PLUG-IN DE INTELIGENCIA ARTIFICIAL PARA NPC EN UN JUEGO RPG DESARROLLADO EN UNREAL ENGINE 4

David Santiago Reina Rojas

Erikson Javier Romero Hernández

Proyecto de grado

Directores:

Ing. Wilman Helioth Sánchez. M.Ed.

Ing. Álvaro Joffre Uribe, Ph.D.

UNIVERSIDAD MILITAR NUEVA GRANADA

FACULTAD DE INGENIERÍA

INGENIERÍA EN MULTIMEDIA

BOGOTÁ

2021

Plug-in de inteligencia artificial para NPC en un juego RPG desarrollado en Unreal Engine 4

David Santiago Reina Rojas

Erikson Javier Romero Hernández

Trabajo de Grado para optar por el título de:

Ingeniero en Multimedia

Dirigida por:

Ing. Wilman Helioth Sánchez. M.Ed.

Ing. Álvaro Joffre Uribe. Ph.D.

Universidad Militar Nueva Granada

2021

Para todos aquellos que mediante la ingeniería quieren buscar una mejor calidad de vida para las generaciones futuras.

David Reina – Erikson Romero

Agradecimientos

A nuestros compañeros y familiares que con el paso de los días y los meses nos incentivaron a tener un trabajo constante y no desfallecer en el desarrollo de este proyecto que en ciertos momentos presento algún obstáculo.

A los maestros Wilman Helioth Sánchez y Álvaro Joffre Uribe que, con sus recomendaciones y apuntes, en conjunto de un constante acompañamiento que llevaron a completar este proyecto de una manera adecuada con muy buenos resultados.

Al programa, el cual, con la enseñanza continua a lo largo de las diferentes asignaturas, permitieron desarrollarnos como profesionales íntegros y éticos que están en la capacidad de abordar un problema de esta índole.

A la universidad y laboratoristas por prestar las instalaciones y equipos que apoyaron nuestro aprendizaje en el transcurso de nuestro tiempo de educación.

Resumen

En el desarrollo de los videojuegos de tipo RPG (Role Play Game por sus siglas en inglés), existe una amplia cantidad de personajes no jugables (NPC por sus siglas en inglés), que en algunas ocasiones no destacan por su inteligencia artificial (IA), llevando a cabo comportamientos incongruentes, lo cual puede romper la inmersión y la jugabilidad. Este tipo de situaciones puede dañar el flujo de la partida en el momento que el jugador interactúa con el NPC.

La configuración de la IA de cada NPC varía de acuerdo a cada título, género, o fin al que se quiere mostrar en el juego, para esto, dentro de los motores de desarrollo de videojuegos, se implementan herramientas que realicen la implementación de esta característica al NPC. Unreal Engine hace manejo de árboles de comportamiento propios, los cuales se encargan de realizar una acción, dependiendo de un conjunto de entradas que son recopiladas por el ambiente y definidas por el desarrollador.

El presente proyecto plantea el diseño e implementación de un complemento de IA para el motor de videojuegos Unreal Engine 4, ya que este editor permite un gran procesamiento visual y de desarrollo, que facilita la implementación de este y futuros proyectos de mayor envergadura; haciendo uso del árbol de búsqueda de Monte Carlo (MCTS por sus siglas en inglés), que permite generar una serie de decisiones aleatorias, escogiendo la mejor a partir de los cálculos basados en simulaciones aleatorias que agilizan la toma de posibles acciones.

Para validar la implementación propuesta en este trabajo de grado, se generó un ambiente de pruebas de tipo RPG para toma de decisiones, acotado por un escenario de casillas con mecánicas basadas en combates por turnos, donde cada NPC cuenta con los conceptos armamentísticos derivados de las antiguas legiones romanas, conceptualizando las funciones de cada legionario junto al equipamiento que los acompañaba durante cada batalla, en pos de dinamizar estos comportamientos de manera lógica, incluyendo un esquema basado en variables que definen una personalidad, junto a paquetes de estado que identifican las cualidades de miedo, fe y venganza que derivan de las acciones del jugador y afectan el comportamiento del actor.

Lo que lleva a corroborar la implementación con un sistema de tres pruebas, las cuales cuentan con distintos entornos construidos en base a la cantidad de NPC presentes en escena, junto a la ubicación de los mismos alrededor de la escenografía y del jugador, donde se clasifican tres tipos de NPC que se diferencian por sus valores de personalidad, los cuales son el IQ (del alemán *Intelligenzquotient*, conocido como *coeficiente intelectual)*, miedo, valentía, venganza, locura, liderazgo y confianza, para así, formar un carácter a cada decisión que se deba tomar. Generando rangos basados en dichas variables que representan el nivel bajo, básico y avanzado, a los cuales hará frente al jugador, con la finalidad de analizar los árboles generados y sus comportamientos dependiendo de dichas configuraciones, arrojando datos que evidencian las relaciones entre actores de tipo NPC, y la varianza en su actitud para afrontar diferentes acciones realizadas por la persona que los confronte.

Índice

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

La historia de los videojuegos inició a mitad del siglo XX, donde todo partió con el desarrollo realizado en Tennis for Two, que sirvió de inspiración para el juego conocido como Pong, en donde el físico nuclear William Higinbotham presentó un juego en el cual dos competidores dirigen el rebote de un punto verde sobre una pantalla negra, mediante dos controladores conectados a un osciloscopio de cinco pulgadas (Bossom et al., 2016). De esta época en adelante, la industria de los videojuegos se mantiene en constante crecimiento, aún más con la tasa de incremento en la distribución digital (Clement, 2021), y esto acompañado de un crecimiento monetario que en el año 2006 representó en la industria norteamericana un valor de más de $12.5 miles de millones de dólares (Wolf, 2008, p. 1), y en el año 2019 recaudó $120.1 miles de millones de dólares (Laguna, 2020), suma que se puede comparar con la de otros medios de entretenimiento masivos, haciendo más relevante este rubro dentro de la economía actual (Malim, 2018).

Dentro del desarrollo de proyectos de esta índole, el uso adecuado de los elementos para diseñar un videojuego, como lo son el jugador, los objetivos, las reglas, los conflictos y los límites de este, pueden enriquecer o romper la inmersión del jugador con respecto a su experiencia, por lo cual, para mantener el jugador interesado, se debe balancear la dificultad y habilidad requerida para completar los objetivos. Esta relación es directamente proporcional y adecuada para mantener un flujo constante de emociones e inmersión, ya que una habilidad alta y un reto bajo generan aburrimiento, y una habilidad baja y un reto alto generan ansiedad, este concepto se conoce como “Flow” (Baron, 2012).

Identificar el público objetivo, considerando un rango de edades, temáticas y tonalidades asociados a las mecánicas, gráficas o historia, juegan un papel importante para determinar las mecánicas y respuestas del juego. Como parte del desarrollo de videojuegos, se cuenta con herramientas como la IA (Rockstar, 2021), la cual es utilizada en su mayoría para dotar de comportamientos lógicos a NPC. Esto permite que la experiencia jugable tenga un mayor dinamismo e involucre más a los jugadores con su entorno digital, ya que el jugador detecta un comportamiento creíble para verse inmerso en el mundo del juego, gracias a la investigación realizada por los métodos de IA (Gunn et al., 2009).

En algunos casos, el entorno y flujo de acciones bajo el cual se encuentran los actores, evidencia comportamientos que realmente no demuestran lógica por parte de las acciones del NPC, repercutiendo en la experiencia de usuario y las características que el desarrollador quiera darle a estos individuos virtuales (Yannakakis et al., 2015). Hay varios casos en los cuales los personajes están diseñados para tener un comportamiento lineal, ya que se encuentran basados en máquinas de estado finitas, que direccionan la respuesta mediante una acción específica del jugador. Con el paso del tiempo nuevas lógicas que complementan y toman como base este concepto, han sido añadidas a fin de brindar una aleatoriedad que interactúe de mejor manera con el jugador y con un comportamiento más orgánico (Shummon et al., 2019).

La falta de esta aleatoriedad genera ataques y acciones repetitivas, sin importar el entorno o recursos para el NPC, como se observa en Galaga, videojuego en el cual, los enemigos siguen un patrón de ataque establecido sin varianzas (Véase figura 1). Sin embargo, los patrones con pocas

variaciones, han sido reemplazados por aquellos más complejos con acciones robustas, a fin de representar un mayor reto con respecto al jugador, como se puede observar en el videojuego For Honor, en el cual el NPC busca mediante sus acciones romper la guardia del jugador para poder acertar un ataque (Véase figura 2) (Vossen, 2015).

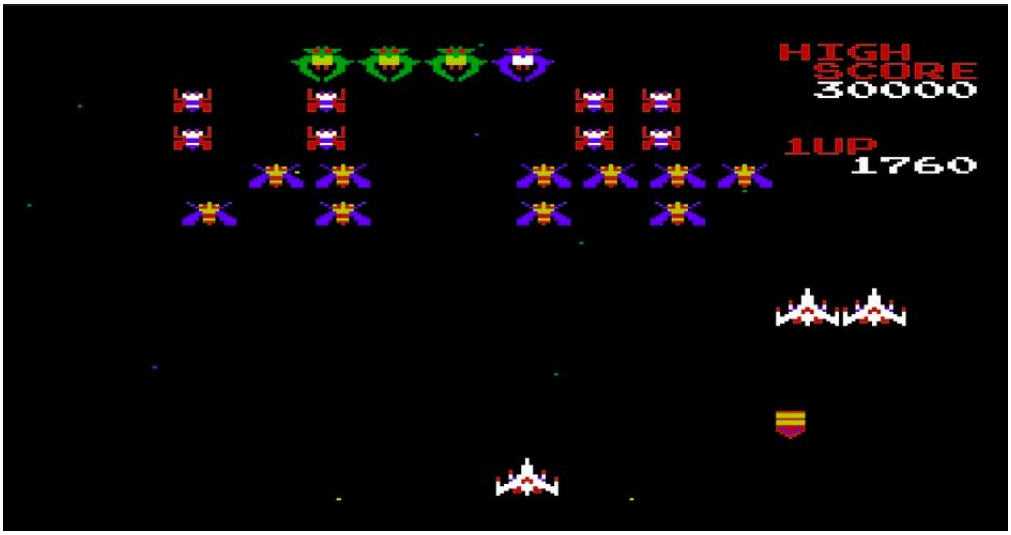

Figura 1. Galaga. Representación de las mecánicas del combate espacial y los ataques enemigos. Recuperado de https://galaga.fandom.com/wiki/Galaga?file=Galaga_NES.png

Figura 2. For Honor. Combate cuerpo a cuerpo. Recuperado de http://gameaccessibilityguidelines.com/for-honor-ui-contrast/

Algunos desarrolladores han optado por el uso de metodologías heurísticas de IA para la búsqueda de acciones a realizar, como lo son los árboles de comportamiento. Los arboles permiten realizar una toma de decisiones bajo ciertas circunstancias en un ámbito jugable de manera

controlada, diversificando la cantidad de acciones que un NPC puede realizar. Como resultado, se puede generar una amplia cantidad de problemáticas y retos a superar a nivel de máquina, lógica y credibilidad del organismo NPC, (Adamchik, 2009).

El mayor problema presente en los árboles de comportamiento es lograr generar acciones aleatorias que el jugador no espera y que a su vez tengan sentido (Akhtar, 2019). Dentro de estos, existen algunos como el MCTS, cuya heurística logra prever acciones con el paso del tiempo. Este proyecto busca hacer uso de este árbol de comportamiento, con el fin de implementar un complemento que permita generar el comportamiento de un NPC, facilitando la creación de NPCs por parte del desarrollador, y evaluar mediante pruebas y test en un ambiente controlado.

Conceptos fundamentales

Videojuegos

La forma de representar un juego de manera audiovisual, está basado en una historia que es acompañada de mecánicas que permiten interactuar con el escenario y tener una inmersión en la jugabilidad (Esposito, 2005). Acompañado del componente virtual, que es gobernado por la máquina que es representada tanto por el hardware como el software que permite la realización del concepto, siendo esta quien se dedique a presentar el contexto que se muestra en animaciones y gameplay a fin de brindar la diversión que se busca para estos casos (Wolf, 2008) .

Los elementos característicos de un videojuego, que se identifican según (Nacke, 2014) como:

- El jugador como el personaje que se ve inmerso en la partida y dependiendo de su rol se pueden afectar sus acciones durante esta.
- Los objetivos que tienen la capacidad de presentar un reto al jugador a fin de engancharlo y ofrecer cierto nivel de dificultad, donde podemos encontrar diferentes tipos de estos, como lo son la captura de un objeto o u terreno, cazar a un jugador o enemigo, ganar una carrera con el fin de alcanzar una meta antes que alguien más, rescatar o escapar de unidades u objetos, construir, mantener o manejar objetos del juego, explorar zonas nuevas para el jugador y solucionar un puzle que se presente en escena.
- Las reglas son las que especifican los procedimientos que los actores y objetos pueden o no realizar, que se definen en el inicio del juego, durante su progresión, si hay un evento especial en el estado del juego y en las resoluciones que llevan a cumplir los objetivos del juego, lo cual va ligado a los recursos que posea y afecten al actor de manera directa, tal como la vida, el inventario, el tiempo, etc.
- Los conflictos que se ven influenciados por los procedimientos, reglas y objetivos, previenen que un jugador alcance una meta, comúnmente se ven referenciados como obstáculos, oponentes y dilemas que aparecen en la historia.
- Los límites del juego, establecen espacial y conceptualmente, hasta dónde puede el juego llegar a abarcar las situaciones y acciones dentro de este.

RPG

Role-Playing Game, son un modelo de juego, en el cual el jugador asume el rol de un personaje que se encuentra preestablecido por el juego, y permite diferentes opciones de personalización, como, por ejemplo, en cuanto a armamento y armadura. Adicionalmente, la narrativa permite la ramificación de acuerdo a ciertas decisiones (Liberty, 2017).

NPC

Non-Player Characters, son aquellos personajes que están presentes en el entorno virtual pero carecen de control por parte del jugador, siendo esto realizado por la máquina (Warpefelt, 2016), junto a esto se ha determinado que la falta de inteligencia o de personalidad dificultan la

inmersión en el juego hacia el jugador (Loyall, 1997), intentando ofrecer opciones lógicas para mitigar este defecto o falla.

Máquinas de estado finitas

En el área de la ciencia computacional se manejan modelos basados en estados que pueden ser activados uno a la vez, y que de igual manera tienen una secuencia de entradas que son enviadas a la máquina para que esta salida se de en respuesta a un estímulo que llega (Bors, 2018).

Flow

Para el desarrollo de videojuegos se considera tanto la habilidad del jugador, como la dificultad del reto, para generar un balance entre ambos, evitando generar aburrimiento o ansiedad que afecten la experiencia de usuario e interrumpa la inmersión, en una relación directamente proporcional entre estos componentes, siendo esto el equilibrio entre estas dos vertientes (Baron, 2012).

Unreal Engine

Es conocido como un motor de videojuegos altamente potente, usado en gran cantidad por los desarrolladores acorde a las necesidades de sus proyectos (Denham, 2020), cuenta con capacidades de manejar eventos, interfaces y otros aspectos necesarios mediante el uso de programación a partir del lenguaje C++ o de los blueprints con los cuales cuenta, siendo estos bloques, funciones predeterminadas que facilitan el desarrollo al momento de poder compactarlos para poder formar un componente que tenga un amplia rango de desempeño acorde a sus necesidades prácticas, junto a esto se encuentra su desarrollo multiplataforma que genera un amplio mercado para los usuarios facilitando el despliegue de sus proyectos siendo este aplicado a consolas o a equipos de cómputo (Epic Games, 2004).

Inteligencia artificial

Inicialmente, para lograr comprender lo que abarca la IA, se puede remontar dicho concepto a lo que la inteligencia se refiere (Kaplan, 2016, p. 5), abstrayendo dicho concepto al mundo computacional, en donde se busca dotar a las máquinas de una inteligencia o razonamiento semejante al humano, aprendiendo y actuando en base a su entorno (Gambus et al., 2018).

Agentes inteligentes

En los contextos de la IA, la percepción del ambiente es un factor crucial para poder tener la toma de decisiones de una manera oportuna y precisa al momento que se identifique una posible acción, es por esto que en el medio computacional se aplica el concepto de agente inteligente, a fin de realizar una presentación semejante a como un humano percibe el mundo mediante sus sentidos, percibiendo lo que su entorno comprende y facilitando la elección, por medio de sensores con los que interactúa con otros actores (Mbaabu, 2020), todo esto buscando que su comportamiento se realice de manera ideal frente al obstáculo general y los posibles problemas que pueda llegar a encontrar en el camino.

Técnicas duras

Son conocidas como las aproximaciones que recibe la IA mediante el uso de árboles de búsqueda, que enfatizan la selección de una acción sobre las demás, y que están basados en el emparejamiento de patrones junto a un sistema basado en reglas (Cant et al., 2001).

Técnicas suaves

A diferencia de las técnicas duras, se genera una aproximación probabilística que busca afrontarse a problemas de la vida, emulando el no tener una respuesta 100% correcta, ya que se guía a partir de una aleatoriedad total o parcial para los números que fomentan sus decisiones, siendo visto en ejemplo como redes neuronales, algoritmos genéticos o técnicas evolutivas (Cant et al., 2001).

Árboles de comportamiento

Árboles de comportamiento eliminan la manera en cómo se ejecutaban los procesos ya definidos en las máquinas de estado para así mediante nodos intermedios, se pueda definir el orden de pasos a ejecutar, retornado en cada iteración si en ese nivel define un estado de éxito o de fracaso (Sagredo Olivanza et al., 2014).

MCTS

Dados los diferentes tipos de árboles de comportamiento, que se concentran en optimizar cierto tipo de funciones para adaptarse mejor a algunos escenarios, se encuentra el Monte Carlo Tree Search, el cual a partir de una función matemática logra dar una aproximación de simulación óptima para los estímulos que recibe tomando como elección de expansión en sus nodos raíz la opción más prometedora, y teniendo en consideración que el algoritmo funciona correctamente bajo las condiciones de: “1) La puntuación del juego está acotada; 2) Las reglas son conocidas y 3) Las simulaciones terminan relativamente rápido (la longitud del juego es limitada)”, (Embid, 2012, p. 19).

Antecedentes

En esta última década, la industria de los videojuegos ha experimentado un crecimiento exponencial y un movimiento empresarial gigantesco, por lo que, la tecnología ha avanzado significativamente, en particular en lo referente a procesamiento gráfico, como resultado se ha obtenido realismo en las imágenes renderizadas en tiempo real. Otros campos, como la IA han aumentado como resultado de la capacidad de procesamiento en paralelo (Statt, 2019).

El lento avance de la IA fue expuesto por (Kelleher et al., 2011), donde justifican la recurrencia de NPCs predecibles a la vista de los jugadores. Adicionalmente desarrollan una demostración de un juego de disparos en primera persona, donde incorporan un NPC que evolucionaba con el progreso del jugador. El uso de diferentes técnicas para alimentar al agente para permitir que el NPC fuera capaz de confrontar a un jugador real, dando como resultado una serie de información que ayudó a concluir que la suma de muchas técnicas de IA pueden llegar a ser la forma más eficiente de generar NPCs autónomos, creíbles y adaptativos.

En el campo de la IA, se hace uso de técnicas suaves y duras para generar un NPC autónomo y adaptativo (Černý et al., 2017). Por ejemplo, el manejo de objetos y áreas inteligentes dentro del desarrollo de juegos de mundo abierto, funciona como interruptores para activar eventos del NPC, en el cual encapsulan varios árboles de comportamiento con la finalidad de efectuar una acción determinada, pero de manera aleatoria, y completamente basada en el tiempo, y el enriquecimiento de la representación del NPC dentro del juego.

Los arboles de comportamiento han sido una gran alternativa dentro del desarrollo de NPCs, pues logran dar aleatoriedad dentro del juego emulando la intuición humana, basándose en decisiones resultantes de sus árboles de comportamiento (Tomai et al., 2014). Donde se realizó un desarrollo de NPCs en un juego Massive Multiplayer Online Role-Playing Game (MMORPG), en el cual encontraron que estos algoritmos, pueden ser capaces de procesar acciones realizadas por jugadores, adaptando el árbol de comportamiento y cambiando sus acciones en base a estas.

Las variables de un NPC determinan su personalidad y la habilidad de responder a los estímulos emitidos por las acciones de un jugador. Como resultado el NPC capta estas acciones en un tiempo concreto, de forma que el árbol de comportamiento como el MCTS responde de manera adecuada (Kim et al., 2017). Se añadieron tablas de acciones, en las cuales se controlaba un combate entre NPC, y clasificado qué acciones realizar en base a lo que otras IA combatían, dando maravillosos resultados puesto que su implementación obtuvo el segundo puesto en el torneo internacional de IA de la IEEE CIG del año 2016.

Planteamiento del problema

La IA es una ciencia computacional que busca emular tareas y características humanas por medio de hardware y software, como la toma de decisiones dependiendo de conocimiento adquirido a lo largo de un proceso planeado por una heurística (Lucci et al., 2009). Para el caso concreto de la IA en videojuegos, se debe tener en cuenta que esta se aplica para generar entornos más creíbles y comportamientos casi humanos a los NPC.

La implementación de un NPC, se ve afectada por las decisiones que se toman como respuesta a las acciones realizadas por el jugador, lo que en ocasiones afecta su percepción de la actitud frente a este. En caso de tomar un accionar muy semejante a la de un jugador, esto se denomina como un comportamiento de valle inquietante, dentro del cual el parecido llega a molestar, por la falta de realismo, resultando en un bajo nivel que no cumple con las expectativas (Sangyeob Lee et al., 2015).

En pos de cumplir con mejores niveles de comportamiento, los motores de videojuegos han implementado mejores formas de representar la toma de decisiones, a fin que los desarrolladores hagan uso de estos conceptos en sus proyectos. Así mismo, Unreal Engine 4 hace manejo de árboles de comportamiento que se basan en máquinas de estado finito, con el objetivo de realizar una acción rápida (Epic Games, 2021), esto complementado con otras herramientas como explica (Newton et al., 2016):

- NavMesh o mallas de navegación, las cuales delimitan las áreas sobre las cuales los NPC se pueden mover.
- Path Following o trazado de rutas, que define la zona sobre la cual la IA puede realizar un desplazamiento.
- Steering behaviors o comportamiento de dirección, en donde se define una cualidad bajo la cual se realizará todo el desplazamiento, como un movimiento directo o la búsqueda de coberturas durante el trayecto.
- Sistemas de sensores, los cuales están atentos a los cambios del ambiente, mediante la percepción de jugadores cercanos, niveles de sonido, cobertura próxima, etc.

