VOLUMEN 28
Verano de la Ciencia XXIX
ISSN 2395-9797
www.jóvenesenlaciencia.ugto.mx

Inteligencia Artificial como soporte para la creación de contenido en el arte y los videojuegos

Artificial Intelligence as a support for content creation in art and videogames

Martín St. John Clarke Bideau[1**], Juan Aguilera Huerta[2*], Edmundo Rafael Medel Sánchez[3*], Luis Armando Sánchez Hernández[4****], Marco Antonio Escobar Carmona[5*], Rocio Alfonsina Lizárraga Morales[6***], Uriel Haile Hernández Belmonte[7***]

[*]Lic. En Ingeniería En Sistemas Computacionales, División de Ingenierías Campus Irapuato-Salamanca, Universidad de Guanajuato,
[**]Lic. En Artes Digitales, División de Ingenierías Campus Irapuato-Salamanca, Universidad de Guanajuato.
[***]Departamento de Arte y Empresa, División de Ingenierías Campus Irapuato-Salamanca, Universidad de Guanajuato.
[****] Ingeniería Informática, Instituto Tecnológico Superior de la Región Sierra.

m.stjohnclarkebideau@ugto.mx[1], j.aguilerahuerta@ugto.mx [2], er.medelsanchez@ugto.mx[3], veranocientifico79@gmail.com[4], ma.escobarcarmona@ugto.mx [5], ra.lizarragamorales@ugto.mx [6], uh.hernandez@ugto.mx [7]

Resumen

En los últimos años, la inteligencia artificial ha irrumpido en distintos campos del conocimiento que se pensaban exclusivos de los seres humanos. Ya se comercializan carros que se manejan de forma autónoma, sistemas de recomendación basados en nuestros hábitos de compras y recientemente, sistemas que pueden responder preguntas complejas y generar todo tipo de contenido multimedia. Estos avances tecnológicos han comenzado a cambiar de forma drástica la forma en la cual se produce el contenido digital. Actualmente, una computadora puede servir como lienzo, como pincel, como lápiz, como instrumento musical, como asistente creativo e inclusive, también puede aprender a realizar nuevas acciones. La inteligencia artificial ha implicado cambios disruptivos en diversos dominios creativos, detonando amplias discusiones sobre el presente y futuro en la creación de contenido digital. Esto nos lleva a explorar una conexión más profunda entre la Inteligencia Artificial y la creatividad. El objetivo de este proyecto es la exploración y aplicación de diferentes algoritmos de Inteligencia artificial enfocados al desarrollo de videojuegos y generación de contenido en diferentes áreas de las artes digitales. Específicamente, se propone una exploración de aprendizaje por refuerzo para agentes en conjunto con un videojuego llamado *Sporing* desarrollado en la plataforma *Unity*. En *Sporing*, existen personajes de soporte que son agentes inteligentes capaces de navegar de manera autónoma a través de un entorno. Asimismo, la ambientación sonora se implementó utilizando un algoritmo de composición generativa. Los resultados nos indican que el uso de estas técnicas en áreas como el arte y los videojuegos representa un área de oportunidad para la exploración y desarrollo de nuevas experiencias.

Palabras clave: inteligencia artificial, aprendizaje por refuerzo, videojuegos, agentes, generación de contenido, algoritmo, PPO, NavMesh, NPC.

Introducción

En la última década, gran parte de los avances en inteligencia artificial (IA) han ido dirigidos al medio artístico y al audiovisual, impactando en diferentes áreas como la creación de imágenes, música y por supuesto, videojuegos. Cada uno de estos enfoques necesita de distintas técnicas de IA para resolver los problemas específicos que conllevan. Por un lado, para la generación de contenido artístico se requiere de grandes modelos de aprendizaje automático que identifican los estilos o patrones presentes en una inmensa cantidad de muestras. Esto mismo ocasiona que los modelos imiten deliberadamente las características de material ya existente, lo que ha sido objeto de controversia en más de una ocasión. Por el contrario, en el caso de la música, se ha demostrado el potencial que se tiene para crear piezas musicales originales mediante la generación de música simbólica. Esto se refiere a un proceso donde modelos de aprendizaje son capaces de reconocer y representar propiedades de la música; notas, ritmos, acordes, dinámicas, etc., a través de secuencias de símbolos. La música simbólica se caracteriza por considerar la estructura e interpretación de las piezas, lo que permite producir música coherente que inclusive puede tener diferentes niveles de repetición y modulación (Zhao, y otros, 2022).

Otra de las áreas donde la inteligencia artificial ha sido mayormente implementada es en la creación de sistemas para videojuegos. En un principio, estos sistemas se basaron en algoritmos simples que mejoraban ligeramente la lógica de enemigos u obstáculos básicos y, por ende, no requerían de comportamientos más avanzados. No fue hasta la década de 1980 que empezaron a introducirse sistemas de decisiones que aparentaban una mayor inteligencia, siendo el ejemplo más popular Pac-Man. En este juego, cada enemigo o "fantasma" tiene un patrón de movimiento diferente para atrapar al jugador, pero a la vez puede entrar en otros estados derivados de acciones especiales que ejecute el jugador, formando una máquina de estados. Los videojuegos siguieron integrando IA partiendo de este mismo fundamento, pero combinando otros sistemas más sofisticados como búsqueda de caminos o simulación sensorial. Algunos ejemplos que basaron su diseño en esta última técnica incluyen juegos como: *Goldeneye 007* (Rare Ltd., 1997), *Metal Gear Solid* (Konami Corporation, 1998) y *Thief: The Dark Project* (Looking Glass Studios, Inc., 1998) (Millington & Funge, 2009). Conforme las limitaciones tecnológicas eran menores, los desarrolladores de videojuegos han podido darle varios enfoques a la inteligencia artificial más allá de los enemigos "autónomos"; comportamiento de personajes no jugables (NPCs), generación procedural de niveles, escalas de dificultad dinámicas, y narrativa son algunos de ellos. No obstante, a pesar de la aparición de nuevos algoritmos y modelos inteligentes; varias de estas características son implementadas con técnicas rudimentarias que se alejan del concepto de "inteligencia artificial" que se tiene actualmente.

