UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA EN INFORMÁTICA

PROYECTO FIN DE CARRERA

THE RIFT: DISEÑO DE UN JUEGO DE ESTRATEGIA BASADO EN CARTAS, DESARROLLO DEL PROTOTIPO E IMPLEMENTACIÓN DE UN JUGADOR INTELIGENTE

Autor: Denis Asensio Palacios

Director: Daniel Borrajo Millán

Agradecimientos

Sin duda, a mis padres, por su infinita paciencia, su apoyo y su confianza incondicional. A mi familia en general, por la fe depositada a lo largo de estos años.

Por supuesto, también a mi tutor, por haber aguantado los numerosos mails y, a pesar de mi intermitencia con el PFC, haber estado atento en todo momento.

Resumen

El mercado de los videojuegos es un sector en constante crecimiento y renovación. En concreto, el sector de los juegos de estrategia ha sido durante mucho tiempo uno de los campos más exitosos y cuenta con algunos de los nombres más conocidos de los últimos años [1] [2]. Dentro de la temática de los juegos de estrategia, se encuentra un sub-género relativamente amplio, que será objeto de análisis en las primeras fases del documento: los juegos de estrategia basados en cartas coleccionables o transferibles, conocidos como CCG y TCG respectivamente. Se cuenta con la presencia de CCG's y TCG's tanto en el sector del ocio informático, por ejemplo con el reciente y exitoso *Hearthstone*, como en el sector del ocio común, con ejemplos muy conocidos como *Magic: The Gathering* o *Yugi-Oh*, que incluso tienen también adaptaciones como videojuegos.

En resumen, el mercado de los videojuegos de estrategia basados en cartas está representado por muchos juegos de muy diverso tipo, siendo un reto el diseño de un juego novedoso y entretenido. El principal objetivo del PFC es sentar las bases para el futuro diseño y desarrollo de un videojuego de estas características, creando un prototipo que pudiera servir para el desarrollo de un juego independiente. Tras el diseño de la estructura y mecánicas del videojuego, de nombre "The Rift", se realizará una aplicación que permita al jugador tener un primer acercamiento al juego. Para ello, será importante, además de proporcionar una funcionalidad básica, la creación de un jugador inteligente de nivel medio y de una infraestructura que permita el juego remoto.

[1] Sagas *Age of empires*, *Starcraft*, *Warcraft*, *Command&Conquer*, *Civilization*, etc.
[2] (Metacritic). *Metascore* del género estrategia.

Contenido

Gestión del proyecto:

Bibliografía:

Anexos:

Introducción y objetivos

En esta sección se introduce la motivación del proyecto y se definirán los objetivos a tratar, así como la estructura general del documento.

1.1 Introducción

Desde hace años, la informática se ha extendido hasta convertirse en pieza clave de numerosos sectores, siendo uno de los más notorios el sector del ocio y el entretenimiento. La tasa de crecimiento de la industria del videojuego ha sido tal que, en la década de 2000, los videojuegos generaron más dinero que las industrias del cine y la música juntas.

A pesar de que en las primeras supercomputadoras programables ya se realizaron intentos de creación de software de carácter lúdico, no fue hasta la década de los 60 cuando la industria comenzó a formarse con la creación de los primeros videojuegos modernos, iniciando una era de comercialización y progreso tecnológico a tal escala que, apenas medio siglo después, el videojuego goza del estatus de medio artístico. Además, el carácter adaptativo y de innovación de dicha industria la sitúan en el primer puesto en cuanto a mayor proyección de crecimiento.

Dentro de la industria de los videojuegos, destaca entre otros el género de los juegos de estrategia, con una diversidad apabullante que lleva desde los videojuegos de estrategia por turnos[3] a los más modernos "tower defense"[4], pasando por adaptaciones de juegos de mesa de lo más variopinto. Dichas adaptaciones, en principio orientadas a juegos clásicos como el Ajedrez o los juegos de naipes, han inspirado la creación de juegos de tablero exclusivamente para computadora, que se han hecho hueco en el mercado a base de grandes éxitos o incluso como mini-juegos dentro de otras superproducciones. En concreto, la saga "Final Fantasy" se caracteriza por la inclusión de varios mini-juegos en muchas de sus creaciones, siendo algunos de ellos considerados de culto por la complejidad y horas de juego ofrecidas por un mini-juego. Uno de ellos, un juego de cartas coleccionables que conseguías a lo largo de la aventura principal del "Final Fantasy

[3] Por ejemplo X-COM o Final Fantasy Tactics.
[4] Como el exitoso "Plants vs. Zombies".

VIII" llamado "Triple Triad"[5], tuvo especial éxito, inspirando incluso alguna implementación de carácter amateur y gratuita para su uso fuera del juego original[6]. Sin embargo, a pesar de su éxito y de ser un juego de relativa complejidad, la falta de un horizonte claro y un sistema de reglas equilibrado y fácilmente ampliable condenaron el juego a permanecer como lo que era, un mini-juego.

En el proyecto expuesto en este documento se propone el reto de tomar las reglas más elementales del juego "Triple Triad", que le concedieron tal éxito parcial, y construir un juego de cartas equilibrado y de fácil ampliación, que no solo permita la creación de un videojuego a partir de un set de reglas y una base de datos de cartas determinados, sino que esté abierto a nuevas reglas y expansiones con conjuntos de nuevas cartas, en la línea de otros juegos de éxito en el mercado. La implementación del videojuego será en el lenguaje JAVA, y se realizará de tal forma que esté encapsulada en varios módulos que permitan el juego remoto y la clara diferenciación en las funciones de cada clase: Motor de juego, Servidor y Cliente/s. Además del diseño e implementación del videojuego en cuestión, se propone implementar un jugador inteligente básico que permita a un solo jugador humano realizar partidas de un nivel aceptable contra la CPU.

1.2 Objetivos

Los objetivos de este proyecto están divididos en tres fases muy diferenciadas.

1.2.1 Diseño del videojuego

Esta primera fase trata los objetivos característicos de la creación de una base para un juego de cartas exitoso, siendo estos:

- **Sencillez:** Las reglas básicas deben ser sencillas, claras e intuitivas.
- **Complejidad:** A pesar de la sencillez de sus reglas, las posibilidades en partida no deben ser de carácter lineal y totalmente previsible.
- **Equilibrio:** Prácticamente toda carta debe tener su utilidad o estar valorada de tal forma que su uso pueda salir rentable en alguna situación. Por extensión, ninguna carta debe ser siempre la mejor opción.
- **Escalabilidad:** Aprovechando la sencillez del juego, se establecerán varias reglas y/o habilidades más avanzadas que servirán como ejemplo para futuras creaciones de nuevas reglas o habilidades. Así mismo, se crearán las

[5] http://finalfantasy.wikia.com/wiki/Triple_Triad
[6] http://www.tripletriadflashonline.com/es/play

bases para poder profundizar en ámbitos como la creación de mazos equilibrados o la creación de nuevas cartas para añadir a la base de datos inicial.

- **Bien enfocado:** Debe aprender de los errores cometidos por el juego en el que está basado el nuevo diseño, tales como la predictibilidad y falta de incertidumbre, o la falta de un factor aleatorio que diferencie una partida de la siguiente.
- **Exportable:** Se tendrá muy en cuenta la posibilidad del cambio de formato del juego en un futuro. Es decir, se planteará el juego de forma que, a pesar de ser un juego de ordenador, se pudiera crear un juego de mesa de cartas coleccionables físicas.

1.2.2 Implementación

En la segunda fase se abordará la programación en JAVA de un software que permita poner en práctica el juego diseñado previamente:

- **Encapsulado:** La implementación se dividirá en tres módulos (Motor, Servidor y Cliente) aislados con respecto a los demás módulos, que compartirán información exclusivamente a través de canales creados para la transmisión de la misma.
- **Juego remoto:** Permitir la comunicación e intercambio de información entre módulos físicamente no interconectados.
- **Concurrencia básica:** A pesar del carácter secuencial de los juegos por turnos, se establecerá una base que permita a cada jugador-cliente comunicarse con el servidor de forma concurrente, previendo la necesidad de dicha característica en posibles futuras implementaciones.
- **Interfaz básica:** Se creará una interfaz gráfica mediante caracteres ASCII que se representará en consola y proporcionará una jugabilidad cómoda aunque básica.
- **Varias opciones de juego:** Se implementarán varios tipos de jugador, que serán elegidos al comienzo del juego, y se permitirá cualquier combinación de ellos: Jugador Humano (control manual), Jugador aleatorio (elección de movimiento aleatoria) y Jugador inteligente (elección de la jugada mediante un sistema de Inteligencia Artificial).
- **Control de errores:** Se acotarán las opciones del cliente-jugador para que no pueda provocar un error en el servidor y/o motor con entradas de datos incorrectas.

- **Otros:** Se implementarán aspectos del programa que aumenten la usabilidad, tales como mensajes aclaratorios o la posibilidad de repetir una partida con los jugadores y mazos de la partida previa.

1.2.3 Jugador inteligente

En la última fase se programará un agente inteligente capaz de jugar al software implementado, cuyos objetivos son:

- **Evaluación:** No se asegura que el jugador inteligente tome siempre la elección correcta, básicamente porque en un entorno con tanta incertidumbre dicha opción no siempre será factible, si no que evalúe las posibles opciones y tome la que parezca más efectiva.
- **Incertidumbre:** Se espera al menos una mínima forma de manejar la incertidumbre, poder sopesar jugadas teniendo también en cuenta la información a la cual no se tiene acceso.
- **Planificación básica:** No solamente se tendrá en cuenta que la jugada afecte en gran medida al tablero en el turno en el que se realiza la evaluación, sino también el efecto que dicha jugada podría tener en turnos consiguientes, especialmente el turno posterior.
- **Humano:** Se intentará crear un sistema de inteligencia que emule en parte el método de razonamiento humano, en el caso del juego en cuestión, una valoración de las posibles jugadas con un refuerzo positivo o negativo similar a los “pros” y “contras” que un humano tendría en cuenta al evaluar un movimiento. Además, este método de razonamiento “humano” implicaría no tener un acceso privilegiado a información que pudiera no resultar intuitiva para un jugador experto, por ejemplo, al número exacto de cartas que posean el valor “5” como atributo más alto.
- **Calidad mínima:** El éxito mínimo esperado es un diseño de Jugador Inteligente que supere con creces a un jugador que realice movimientos aleatorios o con evaluación mínima. Sin embargo, se aspira a tener un comportamiento capaz de dar problemas a jugadores de nivel medio.

1.3 Estructura del documento

El documento se encuentra dividido en tres partes diferenciadas, más un cuarto apartado que contiene los resultados de la fase de pruebas:

- En el primer apartado, se explicará la fase de creación y diseño del juego, haciendo un repaso del mini-juego original del cual surgió la idea y posteriormente detallando las reglas y las decisiones de diseño que se llevaron a cabo.
- En el segundo apartado, se explicará el funcionamiento del Jugador Inteligente, empezando por un breve repaso de la teoría que ha ayudado a su diseño, para acabar detallando la estrategia seguida por el Jugador Inteligente y cada uno de los métodos que lo componen.
- En el tercer apartado se detallará la estructura de la aplicación, tanto desde el punto de vista de la arquitectura modular, como desde un punto de vista de bajo nivel (diagramas de clases, explicación de cada método relevante).
- Por último, el cuarto apartado recoge los resultados obtenidos en la fase final de testeo, en la que se pretendía especialmente conseguir una idea de las limitaciones del Jugador Inteligente y comprobar qué objetivos se habían cumplido. En menor medida, se detallan posibles fallos de diseño o mejoras de cara a la siguiente versión del juego.

Diseño del juego

En esta sección se estudiarán las fases seguidas en el diseño del juego y las elecciones realizadas, así como las motivaciones que han llevado a tomar cada una de ellas.

2.1 Análisis

Como se comentó en apartados anteriores, la idea de la creación de "The Rift" surge de un mini-juego llamado "Triple Triad", que formaba parte de un exitoso juego de la plataforma *Play Station*. Por lo tanto, el primer paso hacia la creación de un videojuego con alguna expectativa de futuro sería analizar las características del juego original, tanto las mecánicas que lo convertían en un mini-juego atractivo y con cierta "re-jugabilidad" (del inglés *replayability*), como las razones por las que el mini-juego nunca tuvo la oportunidad de crear un mayor impacto. Por lo tanto, en este apartado se analizarán tanto las reglas y características básicas como las reglas avanzadas y detalles que en principio podrían resultar menos relevantes.

2.1.1 Reglas Básicas

Triple Triad es un mini-juego que se activaba de forma opcional al hablar con otro personaje que también dispusiera de cartas coleccionables. Una vez aceptada la partida, el jugador era trasladado a la pantalla del mini-juego, la cual mostraba un tablero 3x3 en el centro y, en el lado izquierdo, la mano del jugador, mientras que en el lado opuesto se mostraba la mano del oponente (ver figura).

Figura 1: Tablero de juego

La mano de cada jugador estaba formada por exactamente cinco cartas, las cuales eran elegidas a la hora de crear el mazo. El objetivo del jugador era conseguir que la mayor parte del tablero estuviera ocupada por cartas controladas por él. Para ello, los jugadores iban jugando cartas por turnos, las cuales "luchaban" entre ellas, hasta que el tablero estaba completo.

En cualquiera de los ejemplos de la figura 2 se puede observar la estructura de una carta, siendo ésta: 4 valores (*Ranks*), situados en la parte superior, inferior, izquierda y derecha; una imagen representativa del personaje o monstruo en cuestión (*Illustration*); un icono representando la afinidad elemental de dicha carta (*Elemental Attribute*), y una franja inferior con datos varios sobre la carta, que no influyen en el combate (*Title bar*).

Figura 2: Estructura de una carta

Una vez dentro de la pantalla de partida, el jugador inicial se decidía con un lanzamiento de moneda. El jugador inicial procedía pues a elegir una de las cinco cartas de su mano y colocarla en una de las nueve casillas del tablero. A continuación, el otro jugador realizaba su movimiento de la misma forma. En caso de que una carta recién jugada se enfrentara a (compartiera lateral con) una carta rival, si el valor de la carta jugada era mayor vencía, y hacía que la otra carta pasara a su bando. Si el valor era menor o igual no pasaba nada, ambos jugadores conservaban su carta. Este proceso se repetía un total de nueve turnos, hasta que el tablero quedaba completo, en cuyo caso el jugador que poseía más cartas de su color ganaba la partida. En la figura se muestra un ejemplo de una partida a medias.

Figura 3: Quinto turno de una partida

En el juego original Triple Triad se podía dar el evento de empate, siendo éste un caso muy común, dado que la única carta que no se jugaba (cada jugador poseía 5 cartas y solo 9 son jugadas) también contaba para el cómputo general de la partida. En caso de empate, la partida contaba como nula a fin de determinar un ganador, o bien se activaba la regla especial (ver el apartado siguiente) “Sudden Death”, que reiniciaba la partida pero distribuía las cartas dependiendo del color con el que finalizaron la partida. En la próxima figura se puede ver un ejemplo de partida finalizada.

Figura 4: Partida finalizada con victoria del jugador azul

Al finalizar, el ganador obtenía la recompensa, que consistía normalmente en una o varias cartas del mazo rival.

2.1.2 Reglas avanzadas

A lo largo del juego se iban añadiendo reglas cada vez más complejas y en la mayoría de casos confusas, que se propagaban de región a región dentro del juego y se iban acumulando, hasta formar un conjunto de reglas totalmente anti intuitivas que, si bien es cierto que una a una ofrecían un reto y una novedad, cuando se aplicaban varias cambiaban el juego totalmente. El conjunto de reglas era el siguiente:

- **Open:** Es la primera regla y prácticamente se aplicaba a todas las partidas, con lo cual se podría tratar como una regla básica. Su función era que ambos jugadores pudieran ver las cartas de la mano del oponente. Como se puede observar, la regla más aplicada y más básica eliminaba por completo cualquier tipo de incertidumbre, cambiando drásticamente el concepto inicial del juego.
- **Same:** Si una carta tocaba a dos o más cartas enemigas, y los números que hacían contacto eran exactamente los mismos, las cartas enemigas se volteaban (se ganaba el control de dicha carta).
- **Same wall:** Mismo concepto que la anterior, solo que los muros exteriores del tablero contaban como una "A" (máxima puntuación posible, por encima del 9) a objeto de voltear cartas con la regla "Same".
- **Sudden death:** Ya comentada anteriormente. En caso de empate se reparten las cartas que controla cada jugador en ese momento y se comienza una nueva partida.
- **Random:** Se crea un nuevo mazo con cartas aleatorias de la colección del jugador, y se utilizará para la partida.
- **Plus:** Similar a "Same". Si la carta hace contacto con dos cartas enemigas, y la suma de ambos valores de la carta jugada igualan a la suma de los valores en contacto de las cartas enemigas, ambas son volteadas.
- **Combo:** Las cartas capturadas mediante "Same", "Same wall" y "Plus" cuentan como si se hubieran jugado ese turno, es decir, voltean a las cartas enemigas con un atributo menor en contacto.
- **Elemental:** El tablero contiene casillas con un determinado símbolo elemental. Si una carta de ese elemento es colocada en dicha casilla, obtiene +1 a todos los rangos.

2.2 Diseño

A partir de las reglas básicas y avanzadas analizadas previamente, se procederá a diseñar un juego que cumpla con los objetivos propuestos, mejorando en los aspectos que se consideran importantes para la realización de un juego

escalable y con futuro. El primer paso para decidir qué mecánicas conservar y cuáles añadir es analizar las virtudes y los defectos del juego original.

2.2.1 Virtudes

Se han considerado como tales las siguientes características:

- **Ágil:** Las partidas son cortas y dinámicas, lo que minimiza el tedio provocado por las fases más pesadas de los juegos con partidas largas, y aumenta la re-jugabilidad.
- **Simple:** Las reglas básicas son fáciles de aprender y poner en práctica.
- **Complejo:** A pesar de la simplicidad de las reglas básicas, se consigue cierta complejidad en las partidas.
- **Empate:** El juego posee un sistema capaz de igualar el desequilibrio provocado por la ventaja de poder jugar una carta más en el tablero (número de turnos impares). Sin embargo, esto se podría considerar un arma de doble filo, dado que si el juego pretende determinar un vencedor debe haber alguna forma de desempatar.

2.2.2 Defectos

Por otro lado, los defectos encontrados en el juego y a los que debe hacer frente el nuevo diseño son:

- **Predictibilidad:** Probablemente el principal problema que se ha querido paliar, y el que evita que el juego sea escalable y tenga un futuro mayor. El juego se torna predecible debido a varias características, entre ellas el momento en el cual la primera regla avanzada es aplicada: Open. Esta regla elimina totalmente la incertidumbre y, unido al hecho de que los mazos son de cinco cartas elegidas por cada uno de los jugadores, provoca que las partidas entre dos mismos jugadores sean siempre la misma partida o variaciones muy ligeras. A esto se le suma también la falta total de aleatoriedad, lo cual no se considera categóricamente negativo (por ejemplo en el caso del Ajedrez), pero que en un juego con un número de jugadas tan limitadas y con las características ya mencionadas en este apartado acaba creando la sensación de estar jugando la misma partida una y otra vez.
- **Escalabilidad:** Como ya se ha comentado, el juego está diseñado para ser un mini-juego. Se encuentran múltiples problemas a la hora de crear nuevas reglas o ampliar las existentes. Por ejemplo, el sistema es muy cerrado en el sentido en el que no se podría crear un tablero 4x4 en el que se pudiera

jugar, o habría reglas que dejarían de tener sentido o habría que eliminar completamente.

- **Opacidad:** Con la adición de las reglas especiales, particularmente cuando se aplican en conjunto, el juego se torna confuso, poco intuitivo. A esto ayuda el hecho de que las reglas avanzadas se aplican obligatoriamente a todas las cartas. En lugar de poseer cierta carta una habilidad que cambie el transcurso de la partida, las reglas se aplican a todas las cartas, lo que hace que la creación de nuevas reglas o habilidades sea imposible sin aumentar más aún la opacidad del juego. Esto afecta directamente a la escalabilidad.
- **Elección de mazo:** Otro de los aspectos clave, especialmente en el caso de un juego de cartas coleccionables, es la posibilidad de configurar y crear un mazo con una estrategia determinada, donde cada carta de tu colección tenga la posibilidad de formar parte del mazo. El hecho de que solo haya dos opciones (mazo de cinco cartas elegidas por el jugador o mazo totalmente aleatorio si la regla "Random" está activa) va totalmente en contra del espíritu de los CCGs.
- **Cartas simples:** Aunque es parte de las características positivas del juego, la falta de habilidades o reglas especiales en las cartas las convierte al final en conjuntos de números sin más "personalidad".

2.2.3 Conclusiones y nuevo diseño

Tras valorar las virtudes y defectos del juego original, se procedió a diseñar las mecánicas de "The Rift".

El primer objetivo era que el nuevo diseño conservara las características más genuinas de "Triple Triad". Las reglas de volteo de una carta son las mismas que en el juego original, es decir, una carta enemiga es volteada si se juega una carta cuyo atributo en contacto con el atributo de la carta enemiga es mayor. También se conservan las características básicas del desarrollo de la partida, donde cada turno el jugador activo situará una carta de su mano directamente en el tablero, y a continuación se procederá al siguiente turno, donde actuará el que anteriormente era el jugador pasivo. La partida acaba cuando el tablero esté completo, siendo el jugador con más cartas bajo su control el ganador de la partida. Sin embargo, en el juego original se contabilizaba la carta que quedaba en la mano del jugador que actuó en último lugar, lo cual no es posible en el nuevo diseño debido a varios cambios que se comentarán más adelante. Por lo tanto, en "The Rift" no se concibe la posibilidad de un empate, evitando al mismo tiempo la carga de conseguir un método de desempate justo.

Otra de las decisiones de diseño orientadas a perpetuar las características genuinas del juego original es la conservación del diseño básico del tablero, un formato 3x3 que mantendría el modo de juego ágil, con partidas cortas y dinámicas. En diversas fases del desarrollo se meditó la posibilidad de aumentar el tablero a un formato 4x4, debido entre otras razones a un problema surgido con el desequilibrio causado por la ventaja de jugar primero (ergo, jugar una carta más en total). Sin embargo, la pérdida de dinamismo en las partidas y el nuevo problema añadido de determinar un vencedor en caso de empate no acababa de resultar rentable. Se pensaron varias formas de desempate, desde calcular el sumatorio del atributo "valor" (ver más adelante en la descripción del diseño de las cartas) de todas las cartas y premiar al que poseyera las cartas más valiosas bajo su control, hasta calcular el sumatorio del valor de las cartas jugadas y otorgar la victoria al jugador que hubiera usado las cartas de menos valor total, considerando el mayor mérito de vencer con cartas en principio menos valiosas. Dado que todos los métodos de desempate planteados eran en cierta manera injustos, ligeramente arbitrarios o incluso alteraban el modo de juego y la táctica a seguir, al final se decidió conservar el diseño del tablero 3x3 original. Cabe mencionar que a pesar de optar por el tablero 3x3, el diseño del resto de mecánicas, cartas y habilidades se realizó teniendo en cuenta la posibilidad de desarrollar nuevos modos de juego en un futuro, entre ellos los tableros 4x4, 5x5, tableros asimétricos o incluso con casillas bloqueadas.

Una vez establecido el tablero, se procedió a diseñar el modelo de carta. La idea original parecía acercarse mucho a la meta del nuevo juego, al menos en lo básico: cartas con cuatro atributos, uno para cada lado de la carta, que permitieran a éstas confrontarse con las cartas rivales midiendo los atributos en contacto. Sin embargo, se decidió ampliar la idea original de la carta añadiendo más características clave, ya que esto solucionaba varios de los defectos contemplados en el análisis del diseño original. El defecto quizá más obvio que se paliaba era la ya comentada falta de "personalidad" de las cartas, es decir, algún detalle que las diferenciara más allá de los atributos numéricos. A continuación se comentarán las características que conformarán el diseño de la carta en esta primera versión, con una explicación de lo que aporta dicha característica, si se considera relevante.