Estas herramientas ayudan en el desempeño del NPC para ejecutar una acción determinada, de forma que la activación de estos comportamientos en el árbol, se efectúen para el momento designado. Ahora bien, la implementación de un algoritmo MCTS en ciertos proyectos, ayuda a seleccionar mejores movimientos basados en el estado de juego, algo que mejora el nivel de toma de decisiones, pero que específicamente para Unreal Engine 4, no está implementado de forma nativa, lo que conduce a usuarios del motor a realizar desarrollos por su parte y aplicarlos a su propia solución.

El presente documento se enfoca en, ¿Cómo diseñar un plug-in que permita introducir MCTS a la IA de un NPC, para poder elegir acciones basadas en la personalidad impuesta a este y que sirvan en un juego RPG para el motor de videojuegos Unreal Engine 4?

Justificación

En el mundo de los videojuegos, el comportamiento de los NPC afecta el éxito del juego, por ejemplo, en *The Elder Scrolls Skyrim*, los NPC no se destacan por tener una lógica acorde a las situaciones (Rodríguez, 2015), lo cual fue mejorado con una modificación hecha por terceros, que válida el estado del NPC en comparación al del jugador y con una personalidad que lo caracterice a fin de determinar su accionar (DOrchymont, 2018).

Este manejo no es aplicado a un desarrollo en específico, ya que puede ser desarrollado e implementado para uno o varios motores de videojuegos, a fin de mejorar los futuros proyectos con este tipo de desarrollos. Este es el caso del plug-in realizado por *Nvidia*, en el cual se aplica un concepto que ya había sido implementado en sus productos, como lo es el *Deep Learning Super Sampling* (DLSS), el cual escala la cantidad de fotogramas por segundo para mejorar la experiencia en videojuegos con definición 4K, y ahora puede ser utilizado dentro del motor Unreal Engine 4, lo cual permite que el uso de esta tecnología no se limite a una cierta cantidad de productos, sino a una mayor variedad de títulos (Sarkar, 2021).

En búsqueda de que un complemento para Unreal Engine 4, que pueda implementarse en una IA coherente a las mecánicas de un juego, que mejore las decisiones que tomara un NPC, este proyecto plantea diseñar e implementar un plug-in en el motor, que contemple las características de una IA para un juego RPG, permitiendo que el usuario genere NPC que puedan cumplir sus objetivos, y se acoplen a los lineamientos previamente establecidos por el desarrollador.

Objetivo general

Diseñar e implementar un complemento que introduzca IA personalizable, en personajes no jugables, en un juego de Acción RPG mediante la implementación del árbol de comportamiento en Unreal Engine 4 con MCTS.

Objetivos específicos

- Analizar e implementar la lógica de MCTS con árboles de comportamiento en Unreal Engine 4.
- Diseñar y desarrollar árboles de comportamiento inteligentes orientados a los enemigos en cuestión.
- Corroborar el funcionamiento de la IA mediante pruebas y test en un ambiente controlado.

Capítulo 2. Marco teórico

Para el desarrollo del proyecto, se tomó como base los juegos de tipo RPG basados en un sistema por turnos, acompañado de conceptos armamentísticos de las legiones romanas, a fin de utilizar diferentes niveles de daño y defensa como recursos del juego. En este capítulo se presenta un análisis con respecto a los videojuegos, la conceptualización del tipo RPG, manejo de IA en videojuegos de diferentes tipos, así de cómo ha sido su avance, el manejo de árboles de comportamiento, para lograr mejorar la toma de decisiones en las interacciones con el jugador, la implementación de MCTS, y por último, el inventario que acompañaba a los legionarios romanos en sus cruzadas.

RPG

Dentro del universo que son los videojuegos, se encuentra el género de los juegos Role-Playing Game, el cual se caracteriza por ofrecer al jugador un mundo en el cual él puede decidir qué papel desempeñar, mediante la creación o elección de un personaje, que se caracteriza por tener atributos que desempeñan un rol específico para cada manera de jugar. Dentro de estos atributos, comúnmente se encuentran especificaciones como la raza, especie, género, facción, labor, etc., junto a algunas habilidades como fuerza, velocidad, destreza, magia, etc., (Wolf, 2008), lo que con el tiempo detonó en una subdivisión que abarcara ciertos grupos y temáticas.

En torno a la conceptualización que se realiza para cada juego desarrollado, se encuentran varios sub géneros que cuentan con mecánicas que los diferencian entre sí, dentro de estos tenemos:

- Action Role-Playing Game: Este sub género se caracteriza por no detener los enfrentamientos mediante un sistema de turnos, basándose en combates en tiempo real, que forzan al jugador a realizar decisiones, ya que no cuenta con interfaces o ayudas tácticas visuales, que logren aportar información relevante durante el combate (Xbox Wire Staff, 2015).

- Turn bassed RPG: Para este apartado, se cuenta con características que emulan los conceptos dados por un juego de rol semejante a Dungeons and Dragons, en el cual se cuenta con un tablero y un sistema de turnos, que busca ofrecer un combate dividido entre personajes no jugables y jugadores (Hovermale, 2019).

- Tactical RPG: Es un género que consta de un sistema parecido al ajedrez, en donde el jugador debe buscar la forma de sobreponerse en combate al enemigo, en donde las tácticas y la información relevante del enfrentamiento, son mostrados en pantalla con el fin de buscar plantear la mejor manera para aniquilarlo, todo esto sin sufrir mucho daño y considerando que cada personaje cuenta con su rol respectivo (Grosso, 2015).

- MMORPG: Massively-Multiplayer Online Role-Playing Game, considerado como el ambiente en línea, bajo el cual mediante un RPG se pueden conectar masivamente varios usuarios, en donde dependiendo de las mecánicas se puede generar un PvP,

conocido como el player vs player, ó el PvE, conocido como Players vs Enviroment, que se traduce en un conjunto de jugadores que buscan realizar objetivos en grupo, a fin de avanzar en sus metas (Kaelin, 2006).

Inteligencia artificial

Desde los avances en desarrollo de lógica, la IA ha tenido en los últimos años una implicación cada vez mayor en el área de desarrollo de software, partiendo del punto de llegar a un sólo objetivo, siendo este, el dotar de inteligencia a agentes no vivos, y que ésta se acerque a la de un ser humano. Ese anhelo por crear software que introduzca procesos que solamente la inteligencia humana puede realizar es un reto, que se ha convertido en una actividad atractiva en la que muchos ya están dando sus primeros pasos, ya la IA abre un mundo de posibilidades a quien conoce su potencial, ya que proporciona un amplio conjunto de métodos, técnicas y algoritmos, que mediante su estudio exhaustivo y cuidadoso, pueden ser incluidas en distintas aplicaciones financieras, educativas, de seguridad informática, hasta videojuegos, entre otros (Johari, 2020).

Para el apartado de los videojuegos, se ha visto un crecimiento en el manejo de la IA a través del tiempo, aunque para los primeros títulos que fueron creados, no se tenían implementadas dichas características dada la complejidad y la forma en que estos títulos se basaban, siendo un juego para dos o más jugadores (Xu, 2018), pero con el paso del tiempo, los diseñadores vieron el potencial que implementar una IA abarca, por lo que fue aplicada en primer medida al juego de Atari *Computer Space* en 1970, para luego ser aplicada en demás juegos como *Pong, Space Invaders y Donkey Kong* (Xu, 2018), donde la elección no existía y era más seguir el patrón definido por el script que el desarrollador definió para sus NPC.

Logrando una conceptualización de IA que va ligada al manejo que se haga de esta, donde se concibe a partir del uso que se le dé durante su aplicación, y que en muchos casos no debe ser la mejor sino la más disfrutable, al intentar igualar el comportamiento del jugador para que este logre aprender y tener un Flow apropiado (Shummon et al., 2019). Según (Straeubig, 2019), el desarrollo de la IA aplicada en los videojuegos se puede ver clasificada en los siguientes aspectos:

- Mecánicas: Hacen referencia a las características que hacen parte tanto de las cualidades del NPC, como del manejo del entorno que se le da al juego, como lo pueden ser algoritmos para encontrar el mejor movimiento, movimiento de enjambres o multitudes, etc.
- Alter/Ego: Se define como la entidad que aparece en el mundo del jugador, simulando ser semejante al personaje, pero que es construido en base a algoritmos que simulan características que lo hacen familiar con relación al jugador.
- Observador: Caracterizada como la IA que se encuentra por fuera de la partida, mundo o del juego en sí, para lograr obtener conocimiento mediante observar el comportamiento del jugador en una situación específica.
- Protector: Se referencia como aquella entidad que busca hacer respetar las reglas del juego, para poder evitar jugadores que cometan fraude o hagan trampa alguna.

- Jugador: Busca reemplazar al jugador, a fin de aprender las mecánicas del juego al que se va a enfrentar por sí mismo.
- Creador: Es conocida como la aplicación que se basa en generar contenido acorde a las reglas del juego, que logren contar con sentido en su ejecución, y que es regida por una generación de contenido procedural, lo que se entiende como contenido generado por procedimientos.

Definido así cada aspecto que determina su rol dentro del juego, lo que además incluye la metodología bajo la cual será implementada la IA, teniendo la capacidad de ser o no de dominio supervisado, definido como la capacidad que tiene el problema para que se conozca su posible resultado antes de realizar la ejecución del algoritmo, en donde el supervisado se encuentra representado por árboles de decisión o clasificación, y para el no supervisado se cuenta con algoritmos como el clustering o agrupamiento, del cual no se conoce el resultado final sino hasta la finalización de su ejecución (Skinner et al., 2019).

A la par de la evolución de la IA en relación con los videojuegos, algoritmos destacables también se han enriquecido a lo largo del tiempo, como el Alpha Go, que según Jiang, este algoritmo busca generar un oponente capaz de ganarle a competidores humanos en el juego de mesa chino Go, pues en sus inicios realizaba búsquedas de posibilidades haciendo uso del el árbol de búsquedas minmax, pero opto luego por el MCTS, dando este último excelentes resultados al momento de ejecutar movimientos que lleven a la victoria, y es ahora, en su versión AlphaGo Zero, donde este algoritmo hace uso de aprendizaje reforzado por sí mismo, donde una red neuronal, aprende de jugar con muchos jugadores, en pos que escoger incrementalmente con MCTS la mejor opción para ganar una partida (2020).

Esta carrera de ciertas IA para superar la capacidad humana en ciertos juegos, no solo se puede notar en aplicaciones de juegos de tablero, sino también en otros más enfocados en juegos comerciales, como lo es la herramienta ViziDoom, que integra un agente que reemplaza al jugador en el video juego Doom clásico, el cual realiza primero una navegación por el mapa, para luego en una segunda fase, disparar a enemigos que observe. Al alternar entre una fase y otra este agente demostró ser muy superior frente a un jugador humano, realizando más asesinatos y muriendo menos veces en su partida (Jiang, 2020). Fue desarrollado haciendo uso un modelo de Reforzado profundo con QNetworks, el cual se basa en el uso de múltiples iteraciones, con el fin de que un agente aprenda una tarea específica, para luego validar los resultados haciendo uso de una función que premia o resta puntaje dependiendo de si logró o no sus objetivos (Alchalabi et al., 2019).

Árbol de comportamiento

A lo largo de los desarrollos que se han hecho de los videojuegos, se ha aplicado mejoras que evolucionan a medida que la generación de hardware crece, así como de nuevas ideas que se elaboran. En relación a los NPC se ha dado la aplicación de diferentes conceptos dentro de los cuales estaban las máquinas de estado finitas, que brindaban una posible selección y cambio de estado como su nombre lo indica, por parte del NPC al momento de recibir los estímulos

correspondientes (Pardo, 2013), aunque en el año 2004, con el lanzamiento del videojuego Halo 2, se apropió el uso de los árboles de comportamiento o behavior trees, siendo de los pioneros en el uso de estos (Torres, 2020). Esto llevó a una nueva forma de uso de la IA en los videojuegos, que actualmente es muy utilizada en la mayoría de videojuegos lanzados, tal como soporte en los motores de videojuegos como UE, Unity y CryEngine (Robertson et al., 2015).

Para que la IA funcione de acuerdo a los requerimientos dados por el diseñador, el programador aplica diferentes características de esta, a fin de ser implementada de una manera que sea disfrutable para el jugador, teniendo comportamientos lógicos en las mecánicas del juego que lleven a retener la inmersión en el mismo, acompañado de la toma de decisiones que está configurada para el NPC. Es por esto que al aplicar los árboles de comportamiento, se ejecuta el recorrido del árbol hasta sus hojas, sitio donde reside la decisión final y la toma de decisiones de cada nodo, a fin de seleccionar el mejor camino. Con la definición del camino ideal, se genera una serie de tareas que determinan una selección idónea a lo largo del árbol.

Durante la selección de las ramas, se priorizan dos tipos de misiones que harán parte del proceso de decisión para el NPC, para esto se tiene:

- Acciones: Estos nodos, están definidos a partir de los posibles cambios en el estado del NPC, realizando los movimientos viables a fin de realizar una aproximación a un comportamiento apropiado, de acuerdo a la lógica que se le brinda mediante el árbol, siendo estos los nodos terminales de todo el proceso.
- Condiciones: Están definidos como los nodos intermedios del procesamiento lógico, en donde se valida el estado del NPC en dicha secuencia, llegando a procesar las tomas de decisiones que se han efectuado en el nodo padre y enlazaran con el nodo hijo.

Es por esto, que a partir de la selección de cada nodo, sea por parte de acción o de condición, se da el estado mediante el cual se define el éxito o fallo de cada movimiento a realizar, validando si el proceso debe continuar o no por un número determinado de posibles iteraciones, a fin de desembocar en la opción más prometedora (Sagredo Olivenza et al., 2015). Con lo cual se define con este proceso, que para que la ejecución en los nodos condiciones o intermedios, se dé de una manera óptima, el uso de una serie de opciones como lo son:

Secuencia: Como su nombre lo indica, define una estructura en sus comportamientos a lo largo del proceso de ejecución del árbol, en donde se ejecuta los nodos y sus hijos, y que en caso de retornar éxito o success el nodo donde se encuentra, ejecuta al siguiente en cadena hasta finalizar con un resultado satisfactorio durante toda la secuencia, y de ser el caso opuesto, se comienza con la siguiente cadena.

- Selector: Dada su lista de comportamientos a ejecutar, se procesa dicha secuencia en cada nodo, en donde a partir de la selección y ejecución de estos, se define cada secuencia a partir del hijo con un resultado satisfactorio, a partir del posible éxito de un solo nodo durante la secuencia.
- Paralelo: Son procesos que se ejecutan en conjunto, pero no necesariamente son de manera estrictamente paralela o en varios núcleos, sino que se puede realizar de

manera entrelazada, en donde para definir el procedimiento de éxito se puede validar de manera secuencial o de selector.

- Selector con prioridad: De manera semejante a como se hace en el selector, se ejecutan las secuencias de nodos en donde el caso de éxito se define el proceso, pero con la diferencia con respecto al selector normal, de que cada nodo cuenta con su prioridad jerárquica con respecto a los demás, y en este apartado se ejecutaran dado la jerarquía con el que este cuente.
- Decoradores: Estos son nodos que se añaden bajo casos especiales, que se agregan a un nodo en donde este pueda tener solo uno de estos, añadiendo condiciones especiales, entre las cuales se encuentran añadir una condición adicional que valide el caso de éxito para el nodo, invertir el caso de resultado del nodo (de éxito a fallo, o viceversa), o repetir un número determinado de veces al nodo.

Donde dichos estos conceptos, se genera la lógica necesaria para ejecutar las decisiones que el árbol considera propicias, para manejar los NPC de manera coherente de acuerdo a sus percepciones.

Monte Carlo Tree Search

Dentro de la IA y los diferentes tipos de árboles, se encuentra el MCTS, el cual hace uso de diferentes aspectos como lo son la simulación, selección y retro propagación, haciendo manejo de cálculos matemáticos como el UCT (Upper Confidence Tree) o UCB_1 (Upper Confidence Bound), lo que define las decisiones oportunas a fin de un razonamiento que tienda a ser lógico y con mayor concordancia en las acciones realizadas.

Para lograr realizar dichas acciones lo que se implementa es una construcción de un árbol asimétrico y heurístico, denotando un crecimiento que se ahonda en aquellas acciones que tengan un mejor prospecto para el desarrollo del problema, y que, a su vez, es posible no dotarlo de gran conocimiento del problema, para lograr con ligeras indicaciones en su implementación, la solución en diferentes tipos de problemas, llegando a casos de éxito debido al buen desempeño de este.

El proceso que el árbol emplea, es ampliamente usado en campos donde la toma de decisiones parte desde un dominio conocido, y que las acciones que repercutan estén definidas, a fin de determinar dentro del conjunto la acción oportuna para lograr dar solución al problema (James et al., 2017), en donde se maneja la definición matemática necesaria para el desarrollo, sea UCT o UCB_1, a fin de que la elección de las opciones se dé en forma en que la predicción sea la más valorada posible, donde tenemos que UCB_1 se define como (Levine, 2017):

$$X = \frac{\omega_i}{n_i} + c\sqrt{\frac{ln\, N_i}{n_i}}$$

Figura 3. Fórmula UCB1

Donde:

ω_i: El número de valores de ganancia que ha tenido el nodo durante la retro propagación para cada movimiento $_i$.

n_i: El número de veces que ha sido visitado el nodo por cada movimiento $_i$.

N_i: Número de veces que ha sido visitado el nodo Padre

C: Es el parámetro de exploración, teóricamente debe ser 2, para que funcione empíricamente.

A partir de la secuencia de procedimientos matemáticos que se dan para cada nodo hijo, se genera las diferentes fases bajo las cuales el árbol va a regir su comportamiento, desde el estado inicial hasta que encuentre la decisión acertada bajo la cual se dará uso del mismo. Es por esto que su comportamiento es definido en cuatro fases básicas, bajo las cuales se dará su crecimiento y selección de nodos, para estas tenemos:

Selección: Esta opción parte desde la creación del nodo raíz, el cual se define como el primero en dar forma al árbol y del cual partirán todas las elecciones posibles, este no considera acción alguna, ya que su función es crear los hijos que dan inicio al planteamiento del problema, para así, lograr definir el árbol mediante la selección del hijo con mayor valor de ganancia, para comenzar a expandir por esa rama *(Véase figura 4),* esto se valida con el parámetro ω_i, que es incremental mediante es visitado nuevamente dicho nodo, sumando el valor actual al que retorna sus hijos para lograr acotar en mayor medida el camino, obteniendo una mejor proyección para las posibles acciones que cuenta al momento.

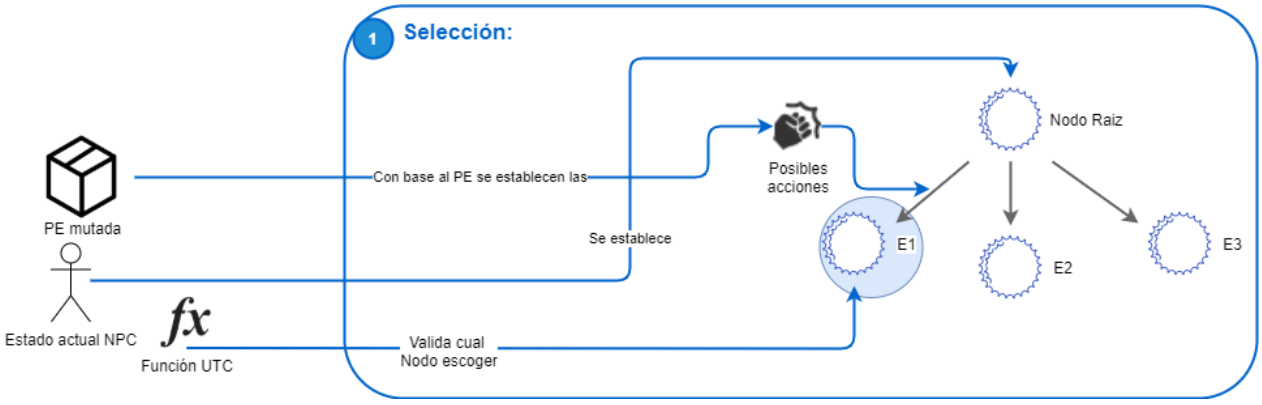

Figura 4. Proceso de selección de MCTS

Expansión: En el momento de que el nodo represente un caso de éxito, este genera nodos con los mismos casos de victoria, a fin de seguir expandiendo y llegando a las hojas del árbol con la mejor proyección *(Véase figura 5),* lo que define como un caso de éxito, tener el mayor valor a partir de la acotación numérica, en donde se define que cualquier nodo que tenga una cantidad de visitas en 0 ó n_i=0, representa una división que tiende a indefinido, por lo que se asegurara de recorrer la secuencia total de hojas por cada proceso de expansión, pero una vez esta fila de nodos tenga sus respectivos hijos, optará por ir expandiendo la rama con mejor tendencia en valores.

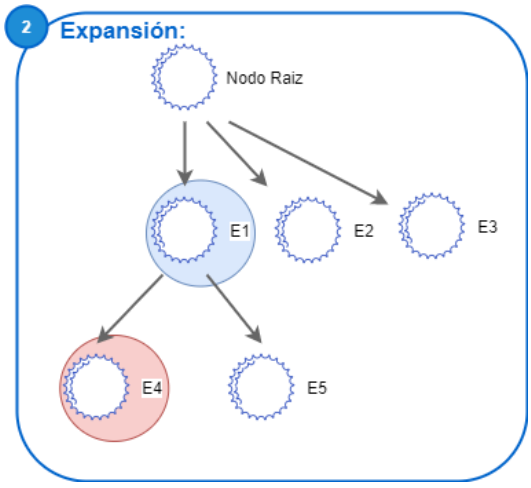

Figura 5. Proceso de Expansión de MCTS

Simulación: Genera una simulación de casos, bajo los cuales se ve cuál puede ser el caso de éxito a partir de unas "recompensas", siendo otorgadas mediante completar ciertos objetivos a lo largo del proceso, para saber qué acciones tienen un mejor valor a fin de retornar hasta el nodo bajo el cual se inició el proceso. Este proceso da pie para que se inicie y valide la retro propagación *(Véase figura 6).*

Figura 6. Proceso de Simulación de MCTS

Retro propagación: Es un proceso bajo el cual el árbol, al tener los valores del nodo con la mejor proyección, retorna este dato al nodo padre y así sucesivamente, para lograr llegar al nodo raíz, en donde va incrementando las visitas de cada nodo y su respectivo valor que fue aumentando secuencialmente durante el recorrido *(Véase figura 7).*

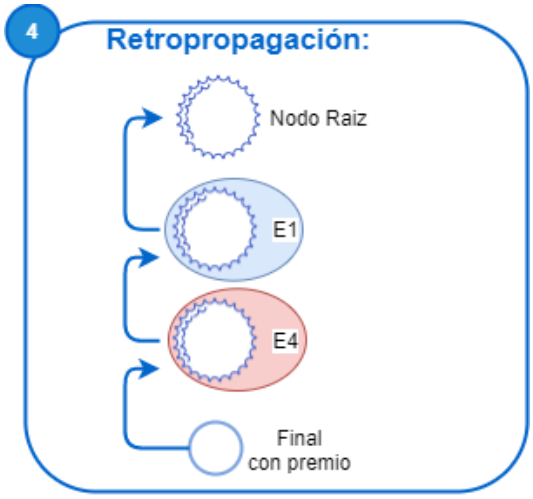

Figura 7. Proceso de Retro propagación MCTS

Arte del combate romano

Durante el tiempo de los diferentes conflictos que se desarrollaron durante la época del gran imperio romano, se generó la pérdida de innumerables vidas y avances en el arte de la guerra, a fin de perfeccionar el ejército y lograr tener un mayor porcentaje de victorias en sus enfrentamientos. Todo esto llegó a intrigar a diversos cineastas que han hecho una gran variedad de largometrajes, que pretenden recrear estos combates de manera dinámica y ficticia con respecto a lo que en realidad se vivió, junto a este trabajo, las legiones romanas han tenido un gran impacto en el desarrollo de videojuegos, los cuales se inspiran en el manejo que esta armada presentaba durante sus combates, lo que se puede ver en el título *Ryse son of Rome,* que cuenta con una ambientación en la antigua Roma durante el reinado de Nero (Nichols, 2014).

