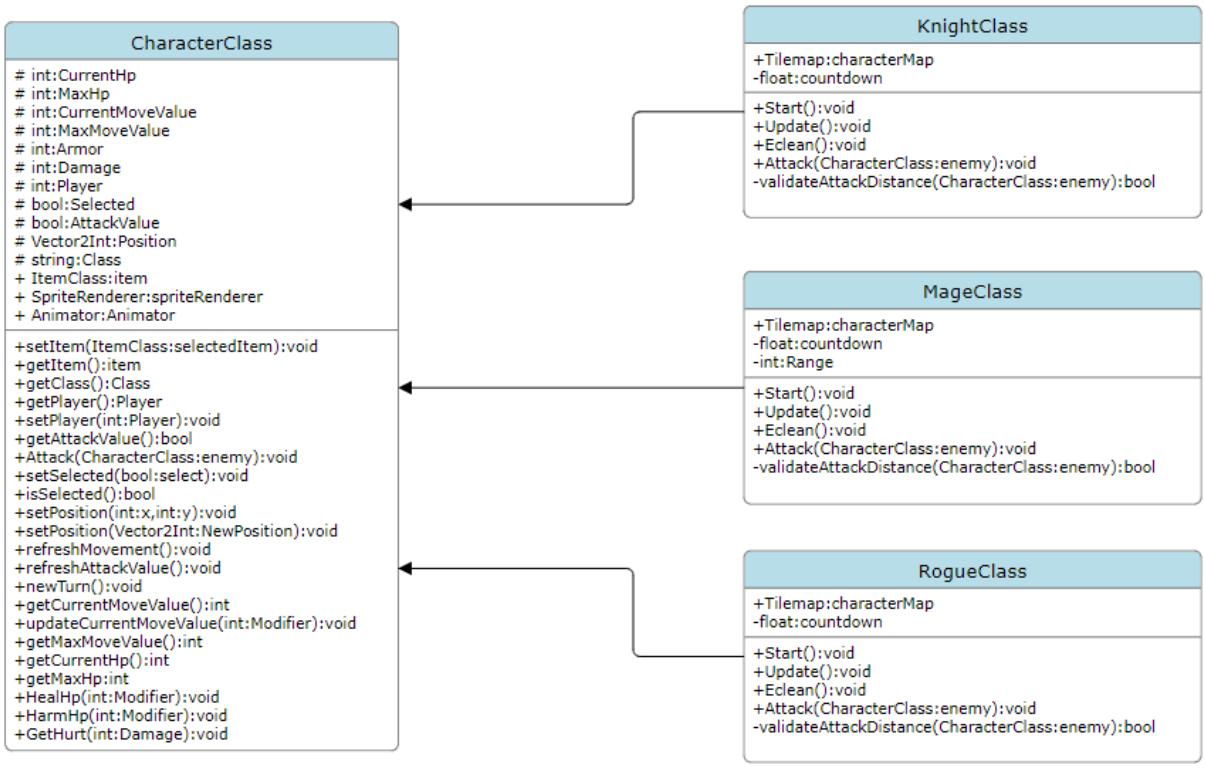

Figura 7.3.3.3 – Diagrama UML del manejo y comportamiento de los personajes

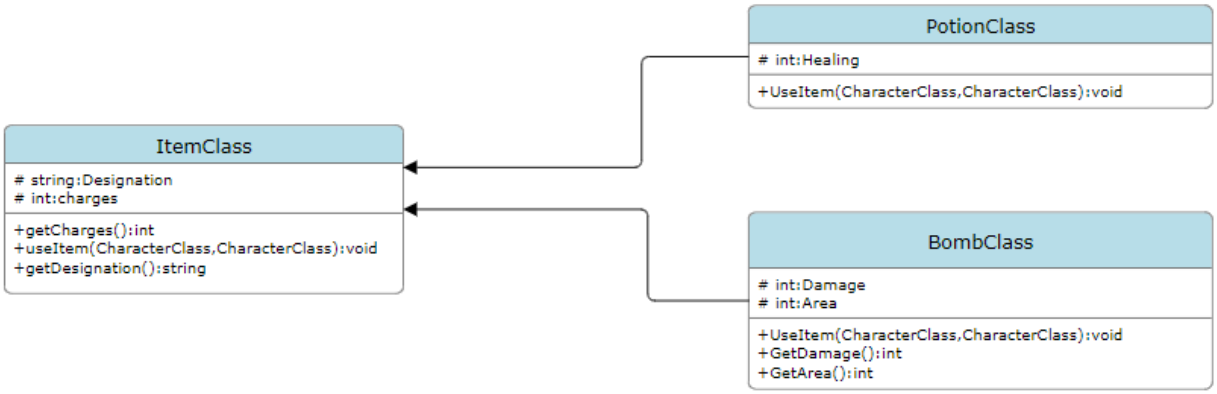

Figura 7.3.3.4 – Diagrama UML del manejo y comportamiento de los ítems de los personajes

El sistema de batalla determina: la creación de los personajes dentro del tablero en el método ¨Start¨, también determina: si va a atacar, usar un item o actualizar la interfaz usuario dependiendo del contexto.