- **Nombre:** Nombre de la carta.
- **Atributo superior:** Valor del atributo que presenta la carta en la parte de arriba.

- **Atributo inferior:** Valor del atributo que presenta la carta en la parte de abajo.
- **Atributo derecho:** Valor del atributo que presenta la carta en el lateral derecho.
- **Atributo izquierdo:** Valor del atributo que presenta la carta en el lateral izquierdo.
- **Habilidades:** Lista de habilidades poseídas por la carta. Las habilidades son reglas especiales que se aplican solo cuando la carta que las posee sea utilizada. Se considera una extensión de las "Reglas avanzadas" del juego original, con la ventaja de que permite una cantidad de nuevas mecánicas mucho mayor y una personalización de mazos y estrategias a un nivel totalmente diferente. Las mecánicas ya no resultarán confusas dado que no se aplicarán todos los turnos a todas las cartas, sino que a la hora de diseñar una nueva carta se decidirá el nivel de "opacidad" que dicha carta aplicará a la partida, desde no poseer ninguna habilidad especial hasta presentar varias mecánicas adicionales que aumenten la complejidad. Esta simple mejora elimina los defectos de *opacidad* de la partida y *simplicidad* de las cartas, a la vez que dispara la *escalabilidad* del juego y mejora la *elección de mazo*. La decisión de añadir la posibilidad de que una carta pueda poseer habilidades especiales es de tal relevancia que se comentará en mayor profundidad en un apartado posterior de esta misma sección.
- **Valor:** Atributo numérico que determina la calidad de la carta en función de una valoración aproximada de sus atributos y habilidades. A pesar de no presentar una utilidad real en esta primera versión, se pretende utilizar este parámetro como una medida a la hora de limitar la construcción de mazos. Los jugadores deberán elegir las cartas que conforman su mazo de forma que la suma de valores no supere el límite de puntos permitidos por la modalidad de juego. Cada modalidad tendrá un límite de puntos más o menos restrictivo, creando meta-juegos (del inglés *metagame*) diferentes para cada modalidad. Los límites concretos y el número de modalidades aún están por definir, necesitando una cantidad de testeo mucho mayor para tener una primera aproximación. Otra aplicación de este parámetro podría ser el ya mencionado desempate en tableros con número de casillas pares.
- **Rareza:** Representa la dificultad de obtención de dicha carta. Aun siendo una primera fase del juego, uno de los objetivos era asentar las bases para un juego escalable y para nada limitado, con lo cual la creación de dicho parámetro establece la rareza de la carta en un gran rango de posibles

ámbitos: desde un juego de ordenador RPG[7] donde se ganen cartas según se avance en la historia o se suba de nivel, hasta la venta de sobres o packs de cartas en caso de evolucionar a un juego de cartas coleccionables en formato físico. Dicho parámetro puede tomar los siguientes valores, en orden ascendente de rareza: Común, Rara, Épica y Única.

- **Facción:** Parte del "lore" o historia del juego. A pesar de tener una idea muy básica de la ambientación del juego, se ha incluido éste parámetro con valores representando varias facciones básicas a las que puede pertenecer una carta. Aunque no tiene ningún efecto actual en partida, en un futuro habrá habilidades que sirvan de apoyo para las cartas que compartan facción con la carta que posea la habilidad, o incluso habilidades que añadan bonificadores a la hora de enfrentarse a cartas de determinadas facciones.
- **Imagen:** Ilustración representativa de la carta. En la primera versión del juego no hay ilustraciones disponibles.

En la figura siguiente se muestra un boceto de una posible carta[8]. El nombre ("Hambre insaciable") y los atributos se distinguen claramente. La rareza se indicaría con el color del borde de la carta, y en el espacio inferior se indicarían las habilidades ("ECO", en este caso). A la derecha del nombre se encontraría el símbolo representativo de la facción.

Figura 5: Ejemplo de carta

[7] Role Playing Game: un género de videojuego caracterizado por el desarrollo de uno o varios personajes y sus características de forma paralela al avance de la historia.

[8] Se desconoce al autor de la imagen. En ningún momento se pretende reclamar la autoría sobre ella.

Una vez definido el curso de una partida, el tablero y definidas las cartas, se procedió a solventar los dos únicos pero relevantes defectos que quedaban sin paliar: la predictibilidad y la falta de incertidumbre. Para ello se abordan los temas de la construcción de mazos y manejo de la "mano", estrechamente relacionados. El principal problema que poseía el juego original era la poca variabilidad entre una partida y la siguiente, debido a que el jugador elegía solamente cinco cartas de entre su colección sin restricción alguna (o con una restricción de elección aleatoria que eliminaba cualquier estrategia posible), con lo cual el jugador se preocupaba de elegir las cinco mejores cartas, que conformarían su mazo todas las partidas. Y esas cinco cartas no solo serían su mazo, sino que además conformarían su mano en cada una de dichas partidas, dado que no hay diferenciación entre mazo y mano. Además, las cartas estarían a la vista del rival en la mayoría de casos, dado que la regla "Open" se aplicaba en todas las partidas, con lo cual no solo se eliminaría la variabilidad y el factor aleatorio si no también la incertidumbre.

Por las razones recién comentadas, la primera medida fue la de diferenciar claramente entre "mazo" y "mano". En el nuevo diseño, el mazo estaría formado por un total de veinte cartas elegidas por el jugador del conjunto de cartas que forma su colección, y la mano inicial estaría formada por las primeras cinco cartas robadas de dicho mazo. La construcción del mazo estaría delimitada por las reglas de la modalidad de juego pertinente, siendo lo más común un límite numérico al sumatorio del atributo "valor" de las cartas del mazo y un límite de dos cartas iguales por mazo. Además, cada turno después del primero, el jugador activo procederá a robar una carta de su mazo y añadirla a su mano, con lo cual no solo se juega con las cinco cartas iniciales, si no que existe la posibilidad de reforzar tu mano en los turnos consiguientes. Con estas medidas lo que se pretende es encontrar un equilibrio entre el factor aleatorio de una partida, que contribuye directamente a diferenciar una partida de otra, y el factor determinado por la calidad del jugador. Se pretende encontrar un equilibrio en el cual un jugador de menor calidad pudiera ganar a un jugador de mayor calidad, aunque normalmente con una dosis de suerte relevante. Otra de las ventajas de este sistema es que se puede jugar con el número de cartas iniciales para otorgar cierta compensación al jugador que juega en segundo lugar, y que por lo tanto tendrá menos cartas en mesa. Actualmente, la ventaja del segundo jugador se limita a que el jugador inicial no roba carta en el primer turno, pero se sospecha, a falta de más testeo y estadísticas,

que dicha ventaja no es suficiente como para equilibrar las posibilidades de ambos jugadores.

2.2.4 Reglas avanzadas

Como ya se ha comentado en los apartados anteriores, el diseño del juego pretende eliminar completamente las reglas especiales que afectan a la partida de forma global, en pos de implementar habilidades en las cartas que sean las encargadas de dotar al juego de mayor complejidad y frescura. A pesar de haber solo cuatro habilidades diseñadas por ahora, se posee una gran cantidad de habilidades esperando a ser testeadas para luego implementarlas en el juego. A continuación se comentan las habilidades diseñadas y su motivación:

- **Robar X:** Esta habilidad proporciona al jugador la capacidad de robar el número de cartas determinado por el valor "X". La habilidad entra en efecto cuando la carta que la posee es jugada, y las cartas se roban inmediatamente. La motivación de esta habilidad es la de proporcionar a los jugadores la opción de utilizar jugadas en principio menos rentables en pos de conseguir una ventaja de cartas en mano que pueda ser relevante en un futuro. Añade un componente estratégico muy relevante con una habilidad sencilla.
- **Defensor:** Esta habilidad se puede considerar una característica negativa o "pega". Una carta con la habilidad "Defensor" no puede voltear cartas rivales de ninguna forma. Las cartas con defensor suelen tener, a cambio, unos atributos mayores y/o un valor de carta menor. La motivación de esta habilidad es la de dotar de otro aspecto estratégico tanto a partidas como a construcción de mazo. Al tener valores normalmente reducidos y/o atributos elevados, permite el uso de cartas muy eficaces defensivamente hablando y con un coste de requisito de construcción de mazo menor.
- **Fuerza:** Esta habilidad otorga la capacidad de voltear cartas rivales en caso de poseer un atributo mayor o igual al del enemigo, en lugar de voltear dicha carta solamente en caso de ser mayor. La motivación de esta habilidad es la de conseguir cartas de uso prioritariamente ofensivo, de forma similar al caso de las cartas con la habilidad "Defensor" y su motivación defensiva.
- **Eco:** Esta es la primera de las habilidades diseñadas que contiene una mecánica compleja y con la capacidad de cambiar el rumbo de la partida desmesuradamente. Si una carta con "Eco" vence a una o más cartas rivales, cada carta vencida realiza un "ataque" a otra carta adyacente, pudiendo voltear otra carta enemiga ese mismo turno. Es decir, por cada carta vencida

(volteada) se elegirá una carta enemiga del tablero que esté en contacto con dicha carta vencida, y será volteada si su atributo es menor al de la carta vencida. No se aplicarán habilidades de cartas de esta forma, es decir, las cartas vencidas no podrán utilizar “Fuerza” ni “Robar”, y de la misma forma una carta vencida con la habilidad “Defensor” podrá utilizarse ofensivamente. A cambio de la capacidad de cambiar partidas otorgada por esta habilidad, las cartas que la poseen tienden a tener atributos extremadamente bajos, dificultando enormemente la tarea de voltear una carta enemiga, pero maximizando el resultado en caso de conseguirlo. Debido al delicado equilibrio de estas cartas, la regla no se ha implementado en la primera versión del juego a falta de un testeo mayor.

2.2.5 Problemática del diseño

Las principales dificultades observadas en el diseño y a las que se deberá hacer frente en próximas versiones u observar detalladamente en el proceso de testeo son las siguientes:

- Posible desequilibrio entre el primer y segundo jugador.
- Posible desequilibrio de las cartas con la regla “Eco”.
- Problema de representación del control de las cartas. Para saber si una carta en mesa pertenece a uno u otro jugador hay que recurrir a métodos externos como el uso de fundas de color u objetos diferenciadores en caso de juego físico, o en el caso del software cambiar el color de la fuente o añadir el nombre del controlador a cada carta.
- Incomodidad al tener que observar el tablero desde el mismo ángulo. Éste problema solo se encuentra en el formato de juego físico.

Jugador Inteligente

En este apartado se definirá el concepto de "Jugador inteligente" utilizado en el documento y se comentarán las estrategias de juego que se han propuesto aplicar, así como posibles mejoras y cómo implementarlas en un futuro.

3.1 Jugador Inteligente como Agente Inteligente

En este apartado se comenzará definiendo "Jugador inteligente" como una entidad capaz de percibir su entorno y responder ante él, tomando elecciones capaces de maximizar el resultado esperado. Ésta definición está sacada del término "Agente inteligente", utilizado por algunos autores para referirse concretamente a una entidad física, más que virtual, que percibe el entorno mediante el uso de sensores y responde mediante actuadores. Por lo tanto, para definir al *Jugador inteligente* sería más adecuado el uso de los a veces llamados Agentes Inteligentes Abstractos (AIA), para diferenciarlos de sus implementaciones físicas. A pesar de la falta de consenso, *Franklin* y *Graesser* [94] argumentan que una definición unificada de agente inteligente podría ser la siguiente:

> *"Un agente es un sistema computacional capaz de actuar de manera autónoma para satisfacer sus objetivos y metas, mientras se encuentra situado persistentemente en su medio ambiente."*

A pesar de ser ésta una definición muy abstracta de Agente inteligente, mostraría el propósito del agente sin entrar en definiciones más complejas como la de "inteligencia" o "racional".

Para el problema que se aborda en este documento es útil recalcar una de las características de un agente inteligente, concretamente, que la efectividad de un agente se calcula en función del éxito esperado y de su capacidad de percepción. Es decir, la efectividad está delimitada por la cantidad de información accesible y por la medida de desempeño establecida como estándar de éxito. En este sentido es importante decir que, el *Jugador Inteligente* a implementar, ni tiene acceso a toda la información del entorno (incertidumbre) ni pretende hallar un resultado óptimo para cada jugada. En la figura 6 se puede observar el comportamiento estándar de un agente inteligente.

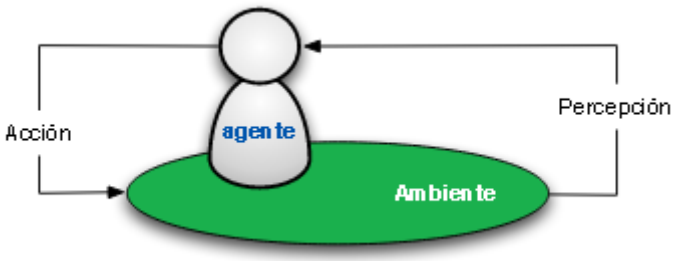

Figura 6: Comportamiento de un agente inteligente

3.2 Teoría de la decisión: Riesgo e incertidumbre

Una vez definido el término "Jugador Inteligente", se procede a describir el estado de la cuestión en lo referente a toma de decisiones y sistemas de evaluación en juegos.

Uno de los factores clave a la hora de definir el *Jugador Inteligente*, y muy relevante a la hora de diseñar una función de evaluación eficiente, es la existencia y grado de incertidumbre en el entorno. Se define la teoría de la decisión con incertidumbre o riesgo como una teoría en la que se analizan decisiones con cierta aleatoriedad o incertidumbre en los resultados, de modo que las consecuencias de una decisión no están determinadas de antemano. La distinción entre riesgo e incertidumbre se establece en 1921 en la obra *Risk, Uncertainity and Profit*, por *F. Knight* [94]. En ella, se define riesgo como una situación en la que no hay certeza alguna sobre el resultado de una decisión, pero hay al menos una estimación de las probabilidades de los distintos resultados alternativos (distribución probabilística conocida). En los procesos de decisión con incertidumbre, sin embargo, el decisor conoce los posibles resultados, aunque no hay una estimación de la probabilidad de que suceda cada uno de ellos.

Otro de los bloques estudiados en teoría de la decisión son los enfocados a decisiones multicriterio, en las que si bien las consecuencias de una decisión están definidas, no se puede obtener una valoración objetiva, dado que hay varias metas en conflicto.

El campo o estudio quizá más cercano al problema tratado en este documento, sería el campo denominado como *Teoría de juegos*. La teoría de juegos estudia las decisiones que no dependen únicamente de la decisión adoptada, sino también de la que tomen los demás jugadores.

3.3 Teoría de juegos

La teoría de juegos es un área de las matemáticas dedicada a estudiar y optimizar las posibles interacciones y toma de decisiones surgidas en los problemas de incentivos llamados "juegos". Aunque la teoría de juegos fue ideada en un principio como herramienta para entender el comportamiento de la economía, su uso se ha extendido a varios campos de diversa índole, como la psicología, la biología o el sector que nos atañe: la inteligencia artificial aplicada a juegos. Una situación de teoría de juegos se define como aquella en la que hay un conflicto y una mayor o menor oposición entre los intereses de los decisores (jugadores).

A pesar de que en un principio podría parecer similar a la teoría de la decisión, las decisiones en teoría de juegos suceden en entornos donde el coste y el beneficio no están prefijados de antemano. Esto resulta clave a la hora de estudiar la viabilidad de los diferentes criterios y a la hora de crear una función de evaluación para el juego, dado que la forma de evaluar la calidad de una jugada es meramente especulativa, y además depende de las decisiones tomadas por un ente ajeno.

Los juegos pueden ser clasificados atendiendo a varios criterios:

- Número de jugadores.
- Número de estrategias de los jugadores.
- Por su evolución en el tiempo (estáticos o dinámicos). Los juegos dinámicos son aquellos en los que en el transcurso del juego existe una ganancia de información por parte del jugador.
- Por el grado de cooperación entre jugadores.
- Por el grado de riqueza del conjunto de jugadores. Se define como suma constante vs suma no constante, en el que la suma constante sería la situación en la que siempre hay una cantidad fija de recursos a repartir, y el problema recae en la forma de repartir la riqueza. Además, se denomina "suma nula" al problema en el que hay cero riqueza a repartir, es decir, en el que la ganancia de un jugador es pagada directamente por el rival.
- Por la cantidad de información adquirida durante el juego. Se define como información perfecta si se tiene acceso completo y correcto a todo el estado del juego.

3.4 Teoría de juegos aplicada al Jugador Inteligente

A la hora de aplicar la teoría mostrada en las secciones anteriores al Jugador Inteligente, se podría resumir el problema a tratar como la combinación de varias características:

1. Dos jugadores: Aunque bastante obvia, es una característica clave, dado que define la necesidad de tener en cuenta a un ente ajeno que afecta al entorno.
2. Incertidumbre: Aunque se conocen los posibles resultados de una partida, no se puede saber, a priori, cuál ocurrirá.
3. Multicriterio: A la hora de realizar una jugada, se tienen en cuenta varios objetivos que entran en conflicto entre ellos (por ejemplo, asegurar una carta propia vs conseguir una carta rival).
4. Estático/Dinámico: Aunque se podría extraer información de los intereses y posibilidades del rival según avanza la partida, el grado de especulación al respecto sería muy elevado y no solo influiría poco en las decisiones a tomar, si no que el análisis podría ser totalmente erróneo el siguiente turno. Por ejemplo, si el jugador asume que el rival no tiene cartas con un valor izquierdo elevado dado que no ha intentado vencer a su carta con un valor derecho bajo, sería probable que el rival robara una carta con un valor izquierdo medio o elevado en su próximo turno, con lo cual la poca e incierta información recibida sería errónea. Por ello, se tratará el problema desde un punto de vista estático.
5. Información perfecta: Se puede tener acceso a todas las jugadas realizadas durante la partida.
6. Suma nula: En el juego, la riqueza a conseguir es una cantidad fija (el control de cada una de las casillas con una carta propia), y su adquisición repercute directamente en una pérdida equivalente para el jugador. Básicamente, esto concluye que una jugada puede ser igualmente buena por asegurar la posición del jugador como por desestabilizar los intereses del contrario.
7. Criterio pesimista: La forma de buscar estrategias será similar a un criterio pesimista, dado que para cada jugada se espera que el jugador rival realice su mejor jugada (lo peor para el decisor actual). El criterio pesimista es adecuado para situaciones en las que el grado de incertidumbre es muy elevado.

3.5 Objetivos, mecánica y estrategia

Una vez definido el problema a tratar dentro de un marco teórico, se plantearan los objetivos concretos que se consideran factibles para este proyecto y que se espera que la inteligencia artificial pueda conseguir. Al mismo tiempo, se estudiarán las posibles estrategias para la mecánicas de juego dadas, y se sacará una conclusión sobre qué tipo Jugador Inteligente implementar y cómo hacerlo.

3.5.1 Objetivos:

Las metas a lograr son:

- **Limitado:** No se pretende conseguir una inteligencia imbatible, sino una experiencia parecida a la de enfrentarse a un jugador de nivel medio, principalmente por la imposibilidad de conseguir unas jugadas óptimas dado el marco en el que se desarrolla el juego.
- **Evaluación de la situación:** Es importante no tener un sistema de reglas reactivo totalmente prefijado, si no poder evaluar cada situación y asignar más o menos relevancia a las jugadas dependiendo de las circunstancias.
- **Humano:** Se pretende emular el flujo de pensamiento de un jugador humano intentando valorar las opciones con un sistema de “pros” y “contras”.
- **Manejo de la incertidumbre:** Capaz de tener en cuenta en mayor o menor medida las posibles jugadas del rival, juzgando la posibilidad de que una jugada tenga un mayor o menor éxito a pesar de la falta de información.
- **Planificación:** Se requiere un mínimo de anticipación, no solo centrándose en conseguir la mayor ventaja el turno actual, si no estableciendo unas bases sólidas para turnos venideros. Para ello será importante tener en mente un criterio pesimista, en el que se tiene muy en cuenta la posibilidad del rival de tener una jugada que cambie el rumbo de la partida. Esto se realizará mediante un estudio aproximado de probabilidades, en el que la función de evaluación no solo estará influida por el resultado de la jugada, si no por las posibles consecuencias o “contra-ataques” del rival.

3.5.2 Mecánica y estrategia

Como se comentó en apartados anteriores, una de las principales metas del diseño era conseguir un juego equilibrado donde cada carta pudiera tener su función y utilidad sin necesidad de ser considerada de la misma calidad. Tras un testeo intensivo, se demostró que las cartas con valores medios bajos podían llegar a ser tan útiles como otras cartas con valores más altos. Esto se debe a la naturaleza misma del juego, en la que el hecho de que el rival gane el control de una carta no significa que esté perdida, y la posibilidad de jugar con este concepto es la novedad frente a otros juegos del mismo género: las cartas con mayor valor no siempre son mejores, por lo tanto, las jugadas que podrían parecer menos intuitivas para un jugador novato podrían ser las más eficientes.

El reto, por lo tanto, es conseguir una función de evaluación que represente este concepto tan inusual, que sepa identificar cuándo una carta con atributos mayores realmente tiene un mayor valor. Para ello, se empezará definiendo las posibles situaciones intentando descubrir, desde el punto de vista de un jugador humano, cuáles serían las posibles tácticas a aplicar.

Para comenzar a afrontar este problema, primero hay que comprender bien las mecánicas que maximizan las posibilidades de victoria. Siendo un juego que cuenta con tan pocas jugadas (5 y 4, dependiendo del jugador inicial) y en el cual el jugador que controle más cartas se alzará con la victoria, cada carta debe conseguir un objetivo muy claro: intentar decantar y maximizar la diferencia de cartas controladas a tu favor (juego de suma nula). Ésta sería una tarea muy simple si se ignorara por completo la planificación de un estado favorable para jugadas futuras, ya que solo habría que hacer la jugada que más cartas volteara en el turno actual. Sin embargo, aunque una parte extremadamente relevante es el número de cartas volteadas, hay situaciones en las que puede no salir rentable, o simplemente hay varias jugadas que consiguen el mismo número de cartas pero algunas de las jugadas son mucho mejores que las otras. Por lo tanto, la evaluación de la jugada actual no solo pasa por maximizar el resultado actual, si no en mantener una defensa sólida ante las futuras e inciertas jugadas del rival.

En resumen, se pueden identificar varias directrices que se complementan entre sí y que hay que valorar en función de la situación. Estas podrían agruparse en cinco grupos:

- **Maximizar el número de cartas ganadas:** Está claro que sigue siendo una directriz prioritaria, pues el número de turnos es muy limitado y las posibilidades de voltear más de una carta por turno escasas.

- **Seguridad de la carta jugada:** Una jugada que proporcione una ganancia de +1 carta pero que al turno siguiente signifique con seguridad la pérdida de la carta jugada, y muchas veces con facilidad y flexibilidad para el rival, pierde gran parte del valor conseguido. Otra forma de afrontar la seguridad de la carta es no teniendo en cuenta solo la posibilidad de perderla, sino más bien la probabilidad de recuperarla en un futuro próximo.

- **Seguridad de tus cartas afectadas:** No solo hay que tener en cuenta la carta jugada y las que se ganen con dicho movimiento, sino que es posible que una jugada a priori peor asegure el control de otras cartas que se podrían haber perdido fácilmente. Esto es algo muy común cuando con una jugada se "aísla" una carta, de forma que ya no posee casillas adyacentes libres y nunca podrá ser recuperada por el rival. Esto se aplica también a la carta que se haya volteado/conseguido este turno, el valor de la jugada es mucho mayor si dicha carta queda "cerrada" bajo el control del jugador. Sin embargo, del mismo modo que una jugada puede asegurar las cartas de casillas adyacentes, también las puede dejar en una situación precaria.