Aunque las representaciones no pueden estar más alejadas de la realidad, dado a que en su estrategia, los soldados andan en una búsqueda constante de desestabilizar a su adversario, mediante movimientos continuos que representen un mayor esfuerzo físico y estrés al contrincante, desencadenando todo esto en un combate frenético a fin de lograr aniquilar al oponente de la manera más pronta posible (Batán, 2016).

Al tener enfrentamientos cortos y estratégicos, la armada busca en primera medida generar un ataque directo sobre su oponente, en donde las armas pesadas y de larga distancia, como lo son el onagro *(véase figura 8)* y las balistas (*véase figura 9),* definen la primera línea de ataque sobre la formación enemiga, dando el ataque inicial directo, que afecta a las líneas contrarias dando una desestabilización a su posición y generando paso a que la *Équite,* una subdivisión montada en equinos, que generalmente hacía uso del pilum *(véase figura 12),* dispersara las masas de soldados en subgrupos para que la propia legión romana no tuviera inconvenientes en sus números (Arre Caballo, 2017).

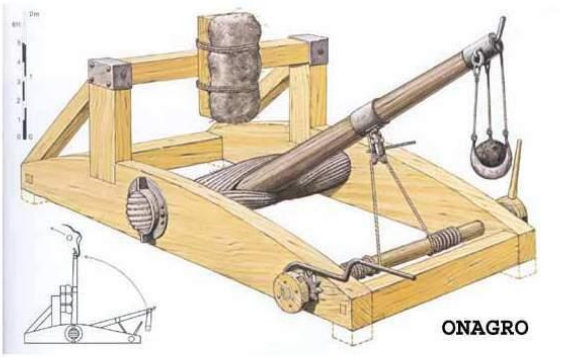

Figura 8. Onagro. Recuperado de https://www.ecured.cu/Archivo:Onagro.jpg

Figura 9. Ballista. Recuperado de http://legionarioderoma.blogspot.com/2013/04/ballista.html

Una vez la estrategia se ha completado y se tienen divisiones manejables de armada enemiga, el combate cuerpo a cuerpo inicia, caracterizado por un combate marcado por momentos de frenesí (Batán, 2016). En este tipo de combate, la condición física y el estado mental en un enfrentamiento, pueden llegar a determinar una victoria de una derrota, por lo que el uso correcto de las armas y equipamientos facilita la ardua labor del combate, como las armas arrojadizas o armas de distancia, a fin de limitar el esfuerzo físico que representa el combate cuerpo a cuerpo. La armada romana, se valía de armas que pudieran generar bajas o el mayor daño posible sin tener que entrar en un ataque directo, de las cuales podemos destacar:

Hastae o Hasta

Es una lanza de acero, que medía aproximadamente seis pies romanos de largo, que eran usados por veteranos de la guerra, implementados como escuadras de reserva que combatían exclusivamente de ser necesario (*Véase figura 10)* (Batán, 2016).

Figura 10. Hasta. Recuperado de https://www.taringa.net/+ciencia_educacion/armas-y-armamento-del-ejercito-romano_hrxdk

Verutum

Era una jabalina que no tenía una longitud superior a un metro, considerada muy flexible dada su disminución en peso comparada con el pilum, y era usada por la clase pobre ya que brindaba una gran capacidad de maniobrabilidad contra el enemigo *(Véase figura 11)* (Batán, 2016).

Figura 11. Verutum. Recuperado de https://beacon.by/revistadehistoria/revista-de-historia-especial-legiones-de-roma/3#/3

Pilum

Era el arma preferida por la armada romana al tener la capacidad de ser arrojada y volverse inútil, dado a que se presentaba una curvatura al realizar el impacto. Aunque este hecho no se realizaba de manera intencionada sino por un defecto en el metal bajo el cual esta se fabricaba. Se basa en una punta piramidal y una longitud de metro y medio a dos metros aproximadamente, con

un peso que oscilaba entre dos y cinco kilos, que cuenta con la capacidad de atravesar tres cm de madera aproximadamente. *(Véase figura 12)* (Desperta Ferro Ediciones, 2018).

Figura 12. Pilum, Recuperado de https://www.tienda-medieval.com/blog/el-pilum-romano.html

Contus

Es un arma pesada que debía ser usada a dos manos, usada por los romanos, griegos y bizantinos, siendo utilizada por los ejércitos romanos como principal arma de la caballería para perseguir enemigos que escaparan del combate *(Véase figura 13)* (Batán, 2016).

Figura 13. Contus. Recuperado de https://www.taringa.net/+ciencia_educacion/armas-y-armamento-del-ejercito-romano_hrxdk

Spiculum

Es la innovación del Pilum, en donde los historiadores creen que pudo ser la mezcla entre jabalina romana y lanzas germánicas, logrando así poder ser utilizada como arma cuerpo a cuerpo o arrojadiza *(Véase figura 14)* (Batán, 2016).

Figura 14. Spiculum. Recuperado de https://beacon.by/revistadehistoria/revista-de-historia-especial-legiones-de-roma/3#/3

Plumbata

Traducida como lanza emplomada o dardo de Marte, es un dardo con un lastre de hierro que le daba el peso suficiente para lograr atravesar escudos enemigos, contando con la cualidad de poder ser lanzada con una sola mano y alcanzando una distancia de aproximadamente 60 m *(Véase figura 15)* (Desperta Ferro Ediciones, 2018).

Figura 15. Plumbata. Recuperado de http://amodelcastillo.blogspot.com/2014/04/las-espinas-de-marte.html

Sagittarii

División de la caballería especializada en el uso del arco y armas a distancia, el cual a diferencia de los arcos del momento, se componía de diferentes materiales, lo que algunos aseguran le daba un mayor rango de alcance. (*Véase figura 16)* (Batán, 2016).

Figura 16. Sagittarii. Recuperado de https://beacon.by/revistadehistoria/revista-de-historia-especial-legiones-de-roma/3#/3

Dado la gran cantidad de armas a distancia romanas, no se debe mitigar el impacto que las armas cuerpo a cuerpo han tenido en los enfrentamientos bélicos, contando con la amplia cantidad de bajas que se podían causar al hacer un uso adecuado de estas, contando con avances tecnológicos que facilitarán su manejo, de acuerdo a esto se tienen un conjunto de los siguientes tipos de armas:

Gladius

Arma predilecta por la armada romana debido a su gran eficacia y facilidad para ejecutar bajas enemigas, compuesta por un mango y una daga con una media de 50 cm de longitud y seis cm de ancho, con un doble filo lo que brindaba una gran facilidad para realizar estocadas al momento de que el oponente se descubriera en un movimiento en falso *(véase figura 17)* (Arteaga, 2011).

Figura 17. Gladius. Recuperado de https://es.wikipedia.org/wiki/Gladius

Spatha

En cambio de la gladius, este espadón contaba con una longitud mayor, con aproximadamente una longitud de entre 60 y 90 cm, convirtiéndose en la espada principal de las legiones, dada a la decadencia que ocurrió con el paso de los años en el imperio y la conformación de estas por soldados celtas y germanos, los cuales influyeron en los equipamientos con los cuales contaban *(véase figura 18)* (Batán, 2016).

Figura 18. Spatha. Recuperado de https://beacon.by/revistadehistoria/revista-de-historia-especial-legiones-de-roma/3#/7

Pugio

Era un puñal corto, de aproximadamente 20 a 24 cm de longitud con seis cm de ancho, utilizado en el momento donde fuera la única defensa por parte del legionario o se desease realizar un ataque certero sobre el contrincante, con el fin de acelerar su muerte. *(Véase figura 19)* (Batán, 2016).

Figura 19. Pugio. Recuperado de https://beacon.by/revistadehistoria/revista-de-historia-especial-legiones-de-roma/3#/7

Maza

Es conocida como un arma de combate pesado, compuesto por un mango que generalmente está compuesto por hierro, terminando en una cabeza pesada que puede estar elaborada en bronce, cobre o hierro, que con el paso del tiempo adoptó cuchillas, pinchos u otro tipo de implemento que aumenta la letalidad de esta *(véase figura 20)* (Merino et al., 2011).

Figura 20. Maza. Recuperado de https://beacon.by/revistadehistoria/revista-de-historia-especial-legiones-de-roma/3#/7

Como se ha visto, el poder armamentístico tanto del imperio, como de diferentes armadas que han sido recreadas y estudiadas en otros campos, generaba un amplio daño en la condición física de quien recibía los ataques, y así, a partir de las heridas sufridas se podía llegar a tener fiebre, infecciones y otras posibles enfermedades por el intenso combate llevado a cabo contra el enemigo, (Batán, 2016) es por esto que se generó una mezcla de armaduras de diferentes culturas para poder proteger de la mejor manera a las unidades, y de todo esto podemos rescatar los siguientes elementos:

Lorica Hamata

Armadura usada por las tropas legionarias, así como las tropas auxiliares, formado por una cota de malla y anillos metálicos destinados para cubrir tanto la parte frontal, como la posterior del torso, cubriendo contra corte de espadas, martillo y hachas, pero con gran debilidad frente a flechas o armas punzantes arrojadizas. (*Véase figura 21)* (Batán, 2016).

Figura 21. Lorica Hamata. Recuperado de https://beacon.by/revistadehistoria/revista-de-historia-especial-legiones-de-roma/3#/4

Lorica Squamata

Cuenta con características similares a la lorica hamata, con la diferencia de estar compuesta por segmentos metálicos a manera de escamas, cocidas sobre una base de tela, siendo usadas por músicos, centuriones, tropas auxiliares y caballería, con un grosor de aproximadamente 0.5 mm (*Véase figura 22)* (Bofill, 2019).

Figura 22. Lorica Squamata. Recuperado de https://beacon.by/revistadehistoria/revista-de-historia-especial-legiones-de-roma/3#/4

Lorica Plumata

Este nombre viene de una disposición de escamas que son conformadas por anillos metálicos, dando un mejor acabado que la lorica squamata, la cual asemeja a las plumas de un ave. Debido a su alto costo de producción se delegó a los oficiales de alto rango. (*Véase figura 23)* (Batán, 2016).

Figura 23. Lorica Plumata. Recuperado de https://beacon.by/revistadehistoria/revista-de-historia-especial-legiones-de-roma/3#/4

Lorica Musculata

Es la armadura más conocida debido a la industria cinematográfica, la cual es una pieza de bronce macizo que cubre todo el torso y retrata tanto los pectorales como los abdominales, gracias a las amplias habilidades de los maestros herreros, siendo usada por emperadores y oficiales de un muy alto rango.*(Véase figura 24)* (Batán, 2016).

Figura 24. Lorica Musculata. Recuperado de https://beacon.by/revistadehistoria/revista-de-historia-especial-legiones-de-roma/3#/4

Lorica Segmentata

Usada por las unidades especiales, es una armadura compuesta por varios segmentos metálicos que disponían de un movimiento de acuerdo a las piezas, elaborada en metales no endurecidos en el proceso de la forja, para que absorbieran el impacto, lo que le daba una amplia protección y con un peso menor a la cota de malla, ya que esta pesa 16 kg, mientras que la segmentata aproximadamente nueve kilogramos. *(Véase figura 25)* (Arribas, 2017).

Figura 25. Lorica Segmentata. Recuperado de https://beacon.by/revistadehistoria/revista-de-historia-especial-legiones-de-roma/3#/4

Otras armaduras

Armaduras usadas por la infantería más ligera, basadas en un bajo costo y de fácil producción, por lo que no permitían una protección avanzada, pero una movilidad amplia con materiales como cuero o bronce. *(Véase figura 26)* (Batán, 2016).

Figura 26. Otro tipo de armaduras. Recuperado de https://beacon.by/revistadehistoria/revista-de-historia-especial-legiones-de-roma/3#/4

Figura 27. Escudos romanos. Recuperado de https://www.pinterest.ca/pin/547398529694564670/

Scutum

Es el escudo más conocido de las legiones, contaba con un tamaño que cubría de hombros a la parte superior de la rodilla, con una curvatura para proteger desde varios flancos y poder hacer uso de la gladius en su momento, con una pieza en bronce llamado umbo ubicado en todo el centro de su parte frontal, que permitía el desplazamiento del enemigo mediante un empuje en su contra. *(Véase figura 27)* (D'amato, 2009).

Cetratus

Destinado a las tropas auxiliares, siendo un escudo de cuero y tela revestido por metal, poco pesado y fácil de usar con una menor resistencia comparada frente al scutum. *(Véase figura 27)* (Batán, 2016).

Parma

Escudo circular usado por la caballería y la infantería auxiliar, fabricado de forma simple y con un diámetro de aproximadamente 80 cm. *(Véase figura 27)* (D'amato, 2009).

Capítulo 3. Materiales y Métodos

Desarrollo

En este capítulo se presenta el desarrollo de un juego RPG táctico por turnos. El cual cuenta con modelos basados en geometrías básicas, implementadas en Unreal Engine 4 a fin de simular el entorno de un juego RPG. Como primer paso se toma de referencia la generación de un mapa tridimensional que cuenta con mecánicas de desplazamiento sobre su superficie *(véase figura 28),* contando con una cuadrícula de casillas sobre la cual se moverán y ejecutarán sus acciones tanto los NPC como el jugador.

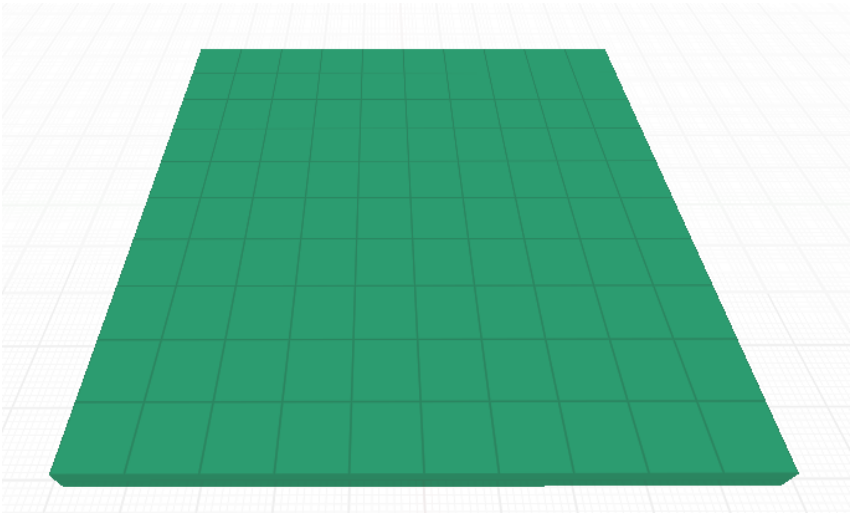

Figura 28. Mapa del juego

Situando una cámara sobre el mapa, a fin de tener una visión del escenario con geometría perspectiva y contar con la posibilidad de girar la posición de la cámara alrededor de este. El vector de orientación se mantiene en el mismo punto para manejarlo a forma de pivote, y poder ver desde las cuatro posibles vistas el mapa evitando obstáculos visuales. Ya que en escena es posible encontrar geometrías que emulan elementos del entorno, como naturaleza, murallas, escombros, etc. Así mismo el NPC y su sombra están situadas en el campo *(véase figura 29).*

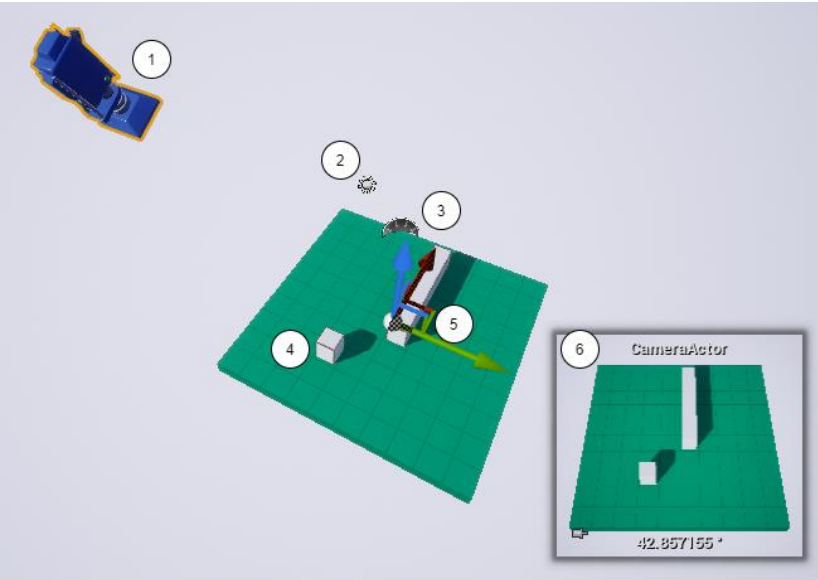

Figura 29. Implementación de la cámara, (1. Cámara, 2. Luz direccional, 3. Luz ambiente, 4. Obstáculos, 5. Eje coordenado, 6. Punto de la vista de la cámara)

A seguir se desarrolla el avatar que representa al jugador, el cual para fines prácticos tiene una coloratura azul, que puede ser transformada en algún otro tono deseado. En este paso se le otorgan atributos al avatar referentes al rango de visión. Y que en el editor se observa como las circunferencias azules que rodean el perímetro exterior, acompañado del perímetro inferior, representado de color rojo que delimita el rango de acción de cada arma. Adicionalmente se incluye el colisionador que determina sobre que casilla se encuentra ubicado el actor, representado por el contorno que rodea la geometría del jugador *(véase figura 30).*

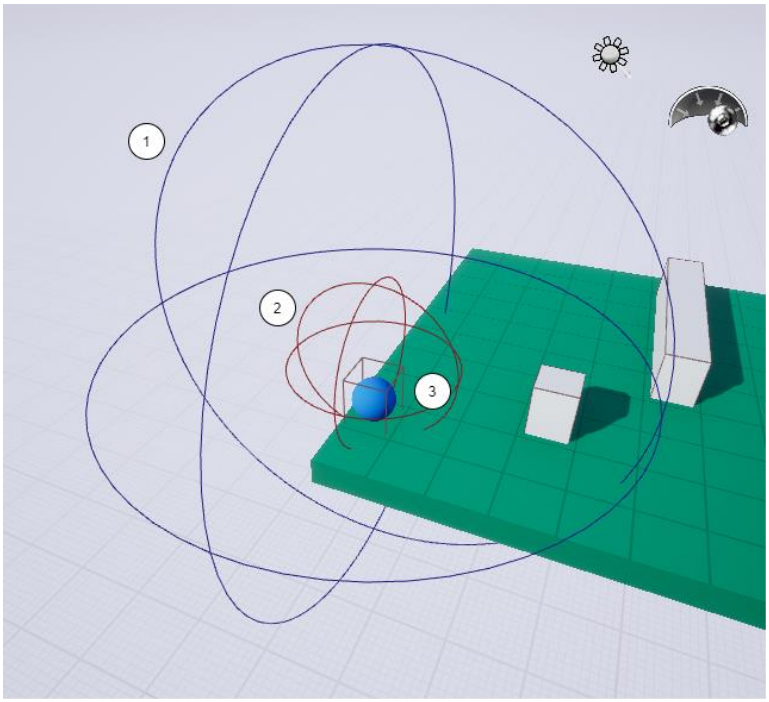

Figura 30. Implementación del jugador, (1. Rango de visión, 2. Rango de acción, 3. Colisionador)

El actor cuenta con atributos modificables en el editor, que hacen referencia a las configuraciones hechas en el código para este, como lo son el estado, que cuenta de nivel, salud máxima, salud actual, equipamiento máximo, equipamiento actual, poder de taque y su defensa, siendo los recursos que afectan al actor durante la partida *(véase figura 32a)*. El estado, también se ve influenciado por el inventario con el que cuente al combatir contra el NPC, incluyendo armadura, armas y escudos, que definen la cantidad de daño infligido o recibido para el personaje *(véase figura 32b)*. Finalmente, en la configuración del equipo (la cual es establecida al inicio del juego), se determina qué herramientas llevará puestas de su inventario *(véase figura 32c)*, evento que ocurre al iniciar la partida y configura el esquema que tendrá el actor para enfrentar a los NPC *(véase figura 33)*.