Un pensamiento recurrente entre desarrolladores y consumidores es que el avance de los videojuegos se ve determinado por su mejora en el apartado gráfico. Aunque esta afirmación no es del todo falsa, los videojuegos han llegado a un punto donde las diferencias en gráficos entre diferentes productos son mínimas. Alrededor de mundo, equipos de desarrollo han dirigido sus esfuerzos en buscar aquello que haga destacar a sus juegos entre los demás. Es aquí donde los avances en inteligencia artificial pueden ser la solución (Ocio Barriales, 2014), siendo una herramienta que daría lugar a experiencias más complejas. Si bien hay géneros de videojuegos que no necesitarían de un modelo o método de aprendizaje avanzado para manejar sus mecánicas simples; ya que sería un desperdicio de recursos e infraestructura, existen otros que representan un área de oportunidad para mejorar aspectos cruciales de su jugabilidad.

Los juegos modernos de exploración, estrategia, simulación e incluso *shooters*, han empleado técnicas de inteligencia artificial que provienen de otras áreas de investigación, como la robótica y la navegación. Una de estas técnicas que han destacado por su fácil implementación en agentes móviles es el aprendizaje por refuerzo; el cual es un tipo de aprendizaje automático donde, mediante un proceso de prueba y error, un agente adquiere conocimiento de un ambiente dinámico para alcanzar una meta de la mejor manera (Kanade, 2022). De forma general, un algoritmo de aprendizaje por refuerzo consta de tres componentes: un método de exploración simple para intentar acciones variadas, una función de refuerzo o recompensa que determinará que tan buena o mala es cada acción realizada, y una regla o política que une a estas dos para que el agente pueda "aprender" (Millington & Funge, 2009). La base para este tipo de aprendizaje es el

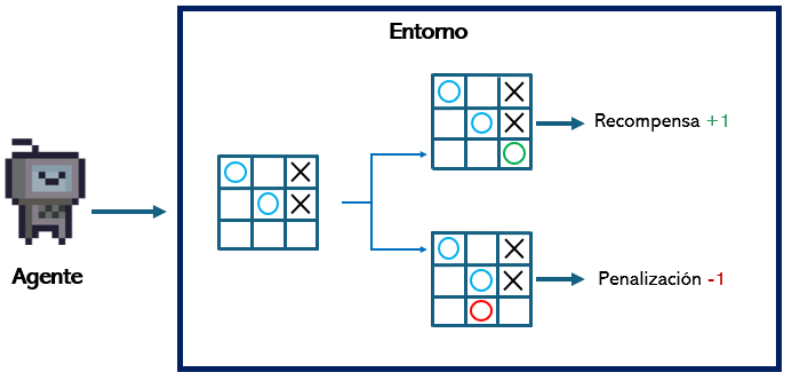


Figura 1. Explicación básica del funcionamiento del aprendizaje por refuerzo.

sistema de recompensas numéricas, siendo la manera en la que se ajusta el comportamiento de un agente para optimizar su función objetivo. El entorno envía estas recompensas con cada interacción positiva del agente, por lo que buscará maximizarlas, caso contrario si se realiza una acción que lo aleja de cumplir con el objetivo, donde se responderá con una penalización.

En este documento, se presenta el proceso que se siguió para dar lugar a la creación de una experiencia interactiva implementando algunos elementos que funcionan con inteligencia artificial, los cuales tienen dos enfoques distintos:

1. Implementación de aprendizaje por refuerzo para el movimiento de asistentes autónomos.
2. Creación de música generada dinámicamente en base a la exploración del entorno.

El contenido de este artículo será mostrado de la siguiente forma: en la sección 2 se abarcará cada aspecto importante de la demostración creada por separado, tales como la concepción de la idea y los pasos que se siguieron para implementar inteligencia artificial. En la sección 3 se exhibirán los resultados obtenidos de cada integración, y finalmente en la sección 4, se expondrán las conclusiones a las que se llegó con en este proyecto.

Metodología

Durante la fase de planificación, fueron consideradas diversas perspectivas en las que la inteligencia artificial pudiera enriquecer la experiencia jugable; la propuesta más viable fue añadir un sistema de juego que involucrase personajes que no se pudiesen controlar, por lo menos de manera directa. Esto con el objetivo de que estos personajes, o entidades, tuvieran un margen para demostrar inteligencia, pues tendrían la capacidad de realizar tareas con poca o nula intervención del jugador. A partir de esta idea, se analizaron videojuegos existentes con una propuesta semejante. Entre los cuales destacó la saga *Pikmin*. Cuya mecánica principal gira entorno a criaturas conocidas como *pikmin*, que son capaces de asistir al jugador en una variedad de situaciones. Estas criaturas tienen la peculiaridad de clasificarse en tipos dependiendo de sus atributos y debilidades, diferenciándose con un color distinto. Por lo que habrá tareas específicas que sólo algunos *pikmin* podrán hacer. Esta mecánica expresa a la perfección el concepto de asistentes inteligentes y resulta un terreno ideal de exploración a fin de involucrar un sistema de inteligencia artificial más moderno.

