



UNIVERSIDAD DE GRANADA

E.T.S de Ingenierías Informática y de Telecomunicación, Facultad de Ciencias

DOBLE GRADO EN INFORMÁTICA Y MATEMÁTICAS

TRABAJO DE FIN DE GRADO

Bardtastic: Diseño e implementación de un *deckbuilder* narrativo con IA basada en exploración de árboles

Presentado por:
Arturo Cáceres Arias

Curso académico 2024-2025

Bardtastic: Diseño e implementación de un *deckbuilder* narrativo con IA basada en exploración de árboles

Arturo Cáceres Arias

Arturo Cáceres Arias *Bardtastic: Diseño e implementación de un deckbuilder narrativo con IA basada en exploración de árboles.*

Trabajo de fin de Grado. Curso académico 2024-2025.

Responsable de tutorización

Antonio Miguel Mora García
Departamento de Teoría de la Señal, Telemática y Comunicaciones
Lidia Fernández Rodríguez
Departamento de Matemática Aplicada

Doble Grado en Informática y Matemáticas

E.T.S de Ingenierías
Informática y de
Telecomunicación,
Facultad de Ciencias

UGR

DECLARACIÓN DE ORIGINALIDAD

D./Dña. Arturo Cáceres Arias

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2024-2025, es original, entendido esto en el sentido de que no he utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 10 de noviembre de 2025

Fdo: Arturo Cáceres Arias

A mi familia y amigos, por aguantarme y acompañarme durante todo este proceso.

En especial, gracias a Marta, lo mejor que me llevo del DGIIM. T#T.

Gracias a Aicha por ser la mejor ex y playtester que se puede pedir.

Gracias también a Lin, por ponerle banda sonora a tantas tardes escribiendo como si se me acabase el tiempo.

Nos vemos en el próximo jueguito.

Agradecimientos

En primer lugar, me gustaría agradecer a la Universidad de Granada por haberme ofrecido la oportunidad de formarme en un entorno exigente y enriquecedor. En particular, a todos los docentes del Doble Grado de Ingeniería Informática y Matemáticas, por su paciencia y dedicación para transmitir sus conocimientos y pasión.

A mis tutores, Lidia Ortega y Antonio Miguel Mora, por su constante orientación, tiempo y ánimo a lo largo de este proyecto.

A la incubadora de Power Up, por ofrecerme su impulso y espacio para dar forma a estas ideas. Muy especialmente gracias a Laura, por su apoyo, confianza y entusiasmo.

Por último, gracias también a todas aquellas personas que, de una forma u otra, han contribuido a que este trabajo comience con buen pie: por sus consejos, su compañía o simplemente por recorrerme detrás de cada línea de código hay algo más importante, las personas que te acompañan mientras la escribes.

Summary

This project presents Bardtastic, a narrative deck-building game developed as a prototype within the framework of a Final Degree Project in the Double Degree in Computer Engineering and Mathematics. The work combines game design, software engineering, artificial intelligence, and mathematical modelling to create a complete playable prototype where the player competes against an AI-controlled bard by constructing verses, managing emotions, and capturing the audience's attention.

From the mathematical perspective, the project frames the decision-making process of the rival within game theory, formally characterising the encounter as a finite, adversarial, turn-based game. Concepts such as strategies, expected utilities, and rational choice are introduced, and the link between minimax reasoning and equilibrium behaviour is analysed. Although Bardtastic is not a strict zero-sum game, this theoretical foundation provides a rigorous framework for structuring the agent's behaviour and for motivating the use of heuristic search guided by an approximate utility function.

The project includes the full pipeline from conceptual design and paper prototyping to an implementation in Unreal Engine based on a modular architecture that separates gameplay, interface, and AI logic. A deliberative agent was developed to act as the rival, using a depth-limited search guided by a heuristic utility function that balances immediate audience gain, narrative progress, and hand-cycling potential. This approach achieves rational but not omniscient behaviour, offering a believable and strategically coherent opponent grounded in the principles of bounded rationality.

The results validate the core mechanics and demonstrate that well-structured, extensible software enables future expansion of the game, including new cards, rules, and more advanced AI strategies. Potential future work includes refining the narrative progression, improving the AI's reasoning, formalising the mathematical model of the encounter, and extending the prototype toward a full Vertical Slice suitable for presentation to publishers.

Índice general

Agradecimientos	V
Summary	VII
Introducción	XIII
1 Motivación y contexto	XIII
2 Objetivos del trabajo	XIII
3 Estructura del documento	XIV
1 ¿Qué es un juego?	1
1.1 Teoría de Juegos	1
1.1.1 Definición	1
1.1.2 Tipos de juegos	2
1.1.3 Representación formal	2
1.2 Pero, ¿qué es realmente un juego?	4
1.2.1 Juegos como actividades culturales y sociales	4
1.2.2 Juegos como superación de obstáculos	5
1.2.3 La necesidad de un vocabulario para diseñar	5
1.2.4 Definición operativa	6
2 Utilidad, soluciones y equilibrio	8
2.1 Utilidad	8
2.1.1 Utilidad esperada	8
2.2 Concepto de solución	9
2.2.1 Principales conceptos de solución	9
2.3 Dominación y racionalización	9
2.3.1 Estrategias dominadas	10
2.3.2 Eliminación iterada	10
2.4 Equilibrio de Nash	10
2.4.1 Juegos de suma cero	13
2.5 Eficiencia de Pareto	15
2.6 Equilibrio bayesiano	15
3 Ejemplos de juegos	17
3.1 Juegos estáticos	17
3.1.1 Juegos de información completa	17
3.1.2 Juegos de información incompleta	20
3.2 Juegos dinámicos	22
3.2.1 Juegos de información completa	22
3.2.2 Juegos de información incompleta	24
4 Algoritmos para IA en juegos	27
4.1 Agentes	27
4.2 Algoritmos para juegos con información completa	27
4.2.1 Minimax	27
4.2.2 Poda α - β	29

4.2.3 Heurísticas	32
4.3 Algoritmos para juegos con información incompleta	33
4.3.1 Monte Carlo	33
4.3.2 Algoritmos Evolutivos y Genéticos	34
5 Nuestro caso: <i>Bardtastic</i>	36
5.1 Descripción general del juego	36
5.2 Dinámica de los enfrentamientos	37
5.2.1 Desarrollo de un turno	38
5.2.2 Fin de la partida	38
5.2.3 Tipos de cartas	38
5.3 Clasificación teórica del juego	41
5.4 Agente desarrollado: El rival	41
5.4.1 Función de utilidad (criterios de valoración)	41
5.4.2 Ámbitos de decisión	42
5.4.3 Complejidad computacional del agente	44
6 Estado del arte	45
6.1 Diseño y género	45
6.1.1 Origen y evolución del <i>deckbuilding roguelike</i>	45
6.1.2 <i>Slay the Spire</i>	46
6.1.3 <i>Balatro</i>	47
6.1.4 <i>Storyteller</i>	48
6.2 Síntesis comparativa	48
6.2.1 Conclusiones	49
6.3 Artístico y temático	49
7 Planificación y diseño	51
7.1 Diagrama de <i>Gantt</i>	51
7.2 Presupuesto	52
7.3 Metodología de diseño y desarrollo	52
7.4 Análisis de requisitos	53
7.4.1 Requisitos funcionales (RF)	53
7.4.2 Requisitos no funcionales (RNF)	54
7.4.3 Modelo de casos de uso	54
7.4.4 Matriz de trazabilidad UC ↔ RF	57
7.4.5 Criterios de verificación	57
7.5 Diseño del juego	57
7.5.1 Pilares de diseño	57
7.5.2 Referencias de diseño	58
7.5.3 Prototipo en papel	59
7.6 Diagrama de clases	62
7.6.1 Alcance y criterios de modelado	62
8 Herramientas	65
8.1 Justificación de la elección tecnológica	65
8.2 Prototipado y velocidad de iteración	65
8.3 <i>Pipeline</i> de arte y contenido	66

Índice general

8.4 Diseño 2D y UI	66
8.5 Audio	67
8.6 Herramientas de producción	67
8.7 Conclusiones	67
9 Software desarrollado	68
9.1 Descripción general	68
9.2 Enfrentamientos	68
9.2.1 Interfaz de usuario	69
9.2.2 Cartas	71
9.2.3 Jugadores	73
9.3 Agente Rival	74
9.3.1 Arquitectura y desacoplamiento	74
9.3.2 Flujo de decisión en turno	75
9.3.3 Notas de complejidad y rendimiento	75
9.3.4 Puntos de extensión	75
9.4 Eventos	76
9.4.1 Arquitectura general	76
9.4.2 Lógica del evento	76
9.4.3 Extensibilidad del sistema	77
9.4.4 Integración con la progresión del juego	77
9.5 Sistema de guardado	78
9.5.1 Modelo general en Unreal	78
9.5.2 Guardado actual: SavedRun	78
9.5.3 Múltiples guardados lógicos	78
9.5.4 Ciclo de vida de guardado y carga	79
9.5.5 Rendimiento y restricciones de plataforma	79
9.5.6 Estado actual y ampliaciones previstas	79
9.6 Menú principal	80
10 Evaluación y pruebas	81
10.1 Objetivos de la evaluación	81
10.2 Estrategia empleada (prototipo)	81
10.2.1 Pruebas exploratorias	81
10.2.2 <i>Playtesting</i> supervisado	82
10.2.3 Trazas y depuración	82
10.3 Cobertura funcional alcanzada	82
10.4 Casos de prueba manuales (muestra)	83
10.5 Limitaciones del enfoque actual	83
10.6 Plan de formalización de pruebas (<i>Vertical Slice</i>)	83
10.6.1 Niveles y tipos de prueba	83
10.6.2 Herramientas de Unreal recomendadas	84
10.6.3 Matriz mínima de pruebas (RF → TC)	84
10.6.4 Criterios de aceptación y métricas	84
10.6.5 Procedimiento de <i>playtest</i> estructurado	84
11 Conclusiones y trabajo a futuro	85
11.1 Síntesis y logros del proyecto	85

Índice general

11.2 Valoración de los objetivos	85
11.3 Trabajo a futuro	86
Bibliografía	87

Introducción

Los videojuegos constituyen hoy un espacio privilegiado para la experimentación en diseño interactivo y en la aplicación práctica de algoritmos de decisión. Como medios computacionales complejos, permiten estudiar cómo reglas formales, recursos limitados y elecciones encadenadas dan lugar a comportamientos estratégicos. Dentro de este amplio panorama, los juegos basados en cartas y en estructuras combinatorias ofrecen un entorno especialmente adecuado para analizar procesos de razonamiento y para introducir agentes capaces de actuar bajo incertidumbre.

1. Motivación y contexto

En la última década, el auge del género *deckbuilder roguelike* ha transformado la escena del videojuego independiente, combinando construcción de mazos, progresión procedural y toma de decisiones estratégicas. Títulos como *Slay the Spire* o *Balatro* han demostrado la viabilidad de sistemas basados en probabilidad y optimización. Sin embargo, en la mayoría de estos títulos, la inteligencia artificial cumple un papel puramente reactivo o estadístico.

En este marco surge la motivación de crear un prototipo de *deckbuilder* narrativo con mecánicas originales que sirva como base para un desarrollo futuro. Paralelamente, se explora cómo un agente deliberativo, inspirado en los modelos de decisión racional de la Teoría de Juegos, puede integrarse en este tipo de experiencias para generar comportamientos adaptativos, coherentes y competitivos frente al jugador humano.

2. Objetivos del trabajo

El presente Trabajo de Fin de Grado se articula en torno a dos objetivos generales, coherentes con la doble formación en Matemáticas e Ingeniería Informática. A partir de ellos se derivan una serie de subobjetivos específicos que guían el desarrollo teórico y práctico del proyecto.

Objetivo general 1: Fundamentación matemática y teórica

Estudiar y presentar los conceptos esenciales de la teoría de juegos que permiten describir formalmente un juego, sus estrategias y sus equilibrios, y utilizar este marco para clasificar el enfrentamiento que plantea *Bardtastic*. Esta fundamentación proporciona una justificación teórica al diseño del agente de inteligencia artificial y a la elección de los modelos de decisión empleados.

Subobjetivos asociados:

- Introducir las definiciones de juego, utilidad, estrategias puras y mixtas.
- Exponer los conceptos de equilibrio de Nash, estrategias dominadas y resultados relevantes como el teorema del minimax.
- Clasificar formalmente distintos tipos de juegos (estáticos, dinámicos, de información completa o incompleta, de suma cero o no suma cero).
- Enmarcar *Bardtastic* dentro de esta clasificación y analizar las implicaciones para el diseño del agente rival.

Objetivo general 2: Diseño e implementación del prototipo y del agente

Desarrollar un prototipo funcional del videojuego *Bardtastic*, integrando narrativa, mecánicas de cartas y progresión *roguelike*, así como un agente de inteligencia artificial que utilice exploración de árboles y heurísticas inspiradas en la Teoría de Juegos para tomar decisiones racionales bajo información parcial.

Subobjetivos asociados:

- Diseñar las reglas del juego, los tipos de carta y las mecánicas centrales (rimas, atención, emociones, historia).
- Construir una arquitectura modular en Unreal Engine que separe lógica de juego, interfaz y comportamiento del agente.
- Implementar un agente deliberativo basado en *Depth-First Search* y en una función de utilidad que evalúa atención, progreso narrativo y ciclo.
- Realizar pruebas funcionales y sesiones de *playtesting* que permitan ajustar tanto el juego como el comportamiento del agente.

En conjunto, estos objetivos combinan la modelización matemática de la decisión estratégica con los principios de la ingeniería del software y el diseño de videojuegos, dando lugar a un prototipo jugable fundamentado teóricamente y técnicamente sólido.

3. Estructura del documento

El documento se organiza de forma progresiva, comenzando por los fundamentos teóricos y avanzando hacia el diseño, la implementación y la validación del prototipo.

El **Capítulo 1** introduce las nociones básicas de la Teoría de Juegos necesarias para el desarrollo del trabajo. En él se presentan las definiciones formales de juego, estrategias, utilidades y resultados preliminares que permiten describir interacciones estratégicas de manera rigurosa.

El **Capítulo 2** profundiza en los conceptos de equilibrio, especialmente el equilibrio de Nash, así como en criterios de optimalidad como las estrategias dominadas y el teorema del minimax. Este marco sirve de base matemática para el análisis del agente rival utilizado en el videojuego.

El **Capítulo 3** recopila distintos ejemplos de juegos estáticos y dinámicos que ilustran la teoría expuesta. Su propósito es mostrar cómo los conceptos anteriores se aplican en situaciones concretas y cómo pueden modelarse problemas reales de decisión.

El **Capítulo 4** presenta varios algoritmos clásicos empleados en la resolución de juegos y en la toma de decisiones computacional, incluyendo métodos basados en exploración de árboles y heurísticas. Estas técnicas inspiran directamente el diseño del agente implementado posteriormente.

El **Capítulo 5** introduce en detalle el proyecto *Bardtastic*. Se describen sus reglas, el funcionamiento de sus mecánicas principales y su clasificación formal desde la Teoría de Juegos, estableciendo el puente entre los conceptos teóricos y el caso de estudio.

El **Capítulo 6** presenta el estado del arte tanto en diseño de videojuegos como en enfoques de inteligencia artificial aplicados a juegos. Se revisan los géneros que inspiran el proyecto, los referentes mecánicos y narrativos, así como los principales modelos de agentes deliberativos empleados en entornos interactivos.

El **Capítulo 7** describe la planificación del desarrollo y la metodología empleada. Se detallan las fases de trabajo, la organización del proyecto, las herramientas utilizadas y el uso de enfoques ágiles adaptados a un proceso de producción individual.

El **Capítulo 8** expone las decisiones de diseño que dan forma al prototipo: los pilares conceptuales, las mecánicas centrales, la identidad narrativa y la transición desde el prototipo en papel hasta su implementación digital. Este capítulo establece el marco conceptual que guía las elecciones técnicas posteriores.

El **Capítulo 9** describe la implementación técnica del prototipo en Unreal Engine, con especial atención a la arquitectura del software, el sistema de cartas y el comportamiento del agente rival.

El **Capítulo 10** presenta la evaluación del prototipo mediante pruebas funcionales y sesiones de *playtesting*, analizando tanto el rendimiento del agente como la adecuación de las mecánicas.

Finalmente, el **Capítulo 11** recoge las conclusiones generales del trabajo y propone varias líneas de continuación, incluyendo la extensión del prototipo hacia una *Vertical Slice*, la ampliación del contenido jugable y el diseño de agentes más sofisticados.

Capítulo 1: ¿Qué es un juego?

“La cultura humana surge del juego y, a su vez, se desarrolla en él. El juego es más viejo que la cultura, pues esta presupone siempre una sociedad humana; pero los animales no han esperado al hombre para jugar.”

Johan Huizinga, *Homo Ludens* (1938) [27]

Para entender lo que es un juego, en primera instancia, debemos definir el concepto con el fin de concretar a qué nos referimos al hablar de uno. Tras esto, abordaremos el desarrollo de un juego y el diseño de un agente capaz de jugarlo.

En este capítulo se analiza la definición de “juego” desde el punto de vista de la Teoría de Juegos. A partir de esta definición, se propone una clasificación y se presentan varias formas de representación formal. Por último, se ofrece una definición del concepto de “juego” desde la perspectiva del diseño, a pesar de la dificultad que conlleva dicha tarea.

1.1. Teoría de Juegos

Esta sección se basa en los contenidos de *Teoría de Juegos* (Cerdá, Pérez y Jimeno, 2004) [10].

El concepto de juego en el ámbito matemático está estrechamente relacionado con la Teoría de Juegos, disciplina que surge en 1944 con la publicación de *Game Theory and Economic Behavior* [52], de Von Neumann y Morgenstern. Aunque existen trabajos anteriores que tratan ideas similares, no será hasta esta obra cuando se asienten las bases de la Teoría de Juegos moderna, centrada en el análisis y la resolución de conflictos estratégicos.

Aunque la Teoría de Juegos no se desarrolló con el propósito de estudiar la faceta lúdica de los juegos, sino su componente estratégico y sus aplicaciones al análisis de sistemas complejos en economía, biología o sociología, en capítulos posteriores abordaremos precisamente esta dimensión lúdica, desde la perspectiva del diseño de juegos.

1.1.1. Definición

Entendemos “juego” como una situación en la que cada participante busca lograr el mejor resultado posible (maximizar su utilidad), sabiendo que el resultado no depende únicamente de sus propias decisiones, sino también de las decisiones de los demás. Dentro del marco de la Teoría de Juegos, nos centraremos en el análisis de la interacción estratégica entre los jugadores.

De este modo, la Teoría de Juegos puede considerarse como una teoría de la decisión interactiva, diferenciada de la teoría de la decisión individual.

Para este primer acercamiento, es conveniente explicitar algunos conceptos básicos:

- **Jugadores:** Participantes del juego que toman decisiones buscando maximizar su utilidad. Pueden ser dos o más.
- **Acciones:** Conjunto de decisiones disponibles para un jugador en un momento determinado.

1 ¿Qué es un juego?

- **Resultados:** Conjunto de estados en los que se da por finalizado el juego.
- **Pagos:** Ganancias o pérdidas que recibe cada jugador al terminar la partida. Estos dependen del resultado obtenido y nos dan la utilidad que cada jugador va a asociar a ese estado.
- **Estrategias:** Plan completo de acción que sigue un jugador al participar en el juego. Podemos agrupar las estrategias en perfiles de estrategias seguidos por cada jugador.

1.1.2. Tipos de juegos

La primera gran distinción que se puede hacer dentro del ámbito de los juegos es entre **cooperativos y no cooperativos**.

En los juegos **cooperativos**, los jugadores coordinan sus estrategias o establecen acuerdos para maximizar un beneficio conjunto. Por el contrario, en los **no cooperativos**, cada jugador toma decisiones de forma independiente, buscando su propio beneficio.

Este trabajo se centra en los juegos **no cooperativos**, ya que nos enfocamos en aquellos juegos en los que los jugadores están en enfrentamiento directo. Dentro de este ámbito, se distinguen cuatro categorías generales:

- **Juegos estáticos:** Los jugadores deciden simultáneamente, sin conocer las decisiones ajenas.
- **Juegos dinámicos:** Un jugador puede llegar a conocer las decisiones de otro participante antes de tomar su decisión. Usualmente, son juegos que pueden contar con turnos.
- **Juegos con información completa:** Todos los jugadores conocen el estado completo del juego y, por tanto, de las posibles consecuencias de sus acciones.
- **Juegos con información incompleta:** Al menos un jugador no conoce el estado completo del juego, impidiendo así la posibilidad de conocer las consecuencias de alguna acción.

1.1.3. Representación formal

Una vez expuesto qué es un juego, cuáles son sus elementos principales y su clasificación, se plantea a continuación cómo podemos representar uno de forma rigurosa.

Pueden ser representados de manera **estratégica** o **extensiva**. En ambas se explicitan los jugadores, las acciones y los pagos.

La **estratégica** utiliza una matriz que recoge todas las combinaciones posibles de acciones y los pagos asociados, permitiendo visualizar de manera compacta las estrategias disponibles para cada jugador.

La **extensiva**, en cambio, representa el juego mediante un árbol de decisiones que muestra de forma secuencial el desarrollo del mismo.

De manera general, se emplea la representación estratégica para juegos estáticos y la extensiva para los dinámicos.

Primero, se definen los siguientes elementos, que son comunes a ambas formas de representación:

- **El conjunto de jugadores.** J. Podemos escribir $J = \{j_1, \dots, j_n\}$ para un juego con n jugadores. Por simplicidad en la notación, es conveniente que cada $j_i = i$ sea entero. Por tanto, un juego de n jugadores quedaría como $J = \{1, 2, \dots, n\}$.
- **Los conjuntos de acciones.** Cada jugador $i \in J$ tiene un conjunto de acciones A_i posibles ante cada situación.
- **Los conjuntos de estrategias.** Cada jugador $i \in J$ tiene un conjunto de estrategias S_i . Las estrategias definen qué acciones debe tomar ese jugador ante una situación.
- **Los perfiles de estrategias.** Cada perfil $s = (s_1, \dots, s_n)$ es la n -upla que describe la estrategia seguida por cada uno de los n jugadores. El perfil de estrategias nos da los posibles desarrollos del juego.
- **Las funciones de pagos.** Habrá un pago para cada jugador $i \in J$ por jugar cada perfil de estrategias s , $\{u_i(s)\}_{i \in J}$.

Con todo esto presente, describimos el juego G como:

$$G = \left\{ J, \{S_i\}_{i \in J}, \{u_i(s)\}_{i \in J} \right\}$$

Forma estratégica:

Para ver su representación en una matriz bidimensional, suponemos $J = \{1, 2\}$.

Si el jugador 1 tiene estrategias ($S_1 = s_{11}, s_{12}, \dots, s_{1m}$) y el jugador 2 tiene estrategias ($S_2 = s_{21}, s_{22}, \dots, s_{2n}$), la matriz de pagos es:

		J2		
		s_{21}	s_{22}	\dots
J1	s_{11}	$(u_1(s_{11}, s_{21}), u_2(s_{11}, s_{21}))$	$(u_1(s_{11}, s_{22}), u_2(s_{11}, s_{22}))$	\dots
	s_{12}	$(u_1(s_{12}, s_{21}), u_2(s_{12}, s_{21}))$	$(u_1(s_{12}, s_{22}), u_2(s_{12}, s_{22}))$	\dots
	\vdots	\vdots	\vdots	\ddots
	s_{1m}	$(u_1(s_{1m}, s_{21}), u_2(s_{1m}, s_{21}))$	$(u_1(s_{1m}, s_{22}), u_2(s_{1m}, s_{22}))$	\dots

Cada celda corresponde al par de pagos $((u_1, u_2))$ resultante del perfil de estrategias escogido.

En el Capítulo 3 se exponen ejemplos concretos de juegos representados de forma estratégica como el dilema del prisionero 3.1.1.1 o piedra papel y tijeras 3.1.1.2.

■ **Forma extensiva:** En la forma extensiva nos apoyamos en un árbol donde los jugadores toman decisiones y van ejecutando acciones sucesivamente.

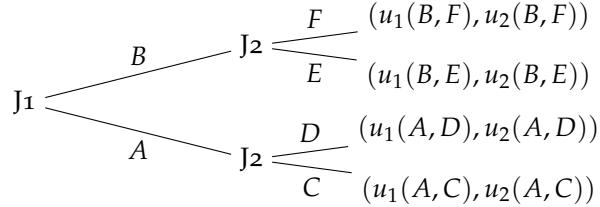
Veamos cómo sería un juego $G = \{\{1, 2\}, \{S_i\}_{i \in J}, \{u_i(s)\}_{i \in J}\}$ donde las acciones posibles son $A_1 = \{A, B\}$ para el J1 y $A_2 = \{C, D, F, G\}$ para J2. Como no podemos representar todos los perfiles de estrategias, a continuación representamos uno donde:

- J1 elige entre A y B.

1 ¿Qué es un juego?

- Si elige A , entonces J2 elige entre C y D .
- Si elige B , entonces J2 elige entre E y F .

Al final de cada rama hay un vector de pagos (u_1, u_2) .



De esta forma, podemos representar las acciones de forma secuencial y los pagos a los que llega cada jugador.

En el Capítulo 3 se exponen ejemplos concretos de juegos representados de forma extensiva como el tres en raya 3.2.1.1 o el ajedrez 3.2.1.2.

1.2. Pero, ¿qué es realmente un juego?

Definir qué es un juego puede parecer trivial a primera vista, pero se trata de una tarea sorprendentemente compleja y profundamente relevante para cualquier diseñador. La necesidad de una definición precisa no es meramente académica: no podemos aspirar a mejorar en la creación de juegos si no somos capaces de articular qué es lo que estamos diseñando. En este sentido, disponer de un vocabulario común y bien delimitado permite no solo comunicar ideas, sino también evaluar y criticar mecánicas, estructuras y experiencias lúdicas de manera rigurosa. [13]

A lo largo de la historia, distintos autores han intentado capturar la esencia del juego desde perspectivas diversas:

1.2.1. Juegos como actividades culturales y sociales

Joh Huizinga, en su influyente obra *Homo Ludens* (1938) [27], propone una definición amplia y abarcativa:

“El juego, en su aspecto formal, es una acción libre ejecutada «como sí» y sentida como situada fuera de la vida corriente, pero que, a pesar de todo, puede absorber por completo al jugador, sin que haya en ella ningún interés material ni se obtenga en ella provecho alguno, que se ejecuta dentro de un determinado tiempo y un determinado espacio, que se desarrolla en un orden sometido a reglas y que da origen a asociaciones que propenden a rodearse de misterio o a disfrazar para destacarse del mundo habitual.”

Según Huizinga, el juego es:

- Libre y voluntario.
- Desarrollado dentro de límites de tiempo y espacio.
- Regido por reglas.

- Sin finalidad práctica, es decir, un fin en sí mismo.
- Acompañado de tensión, alegría y separación de la vida ordinaria.

Si bien esta definición es profunda y culturalmente rica, resulta demasiado amplia para el contexto del diseño de videojuegos, ya que engloba prácticas como el teatro, rituales y otras formas de juego simbólico que no necesariamente comparten las propiedades interactivas y mecánicas de los videojuegos modernos.

1.2.2. Juegos como superación de obstáculos

Bernard Suits, en *The Grasshopper: Games, Life and Utopia* (1978) [45], propone un enfoque más formal y filosófico:

“Playing a game is the voluntary attempt to overcome unnecessary obstacles.”

Suits desarrolla esta definición mediante cuatro componentes fundamentales:

1. Una actividad en la que los participantes intentan alcanzar un estado meta (*prelusory goal*).
2. Usando solo los medios permitidos por las reglas (*lusory means*).
3. Donde las reglas prohíben usar medios más eficaces para alcanzar la meta.
4. Y los jugadores aceptan estas reglas precisamente porque hacen posible la actividad (*lusory attitude*).

Aunque la definición de Suits se ajusta bien a juegos de mesa o deportivos, se queda corta para capturar las dinámicas interactivas y experienciales de los videojuegos modernos, donde elementos como la narrativa, la exploración o la elección del jugador son fundamentales y no siempre encajan en la idea de “obstáculos innecesarios”.

Esta definición resultó especialmente útil para el filósofo C. Thi Nguyen en su obra *Games: Agency As Art (Thinking Art)*, donde reflexiona sobre los juegos como un medio artístico que explora la agencia humana. Nguyen reconoce desde el inicio la dificultad de proporcionar una definición satisfactoria de este fenómeno. Apoyándose en la concepción de Suits, introduce el término *suitsian game* para referirse a aquellos juegos que poseen reglas formales y objetivos definidos, diferenciándolos del juego libre o abierto a la improvisación.

1.2.3. La necesidad de un vocabulario para diseñar

El artículo *I Have No Words and I Must Design* [13] de Greg Costikyan aborda de manera explícita la dificultad de definir juegos y la importancia de contar con un vocabulario compartido en el diseño. Costikyan propone una definición más centrada en los elementos relevantes para el diseño:

“[A game is] an interactive structure of endogenous meaning that requires players to struggle toward a goal.”

Se identifica varias propiedades que debe tener un juego:

- Interacción significativa entre el jugador y el sistema.

1 ¿Qué es un juego?

- Existencia de objetivos claros.
- Presencia de conflictos o desafíos.
- Estructura y reglas que delimitan la experiencia.
- Significado endógeno: los jugadores entienden y valoran la experiencia desde las propias reglas.

