Trabajo Fin de Grado

HLD: Explorando distintas estrategias de IA para juegos de cartas

HLD: Exploring different AI strategies for card games

Autor

Álvaro Pérez Martínez

Director

Carlos Bobed Lisbona

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2024

HLD: Explorando distintas estrategias de IA para juegos de cartas

Álvaro Pérez Martínez

19 de julio de 2024

Agradecimientos

A Carlos Bobed, por el apoyo y los consejos
que me ha ofrecido durante todo el proyecto.
A Claudia, por confiar siempre en mí.

Resumen

En los últimos años, la industria de los videojuegos ha experimentado un crecimiento exponencial, convirtiéndose en una de las principales formas de entretenimiento digital. Paralelamente, los juegos de mesa, a pesar de ser una forma de ocio tradicional, han sabido adaptarse y evolucionar en este contexto digital. La convergencia entre ambas formas de entretenimiento ha dado lugar a una tendencia creciente: la digitalización de los juegos de mesa.

Dado el contexto nombrado anteriormente, en este proyecto se ha planteado la virtualización del juego Happy Little Dinosaurs, un juego de mesa competitivo en el que los jugadores deben tomar decisiones para lograr la supervivencia de sus personajes dinosaurio.

Dicha virtualización ha requerido, por un lado, la implementación de diversas escenas o interfaces, prestando especial atención a la información sobre las cartas jugadas en cada ronda.
Por otro lado el desarrollo del videojuego, se ha compuesto de 2 fases para el desarrollo de bots que usan Inteligencia Artificial (IA):

- En la primera fase del proyecto se diseñaron e implementaron la lógica y el motor del juego, desarrollando una interfaz de usuario intuitiva, ayudado por el entorno *Unity*, herramienta muy usada para crear videojuegos.

- En la segunda fase, se utilizaron técnicas de IA aplicada a Videojuegos, que permitieran al jugador desafiar a la máquina. Para ello se evaluaron 3 algoritmos diferentes: algoritmo MinMax, el cual resultó complicado de implementar al ser un juego de cartas de más dos jugadores y con información imperfecta u oculta; el algoritmo de Monte Carlo, el cual resultó útil ya que era capaz de determinar la probabilidad de las cartas para ganar la ronda; y el algoritmo de Aprendizaje por Refuerzo, el cual resultó muy prometedor, pero quedó finalmente fuera del alcance del proyecto.

Este trabajo ha permitido experimentar la realización de un proyecto personal de escala media, con especial énfasis en la organización del tiempo, los esfuerzos y la virtualización de un videojuego completo.

Índice general

1. Introducción

En los últimos años, la industria de los videojuegos ha experimentado un crecimiento muy significativo, el cual paralelamente ha ido acompañado de un impulso del mercado de juegos de mesa. Esto ha hecho que surgiera una simbiosis de los dos sectores y se virtualizaran grandes títulos de los juegos de mesa como puedan ser *Monopoly, Risk, Uno, Catan....* Fue este preciso motivo lo que propició la decisión de elaborar en este trabajo la virtualización de un juego de mesa.

1.1. Análisis de opciones

En un primer momento, esta búsqueda planteó 4 posibles opciones de virtualización de juegos de mesa:

- Munchkin: es un juego de cartas por turnos cuyo objetivo es alcanzar el nivel 10, para lo cual los jugadores deben enfrentarse a diferentes monstruos, pudiendo o no crear alianzas por el camino.
  Este juego presentaba muchas dificultades para su virtualización debido al gran número de cartas y efectos que lo componían.

- Código Secreto: es un juego de deducción en el que los jugadore se dividen en dos equipos, cada uno con un jefe de espías, quien debe aportar pistas a sus compañeros para que sean capaces de localizar todas las palabras clave propias de su equipo.
  La dificultad principal de este juego radicaba en la implementación de la IA para que actuase como jefe de espías, capaz de agrupar las palabras del equipo y ofrecer pistas óptimas. Además, se requiere de otro algoritmo que, a partir de una pista del jefe de espías pueda encontrar las palabras a las que se refiere. Ambas tareas eran demasiado complejas en ese momento.

- Here to Slay: es un juego de cartas estratégico por turnos, en el que para ganar hay que derrotar 3 monstros, o reunir los 6 tipos de héroes diferentes. Sin embargo, los demás jugadores pueden dificultar esta tarea al interferir en la eliminación de monstruos, robar héroes o forzar el descarte de cartas.
  Aunque este es el juego más simple de los propuestos, y cuenta con las funcionalidades básicas necesarias, contaba con una cantidad de cartas demasiado amplia.

- Happy Little Dinosaurs (HLD): es un juego de cartas por turnos con toques de humor, basado en la supervivencia de sus personajes dinosaurio ante diversos peligros. Para ello el juego cuenta con 3 tipos de cartas (Cartas de Poder, Cartas Instantáneas y Cartas Desastre), lo que hace que la complejidad de los algoritmos de la IA recaiga tanto en la elección de las cartas, como en los posibles efectos que puedan poseer.

Es por ello que, tras una exhaustiva búsqueda inicial de información, se optó finalmente por virtualizar el juego Happy Little Dinosaurs.

1.2. Objetivos

Este proyecto tiene 2 objetivos principales:

- Virtualizar el juego Happy Little Dinosaurs (HLD) en formato digital, implementado diversas escenas o interfaces para mostrar el desarrollo del juego.

- Implementar diferentes niveles de bots con IA con diferentes niveles de dificultades, barajando los algoritmos y técnicas:
  - MinMax
  - Monte Carlo
  - Aprendizaje por Refuerzo

1.3. Tecnología utilizada

Para este proyecto se empleó el entorno de desarrollo *Unity* para realizar las escenas y la vinculación de sus elementos con el motor del juego. Para desarrollar el motor se empleó el editor de código *Visual studio Code*, en el cual se utilizaron 2 lenguajes de programación diferentes:

- C#: el cual se empeló tanto para desarrollar el motor, como para el apartado gráfico y la implementación de la IA, es decir, el 90 % del proyecto fue escrito con este lenguaje.

- Python: se utilizó para crear un nivel de dificultad de la Inteligencia Artificial, ayudado de la librería RLCard.

Para la obtención de las imágenes de las cartas y del tablero se realizó ingeniería inversa, a través de los propios archivos del videojuego *TableTop*.

1.4. Estructura de la memoria

A continuación, en la Sección 2 se describen las reglas básicas del juego, y definen algunos conceptos básicos para facilitar la comprensión del mismo.

Posteriormente, en la Sección 3 se detallan los requisitos funcionales y no funcionales del proyecto, así como las diferentes decisiones de diseño e implementación del juego.

Tras ello, en la Sección 4, se detalla la información obtenida tras una búsqueda de información de los algoritmos de IA, así como las conclusiones obtenidas.

Por último, la Sección 5 recoge las conclusiones finales sobre el trabajo, así como una visión de posibles futuras líneas de implementación.

2. Happy Little Dinosaurs (HLD)

Happy Little Dinosaurs es un juego de mesa competitivo, basado en la supervivencia de sus personajes dinosaurio, creado por la compañía Unstable Games.

En este juego, los jugadores deben tomar decisiones sobre cómo usar sus cartas y habilidades, con el objetivo de ser el último dinosaurio con vida o llegar a obtener 50 puntos antes que los demás, mientras intentan evitar Desastres y peligros: el mundo de los dinosaurios está lleno de pozos de alquitrán, ardillas asesinas y cortes de pelo realmente malos, sin mencionar la inminente amenaza de la extinción.

2.1. Dinosaurios

Los jugadores tendrán que elegir entre los 4 dinosaurios disponibles, los cuales se muestran a continuación en la *Figura* 1:

Figura 1: Dinosaurios

Como se muestra en la *Figura* 2, cada dinosaurio tendrá sus rasgos descritos en la esquina inferior izquierda de su tablero, y servirán para dar ventajas o desventajas al enfrentarse a ciertos tipos de Cartas Desastre. Las ventajas y desventajas de los rasgos de dinosaurio se suman o restan al valor de las Cartas de Poder en cada ronda.
Cada jugador dispone, en su propio tablero, de 3 zonas diferentes:

- Zona izquierda: su personaje, junto con la habilidad que posee.
- Zona central: la ruta de escape, por la cual se avanzará a lo largo de los turnos.
- Zona derecha: el Área de Desastre, donde se sitúan las Cartas Desastre que se obtengan en cada ronda perdida.