Figura 2. En Pikmin, la jugabilidad recae en diferentes tipos de asistentes.

Un componente importante para acompañar a la jugabilidad en casi todo momento es el apartado musical y sonoro. Normalmente, los videojuegos no poseen una línea narrativa con un tiempo fijo, lo que significa que no puedes decidir cuánto tiempo el jugador va a estar en cada situación, pues en muchos casos solamente él tiene el control. A la larga, esto conlleva a la repetición de los mismos efectos de sonido y la misma música, provocando que el interés en este apartado disminuya. Si bien no toda repetición se debe considerar negativa, también es cierto que los cambios en la música o el sonido que pudieran las interacciones que puede tener el jugador, pueden mejorar su experiencia general. Además de reducir la repetición, sería posible transmitir información contextual en tiempo real que estimule su atención mediante el sonido (Lopez Duarte, 2020).

Con todo lo discutido anteriormente, se decidió que esta demostración consistirá en una experiencia de exploración, donde el jugador dependa del soporte de múltiples tipos de asistentes inteligentes, a los que se les llamará *fungis*, que puedan responder a órdenes generales de parte del jugador para interactuar con objetos del ambiente. Asimismo, la música y efectos de sonido se verán afectados por las zonas y elementos cercanos al jugador.

Aprendizaje y sistema de navegación de los asistentes

Para el desarrollo de la IA de los agentes, estos funcionarán con el método de aprendizaje por refuerzo, que puede ofrecer tomas de decisiones más “inteligentes” al enfrentarse con varias posibilidades en un entorno. Existen diferentes tipos algoritmos que emplean métodos variados para garantizar el aprendizaje del agente, pero en este caso será usado el algoritmo PPO (*Proximal Policy Optimization*). Este algoritmo tiene como prioridad la optimización directa de las

políticas que determinan las acciones, en lugar de basarse en los valores obtenidos tras ejecutar alguna acción. Para ello, optimiza una función objetivo para ajustar los parámetros de la política que sigue el agente y así maximizar la recompensa.

La función objetivo de PPO involucra dos componentes principales:

1. **Función de política**. Define la estrategia que impulsa el comportamiento del agente dado el estado en el que se encuentre. Se representa como una función que arroja la probabilidad de tomar una acción a cuando se encuentre en un estado *s:* $\pi_\theta(a|s)$, donde π_θ es la política parametrizada por θ, que es el peso del modelo.
2. **Función de valor**. Estima la recompensa acumulada esperada de tomar una acción en un estado particular, considerando la política actual. Se representa como $V(s)$ (Optimización de políticas próximas (PPO) en el aprendizaje por refuerzo, s.f.).

PPO restringe cambios drásticos en su política siguiendo una función de pérdida conocida como *clipping*. Esta función hace que el agente explore su entorno con acciones acorde a una probabilidad sin desviarse de su política actual. La probabilidad de elegir una acción dependerá tanto de las condiciones iniciales como del proceso de entrenamiento. Conforme avance el entrenamiento, la política encamina a cometer acciones menos aleatorias, dándole prioridad a las políticas ya conocidas que generen mayores recompensas. Es aquí donde el agente adquiere una política óptima (OpenAI, 2018).

En un videojuego 3D, otro componente importante para planificar las rutas de los agentes a través de cualquier ambiente es sistema de navegación *NavMesh*. El *NavMesh* hace referencia a una estructura de datos que utiliza un conjunto de regiones bidimensionales (2D) para crear una malla simplificada que representa el espacio transitable del entorno (Van Toll, Cook, & Geraerts, 2012). Esta malla se compone de polígonos (generalmente triángulos) que cubren la superficie a transitar del entorno, además de incluir zonas con obstáculos y áreas no navegables. Estos últimos se excluyen o se marcan como intransitables en el *NavMesh*, ayudando a los agentes a evitar colisiones no deseadas durante la navegación. En conjunto con algún algoritmo, en este caso el algoritmo PPO, la malla del NavMesh permitió determinar los caminos óptimos para que los agentes o "*fungis*" pudieran moverse entre dos puntos en caso de que su tarea lo requiera.

Implementación con Unity

Gracias a la capacidad de los motores de videojuegos de renderizar gráficos y simular físicas como *Unity*, *Unreal Engine* o *Godot* (por mencionar algunos), se tiene la posibilidad de simular ambientes a gran detalle y con total libertad de modificar y crear un entorno virtual con todo lo necesario. En este caso, entrenar agentes para determinadas tareas a través del aprendizaje por refuerzo, todo esto se realiza mediante el Unity ML Agents Toolkit, "un proyecto de código abierto que permite a investigadores y desarrolladores crear ambientes simulados con el Editor de Unity e interactuar con ellos a través de una API de Python. Este conjunto de herramientas proporciona el ML-Agents SDK que contiene toda la funcionalidad necesaria para definir ambientes con el Editor de Unity junto a los códigos base de C# para construir una ruta de aprendizaje" (Juliani, y otros, 2020).

