UNIVERSIDAD ANDRÉS BELLO

FACULTAD DE INGENIERÍA

INGENIERÍA CIVIL INFORMÁTICA

Implementación de Técnicas de Inteligencia Artificial para el Comportamiento Autónomo de un NPC en Videojuegos

Raimundo Armijo Undurraga

Profesor guía: Doctor Pablo Schwarzenberg

SANTIAGO – CHILE

Diciembre, 2024

Implementación de Técnicas de Inteligencia Artificial para el Comportamiento Autónomo de un NPC en Videojuegos

RAIMUNDO ARMIJO, Universidad Andrés Bello, Santiago de Chile

Abstract El desarrollo de agentes virtuales con comportamiento autónomo en videojuegos plantea un desafío significativo dentro del campo de la inteligencia artificial (IA). Este estudio aborda la creación e implementación de un agente enemigo no jugable (NPC) en un videojuego 2D con temática medieval, utilizando técnicas de aprendizaje por refuerzo Q-learning y árboles de decisión. Estas herramientas permiten al agente tomar decisiones autónomas y optimizar su comportamiento en función de recompensas dinámicas, adaptándose a escenarios variables. Además, se integró un sistema de reseteo que fomenta la exploración continua y evita la sobreespecialización en patrones de comportamiento. Los resultados muestran que el agente puede equilibrar eficazmente entre exploración y explotación, dependiendo de las configuraciones de recompensas asignadas, y que el sistema de reseteo potencia la diversidad de estrategias. Este trabajo contribuye al diseño de NPCs más realistas y dinámicos, fortaleciendo las capacidades interactivas y adaptativas en videojuegos.

CCS Concepts: • **Mathematics of computing** → **Statistical software**; • **Applied computing** → **Education**; • **Human-centered computing** → *Empirical studies in HCI*; • **Software and its engineering** → *Domain specific languages.*

Additional Key Words and Phrases: comma, separated

ACM Reference Format:
Raimundo Armijo. 2024. Implementación de Técnicas de Inteligencia Artificial para el Comportamiento Autónomo de un NPC en Videojuegos. 1, 1 (December 2024), 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Contents

Author's address: Raimundo Armijo, r.armijoundurraga@uandresbello.edu, Universidad Andrés Bello, Santiago de Chile.

XXXX-XXXX/2024/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

List of Figures

1 INTRODUCCIÓN

Los avances en inteligencia artificial (IA) han transformado diversos campos, y el desarrollo de agentes virtuales es un ejemplo destacado de esta evolución. Un agente virtual, o agente autónomo, es un personaje no jugable (NPC) que interactúa en entornos digitales imitando comportamientos humanos o animales. Estos agentes están diseñados para tomar decisiones y adaptarse a su entorno gracias a algoritmos avanzados de IA, como el aprendizaje por refuerzo (reinforcement learning).

Entre las técnicas utilizadas para el diseño de agentes virtuales, los árboles de decisión y el Q-learning [5] destacan por su efectividad. Los árboles de decisión, como algoritmos de aprendizaje supervisado, permiten evaluar condiciones específicas y tomar decisiones estructuradas basadas en criterios jerárquicos. Por otro lado, el Q-learning, una técnica de aprendizaje por refuerzo, ayuda a los agentes a explorar y aprender de su entorno mediante un sistema de recompensas. Al combinar ambas técnicas, los agentes pueden adaptarse a contextos complejos, optimizando su comportamiento a lo largo del tiempo.

En el ámbito de los videojuegos, los agentes virtuales son fundamentales para enriquecer la experiencia del jugador, ya sea como aliados, enemigos o personajes secundarios. Sin embargo, replicar un comportamiento que sea no solo funcional sino también convincente y humano sigue siendo un desafío. Esto se debe a la necesidad de diseñar sistemas capaces de analizar su entorno,

tomar decisiones óptimas y mejorar a partir de la experiencia acumulada.

El presente trabajo aborda este desafío mediante el desarrollo de un agente virtual controlado por IA en un videojuego 2D con temática medieval. El objetivo principal es implementar y evaluar cómo el uso combinado de un árbol de decisión y una Q-table permite al agente aprender y tomar decisiones autónomas basadas en parámetros dinámicos [2]. A lo largo del juego, el agente deberá adaptarse a diferentes escenarios y comportamientos del jugador, como atacar, huir o perseguir, utilizando un sistema de memoria y reseteo que fomenta un aprendizaje continuo y diverso.

Con este proyecto, se busca no solo mejorar el diseño de agentes virtuales en videojuegos, sino también explorar su potencial como herramienta educativa. Esto incluye demostrar cómo estos algoritmos pueden ser empleados para enseñar conceptos básicos de inteligencia artificial y machine learning, integrando la teoría en un entorno interactivo que fomente el aprendizaje práctico.

