ESTUDIO E IMPLEMENTACIÓN DE *MACHINE LEARNING* EN EL DESARROLLO DE VIDEOJUEGOS

Juan Diego Duarte Antolinez

Trabajo de grado para optar al título de Ingeniero de Sistemas

Director:

Nitae Andrés Uribe Ordoñez

Universidad Autónoma de Bucaramanga.

Facultad de Ingeniería.

Programa de Ingeniería de Sistemas.

Proyecto de Grado en Ingeniería de Sistemas.

Junio 2019

Dedicatoria

A mi familia

A mis profesores

En especial a mi madre

Agradecimientos

A mi familia, por ser un apoyo incondicional en toda mi trayectoria tanto académica como personal, acompañando la realización de todas mis metas entre las cuales está la presentación de este trabajo de tesis.

A mis maestros, por el apoyo constante en cada una de sus especialidades con las que logré direccionar el rumbo de esta investigación como base de mi crecimiento profesional.

A mi madre, cuya solicitud e insistencia permanente hacia mí y este proceso fue el de estudiar, para lograr estructurar, profundizar y aprender de lo que inicialmente fue tan solo una idea.

Al ingeniero Nitae Uribe, director de este proyecto de grado, por compartir su conocimiento, intencionalidad, dedicación y permanentes aportes, porque gracias a él logré clarificar la temática y profundizar acerca de la automatización, además de la confianza en mi potencial.

A mis amigos, que escucharon incontables versiones de este documento con el fin de tener un documento completo como anexo a mi carrera profesional.

Tabla de contenido

Resumen

En el presente se hace una breve investigación acerca de los métodos más comunes para el estudio de datos en *Machine Learning*, con un enfoque al desarrollo de video juegos en la herramienta Unity 3D. El tipo de aprendizaje más común en la industria de los video juegos es el aprendizaje reforzado que usa los algoritmos para evaluar repetidamente un escenario de datos. Por medio de un prototipado de aprendizaje basado en agentes, se busca como resultado una plantilla que funcione como base de futuros proyectos de *Machine Learning* aprovechando las ventajas de Unity 3D como plataforma de simulación, tomando como caso de estudio la empresa colombiana de juego FRYOS STUDIOS S.A.S.

Lista de tablas

Lista de figuras

Introducción

El desarrollo del proyecto de grado propuesto está orientado al estudio del *machine learning* y su aplicación en los video juegos. *Machine learning* es un campo dentro de la inteligencia artificial, que reúne algoritmos capaces de aprender e imitar un comportamiento racional acorde a una situación de manera autónoma. Su principal característica es la capacidad de mejorar autónomamente sus comportamientos basado en la experiencia y sin la intervención humana. (Martínez, 2017)

Por otro lado, un video juego está compuesto por dos áreas: (1) Área técnica: control de gráficas, sonidos, simulaciones físicas, entre otras (FUNGE, ARTIFICIAL INTELLIGENCEFOR GAMES, 2009); (2) Mecánicas: son aquellos aspectos determinantes de la esencia del video juego, tales como reglas, retos y finalidades que rigen su contexto para construir una experiencia única de usuario. Su programación debe ser soportada por las capacidades técnicas del hardware, y su estructura interna está basada en recursos (sonidos, los modelos 3d o texturas, entre otras) (Leo Galway, 2009). El video juego recibe estímulos de un jugador a través de un periférico como lo puede ser un mando portátil, el teclado, u otro hardware compatible, para procesar las acciones que deben ser interpretadas por la mecánica. Las mecánicas son el componente que aporta el insumo diferenciador del video juego, es decir, lo que hace único a cada uno. (FUNGE, ARTIFICIAL INTELLIGENCEFOR GAMES, 2009)

FRYOS STUDIOS S.A.S. es una compañía que pertenece al sector de software y servicios informáticos (SSI), su objetivo principal es realizar productos de contenido digital, tales como

videojuegos, aplicativos móviles y páginas webs. Desde el 2015 ha estado realizando diversos desarrollos en la industria de los videojuegos con enfoque móvil por medio de la plataforma Unity 3D, un software de desarrollo de video juegos multiplataforma. Gran parte de las mecánicas en los video juegos de FRYOS STUDIOS S.A.S. son basadas en agentes, como lo es un personaje con el que un jugador pueda sentirse identificado, un enemigo que represente un reto, o un PNJ (personaje no jugable) que pueda ayudar al jugador con el reto que deba resolver en un escenario. La programación actual que se implementa en el desarrollo de las mecánicas de los videojuegos que produce FRYOS STUDIOS S.A.S. son programadas de manera convencional, lo cual se traduce en tiempos de desarrollo más extensos y menos dinámicos que impactan en los resultados de la experiencia de usuario, además de temas financieros en desarrollo de sus proyectos.

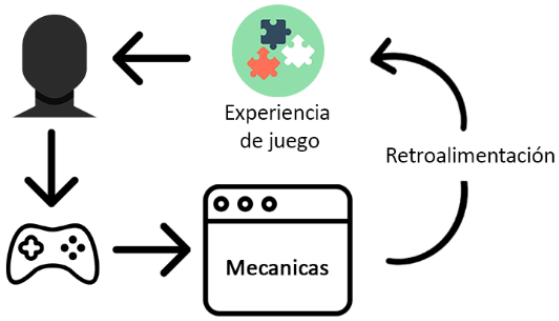

Figura 1. Un video juego visto como sistema. Fuente: Autor

Un video juego visto como un sistema, recibe información del periférico para evaluarla según la mecánica, y acorde a cada decisión tomada en cada instante del videojuego, cambia el estado y la experiencia de usuario dado que dentro de este sistema existe un número finito de agentes con un número significativo de acciones que puede realizar de acuerdo a las mecánicas, escenarios, usuario, entre otros, que para correcto funcionamiento se debe programar, y es ahí, donde la

inteligencia artificial, más exactamente *machine learning,* permitiría de manera más versátil entrenar los agentes del sistema para que respondan de manera autónoma ante una situación posible como resultado de las mecánicas del videojuego.

Para atender la problemática anteriormente descrita, el trabajo de grado realizado consistió en el desarrollo de una librería para la programación de videojuegos en Unity 3D empleando técnicas de *Machine Learning*, direccionado a la esencia de los videojuegos desarrollados en FRYOS STUDIOS S.A.S.

Planteamiento del problema

Los agentes de los video juegos se comportan de distinta manera según el contexto del proyecto, es decir, que los personajes jugables puedan moverse, saltar, agacharse, entre otras acciones; mientras que los personajes no jugables deban atacar o esquivar, incluso una plataforma que deba subir o bajar por la presencia de un jugador. Cada agente enfrenta una situación en la cual debe tomar una decisión, la programación de estos agentes en los video juegos consume gran una parte de los tiempos de desarrollo, lo que los convierte en un punto clave para la construcción de la experiencia de usuario que es el resultante del video juego visto como un sistema.

El tiempo total de desarrollo de un videojuego para dispositivos móviles en FRYOS STUDIOS S.A.S. se estima en un promedio de ocho (8) meses, dentro del cual, a nivel general, las bases de programación en una mecánica estándar de agentes consumen entre uno (1) y tres (3) meses, acorde a la complejidad de las situaciones y acciones que debe tomar el agente. Lo anterior conlleva a formular la siguiente pregunta: **¿De qué manera se puede disminuir los tiempos de desarrollo de programación de la mecánica de los agentes a partir de una librería que implemente *machine learning* en Unity 3D?**

Justificación

La implementación de *machine learning* en el desarrollo de video juegos puede conseguir comportamientos de inteligencia más enriquecidos que contribuyen a la construcción de una mejor experiencia de usuario. Empresas dedicadas al desarrollo de video juegos que usen Unity 3d, como es el caso de FRYOS STUDIOS S.A.S. podrían hacer uso del *machine learning* por medio de una librería que se integre adecuadamente con la plataforma de desarrollo unity 3d, con el fin de enriquecer la inteligencia artificial programada en los agentes de sus video juegos. En la figura 2, se ilustra como el uso de la librería no está limitada por la etapa del proyecto, pudiéndose integrar al principio o en medio de un desarrollo.

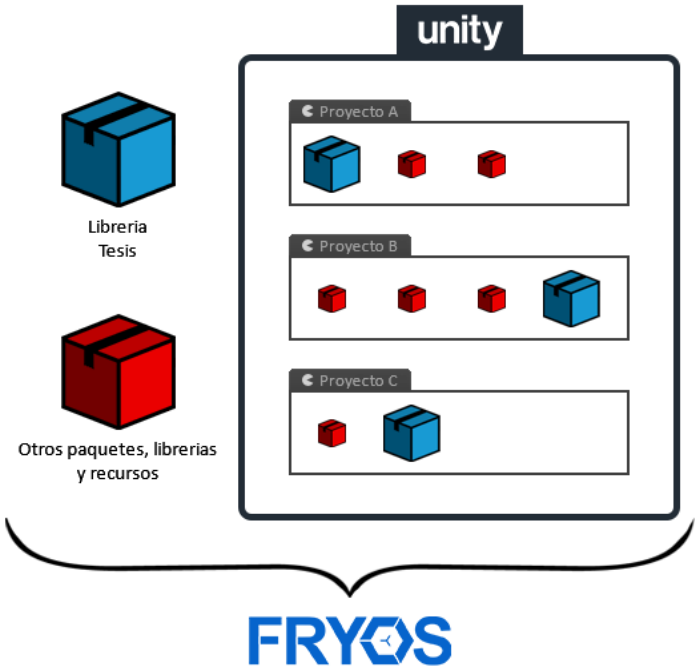

Figura 2. Uso de librería machine learning en Unity 3D. Fuente: Autor

Objetivos

Objetivo General

- Desarrollar una librería para la programación de videojuegos en Unity 3D empleando técnicas de *Machine Learning*.