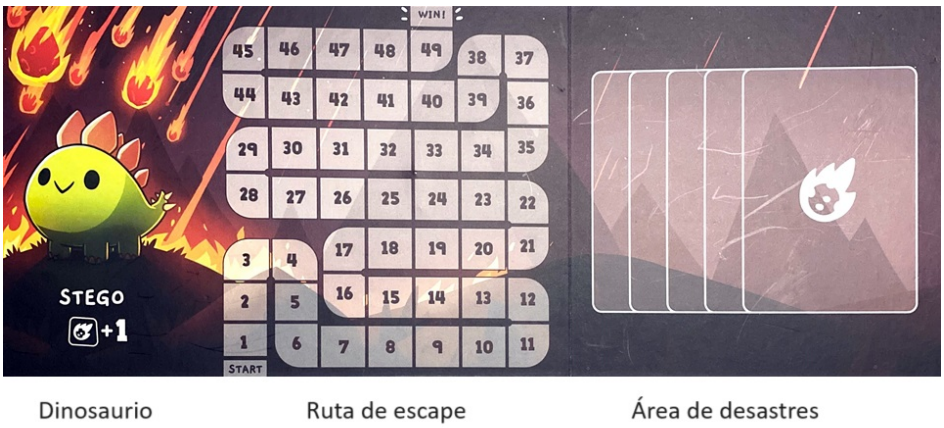

Figura 2: Tablero del jugador

2.2. Tipos de cartas

Este juego posee 3 tipos de cartas diferentes: las Cartas Desastre, comunes a todos los jugadores, y que tendrán que pelear para no tener que llevarse a su Área de Desastres; las Cartas de Poder, que son las más comunes y es necesario tener al menos una en la mano para poder jugar la ronda; y las Cartas Instántaneas, que añaden, quitan o intercambian las puntuaciones de los jugadores.

Cartas Desastre

A continuación en la *Figura* 3 se puede ver que existen 4 tipos de Cartas Desastre, cada una de las cuales se indica con un color:

- Rojo: Amenaza física.
- Azul: Amenaza emocional.
- Verde: Amenaza natural.
- Negro: Meteorito.

En caso de que un jugador recolecte tres Cartas Desastre del mismo tipo, o tres tipos diferentes de Cartas Desastre, quedará eliminado del juego y deberá descartar inmediatamente su mano.

Las Cartas Meteorito son un tipo de Cartas Desastre especiales, que actúan como cualquiera de los tres tipos de Cartas Desastre.

Cada vez que un jugador añada una Carta Desastre a su Área de Desastres, deberá mover su dinosaurio hacia adelante una casilla en su Ruta de Escape al final de cada ronda.

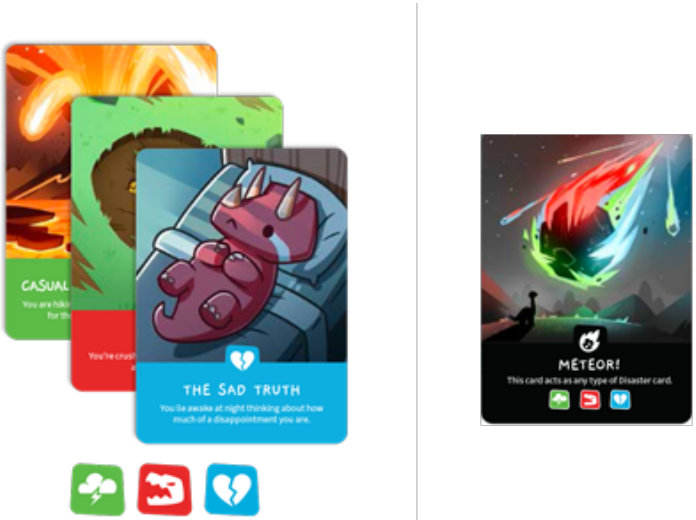

Figura 3: Ejemplos de Cartas Desastre y sus simbolos

Cartas de Poder

Como se puede ver en la *Figura* 4, las Cartas de Poder tienen un valor de entre 0 y 9, que se utilizan para anotar durante la ronda. Algunas de ellas sirven exclusivamente para obtener ese valor, mientras que otras tienen, además, un efecto de carta, que pueden permitir a los jugadores robar una carta, descartarse una carta de la mano, intercambiar Cartas de Poder con otro jugador o incluso alterar las reglas de puntuación de esa ronda.

A menos que se especifique lo contrario, los jugadores pueden usar el efecto de su Cartas de Poder después de revelarla, pero antes de que se produzca la puntuación en la ronda. En el caso de que varios jugadores jueguen Cartas de Poder con efectos en la misma ronda, lo usará primero el jugador con el valor más bajo de su carta.
Los efectos de las Cartas de Poder solo se pueden usar una vez por ronda.

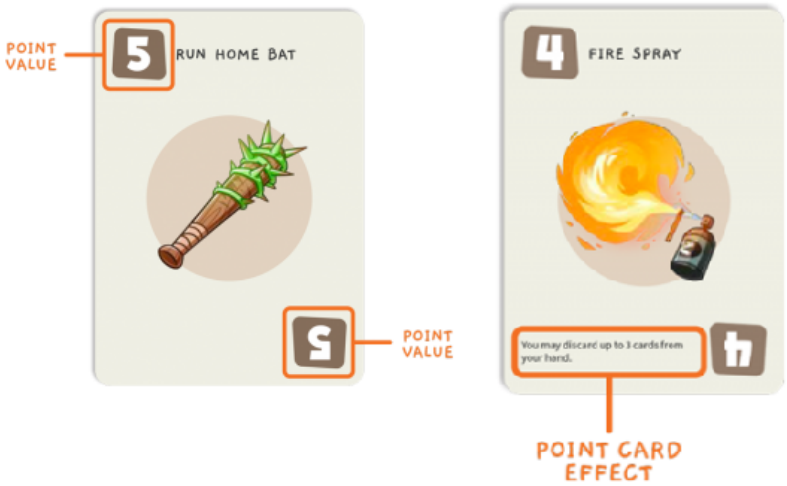

Figura 4: Cartas de Poder

Cartas Intántaneas

Este tipo de cartas se encontrarán en la mesa, en el mazo principal junto a las Cartas de Poder, ya que ambas podrán estar en la mano de los jugadores.

Las Cartas Instántaneas (*Figura* 5) son muy versátiles ya que, dependiendo de lo que indique la propia carta, pueden tener varios efectos posibles, como pueden ser desviar desastres, modificar puntos, afectar a las cartas que hay en juego en esa ronda o cambiar las reglas del juego. Pueden ser usadas en cualquier momento del juego, incluso fuera del turno. Su efecto es inmediato, y una vez jugadas se descartan y no vuelven al juego. Además, pueden usarse como respuesta a otras acciones que haga otro jugador que puedan afectarte negativamente.

Figura 5: Cartas Instantáneas

2.3. Reglas del juego

En el juego habrá 2 mazos de cartas: uno que conste de las Cartas Desastre y otro que esté formado aleatoriamente por el resto de cartas (Cartas de Poder y Cartas Instantáneas).

El juego comienza eligiendo cada jugador a un dinosaurio que le represente, tras lo cual recibe en su mano 5 cartas del mazo aleatorio. Cada ronda consta de 2 fases, tras las cuales se hace un recuento de la puntuación:

- En la primera fase de la ronda los jugadores se enfrentarán a un Peligro, es decir, a una carta que se extraiga de forma común para todos los jugadores del mazo de Cartas Desastre, y que se colocará boca arriba y de forma central en el tablero. En ese momento, cada uno de los jugadores deberá enfrentarse a ese peligro empleando una de sus Cartas de Poder en mano y colocándola boca abajo en el tablero, con la esperanza de acumular puntos y evitar así el impacto de los Peligros. En caso de que un jugador no tenga ninguna Carta de Poder en su mano cuando deba usarla, es decir, en los inicios de ronda o en las rondas empate, se descartará todas las cartas de su mano y robará 5 cartas nuevas (si fuera necesario, se repetirá el proceso hasta que tenga una Carta de Poder que pueda utilizar). En el momento en el que todos los jugadores hayan colocado

una Carta de Poder boca abajo, se revelarán todas las cartas al mismo tiempo, terminando así la primera fase.

- En la segunda fase de la ronda, se jugarán los diferentes efectos, en caso de alguna de las cartas los contenga. Seguidamente, se dará la posibilidad de jugar tantas cartas de hechizos como quieran los jugadores.

- Tras acabar la segunda fase se procede al recuento de puntos: la puntuación de cada jugador es igual al valor de la Cartas de Poder que ha utilizado en la ronda, pero esa puntuación puede aumentar o disminuir según las bonificaciones de los rasgos de su personaje dinosaurio, los efectos de las Cartas de Poder o los efectos de las cartas Instantáneas, lo cual se explica más detalladamente en secciones posteriores. El jugador con la puntuación más alta en la ronda acumula el número de poder que posea su carta y debe mover su dinosaurio a lo largo de la ruta de escape en su tablero de jugador en consecuencia. Por el contrario, el jugador con la puntuación más baja añadirá la Carta Desastre boca arriba a su Área de Desastres, y podrá de forma opcional descartar una carta de su mano.

Por último, antes de comenzar la siguiente ronda, volviendo a iniciar la primera fase, cada jugador deberá robar cartas del mazo principal hasta tener 5 cartas en su mano, y se limpiará el tablero moviendo todas las Cartas de Poder e Instantáneas que se hayan empleado a la pila de descarte.

Si en algún momento no quedan cartas en el mazo principal, se barajará la pila de descartes y se colocará boca abajo para formar un nuevo mazo principal.