1.1 Problema

Problemática abordada hacia el aprendizaje reforzado
Los agentes virtuales han ganado popularidad en los últimos años, especialmente en el sector de servicio al cliente, donde tienen la capacidad de mantener conversaciones personalizadas para mejorar la experiencia del usuario. A estos agentes se les programa con tecnologías avanzadas de aprendizaje automático (Machine Learning, ML) para procesar y comprender el lenguaje de los usuarios e identificar soluciones a sus problemas. Es importante no confundir a estos agentes con los chatbots, que son códigos diseñados únicamente para chatear, mientras que los agentes virtuales avanzados actúan como un sustituto de un agente humano.

Los agentes virtuales también están insertos en el mundo de los videojuegos, presentándose como aliados, enemigos y como NPCs en el mundo virtual, dentro de las últimas dos décadas, los desarrollos de videojuegos han evolucionado dentro de un constante patrón, hacer los mundos cada vez más realistas y vivos, en uno de estos estos detalles, la idea de constantemente hacer que los NPCs que tengan cada vez comportamientos y patrones casi idénticos al ser humano es un objetivo que muchas desarrolladoras tienen como meta, como gran ejemplo esta Red Dead Redemption 2, un juego de acción/aventuras al estilo western en el cual todos los NPCs tienen un patrón de comportamiento idénticos a un ser humano, en el cual tienen un ciclo de dia/noche, comidas y actividades, y esto es posible en gran parte debido a que los NPCs tienen la habilidad de recordar patrones de comportamiento y además eventos en los cuales el jugador protagonista haya participado.

La problemática a resolver en esta investigación consiste en evaluar y posteriormente analizar el desempeño de un agente virtual enemigo en un videojuego de 2D , desarrollado y diseñado en Unity, que emplea un árbol de decisión y Q-learning con distintos parámetros. Se buscará determinar si el desempeño de un agente virtual con estos algoritmos es capaz de aprender un patrón de comportamiento similar al de un humano en un escenario en el cual el personaje tendrá que además recordar estado de acciones aprendidas anteriormente y además ser capaz de tomar decisiones adecuadas en el momento oportuno.

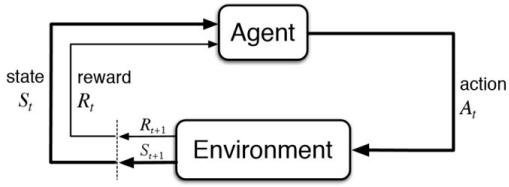


Fig. 1. Q-learning

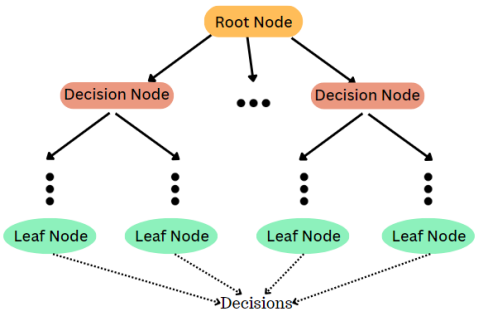


Fig. 2. Decision Tree

1.2 Revisión de material Bibliográfico

Los avances en inteligencia artificial (IA) han permitido el desarrollo de agentes virtuales cada vez más sofisticados para su integración en videojuegos.

Sin embargo, cada implementación revela tanto los logros como los desafíos inherentes a la creación de sistemas autónomos que interactúan eficazmente en entornos dinámicos, al tiempo que intentan simular un comportamiento humano convincente. Los trabajos de Di Wang et al. [4], Valle Guevara y Correa Madrigal [3], y Fernández et al. [1] abordan esta problemática desde distintas perspectivas, ofreciendo una visión complementaria que guía la evolución de este campo.
El trabajo de Di Wang et al. [4] presenta un agente virtual diseñado con redes autoorganizadas FALCÓN, específicamente desarrollado para competir en el torneo Unreal 2004 y participar en la competencia 2K BotPrize. Este evento desafió a los desarrolladores a crear agentes que pudieran ser confundidos con jugadores humanos. Para ello, el agente implementó un sistema reactivo que le permitió adaptarse en tiempo real, seleccionando entre cuatro acciones principales: moverse, recolectar objetos, huir de combates o participar en el fuego cruzado. Las decisiones se tomaban basándose en entradas clave como el nivel de salud, la visibilidad del oponente y la disponibilidad de munición. Aunque el agente logró la mayor puntuación del torneo y convenció a varios jueces de que era humano, su comportamiento no alcanzó la completa naturalidad. Este caso evidenció cómo los agentes pueden desempeñarse eficazmente en un entorno competitivo al adoptar y ajustar reglas en tiempo real, pero también subrayó las limitaciones al replicar el comportamiento humano de forma convincente.