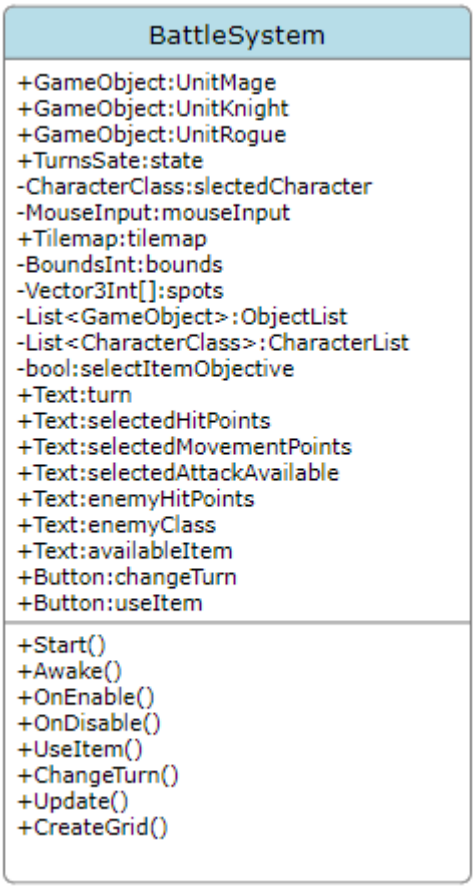

Figura 7.3.3.5– Diagrama UML del manejo y comportamiento de los personajes dentro del juego

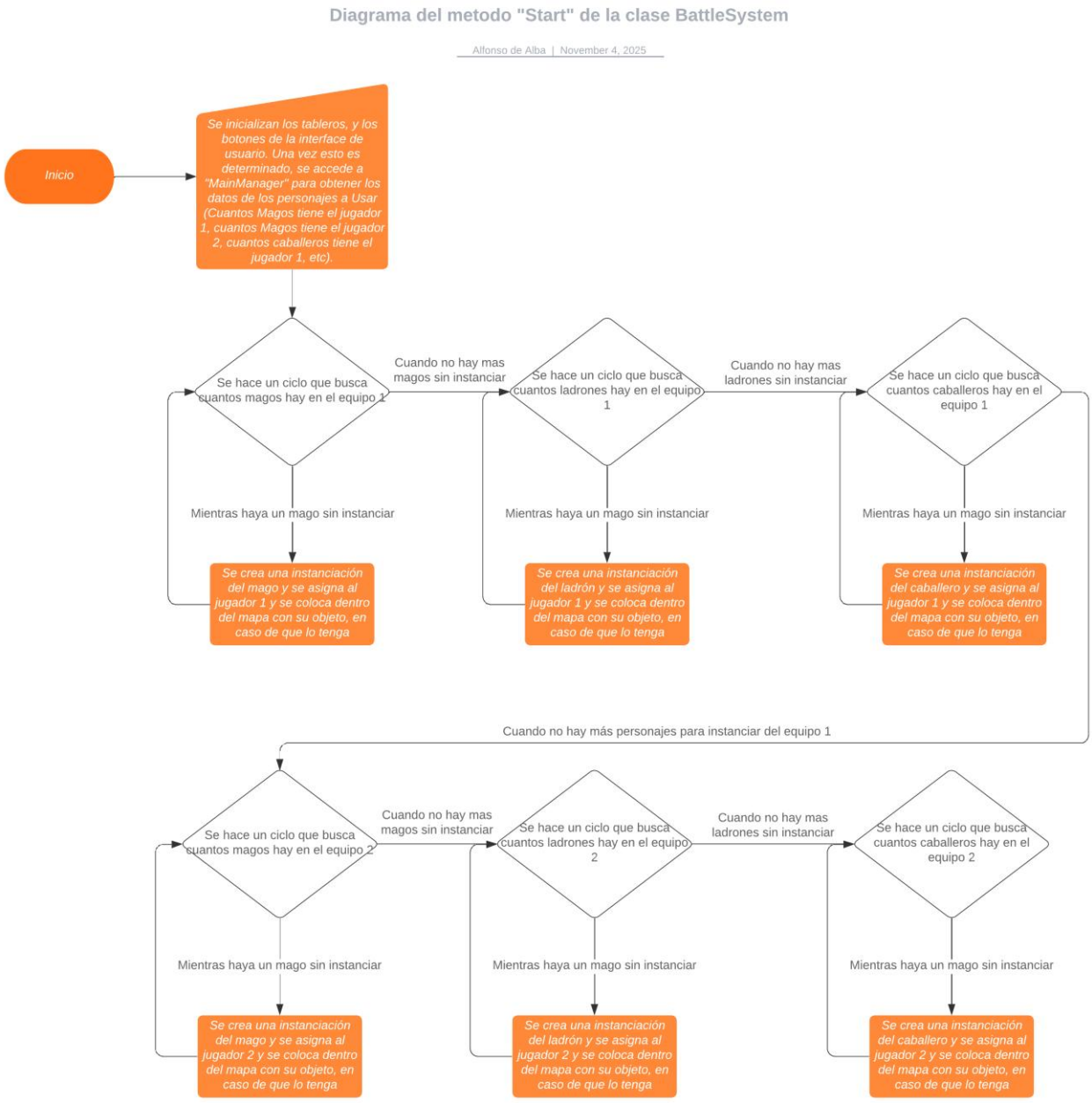

Figura 7.3.3.6– Método "Start" de la clase BattleSystem

El sistema de batalla determina: la creación de los personajes dentro del tablero en el método ¨Start¨, también determina: si va a atacar, usar un item o actualizar la interfaz usuario dependiendo del contexto.

El método "Start" es invocado solo una vez al iniciar el escenario. Se obtienen los datos requeridos en forma de una lista, en la que se especifica cuantos magos, caballeros y ladrones hay por equipo y qué objetos tendrá equipada cada clase.