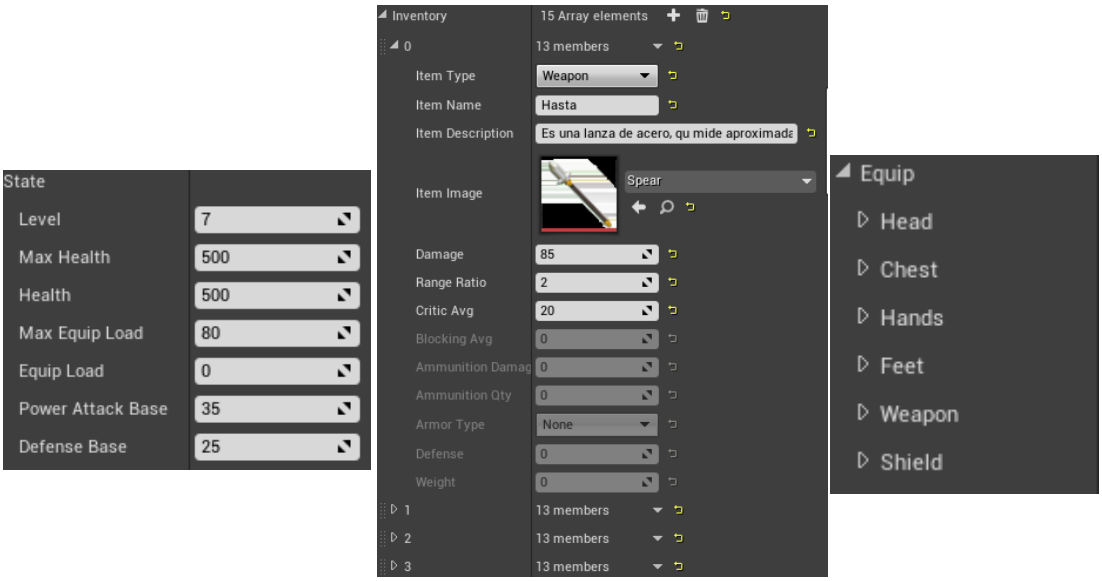

Figura 31. a) Estado del jugador, b) Inventario del jugador, c) Equipo del jugador

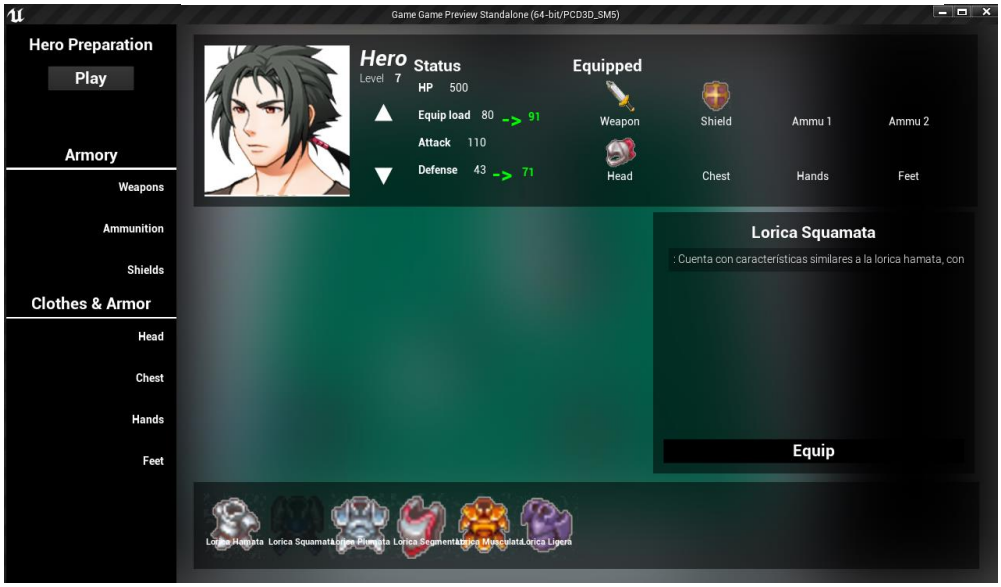

Figura 32. Menú de equipamiento

Para cada enfrentamiento que se genera entre los actores, del cual la acción proviene del jugador, se permite seleccionar la acción que este ejecutara en contra del NPC. Dentro de este catálogo de acciones contamos con el ataque, la defensa, desplazamiento, y también el ignorar que representa al saltar un turno *(véase figura 33)*. Junto a esto, el desarrollo empleado para el movimiento de ambos para actores, aplica el cálculo del desplazamiento mediante la distancia Manhattan, la cual considera el espacio entre dos puntos (entendidos como A y B), siendo estos vectores bidimensionales, en los cuales la resta de sus componentes representa la distancia final *(véase figura 34)*.

Figura 33. Acciones por parte del jugador

$$d(A,B) = \sum_{i=1}^{n} |A_i - B_i|$$

Figura 34. Fórmula Distancia Manhattan

Luego de definir el movimiento y las acciones para el personaje que manipula el jugador, se genera el actor NPC el cual es una nueva clase de objeto que hereda todos los recursos que caracterizan al jugador, cambiando su color de textura a rojo, para así poder distinguirlo, y añadiendo características únicas que solo el NPC puede realizar (*véase Figura 35*).

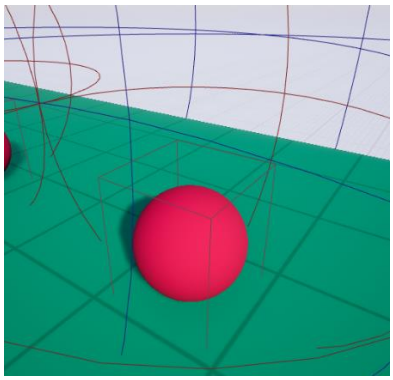

Figura 35. NPC en escena

Se realiza una configuración de las acciones del NPC, para integrar las posibles opciones que determinan el comportamiento que este tendrá. Las acciones incluyen desplazamientos que lo acerquen o alejen con respecto a la posición del jugador, ataques o bloqueos que suben la defensa, y por último, el observar, que se basa en un análisis del desarrollo de su ambiente *(véase figura 36).* Las acciones se procesan mediante la lógica del árbol y los paquetes de estado, que a lo largo del proceso, alteran sus valores, con el fin de realizar un movimiento estratégico por parte del NPC.

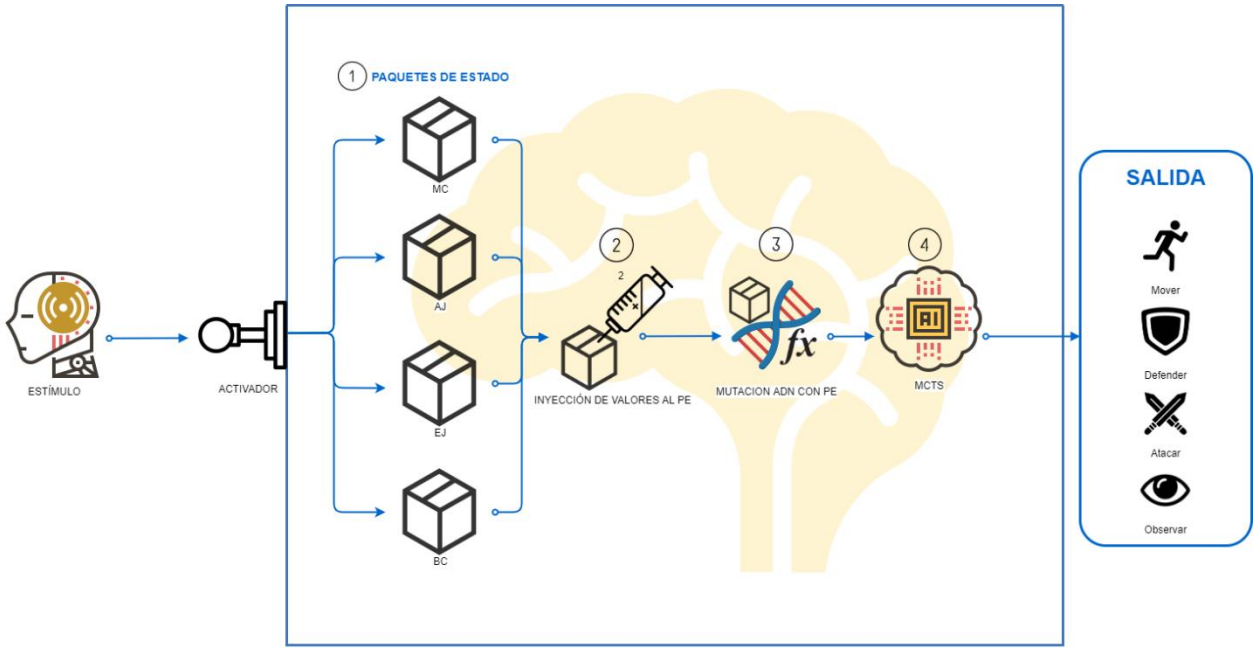

Figura 36. Diagrama de caja blanca del modelo del sistema

Diagrama Sistémico de Componentes

La secuencia para ejecutar el proceso, permite obtener los valores adecuados para manipular los datos y tener información tangible para tomar la decisión correcta, siguiendo el flujo a lo largo del sistema. Se tiene como primer proceso la definición del paquete de estado, que se conceptualiza mediante la definición de los rangos de miedo, venganza y confianza. *(Véase figura 37).* Para este desarrollo, los rangos se definen como las variables que son afectadas por la personalidad del NPC durante un combate y se ven alteradas bajo cuatro tipos de estímulos. Los estímulos que afectan al NPC incluyen la muerte de un compañero (MC), ataque de jugador (AJ), equipo con el que cuenta el jugador (EJ) y berserker (BC), siendo entendido como un estado frenético *(véase figura 36)*, siendo estos los eventos subyacentes de las mecánicas de juego.

Teniendo como referencia los eventos que generan los estímulos, el activador inicia un proceso de inyección de valores a cada tipo de paquete de estado, los cuales contarán con un incremento o decremento porcentual, que toma de base los recursos tanto del jugador como del NPC, junto al entorno y cantidad de compañeros en escena *(véase figura 37)*. Detonando en un paquete de estado final que determina con que valores el árbol trabajará a lo largo de su ejecución *(véase figura 38)*.

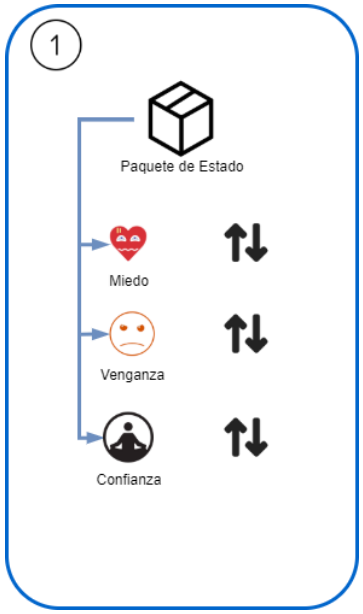

Figura 37. Diagrama del primer componente, Paquete de estado

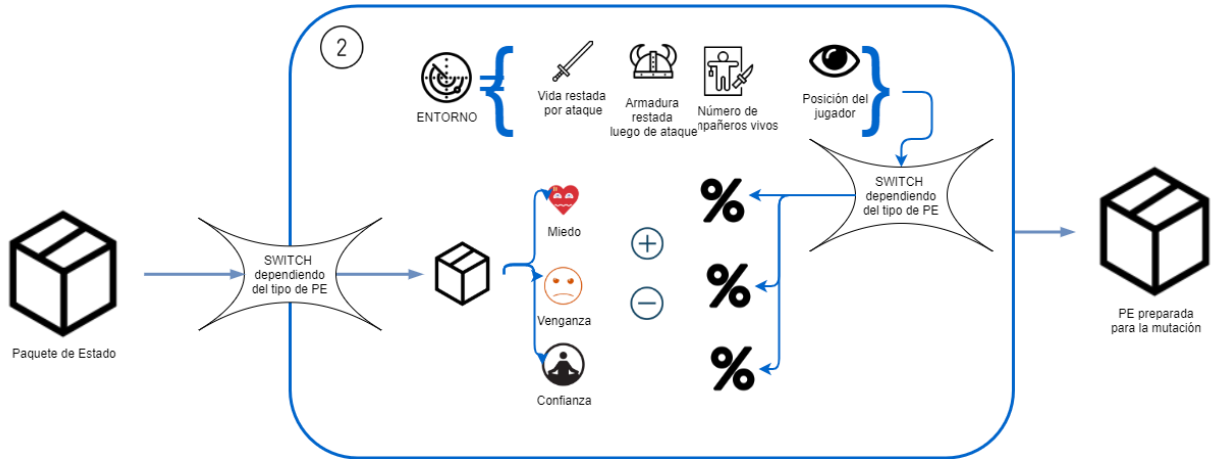

Figura 38. Diagrama del segundo componente, inyección de valores al paquete de estado

Con el paquete de estado definido, se procede a realizar la mutación en sí, la cual consta de afectar los datos que definen la personalidad del NPC, que según el desarrollo planteado están definidos como el IQ, miedo, valentía, venganza, locura, respeto, liderazgo y confianza. Variables de personalidad que se tomaron en cuenta con el fin de poder parametrizar varias personalidades con base a los tipos que requiera el desarrollador *(véase figura 39).*

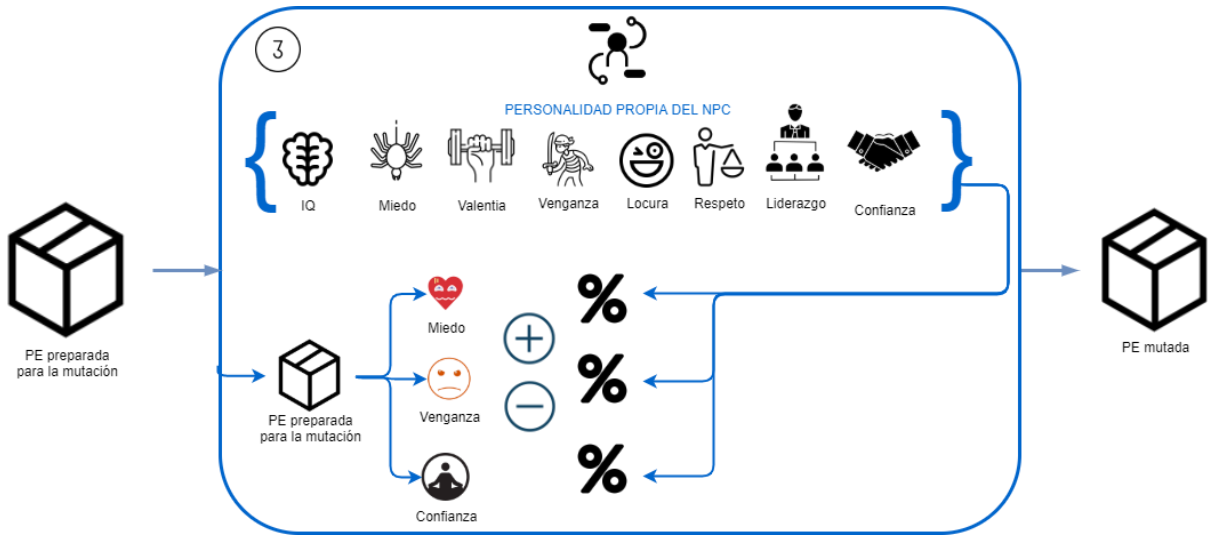

Figura 39. Diagrama del tercer componente, mutación de valores al paquete de estado

Por último, la función UCT del MCTS, determina los posibles caminos que pueden dar con la mejor respuesta, donde el propio árbol genera la selección de cada nodo hijo, para cada una de las posibles acciones que pueda llegar a realizar. Luego se procede a expandir, simular y validar cual es la mejor opción mediante la retro propagación de valores *(véase figura 40)*. Con la secuencia de procedimientos establecida, el procedimiento general se determina a partir de cada componente establecido en el que se procesa la percepción de un estímulo, para culminar en la salida de una acción que sea propicia con respecto al fenómeno percibido *(véase figura 41)*.

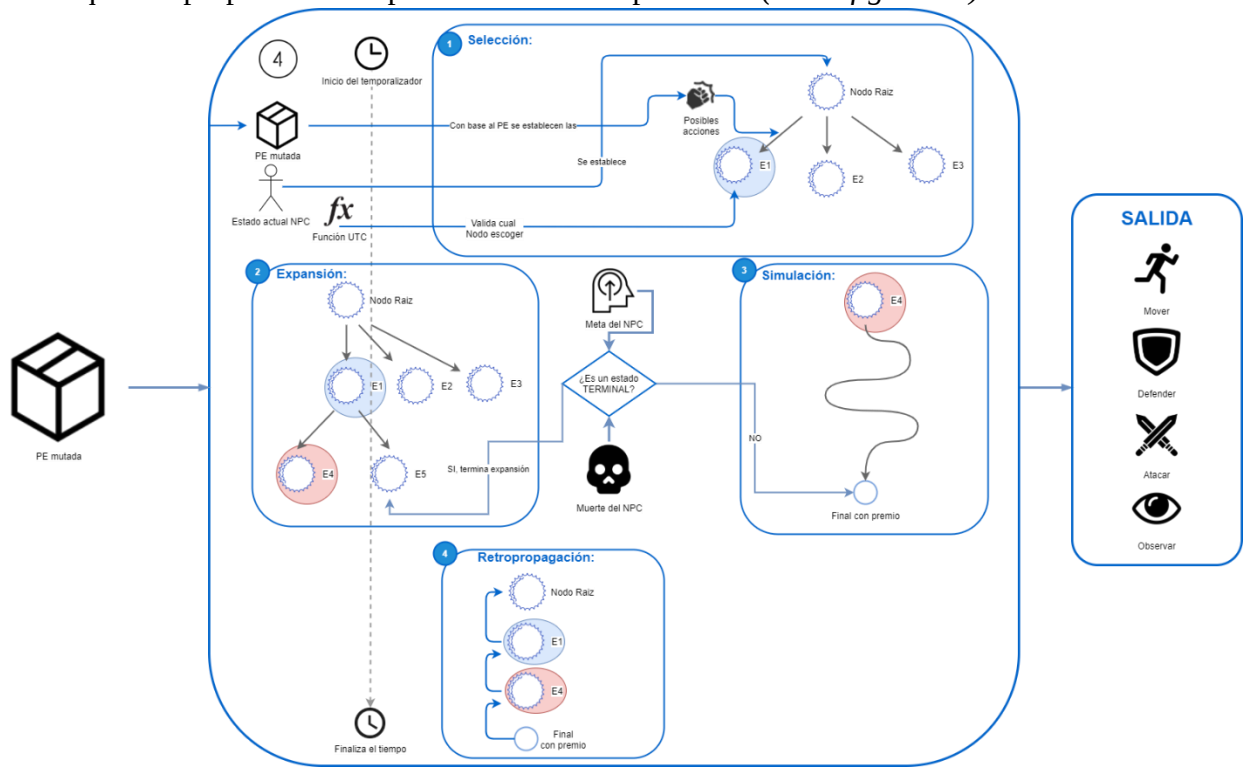

Figura 40. Diagrama del cuarto componente, MCTS

Método computacional

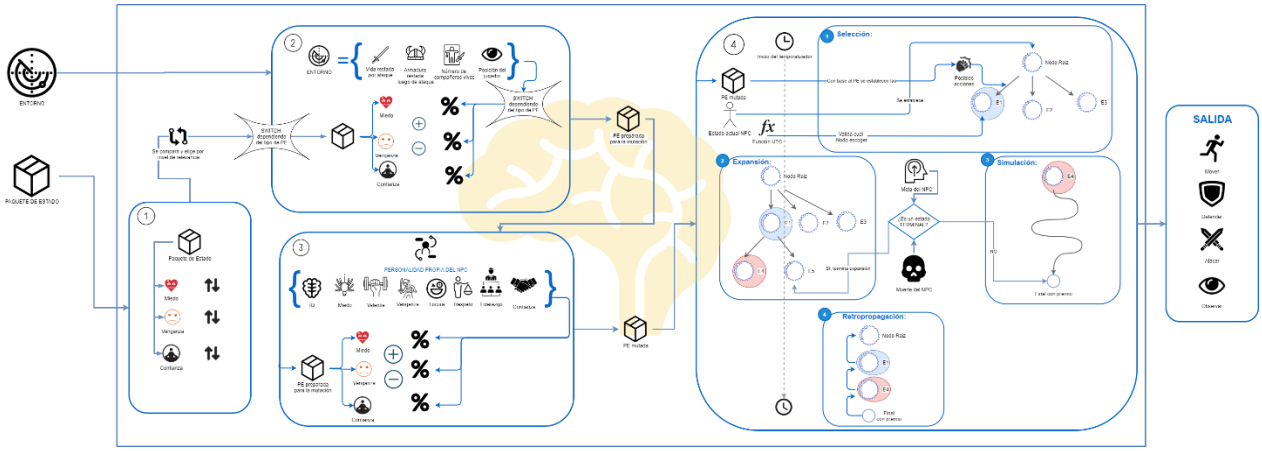

Figura 41. Diagrama de caja blanca del sistema completo

Configuración del plug-in

Para lograr hacer uso del plug-in dentro de un proyecto, cabe resaltar que como procedimiento inicial se debe tomar el código fuente de este, e importarlo dentro del directorio *Plugins*, el cual habilitará una opción en el editor que permite realizar la activación del mismo, al cual se puede acceder en la barra de tareas en la opción *Edit*, para seleccionar posteriormente la opción *plugins*, la cual abrirá el modal correspondiente en donde al final de la lista de todos los complementos que ofrece el motor, se encuentran aquellos que han sido importados para ser activados *(véase figura 42 y anexo 1)*.

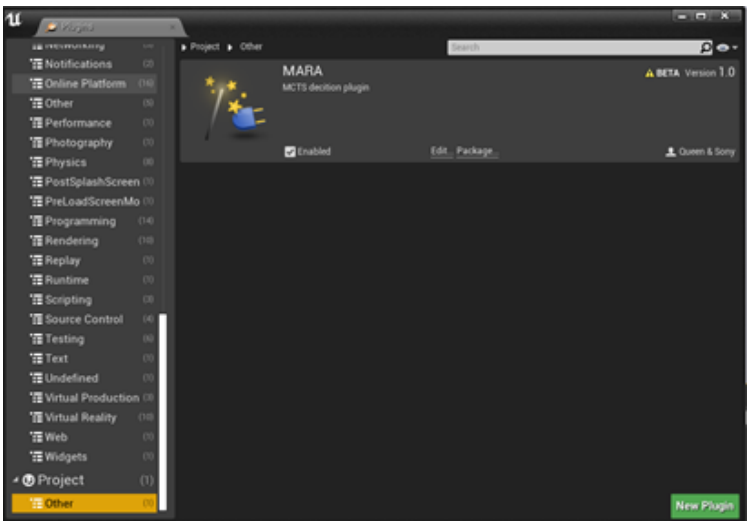

Figura 42. Activación del plugin en el motor

Protocolo de pruebas

Para realizar la validación del desempeño del plug-in, se genera un conjunto de pruebas basadas en el enfrentamiento del jugador con uno o más NPC, los cuales varían en los datos de su personalidad y equipamiento. Con el fin de validar el comportamiento, se definen las características estándar bajo las cuales el jugador tendrá todos sus enfrentamientos. El jugador tiene un nivel de 7, salud de 500, equipamiento de 80, ataque de 35, defensa de 25 y una armadura que se encuentra en la media de las posibilidades compuesto por el casco imperial, lorica squamata, hastae y scutum *(véase tabla 1).*

JUGADOR			
Level	7	Salud	500
Equipamiento	80	Ataque	35
Defensa	25	Casco	Casco Imperial Defensa = 8
Armadura	Lorica Squamata Defensa = 28	Arma	Hastae Daño = 85 Rango de ataque = 2 Daño crítico = 20
	Escudo	Scutum Bloqueo = 60	

Tabla 1. Estado del jugador

Junto a los parámetros de prueba con los cuales el jugador contará, se hace la esquematización de datos para el NPC en los diferentes tipos de respuestas para enfrentar al actor manipulado por el jugador. Se definen tres tipos posibles de NPC que, varían según el nivel que determina su personalidad y sus características de estado. La concepción de los NPC incluye una esquematización de bajo rango, con el fin de tener una personalidad con un alto valor de miedo y respeto para referenciar a un soldado de bajo rango semejante a las legiones romanas *(véase tabla*

2), y también de un nivel bajo con una salud que corresponde a la mitad de la del jugador y de un ataque y defensa mínimo y una armadura de baja categoría *(véase tabla 3).*

NPC - Nivel bajo - Personalidad			
IQ	0.35	Miedo	0.68
Valentia	0.10	Venganza	0.40
Locura	0.20	Respeto	0.80
Liderazgo	0.10	Confianza	0.70

Tabla 2. Personalidad NPC nivel bajo

NPC - Nivel bajo - Estado			
Level	2	Salud	250
Equipamiento	30	Ataque	10
Defensa	12	Casco	Casco Imperial Defensa = 8
Armadura	Lorica Hamata Defensa = 20	Arma	Spatha Ataque = 75 Rango ataque = 1 Daño crítico = 10
Escudo	Parma Bloqueo = 45	Percepción	3

Tabla 3. Estado NPC nivel bajo

Para el segundo NPC, se toma como referencia las características del jugador a fin de llevar un enfrentamiento equivalente entre ambos actores *(véase tabla 4).* En este, la única característica que cambia es la percepción del NPC, en donde se le asigna una cantidad de cinco unidades *(véase tabla 5).* Esta concepción, busca ver el comportamiento entre iguales y cómo se maneja el plug-in frente a dicha circunstancia, para que así se pueda culminar con la implementación de un actor mejorado, el cual presentará el mayor reto para el jugador.