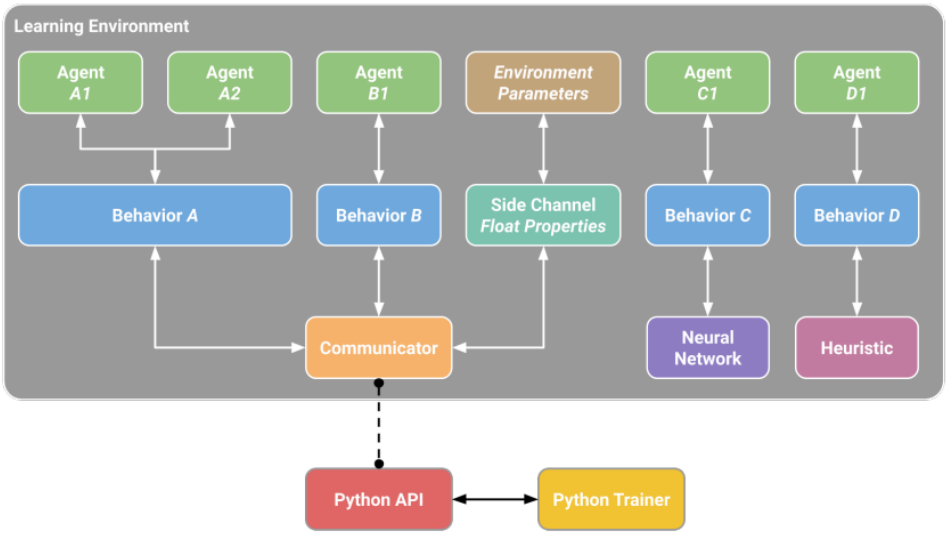


Figura 3. Diagrama del funcionamiento del entorno de simulación que provee el Unity ML-Agents Toolkit.

Se aprovechó esta herramienta para poner en práctica los conocimientos teóricos del *machine learning* a través del aprendizaje por refuerzo. Con esto se creó un agente con forma de cubo capaz de mover una esfera hacia una dirección específica a través de un sistema de recompensas y sesiones de entrenamiento incremental.

Se ha dividido en partes el desarrollo de este proceso con el fin de llevar un control detallado de cada paso. A continuación, se explican en detalle cada una de ellas:

I. **Entrenamiento básico.** Se partió de lograr que el agente pueda ser capaz de empujar una pelota hacia la pared que tiene frente a él, a partir de esta base, se comenzó a desarrollar el proceso de aprendizaje desde lo más básico a lo más complejo.

   a) Ubicaciones estáticas. Se establece que el agente y la pelota siempre estarán separados por una misma distancia. De igual forma, la pared a donde el agente debe llevar la pelota siempre se encuentra frente a ambos.
   b) Ubicación aleatoria hacia las cuatro paredes. Con los conocimientos adquiridos en las sesiones de entrenamiento de la etapa A, se agregó una dificultad extra al cambiar la posición estática del agente por un arreglo que contiene 4 posiciones diferentes apuntan hacia cada una de las paredes. De este arreglo es posible seleccionar de forma pseudoaleatoria alguna de las cuatro ubicaciones desde las que se pueda completar la misma acción, pero en diferentes puntos del plano.
   c) Ubicación aleatoria del agente en un rango. Por último, dentro de este ciclo de entrenamientos se le da más libertad a la posición del agente, aunque sigue bajo la misma condición de la pared. La ubicación del agente estará oscilando entre un determinado rango de valores para que aprenda a apuntar en diferentes direcciones y no en una línea recta.

II. **Entrenamiento medio**. Una vez cubierto el movimiento recto y desde diferentes direcciones para empujar una pelota hacia una misma pared, el siguiente paso fue la modificación del código para tratar de buscar que el agente se enfrente a escenarios donde deba enviar la pelota hacia una pared que esté detrás de él. Para desarrollar este ambiente de aprendizaje, se siguieron los mismos principios (a,b,c) del punto anterior, partiendo de posiciones y situaciones estáticas a las que cada vez se le van incluyendo más condiciones pseudoaleatorias. El aprendizaje deseado para el agente se enfoca en que este pueda desarrollar estos conocimientos para empujar una bola hacia una pared que está delante de él desde cualquier dirección, pero que también pueda hacerlo con paredes que tenga en el sentido contrario.

III. **Entrenamiento avanzado**. El último paso desarrollado en estos ciclos de entrenamientos, consiste en una síntesis final de todos los escenarios, es decir, agregar condiciones lo más variadas posibles para someter al agente a escenarios muy distintos entre sí; que pueda utilizar todas las habilidades básicas que adquirió con los

procesos previos de entrenamiento y así lograr llevar la pelota hacia la pared correcta bajo cualquier escenario posible, sin importar si deba empujarla hacia adelante, cambiar la dirección a la que apunta la esfera para llevarla a una pared desde otro ángulo, cualquier caso. Para esto agregamos un margen dentro de todo el plano para los puntos de aparición de la pelota y la esfera, sin importar dónde se encuentren, el agente deberá ser capaz de encontrar un camino que le permita completar su objetivo.