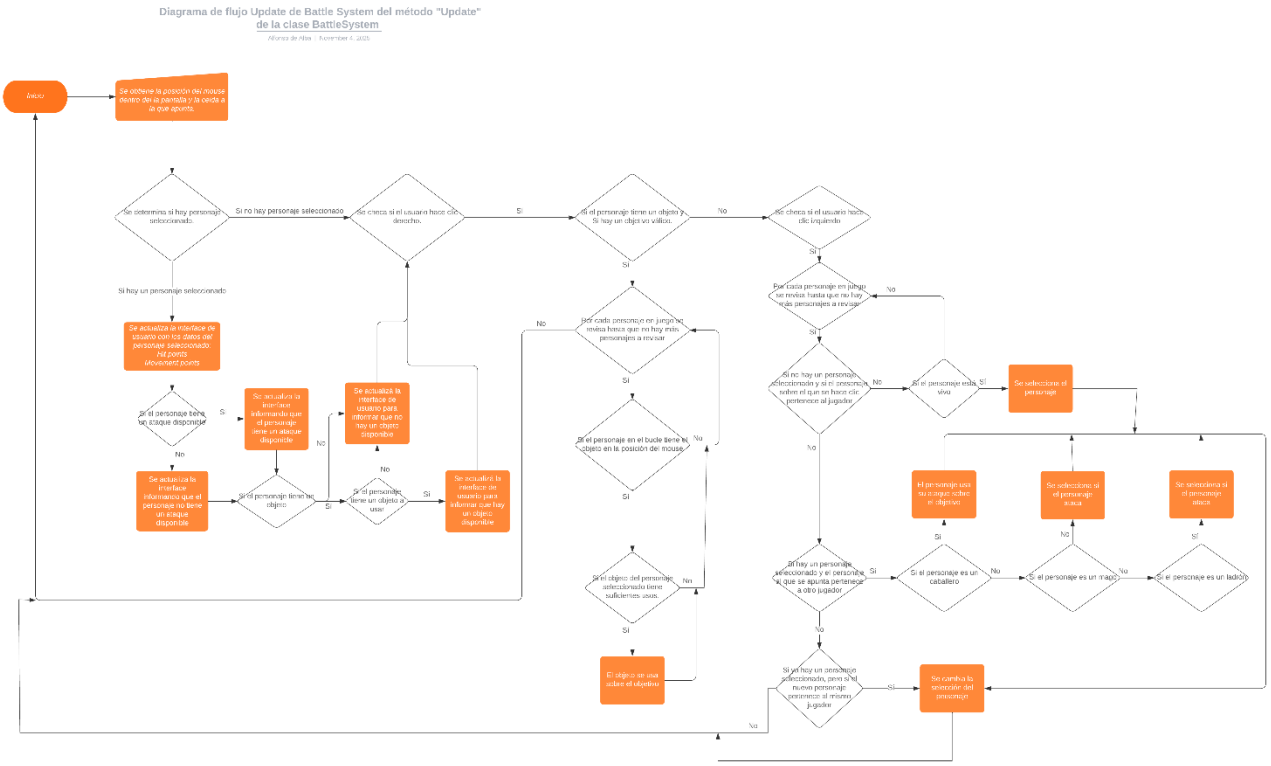

Figura 7.3.3.7– Método "Update" de la clase BattleSystem

El método "Update" es invocado cada que se actualiza la pantalla, cada "frame". Este método administra ¿Que personajes son seleccionables? tomando la posición del personaje seleccionado y asegurando que el personaje pertenece al jugador.

Una vez seleccionado el jugador podrá:

- Atacar a un personaje de otro jugador, si se selecciona a un personaje sobre el que el jugador no es el dueño: usando clic izquierdo.

- Cambiar de personaje seleccionado si se selecciona un personaje, si el jugador hace clic izquierdo sobre otro personaje, sobre el cual el jugador tiene derechos como dueño.

- Usar un ítem si el jugador usa el clic derecho. El juego verifica si el item es existente, si está disponible y si el objetivo es válido.

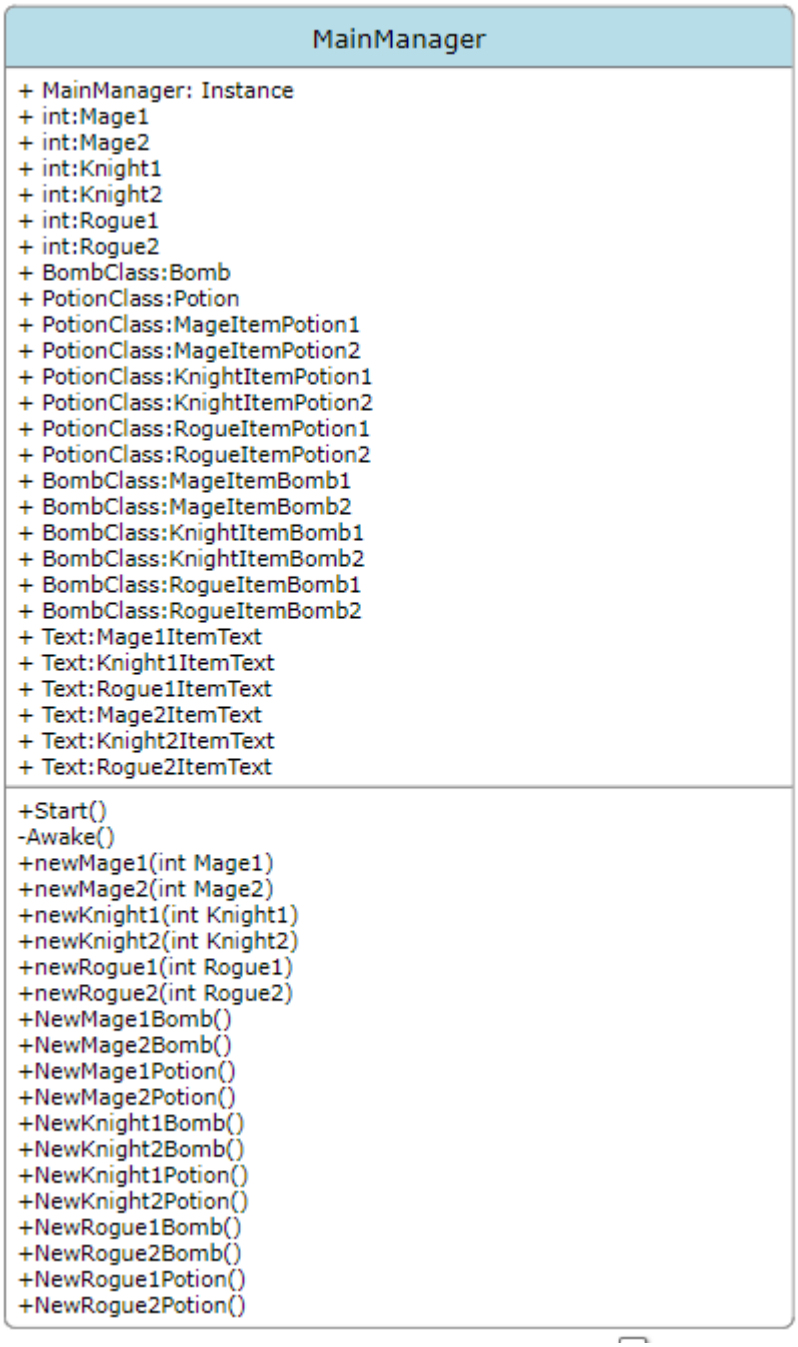

Figura 7.3.3.8– Diagrama UML de la clase MainManager.