Este enfoque permite entender mejor por qué algunas actividades, aunque entretenidas, no califican como juegos según criterios de diseño: deben ofrecer desafíos interactivos y metas que generen *engagement* mediante reglas que los jugadores comprendan y acepten.

1.2.4. Definición operativa

Para los fines del diseño de videojuegos, la definición más práctica y operativa que hemos encontrado es la de Jesse Schell en *The Art of Game Design: A Book of Lenses* [40]:

“A game is a problem-solving activity, approached with a playful attitude.”

Schell propone, a partir de un análisis comparativo de múltiples definiciones, diez cualidades esenciales que caracterizan a un juego:

1. Los juegos son iniciados voluntariamente.
2. Los juegos poseen objetivos claros.
3. Los juegos presentan conflictos.
4. Los juegos están regidos por reglas.
5. Los juegos pueden ser ganados o perdidos.
6. Los juegos son interactivos.
7. Los juegos implican desafíos.
8. Los juegos generan valor interno propio.
9. Los juegos logran involucrar a los jugadores (*engagement*).
10. Los juegos constituyen sistemas formales y cerrados.

Schell reconoce que incluso esta lista extensa puede no ser exhaustiva, y que su propósito principal es ofrecer un marco práctico para evaluar y diseñar juegos. Según el propio autor, si se necesitan muchos elementos para definir algo, probablemente convenga reagrupar y simplificar las ideas para capturar la esencia de manera más efectiva. Esta aproximación proporciona una definición operativa robusta y aplicable al diseño de videojuegos.

En resumen, la literatura ha explorado diversas concepciones de juego:

- Huizinga ofrece una visión cultural y social amplia, pero demasiado inclusiva para nuestro propósito.

1.2 Pero, ¿qué es realmente un juego?

- Suits formaliza la estructura de los juegos clásicos, pero no cubre del todo las particularidades de los videojuegos.
- Costikyan y Schell centran la definición en la interacción, el objetivo y la experiencia del jugador, proporcionando un marco útil para el diseño.

A partir de estas reflexiones, consideramos que un juego, desde la perspectiva del diseño de videojuegos, es una actividad que presenta un desafío a ser resuelto por el jugador, el cual se aborda de forma voluntaria y con intención creativa. La definición de Schell, con sus puntos de evaluación, será nuestro punto de referencia para el análisis posterior.

Capítulo 2: Utilidad, soluciones y equilibrio

En el capítulo anterior se ha definido formalmente qué se entiende por un juego y sus componentes principales: el conjunto de jugadores J , los conjuntos de estrategias $\{S_i\}_{i \in J}$, y las funciones de pagos $\{u_i(s)\}_{i \in J}$. A partir de esta base, se introducen los conceptos de *utilidad*, *concepto de solución* y *equilibrio*, que constituyen los pilares analíticos de la Teoría de Juegos.

A partir de este punto se dejan de lado los aspectos de diseño para centrarse en la formalización teórica, cuya aplicación práctica se retomará posteriormente durante el desarrollo del *software* (véase el capítulo 7).

Los conceptos principales de este capítulo se basan en lo expuesto en *Teoría de Juegos* (Cerdá, Pérez y Jimeno, 2004) [10].

2.1. Utilidad

En la Teoría de Juegos, la **utilidad** es la medida numérica que representa las preferencias o el grado de satisfacción de cada jugador respecto a los posibles resultados del juego. Dado un conjunto de resultados X , cada jugador $i \in J$ posee una **función de utilidad**

$$u_i : X \rightarrow \mathbb{R},$$

que asigna un valor real a cada resultado $x \in X$, de manera que

$$u_i(x') > u_i(x'') \quad \text{si y sólo si el jugador } i \text{ prefiere } x' \text{ a } x''.$$

Esta función de utilidad puede interpretarse como una representación *ordinal* de las preferencias del jugador: sólo importa el orden, no la magnitud de los valores asignados. Cualquier transformación estrictamente creciente de u_i representará las mismas preferencias [10].

En el contexto de un juego, los resultados dependen del perfil de estrategias s donde:

$$s = (s_1, \dots, s_n) \in S = S_1 \times \dots \times S_n.$$

Así, la utilidad de un jugador se expresa como:

$$u_i(s) = u_i(s_1, s_2, \dots, s_n),$$

que indica lo que obtiene el jugador i cuando todos los jugadores eligen las estrategias especificadas por el perfil s .

2.1.1. Utilidad esperada

Cuando existe incertidumbre sobre las acciones de los demás jugadores o sobre los resultados, se trabaja con **utilidades esperadas**. Si un jugador i asocia una probabilidad $p(s)$ a cada

perfil de estrategias s , la utilidad esperada se define como:

$$\mathbb{E}[u_i] = \sum_{s \in S} p(s) u_i(s).$$

Esta formulación supone que los jugadores cumplen los axiomas de racionalidad, continuidad e independencia, y permite representar sus preferencias sobre distribuciones probabilísticas de resultados mediante una función lineal en las probabilidades. En este caso, las utilidades $u_i(s)$ se denominan utilidades de Von Neumann-Morgenstern, únicas salvo una transformación afín positiva.

2.2. Concepto de solución

El objetivo de la Teoría de Juegos es identificar qué resultados pueden considerarse razonables o estables bajo el supuesto de que los jugadores actúan racionalmente. Para ello, se introducen los **conceptos de solución**, entendidos como criterios formales que permiten identificar los perfiles de estrategias más plausibles, dados los supuestos del juego [10, 36].

El concepto de solución asocia a cada juego

$$G = \left\{ J, \{S_i\}_{i \in J}, \{u_i(s)\}_{i \in J} \right\}$$

un subconjunto $S^* \subseteq S$ de perfiles de estrategias considerados razonables o estables.

Se entiende por comportamiento racional aquel orientado a maximizar la utilidad. Se asume, por tanto, que los jugadores actúan de manera racional e independiente. Aunque este supuesto no siempre se cumple en contextos reales, resulta una abstracción necesaria para poder aplicar plenamente el marco analítico de la teoría.

2.2.1. Principales conceptos de solución

Entre los conceptos de solución más relevantes se encuentran los siguientes:

- **Estrategias dominantes y eliminación iterada.** Se eliminan las estrategias que resultan peores que otras para el mismo jugador, independientemente de lo que hagan los demás.
- **Equilibrio de Nash.** Se alcanzan perfiles de estrategias en los que ningún jugador puede mejorar su utilidad desviándose unilateralmente.
- **Óptimo de Pareto.** Describe situaciones donde no se puede mejorar la utilidad de un jugador sin empeorar la de otro [36].
- **Equilibrio bayesiano.** Extiende el equilibrio de Nash a juegos con información incompleta, en los que los jugadores tienen creencias probabilísticas sobre los tipos o características de los demás.

2.3. Dominación y racionalización

Antes de buscar equilibrios, puede simplificarse el conjunto de estrategias eliminando aquellas que nunca serían elegidas racionalmente. Este proceso se conoce como **eliminación**

iterada de estrategias dominadas.

2.3.1. Estrategias dominadas

Sea S_i el conjunto de estrategias de un jugador i . Se dice que una estrategia s'_i está **estrictamente dominada** por otra s''_i si:

$$u_i(s_1, \dots, s'_i, \dots, s_n) < u_i(s_1, \dots, s''_i, \dots, s_n) \quad \forall s_j \mid j \neq i.$$

En ese caso, un jugador racional nunca elegiría s'_i .

Si la desigualdad es no estricta para todos los s_j con $j \neq i$ y estricta para al menos uno, se dice que s'_i está **débilmente dominada** por s''_i :

$$u_i(s_1, \dots, s'_i, \dots, s_n) \leq u_i(s_1, \dots, s''_i, \dots, s_n) \quad \forall s_j \mid j \neq i, \text{ con } > \text{ para algún } s_j.$$

2.3.2. Eliminación iterada

La **eliminación iterada de estrategias estrictamente dominadas** consiste en:

1. Eliminar de cada jugador las estrategias estrictamente dominadas.
2. Repetir el proceso en el juego reducido.

Si este procedimiento conduce a un único perfil de estrategias, se considera una **solución racionizable**. En caso contrario, el conjunto restante representa las estrategias que podrían jugarse razonablemente sin contradecir la lógica de optimización individual.

2.4. Equilibrio de Nash

El **equilibrio de Nash** constituye el concepto de solución fundamental en el análisis de juegos no cooperativos. Un perfil de estrategias $s^* = (s_1^*, \dots, s_n^*)$ constituye un equilibrio si ningún jugador puede mejorar su utilidad modificando unilateralmente su estrategia:

$$u_i(s_1^*, \dots, s_i^*, \dots, s_n^*) \geq u_i(s_1^*, \dots, s_i, \dots, s_n^*) \quad \forall s_i \in S_i, \forall i \in J.$$

Vamos a introducir algunas definiciones y resultados previos que serán necesarios para enunciar y demostrar los teoremas relativos al equilibrio de Nash.

Diremos que una función $f : S \rightarrow \mathbb{R}$ es **cuasicóncava** si, para todo par de puntos $x, y \in S$ y para todo $\lambda \in [0, 1]$, se cumple que

$$f(\lambda x + (1 - \lambda)y) \geq \min\{f(x), f(y)\}.$$

La cuasiconcavidad es una propiedad más débil que la concavidad, pero suficiente para asegurar que los conjuntos de mejores respuestas (es decir, los conjuntos de estrategias que maximizan la función de utilidad) sean **convexos**. Esta convexidad es un elemento clave en la demostración del teorema de existencia de equilibrio de Nash, ya que permite aplicar resultados de punto fijo (como el teorema de Kakutani).

De forma complementaria, introducimos el concepto de **correspondencia**. Diremos que una correspondencia de un conjunto X en un conjunto Y es una aplicación multivaluada $F : X \rightrightarrows Y$ que asigna a cada elemento $x \in X$ un subconjunto $F(x) \subseteq Y$. A diferencia de una función ordinaria, una correspondencia puede asociar varios valores posibles a un mismo elemento de su dominio.

Por otro lado, diremos que una correspondencia $F : X \rightrightarrows Y$ es **hemicontinua superiormente** si para toda sucesión $\{x^k\} \subset X$ que converge a x , y para toda sucesión $\{y^k\} \subset Y$ con $y^k \in F(x^k)$ que converge a y , se cumple que $y \in F(x)$. Intuitivamente, esta propiedad garantiza que los valores de la correspondencia no varían bruscamente ante pequeñas modificaciones de su argumento, asegurando la estabilidad de sus imágenes en el límite.

Estas nociones resultan fundamentales para aplicar el siguiente resultado, que constituye la base de la demostración del teorema de existencia de equilibrio de Nash.

Teorema 2.1 (Teorema del punto fijo de Kakutani, [10], Teorema 3.2). *Sea X un subconjunto no vacío, compacto y convexo de \mathbb{R}^n , y sea $F : X \rightrightarrows X$ una correspondencia tal que:*

- $F(x)$ es no vacío, convexo y compacto para todo $x \in X$;
- el grafo de F es cerrado, o equivalentemente, F es hemicontinua superiormente.

Entonces, existe al menos un punto $x^* \in X$ tal que $x^* \in F(x^*)$.

Este teorema garantiza la existencia de un punto fijo para correspondencias que cumplen condiciones de continuidad y convexidad, y será la herramienta central para demostrar la existencia del equilibrio de Nash en juegos continuos.

Denotaremos por G un juego definido como:

$$G = \left\{ J, \{S_i\}_{i \in J}, \{u_i(s)\}_{i \in J} \right\}.$$

A partir de los conceptos introducidos, podemos enunciar y demostrar el siguiente resultado sobre la existencia de equilibrio de Nash.

Teorema 2.2 (Existencia de equilibrio de Nash en juegos continuos [10], Teorema 3.3). *Sea G un juego en el que para cada jugador i se cumple que S_i es no vacío, compacto y convexo, y que u_i es continua en S y cuasicóncava en su propia estrategia s_i . Entonces, existe al menos un equilibrio de Nash en estrategias puras.*

Demostración. Sea G un juego que verifica las hipótesis del teorema. Para cada jugador $i \in J$, el conjunto de estrategias S_i es no vacío, compacto y convexo, y la función de utilidad $u_i : S \rightarrow \mathbb{R}$ es continua en $S = S_1 \times \dots \times S_n$ y cuasicóncava en su propia estrategia s_i .

Para cada jugador i definimos su correspondencia de respuesta óptima R_i , que asigna a cada perfil de estrategias $s = (s_1, \dots, s_i, \dots, s_n)$ el conjunto de estrategias de i que son respuesta óptima a dicho perfil:

$$R_i(s) = \{x_i \in S_i \mid u_i(s_1, \dots, x_i, \dots, s_n) \geq u_i(s_1, \dots, y_i, \dots, s_n), \forall y_i \in S_i\}.$$

Definimos la correspondencia global de respuesta óptima R como:

$$R(s) = R_1(s) \times \dots \times R_n(s) = \{t = (t_1, \dots, t_n) \in S \mid t_i \in R_i(s), \forall i \in J\}.$$

Esta correspondencia asigna a cada perfil de estrategias s el producto cartesiano de los conjuntos de respuesta óptima individuales.

2 Utilidad, soluciones y equilibrio

A continuación, comprobaremos que R cumple las condiciones del *teorema del punto fijo de Kakutani*, lo que garantizará la existencia de un punto fijo $s^* \in S$ tal que $s^* \in R(s^*)$.

1. **El conjunto S es no vacío, compacto y convexo.** Por hipótesis, cada S_i cumple estas propiedades; por tanto, su producto cartesiano $S = S_1 \times \dots \times S_n$ también lo es.
2. **$R(s)$ es no vacío para todo $s \in S$.** Dado que $u_i(\cdot)$ es continua en el conjunto compacto S_i , el teorema de Weierstrass garantiza que el máximo se alcanza. Por tanto, $R_i(s) \neq \emptyset$ para cada jugador i , y en consecuencia $R(s) \neq \emptyset$.
3. **Cada $R(s)$ es convexo.** La cuasicóncavidad de u_i respecto a su propia estrategia s_i implica que el conjunto de maximizadores $R_i(s)$ es convexo. Por tanto, el producto cartesiano $R(s) = R_1(s) \times \dots \times R_n(s)$ es también convexo.
4. **La correspondencia R es hemicontinua superiormente.** Sea una sucesión $\{s^k\} \subset S$ tal que $s^k \rightarrow s$, y una sucesión $\{t^k\}$ con $t^k \in R(s^k)$ y $t^k \rightarrow t$. Para cada jugador i , se cumple:

$$u_i(s_1^k, \dots, t_i^k, \dots, s_n^k) \geq u_i(s_1^k, \dots, y_i, \dots, s_n^k), \quad \forall y_i \in S_i.$$

Al tomar el límite cuando $k \rightarrow \infty$ y usar la continuidad de u_i , se obtiene:

$$u_i(s_1, \dots, t_i, \dots, s_n) \geq u_i(s_1, \dots, y_i, \dots, s_n), \quad \forall y_i \in S_i,$$

de modo que $t_i \in R_i(s)$ para todo i . Por tanto, $t \in R(s)$ y el grafo de R es cerrado, es decir, R es hemicontinua superiormente.

Dado que R es una correspondencia definida sobre un conjunto no vacío, compacto y convexo, con valores no vacíos, convexos y un grafo cerrado, se cumplen las condiciones del *teorema del punto fijo de Kakutani*. En consecuencia, existe $s^* \in S$ tal que $s^* \in R(s^*)$.

Podemos afirmar entonces que R tiene al menos un punto fijo $s^* = (s_1^*, \dots, s_n^*)$. Por tanto, $s_i^* \in R_i(s^*) \forall i \in J$, es decir, s_i^* es una respuesta óptima frente a s^* para cada jugador. En otras palabras, s^* constituye un **equilibrio de Nash en estrategias puras** para el juego G .

□

Este resultado garantiza la existencia de equilibrio en juegos con espacios de estrategias continuos, siempre que las preferencias sean bien comportadas.

Podemos aplicar este Teorema para obtener un nuevo resultado sobre juegos con estrategias mixtas.

Teorema 2.3 (Existencia de equilibrio de Nash en juegos finitos [10], Teorema 3.4). *En todo juego finito G existe al menos un equilibrio de Nash en estrategias mixtas.*

Demostración. Dado el juego finito

$$G = \{J, \{S_i\}_{i \in J}, \{u_i\}_{i \in J}\},$$

con $S_i = \{s_i^1, \dots, s_i^{m_i}\}$ finito para cada i , consideremos el juego $B(G)$, que tiene los mismos jugadores que G y definimos como:

Para cada jugador i , sea $B(S_i)$ el conjunto de distribuciones de probabilidad sobre S_i (estrategias mixtas de i). El espacio de perfiles mixtos es $B(S) = B(S_1) \times \dots \times B(S_n)$. Para $p = (p_1, \dots, p_n) \in B(S)$, el pago esperado de i viene dado por

$$U_i(p_1, \dots, p_n) = \sum_{s \in S} \left(\prod_{j \in J} p_j(s_j) \right) u_i(s),$$

donde $s = (s_1, \dots, s_n)$.

Comprobamos ahora las hipótesis del Teorema 2.3 continuo (Teorema 3.3):

- (a) Para cada i , $B(S_i)$ es un subconjunto no vacío, compacto y convexo de \mathbb{R}^{m_i} .
- (b) U_i es continua en todo $B(S)$ y, fijados p_j para $j \neq i$, la aplicación $p_i \mapsto U_i(p_1, \dots, p_i, \dots, p_n)$ es *afín* en p_i ; por tanto, es cuasicónica en p_i .

En conclusión, por el teorema anterior, existe un equilibrio de Nash en estrategias puras para $B(G)$. Por tanto, existe un equilibrio de Nash en estrategias mixtas de G .

□

Por último, vamos a enunciar, sin demostrar, un resultado importante acerca de los juegos simétricos.

Teorema 2.4 (Equilibrio de Nash simétrico [10], Teorema 3.5). *Si el juego G es simétrico, donde $S_i = A$ para todo i , y si A es compacto y convexo y las funciones u_i son continuas y cuasicónicas en su propia estrategia, entonces existe al menos un equilibrio de Nash simétrico. Si el juego es finito, existe al menos un equilibrio de Nash simétrico en estrategias mixtas.*

Los juegos simétricos son frecuentes en la economía y la biología, ya que modelan situaciones donde todos los jugadores enfrentan el mismo conjunto de estrategias y pagos estructuralmente equivalentes.

2.4.1. Juegos de suma cero

Antes de presentar un resultado clásico que conecta el equilibrio de Nash con las estrategias *maximín*, conviene introducir formalmente el concepto de **juego de suma cero**.

Un juego bipersonal es de suma cero si las utilidades de ambos jugadores verifican que:

$$u_1(s_1, s_2) + u_2(s_1, s_2) = 0 \quad \forall (s_1, s_2) \in S_1 \times S_2.$$

Es decir, cualquier ganancia de un jugador implica una pérdida equivalente del otro. En este tipo de juegos el conflicto es total y los intereses de los jugadores son estrictamente opuestos. Por tanto, basta con analizar la función de pago de un solo jugador, ya que la del otro se deduce automáticamente como su negación.

En los juegos de suma cero, cada jugador trata de optimizar su resultado bajo la hipótesis de que el adversario juega del modo más desfavorable posible. Así, el jugador 1 buscará maximizar su ganancia mínima esperada:

$$v_1 = \max_{p_1 \in \Delta(S_1)} \min_{p_2 \in \Delta(S_2)} U_1(p_1, p_2),$$

2 Utilidad, soluciones y equilibrio

mientras que el jugador 2 intentará minimizar la ganancia máxima de su oponente:

$$v_2 = \min_{p_2 \in \Delta(S_2)} \max_{p_1 \in \Delta(S_1)} U_1(p_1, p_2).$$

Las estrategias que alcanzan esos valores se denominan, respectivamente, *maximín* y *minimax*. En general, $v_1 \leq v_2$, y si ambos valores coinciden, el juego se dice que tiene un **valor** $v = v_1 = v_2$.

En este contexto cobra especial relevancia el teorema del *minimax*, resultado fundamental debido a John von Neumann, que garantiza precisamente la existencia de ese valor en todo juego finito de suma cero. El siguiente resultado presenta su formulación más general.

Teorema 2.5 (Teorema del minimax [10], Teorema 3.6). *Dado un juego bipersonal finito de suma cero*

$$G = \{\{1, 2\}, S_1, S_2; u_1, u_2\},$$

dicho juego tiene un **valor**. Es decir, existe un $v \in \mathbb{R}$ tal que $v_1 = v_2 = v$, siendo v_1 y v_2 los valores maximín y minimax.

El teorema anterior establece la existencia de una situación de equilibrio en términos de expectativas: el jugador 1 puede garantizarse una ganancia no inferior a v , mientras que el jugador 2 puede limitar la ganancia de su adversario a lo sumo a v . El número v se denomina **valor del juego** y constituye la solución natural de los juegos de conflicto puro.

A partir de esta idea se puede establecer una conexión directa entre el equilibrio de Nash y las estrategias *minimax*. El siguiente resultado demuestra que, en los juegos de suma cero, ambos conceptos son equivalentes: las estrategias que satisfacen la condición del teorema del *minimax* son exactamente las que forman parte de los equilibrios de Nash.

Teorema 2.6 (Equilibrio y estrategias *minimax* [10], Teorema 3.7). *En los juegos bipersonales finitos de suma cero, las estrategias minimax forman parte de los equilibrios de Nash, y únicamente ellas.*

Demostración. Sea el juego bipersonal finito de suma cero

$$G = \{\{1, 2\}, S_1, S_2; u_1, u_2\},$$

donde

$$u_1(s_1, s_2) + u_2(s_1, s_2) = 0 \quad \forall (s_1, s_2) \in S_1 \times S_2.$$

Denotemos por A_1 la matriz de pagos del jugador 1, es decir,

$$A_1 = (u_1(s_1, s_2))_{s_1 \in S_1, s_2 \in S_2}, \quad A_2 = -A_1.$$

Sean p_1 y p_2 estrategias mixtas genéricas de 1 y 2, es decir, distribuciones de probabilidad sobre sus respectivos conjuntos de estrategias puras.

(a) Si (p_1^*, p_2^*) es un equilibrio de Nash, entonces p_1^* y p_2^* son estrategias *maximín* y *minimax*, y el pago corresponde al valor del juego.

Por definición de equilibrio de Nash,

$$U_1(p_1^*, p_2^*) = p_1^* A_1 p_2^{*t} = \max_{p_1 \in \Delta(S_1)} p_1 A_1 p_2^{*t} = \min_{p_2 \in \Delta(S_2)} p_1^* A_1 p_2^t.$$

Denotando por v_1 y v_2 los valores *maximín* y *minimax* definidos como

$$v_1 = \max_{p_1} \min_{p_2} p_1 A_1 p_2^t, \quad v_2 = \min_{p_2} \max_{p_1} p_1 A_1 p_2^t,$$

se deduce que $v_1 = v_2 = v$, por el teorema del minimax anterior. Por tanto,

$$U_1(p_1^*, p_2^*) = v, \quad U_2(p_1^*, p_2^*) = -v,$$

y cada jugador obtiene el valor del juego.

(b) Recíprocamente, si (p_1^*, p_2^*) son estrategias *maximín* y *minimax*, entonces forman un equilibrio de Nash.

Si p_1^* y p_2^* alcanzan los extremos del teorema del minimax, se cumple para todo par (p_1, p_2) :

$$U_1(p_1, p_2^*) \leq v \leq U_1(p_1^*, p_2),$$

lo que implica que p_1^* es una mejor respuesta a p_2^* y viceversa. Por definición, (p_1^*, p_2^*) constituye entonces un equilibrio de Nash.

En resumen, en los juegos bipersonales finitos de suma cero, las estrategias *maximín* y *minimax* coinciden con las estrategias de equilibrio de Nash, y el valor del juego es único. \square

Los resultados anteriores son relevantes para el desarrollo de agentes en juego bipersonales. En capítulos posteriores (véase la Sección 4.2.1) se presentará su implementación práctica, donde el algoritmo *minimax* se aplica a la toma de decisiones del agente mediante exploración de árboles de juego y poda alfa–beta, buscando reproducir un comportamiento racional coherente con el valor teórico del juego.

2.5. Eficiencia de Pareto

Un perfil de estrategias $s \in S$ es **óptimo de Pareto** si no existe otro perfil s' tal que:

$$u_i(s') \geq u_i(s) \quad \forall i \in J, \quad \text{y} \quad u_j(s') > u_j(s) \text{ para algún } j \in J.$$

Es decir, no puede mejorarse la situación de un jugador sin perjudicar a otro.

En general, los equilibrios de Nash no son necesariamente óptimos de Pareto, como ilustra el clásico dilema del prisionero (ver sección 3.1.1.1). La diferencia entre ambos conceptos radica en que el equilibrio de Nash se centra en la estabilidad individual, mientras que la eficiencia de Pareto mide la eficiencia social.

2.6. Equilibrio bayesiano

En los juegos con información incompleta, los jugadores desconocen ciertos elementos del entorno o de los demás. Cada jugador i tiene un **tipo** $t_i \in T_i$, y una estrategia que depende de él:

$$s_i : T_i \rightarrow S_i.$$

2 Utilidad, soluciones y equilibrio

Sea $p(t)$ la distribución común de probabilidad sobre los tipos $t = (t_1, \dots, t_n)$. La utilidad esperada de un jugador i , dado su tipo, es:

$$\mathbb{E}_{t_{-i}}[u_i(s_i(t_i), s_{-i}(t_{-i}), t_i, t_{-i})].$$

Un perfil (s_1^*, \dots, s_n^*) constituye un **equilibrio bayesiano** si, para todo jugador i y tipo t_i ,

$$\mathbb{E}_{t_{-i}}[u_i(s_i^*(t_i), s_{-i}^*(t_{-i}), t_i, t_{-i})] \geq \mathbb{E}_{t_{-i}}[u_i(s_i(t_i), s_{-i}^*(t_{-i}), t_i, t_{-i})],$$

para toda estrategia alternativa s_i .

Este equilibrio generaliza el de Nash a entornos de información asimétrica. Su existencia se apoya, conceptualmente, en el mismo principio que el **Teorema de existencia de Nash en estrategias mixtas** (Teorema 2.3), ya que los jugadores, al desconocer los tipos ajenos, deben formar distribuciones de probabilidad sobre ellos. Así, el equilibrio bayesiano puede de entenderse como una extensión del equilibrio mixto de Nash al caso de incertidumbre estructurada mediante creencias probabilísticas.

Este concepto resulta fundamental en contextos como subastas, selección adversa o negociación.

Capítulo 3: Ejemplos de juegos

Una vez establecidos los conceptos fundamentales, resulta pertinente presentar diversos ejemplos de juegos. Con el propósito de ilustrar la clasificación expuesta en el primer capítulo, se introduce un ejemplo representativo para cada categoría considerada.

3.1. Juegos estáticos

En esta sección se presentan varios juegos que ejemplifican la categoría de juegos estáticos. Se exponen dos casos clásicos y ampliamente utilizados en la literatura académica para ilustrar conceptos fundamentales: el dilema del prisionero y el mercado del limón. Para complementar, se muestran también algunos juegos reales, más sencillos y conocidos, que no sólo pueden modelarse dentro de este marco teórico, sino que además son jugados con fines recreativos.

3.1.1. Juegos de información completa

3.1.1.1. Dilema del prisionero

El dilema del prisionero es un ejemplo clásico y sencillo de juego de información completa.

El siguiente ejemplo está tomado textualmente de [10], (ejemplo 2.1).

“ Dos delincuentes habituales son apresados cuando acaban de cometer un delito grave. No hay prueba clara contra ellos, pero sí indicios fuertes de dicho delito y además hay pruebas de un delito menor. Son interrogados simultáneamente en habitaciones separadas. Ambos saben que si los dos se callan serán absueltos del delito principal por falta de pruebas, pero condenados por el delito menor (1 año de cárcel), que si ambos confiesan, serán condenados por el principal pero se les rebajará un poco la pena por confesar (4 años), y finalmente, que si sólo uno confiesa, él se librará de penas y al otro «se le caerá el pelo» (5 años).”

Es claro que se trata de un juego estático, ya que ambos jugadores eligen sus estrategias de manera simultánea, sin conocer la decisión del otro. Además, es un juego de información completa, puesto que ambos jugadores conocen las estrategias disponibles y los pagos asociados a cada perfil de estrategias.

Para representarlo de forma estratégica, primero definimos el conjunto de jugadores:

$$J = \{1, 2\}$$

Cada jugador tiene dos estrategias disponibles:

$$S_1 = S_2 = \{C, E\},$$

donde C significa “callarse” y E significa “entregarse”. Denotamos por $s = (s_1, s_2)$ un perfil de estrategias. Tomamos como función de pagos la negación de los años de prisión (utilidad negativa), de modo que mayores años representan peores pagos:

	C_2	E_2
C_1	(-1, -1)	(-5, 0)
E_1	(0, -5)	(-4, -4)

La entrada (u_1, u_2) indica las utilidades de los jugadores 1 y 2 respectivamente. Por ejemplo, si ambos eligen C reciben cada uno -1 (un año de prisión). Si ambos confiesan (E, E) reciben -4 (cuatro años). Si uno confiesa y el otro no, el confesor recibe 0 y el otro -5.