Diseño del ambiente de aprendizaje

La parte más importante durante el entrenamiento de un agente es su ambiente de aprendizaje, este será el encargado de proporcionarle al agente todas las condiciones necesarias sobre las que él podrá recolectar información, tomar decisiones y generar acciones que serán evaluadas en el **sistema de recompensas** (del cual se hablará a detalle más adelante). El ambiente simulado de aprendizaje consta de un cubo con cuatro paredes que delimitan su área de acción, mostrado en la Figura 4. Esto no significa que el cubo aprenderá a moverse únicamente por espacios como el propuesto, pero sí con las condiciones más simples para empezar a formar una estructura básica de las habilidades que se busca adquiera el agente.

Componentes del agente

El agente consiste en un *GameObject* con forma cúbica que contiene elementos básicos del mismo, tales como un *RigidBody* que permitirá al agente interactuar con las físicas de un espacio 3D para su movimiento. Así mismo, es importante destacar que fueron incluidos dos sensores de observación denominados *Raycast*, que permitirán recopilar información de las paredes que nuestro agente tiene en un campo de visión de 180° y determinar cuándo tiene la pelota frente a él.

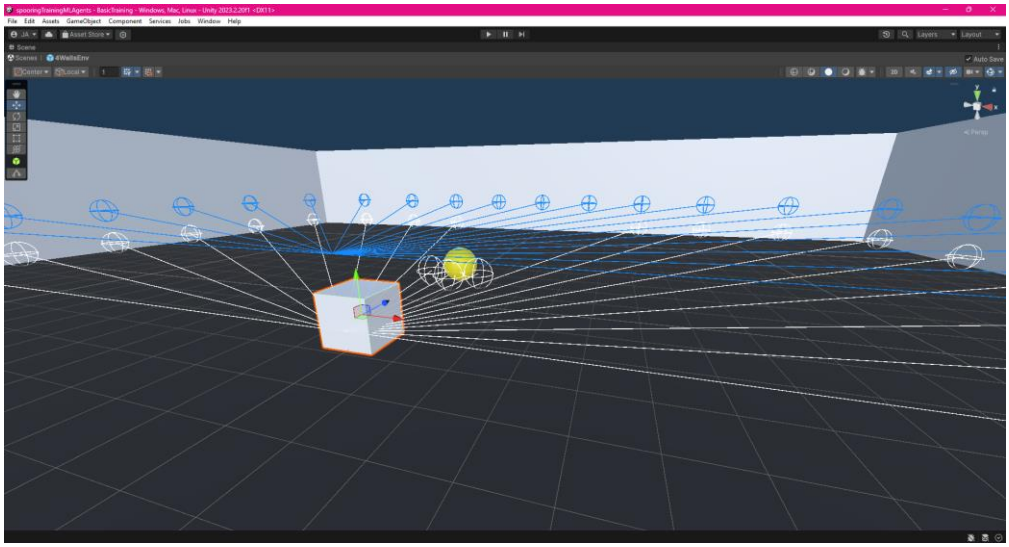


Figura 4. Vista del agente con sus sensores en el escenario.

Las tareas del agente serán segmentadas para incrementar la dificultad a medida que una tarea ayude a que exista un aprendizaje. ML-Agents toolkit proporciona una configuración que permite empezar a entrenar una red neuronal nueva a partir del conocimiento adquirido por alguna previa, así que esta experiencia será transferida a través de todas las sesiones de entrenamiento.

Datos de observación

Para que una inteligencia artificial pueda aprender a desarrollar una tarea, el paso más importante son los parámetros, o en este caso, las observaciones del entorno enviadas a la red neuronal del agente. Estas observaciones sirven para formar la. En el contexto de un videojuego, existen múltiples factores que determinarían estos estados: posición, proximidad a

un elemento, barra de salud, nivel, entre otros, y que darán una ventaja en el aprendizaje del agente (Millington & Funge, 2009).

Es importante escoger las observaciones que en verdad ayudarían a la red neuronal a tener una noción más clara del espacio y sus componentes, descartando aquellas que no aporten información relevante. Para este caso en concreto, los valores a tomar en cuenta son los siguientes:

- Posición del agente
- Posición de la pared a la que se debe llevar la esfera
- Velocidad del agente
- Velocidad de la esfera
- Dirección del agente con respecto a la esfera
- Dirección hacia la que la esfera está apuntando
- Posición de la esfera

Basándose en estos datos, la red neuronal determinará que variables alterar con tal de decidir sobre los movimientos o rotación del agente para desencadenar acciones que lo recompensen o penalicen hasta encontrar la estrategia correcta.

SISTEMA DE RECOMPENSAS

Se establece la tarea como completada al detectar una colisión entre la pelota y la pared, que es de color verde dentro de la simulación. Esto sumará una recompensa de 1. La otra manera en la que el agente es recompensado, se divide un punto entre la cantidad de veces que el agente haya colisionado con la pelota, esto garantiza que la inteligencia artificial se asegure de recibir incentivos para que colisione con la esfera pero que busque la manera óptima de tocarla lo menos posible.