MainManager como se muestra en la Figura 7.3.3.8, es una clase constante entre escenarios. Se usa principalmente para guardar la cantidad de personajes por clase por equipo (1 o 2) y el item que fue escogido para ese personaje.

En estética esta iteración debe incluir.

- Personaje. Debe ser un mago como es imaginado en escenarios de fantasía, usando un estilo de arte en píxel. Usando el gráfico de ejemplo en el Anexo 2.

- Mapa táctico. Debe tener una cuadrícula que lo defina y estar construido usando un arte en píxel.

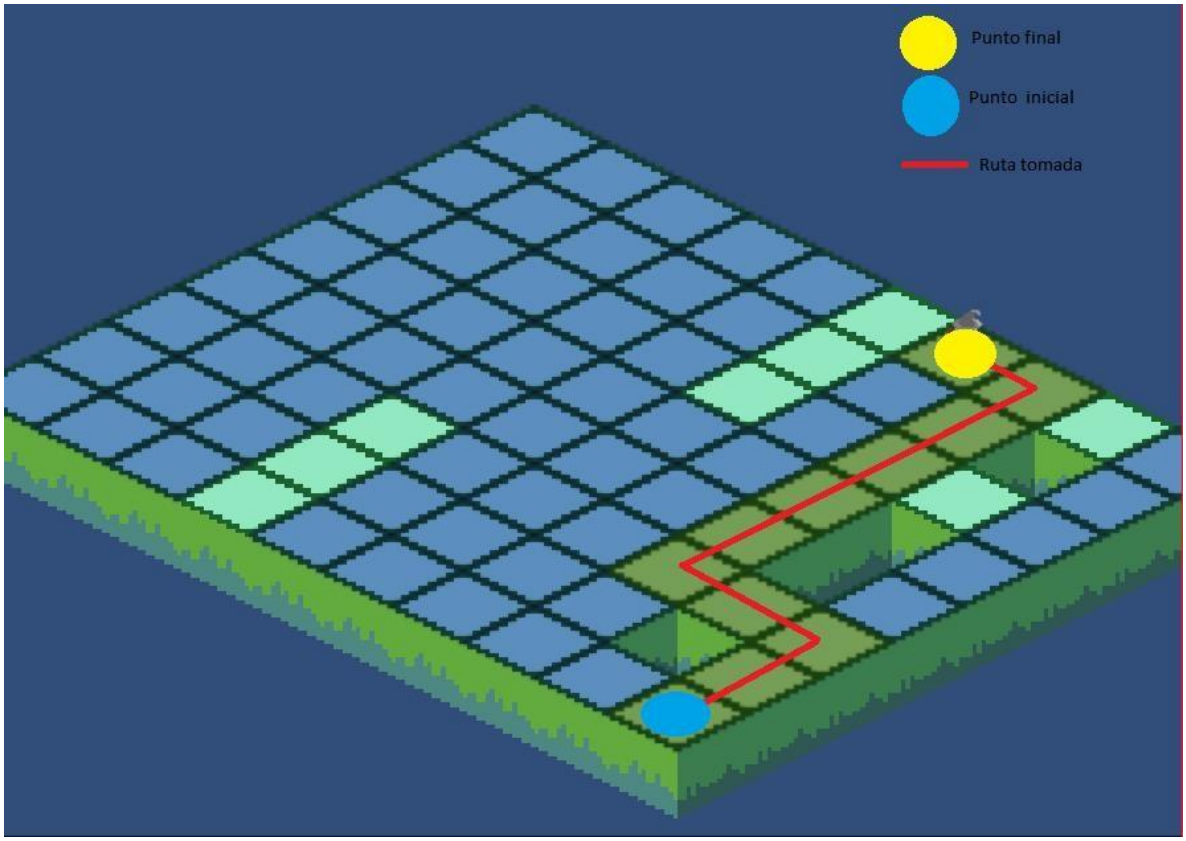

*Figura 7.3.3.9 - Ejemplo del movimiento de personajes usando algoritmo A**

No habrá narrativa durante está iteración.

7.4 Proceso de prototipos

7.4.1 Codificación del prototipo

La parte más importante de la codificación durante el proyecto fue el algoritmo A* (A star o A estrella), que permitirá el movimiento correcto de las unidades. Esto se considera la parte más importante dentro del escenario, por que afectara a todos los escenarios siguientes.

El algoritmo A* es un algoritmo de búsqueda heurístico, es decir: de descubrimiento, que busca encontrar el camino más corto entre un punto A y punto B en una rejilla. Sin embargo; a diferencia del algoritmo clásico (o como normalmente se codifica) se debe de tomar en cuenta 2 casos especiales.

- Los personajes no se pueden mover en forma diagonal, solo se pueden mover de forma lateral y vertical desde la perspectiva de uno de los bordes de la rejilla.
- Los personajes pueden tener obstáculos en el camino que deben de rodear.

Abajo se muestra el resultado final del algoritmo en función, tomando en cuenta que el personaje, solo puede moverse sobre los cuadros azules, cuadros verdes claro son obstáculos (el personaje no puede pisarlos) y cuadros verdes oscuro es la senda que el personaje ha tomado.