Para cada jugador i comparar las utilidades entre C y E muestra que E es estrategia estrictamente dominante:

- Si el otro juega C : $u_i(E, C) = 0 > u_i(C, C) = -1$.
- Si el otro juega E : $u_i(E, E) = -4 > u_i(C, E) = -5$.

Por tanto el único equilibrio de Nash en estrategias puras es (E, E) .

El perfil (C, C) es Pareto-óptimo frente a (E, E) porque $-1 > -4$ para ambos. Sin embargo, la estructura de incentivos lleva a ambos jugadores hacia (E, E) pese a que (C, C) sería mejor colectivamente. Esto ejemplifica el conflicto entre interés individual y bienestar social que caracteriza al dilema del prisionero.

3.1.1.2. Piedra papel o tijeras

El juego de piedra, papel o tijeras es otro ejemplo de juego estático y de información completa, ampliamente conocido por su sencillez y simetría.

En este juego, ambos jugadores eligen simultáneamente una de las tres posibles acciones: piedra, papel o tijeras. Las reglas son bien conocidas: la piedra gana a la tijera, la tijera gana al papel y el papel gana a la piedra. Si ambos jugadores eligen la misma opción, el resultado es un empate.

Formalmente, definimos el conjunto de jugadores:

$$J = \{1, 2\}$$

Y los conjuntos de estrategias

$$S_1 = S_2 = \{R, P, S\},$$

donde R representa piedra (*rock*), P papel (*paper*) y S tijeras (*scissors*). Cada jugador recibe un pago de 1 si gana, -1 si pierde y 0 si empata. La matriz de pagos puede representarse del siguiente modo:

	R_2	P_2	S_2
R_1	(0, 0)	(-1, 1)	(1, -1)
P_1	(1, -1)	(0, 0)	(-1, 1)
S_1	(-1, 1)	(1, -1)	(0, 0)

Cada entrada (u_1, u_2) indica las utilidades obtenidas por los jugadores 1 y 2 respectivamente. Por ejemplo, si el jugador 1 elige piedra y el jugador 2 tijeras, el primero gana y obtiene 1, mientras que el segundo pierde y obtiene -1.

Cada entrada (u_1, u_2) indica las utilidades obtenidas por los jugadores 1 y 2, respectivamente. Por ejemplo, si el jugador 1 elige piedra y el jugador 2 tijeras, el primero gana y obtiene 1, mientras que el segundo pierde y obtiene -1.

Observamos que este juego no posee un equilibrio de Nash en estrategias puras, ya que para cualquier combinación determinista alguno de los jugadores puede mejorar su resultado cambiando unilateralmente su elección. Formalmente, para cualquier par $(s_1, s_2) \in S_1 \times S_2$, existe una desviación $s'_1 \in S_1$ tal que:

$$u_1(s'_1, s_2) > u_1(s_1, s_2),$$

y de modo análogo para el jugador 2.

Sin embargo, existe un equilibrio en estrategias mixtas, en el que cada jugador elige piedra, papel o tijeras con igual probabilidad $\frac{1}{3}$.

En este caso, en lugar de trabajar con estrategias puras, consideraremos funciones de distribución de probabilidad sobre $S_i = \{R, P, S\}$:

$$\sigma_i(s) = \begin{cases} p_r & \text{si } s = R, \\ p_p & \text{si } s = P, \\ p_s & \text{si } s = S. \end{cases}$$

Las funciones de utilidad esperada vienen dadas por:

$$\mathbb{E}[u_i(\sigma_1, \sigma_2)]$$

donde la esperanza se calcula sobre las distribuciones de probabilidad de ambos jugadores.

Podemos tomar la distribución que asigna la misma probabilidad a cada estrategia:

$$\sigma_i^*(s) = \begin{cases} \frac{1}{3} & \text{si } s = R, \\ \frac{1}{3} & \text{si } s = P, \\ \frac{1}{3} & \text{si } s = S. \end{cases}$$

La utilidad esperada para cada jugador viene dada por:

$$\mathbb{E}[u_i(\sigma_1, \sigma_2)] = \sum_{s_1 \in S_1} \sum_{s_2 \in S_2} \sigma_1(s_1) \sigma_2(s_2) u_i(s_1, s_2).$$

Sustituyendo las distribuciones uniformes, obtenemos:

$$\mathbb{E}[u_i(\sigma_1^*, \sigma_2^*)] = 0 \quad \forall i \in J.$$

Este perfil de estrategias mixtas constituye un equilibrio de Nash, ya que ningún jugador puede mejorar su utilidad esperada mediante una desviación unilateral:

$$\mathbb{E}[u_i(\sigma_1^*, \sigma_2^*)] \geq \mathbb{E}[u_i(\sigma_1, \sigma_2^*)] \quad \forall \sigma_1 \in \Delta(S_1)$$

donde $\Delta(S_1)$ denota el conjunto de distribuciones de probabilidad sobre el conjunto de estrategias puras S_1 . El mismo razonamiento es válido para el jugador 2.

En este equilibrio, cada jugador selecciona piedra, papel o tijeras con igual probabilidad, de modo que el juego permanece perfectamente simétrico y de suma cero. Así, piedra, papel o tijeras constituye un ejemplo clásico de equilibrio mixto en un juego competitivo simple.

3 Ejemplos de juegos

El estudio de juegos como piedra, papel o tijeras resulta especialmente útil porque ejemplifica un patrón de relaciones cíclicas de ventaja/desventaja que aparece en muchos sistemas lúdicos reales. En efecto, numerosos subsistemas de juegos más complejos se fundamentan en equilibrios similares, en los que distintas categorías interactúan mediante ventajas y debilidades recíprocas. Por ejemplo, esta estructura está presente en los tipos elementales de *Pokémon* [56] y en los roles o tipos de campeones en los juegos MOBA (Massive Online Battle Arena), donde cada tipo puede tener ventaja frente a uno y desventaja frente a otro, generando una dinámica de contrarrestes mutuos. [4, 57]

3.1.2. Juegos de información incompleta

3.1.2.1. Mercado del limón

El mercado del limón es un ejemplo clásico de juego de información incompleta, introducido por George Akerlof en su artículo *The Market for “Lemons”* [5]. Este modelo ilustra cómo la asimetría de información puede provocar un fallo de mercado.

Supongamos un mercado de automóviles usados en el que hay dos tipos de coches: malos y buenos (llamados “limones” y “melocotones” respectivamente). Los vendedores conocen la calidad real de su coche, pero los compradores no pueden distinguir entre un coche bueno y uno malo antes de la compra. Ambos saben, sin embargo, la proporción de coches buenos y malos en el mercado, así como los valores medios esperados.

En este contexto, la información sobre la calidad del coche no es completa: los vendedores conocen más que los compradores. Por tanto, el juego presenta información incompleta.

Podemos representarlo de manera simplificada como un juego estático de tipo bayesiano con dos jugadores:

$$J = \{\text{Vendedor, Comprador}\}.$$

El vendedor puede tener dos tipos posibles:

$$T_{\text{Vendedor}} = \{\text{Melocotón, Limón}\}.$$

El comprador no conoce el tipo del vendedor, pero tiene una creencia inicial sobre la probabilidad de que el coche sea bueno o malo. Denotamos estas probabilidades por p y $1 - p$, respectivamente.

Las estrategias posibles son:

$$S_{\text{Vendedor}} = \{\text{Ofrecer, No ofrecer}\}, \quad S_{\text{Comprador}} = \{\text{Comprar, No comprar}\}.$$

Si el comprador adquiere un coche bueno, su utilidad neta es el valor del coche v_b menos el precio p_c . Si compra un coche malo, obtiene $v_m - p_c$, con $v_m < v_b$. Para el vendedor, el pago depende del precio recibido menos su valoración del coche.

Debido a la asimetría de información, el comprador anticipa que, a cualquier precio intermedio, los vendedores de coches buenos tendrán menos incentivo para vender que los de coches malos. Esto reduce la calidad esperada de los coches ofertados y, en equilibrio, puede hacer que desaparezca el mercado: sólo se venden coches malos.

En resumen, el *mercado del limón* muestra cómo la falta de información simétrica puede generar un equilibrio ineficiente, en el que las transacciones potencialmente mutuamente

beneficiosas no se llevan a cabo. Este resultado pone de manifiesto que la información incompleta puede alterar de forma sustancial el equilibrio de un mercado.

Los sistemas de apuestas resultan de gran interés tanto en contextos económicos reales como en el diseño de juegos de mesa modernos. En particular, constituyen una mecánica habitual dentro del género de los *Eurogames*. Este subgénero se caracteriza, entre otros aspectos, por priorizar la solidez mecánica frente a la temática y por propiciar formas de interacción más sutiles entre los jugadores, basadas en el conflicto indirecto, lo que explica la popularidad de las mecánicas de pujas y apuestas [54]. Algunos diseñadores de reconocido prestigio, como Reiner Knizia (matemático de formación y autor de más de setecientos juegos publicados), incorporan principios relacionados con el equilibrio bayesiano en los mecanismos de pujas y puntuación, con el objetivo de generar sistemas estratégicamente ricos y dinámicos [21].

3.1.2.2. Póker a una ronda o *Kuhn*

El póker a una ronda, también conocido como *Kuhn poker*, constituye un ejemplo paradigmático de juego con información incompleta y componente estratégico de engaño, ampliamente estudiado en la Teoría de Juegos como modelo de interacción con información privada y señales. A diferencia de juegos como el dilema del prisionero o piedra, papel o tijeras, aquí los jugadores no disponen de información completa sobre el estado del juego, lo que introduce incertidumbre y estrategias mixtas basadas en probabilidades y credibilidad.

Consideremos un juego simplificado de póker entre dos jugadores. Cada uno recibe una carta del mazo, la cual puede ser alta (A) o baja (B), con igual probabilidad. El valor de la carta es conocido solo por el jugador que la recibe. A continuación, el jugador 1 puede apostar (subir) o pasar. Si pasa, el jugador 2 también pasa y ambos muestran sus cartas: gana el jugador con la carta más alta. Si el jugador 1 apuesta, el jugador 2 puede retirarse o igualar la apuesta. Si se retira, el jugador 1 gana automáticamente; si iguala, ambos muestran sus cartas y el jugador con la carta más alta gana el bote [43].

Este escenario es un juego de información incompleta, ya que cada jugador conoce su propia carta pero desconoce la del oponente. La incertidumbre sobre la mano del rival da lugar a estrategias de *bluffing* (faroles), en las cuales un jugador con una carta baja puede apostar intentando hacer creer al otro que posee una carta alta.

Formalmente, podemos definir el juego como:

$$J = \{1, 2\}, \quad T_i = \{\text{Alta, Baja}\} \text{ para } i = 1, 2.$$

Cada jugador conoce su tipo, pero desconoce el del rival. Las estrategias disponibles son:

$$S_1 = \{\text{Apostar si Alta, Apostar si Baja, Pasar}\}, \quad S_2 = \{\text{Igualar, Retirarse}\}.$$

El póker a una ronda es un juego bayesiano simple: cada jugador dispone de información privada (su carta) y toma decisiones estratégicas en función de sus creencias sobre el oponente. En equilibrio bayesiano, las estrategias suelen ser mixtas e incluyen, ocasionalmente, apuestas con manos débiles (faroles) para preservar la incertidumbre. No obstante, la forma concreta del equilibrio depende críticamente de los parámetros del modelo, como la probabilidad *ex ante* de cada tipo y el tamaño relativo de las apuestas. [36]

3 Ejemplos de juegos

Este modelo, formulado originalmente por John C. Harsanyi como ejemplo de juego bayesiano, muestra cómo la información privada y las estrategias de señalización pueden influir de manera decisiva en el resultado del juego [24].

3.2. Juegos dinámicos

En esta sección se presentan diversos juegos que ilustran la categoría de juegos dinámicos. Los ejemplos seleccionados no solo poseen relevancia en la literatura académica, sino que también constituyen elecciones populares en contextos recreativos y competitivos.

3.2.1. Juegos de información completa

3.2.1.1. Tres en raya

El tres en raya es un ejemplo clásico de juego de información completa, finito y determinista. Ambos jugadores observan todas las jugadas y conocen exactamente el estado del tablero en cada momento.

Dos jugadores, X y O , alternan turnos colocando su símbolo en una cuadrícula de 3×3 . Gana quien consiga alinear tres de sus símbolos en fila, columna o diagonal. Si todas las casillas se llenan sin que nadie consiga alinear tres, el juego termina en empate.

Podemos representar el juego en forma extensiva, donde cada nodo corresponde a un estado del tablero y cada rama a una jugada posible. Dado que ambos jugadores observan el tablero completo, no hay información oculta.

En términos formales:

$$J = \{X, O\}, \quad S_X, S_O = \text{conjunto de secuencias posibles de movimientos válidos.}$$

Los pagos se definen como:

$$u_X = \begin{cases} 1 & \text{si } X \text{ gana,} \\ 0 & \text{si hay empate,} \\ -1 & \text{si } O \text{ gana,} \end{cases}$$

$$u_O = -u_X.$$

La representación en forma extensiva permite visualizar las decisiones secuenciales y el carácter de información perfecta del juego. En la Figura 3.1 se muestra un fragmento del árbol de decisión correspondiente al tres en raya.

El juego es de suma cero y simétrico. Su análisis permite determinar estrategias dominantes y soluciones mediante retroinducción. De hecho, se sabe que, con juego perfecto de ambos lados, el resultado siempre será un empate. El tres en raya ilustra un caso de equilibrio de Nash puro, alcanzado cuando ninguno de los jugadores puede mejorar su resultado cambiando unilateralmente de estrategia.

Resulta pertinente retomar las definiciones expuestas en la sección 1.2, pues el tres en raya constituye un ejemplo limítrofe entre el concepto de “juego” y el de “puzzle”. Dado que

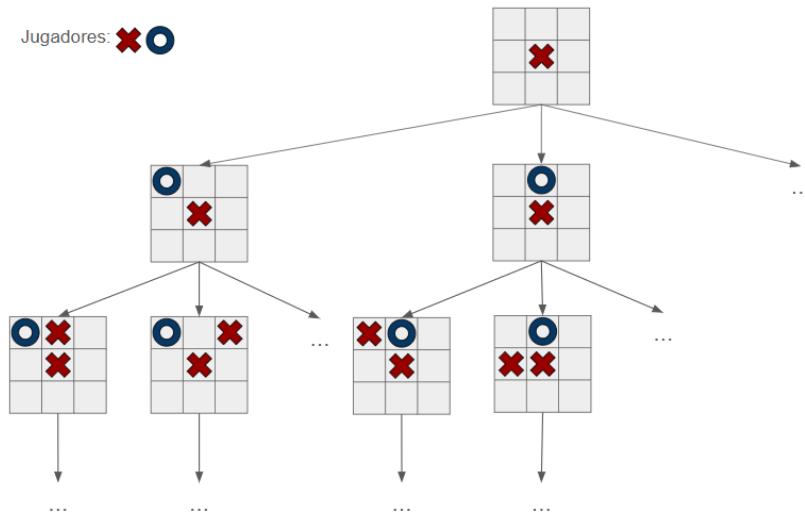


Figura 3.1: Representación parcial en forma extensiva del juego de tres en raya. Creación propia.

posee un equilibrio perfectamente definido: en el que, al jugar de forma óptima, la partida siempre concluye en empate. La interacción estratégica y la resolución de problemas son prácticamente inexistentes. No obstante, para jugadores que desconocen dicha estrategia óptima, como puede ser el caso de los niños pequeños, el tres en raya sí cumple las condiciones necesarias para ser considerado un juego. Este ejemplo pone de manifiesto que la naturaleza lúdica depende tanto de la experiencia del jugador como del sistema de reglas en sí. [35]

3.2.1.2. Ajedrez

El ajedrez es otro ejemplo representativo de juego de información completa, aunque con una complejidad combinatoria mucho mayor que el tres en raya. En este caso, los jugadores también observan completamente el tablero, las piezas y los movimientos posibles del adversario.

Dos jugadores, Blancas y Negras, alternan turnos moviendo sus piezas sobre un tablero de 8×8 siguiendo reglas bien definidas. El objetivo es dar jaque mate al rey contrario, es decir, situarlo bajo amenaza de captura sin posibilidad de escape.

Formalmente:

$$J = \{\text{Blancas, Negras}\}, \quad S_i = \text{conjunto de secuencias legales de movimientos.}$$

3 Ejemplos de juegos

Los pagos pueden definirse como:

$$u_{\text{Blancas}} = \begin{cases} 1 & \text{si Blancas ganan,} \\ 0 & \text{si hay tablas,} \\ -1 & \text{si Negras ganan,} \end{cases}$$

$$u_{\text{Negras}} = -u_{\text{Blancas}}.$$

Aunque el ajedrez es teóricamente resoluble por ser un juego finito de información perfecta, su tamaño hace inviable el cálculo completo de la forma extensiva o del árbol de decisión. Sin embargo, la teoría garantiza que existe un equilibrio de Nash en estrategias puras: una secuencia de jugadas óptimas para ambos jugadores que determina el resultado final (victoria de uno o tablas).

El ajedrez es, por tanto, un ejemplo paradigmático de cómo los juegos de información completa pueden ser analizados mediante los principios de la Teoría de Juegos, aunque su resolución práctica requiera aproximaciones computacionales o heurísticas.

3.2.2. Juegos de información incompleta

3.2.2.1. Póker Texas Hold'em

El *Texas Hold'em* es la variante de póker más popular tanto en entornos recreativos como competitivos. Se trata de un juego de información incompleta, en el que cada jugador dispone de información privada (sus cartas ocultas) y de información pública (las cartas comunitarias visibles por todos). El desarrollo secuencial de las apuestas genera un espacio de decisión extenso que combina incertidumbre y estrategia probabilística.

El juego utiliza una baraja estándar de 52 cartas y puede involucrar entre dos y diez jugadores. Cada uno recibe dos cartas privadas, conocidas como *hole cards*, mientras que cinco cartas comunitarias se revelan progresivamente en el centro de la mesa. Los jugadores forman su mejor mano de cinco cartas combinando sus dos cartas privadas con las cinco comunitarias.

Cada ronda de *Texas Hold'em* se compone de las siguientes fases [23]:

1. **Apuestas iniciales y ciegas.** Antes de repartir las cartas se colocan dos apuestas forzadas: la *ciega pequeña* (*small blind*) y la *ciega grande* (*big blind*). Estas apuestas garantizan la existencia de un bote inicial e introducen incentivos estratégicos desde el primer momento.
2. **Pre-flop.** Cada jugador recibe sus dos cartas privadas y decide si retirarse (*fold*), igualar la apuesta (*call*) o subir (*raise*).
3. **Flop.** Se descubren tres cartas comunitarias boca arriba. Tiene lugar una nueva ronda de apuestas en la que los jugadores actualizan sus creencias sobre la fuerza de su mano.
4. **Turn.** Se revela una cuarta carta comunitaria, seguida de otra ronda de apuestas.
5. **River.** Se muestra la quinta carta comunitaria y se realiza la última ronda de apuestas.
6. **Showdown.** Si quedan varios jugadores activos, muestran sus cartas privadas y gana quien tenga la mejor combinación de cinco cartas.

Ejemplo.

Consideremos una mano simplificada entre tres jugadores: A, B y C. Las ciegas son de 50 y 100 fichas, y el juego es *No-Limit*.

- A recibe As♣ y 10♥, B recibe 7♣ y 7♦, C recibe K♣ y 9♣.
- En el pre-flop, A sube a 300 fichas, B iguala y C se retira.
- En el flop (10♦, 7♣, 3♣), B obtiene un trío de sietes y A una pareja de dieces. A apuesta 400 fichas y B iguala.
- En el turn (Q♣), B mantiene la ventaja.
- En el river (10♣), A mejora a trío de dieces y gana la mano, obteniendo todas las fichas apostadas.

Este ejemplo ilustra cómo la información incompleta y las apuestas sucesivas inducen comportamientos estratégicos complejos: faroles, semifaroles, gestión del riesgo y deducción probabilística del rango del oponente.

Siguiendo la formulación de Harsanyi para juegos bayesianos [24], en el *Texas Hold'em* se identifican los siguientes elementos:

$$J = \{1, 2, \dots, n\},$$

donde n es el número de jugadores.

Cada jugador i recibe un tipo privado $t_i \in T_i$, que corresponde a sus dos cartas iniciales (*hole cards*). La distribución de tipos está dada por una probabilidad comúnmente conocida por combinatoria, lo que garantiza información común sobre la estructura del azar.

Las estrategias puras de un jugador i pueden definirse como funciones:

$$s_i : T_i \times H_i \rightarrow A_i,$$

donde H_i representa el historial público de apuestas y cartas reveladas, y A_i el conjunto de acciones posibles en cada ronda: $A_i = \text{retirarse, igualar, subir}$.

Los pagos u_i dependen del resultado final (la mejor mano de cinco cartas y el tamaño del bote) y se distribuyen como:

$$u_i(s_1, \dots, s_n, t_1, \dots, t_n) = \begin{cases} b_i, & \text{si el jugador } i \text{ gana el bote,} \\ -b'_i, & \text{si pierde la mano,} \\ 0, & \text{en caso de empate.} \end{cases}$$

El equilibrio de Bayes-Nash se alcanza cuando cada jugador elige su estrategia óptima dada su información privada y las creencias sobre las estrategias de los demás.

El *Texas Hold'em* combina información privada, azar y señales públicas en una estructura secuencial. Su análisis dentro de la Teoría de Juegos permite estudiar conceptos como la actualización bayesiana de creencias, la racionalidad limitada y el equilibrio estratégico bajo incertidumbre. Por ello, constituye uno de los modelos más utilizados en la investigación sobre decisión estratégica y Teoría de Juegos aplicada a la economía del comportamiento [11].

3.2.2.2. Magic: The Gathering

Magic: The Gathering (MTG)¹ [3] es un *trading card game* (TCG)² o juego de cartas colecciónables de temática fantástica cuyo objetivo consiste en derrotar al oponente reduciendo sus puntos de vida a cero mediante el uso de criaturas, conjuros y encantamientos. Diseñado en 1993 por Richard Garfield [2], MTG fue el título que popularizó el género y estableció muchas de las bases mecánicas que aún hoy definen los TCG. Su influencia en el diseño moderno de juegos de mesa y cartas es incuestionable: resulta difícil encontrar un diseñador que no cite *Magic: The Gathering* como una de sus principales referencias, o un jugador que no haya oído hablar de él. [20]

Los trading card games (TCCG o TCG) son juegos de cartas en los que los participantes construyen sus propios mazos a partir de una amplia colección de cartas con diferentes habilidades, valores y funciones. Estas cartas suelen obtenerse mediante sobres aleatorios o intercambios entre jugadores, lo que introduce un componente económico y de coleccionismo además del estratégico. El diseño de un TCG combina gestión de recursos, construcción de mazos y toma de decisiones tácticas, generando una profundidad de juego que equilibra azar y habilidad.

En el caso de MTG, en su modalidad *standard*, dos jugadores se enfrentan con mazos de al menos 60 cartas y una reserva inicial de 20 puntos de vida. El desarrollo de la partida se organiza en turnos compuestos por varias fases: mantenimiento, principal y combate. En los que los jugadores pueden desplegar criaturas, lanzar conjuros o activar habilidades. Uno de los elementos más característicos del sistema de reglas es la gestión del maná, recurso con el que se pagan los costes de las cartas. Este se obtiene de las cartas de *Tierra*, que se pueden jugar una por turno y deben “girarse” para generar maná de distintos colores. La correcta administración de este recurso determina las acciones disponibles en cada turno y condiciona la estrategia general, ya que las cartas más poderosas exigen mayores cantidades o combinaciones específicas de maná [38].

MTG combina elementos de información incompleta, el contenido del mazo y la mano de cada jugador permanecen ocultos, con un marcado componente de planificación previa, representado por la construcción del mazo o *deckbuilding*. Esta doble dimensión, estratégica y táctica, genera un espacio de decisión extenso y altamente complejo. Según Ganivet et al. [38], la estructura secuencial y la dependencia contextual de las decisiones hacen de MTG un entorno idóneo para el estudio de la inteligencia artificial y la optimización heurística, ya que el número de posibles configuraciones supera ampliamente cualquier capacidad de cálculo exhaustivo.

En consecuencia, MTG puede considerarse un juego con múltiples equilibrios dinámicos, donde las estrategias óptimas dependen del contexto: composición del mazo, tipo de oponente y evolución de la partida. Los estudios recientes demuestran que incluso agentes basados en reglas o algoritmos evolutivos tienden a converger hacia comportamientos *cuasi-óptimos* sin alcanzar un equilibrio estable en el sentido clásico. Esta propiedad lo convierte en un modelo de referencia para el análisis de sistemas con información oculta y toma de decisiones secuencial. En nuestro caso, el paralelismo con MTG es directo: el jugador debe gestionar información incompleta, optimizar recursos limitados y equilibrar riesgo y recompensa dentro de un entorno con resultados parcialmente impredecibles. [9]

¹A partir de ahora usaremos MTG para referirnos a *Magic: The Gathering*

²A partir de ahora usaremos TCG para referirnos a los *trading card games*

Capítulo 4: Algoritmos para IA en juegos

Una vez expuestos los distintos tipos de juegos y sus características fundamentales, el siguiente paso consiste en analizar los mecanismos mediante los cuales pueden ser jugados o resueltos. Si bien algunos juegos permiten identificar soluciones de equilibrio de forma analítica, en otros casos resulta necesario recurrir a agentes capaces de determinar y ejecutar estrategias de manera autónoma. El estudio de estos agentes permite aproximarse a la resolución de juegos complejos, incluso cuando no es posible obtener una solución exacta. En tales situaciones, el objetivo pasa a ser el diseño de agentes que actúen de forma racional dentro del entorno del juego, optimizando su comportamiento.

4.1. Agentes

En inteligencia artificial, un **agente** es una entidad que percibe su entorno mediante sensores y actúa sobre él a través de actuadores [39]. El comportamiento del agente está determinado por una función agente que asigna una acción a cada secuencia posible de percepciones. Un **agente racional** elige las acciones que maximicen su medida de rendimiento esperada, en función de las percepciones y el conocimiento disponible.

Según su complejidad, pueden distinguirse agentes reactivos simples, basados en modelos, basados en objetivos y basados en utilidad. Cada tipo amplía la capacidad del agente para razonar sobre el entorno, planificar y adaptarse, acercándose a un comportamiento racional en contextos dinámicos e inciertos.

4.2. Algoritmos para juegos con información completa

Esta sección se basa en los contenidos de *Inteligencia Artificial: Un enfoque moderno* (Russell, Stuart J. and Norvig, Peter, 2022) [39].

Los algoritmos que se presentan a continuación suponen un entorno competitivo de juego de suma cero, tal como se definió en la sección anterior y se analizó en el *Teorema de equilibrio y estrategias minimax* (Teorema 2.6). En este tipo de juegos, las ganancias de un jugador equivalen exactamente a las pérdidas del otro, por lo que el análisis puede realizarse desde la perspectiva de un único jugador: maximizar su utilidad implica simultáneamente minimizar la de su oponente.

4.2.1. Minimax

El algoritmo **minimax** constituye la base de la toma de decisiones óptima en juegos secuenciales de información completa con suma cero. Su objetivo es determinar la estrategia que maximiza la ganancia mínima posible del jugador, asumiendo que el oponente juega de forma perfecta.

Sea un juego representado de forma extensiva por un árbol de decisión en el que los nodos representan estados n y los arcos representan acciones posibles. Los nodos terminales están asociados a valores de utilidad $u(s)$, que cuantifican el resultado final para el jugador maximizador. En cada nivel del árbol, los jugadores alternan su turno:

- El jugador **MAX** elige la acción que **maximiza** el valor esperado.

4 Algoritmos para IA en juegos

- El jugador **MIN** elige la acción que **minimiza** dicho valor, buscando reducir la ganancia del rival.

Formalmente, el valor de un estado n se define recursivamente como:

$$V(n) = \begin{cases} u(n) & \text{si } n \text{ es terminal,} \\ \max_{s \in \text{Sucesores}(n)} V(s) & \text{si } n \text{ es un estado de MAX,} \\ \min_{s \in \text{Sucesores}(n)} V(s) & \text{si } n \text{ es un estado de MIN.} \end{cases}$$

Llamamos a $V(n)$ el **valor minimax** del nodo n . Si n es un nodo terminal, $u(n)$ representa la utilidad asociada al resultado alcanzado siguiendo la secuencia de decisiones que conduce hasta n .

De esta forma, el algoritmo explora el árbol de juego hasta las hojas y propaga hacia arriba los valores de utilidad, seleccionando en la raíz la acción con el valor máximo.

El agente que aplica el algoritmo minimax puede interpretarse como un agente basado en modelos, ya que mantiene una representación explícita del entorno del juego (el árbol de estados) y razona sobre él para seleccionar la acción óptima. Este agente asume que el oponente también actúa racionalmente, maximizando su propia utilidad, lo que lo convierte en un agente racional en el sentido definido al inicio del capítulo.

4.2.1.1. Minimax en el tres en raya.

Se desarrolla ahora el caso del tres en raya, definido anteriormente.

Un agente minimax para este juego actúa de la siguiente forma:

1. Genera todos los movimientos posibles desde el estado actual.
2. Evalúa los estados terminales según la función de utilidad definida.
3. Propaga los valores hacia atrás aplicando las reglas de maximización y minimización.
4. Selecciona el movimiento con el valor $V(s)$ máximo.