Objetivos Específicos

- Estudiar el estado del arte de las técnicas de *Machine Learning* y su aplicación en la creación de video juegos en Unity 3d.

- Implementar un escenario de entrenamiento de agentes para el uso de *Machine Learning* en Unity 3D.

- Analizar los resultados obtenidos sobre la implementación del *Machine Learning* en base al escenario de entrenamiento.

Metodología

El proyecto de grado realizado es de enfoque mixto puesto que integra el enfoque cuantitativo y cualitativo, a través de los procesos inductivo, deductivo y secuencial. Desde el enfoque cualitativo, a partir de revisión documental en fuentes primarias, secundarias y terciarias de información fueron exploradas las técnicas de *machine learning* y su aplicación en la creación de videojuegos. Desde el enfoque cuantitativo, fue realizada la recolección de datos de librería de código fuente del funcionamiento completo de Unity 3D *toolkit*, las cuales fueron analizadas para el desarrollo de una librería propia para automatizar la programación de las mecánicas de los agentes.

El alcance de la investigación es de tipo experimental, debido que, a partir del trabajo realizado, se desarrolló una librería base para la programación de agentes aplicados en un escenario prototipado en Unity 3D.

El resultado principal del proyecto consistió en el desarrollo de una librería para la programación de videojuegos en Unity 3D empleando *Machine Learning*.

Para el desarrollo del proyecto de grado, se consideraron un total de tres (3) fases relacionadas directamente con los objetivos específicos del proyecto, las cuales son presentadas a continuación:

Primera Fase: Estudio del estado del arte de las técnicas de *Machine Learning* y su aplicación en la creación de video juegos en Unity 3D. Para el desarrollo del objetivo, se definen las siguientes actividades:

- Revisión bibliográfica de fuentes primarias, secundarias y terciarias de información sobre: (1) Técnicas de *Machine Learning*; (2) Aplicación de *Machine Learning* en la creación de videojuegos e; (3) Implementación en la estructura de Unity 3D.

- Análisis de la información recuperada a partir de los siguientes ejes temáticos: (1) *Machine Learning*: Análisis de la inteligencia Artificial en video juegos, revisión Unity *Machine Learning* toolkit y; (2) Mecánicas de video juegos: estructura de la programación convencional, brechas de mejoramiento, ambientes automatizables, entre otras.

Segunda Fase: Implementación un escenario de entrenamiento de agentes para el uso de *Machine Learning* en Unity 3D. Para el desarrollo del objetivo, se definen las siguientes actividades:

- Diseño del escenario de aprendizaje
- Determinación del agente y del escenario de aprendizaje
- Definición del objetivo del agente dentro del escenario de aprendizaje
- Determinación de los atributos del sistema desde el enfoque del agente desde las técnicas de *Machine Learning*
- Entrenamiento del agente a partir de la implementación de las del sistema desde el enfoque del agente

Tercera Fase: Análisis de los resultados obtenidos sobre la implementación del *Machine Learning* en base al escenario de entrenamiento. Para el desarrollo del objetivo, se definen las siguientes actividades.

- Análisis comparativo del comportamiento del agente a partir de la programación convencional y las técnicas de *Machine Learning*

Marco de referencia

Marco conceptual

Inteligencia artificial

La inteligencia artificial se refiere a la imitación de procesos inteligentes por parte de máquinas (FUNGE, ARTIFICIAL INTELLIGENCEFOR GAMES, 2009), entre los elementos que componen el sistema que realiza esto procesos destacan el lenguaje, la información, el entendimiento y la realimentación. Un sistema de inteligencia artificial recibe una información que debe ser entendida en un lenguaje recibiendo una realimentación sobre el proceso. Ahora bien, desde el enfoque de la solución de un problema, podría contextualizarse en una búsqueda de soluciones por parte del sistema con base en su información. De esta forma, el concepto de búsqueda se hace importante debido a que a raíz de este se ramifican las diferentes diciplinas que permiten modelar la inteligencia en una máquina.

Los métodos de búsqueda heurísticas están orientados a reducir la cantidad de búsqueda requerida para encontrar una solución. Feigenbaum y Feldman definen la heurística como sigue: "Una heurística es una regla para engañar, simplificar o para cualquier otra clase de ardid el cual limita drásticamente la búsqueda de soluciones en grandes espacios de estados". (Herrera, 2015)

En esencia una heurística es simplemente un conjunto de reglas que evalúan la posibilidad de que una búsqueda va en la dirección correcta. Generalmente los métodos de búsqueda heurísticas se basan en maximizar o minimizar algunos aspectos del problema. (Martínez, 2017). La búsqueda

heurística no garantiza un resultado del todo acertado, se basa en la media de los posibles resultados buscando el matemáticamente optimo basado en ciertas restricciones, el estudio de estos métodos en búsqueda del modelamiento de la inteligencia ha sido ampliamente estudiados a lo largo de la historia. Dado que está dividido en ramas, el concepto de inteligencia fluctúa dependiendo del análisis, todas parecen acercarse a un campo de estudio donde su principal objetivo es encontrar métodos efectivos para aplicar inteligencia de resolución de problemas y otras ramas a una amplia gama de problemas prácticos. Algunas de las áreas de aplicación para la inteligencia artificial son: (Leo Galway, 2009)

- *The use of computers to do symbolic reasoning*
- *A concern with problem solving using inexact missing or poorly defined information*
- *An effort to capture and manipulate the significant qualitative features of a situation rather than relying on numeric methods*
- *Answers that are neither exact nor optimal, but are in some sense "Sufficient"*

Machine learning

Machine learning es una rama de la inteligencia artificial encargada de estudiar técnicas que permitan modelar el conocimiento a partir del aprendizaje. El *machine learning* fue infundado principalmente por el desarrollo y las capacidades tecnológicos que fue adquiriendo la humanidad con el tiempo, tiene una gran variedad de áreas de aplicación, entre las más destacadas está el campo de la genética, la seguridad informática, el transporte autónomo, entre otras, y sus objetivos pueden ser de predicción, o identificación de patrones complejos basados en los datos, por

mencionar algunos (Martínez, 2017). En un ejemplo práctico de un problema con dos variables, el espacio de datos se representa en un plano cartesiano, como se observa en la figura 3 donde hay 3 distintos puntos ubicados en la gráfica.

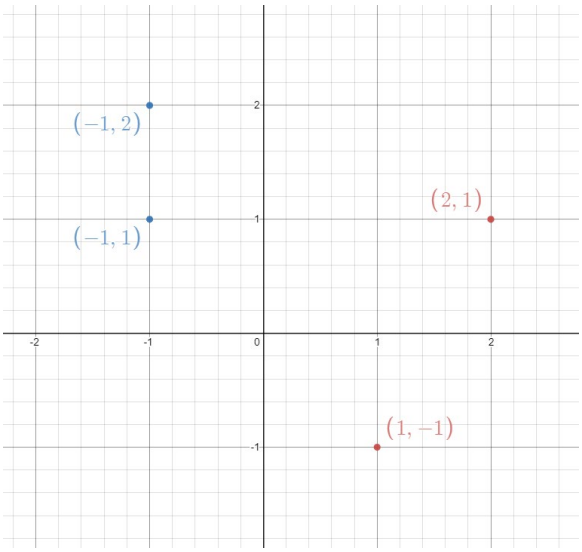

Figura 3. Plano de dos dimensiones. Fuente: Autor

Cada uno de los puntos graficados en esos espacios será usado para modelar bajo cierto criterio el conocimiento que podría estar implícito en ellos, pero la interpretación de estos datos puede variar dependiendo del contexto del problema. Existen diversas técnicas para agrupar problemas de *machine learning* entre las cuales destacan, el aprendizaje supervisado, el no supervisado, el semi supervisado y el reforzado (Martínez, 2017).

Marco teórico

Aprendizaje Supervisado

El aprendizaje supervisado se caracteriza por etiquetar cada uno de los datos en el espacio bajo criterios determinados, este aprendizaje generaliza el conocimiento a partir de reglas que clasifican la salida del sistema como correcto o incorrecto según el contexto. Esta salida no está limitada a un valor discreto, como podría ser VERDADERO o FALSO, pero estará catalogada como una de las posibles salidas contempladas desde el inicio para el sistema (John Schulman, 2017)

En un problema de 2 variables, cuyos datos han sido catalogados con 2 posibles valores [AZUL, NARANJA], el grafico que representaría los datos basados en el coeficiente de cada una de sus variables se puede observar en la figura 4.