7.5 Descripción del proyecto

En este caso definimos las vistas, en específico y en orden de desarrollo:

1.- Menú de inicio

1.1- Diseño UI

1.2- Funcionalidad

1.3 – Assets

2.- Menú de configuración

1.1- Diseño UI

1.2 Funcionalidad

1.3- Assets

3.- Juego.

3.1- Mapa táctico

3.2- Personajes

3.3- Funcionalidades (Movimiento, animaciones habilidades)

7.6 Reglas de uso

Se le enseñará al jugador a usar las funcionalidades como:

- Participar en el juego:
  - Mover sus unidades. El jugador debe ser capaz de mover sus unidades en el mapa.
  - Atacar unidades enemigas. El jugador debe ser capaz de instruir a sus unidades a atacar las unidades enemigas en el mapa.
  - Las condiciones de victoria. El jugador debe ser capaz de reconocer las condiciones de victoria en el mapa.
  - Las condiciones de derrota. El jugador debe ser capaz de reconocer las condiciones de derrota en el mapa.
- Guardar avances. El jugador debe de ser capaz de guardar el estado del juego en un archivo que sea legible por el juego en sesiones futuras.
- Cargar avances. El jugador debe ser capaz de recuperar el estado del juego guardado en una sesión previa.

8 Conclusiones

Se han alcanzado los objetivos generales: (como es evidenciado en los anexos mostrando los gráficos de terreno usados, personajes, movimiento, codigo, etc) especificados al inicio del documento. Y de forma exitosa a producir un prototipo del producto, lo que sugiere que la metodología es suficiente para establecer las bases para el desarrollo independiente de videojuegos.

La metodología creada, ofrece una base para empezar a desarrollar el proyecto, desde las herramientas a usar y las habilidades de los desarrolladores. También se ha permitido definir la visión y el alcance del proyecto, en términos artísticos, tecnológicos y financieros, lo que introduce la disciplina personal o de equipo, según sea el caso, requerida para aumentar las probabilidades de éxito del proyecto.

Durante el desarrollo de la metodología propuesta, hubo varios retos y lecciones aprendidas, entre ellas:

- Generar diseños estéticos (valores) Conforme vayan siendo necesarios. Un juego es un proyecto extenso y sujeto a cambios. Por este motivo, si bien sería más organizado crear todos los diseños que esperamos implementar en el juego.
- Hacer lo más complicado primero. De esta manera cualquier error será encontrado más prontamente y se tendrán soluciones lo más temprano posibles. Se tendrá: una primera versión muy tosca, pero conforme el proyecto avance, no habrá necesidad de regresar a repetir el trabajo nuevamente. Cabe aclarar que esto es en base a un estimado de “dificultad”, que depende de la experiencia de los desarrolladores y las estimaciones que se hagan antes de empezar.
- Buscar y encontrar herramientas fue relativamente fácil, pero también fue requerido evaluarlas y para lograr una evaluación válida se ha de determinar: cuáles son las características que ofrecen más valor para el equipo de desarrollo, tomando en cuenta las limitantes que ya se habían definido.
- Ser fieles al concepto central del proyecto, pero no estar apegados a todo lo que le rodea. Con esto se expresa que deben de estar preparados para hacer cambios según nos llegan las retroalimentaciones, pero esto no requiere que tengamos que redefinir todo el proyecto. Parte de lo que les da su valor a los proyectos independientes, es que tienen la flexibilidad de tomar riesgos e intentar cosas nuevas, con las que el usuario puede que no esté familiarizado.
- Es imposible no iterar y esperar lograr un resultado de calidad. Si bien el método “en cascada” es eficiente en términos de recursos, que son la limitante más importante del proyecto, es todavía más valioso obtener una retroalimentación de lo que se ha obtenido, para hacer cambios que sean convenientes.
- Así sea un proyecto relativamente humilde, es imposible terminarlo en un periodo menor a un año. Se requiere: diciplina personal y perseverancia para llevar el producto a término.
- Es necesario tener una mente abierta, sobre cualquier tipo de habilidad que sea requerida para poder completar el proyecto, ya sean habilidades artísticas gráficas, de sonido o ingenieriles: como es la codificación y aprender a usar herramientas de forma correcta aplicando las buenas prácticas.

9 Bibliografía

Adobe. (s.f.). *https://www.adobe.com/mx/products/photoshop.html?sdid=KQPQY&mv=search&ef_id=003c454a31d210a3b99827323ebbea89:G:s&s_kwcid=AL!3085!10!78958807024752!78958691240109*. Obtenido de https://www.adobe.com/mx/products/photoshop.html?sdid=KQPQY&mv=search&ef_id=003c454a31d210a3b99827323ebbea89:G:s&s_kwcid=AL!3085!10!78958807024752!78958691240109: https://www.adobe.com/mx/products/photoshop.html?sdid=KQPQY&mv=search&ef_id=003c454a31d210a3b99827323ebbea89:G:s&s_kwcid=AL!3085!10!78958807024752!78958691240109

Aseprite. (s.f.). *https://www.aseprite.org/*. Obtenido de https://www.aseprite.org/: https://www.aseprite.org/

Borromeo, N. A. (2020). Hands on Unity 2020 Game development. En N. A. Borromeo, *Hands on Unity 2020 Game development* (pág. 798). Brimingham: Packt Publishing Ltd.

Bradfield, C. (2018). Godot Engine Game Development Projects: Build five cross-platform 2D and 3D games with Godot 3.0. En C. Bradfield, *Godot Engine Game Development Projects: Build five cross-platform 2D and 3D games with Godot 3.0* (pág. 300). Packt Publishing.

Clevelantd Institute of Art. (04 de March de 15). *Clevelantd Institute of Art*. Obtenido de Clevelantd Institute of Art: https://www.cia.edu/blog/2015/04/the-elemental-tetrad-of-games

Cordone, R. (2019). *Unreal Engine 4 Game Development Quick Start Guide: Programming professional 3D games with Unreal Engine 4.* Packt Publishing.