La representación en forma extensiva permite visualizar las decisiones secuenciales y el carácter de información perfecta del juego. En la Figura 4.1 se muestra un fragmento del árbol de decisión correspondiente a una partida que termina en empate. Cada nodo alterna entre decisiones de MAX (X) y MIN (O), y las utilidades en las hojas determinan la estrategia óptima.

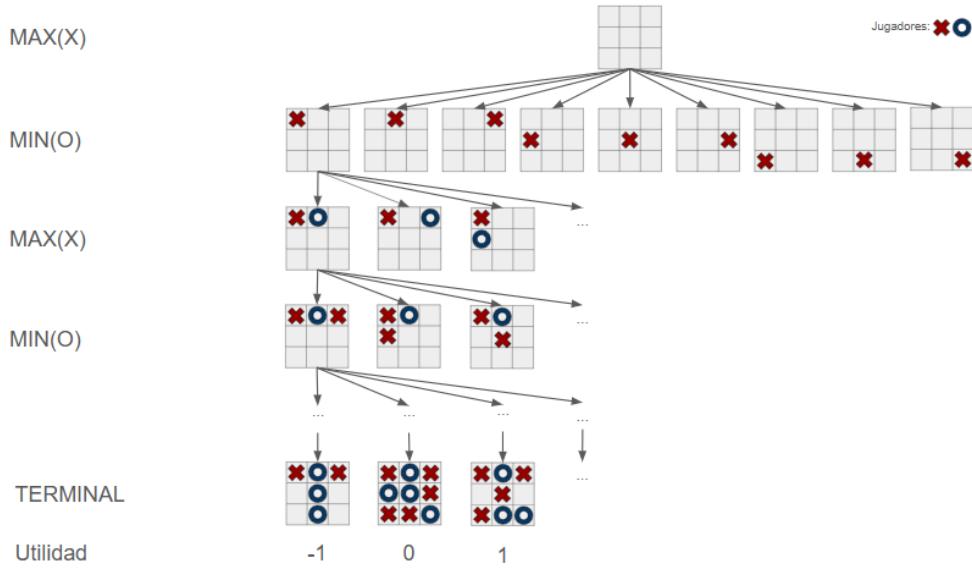


Figura 4.1: Un árbol (parcial) de búsqueda minimax para el juego de tres en raya. Creación propia.

4.2.2. Poda α - β

La poda α - β es una optimización del algoritmo minimax que reduce el número de nodos evaluados sin alterar el resultado final óptimo. Se basa en el hecho de que, en muchos casos, es posible detectar que cierta rama del árbol no puede cambiar la decisión definitiva y descartarla anticipadamente.

La poda α - β debe su nombre de los dos parámetros que describen los umbrales sobre los valores hacia atrás que aparecen durante la exploración:

- α : el valor (máximo) garantizado hasta ahora para el jugador MAX.
- β : el valor (mínimo) garantizado hasta ahora para el jugador MIN.

La poda α - β actualiza el valor de α y β según se va explorando el árbol y poda (es decir, deja de recorrer), las ramas restantes en un nodo en el momento en el valor del nodo actual es peor que el actual valor α o β para MAX o MIN, respectivamente.

Formalmente, la función recursiva se puede escribir como:

$$\text{AlphaBeta}(n, \alpha, \beta) = \begin{cases} u(n), & \text{si } n \text{ es terminal,} \\ \max_{s \in \text{Sucesores}(n)} \text{AlphaBeta}(s, \alpha, \beta), & \text{si turno = MAX,} \\ \min_{s \in \text{Sucesores}(n)} \text{AlphaBeta}(s, \alpha, \beta), & \text{si turno = MIN,} \end{cases}$$

pero con las condiciones de poda:

4 Algoritmos para IA en juegos

- En el caso MAX: cuando calculamos un sucesor s y obtenemos un valor $v = \text{AlphaBeta}(s, \alpha, \beta)$, si $v > \alpha$ actualizamos $\alpha = v$. Si en algún momento $\alpha \geq \beta$, rompemos el bucle de sucesores (corte).
- En el caso MIN: cuando obtenemos un sucesor con valor v , si $v < \beta$ actualizamos $\beta = v$. Si $\beta \leq \alpha$, hacemos corte (podar).

El agente que utiliza poda $\alpha - \beta$ mantiene el mismo tipo de racionalidad que el agente minimax, pero optimiza su proceso de decisión reduciendo la cantidad de estados explorados. Se trata, por tanto, de un agente basado en modelos que emplea un mecanismo de búsqueda más eficiente para alcanzar decisiones racionales en tiempo finito.

Es por esto que se utiliza ampliamente para juegos complejos bipersonales de suma cero, como el ajedrez o el tres en raya.

4.2.2.1. El algoritmo de búsqueda $\alpha - \beta$

Algorithm 1 Búsqueda Alfa-Beta (adaptado de [39]).

```

1: function BÚSQUEDA-ALFA-BETA(estado)
2:   entrada: estado, estado actual del juego
3:    $v \leftarrow \text{MAX-VALOR}(\text{estado}, -\infty, +\infty)$ 
4:   return la acción de SUCESORES(estado) con valor  $v$ 
5: end function

6: function MAX-VALOR(estado,  $\alpha, \beta$ )
7:   entrada: estado,  $\alpha$  valor de la mejor alternativa para MAX,  $\beta$  para MIN
8:   if TEST-TERMINAL(estado) then
9:     return UTILIDAD(estado)
10:  end if
11:   $v \leftarrow -\infty$ 
12:  for all  $s$  en SUCESORES(estado) do
13:     $v \leftarrow \max(v, \text{MIN-VALOR}(s, \alpha, \beta))$ 
14:    if  $v \geq \beta$  then return  $v$ 
15:    end if
16:     $\alpha \leftarrow \max(\alpha, v)$ 
17:  end for
18:  return  $v$ 
19: end function

20: function MIN-VALOR(estado,  $\alpha, \beta$ )
21:   entrada: estado,  $\alpha$  valor de la mejor alternativa para MAX,  $\beta$  para MIN
22:   if TEST-TERMINAL(estado) then
23:     return UTILIDAD(estado)
24:   end if
25:    $v \leftarrow +\infty$ 
26:   for all  $s$  en SUCESORES(estado) do
27:      $v \leftarrow \min(v, \text{MAX-VALOR}(s, \alpha, \beta))$ 
28:     if  $v \leq \alpha$  then return  $v$ 
29:     end if
30:      $\beta \leftarrow \min(\beta, v)$ 
31:   end for
32:   return  $v$ 
33: end function

```

4.2.2.2. $\alpha-\beta$ en el tres en raya

A diferencia del ejemplo de minimax, donde mostramos únicamente un fragmento del árbol debido a su tamaño, la poda $\alpha-\beta$ nos permite representar la exploración completa del juego de manera eficiente. En la Figura 4.2 se muestra un árbol de decisión completo para una partida de tres en raya, donde se indica en cada nodo el valor actual de α y β durante la búsqueda.

Este ejemplo ilustra cómo la poda evita explorar ramas innecesarias, reduciendo el número de nodos evaluados sin afectar el resultado final. Todos los caminos analizados conducen a

4 Algoritmos para IA en juegos

un empate, lo que confirma que el tres en raya es un juego con equilibrio: si ambos jugadores actúan de manera racional, el resultado óptimo es siempre el empate. La figura permite visualizar de forma clara la dinámica de actualización de los valores α y β en cada nivel del árbol, y cómo se producen las podas en las ramas que no pueden mejorar la decisión del jugador MAX o MIN.

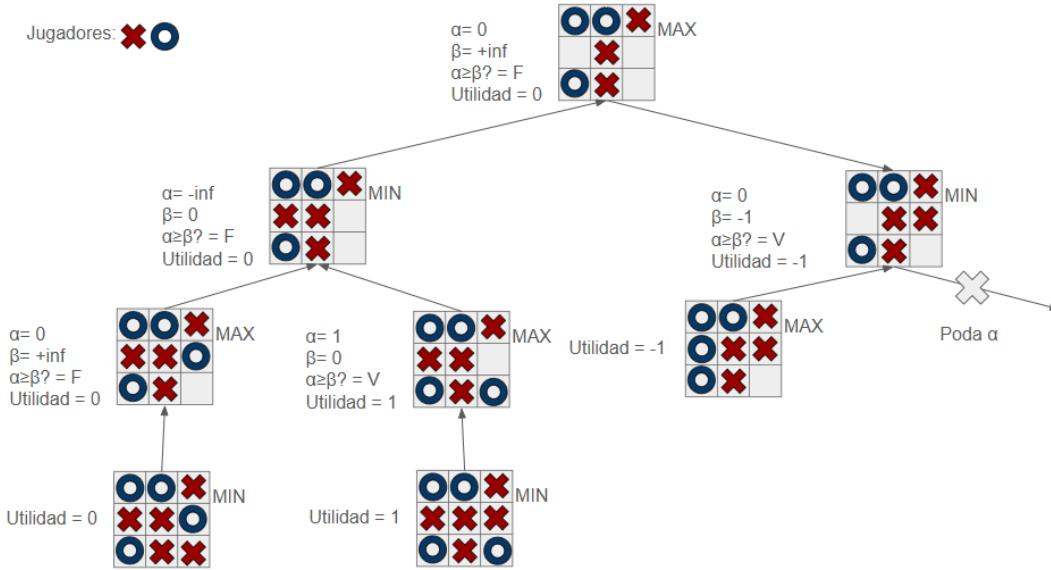


Figura 4.2: Árbol completo de búsqueda α - β para el juego de tres en raya. Cada nodo muestra los valores α y β en el momento de la evaluación, y las ramas podadas se omiten para indicar la eficiencia de la búsqueda. Creación propia.

4.2.3. Heurísticas

La búsqueda informada usa conocimiento específico del problema para guiar la búsqueda hacia el objetivo más eficientemente que con estrategias sin información adicional. Se denomina también búsqueda primero el mejor (*best-first search*).

Una **función heurística** $h(n)$ estima el costo más bajo desde un estado n hasta un estado meta. Debe cumplir al menos:

- $h(n) \geq 0$ para todo n .
- Si n es meta, $h(n) = 0$.

Dos propiedades importantes:

- **Admisibilidad:** nunca sobreestima el costo real mínimo hasta la meta.
- **Consistencia** (o monótona): para cada transición de n a n' con costo $c(n, n')$,

$$h(n) \leq c(n, n') + h(n').$$

4.2.3.1. Algoritmos heurísticos destacados

- **Búsqueda local voraz:** selecciona el nodo con heurística $h(n)$ mínima en cada paso. Se enfoca fuertemente en acercarse al objetivo pero puede no encontrar un camino óptimo o quedarse atascado en ciclos.
- **Búsqueda A^* :** combina el costo hasta el nodo $g(n)$ con la heurística $h(n)$. Usa la función de evaluación

$$f(n) = g(n) + h(n).$$

Si la heurística es admisible (y consistente), A^* garantiza encontrar un camino óptimo.

Los algoritmos voraces pueden ser muy rápidos en algunos casos, pero ni óptimos ni completos en general.

A^* es más costoso en espacio y tiempo que los voraces, pero ofrece garantía de optimidad si la heurística lo permite. En comparación con búsqueda sin información, las heurísticas bien diseñadas reducen dramáticamente el número de nodos explorados.

La calidad de la heurística (qué tan cerca está de la estimación real) determina cuánta mejora se logra. En muchos casos, estas heurísticas se obtienen al analizar las reglas del juego combinado con ensayo y error.

Los agentes que emplean heurísticas pueden considerarse agentes basados en objetivos o en utilidad, ya que no sólo razonan sobre estados posibles, sino que evalúan su conveniencia mediante una función de utilidad o de costo estimado. Esto les permite actuar de forma racional incluso sin explorar exhaustivamente todos los estados posibles

4.3. Algoritmos para juegos con información incompleta

En los juegos de información completa, como el ajedrez o el tres en raya, los algoritmos basados en minimax y las técnicas de poda α - β permiten explorar el árbol de decisiones de forma más o menos exhaustiva para determinar secuencias óptimas de jugadas. Sin embargo, cuando el juego presenta información incompleta (que contienen, por ejemplo, cartas ocultas, información asimétrica o elementos aleatorios), el árbol de juego tradicional se transforma en una estructura con nodos de información y nodos de azar, haciendo inviable su exploración completa debido al crecimiento combinatorio y a la incertidumbre inherente al entorno.

Por ello, en los juegos con información incompleta se desarrollan alternativas basadas en agentes de aprendizaje, que pueden ser de dos tipos principales: basados en modelos probabilísticos o de aprendizaje adaptativo. Estos agentes emplean simulaciones y muestreo estadístico para estimar el valor esperado de cada decisión sin necesidad de recorrer exhaustivamente todas las ramas posibles del juego. Dentro de este enfoque destacan los métodos de búsqueda por árboles Monte Carlo y los algoritmos evolutivos, ampliamente utilizados en juegos de tablero, estrategia y cartas.

4.3.1. Monte Carlo

Los métodos *Monte Carlo* se basan en la idea de estimar el valor de una acción mediante simulaciones aleatorias repetidas. En lugar de explorar de manera exhaustiva todas las posibles jugadas, el algoritmo ejecuta múltiples partidas simuladas (*rollouts*) hasta alcanzar

estados terminales y promedia los resultados para aproximar el valor esperado de cada decisión. Esto permite que los agentes actúen de forma razonable incluso cuando el espacio de estados es muy amplio o incierto.

En juegos con información oculta, como el *Texas Hold'em*, los enfoques basados en *Monte Carlo Tree Search* (MCTS)¹ se han consolidado como herramientas eficaces para aproximar estrategias de equilibrio. Heinrich y Silver [25] evaluaron MCTS en variantes de póker como *Kuhn Poker* y *Limit Texas Hold'em*, introduciendo una versión modificada denominada *Smooth UCT*. Esta variante combina los principios de MCTS con técnicas de *fictitious play*, un método clásico de Teoría de Juegos en el que cada jugador asume que los demás mantendrán estrategias promedio basadas en su comportamiento pasado, suavizando la selección de acciones para mejorar la estabilidad del aprendizaje en entornos parcialmente observables.

En *Limit Texas Hold'em*, los autores entrenaron agentes mediante auto-juego (*self-play*) durante miles de millones de episodios. Dada la enorme complejidad del juego completo, aplicaron abstracciones del espacio de información, agrupando manos con valores estratégicamente similares en categorías o *buckets*. Esta reducción permitió que MCTS centrara su búsqueda en las regiones más relevantes del espacio estratégico, evitando el crecimiento exponencial del árbol.

El entrenamiento mediante *self-play* permitió que los agentes actualizaran sus políticas tras cada simulación. Los resultados mostraron que tanto el UCT estándar como *Smooth UCT* alcanzaron políticas competitivas frente a bots de referencia, aproximando comportamientos cercanos al equilibrio de Nash. Aunque no se demostró formalmente la convergencia teórica en juegos con información imperfecta, los resultados empíricos evidencian que MCTS es capaz de generar estrategias robustas mediante aprendizaje autónomo y simulación masiva.

Estos avances consolidan el uso de MCTS en juegos con información incompleta, donde la complejidad combinatoria y la incertidumbre hacen inviable el uso de métodos deterministas. El caso del *Texas Hold'em* ejemplifica cómo el muestreo Monte Carlo permite equilibrar exploración y explotación para crear estrategias efectivas en entornos parcialmente observables.

4.3.2. Algoritmos Evolutivos y Genéticos

Los algoritmos evolutivos (AE)² constituyen una familia de métodos de optimización inspirados en los procesos biológicos de la evolución natural. Su objetivo es encontrar soluciones óptimas o quasi-óptimas a problemas complejos mediante la iteración de una población de posibles soluciones que se modifican a lo largo del tiempo según principios de selección, mutación, recombinación y supervivencia del más apto. Entre sus variantes más destacadas se encuentran los algoritmos genéticos (AG)³, introducidos por John Holland en la década de 1970, los cuales utilizan representaciones cromosómicas (normalmente cadenas de bits, enteros o estructuras más complejas) para modelar el espacio de búsqueda [26].

El funcionamiento general de un algoritmo genético se basa en la generación inicial de una población de soluciones candidatas que se evalúan mediante una función de aptitud o *fitness function*. A partir de dicha evaluación, las soluciones con mejor desempeño se seleccionan para reproducirse, combinando sus características (cruce o *crossover*) y generando descendientes que pueden experimentar pequeñas alteraciones aleatorias (mutaciones). Este ciclo

¹A partir de ahora usaremos MCTS para referirnos a *Monte Carlo Tree Search*

²A partir de ahora usaremos AE para referirnos a los algoritmos evolutivos

³A partir de ahora usaremos AG para referirnos a los algoritmos genéticos

de selección y variación se repite durante múltiples generaciones, lo que permite explorar el espacio de búsqueda de manera estocástica y adaptativa hasta converger en soluciones de alta calidad.

4.3.2.1. Algoritmos Evolutivos en MTG

En el ámbito de los juegos, los algoritmos evolutivos han demostrado ser especialmente útiles para abordar problemas en los que el espacio de decisiones es amplio, no lineal y dependiente del contexto. Según Ganivet et al. [38], *Magic: The Gathering* (MTG) constituye un caso paradigmático en este sentido: la combinación de información incompleta (mazo y mano oculta), secuencias de decisiones interdependientes y un número astronómico de configuraciones posibles hacen inviable la aplicación de métodos de búsqueda exhaustivos. Por ponerlo en perspectiva, en una partida del formato *standard* suelen existir alrededor de 2000 cartas legales [1]. Si consideramos modalidades más amplias como *legacy*, en las que se permite utilizar prácticamente todas las cartas impresas a lo largo de la historia del juego, la cifra se incrementa de forma considerable.

En este contexto, los algoritmos genéticos permiten simular la evolución de estrategias a partir de un conjunto inicial de agentes que aprenden mediante iteraciones sucesivas, evaluando su desempeño frente a diferentes oponentes y adaptándose progresivamente a entornos cambiantes.

El uso de algoritmos evolutivos en MTG, y en general en juegos con alto grado de complejidad estratégica, se justifica por su capacidad para aproximarse a equilibrios dinámicos sin requerir un conocimiento total del sistema. A diferencia de enfoques deterministas, estos algoritmos pueden descubrir patrones emergentes y comportamientos eficaces a partir de datos incompletos, imitando en cierto modo la adaptación de los propios jugadores humanos.

En conclusión, los algoritmos evolutivos y genéticos resultan herramientas especialmente adecuadas para juegos con información oculta, donde la optimización directa resulta inviable. Además, su eficacia aumenta en dominios donde existe acceso a un gran volumen de datos de partidas reales, ya que estos pueden emplearse para entrenar o calibrar las funciones de evaluación, mejorando la capacidad del algoritmo para generalizar estrategias en contextos diversos.

Capítulo 5: Nuestro caso: *Bardtastic*

"Who lives, who dies, who tells your story?"

Lin-Manuel Miranda, Hamilton (2015) [33]

Con el marco teórico establecido, este capítulo presenta el caso práctico de *Bardtastic*, analizando su naturaleza como juego y su clasificación dentro de los modelos formales expuestos en capítulos anteriores. Este análisis permite situar el proyecto tanto desde una perspectiva de producto de software como desde la teoría de juegos, facilitando la justificación del algoritmo implementado para el agente inteligente desarrollado.

El objetivo es doble: en primer lugar, describir *Bardtastic* como producto y como sistema de reglas; en segundo lugar, identificar su tipología de juego para determinar las condiciones teóricas bajo las cuales opera el agente. Una vez caracterizado el entorno, se podrá discutir con precisión la estrategia de resolución y el comportamiento racional del agente dentro de él.

Aunque *Bardtastic* se presenta aquí bajo una doble lente técnica y formal, el análisis de las decisiones de diseño desde la perspectiva lúdica se abordará en capítulos posteriores.

5.1. Descripción general del juego

Bardtastic es un videojuego del género *roguelike deckbuilder*, ambientado en un universo de fantasía medieval poblado de bardos y cuentacuentos. El jugador encarna a un joven bardo que viaja desde su aldea hasta la capital del reino, compitiendo en distintas tabernas contra otros narradores con el objetivo de ganar fama, recursos y reconocimiento suficiente para abrir su propio teatro.

En términos de diseño, el juego combina dos estructuras principales:

- **Roguelike:** se estructura en partidas o *runs* independientes de duración media, donde cada intento presenta variaciones en el recorrido, los oponentes y las cartas disponibles. La progresión está sujeta a decisiones estratégicas y a elementos aleatorios que garantizan la rejugabilidad [46].
- **Deckbuilder:** el núcleo del sistema de juego consiste en la *construcción y gestión de mazos*. A lo largo de la partida, el jugador adquiere nuevas cartas, modifica su mazo y adapta su estrategia narrativa para mejorar sus probabilidades de éxito frente a distintos rivales [34].

Durante la partida, el jugador alterna entre enfrentamientos contra otros bardos y secciones de *deckbuilding*, que funcionan como eventos narrativos que representan su viaje entre tabernas. En estas fases es posible modificar el mazo, añadiendo, eliminando o reemplazando cartas.

Al finalizar cada partida se desbloquean elementos persistentes, como nuevos jefes, cartas o eventos, que estarán disponibles en intentos futuros. El juego cuenta con un final canónico, que se alcanza al vencer el enfrentamiento en la capital y su correspondiente viaje de vuelta.

Para desbloquearlo es necesario haber completado con éxito varias partidas y derrotado a los jefes intermedios.

Este apartado ofrece una visión general de la progresión del juego, aunque dicha progresión será retomada posteriormente al tratar el diseño. A continuación, se describe la dinámica de los enfrentamientos, donde interviene el agente de inteligencia artificial.

5.2. Dinámica de los enfrentamientos

Cada enfrentamiento tiene lugar en una taberna, donde el jugador se enfrenta a otro bardo controlado por la IA. Ambos disponen de un mazo de cartas que representa los elementos narrativos de su historia y las formas de ponerla en escena. Durante su turno, el jugador selecciona y juega cartas para construir y relatar su historia ante la audiencia. Para poder ser jugadas, las cartas deben rimar entre sí, siguiendo una estructura formal. Cada carta posee una rima anterior y una posterior, que determinan las secuencias válidas.

El objetivo es captar y mantener la atención del público mejor que el rival. La audiencia está compuesta por cinco miembros, que pueden prestar atención al jugador o a su oponente. La atención del público actúa como el recurso central y la métrica de éxito en cada enfrentamiento. Las cartas pueden producir efectos variados: modificar la atención ganada, robar nuevas cartas, alterar cartas ya jugadas o influir sobre el mazo adversario.

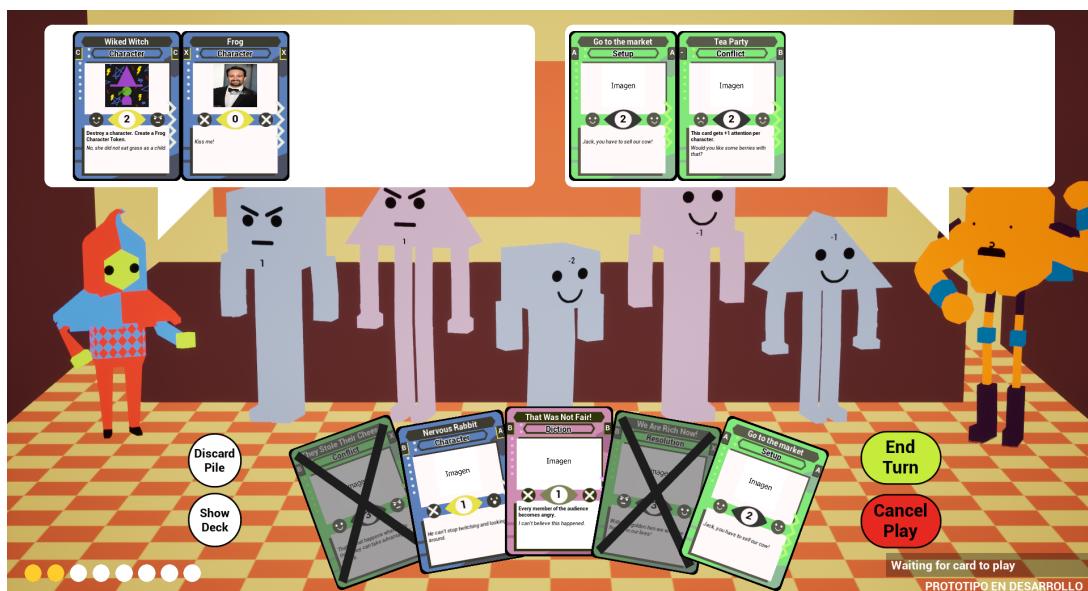


Figura 5.1: Pantalla principal de los enfrentamientos en *Bardtastic*. Creación propia.

Como se muestra en la Figura ??, la interfaz principal de los enfrentamientos presenta a los dos bardos: el jugador, situado a la izquierda, y su rival, a la derecha. En la parte inferior aparece la mano del jugador, desde la cual se seleccionan las cartas que compondrán la jugada.

Sobre cada personaje se encuentran las mesas de juego, representadas mediante bocadillos, que contienen las cartas permanentes que conforman la historia en construcción. En

el centro de la pantalla se sitúa la audiencia, cuyos miembros muestran tanto su nivel de atención como su emoción actual.

Todos estos elementos y su funcionamiento se describen con mayor detalle en los apartados siguientes.

5.2.1. Desarrollo de un turno

El flujo del juego se desarrolla por turnos alternos, donde cada jugador toma decisiones basadas en la información disponible. La secuencia típica de una ronda es:

1. Robar cartas hasta tener cinco en la mano.
2. Seleccionar una secuencia válida de cartas según sus rimas.
3. Resolver los efectos asociados a las cartas jugadas.
4. Pasar el turno al oponente.

5.2.2. Fin de la partida

El ciclo continúa hasta que ambos jugadores han completado siete turnos. En determinadas tabernas pueden aparecer turnos con efectos globales, como la duplicación de la puntuación o la modificación de las reglas de atención.

Gana el jugador que, al finalizar la partida, posea la mayoría de los miembros de la audiencia a su favor y una historia completa. Se considera historia completa aquella que contiene inicio, nudo y desenlace, además de al menos un personaje y un pensamiento.

Si ninguno de los dos jugadores ha completado la historia, gana el que tenga mayoría en la audiencia.

5.2.3. Tipos de cartas

Existen dos tipos principales de cartas: **permanentes e instantáneas**.

Las cartas permanentes permanecen en la mesa tras su uso. Se clasifican en:

- **Historia:** representan las partes narrativas (inicio, nudo y desenlace). Solo puede haber una de cada tipo, y deben jugarse en orden.
- **Personaje:** representan los personajes involucrados en la historia. No hay límite en su número.
- **Pensamiento:** expresan los temas o mensajes subyacentes de la narración. No hay límite en su número.

Las cartas instantáneas se descartan tras su uso. Se dividen en:

- **Melodía:** representan los elementos musicales de la narración.
- **Dicción:** representan los recursos discursivos empleados.
- **Espectáculo:** representan los recursos escénicos o visuales.

Como se muestra en la Figura 5.2, cada carta recoge sus propiedades principales:

1. Nombre.
2. Tipo.
3. Valor de atención (puntos que pueden asignarse a miembros de la audiencia).
4. Rimas (anterior y posterior).
5. Emociones (antigua y nueva).
6. Texto descriptivo (efecto de la carta).
7. Texto de ambientación (sin impacto en la mecánica).
8. Imagen asociada.



Figura 5.2: Ejemplo de carta de *Bardtastic*. Cada carta contiene sus propiedades principales: tipo, atención, rimas y emociones. Creación propia.

5.2.3.1. Rimas

Para jugar una carta, debe rimar con la anterior en la secuencia activa. Las rimas posibles son A, B, C, X (sin rima) y “-” (rima libre). Cada carta posee una rima anterior y una posterior; la carta jugada debe coincidir en su rima anterior con la rima posterior de la carta precedente. Algunas cartas no pueden iniciar una secuencia y requieren ser enlazadas mediante rima.

5.2.3.2. Audiencia

La audiencia está compuesta por cinco miembros, cada uno con una puntuación de atención, representada por un entero en $[-3, 3]$. Un valor de -3 indica máxima atención al rival, mientras que 3 indica atención total hacia el jugador. Se considera que un miembro de la audiencia está de parte del jugador si su atención es > 0 y del rival si es < 0 . Una puntuación de 0 representa un miembro que no está de parte de ninguno.

De esta forma, por ejemplo, si los miembros de la audiencia tienen unas puntuaciones de atención de $1, 0, 3, -1$ y -2 , tendremos a dos miembros de nuestra parte, el rival tendrá otros dos, y habría un miembro que no está de parte de ninguno.

Cada miembro muestra además una emoción, que puede cambiar durante la partida.