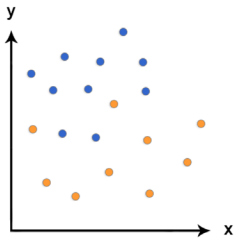

Figura 4. Datos en el aprendizaje supervisado. Fuente: Autor

Una vez que los datos están catalogados los sistemas de aprendizaje supervisado podrán modelar las reglas que determinaran sus salidas condicionadas a su vez por el contexto de problema, existen

dos grandes grupos de problemas en el aprendizaje supervisado: (1) Problemas de regresión y; (2) Problemas de clasificación.

Problemas de regresión

En los problemas de regresión, la salida del sistema realiza una predicción con base en la relación que existe entre cada una de las variables de los datos, en el ejemplo anterior trazando una recta sobre el espacio bidimensional que aborde la mayor cantidad de datos, el sistema podría intuir para valor X un valor Y, y viceversa, ver figura 5. (Fonseca-Reyna, 2017)

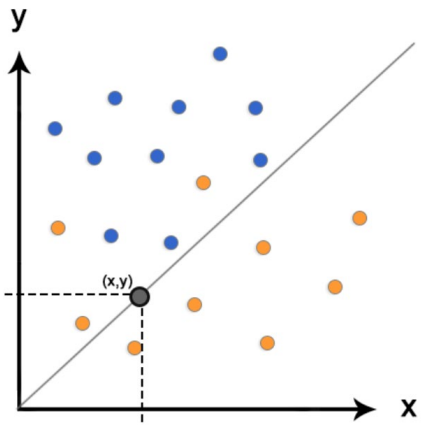

Figura 5. Gráfico de regresión lineal. Fuente: Autor

Problemas de clasificación

En los problemas de clasificación, el objetivo es catalogar un nuevo dato dentro de uno de los grupos definidos en el planteamiento del problema, en este caso la entrada del sistema es un nuevo punto que puede ser catalogado según predicciones en un valor discreto (Herrera, 2015) Con los

datos en el ejemplo anterior al trazar una recta que divida la concentración de datos que sean de un grupo u otro se obtendría una gráfica como la que se puede observar en la figura 6.

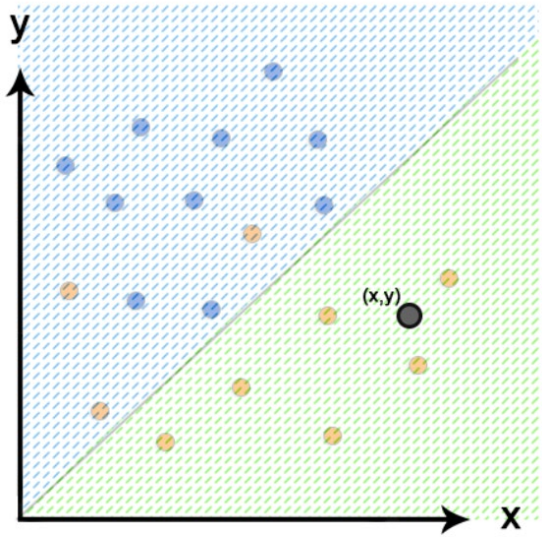

Figura 6. Gráfico de clasificación. Fuente: Autor

De esta forma para cada nuevo dato le corresponderá un grupo en el espacio permitiendo así catalogarlos, el problema este caso depende de ambas variables para procesar un nuevo resultado.

Algoritmos aprendizaje supervisado

- Algoritmo regresión lineal
- Algoritmo regresión logística
- Cuantificador Bayesiano Ingenuo
- Algoritmo k vecinos más próximos
- Algoritmo arboles de decisión

- Algoritmo bosques Aleatorios
- Redes neuronales

Aprendizaje no supervisado

La principal característica del aprendizaje no supervisado es que los datos en el espacio no están catalogados, contrario al ejemplo anterior cada dato tiene un valor según la cantidad de variables que haya en el sistema, pero más allá de ocupar un espacio no tienen una categoría o parámetro que permita relacionarlo (John Schulman, 2017). Por medio de un plano cartesiano de dos dimensiones un grupo de datos en el aprendizaje no supervisado se graficaría como se observa en la figura 7.

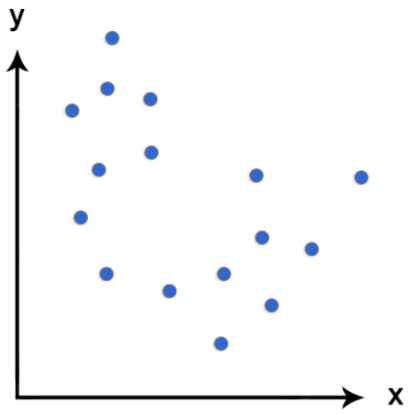

Figura 7. Datos en el aprendizaje no supervisado. Fuente: Autor

El objetivo del aprendizaje no supervisado es identificar las relaciones implícitas que puedan tener los datos, la forma en la que el sistema interpreta estos los valores de cada dato se dividen en dos grandes grupos, problemas de agrupación y de reducción de dimensionalidad.

Problemas de agrupación

Los sistemas basados en el problema de agrupación buscan patrones o estructuras que definan un comportamiento recurrente en los datos (Fonseca-Reyna, 2017), para el ejemplo anterior con múltiples datos en un espacio bidimensional un sistema de aprendizaje no supervisado desde el punto de vista de un problema de agrupación podría establecer dos posibles grupos basados en la ubicación en el espacio, el grafico que representa se observa en la figura 8.

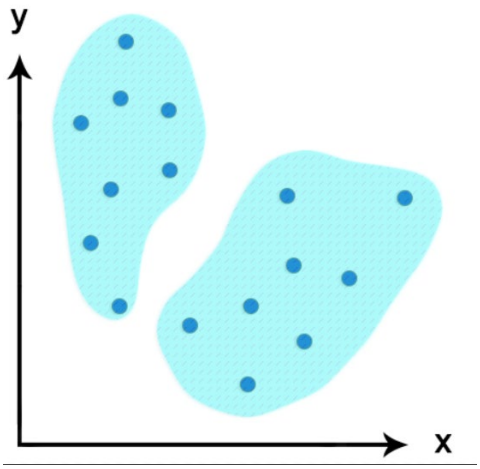

Figura 8. Agrupación no supervisada. Fuente: Autor

El sistema en este punto habrá encontrado que entre algunos puntos sin catalogar existen dos posibles grandes categorías, estas categorías se basan en similitudes en los datos que aun sin conocer el propósito real de la categoría, consigue agrupar datos suficientes para hacerla relevante.

Problemas de reducción de dimensionalidad

Este tipo de problema también es conocido como análisis de componentes principales, el cual tiene como objetivo analizar los datos en busca de aquellas variables menos representativas para modelar un comportamiento o patrón dentro del espacio, la más notoria ventaja de este planteamiento son la reducción de complejidad tanto para el análisis como para el cálculo de los datos (Fonseca-Reyna, 2017). De acuerdo con el ejemplo anterior el sistema se denota la correlación entre los puntos que probablemente puede ser representada por 3 nuevas variables fundamentales que rigen el funcionamiento de los datos, estas 3 variables hipotéticas serian graficadas en un plano como se observa en la figura 9.

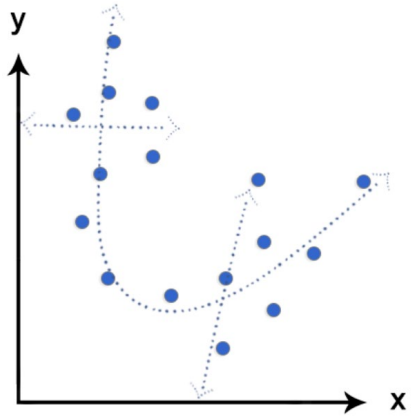

Figura 9. Gráfico de dimensionalidad. Fuente: Autor

Estas 3 nuevas variables que el sistema pudo haber encontrado pueden llegar a ser la clave para predecir, clasificar o modelar el comportamiento de los datos, simplificando el problema a uno más estable.

Algoritmos aprendizaje no supervisado

- Análisis de Correspondencias Simple y Múltiple
- Escalamiento Multidimensional
- Análisis Clúster Jerárquico y No Jerárquico
- Análisis de componentes principales
- Análisis Factorial

Aprendizaje semi supervisado

El aprendizaje semi supervisado abarca todo tipo de problema de automatización en el cual los datos pueden estar o no catalogados en el mismo espacio (Michael Bowling, 2006). En un plano bidimensional dependiente de dos variables, podrían existir distintos datos, algunos de ellos sin una clasificación definida. Aunque los datos no catalogados limitan el análisis de los datos, se considera que los parámetros de todos los datos se correlacionan de alguna manera, encontrando así, nuevas categorías, o la categorización de aquellos datos no etiquetados en una de las categorías presentes en el problema. Los datos catalogados y no catalogados se grafican en un plano de dos dimensiones como se observa en la figura 10.