2.4. Empate

En el caso de que en una ronda varios jugadores tengan la misma puntuación al usar sus Cartas de Poder, teniendo en cuenta las penalizaciones de las Cartas Desastre, las características de los Rasgos de Dinosaurio de cada personaje, y los efectos de las cartas Instantáneas, pueden darse varios casos:

- Si varios jugadores empatan en la puntuación más alta en una ronda, ambos acumulan el mismo número de puntos y mueven su dinosaurio hacia adelante en su Ruta de Escape las casillas correspondientes a ese valor.

- Si varios jugadores empatan en la puntuación más baja en una ronda, se produce una Muerte Súbita, por lo que cada jugador empatado deberá jugar otra de sus Cartas de Poder boca abajo de su mano y voltearlas simultáneamente.
  En este momento, el jugador con el valor de poder más bajo agregará la Cartas Desastre a su Área de Desastres. Durante la Muerte Súbita, los Rasgos de Dinosaurio, los efectos de las Cartas de Poder y las Carta Instantáneas no se aplican, y ningún jugador involucrado en Muerte Súbita acumulará puntos para la ronda, incluso si su nueva tarjeta de poder excede la puntuación del jugador con mayor puntuación en la ronda.
  Si se produce un empate durante la Muerte Súbita, se repetirá el proceso hasta que se rompa el empate, y si en algún momento uno de los jugadores involucrados

en Muerte Súbita no tiene una Carta de Poder en su mano para jugar, ese jugador perderá la ronda y deberá agregar la Carta Desastre a su Área de Desastres. En caso de que más de un jugador se queden sin Cartas de Poder en su mano durante la Muerte Súbita, la Carta Desastre se colocará boca arriba al final del mazo de Cartas Desastre y la ronda se dará por finalizada.

- Si, en caso de empate en cartas más altas o más bajas, se utiliza una Carta Instantánea que invierte las puntuaciones, esto afectará a las puntuaciones de todos los jugadores involucrados, es decir, si por ejemplo dos jugadores están empatados en la puntuación más baja de la ronda con 3 puntos cada uno y la puntuación más alta en esa ronda es de 9 puntos, ambos jugadores originalmente empatados en la puntuación más baja reciben una puntuación de 9 puntos para la ronda, y el jugador que originalmente tenía una puntuación de 9 puntos ahora recibe una puntuación de 3 puntos.

- Si todos los jugadores empatan en una ronda, la Carta Desastre se colocará boca arriba al fondo del mazo de Desastres, y la ronda se dará por finalizada sin que ningún jugador acumule puntos esa ronda.

2.5. Cómo ganar

Las dos opciones posibles para coronarse cono ganador del juego serán, o bien siendo el primer Dinosaurio en alcanzar los 50 puntos en tu Ruta de Escape para así completarla, o bien siendo el último dinosaurio que queda en el juego con vida, evitando obtener Cartas Desastre durante las rondas.

3. Diseño e implementación

En este apartado se detallará el proceso de diseño de la implementación, explicando los requisitos funcionales y no funcionales, y se explicará el método empleado para la virtualización del juego, acompañado de forma visual con diagramas de secuencia para mostrar como interactúan las clases del proyecto.

Esta es una fase crucial del desarrollo, donde las especificaciones teóricas se convierten en un producto funcional.

3.1. Requisitos funcionales y no funcionales

El sistema debe ser capaz de cumplir los siguientes requisitos funcionales pertenecientes a la *Tabla* 1:

RF-Número	**Requisito Funcional**
RF-01	El sistema debe poder permitir robar cartas del mazo a los jugadores.
RF-02	El usuario debe poder ver las puntuaciones y las Cartas Desastre de cada jugador.
RF-03	El sistema debe permitir a los jugadores robar cartas a otros jugadores.
RF-04	El sistema debe permitir jugar una carta de poder a los jugadores en cada ronda.
RF-05	El sistema debe permitir la ejecución de los diferentes efectos de las cartas jugadas.
RF-06	El usuario tiene que poder jugar Cartas Instantáneas contra cualquier jugador.
RF-07	El usuario tiene que poder descartar una carta si se ha perdido la ronda.
RF-08	El sistema debe permitir añadir una Carta Desastre si se ha perdido la ronda y no ha sido descartada.
RF-09	El usuario tiene que poder escoger un nivel de dificultad de la IA.
RF-10	El usuario tiene que poder consultar sus diferentes acciones.
RF-11	El sistema tiene que llevar la cuenta de los puntos de cada jugador.
RF-12	El sistema tiene que identificar cuando finalizar la partida.
RF-13	El sistema debe permitir jugar a un jugador humano contra 3 bots.

Tabla 1: Requisitos funcionales

Además, el sistema también debe cumplir los siguientes requisitos no funcionales de la *Tabla* 2:

RNF-Número	Requisito No Funcional
RNF-01	La interfaz de usuario debe ser fácil de comprender y usar.
RNF-02	El sistema debe garantizar una experiencia de juego fluido sin retrasos.
RNF-03	El sistema debe funcionar y mostrarse correctamente independientemente de la resolución de pantalla.
RNF-04	El sistema debe permitir jugar sin conexión a internet.
RNF-05	El sistema debe permitir jugar en computadores con sistema operativo Windows.
RNF-06	El usuario tiene que poder regular el volumen de la música.
RNF-07	El usuario tiene que poder elegir el tamaño de pantalla.

Tabla 2: Requisitos no funcionales

3.2. Escenas del juego

La primera pantalla (*Figura* 6) que se mostrará será la de selección del personaje, la cual solo aparecerá al iniciar el juego. Será aquí cuando se muestren las opciones de dinosaurios posibles con los que se puede jugar, así como las diferentes opciones y ajustes del propio juego:

- Dificultad de los bots.
- Resolución de pantalla.
- Pantalla completa.
- Volumen de la música.

Figura 6: Escena de selección del personaje

Tal y como se muestra en la *Figura* 7, al seleccionar un dinosaurio se mostrará la segunda pantalla, la escena puntuaciones, la cual se verá regularmente, ya que se mostrará al final de cada ronda.

Figura 7: Diagrama de escena 1

En la *Figura* 8 de puntuaciones se puede ver, como su nombre indica, la puntuación actual del jugador y sus cartas desastre, y permite navegar entre las pantallas de resultados del resto de jugadores pulsando los botones en forma de flechas.

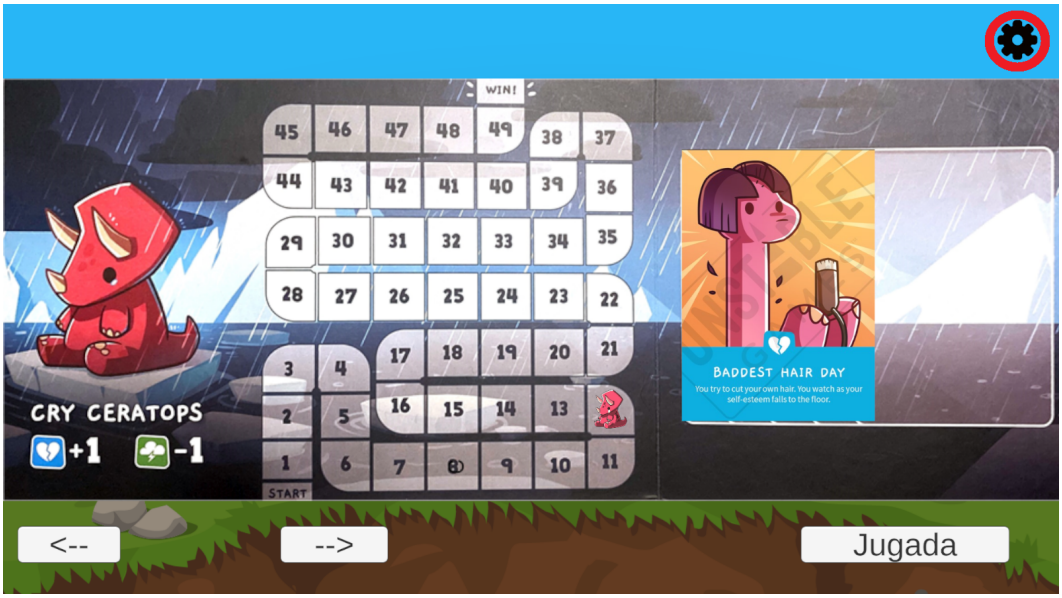

Figura 8: Escena de puntuaciones

Una vez en esta escena, existen 3 opciones posibles, como se muestra en la *Figura* 9:

- Seleccionar el botón con forma de engranaje negro situado arriba a la derecha, lo cual nos lleva a la escena de opciones.
- Seleccionar el botón *Jugada*, lo cual nos lleva a la escena jugar ronda.
- Que, sin pulsar ningún botón, esta escena dé paso a la escena final (escena ganadora o perdedora) para mostrarnos las puntuaciones.