En una línea complementaria, Valle Guevara y Correa Madrigal [3] se enfocaron en el diseño de sistemas de percepción avanzados para agentes autónomos en videojuegos serios. Su propuesta

ofrece un enfoque jerárquico con distintos niveles de complejidad para percibir elementos dinámicos del entorno, así como herramientas para capturar datos topológicos y medioambientales. Lo que destaca de este trabajo es la importancia que otorgan a la cooperación entre agentes y a la coherencia temporal. A través de un algoritmo específico, los agentes pueden actualizar únicamente los datos relevantes, reduciendo así la carga computacional. Este enfoque proporciona una base sólida para diseñar agentes que no solo interactúan con el entorno, sino que lo hacen de forma eficiente y en colaboración con otros, lo cual es esencial en entornos de simulación más complejos.

Por otro lado, Fernández et al. [1] introducen un enfoque innovador al desarrollar AI-live, un videojuego inspirado en The Sims. Su principal objetivo es construir personajes impulsados por IA mediante técnicas de planificación. Para ello, integraron un planificador de tareas y un planificador de rutas, logrando que los agentes tomen decisiones estratégicas mientras se desplazan de forma eficiente en el entorno del juego. Además, diseñaron una arquitectura basada en un servidor central y tres tipos de clientes (basados en reglas, planificación y una interfaz gráfica). Aunque este proyecto se encuentra en etapas iniciales, los autores anticipan ampliaciones significativas, como la inclusión de relaciones sociales entre personajes y la transición a un entorno 3D. Este trabajo ejemplifica cómo la IA puede utilizarse no solo para simular comportamientos, sino también para crear experiencias interactivas que evolucionen con el tiempo.

En conjunto, estos estudios resaltan tanto los avances como las limitaciones actuales en el desarrollo de agentes virtuales. Si bien demuestran que es posible implementar comportamientos complejos en sistemas autónomos, también evidencian que el diseño de agentes capaces de aprender y adaptarse de manera completamente autónoma sigue siendo un desafío. Ninguno de los trabajos revisados aborda la creación de un videojuego donde el agente virtual no solo interactúe, sino que aprenda con el tiempo y, además, funcione como una herramienta educativa. Esto abre una oportunidad para innovar en la creación de un sistema con Q-learning que fomente el cómo funciona un agente virtual con inteligencia artificial y machine learning. El enfoque con agentes capaces de enseñar sus propias funciones no solo llenaría un vacío en la literatura, sino que también ofrecería una solución práctica para enseñar estas tecnologías de manera dinámica y accesible.

1.3 Objetivos

Desarrollar un sistema de agente virtual el cual se implementará en un videojuego enfocado en el aprendizaje por reforzamiento, para lograr este objetivo se trabajará desarrollando una IA avanzada, para esto se ocupará árbol de decisión y Q-learning, las cuales, entrelazadas, permitirán que el agente pueda tomar decisiones por su cuenta, en base a una serie de estado de acciones con recompensas impuestas en el algoritmo, además, el agente deberá ser capaz de aprender la mejor acción en el momento adecuado, esto también será posible gracias a un sistema de memoria y reseteo, en el cual el agente deberá retener la información aprendida sobre una acción, para posteriormente, hacer un reseteo en el cual se le permita a este aprender una nueva. Para el videojuego, será uno con estilo medieval, 2D, en el cual el jugador pueda explorar un mapa y probar las animaciones del personaje, así como observar el comportamiento del enemigo, para posteriormente hacer un análisis sobre cómo el enemigo reacciona a los distintos parámetros y si es capaz de emular un comportamiento realista en base a esos parámetros. El objetivo total de este juego es implementar, observar y analizar cómo se comporta un agente virtual en un videojuego dado los parámetros dados y con sistemas avanzados de IA.

Primer objetivo específico: Diseñar mecánicas alineadas a lo que quiero lograr.

Para esta investigación se ocuparan y emplearán distintas técnicas y mecánicas para que la demostración del juego y del NPC queden mecánica y visualmente coherentes con la temática del juego y con lo que desea implementar. Lo primero es lograr que el personaje tanto controlable por el jugador, como el agente (en este caso el enemigo), tenga un diseño visualmente agradable, para esto se ocupara un sprite con diseño medieval para el personaje jugable, el cual estará ligeramente basado en el personaje principal de el videojuego Dark Souls, luego, el enemigo estará ligeramente basado en un duende pero grande, con espada.

Como mecánica, el personaje protagonista tendrá una espada y un escudo, podrá saltar y rodar y tendrá animaciones definidas para ello, en el caso del enemigo, el agente podrá atacar y recibir daño, en base a una barra de salud en el código, la cual estará establecida en 100, el resto del movimiento del agente recaerá en los algoritmos y en la respectiva recompensa que se le quiere dar.

Segundo objetivo específico: Desarrollar juego.

¿Cómo el agente virtual seguirá la lógica de poder tomar decisiones por el mismo?, para esto se utilizará un algoritmo de aprendizaje con refuerzo llamado Q-learning, que permite al agente tomar sus propias decisiones óptimas a través de la experiencia. El objetivo del Q-learning es ayudar al agente a maximizar una recompensa acumulada en el tiempo. Esta recompensa se asigna en función de las acciones que toma el agente en diferentes situaciones.