Eclipse. (08 de 05 de 2022). *https://www.eclipse.org/ide/*. Obtenido de https://www.eclipse.org/ide/: https://www.eclipse.org/ide/

Felicia, P. (2020). Godot from Zero to Proficiency (Foundations): A step-by-step guide to create your game with Godot. En P. Felicia, *Godot from Zero to Proficiency (Foundations): A step-by-step guide to create your game with Godot* (págs. 30-33). Independently published.

Fullerton, T. (2008). *Game Design Woprkshop.* Danvers: CRC Press.

gamefromscratch. (26 de 11 de 2019). *Why Chose the Godot Game engine over Unity or Unreal Engine*. Obtenido de GameFromScratch.com: https://gamefromscratch.com/why-choosethe-godot-game-engine-over-unity-or-unreal-engine/

Gibson, J. (2015). Introduction to game design, prototyping and development. En *Introduction to game design, prototyping and development* (pág. 230). Upplr Saddle River: Addison-Wesley. gibson, J. (2015). Introduction to Game Design, Prototyping, and Development. En J. Gibson, *Introduction to Game Design, Prototyping, and Development* (págs. 237-238). New Jersey: Addison-Wesley.

Hammad Fozi, G. M. (2020). *Game Development Projects with Unreal Engine: Learn to build your first games and bring your ideas to life using UE4 and C++.* Packt Publishing.

help.bandlab.com. (01 de 05 de 2022). *https://help.bandlab.com/hc/en-us/articles/360036010533Supported-Import-and-Export-File-Formats-*. Obtenido de https://help.bandlab.com/hc/enus/articles/360036010533-Supported-Import-and-Export-File-Formats-: https://help.bandlab.com/hc/en-us/articles/360036010533-Supported-Import-andExport-File-Formats- *https://www.patreon.com/c/gaming*. (s.f.). Obtenido de patreon. *https://www.venturecapital.game/*. (s.f.). Obtenido de venturecapítal.game.

itch.io. (20 de May0 de 2022). *https://itch.io*. Obtenido de itch.io: https://itch.io/games/made-withunity/tag-pixel-art/tag-turn-based-strategy

itch.io. (s.f.). *Most Used Engines*. Obtenido de itch.io: https://itch.io/gamedevelopment/engines/most-projects jetbrains. (13 de August de 2021). *https://www.jetbrains.com/help/rider/Languages_and_Frameworks.html*. Obtenido de https://www.jetbrains.com/help/rider/Languages_and_Frameworks.html: https://www.jetbrains.com/help/rider/Languages_and_Frameworks.html

jetbrains. (s.f.). *https://www.jetbrains.com/store/#personal*. Obtenido de https://www.jetbrains.com/store/#personal: https://www.jetbrains.com/store/#personal

Lily. (07 de 2021). *Can I make a commercial game with Unity Free/Personal Edition?* Obtenido de support.unity.com: https://support.unity.com/hc/en-us/articles/205253119-Can-I-make-acommercial-game-with-Unity-Free-Personal-Edition-?source=search&auth_token=eyJhbGciOiJIUzI1NiJ9.eyJhY2NvdW50X2lkIjoyNzI3MDEsInVzZXJfaWQiOjQwMTgwMTc5MTUzMSwidGlja2V0X2lkIjoxMDAzNjkyLCJjaGFu lmms.io. (s.f.). *https://docs.lmms.io/user-manual/getting-started/faq*. Obtenido de https://docs.lmms.io/user-manual/getting-started/faq: https://docs.lmms.io/usermanual/getting-started/faq

Microsoft Corporation. (10 de December de 2021). *Microsoft Corporation*. Obtenido de Microsoft Corporation docs web site: https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions

Resarch Optimus. (s.f.). *Resarch Optimus*. Obtenido de Resarch Optimus Web siter: https://www.researchoptimus.com/article/primary-research-methods.php

Roebolledo, R. A. (11 de 08 de 2017). *eleconomista*. Obtenido de eleconomista: https://www.eleconomista.com.mx/arteseideas/Cual-es-el-problema-de-la-fuga-decerebros--20170811-0069.html

Schell, J. (2019). *The Art of Game Design.* A K Peters/CRC Press; 3rd edition (August 27, 2019).

Toftedahl, M. (30 de 07 de 2019). *production*. Obtenido de gamedeveloper: https://www.gamedeveloper.com/production/which-are-the-most-commonly-usedgame-engines-

Toftedahl, M. (30 de 09 de 2019). *Which are the most commonly used Game Engines?* Obtenido de gamasutra: https://www.gamasutra.com/blogs/MarcusToftedahl/20190930/350830/Which_are_the_most_commonly_used_Game_Engines.php

unity. (18 de 04 de 2019). *https://docs.unity3d.com/Manual/ScriptingToolsIDEs.html*. Obtenido de unity3d.com: https://docs.unity3d.com/Manual/ScriptingToolsIDEs.html

Unity. (s.f.). *https://support.unity.com/hc/en-us/articles/206484803-What-are-the-supportedAudio-formats-in-Unity-*. Obtenido de https://support.unity.com/hc/enus/articles/206484803-What-are-the-supported-Audio-formats-in-Unity-: https://support.unity.com/hc/en-us/articles/206484803-What-are-the-supported-

Audioformats-in-Unity- *unity3d*. (s.f.). Obtenido de unity3d.com:

https://docs.unity3d.com/Manual/ScriptingToolsIDEs.html

unrealengine. (s.f.). *Get started with Unreal Engine*. Obtenido de unrealengine.com: https://www.unrealengine.com/en-US/download

Videojuegos, A. M. (2017). Obtenido de http://mexicogames.org/wp-content/themes/accesspressmag/images/MexicoGamesIndustry.pdf

Videojuegos, A. M. (20 de 02 de 2020). Obtenido de https://amexvid.com/2020/02/20/opcionesde-

funding-para-el-desarrollo-de-videojuegos/ *visualstudio*. (s.f.). Obtenido de

visualstudio.microsoft.com:

https://visualstudio.microsoft.com/es/vs/features/web/languages/ *visualstudio*.