AUDIO GENERATIVO

Para crear e integrar la música y los efectos de sonido dentro de esta demostración fue utilizado el programa *SuperCollider*. Este software de uso libre consiste en un entorno de programación para la síntesis de audio y composición algorítmica. *SuperCollider* se sustenta de su propio lenguaje de programación (sclang), su servidor de audio en tiempo real scsynth y su editor de código *scide*, para la creación de componentes de audio conocidos como *Synths*. Cuando un componente o *Synth* es instanciado, puede recibir algunos argumentos que provoquen un cambio en los sonidos generados durante su ejecución. Algunos de estos componentes fueron seleccionados de la colección SCLOrkSynths, un proyecto que reúne algunos aportes de la comunidad de SuperCollider (McCartney, 1996). Estos componentes fueron modificados para adecuarse a la visión que se tenía del proyecto, aunque otros se usaron tal y como fueron definidos por sus creadores, únicamente cambiando el valor de sus parámetros al instanciarlos.

En cuanto a la composición del tema principal, se utilizaron procesos algorítmicos de la mano con conceptos de teoría musical. El tema está constituido de cuatro acordes pertenecientes a la escala de Do mayor, que son complementados por sus séptimas, novenas y onceavas notas para darles corporeidad y presencia. Estos cuatro acordes son la base sobre la que todos los *Synths*, o instrumentos, construyen una atmósfera relajante propia de un mundo de fantasía con elementos de naturaleza. Al instanciar cada instrumento, algunos de los parámetros para su llamada son definidos de forma pseudoaleatoria, tales como el espacio entre las notas que componen a un acorde, la duración de estas, sus frecuencias, volúmenes, paneo, entre otras propiedades del sonido. Así, este tema puede reproducirse indefinidamente sin repetirse hasta que el programa se detenga.

Al momento de reproducir los efectos de sonido, también creados a través de *Synths*, es utilizado el protocolo OSC. *Open Sound Control* es un protocolo de red que establece comunicación entre aplicaciones multimedia, computadores y sintetizadores en tiempo real, siendo el puente que conecta Unity con SuperCollider para la gestión de audio interactivo. Por un lado, SuperCollider actúa como servidor y escucha mensajes OSC en un puerto específico, los cuales contienen comandos para reproducir sonidos y cambiar parámetros de síntesis. Desde el otro extremo, Unity envía mensajes OSC al servidor de SuperCollider empleando la biblioteca *extOSC*, que facilita el envío de mensajes por medio de este protocolo. Estos mensajes son desencadenados por eventos dentro del ambiente de la experiencia y contienen rutas de destino con parámetros para la ejecución de funciones desde SuperCollider. Por todo lo anterior, el protocolo OSC resulta ser bastante flexible y eficaz para desarrollar un sólido sistema de audio interactiva.

En cada instrumento, existe una dirección "/amp[Instrumento]", a la cual se envía una señal de activación cada vez que ocurre un evento en específico, en este caso cuando el jugador se encuentra con un nuevo asistente o "fungi", y aumenta el volumen de dicho instrumento en su totalidad, haciendo que comience a sonar y el tema principal vaya tornándose más complejo al avanzar en el juego. Con cada movimiento del jugador, la distancia de este hasta un área deseada es calculada, y de ser menor a 40 unidades, un mensaje hacia la dirección "/waterProximity" es enviado junto con esta distancia, para lograr que el volumen y otras propiedades de la síntesis granular de la secuencia de sonidos de río aumente o disminuya. De igual manera, cada vez que el personaje principal toca el terreno a avanzar, una señal es enviada a la dirección "/walk", lo que reproduce un sonido de contacto con el suelo. Finalmente, cada tipo de asistente tiene una paleta de sonidos para hablar o interactuar, que es utilizada al enviar mensajes a las direcciones "/[fungi]Talk", para hacer que se reproduzca una serie de notas pseudoaleatorias con las respectivas voces de los "fungis", simulando su habla, y "/[fungi]No", cuando un asistente no puede llevar a cabo una orden recibida. Además, se envía un mensaje a la dirección "/fungiJump" cuando dos Fungis establecen relaciones diplomáticas y comienzan a saltar alegremente. Este mensaje contiene un valor que aumenta o decrementa el tono de reproducción del sonido de salto.