Entre los conceptos básicos de Q-learning que se aplicaran en el juego se encuentran los siguientes:

- Estados: Representan diferentes situaciones o condiciones en las que el agente puede encontrarse. Los cuales podrían incluir situaciones como “el jugador está cerca”, “el jugador está herido” o “el agente recibió daño”.

- Acciones: Estas son las diferentes opciones de comportamiento que el agente puede elegir en cada estado. Por ejemplo, optar por atacar, huir, o defenderse.

- Recompensa: Se da una recompensa (o penalización) al agente por realizar una acción en un estado. Si el agente realiza una acción que lo acerca a su objetivo (por ejemplo, inflige daño al jugador), puede recibir una recompensa positiva. Si realiza una acción desfavorable (por ejemplo, recibe daño), puede recibir una penalización.

- Q-table: La Q-table es una tabla en la que el agente almacena sus experiencias. Cada combinación de estado y acción tiene un valor Q, que representa el “valor esperado” de tomar acción en ese estado. A medida que el agente explora y aprende, actualiza estos valores para reflejar las acciones más efectivas en cada estado.

  Al implementar el Q-learning en el juego, la tabla actúa como una guía para que el NPC enemigo aprenda y mejore su comportamiento con el tiempo. Inicialmente los valores Q en la tabla pueden ser aleatorios o neutrales, en este caso se definirán en el código distintos valores en la tabla para después analizar y comparar sus respectivos resultados. A medida que el enemigo interactúa con el jugador y el entorno, el algoritmo ajusta los valores Q para reflejar

las consecuencias de las acciones del NPC, mejorando sus decisiones futuras. Sin embargo, estas decisiones no durarán demasiado debido a un sistema de reseteo en el cual el agente podrá empezar de cero para aprender una nueva acción pero retendrá cierto aprendizaje.

Tercer objetivo específico: Evaluar el juego.

Evaluación del Juego Basada en las 10 Heurísticas de Usabilidad de Jakob Nielsen
Debido a restricciones de tiempo, no se llevaron a cabo pruebas formales en fases alpha ni beta. Sin embargo, la evaluación del juego se planteó siguiendo las 10 heurísticas de usabilidad de Jakob Nielsen, identificando puntos clave a considerar para garantizar que el diseño y la experiencia del usuario cumplan con los estándares de accesibilidad, intuitividad y funcionalidad. Estas consideraciones sirvieron como guía para el desarrollo del videojuego y su agente virtual.

(1) **Visibilidad del estado del sistema**
Se aseguró que el juego proporcionara información constante sobre el estado del jugador y del agente enemigo. Se incluyeron indicadores de heridas en el personaje protagonista debido a que sangra cuando es atacado y notificaciones sobre el estado del enemigo en la consola del unity (Idle, Pursue, Attack, Flee), con el objetivo de mantener al usuario informado de manera clara y precisa.

(2) **Relación entre el sistema y el mundo real**
El diseño del juego utilizó una temática medieval, integrando elementos visuales y terminología fácilmente reconocibles para los jugadores. Las acciones del agente, como "Atacar" o "Huir", y los elementos visuales, como espadas y escudos, se alinearon con este contexto, facilitando la inmersión y comprensión del entorno.

(3) **Control y libertad del usuario**
Se diseñaron mecánicas que permitieran al jugador controlar sus acciones, como atacar, bloquear y esquivar, sin restricciones excesivas. En caso de errores del usuario, el sistema debía permitir retomar el flujo de juego rápidamente, asegurando una experiencia sin interrupciones.

(4) **Consistencia y estándares**
El diseño mantuvo un estilo coherente en todos los elementos visuales y funcionales del juego. Los menús, controles y animaciones se diseñaron siguiendo estándares familiares en juegos de temática medieval, garantizando una experiencia fluida y predecible para el jugador.

(5) **Prevención de errores**
Se implementaron sistemas básicos para prevenir errores comunes, como acciones no válidas o inconsistentes. Por ejemplo, el agente enemigo no realiza ataques cuando su estado es "Huir", lo que refuerza la lógica de su comportamiento basado en la IA.

(6) **Reconocimiento antes que recuerdo**
El juego incluyó indicadores visuales y contextuales que evitaban la necesidad de memorizar información clave. Por ejemplo, las recompensas obtenidas por el agente y los estados actuales eran visibles en pantalla, proporcionando claridad durante la experiencia de juego.

(7) **Flexibilidad y eficiencia de uso**
Aunque no se alcanzó a implementar personalización avanzada, se consideró que el sistema debía ser accesible tanto para jugadores novatos como avanzados. Esto incluyó controles simples y directos que facilitaran la interacción con el juego.