NPC Básico - Personalidad			
IQ	0.60	Miedo	0.30
Valentia	0.20	Venganza	0.15
Locura	0.5	Respeto	0.25
Liderazgo	0.10	Confianza	0.20

Tabla 4. Personalidad NPC nivel básico

NPC Básico - Estado			
Level	7	Salud	500
Equipamiento	80	Ataque	35
Defensa	25	Casco	Casco Imperial Defensa = 8
Armadura	Lorica Squamata Defensa = 28	Arma	Hastae Daño = 85 Rango de ataque = 2 Daño crítico = 20
Escudo	Scutum Bloqueo = 60	Percepción	5

Tabla 5. Estado NPC nivel bajo

Para definir una personalidad robusta y que haga contraposición al jugador, se define un esquema con un valor del 30% en miedo y con una tendencia a ser un actor firme en el combate, teniendo un 70% de liderazgo, 60% de IQ y 67% de valentía entre otros *(véase tabla 6).* Tomando como referencia un miedo bajo y opciones de combate altas que fomenten que en los enfrentamientos se den acciones que incentiven el ataque constante, potenciando todas las variables del NPC, subiendo cinco puntos el nivel, 600 puntos la salud, 40 puntos el equipamiento, 15 puntos el ataque y 45 la defensa *(véase tabla 7).*

NPC - Mejorado - Personalidad			
IQ	0.60	Miedo	0.30
Valentia	0.67	Venganza	0.15
Locura	0.5	Respeto	0.25
Liderazgo	0.7	Confianza	0.4

Tabla 6. Personalidad NPC nivel mejorado

NPC - Mejorado - Estado			
Level	12	Salud	1100
Equipamiento	120	Ataque	50
Defensa	70	Casco	Casco Gálico Imperial Defensa = 18
Armadura	Lorica Musculata Defensa = 35	Arma	Maza Daño = 100 Rango ataque = 1 Daño crítico = 25
Escudo	Cetratus Bloqueo = 70	Percepción	5

Tabla 7. Estado NPC nivel mejorado

Al tener definidos los tres tipos de posibles NPC, se desarrollarán los escenarios en los cuales se darán los enfrentamientos. Tres escenarios en los cuales intercambian los NPC manteniendo como constante la posición del jugador (casilla azul) y de los obstáculos en escena (casillas verdes). Para el primer conjunto de pruebas se define un NPC en escena (casilla roja), del cual derivan tres sub escenarios posibles en donde se irán alternando en los diferentes rangos mencionados anteriormente y manteniendo su posición *(véase figura 43).*

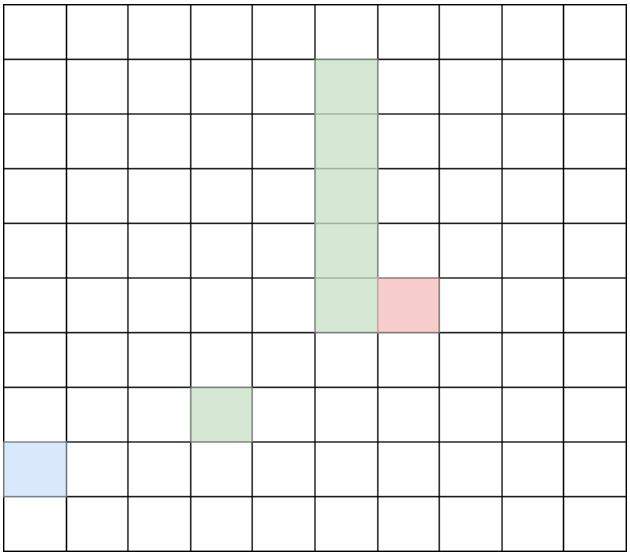

Figura 43. Distribución para el primer caso de prueba

Para el segundo escenario el número de NPC aumenta a dos *(véase figura 44)*, y se generan tres momentos de manera semejante al grupo anterior. El primer escenario presenta un personaje nivel bajo y uno nivel medio, el segundo escenario presenta un NPC de nivel bajo y uno de nivel alto, y para finalizar, se termina con un NPC de nivel medio y uno de nivel alto, donde se evidencie el momento donde el jugador derrota al personaje no jugable con mayor rango, a fin de ver el comportamiento de su compañero en dicha situación. Junto a estos enfrentamientos, el tercer caso presenta los tres tipos de actores *(véase figura 45)* donde el jugador igual que en el ejemplo anterior, atacará al NPC de menor rango de manera inicial para así ir atacando a los demás personajes no jugables en escena y determinar el ganador de dicho enfrentamiento.

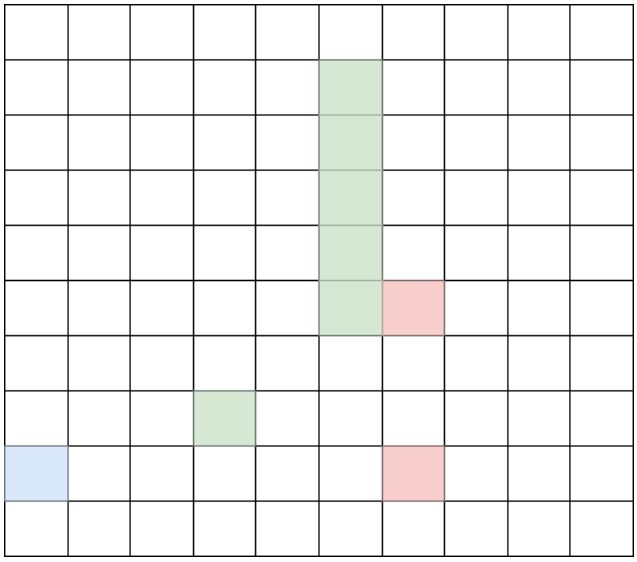

Figura 44. Distribución para el segundo caso de prueba

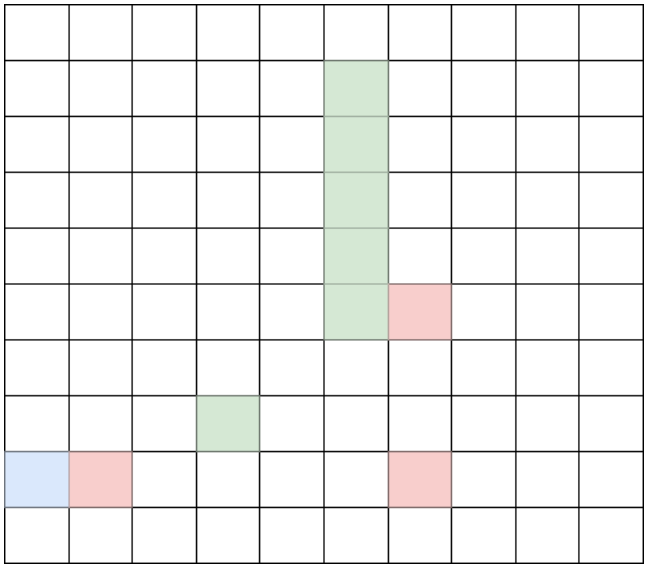

Figura 45. Distribución para el tercer caso de prueba

Implementación

Para fines prácticos del proyecto se definen tres tipos de objetos en el mundo localizados sobre el tablero, dentro de los cuales se encuentra el jugador (casilla color azul), NPC (casilla color naranja) y obstáculos presentes en el escenario (casillas color verde) siendo estos elementos del paisaje como árboles o escombros. Los obstáculos serán representados por un prisma cúbico, logrando obstaculizar la vista para los NPC e inhabilitando localizarse en dicha posición *(véase figura 46),* dando así las pautas para efectuar ataques sorpresa o brindar de dinamismo a los movimientos de cualquiera de los dos personajes.

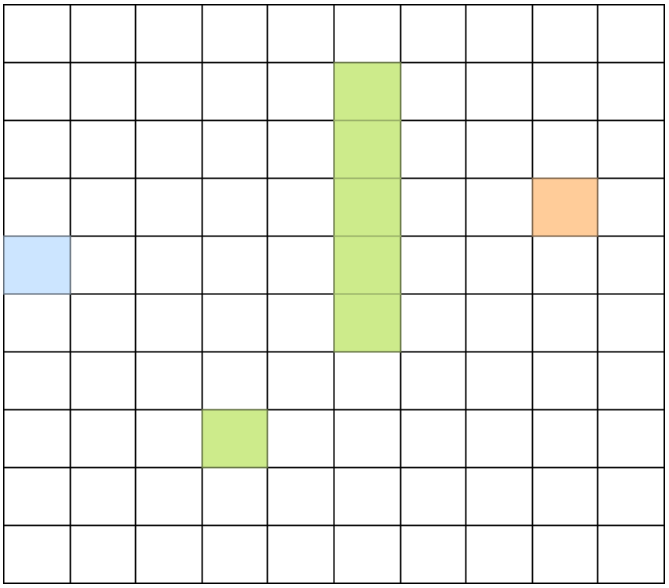

Figura 46. Esquema general del juego

Dadas dichas mecánicas, los dos tipos de personajes presentes en el prototipo cuentan con características que identifican su comportamiento frente a cada situación, validando sus respuestas con respecto al movimiento previo realizado por el jugador. Dando así enfrentamientos que tengan una aproximación lógica a los enfrentamientos que se generen durante la partida, resaltando el carácter de los NPC para su ataque y defensa, teniendo una aproximación a las referencias de las legiones romanas tratadas previamente en el documento.

Haciendo un uso esquemático de los conceptos armamentísticos de la armada, se aislaron los elementos defensivos y ofensivos. Las características principales para el combate se llevarán a métricas para afectar las características propias del actor, como su nivel de experiencia (level), nivel de salud (Health), nivel de equipamiento (EquipLoad), ataque básico (PowerAttackBase), defensa (DefenseBase) y la cantidad de turnos respectivos durante cada partida (TurnsNum). Todo esto, para lograr obtener los promedios de cada variable mediante el avance de los turnos, para lograr definir si el jugador o el propio NPC están sufriendo cambios negativos en su condición.

El peso, resistencia y daño de los equipos, se tomaron como referencias para los principales objetos que caracterizaron las legiones durante sus combates. Estos incluyeron el Hastae, Pugio, Spatha y Maza, siendo las armas predilectas por los diferentes legionarios, llevando a un incremento de 120, 75, 110 y 135 los valores de ataque respectivamente. Adicionalmente se toma como referencia para el apartado defensivo los escudos Scutum, Cetratus y Parma, los cuales entran en acción al momento que el enemigo realice una maniobra de ataque, donde el ajuste de sus valores se da mediante el editor para que cada desarrollo ajuste dichas métricas a su preferencia.

En cuanto a los objetos de defensa corporal, se tomó como referencia los usados por los diferentes rangos de las escuadras, para que de dicha forma esto se viera reflejado acorde al rango y experiencia de cada actor, teniendo como referencias los cascos, como lo son el imperial, con un peso que incrementa en 15 los valores de nivel de equipamiento y en 33 la defensa, el casco gálico imperial, con unas estadísticas de 18 puntos para el nivel de equipamiento y 43 para la defensa, ligado a protección del torso con los diferentes tipos de loricas, como lo son Ligera, Hamata, Squamata, Plumata, Segmentata y Musculata, donde se inicia con métricas de 45 para su nivel de

equipamiento y 40 para la defensa, con un incremento de cinco en cada valor respectivamente, teniendo así el equipamiento total acorde a su ataque y defensa, bajo el cual afronta las situaciones de peligro.

Tomando como referencia los aspectos característicos de un actor que posee la cualidad de ser manipulado por el jugador, se abstraen dichos conceptos mediante herencia al comportamiento y propiedades del NPC, con distinción de armaduras y atributos de ataque o defensa, pero contando con el agregado de tener personalidad, donde estos atributos definen su comportamiento frente a las diferentes situaciones que el jugador le plantee.

De acuerdo a las diversas situaciones que se pueden generar debido a una acción ofensiva o defensiva de alguno de los dos actores, y el peso que puede llegar a tener el equipamiento indicado, se debe tener en cuenta la percepción espacial del tablero, para así lograr identificar la ubicación de cada objeto que influya directa o indirectamente en las acciones de respuesta, por lo que se determina el manejo del tablero a manera de matriz numérica, en donde de acuerdo a su valor se puede identificar qué tipo de objeto se encuentra sobre él, tomando valores en un rango de menos uno a uno, siendo cero el caso de no contar con un objeto sobre él o el propio jugador, un valor de uno para los casos de un NPC, y de menos uno para todos los obstáculos posibles en escena *(véase figura 47).*

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	-1	0	0	0	0
0	0	0	0	0	-1	0	0	0	0
0	0	0	0	0	-1	0	0	1	0
0	0	0	0	0	-1	0	0	0	0
0	0	0	0	0	-1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figura 47. Valores de casillas en el tablero

Al contar con la ubicación de los elementos sobre todo el tablero, se debe considerar la percepción de cada NPC para lograr identificar el área bajo la cual pueda percibir a su rival, con el fin de ejecutar alguna acción. Para esto, la percepción o visión para detectar enemigos (EnemyPerception). La detección toma como base números enteros, ya que se referencia cono la cantidad de casillas que tendrá su radio de visión, para definir la cantidad de casillas bajo las cuales puede percibir la proximidad con respecto a su entorno *(véase figura 48)*, dando el campo de visión en dichas zonas mediante la incidencia de la circunferencia a lo largo del espacio.

Incluyendo los diferentes tipos de objetos en la proximidad del NPC, se logra percibir los obstáculos, compañeros o el mismo jugador, efectuando proyecciones del radio a la altura media del NPC con diferentes ángulos, delimitando y abarcando el área en su totalidad, para así lograr identificar los choques con los distintos elementos presentes en el escenario, donde para efectos prácticos, de contar con elementos en su percepción, se crea una colisión entre el ente emisor y el receptor, llevando a interrumpir la proyección de los vectores dado su choque.

Lo que toma provecho de la unión del sistema de objetos presentes en el tablero, ya que mediante sus colliders se crean colisiones entre los elementos y la casilla sobre la cual se encuentran, iniciando una activación de un estado, el cual depende de la clasificación del modelo, para lograr saber si es un obstáculo o nos enfrentamos al jugador, por lo cual se buscan estos choques y cómo pueden generar sombra al tener la interacción entre la fuente y el elemento con el que colisiona. Para esto se crean proyecciones de los vectores normales del plano creado por la circunferencia con respecto al tablero, a fin de encontrar las incidencias mediante la procedencia de la zona del plano en el mapa.

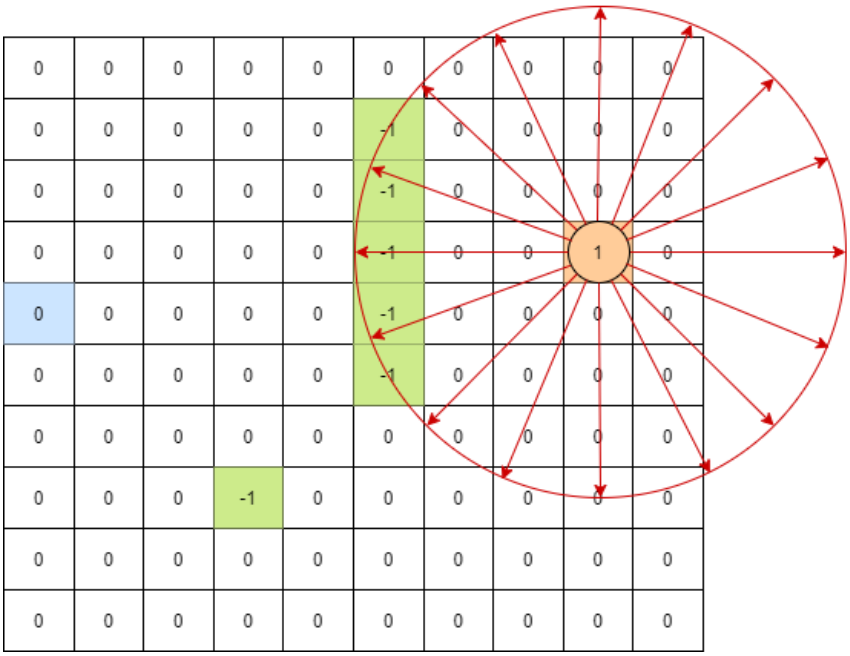

Figura 48. Percepción del NPC

Estos choques llegan a catalogarse de dos formas diferentes, determinados por las proyecciones que se dan luego de la colisión, lo que segmenta el tramo mediante dos secciones, siendo el tramo sin colisión y aquel que ya parte de una, lo que incide directamente en la proyección de sus normales, ya que se caracterizan con el mismo comportamiento del tramo padre, tal como se explicó previamente. Tomando como referencia la cantidad de choques con la casilla, se logra definir la cantidad de visión que se aplica a esta, todo esto, mediante el porcentaje de colisiones que tuvo en cuanto a cada tipo.

Proporcionando un total de 13 choques en las casillas donde tiene una cobertura completa, y de un menor número en aquellas que su área de incidencia no abarque su totalidad, logrando obtener un porcentaje de colisiones que llevan a determinar qué tipo de percepción prevalece más. En base a estos parámetros, se validan los casos bajo los cuales una casilla puede estar afectada,

teniendo una propiedad de visión completa en aquellas donde la mayoría de los vectores vengan de una proyección que no ha sido intervenida por un choque, con percepción parcial en aquellas que no se tenga la totalidad de vectores, y de percepción nula en aquellas que los vectores provienen en su totalidad o mayoría de proyecciones colisionadas *(véase figura 49).*

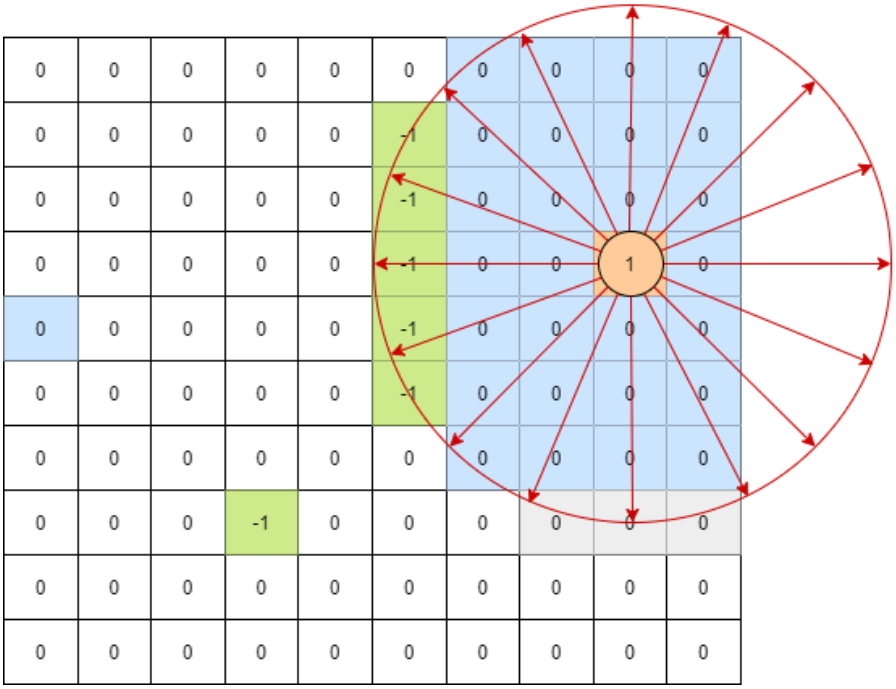

Figura 49. Distribución de la percepción al tablero

Llevando a percibir al jugador que se presente en alguna de las casillas donde se tiene visión, esto con el fin de lograr efectuar alguna acción que dé respuesta al acercamiento realizado por este actor, para lograr efectuar un combate directo o desplazamientos que logren encontrar una ubicación mejor para el NPC. Todo esto se valida con la locación de ambos actores, a fin de determinar el correcto movimiento para que sea representado tal como el árbol lo determinó, contando con un acercamiento estrictamente directo hacia el jugador, mientras que para la huida, se tiene una variedad de posibilidades en las direcciones, de acuerdo al arco posterior que separa a estos actores *(Véase figura 50 y 51).*

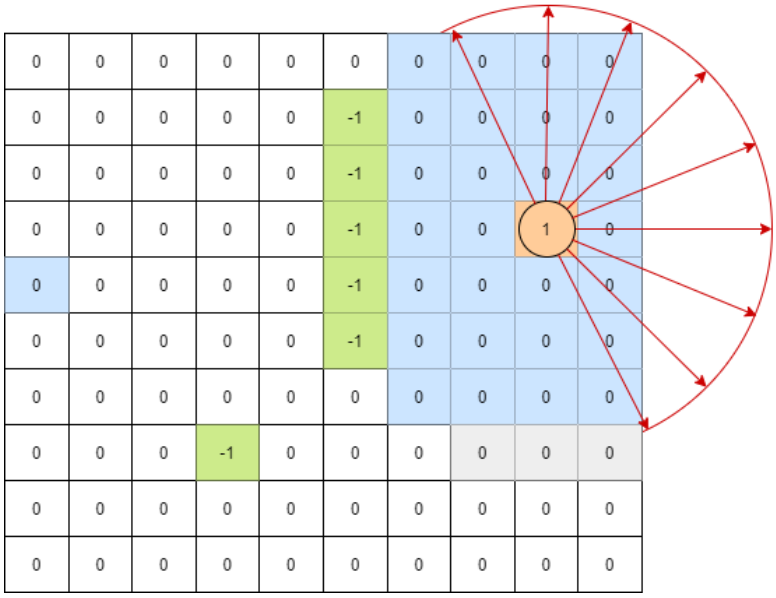

Figura 50. Distanciamiento del NPC con respecto al jugador

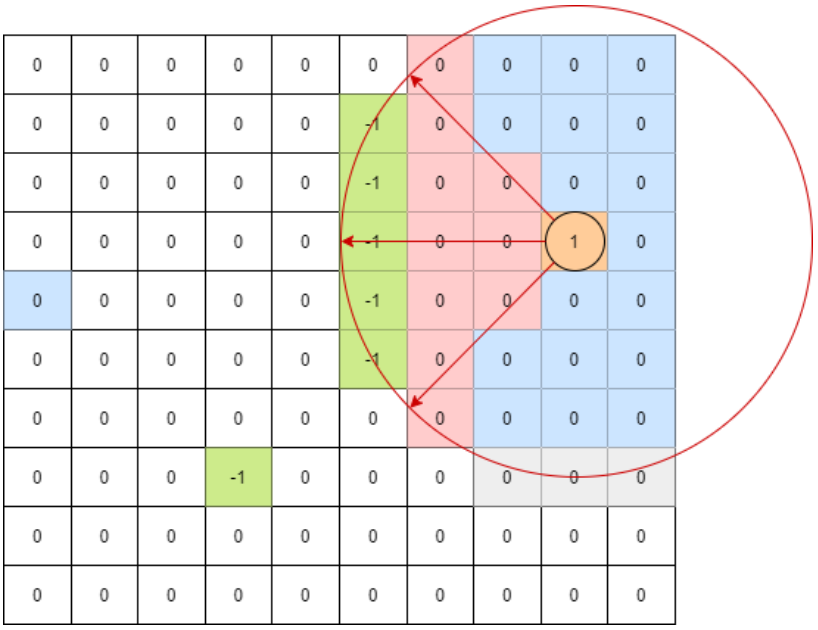

Figura 51. Acercamiento del NPC con respecto al jugado

Donde el sistema de turnos se genera aleatoriamente *(véase figura 51)*, tomando como valor base dos movimientos de 15 segundos cada uno, pero en casos extraordinarios se puede obtener tres rondas mediante el cálculo aleatorio que define la cantidad de movimientos que tendrá cada actor, lo que lleva a una alternancia entre rondas para los objetos entre los cuales está el jugador y la distinta cantidad de NPC que pueda llegar a tener todo el nivel o mapa *(véase tabla 8)*, en donde para cada sucesión de opciones para el personaje no jugable se validan los estados en su entorno, como lo puede ser la muerte de un compañero, ataque por parte del jugador, el equipamiento del

mismo o su entrada en el modo Berserker tal y como se ha explicado previamente que comunica con el plug-in en donde envía las variables para el posterior procesamiento y toma de decisiones.