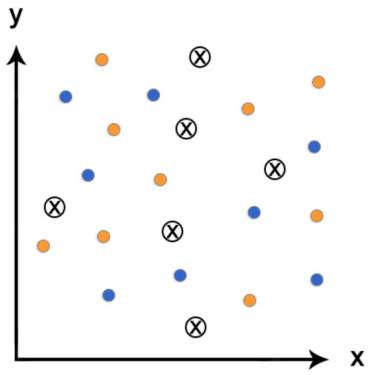

Figura 10. Datos en el aprendizaje semi supervisado. Fuente: Autor

Problemas transducidos (clasificación)

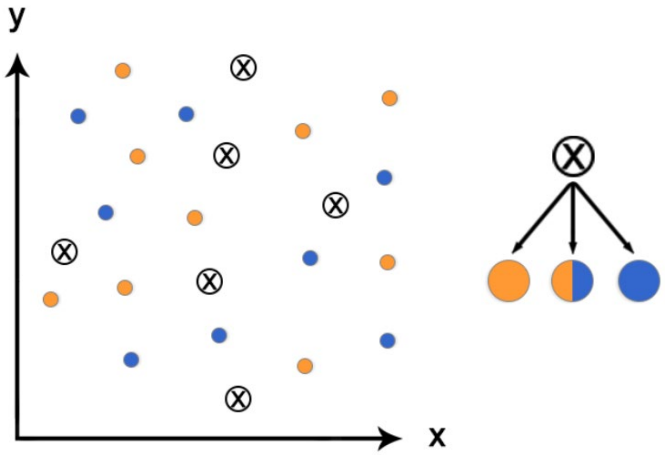

Figura 11. Gráfico de problemas transducidos. Fuente: Autor

El análisis de estos problemas tiene como objetivo el categorizar aquellos elementos en uno de los grupos definidos en el planteamiento o agruparlos en nuevas categorías basadas en su relación con las categorías vigentes.

A partir de las relaciones que existan en los datos el sistema podría ser capaz de determinar un nuevo grupo a partir de la relación de los parámetros presentes en los datos ya catalogados.

Problemas inductivos (predicción)

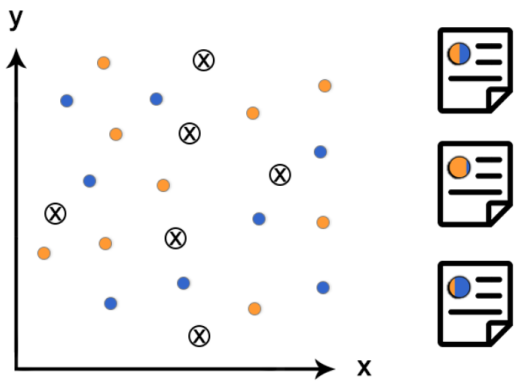

Figura 12. Gráfico de problemas inductivos. Fuente: Autor

En los problemas inductivos el objetivo del sistema es encontrar las generalidades que permiten la categorización, haciendo uso de los elementos no etiquetados reúne aquellos parámetros que podrían deducir la categorización de nuevos elementos (Fonseca Reyna, 2015).

De esta forma, se podría determinar la naturaleza de los elementos etiquetados a partir del estudio de los elementos no etiquetados en función de la relación con los elementos etiquetados.

Algoritmos aprendizaje semi supervisado

- Algoritmo de auto entrenamiento
- Algoritmo de ensamble
- Algoritmo de repeso

Aprendizaje reforzado

El aprendizaje reforzado aborda una situación en la que, en base a un objetivo, se acumula experiencia en un escenario para generar futuras decisiones a partir de la retroalimentación de esas experiencias, de esta forma en cada iteración el sistema podrá modelar las circunstancias que le permitieron conseguir su objetivo. Los tres factores clave que rigen la automatización de este tipo de aprendizaje son: (1) Claridad de un objetivo para el agente; (2) La prueba y error (recompensa o castigo) que reciba el agente en cada iteración sobre el escenario y; (3) La retroalimentación que el sistema le dé al agente basado en su objetivo (Jennifer Hernández Bécares, 2016).

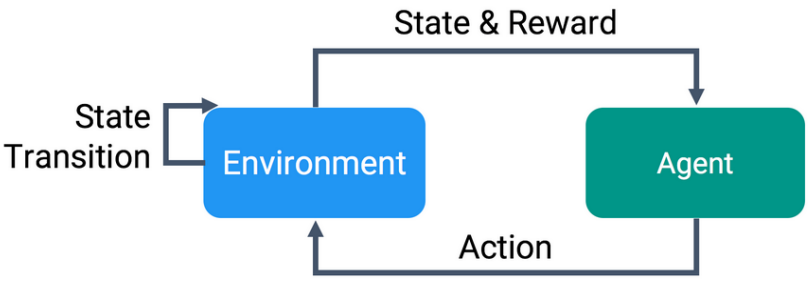

Figura 13. Ciclo de entrenamiento de aprendizaje reforzado. Fuente: Unity 3D

Algoritmos aprendizaje reforzado

- Criterio de optimalidad
- Acercamiento al valor de la función
- Método de Montecarlo
- Métodos de diferencias temporales

- Búsqueda política directa

Estado de arte

Fuente	Tema central	Aporte
Strategy research on the performance optimization of 3D mobile game Development based on Unity **(Zhu, 2011)**	Un estudio sobre métodos óptimos de desarrollo sobre la plataforma de unity 3d, para el desarrollo de proyectos con alta complejidad lógica.	Un control prematuro sobre la optimización permite que el proceso de aprendizaje tenga mayor protagonismo sobre el uso de memoria que consume el video juego como software.
Game Design Narrative for Learning **(Dickey, 2006)**	Narrativa y contexto en el desarrollo de los video juegos	Una visión amplia sobre la narrativa necesaria para conectar una mecánica con el objetivo de un jugador como agente o el agente como un sistema dentro del video juego.
A fast learning algorithm for deep belief nets **(Osindero, 2006)**	Estudio sobre aprendizaje profundo que evaluaba	El aprendizaje reforzado y la importancia en el balance de

	técnicas de fortalecer los sistemas de entrenamiento	los datos sobre la curva de aprendizaje
Artificial Intelligence For games **(FUNGE, Artificial intelligence for games, 2009)**	Donde esta la inteligencia artificial en los video juegos y que técnicas se han usado para lidiar con las limitantes de tecnología al largo del tiempo	Las mecánicas y la estructura de como se desenvuelve el juego para encontrar el punto clave para la implementación de machine learning

Resultados obtenidos

A partir de las actividades realizadas en cada una de las fases definidas, se obtuvo los siguientes resultados:

Resultado de la Primera Fase

Estudio del estado del arte de las técnicas de *Machine Learning* y su aplicación en la creación de video juegos en Unity 3D. De la cual se obtuvo la siguiente síntesis:

Revisión bibliográfica

Existen dos formas básicas de aprendizaje, la adquisición de conocimiento y el refinamiento de habilidades. Por un lado, la adquisición de conocimiento busca acumular suficientes datos para hacer uso efectivo de ellos. Uno de los ejemplos más utilizados es el

estudio de la matemática, donde se adquiere un gran número de fórmulas, ecuaciones, equivalencias y demás recursos propios de la matemática, para hacer uso efectivo de esta información. Por otro lado, el refinamiento de habilidades se refiere a la mejora gradual de habilidades motoras y cognoscitivas, donde la adquisición de conocimiento corresponde solo a la primera etapa del aprendizaje, un ejemplo es la conducción de un vehículo, donde la teoría del manejo no es suficiente para un buen desempeño como piloto. En la actualidad, el estudio del *machine learning* gira sobre tres ejes:

- Estudios orientados a las tareas: A partir de una tarea predeterminada analizar los sistemas de aprendizaje aplicados para mejorar el comportamiento en el desarrollo de esa tarea.
- Simulación cognoscitiva: La investigación y simulación informática proceso de aprendizaje humano.
- Análisis teórico: La exploración teórica del espacio de los posibles métodos de aprendizaje y los algoritmos con independencia del dominio de las aplicaciones.

En los video juegos, el jugador pone a prueba sus habilidades, alterando con cada decisión el flujo del juego, esta dependencia hace que el tipo de conocimiento este catalogado como un refinamiento de habilidades, por la imposibilidad de conocer todos los estados en los que se podría desenvolver la partida de un video juego. Los algoritmos que pueden manejar grandes cantidades de datos como en los video juegos u otros usos del aprendizaje están catalogados según los datos que alimentan el sistema. Algunas de las más reconocidas

técnicas de aprendizaje en el *machine learning* son: (1) Aprendizaje supervisado; (2) No supervisado; (3) Semi supervisado y; (4) Aprendizaje reforzado.

El tipo de aprendizaje en las mecánicas de los video juegos debe ser entendido como un aprendizaje reforzado, evaluando los posibles estados de un video juego mientras refina las habilidades para optimizar la obtención de un objetivo planteado en el videojuego.