(8) **Estética y diseño minimalista**
El diseño visual del juego mantuvo un enfoque minimalista, evitando sobrecargar al jugador con información innecesaria. Los elementos medievales reforzaron la narrativa sin distraer de los objetivos principales del juego.

(9) **Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores**
Se desarrollaron mensajes básicos para notificar al jugador sobre situaciones específicas, como "El personaje está muerto". Aunque limitados, estos mensajes se diseñaron para ser claros y funcionales, ayudando al jugador a corregir sus acciones.

(10) **Ayuda y documentación**
Se contempló la inclusión de tutoriales y menús de ayuda en fases futuras. Estos recursos explicarían las mecánicas del juego, el comportamiento del agente enemigo y los objetivos principales, con el propósito de facilitar la curva de aprendizaje.

Limitaciones
Debido a la falta de pruebas formales en usuarios reales, la evaluación del sistema se realizó de manera interna, basándose en análisis cualitativos durante el desarrollo. Aunque las mecánicas y funcionalidades fueron validadas a nivel técnico, los aspectos de usabilidad requieren pruebas adicionales para confirmar su efectividad en un entorno práctico.

En el futuro, sería recomendable realizar pruebas alpha y beta para obtener retroalimentación directa de los usuarios, identificar mejoras necesarias y garantizar que el sistema cumpla con las expectativas tanto educativas como de experiencia de usuario.

2 METODOLOGÍA

La metodología seguida en esta investigación combina el diseño, desarrollo y evaluación de un videojuego prototipo que utiliza un agente virtual controlado mediante algoritmos de inteligencia artificial avanzada. El objetivo principal es implementar y analizar el desempeño de un NPC (Non-Playable Character) enemigo en un entorno 2D, integrando árboles de decisión y Q-learning para fomentar comportamientos autónomos, realistas y adaptativos.

2.1 Diseño del Experimento

El enfoque metodológico incluye las siguientes etapas:

(1) **Definición del agente virtual:**
Se diseñó un NPC enemigo con un comportamiento autónomo capaz de interactuar con un entorno medieval 2D desarrollado en Unity. Este agente tiene la capacidad de atacar, huir, perseguir y permanecer inactivo, con decisiones basadas en un árbol de decisión y una Q-table que optimizan sus acciones a través del aprendizaje por refuerzo.

(2) **Implementación de algoritmos:**
El agente emplea un árbol de decisión para evaluar estados específicos del entorno, como la proximidad del jugador, la salud actual del agente y las acciones recientes. Paralelamente, la Q-table gestiona las recompensas acumuladas para mejorar las decisiones del NPC, permitiendo que este aprenda patrones óptimos de comportamiento mediante un sistema de refuerzo positivo y negativo.
(3) **Sistema de reseteo:**
Se integró un mecanismo de reseteo en el que el agente, tras dominar una acción específica, "olvida" su aprendizaje previo y reinicia el proceso de exploración. Esto fomenta el descubrimiento de nuevas estrategias y reduce la sobreespecialización en un solo patrón.

2.2 Desarrollo del Videojuego

El videojuego se diseñó en Unity utilizando C++ para la programación de la lógica y las animaciones. El entorno del juego incluye un escenario medieval donde el jugador controla un personaje principal, mientras que el enemigo es controlado por la IA. Las principales características del diseño incluyen:

- **Componentes visuales y mecánicos:**
El personaje principal cuenta con animaciones para moverse, atacar, defenderse y esquivar, mientras que el enemigo tiene la capacidad de atacar y moverse autónomamente según las decisiones de la IA. Ambos personajes cuentan con una barra de salud que define el impacto de las acciones.
- **Parámetros del agente:**
El agente enemigo utiliza cuatro estados principales:
Idle: Mantenerse inactivo.
Pursue: Perseguir al jugador.
Attack: Atacar al jugador.
Flee: Huir cuando se encuentra en desventaja.

- **Asignación de recompensas:**
Cada acción está asociada a una recompensa o penalización dependiendo de su efectividad en el contexto del juego. Por ejemplo, atacar exitosamente al jugador genera una recompensa positiva, mientras que recibir daño penaliza al agente.

2.3 Evaluación del Sistema

Para evaluar el desempeño del agente, se realizaron dos evaluaciones principales:

(1) **Pruebas antes y después del reseteo:**
Se analizaron los valores de la Q-table antes y después de que el agente reiniciará su aprendizaje. Esto permitió observar cómo el agente prioriza acciones iniciales y cómo explora nuevas estrategias tras el reseteo.

(2) **Simulaciones con parámetros variables:**
Se modificaron las recompensas asociadas a cada estado para observar cómo influyen en las decisiones del agente. Esto permitió ajustar las configuraciones para lograr un equilibrio entre exploración y explotación.

2.4 Análisis de Resultados

Los datos obtenidos se analizaron a través de mapas de calor que mostraban las decisiones del agente en función de los valores de la Q-table. Esto permitió evaluar:

- La efectividad del agente para adaptarse a diferentes situaciones.
- La capacidad del sistema de reseteo para fomentar nuevos aprendizajes.
- El balance entre acciones agresivas (ataque) y defensivas (huida).