(s.f.). Obtenido de visualstudio.microsoft.com/:

https://visualstudio.microsoft.com/es/free-developer-offers/

Wijman, T. (8 de May de 2020). *newzoo*. Obtenido de newzoo: https://newzoo.com/insights/articles/newzoo-games-market-numbers-revenues-andaudience-2020-2023/

Wikipedia. (s.f.). Obtenido de https://en.wikipedia.org/wiki/Game_engine

Wikipedia. (2022). Obtenido de https://en.wikipedia.org/wiki/Kerbal_Space_Program

Wikipedia. (s.f.). *https://es.wikipedia.org/wiki/Microsoft_Paint*. Obtenido de https://es.wikipedia.org/wiki/Microsoft_Paint: https://es.wikipedia.org/wiki/Microsoft_Paint

Xantomila, J. (27 de 09 de 2020). *La jornada*. Obtenido de jornada.com.mx: https://www.jornada.com.mx/ultimas/sociedad/2020/11/27/probable-aumento-de-fugade-cerebros-debido-a-crisis-cedat-8609.html

Anexos.

Anexo 1. Cuestionario.

Se usará una encuesta para diagnosticar si el producto es viable o no:

1.- ¿Consideras qué el “gaming” es su forma principal de entretenimiento? (Si o No)

1.1.- ¿Podría compartir su país?

1.2.- ¿Podría compartir su sexo?

2.- ¿Qué género de videojuego prefiere? (ROL, estrategia, rompecabezas, plataformas, multijugador, etc.)

3.- ¿Usualmente cómo determina qué juegos comprar?

a) Revisión de usuarios

b) Revisión de críticos profesionales

c) “Let’s plays”

d) Revistas informativas del tema

e) Recomendación de conocidos

d) Otro _____

4.- ¿Qué opinaría de un desarrollo de un juego independiente que combina un solo jugador, rol y elementos de estrategia?

5.- ¿Qué sería más importante para usted en su juego? (valor de rejuego, dificultad, historia, visualización, etc.)

6.- ¿Esta propuesta le suena a algún otro juego en el mercado?

6.1- Por favor menciónala y da una descripción breve de su opinión al respecto.

Anexo 2. Gráficos

Terreno

Estos son los gráficos usados para determinar el tablero de juego. Se usa cada uno de estos para ocupar una celda en específico.

Figura Anexo 2.1 Terreno – Diseño de terreno

Personajes

Se han dibujado los gráficos de 2 personajes, un caballero y un mago, cada uno viendo a 4 direcciones, respectivamente, al frente a la izquierda, frente a la derecha, atrás a la izquierda y atrás a la derecha.

Caballero

Figura Anexo 2.2 Caballero – Diseño de personaje caballero

Mago

Figura Anexo 2.3 Mago – Diseño de personaje Mago

Movimiento del personaje

Se muestra al personaje en su posición inicial al medio del tablero. Solo se puede mover sobre las casillas azules. No se puede mover sobre casillas sin terreno o las casillas verdes claro.

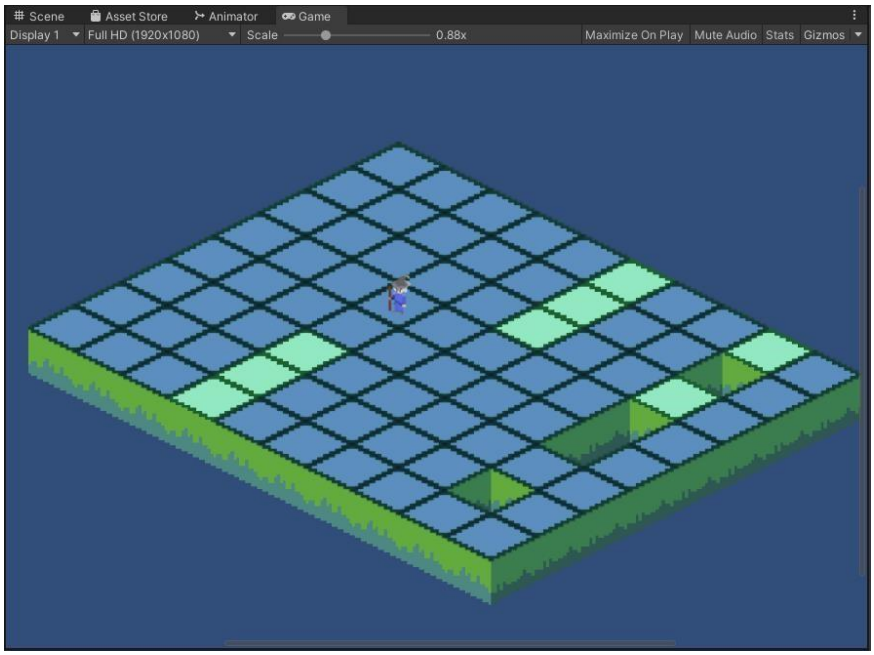

Figura Anexo 2.4 Movimiento de personaje – Posición inicial

Se ha decidido mover al personaje a la esquina de la derecha.