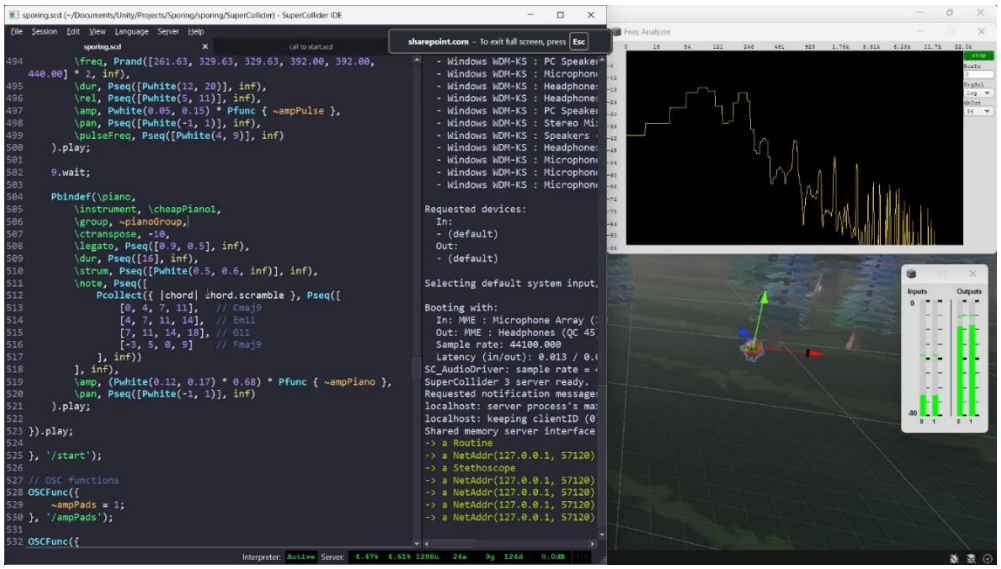


Figura 5. Entorno de trabajo en SuperCollider y Unity

La síntesis de diferentes herramientas de IA en Sporing

Una vez realizamos una exploración de las herramientas que estaban a nuestro alcance, buscamos diseñar y unificar estos conocimientos en un videojuego que les sacara provecho. De tal forma que pudiera demostrar cómo es que, gracias a la Inteligencia Artificial, podemos complementar las experiencias de juego, y así, generar ambientes e interacciones más inmersivas y personalizadas para cada jugador.

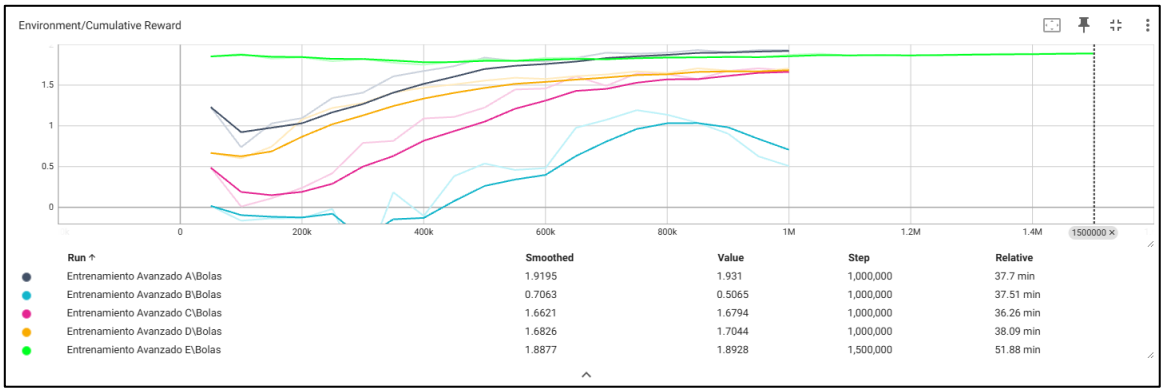
Es ahí cuando nace *Sporing*, un juego en el que controlas a un pequeño pájaro que necesita descubrir e interactuar con diferentes tipos de hongos con cualidades únicas, para traducir, saltar y servir como plataforma para alcanzar zonas altas del mapa, ya que nuestro personaje principal carece de muchas habilidades y no podría avanzar sin su ayuda. Todos los movimientos y mecánicas de los hongos son controladas a través del *Navmesh***,** herramienta de la cual ya se dieron especificaciones sobre su funcionamiento. La música ambiental del juego, desarrollada en SuperCollider, agrega una interactividad dinámica con el audio, que ofrece melodías que se adaptan a las interacciones que el jugador tiene con el entorno del videojuego. Buscamos generar en cada jugador sensaciones de misterio, paz, armonía y exploración, que le permitieran olvidarse de muchas de las convenciones usados en la mayoría de los videojuegos actuales. No quisimos recurrir a decirle al jugador de manera obvia hacia dónde debe dirigirse, sino que conociera y apreciara todo lo que tiene a su al rededor por voluntad y habilidad propia.

Eso es Sporing, una obra con un diseño de juego detallado, que se sirve de tecnologías actuales como la Inteligencia Artificial y la generación procedural, para reforzar las intenciones que tenemos en la construcción de la experiencia que el juego abarca.

Figura 6. Personaje principal y hongos en Sporing

Resultados del sistema y entrenamiento

Además de lograr complementar un concepto de juego con herramientas de la Inteligencia Artificial, mostramos algunos de los resultados obtenidos en la etapa de exploración con las sesiones de entrenamiento de los agentes autónomos con ML-Agents. Gracias a las diferentes etapas en las que dividimos estas tareas, pudimos llegar hasta puntuaciones muy cercanas con 1.92 con respecto al valor máximo de 2.0, que representa la manera óptima de ejecutar la tarea.



Run	Smoothed	Value	Step	Relative
Entrenamiento Avanzado A\Bolas	1.9195	1.931	1,000,000	37.7 min
Entrenamiento Avanzado B\Bolas	0.7063	0.5065	1,000,000	37.51 min
Entrenamiento Avanzado C\Bolas	1.6621	1.6794	1,000,000	36.26 min
Entrenamiento Avanzado D\Bolas	1.6826	1.7044	1,000,000	38.09 min
Entrenamiento Avanzado E\Bolas	1.8877	1.8928	1,500,000	51.88 min

Figura 7. Resultados de diferentes sesiones de entrenamiento con los ml-agents

Conclusiones

En este trabajo, se presentó una demostración técnica de un videojuego cuyas características principales fueron desarrolladas con herramientas de inteligencia artificial. Por un lado, se buscó mejorar el apartado sonoro y musical con el objetivo de crear un entorno auditivo cambiante e inmersivo en función de la exploración. El entorno de *SuperCollider* resultó ser una buena opción para lograr esto debido a su eficacia al sintetizar y procesar audio en tiempo real para responder al entorno, además de que ofrece una vasta colección de bloques a fin de construir diversos sonidos. Aunque *SuperCollider* no implemente IA de manera nativa, gracias a su compatibilidad con otros entornos de desarrollo como Unity, puede combinarse con recursos externos de este tipo para dar lugar a proyectos que fusionen la síntesis y el procesamiento de audio avanzado con métodos de aprendizaje automático.