Esta metodología asegura un enfoque estructurado para diseñar y evaluar el rendimiento del agente virtual, contribuyendo al desarrollo de sistemas de IA más avanzados y educativos.

3 DISEÑO DEL VIDEOJUEGO

Configuracion experimental

El videojuego esta diseñado en unity, una plataforma para desarrolladores. Ademas, la especificaciones del equipo en la cual se esta haciendo el proyecto son las siguientes:

- Procesador 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.80 GHz
- Ram de 16,0 GB (15,7 GB utilizable)
- Windows 11 Pro con sistema operativo de 64 bits, procesador x64

Programacion y diseño

El videojuego en sus etapas de prototipado se ideó que fuera 2D, esto principalmente debido a que es un proyecto de corto alcance y un juego 3D pero más largo requeriría un alcance mayor del proyecto. Sobre el videojuego en sí, se han utilizado una variedad de assets de carácter medieval, el cual es la principal fuente de inspiración visual del videojuego tal como se puede ver en la figura 1. Esta idea de que sea medieval nace por una cierta atracción de los jugadores a videojuegos con esta temática, tales como The Elder Scrolls, Dark Souls, entre otros.

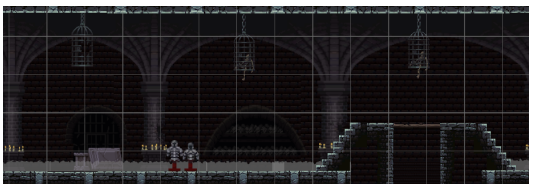
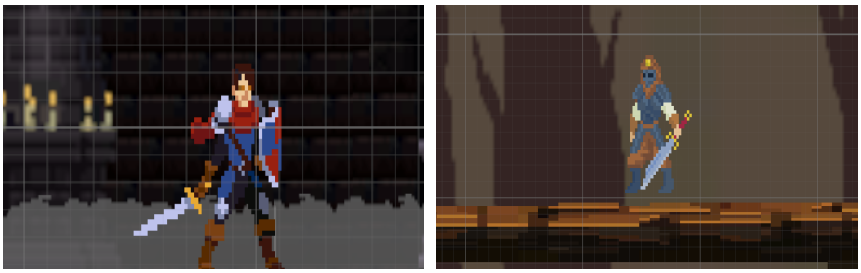


Fig. 3. Assets utilizados para el videojuego

La inspiración para crear al personaje viene también de lo medieval, como se puede ver en la figura 2. El personaje el cual sus animaciones pueden hacer que este salte, defienda con su escudo, permita dar una vuelta y tenga tres ataques con la espada. Cada una de las animaciones son coherentes con lo que se quiere demostrar, en el caso de el enemigo, este tambien se podra mover y atacar, pero en su totalidad en solo una base preestablecida, debido a que el movimiento se controla por algoritmo, poniendo énfasis en atacar al jugador principal y perseguir u huir al jugador.

El videojuego está Programado en su totalidad en C + +, el cual es el lenguaje de Unity. Para la implementación de la Q-table o también llamado aprendizaje por reforzamiento se hará una integración del código del enemigo con sus respectivas animaciones más el aprendizaje, el enemigo irá reconociendo el mapa y tomará decisiones en base a la mayor recompensa posible, lo



(a) Personaje selecto para el jugador (b) Personaje selecto para el Agente Virtual

Fig. 4. Personajes del videojuego

más importante es que tenga una tasa de aprendimiento rápido, ya que eso hará posible el que el videojuego sea dinámico y a su vez interactivo para el aprendizaje.

4 DESARROLLO

La implementación de la Q-table el NPC enemigo trajo consigo sus ventajas y desventajas, primero, se le introdujo un sistema de recompensas en el cual se definieron antes los estados (states) a utilizar para que el personaje aprenda, estos estados fueron: atacar, huir, perseguir, y el estado Idle del personaje el cual es por default inactivo. Terminada la integración de la de la Q-table con el sistema de recompensas, se le integró al código un sistema de reseteo, que fomenta la exploración continua y evita la sobreespecialización.

4.1 Impacto del Sistema de Reseteo

El sistema de reseteo tiene el propósito de:

- Reiniciar el conocimiento acumulado del agente.
- Promover la exploración y descubrimiento de nuevas estrategias.
- Minimizar la sobreespecialización en patrones de comportamiento específicos.