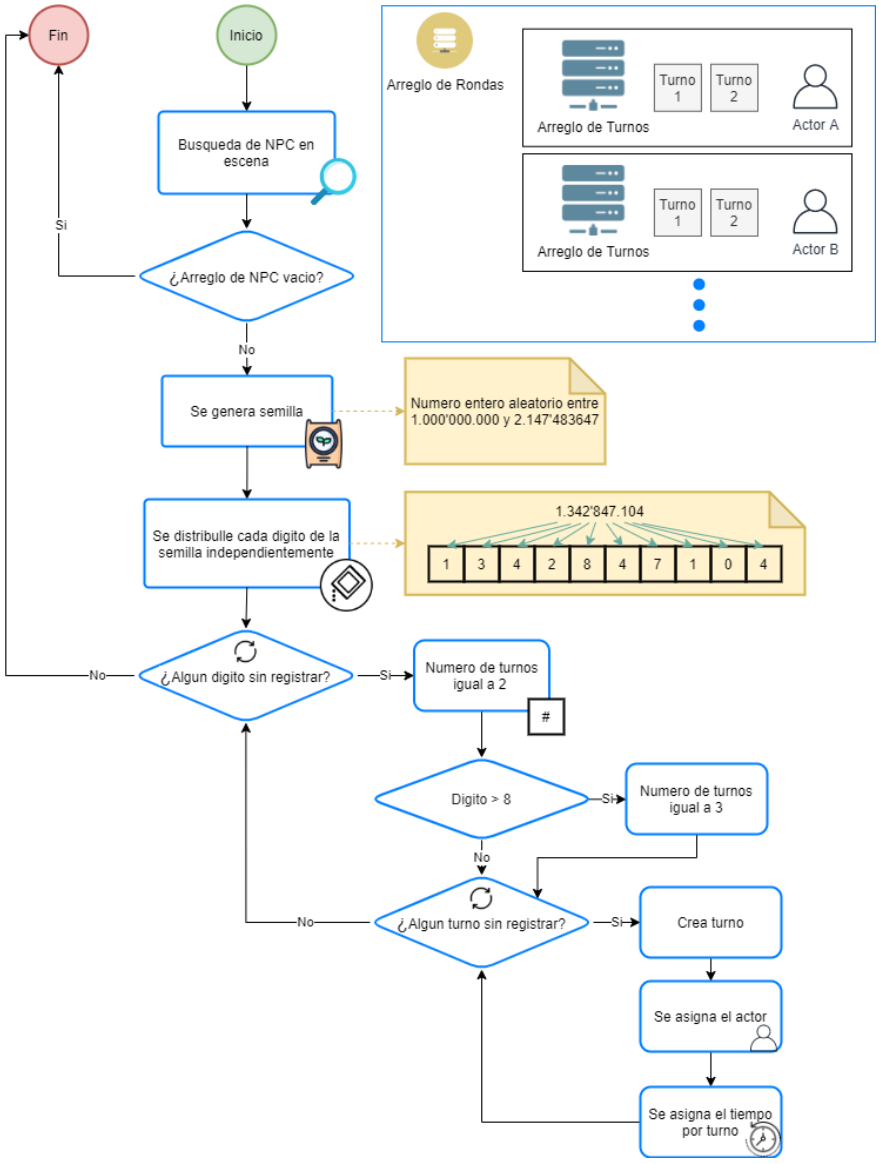

Figura 52. Flujo de generación aleatoria de turnos

PROCESO DE TURNOS	
JUGADOR	TURNO 1 - 15 SEG
	TURNO 2 - 15 SEG
NPC 1	TURNO 1 - 15 SEG
	TURNO 2 - 15 SEG
NPC 2	TURNO 1 - 15 SEG
	TURNO 2 - 15 SEG
JUGADOR	TURNO 1 - 15 SEG
	TURNO 2 - 15 SEG
	TURNO 2 - 15 SEG

Tabla 8. Sistema de gestión de turnos

Para el procesamiento de las acciones que ha de tomar el NPC en las diferentes circunstancias que el jugador plantee, se obtienen las métricas de vida propia, equipamiento del jugador y cantidad compañeros. Estas métricas se registran al comenzar el turno, siempre y cuando esté añadido el complemento de cerebro en el actor no jugable, el cual se puede incluir desde el contenedor de medios del plug-in. Este se encarga del cálculo de sus respectivos porcentajes mediante el valor actual comparado al base definido en su creación, en donde cada uno de estos datos afecta los valores del paquete de estado que tiene cada actor no jugable.

Los valores del paquete de estado son afectados por las métricas que se perciben en el entorno y de cada actor, el miedo se ve afectado por la distancia entre el jugador y el NPC, junto al porcentaje de vida actual propia y del enemigo, la fe cambia de acuerdo a la vida que se tenga en dicho turno, la capacidad de armadura del enemigo y la distancia que los separa, y la venganza se referencia directamente con la cadencia en sus compañeros, esto debido a los ataques del enemigo que lleguen a acabar con ellos. Esto determina las variables del paquete de estado, siendo modificadas para definir la personalidad del NPC.

Una vez el paquete esté listo para ser mutado por la personalidad definida para el NPC, se inicia un proceso para identificar el cambio que pueda llegar a tener la identidad. Mediante un crecimiento o decremento en cada parámetro se determina si la personalidad se ve alterada de alguna de estas formas, basado en la acción que el jugador acaba de plantear sobre él *(véase figura*

53). Adicionalmente, se generan los cambios finales al paquete de estado para ser enviado al árbol de decisión, segmentando las acciones posibles y decidiendo cuál es mejor a partir de su funcionamiento interno.

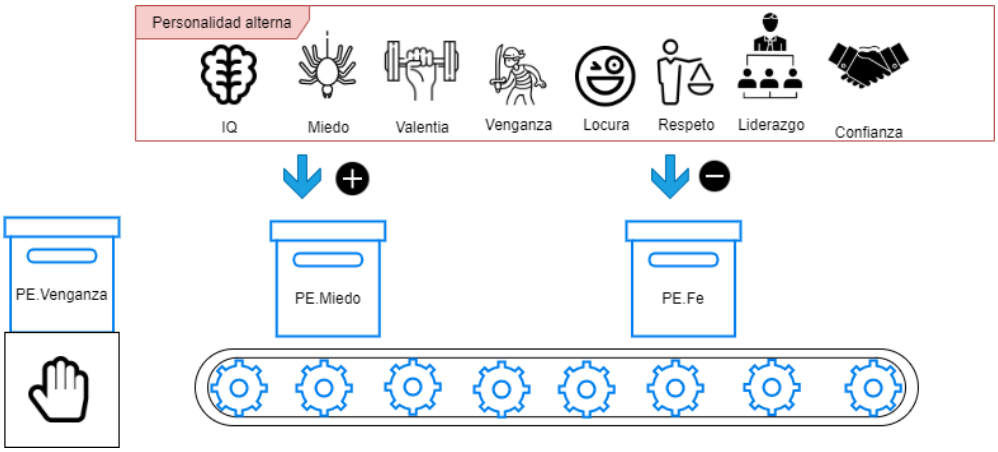

Figura 53. Sistema de mutación de paquete de estado

Al recibir el paquete de estado en el árbol este ejecuta la creación del nodo raíz que parte del estado actual del NPC siendo este referenciado como un objeto que hospeda las variables de nivel, vida máxima y actual, equipo máximo y actual, poder de ataque, nivel de defensa, número de turnos, rango de ataque y número de movimientos realizados por el NPC. Así mismo la vida, poder de ataque y defensa del enemigo, por lo cual en su estado de padre comience la creación de las posibles acciones que se pueden realizar tomando como base cada valor del paquete de estado. Como ejemplo el caso en el que un valor elevado de miedo, no se realice un acercamiento, y que la acción de alejarse no pueda ser aplicada por más de un turno para poder generar movimiento en las acciones del NPC *(véase figura 54)*, lo que detona en recompensas por cada simulación, seleccionando la opción óptima a retornar finalmente *(véase figura 55)*.

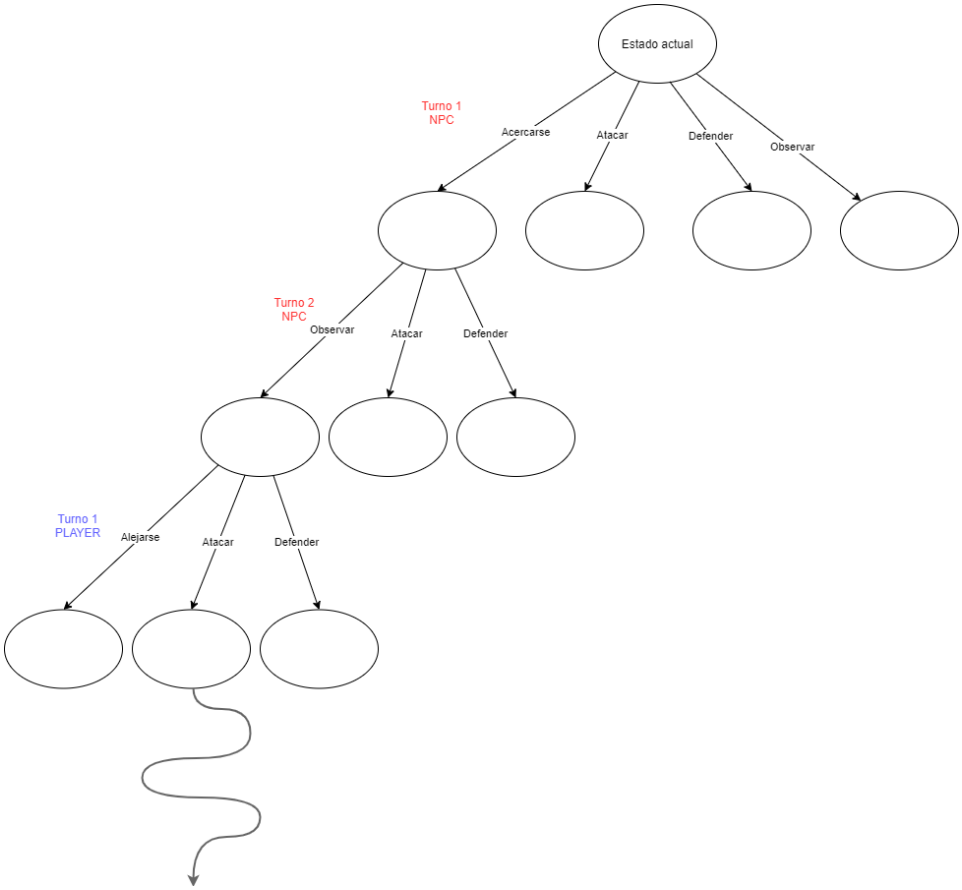

Figura 54. Proceso de simulación gráfico

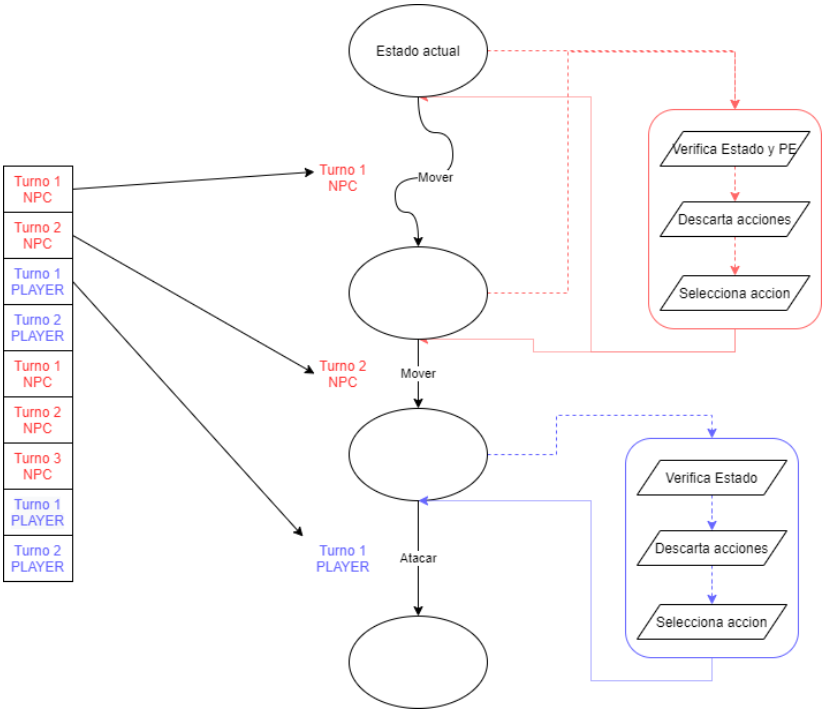

Figura 55. Validación de proceso de simulación

Determinando el peso que va a tener cada acción que se llegue a tomar durante la simulación, se valida el proceso a partir de la premisa de descartar acciones que otorguen un comportamiento errático, o que fomenten la eliminación del enemigo en una cantidad máxima de movimientos para el NPC. Es por esto que, durante las diferentes acciones, los valores de vida representan una varianza de cada puntaje interno del nodo, variando de acuerdo a las acciones que genera el enemigo, esto en caso de atacarse incrementalmente llegando a conmutar dichos valores en el porcentaje de vida para definir la cantidad restante. Lo que brinda un porcentaje mayor o menor de acuerdo a la intención de ataque o defensa, por lo que las acciones de movimiento y observación no representan una variación en el premio directamente, pero si al aumentar la cantidad de pasos.

Junto al cálculo de movimientos durante la simulación y el manejo del premio, se define un ajuste final para poder dar inicio al proceso de retro propagación, en donde se valida la cantidad de movimientos que empleó para tomar la decisión seleccionada, tomando en cuenta el cambio en la vida del jugador, así como la del propio NPC. En este se valida la profundidad del árbol, para obtener una relación de la cantidad de movimientos realizados con respecto a los posibles, ya que el realizar la acción seleccionada en un menor número de turnos, obtiene una mejor recompensa, mientras que al realizarla en una cantidad mayor el valor del premio disminuye.

Al contar con el valor numérico de la mejor opción encontrada por el MCTS, este se encarga de enviar la mejor acción a manera de enumerador al juego, con el fin de que el NPC realice la acción para dicho turno, en donde al aplicarlo al demo desarrollado para las pruebas del mismo, el NPC realiza esta acción, lo que consume uno de los turnos que tiene habilitados. Lo que se repite la cantidad de rondas y turnos que llegue a durar la partida, determinando así un ganador que puede ser el jugador o el NPC.

Capítulo 4. Resultados

La primera prueba requirió que el jugador realizara un acercamiento hacia el NPC llegando a quedar en la casilla junto a él *(véase figura 56)*, para luego efectuar un ataque hacia el NPC. El proceso que se repite en los tres subgrupos de esta primera prueba, para esto se definió un NPC de nivel bajo, el cual activo el plug-in para poder determinar su primera acción en el momento que el jugador entra a su área de visión. El primer movimiento realizado por el contrincante, da inicio al proceso, y se verifican los recursos de ambos actores, lo que genera la ejecución del árbol con el fin del retorno de la acción que ha de ejecutar. Al ejecutar el árbol se determina cuál es su respectivo paquete de estado al verse afectado con el acercamiento y el ataque realizado por el jugador como se explicó en el protocolo de pruebas.

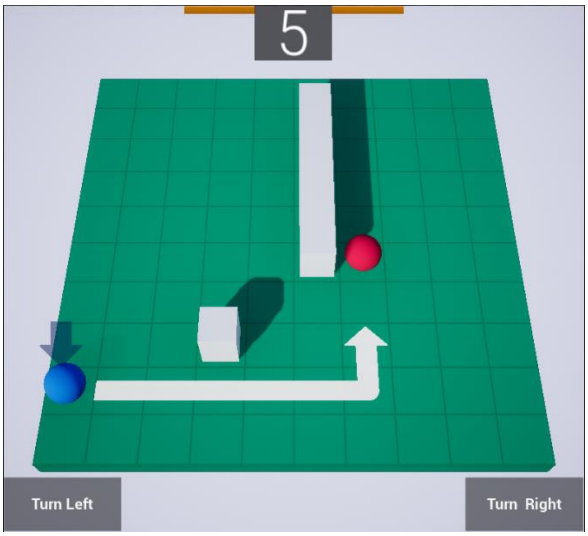

Figura 56. Movimiento prueba 1

Las acciones afectan al paquete de estado en sus variables, donde se define una cantidad de miedo que sobrepasa el 50% de su máximo valor, un valor de fe que de igual manera no sobrepasa la media de su valor total y con una venganza que expresa su cantidad con el mayor número posible en el caso del NPC de nivel básico. Para el segundo caso de prueba se tiene una proyección de miedo que es inferior a la media del valor, una cantidad de fe semejante a la de miedo. Finalizando se tiene al NPC avanzado que sobrepasa las características que tiene el jugador, para este actor se tiene un valor porcentual superior a la media del miedo, una fe superior a la de los tres casos, y de igual manera un nivel de venganza total *(véase tabla 9)*.

	Prueba 1		
	Entorno 1	Entorno 2	Entorno 3
	NPC Básico	NPC Medio	NPC Avanzado
 Miedo	0.5902	0.4751	0.5188
 Fe	0.3657	0.4265	0.5311
 Venganza	1	1	1

Tabla 9. Paquetes de estado prueba 1

De acuerdo a los paquetes de estado que definen a cada NPC en la validación de la respuesta para la acción realizada por el jugador, se determinó el proceso secuencial que dará el indicador para poder ejecutar el movimiento en el tablero. El entorno número uno donde se define la reacción del personaje de nivel bajo, con un árbol que determina dos posibles acciones que son el ataque y el bloqueo, determinando las mejores opciones y el peso que cada movimiento influencia en la acción final, el cual se inclina por el bloqueo ya que cuenta con un puntaje de 321.9 puntos contra los 2.45 del posible ataque *(véase figura 57)*, junto a esto, para el entorno número dos se determina la secuencia de manera semejante, en donde en contraposición al comportamiento del NPC de menor experiencia, se determina que su mejor acción es realizar un ataque, en donde se percibe un número seis veces mayor *(véase figura 58).*

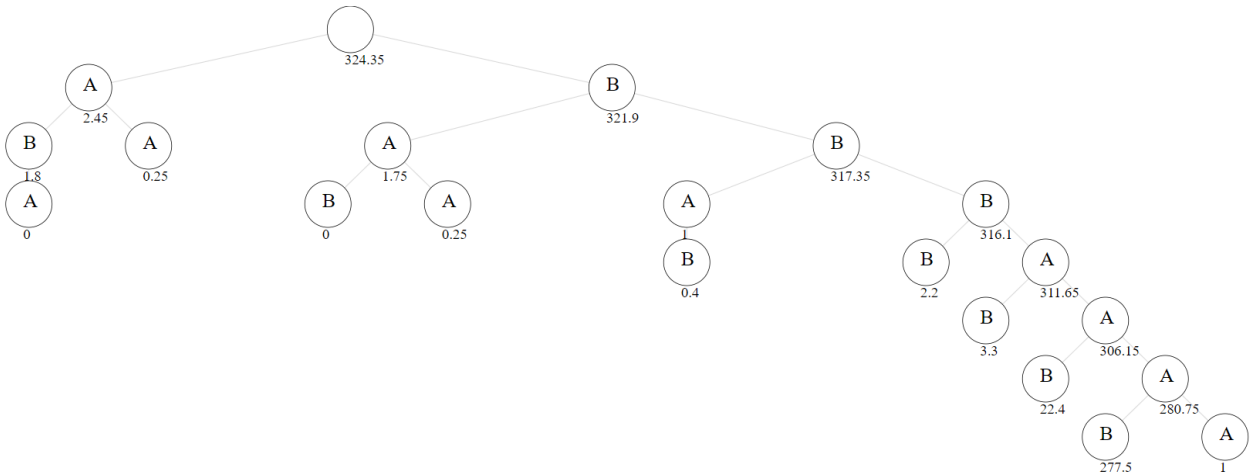

Figura 57. Árbol prueba 1 - NPC bajo

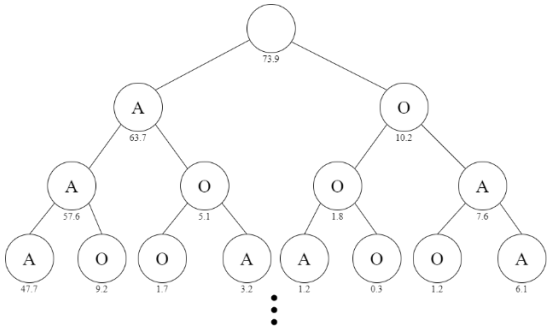

Figura 58. Árbol prueba 1 - NPC medio

En el caso del actor con mayor rango que sus compañeros e incluso que el propio jugador, el comportamiento del árbol se da de manera semejante al actor de nivel medio, en donde la mayor cantidad de nodos se enfocan en el ataque. Esto resultó en una mayor concentración en su propagación y determinando un premio de 99.9 puntos sobre los 27.7 que tiene la opción de observar que logró una profundidad de cuatro niveles en su expansión contra los seis niveles de la acción adversa *(véase figura 59).*

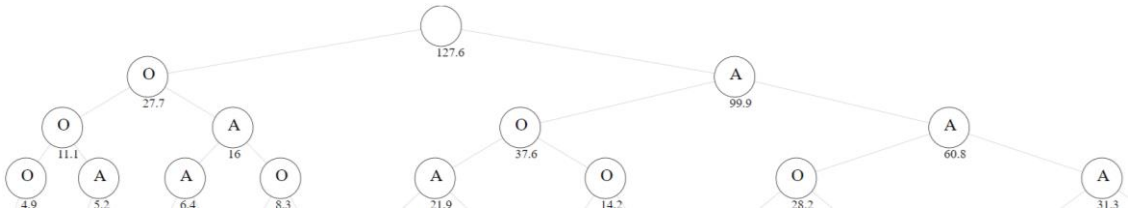

Figura 59. Árbol prueba 1 - NPC avanzado

Para la segunda prueba se realizó un proceso semejante al de la práctica anterior, en donde para los dos NPC que se encuentran presentes en escena, se determinó un ataque al actor que se encuentre en la fila superior, pero permaneciendo entre ambos NPC *(véase figura 60)*. En el entorno uno se observa que las bajas características que tiene el NPC de nivel bajo se presenta una cantidad del 43% de miedo mientras que su contraparte de nivel medio demuestra miedo en su máxima expresión, la fe de ambos se encuentra sobre el 75% o más llegando a ser del 100% para el caso del rango intermedio, y con una venganza nula dado que ninguno ha sido masacrado.