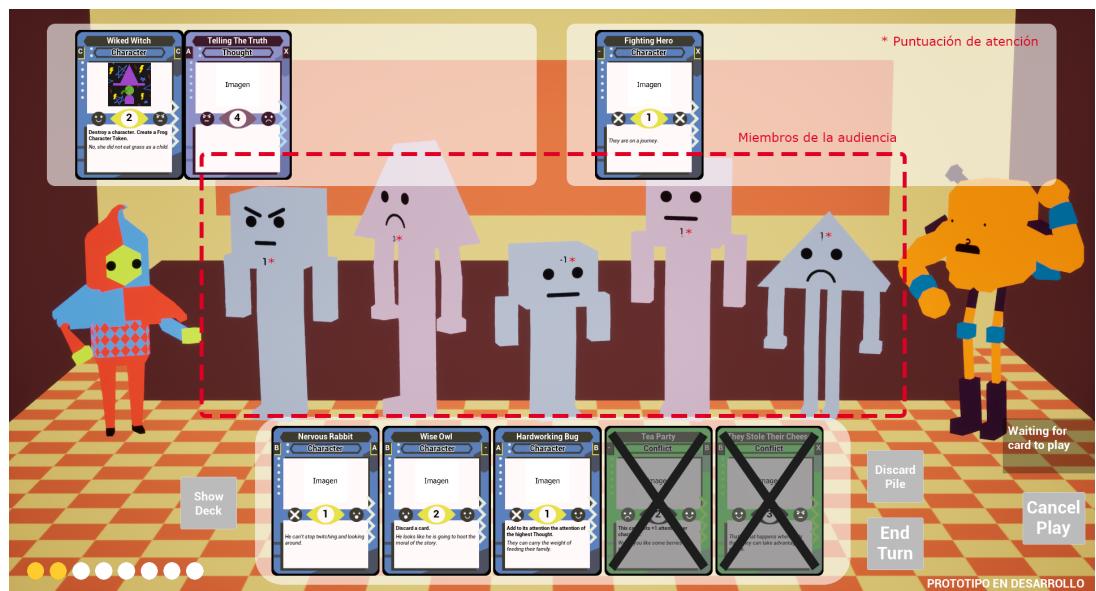


Figura 5.3: Ejemplo de audiencia en *Bardtastic*. Cada miembro tiene su puntuación de atención y muestra una emoción. Creación propia.

5.2.3.3. Emociones

Las cartas poseen una emoción antigua y una nueva, entre las siguientes: feliz, triste, enfadado o sorprendido. Si un miembro de la audiencia presenta la emoción antigua de la carta jugada y se le aplica atención, se obtiene un punto adicional de atención. Por ejemplo, si una carta con emoción antigua "triste" aplica atención a un espectador "triste" con valor -3 , este pasará a -1 . Si el espectador queda del lado del jugador tras el cambio, su emoción se actualiza a la nueva de la carta. Algunas cartas pueden afectar emociones globalmente o interactuar con ellas de manera indirecta.

5.3. Clasificación teórica del juego

Desde una perspectiva formal, los enfrentamientos en *Bardtastic* pueden modelarse como un juego dinámico de dos jugadores con información incompleta:

- Es **dinámico** porque las decisiones se toman de forma secuencial y los estados del juego dependen de las acciones previas de ambos jugadores.
- Es de **información incompleta** porque cada jugador desconoce las cartas que el oponente tiene en mano o las que quedan en su mazo. Además, el robo de cartas introduce incertidumbre.

El juego presenta **aleatoriedad limitada**, derivada del orden de las cartas y de ciertos efectos de azar, lo que incrementa su complejidad estratégica. Dado que los jugadores compiten por un recurso común (la atención de la audiencia), puede considerarse un **juego de suma cero**: la ganancia de atención de un jugador implica la pérdida equivalente del otro. Sin embargo, a diferencia de los juegos deterministas como el ajedrez o el tres en raya, *Bardtastic* incorpora incertidumbre y aleatoriedad, lo que condiciona la planificación óptima y justifica el uso de un agente basado en búsqueda y evaluación heurística.

5.4. Agente desarrollado: El rival

El agente de *Bardtastic* representa al oponente controlado por la máquina durante los enfrentamientos. Su objetivo de diseño no es la perfección algorítmica, sino emular a un bardo rival creíble que toma decisiones estratégicas y racionales coherentes con las reglas del juego.

Objetivo y alcance

En cada turno, el agente decide qué secuencia de cartas jugar en función del estado observable (mano, cartas en mesa, audiencia y emociones), sujeto a las mismas limitaciones de información que un jugador humano. Su ámbito se restringe al enfrentamiento; no gestiona la progresión global de la *run*.

Modelo deliberativo (visión conceptual)

El agente sigue un esquema deliberativo de horizonte corto: percibir el estado, generar acciones candidatas (secuencias válidas por rima y tipo), evaluarlas con una heurística y ejecutar la mejor. La racionalidad es limitada en el sentido de Simon [42]: opera con incertidumbre y recursos acotados para mantener flujo de juego y reto significativo [13, 40].

5.4.1. Función de utilidad (criterios de valoración)

La valoración de una secuencia candidata combina atención inmediata, progreso narrativo y calidad de mano futura. Formalmente:

$$U = w_1 \cdot \text{Atención} + w_2 \cdot \text{Longitud} + w_3 \cdot \text{ProgresoHistoria} + w_4 \cdot \text{Ciclo}, \quad (5.1)$$

donde:

- **Atención:** ganancia neta sobre la audiencia.
- **Longitud:** número de cartas jugadas (aprovechamiento del turno).

- **ProgresoHistoria:** avance hacia historia completa (inicio, nudo, desenlace, personaje y pensamiento).
- **Ciclo:** capacidad de robar/descartar para mejorar opciones futuras.

Los pesos w_i pueden interpretarse como la importancia relativa de cada criterio en la toma de decisiones del agente. Estos se fijan en el prototipo para favorecer secuencias estructuralmente valiosas sin descuidar la eficacia inmediata.

5.4.2. Ámbitos de decisión

El comportamiento se articula en tres ámbitos: (i) **selección de jugada** (secuencia de cartas), (ii) **asignación de atención** a espectadores cuando procede, y (iii) **decisiones auxiliares** (robo y descarte) basadas en utilidad esperada por carta. Los algoritmos concretos y su materialización software se describen en la sección 9.3.

5.4.2.1. Selección de jugada

La decisión principal del agente consiste en determinar qué secuencia de cartas jugar durante su turno. Para ello, se realiza una búsqueda en profundidad sobre el conjunto de combinaciones posibles generadas a partir de las cartas en mano y las reglas de encadenamiento (rima y tipo). Cada secuencia candidata se evalúa mediante la función de utilidad descrita en la sección 5.4.1, que pondera factores como la atención obtenida, la longitud de la jugada, el progreso narrativo y el potencial de ciclo.

El agente selecciona la secuencia con la puntuación heurística más alta, es decir, aquella que ofrece el mejor equilibrio entre ganancia inmediata y construcción narrativa a medio plazo. Este enfoque no garantiza la solución óptima global, pero proporciona una **racionalidad limitada** adecuada para un entorno interactivo: mantiene el desafío del jugador y asegura tiempos de respuesta compatibles con el ritmo de juego.

5.4.2.2. Asignación de atención

Tras elegir la jugada, el agente decide a qué miembro de la audiencia dará cada punto de atención obtenido.

Para cada espectador posible, se simula el efecto de la acción teniendo en cuenta la emoción asociada a la carta actual y se obtiene un nuevo nivel de atención simulado.

La elección se realiza con una heurística por etapas:

1. se prioriza al espectador cuyo nivel simulado queda más cerca del objetivo del bando propio (+1 si el agente actúa como jugador 1, -1 si lo hace como jugador 2).
2. en caso de empate, se prefiere a quien *cruza* al lado del agente (de atención no favorable a favorable)
3. si persiste el empate, se elige el cambio de mayor magnitud en la
4. finalmente, se deshacen empates atendiendo al nivel simulado más alineado con el bando del agente.

Este criterio local garantiza una decisión coherente con el objetivo táctico del turno: *acercar* a un espectador concreto hacia la zona favorable y, cuando es posible, provocar el cruce a nuestro lado, aprovechando el efecto emocional de la carta actual.

5.4.2.3. Decisiones auxiliares

Además de la elección de jugada y de la asignación de atención, el agente debe resolver decisiones complementarias que afectan a su estado futuro: robar o descartar cartas. Estas decisiones se abordan mediante una valoración heurística que estima la utilidad potencial de cada carta en el contexto actual, ponderando distintos criterios.

Selección de carta a robar.

Cuando el agente puede elegir entre varias cartas para incorporar a su mano, se calcula una puntuación para cada opción combinando tres factores:

- **Necesidad narrativa:** prioridad a las cartas que permiten completar una historia coherente (por ejemplo, si falta un *conflicto* o un *desenlace*).
- **Atención potencial:** incremento esperado de atención del público al jugar la carta, considerando sus efectos.
- **Capacidad de ciclado:** utilidad para renovar la mano o acceder a combinaciones futuras.

Las cartas se ordenan según esta puntuación compuesta (primero por necesidad narrativa, después por atención y, en último lugar, por capacidad de ciclo) y el agente selecciona la mejor. De esta forma, se asegura de avanzar hacia la condición de victoria, ya que necesitamos tener una historia completa para ganar (premia las cartas que le faltan y el ciclo, pues es una posibilidad de obtener una carta necesaria) y tener mayoría de audiencia (evalúa la atención potencial).

Selección de carta a descartar.

De modo análogo, al descartar cartas se invierte el criterio: se prioriza eliminar aquellas con menor contribución a la estrategia global. Se consideran principalmente dos aspectos:

- **Redundancia:** cartas permanentes cuyo tipo ya está representado en la historia o sobre la mesa, y por tanto aportan poco valor adicional.
- **Baja atención:** cartas que generan un efecto de atención reducido o que interrumpen la progresión temática.

El agente descarta la carta con peor balance entre estos factores. Da prioridad a la redundancia (si ya tiene de ese tipo, la puede descartar sin problema) y luego a la atención.

En conjunto, estas decisiones auxiliares refuerzan la coherencia interna del comportamiento del agente: su política de robo maximiza el potencial de progreso a medio plazo, mientras que su política de descarte minimiza redundancias y mantiene una mano eficiente. El resultado es un oponente que, aunque no persigue la optimización perfecta, exhibe una racionalidad perceptible y mantiene el flujo del juego, ajustándose al objetivo de ofrecer un desafío creíble y consistente con la ambientación de *Bardtastic*.

En los siguientes capítulos se detalla cómo este modelo deliberativo se implementa en el sistema software del juego, traduciendo los conceptos teóricos de utilidad y racionalidad limitada en un flujo de decisión ejecutable en tiempo real (véase la sección 9.3).

5.4.3. Complejidad computacional del agente

El agente rival emplea una búsqueda en profundidad (*Depth-First Search*, DFS) para explorar las secuencias jugables de cartas en cada turno. La elección de este enfoque está justificada por el tamaño acotado del espacio de búsqueda: dado un máximo de cinco cartas en mano, y considerando que el orden importa y no se permite la repetición, el número máximo de secuencias posibles corresponde a las variaciones sin repetición de cinco elementos,

$$V(5,5) = \frac{5!}{(5-5)!} = 5! = 120.$$

Este límite superior garantiza que incluso en el peor caso la exploración exhaustiva es perfectamente abordable en tiempo real. Además, el propio sistema de rimas y tipos de carta introduce una poda natural: en cuanto una secuencia candidata deja de ser extensible según las reglas del juego, la rama se descarta inmediatamente sin necesidad de continuar la exploración.

En cuanto a las decisiones auxiliares, su complejidad es también reducida y estable. El algoritmo de asignación de atención debe evaluar únicamente a los cinco miembros de la audiencia, por lo que su coste es constante,

$$O(1) \text{ (cinco evaluaciones).}$$

Del mismo modo, los procesos de selección de carta a robar o descartar analizan únicamente las opciones disponibles en ese momento. Las manos están limitadas a un máximo de cinco cartas y los efectos que permiten elegir entre varias opciones rara vez superan las diez alternativas. Incluso considerando el tamaño máximo de un mazo (30 cartas), el análisis sigue siendo lineal en el número de candidatos y completamente compatible con los requisitos de interacción en tiempo real.

En conjunto, tanto la exploración deliberativa de secuencias como las decisiones auxiliares operan dentro de un rango de complejidad muy reducido y fuertemente acotado. Esto permite al agente responder de forma inmediata durante el transcurso de la partida sin comprometer el ritmo del juego ni la fluidez del enfrentamiento.

Capítulo 6: Estado del arte

El presente capítulo examina el panorama actual de los videojuegos que comparten fundamentos conceptuales o mecánicos con *Bardtastic*. El objetivo es identificar los títulos más relevantes dentro del género *deckbuilder* y de los juegos con mecánicas narrativas, analizar sus aportaciones y limitaciones, y situar a *Bardtastic* en relación con ellos.

El análisis se estructura en dos apartados complementarios. En primer lugar, se estudia el contexto de diseño y la evolución del género *roguelike deckbuilder*, comparando *Bardtastic* con obras representativas como *Slay the Spire*, *Balatro* y *Storyteller*. En segundo lugar, se examinan los referentes artísticos y temáticos que inspiran su identidad visual y tono humorístico.

6.1. Diseño y género

6.1.1. Origen y evolución del *deckbuilding roguelike*

El concepto de *deckbuilding* (la construcción de un mazo de cartas como núcleo de la experiencia jugable) tiene su origen en los juegos de mesa. El punto de partida más reconocido es *Dominion* (Vaccarino, 2008), que introdujo la idea de que los jugadores comienzan con un conjunto de cartas básicas y, durante la partida, adquieren nuevas cartas que definen su estrategia [50]. Esta mecánica trasladó la gestión y optimización de recursos al nivel de la propia estructura del mazo, generando una capa de planificación y rejugabilidad inédita en los juegos de cartas tradicionales.

Posteriormente, títulos como *Ascension* (Gary Games, 2010) [19] consolidaron y popularizaron el género, incorporando variaciones en la dinámica de adquisición y uso de cartas. A partir de ahí, surgieron múltiples reinterpretaciones que ampliaron el concepto, como *Quarriors!* (Elliott y Kovalic, 2011) [14] o *Marvel Dice Masters* (Elliott y Wilson, 2014) [15], donde la construcción se centra en conjuntos de dados o fichas en lugar de cartas.

Por su parte, el término *roguelike* proviene del clásico *Rogue* (Toy, Wichman y Arnold, 1980) [48], caracterizado por la generación procedural de niveles, la *permadeath* (al morir en una partida, se pierde todo el progreso) y la estructura de partidas autoconclusivas. Esta filosofía de diseño, basada en la repetición significativa y la toma de decisiones bajo incertidumbre, fue heredada por títulos contemporáneos como *The Binding of Isaac* (McMillen, 2011) [31] o *Hades* (Supergiant Games, 2020) [17], que consolidaron el modelo de progresión mediante *runs* independientes y mejora acumulativa del jugador.

La convergencia entre ambos géneros culminó con *Slay the Spire* (MegaCrit, 2019) [32], que estableció un nuevo estándar al combinar la construcción progresiva de mazos con la estructura de juego *roguelike*. Este título demostró que las decisiones estratégicas propias del *deckbuilding* podían integrarse eficazmente en un ciclo de exploración y riesgo característico de los *roguelikes*, generando un alto valor de rejugabilidad y profundidad táctica.

En años recientes, el género ha continuado diversificándose. Obras como *Balatro* (LocalThunk, 2024) [30] o *Dungeon Clawler* (Stray Fawn Studio, 2023) [49] han reinterpretado la fórmula introduciendo estéticas más ligeras, sistemas de puntuación o capas narrativas simplificadas.

En este contexto, *Bardtastic* se presenta como una evolución natural del género, que introduce a su vez un enfoque innovador: mantiene la estructura estratégica y la progresión

6 Estado del arte

propias del *deckbuilder roguelike*, pero desplaza el foco hacia la creatividad narrativa, el humor y la riqueza temática.

6.1.2. *Slay the Spire*



Figura 6.1: Captura de pantalla de *Slay the Spire* [32].

Slay the Spire es considerado el referente contemporáneo del género *roguelike deckbuilder*. Su estructura combina la progresión a través de combates por turnos con la construcción dinámica de un mazo que evoluciona a medida que el jugador avanza. La rejugabilidad proviene de la aleatoriedad en las rutas, enemigos y cartas disponibles, lo que exige una gestión estratégica del riesgo y de los recursos por parte del jugador [32].

Relación con *Bardtastic*. Ambos títulos comparten la naturaleza *roguelike deckbuilder* y un núcleo de decisión basado en la optimización del mazo. Además, ambos presentan la estructura de un viaje en el que el jugador puede modificar la ruta a seguir, equilibrando así la relación entre riesgo y recompensa. Sin embargo, mientras que *Slay the Spire* traduce cada carta en acciones de combate, *Bardtastic* las convierte en fragmentos narrativos y recursos expresivos. Por otra parte, la carga narrativa en *Bardtastic* no se limita a los enfrentamientos principales, sino que se expande a través de una trama más amplia, poblada de personajes y situaciones que evolucionan a lo largo de la partida. En contraste, la propuesta narrativa de *Slay the Spire* resulta más lineal y funcional, centrada únicamente en el ascenso a una torre habitada por monstruos y cultistas.

Limitaciones detectadas.

- Ausencia de un componente narrativo emergente.
- Repetitividad temática: las acciones se articulan casi exclusivamente en torno al combate.

- Falta de un tono humorístico o identitario que trascienda la fantasía genérica.

6.1.3. *Balatro*



Figura 6.2: Captura de pantalla de *Balatro* [30].

Balatro expande el concepto de *deckbuilder* aplicándolo a las combinaciones del póker, mediante un sistema basado en la puntuación y en la optimización de sinergias entre cartas. Su éxito radica en la accesibilidad, el ritmo ágil de las partidas y el carácter adictivo derivado de la mejora incremental del mazo [30].

Relación con *Bardtastic*. Ambos títulos comparten un enfoque centrado en el uso expresivo de las cartas como un sistema de reglas autosuficiente, así como una estética desenfadada que atenúa la carga estratégica. No obstante, *Balatro* se orienta casi por completo hacia la optimización numérica, mientras que *Bardtastic* traslada esa estructura al terreno de la interpretación, la narrativa y la creatividad. Por otra parte, *Balatro* carece de cualquier tipo de narrativa o representación diegética del jugador, mientras que *Bardtastic* incorpora una historia y un elenco de personajes que contextualizan las acciones y refuerzan la dimensión expresiva de la experiencia.

Limitaciones detectadas.

- Ausencia de progresión narrativa o contexto temático.
- Falta de interacción directa con oponentes.
- Enfoque puramente abstracto y numérico, sin dimensión narrativa o temática.

6.1.4. Storyteller



Figura 6.3: Captura de pantalla de *Storyteller* [8].

Storyteller propone un enfoque radicalmente distinto: el jugador compone historias a partir de módulos predefinidos (personajes, escenarios y acciones) con el objetivo de cumplir determinadas condiciones narrativas [8]. En este caso, no se trata de un *roguelike deckbuilder*, sino de un juego de puzzles cuyo valor reside en el uso de la narrativa como mecánica central, invitando a la experimentación dentro de un sistema cerrado de combinaciones posibles.

Relación con *Bardtastic*. Ambos juegos sitúan la narración en el núcleo de la experiencia jugable. Sin embargo, mientras *Storyteller* se estructura como una sucesión de desafíos con soluciones predefinidas, *Bardtastic* propone una narrativa competitiva y emergente, donde la historia surge de la combinación libre de cartas y de la interacción entre los jugadores.

Limitaciones detectadas.

- Escasa libertad creativa más allá de las soluciones previstas.
- Ausencia de una dimensión estratégica o de construcción de mazos.
- Falta de un sistema competitivo o de valoración del relato.

6.2. Síntesis comparativa

La Tabla 6.1 resume las principales características de los títulos analizados en relación con *Bardtastic*.

Característica	Slay the Spire	Balatro	Storyteller	Bardtastic
<i>Deckbuilding</i>	Sí	Sí	No	Sí
<i>Mecánicas Narrativas</i>	No	No	Parcial	Sí
<i>Competitividad directa</i>	No	No	No	Sí
<i>Tono cómico</i>	No	Parcial	Sí	Sí
<i>Creatividad del jugador</i>	Media	Baja	Alta	Alta
<i>Narrativa global</i>	Parcial	No	No	Sí

Tabla 6.1: Comparativa de características entre obras de referencia y *Bardtastic*.

6.2.1. Conclusiones

El análisis evidencia que los títulos existentes tienden a concentrarse en uno de los tres ejes principales: la estrategia del mazo (*Slay the Spire*, *Balatro*) o la construcción narrativa (*Storyteller*). Ninguno integra ambos de manera orgánica ni introduce una dimensión competitiva centrada en la interpretación y el humor.

Bardtastic se diferencia al situar la creación narrativa como núcleo mecánico dentro de una estructura *deckbuilder roguelike*, dotando a cada carta de una función expresiva y temática. El resultado es un sistema que combina estrategia y creatividad, invitando al jugador no solo a optimizar su mazo, sino a construir relatos coherentes, ingeniosos y cómicos en un contexto competitivo.

Esta síntesis entre planificación estratégica, libertad narrativa y tono humorístico posiciona a *Bardtastic* como una propuesta singular dentro del panorama actual, ampliando el potencial expresivo del *deckbuilding* más allá de la mera optimización mecánica.

6.3. Artístico y temático

El apartado artístico de *Bardtastic* se nutre de referentes que combinan fantasía, humor y absurdo desde una perspectiva visual. Títulos como *Munchkin* (Steve Jackson Games, 2001) [28] y series como *Adventure Time* (Pendleton Ward, 2010) [53] constituyen influencias directas en el tono y la dirección artística del proyecto. Ambos ejemplos se caracterizan por una estética caricaturesca y colorida que, lejos de trivializar la experiencia, refuerza la dimensión cómica y accesible del universo que representan.

Otra de las principales referencias estéticas de *Bardtastic* es la corriente arquitectónica y decorativa desarrollada por el Grupo Memphis. Este movimiento se distingue por el uso de formas geométricas audaces, colores vibrantes y patrones de alto contraste, en una búsqueda deliberada de ruptura con la sobriedad del diseño modernista [44]. En el contexto del juego, se busca integrar esta filosofía visual con los motivos de la fantasía medieval, generando una identidad nueva y reconocible que combine lo lúdico con lo excéntrico.

Esta elección estética contrasta con el tono sombrío y serio predominante en muchos *deckbuilders roguelike* contemporáneos, proponiendo en su lugar una atmósfera más ligera y desenfadada. La fantasía de *Bardtastic* no aspira a la épica heroica oscura, sino a la improvisación, el ingenio y la exageración, en coherencia con su temática de bardo y su énfasis en la creatividad narrativa.

El resultado es una identidad visual y tonal que refuerza la propuesta mecánica del título: mientras el jugador construye historias, el mundo que las enmarca mantiene un tono humo-

6 Estado del arte

rístico, irreverente y expresivo, fomentando una experiencia coherente entre arte, jugabilidad y mensaje.

Capítulo 7: Planificación y diseño

*"To remember to only work on what is important, ask yourself this question:
Is making this game worth my time?"*

Jesse Schell, The Art of Game Design: A Book of Lenses (Lens #99 – The Lens of the Raven)

El desarrollo de *Bardtastic* se ha planificado siguiendo un enfoque estructurado de ingeniería del software, con el objetivo de garantizar una organización eficiente del trabajo, una adecuada gestión del tiempo y una distribución coherente de los recursos. A lo largo del proceso se ha definido un calendario de producción y un presupuesto estimado que reflejan tanto la carga de trabajo del estudiante como las necesidades de desarrollo del proyecto.

Antes de poder empezar a desarrollar, en la fase de pre-producción, se han definido el género, la temática y las principales referencias artísticas y mecánicas del proyecto, así como su identidad visual y tono narrativo. También se ha desarrollado y *testeado* un prototipo en papel que permitió explorar las dinámicas de juego y validar las ideas iniciales, aunque este prototipo se abordará con mayor detalle en apartados posteriores.

7.1. Diagrama de Gantt

La planificación del desarrollo se ha realizado en colaboración con una mentora del programa *Power Up Plus* [37], que ha acompañado el proceso de producción desde el mes de julio. El plan se ha estructurado en fases orientadas a la creación de un primer prototipo, con especial atención a la organización iterativa del trabajo y a la combinación equilibrada de tareas de diseño, implementación, pruebas y documentación.

Aunque el diagrama de planificación solo abarca hasta el prototipo previsto para noviembre, el objetivo final de las mentorías es finalizar diciembre con una *Vertical Slice* que permita presentar el juego ante posibles *publishers* y despertar el interés del público. Si bien el desarrollo de este Trabajo Fin de Grado se centra únicamente en la producción del primer prototipo, algunos elementos de planificación, como el presupuesto, hacen referencia a la *Vertical Slice*, considerada la próxima gran *milestone* del proyecto.

Una *Vertical Slice* se define como una versión reducida y completamente funcional del videojuego que incluye todas las mecánicas, sistemas y elementos visuales esenciales, aunque en una escala limitada. Su propósito es demostrar la viabilidad técnica y artística del proyecto, así como establecer una base sólida para su desarrollo completo posterior.

El diagrama de *Gantt* de la Figura 7.1 muestra la distribución temporal de las principales tareas del proyecto, abarcando el periodo comprendido entre julio y noviembre. Cada bloque temporal corresponde a un conjunto de hitos o entregables parciales, definidos para facilitar la supervisión del progreso y la evaluación de resultados.

7 Planificación y diseño

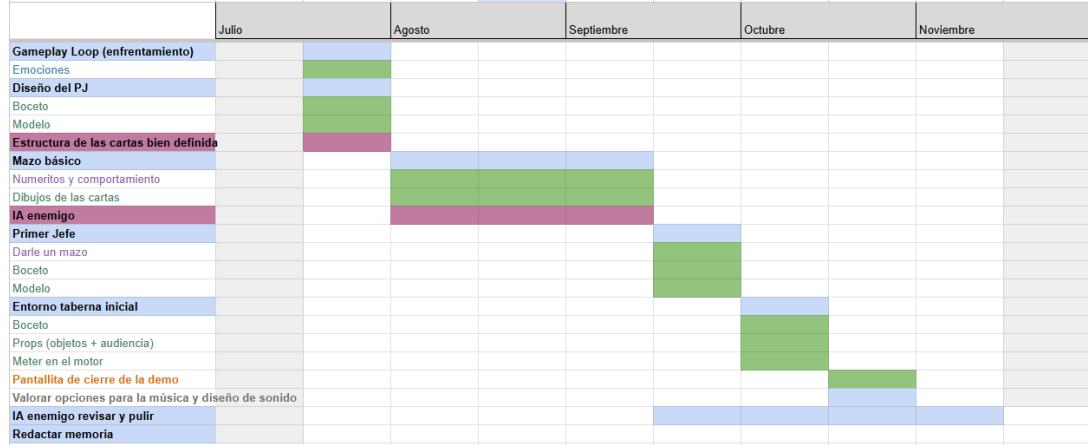


Figura 7.1: Diagrama de Gantt con el plan de producción desde julio a noviembre.

7.2. Presupuesto

Se presenta un presupuesto para el prototipo que abarca desde inicios de julio a noviembre.

El cálculo se basa en una estimación razonable de horas dedicadas a tareas de diseño, programación, arte y producción, junto con los gastos asociados a marketing, sonido y otros servicios necesarios para completar el prototipo vertical. La Figura ?? muestra el desglose presentado en la solicitud de financiación.

Tabla 7.1: Desglose del presupuesto estimado para el prototipo.

Concepto	Coste	Notas
Sueldo desarrollador (5 meses)	7.500 €	5 meses de programación, diseño y producción (1.500 €/mes netos aprox.)
Sueldo analistas (5 meses)	10.000 €	5 meses de análisis y consultas (2.000 €/mes netos aprox.)
Banda sonora	1.400 €	Encargo de 6~ temas completos (menú, enfrentamientos, etc.)
Efectos de sonido (SFX)	600 €	Encargo de diseño de SFX
Portada	200 €	Diseño de portada, <i>key art</i> y material promocional
Logo	100 €	Diseño vectorial, variantes y guía de estilo
Total	19.800 €	

7.3. Metodología de diseño y desarrollo

Para la planificación y organización del trabajo se ha seguido una metodología de carácter ágil, adaptada a las particularidades del desarrollo en solitario. Este enfoque permite mantener la flexibilidad necesaria para iterar sobre ideas de diseño, ajustar prioridades y

validar mecánicas a partir de prototipos jugables. Siguiendo la filosofía del modelo *iterativo-incremental* [29], el desarrollo se ha estructurado en ciclos cortos en los que cada versión del juego amplía y refina la anterior, reduciendo riesgos y permitiendo detectar problemas de diseño de forma temprana.

La gestión del trabajo diario se ha apoyado en principios de *Kanban* [6], utilizando un tablero visual que facilita el seguimiento del estado de cada tarea y ayuda a limitar el número de actividades en curso. Este método resulta especialmente adecuado en un proyecto individual, ya que proporciona claridad sobre el flujo de trabajo y permite reorganizar prioridades de manera dinámica.

Aunque no se aplica el marco de *Scrum* en su forma completa, sí se han integrado algunos de sus elementos más eficaces, como el mantenimiento de un *product backlog* priorizado, la planificación mediante iteraciones breves y la revisión periódica del progreso, en línea con las recomendaciones de la guía oficial [41]. Estas prácticas se han adaptado a un entorno sin equipo ni reuniones formales, pero permiten conservar la filosofía de mejora continua y entrega frecuente de resultados funcionales.

En conjunto, esta combinación ligera de enfoques ágiles ha permitido un proceso de desarrollo flexible, compatible con la naturaleza experimental del diseño de videojuegos y adecuado para la construcción progresiva de prototipos evaluables.