4.2 Simulaciones

Para los análisis Se realizaron tres simulaciones para evaluar el desempeño del agente antes y después del reseteo, en los cuales en cada uno tenía unos valores distintos asociados a las recompensas

Para la primera de las simulaciones se utilizaron los siguientes parámetros:

```
case State.Attack:
    return 15.0f;  // Aumento de recompensa por atacar
case State.Flee:
    return -5.0f;   // Penalización moderada por huir
case State.Pursue:
    return 10.0f;   // Aumento de recompensa por perseguir
default:
    return 1.0f;    // Baja recompensa por estar inactivo
```

Fig. 5. Estados y acciónes del videojuego

Dando paso a los siguientes resultados en la matriz de calor:

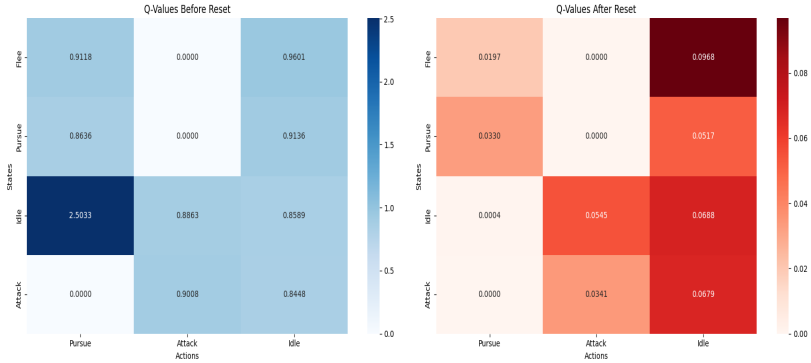


Fig. 6. Antes y despues del reinicio

4.3 Análisis y Resultados

- Antes del reinicio: el agente priorizaba estados defensivos como Idle y Flee, con Pursue siendo el siguiente estado preferido. Esto refleja un aprendizaje previo que lo llevó a evitar riesgos y conservar recursos.

- Después del reinicio, el reinicio del Q-learning restableció las preferencias del agente, distribuyendo los valores de manera uniforme y fomentando una exploración más activa de todas las acciones posibles.

Para la segunda de las simulaciones se utilizaron los siguientes parámetros:

```
case State.Attack:
    return 30.0f;  // Aumento de recompensa por atacar
case State.Flee:
    return -10.0f;   // Penalización moderada por huir
case State.Pursue:
    return 20.0f;   // Aumento de recompensa por perseguir
default:
    return 1.0f;    // Baja recompensa por estar inactivo
```

Fig. 7. Estados y acciónes del videojuego

Los parámetros de recompensa definidos en esta ocasión son valores altos tanto positivamente como negativamente para que el agente tenga un incentivo mayor al realizar las acciones.

Dando paso a los siguientes resultados en esta matriz de calor:

Fig. 8. Antes y despues del reinicio

4.4 Análisis y Resultados

- Antes del reinicio: Los valores elevados de las acciones de “Attack” y “Pursue” indican que el agente tiende a preferir estas acciones en los estados correspondientes , por lo que el comportamiento del bandit antes del reset estaba probablemente orientado hacia un enfoque más agresivo o de persecución activa.

- Después del reinicio: Dado que los valores son significativamente menores, el agente estará menos inclinado a “decidir” acciones agresivas o de persecución de inmediato. Esto implica que el Bandit (agente) ahora probablemente adopte un comportamiento de exploración mayor, probando más combinaciones de estado-acción para identificar nuevamente los mejores caminos.

Para la tercera de las simulaciones se utilizaron los siguientes parámetros:

```
case State.Attack:
    return 10.0f;  // Aumento de recompensa por atacar
case State.Flee:
    return 30.0f;   // Penalización moderada por huir
case State.Pursue:
    return 70.0f;   // Aumento de recompensa por perseguir
default:
    return 1.0f;    // Baja recompensa por estar inactivo
```

Fig. 9. Estados y acciónes del videojuego

En este caso, los valores definidos en la recompensa son valores altos en los estados de Flee y Pursue.

Dando paso a los siguientes resultados en la matriz de calor:

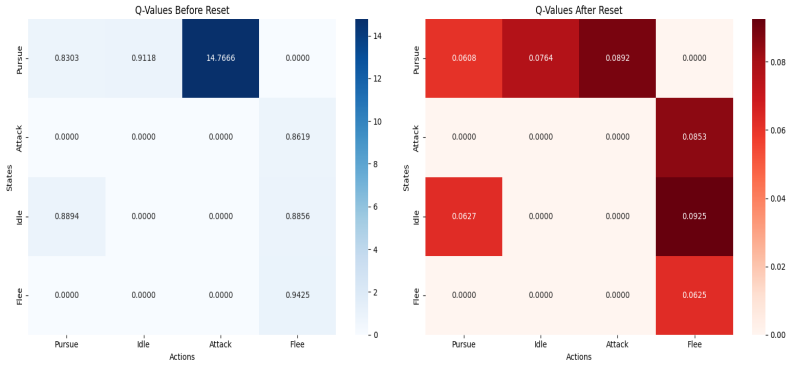


Fig. 10. Antes y despues del reinicio

4.5 Análisis y Resultados

- Antes del reinicio: El Bandit había desarrollado un comportamiento altamente optimizado, priorizando “Pursue” combinado con “Attack” como la estrategia más rentable debido al alto valor de estos. Las acciones pasivas como “Idle” y “Flee” son menos favorables a menos que el contexto específico las hiciera útiles.