- **Pérdida de seguridad de las cartas del rival:** Del mismo modo que con una jugada se intenta maximizar la seguridad de las propias cartas, es apropiado pensar que dejar al rival en una situación de peligro, a poder ser en más de una carta, será ventajoso. Sin embargo se le da menos importancia en la función de evaluación, dado que el siguiente jugador en mover es el rival y se tiene un grado de incertidumbre muy elevado sobre las posibilidades de éste. En resumen y siguiendo el criterio pesimista, es mejor asegurar cartas tuyas que dejar las cartas del rival en una situación precaria.

- **Otras:** Hay una serie de detalles, a priori de menor relevancia, que pueden decantar la balanza en el caso de tener varias jugadas de evaluación similar. Por ejemplo las habilidades de las cartas, como "Robar", añaden cierto atractivo para el jugador humano a la hora de escoger una jugada, aunque es difícil de cuantificar.

Para algunos de los puntos, hay que tener muy en cuenta la llamada "regla de los lados libres", que se tratará a fondo en el próximo apartado y que constituye una de las bases de la complejidad estratégica de un juego a priori simple.

3.5.3 La regla de los lados libres

En el apartado anterior se han definido las estrategias básicas para optimizar la situación del tablero tras una jugada, teniendo en cuenta tanto el estado actual como el futuro incierto. Pero, aunque haya conceptos fáciles de valorar como el volteo de una carta, ¿de qué forma se pueden cuantificar conceptos tales como la "seguridad" de una carta propia o la "precariedad" de una carta rival? Para poder empezar a pensar en un cálculo, se define la "seguridad" de una carta como la probabilidad P de que el rival no pueda voltearla en un futuro próximo. Por extensión, la precariedad de una carta se definiría como (1-P), la probabilidad de que el rival pueda voltearla.

Antes de empezar a hablar de probabilidades como forma de cálculo para modelar la función de evaluación, cabe destacar el uso laxo que se hará de dichas probabilidades, utilizándose como **estimaciones de probabilidad** o **valores de incertidumbre**.

La probabilidad de que una carta que tiene un lado libre sea ganada y por lo tanto conservada por el rival es relativamente trivial de calcular si se tiene acceso al número de cartas con valores mayores que el atributo "libre", ya sea con un cálculo exacto o aproximado, se podría estimar la probabilidad de perder dicha carta el próximo turno dado el número de cartas en la mano del rival. Para ejemplificar el cálculo de dicha probabilidad, se parte del caso base en el que el rival tenga una sola carta en mano. Asumiendo, por ejemplo, que hemos dejado un valor libre con un 70% de valores mayores (P = 0.3, 1-P = 0.7), el rival tendría un 70% de probabilidad de poder vencer. Si el rival tuviera dos cartas en mano, la probabilidad de que pudiera vencer sería "P = 0.7 + 0.7 – 0.7*0.7", o lo que es lo mismo, "P = 1 – (0.3 * 0.3)", es decir, que la probabilidad de que el jugador inicial conserve su carta va disminuyendo a razón de $\mathbf{1-P^n}$, siendo P^n la probabilidad P elevada al número de cartas en la mano del jugador rival. Por lo tanto, es lógico pensar en una correlación entre el valor del atributo libre y las posibilidades de conservar la carta, siendo éstas mayores a mayor valor del atributo.

La problemática, sin embargo, adquiere otra magnitud cuando el número de lados libres es mayor a uno. ¿Qué probabilidad hay de que el rival pueda superar al menos el valor más bajo de la carta? Y en caso de superarlo, ¿Qué posibilidades tendría el jugador actual de recuperarla en su próximo turno? ¿Hay una correlación tan obvia como en el caso anterior? El resultado de ésta cuestión puede parecer poco intuitivo o incluso paradójico, pero su comprensión es muy importante para el correcto aprendizaje del juego y la correcta interpretación del Jugador Inteligente: NO hay una correlación directa que asegure mayor probabilidad de éxito a 1-2 valores altos que a los correspondientes valores bajos. De hecho, probabilísticamente hablando y sin tener en cuenta variables externas, es mucho más rentable dejar dos valores muy bajos libres (por ejemplo, 0-0) que una dupla de valores medios-altos (por ejemplo, 5-9). El razonamiento no solo resulta sencillo, si no que el cálculo de las probabilidades de conservar la carta al final de la partida lo secunda. Una carta de un propietario X con dos valores muy bajos será muy fácil de ganar para el rival, pero en el momento en el que lo haga, el propietario X podrá recuperar el control de dicha carta con total facilidad y flexibilidad, no solo recuperando el control si no "cerrando" dicha carta y negando la posibilidad de que el rival pudiera recuperarla. Analizando la casuística, se revela pues que los mejores casos para dos lados libres son dos: dos números muy altos que aseguren el control de dicha carta (preferiblemente 9-9), prácticamente imposibles de superar, o dos números muy bajos que aseguren una recuperación sencilla de la carta. Así mismo, dos valores libres iguales o muy similares es también una posición ventajosa, sobre todo si se posee ventaja de cartas en mano (el jugador tiene acceso en todo momento a dicha información, pudiendo ver cuantas cartas tiene el rival en mano), dado que el rival tendrá las mismas facilidades/dificultades para ganar la carta que el propietario inicial para recuperarla, con la diferencia de que una vez recuperada, esta carta no podrá volver a ser vencida al no tener más lados abiertos. De este análisis se extrapola también que la peor jugada sería dejar un número muy bajo y un número muy alto (0-9 se llevaría la peor puntuación), dado que es muy fácil para el rival ganar la carta, y el propietario inicial tendría una dificultad extrema a la hora de recuperarla.

El análisis para la situación de dos lados libres se convierte entonces en una combinación de dos posibilidades favorables para el jugador activo: primero, la posibilidad de que el rival no pueda vencer a ninguno de los valores, y segundo, que el rival gane la carta pero el jugador pueda recuperarla. Cabe hacer hincapié en que, debido a factores difíciles de

calcular, **no se pretende conseguir la probabilidad exacta** de que una carta con dos valores abiertos sea conservada al final de la partida. Por ejemplo, se da la situación en la que el rival no intente ganar la carta en el turno siguiente, pero por la imposibilidad de calcular una probabilidad aproximada para dicho suceso y por el relativo poco impacto de dicha elección (si no elige atacar a la carta recién puesta este turno, normalmente tendrá que hacerlo más adelante, y el número de cartas en mano no variará a no ser que use cartas con la habilidad "ROBAR"), se ha decidido desechar dicha posibilidad y otros factores menos relevantes a la hora de realizar el cálculo. Precisamente por esta razón y otras similares, es importante que el comportamiento resultante del Jugador Inteligente sea de mentalidad pesimista, dado que hay factores inciertos que no se han podido cuantificar.

Una vez claras las ventajas e inconvenientes de dejar dos lados libres, se intentará usar dicho análisis para extrapolar una posible evaluación a la situación de dejar tres lados libres. En este caso, la complejidad se dispara sobremanera, dadas las muchas variables y posibilidades ofrecidas al rival, todas plagadas de incertidumbre difícilmente calculable.

En principio, para la situación de tres lados libres se pensó que la forma más acertada de realizar la evaluación era la siguiente: En caso de que el rival ganara la carta (se calcularía la probabilidad de que ocurriera para cada uno de los valores libres), ¿en qué situación quedaría dicha carta para él? ¿Sería una carta fácil o difícil de conservar? Básicamente, se deduciría que tras ganar la carta el rival la dejaría en una situación de "dos lados libres", con lo cual la forma de maximizar la evaluación sería MINIMIZAR la probabilidad de que el rival conservara su carta, es decir, maximizar (1-P), donde P sería la evaluación de la situación de "dos lados libres" para la carta del rival. Dicho de forma más intuitiva, cuantos más problemas tuviera el rival para conservar la carta que acaba de ganar, mejor sería para el jugador. A esto hay que sumar la probabilidad de que el rival no pudiera vencer ante ninguno de los tres valores libres, lo cual sería harto improbable para la gran mayoría de situaciones, pero habría que tener en cuenta dicha posibilidad.

Tras implementar un algoritmo para la situación de tres lados libres basado en las premisas anteriores, se probó en multitud de partidas, y no solo daba valores extremadamente bajos por norma general, sino que además en situaciones concretas valoraba de forma positiva jugadas que serían realmente malas. No se sabe con certeza si la falta de éxito se debe a una mala implementación, al intento de conseguir tener en cuenta la posibilidad de que el rival no pudiera vencer a ninguno de los tres valores, o

la más probable, al intento de dar relevancia (aunque diferente, dependiendo de la situación) a cada una de las posibilidades del rival (recuperar la carta por el lado mayor, en lugar de tener en cuenta solo el caso en el que recupera la carta por valor menor, o incluso el medio). Probablemente el fallo fuera al intentar tener en cuenta todas las opciones, cosa que ni siquiera en el algoritmo de dos lados libres sucede (no se valora que el jugador rival recupere la carta por el lado menos rentable, aunque sea la única forma que tiene de recuperar la carta, lo cual obviamente es una opción), pero el caso era que no se valoraban correctamente las opciones, o al menos se valoraban de una forma muy diferente a lo pretendido al implementar el algoritmo.

Tras probar el algoritmo ya comentado para valorar la situación de tres lados libres, se buscaron varias formas de valorar la jugada lo más acertadamente posible. Al final, se optó por simplificar el algoritmo sobremanera, centrándose en un patrón claro de actuación: la probabilidad de que el rival venza a los dos valores extremos (el más bajo para ganarla por primera vez, y posteriormente el más alto para recuperarla) y la probabilidad de que el jugador actual venza al valor medio. Esto desemboca en una fórmula que se explicará más adelante, en el apartado de implementación.

Cabe decir que, si calcular la probabilidad de conservar una carta con tres lados libres dada la incertidumbre de no saber por qué lado va a realizar el rival su envite ya era una tarea arduo compleja, calcular las posibles variantes de situar una carta en el centro en las primeras fases de la partida, es decir, dejar cuatro lados libres, es prácticamente imposible con el sistema actual. Además de la dificultad de los cálculos, dicha jugada es tan impredecible que se desaconseja su uso prácticamente en cualquier situación.

3.5.4 Implementación

Una vez establecidas las directrices que guiarán el modus operandi del Jugador Inteligente, se procede a explicar las decisiones tomadas a la hora de implementarlo. En primer lugar se realizará una exposición de la mecánica general de evaluación utilizada por el agente y de las razones que han llevado a hacer esa elección. A continuación, se plasmarán los diferentes objetivos explicados en las secciones anteriores, y se implementará una forma de valorar cuáles perseguir con cada acción

(estrategia multicriterio), por ejemplo, si priorizar la seguridad de una o dos cartas propias, voltear una carta rival o dejar varias cartas rivales en situación precaria. Para terminar, se propondrá de forma breve una serie de mejoras o implementaciones que se han tenido en cuenta.

En un principio, se valoró la posibilidad de realizar solamente un agente reactivo simple, basado en un sistema de reglas del tipo if-then que valorara cada jugada en función de una prioridad asignada previamente a cada uno de los objetivos. Aunque la relación complejidad de la IA – eficiencia del agente era notable, había situaciones en las que realmente se necesitaba un análisis más profundo de qué jugadas eran más valiosas, y la sensación era, si acaso, la de un jugador novel con ciertos conocimientos tácticos. Por esa misma razón, se decidió invertir más tiempo en la IA y conseguir un Jugador Inteligente más humano, basado en evaluaciones positivas y negativas dependiendo de su percepción del estado del tablero, que reflejara la reacción y método de elección de un jugador medio ante las situaciones comunes. El resultado fue el diseño de una función de evaluación que valorara, para cada jugada posible en dicho turno, cada uno de los objetivos tratados en puntos anteriores, que llamara a sub-funciones para calcular cuan efectiva era la jugada a la hora de perseguir esa meta (por ejemplo, la meta de que la carta jugada quede en una posición segura) y aplicara un modificador para ajustar el peso de dicho objetivo en el resultado final. Tras revisar cada uno de los objetivos, se mostraría un resultado numérico final positivo o negativo en función de la efectividad general de la jugada. Como se puede ver, esto refleja el método de valoración humano: pensar en las ventajas y desventajas de una jugada en función de lo que se espera conseguir, valorarlas y compararlas con otras posibles jugadas.

A continuación se hará un resumen en pseudocódigo del bucle general que se aplica cada turno del agente inteligente. Más tarde se detallará cada una de las funciones:

- Cada jugada (una iteración del bucle) es una dupla carta-casilla, y se aplica para carta en la mano y cada posible casilla donde pueda jugarse.
- Siendo “tablero” la situación inicial del tablero antes de jugar ninguna carta, ”tableroPosterior” correspondería a la situación final del tablero tras realizar la jugada que se está valorando.
- “jugActivo” se refiere al jugador que hace la jugada, y “rival” al oponente.

```
While (jugActivo.cartasEnMano)
        For i= 0; i<numJugadasPosibles
                Jugada = New Jugada(i);
                Evaluación = Jugada.evaluarJugada();
                listaEvaluaciones.add(Evaluación)
        End for
End while


evaluarJugada(){
        evaluación = 0;
        tableroPosterior = tablero.voltearCartas();
        evaluación = evaluación + calcularNumeroCartasVolteadas();
        evaluación = evaluación + bonusHabilidadesUtiles();
        evaluación = evaluación +modificadorSeguridadCartaJugada();
        evaluación = evaluación +modificadorSeguridadCartasAdyacentes();
        evaluación = evaluación +modificadorPrecariedadCartasRivales();
        return evaluación;}
```

- **Voltear cartas:** Esta función se encarga de simular cual sería el resultado final de realizar la jugada que se está calculando. Para ello, se voltean todas las cartas que dicha jugada conseguiría, y el estado se guarda en tableroPosterior.

- **Cálculo de número de cartas volteadas:** Esta función se encarga de comparar el tablero inicial y el tablero posterior tras realizar la jugada, y contabilizar así el número de cartas ganadas. Por comodidad y eficiencia, en el bucle del programa se realiza al mismo tiempo que se voltean las cartas. El resultado del cálculo se establecerá como la *puntuación base* a la cual se irán añadiendo el resto de evaluaciones ponderadas. El resultado del cálculo es igual al número de cartas volteadas, es decir,

cero cartas ganadas sería un valor inicial de evaluación de 0, dos cartas volteadas sería un valor igual a 2, etc.

- ***Bonus* por habilidades útiles:** En esta función se realizan dos valoraciones, una para la habilidad "Robar" y otra para la habilidad "Defensor".
  En el caso de "Robar", a la función de evaluación se le añadirá una puntuación de 0.05 por cada carta robada gracias a la jugada realizada. Además, según avance la partida, el *bonus* recibido irá decreciendo de forma lineal, dado que la importancia de tener ventaja de cartas en mano es mucho mayor en las primeras fases de la partida, cuando aún quedan muchas jugadas por hacer, y decrece a medida que van pasando los turnos. El valor de 0.05 por carta no es la conclusión de ningún cálculo probabilístico, sino de un análisis mediante ensayo y error de la relevancia de la habilidad, y de la comparación con otros *bonus* recibidos por metas diferentes. La fórmula final sería:

  *Bonus Robar= 0.05 * valorRobar * (1 – (numCartasTablero/10))*

  En el caso de la habilidad "Defensor", el *bonus* recibido solo es un *tie-breaker*. Es decir, en caso de no haber ninguna jugada ofensiva y tener la opción de jugar dos cartas defensivas con los mismos atributos, pero una con "defensor" y la otra sin, es más rentable utilizar la que tiene defensor, dado que la otra carta podría ser útil en algún otro turno en el que se necesitara una fuerza ofensiva que la carta con "Defensor" no podría cumplir. Por lo tanto, se aplica un *bonus* de 0,01 para que la fórmula priorice ligeramente a cartas con la habilidad "Defensor". La fórmula para defensor y la fórmula final se ven a continuación:

  Bonus Defensor= 0.01 (if Defensor)
  Bonus Total = BonusRobar + BonusDefensor

- **Seguridad de la carta jugada:** Este es el método que marca la diferencia entre un Jugador Inteligente simple y uno avanzado. La seguridad de la carta recientemente jugada es clave, dado que será el principal cambio aplicado al estado actual de la partida y el más susceptible a posibles reacciones del rival. El gran rango que abarca las posibles jugadas hace de esta meta un problema muy marcado por el grado de incertidumbre. Al ser la incertidumbre directamente proporcional al grado de amenaza y

posibles opciones del rival, uno de los factores claves para cuantificar el riesgo es el número de cartas en mano del rival. El segundo factor clave, como se puede deducir, es la calidad (en forma de valor de los atributos libres) de la carta jugada. Así pues, teniendo claros los dos factores que determinarán la probabilidad de que el rival consiga la carta jugada, se procede a mostrar los pasos requeridos para definir la fórmula:

- Primero, se ha realizado un estudio sobre la base de datos que indica el número de cartas superiores a cada uno de los posibles valores de un atributo, siendo estos [0-9].
- Posteriormente, se convirtió dicha información en datos porcentuales, con lo cual dado un valor X se podría saber la probabilidad de que otro valor aleatorio de la base de datos fuera mayor.
- Concretamente, el array utilizado para los cálculos muestra la probabilidad, en una escala de 0-1, de que UNA carta del rival no supere un valor dado. El array es el siguiente, con la posición 0 mostrando la probabilidad para el valor 0, etc:

  P = {0.1625, 0.25625, 0.35, 0.50625, 0.575, 0.7125, 0.8125, 0.9125, 0.9625, 1}
- Una vez obtenida dicha información, se podría calcular la probabilidad aproximada de que el rival no tuviera NINGUNA carta que pudiera voltear la carta el siguiente turno. Esta se calcularía encadenando "n" veces la probabilidad de que una carta no supere ese valor, donde *n* es el tamaño de mano rival. Por lo tanto, la probabilidad de que el rival no tuviera ninguna carta que superara a la recién puesta sería:

  $$Pn = P^n$$

  Siendo P la probabilidad obtenida anteriormente gracias al array de probabilidades. Hay que tener en cuenta que no es la probabilidad exacta (ni se pretende), dado que hay cierta dependencia entre sucesos: si una de las cartas del rival es una de las X cartas en la base de datos que pueden superar la carta jugada, la próxima carta de su mano no tendrá exactamente la misma probabilidad que la anterior (aunque muy similar en la mayoría de casos).

  Aunque pueda parecer ilógico que las probabilidades que resultan son muy bajas (según los cálculos, un número "7" al inicio de la partida tendría alrededor de un 40% de probabilidades de ser

vencido por el rival), la explicación puede resultar más obvia si se tienen en cuenta situaciones parecidas, como la *paradoja del cumpleaños*[9].

A partir de este momento, se referirá a esta probabilidad como *Probabilidad de mantener la carta*, abreviada como *Pm* para los cálculos.

- Cabe mencionar que la razón por la que no se hizo un estudio de probabilidad para cada uno de los cuatro valores de la carta por separado, fue que los datos hubieran dotado de un conocimiento muy poco humano y mucho más privilegiado a la inteligencia artificial. Lo que se pretendía más bien era obtener una medida aproximada "natural" de la probabilidad de que un valor pudiera ser vencido por otro valor. El nivel de detalle se cree suficiente como para no considerarlo información privilegiada, dado que cualquier ser humano puede pensar que aproximadamente el 50% de atributos serán iguales o mayores que "5", y en efecto se acercará mucho a la cifra exacta (50.225%), pero sería poco realista pensar que un ser humano tendría memorizado el número exacto de valores mayores a 5 en el lado derecho, izquierdo, superior e inferior por separado.

Una vez conseguida la medida probabilística *Pm*, el siguiente paso es mostrar su influencia para cada una de las posibilidades contempladas por la IA, siendo estas las mencionadas en el apartado anterior, *Regla de los lados libres*.

Así pues, comenzando por la situación en que la carta queda *cerrada*, es decir, sin ningún lado libre, la puntuación de la seguridad de la carta sería máxima, dado que no hay forma de perderla. El rango de puntuaciones es [-0.5, 0.5], con lo cual en este caso se sumaría un +0.5 a la función de evaluación. En este caso en concreto pasa algo similar al caso de una carta con defensor: se añadirá un *bonus* del tipo *tie-breaker* valorado en +0.01, para que se priorice dejar la carta sin lados abiertos por encima de dejar la carta con un 9 abierto, lo cual resultaría en un +0.51. El bonus es simplemente para desempatar, dado que las posibilidades de perder dicha carta con un 9 abierto son nimias, y solo factibles si el valor está en un lado determinado y ante una de las dos cartas con una puntuación de 9 en dicho lado y la habilidad fuerza.

[9] Paradoja del cumpleaños explicada

Siguiendo con la situación en la que la carta jugada deja un solo lado libre, la probabilidad de que el rival la supere es trivial: *Pm*. Al mostrarse la probabilidad en el rango [0-1], bastaría con restar 0.5 puntos al resultado obtenido en *Pm* para reflejar el sistema de "pros" y "contras", que abarca valores entre -0.5 y +0.5.

Sin embargo, la fórmula para dos lados libres no es ni tan sencilla ni tan exacta, dado que hay un grado de incertidumbre mucho más difícil de contemplar. Dados dos valores abiertos, el movimiento óptimo para el rival sería superar el valor más bajo, y dejar el valor más elevado para así dificultar la recuperación de la carta. La otra posibilidad, la de vencer al número más alto y dejar el menor abierto, también sería factible, aunque mucho peor. La existencia de dicha posibilidad es un problema, dado que sería imposible cuantificar el nivel táctico del rival (o resultaría mucho más complejo y requeriría un estudio previo mucho mayor), y sería muy complicado calcular la cantidad de veces que el rival tendría la posibilidad y se vería forzado o elegiría superar el mayor valor, en lugar del menor. Por la poca relevancia estratégica de esta posibilidad y la dificultad de cuantificarla, se ha decidido asumir el peor caso (criterio pesimista) a la hora de realizar los cálculos, es decir, calcular la probabilidad de que el oponente pueda superar el MENOR de los atributos y dejar el mayor atributo como barrera para la recuperación. Por lo tanto, la fórmula de evaluación final para dos lados libres sería:

$$P2 = 1 - ((1-Pm)*Pmmia)$$

donde *Pmmia* es la probabilidad de que el jugador actual pudiera recuperar la carta perdida, superando el valor mayor. Es decir, la fórmula calcularía la probabilidad de que el rival SI pudiera ganar al menor valor y el jugador actual NO pudiera recuperarla. La inversión de dicha probabilidad denotará la evaluación final, *P2*, de seguridad de la carta conseguida con la jugada. A dicha evaluación se le aplicará, de nuevo, el modificador de -0.5 para adecuar el peso al rango [-0.5, 0.5].

El último caso, el de tres lados libres, es por lo tanto el más complejo de calcular, aunque afortunadamente también el menos relevante de los casos expuestos, dado que las mejores jugadas tienden a ser las que dejan dos lados libres, un lado libre con un valor muy elevado, o la carta cerrada. Para cuantificar la calidad de la jugada, hay dos

posibilidades implementadas. Anteriormente se realizaba la suma probabilística de dos eventos: que el rival no pudiera superar ninguno de los valores libres (llamado *Pns*, el equivalente de *Pm* para los tres valores) y que el rival, tras superar uno de los valores, dejara la carta en una situación precaria. El primer factor, *Pns*, era sencillo de calcular. Para el segundo factor, bastaba con calcular la probabilidad de que el rival superara uno de los factores y lograra mantener la carta, que tendría dos valores abiertos. La probabilidad de que el rival mantuviera la carta se calculaba de igual forma que se calculaba para el jugador actual en el caso de dos lados libres. Cabe denotar que, de nuevo, al haber tres casos posibles y no poder tener en cuenta el nivel táctico del rival, a lo que se añade un aun mayor grado de incertidumbre, se asumía por sencillez y por ponerse en el peor caso que el rival intentaría realizar la jugada óptima. Por lo tanto, de las 3 probabilidades calculadas se usaba la más contraproducente para los intereses del agente inteligente, denotada como *PMrival* en la fórmula. Esto no solo asumía un alto nivel táctico del rival y un pesimismo útil, si no que hacía que la IA penalizara las jugadas con incertidumbre absurdamente elevada y priorizara las jugadas más seguras. Por lo tanto, la fórmula final era:

*Pns + PMrival – Pns*Pmrival*

a la cual se aplicaba también el valor modificador de -0.5 para corregir el rango.