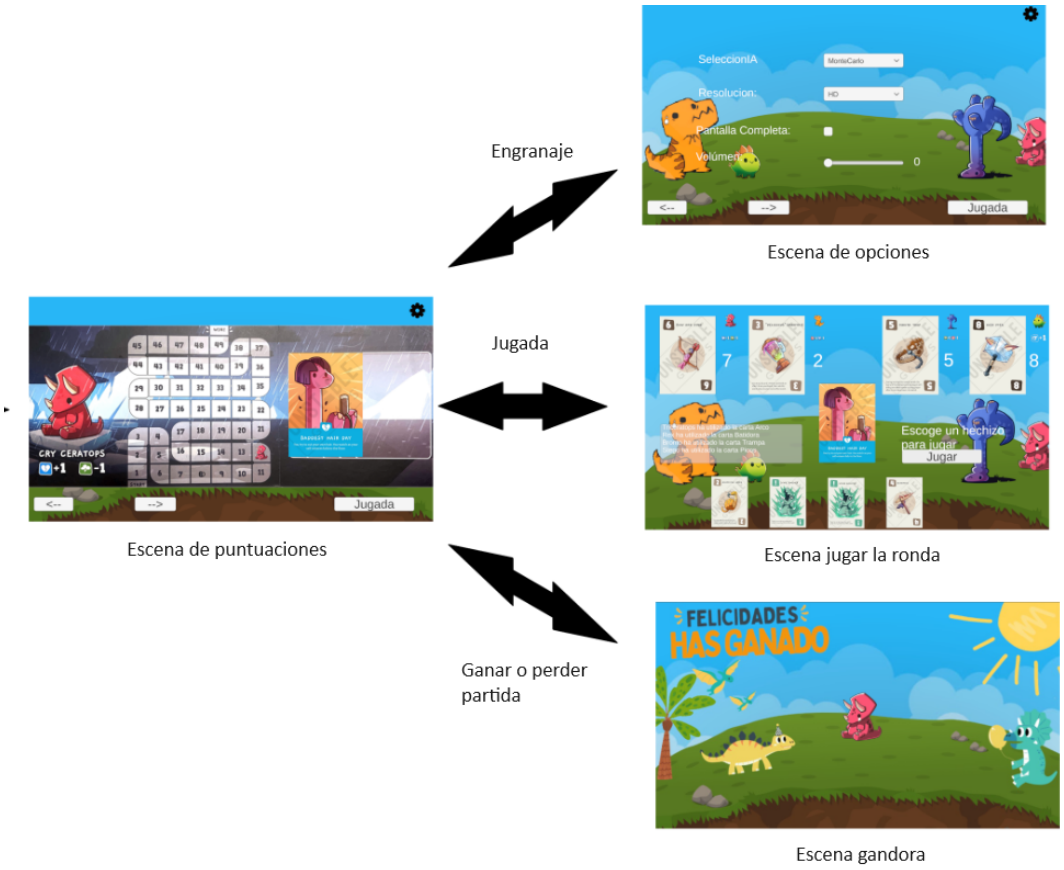

Figura 9: Diagrama de escena 2

A continuación, en la *Figura* 10 se muestra la primera de las opciones nombradas anteriormente, la cual trata de que al seleccionar el botón con forma de engranaje negro situado en la parte superior derecha de la pantalla si en algún momento se desea ver o modificar los ajustes elegidos, lo cual da paso a la escena opciones.

Figura 10: Escena opciones

La segunda de las opciones sería seleccionar el botón jugada, lo cual dirige la pantalla a la escena jugar ronda, la cual es la escena principal del juego ya que sobre ella se desarrollarán las jugadas de cada turno: las Cartas de Poder sacadas por cada jugador, así como los efectos de las cartas, las Cartas Instantáneas y los empates.

La *Figura* 11 se mostrará continuamente a lo largo de todas las rondas de la partida, y el hilo de ejecución será el siguiente: el jugador escogerá una de las 5 cartas que tenga en mano, que se mostrarán en la parte inferior de la pantalla, pulsando sobre ella. En caso de que el jugador decida cambiar su jugada y elegir otra carta, solo deberá pulsar la carta que ha seleccionado para retirarla y seleccionar la carta elegida de nuevo.

Se mostrarán los diferentes efectos de las cartas elegidas, en caso de que alguna de ellas lo posea, y a continuación se tendrá posibilidad de jugar tantas Cartas Instantáneas como quieran los jugadores.

Para acabar, se obtiene uno o varios ganadores de la ronda, según la mayor puntuación obtenida, así como uno o varios jugadores que hayan perdido la ronda, según la menor puntuación obtenida. Es en este momento cuando, en caso de ser necesario, se procede a jugar la Muerte Súbita.

Figura 11: Escena jugar ronda

Al finalizar la partida, ya sea porque uno de los jugadores haya obtenido 3 Cartas Desastre, por lo que ha perdido la partida, o porque uno de los jugadores haya conseguido 50 puntos, por lo que ha ganado, aparecerá la última pantalla del juego (*Figura* 12) con el resultado de la partida para el jugador humano.

Figura 12: Escena final ganadora

3.3. Diagrama de secuencia

En este apartado se explicarán, haciendo alusión a las escenas nombradas anteriormente, los diagramas de secuencia que muestran todos los posibles caminos a tomar a lo largo del juego.

Como se puede observar en la *Figura* 13, en la primera pantalla existe interacción de clases en relación a la elección del personaje dinosaurio, lo cual hace que el programa avance a la segunda escena, la escena de puntuaciones

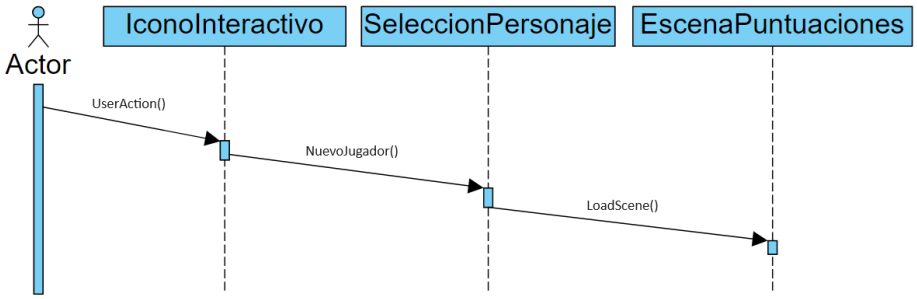

Figura 13: Diagrama secuencia inicio del juego

Para facilitar su lectura, la ronda de juego se ha dividido en 4 diagramas:

En el primer diagrama (*Figura* 14), se muestra cómo, una vez en la escena de puntuaciones, al seleccionar el botón jugada, la escena avanza a la siguiente (es decir, a la escena jugar ronda) donde previo al inicio de la ronda el jugador deberá seleccionar una de sus Cartas de Poder en mano.

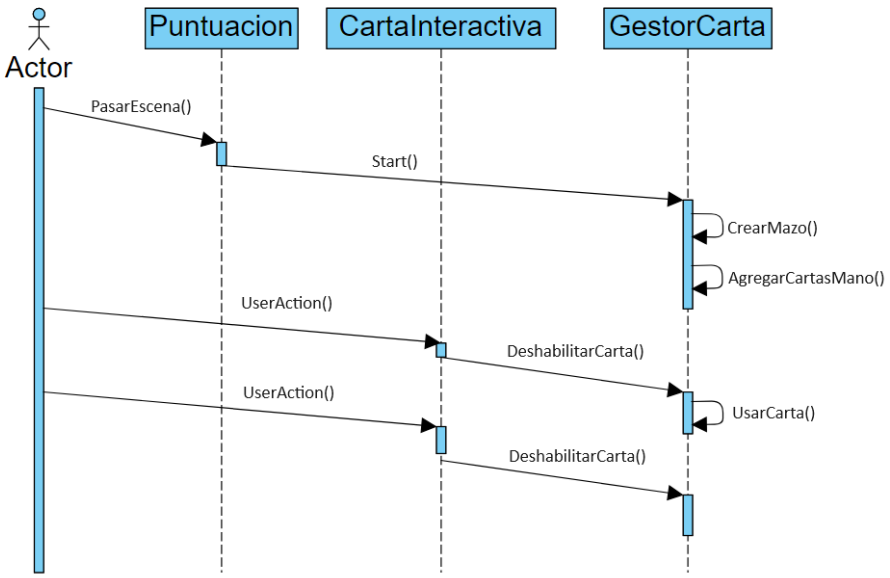

Figura 14: Diagrama secuencia escoger carta

Una vez seleccionada la Carta de Poder e iniciada la ronda, como se muestra en la *Figura* 15, Gestor Jugada es el encargado de llevar a cabo los diferentes efectos.