Unity Technologies cuenta con un kit de desarrollo para la implementación de machine learning en su plataforma unity 3d, está enfocado en el entrenamiento de agentes por medio de aprendizaje reforzado, este tipo de aprendizaje es alimentado por una serie de recompensas que permiten construir un conocimiento en base a las situaciones experimentadas por el agente.

Mecánicas de video juegos implementando *Machine Learning*

Con el fin de explicar el concepto de mecánicas en los video juegos y donde está la inteligencia artificial en ellas, se tomará como ejemplo Cube Coliseum, un video juego desarrollado por FRYOS STUDIOS S.A.S. En este juego existe una plataforma cuadrada dividida en 16 espacios, sobre esta plataforma un jugador deberá moverse sobre ella en busca de monedas que irán cayendo junto con cubos, que en caso de caer encima del jugador darían por terminada la partida. El juego está basado en turnos un grupo de cubos y monedas caen en la plataforma por cada turno, si el jugador consigue evitar que cualquier cubo lo golpee pasara a la siguiente ronda.

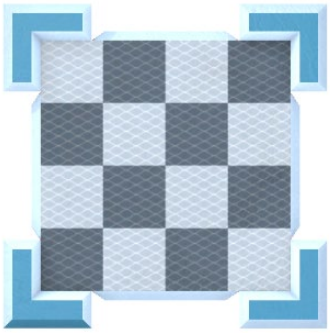

Figura 14. Plataforma cube Coliseum en vista superior. Fuente: Fryos Studios

Por un lado, la mecánica de la plataforma crea una cantidad de cubos que caerán con la intención de eliminar al jugador, esta cantidad es progresiva y aumenta a medida que avanzan los turnos. La posición de caída de los cubos es aleatoria, pueden caer en algunas de las 16 divisiones de la plataforma y cada cubo tiene un peso distinto lo que hace que algunos caigan con más rapidez que otros. En la figura h, se puede observar 4 de los posibles cubos que pueden caer en la plataforma, cada cubo con características distintas.

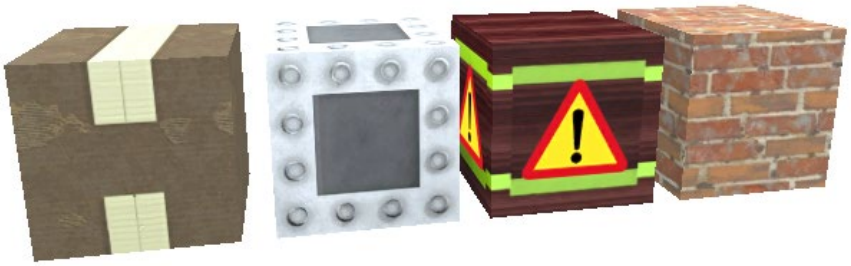

Figura 15. Cubos cube Coliseum. Fuente: Fryos studios

Por otro lado, está la mecánica del jugador, quien debe moverse sobre la plataforma evitando que los cubos caigan sobre él y así pasar al siguiente turno, el turno termina una

vez todos los cubos caigan por completo sobre la plataforma, una vez eso sucede la plataforma sube con el jugador, liberando los 16 espacios para un nuevo turno donde la mecánica de la plataforma creara nuevos cubos que caerán con la intención de eliminar al jugador. En la figura 12 se puede ver un nivel avanzado, donde todos los turnos anteriores están bajo la plataforma, y el jugador ha logrado sobrevivir a la caída de cubos de ese turno.

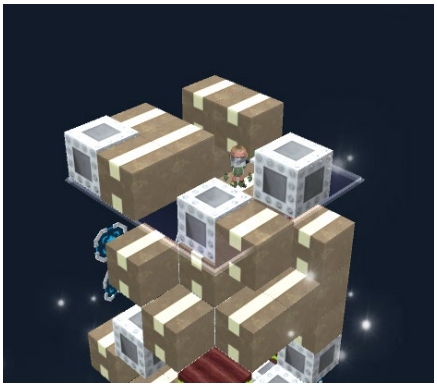

Figura 16. Partida cube Coliseum. Fuente: Fryos Studios

Las monedas que el jugador recoge por cada turno representan un reto por el riesgo que algún cubo pueda caer sobre él. De esta forma tenemos la siguiente tabla que resume las labores de ambas mecánicas

Agente	**Mecánica**	**Objetivo**
Plataforma	Crear una cantidad cubos por turno	Golpear al jugador con uno de los cubos
	Crear una cantidad monedas por turno	Retar al jugador para que sea más probable golpearlo

Jugador	Moverse/Esquivar	Pasar al siguiente turno
		Conseguir las monedas

Tabla 1. Comparativo mecánicas cube Coliseum

Teniendo en cuenta que un agente tiene un grupo de acciones y un grupo de estados, tanto la plataforma como el jugador podrían tomar acciones basadas en un cerebro autónomo, construido en un en un escenario de entrenamiento donde cada agente conoce sus acciones y buscara maximizar su recompensa. Por ejemplo, la plataforma podría identificar los patrones de movimiento del jugador, encontrando los lugares frecuentados al momento de evitar el golpe de los cubos. El jugador por su parte podría encontrar las áreas en donde es más probable que caiga una moneda.

Es importante tener en cuenta que si ambas mecánicas son entrenadas simultáneamente el juego estaría desbalanceado, pues el aprendizaje de cada mecánica será constantemente superado por la otra. Además, un agente no es necesariamente un jugador, como se observó antes cualquier sistema con acciones y estados puede entrenar un cerebro autónomo.

Unity Machine Learning ToolKit

Unity *Machine Learning Toolkit* está orientado al entrenamiento de agentes. Un agente tiene un grupo de estados y un grupo de acciones, cada acción realizada lo llevara a un estado diferente. El entrenamiento dentro de unity 3d es un ciclo de pruebas realizadas en

cada fotograma con las acciones posibles del agente recibiendo recompensas por cada acción que haya tomado. Un escenario de entrenamiento es donde un agente pone a prueba sus acciones cíclicamente, recibiendo recompensas positivas y negativas, con el fin de construir un cerebro autónomo capaz de maximizar la recompensa final basado en los estados que recorre hasta cumplir un objetivo, ver figura 17.

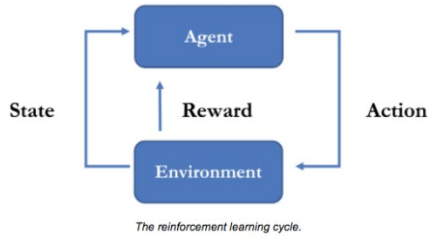

Figura 17. El ciclo del aprendizaje reforzado. Fuente: Unity 3D

En cada ciclo de entrenamiento las acciones tomadas por el agente son evaluadas de forma voraz, buscando sobre cada secuencia de estados, aquellos que consiguen otorgar la mejor recompensa al agente. El aprendizaje reforzado también tiene en cuenta aquellos estados que no consiguen una recompensa alta directamente, sino son el resultante de una serie de estados que lograron maximizar su recompensa, en ese sentido la recompensa esta retrasada y solo será optima recorriendo esa serie de estados. Además, existen estados que no están contemplados inicialmente, son creados en base a acciones que durante el entrenamiento lograron conseguir recompensas aún más altas que los estados iniciales, de esta forma el agente consigue maximizar su recompensa en el entrenamiento, mientras prueba cíclicamente sus acciones en un escenario.

Dentro del ciclo de entrenamiento el agente hace uso de unas observaciones dadas por un escenario de entrenamiento, estas observaciones son las que permiten explorar que acciones consiguen obtener las mejores recompensas, dentro de la librería de Unity, las observaciones pueden ser de 7 tipos distintos tipos lógicos de datos vistos en la tabla 2.

Tipo de dato	Descripción
Booleano	Valores con dos posibles estados (encendido, apagado) (arriba, abajo) (izquierda, derecha)
Flotante	Valores de fracción, como distancias que necesitan precisión con los decimales.
Entero	Valores enteros, como cantidad de algún objeto, o número de puntos
Cuaternión	Valor de 4 dimensiones que representa la rotación de un objeto en la plataforma de unity 3d
Vector de dos dimensiones	Vector común de dos dimensiones
vector de tres dimensiones	Vector común de 3 dimensiones
Listado de flotantes	Listado de valores, único grupo de datos que pueden ser usados como observación para los agentes en unity 3d.

Tabla 2. Tipos de dato para las observaciones

Además, los datos más representativos dentro del kit de desarrollo que permiten controlar el proceso en los algoritmos de aprendizaje reforzado están descritos en la tabla 3.

Configuración	Descripción
buffer_size	El número de experiencias recolectadas antes de realimentar el cerebro
epsilon	Determina la rapidez con la que el modelo generado en el entrenamiento evoluciona
lambd	Valor de regulación de aprendizaje
learning_rate	El valor inicial que tienen todos los estados antes de realimentar el sistema
max_steps	La cantidad total de acciones tomadas en el entrenamiento

Tabla 3. Datos de configuración de aprendizaje

El sistema de aprendizaje reforzado en unity *machine learning toolkit* tiene 3 elementos fundamentales, el agente, el cerebro y la academia. Un agente pone a prueba sus acciones en un escenario, entrenando un cerebro según los parámetros de una academia, este cerebro una vez terminado el entrenamiento podrá tomar acciones que maximicen la recompensa en el escenario. En la tabla 4 se observa de manera independiente la función de cada uno de los elementos.