Sin embargo, como ya se comentó anteriormente, la fórmula no era todo lo eficiente que debía ser. Por ello se buscaron varias posibilidades y se compararon entre ellas. Al final, la que demostró mejores resultados, fue sorprendentemente la más simple. Como ya se comentó, la fórmula de evaluación para tres lados libres consiste en evaluar de forma positiva la dificultad del rival por vencer al valor más alto y al más bajo, mientras que se evalúa de forma negativa la dificultad del jugador actual de vencer a la carta de nivel medio. La fórmula resultante sería:

Evaluacion = probabilidadNoMayor[valorMayor] - probabilidadNoMayor[valorMedio] + probabilidadNoMayor[valorMenor] + factorCorrector;

factorCorrector = (numCartasRival – numCartasJugador) /10

A esta fórmula se le ha aplicado un factor corrector arbitrario que disminuye el valor de la jugada en caso de que el rival tenga más cartas en mano que el jugador actual (0.1 por cada carta), dado que aumenta la incertidumbre. A pesar de que la fórmula parece extremadamente simple y muy concreta, se ha estudiado a fondo la casuística y proporciona un valor de evaluación adecuado en cada caso. Por ejemplo, una buena jugada sería dejar libres los valores (0, 0, 9), ya que el enemigo vencería fácilmente a uno de los ceros pero el jugador recuperaría la carta también fácilmente, al mismo tiempo que deja un valor alto abierto. En la fórmula, los dos ceros actuarían de valor menor y valor medio, dejando una muy buena puntuación al evaluar la probabilidad de perder un nueve (cuasi nula). Sin embargo, si el tercer valor fuera también un 0 la jugada sería muy mala, y así lo reflejaría la fórmula.

- **Seguridad de las cartas adyacentes:** Exactamente de la misma forma que se calcula la seguridad de la carta jugada, se calculará la seguridad de cada carta a la que haya afectado la jugada. Como ya se ha comentado anteriormente, una carta en una posición a priori segura podría acabar en una situación precaria por la jugada evaluada, y al contrario. Utilizando el mismo sistema de cálculo para la seguridad de la carta jugada se obtendrán todas las evaluaciones de seguridad de las cartas circundantes, a las cuales se les aplicará un modificador de 1/3. Dicho modificador es, como en el caso del aplicado a las cartas con "Robar", resultado del análisis de los datos obtenidos en numerosas partidas y el intento de ponderarlo y asemejarlo al razonamiento de un jugador humano. La fórmula que representa la seguridad de las cartas adyacentes es la siguiente:

  SegCartasAdyacentes = SegCartaAdyacenteA/3 + SegCartaAdyacenteB/3 + SegCartaAdyacenteI/3 + SegCartaAdyacenteD/3

  teniendo en cuenta que, en caso de no haber una carta adyacente en una de las cuatro direcciones, el factor de dicha dirección sería 0.

- **Precariedad de las cartas rivales:** El estado final de las cartas rivales circundantes tras realizar la jugada es de relevancia similar al punto anterior. Naturalmente, cerrar una carta del rival nunca es un factor positivo, pero dejarla en una situación incómoda si puede serlo. Se realizan exactamente los mismos cálculos de seguridad de la carta que en el punto anterior, se aplica también el modificador -0.5 seguido del

de 1/3, pero en este caso el resultado se resta a la función de evaluación final. La razón es que a mayor seguridad de la carta rival, peor jugada resultará para el jugador actual.

Una vez establecidas todas las evaluaciones meta por meta y añadidas a la evaluación, el resultante será el valor por el cual se compararán las duplas carta-casilla, denominadas *jugadas*.

3.5.5 Posibles mejoras

A la hora de diseñar el agente, se tuvieron en cuenta varias mejoras que se podrían haber implementado. Muchas de ellas se descartaron por suponer un gran aumento de la complejidad, tanto de implementación como del agente en sí, y otras se desecharon por la incertidumbre sobre su correcto funcionamiento.

- **Valores altos desperdiciados:** En contadas ocasiones, cuando dos jugadas obtienen una evaluación similar y se elige a la mejor valorada, se incurre en un error no muy común y de relevancia media, pero muy notorio: se utiliza una carta que podría haber resultado más útil en otro momento de la partida. Un claro ejemplo es cuando en el inicio de la partida se utiliza una carta con dos valores bajos en una esquina. Es sin duda una buena jugada, gracias a la regla de los lados libres, pero ¿y si los valores que han quedado inutilizados eran dos valores elevados? En las fases finales de la partida, dicha carta podría haber tenido una utilidad mucho mayor. El modificador que habría que aplicar a dichas cartas para no desaprovecharlas dependería de tantas variables que el resultado sería incluso contraproducente en un gran número de partidas, además de la complejidad de cálculos que requeriría para obtener un resultado medianamente fiable para un factor no tan relevante en una partida.

- **Casillas peligrosas:** Es difícil de concretar, pero hay veces que aunque una jugada pueda parecer muy rentable, a la larga podría resultar en estado del tablero con una sola debilidad, pero de relevancia crítica. Un ejemplo sería ir encadenando buenas jugadas en forma de dos lados libres de valor bajo, pero dejando una casilla concéntrica en la que, en cierto momento, el rival pondría una carta elevada y convertiría un conjunto de teóricamente buenas jugadas en una partida perdida. Según se acerca dicho momento, la casilla peligrosa iría quedando mejor

identificada, pero sería difícil de concretar cuál de los movimientos anteriores había influenciado más, y desde luego sería tarde para aplicar una evaluación negativa. Además, dichos momentos serían lo suficientemente poco frecuentes y complejos de implementar como para que no saliera rentable, y hay otras evaluaciones que en ocasiones colaboran a la hora de "cerrar" esas posibles casillas peligrosas. Un ejemplo es la meta de asegurar las cartas circundantes. Si existiera una casilla crítica en construcción, cada vez resultaría más rentable situar una carta en el centro e ir asegurando dichas cartas. Aun así, en versiones posteriores se podría implementar algún método que hiciera al agente más consistente en estas situaciones.

- **Modificadores positivos por estrategias a largo plazo:** Aunque requeriría un mayor grado de planificación y un mayor análisis de las cartas, se podría establecer una meta a largo plazo que fuera proporcionando modificadores positivos a las jugadas según fueran cumpliendo dicho objetivo. Un ejemplo sería si las cartas del agente tuvieran valores muy elevados en el lado superior. En este caso, sería ideal ir construyendo la partida de arriba hacia abajo, para que en las etapas finales las cartas del agente fueran clave a la hora de voltear las cartas, dado que estarían situadas en la parte superior del tablero. Con este ejemplo, se irían priorizando y premiando las jugadas en la parte superior del tablero. Dado el grado de planificación necesario, y sobre todo de complejidad a la hora de designar los modificadores para cada caso concreto, es posible que la implementación de dicha función no fuera óptima para éste agente, o la varianza de la efectividad fuera muy elevada. En cualquier caso, sería un objeto digno de estudio para versiones posteriores.

- **Mayor control de las cartas en mano:** Las cartas en mano proporcionan una información que en ocasiones puede declinar la balanza y que no se tiene en cuenta. Un ejemplo sería a la hora de calcular la probabilidad de recuperar una carta en caso de perderla. No es lo mismo calcular la probabilidad de que se pueda recuperar dicha carta que tener la certeza de que, en caso de perderla, habría una carta en tu mano capaz de recuperarla. No sería extremadamente difícil llevar un control de este tipo, y aumentaría el grado de planificación y el rendimiento general del agente. En la opinión del autor, éste sería el próximo paso a implementar en caso de haber una próxima versión del agente.

- **Posible visión "global" del estado del tablero:** En lugar de evaluar una jugada en función de la casilla donde se juega y sus efectos en casillas adyacentes, podría resultar más sencillo evaluar el tablero global tras la jugada. Esto representaría quizá alguna ventaja, además de en sencillez, en sectores como la identificación de zonas peligrosas, pero probablemente diera menor relevancia a la jugada actual, al no distinguirla de las jugadas previas. Es posible que una combinación de ambos sistemas fuera óptima. De nuevo, sería un objeto digno de estudio para versiones posteriores.

- **Diferente estrategia según el jugador inicial:** En la etapa de testeo, una de las conclusiones a las que se ha llegado es que el jugador inicial tiene una gran ventaja. Esto no es un gran inconveniente, en cuestión de equilibrado del juego, si el ganador se determina al mejor de X partidas y el jugador inicial va cambiando, en lugar de jugar una sola partida. Es un hecho que se observa en la mayoría de juegos del mercado, por ejemplo en *Magic: the Gathering* o *Hearthstone*, en los cuales a pesar de haber desarrollado un sistema para equilibrar dicha desventaja (ambos basados en otorgar más cartas al segundo jugador), el jugador inicial sigue teniendo ventaja.

  Sin embargo, la información obtenida tras el testeo no es tan relevante para el diseño del juego como lo es para el del Jugador Inteligente. Tras una gran cantidad de partidas, se ha observado que la IA se defiende cuasi perfectamente en situaciones en las que tiene ventaja (comenzando primero, por ejemplo), dado que la táctica de asumir que va a perder la carta y centrarse en recuperarla se adecúa sobremanera a esta situación. Sin embargo, la contraparte es que en caso de ir por detrás, el Jugador Inteligente sigue con la misma estrategia, que se ha demostrado es más adecuada para situaciones en las que va por delante, dado que no arriesga en ningún momento para sobreponerse a esa situación desventajosa. Por lo tanto, una de las mejoras más relevantes que se podrían implementar en siguientes versiones sería la de establecer dos estrategias diferentes, una para cada situación, y la estrategia implementada actualmente sería utilizada para el caso en el que se va por delante.

Desarrollo de la aplicación

En esta sección se analizará la aplicación desarrollada para el juego “The Rift”. Se realizará una pequeña introducción, en la cual se explicarán las funciones que deberá cumplir y se comentarán las elecciones realizadas. Posteriormente, se desglosará la aplicación y se mostrara su estructura e implementación en mayor detalle.

4.1 Introducción

La funcionalidad principal pretendida por la aplicación a desarrollar es la de proporcionar un prototipo que permita probar el juego ante una inteligencia artificial y ante otro jugador humano. Para ello, será necesario tanto el diseño e implementación de un jugador inteligente como un diseño distribuido de la aplicación, que permita a dos jugadores jugar desde terminales diferentes. El diseño del jugador inteligente se comentará en la sección designada para tal cometido, mientras que este apartado tratará de la implementación de la aplicación distribuida.

Como ya se ha comentado, se pretendía crear una aplicación que sirviera para probar de forma interactiva tanto el juego como la inteligencia artificial, por lo tanto no se tuvieron en cuenta aspectos secundarios tales como el aspecto gráfico o la creación de mazos personalizados. Sin embargo, sí que se consideró importante proporcionar ciertas facilidades y funcionalidades extra, como por ejemplo la posibilidad de juego remoto o el manejo de errores realizados por el jugador a la hora de introducir el movimiento deseado. Dichas funcionalidades secundarias también se comentarán, mediante una descripción a alto nivel, en el apartado correspondiente.

4.2 Arquitectura y descripción de la aplicación

La estructura de la aplicación, desde el punto de vista más abstracto, consta de tres módulos muy diferenciados, como se puede apreciar en la figura 7. Cada uno de estos módulos presenta una funcionalidad muy distinta, y se comunican entre ellos mediante *Sockets* para mantener la coherencia en la aplicación. En primer lugar, el módulo “Motor” se encarga de realizar todas las operaciones necesarias para el desarrollo de la partida. Los módulos “Client” representan a cada

uno de los (dos) jugadores de la partida, y se encargan de proporcionar las jugadas al motor para que desarrolle cada turno. El módulo “Server” es un intermediario, y se encarga de mantener la comunicación entre cliente y motor.

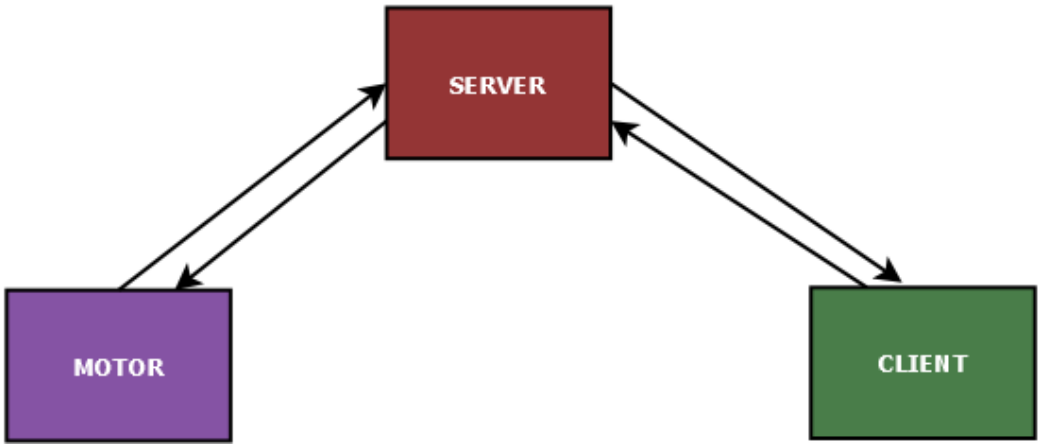

Figura 7: Arquitectura modular de la aplicación

A continuación se describe el funcionamiento a alto nivel de cada módulo, con las correspondientes entradas y salidas.

4.2.1 Motor

Es el núcleo de la aplicación, el módulo encargado de todas las operaciones relativas al desarrollo de la partida, exceptuando la elección de las jugadas, que se realiza en los módulos Client.

El flujo de ejecución del módulo Motor se podría resumir de la siguiente forma:

- Carga de la base de datos.
- Conexión mediante Sockets al servidor creado en el módulo Server.
- Petición de datos necesarios para la creación del perfil de cada jugador (Nombre y lista de cartas que componen el mazo).
- Creación del tablero. Iniciando partida.
- Comienzo del bucle de partida, consistiendo éste en una sucesión de nueve turnos. Cada turno se realizan las siguientes acciones:
    - Fase de robo. El jugador activo robará una carta.
    - Fase de jugar carta. El jugador activo jugará una carta. En caso de ser del tipo “Humano”, *Motor* conectará con *Server*, y éste a su

vez con el cliente adecuado, para solicitar la jugada a realizar (tras informar del estado actual del tablero y de su mano).
    - Fase de activación de habilidades. Las habilidades se activan previamente al volteo de cartas.
    - Fase de volteo. Se comprueban las cartas que han sido vencidas y se voltean.
    - Fase fin de turno. Se cambia el jugador activo.
- Al finalizar la partida (tablero lleno), comunicación del resultado a los jugadores (en caso de ser del tipo “Humano”).
- Dar la opción de volver a jugar una nueva partida con los mismos jugadores y mazos.
- Cerrar la conexión cuando no se quieran jugar más partidas.

Por lo tanto, el módulo Motor consta de las siguientes entradas y salidas (ordenadas cronológicamente), todas realizadas desde y hacia el módulo Server tras establecer la conexión. Se asume una partida con al menos un jugador Humano, para poder explicar las entradas y salidas relativas a la petición de jugadas:

- **Entrada:** Nombre del jugador 1.
- **Entrada:** Bucle de entradas, hasta que no se reciba el código de finalización. Cada una de las entradas corresponderá al nombre de una carta del mazo del jugador 1. Al recibir el código que marca el final del mazo, se procederá a reconstruir el mazo recibido y asignarlo al jugador 1.
- **Entrada:** Nombre del jugador 2.
- **Entrada:** Bucle de entradas para construir el mazo del jugador 2, de la misma forma que con el anterior jugador.
- **Bucle de partida**, consistente en 9 turnos. En cada uno de los turnos se realizan las entradas y salidas siguientes:
    - **Salida:** Codificación del estado del tablero actual. Será descodificado al llegar al módulo del jugador (Client).
    - **Salida:** Codificación del estado de la mano del jugador activo. Será descodificado al llegar al módulo del jugador (Client).
    - **Salida:** Petición de movimiento al jugador actual (enviado al intermediario “Server”, como siempre). Esta petición solo se realizará a jugadores humanos, no a jugadores del tipo IA (Jugador Inteligente) o Random (Movimientos aleatorios).
    - **Entrada:** Carta que el jugador activo quiere jugar.

    - **Entrada:** Posición X del tablero donde situar la carta.
    - **Entrada:** Posición Y del tablero donde situar la carta.
- **Salida:** Notificación de final de partida y del resultado a cada uno de los jugadores.

4.2.2 Client

El módulo *Client* representa a un jugador concreto, proporcionando las funcionalidades necesarias para que dicho jugador pueda iniciar una partida y elegir su acción cada turno. El módulo *Server* aceptará dos clientes, uno por cada jugador.

El flujo de ejecución del módulo en cuestión es el siguiente:

- Conexión al módulo *Server* mediante *Sockets*.
- Creación del perfil, consistiendo éste en un nombre identificativo introducido por el jugador y un mazo creado aleatoriamente. Para esta versión no se considera importante la posibilidad de personalizar el mazo.
- Envío del perfil de jugador a *Server*, que a su vez lo retransmitirá a *Motor*.
- Comienzo del bucle de partida. Se recibirá un código indicando la acción pertinente, tras lo cual se realizará alguna de las siguientes acciones, hasta que finalice la partida:
    - Se reciben la información del estado de la partida (tablero actual y mano del jugador).
    - Se imprime la mano del jugador, el tablero en su estado actual y se solicita al jugador un movimiento a realizar. El movimiento consiste en un número identificativo de una carta de su mano, más una dupla X-Y que determina la posición del tablero donde quiere jugar la carta.
        - Cabe mencionar que se comprueba que el movimiento sea realizable, y en caso contrario se vuelve a pedir al jugador que introduzca los datos.
    - Se recibe el resultado de la partida y se imprime el tablero final por pantalla (solo al finalizar la partida).
- Cierre de la conexión.

Por ende, las entradas y salidas del módulo *Client* son las siguientes:

- **Salida:** Envío del nombre del jugador.

- **Salida:** Bucle que envía el nombre de cada una de las cartas que componen el mazo.
- **En bucle de partida:**
    - **Entrada:** Estado del tablero actual codificado. Se reconstruirá para poder imprimirlo por pantalla.
    - **Entrada:** Estado de la mano del jugador, también codificado.
    - **Salida:** Valor numérico que representa el número de carta que desea jugar.
    - **Salida:** Valor numérico X, representando la columna donde se quiere situar la carta.
    - **Salida:** Valor numérico Y, representando la fila donde se quiere situar la carta.
- **Entrada:** Resultado final de la partida y estado final del tablero.

4.2.3 Server

El módulo *Server* hace de intermediario entre los módulos *Motor* y *Client*, siendo el servidor que acepta las conexiones por *Sockets* y maneja la comunicación. Es un módulo que se comporta en mayor parte de forma pasiva, dado que en lugar de pedir información, espera a recibirla y la retransmite según debe.

El flujo de ejecución del módulo sería el siguiente:

- Aceptar conexión del módulo *Motor* (dos Sockets) y de ambos *Client*.
- Creación de un "Lock" auxiliar, que ayudará a determinar quién tiene derecho de transmisión/recepción.
- Creación de un *Thread* por cada uno de los dos jugadores, que pertenecerá a dicho jugador desde ese momento. Conecta cada uno de los *Threads* con un *Socket* "*Motor-Server*".
- Cuando ambos jugadores se han conectado con éxito, inicialización de los *Threads*. El flujo de ejecución de cada *Thread* es el siguiente:
    - De forma similar al bucle de partida del módulo *Client*, recibe un código que indica las acciones a realizar. Las posibles acciones son las siguientes:
        - Envío del estado del tablero al jugador. No hay petición de movimiento a realizar.
        - Envío del estado actual de su mano al jugador.
        - Petición de carta a jugar al cliente, y casilla donde jugarla.
        - Comunicación al cliente de que ha ganado o perdido la partida.

        - Mensaje de finalización de la partida y del hilo de ejecución del motor (se envía si no se quieren jugar más partidas).
    - Cierre del *Thread*.
- Espera a la finalización de los *Threads*, que solo finalizan al acabar la partida y no comenzar otra.
- Cierre de la conexión.

Los flujos de entrada y salida no se especificarán para el módulo *Server*, dado que es simplemente un intermediario y ya se ha especificado el flujo de información tanto para el módulo *Motor* como para *Client*. Por lo tanto, el flujo de información del módulo Server se extrapola directamente de los flujos de los otros módulos.

La figura 8 resume los flujos de entrada y salida en una arquitectura modular ligeramente extendida, mostrando que cada *Thread* actúa realmente como un intermediario entre su cliente asignado y el motor:

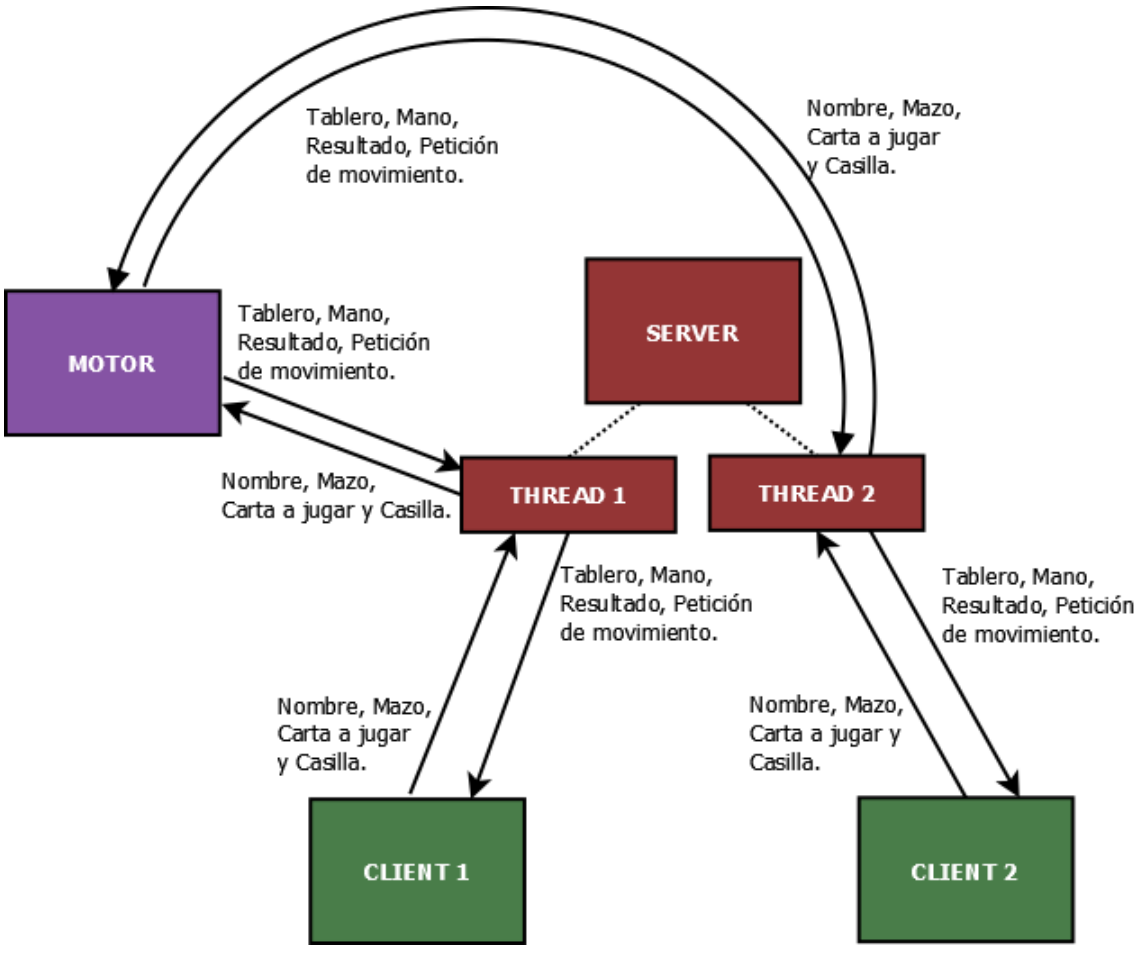

Figura 8: Flujo de información en una arquitectura modular extendida.