7.4. Análisis de requisitos

El presente apartado identifica y define los requisitos del sistema asociados a *Bardtastic*, considerando tanto su naturaleza como producto software jugable como el componente de inteligencia artificial encargado de controlar a los oponentes. El objetivo es especificar de forma clara las funcionalidades necesarias para garantizar una experiencia de juego coherente con la propuesta conceptual y plenamente operativa para la evaluación del agente inteligente.

Los requisitos se han clasificado en dos categorías principales:

- **Requisitos funcionales (RF)**: describen los comportamientos observables del sistema, condicionados por las reglas de juego y las interacciones del usuario y de la IA.
- **Requisitos no funcionales (RNF)**: definen atributos de calidad, restricciones tecnológicas y métricas asociadas al rendimiento, facilidad de uso o mantenimiento.

7.4.1. Requisitos funcionales (RF)

RF-1 El sistema debe permitir partidas con rondas y **turnos alternos** entre Jugador e IA.

RF-2 Al inicio de cada turno del jugador, el sistema debe **robar cartas** hasta tener 5 en mano (si es posible).

RF-3 El sistema debe **validar la secuencia** de cartas jugadas según la regla de rima (la rima posterior de la carta previa coincide con la rima anterior de la siguiente).

RF-4 El sistema debe **resolver los efectos** de las cartas jugadas sobre audiencia, mano, mazo y mesa.

RF-5 El sistema debe **gestionar los tipos de carta**: permanentes (Historia, Personaje, Pensamiento) e instantáneas (Melodía, Dicción, Espectáculo).

- RF-6** El sistema debe **actualizar la atención** de cada espectador en el rango $[-3, 3]$ de forma inmediata tras cada efecto.
- RF-7** El sistema debe **gestionar emociones** por espectador y aplicar las **transiciones** indicadas por las cartas.
- RF-8** El sistema debe permitir que la **IA** seleccione y juegue **secuencias válidas** de cartas siguiendo su estrategia.
- RF-9** El sistema debe **detectar el fin de partida** al completar 7 turnos por jugador o al cumplirse reglas globales de taberna.
- RF-10** El sistema debe **calcular el ganador** por mayoría de audiencia; en caso de empate, prevalece quien tenga **historia completa** (inicio, nudo, desenlace + $ge \geq 1$ personaje + $ge \geq 1$ pensamiento).

7.4.2. Requisitos no funcionales (RNF)

- RNF-1 Usabilidad:** interfaz clara que muestre mano, audiencia y estado narrativo (cartas permanentes en mesa).
- RNF-2 Rendimiento:** respuesta interactiva sin latencias perceptibles al jugar/validar/actualizar (p. ej., < 100 ms por acción típica).
- RNF-3 Portabilidad:** ejecutable en PC (Windows/Linux) con dependencias controladas.
- RNF-4 Mantenibilidad:** arquitectura modular (lógica de juego, IU, IA) para facilitar extensiones (nuevos rivales/cartas/reglas).
- RNF-5 Rejugabilidad:** variación entre *runs* (mazos, rutas, oponentes).
- RNF-6 Extensibilidad del agente:** parámetros y estrategias de IA configurables.

7.4.3. Modelo de casos de uso

El modelado de casos de uso se usa para **delimitar el sistema, describir el punto de vista del usuario y trazar requisitos a funcionalidades**. A continuación se describen los casos de uso principales (plantilla básica y extendida según Tema 2)

UC-01 — Iniciar turno del jugador

Actores	Jugador (iniciador)
Precond.	Es el turno del jugador; la partida está en curso.
Postcond.	Mano actualizada hasta 5 cartas; estado listo para selección de jugada.
Propósito	Preparar al jugador con cartas suficientes para decidir su jugada.
Flujo básico	1) El sistema roba cartas del mazo del jugador hasta 5 (RF-2). 2) Actualiza la interfaz con la nueva mano (RNF-1).
Excepciones	E1) Mazo agotado: no se alcanza 5; el sistema continúa con la mano disponible.
Requisitos	RF-2, RNF-1, RNF-2.

UC-02 — Seleccionar y jugar secuencia de cartas

Actores	Jugador (iniciador)
Precond.	UC-01 completado; hay cartas en mano.
Postcond.	Secuencia válida colocada; efectos aplicados; mano/mesa/mazo/audiencia actualizados.
Propósito	Permitir al jugador ejecutar una jugada conforme a las reglas de rima y efectos.
Flujo básico	<ul style="list-style-type: none"> 1) El jugador selecciona una secuencia en la mano. 2) El sistema valida rimas entre cartas adyacentes (RF-3). 3) El sistema resuelve efectos de cada carta (RF-4). 4) Actualiza atención y emociones por espectador (RF-6, RF-7). 5) Actualiza tipos permanentes en mesa e instantáneas al descarte (RF-5).
Cursos alternativos	A1) Validación de rima falla: se informa y se impide la jugada (RF-3, RNF-1). A2) Efecto apunta a espectador inexistente: se cancela ese sub-efecto.
Requisitos	RF-3, RF-4, RF-5, RF-6, RF-7, RNF-1, RNF-2.

UC-03 — Jugar turno de la IA

Actores	IA (iniciador implícito)
Precond.	Turno de la IA; estado consistente tras el turno del jugador.
Postcond.	Secuencia válida jugada por IA; estado actualizado.
Propósito	Permitir a la IA tomar decisiones y ejecutar una jugada válida.
Flujo básico	<ul style="list-style-type: none"> 1) La IA evalúa su mano y el estado de audiencia/mesa. 2) Selecciona una secuencia válida (RF-8, RF-3). 3) El sistema resuelve efectos y actualiza atención/emociones (RF-4, RF-6, RF-7).
Excepciones	E1) No hay secuencia válida: la IA pasa.
Requisitos	RF-8, RF-3, RF-4, RF-6, RF-7, RNF-2, RNF-6.

UC-04 — Actualizar atención de la audiencia

Actores	Sistema
Precond.	Se ha jugado una carta con efectos de atención.
Postcond.	Cada espectador refleja su nueva atención en $[-3, 3]$ y posible cambio de bando.
Propósito	Mantener la métrica central de éxito del enfrentamiento.
Flujo básico	<ul style="list-style-type: none"> 1) Calcular contribuciones de atención según efecto y emoción antigua (RF-4, RF-7).

	2) Saturar a $[-3, 3]$ y registrar el resultado (RF-6). RF-4, RF-6, RF-7, RNF-2.
--	---

UC-05 — Gestionar emociones del espectador

Actores	Sistema
Precond.	Se ha aplicado atención que pueda provocar transición emocional.
Postcond.	Emociones actualizadas (antigua → nueva) cuando proceda.
Propósito	Aplicar reglas de transición emocional ligadas a cartas.
Flujo básico	1) Detectar si el espectador queda del lado del jugador tras el cambio (RF-7). 2) Si procede, establecer emoción nueva indicada por la carta (RF-4). RF-7, RF-4.
Requisitos	

UC-06 — Gestionar tipos de carta

Actores	Sistema
Precond.	Se ha jugado una carta.
Postcond.	Permanentes permanecen en mesa; instantáneas al descarte.
Propósito	Mantener la consistencia del tablero según tipo de carta.
Flujo básico	1) Si la carta es Historia/Personaje/Pensamiento, ubicarla en mesa (RF-5). 2) Si es Melodía/Dicción/Espectáculo, enviar a descarte (RF-5). RF-5.
Requisitos	

UC-07 — Detectar fin de partida

Actores	Sistema
Precond.	Cada jugador ha jugado su turno en la ronda actual.
Postcond.	Si se cumplen condiciones de fin, se bloquea nueva ronda y se pasa a UC-08.
Propósito	Concluir el enfrentamiento cuando corresponda.
Flujo básico	1) Comprobar turnos acumulados por jugador (RF-9). 2) Comprobar reglas globales de taberna (si aplican) (RF-9). RF-9.
Requisitos	

UC-08 — Calcular y anunciar ganador

Actores	Sistema
Precond.	UC-07 ha determinado fin de partida.
Postcond.	Se determina ganador por mayoría de audiencia y completitud de historia.
Propósito	Determinar el resultado de la partida según las reglas.
Flujo básico	1) Contar espectadores a favor de cada bando (RF-6). 2) Si hay empate, comprobar historia completa (RF-10). 3) Mostrar resultado en interfaz (RNF-1).

Requisitos **RF-10, RF-6, RNF-1.**

7.4.4. Matriz de trazabilidad UC ↔ RF

RF	UC-01	UC-02	UC-03	UC-04	UC-05	UC-06	UC-07	UC-08
RF-1	X		X					
RF-2	X							
RF-3		X	X					
RF-4		X	X	X	X			
RF-5	X					X		
RF-6	X	X	X				X	
RF-7	X	X			X			
RF-8		X						
RF-9						X		
RF-10							X	

7.4.5. Criterios de verificación

Cada RF/RNF es verificable mediante pruebas funcionales (para RF) y métricas/inspecciones (para RNF), alineado con las propiedades de especificación (completa, consistente, verifiable, trazable).

7.5. Diseño del juego

El diseño de *Bardtastic* se ha desarrollado siguiendo una fase de preproducción centrada en validar las ideas principales antes de acometer su implementación en Unreal Engine. Durante esta etapa se elaboró un prototipo en papel que permitió explorar y depurar las dinámicas narrativas y competitivas del sistema de cartas. La transición de este prototipo analógico al prototipo digital descrito en el Capítulo 9 ha supuesto una primera materialización del *loop* jugable, con el alcance necesario para evaluar la viabilidad de las mecánicas básicas y del comportamiento del agente inteligente.

La propuesta de diseño se articula en torno a tres pilares de diseño que sirven como guía conceptual y criterio de validación durante el desarrollo. Estos pilares garantizan que las decisiones relacionadas con mecánicas, arte e interfaz mantienen una coherencia temática y una identidad clara del proyecto.

7.5.1. Pilares de diseño

Los tres pilares de diseño definidos para *Bardtastic* se inspiran en el marco conceptual planteado por Taylor [47], adaptado aquí al contexto específico de un *deckbuilder* con énfasis narrativo. Este enfoque invita a los diseñadores a concretar qué tipo de experiencia se desea provocar en el jugador, más allá de la noción genérica de “diversión”.

En este sentido, se ha considerado también la clasificación de las catorce formas de diversión propuesta por Garneau [18], que facilita un análisis más granular del tipo de satisfacción que el juego puede ofrecer. Ambas referencias han servido de guía para identificar los elementos que resultan más relevantes en la experiencia que *Bardtastic* busca construir:

creatividad en la construcción del mazo, toma de decisiones con impacto estratégico y un tono humorístico que acompañe a la fantasía de actuar como un bardo en un entorno teatral.

7.5.1.1. Creación

El núcleo de un *deckbuilder* reside en permitir que el jugador construya su propio enfoque estratégico. Cada decisión sobre qué cartas incluir, mejorar o descartar contribuye a la creación de su “voz” como bardo. Este proceso de construcción mecánica se complementa con la fantasía de “contar una historia”, reforzando la sensación de autoría y de creatividad emergente a lo largo de una partida.

7.5.1.2. Resolución de problemas

Los enfrentamientos plantean un reto táctico constante: la gestión de recursos (cartas, rimas válidas, emociones del público) obliga al jugador a analizar el estado de la partida y tomar decisiones que maximizan su impacto narrativo frente al rival. La inteligencia artificial contribuye además a evitar patrones triviales, incentivando la adaptación y la planificación del jugador.

7.5.1.3. Comedia

La ambientación ligera y humorística, la exageración teatral y la interacción con una audiencia caricaturesca buscan generar una experiencia divertida, accesible y expresiva. La comedia actúa como estabilizador del tono, incluso en situaciones competitivas, alineándose con la naturaleza desenfadada del proyecto y su aspiración a poder ser disfrutado por un público amplio.

Estos tres pilares actúan como referencia constante: cualquier mecánica o decisión visual debe servir al menos a uno de ellos para ser considerada coherente con el diseño global del juego.

7.5.2. Referencias de diseño

El diseño de *Bardtastic* se apoya en una combinación de referentes mecánicos, estilísticos y estructurales dentro del ámbito de los juegos de cartas y de los *roguelites*. A nivel mecánico, se toma como base la construcción de mazos inspirada en *Slay the Spire* [32], especialmente en lo relativo a la toma de decisiones tácticas y al acceso progresivo a cartas que modifican el estilo de juego. La creatividad sobre la composición de la historia en cada partida bebe de propuestas como *Storyteller* [8], que explora la recombinación narrativa a través de piezas modulares.

Asimismo, el juego incorpora elementos clásicos del género *deckbuilder* consolidados en *Ascension* [19] y *Magic: The Gathering* [3], aunque adaptándolos a un contexto competitivo asimétrico basado en la atención del público en lugar del daño directo. La progresión entre partidas se inspira en la estructura de *Slay the Spire* y en la tradición *roguelite* representada por *The Binding of Isaac* [31], particularmente en cómo la narrativa avanza mediante la aparición de nuevos jefes y contenidos desbloqueables.

Finalmente, se reconoce también la influencia de juegos humorísticos como *Munchkin* [28], que combina interacción directa entre jugadores con un tono desenfadado y paródico, reforzando así uno de los pilares del diseño de *Bardtastic*: la comedia.

7.5.3. Prototipo en papel

Antes de iniciar la implementación digital, se desarrolló un prototipo en papel con el objetivo de explorar las bases narrativas y mecánicas del sistema de juego. Esta metodología permitió iterar rápidamente sobre las reglas, validar hipótesis de diseño y descartar ideas que no contribuían a la experiencia deseada.



Figura 7.2: Prototipo a papel de *Bardtastic*. Creación propia.

7.5.3.1. Diseño narrativo inicial

El primer reto consistió en definir qué significa “contar una historia” a través de cartas. Para ello se llevó a cabo una revisión de referentes teóricos sobre estructura narrativa:

- **Aristóteles — *Poética*** [7]: identificación de los seis elementos fundamentales del drama (acción, personajes, pensamiento, expresión, música y espectáculo), adoptados como base para los tipos de carta.

- **Christopher Vogler — *The Writer's Journey* [51]**: refuerza la dimensión arquetípica del relato y la universalidad de ciertas estructuras que pueden activarse mediante mecánicas emergentes del mazo.
- **John Yorke — *Into the Woods* [58]**: ofrece una lectura moderna de la estructura clásica en tres actos, muy útil para organizar la secuencia narrativa de las cartas de Historia (inicio, nudo y desenlace).
- **Julian Woolford — *How Musicals Work and How to Write One* [55]**: proporciona un puente entre la teoría clásica y la puesta en escena contemporánea, reforzando la importancia del espectáculo en la recepción del público.
- **Monomito aplicado al viaje del héroe [12]**: ofrece una referencia estructural para la **narrativa macro** del juego (progresión entre tabernas, ascenso hacia un gran reto final), que se explora en la planificación futura del *roguelike*.

Esta revisión permitió trasladar principios narrativos clásicos al dominio mecánico de las cartas, manteniendo una coherencia temática entre la forma de jugar y la fantasía que se propone al jugador.

A partir de esta revisión, se definieron los seis tipos de cartas empleados en el prototipo, cada uno asociado a uno de los componentes dramáticos clásicos. Esta decisión vincula directamente la mecánica con la fantasía de juego: construir una historia frente a un público.

7.5.3.2. Atención como recurso central

En las primeras iteraciones se experimentó con diferentes representaciones de la respuesta del público: una puntuación global, medidores individuales por jugador o un sistema basado en espectadores múltiples. Finalmente, se optó por una audiencia compuesta por varios miembros con atención individual, al proporcionar:

- mayor expresividad estratégica,
- toma de decisiones más rica al elegir a quién influir,
- coherencia temática con el espectáculo teatral.

Así, la **atención** se erige como el recurso principal del enfrentamiento, sustituyendo a los tradicionales puntos de vida en juegos de cartas competitivos.

7.5.3.3. Construcción del mazo y ciclo de cartas

Definida la atención, se generaron dos primeros mazos con efectos básicos (robar, descartar, interactuar con cartas en mesa, etc.) y se realizaron sesiones de prueba con jugadores reales. Estas pruebas permitieron ajustar aspectos fundamentales del flujo de partida, como:

- establecer un número fijo de cartas en mano,
- robar al inicio del turno hasta recuperar dicho límite,
- mantener un descarte funcional que permita el ciclado del mazo.

Este diseño garantiza que las cartas de gestión de mano y mazo tengan relevancia táctica en la toma de decisiones.

7.5.3.4. Rimas como sistema de encadenado

Durante el proceso se descartó el uso de un sistema de recursos equivalente al “maná” de *Magic: The Gathering* debido a sus problemas de flujo y variabilidad no controlada [3]. En su lugar, se buscó un mecanismo temático y mecánicamente expresivo que regulara la capacidad de juego:

- el sistema de rimas define qué cartas pueden jugarse en secuencia,
- promueve la planificación sin bloquear en exceso al jugador,
- refuerza la fantasía de recitar una historia en verso.

El encadenado de rimas actúa, por tanto, como alternativa narrativa al consumo de recursos tradicional.

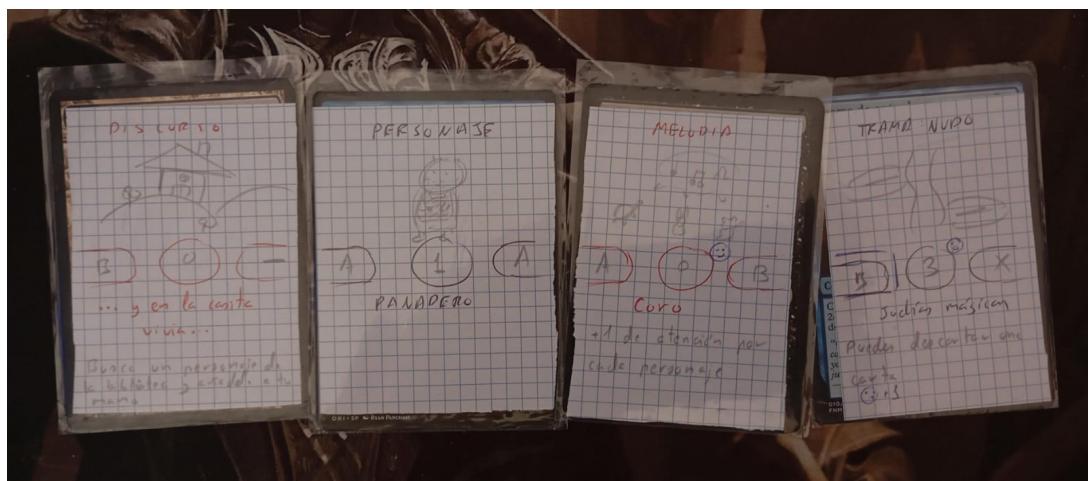


Figura 7.3: Jugada del prototipo a papel de *Bardtastic*. Creación propia.

7.5.3.5. Emociones de la audiencia

Tras varias iteraciones se identificó la necesidad de añadir una capa más de interacción, ya que influir únicamente en la atención individual resultaba tácticamente limitado. Así, se incorporó un sistema de **emociones** que modifica:

- la efectividad con la que una carta puede afectar al público,
- la lectura estratégica del tablero por parte del jugador,
- la expresividad temática y humorística del enfrentamiento.

Este añadido refuerza tanto la toma de decisiones como la coherencia con la premisa narrativa del juego.

7.5.3.6. Conclusión de la fase de prototipado

El prototipo en papel permitió alcanzar un conjunto de mecánicas cohesionado y alineado con los pilares de diseño del proyecto. Una vez validados:

- los tipos de carta y su fundamento narrativo,
- el flujo de robo, mano, mesa y descarte,
- la atención y emociones como recursos interactivos,
- el sistema de rimas como regulador jugable,

se establecieron las bases suficientes para iniciar la implementación digital del prototipo jugable en Unreal Engine.

7.6. Diagrama de clases

El diagrama de clases recoge únicamente los componentes C++ del proyecto, es decir, la lógica central del juego y de la IA. La capa de presentación y flujo visual (animaciones, UI y orquestación de acciones) se aborda principalmente con *Blueprints* y, por tanto, no se incluye aquí. Este enfoque permite analizar la *estructura de datos y responsabilidades* que sostienen el prototipo, manteniendo la separación de intereses entre reglas, decisión e interfaz (véase la relación con los RNF en la Sección 7.4.2).

7.6.1. Alcance y criterios de modelado

El modelo UML se ha construido con los siguientes criterios:

- **Foco en el dominio:** clases que representan cartas, mazo/descartes, audiencia y jugador controlado por lógica.
- **Decisión desacoplada:** la IA rival se modela como una *estrategia* independiente del jugador, con utilidades auxiliares para evaluar jugadas.
- **Mutaciones controladas:** las simulaciones para puntuar secuencias no alteran el estado real del juego; se emplean estructuras ligeras o copias de trabajo.

7.6 Diagrama de clases

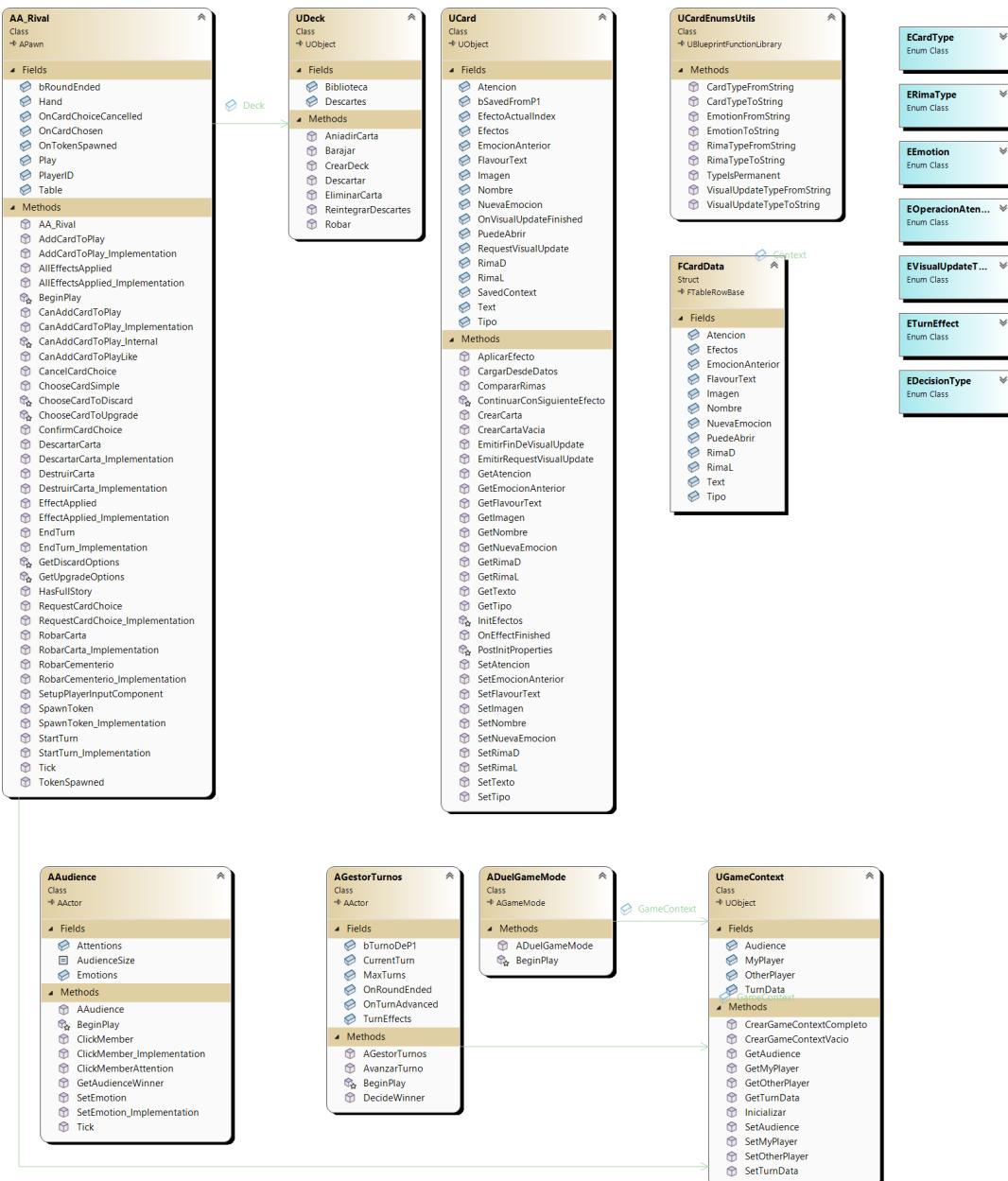


Figura 7.4: Diagrama de clases UML (bloque 1: dominio y contexto). Creación propia.

7 Planificación y diseño

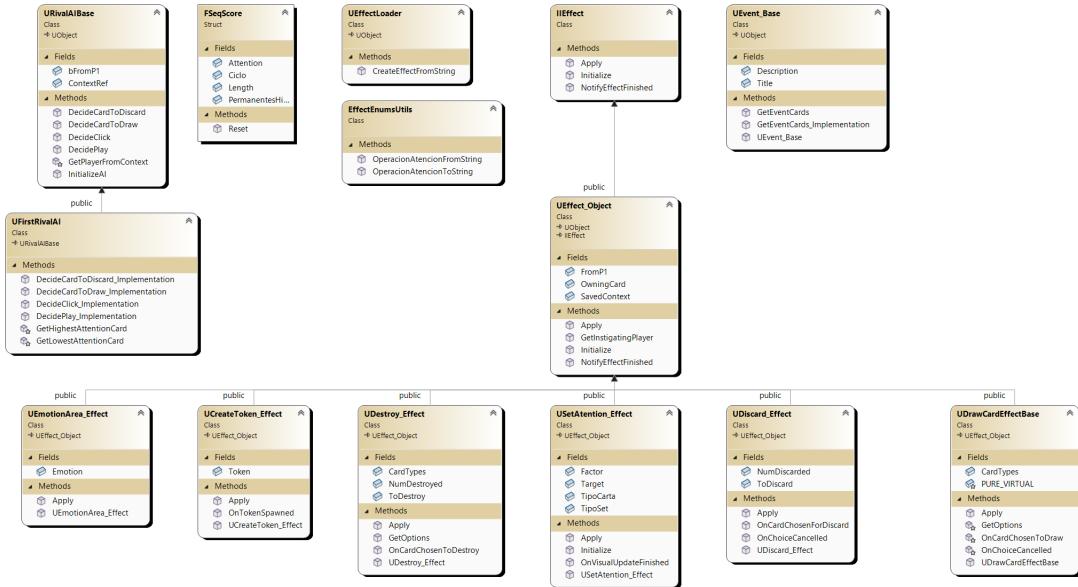


Figura 7.5: Diagrama de clases UML (bloque 2: IA y utilidades). Creación propia.

Capítulo 8: Herramientas

El desarrollo de *Bardtastic* se ha realizado mediante un conjunto de herramientas ampliamente utilizadas tanto en la industria como en el ámbito independiente del desarrollo de videojuegos. La combinación de estas tecnologías ha permitido abordar el proyecto desde una perspectiva integral que abarca programación, diseño visual, producción de contenidos y gestión del desarrollo.

8.1. Justificación de la elección tecnológica

La elección del motor de juego no responde únicamente a criterios de difusión industrial, sino también a la experiencia previa del desarrollador. Unreal Engine [16] ha sido seleccionado porque es la herramienta en la que el autor ya posee conocimientos previos, especialmente en el uso de *Blueprints*, lo que permite una curva de desarrollo más eficiente en fases iniciales. Los *Blueprints* constituyen el sistema de programación visual de Unreal: un lenguaje basado en nodos que permite crear lógica de juego, interfaces y comportamientos complejos sin necesidad de escribir código C++, facilitando el prototipado rápido y la iteración sobre mecánicas.

Aunque Godot [22] constituye una alternativa emergente y atractiva en el desarrollo independiente, particularmente por su ligereza, flexibilidad y licencia completamente abierta, presenta todavía limitaciones en cuanto a exportación nativa a plataformas de consola, lo cual condiciona potencialmente la escalabilidad comercial del proyecto. Unreal, en cambio, dispone de *toolchains* y soporte oficial para Xbox, PlayStation y Nintendo Switch, lo que refuerza la viabilidad a largo plazo de *Bardtastic*.

Los puntos clave de esta decisión son:

- **Prototipado rápido con Blueprints**, que permite experimentar con mecánicas, IA y UI sin coste elevado.
- **Posibilidad futura de publicar en consolas** sin rehacer el pipeline técnico ni migrar el proyecto.
- Convergencia entre industria AAA y estudios indie que adoptan Unreal como motor principal, lo que garantiza documentación, soporte y comunidad.

8.2. Prototipado y velocidad de iteración

El componente más ventajoso de Unreal para este proyecto son los *Blueprints*. Permiten:

- Probar mecánicas en cuestión de minutos.
- Reimplementar funciones y métodos para alterar comportamientos sin recompilar C++.
- Facilitar los momentos del desarrollo más centrados en la parte artística y el diseño de niveles.

En *Bardtastic*, las reglas del juego (validación de rimas, efectos de emociones y atención) requieren numerosos ajustes y testeo continuo. Realizar estos cambios únicamente en C++ ralentizaría el flujo de trabajo. La arquitectura híbrida permite así reservar el código nativo para lo que realmente lo requiere: el motor lógico y la IA.

En resumen:

- **BluePrints** → experiencia de usuario, UI, flujo visual del combate.
- **C++** → lógica del sistema, reglas de victoria y comportamiento inteligente.