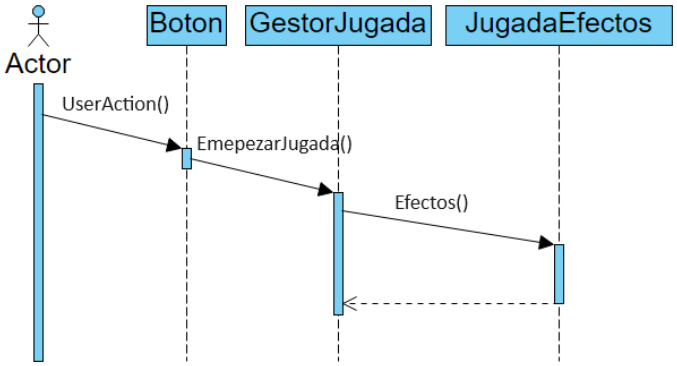

Figura 15: Diagrama secuencia empezar jugada y efectos

Tras haberse realizado los efectos de las cartas, en la *Figura* 16 los jugadores deciden si jugar una de sus Cartas Instántaneas. Si un jugador juega una de estas cartas, se avisará a todos los jugadores de ello, y se les volverá a dar la opción de decidir si quieren jugar una Carta Instántanea.
Existen 3 tipos de Cartas Instantáneas que se pueden jugar en este momento: Score Booster y Score Sapper, cartas con las que en ambos casos el jugador que ha decidido utilizar dicha carta debe escoger tambíen a quién desea que se le aplique el efecto, y por otro lado Score Inversion, con la cual el jugador solo debe escoger la carta.

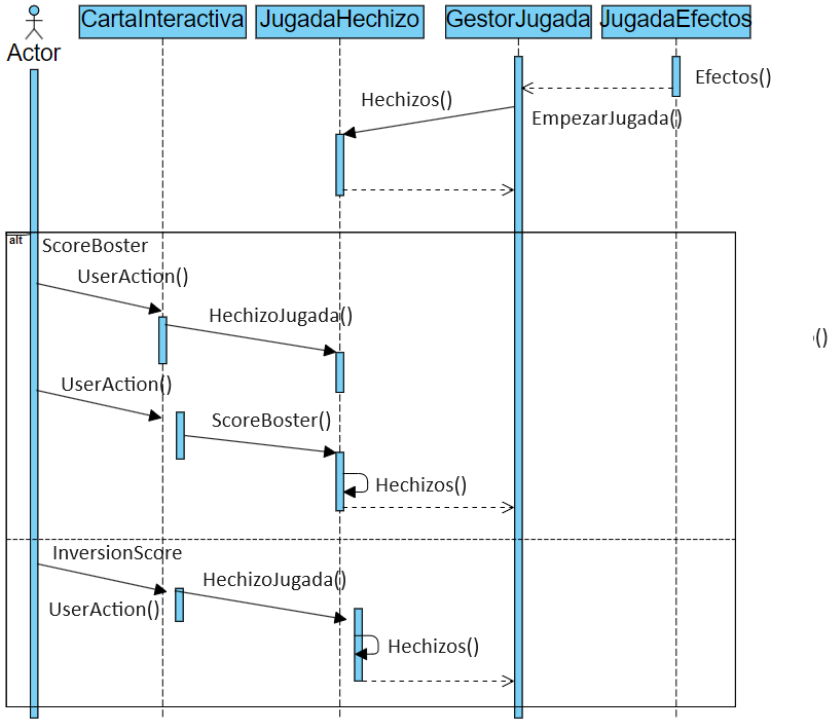

Figura 16: Diagrama secuencia cartas Instántaneas

En caso de que ningún jugador quiera emplear una de sus Cartas Instantáneas, o ya se hayan jugado, a continuación se obtienen los ganadores de la ronda, como se muestra en la *Figura* 17. En caso de haber más de un perdedor, se llama a GestorJugada para realizar una Muerte Súbita, por lo que se vuelve a obtener ganadores. Tras obtener un único perdedor de la ronda, esta finaliza y se retorna a la escena de puntuaciones.

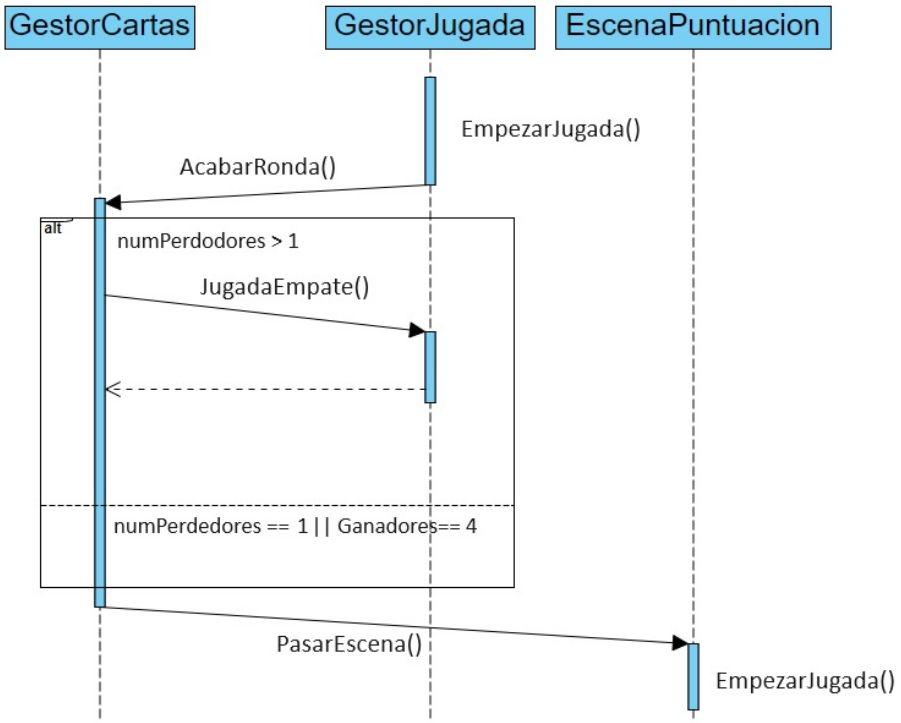

Figura 17: Diagrama secuencia acabar ronda

3.4. Diagrama de módulos

La estructura del juego se puede observar en el siguiente *Diagrama de módulos* (*Figura* 18):

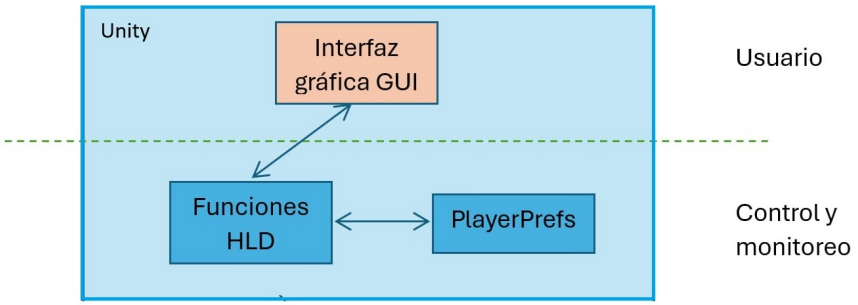

Figura 18: Diagrama de módulos

- En primer lugar, englobando el proyecto se sitúa el entorno de trabajo *Unity*, que se encarga de la comunicación entre los módulos.
- Por otro lado, el módulo Interfaz gráfica (GUI) hace referencia a la funcionalidad de las escenas mostradas en apartados anteriores.
- A continuación, el módulo Funciones HDL engloba el motor del juego. Este está comunicado tanto con la GUI, como con el módulo *PlayerPrefs*.
- Por último, *Unity* incluye el módulo *PlayerPrefs*, el cual se encarga de guardar y gestionar la información del módulo Funciones para no perder información, y que se pueda acceder a ella entre las diferentes escenas.

4. Implemetación de bots: IA

En este apartado se detallarán los diferentes algoritmos de IA que se barajaron para el proyecto, así como las conclusiones que se obtuvieron a partir de su estudio.

4.1. Algoritmo MinMax

El algoritmo MinMax es una técnica de toma de decisiones utilizada en computación, empleado especialmente en juegos de 2 jugadores como el *Ajedrez* o el *Go*, cuyo objetivo es determinar la mejor estrategia para un jugador, asumiendo que el oponente también juega de manera óptima.

Este algoritmo se basa en generar un árbol de decisiones y explorar todos los posibles movimientos del juego hasta una cierta profundidad variable, calculando así el valor de cada hoja según las posibles jugadas futuras, recreando 2 jugadores, a los que denomina Max y Mín, donde Max intenta maximizar su ganancia y Min intenta minimizar la ganancia de Max.

Generación de arbol de juego

El juego se representa como un árbol, donde cada nodo es una posible jugada y los hijos de cada nodo son las posibles jugadas que se pueden llevar a cabo tras realizar esa primera jugada.

Función de evaluación

Esta función evalúa el valor de cada hoja, alternando en cada fila la perspectiva del jugador Max y la perspectiva del jugador Min.

En juegos simples como el tres en raya, esta función asigna un valor positivo a las posiciones ganadoras para Max, un valor negativo a las posiciones perdedoras, y 0 a las posiciones de empate. En juegos más complejos como el *Ajedrez*, eliminar una ficha rival o realizar jaque asignaría un valor positivo y perder una ficha asignaría un valor negativo, pero mover una ficha podría tener distintos tipos de valores.