4.3 Modelo de conocimiento de la aplicación

En este apartado se mostrará un diagrama UML resumido de la aplicación, y a continuación se explicarán las clases más significativas de cada módulo y su función.

4.3.1 Motor

El diagrama de clases del módulo *Motor* se puede resumir con el esquema UML representado en la figura 9.

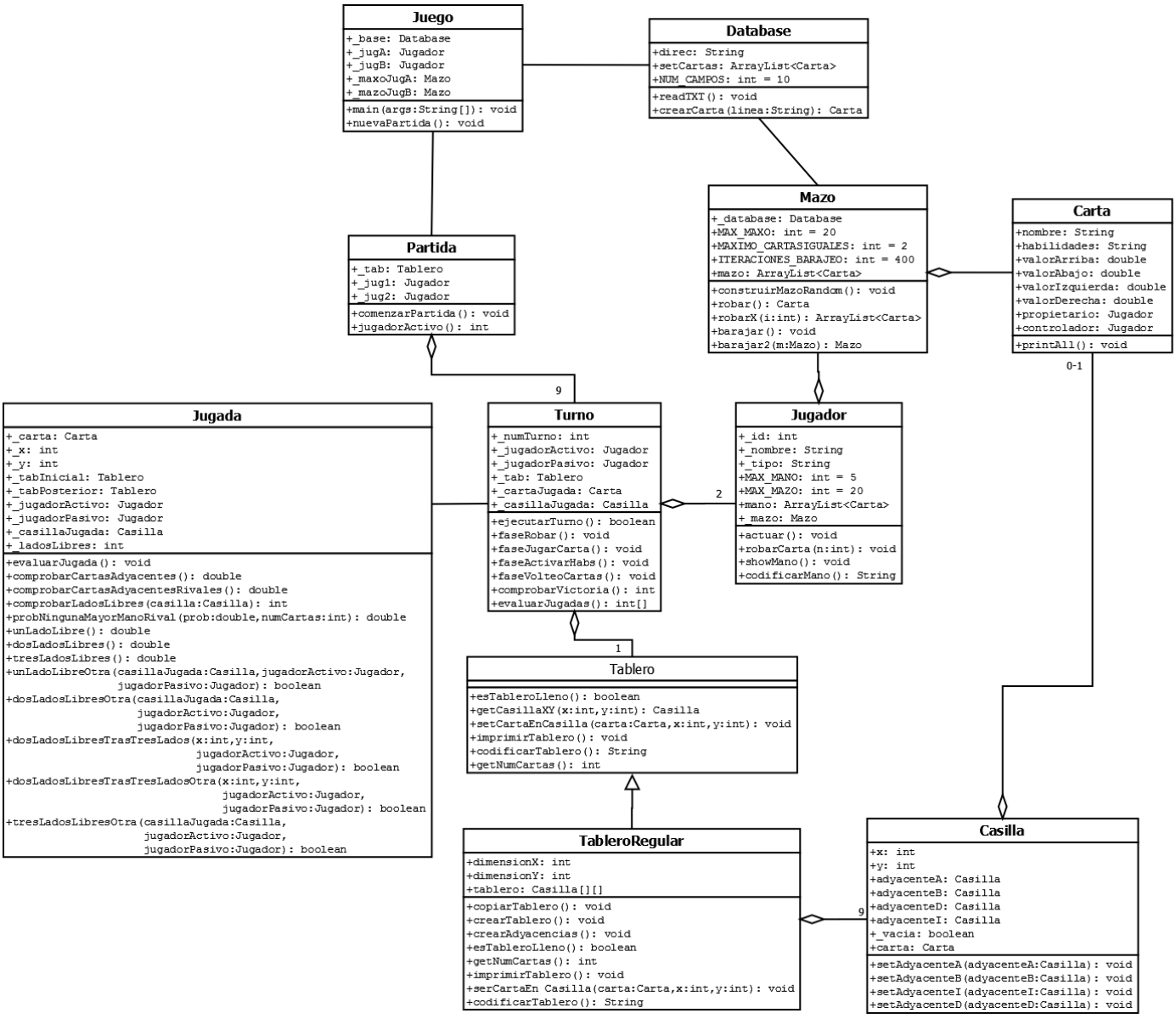

Figura 9: Diagrama de clases del módulo *Motor*.

A continuación se comentará brevemente la función de cada una de las clases:

- **Juego:** La clase *main*. Se encarga de conectar dos Sockets al módulo Server (se asignará uno a cada Thread), a través de los cuales recibirá la información de cada uno de los jugadores (nombre y mazo). Tras recibir la información, llamará a la función *nuevaPartida()* que creará un objeto *Partida.*
- **Partida:** Inicia la partida, asignando la mano inicial a cada jugador. Se encarga de imprimir el tablero antes de llamar a la clase Turno, que se encarga de las operaciones relativas al transcurso del turno. También decide el jugador inicial de forma aleatoria e informa a cada jugador del fin de la partida y de su resultado.
- **Turno:** En una partida, la clase *Partida* crea 9 objetos "Turno" nuevos, uno por cada turno. Esta clase se encarga de las acciones que transcurren en un turno. Comienza con la fase de robar cartas, en la que el jugador activo roba una carta. Después, dependiendo del tipo de jugador (*Humano*, *IA* o *Random*) jugará una carta o solicitará al jugador activo el movimiento a realizar. Tras ello se activarán las habilidades (como por ejemplo robar cartas extra). Por último, se procederá a voltear las cartas enemigas que han sido superadas, finalizando el turno y volviendo al bucle de la clase Partida, que creará el próximo *Turno*.
- **Database:** Clase que se encarga de leer una base de datos de cartas en un formato específico y crear un *Array* con todas las cartas. Se utiliza el formato ".txt" por defecto al exportar una base de datos de Excel. En la entrega se adjuntan los archivos Excel y ".txt" utilizados.
- **Jugador:** El objeto que representa a uno de los jugadores conectados desde el módulo *Client*. Contiene atributos que identifican al jugador (nombre e Id.) y otros igual de relevantes, como el mazo o el tipo de jugador. Los métodos más importantes son los que permiten actuar con la mano del jugador (robar carta, codificar la mano para posteriormente enviarla e imprimir la mano por pantalla), representada por un atributo *ArrayList<Carta>*.
- **Mazo:** Clase que representa el mazo de cartas del jugador. Contiene los métodos de barajeo y robo de cartas. También tiene un importante método para construir mazos de cartas aleatorios, pero en esta versión dicho método es utilizado por la clase *Mazo* del módulo *Client*, dado que se ha pretendido que el jugador cree el mazo y lo envíe al servidor como parte de su información (y posteriormente al motor).

- **Carta:** Objeto que representa una carta, conteniendo todos los atributos necesarios, desde los valores de la carta, habilidades o posición que ocupa en el tablero, hasta su propietario o su controlador actual. Prácticamente todos sus métodos son del tipo *get* y *set*.
- **TableroRegular:** Especificación de un tipo de tablero, el tablero de dimensiones 3x3, el único implementado para esta versión del juego. Hereda de la interfaz "Tablero", creada para favorecer una estructura donde se puedan añadir otros tipos de tablero de forma más sencilla. Contiene tanto los métodos para crear el tablero como una gran cantidad de métodos auxiliares (comprobadores, métodos para codificar o imprimir resultados por pantalla…). El tablero se crea como *Array* bidimensional 3x3 de objetos "Casilla", tras lo cual a cada casilla se le asignan unas adyacencias dependiendo de las casillas con las que tiene contacto. Se utiliza para comprobar rápidamente las adyacencias de una casilla.
- **Casilla**: Objetos que conforman un tablero de juego. Cada casilla tendrá, aparte de los atributos que indican su posición en el tablero y la carta que ocupa dicha casilla, cuatro atributos que guardarán sus casillas adyacentes, siendo *null* en caso de no tener ninguna casilla adyacente en esa dirección.
- **Jugada:** Cuando un jugador del tipo "IA" debe realizar un movimiento, la clase *Turno* creará un objeto "Jugada" por cada una de las posibles jugadas a realizar. Básicamente, consiste en una dupla Carta-Casilla que guarda una puntuación de evaluación en un atributo. Con la función de evaluación implementada se evaluará la viabilidad de dicha jugada, es decir, de utilizar dicha carta en dicha casilla. Esta clase contiene la implementación de la inteligencia artificial, pero no se hablará de cada uno de los métodos en esta sección, dado que hay una sección especial dedicada a hablar del Jugador Inteligente implementado.

4.3.2 Client

En la figura 10 se muestra el diagrama de clases en formato UML para el módulo *Client*.

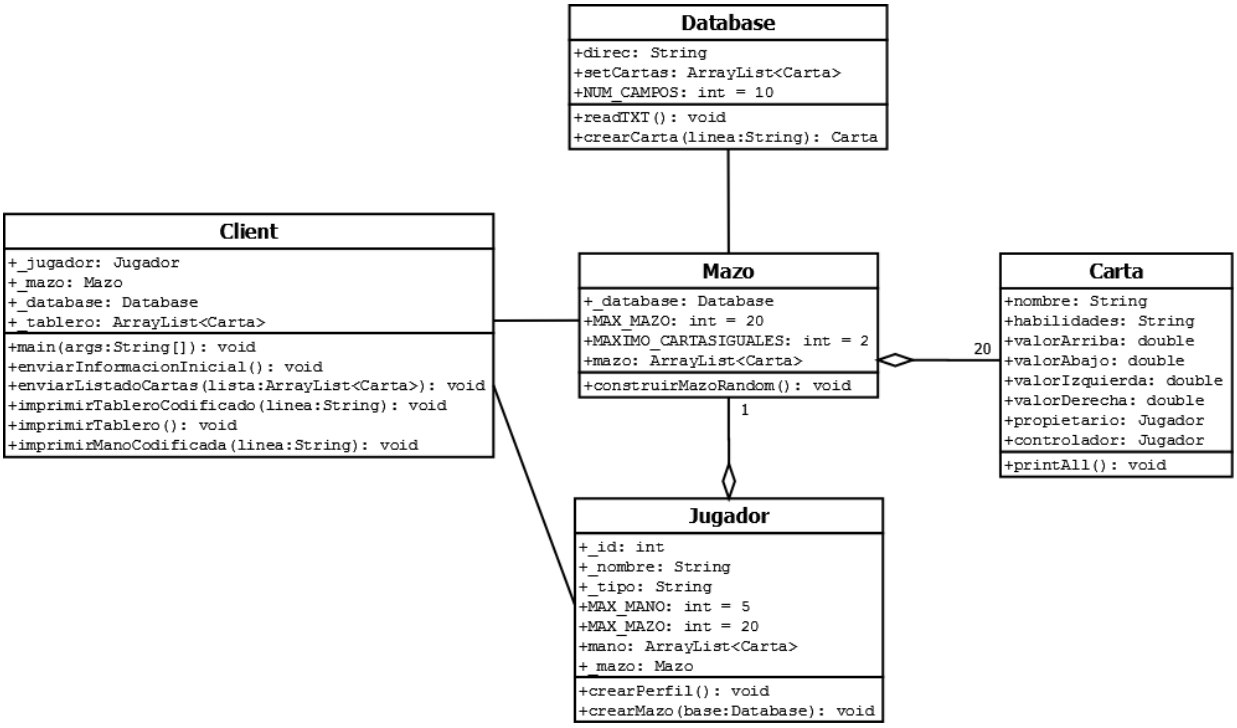

Figura 10: Diagrama de clases del módulo *Client*.

- **Client:** La clase main. Se encarga de la mayoría de funciones del cliente, además de realizar la conexión al *Server* mediante *Sockets*. Comienza creando el perfil del jugador, que consiste en un nombre introducido por consola y un mazo creado de forma aleatoria mediante la clase *Mazo*. Tras enviarlo al Server, comienza la partida y entra en un bucle en el que responde a los envíos y peticiones que le llegan desde Server en forma de códigos. Las funciones que realiza son: imprimir el tablero, imprimir el tablero y pedir un movimiento al jugado, imprimir la mano del jugador, informar del resultado de la partida, y finalizar el programa.
- **Database:** Clase clónica de la clase del mismo nombre del módulo *Motor*. Es importante porque el mazo aleatorio se genera en el cliente, y necesita acceso a la base de datos de cartas.
- **Jugador:** Muy parecida en estructura a la clase del mismo nombre del módulo *Motor*, sin embargo no se hace uso de funciones relativas a la partida (como se comentó anteriormente, la clase Jugador del módulo

Motor manejaba varias funciones relacionadas con la mano del jugador), como robar carta, mientras que se hace uso de otras funciones como la de crear perfil.

- **Mazo:** Clase similar a su correspondiente en el módulo *Motor*, exceptuando el uso de "construirMazoRandom()" y la ausencia de funciones relativas a las operaciones de la partida, como robar y barajar.
- **Carta:** Prácticamente una clase clónica de su correspondiente en el módulo *Motor*. De nuevo, sirve como un "esqueleto", que en lugar de proporcionar operaciones proporciona una estructura con un gran número de atributos para especificar el contenido de una carta.

4.3.3 Server

En la figura 11 se muestra el diagrama de clases en formato UML para el módulo *Server*.

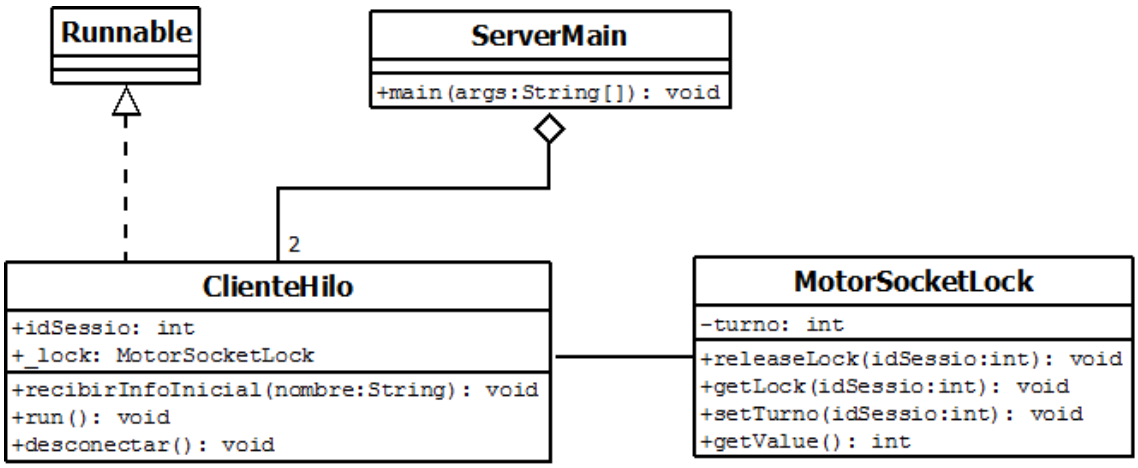

Figura 11: Diagrama de clases del módulo *Server*.

- **ServerMain:** La clase *main* del módulo. Se encarga de aceptar las conexiones tanto de parte del motor como de los clientes, para luego crear dos hilos (uno por jugador) que hagan de intermediarios entre *Motor* y ambos *Client*.
- **ClienteHilo:** La clase que se encarga de tratar y retransmitir el flujo de información entre uno de los dos clientes y el *Socket* del módulo *Motor*.
- **MotorSocketLock:** Clase de seguridad que evita que el motor reciba información de ambos clientes al mismo tiempo, mediante el uso de una variable "lock".

Resultados

En este apartado se mostrará una partida de ejemplo, tras la cual se analizará una gran cantidad de resultados de varios tipos, con la intención de mostrar la efectividad del Jugador Inteligente tanto en partidas contra jugadores humanos como en partidas donde el oponente mueve aleatoriamente.

5.1 Ejemplo

En esta sección se explicará un ejemplo de ejecución del programa, haciendo uso de las impresiones por pantalla de la aplicación y haciendo explicaciones breves de lo ocurrido. Para ello, se mostrará el resultado del módulo *Motor*, ya que es la parte más relevante de la aplicación y contiene impresiones por pantalla muy útiles para el entendimiento del programa.

En el ejemplo en cuestión, se mostrará una partida entre dos jugadores del tipo “IA”, es decir, dos jugadores inteligentes enfrentados.

Se podrán identificar los comentarios por estar en negrita.

```
Cargando base de datos.
```

En primer lugar, se procede a cargar la base de datos desde el archivo correspondiente. Las siguientes líneas no son más que una impresión de las cartas cargadas.

```
Soldado entrenado, 5.0, 5.0, 5.0, 5.0, ---, 3.0, Comun, Trecia, 5.0
Ratido de biblioteca, 3.0, 3.0, 4.0, 3.0, ROBAR 1, 3.0, Comun, La Plaga, 3.5
Patrulla de La Corriente, 7.0, 3.0, 1.0, 3.0, ECO, 5.0, Rara, La Corriente,
3.5
Defensor de la tierra, 8.0, 5.0, 7.0, 5.0, DEFENSOR, 3.0, Rara, Landuin, 6.75
Ogro suturado, 9.0, 6.0, 5.0, 6.0, FUERZA, 8.0, epica, Hordas del olvido, 5.5
Valeria "la justa", 7.0, 7.0, 7.0, 7.0, ---, 5.0, unica, Sin afiliacion, 7.0
Soldado aguerrido, 3.0, 7.0, 7.0, 7.0, ---, 4.0, Comun, Trecia, 4.75
Explorador de Landuin, 8.0, 2.0, 3.0, 2.0, ---, 4.0, Comun, Landuin, 4.0
Tirador de Landuin, 7.0, 3.0, 6.0, 3.0, ---, 3.0, Comun, Landuin, 4.75
Esclavo de La Corriente, 1.0, 3.0, 2.0, 3.0, ECO, 2.0, Comun, La Corriente,
2.0
Alimana, 3.0, 2.0, 4.0, 2.0, ---, 1.0, Comun, La Plaga, 3.25
Ratido apestoso, 4.0, 1.0, 1.0, 1.0, ---, 0.0, Comun, La Plaga, 2.75
Ratido devorador, 6.0, 6.0, 2.0, 6.0, ---, 3.0, Comun, La Plaga, 4.25
Guardia de la muralla, 3.0, 7.0, 7.0, 7.0, DEFENSOR, 2.0, Comun, Trecia, 5.5
Aprendiz de asesino, 6.0, 1.0, 1.0, 1.0, ---, 2.0, Comun, Trecia, 2.25
Tirador novato, 4.0, 5.0, 5.0, 5.0, ---, 2.0, Comun, Trecia, 3.75
Ogro desfigurado, 3.0, 5.0, 7.0, 5.0, FUERZA, 4.0, Comun, Landuin, 4.75
Combatiente putrido, 2.0, 4.0, 6.0, 4.0, ---, 3.0, Comun, Hordas del olvido,
4.5
```

```
Bardo temeroso, 2.0, 8.0, 7.0, 8.0, DEFENSOR, 3.0, Comun, Landuin, 4.5
Cabecilla ratido, 5.0, 6.0, 4.0, 6.0, ---, 4.0, Comun, La Plaga, 5.0
Buscador del estertor, 3.0, 3.0, 0.0, 3.0, ECO, 3.0, Comun, Hordas del olvido, 1.75
Renegado, 3.0, 6.0, 6.0, 6.0, ---, 3.0, Comun, Sin afiliacion, 4.0
Buscon apatrida, 3.0, 1.0, 1.0, 1.0, ---, 2.0, Comun, Sin afiliacion, 3.0
Historiador treciano, 2.0, 0.0, 4.0, 0.0, ROBAR 1, 2.0, Comun, Trecia, 2.75
Esclavo ignorante, 0.0, 5.0, 0.0, 5.0, ---, 0.0, Comun, La Corriente, 2.5
Homunculo, 6.0, 1.0, 5.0, 1.0, ---, 2.0, Comun, La Corriente, 3.25
Asesino contratado, 0.0, 0.0, 0.0, 0.0, ---, 3.0, Rara, Trecia, 2.25
Visionario, 4.0, 6.0, 2.0, 6.0, ECO, 6.0, Rara, Landuin, 4.5
Estratega de Landuin, 5.0, 9.0, 9.0, 9.0, DEFENSOR, 5.0, Rara, Landuin, 7.0
Comandante treciano, 7.0, 8.0, 3.0, 8.0, FUERZA, 5.0, Rara, Trecia, 5.75
Hambre insaciable, 5.0, 2.0, 2.0, 2.0, ECO, 3.0, Rara, Hordas del olvido, 3.5
Miasma viviente, 5.0, 8.0, 3.0, 8.0, ---, 4.0, Rara, La plaga, 6.0
Brujo del olvido, 0.0, 3.0, 3.0, 3.0, ROBAR 2, 4.0, Rara, Hordas del olvido, 2.25
Soldado de fase, 0.0, 9.0, 0.0, 9.0, ---, 5.0, Rara, La Corriente, 4.5
Patron del gremio de asesinos, 8.0, 5.0, 6.0, 5.0, ---, 5.0, epica, Trecia, 6.5
Alima¤a sedienta, 8.0, 1.0, 6.0, 1.0, ROBAR 1, 4.0, epica, La Plaga, 5.0
Chaman ciego, 2.0, 2.0, 7.0, 2.0, ROBAR 3, 6.0, epica, La Corriente, 4.25
Acechador, 3.0, 2.0, 9.0, 2.0, FUERZA, 3.0, epica, Hordas del olvido, 4.25
Kalar'Zar, piedra guia, 9.0, 9.0, 9.0, 9.0, DEFENSOR, 10.0, unica, La Corriente, 9.0
Locura encarnada, 7.0, 8.0, 9.0, 8.0, ---, 6.0, unica, La Corriente, 6.75
```

Una vez cargadas las cartas, se crean dos conexiones por *Sockets* al módulo *Server*.

```
Conectando al puerto 5001
Conectando al puerto 5002
Motor: Aceptado por servidor
```

A continuación, se inicia la recepción de datos.

```
Iniciando nueva partida.
```

Para empezar, se reciben los datos del que es denominado “Jugador 1”. Primero el nombre, y a continuación las cartas pertenecientes al mazo. Se omitirá parte de las cartas añadidas por la irrelevancia de la acción.

```
Recibiendo en la primera conexion
Numero de jugador igual a 1
Nombre de jugador igual a a
Añadida la carta Valeria "la justa"
Añadida la carta Esclavo de La Corriente
...
Añadida la carta Hambre insaciable
Mazo leido con éxito
```

En segundo lugar, el mismo proceso para el “Jugador 2”.

```
Recibiendo en la segunda conexion
Numero de jugador igual a 2
Nombre de jugador igual a b
Añadida la carta Soldado de fase
Añadida la carta Esclavo ignorante
...
Añadida la carta Ogro suturado
PERFILES DE JUGADORES CREADOS!
```

Tras crear los perfiles de jugador, se llama a la clase *Partida*, que crea el tablero inicial y las manos de cada jugador, y posteriormente llama a la clase *Turno* para dar comienzo al primer turno.

```
Creando tablero.
Comenzando partida.
Robando cartas iniciales.
```

La clase *Turno* realiza siempre un bucle explicado en secciones anteriores, en el que roba cartas, elige jugadas (la elección se hace dependiendo del tipo de jugador), aplica las habilidades y el volteo de las cartas.
Se puede ver el tablero correspondiente al turno 1 (vacío).

```
INICIANDO TURNO NUMERO 1
Mostrando tablero
     1     2     3
  +-----+-----+-----+
  |     |     |     |
1 |     |     |     |
  |     |     |     |
  +-----+-----+-----+
  |     |     |     |
2 |     |     |     |
  |     |     |     |
  +-----+-----+-----+
  |     |     |     |
3 |     |     |     |
  |     |     |     |
  +-----+-----+-----+
```

En este ejemplo, el Jugador 2 ha sido elegido para empezar la partida. Se muestra la mano al jugador correspondiente. En la clase *Motor* se realiza un seguimiento complementario de la partida, ya que en cada cliente se puede ver la información relativa a cada jugador, por ejemplo, la mano que se muestra a continuación también se mostraría en el cliente correspondiente.

```
Turno del Jugador 2, b

            2
0   Historiador treciano   5    Habilidades: ROBAR 1
            4

            0
0   Asesino contratado   9    Habilidades: ---
            0

            3
6   Renegado   1    Habilidades: ---
            6

            8
1   Alima¤a sedienta   5    Habilidades: ROBAR 1
            6

            5
6   Cabecilla ratido   5    Habilidades: ---
            4
```

A continuación, al ser un jugador del tipo "IA", se crearán una lista de todas las posibles jugadas (cada carta en cada posible casilla), y se evaluarán mediante la función de evaluación.
Se muestra la impresión por consola de algunos cálculos de probabilidades y evaluaciones que, aunque de forma bastante oscura, pueden ayudar a entender el funcionamiento de la inteligencia artificial. Esto solo se realizará para la primera jugada, debido a la gran cantidad de espacio que ocuparía.