- Después del reinicio: La matriz refleja un estado inicial sin aprendizaje significativo. Esto confirma que el reinicio eliminó todo conocimiento previo, afectando significativamente los estados de “Pursue” y “Attack”, volviendo a tener que reaprender nuevos patrones y comportamientos adaptados a su entorno.

5 CONCLUSIONES

Los resultados obtenidos a partir de las simulaciones permitieron evaluar la efectividad del agente virtual implementado y extraer conclusiones relevantes sobre su desempeño. A continuación, se presentan los hallazgos clave:

(1) **Patrones de Comportamiento del Agente:**

- El agente muestra un comportamiento racional basado en las recompensas asignadas, priorizando acciones como Flee en situaciones defensivas y Pursue cuando inicia la interacción. Esto refleja una capacidad de adaptación eficiente a las condiciones del entorno.

- En las configuraciones con altas recompensas para acciones agresivas (Attack y Pursue), el agente tiende a desarrollar estrategias más ofensivas, mientras que recompensas balanceadas favorecen un enfoque defensivo.

(2) **Impacto del Sistema de Reseteo:**

- El reseteo del sistema fomenta la exploración de nuevas estrategias y evita la sobreespecialización en un solo patrón de comportamiento. Esto resulta en un equilibrio entre acciones ofensivas y defensivas, aunque con una pérdida temporal de desempeño tras el reinicio.

- Los valores significativamente menores después del reseteo demuestran que el agente retorna a un estado de aprendizaje inicial, favoreciendo el descubrimiento de combinaciones de estado-acción previamente ignoradas.

(3) **Balance entre Exploración y Explotación:**

- El agente logra un equilibrio adecuado entre explorar nuevos estados y explotar estrategias conocidas cuando las recompensas están configuradas de manera equilibrada. Esto es esencial para mantener la dinámica y la diversidad en el comportamiento del agente.

- En las simulaciones donde se incentivaron valores altos para Flee, el agente mostró una tendencia a evitar riesgos, lo cual puede ser útil en escenarios de supervivencia, pero podría limitar la agresividad necesaria en otros contextos.

(4) **Áreas de Mejora Identificadas:**

- **Ajuste de Recompensas:** Es necesario refinar los valores asignados a las recompensas para equilibrar mejor la agresividad y la defensa del agente, especialmente en escenarios que requieren mayor proactividad.

- **Optimización del Aprendizaje Tras el Reseteo:** Implementar mecanismos que permitan retener parcialmente el conocimiento adquirido podría reducir la pérdida de desempeño después del reseteo, acelerando el proceso de reaprendizaje.

6 TRABAJOS FUTUROS

En el futuro, se podrían optimizar las capacidades del agente mediante la integración de mecanismos de retención parcial del conocimiento previo al reseteo, lo que reduciría la pérdida de desempeño y aceleraría el reaprendizaje. Asimismo, explorar modelos avanzados como el Deep Q-Learning permitiría manejar entornos más complejos y tomar decisiones más precisas. Realizar pruebas con usuarios en etapas alpha y beta facilitaría evaluar la jugabilidad y el impacto del comportamiento del agente en escenarios dinámicos. La adaptación de recompensas en tiempo real y la incorporación de interacciones sociales con otros NPCs y jugadores abrirían nuevas posibilidades en términos de diseño y experiencia. Finalmente, sería valioso analizar el desempeño del agente en sesiones de juego prolongadas, evaluando su capacidad de memoria y ajuste continuo a contextos cambiantes.

7 AGRADECIMIENTOS ESPECIALES

A mi profesor guía, mi familia y mis amigos.....muchas gracias por todo.

REFERENCES

[1] Susana Fernández, Roberto Adarve, Miguel Pérez, Martın Rybarczyk, and Daniel Borrajo. 2006. Planning for an AI based virtual agents game. In *Proceedings of the Workshop AI Planning for Computer Games and Synthetic Characters in the ICAPS*. Citeseer.

[2] Kao-Shing Hwang, Yu-Jen Chen, Wei-Cheng Jiang, and Tsung-Wen Yang. 2012. Induced states in a decision tree constructed by Q-learning. *Information Sciences* 213 (2012), 39–49.

[3] Yenifer del Valle Guevara and Omar Correa Madrigal. 2013. *Sistema de percepcion generico para agentes autonomos en videojuegos serios.* Master's thesis. Universidad de las ciencias informaticas.

[4] DI Wang, Budhitama Subagdja, Ah-Hwee Tan, and Gee-Wah Ng. 2009. Creating human-like autonomous players in real-time first person shooter computer games. In *Twenty-First IAAI Conference*.

[5] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8 (1992), 279–292.