Recursividad

Si la posición actual es una posición terminal (ganadora, perdedora o empate), se devuelve el valor de la función de evaluación para esa posición, es decir, si es el turno de Max, se exploran todas las posiciones posibles resultantes de los movimientos legales y se selecciona el valor máximo de entre los valores MinMax de las posiciones hijas. En cambio, si es el turno de Min, se exploran todas las posiciones posibles resultantes de los movimientos legales y se selecciona el valor mínimo de entre los valores MinMax de las posiciones hijas.

Conclusión

El método de MinMax podría resultar muy útil en un juego de cartas de 2 jugadores, con información perfecta, pero en este caso, al ser un juego de cartas de más de 2 jugadores, con información imperfecta u oculta, la implementación del juego se podría complicar eligiendo varias veces el menor valor de cada nodo hijo, por lo que el algoritmo de MinMax no se adecúa a nuestros requisitos para el juego.

4.2. Algoritmo de Monte Carlo

El algoritmo de Monte Carlo es un algoritmo de toma de decisiones, utilizado para estimar resultados a través de la generación de muestras aleatorias y el análisis estadístico.

Pasos del algoritmo

El primer paso trata de definir el problema en una función matemática, incluyendo parametros y variables involucradas en el problema.

A continuación genera un número de muestras aleatorias, cada una de las cuales se evalúa utilizando la función definida, y calcula los resultados, obteniendo diversos datos estadísticos como son la media o la varianza.

Por último, el algoritmo realiza inferencias sobre el conocimiento del sistema o de la solución, con las estadísticas obtenidas de cada muestra. Con estas inferencias se pueden estimar las probabilidades para optimizar la respuesta del algoritmo.

Information Set Monte Carlo Tree Search (IS-MCTS)

El algoritmo de Monte Carlo funciona adecuadamente con juegos de información perfecta como son el *Ajedrez* o el *Go*, es decir, cuando toda la información está disponible para los jugadores.

Para casos en el que la información sea imperfecta u oculta, los algoritmos IS-MCTS han sido adaptados, de manera que el algoritmo intenta adivinar aleatoriamente la información oculta, pero en lugar de tratar cada posible ronda como un juego independiente, utiliza todas las rondas para guiarse, es decir, utiliza un árbol en el cual, en cada hoja, los jugadores utilizan una carta diferente.

Para realizar los algoritmos de IS-MCTS es necesario definir 4 funciones:

- `CloneAndRandomize`: Crea una copia del estado y adivina la información oculta.
- `GetMoves`: Devuelve una lista con los posibles movimientos.
- `DoMove`: Aplica un movimiento y actualiza el jugador que realiza el movimiento.
- `GetResult`: Si el estado es terminal y gana obtiene un 1, si pierde un 0 y si empata 0.5.

Para obtener los resultados se han jugado partidas de IA contra IA, comparando el algoritmo de IS-MCTS con 2 jugadores sin información adicional contra otros 2 jugadores que sí tenían información adicional (cómo por ejemplo, saber las cartas que ya se han jugado y no se han vuelto a añadir al mazo principal), lo que tras programarlo para que jugara 100 partidas, obtuvo como resultado que en un 63 % de los casos, ganó el algoritmo que no utilizaba información adicional.

Conclusión

El algoritmo Monte Carlo es útil para ser utilizado en este proyecto, ya que es capaz de determinar la probabilidad de las cartas para ganar la ronda.

Por ello, se entrenó este algoritmo simulando partidas, jugando cartas y esperando la respuesta de los contrincantes, guardando las veces que una carta ha dado un resultado ganador, perdedor o no ha obtenido ninguno de estos resultados en cada una de las rondas.

4.3. Algoritmo de Aprendizaje por Refuerzo

Este algoritmo es una rama del Aprendizaje Automático que se ocupa de las acciones que un agente debe realizar en función de su observación de un entorno para maximizar una recompensa.

Componentes del algoritmo

Los algoritmos de Aprendizaje por Refuerzo se componen de:

- Agente: encargado de tomar la decisión de la siguiente acción a realizar, dependiendo de la información de la que disponga, aunque cabe la posibilidad de su información sea nula. Solo puede realizar acciones legales.
- Entorno: está formado por todo lo que le rodea al agente, incluyendo el estado del sistema, así como las recompensas dependiendo de la decisión tomada.
- Acción: cada una de las posibles decisiones dado un determinado estado.
- Estado: representa la situación actual del entorno.

- Recompensa: respuesta que recibe el agente tras tomar una decisión determinada. Para acciones que se acerquen al estado ganador recibe valores positivos, mientas que, si se aleja o el resto de los jugadores se acercan más al estado ganador, recibe valores negativos.
- Política: estrategía escogida por el agente para la toma de decisiones.

Reinforcenment Learning Card (RLCard)

RLCard es una biblioteca de Python diseñada para el Aprendizaje por Refuerzo especializado en juego de cartas, que se ha utilizado en juegos como el *Póker*, el *Bridge* o el *Uno*. Utilizando el Uno como ejemplo, estos serían sus componentes:

- El entorno sería el propio Uno.
- El estado haría referencia a la última carta jugada, así como a las cartas que el jugador tiene en su mano
- La acción sería jugar una carta determinada de la mano o robar una carta del mazo.
- La recompensa sería una función que permita calcular el beneficio que aporta realizar una jugada concreta.
- Para la política, RLCard ofrece varias opciones, entre ellas DQN (Deep Q Network) o CFR (Counterfactual Regret Minimization).

RLCard ofrece entornos, Agentes y las reglas de diversos juegos, como son el *Blackjack*, el *Uno*, el *Bridge*..., listos para ser implementados. En caso de querer usar la biblioteca para otro juego, RLCard ofrece varias clases del entorno, agente y modelo para heredar de ellas, detallando en su apartado de Desarrollo varios requisitos para hacer funcionar la biblioteca, los cuales son:

- El agente necesita tener una función `step`, `eval_step` y `use_raw`.
- Se debe implementar un juego que contenga las clases Game, Dealer, Round, Player y Judger.
- Cada juego debe estar dentro de un entorno, el cual debe tener las funciones `_extract_state`, `_decode_action` y `get_payoffs`.

Conclusion

El algoritmo RLCard proporciona una serie de beneficios muy significativos, como su facilidad de implementación, ambiente estandarizado, capacidad de comparar algoritmos, recursos de aprendizaje disponibles, y posibilidad de realizar simulaciones rápidas y extensas, lo cual justifica su uso en la fase de investigación y desarrollo de la IA. A pesar de ello, y de haber conseguido implementar una comunicación de C# hacia Python, finalmente no se consiguió entrenar este algoritmo por falta de tiempo.

4.4. Implementación

Tras realizar una búsqueda bibliográfica, y probar los diferentes algoritmos de aprendizaje se optó por crear 3 niveles de dificultad para la IA

El primer nivel, y el más sencillo, se elaboró utilizando unas reglas básicas para la elección de carta, que se regiría por la norma de que la carta con más poder fuera la que se jugara, y el uso de sus efectos, lo cual se regiría por las siguientes normas:

- Carta con poder 0 y 3: Se descartará otra carta de poder de la mano, añadiendo el poder de dicha carta a la ronda. En caso de haber usado la carta de poder 3, la carta descartada debe tener un efecto.
- Carta con poder 1: Al finalizar la ronda robará la mejor carta que posea un rival al azar.
- Carta con poder 2: Puede intercambiar la carta utilizada en la ronda por la de un jugador situado a su izquierda o su derecha, siempre que esa carta también sea de poder.
- Carta con poder 4: Descarta las Cartas de Poder que tengan menos de 5 puntos.
- Carta con poder 7: Roba 2 escogiendo la carta con poder más alto.
- Carta con poder 8: Si no va ganando intercambia la carta para ganar y si no intercambia dos cartas aleatoriamente que no sea la suya.

Para el segundo nivel, de dificultad intermedia, se empleó el método Monte Carlo, con los pesos generados en su entrenamiento para seleccionar la carta disponible de mayor peso. Determinadas cartas, como es el caso de las Cartas de Poder 0 y 3, varían su peso en función de las cartas que el propio jugador posea en su mano.

Por último, en el tercer nivel, que sería el de mayor dificultad, aunque el algoritmo de Aprendizaje por Refuerzo resultó útil y se logró implementar su biblioteca, no se consiguió entrenar su herramienta RLCard.

Cabe destacar que la dificultad del nivel complica la elección de carta en las rondas que no se tratan de Muerte Súbita, en cuyo caso la elección de jugar Cartas Instantáneas y rondas de Muerte Súbita se rige por dos normas muy sencillas.

- Solo se juegan Cartas Instantáneas si con ellas se gana o no pierde.
- Para la ronda de Muerte Súbita, como se explica en apartados previos, solo se tiene en cuenta el poder de la carta, por lo que se juega la carta de poder más alta que posea el jugador.

5. Conclusiones

Como conclusiones principales que se obtuvieron tras la elaboración de este proyecto encontramos que fue posible desarrollar el juego Happy Little Dinosaurs (HLD) en formato digital para su uso sin red, aplicando diferentes algoritmos de entrenamiento de IA para ello. Por un lado, se concluye que el algoritmo MinMax no fue útil para el desarrollo de este proyecto. Por otro lado, el algoritmo Monte Carlo sí que resultó tanto útil como posible de implementar para la elaboración de la IA.