Elemento	Función
Agente	Poseen un grupo de acciones. Durante el entrenamiento reciben recompensas positivas y negativas por cada acción que tomen basados en sus observaciones.

Cerebro	El cerebro es el modelo creado por el entrenamiento que analiza los estados alcanzados por el agente que consiguieron maximizar la recompensa.
Academia	La academia controla los aspectos más técnicos del entrenamiento, como los tiempos de toma de acciones, el número de acciones que debe tomar un agente antes de completar su entrenamiento, entre otros parámetros.

Tabla 4. Elementos fundamentales en el aprendizaje reforzado

Dentro de una academia pueden existir diferentes cerebros siendo entrenados bajo los mismos parámetros, y cada cerebro usa las recompensas obtenidas por uno o más agentes. De esta forma varios agentes pueden construir un único cerebro, beneficiándose de las recompensas que consiga cada agente de manera independiente. En la figura 9 se observa la relación entre los 3 elementos fundamentales para el aprendizaje reforzado.

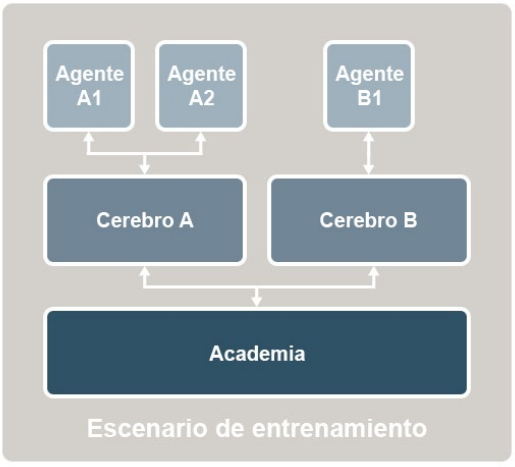

Figura 18. El escenario de aprendizaje. Fuente: Unity 3D

Resultado de la Segunda Fase:

Implementación un escenario de entrenamiento de agentes para el uso de Machine Learning en Unity 3D.

Diseño del escenario de aprendizaje

El escenario propuesto es una matriz rectangular compuesta por un numero finito de posiciones, en cada una de estas posiciones existen dos tipos de dato, uno de ellos es un cubo que representa uno de los posibles lugares donde un agente podría estar ubicado, el otro es una posición inalcanzable para el agente, dentro de la matriz los cubos forman una isla que un agente podrá recorrer con el fin de cumplir un objetivo.

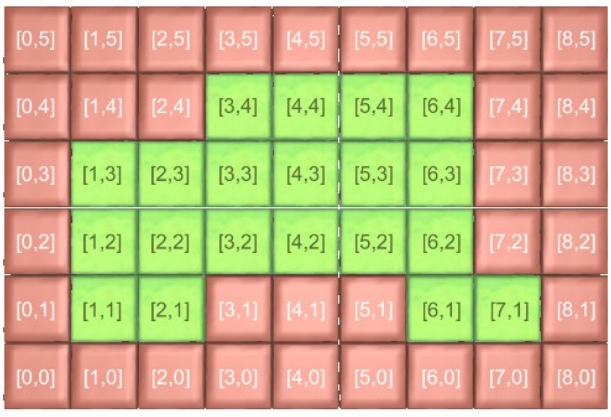

Figura 19. Isla de cubos. Fuente: Autor

Se desarrollo un sistema con la capacidad de reconocer los cubos que el agente puede recorrer y aquellos espacios inalcanzables. Para cualquier isla creada los bordes siempre

estarán catalogados dentro de la matriz de datos como posiciones inalcanzables, las otras posiciones inalcanzables corresponden a los espacios dentro de la isla que se necesitan para conservar la proporción rectangular de la matriz como se observa en la figura 20. El sistema construye una matriz rectangular según los datos, asignando a cada posición un valor en la matriz, que será usada por el agente para evaluar sus acciones en el escenario.

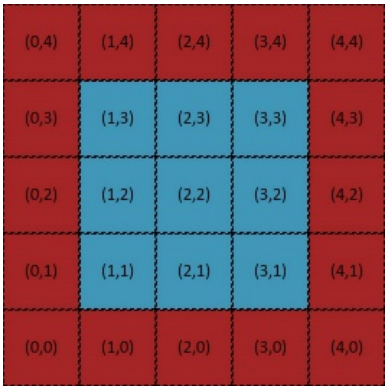

Figura 20. Matriz lógica construida. Fuente: Autor

Cada isla de cubos es creada desde unity, el sistema descrito anteriormente se encarga de construir la matriz de datos que se usará en el entrenamiento, dentro del editor la isla, solo tendrá visibles los cubos propios de la isla ocultando los lugares inalcanzables, pero manteniéndolos referenciados en la matriz de datos.

Figura 21. Isla visual de 16 cubos. Fuente: Autor

Con el fin de simplificar el estudio de la implementación de *machine learning* en unity 3d, se modelo un escenario basado en matrices, estas son ampliamente utilizadas en los videojuegos por la versatilidad en su implementación, sin limitarse a lo propuesto en este escenario. Se puede denotar en cualquier video juego de carreras, donde todo el escenario puede ser simplificado a un matriz que represente la ruta que recorre un agente para completar la carrera como se observa en la figura 22.

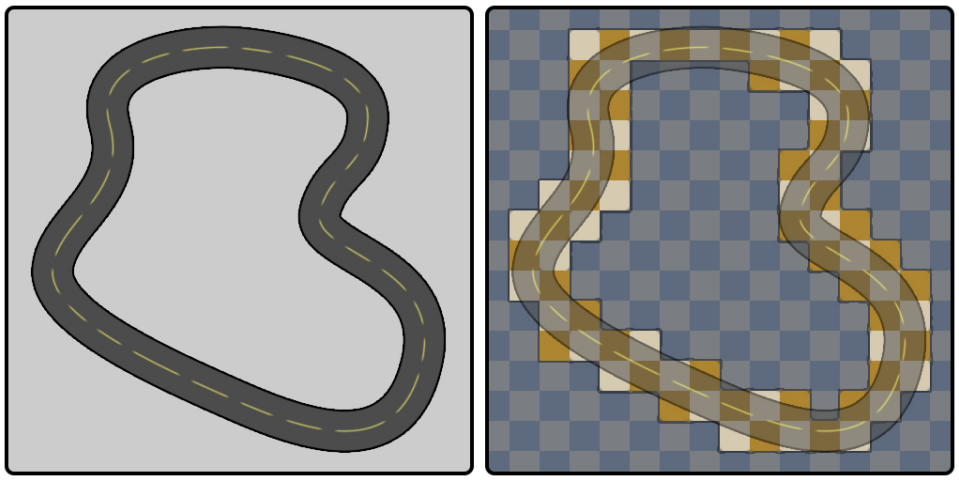

Figura 22. El mapa de un juego de carreas simplificado en una matriz. Fuente: Autor

Otra posible aplicación de las matrices en los videojuegos puede ser en la segmentación de una habitación donde cada posición en la matriz representa un espacio para un agente, como se ve en la figura 23.

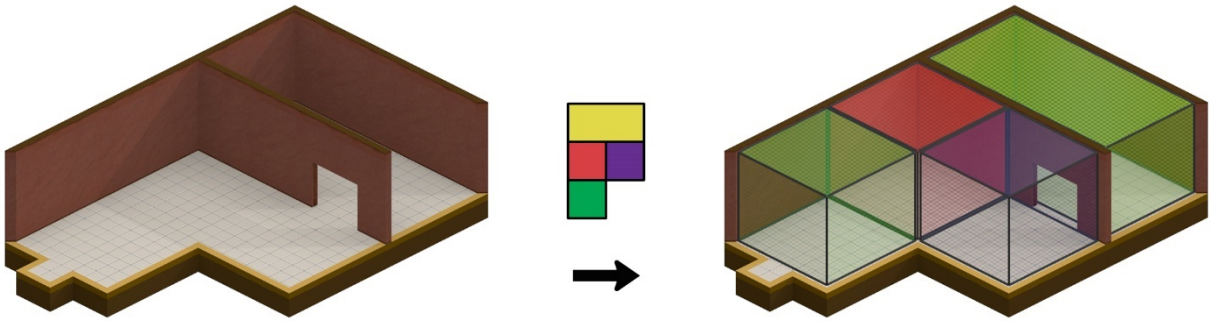

Figura 23. Una habitación vista como una matriz. Fuente: Autor

Determinación del agente y del escenario de aprendizaje

El agente Bob llamado así por el autor, diseñado como un pequeño robot con apariencia amigable es quien tendrá el objetivo de reunir la mayor cantidad de cubos en la isla saltando sobre las distintas posiciones de la matriz. Bob podrá moverse sobre la isla un cubo a la vez en una de las 4 direcciones no diagonales de su orientación. En la figura 24 se observa el modelo renderizado de Bob.