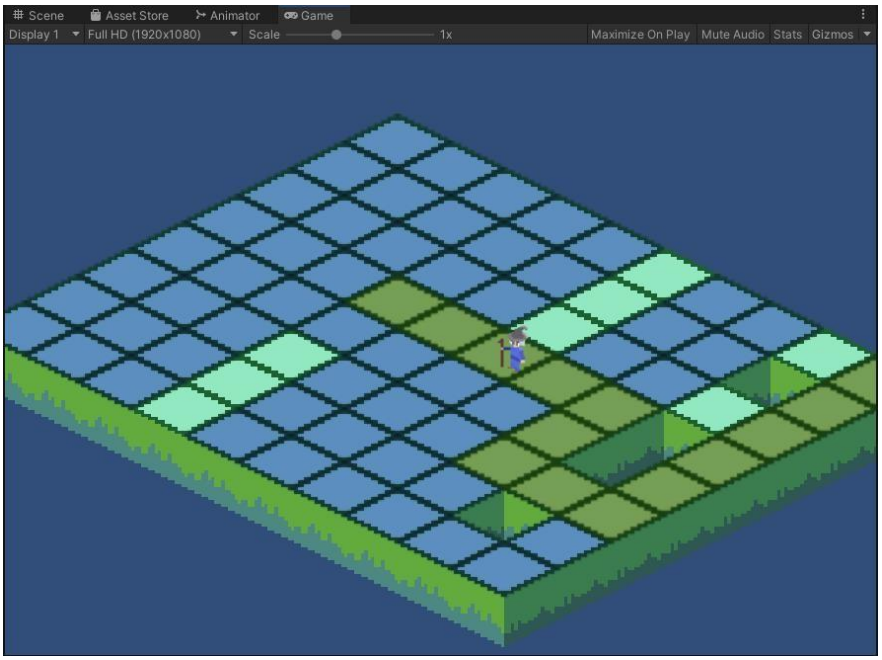

Figura Anexo 2.5 Movimiento de personaje – Movimiento 1

El algoritmo A* determina el camino más corto para nuestro personaje a su destino, y viaja a través del camino, mostrado aquí como un sendero verde obscuro.

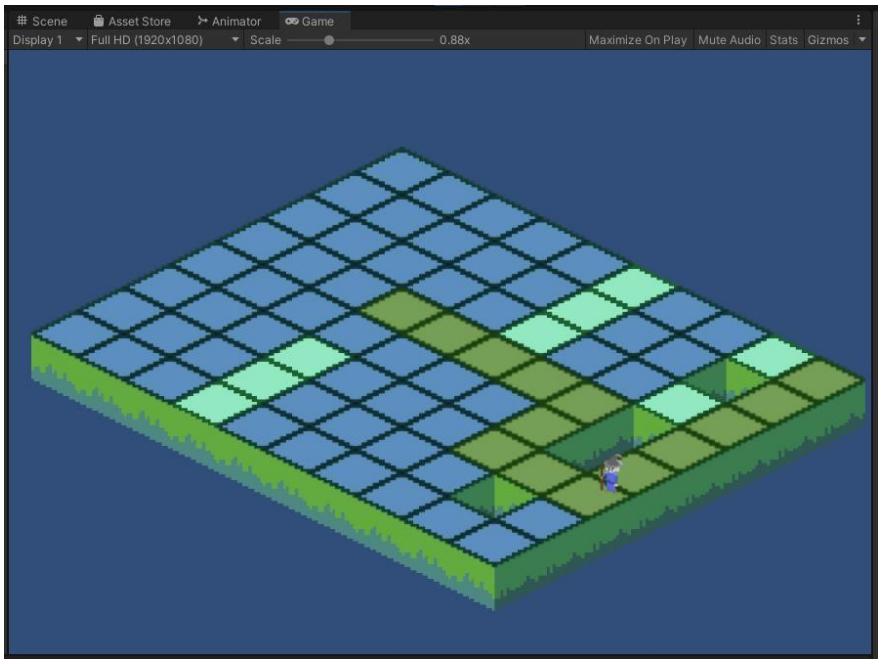

Figura Anexo 2.6 Movimiento de personaje – Movimiento 2

Finalmente, el personaje se detiene al llegar a su destino y se muestra el camino que ha seguido.

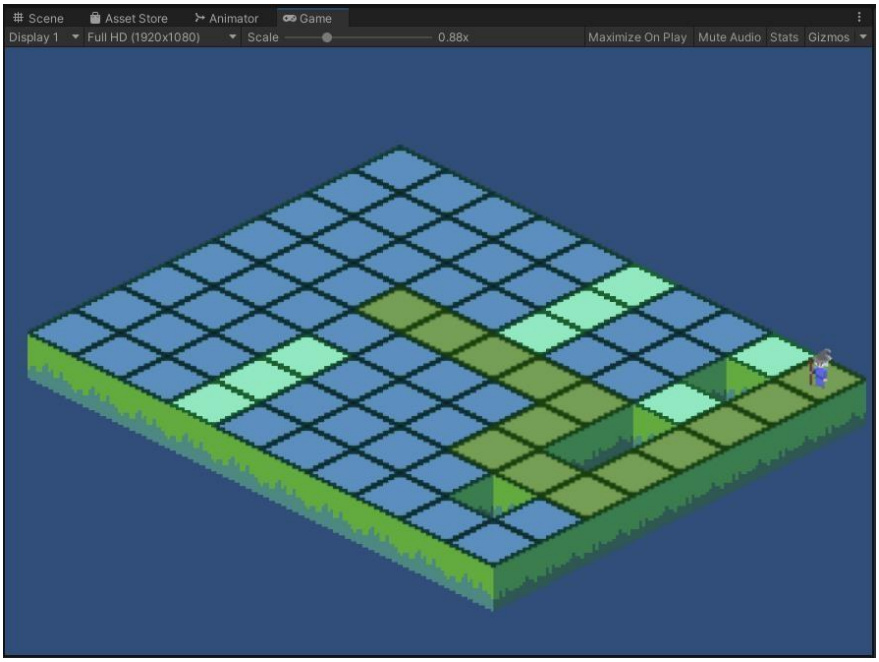

Figura Anexo 2.7 Movimiento de personaje – Posición final

Anexo 3. Librerías externas.

Nombre de librería	Descripción
API updater	Para mejorar la usabilidad y el rendimiento, Unity puede cambiar la forma en que clases, funciones y propiedades funcionan. Para minimizar el impacto de estos cambios API updater identifica y cambia el código obsoleto en los spcrits y ensamblajes.
GridBrush	Herramienta de cepillo usada para pintar o borrar losas y/u objetos en una cuadricula.
PackageManager	Manejador de paquetes para los registros del alcance y modificación de las configuraciones avanzadas del proyecto.
UIElements	Es un módulo que implementa el marco de trabajo de diseño de UI.

Tabla Anexo 3.1 – Librerías externas