Este patrón es considerado una buena práctica en Unreal tanto para equipos independientes como grandes producciones.

8.3. Pipeline de arte y contenido

El proceso de creación artística sigue un flujo iterativo orientado a garantizar la coherencia estética del juego y su adecuación al tono teatral y caricaturesco de la propuesta. Es importante destacar que **todo el arte desarrollado para el prototipo —modelos, texturas, animaciones, iconografía y materiales— es de elaboración propia**, producido específicamente para este Trabajo Fin de Grado.

Flujo de creación

El *pipeline* aplicado es el siguiente:

1. **Moodboard y bocetos**: exploración visual inicial para definir el estilo caricaturesco, la paleta cromática y las referencias teatrales del universo del juego.
2. **Modelado en Blender**: creación desde cero de personajes, objetos y elementos escénicos, con énfasis en formas simples y siluetas expresivas.
3. **Pintado en Substance Painter**: texturizado manual con un acabado *cartoon*, coherente con la dirección artística del proyecto.
4. **Rigging y animación en Blender**: elaboración de esqueletos, pesos y animaciones básicas, centradas en gestos exagerados propios de la interpretación bardesca.
5. **Integración en Unreal Engine**: importación de modelos y texturas, creación de materiales, configuración de *skeletons*, *montages* y controladores de animación.
6. **Asignación de comportamiento**: conexión del contenido artístico con la lógica de juego mediante C++ y *Blueprints*, integrando animaciones, efectos y respuestas a eventos del gameplay.

8.4. Diseño 2D y UI

Para el diseño visual de *Bardtastic* se utiliza Adobe Photoshop como herramienta principal de creación gráfica. En él se desarrollan tanto las cartas y sus ilustraciones como los iconos y botones necesarios para la interfaz del juego. Photoshop constituye además el entorno donde se componen los distintos elementos promocionales y de presentación del proyecto, como la

carátula del juego, materiales para tiendas digitales y posibles publicaciones en redes. Los recursos generados en esta fase se integran posteriormente en el sistema *UMG* de Unreal Engine, desde donde se implementa su comportamiento e interacción dentro del flujo de partida.

8.5. Audio

El apartado sonoro de *Bardtastic* se centra en dos elementos fundamentales: la música de fondo que acompaña a los enfrentamientos y los efectos asociados al uso de las cartas. Para la edición y optimización de estos audios se emplea Audacity, donde se realizan ajustes básicos como recortes, normalización y limpieza de ruido cuando resulta necesario. Una vez preparados, los recursos se integran en el motor de sonido de Unreal Engine, que se encarga de reproducirlos de forma contextual durante la partida y de proporcionar al jugador un apoyo auditivo coherente con el ritmo del combate y la construcción de su historia.

8.6. Herramientas de producción

Para gestionar la evolución del proyecto se integra:

- **Notion:** organización del desarrollo, diseño del juego y documentación.
- **Git:** control de versiones del código y activos con configuración específica para repositorios de Unreal.

La gestión con Notion permite aplicar incrementos iterativos de funcionalidad y contenido, priorizando siempre aquello que aporta valor jugable en cada momento.

8.7. Conclusiones

El conjunto de herramientas seleccionado está alineado con:

- Los **estándares de la industria** en motor y pipeline.
- La **realidad del desarrollo indie**, que exige flexibilidad y rapidez de iteración.
- La experiencia previa del desarrollador, lo que se traduce en mayor eficiencia.

La combinación de **C++** y **Blueprints** en Unreal representa un equilibrio idóneo entre rendimiento, mantenibilidad y velocidad de prototipado, factores decisivos en un juego experimental y basado en toma de decisiones estratégicas como *Bardtastic*.

Capítulo 9: Software desarrollado

En este capítulo se describe el *software* desarrollado como resultado del proyecto, detallando las características del prototipo implementado, su arquitectura interna y los componentes fundamentales que permiten el funcionamiento del juego. El objetivo es exponer una visión estructurada del sistema: cómo se organizan las mecánicas, cómo interactúan sus módulos y qué decisiones técnicas han permitido transformar el diseño conceptual en un producto jugable. No se incluye código fuente completo, salvo fragmentos o referencias cuando es necesario aclarar aspectos concretos del desarrollo.

9.1. Descripción general

El resultado del presente Trabajo de Fin de Grado es un prototipo funcional de *Bardtastic*, un videojuego *roguelike deckbuilder* centrado en el uso estratégico de cartas dentro de enfrentamientos narrativos.

El prototipo implementa el *core loop* del juego: el jugador disputa una sucesión de combates, modifica progresivamente su mazo y se enfrenta a un oponente controlado por inteligencia artificial. Las cartas jugadas producen efectos que alteran la atención del público y modifican el estado de la partida. Tras cada victoria, el sistema presenta un evento que permite mejorar el mazo antes del siguiente combate.

En cuanto al apartado visual, el prototipo incluye ya las bases estéticas de la versión final: personajes animados, público reactivo, cartas ilustradas y un entorno tridimensional que reproduce una taberna teatralizada. La interfaz de usuario es funcional aunque provisional, actuando como guía para la futura *Vertical Slice*.

Este prototipo demuestra que las mecánicas y dinámicas diseñadas son viables, jugables y compatibles con la arquitectura *software* propuesta.

9.2. Enfrentamientos

Los enfrentamientos se ejecutan dentro de un nivel de Unreal Engine que agrupa los *Actors* responsables de la lógica del combate:

- **Jugador:** gestiona el mazo, la mano, la mesa de juego y comunica los cambios al interfaz.
- **Rival (IA):** controla su propio mazo, genera decisiones y ejecuta jugadas de manera autónoma.
- **Gestor de turnos:** alterna el flujo, controla el final de partida y notifica el resultado.
- **Audiencia:** mantiene las puntuaciones de atención y emociones.

El GameMode crea e inicializa el GameContext que actúa como fuente accesible y centralizada de información global. Este patrón se justifica porque el *Game Mode* es un *Singleton* natural del motor y su acceso está garantizado durante toda la partida.

9.2.1. Interfaz de usuario

La interfaz en *Bardtastic* se compone de varios *Widgets* de Unreal (UMG) que presentan, de forma clara, el estado de la partida: la mano del jugador, las cartas permanentes en mesa de ambos bandos y la información de turnos.

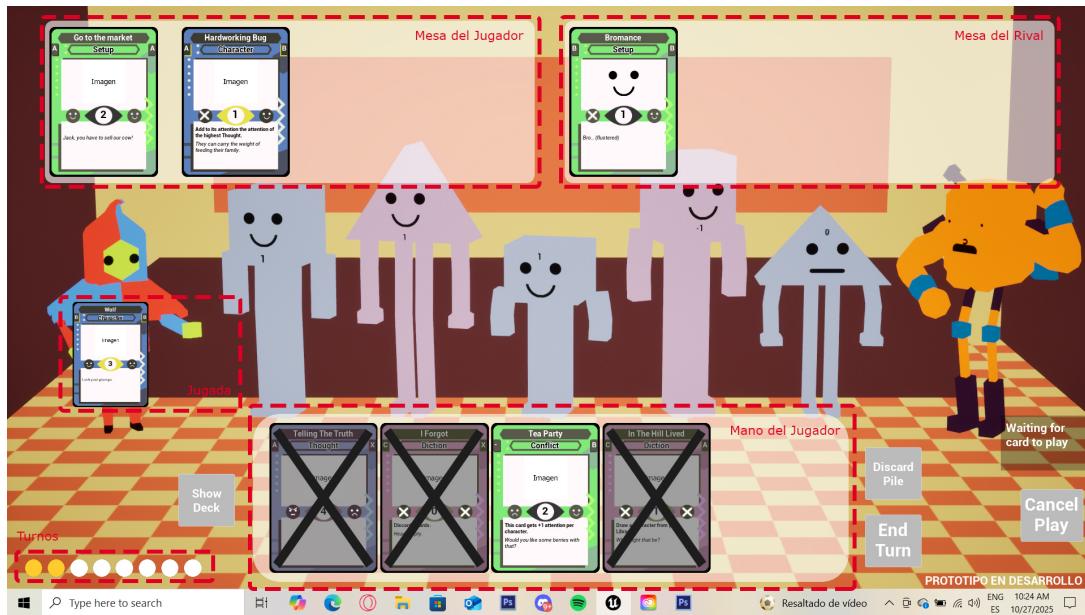


Figura 9.1: Interfaz de los enfrentamientos en *Bardtastic*.

En la Figura 9.1 se aprecian los elementos principales de la pantalla:

- **Mesa del jugador:** zona donde permanecen las cartas de tipo Historia, Personaje y Pensamiento del jugador.
- **Mesa del rival:** zona equivalente para las cartas permanentes del oponente controlado por la IA.
- **Mano del jugador:** conjunto de cartas disponibles para jugar durante el turno actual.
- **Indicador de turnos:** muestra cuántos turnos se han jugado y cuántos restan hasta el final del enfrentamiento.

Al concluir el combate, la interfaz despliega un *Widget* de fin de partida con el resultado y un botón para continuar con el flujo del juego.

9 Software desarrollado

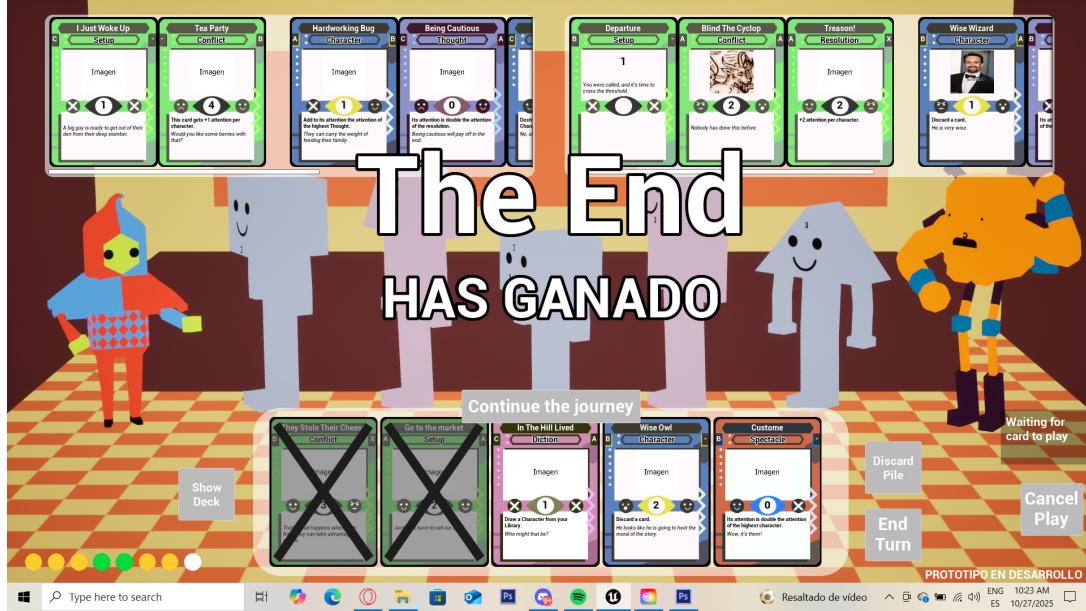


Figura 9.2: Interfaz al finalizar el enfrentamiento (pantalla de resultado).

Durante la partida, el jugador puede consultar en cualquier momento la pila de descartes y las cartas restantes en el mazo mediante sendos menús emergentes, accesibles desde los botones correspondientes de la interfaz.

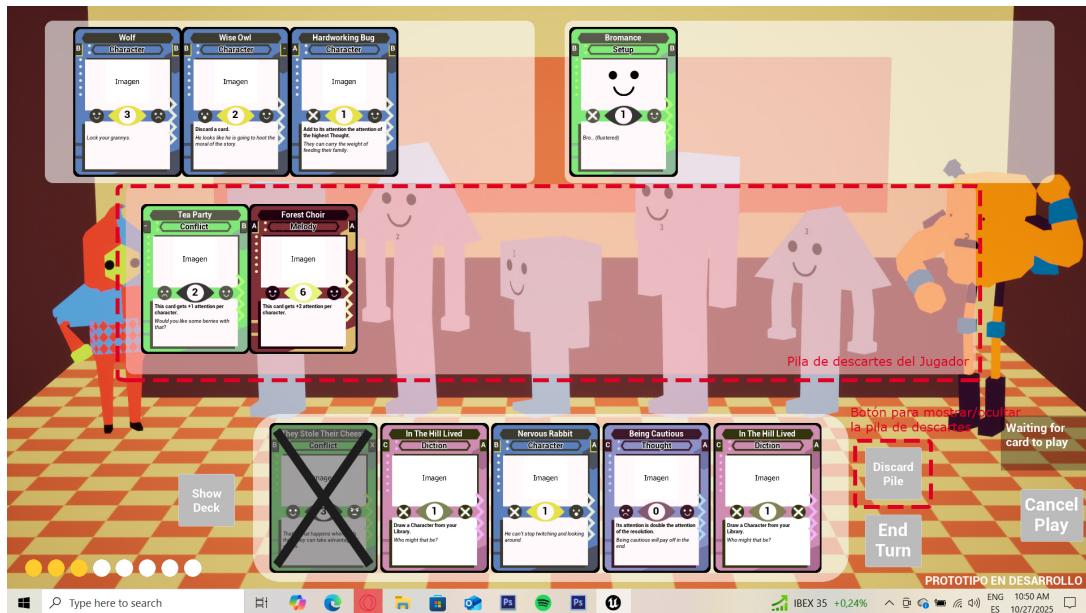


Figura 9.3: Visualización de la pila de descartes.



Figura 9.4: Visualización del mazo restante.

9.2.2. Cartas

Las cartas constituyen el núcleo del diseño del software de *Bardtastic*. Cada carta es una instancia de la clase UCard, derivada de UObject, lo que permite gestionar su ciclo de vida en el motor de forma flexible e independiente de los *Actors* del nivel. La clase encapsula tanto los atributos estáticos de la carta (nombre, tipo, rimas, emociones, texto, imagen) como los elementos de lógica necesarios para su funcionamiento en partida.

9.2.2.1. Representación de los datos y propiedades

Cada carta almacena sus datos básicos como propiedades editables en Unreal, lo que facilita su configuración desde el editor y su exposición a *Blueprints*. Estos atributos describen su funcionalidad y su papel en el enfrentamiento: tipo de carta, atención que aporta, rimas que condicionan la jugada, emociones que pueden provocar en la audiencia y recursos visuales asociados. Gracias a esta estructura, el diseño narrativo y visual puede evolucionar sin necesidad de modificar código nativo.

9.2.2.2. Lógica de juego y efectos

Además de los datos descriptivos, las cartas incluyen un arreglo de efectos, implementados a través de instancias de clases derivadas de UEffector_0bject. Cada efecto sigue el patrón de interfaz común e implementa un método de ejecución asociado al contexto de partida:

```
AplicarEfecto(UGameContext* Context, bool FromP1)
```

La carta actúa como un gestor secuencial de esos efectos, procesándolos uno a uno mediante la función interna ContinuarConSiguienteEfecto(). Para ello utiliza referencias al

9 Software desarrollado

GameContext, donde reside toda la información global del enfrentamiento: la audiencia, la mesa del jugador y del rival, los contadores de turno, etc. De este modo, los efectos conocen únicamente el contexto y no necesitan acceder directamente a otros componentes del juego, lo que reduce el acoplamiento y mejora la mantenibilidad.

Este enfoque permite incorporar nuevos tipos de efectos sin modificar la clase de carta, únicamente añadiendo nuevas clases que implementen la interfaz definida.

9.2.2.3. Creación dinámica y carga desde *DataTables*

Las cartas no están definidas de manera rígida en código, sino que se cargan en tiempo de ejecución a partir de un *DataTable*. Cada entrada del *DataTable* corresponde a la descripción de una carta en formato de datos (FCardData), lo que proporciona tres ventajas principales:

- **Escalabilidad:** permite añadir nuevas cartas sin recompilar el proyecto.
- **Localización:** utiliza el sistema nativo de traducción de Unreal para los textos.
- **Optimización de memoria:** los mazos almacenan únicamente referencias al nombre de la carta.

La función CargarDesdeDatos() inicializa la carta a partir de la fila seleccionada del *DataTable*, y la clase EffectLoader se encarga de instanciar los efectos correspondientes definidos en esa entrada.

9.2.2.4. Interacción con la interfaz y el flujo de partida

La carta incorpora dos *delegates* (RequestVisualUpdate y OnVisualUpdateFinished) que actúan como eventos para coordinar la actualización visual en el cliente y asegurar la sincronización con la lógica de juego. Esto permite, por ejemplo:

- Mostrar animaciones antes de aplicar el siguiente efecto.
- Sincronizar resoluciones entre las cartas del jugador y del rival.
- Notificar al gestor de turnos cuando todos los efectos han terminado.

9.2.2.5. Desacoplamiento y escalabilidad del sistema

El diseño se basa en un patrón de paso de mensajes: las cartas solicitan al jugador o a la IA las decisiones necesarias, pero sólo el GameContext conoce el estado global. Este enfoque facilita:

- La extensión futura del sistema (nuevos efectos, nuevos tipos de carta).
- La integración de UI reactiva sin mezclar responsabilidades.
- La evolución independiente de la IA sin modificar la clase UCard.

En conjunto, la arquitectura de las cartas maximiza la modularidad del sistema y proporciona una base sólida para ampliar el contenido jugable del proyecto sin comprometer su estabilidad.

9.2.3. Jugadores

En el prototipo ambos participantes del enfrentamiento (el jugador controlado por el usuario y el rival controlado por la inteligencia artificial) comparten una arquitectura común. El comportamiento base se implementa en C++ en la clase AA_Rival, que actúa como controlador lógico de un jugador dentro del nivel y gestiona su mazo, mano, mesa de cartas permanentes, descartar, robar y finalizar turno.

A partir de esta clase se ha construido una jerarquía en Blueprint que permite separar la lógica del juego de la presentación visual y de la interacción específica de cada tipo de jugador:

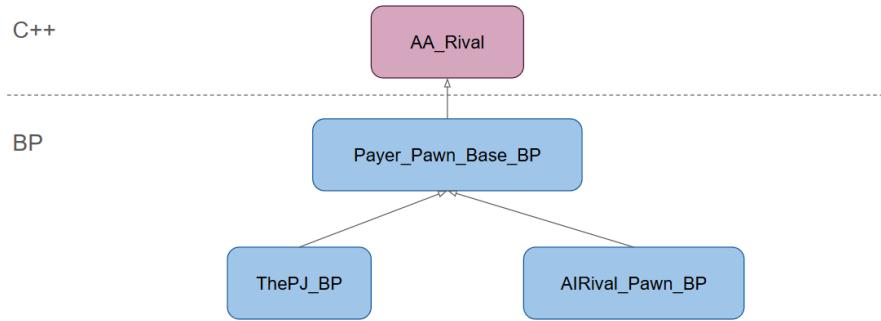


Figura 9.5: Arquitectura general de los jugadores en *Bardtastic*.

Este enfoque híbrido sigue un patrón común en Unreal Engine: la lógica del sistema permanece en C++ para garantizar robustez y rendimiento, mientras que la capa visual y de interacción se desarrolla en Blueprints, facilitando la edición en el editor sin necesidad de recompilar.

9.2.3.1. Jugador

El jugador controlado por el usuario se representa mediante ThePJ_BP. Esta clase hereda todas las funciones jugables desde C++, pero redirige las decisiones al usuario a través de la interfaz (*Widgets*) y gestiona las animaciones asociadas al avatar en el escenario. De esta forma, el usuario puede realizar acciones como seleccionar cartas o confirmar la finalización del turno sin modificar la arquitectura interna.

9.2.3.2. Rival e Inteligencia Artificial

El oponente se representa mediante AIRival_Pawn_BP, que también deriva de Player_Pawn_Base_BP. La interfaz de decisiones de C++ (funciones como RequestCardChoice() o StartTurn()) se implementa aquí conectándose con un componente de IA capaz de decidir qué cartas jugar.

La inteligencia artificial no está integrada en el *Pawn* de manera rígida, sino que se delega en una clase independiente (URivalAIBase), que expone el método:

```
TArray<UCard*> DecidePlay(UGameContext* Context, bool FromP1)
```

permitiendo así sustituir o ampliar la estrategia del rival sin afectar al resto del sistema. La primera implementación del prototipo, UFirstRivalAI, utiliza una lógica básica para seleccionar una secuencia válida de cartas teniendo en cuenta el estado de la partida.

9.2.3.3. Ventajas de esta organización

Esta arquitectura por capas proporciona varios beneficios:

- **Desacoplamiento de responsabilidades:** C++ gestiona reglas y estado; Blueprint gestiona visualización y control específico.
- **Escalabilidad:** es posible añadir nuevos tipos de jugador sin duplicar lógica.
- **Modularidad de IA:** la estrategia puede evolucionar sin tocar el controlador del jugador.
- **Iteración rápida:** cambios en animaciones y UI sin recompilar.

En conjunto, este diseño permite mantener la integridad del sistema de juego mientras se ofrece flexibilidad para continuar expandiendo contenido y mejorar el comportamiento de la IA durante futuros desarrollos del proyecto.

9.3. Agente Rival

El diseño conceptual del agente se ha introducido en la sección 5.4. Aquí se describe su implementación *software*: arquitectura, componentes implicados y flujo de decisión que materializa la heurística de la ecuación (5.1).

9.3.1. Arquitectura y desacoplamiento

La implementación separa estrictamente la lógica de jugador del razonamiento de IA:

- AA_Rival (C++): clase base de jugador. Gestiona mazo (UDeck), mano, mesa, descartes y ciclo de turno (StartTurn(), AddCardToPlay(), CanAddCardToPlay(), EndTurn(), ...).
- Player_Pawn_Base_BP (BP): envoltorio común en Blueprint para presentar/animar acciones de AA_Rival.
- AIRival_Pawn_BP (BP): rival IA (hereda de Player_Pawn_Base_BP); delega la decisión en la estrategia.
- URivalAIBase (C++, UObject): interfaz de estrategia, expone DecidePlay(UGameContext*, bool).
- UFirstRivalAI (C++): estrategia actual; construye la jugada mediante DFS guiada por (5.1).
- AIScoringUtils (C++): utilidades de simulación/puntuación para calcular términos de (5.1) sin mutar el estado real.

Este diseño permite sustituir UFirstRivalAI por variantes (voraz, *beam search*, MCTS parcial) sin tocar ni AA_Rival ni los *Pawns*. De esta forma, permite modificar la inteligencia del rival sin alterar la estructura general del juego, fomentando la reutilización y la experimentación con distintos modelos de decisión.

9.3.2. Flujo de decisión en turno

1. **Inicio de turno** (StartTurn() en AA_Rival): estado consistente (mano, mesa, audiencia).
2. **Cálculo de jugada** (AIRival_Pawn_BP → UFirstRivalAI):
 - Generación: DFS sobre secuencias legales (rimas, tipos, unicidad/orden de Historia).
 - Simulación: cada extensión se valora con AIScoringUtils para obtener Atención, ProgresoHistoria y Ciclo; Longitud se deriva de la propia secuencia.
 - Selección: se conserva la mejor secuencia según (5.1).
3. **Ejecución**: AIRival_Pawn_BP reproduce la secuencia en orden llamando a AddCardToPlay(), coordinada con los *delegates* visuales de las cartas.
4. **Objetivos de atención**: Cuando un efecto requiere escoger espectador, el agente simula cada posible selección y elige la que maximiza la ganancia de atención total, considerando bonificaciones emocionales y equilibrio entre bandos (cruce de bando → ganancia neta → cercanía a +3).
5. **Acciones auxiliares**: en robos/descartes, cada carta se puntúa individualmente (prioridad a completar historia → atención estimada → ciclado).
6. **Fin de turno** (EndTurn()).

La gestión del turno la hace el GestorTurnos, dando paso al agente y el agente le notifica el final.

9.3.3. Notas de complejidad y rendimiento

La mano está acotada a 5 cartas, lo que permite una DFS exhaustiva práctica. El orden “mejor-primer” por (5.1) actúa como poda implícita; además, se penaliza la redundancia (p.ej., repetir una pieza de Historia ya presente) para contener la rama efectiva. La simulación se realiza sobre copias ligeras del estado (no sobre UObject vivos), evitando costes y efectos colaterales.

En la práctica, la exploración completa del árbol de jugadas tiene una complejidad controlada gracias a las restricciones narrativas del juego y al tamaño acotado de la mano, lo que permite mantener una respuesta inmediata sin comprometer la calidad de la decisión.

9.3.4. Puntos de extensión

- **Heurística**: ajuste de w_i y rasgos situacionales (último turno, taberna con reglas globales, etc.).
- **Estrategia**: nuevas subclases de URivalAIBase.
- **Efectos**: ampliar AIScoringUtils para simular efectos futuros sin acoplar la IA a las cartas.

En suma, la implementación traslada el modelo de la sección 5.4 a un sistema modular y extensible que decide en tiempo interactivo, manteniendo la legibilidad del comportamiento y la coherencia con el diseño del juego.

9.4. Eventos

Además de los enfrentamientos, el prototipo incluye un sistema de eventos que permite modificar el mazo del jugador entre partidas. Estos eventos se ejecutan en un nivel independiente del combate y actúan como nodos narrativos donde el jugador toma decisiones que afectan a su progresión.

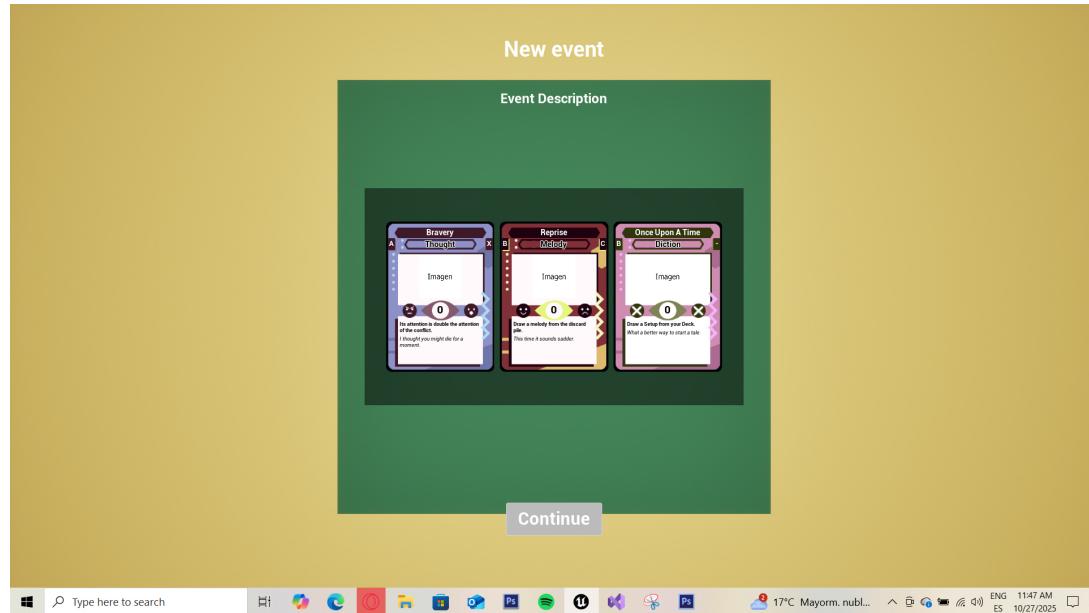


Figura 9.6: Evento donde se nos pide elegir una de entre tres cartas.

9.4.1. Arquitectura general

El sistema se organiza en torno a dos elementos principales:

- Un **Pawn** específico del nivel de eventos (EventPJ_BP), que posee una cámara fija y se encarga de inicializar el evento activo.
- Un **Widget** asociado, responsable de mostrar las opciones disponibles al jugador y de trasladar sus decisiones a la lógica del evento.

EventPJ_BP crea dinámicamente una instancia de evento al comienzo del nivel y establece la referencia correspondiente dentro del widget. Esta separación asegura que la interfaz nunca conozca directamente la lógica del mazo ni del enfrentamiento, manteniendo así un bajo acoplamiento entre subsistemas.

9.4.2. Lógica del evento

La base del sistema está implementada en C++ mediante la clase UEvent_Base, que define los atributos principales de un evento y la interfaz necesaria para su comunicación con la interfaz de usuario:

- **Título y descripción:** información que se muestra en pantalla.
- **GetEventCards():** función que devuelve un conjunto de cartas u opciones disponibles para el jugador.

Actualmente, la elección de la opción del evento se gestiona directamente desde Blueprint, pero la arquitectura está pensada para ampliarse con nuevos métodos que gestionen resultados, costes o consecuencias. En concreto, se prevé incorporar un delegado o función como:

```
OnOptionSelected(UCard* SelectedCard)
```

lo que permitirá que cada evento defina cómo actúa sobre el mazo del jugador u otros sistemas del juego.

9.4.3. Extensibilidad del sistema

Un aspecto clave del diseño es que **los eventos son completamente modularizables**. Cada uno podrá definirse como una subclase de UEvent_Base (ya sea en Blueprint o en C++) y proporcionar su propia implementación de:

- selección de opciones,
- condiciones para mostrar cartas,
- efectos sobre el mazo o sobre elementos persistentes del juego.

Desde la perspectiva de ingeniería del software, esta solución facilita:

- la creación rápida de contenido nuevo mediante diseño visual,
- la incorporación progresiva de mecánicas avanzadas según crezca el proyecto,
- la escalabilidad de la estructura de campaña o de las rutas posibles del jugador.