Siguiendo la misma línea, se determinó que el algoritmo de Aprendizaje por Refuerzo era útil, pero no se consiguió entrenar su herramienta RLCard, a pesar de haber conseguido implementar su biblioteca. Debido a la complejidad de modelar el juego con las herramientas disponibles y a la limitación temporal de este proyecto.

De manera personal para su autor, la realización de este trabajo ha facilitado el fortalecimiento de sus bases teóricas previas y ha permitido una profundización significativa en áreas como el desarrollo de Software, los Videojuegos y la IA. Asimismo, ha proporcionado una visión global más amplia en relación con la trayectoria profesional que deseo seguir en el ámbito de los videojuegos.

Durante la implementación del sistema, se ha profundizado en el diseño de arquitecturas de Software, tomando decisiones cruciales sobre las clases e interfaces para acomodar futuros cambios y facilitar su comprensión. Este proceso ha permitido enfrentar tareas complejas, como la comunicación entre clases en diferentes lenguajes de programación y el entrenamiento de algoritmos de IA.

Se ha investigado exhaustivamente sobre los algoritmos de IA, buscando y comparando información sobre aquellos aspectos aplicables en el contexto de los juegos de cartas. A través de esta investigación, se ha aprendido a identificar los algoritmos más adecuados y a implementar entrenamientos para obtener resultados óptimos, utilizando librerías y recursos disponibles en la Web para mejorar el conocimiento y la comprensión del algoritmo durante su implementación y entrenamiento.

En el ámbito personal, se ha adquirido competencia en un nuevo lenguaje de programación y en un entorno de desarrollo que actualmente está en auge en el sector de los Videojuegos. Además, se ha aprendido a discutir y tomar decisiones de diseño relacionadas con la creación de un videojuego, desarrollando una visión más general sobre la implementación. Esto incluye, no solo cumplir con los requisitos inmediatos, sino también pensar a largo plazo para evitar grandes cambios en caso de futuras modificaciones en la aplicación.

En resumen, este trabajo ha contribuido significativamente al desarrollo de habilidades técnicas y personales, y ha proporcionado una comprensión más profunda y amplia del diseño y la implementación de Software en el ámbito de los Videojuegos y la IA.

5.1. Cronograma

La mayor parte del trabajo se llevó a cabo entre Octubre y mediados de Febrero, cuando se estudió y se elaboró casi por completo el código del juego, junto con el nivel fácil de la IA. A partir de entonces, el hecho de que el autor comenzó a trabajar, unido a que se empezaron a implementar los niveles intermedio y superior de la IA, dificultó el avance del proyecto al ser un campo de trabajo relativamente nuevo.

A continuación, en la *Tabla* 3, se muestra la distribución de horas invertidas en cada tarea:

TAREAS	**HORAS INVERTIDAS**
Análisis	**31**
Aprendizaje Unity y C#	15
Selección del juego	4
Estudio de los algoritmos	12
Arquitectura	**7**
Obtención y elaboración de recursos gráficos	7
Implementación	**255**
Cartas y manos de los jugadores	19
Ejecución de ronda	33
Ganador y perdedor de ronda	13
Efectos de las cartas	49
Pantalla resultados y selección de especie	11
Cartas Instantáneas	17
Pantalla de opciones	8
Algoritmo de Monte Carlo	16
Comunicación C# Python	9
Implementación RLCard	80
Pruebas	**24**
Correción de errores	8
Testeo	16
Memoria	**47**
Redacción	23
Correcciones	24
HORAS TOTALES:	**364**

Tabla 3: Distribución de horas en tareas

En la siguiente *Figura* (19) se puede ver como se repartió el tiempo:

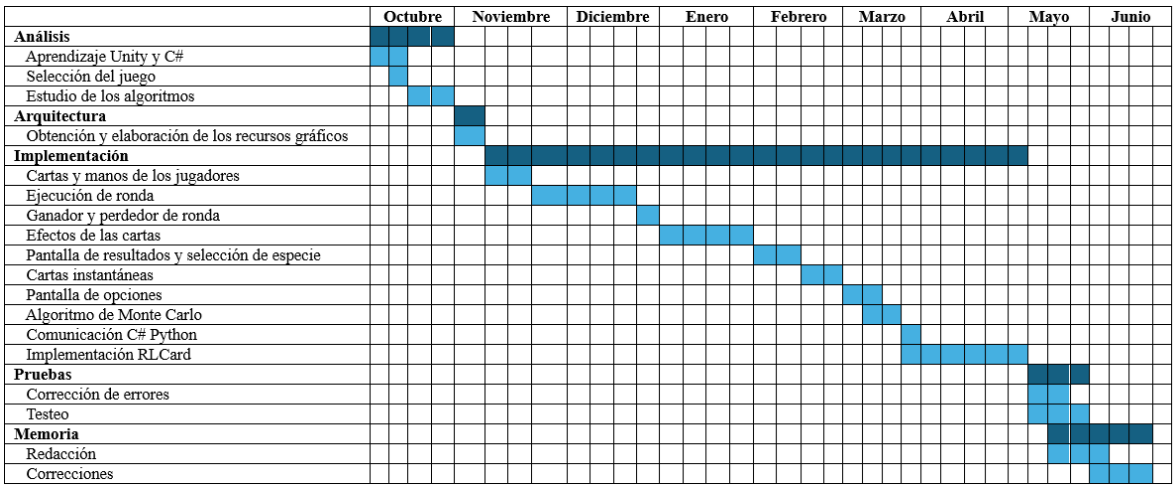

Figura 19: Diagrama Grantt

5.2. Posibles ampliaciones

Tras la elaboración de este proyecto han surgido ideas sobre posibles ampliaciones futuras, las cuales se muestran a continuación divididas en 2 clases:

Referentes al juego:

- Añadir una nueva funcionalidad para que, pasados unos segundos tras la pantalla que muestra el resultado de la partida, se vuelva a la pantalla de selección de especie.
- Incluir nuevas animaciones para dinamizar la partida, como por ejemplo al utilizar Cartas Instantáneas, o mostrando en la pantalla de resultados si alguien ha descartado o redirigido una Carta Desastre.
- Mejorar la pantalla *Jugar la Ronda* para hacerla más intuitiva para el usuario.
- Añadir el modo de 2 jugadores.
- Ampliar las opciones añadiendo el botón *Idioma*, mediante una herramienta de gestión del mismo.

Referentes a la IA:

- Completar el desarrollo del tercer nivel de dificultad, implementando el código del entorno y el agente.
- Utilizar diferentes agentes ya integrados de RLCard.
- Buscar nuevos algoritmos e implementaciones para dotar al juego de más niveles de dificultad.

6. Bibliografía

Best techniques for an AI of a card game. (s. f.). Software Engineering Stack Exchange.
https://softwareengineering.stackexchange.com/questions/213870/best-techniques

Cfa, B. P. (2021, 28 diciembre). Tackling the UNO Card Game with Reinforcement Learning. Medium.
https://towardsdatascience.com/tackling-uno-card-game-with-reinforcement-learning

García-Sánchez, P., Tonda, A., Fernández-Leiva, A. J., & Cotta, C. (2020). Optimizing Hearthstone agents using an evolutionary algorithm. Knowledge-based Systems, 188, 105032.
https://doi.org/10.1016/j.knosys.2019.105032

Helgevold, T. (2023, 8 agosto). Machine Learning and Card Games - Torgeir Helgevold - Medium. Medium.
https://medium.com/tor_92315/machine-learning-and-card-games-6b210f8ec322

In-Op. (s. f.). GitHub - in-op/GameAI: Various C# implementations of game AI. GitHub.
https://github.com/in-op/GameAI

RLCard: A Toolkit for Reinforcement Learning in Card Games — RLcard 0.0.1 documentation. (s. f.).
https://rlcard.org/

Science, W. D. (s. f.). Oh Hell! - Reinforcement Learning for solving card games . Research WDSS.
https://research.wdss.io/oh-hell/#The-Approach

Unity Essentials Pathway - Unity Learn. (s. f.). Unity Learn.
https://learn.unity.com/pathway/unity-essentials

Unity - Introduction. (s.f.).
https://www.tutorialspoint.com/unity/unity_introduction.htm

UnstableGames. (s. f.). Happy little dinosaurs.
https://www.unstablegames.com/collections/happy-little-dinosaurs

Anexos

Diagrama de Clases

Las diferentes clases que tiene el *diagrama de clases* se pueden dividir en:

- Clases del motor: gestionan las funciones del juego, empezar la ronda, jugar efectos, jugar Cartas Instantáneas y obtener ganadores.
  Son las clases GestorJugada, IA, JugadaEfectos, JugadaHechizos, Carta, CartaDesastre y Jugador.

- Clases de interfaz: gestionan la información de la pantalla y las interacciones del jugador humano.
  Son las clases de GestorJugada, SelecciónJugador, EscenaPuntuacion, IconoInteractivo y CartaInteractiva.