En primer lugar (para la primera carta de la mano en la primera casilla del tablero, la 1-1), se calcula la probabilidad de que, puesta en esa casilla, el rival no tuviera ninguna carta en mano que la superara. Como vemos la probabilidad es muy baja, dado que el jugador rival tendrá bastantes cartas en mano y el valor a superar sería un 4.
```
Probabilidad ninguna mayor mano rival: 0.04535479918652205
```

A continuación se calcula la posibilidad de que, dado que el jugador enemigo volteara dicha carta, el jugador NO PUDIERA recuperarla. Al quedar un valor "5" abierto, la probabilidad de NO recuperarla es bastante baja, dado que el jugador tendrá varias cartas en mano (una menos que el rival, con lo cual la probabilidad de no poder superarla aumenta significativamente).
```
Probabilidad ninguna mayor mano mia: 0.21787359775641024
```

Por lo tanto, la evaluación final para el caso de dejar dos lados libres, uno de valor 4 y otro de valor 5, se obtiene al calcular la probabilidad combinada, resultando en:
```
La evaluacion final para dos lados libres es 0.7920080155178768
```

A continuación, y aunque en este caso no sea relevante, se calcula el estado en el que quedan tus cartas adyacentes y las cartas rivales adyacentes, dado que han sido afectadas por el movimiento. Al estar el tablero vacío, estas primeras evaluaciones son iguales a 0.
```
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
```

Por lo tanto, la evaluación final de la jugada, en un rango entre [-0.5, 0.5], es la siguiente:
Evaluacion: 0.3420080155178768
Se puede observar que el valor esperado sería un "0.292008", al adaptar "0.792008" al rango [-0.5, 0.5]. Sin embargo, al tener la habilidad "Robar 1", la evaluación final es mayor, dado que el valor de la carta aumenta. Robar cartas es muy útil, en especial en los primeros turnos de partida, y por ello la función de evaluación aplica modificadores teniéndolo en cuenta.

Tras esta primera evaluación, se evaluaría la misma carta para el resto de casillas. En algún caso muy concreto se puede observar que una evaluación sale del rango [-0.5, 0.5], y se debe a modificadores por habilidades (como robar carta) o a pequeñas modificaciones en la evaluación final en forma de *tie-breakers*.

La siguiente casilla sería la 1-2 (se evalúa de arriba abajo y de izquierda a derecha), con lo cual quedarían tres lados libres. Vemos que la evaluación en este caso no es tan positiva (ronda el 50% de probabilidades de acabar conservando la carta).

```
La evaluacion para tres lados libres es: 0.48750000000000004
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.036500000000000046
```

Las evaluaciones continúan para el resto de casillas, hasta acabar de evaluar la primera carta,

```
Probabilidad ninguna mayor mano rival: 0.0020651588060099572
Probabilidad ninguna mayor mano mia: 0.21787359775641024
La evaluacion final para dos lados libres es 0.7825763458225935
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.3325763458225935
La evaluacion para tres lados libres es: 0.30000000000000004
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: -0.15099999999999997
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
```

Evaluacion: -0.05 **(La casilla central. Se evaluará siempre negativamente, dejándola como último recurso en caso de que todas las jugadas sean malas)**

```
La evaluacion para tres lados libres es: 0.525
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.07400000000000002
Probabilidad ninguna mayor mano rival: 1.464380704717876E-5
Probabilidad ninguna mayor mano mia: 0.07298473432313893
La evaluacion final para dos lados libres es 0.9270163344512279
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.4770163344512279
La evaluacion para tres lados libres es: 0.38749999999999996
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: -0.06350000000000004
Probabilidad ninguna mayor mano rival: 1.464380704717876E-5
Probabilidad ninguna mayor mano mia: 0.005669063389046942
La evaluacion final para dos lados libres es 0.9943310196276235
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.5443310196276235
```

Se ha acabado de evaluar la primera carta. Vemos que la mejor evaluación se da en la última casilla, dejando el 0 y el 2 libres y además robando carta. Es una muy buena jugada, aunque en principio parezca contra-intuitiva, dado que por la denominada "regla de los dos lados", las probabilidades de acabar conservando dicha carta son prácticamente del 100%. Básicamente, es una carta que será muy fácilmente recuperable.

A continuación se evaluará la segunda carta ("Asesino contratado"), que da lugar a un caso curioso.
Se puede observar como la evaluación de la primera casilla es terrible, dado que dejar un 0 y un 9 abiertos es prácticamente la peor jugada posible: el rival podrá ganar sin duda al número "0", mientras que el jugador no podrá vencer al número "9". La probabilidad combinada de conservar la carta es nimia.

```
Probabilidad ninguna mayor mano rival: 1.464380704717876E-5
Probabilidad ninguna mayor mano mia: 1.0
La evaluacion final para dos lados libres es 1.4643807047187707E-5
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: -0.4999853561929528
```

Sin embargo, al evaluar la carta para la casilla 1-2 y dejar tres lados libres, la calidad de la jugada se dispara. Esto se debe a que, al dejar dos lados muy fácilmente recuperables (dos ceros) y un lado "imposible" de recuperar (un nueve), el resultado es prácticamente equivalente al de dejar dos lados libres con dos ceros, y así se refleja en la evaluación final. De hecho, es ligeramente mejor jugada que la mejor evaluada hasta ahora, pero el hecho de poder robar carta decanta la balanza a favor de la anterior.

```
La evaluacion para tres lados libres es: 1.0
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.499
```

A continuación, el resto de jugadas posibles de la segunda carta.

```
Probabilidad ninguna mayor mano rival: 1.464380704717876E-5
Probabilidad ninguna mayor mano mia: 1.0
La evaluacion final para dos lados libres es 1.4643807047187707E-5
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: -0.4999853561929528
La evaluacion para tres lados libres es: 1.0
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.499
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: -0.1
La evaluacion para tres lados libres es: 1.0
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.499
Probabilidad ninguna mayor mano rival: 1.464380704717876E-5
Probabilidad ninguna mayor mano mia: 9.762538031452507E-5
La evaluacion final para dos lados libres es 0.9999023760492927
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.49990237604929266
La evaluacion para tres lados libres es: 0.1625
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: -0.3385
Probabilidad ninguna mayor mano rival: 1.464380704717876E-5
Probabilidad ninguna mayor mano mia: 9.762538031452507E-5
```

La evaluacion final para dos lados libres es 0.9999023760492927
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.49990237604929266

La evaluación de la tercera carta:

Probabilidad ninguna mayor mano rival: 2.859933035714285E-4
Probabilidad ninguna mayor mano mia: 0.4248199679943101
La evaluacion final para dos lados libres es 0.5753015276717597
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.07530152767175968
La evaluacion para tres lados libres es: 0.5625
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.0615
Probabilidad ninguna mayor mano rival: 2.859933035714285E-4
Probabilidad ninguna mayor mano mia: 0.03802551009068277
La evaluacion final para dos lados libres es 0.9619853649505681
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.4619853649505681
La evaluacion para tres lados libres es: 0.25625
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: -0.24475000000000002
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: -0.1
La evaluacion para tres lados libres es: 0.5625
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.0615
Probabilidad ninguna mayor mano rival: 0.37930354285206264
Probabilidad ninguna mayor mano mia: 0.4248199679943101
La evaluacion final para dos lados libres es 0.7363157509402316
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.2363157509402316
La evaluacion para tres lados libres es: 0.50625
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.005249999999999977
Probabilidad ninguna mayor mano rival: 0.020642419763513505
Probabilidad ninguna mayor mano mia: 0.4248199679943101
La evaluacion final para dos lados libres es 0.5839493441089508
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.08394934410895083

La evaluación de la cuarta carta:

Probabilidad ninguna mayor mano rival: 0.16963015825320513
Probabilidad ninguna mayor mano mia: 0.4248199679943101
La evaluacion final para dos lados libres es 0.6472423104056862
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.19724231040568624
La evaluacion para tres lados libres es: 0.8625
Evaluacion del estado de tus cartas: 0.0

Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.41150000000000003
Probabilidad ninguna mayor mano rival: 0.16963015825320513
Probabilidad ninguna mayor mano mia: 0.9439903846153846
La evaluacion final para dos lados libres es 0.21613885371642705
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: -0.23386114628357296
La evaluacion para tres lados libres es: 0.35624999999999996
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: -0.09475000000000004
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: -0.05
La evaluacion para tres lados libres es: 0.50625
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.05524999999999998
Probabilidad ninguna mayor mano rival: 2.859933035714285E-4
Probabilidad ninguna mayor mano mia: 0.4248199679943101
La evaluacion final para dos lados libres es 0.5753015276717597
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.12530152767175967
La evaluacion para tres lados libres es: 0.40625
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: -0.04475
Probabilidad ninguna mayor mano rival: 2.859933035714285E-4
Probabilidad ninguna mayor mano mia: 0.9439903846153846
La evaluacion final para dos lados libres es 0.05627959031325114
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: -0.3937204096867489

La evaluación de la quinta carta:

Probabilidad ninguna mayor mano rival: 0.04535479918652205
Probabilidad ninguna mayor mano mia: 0.21787359775641024
La evaluacion final para dos lados libres es 0.7920080155178768
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.2920080155178768
La evaluacion para tres lados libres es: 0.575
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.07399999999999995
Probabilidad ninguna mayor mano rival: 0.16963015825320513
Probabilidad ninguna mayor mano mia: 0.21787359775641024
La evaluacion final para dos lados libres es 0.8190843351102048
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.31908433511020484
La evaluacion para tres lados libres es: 0.6749999999999999
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.17399999999999993
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0

Evaluacion: -0.1
La evaluacion para tres lados libres es: 0.8125
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.3115
Probabilidad ninguna mayor mano rival: 0.04535479918652205
Probabilidad ninguna mayor mano mia: 0.4248199679943101
La evaluacion final para dos lados libres es 0.5944476563444966
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.09444765634449659
La evaluacion para tres lados libres es: 0.6749999999999999
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.17399999999999993
Probabilidad ninguna mayor mano rival: 0.16963015825320513
Probabilidad ninguna mayor mano mia: 0.4248199679943101
La evaluacion final para dos lados libres es 0.6472423104056862
Evaluacion del estado de tus cartas: 0.0
Evaluacion del estado de las cartas rivales: 0.0
Evaluacion: 0.14724231040568625

Al finalizar las evaluaciones, se muestra el resultado de las que han sido las mejor evaluadas en su momento. Se puede ver como, tras tres jugadas, el resultado de "0.544" no ha podido ser superado.

Nueva evaluacion = 0.3420080155178768
La X temporal seria 0
La Y temporal seria 0
La carta temporal seria Historiador treciano

Nueva evaluacion = 0.4770163344512279
La X temporal seria 2
La Y temporal seria 0
La carta temporal seria Historiador treciano

Nueva evaluacion = 0.5443310196276235
La X temporal seria 2
La Y temporal seria 2
La carta temporal seria Historiador treciano

La clase *Turno* realiza la jugada, utiliza las habilidades y voltea las cartas vencidas, en caso de haberlas. A continuación comprueba si la partida ha finalizado y procede al siguiente turno.

IA: El jugador b ha jugado la carta Historiador treciano
Robando cartas gracias a la habilidad ROBAR de Historiador treciano
Carta jugada en la casilla 2-2
Tablero lleno?: false

Turno número 2:

```
INICIANDO TURNO NUMERO 2
Mostrando tablero
     1     2     3
  +-----+-----+-----+
  |     |     |     |
1 |     |     |     |
  |     |     |     |
  +-----+-----+-----+
  |     |     |     |
2 |     |     |     |
  |     |     |     |
  +-----+-----+-----+
  |     |     |  2  |
3 |     |     |0 b 5|
  |     |     |  4  |
  +-----+-----+-----+
Turno del Jugador 1, a
El jugador a ha robado la carta Estratega de Landuin

            3
6   Renegado   1    Habilidades: ---
            6

            2
8   Bardo temeroso   1    Habilidades: DEFENSOR
            7

            3
7   Soldado aguerrido   2    Habilidades: ---
            7

            4
5   Tirador novato   1    Habilidades: ---
            5

            6
1   Homunculo   1    Habilidades: ---
            5

            5
9   Estratega de Landuin   5    Habilidades: DEFENSOR
            9
```

Se procede a evaluar las jugadas. A partir de ahora solo se mostrarán las evaluaciones claves para el entendimiento de la inteligencia artificial. El resumen del resultado de la evaluación del turno 2 es el siguiente:

```
Nueva evaluacion = 0.47936348383030836
La X temporal seria 0
La Y temporal seria 2
La carta temporal seria Renegado
Nueva evaluacion = 0.9625479196992809
La X temporal seria 1
La Y temporal seria 2
La carta temporal seria Renegado
Nueva evaluacion = 1.1720851031123503
La X temporal seria 1
La Y temporal seria 2
```

```
La carta temporal seria Tirador novato
IA: El jugador a ha jugado la carta Tirador novato
Carta jugada en la casilla 1-2
Tablero lleno?: false
```

Se puede observar que la IA ha optado por jugar la carta “Tirador novato” y dejar abiertos dos valores similares (para que al jugador rival le cueste lo mismo ganarla que al jugador recuperarla), un 4 y un 5, al mismo tiempo que obtiene el control de la carta rival.

```
INICIANDO TURNO NUMERO 3
Mostrando tablero
     1     2     3
  +-----+-----+-----+
  |     |     |     |
1 |     |     |     |
  |     |     |     |
  +-----+-----+-----+
  |     |     |     |
2 |     |     |     |
  |     |     |     |
  +-----+-----+-----+
  |     |  4  |  2  |
3 |     |5 a 1|0 a 5|
  |     |  5  |  4  |
  +-----+-----+-----+
Turno del Jugador 2, b
El jugador b ha robado la carta Tirador de Landuin

            0
0   Asesino contratado   9    Habilidades: ---
            0

            3
6   Renegado   1    Habilidades: ---
            6

            8
1   Alima¤a sedienta   5    Habilidades: ROBAR 1
            6

            5
6   Cabecilla ratido   5    Habilidades: ---
            4

            0
5   Esclavo ignorante   5    Habilidades: ---
            0

            7
3   Tirador de Landuin   3    Habilidades: ---
            6

Nueva evaluacion = 0.499
La X temporal seria 0
La Y temporal seria 1
La carta temporal seria Asesino contratado
Nueva evaluacion = 0.620764565952583
La X temporal seria 1
```

```
La Y temporal seria 1
La carta temporal seria Asesino contratado
Nueva evaluacion = 0.9513767194177352
La X temporal seria 1
La Y temporal seria 1
La carta temporal seria Renegado
Nueva evaluacion = 1.2951928667639443
La X temporal seria 2
La Y temporal seria 1
La carta temporal seria Renegado
Nueva evaluacion = 1.3517044438146009
La X temporal seria 2
La Y temporal seria 1
La carta temporal seria Cabecilla ratido
IA: El jugador b ha jugado la carta Cabecilla ratido
Carta jugada en la casilla 2-1
Tablero lleno?: false
```

De nuevo, el jugador inteligente opta por dejar dos valores similares abiertos, al mismo tiempo que recupera la carta puesta en el primer turno y la cierra, asegurándose su control para el resto de la partida. Esto culmina la "estrategia" de los dos lados, en la que en el primer turno dejó dos lados bajos abiertos para poder cerrar la carta fácilmente en un futuro próximo.
Se procede al turno 4.

```
INICIANDO TURNO NUMERO 4
Mostrando tablero
     1     2     3
  +-----+-----+-----+
  |     |     |     |
1 |     |     |     |
  |     |     |     |
  +-----+-----+-----+
  |     |     |  5  |
2 |     |     |6 b 5|
  |     |     |  4  |
  +-----+-----+-----+
  |     |  4  |  2  |
3 |     |5 a 1|0 b 5|
  |     |  5  |  4  |
  +-----+-----+-----+
Turno del Jugador 1, a
El jugador a ha robado la carta Alimana

            3
6   Renegado   1    Habilidades: ---
            6

            2
8   Bardo temeroso   1    Habilidades: DEFENSOR
            7

            3
7   Soldado aguerrido   2    Habilidades: ---
            7
```

```
            6
1   Homunculo   1    Habilidades: ---
            5

            5
9   Estratega de Landuin   5    Habilidades: DEFENSOR
            9

            3
2   Alimana   4    Habilidades: ---
            4

Nueva evaluacion = 0.12080493542121395
La X temporal seria 0
La Y temporal seria 0
La carta temporal seria Renegado
Nueva evaluacion = 0.14016748546759555
La X temporal seria 1
La Y temporal seria 1
La carta temporal seria Renegado
Nueva evaluacion = 0.8390713904694168
La X temporal seria 2
La Y temporal seria 0
La carta temporal seria Renegado
Nueva evaluacion = 1.2361392025248932
La X temporal seria 2
La Y temporal seria 0
La carta temporal seria Soldado aguerrido
IA: El jugador a ha jugado la carta Soldado aguerrido
Carta jugada en la casilla 2-0
Tablero lleno?: false
```

Se puede ver cómo, en este caso, en lugar de optar por poner la carta en la casilla central y dejar dos lados abiertos (el 3 y el 7, una mala combinación), el jugador inteligente sitúa la carta en la esquina superior derecha, dejando un solo lado abierto con un valor medianamente alto, al mismo tiempo que recupera una carta y la deja con un valor abierto también decente.
Se inicia el turno 5.

```
INICIANDO TURNO NUMERO 5
Mostrando tablero
     1     2     3
  +-----+-----+-----+
  |     |     |  3  |
1 |     |     |7 a 2|
  |     |     |  7  |
  +-----+-----+-----+
  |     |     |  5  |
2 |     |     |6 a 5|
  |     |     |  4  |
  +-----+-----+-----+
  |     |  4  |  2  |
3 |     |5 a 1|0 b 5|
  |     |  5  |  4  |
  +-----+-----+-----+
```

```
Turno del Jugador 2, b
El jugador b ha robado la carta Brujo del olvido

            0
0   Asesino contratado   9    Habilidades: ---
            0

            3
6   Renegado   1    Habilidades: ---
            6

            8
1   Alima¤a sedienta   5    Habilidades: ROBAR 1
            6

            0
5   Esclavo ignorante   5    Habilidades: ---
            0

            7
3   Tirador de Landuin   3    Habilidades: ---
            6

            0
3   Brujo del olvido   3    Habilidades: ROBAR 2
            3



Nueva evaluacion = 0.499
La X temporal seria 0
La Y temporal seria 1
La carta temporal seria Asesino contratado
Nueva evaluacion = 1.6666520230740607
La X temporal seria 1
La Y temporal seria 0
La carta temporal seria Asesino contratado
Nueva evaluacion = 1.7884165890266437
La X temporal seria 1
La Y temporal seria 1
La carta temporal seria Asesino contratado
IA: El jugador b ha jugado la carta Asesino contratado
Carta jugada en la casilla 1-1
Tablero lleno?: false
```

En este caso, se observa cómo las evaluaciones de situar la carta en el centro o una casilla más arriba son muy similares. Ambas son buenas jugadas, dejando dos valores fácilmente recuperables abiertos (dos ceros) al mismo tiempo que se gana una carta y se deja cerrada.

```
INICIANDO TURNO NUMERO 6
Mostrando tablero
     1     2     3
  +-----+-----+-----+
  |     |     |  3  |
1 |     |     |7 a 2|
  |     |     |  7  |
  +-----+-----+-----+
  |     |  0  |  5  |
2 |     |0 b 9|6 b 5|
  |     |  0  |  4  |
  +-----+-----+-----+
  |     |  4  |  2  |
3 |     |5 a 1|0 b 5|
  |     |  5  |  4  |
  +-----+-----+-----+
Turno del Jugador 1, a
El jugador a ha robado la carta Tirador de Landuin

            3
6   Renegado   1    Habilidades: ---
            6

            2
8   Bardo temeroso   1    Habilidades: DEFENSOR
            7

            6
1   Homunculo   1    Habilidades: ---
            5

            5
9   Estratega de Landuin   5    Habilidades: DEFENSOR
            9

            3
2   Alimana   4    Habilidades: ---
            4

            7
3   Tirador de Landuin   3    Habilidades: ---
            6

Nueva evaluacion = 0.9618644146996266
La X temporal seria 0
La Y temporal seria 1
La carta temporal seria Renegado
Nueva evaluacion = 1.0183759917502833
La X temporal seria 0
La Y temporal seria 1
La carta temporal seria Homunculo
Nueva evaluacion = 1.2889196482189251
La X temporal seria 0
La Y temporal seria 1
La carta temporal seria Alimana
IA: El jugador a ha jugado la carta Alimana
Carta jugada en la casilla 0-1
```

```
Tablero lleno?: false
```

En este caso, vemos cómo la IA puede optar por poner la carta “Renegado” en la casilla superior central, ganando una carta y dejando un valor decente abierto (un 6). Sin embargo, tras los cálculos probabilísticos, la probabilidad de recuperar una carta con dos valores bajos abiertos es bastante mayor que la de conservar una carta con un solo número medio abierto, dado que el oponente tiene un número elevado de cartas en mano (seis) y es muy probable que pueda vencer a dicho número.

```
INICIANDO TURNO NUMERO 7
Mostrando tablero
     1     2     3
  +-----+-----+-----+
  |     |     |  3  |
1 |     |     |7 a 2|
  |     |     |  7  |
  +-----+-----+-----+
  |  3  |  0  |  5  |
2 |2 a 4|0 a 9|6 b 5|
  |  4  |  0  |  4  |
  +-----+-----+-----+
  |     |  4  |  2  |
3 |     |5 a 1|0 b 5|
  |     |  5  |  4  |
  +-----+-----+-----+
Turno del Jugador 2, b
El jugador b ha robado la carta Acechador

            3
6   Renegado   1    Habilidades: ---
            6

            8
1   Alima¤a sedienta   5    Habilidades: ROBAR 1
            6

            0
5   Esclavo ignorante   5    Habilidades: ---
            0

            7
3   Tirador de Landuin   3    Habilidades: ---
            6

            0
3   Brujo del olvido   3    Habilidades: ROBAR 2
            3

            3
2   Acechador   3    Habilidades: FUERZA
            9

Nueva evaluacion = 0.4972054375434718
La X temporal seria 0
La Y temporal seria 2
La carta temporal seria Renegado
```

```
Nueva evaluacion = 0.8793035428520627
La X temporal seria 1
La Y temporal seria 0
La carta temporal seria Renegado
Nueva evaluacion = 1.2245474732545045
La X temporal seria 0
La Y temporal seria 2
La carta temporal seria Alima¤a sedienta
IA: El jugador b ha jugado la carta Alima¤a sedienta
Robando cartas gracias a la habilidad ROBAR de Alima¤a sedienta
Carta jugada en la casilla 0-2
Tablero lleno?: false
```