En el desarrollo de agentes inteligentes, el aprendizaje por refuerzo probó ser un método dinámico para que se adaptasen a sus alrededores con tal de acompañar al jugador y cumplir con ciertas indicaciones. Para lograrlo, se recurrió al algoritmo PPO que ofrece el SDK de Unity ML-Agents. PPO mejora la estabilidad del aprendizaje durante al entrenamiento al limitar los cambios en la política o reglas a seguir con el mecanismo de "*clipping*". Por supuesto, este método para el entrenamiento sigue sin ser perfecto, ya que los agentes pueden desarrollar comportamientos impredecibles si no se estiman bien los parámetros y recompensas, sumado a la cantidad de tiempo y recursos que se consumen. A pesar de los desafíos que presenta en términos de costos e imprecisión, el aprendizaje por refuerzo puede dar beneficios significativos que cambien la manera de jugar con cada interacción, repercutiendo en la inmersión y rejugabilidad de un videojuego. Con la evolución continua de modelos de aprendizaje por refuerzo y de aprendizaje automático, se abre la posibilidad de ver una mayor integración de estas técnicas en las mecánicas de múltiples videojuegos en el futuro.

Los videojuegos simbolizan un punto de convergencia para múltiples disciplinas, sea visto desde una perspectiva artística o lógico-matemática. El rol que ha tenido la inteligencia artificial en su evolución es innegable, pero queda claro que los videojuegos aún tienen camino por recorrer en este terreno. Hasta la fecha, muchos de los algoritmos y estrategias utilizadas en sistemas de videojuegos parten prácticamente de la misma base con leves modificaciones entre sí. Esta es una tendencia que está comenzando a cambiar dada la saturación del mercado, donde la búsqueda de innovar en la creación de experiencias de juego es cada vez mayor con tal de sobresalir. En los próximos años, se espera que la IA desempeñe un rol más central en la creación y personalización de contenido, dando espacio a los seres humanos de enfocarse en la creatividad y experimentación, que siempre serán piezas clave en el proceso de diseño de cualquier proyecto artístico.

Referencias

David. (2024, January 28). *Intelligent Play: Intro To NPC Design And AI In Modern Video Games*. Retrieved from Scenegraph Academy: https://scenegraph.academy/article/intelligent-play-intro-to-npc-design-and-ai-in-modern-video-games/

Delgado, M. (2024, April 10). *El papel de la inteligencia artificial en los videojuegos - The Good Gamer*. Retrieved from The Good Gamer: https://thegoodgamer.es/el-papel-de-la-inteligencia-artificial-en-los-videojuegos/

Juliani, A., Berges, V. P., Teng, E., Cohen, A., Harper, J., Elion, C., . . . Lange, D. (2020). Unity: A General Platform for Intelligent Agents. *arXiv*. doi:https://doi.org/10.48550/arXiv.1809.02627

Kanade, V. (2022, September 29). *Everything you should know about reinforcement Learning - Spiceworks Inc.* Retrieved from Spiceworks Inc.: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-reinforcement-learning/

Lopez Duarte, A. E. (2020). Algorithmic interactive music generation in videogames: A modular design for adaptive automatic music scoring. *SoundEffects - An Interdisciplinary Journal of Sound and Sound Experience*, 38-59. doi:https://doi.org/10.7146/se.v9i1.118245

Manju, S., & Punithavalli, M. (2011). An Analysis of Q-learning Algorithms with Strategies of Reward Function. *International Journal on Computer Science and Engineering*, 814-815.

McCartney, J. (1996). *GitHub - supercollider/supercollider: An audio server, programming language, and IDE for sound synthesis and algorithmic composition.* Obtenido de GitHub: https://github.com/supercollider/supercollider

Millington, I., & Funge, J. (2009). *Artificial Intelligence for Games, Second Edition.* Burlington, MA: Morgan Kaufmann Publishers Inc.

Ocio Barriales, S. (2014). Inteligencia Artificial en la industria del videojuego AAA ¿Puede el mundo académico contribuir a su éxito?

OpenAI. (2018). *Proximal Policy Optimization — Spinning Up documentation*. Obtenido de OpenAI Spinning Up: https://spinningup.openai.com/en/latest/algorithms/ppo.html

Optimización de políticas próximas (PPO) en el aprendizaje por refuerzo. (s.f.). Obtenido de Code Labs Academy: https://codelabsacademy.com/es/blog/proximal-policy-optimization-ppo-in-reinforcement-learning

Van Toll, W. G., Cook, A. F., & Geraerts, R. (2012). A navigation mesh for dynamic environments. *Computer Animation and Virtual Worlds*, 535-546. doi:10.1002/cav.1468

Wikipedia, C. d. (s.f.). *Open Sound Control*. Obtenido de Wikipedia, la enciclopedia libre: https://es.wikipedia.org/wiki/Open_Sound_Control

Zhao, Z., Liu, H., Li, S., Pang , J., Zhang, M., Qin, Y., . . . Wu, Q. (2022). A Review of Intelligent Music Generation Systems. *arXiv preprint*. doi:arXiv:2211.09124