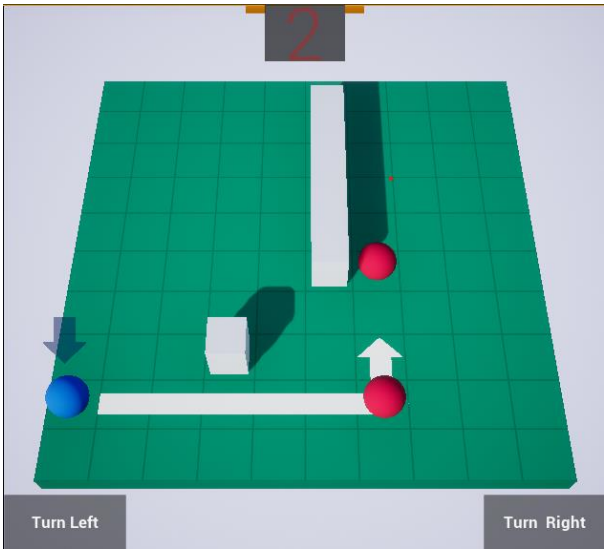

Figura 60. Movimiento prueba 2

En cuanto al entorno dos de la prueba se presentan las combinaciones de un jefe como lo es el NPC nivel avanzado y un subordinado como el de nivel bajo, para el cual se tiene que en ambos casos el nivel de miedo se encuentra superior al 50% pero con un mayor valor para el personaje de nivel bajo, en cuanto a la fe se manejan valores por encima de la media con un 78% para el personaje de nivel bajo, y un 69% para su compañero y de manera semejante a las anteriores, sin porcentaje alguno de venganza. Para el entorno final, se tiene un comportamiento semejante a la prueba realizada en el entorno dos, teniendo valores más altos en miedo el personaje medio y un mayor valor en la fe *(véase tabla 10),* como complemento para el entorno tres, se ejecutó una prueba en la cual se permitía la muerte del NPC avanzado a manos del jugador, lo que disminuyó un 16% para el miedo y un 42% en la fe, junto a un valor de venganza total del 100% *(véase tabla 11).*

	Prueba 2					
	Entorno 1		Entorno 2		Entorno 3	
	NPC Básico	NPC Medio	NPC Básico	NPC Avanzado	NPC Medio	NPC Avanzado
Miedo	0.439	1	0.8163	0.5574	0.7305	0.423
Fe	0.7966	1	0.7891	0.6937	0.8462	0.6732
Venganza	0	0	0	0	0	0

Tabla 10. Paquetes de estado prueba 2

Prueba 2 - Entorno 3 - NPC Medio	
 Miedo	0.5766
 Fe	0.4282
 Venganza	1

Tabla 11. Paquete de estado - prueba 2 - entorno 2 con muerte de compañero (NPC avanzado)

Con la definición del paquete de estado para cada entorno y NPC respectivamente se da origen de un árbol de comportamiento para cada actor y en cada entorno. En primera medida para el entorno uno se observa que el NPC de nivel bajo reporta un accionar de ataque pese a que tuvo tres opciones para realizar, siendo observar, bloquear y atacar *(véase figura 61)*. Para el NPC de nivel medio se presenció que determina como mejor acción acercarse para poder hacer parte del combate *(véase figura 62)*. De manera semejante en el entorno dos se acota la cantidad de posibilidades a realizar, donde aquel que cuenta con el nivel bajo determina que la mejor opción pertenece a observar *(véase figura 63)* y para el actor con mayor nivel se da con tres alternativas, de las cuales opta por atacar con un mayor premio al final de la simulación *(véase figura 64)*.

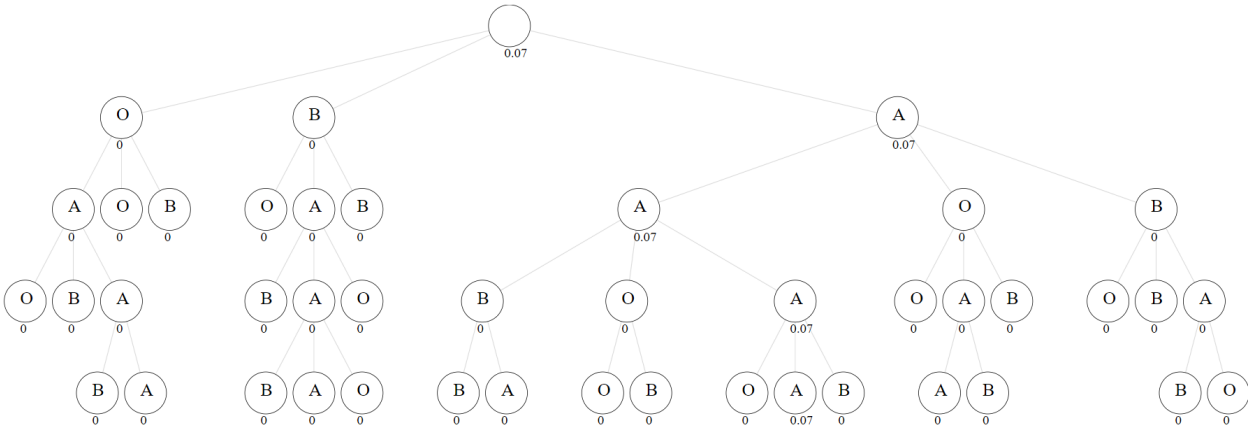

Figura 61. Árbol prueba 2 – entorno 1 – NPC bajo

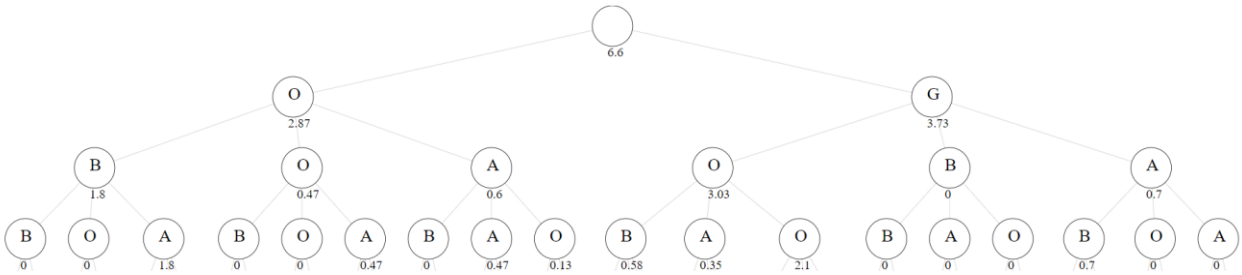

Figura 62. Árbol prueba 2 – entorno 1 – NPC medio

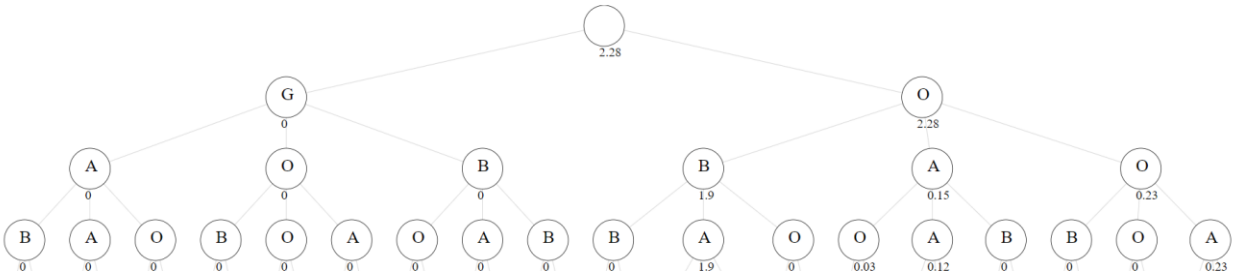

Figura 63. Árbol prueba 2 – entorno 2 – NPC bajo

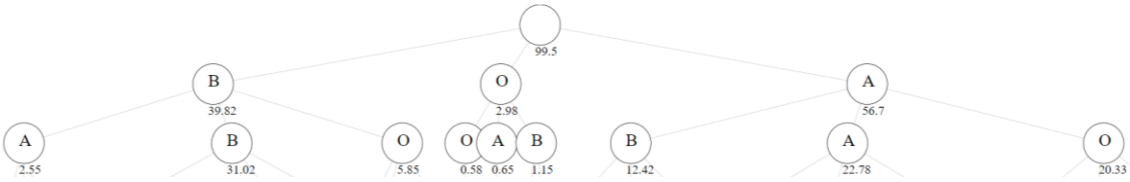

Figura 64. Árbol prueba 2 – entorno 2 – NPC medio

Para el último entorno de la prueba dos, se puede apreciar que en el NPC de nivel medio su árbol desemboca en tres posibles acciones en la cual prima el bloqueo sobre un posible ataque u observar *(véase figura 65).* De manera similar se definen las mismas tres posibilidades que en el árbol de su compañero, más esta vez se opta por un ataque *(véase figura 67).* Junto a esto como se ha mencionado anteriormente, con la prueba se empleó el asesinato del NPC con mayor nivel, lo que lleva a que en segunda medida el personaje no jugable restante determine cual acción lo llevará a tener un mejor enfrentamiento siendo que está solo, lo que conlleva a emplear un ataque por 30 puntos más que la opción de observar *(véase figura 66).*

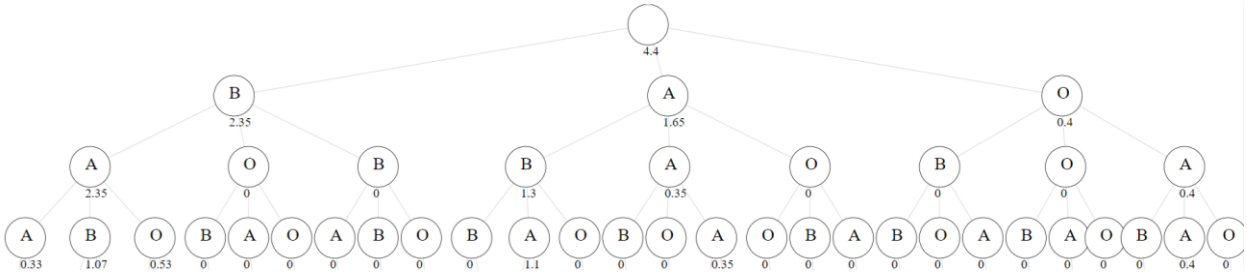

Figura 65. Árbol prueba 2 – entorno 3 – NPC medio

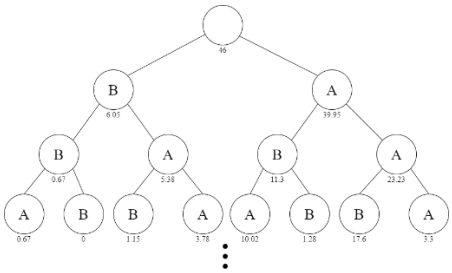

Figura 66. Árbol prueba 2 – entorno 3 – NPC medio – muerte compañero

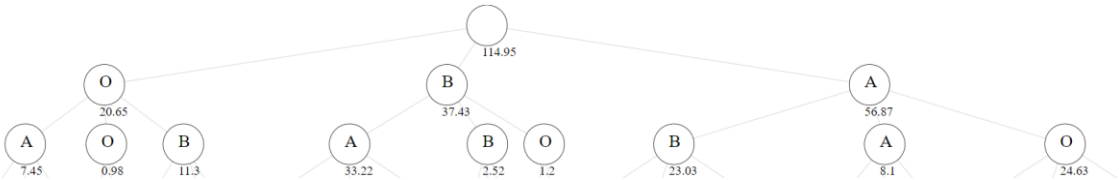

Figura 67. Árbol prueba 2 – entorno 3 – NPC avanzado

Como instancia final, en su tercera fase los movimientos del jugador se aplican al NPC más cercano, siendo este, el situado frente a él con un menor nivel, al que se le ejecutan dos ataques en cadena para así comenzar con el flujo de decisiones correspondiente a cada actor *(véase figura 68).* Identificando cada paquete de estado para los actores, en donde el miedo de cada uno se encuentra bajo la media, aunque el NPC de nivel avanzado tiene una mayor cantidad con 49%, en cuanto a la fe se evidencian valores superiores al 50%, y un 71% para el de mayor nivel, culminando con valores de venganza nulos.

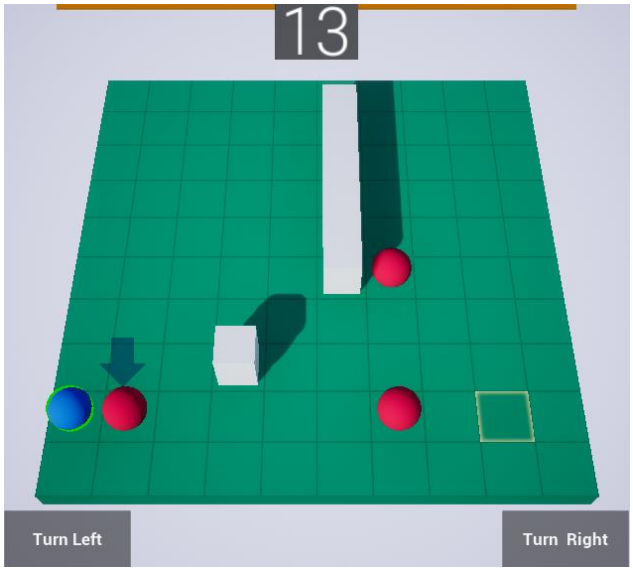

Figura 68. Movimiento prueba 3

	Prueba 3		
	NPC Básico	NPC Medio	NPC Avanzado
 Miedo	0.4594	0.4553	0.4906
 Fe	0.6090	0.5991	0.7066
 Venganza	0	0	0

Tabla 12. Paquetes de estado prueba 3

En cuanto al comportamiento de los árboles, se percibe que para el NPC de nivel bajo la selección de opciones se determina mediante el bloqueo, el ataque u observar, para el cual establece valores bajos en base a la meta, por lo que entre las opciones restantes se destaca como mejor opción el ataque, ya que representa una mejor ganancia *(véase figura 69)*. Referente al NPC de nivel medio se da un manejo de elecciones semejante al anterior, solo que para este caso y con los datos del escenario, tales como de actores, posiciones, etc., decide realizar un bloqueo contra el jugador *(véase figura 70)*, y finalmente se da la misma acción para el personaje nivel avanzado donde la diferencia de valores se caracteriza por no ser muy pronunciada *(véase figura 71).*

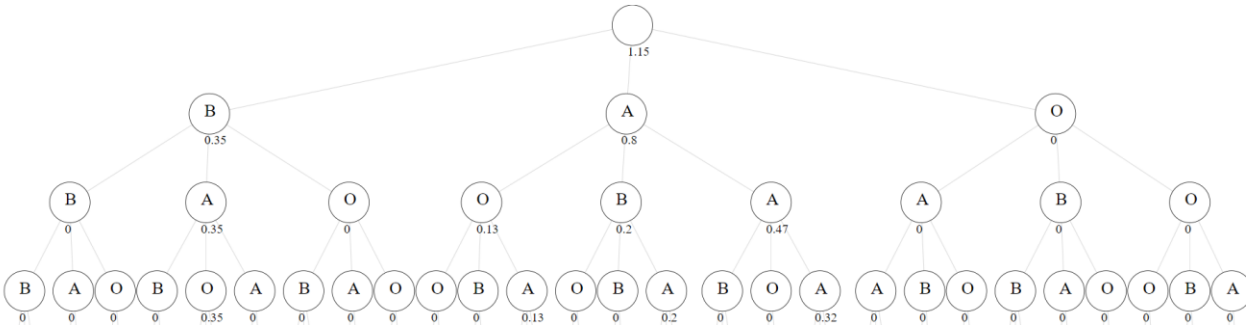

Figura 69. Árbol prueba 3 - NPC bajo

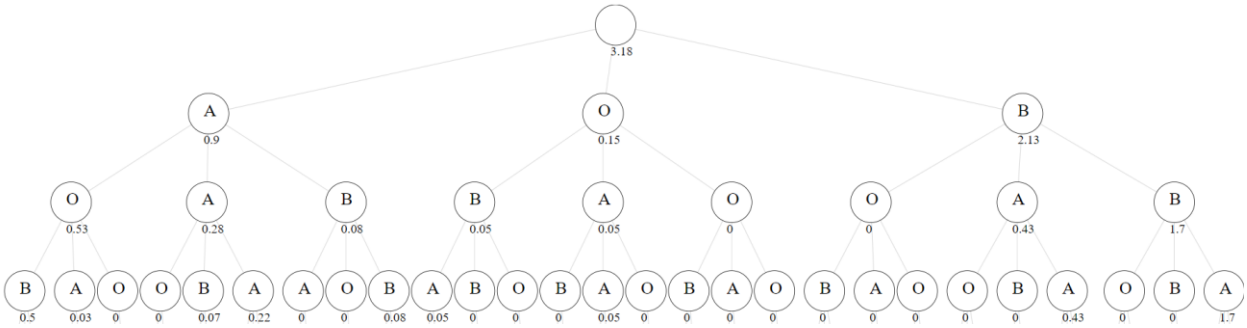

Figura 70. Árbol prueba 3 - NPC medio

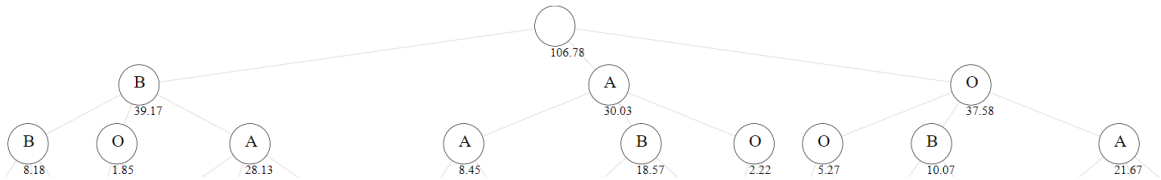

Figura 71. Árbol prueba 3 - NPC avanzado

Capítulo 5. Discusión

Para el conjunto de pruebas aplicado, se percibe que entre las distinciones de rango que se definen en la conceptualización de pruebas, entre mayor nivel tenga el NPC se presentarán tomas de decisiones que afecten más a la distribución de nodos del árbol y que su diferencia de valores no presente una gran dispersión para la toma de decisiones. Cabe destacar que la presencia en escena de dos o más NPC denota un comportamiento diferente a aquel que se ve en las circunstancias donde afrontan al jugador de manera individual, llevando a alterar el valor de venganza o de miedo en base a las etapas de mutación e inyección de valores que se desatan a partir de los actos del jugador.

Como se evidenció en la primera prueba, luego de que el jugador efectuase sus movimientos correspondientes, tales como el acercamiento y ataque a un único NPC en escena, cada actor generó su propio paquete de estado para ser enviado al árbol con su procesamiento lógico, lo que afecta la toma de decisiones, algo que se puede notar es que en los tres casos se proyecta un valor de venganza del 100% *(véase tabla 9).* Este fenómeno sucede al encontrarse sin aliados en escena y el cálculo de la variable depende directamente de esta característica, por lo que su valor es disparado a su máxima expresión. Con base a la aleatoriedad de suma o resta de la variable del miedo, presente en su personalidad, se puede ver que tienen un rango medio y que la diferencia la representa el número aleatorio en la mutación del paquete de estado ya que tenemos rangos de 0.0 a 0.125 para alterar el valor base *(véase tabla 9).*

Junto a la definición del miedo en el paquete de estado, se puede observar que la fe presenta un crecimiento acorde al rango en el cual se encuentra cada actor, aunque no muy sesgado debido a que los datos de la personalidad no se diferencian en gran medida para el apartado de confianza, liderazgo y valentía ya que con estos valores se fomenta el carácter del NPC que afecta directamente la mutación del paquete de estado antes de ser enviado al árbol *(véase tabla 2, 4 y 6).*

Esta mutación en el paquete de estado de cada NPC, representa la base para que las ramas de los árboles de decisión se establezcan y permitan su expansión y simulación, siendo afectados directamente por la personalidad que muto dichas variables. Tal como se evidencia en el árbol del personaje no jugable de nivel bajo, el cual a pesar que sus valores no son muy alejados de los personajes de nivel medio y alto en miedo y fe, se percibe que la variable de venganza al tener una connotación tan amplia determina que el ataque es una buena opción, y por la parte del miedo se asimila un bloqueo para evitar futuros ataques, lo que lleva a decantarse por la rama de bloqueo puesto que en las simulaciones esta rama o lo aleja más de la muerte, permitiendo más posibilidades de jugadas o directamente acabar con el jugador *(véase Figura 47).*

En contraste, los personajes de nivel medio y alto, en pos de acabar con el jugador, optan por las ramas de ataque, pero se diferencian estos dos árboles por los valores que presentan en dichas ramas *(véase Figuras 48 y 49)*, siendo el NPC de mayor rango el que obtiene una mayor recompensa al retro propagar la rama dedicada a atacar, esto debido a que independientemente de la aleatoriedad de la contienda simulada por nodo, que se ve afectada por las variables de IQ, valentía y liderazgo dentro de la fe del paquete de estado *(véase tabla 6)*, y el estado del personaje no jugable favorece a que siempre gane, mientras que el personaje con estadísticas medias se le dificultan más las simulaciones debido a sus valores inferiores en personalidad y ataque en su estado *(véase tabla 4)*, por lo cual el valor obtenido por contienda disminuye al tener que iterar muchas más veces lo que disminuye el valor de recompensa para esta rama de ataque aunque sigue siendo superior a la rama de observar.