En este caso, se ve claramente cómo la mejor opción es vencer a la carta puesta por el oponente en el turno anterior, al mismo tiempo que la carta jugada queda cerrada y se roba carta (aunque al quedar pocos turnos, la efectividad de robar cartas queda muy reducida, y así lo refleja el *bonus* de la función de evaluación).

```
INICIANDO TURNO NUMERO 8
Mostrando tablero
     1     2     3
  +-----+-----+-----+
  |     |     |  3  |
1 |     |     |7 a 2|
  |     |     |  7  |
  +-----+-----+-----+
  |  3  |  0  |  5  |
2 |2 b 4|0 a 9|6 b 5|
  |  4  |  0  |  4  |
  +-----+-----+-----+
  |  8  |  4  |  2  |
3 |1 b 5|5 a 1|0 b 5|
  |  6  |  5  |  4  |
  +-----+-----+-----+
Turno del Jugador 1, a
El jugador a ha robado la carta Locura encarnada

            3
6   Renegado   1    Habilidades: ---
            6

            2
8   Bardo temeroso   1    Habilidades: DEFENSOR
            7

            6
1   Homunculo   1    Habilidades: ---
            5

            5
9   Estratega de Landuin   5    Habilidades: DEFENSOR
            9

            7
3   Tirador de Landuin   3    Habilidades: ---
            6
```

```
            7
8   Locura encarnada   3    Habilidades: ---
            9


Nueva evaluacion = 0.6667402678844975
La X temporal seria 0
La Y temporal seria 0
La carta temporal seria Renegado
Nueva evaluacion = 0.787323717948718
La X temporal seria 1
La Y temporal seria 0
La carta temporal seria Bardo temeroso
Nueva evaluacion = 0.8433333333333333
La X temporal seria 1
La Y temporal seria 0
La carta temporal seria Estratega de Landuin
IA: El jugador a ha jugado la carta Estratega de Landuin
Tablero lleno?: false
```

Este penúltimo turno representa un caso digno de análisis. Se puede ver cómo evalúa positivamente el hecho de jugar una carta en la esquina superior izquierda y recuperar la carta inferior. Sin embargo, dicha jugada dejaría un 1 abierto. Eso quiere decir que, prácticamente con total seguridad, la perdería al siguiente turno, por lo tanto la evaluación es positiva pero claramente mejorable.
A continuación, evalúa la posibilidad de utilizar una carta con "Defensor" que dejaría abierto un valor muy elevado (un 9). No vencería ninguna carta este turno, pero tampoco dejaría 3 cartas suyas fácilmente recuperables al próximo turno (dejaría abiertos un 1, un 0 y un 7, con lo cual dos de las cartas serían vencidas con seguridad). Se puede ver cómo la evaluación es mejor para el caso de la carta con "Defensor", ya que al mismo tiempo que asegura la carta, cierra dos cartas suyas que podrían estar en peligro.
Algo que la IA no tiene en cuenta es que, a pesar de ser ésta la mejor jugada en cuestión de probabilidad, ha perdido la partida con seguridad, dado que solo tiene 4 cartas en su control y necesitaba "arriesgar" para ganar la quinta carta. Si, la probabilidad de conservar las cinco cartas al próximo turno era prácticamente equivalente a 0, pero era un riesgo que un humano hubiera sabido tomar.

```
INICIANDO TURNO NUMERO 9
Mostrando tablero
     1     2     3
  +-----+-----+-----+
  |     |  5  |  3  |
1 |     |9 a 5|7 a 2|
  |     |  9  |  7  |
  +-----+-----+-----+
  |  3  |  0  |  5  |
2 |2 b 4|0 a 9|6 b 5|
  |  4  |  0  |  4  |
  +-----+-----+-----+
  |  8  |  4  |  2  |
3 |1 b 5|5 a 1|0 b 5|
  |  6  |  5  |  4  |
  +-----+-----+-----+
```

```
Turno del Jugador 2, b
El jugador b ha robado la carta Locura encarnada

            3
6   Renegado   1    Habilidades: ---
            6

            0
5   Esclavo ignorante   5    Habilidades: ---
            0

            7
3   Tirador de Landuin   3    Habilidades: ---
            6

            0
3   Brujo del olvido   3    Habilidades: ROBAR 2
            3

            3
2   Acechador   3    Habilidades: FUERZA
            9

            2
0   Historiador treciano   5    Habilidades: ROBAR 1
            4

            7
8   Locura encarnada   3    Habilidades: ---
            9

Nueva evaluacion = 0.501
La X temporal seria 0
La Y temporal seria 0
La carta temporal seria Renegado
Nueva evaluacion = 0.601
La X temporal seria 0
La Y temporal seria 0
La carta temporal seria Brujo del olvido
IA: El jugador b ha jugado la carta Brujo del olvido
Robando cartas gracias a la habilidad ROBAR de Brujo del olvido
Carta jugada en la casilla 0-0
Tablero lleno?: true
```

En el último turno, no hay ninguna jugada que pueda resultar relevante para la partida, ya que no se puede vencer ninguna carta en este caso. Vemos cómo efectivamente, el programa comprueba que el tablero está lleno y que el Jugador 2 (b) controla 5 cartas, mientras que el Jugador 1 (a) controla 4. Esto será comunicado a los clientes a continuación, junto al tablero final de partida.

```
Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5
```

```
     1     2     3
  +-----+-----+-----+
  |  0  |  5  |  3  |
1 |3 b 3|9 a 5|7 a 2|
  |  3  |  9  |  7  |
  +-----+-----+-----+
  |  3  |  0  |  5  |
2 |2 b 4|0 a 9|6 b 5|
  |  4  |  0  |  4  |
  +-----+-----+-----+
  |  8  |  4  |  2  |
3 |1 b 5|5 a 1|0 b 5|
  |  6  |  5  |  4  |
  +-----+-----+-----+
Partida finalizada! El ganador es el Jugador 2
```

El bucle de partida acaba, y se vuelve a la clase *Juego*, que es la encargada de preguntar por consola si se desea jugar otra partida. Esto permite cierta comodidad a la hora de volver a jugar partidas o si se pretende testear una gran cantidad de resultados.

```
¿Desea jugar otra partida con los mismos mazos y jugadores? s/n
```

5.2 IA vs *Random*

En esta sección se mostrará que la inteligencia artificial implementada es significativamente mejor que un oponente que realiza movimientos al azar. Para ello se realizará una batería de pruebas en la que se generarán dos mazos aleatorios y se jugará 10 partidas con dichos mazos, uno controlado por el Jugador Inteligente y otro por el Jugador *Random*. Este proceso se realizará 5 veces, cambiando de mazos cada vez, hasta un total de 50 partidas, para así reducir el componente aleatorio (la posibilidad de que los mazos sean de calidades muy diferentes).

Cabe denotar que, a pesar de intentar reducir la varianza cambiando de mazos, el componente aleatorio es muy elevado, tanto en cuestión de calidad de mazo, como en calidad de mano (puede que el Jugador Inteligente robe una mano de todo defensores, o cartas significativamente bajas, o simplemente cartas que son vencidas fácilmente por las del rival), o como en ciertas situaciones especiales en una partida, por ejemplo la posibilidad de solo tener cartas con un valor derecho elevado y que la partida se desarrolle de izquierda a derecha. También hay que tener en cuenta que el jugador que mueve primero tiene ventaja, y que en 100 partidas puede acabar siendo relevante, sobre todo teniendo en cuenta que el jugador *Random* puede jugar una partida especialmente bien de forma casual. Por lo tanto, un resultado muy elevado de victorias (superior al 95%) se considerará lo suficientemente fiable como para asegurar que hay un salto de calidad elevado entre un Jugador *Random* y el Jugador Inteligente implementado.

A continuación se muestran los resultados de las partidas, siendo el Jugador 1 el Jugador Inteligente, y el Jugador 2 el jugador *Random*:

Dupla de mazos número 1: En esta batería de pruebas, el jugador *Random* jugó en primer lugar prácticamente todas las veces, consiguiendo un resultado de 4-5 en muchos casos. Sin embargo, no consiguió derrotar a la IA en ninguna ocasión.

```
Cartas controladas por el jugador 1: 6 GANADOR IA
Cartas controladas por el jugador 2: 3

Cartas controladas por el jugador 1: 5 GANADOR IA
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 7 GANADOR IA
Cartas controladas por el jugador 2: 2

Cartas controladas por el jugador 1: 5 GANADOR IA
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 6 GANADOR IA
Cartas controladas por el jugador 2: 3

Cartas controladas por el jugador 1: 9 GANADOR IA
Cartas controladas por el jugador 2: 0

Cartas controladas por el jugador 1: 5 GANADOR IA
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 9 GANADOR IA
Cartas controladas por el jugador 2: 0

Cartas controladas por el jugador 1: 7 GANADOR IA
Cartas controladas por el jugador 2: 2

Cartas controladas por el jugador 1: 7 GANADOR IA
Cartas controladas por el jugador 2: 2
```

Dupla de mazos número 2: De nuevo, la cantidad de veces que el jugador *Random* jugó primero fue muy elevada.

```
Cartas controladas por el jugador 1: 5 GANADOR IA
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 7 GANADOR IA
Cartas controladas por el jugador 2: 2

Cartas controladas por el jugador 1: 5 GANADOR IA
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 7 GANADOR IA
Cartas controladas por el jugador 2: 2
```

```
Cartas controladas por el jugador 1: 5 GANADOR IA
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 6 GANADOR IA
Cartas controladas por el jugador 2: 3

Cartas controladas por el jugador 1: 8 GANADOR IA
Cartas controladas por el jugador 2: 1

Cartas controladas por el jugador 1: 7 GANADOR IA
Cartas controladas por el jugador 2: 2

Cartas controladas por el jugador 1: 7 GANADOR IA
Cartas controladas por el jugador 2: 2

Cartas controladas por el jugador 1: 7 GANADOR IA
Cartas controladas por el jugador 2: 2
```

Dupla de mazos número 3:

```
Cartas controladas por el jugador 1: 7 GANADOR IA
Cartas controladas por el jugador 2: 2

Cartas controladas por el jugador 1: 7 GANADOR IA
Cartas controladas por el jugador 2: 2

Cartas controladas por el jugador 1: 8 GANADOR IA
Cartas controladas por el jugador 2: 1

Cartas controladas por el jugador 1: 7 GANADOR IA
Cartas controladas por el jugador 2: 2

Cartas controladas por el jugador 1: 8 GANADOR IA
Cartas controladas por el jugador 2: 1

Cartas controladas por el jugador 1: 6 GANADOR IA
Cartas controladas por el jugador 2: 3

Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5 GANADOR RANDOM
```

Tras estudiar el caso en el que el Jugador Inteligente ha perdido su primera partida, se puede observar que se han dado las circunstancias siguientes: Cartas anormalmente bajas por parte del Jugador Inteligente; cartas anormalmente altas por parte del Jugador *Random*; extraño bloqueo de la partida, en el que prácticamente ningún turno se podía vencer ninguna carta (sobre todo debido a las cartas tan elevadas que ponía el Jugador *Random*); dos jugadas clave excepcionalmente buenas por parte del Jugador *Random*, una en la que vencía a una carta elevada del Jugador Inteligente al mismo tiempo que dejaba un valor alto abierto (un 9), y otra en la que cerraba todas sus cartas débiles con una jugada magistral en una casilla clave. En general, tras revisar el mazo del Jugador *Random*,

se puede ver que cuenta con cartas muy elevadas, y a pesar de ellos las partidas anteriores ha sufrido derrotas claras. Se puede considerar un caso aislado de partida, siempre que no ocurran muchos más de estos.

```
Cartas controladas por el jugador 1: 5 GANADOR IA
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 6 GANADOR IA
Cartas controladas por el jugador 2: 3

Cartas controladas por el jugador 1: 5 GANADOR IA
Cartas controladas por el jugador 2: 4
```

Dupla de mazos 4:

```
Cartas controladas por el jugador 1: 6 GANADOR IA
Cartas controladas por el jugador 2: 3

Cartas controladas por el jugador 1: 6 GANADOR IA
Cartas controladas por el jugador 2: 3

Cartas controladas por el jugador 1: 5 GANADOR IA
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 8 GANADOR IA
Cartas controladas por el jugador 2: 1

Cartas controladas por el jugador 1: 6 GANADOR IA
Cartas controladas por el jugador 2: 3

Cartas controladas por el jugador 1: 7 GANADOR IA
Cartas controladas por el jugador 2: 2

Cartas controladas por el jugador 1: 5 GANADOR IA
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 6 GANADOR IA
Cartas controladas por el jugador 2: 3

Cartas controladas por el jugador 1: 7 GANADOR IA
Cartas controladas por el jugador 2: 2

Cartas controladas por el jugador 1: 6 GANADOR IA
Cartas controladas por el jugador 2: 3
```

Dupla de mazos 5:

```
Cartas controladas por el jugador 1: 7 GANADOR IA
Cartas controladas por el jugador 2: 2

Cartas controladas por el jugador 1: 9 GANADOR IA
Cartas controladas por el jugador 2: 0
```

```
Cartas controladas por el jugador 1: 5 GANADOR IA
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 7 GANADOR IA
Cartas controladas por el jugador 2: 2

Cartas controladas por el jugador 1: 8 GANADOR IA
Cartas controladas por el jugador 2: 1

Cartas controladas por el jugador 1: 6 GANADOR IA
Cartas controladas por el jugador 2: 3

Cartas controladas por el jugador 1: 7 GANADOR IA
Cartas controladas por el jugador 2: 2

Cartas controladas por el jugador 1: 5 GANADOR IA
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 9 GANADOR IA
Cartas controladas por el jugador 2: 0

Cartas controladas por el jugador 1: 6 GANADOR IA
Cartas controladas por el jugador 2: 3
```

Tras finalizar la batería de pruebas, se han obtenido 49/50 victorias, es decir, un resultado equivalente a un 98% de victorias. A pesar de que la cantidad de pruebas realizadas no es suficiente como para asegurar una tasa de victorias prácticamente del 100%, es un buen indicativo de la gran diferencia de calidad entre un jugador inteligente y uno aleatorio, sobre todo teniendo en cuenta el significativo factor de aleatoriedad de las pruebas.

5.2 IA vs Jugador Humano

En esta sección se enfrentará a la inteligencia artificial contra varios jugadores humanos de niveles de juego muy diferentes, jugando un mínimo de 10 partidas y mostrando los resultados.

5.2.1 IA vs jugador experto

En este caso se enfrente al Jugador Inteligente contra un jugador de nivel avanzado, el creador del juego y del Jugador Inteligente, cuyos conocimientos del funcionamiento del agente y del juego en si le proporcionan ciertas ventajas a la hora de jugar contra la inteligencia artificial.

A continuación se muestran los resultados de 25 partidas, cambiando de mazo cada 5 partidas. El jugador 2 es el Jugador Inteligente:

```
Cartas controladas por el jugador 1: 5 GANADOR HUMANO
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 6 GANADOR HUMANO
Cartas controladas por el jugador 2: 3

Cartas controladas por el jugador 1: 3
Cartas controladas por el jugador 2: 6 GANADOR IA

Cartas controladas por el jugador 1: 5 GANADOR HUMANO
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 3
Cartas controladas por el jugador 2: 6 GANADOR IA
```

Cambio de mazos:

```
Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5 GANADOR IA

Cartas controladas por el jugador 1: 5 GANADOR HUMANO
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 5 GANADOR HUMANO
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 3
Cartas controladas por el jugador 2: 6 GANADOR IA

Cartas controladas por el jugador 1: 7 GANADOR HUMANO
Cartas controladas por el jugador 2: 2
```

Cambio de mazos:

```
Cartas controladas por el jugador 1: 5 GANADOR HUMANO
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 3
Cartas controladas por el jugador 2: 6 GANADOR IA

Cartas controladas por el jugador 1: 5 GANADOR HUMANO
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5 GANADOR IA

Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5 GANADOR IA
```

Cambio de mazos:

```
Cartas controladas por el jugador 1: 3
Cartas controladas por el jugador 2: 6 GANADOR IA

Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5 GANADOR IA

Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5 GANADOR IA

Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5 GANADOR IA

Cartas controladas por el jugador 1: 6 GANADOR HUMANO
Cartas controladas por el jugador 2: 3
```

Cambio de mazos:

```
Cartas controladas por el jugador 1: 7 GANADOR HUMANO
Cartas controladas por el jugador 2: 2

Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5 GANADOR IA

Cartas controladas por el jugador 1: 6 GANADOR HUMANO
Cartas controladas por el jugador 2: 3

Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5 GANADOR IA

Cartas controladas por el jugador 1: 7 GANADOR HUMANO
Cartas controladas por el jugador 2: 2
```

Conclusión: El Jugador Inteligente ha conseguido un sorprendente resultado de **13-12 a su favor**, a pesar de enfrentarse a un conocedor del juego y del algoritmo que utiliza. La victoria frente al conocimiento humano se puede explicar, entre otras cosas, por la importancia de jugar en primer lugar. Esto es especialmente importante para la táctica implementada, ya que se centra en poner cartas fácilmente recuperables, lo cual es muy efectivo si se comienza con ventaja de cartas en mesa. Si el Jugador Inteligente es el jugador inicial, es muy complicado darle la vuelta a la partida. Esto da lugar a una necesidad del jugador humano de tener un conocimiento avanzado e intentar que el Jugador Inteligente caiga en una trampa en la que pierda más de una carta en una sola jugada. Este tipo de conocimiento podría intentar implementarse en la inteligencia artificial en una versión posterior,

en la que haya una estrategia diferente para la situación en la que se juega primero y la situación en la que empieza el rival.

Una de las características que más han llamado la atención en el testeo, es que la táctica seguida por la inteligencia artificial es muy difícil de batir cuando es la primera en mover, pero cuando empieza en segundo lugar y va por detrás, al ser una táctica con un criterio pesimista (seguramente pierda la carta, con lo cual voy a centrarme en recuperarla fácilmente), le cuesta más ganar la partida. Esto se debe mayormente a que no arriesga en ningún momento, independientemente de cómo vaya el marcador global, y es algo que un jugador avanzado si podría tener en cuenta fácilmente.

5.2.2 IA vs Jugador Avanzado

En este ejemplo, se enfrenta al jugador inteligente contra un sujeto con bastante experiencia en juegos de cartas y juegos de ordenador en general, con práctica en juegos como *Magic: The Gathering*.

Se realizaron solamente cinco partidas, cambiando de mazos cada una de ellas. Los resultados fueron los siguientes:

```
Cartas controladas por el jugador 1: 3
Cartas controladas por el jugador 2: 6 GANADOR IA

Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5 GANADOR IA

Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5 GANADOR IA

Cartas controladas por el jugador 1: 5 GANADOR HUMANO
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 3
Cartas controladas por el jugador 2: 6 GANADOR IA
```

Conclusión: A pesar de las pocas partidas jugadas, se observó cierto patrón en el modo de juego del Jugador Humano. Como parece lógico en principio, el jugador pretendía dejar valores altos abiertos al mismo tiempo que vencía a una de las cartas del rival. Sin embargo, la inteligencia artificial se centraba en recuperar su antigua carta (NO la elevada carta rival), dado que por norma general aun poseía un valor abierto bajo que le permitía recuperarla. Por lo tanto, los valores altos del jugador humano eran más bien irrelevantes, porque estas jugadas de vencer-recuperar se iban sucediendo turno tras turno, y en caso de que la IA fuera por delante, al jugador humano no se le ocurría forma de darle la vuelta a la partida y se quejaba de

tener malas cartas, cuando en realidad la inteligencia artificial estaba usando cartas mucho peores (con valores bajos, adrede), y solo utilizaba las cartas altas cuando realmente necesitaba vencer a una del jugador humano que tuviera un valor alto.
Cuando el jugador humano iba por delante, no se centraba tanto en mantener la ventaja como en dejar valores altos abiertos, de nuevo, con lo cual la inteligencia artificial le podía dar alguna sorpresa venciendo a dos cartas en un turno y poniéndose por delante.

5.2.3 IA vs Jugador Medio

En este ejemplo se enfrente al Jugador Inteligente a un jugador con cierta facilidad para juegos de inteligencia pero con poca experiencia en juegos de cartas o de estrategia de este tipo.

```
Cartas controladas por el jugador 1: 5 GANADOR HUMANO
Cartas controladas por el jugador 2: 4

Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5 GANADOR IA

Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5 GANADOR IA

Cartas controladas por el jugador 1: 6 GANADOR HUMANO
Cartas controladas por el jugador 2: 3

Cartas controladas por el jugador 1: 3
Cartas controladas por el jugador 2: 6 GANADOR IA
```

Cambio de mazos:

```
Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5 GANADOR IA

Cartas controladas por el jugador 1: 3
Cartas controladas por el jugador 2: 6 GANADOR IA

Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5 GANADOR IA

Cartas controladas por el jugador 1: 6 GANADOR HUMANO
Cartas controladas por el jugador 2: 3

Cartas controladas por el jugador 1: 4
Cartas controladas por el jugador 2: 5 GANADOR IA
```

Conclusión: De nuevo, se observa un comportamiento parecido en el jugador humano, que pretende en todo momento dejar libres los valores

más altos posibles, sin tener en cuenta que una vez el Jugador Inteligente haya recuperado dicha carta, le va a resultar extremadamente difícil recuperarla. Esto hace que en los movimientos finales la calidad de las cartas restantes sea insuficiente para darle la vuelta a la partida, mientras que el Jugador Inteligente aún posee potencia suficiente. El marcador se decanta hacia el Jugador Inteligente.

Conclusiones y futuros trabajos

En este apartado se desglosará el resultado del proyecto y se resumirán brevemente las conclusiones a las que se ha llegado a lo largo de la fase de desarrollo y pruebas. Además, se comentará alguna de las posibles formas de expandir el proyecto.

6.1 Diseño

Desde el punto de vista de diseño y tras realizar innumerables pruebas y comparar con "Triple Triad", el mini-juego original, se extraen varias conclusiones, tanto positivas como negativas:

Positivas:

- Aleatoriedad deseada: Se ha conseguido aumentar la re-jugabilidad y el *scope* del juego original, aumentando en su medida justa el factor aleatorio. Esto hace que cada partida sea lo suficientemente diferente como para que unas mecánicas en principio repetitivas no se vuelvan tediosas tan rápidamente como en el juego original.
- Falta de información: El cambio de juego con información completa a juego con cierta incertidumbre, a pesar de complicar el desarrollo del Jugador Inteligente, ha valido la pena desde el punto de vista de diseño, dado que el factor estratégico se ha multiplicado.
- Habilidades: Las habilidades están equilibradas, a pesar de ser pocas, para proporcionar un comportamiento diferente o una táctica única a las cartas en función de su habilidad. Por ejemplo, las cartas de robar aumentan su utilidad en las etapas iniciales, mientras que las cartas con "Fuerza" son más útiles al final de la partida para dar alguna sorpresa o conseguir una carta que de otra forma era imposible de conseguir (un 9).

Negativas:

- Desequilibrio: Hay un ligero problema de desequilibrio, dado que el jugador inicial consigue poner una carta más que el rival, lo cual suele desembocar en una cantidad de victorias "5-4" bastante elevada. Esto no se ha conseguido equilibrar del todo proporcionando una mayor ventaja de cartas al rival, y es un tema que habría que afrontar en las próximas fases del diseño.

Respecto a futuros trabajos, al respecto del diseño hay un par de posibles vías a expandir.

La primera, sería enfocarse en el problema del equilibrado de los jugadores, consiguiendo un sistema de juego muy similar al actual pero minimizando la ventaja del jugador inicial. Para ello, se podría revisar el planteamiento del primer turno, dando más cartas al segundo jugador o realizando ajustes más radicales, como obligar al primer jugador a situar la primera carta en el centro. Aunque esta última podría ser una buena solución, cambiaría demasiado el juego tal como se concibe actualmente.