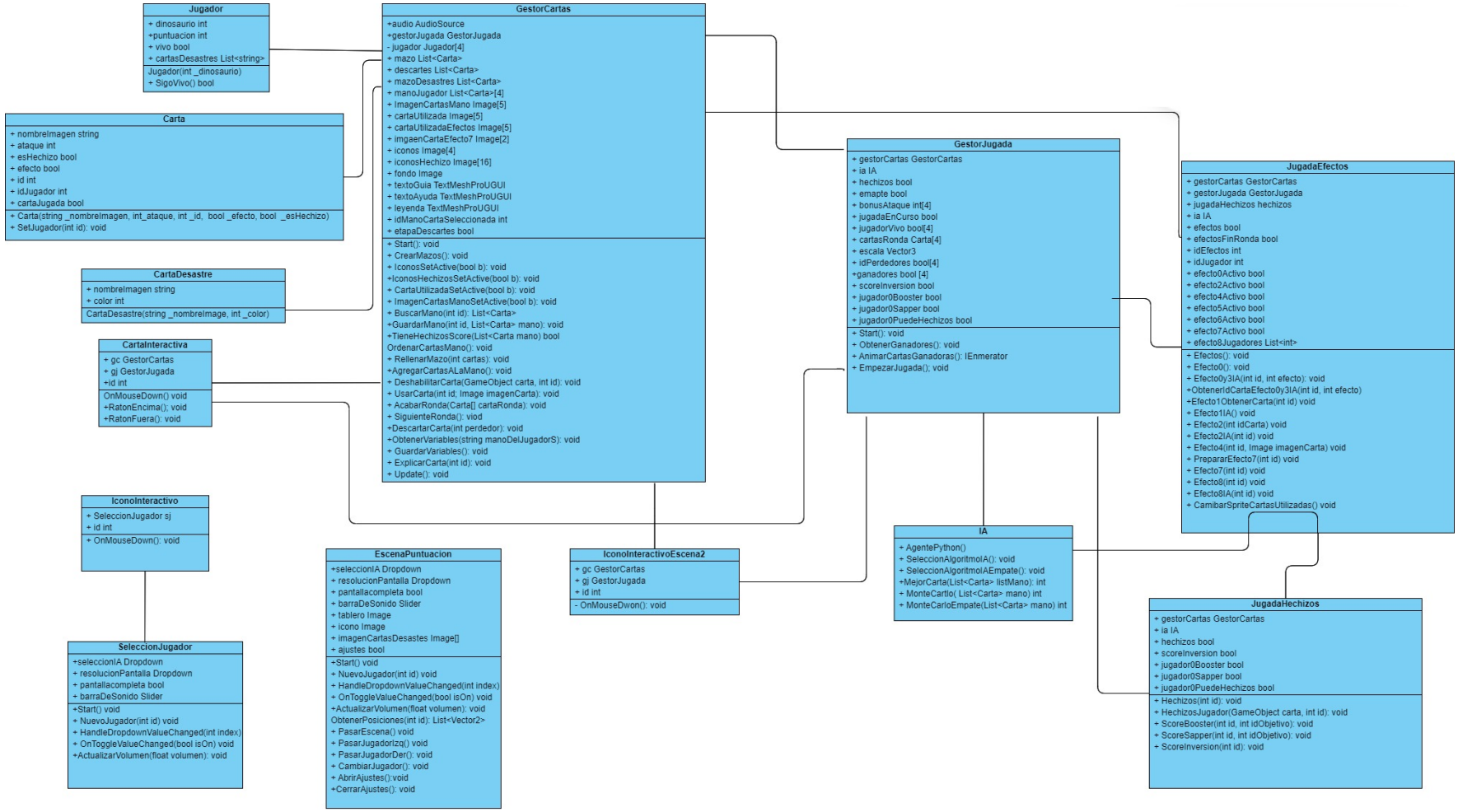

Figura 20: Diagrama de clases del juego de cartas.

Efectos y Cartas Instantáneas

En esta sección se mostrarán los diferentes efectos de las Cartas de Poder y las Cartas Instantáneas, en orden ascendente de poder.

La carta con poder 0, *Figura* 21, tiene el efecto de poder descartar otra Cartas de Poder para añadir su poder al poder actual de la ronda, útil para asegurarte ganar o no perder la ronda, ya que ves el resto de cartas que han utilizado los rivales.

Figura 21: Carta de Poder 0

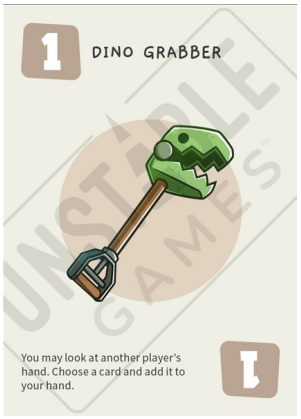

Figura 22: Carta de Poder 1

La carta con efecto de poder 1, *Figura* 22 al acabar la ronda puede elegir un rival para ver su mano y elegir una carta para quedarse con ella.

La carta con efecto de poder 2, Figura 23 puede intercambiar su carta con su rival situado a la derecha o a la izquierda, solo si la cara es una Cartas de Poder con efecto.

Figura 23: Carta de Poder 2

Figura 24: Carta de Poder 3

La carta con efecto de poder 3, Figura 24, tiene un efecto parecido a la Carta de Poder 0, ya que se trata de descartar una carta para añadir su poder al poder actual del turno, pero la carta descartada tiene que ser una carta de poder con efecto.

La carta con efecto de poder 4, *Figura* 25, puede descartar hasta 3 cartas de la mano.

Figura 25: Carta de Poder 4

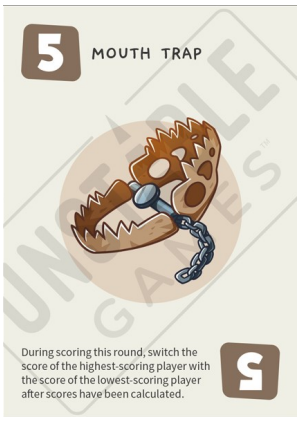

Figura 26: Carta de Poder 5

La carta con efecto de poder 5, *Figura* 26, actúa en el momento de obtener los ganadores ya que intercambia la puntuación más alta por la más baja en caso de que haya un empate, intercambiando ambas puntuaciones.

La carta con efecto de poder 6, *Figura* 27, duplica los efectos de las Cartas Instantáneas Sapper y ScoreBooster, pasando a quitar u otorgar 4 Puntos de Poder en vez de 2.

Figura 27: Carta de Poder 6

Figura 28: Carta de Poder 7

La carta con efecto de poder 7, *Figura* 28, muestra al jugador que la activó las 2 primeras cartas del mazo para que escoga una de ellas.

La carta con efecto de poder 8, *Figura* 29, intercambia dos cartas cualesquiera de la ronda.

Figura 29: Carta de Poder 8

Figura 30: Carta de Poder 9

La carta con efecto de poder 9, *Figura* 30, se activa nada más revelar la carta, y desactiva el resto de los efectos de las cartas de esa ronda.

La Carta Instantáneas Score Booster, *Figura* 31, aumenta en 2 el poder de la ronda actual de cualquier jugador. En caso de estar en juego la carta de poder con efecto 6, el poder de la ronda aumentará en 4.

Figura 31: Carta Instantánea Score Booster

Figura 32: Carta Instantánea Score Sapper

La Carta Instantáneas Score Sapper, *Figura* 32, disminuye en 2 el poder de la ronda actual de cualquier jugador. En caso de estar en juego la carta de poder con efecto 6, en vez de disminuir en 2 el poder de la ronda, aumentará en 4.

La Carta Instantáneas Score Inversion, *Figura* 33, funcíona igual que la carta de poder con efecto 5, intercambia las puntuaciónes más altas y más bajas entre ellas, con la particularidad de que puede servir para contrarrestar esta misma carta o la Carta de Poder con efecto.

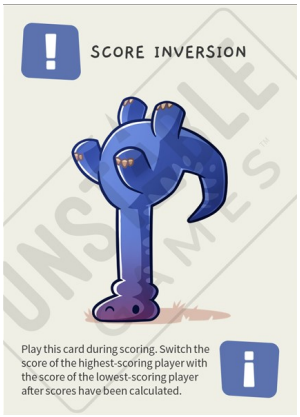

Figura 33: Carta Instantánea Score Inversion

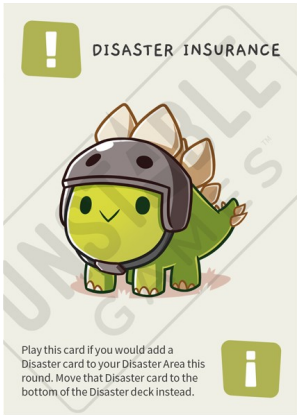

Figura 34: Carta Instantánea Disaster Insurance

La Carta Instantáneas Disaster Insurance, *Figura* 34, evita obtener una Carta Desastre al perder la ronda.

La Carta Instantáneas Disaster Redirect, *Figura* 35, redirige a otro jugador la Carta Desastre que se obtendría al perder la ronda.

Figura 35: Carta Instantánea Score Redirect