Figura 24. El agente Bob. Fuente: Autor

Las acciones que Bob podrá realizar en el escenario son solo de movimiento, Bob podrá escoger entre: al frente, hacia atrás, izquierda o derecha mientras recorre la isla, desde programación los valores que identifican a cada una de estas acciones están representadas con los dígitos (0,1,2,3,4) siendo 0 la acción de quedarse en la posición actual, en la figura 25 se observa la relación de los valores numéricos con las acciones tomadas en el escenario. Las observaciones de Bob serán todos los cubos existentes en la isla.

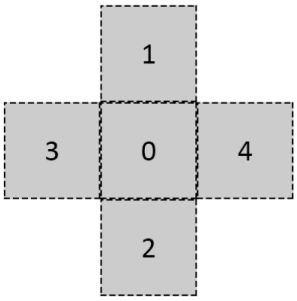

Figura 25. Las acciones de Bob. Fuente: Autor

Definición del objetivo del agente dentro del escenario de aprendizaje

El objetivo de Bob será moverse alrededor del escenario con sus 5 acciones en busca de conseguir todos los cubos sin ser explícitamente programado para eso. Bob debe recorrer cada cubo de la isla repetidas veces hasta que este haga parte de su propiedad, acumulando puntos de propiedad por cada vez que pase sobre un cubo. Durante el entrenamiento se definieron un grupo de estados iniciales que servirán para realimentar en base a recompensas el cerebro de Bob, a continuación, se describen los estados en los que Bob recibirá recompensas en el entrenamiento, ver tabla 5.

Estado	Descripción
Moverse a una posición invalida	Bob recibirá una recompensa negativa cuando intente llegar a un lugar inalcanzable.
Caer sobre un cubo que ya es de su propiedad	Una vez que un cubo está en estado estático, Bob recibirá una recompensa negativa pues en este cubo ya no será posible adquirir más puntos de propiedad
Caer sobre un cubo con algún nivel de propiedad	Bob recibirá una recompensa positiva por agregarle un nuevo punto de propiedad a un cubo que ya tenía al menos un punto de propiedad
Caer sobre un cubo sin ningún nivel de propiedad	Bob recibirá una recompensa positiva por darle un punto de propiedad a un cubo que no tenía ningún punto de propiedad

Ganar la propiedad de un cubo	Una vez que Bob consigue ganar la propiedad de un cubo con la cantidad de puntos definida recibirá una recompensa
Conseguir la totalidad de los cubos	Una vez terminado el escenario, Bob recibirá una recompones positiva por haber cumplido su objetivo
Tiempo en el escenario	Durante todo el entrenamiento Bob recibirá un pequeño coeficiente de recompensa negativa que con el fin de que consiga su objetivo en el menor tiempo posible.

Tabla 5. Estados iniciales en el aprendizaje de Bob

Determinación de los atributos del sistema desde el enfoque del agente desde las técnicas de *Machine Learning*

Cada uno de los datos en la matriz creada por el sistema tendrá una serie de parámetros que el agente usará para realimentar su entrenamiento. Las posiciones inalcanzables tendrán la misma cantidad de parámetros que las posiciones alcanzables, pero mantendrán un valor por defecto que no será cambiado, dado que ningún agente podrá alcanzar ese lugar y solo usará esa información para recibir una recompensa negativa por haber intentado llegar allí. Los parámetros en cada uno de los datos de la matriz son presentados en la siguiente tabla:

Parámetro	Tipo de valor	descripción
Posición x en la matriz de datos	Entero	La posición en la matriz en el eje x

Posición y en la matriz de datos	Entero	La posición en la matriz en el eje y
Cubo congelado	Booleano	Determina si un cubo ya adquirió suficientes puntos de propiedad para estar congelado y es propiedad de una agente
El agente propietario	Cadena de caracteres	Un identificador hexadecimal que determina el id del agente que tomo propiedad del cubo
Número de puntos de propiedad	Entero	La cantidad de puntos de propiedad que tiene el cubo iniciando con 0

Tabla 6. Parámetros para cubo en el escenario de entrenamiento

Entrenamiento del agente a partir de la implementación de las del sistema desde el enfoque del agente

En el escenario propuesto, cada uno de los cubos en la isla podrá pertenecer a un agente, la forma en la que un agente puede tomar propiedad sobre un cubo es pasando sobre él una determinada cantidad de veces, cada vez que un agente pase sobre un cubo, este acumulará un punto de propiedad, una vez que consiga los puntos de propiedad necesarios el cubo pasara a un estado estático que no podrá ser cambiado hasta iniciar un nuevo entrenamiento, en la figura 26, se observan 5 estados de un cubo, inicialmente sin ningún punto de propiedad y luego de acumular 4 puntos de propiedad pasa a un estado en el cual las propiedades se mantienen fijas. una vez que todos los cubos en la isla sean

propiedad del agente se dará por terminado el entrenamiento, el objetivo del agente entonces será reunir la propiedad de todos los cubos en la isla en el menor tiempo posible.

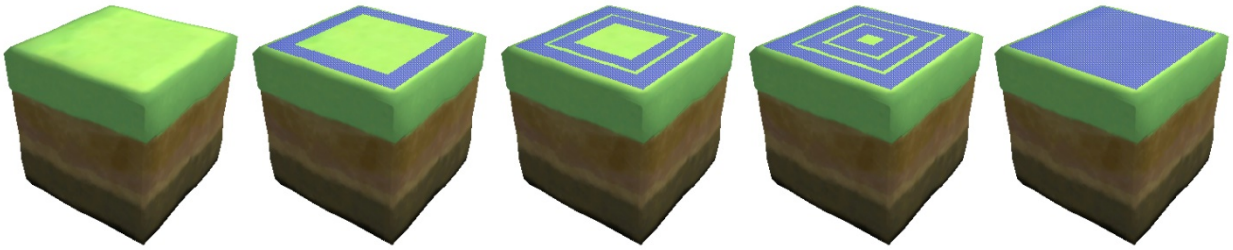

Figura 26. Estados de un cubo con diferentes niveles de propiedad. Fuente: Autor

Resultado de la Tercera Fase:

Análisis de los resultados obtenidos sobre la implementación del *Machine Learning* en base al escenario de entrenamiento.

El aprendizaje de Bob

Una vez que fueron definidas los valores de recompensa para algunos de los estados de Bob definidos en la tabla 5, se realizaron entrenamientos con las siguientes características según la configuración requerida para los entrenamientos en unity 3d Definidos en la tabla 3, asignando los valores de recompones definidos en la tabla 7 a continuación.

Configuración	Descripción
buffer_size	10240
epsilon	0.2
lambd	0.95
learning_rate	0.0003
max_steps	50000

Tabla 7. Datos utilizados para el aprendizaje

En una primera etapa del entrenamiento de Bob de asignaron valores en distintas pruebas para encontrar un balance entre las recompensas positivas y negativa que permitieran una curva de aprendizaje en el escenario, se realizaron 4 pruebas en total donde se iban ajustando los valores hasta encontrar un patrón de comportamiento que evidenciara el cumplimiento del objetivo del agente. Cada prueba tomo aproximadamente 3 horas, en cada prueba 6 agentes realizaban acciones según los parámetros definidos en la tabla 8. distintos escenarios fueron usados para intentar evaluar el comportamiento en diferentes situaciones, la figura 27 se observan algunos de los escenarios creados para la primera etapa del entrenamiento.

Estado	Prueba 1	Prueba 2	Prueba 3	Prueba 4
Moverse a una posición invalida	-1	-1	-2	-1
Caer sobre un cubo que ya es de su propiedad	-0.5	-1	-3	-1

Caer sobre un cubo con algún nivel de propiedad	0.5	1	2	1
Caer sobre un cubo sin ningún nivel de propiedad	1	5	3	2
Ganar la propiedad de un cubo	1	2	2	2
Conseguir la totalidad de los cubos	5	2	10	5
Tiempo en el escenario	-0.001	-0.001	-0.001	-0.001

Tabla 8. Recompensas en el aprendizaje de Bob

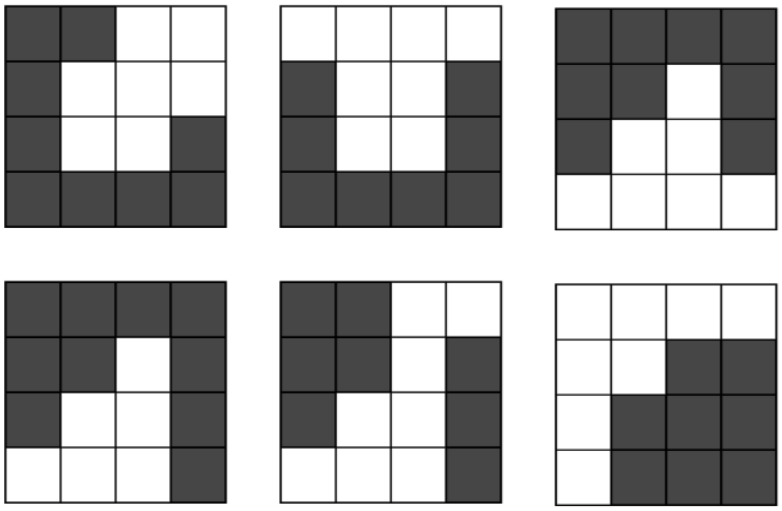

Figura 27. Escenarios para entrenamiento. Fuente: Autor

En la cuarta prueba realizada se evidencio un comportamiento del agente en el escenario suficientemente cercano al cumplimiento del objetivo, siendo estos los valores elegidos para las pruebas extensas de entrenamiento para Bob, los resultados de las pruebas están descritos en la tabla 9.