La segunda vía de expansión sería la creación de nuevo contenido y nuevas mecánicas. Desde especificar bien la incompleta mecánica "Eco" hasta inventar nuevas habilidades que den variedad al juego. De la misma forma, se podría crear nuevo contenido en forma de nuevas cartas, o incluso nuevos tableros jugables, lo cual supondría varios cambios más, tanto en la implementación como en el Jugador Inteligente.

6.2 Implementación

La implementación del software no ha sufrido complicaciones importantes, exceptuando quizá el momento en el que se quiso realizar el cambio de arquitectura del programa de "monolítico" a distribuido, proceso que llevó a cambiar mucho la arquitectura inicial e investigar la mejor opción posible en cuanto al uso de *Multi-Threading* junto a conexiones por *Sockets*.

Un punto a favor de la implementación es la facilidad con la que se podrían añadir nuevas habilidades, dado que desde el punto de vista del equilibrado del Jugador Inteligente bastaría con ajustar la función de evaluación, y en cuanto a implementación existen facilidades para añadir nuevos comportamientos.

Un punto en contra es que la interfaz, a pesar de ser bastante intuitiva para los pocos recursos invertidos, puede resultar menos usable de lo pretendido para un juego (aunque luego proporciona algunas facilidades como control de errores o repetir partidas). Esto, sin embargo, es normal siendo el objetivo inicial la programación de un prototipo jugable, en lugar de un juego completo, centrándose más en los aspectos de diseño y de desarrollo de un Jugador Inteligente.

Respecto a futuros trabajos desde el punto de vista de la implementación, la principal vía de desarrollo podría ser conseguir un juego completo en lugar de un prototipo, es decir, completarlo con una interfaz gráfica usable y un sistema de

juego remoto más elaborado. Otra herramienta a conseguir podría ser un sistema de creación de mazos personalizables.

6.3 Jugador Inteligente

La inteligencia artificial implementada ha resultado ser incluso mejor de lo pretendido en un principio, dado que el objetivo inicial era conseguir desafiar a jugadores de nivel medio, y resulta que el desafío está más orientado a jugadores de nivel alto y con experiencia en este tipo de juegos. Hay que tener en cuenta, de todas formas, que los resultados pueden haber estado sesgados por el fallo de diseño cuya consecuencia es la ventaja obtenida por el jugador inicial, pero se estaría asumiendo que en la totalidad (o al menos gran mayoría) de las pruebas ha habido un sesgo estadístico importante a favor del Jugador Inteligente.

Si hubiera que destacar un hilo a seguir para futuras mejoras del Agente Inteligente, sería sin duda la división de la estrategia actual en dos estrategias diferentes, siendo la actual la estrategia a seguir cuando el Jugador Inteligente es el jugador inicial, y cambiando la función de evaluación para la segunda estrategia (cuando el rival comienza), haciéndola más agresiva y arriesgando más para conseguir la ventaja necesaria. Esto se debe al carácter defensivo-pesimista de la estrategia actual, que pierde sentido cuando la situación es muy desfavorable y la necesidad de arriesgar es mucho mayor.

Desde el punto de vista del diseño de la IA, la principal vía para futuros trabajos sería la de conseguir un agente capaz de jugar al próximo nivel, mucho más cercano a jugadores expertos, sin depender de quién es el jugador inicial. Para ello, la clave podría estar en realizar dos comportamientos diferentes, uno optimista (para cuando va por detrás y necesita arriesgar) y uno pesimista (el actual). Hay otras mejoras sugeridas en apartados anteriores sobre cómo mejorar el Jugador Inteligente. Otras posibles mejoras podrían suponer cambiar totalmente el agente inteligente y utilizar otro tipo de inteligencia artificial, así que no se tratarán en este documento.

Gestión del proyecto

En este apartado se tratará todo lo relativo a la gestión de tiempo y recursos necesarios para la realización del proyecto.

7.1 Planificación inicial

Inicialmente, el proyecto se orientó más hacia la realización de un Jugador Inteligente con aprendizaje automático, dejando en segundo plano la parte del diseño del juego y la implementación del prototipo. Sin embargo, a lo largo del proyecto, tanto el diseño como la implementación fueron cobrando relevancia, quedando a un nivel mucho más importante del que se había pensado, y ocupando alrededor de 2/3 del tiempo dedicado al proyecto.

El planteamiento inicial del proyecto se podría dividir en 3 etapas:

- **Diseño del juego:**
  Una parte leve pero necesaria que identificara las mecánicas básicas a implementar, sin siquiera crear una base de cartas equilibrada ni otras necesidades que se fueron descubriendo a lo largo del desarrollo.
- **Implementación:**
  Aunque era obviamente necesaria, la implementación se veía como una forma de proporcionar un vehículo de prueba para el desarrollo del agente inteligente. Sin embargo, a lo largo del proyecto acabó ocupando alrededor de un 40% del tiempo, y se implementaron comodidades y funcionalidades que no estaban pensadas en un principio.
- **Diseño e implementación de la inteligencia artificial:**
  En un principio se pretendía crear un sistema de IA con aprendizaje, pero los tutores ya avisaron de la magnitud del proyecto tan solo con el desarrollo del juego y de una IA sin aprendizaje, así que la idea se descartó en los primeros meses y se re-equilibró la carga.

7.2 Planificación final

Tras los cambios acontecidos a lo largo del desarrollo del proyecto, éste se podría dividir en 4 etapas de una carga de trabajo similar:

- **Diseño del juego (~25%):**
  El diseño del juego ocupó la primera parte del proyecto y no acabó de cerrarse hasta bien avanzado el desarrollo del prototipo, aunque la idea ya se había formado con antelación y llevaba un tiempo queriendo realizar una adaptación del juego "Triple Triad".
- **Creación de un prototipo funcional (~25%):**
  Para la creación del primer prototipo, se definieron objetivos como los de conseguir un sistema de clases fácilmente adaptables a futuros cambios del diseño, dado que la parte de diseño no acabó de cerrarse hasta las últimas etapas del desarrollo. Se dejaron de lado cosas como la creación de un agente inteligente básico, utilizando un generador de movimientos aleatorios en su lugar para realizar las pruebas. El sistema de clases era bastante "monolítico", sin ningún tipo de división en módulos, centrándose en la versatilidad y funcionalidad en su lugar.
- **Creación de un prototipo modular distribuido (~20%):**
  Una vez presentado el primer prototipo a los tutores, se decidió dedicar menos tiempo al diseño del jugador inteligente y más al refinamiento del programa actual, es decir, la división del sistema en módulos y la orientación a un modelo distribuido y escalable. La adaptación del código anterior fue costosa, ya que hubo muchas cosas que cambiaron con la inclusión de *Threads* y *Sockets*.
- **Diseño e implementación del Jugador Inteligente (~30%):**
  Diseño e implementación se realizaron de forma iterativa, es decir, perfilando un primer diseño e implementándolo, observando los fallos o posibles mejoras a realizar, y volviendo a iterar. De esta forma, se pasó de una IA reactiva basada en reglas simples, a una función de evaluación con algo más de complejidad y unos resultados mejores de lo esperado.

A continuación se muestra el gráfico *Gannt* que muestra la distribución aproximada del trabajo a lo largo del tiempo.

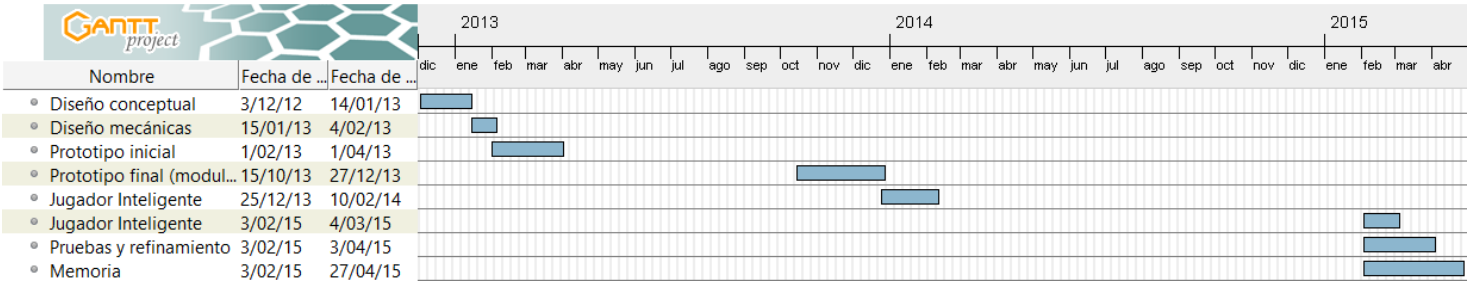

Figura 12: Gráfico *Gannt* del proyecto.

7.3 Presupuesto

En esta sección se realizará un cálculo aproximado del presupuesto necesario para desarrollar un proyecto de esta magnitud. Para empezar, se realiza una estimación del sueldo medio de un programador junior utilizando datos de *Infojobs* del año 2014 en España, de los cuales se puede extrapolar que 15.000€/año es un sueldo medio bastante aproximado. Dicho sueldo se traduce en 6,25€ la hora, que es el valor que se utilizará para estimar el coste del proyecto.

FASES	ESTIMACIÓN HORAS	ESTIMACIÓN COSTE (€)
Diseño	60	375
Prototipo Inicial	60	375
Prototipo Final	50	312’5
Jugador Inteligente	80	500
Documentación	50	312’5
TOTAL	**300**	**1875€**

El resultado muestra que el proyecto requeriría alrededor de 1875€ en gastos de personal. A continuación se calculará de forma aproximada la amortización de los bienes utilizados para el desarrollo del proyecto.

ARTÍCULO	PRECIO (€)	MÍNIMO MESES AMORT.	% MÁXIMO AMORT. ANUAL	MESES	TOTAL AMORT. (€)
Asus N751JK-T7246H i7-4710HQ 8GB/1TB/GTX 850M/17.3"	1099.00	48	25%	10	109.90
MS Office 2013	539.00	48	33%	10	53.90

El presupuesto hasta ahora sería pues de 1875.00€ + 163.80, haciendo un total de **2038.80€**. Sobre dicho total, se aplicaría un cálculo del 20% para reflejar los gastos indirectos.

El presupuesto final sería por lo tanto de 2038.80 + 407.76 = **2446.56€**.

Referencias bibliográficas

Las referencias bibliográficas mostradas a continuación muestran algunas de las webs, libros (en formato .pdf) y documentos recopilados a lo largo de la realización del proyecto.

8.1 Diseño

http://finalfantasy.wikia.com/wiki/Triple_Triad

www.tripletriadflashonline.com/es/play

http://boardgamegeek.com/boardgame/15957/final-fantasy-viii-triple-triad

8.2 Programación

http://docs.oracle.com/javase/tutorial/essential/concurrency/

http://docs.oracle.com/javase/tutorial/networking/

http://www.dlsi.ua.es/asignaturas/sid/JSockets.pdf

http://www.tutorialspoint.com/java/java_thread_communication.htm

8.3 Inteligencia Artificial

http://www.upv.es/sma/teoria/agentes/is%20it%20an%20agent-franklin.pdf

http://books.google.es/books?hl=es&lr=&id=LFsVFS3OG_gC&oi=fnd&pg=PP1&dq=risk+uncertainty+and+profit+risk+vs+uncertainty&ots=pM3-Bif0ZT&sig=q5dr9OVsI3ysxIJZE4D8Xx2lBUU#v=onepage&q=risk%20uncertainty%20and%20profit%20risk%20vs%20uncertainty&f=false

http://www.uv.mx/aguerra/documents/2013-ia2-01.pdf

http://www.fdi.ucm.es/profesor/jpavon/doctorado/arquitecturas.pdf

http://es.wikipedia.org/wiki/Agente_inteligente_%28inteligencia_artificial%29

http://ccia.ei.uvigo.es/docencia/IA/1213/transparencias/Tema4-mini.pdf

http://www.cs.upc.edu/~bejar/ia/transpas/teoria/4-SBC4-rbayesianas.pdf

http://www.mat.ucm.es/~bvitoria/Archivos/a_dt_UCM.pdf

http://www.eumed.net/tesis-doctorales/2006/erbr/2p.htm

http://www.monografias.com/trabajos75/problematica-juegos-inteligencia-artificial/problematica-juegos-inteligencia-artificial2.shtml

Anexos

9.1 Manual de usuario

En esta sección se explicará cómo ejecutar la aplicación y entender la interfaz.

9.1.1 Ejecución de la aplicación

Se comentarán dos posibles vías de ejecución de la aplicación. La primera forma sería a través del entorno "Eclipse"[10], asumiendo que el usuario lo tiene instalado y entiende el funcionamiento básico del programa. La segunda forma sería ejecutándolo por consola (en el ejemplo se usa *Windows*).

- **Mediante *Eclipse*:**
  Una vez añadido el proyecto a *Eclipse*, para ejecutar el programa con la configuración necesaria, habrá que acceder a "Run Configuration". Una de las formas de acceder es haciendo *click* derecho sobre el proyecto a ejecutar y pulsando en la opción adecuada, como se puede ver en la figura 13.

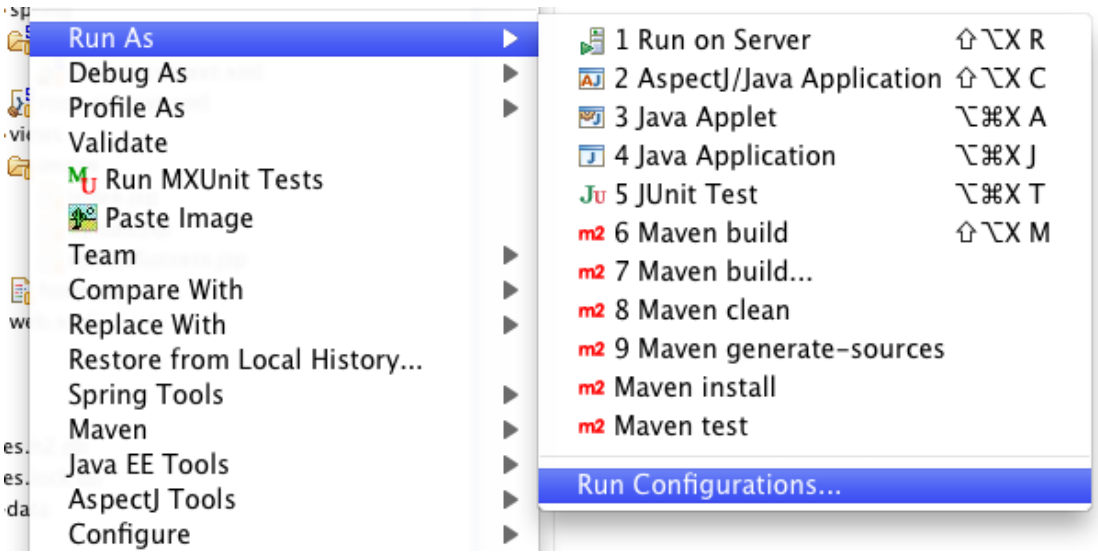

Figura 13: Cómo acceder a la configuración de ejecución.

[10] (Eclipse). Entorno de desarrollo.

A continuación, accederemos a una ventana como la que se muestra en la de la figura 14. Allí habrá que hacer *click* en la clase que queremos ejecutar (debe ser una clase main) y acceder a su pestaña “Arguments”, donde se introducirán los parámetros de ejecución necesarios.

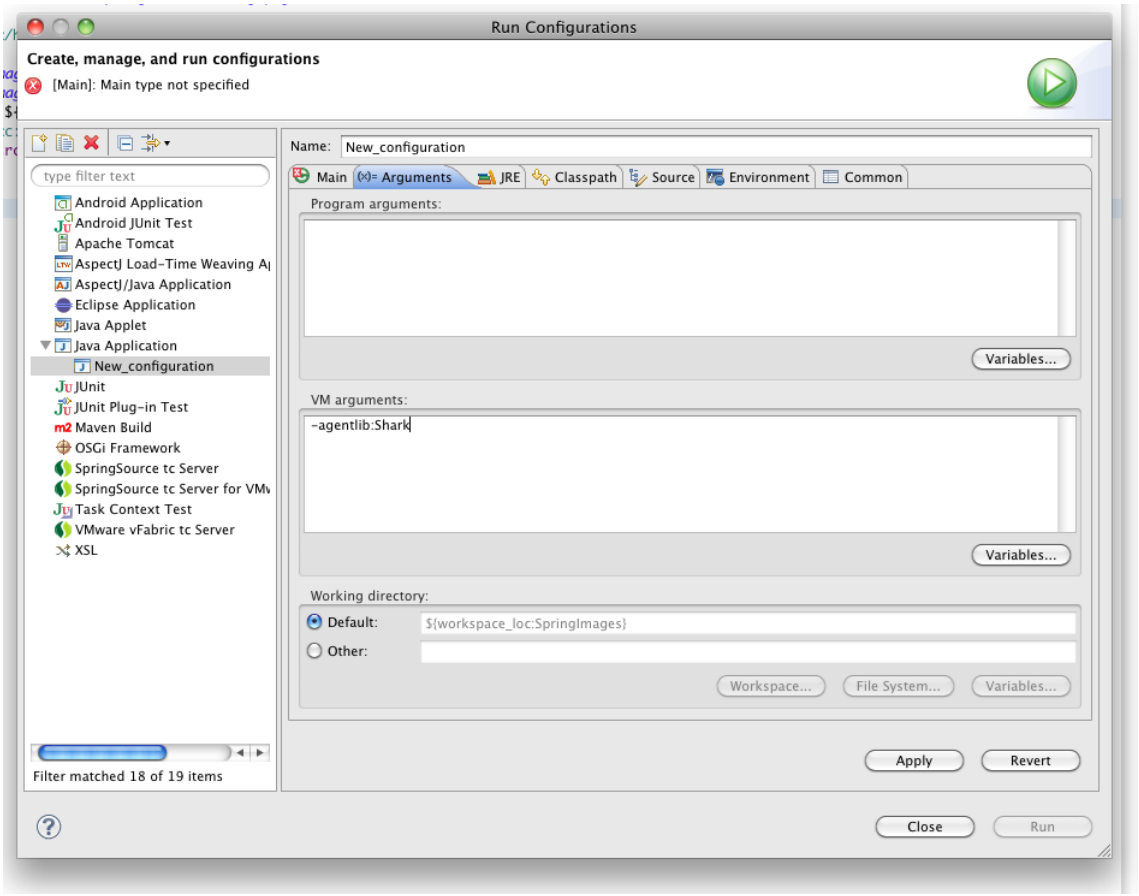

Figura 1412: Ventana de configuración.

Seguidamente, habrá que establecer los parámetros en función del tipo de partida que se desee y de las opciones de ejecución.

- Los parámetros para la clase *ServerMain* son los siguientes:
    - numPuerto: Indica el número del primer puerto que esperará las conexiones provenientes de *Motor* y *Client*. Se puede utilizar, por ejemplo, el puerto 5000.

- Los parámetros para la clase *Client* son los siguientes:
    - txtBaseDatos: Indica el archivo de texto que contiene la base de datos de cartas. La base de datos es proporcionada junto a la aplicación, y su nombre por defecto es "spoilerTXTtoJAVA.txt".
    - Dirección del host: Puede ser tanto una IP remota como el sufijo "localhost", que indica que la conexión se realiza al mismo *host* que ejecuta la aplicación.
    - numPuerto: Indica el número del primer puerto al que se conectará. Se debe utilizar el mismo puerto indicado en la ejecución de *ServerMain*, por ejemplo, el puerto 5000.

- Los parámetros para la clase *Juego* son los siguientes:
    - txtBaseDatos: Indica el archivo de texto que contiene la base de datos de cartas. La base de datos es proporcionada junto a la aplicación, y su nombre por defecto es "spoilerTXTtoJAVA.txt".
    - Dirección del host: Puede ser tanto una IP remota como el sufijo "localhost", que indica que la conexión se realiza al mismo *host* que ejecuta la aplicación.
    - numPuerto: Indica el número del primer puerto al que se conectará. Se debe utilizar el siguiente al puerto indicado en la ejecución de *ServerMain*, por ejemplo, el puerto 5001 (en caso de que se haya usado el 5000 anteriormente).
    - tipoDeJugador1: Indica el tipo de jugador del primer cliente. Se le puede asignar una de las opciones siguientes: Random (La aplicación realizará jugadas aleatorias), IA (el jugador será controlado por el Jugador Inteligente implementado) o Humano (el jugador será un jugador humano que introducirá los datos por pantalla en el módulo *Client*).

- TipoDeJugador2: Indica el tipo de jugador del segundo cliente, con las mismas opciones que en el caso anterior.

Un ejemplo de ejecución podría ser el siguiente, en el cual un jugador humano juega contra la inteligencia artificial en el mismo ordenador, utilizando como puerto inicial el 5000.

- ServerMain: 5000
- Client: spoilerTXTtoJAVA.txt localhost 5000
- Juego: spoilerTXTtoJAVA.txt localhost 5001 Humano IA

Tras haber realizado la configuración de los parámetros, se procede a ejecutar cada una de las clases *main*, por ejemplo en este orden *Server-> Juego-> Client1-> Client2*. Se puede ejecutar en cualquier otro orden, siempre que la clase *Server* se ejecute en primer lugar.

- **Por consola:**
  Estando en *Windows*, abra cuatro ventanas de consola (también llamadas "Símbolo del sistema") diferentes. Para ello, puede acceder a la ventana "Ejecutar" o "Buscar" que se encuentra en antiguas y nuevas versiones de *Windows*, respectivamente, al hacer *click* en el símbolo de *Inicio*. Las figuras 15 y 16 muestran ambas opciones.

Figura 15: Ventana "Ejecutar" en sistemas operativos *Windows* antiguos.

Figura 16: Ventana de búsqueda en sistemas *Windows* modernos.

Una vez en la ventana, escriba “cmd” y presione la tecla de retorno. Realice este proceso cuatro veces, hasta tener cuatro ventanas diferentes.

A continuación, acceda a la ruta que contiene los archivos “.jar” proporcionados con la entrega. Ahora, en cada una de las ventanas ejecute una de las siguientes líneas de comando:

- java –jar Server.jar (parámetros de *Server*, como han sido indicados en el punto anterior)
- java –jar Motor.jar (parámetros de *Motor*, como han sido indicados en el punto anterior)
- java –jar Client.jar (parámetros de C*lient*, como han sido indicados en el punto anterior)
- java –jar Client.jar (parámetros de C*lient*, como han sido indicados en el punto anterior)

Un posible ejemplo de ejecución, el mismo ejemplo que en el punto anterior, sería el siguiente:

- java –jar Server.jar 5000
- java –jar Motor.jar localhost 5001 Humano IA
- java –jar Client.jar localhost 5000
- java –jar Client.jar localhost 5000

Las instrucciones para ejecutar la aplicación en Linux y Mac deberían ser las mismas que en el ejemplo anterior para Windows, ejecutadas desde la consola pertinente a cada sistema.

9.1.2 Comprender la interfaz

Al ejecutar la aplicación, lo primero que tendrá que hacer será proporcionar un nombre representativo del jugador en cada uno de los clientes ejecutados. Una vez asignados los nombres, la información intercambiada (mazo generado y nombre) con el servidor podrá ser visualizada en la consola del módulo *Server*.

Cada consola de cliente representará a un jugador, mostrando la información relativa a la partida que dicho jugador debe saber.

El usuario no tendrá que realizar ninguna otra interacción con la interfaz hasta que finalice la partida, a menos que el cliente sea del tipo "Jugador Humano". En dicho caso, el jugador tendrá que introducir por consola cada jugada cuando se le solicite, utilizando el formato siguiente:

- Número que representa la posición en mano (de 1 a X, siendo X el tamaño de la mano) de la carta a jugar.
- Número X, representando la columna de la casilla donde se desea jugar la carta, en el rango 1-3.
- Número Y, representando la fila de la casilla donde se desea jugar la carta, en el rango 1-3.

En la consola del módulo *Motor* se mostrará información variada del transcurso de la partida. También en dicho módulo se pedirá por consola, al finalizar la partida, si se desea jugar otra partida con los mismos jugadores y mazos.