En la prueba número dos se pueden observar grandes diferencias en los valores arrojados por los paquetes de estado de cada personaje, ya que ahora la variable de número de compañeros es diferente de cero y modifica mucho la base con la cual se recorrerán los respectivos árboles; claramente este cambio se puede evidenciar en el valor constante de venganza, la cual es nula al inicio de la partida *(véase tabla 10),* connotando la dependencia de los NPC con otros de su clase que aporta un mayor dinamismo a la toma de decisiones. También cabe resaltar que al existir codependencia entre estos personajes no jugables, las otras variables como miedo y fe, también son afectadas lógicamente.

El cambio de valor para los apartados de miedo y fe del primer entorno para la prueba dos, más específicamente refiriéndonos al NPC de rango medio encontramos un valor atípico *(véase tabla 10),* esto debido a una asignación que fluctúa en gran medida dado el valor de los números aleatorios sumados en la fase de mutación, por lo cual solo se tomaron en cuenta para comparar los entornos dos y tres de la segunda prueba.

Con el conjunto de los cambios porcentuales del segundo entorno de la prueba *(véase tabla 10),* en la fe y el miedo que corresponden al personaje de nivel bajo contra el de rango alto, se evidencia una relación inversamente proporcional al nivel del NPC respecto al miedo, la fe en paralelo se ve afectada por la presencia de un compañero en escena, llegando a aumentar para el personaje de nivel inferior al contrario que en los casos en los que está solo, dado que su personalidad está determinada por rasgos bajos de liderazgo y de valores altos en confianza *(véase tabla 2),* algo que también llega a certificarse en el tercer entorno de la prueba número dos, lo que influyó directamente sobre las decisiones tomadas a lo largo del recorrido del árbol, lo que determino en su gran mayoría el optar por posiciones de combate, es decir, atacar, bloquear y acercarse al jugador.

Ahora bien, en el caso excepcional de la prueba número dos, en el cual se observaron los valores obtenidos en el paquete de estado y las decisiones tomadas en un NPC, que veía morir a otro de nivel más alto *(véase tabla 11),* este arrojó la información que muestra cómo cambio su estado en referencia al del mismo entorno pero con su compañero aún presente en la escena *(véase tabla 10),* llevando a la alteración de las variables como la venganza que llevo a un incremento del 100% y la fe con un cambio del 85% al 43%, conllevando a una alteración en la generación de un árbol más compacto para la toma de decisiones, concluyendo en un ataque con un valor de retro propagación mucho más alto en comparación a su otra opción de bloqueo, lo que se debe al incremento que tuvo el parámetro de venganza en su respectivo paquete.

En cuanto a los resultados de la tercer prueba, la cual se ejecutó en un único entorno con los tres NPC, se muestran valores muy similares al segundo y tercer entorno de la prueba 2, corroborando la codependencia del número de compañeros al evidenciar un valor nulo en la venganza de todos los NPC en el momento de encontrarse los tres en escena, y donde el miedo se mantuvo equilibrado en todos los personajes de manera semejante a como se presenció en la prueba uno, pues esto solo se ve afectado por su propia personalidad y solo aumenta si el jugador ataca a uno directamente, pero en esta ocasión la fe se vio afectada por una mayor cantidad de personajes no jugables dentro del escenario y cuyas características dan soporte al momento de enfrentar al jugador.

Capítulo 6. Conclusiones

Con este trabajo de grado se presentó un plug-in para una IA en el motor de videojuegos Unreal Engine 4, donde se realizó una abstracción del funcionamiento global de cada proceso, generando la construcción de los estados que se enviaran al árbol para su ejecución. Este proceso permitió realizar una implementación dirigida a la funcionalidad del código mediante el uso de librerías específicas que son proporcionadas por el lenguaje propio de la herramienta de desarrollo.

Junto a la implementación del plug-in como aspecto fundamental para el desarrollo de lo planeado, se da un enfoque propio del actor atribuyéndole una personalidad, junto a un estado que afecta el momento del enfrentamiento, modelando y planteando el impacto que tendrá al instante de usar el árbol y tomar las decisiones, lo que en conjunto con un modelo de pruebas que se enfoca directamente a las decisiones en un combate, se parametriza cada árbol de comportamiento que es creado y ejecutado por cada actor, llevando a realizar cierta acción que le es indicada por su cerebro, que dentro del proyecto es conocido como el plug-in.

Al tomar como referencia una cantidad limitada de NPC para realizar las pruebas, se corrobora que la función del plug-in se da de manera correcta en el desarrollo de un videojuego de tipo RPG sobre este motor, dado que cada entorno de test denota un comportamiento que demuestra procesamientos lógicos que se esperarían de un NPC con un grado de inteligencia, ya que logra responder ataques y realizar un desplazamiento que lo aleje del combate en un caso que lo perjudique.

Estos procesos arrojaron información que demuestra que el plug-in depende de la puesta en escena, ya que si el NPC se encuentra acompañado por más actores de su tipo, el complemento se comporta de una manera más dinámica, logrando una toma de decisiones que equiparen a las acciones del jugador, tal como se pudo ver tanto en la prueba dos como en la tres, afectando de igual manera al comportamiento que sufren en casos de miedo, ya que las relaciones con sus compañeros indican un decremento que puede variar dependiendo de su personalidad, y de la cantidad de estos que se encuentren junto a él.

La personalidad individual de cada NPC influye directamente al paquete de estado que será enviado al árbol de decisiones, donde cada parámetro funciona como un conjunto de relaciones que afectan la manera de procesar cada simulación del MCTS, ya que al tener variables con una amplia dispersión, se genera un comportamiento parecido a otro actor que cuente con características inferiores, pero con un mejor balanceo en sus rangos de personalidad, lo que termina sesgando las acciones en las cuales se expandirá el árbol.

Junto a lo dicho anteriormente, cabe resaltar que la falta de mutación del paquete de estado durante el proceso de simulación y expansión, representa una falta de variedad en estos procesos, ya que este cambio representaría un mayor dinamismo en el valor del premio a retornar, al ver como una acción desencadena otras de acuerdo al estado actual dentro de cada iteración de la simulación.

Capítulo 7. Recomendaciones

Para poder realizar acciones con un mayor peso lógico y coherente a lo que esperaría un jugador, se recomienda realizar una investigación acerca del comportamiento racional humano, enfocándose en como la personalidad de un sujeto afecta las decisiones que toma para reaccionar a una circunstancia, con el fin de mejorar el cálculo que valida la personalidad y el estado del NPC para que así mismo el árbol se genere con las acciones más justificadas.

Junto a esto, también vale la pena plantear un desarrollo para los ambientes de blueprints como de código para la implementación del plug-in, para poder tenerlo esquematizado de dicha forma que el manejo de interfaces de la conexión del jugo con el plug-in de una manera apropiada a las necesidades de quien desee implementarlo, dados los conocimientos con los que cuente el desarrollador.

Por último, se resalta, la posibilidad de realizar un análisis previo para el manejo de la implementación del árbol en su código fuente, ya que dependiendo de cómo se haga el desarrollo, se manejará la conexión entre componentes, y de esta forma lograr minimizar el flujo que deben recorrer los datos para dar respuesta desde el plug-in al juego.

Capítulo 8. Trabajo futuro

Como trabajo futuro y para tener un mejor comportamiento en la toma de decisiones del árbol, se proyecta establecer una mutación del paquete de estado dentro de cada proceso de simulación y expansión, llevando a que las posibles acciones de cada nivel del árbol no siempre tomen como base las elegidas por el primer paquete de estado; lo que puede complementarse disminuyendo la dependencia que cada NPC tiene respecto a sus compañeros en el caso que este inicie la partida solo en escena.

Otro aspecto que se puede implementar, es mejorar la conexión del plug-in con el editor, ya que al momento se permite únicamente su aplicación mediante código, lo que puede llegar a ser engorroso para un desarrollador sin conocimientos de código o que desee aplicarlo por medio de programación por nodos, lo que lleva a complementar el desarrollo incluyendo interfaces que permitan la conexión por el medio gráfico del motor.

Junto a la mejora en la implementación del plug-in, se debe validar el esquema de comportamientos que presenta cada NPC en las diferentes pruebas, a fin de ejecutar cada protocolo en múltiples casos, y llegar a una distribución del comportamiento que se presente en las diferentes ocasiones, para poder categorizar el actuar de una manera más concluyente.

Bibliografía

Adamchik, V. S. (2009). *Game trees*. Retrieved from https://www.cs.cmu.edu/~adamchik/15-121/lectures/Game Trees/Game Trees.html

Akhtar, R. (2019). *Search Algorithms in AI*. Retrieved from https://www.geeksforgeeks.org/search-algorithms-in-ai/

Alchalabi, A. E., Shirmohammadi, S., & Mohammed, S. A. (2019). QNetwork: AI-Assisted Networking for Hybrid Cloud Gaming. *2019 IEEE International Symposium on Measurements and Networking, M and N 2019 - Proceedings*, *July*. doi: 10.1109/IWMN.2019.8804987

Arre Caballo. (2017). *El ejército romano en el Alto Imperio*. Arre Caballo. Retrieved from https://arrecaballo.es/edad-antigua/alto-imperio-romano/el-ejercito-romano-en-el-alto-imperio/#

Arribas, D. (2017). *¿Por qué la lorica segmentata o armadura de placas que vestían las tropas romanas cayó en desuso?* Quora. Retrieved from https://es.quora.com/Por-qué-la-lorica-segmentata-o-armadura-de-placas-que-vestían-las-tropas-romanas-cayó-en-desuso

Arteaga, M. (2011). *Gladius*. EcuRed. Retrieved from https://www.ecured.cu/index.php?title=Gladius&action=history

Baron, S. (2012). *Cognitive Flow: The Psychology of Great Game Design*. March 22, 2012. Retrieved from https://www.gamasutra.com/view/feature/166972/cognitive_flow_the_psychology_of_.php

Batán, T. (2016). *Revista de Historia Especial Legiones de Roma Índice*. Retrieved from https://revistadehistoria.es/especial-legiones-de-roma/

Bofill, M. (2019). *Lorica Squamata*. Gladiatrix En La Arena. Retrieved from https://gladiatrixenlaarena.blogspot.com/2019/04/lorica-squamata.html

Bors, M. L. (2018). *What is a Finite State Machine?* Mar 10, 2018. Retrieved from https://medium.com/@mlbors/what-is-a-finite-state-machine-6d8dec727e2c

Bossom, A., & Dunning, B. (2016). *Video Games* (Issue June). London: Bloomsbury. Retrieved from https://www.taylorfrancis.com/books/9781351299961

Cant, R., Churchill, J., & Al-Dabass, D. (2001). Using hard and soft artificial intelligence algorithms to simulate human go playing techniques. *International Journal of Simulation: Systems, Science and Technology*, *2*(1), 31–49.

Černý, M., Plch, T., Marko, M., Gemrot, J., Ondráček, P., & Brom, C. (2017). Using behavior objects to manage complexity in virtual worlds. *IEEE Transactions on Computational Intelligence and AI in Games*, *9*(2), 166–180. doi: 10.1109/TCIAIG.2016.2528499

Clement, J. (2021). *Console gaming content market value worldwide from 2011 to 2022, by distribution type*. Jan 29, 2021. Retrieved from https://www.statista.com/statistics/292460/video-game-consumer-market-value-worldwide-platform/

D'amato, R. (2009). *Arms and armour of the imperial roman soldier*. London: Frontline Books. Retrieved from https://books.google.es/books?hl=es&lr=&id=tbvgAwAAQBAJ&oi=fnd&pg=PP1&dq=parma+roman+shield&ots=czuCWGMJbH&sig=cspVVMhAri-Z7GhkJhwGi45EdSc#v=onepage&q=parma&f=false

Denham, T. (2020). *What is Unreal Engine?* Retrieved from https://conceptartempire.com/what-

is-unreal-engine/
Desperta Ferro Ediciones. (2018). *Del pilum a la plumbata*. Retrieved from https://www.despertaferro-ediciones.com/2018/pilum-plumbata/
DOrchymont, A. (2018). *Immersive Citizens - AI Overhaul*. 03 July 20155:00PM. Retrieved from https://www.nexusmods.com/skyrim/mods/65013/?tab=articles
Embid, B. N. (2012). *Método de Monte-Carlo Tree Search (MCTS) para resolver problemas de alta complejidad: Jugador virtual para el juego del Go* [Universidad de Zaragoza]. Retrieved from https://zaguan.unizar.es/record/8010/files/TAZ-PFC-2012-393.pdf
Epic Games. (2004). *Unreal Engine*. Retrieved from https://www.unrealengine.com/en-US/
Epic Games. (2021). *Behavior Tree Overview*. Retrieved from https://docs.unrealengine.com/en-US/InteractiveExperiences/ArtificialIntelligence/BehaviorTrees/BehaviorTreesOverview/index.html
Esposito, N. (2005). A short and simple definition of what a videogame is. *Proceedings of DiGRA 2005 Conference: Changing Views - Worlds in Play, January 2005*. Retrieved from https://www.researchgate.net/publication/221217421_A_Short_and_Simple_Definition_of_What_a_Videogame_Is
Gambus, P., & Shafer, S. L. (2018). Artificial intelligence for everyone. *Anesthesiology*, *128*(3), 431–433. doi: 10.1097/ALN.0000000000001984
Grosso, R. (2015). *Playing Roles: On Tactical-RPGs*. 08/21/2015 - 12:00. Retrieved from https://techraptor.net/originals/playing-roles-on-tactical-rpgs
Gunn, E. A. A., Craenen, B. G. W., & Hart, E. (2009). A taxonomy of video games and AI. *Adaptive and Emergent Behaviour and Complex Systems - Proceedings of the 23rd Convention of the Society for the Study of Artificial Intelligence and Simulation of Behaviour, AISB 2009, May 2014*, 4–14.
Hovermale, C. (2019). *Do you prefer turn-based RPGs or action RPGs?* 20/01/2019 14:00:00. Retrieved from https://www.destructoid.com/stories/do-you-prefer-turn-based-rpgs-or-action-rpgs--539499.phtml
James, S., Konidaris, G., & Rosman, B. (2017). An analysis of Monte Carlo tree search. *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, *July*, 3576–3582.
Jiang, C. (2020). Analysis of Artificial Intelligence Applied in Video Games. *Proceedings - 2020 International Conference on Artificial Intelligence and Computer Engineering, ICAICE 2020*, 142–145. doi: 10.1109/ICAICE51518.2020.00033
Johari, A. (2020). *AI Applications: Top 10 Real World Artificial Intelligence Applications*. Sep 18, 2020. Retrieved from https://www.edureka.co/blog/artificial-intelligence-applications/
Kaelin, M. (2006). *Playing a MMORPG is not all fun and games, you better have the right vocabulary*. May 3, 2006. Retrieved from https://www.techrepublic.com/article/playing-a-mmorpg-is-not-all-fun-and-games-you-better-have-the-right-vocabulary/
Kaplan, J. (2016). *Artificial intelligence*. Oxford: Oxford University Press.
Kelleher, B., & Felicia, P. (2011). *Using Artificial Intelligence to create reactive architectures for Non Player Characters (NPCs) in First Person Shooter (FPS) games . Name : Brendan Kelleher Supervisor : Patrick Felicia Table of Contents :* Waterford Institute of Technology.
Kim, M. J., & Kim, K. J. (2017). Opponent modeling based on action table for MCTS-based fighting game AI. *2017 IEEE Conference on Computational Intelligence and Games, CIG 2017*, 178–180. doi: 10.1109/CIG.2017.8080432
Laguna, D. (2020). *Industria de videojuegos generó más de $120 MMDD en 2019*. Retrieved from https://www.levelup.com/noticias/556175/Industria-de-videojuegos-genero-mas-de-120-MMDD-en-2019

Levine, J. (2017). *Monte Carlo Tree Search*. Retrieved from https://www.youtube.com/watch?v=UXW2yZndl7U&feature=youtu.be

Liberty, S. (2017). *What Are Role-Playing Games Even? How are they that?* Medium. Retrieved from https://medium.com/@SA_Liberty/what-are-role-playing-games-even-how-are-they-that-50071c5552e2

Loyall, A. B. (1997). Believable Agents: Building Interactive Personalities [Carnegie Mellon]. In Proceedings of the International Conference on Autonomous Agents (Issue May). doi: 10.1145/267658.267681

Lucci, S., & Kopec, D. (2009). *Artificial Intelligence in the 21st Century* (2nd ed.). Dulles, Boston, New Delhi: David Pallai.

Malim, G. (2018). *Video games market is worth more than music and movies combined so why aren't CSPs launching games services?* 05 July, 2018 at 1:28 PM. Retrieved from https://www.vanillaplus.com/2018/07/05/40093-video-games-market-worth-music-movies-combined-arent-csps-launching-games-services/

Mbaabu, O. (2020). *Intelligent Agents in Artificial Intelligence*. December 1, 2020. Retrieved from https://www.section.io/engineering-education/intelligent-agents-in-ai/

Merino, M., & Pérez, J. (2011). *Definición de Maza*. Definición De. Retrieved from https://definicion.de/maza/

Nacke, L. (2014). *The formal systems of games and game design atoms. The Acagamic*. September 12, 2014. Retrieved from The formal systems of games and game design atom

Newton, P. L., & Feng, J. (2016). *Unreal Enginee 4 AI Programming Essentials*. Birmingham: Packt Publishing Ltd. Retrieved from https://www.packtpub.com/

Nichols, D. (2014). *History Behind The Game – Ryse: Son of Rome*. February 8, 2014 8:31 AM. Retrieved from https://venturebeat.com/community/2014/02/08/history-behind-the-game-ryse-son-of-rome/

Pardo, A. (2013). *Finite State Machines explained*. YouTube. Retrieved from https://www.youtube.com/watch?v=hJIST1cEf6A

Robertson, G., & Watson, I. (2015). Building behavior trees from observations in real-Time strategy games. *INISTA 2015 - 2015 International Symposium on Innovations in Intelligent SysTems and Applications, Proceedings*. doi: 10.1109/INISTA.2015.7276774

Rockstar. (2021). *Rockstar New England.Code AI Programmer*. 2021. Retrieved from https://www.rockstargames.com/careers/openings/position/4064535003

Rodríguez, V. (2015). *Un nuevo mod de Skyrim mejora la IA de los NPCs*. 12 Julio 2015. Retrieved from https://areajugones.sport.es/videojuegos/un-nuevo-mod-de-skyrim-mejora-la-ia-de-los-npcs/

Sagredo Olivanza, I., Gómez Martín, M. antonio, & González Calero, P. (2014). *Un modelo integrador de máquinas de estados y árboles de comportamiento para videojuegos*. Retrieved from http://gaia.fdi.ucm.es/sites/cosecivi14/es/papers/27.pdf

Sagredo Olivenza, I., Flórez Puga, G., Gómez Martín, M. A., & González Calero Pedro. (2015). *Implementación de nodos consulta en árboles de comportamiento*. Retrieved from http://ceur-ws.org/Vol-1394/paper_14.pdf

Sangyeob Lee, M., & Heester, C. (2015). *Cognitive Intervention and Reconciliation : NPC Believability in Single-Player RPGs believable characters parallels the importance of lack. 5*, 47–65. Retrieved from http://ijrp.subcultures.nl/wp-content/uploads/2015/01/IJRP5_LeeHeeter.pdf

Sarkar, S. (2021). *Nvidia's performance-boosting DLSS tech is now easier to add to some games*. Feb 16, 2021, 2:52pm EST. Retrieved from

https://www.polygon.com/2021/2/16/22285726/nvidia-dlss-unreal-engine-4-plugin
Shummon, L., & Luc, A. (2019). *Artificial Intelligence in Video Games*. Jul 1, 2019. Retrieved from https://towardsdatascience.com/artificial-intelligence-in-video-games-3e2566d59c22
Skinner, G., & Walmsley, T. (2019). Artificial intelligence and deep learning in video games a brief review. *2019 IEEE 4th International Conference on Computer and Communication Systems, ICCCS 2019*, 404–408. doi: 10.1109/CCOMS.2019.8821784
Statt, N. (2019). *HOW ARTIFICIAL INTELLIGENCE WILL REVOLUTIONIZE THE WAY VIDEO GAMES ARE DEVELOPED AND PLAYED*. Mar 6, 2019, 10:00am. Retrieved from https://www.theverge.com/2019/3/6/18222203/video-game-ai-future-procedural-generation-deep-learning
Straeubig, M. (2019). *Games, AI, and Systems*. *10*(1), 141–160.
Tomai, E., & Flores, R. (2014). Adapting In-Game Agent Behavior by Observation of Players Using Learning Behavior Trees. *Fdg*. Retrieved from https://pdfs.semanticscholar.org/c9a7/5c6280536a7a95b707751e919bec8f68e7b3.pdf
Torres, J. L. H. (2020). *Arboles de Comportamiento* [Morelia]. Retrieved from https://www.docsity.com/es/arboles-de-comportamiento/5372577/
Vossen, B. (2015). *Enemy design and enemy AI for melee combat systems*. Retrieved from https://www.gamasutra.com/blogs/BartVossen/20150504/242543/Enemy_design_and_enemy_AI_for_melee_combat_systems.php
Warpefelt, H. (2016). *The Non-Player Character* (Issue 16). Publit.
Wolf, M. J. P. (2008). The videogame explosion - A History from PONG to PlayStation® and Beyond. In M. J. P. Wolf (Ed.), Greenwood Press. Westport, Connecticut.
Xbox Wire Staff. (2015). *Know Your Genres: Action Role-Playing Games*. Aug 22, 2015 @ 12:00am. Retrieved from https://news.xbox.com/en-us/2015/08/22/games-know-your-genres-action-role-playing-games/
Xu, S. (2018). *History of AI design in video games and its development in RTS games*. Retrieved from https://sites.google.com/site/myangelcafe/articles/history_ai
Yannakakis, G. N., & Togelius, J. (2015). A Panorama of Artificial and Computational Intelligence in Games. *IEEE Transactions on Computational Intelligence and AI in Games*, *7*(4), 317–335. doi: 10.1109/TCIAIG.2014.2339221

Anexos

Anexo 1. Link de referencia, al video que detalla gráfica y verbalmente el uso e implementación del plug-in en Unreal Engine 4.

LINK