Prueba	Resultado
Prueba 1	Se movía libremente por el escenario sin un objetivo aparente
Prueba 2	Prefirió quedarse quieto en vez de moverse en busca de cubos
Prueba 3	Repetía constantemente los cubos ignorando aquellos sin ningún nivel de propiedad
Prueba 4	Consiguió moverse en gran parte del escenario recolectando un número considerable de cubos

Tabla 9. Tabla de resultados sobre pruebas

Los entrenamientos que se realizaron posteriormente fueron realizados en matrices de 11x11 contando aquellos valores dentro de la matriz de datos que no corresponden a cubos validos como posiciones en el espacio, en la figura 28 se observan algunos de los escenarios creados para el entrenamiento final de Bob en la isla matricial.

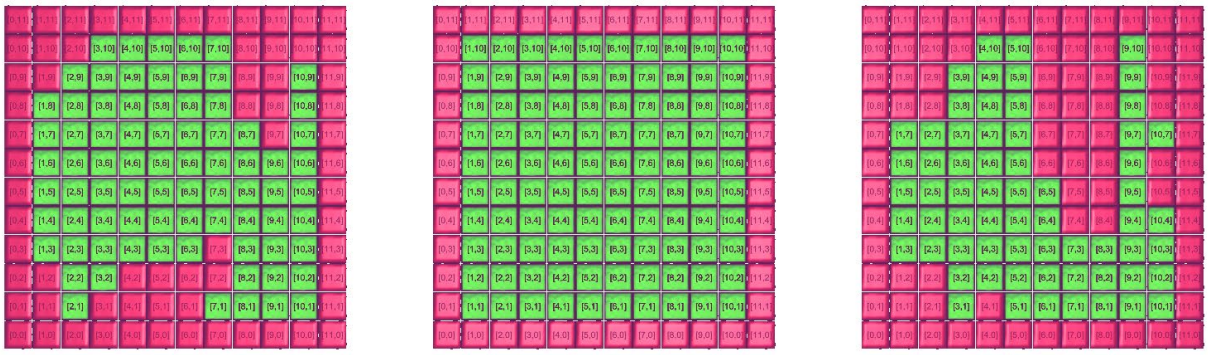

Figura 28. Pruebas generadas para la construcción en diferentes sistemas. Fuente: Autor

La librería de *machine learning*

La librería resultante de esta investigación tiene una clase extra que se vincula con las clases propias de la librería de machine learning toolkit de Unity, además de 2 objetos programables con los cuales se podrá llevar un control más cercano de los parámetros dinámicos que dentro del entrenamiento es necesario cambiar con regularidad en busca del balance de la curva de aprendizaje, el resumen de los componentes más relevantes de la librería está tabulados en la tabla 10.

Clases en programación	Objetos programables
Agente	Recompensas
Academia	Sonidos
Acciones de agente	Cerebro

Tabla 10. Resumen librería

El grupo de recompensas permitirá que múltiples agentes dependiendo de su escenario puedan ser entrenados simultáneamente con distintos grupos de recompensa, permitiendo así mayor flexibilidad al momento de llevar el control en el aprendizaje, en la figura 29 se observa una ventana dentro del software de unity con el objeto programable de recompensas que fue utilizado en el entrenamiento de Bob.

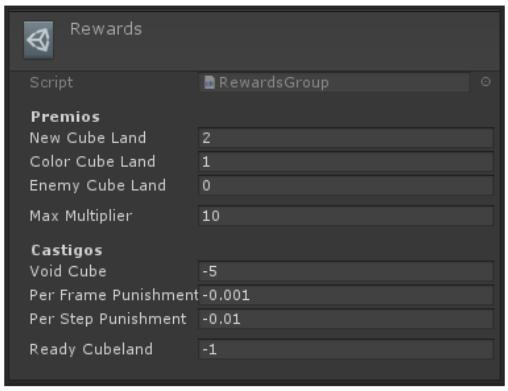

Figura 29. Objeto programable de recompensas. Fuente: Autor

El archivo que almacena todos los recursos de la librería puede ser tratado como cualquier otro archivo, en la figura 31 se observa el icono del archivo sobre Windows luego de haber sido exportado para futuros usos en unity 3d.

Figura 30. Archivo de librería en Windows. Fuente: Autor

Conclusiones

A partir de la revisión bibliográfica realizada, se conocieron aspectos relacionados con los algoritmos de *Machine Learning,* los cuales son elegidos según el contexto de los datos y las necesidades de la situación, no existió un algoritmo o técnica de *Machine Learning* mejor que otra u óptima de manera global, sino que, por el contrario, los usos y métodos de aprendizaje tuvieron mejores o peores resultados dependiendo de la situación o contexto donde se implementen.

La implementación de un escenario de entrenamiento de agentes para el uso de *Machine Learning* en Unity 3D, debe atravesar diferentes etapas que inician en el diseño del ambiente de aprendizaje, la determinación del agente y la definición del objetivo del agente dentro del escenario, para el cual se evidenció que en la práctica la utilización de recompensa negativa (castigo) es muy usual para conseguir una curva de aprendizaje en el agente, sin embargo, este grupo de reglas (recompensas positivas o negativas) deben estar completamente balanceadas para no perder de vista el objetivo real del agente.

De acuerdo con los resultados de la implementación de *Machine Learning* en base al escenario diseñado, se pudo desarrollar una librería para la programación de videojuegos en Unity 3D empleando técnicas de *Machine Learning*, la cual permitirá reducir al menos 2 semanas de desarrollo, que corresponden a la programación base que debe estar integrada con el sistema de *machine learning toolkit*.

Trabajo a futuro

En el futuro se espera poder categorizar las funciones de la librería de acuerdo con las necesidades propias de futuros desarrollos, con un enfoque al estudio de nuevas mecánicas y métodos para retar al jugador, estudiando a profundidad la visión del agente dentro de su ambiente digital con el fin de tener un control más amplio sobre la curva de aprendizaje, reduciendo los tiempos que tomo la primera etapa de ajuste en el entrenamiento de Bob.

Referencias

Dickey, M. (2006). Game Design Narrative for Learning. *Appropriating Adventure Game Design*, 20.

Fiszelew, A. a.-M. (2016). GENERACIÓN AUTOMÁTICA DE REDES NEURONALES CON AJUSTE DE PARÁMETROS BASADO EN ALGORITMOS GENÉTICOS. *Instituto Tecnológico de Buenos Aires*, 21.

Fonseca Reyna, Y. C. (2015). A Reinforcement Learning Approach for Scheduling Problems. *Investigación Operacional*, 36.

Fonseca-Reyna, Y. C.-J. (2017). Q-Learning Algorithm Performance for M-Machine, N-Jobs Flow Shop Scheduling Problems to Minimize Makespan. *Investigación Operacional*, 290.

FUNGE, I. M. (2009). Artificial intelligence for games. *Morgan Kaufmann*, 800.

FUNGE, I. M. (2009). *ARTIFICIAL INTELLIGENCEFOR GAMES.* Burlington: Elsevier Inc.

Herrera, F. (2015). Un estudio emp´ırico preliminar sobre los tests. *Diseños experimentales*, 10.

J.R, Q. (1986). Induction of Decision Trees. *Kluwer Academic*, 26.

Jennifer Hernández Bécares, L. C. (2016). An approach to automated videogame beta testing. *Entertainment Computing*, 92.

John Schulman, F. W. (2017). Proximal Policy Optimization Algorithms. *OpenAI*, 12.

Leo Galway, D. C. (2009). Machine learning in digital games: a survey. *Springer Science*, 2.

Martínez, G. (2017). UNA VISION GLOBAL DEL APRENDIZAJE AUTOMATICO. *APRENDIZAJE AUTOMATICO*, 14.

Michael Bowling, J. F. (2006). Machine learning and games. *Springer Science*, 5.

Mitchell. (1978). Version Spaces: An Approach to Concept. *Stanford University*, 200.

Osindero, G. E. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 16.

Pavón, R. (2010). INTELIGENCIA ARTIFICIAL. *Resumen de Tesis*, 4.

Sohn, I. (2018). A robust complex network generation method based on neural. *Division of Electronics and Electrical Engineering*, 9.

Storkey, C. C. (2015). Teaching Deep Convolutional Neural Networks to Play Go. *University of Edinburgh*, 9.

Y., Y. X. (1998). Toward Designing Artificial Neural Networks by Evolution. *Applied Mathematics and Computation,*, 91.

Zhu, P. H. (2011). Strategy research on the performance optimization of 3D mobile game development based on Unity. *Oxbridge College*, 7.