9.4.4. Integración con la progresión del juego

Tras finalizar un combate, se carga este nivel de evento que permite:

- eliminar cartas poco útiles,
- obtener nuevas cartas del catálogo,
- realizar mejoras simples (en futuras iteraciones),
- tomar decisiones estratégicas antes del siguiente enfrentamiento.

Una vez tomada una decisión, el widget invoca la función correspondiente en UEvent_Base, se modifica el mazo del jugador a través del sistema de guardado y se regresa al nivel de combate para continuar la progresión.

En resumen, el sistema de eventos introduce variabilidad entre partidas y refuerza la dimensión estratégica del juego, permitiendo que el jugador adapte su mazo y desarrolle una estrategia progresiva a lo largo de la aventura.

9.5. Sistema de guardado

El prototipo utiliza el sistema *SaveGame* de Unreal Engine para persistir el estado entre sesiones. Este mecanismo se basa en clases de datos serializables derivadas de *USaveGame*, que se crean, leen y escriben mediante funciones utilitarias del motor (*CreateSaveGameObject*, *SaveGameToSlot*, *LoadGameFromSlot*, etc.). El flujo general consiste en: (i) instanciar un objeto *SaveGame*, (ii) poblar sus campos con el estado del juego que se desea conservar, (iii) serializarlo a disco bajo un *slot* con nombre, y (iv) cargarlo en el arranque o cuando sea necesario restablecer la partida.

9.5.1. Modelo general en Unreal

En Unreal, un guardado se representa como una clase de datos (UCLASS derivada de *USaveGame*) que contiene únicamente la información que debe persistir. La E/S se gestiona desde el código de juego o *Blueprints*, invocando a:

- `UGameplayStatics::CreateSaveGameObject(ClaseSave)` para crear el contenedor.
- `UGameplayStatics::SaveGameToSlot(Obj, SlotName, UserIndex)` para escribir.
- `UGameplayStatics::LoadGameFromSlot(SlotName, UserIndex)` para leer.

Cada *slot* define un archivo independiente, lo que permite manejar múltiples guardados lógicos (p. ej., *Run* actual, progreso global, ajustes del usuario). Por buenas prácticas, el objeto *USaveGame* almacena tipos *POD* o identificadores estables (nombres de filas de *DataTables*, *FString*, *int32*, etc.), evitando referencias directas a *UObject*/*AActor* vivos. La reconstrucción en memoria se realiza posteriormente, resolviendo esos ID contra catálogos o *DataTables*.

9.5.2. Guardado actual: SavedRun

El proyecto implementa un guardado básico centrado en la *run* en curso, llamado *SavedRun*. Su responsabilidad es persistir el estado del mazo del jugador entre combates. Para ello, el mazo se almacena como una lista de identificadores de carta (p. ej., nombres de fila en el *DataTable* de cartas), junto con la información mínima necesaria para reconstruir el estado jugable al cargar:

- **Deck:** secuencia de IDs de carta (referencias a filas del *DataTable*).
- (Opcional futuro) **Semilla RNG:** para reproducibilidad entre cargas.
- (Opcional futuro) **Pista de progreso:** taberna/encuentro actual, turnos restantes.

Esta elección (IDs en lugar de punteros) garantiza independencia entre la persistencia y los objetos en memoria. Al cargar, el cada jugador crea su mazo y resuelve cada ID contra el *DataTable* de cartas, instancia *UCard* y repuebla las estructuras (*UDeck*, mano, etc.) según corresponda.

9.5.3. Múltiples guardados lógicos

El sistema está preparado para crecer mediante múltiples *slots* especializados. Además del guardado de *run* actual (*SavedRun*), es habitual en *roguelike deckbuilders* disponer de:

- **Progreso global** (`SavedProfile`): cartas desbloqueadas, estadísticas del jugador, configuraciones de accesibilidad, opciones.
- **Opciones del usuario** (`SavedSettings`): audio, gráficos, controles.

Separar estos guardados permite borrar o reiniciar la *run* sin perder el progreso global, y viceversa. Cada clase `USaveGame` se guarda en su *slot* (p.ej., `RunSlot`, `ProfileSlot`, `SettingsSlot`), con su propio ciclo de carga/guardado.

9.5.4. Ciclo de vida de guardado y carga

Para evitar E/S redundante, el ciclo recomendado es:

1. **Carga temprana**: al iniciar el juego o al entrar en la *run*, intentar `LoadGameFromSlot`. Si no existe, inicializar valores por defecto y crear un guardado nuevo.
2. **Uso en memoria**: mantener el estado en memoria durante la sesión.
3. **Guardado explícito y atómico**: escribir en momentos concretos (fin de combate, tras un evento, al volver al menú). Evitar guardados continuos.

Este patrón reduce accesos a disco y minimiza riesgos de corrupción por interrupciones durante la escritura.

9.5.5. Rendimiento y restricciones de plataforma

De cara a una futura exportación a Nintendo Switch, es crucial limitar el número y la duración de las operaciones de E/S:

- **Agrupar escrituras**: guardar al principio/final de la transición de nivel (menú ↔ combate ↔ evento) y no tras cada acción.
- **Tamaño de guardado**: mantener objetos `USaveGame` pequeños (IDs y datos primitivos). Evitar volcar estados completos de `UObject`.

En plataformas con E/S restringida, este enfoque reduce latencias visibles y evita bloqueos por tiempo de acceso a disco limitado.

9.5.6. Estado actual y ampliaciones previstas

En la versión actual se persiste `SavedRun` con el Deck del jugador. La arquitectura está preparada para añadir en el futuro:

- **Progreso global**: catálogo de cartas desbloqueadas, estadísticas de partidas.
- **Metajuego**: mejoras permanentes, *traits* o bonificadores.
- **Opciones de accesibilidad y gráficos**: perfiles por usuario.

Todas estas extensiones reapruechan el mismo patrón de *slots* y la resolución por IDs, manteniendo el impacto en rendimiento bajo y el sistema de guardado cohesionado.

En conclusión, el *SaveGame* de Unreal ofrece una solución directa y eficiente para la persistencia. El diseño adoptado en *Bardtastic* favorece la estabilidad (IDs y *DataTables*), la escalabilidad (múltiples *slots* lógicos) y el rendimiento (guardados oportunos y compactos), con especial atención a las restricciones de E/S en plataformas como Nintendo Switch.

9.6. Menú principal

El acceso al juego se realiza a través del nivel MainMenu, que actúa como punto de entrada al prototipo. En este nivel se utiliza un Pawn específico, denominado MenuPJ_BP, que incluye una cámara estática y es responsable de inicializar la interfaz del menú.

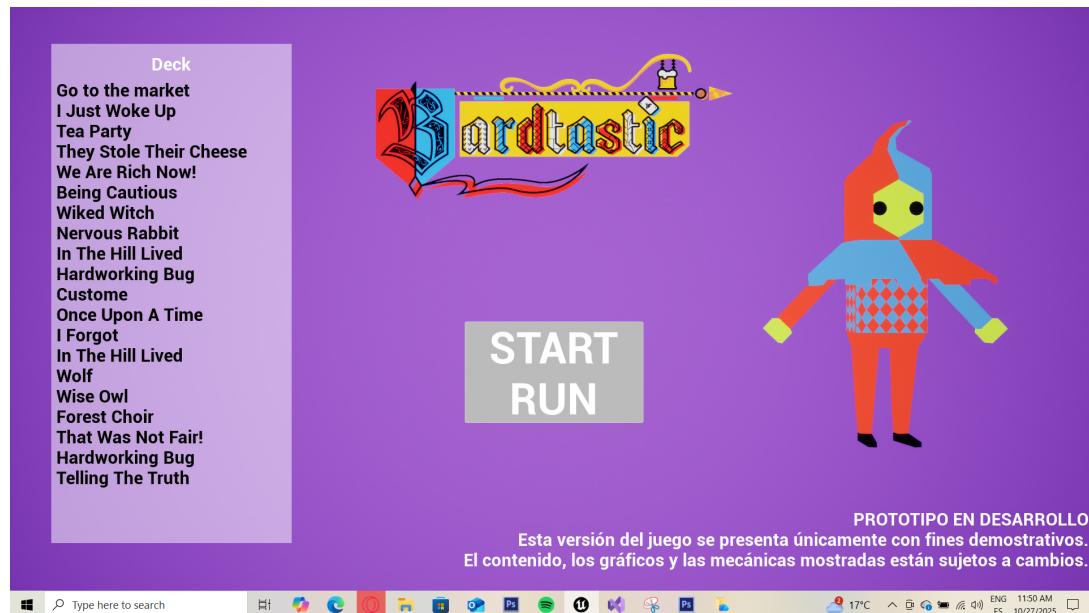


Figura 9.7: Menú principal.

El MenuPJ_BP crea el Widget asociado al menú principal, el cual contiene un botón que permite comenzar la partida y cargar el nivel de enfrentamiento correspondiente. Además, con el objetivo de facilitar las pruebas durante el desarrollo, el menú permite visualizar el mazo actual del jugador, representando las cartas que se usarán en la siguiente partida.

Dado que el proyecto se encuentra todavía en fase de prototipado, el diseño visual del menú es provisional y cumple principalmente una función de validación de flujo y accesibilidad a las mecánicas principales. La interfaz final (incluyendo una presentación más trabajada, animaciones, navegación ampliada y una organización más completa del contenido) está prevista para su desarrollo durante la construcción de la *Vertical Slice*.

Capítulo 10: Evaluación y pruebas

Este capítulo describe el enfoque de verificación seguido durante el desarrollo del prototipo de *Bardtastic*, los resultados obtenidos y un plan estructurado para formalizar las pruebas de cara a la futura *Vertical Slice*. Dado el carácter prototípico del software, la validación se ha centrado en **pruebas exploratorias y sesiones de juego supervisadas**, apoyadas por un uso intensivo de *logging* y trazas en tiempo real. A continuación se sistematiza lo realizado y se proponen medidas concretas para elevar el nivel de aseguramiento de calidad.

10.1. Objetivos de la evaluación

- Verificar la **viabilidad jugable** del *core loop* (robar, validar rimas, aplicar efectos, alternar turnos, determinar ganador).
- Asegurar la **consistencia de estado** (mazo, mano, mesa, descarte, audiencia, emociones).
- Detectar **errores críticos** que impidan terminar una partida.
- Recoger **feedback de usabilidad** durante sesiones de *playtest*.

10.2. Estrategia empleada (prototipo)

Durante esta fase no se han implementado tests unitarios automatizados. La validación se ha realizado mediante:

10.2.1. Pruebas exploratorias

La validación se ha centrado en un enfoque de pruebas exploratorias, en el que la propia desarrolladora interactúa con el sistema sin un guion rígido, observando el comportamiento del juego y buscando posibles fallos en sus mecánicas principales. Dentro de este enfoque se han aplicado tres técnicas complementarias:

- **Pruebas sistemáticas de juego:** sesiones completas y repetidas de enfrentamientos cubriendo tanto flujos habituales como situaciones límite (mazo agotado, rima inválida, empate en audiencia, etc.).
- **Smoke tests:** se ejecutan de manera frecuente al inicio de cada iteración de desarrollo para comprobar que las funcionalidades esenciales no han dejado de funcionar tras introducir cambios. En el contexto del prototipo, estos tests consisten en:
 - cargar el menú principal,
 - iniciar un combate,
 - jugar varias rondas con alternancia de turnos,
 - determinar un ganador y volver al flujo del juego.

El objetivo es detectar fallos críticos de manera temprana y garantizar que el juego sigue siendo ejecutable en todo momento.

- **Error guessing:** técnica basada en la experiencia de la desarrolladora para anticipar dónde es más probable que aparezcan errores. Se busca “romper” el sistema forzando situaciones anómalas, como:

- descartar todas las cartas de la mano en un punto no previsto,
- generar cadenas largas de efectos consecutivos,
- aplicar atención que supere los límites $[-3, 3]$,
- provocar que la IA no disponga de una secuencia válida.

Esta técnica resulta especialmente útil en fases tempranas, cuando aún no se cuenta con una batería completa de casos de prueba formales.

10.2.2. Playtesting supervisado

Se realizaron pruebas informales con usuarios (principalmente compañeros) con el objetivo de observar la comprensibilidad de la interfaz, la claridad del *feedback* y el ritmo de juego. Durante estas sesiones se registraron observaciones cualitativas relacionadas con posibles confusiones, tiempos de decisión, errores de lectura de rimas y la claridad del resultado final.

10.2.3. Trazas y depuración

Se hizo un uso intensivo de **logs** y **prints** para seguir el flujo de turnos, la validación de rimas y la ejecución secuencial de efectos. Además, se emplearon marcadores visuales (*debug draws* o *widgets* temporales) para confirmar los cambios de atención y de emociones durante la ejecución.

10.3. Cobertura funcional alcanzada

A nivel práctico se han verificado los siguientes bloques:

- **Flujo de turnos:** alternancia Jugador/IA, inicio y fin de turno, conteo de 7 turnos por bando.
- **Cartas:** carga desde *DataTables*, creación en tiempo de ejecución, validación de rimas, aplicación de efectos.
- **Audiencia:** actualización de atención con saturación $[-3, 3]$ y transición de emociones básica.
- **Resultado:** cálculo de ganador por mayoría de audiencia; desempate por historia completa.
- **Eventos:** menú intermedio para modificar el mazo (eliminar/añadir según evento actual).
- **Guardado:** persistencia básica de la *run* (SavedRun con Deck).

10.4. Casos de prueba manuales (muestra)

A continuación se documentan casos representativos usados de forma manual durante el prototipado. Sirven como base para su futura automatización.

ID	Área	Descripción	Resultado
TC-01	Turnos	Iniciar partida y completar 14 turnos (7 por bando) sin errores.	OK
TC-02	Rimas	Intentar jugar una carta con <i>rima anterior</i> distinta de la <i>rima posterior</i> de la carta previa.	Rechazo/OK
TC-03	Efectos	Jugar carta que suma atención a varios espectadores y verificar saturación en $[-3, 3]$.	OK
TC-04	Historia	Completar historia (inicio, nudo, desenlace + 1 personaje + 1 pensamiento) y forzar empate de audiencia.	Desempate/OK
TC-05	Mazo	Agotar mazo y reconstruir desde descarte; continuar turno actual.	OK
TC-06	IA	Forzar mano de IA con varias secuencias válidas; comprobar selección coherente.	OK (heur.)
TC-07	Evento	Tras combate, abrir evento, aplicar modificación de mazo y volver a combate.	OK
TC-08	Guardado	Guardar tras evento y cargar; verificar integridad del mazo.	OK

10.5. Limitaciones del enfoque actual

- **Sin pruebas unitarias/funcionales automatizadas:** dependencia total de tests manuales.
- **Sin medición sistemática de rendimiento:** percepción subjetiva de fluidez.
- **Playtesting no estructurado:** muestras pequeñas y sin métricas cuantitativas.

10.6. Plan de formalización de pruebas (*Vertical Slice*)

Para la siguiente fase se propone un plan incremental, compatible con Unreal:

10.6.1. Niveles y tipos de prueba

- **Unitarias (C++):** validación de utilidades puras (comparación de rimas, cálculo de atención, saturación, orden de efectos).
- **Funcionales/UI (Blueprint & Functional Testing):** flujo *end-to-end* en mapas de test (cargar carta, jugar, actualizar audiencia, fin de turno).

- **Integración** (sistemas): UCard + EffectLoader + GameContext + Deck.
- **Experiencia de usuario**: *playtests* con protocolo, métricas y cuestionario SUS/attrak-Diff.

10.6.2. Herramientas de Unreal recomendadas

- **Automation System (C++)**: pruebas unitarias con IMPLEMENT_SIMPLE_AUTOMATION_TEST.
- **Functional Testing / Automation Spec**: mapas de prueba con *steps* de juego.
- **Gauntlet** (si aplica): ejecución automatizada en *builds*.
- **Commandlets simples**: generación/carga de mazos de prueba.

10.6.3. Matriz mínima de pruebas (RF → TC)

RF	TC-01	TC-02	TC-03	TC-04	TC-05	TC-06	TC-07	TC-08
RF-1 Turnos	X							
RF-2 Robo					X			
RF-3 Rimas		X				X		
RF-4 Efectos			X					
RF-5 Tipos			X					
RF-6 Atención			X					
RF-7 Emociones			X					
RF-8 IA						X		
RF-9 Fin	X							
RF-10 Ganador				X				

10.6.4. Criterios de aceptación y métricas

- **Funcional**: o *blockers* y o *críticos* tras batería de TC-01..08.
- **Rendimiento**: acción típica (validar rima + aplicar efectos) < 100 ms en PC objetivo.
- **Playtest**: ≥ 80 % de jugadores comprende reglas de rima sin explicación adicional; tasa de finalización de partida ≥ 90 %.

10.6.5. Procedimiento de *playtest* estructurado

- **Protocolo**: briefing (1 min) → partida (10–15 min) → encuesta corta (SUS) + entrevista (3 min).
- **Instrumentación**: contadores de turnos, tiempo por decisión, errores de rima, impulsos de atención por carta.
- **Registro**: hoja de incidencias (reproducible/pasos/esperado/obtenido).

Capítulo 11: Conclusiones y trabajo a futuro

El presente Trabajo de Fin de Grado ha tenido como objetivo el diseño e implementación de un prototipo funcional del videojuego *Bardtastic*, un *deckbuilder* narrativo en el que el jugador encarna a un bardo que debe entretenecer a una audiencia mediante el uso estratégico de cartas que contienen elementos narrativos y puestas en escena. El proyecto ha combinado la faceta creativa del diseño de juegos con el rigor técnico de la Ingeniería Informática y los fundamentos matemáticos de la Teoría de Juegos y la Inteligencia Artificial.

11.1. Síntesis y logros del proyecto

A lo largo del desarrollo se ha abordado la construcción de un prototipo completo que incluye la lógica de juego, la gestión de cartas, la interfaz de usuario y un agente rival controlado por IA. El prototipo cumple plenamente su función como **prueba de concepto**: valida la idea jugable y demuestra la viabilidad de un sistema narrativo competitivo basado en reglas probabilísticas y toma de decisiones racional.

En el plano teórico, se han aplicado formalismos propios de la Teoría de Juegos, concretamente los modelos de juegos de suma cero y los algoritmos de exploración de árboles de decisión. El agente implementado utiliza una exploración de árboles con una función heurística, ajustada a la naturaleza aleatoria y a la información oculta del juego, y ha mostrado un comportamiento **coherente, racional y competitivo**, capaz de evaluar las jugadas del jugador humano y responder con decisiones óptimas dentro de la información disponible.

Desde el punto de vista de la ingeniería, se ha puesto especial énfasis en la **arquitectura modular y extensible del software**. El diseño basado en clases bien encapsuladas, el uso de componentes reutilizables y la separación entre la lógica del juego, la IA y la presentación visual, permiten que el sistema pueda ampliarse fácilmente con nuevos elementos: cartas, mecánicas, tipos de audiencia o incluso agentes con distintos estilos de juego. Este principio de modularidad garantiza la mantenibilidad del código y su posible reutilización en versiones futuras o proyectos derivados.

11.2. Valoración de los objetivos

Los objetivos establecidos al inicio del TFG se han alcanzado satisfactoriamente:

- Se ha diseñado e implementado un videojuego tipo *deckbuilder* con una propuesta original y temática coherente.
- Se ha aplicado la Teoría de Juegos para modelar la toma de decisiones de los oponentes.
- Se ha desarrollado un agente de IA funcional, basado en exploración de árboles, capaz de jugar de forma racional contra el usuario.
- Se ha validado experimentalmente el concepto de juego, confirmando la viabilidad de su mecánica central y del modelo de atención del público.

11 Conclusiones y trabajo a futuro

Estos resultados integran los conocimientos adquiridos en el grado, demostrando la capacidad de aplicar conceptos teóricos a un desarrollo *software* real y funcional. Además, se amplían y combinan hacia ámbitos como el diseño.

11.3. Trabajo a futuro

El proyecto deja abiertas varias líneas de desarrollo que pueden explorarse en trabajos posteriores:

- **Ampliación del contenido jugable:** incorporación de más cartas, efectos y sinergias que profundicen en la estrategia del jugador y aumenten la rejugabilidad.
- **Evolución de la IA:** sustitución del algoritmo determinista por aproximaciones estocásticas basadas en *Monte Carlo Tree Search (MCTS)*, capaces de manejar incertidumbre y explorar grandes espacios de decisión.
- **Optimización del rendimiento:** análisis de costes computacionales de la exploración de árboles y paralelización de cálculos en GPU o hilos independientes.
- **Integración narrativa y visual:** mejora el diseño de personajes, arte y sonido para convertir el prototipo en una obra más llamativa.
- **Comercialización:** pasar de prototipo a juego completo y viable para lanzar en el mercado de PC y consolas.

En conjunto, *Bardtastic* demuestra que un videojuego puede servir como banco de pruebas para conceptos avanzados de decisión racional y equilibrio estratégico. El enfoque modular adoptado y la base teórica empleada aseguran que el proyecto pueda crecer, diversificarse y evolucionar tanto en el terreno académico como en el creativo.

Bibliografía

- [1] Number of cards legal in Standard over time - AFR Standard was the largest in history. Consultado el 17 de octubre de 2025. Gráfico construido en base a los datos del artículo de Wikipedia "History of Standard", que ya no se encuentra disponible en línea.
- [2] Richard Garfield | Board Game Designer | BoardGameGeek. Consultado el 14 de octubre de 2025.
- [3] Magic: The Gathering | el sitio oficial de las noticias, colecciones y eventos de mtg, 1993-2025. Consultado el 14 de octubre de 2025.
- [4] 8o Level Editorial Team. How Strategy Games Apply the Rock Paper Scissors Mechanic, 2022. Consultado el 9 de octubre de 2025.
- [5] G. A. Akerlof. The market for "lemons": Quality uncertainty and the market mechanism. *The Quarterly Journal of Economics*, 84(3):488–500, 1970.
- [6] D. J. Anderson. *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.
- [7] Aristotle. *Poetics*. Penguin Classics, London, 1996. Edición moderna consultada.
- [8] D. Benmergui. Storyteller. <https://store.steampowered.com/app/1624540/Storyteller>, 2023. Indie developer.
- [9] T. Bertram. Urzagpt: Lora-tuned large language models for card selection in collectible card games. *arXiv preprint arXiv:2508.08382*, 2025. Submitted on 11 Aug 2025.
- [10] E. Cerdá Tena, J. Pérez Navarro, and J. L. Jimeno Pastor. *Teoría de Juegos*. Pearson Educación, S.A., Madrid, 2004.
- [11] B. Chen and J. Ankenman. *The Mathematics of Poker*. ConJelCo, LLC, Pittsburgh, PA, 2006.
- [12] E. Coble. *The Hero's Journey of Odysseus: A Monomyth Guide to the Iliad and Odyssey*. Independently published, 2021. Guía moderna sobre la lectura monomítica.
- [13] G. Costikyan. I have no words and i must design. 2002. Disponible en línea: <http://www.costik.com/nowords.html>.
- [14] M. Elliott and J. Kovalic. Quarriors! <https://boardgamegeek.com/boardgame/91536/quarriors>, 2011. WizKids.
- [15] M. Elliott and E. M. Lang. Marvel dice masters: Avengers vs. x-men. <https://boardgamegeek.com/boardgame/157707/marvel-dice-masters-avengers-vs-x-men>, 2014. WizKids.
- [16] Epic Games. Unreal engine. <https://www.unrealengine.com>, 2025. Accedido: <poner fecha de acceso>.
- [17] S. Games. Hades. <https://store.steampowered.com/app/1145360/Hades/>, 2020. Supergiant Games.
- [18] P. Garnaeu. Fourteen Forms of Fun. <https://www.gamedeveloper.com/design/fourteen-forms-of-fun>, Oct. 2001. Consultado el 27 de octubre de 2025.
- [19] J. Gary. Ascension: Deckbuilding game. <https://boardgamegeek.com/boardgame/69789/ascension-deckbuilding-game>, 2010. Stone Blade Entertainment.
- [20] J. Gary. Richard Garfield — Delving into Game Narratives, The Essence of Memorable Gameplay, and Building Bridges with Fellow Designers (#11), Dec. 2019. Consultado el 14 de octubre de 2025.
- [21] J. Gary. Reiner Knizia — Systems for Publishing 700+ Games, Crafting Profound Gameplay from Simple Rules, Innovations in Scoring and Auction Systems, and The Joy of Tabletop Gaming (#52), 2023. Consultado el 9 de octubre de 2025.

Bibliografía

- [22] Godot Engine Community. Godot engine. <https://godotengine.org>, 2025. Accedido: <poner fecha de acceso>.
- [23] M. Harlan. *Texas Hold'em For Dummies*. Wiley Publishing, Hoboken, NJ, 2006.
- [24] J. C. Harsanyi. Games with incomplete information played by "bayesian" players, i-iii. *Management Science*, 14:486–502, 1968. Presenta la formulación de los juegos de información incompleta (juegos bayesianos).
- [25] J. Heinrich and D. Silver. Self-play Monte-Carlo Tree Search in Computer Poker. In *Proceedings of the AAAI Workshop on Computer Poker and Imperfect Information Games*, pages 1–9. AAAI Press, 2014.
- [26] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975. Reeditado por MIT Press en 1992.
- [27] J. Huizinga. *Homo Ludens*. Alianza Editorial, Madrid, 2012.
- [28] S. Jackson and J. Kovalic. Munchkin. <https://boardgamegeek.com/boardgame/1927/munchkin>, 2001. Steve Jackson Games.
- [29] C. Larman. *Agile and Iterative Development: A Manager's Guide*. Addison-Wesley, 2004.
- [30] LocalThunk. Balatro. <https://store.steampowered.com/app/2379780/Balatro/>, 2024. Playstack.
- [31] E. McMillen. The Binding of Isaac. https://store.steampowered.com/app/113200/The_Binding_of_Isaac/, 2011. Edmund McMillen.
- [32] MegaCrit. Slay the Spire. https://store.steampowered.com/app/646570/Slay_the_Spire/, 2019.
- [33] L.-M. Miranda. Hamilton: An american musical. Musical, 2015. Richard Rodgers Theatre, Broadway.
- [34] A. Nealen. Ascension: a Case Study in Deckbuilding Games. In *Proceedings of DiGRA 2013: DeFrangling Game Studies*, 2013. Available at: <https://www.nealen.net/ascension.pdf>.
- [35] C. T. Nguyen. *Games: Agency As Art*. Oxford University Press, Oxford, 2020. Disponible en línea: <https://global.oup.com/academic/product/games-9780190052089>.
- [36] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, Cambridge, MA, 1994.
- [37] Power Up Plus. Programa de mentoría y aceleración para estudios de videojuegos independientes, 2024. Consultado el 24 de octubre de 2025.
- [38] P. G. Pérez, A. M. M. García, and N. M. Medina. Diseño y optimización de un agente autónomo para el juego magic: The gathering. In *CEUR Workshop Proceedings*, 2023.
- [39] S. J. Russell and P. Norvig. *Inteligencia Artificial: Un enfoque moderno*. Pearson Educación, Madrid, 4 edition, 2021.
- [40] J. Schell. *The Art of Game Design: A Book of Lenses*. CRC Press, Boca Raton, FL, 2008.
- [41] K. Schwaber and J. Sutherland. The scrum guide. <https://scrumguides.org/>, 2020. Último acceso: <poner fecha>.
- [42] H. A. Simon. *Models of Man: Social and Rational*. Wiley, New York, 1957.
- [43] D. Sklansky. *The Theory of Poker*. Two Plus Two Publishing, Las Vegas, NV, 4 edition, 1994. Referencia clásica sobre las reglas y fundamentos estratégicos del póker.
- [44] E. Sottsass and M. Group. Memphis Design Movement. <https://memphis-milano.com/>, 1981. Corriente de diseño y arquitectura posmoderna surgida en Italia.
- [45] B. Suits. *The Grasshopper: Games, Life and Utopia*. University of Toronto Press, Toronto, 1978.
- [46] G. N. Szabados. Roguelike games: Specifications of the genre and empirical research on gamer relations. Master's thesis, University of Debrecen, 2023. DEA Repository item: 135a6032-33e8-4d57-adab-92628398f7a8.

- [47] L. Taylor. Pillars of Design. <https://boardgamedesignlab.com/wp-content/uploads/2023/11/Pillars-of-Design.pdf>, 2023. Consultado el 27 de octubre de 2025.
- [48] M. Toy, G. Wichman, and K. Arnold. Rogue. [https://en.wikipedia.org/wiki/Rogue_\(video_game\)](https://en.wikipedia.org/wiki/Rogue_(video_game)), 1980. Epyx.
- [49] J. Tyroller. Dungeon Clawler. https://store.steampowered.com/app/1921660/Dungeon_Clawler/, 2023. Indie developer.
- [50] D. X. Vaccarino. Dominion. <https://boardgamegeek.com/boardgame/36218/dominion>, 2008. Rio Grande Games.
- [51] C. Vogler. *The Writer's Journey – 25th Anniversary Edition: Mythic Structure for Writers*. Michael Wiese Productions, Los Angeles, CA, 2020.
- [52] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ, 1944.
- [53] P. Ward. Adventure Time. <https://www.imdb.com/title/tt1305826/>, 2010. Cartoon Network Studios.
- [54] S. Woods. *Eurogames: The Design, Culture and Play of Modern European Board Games*. McFarland & Co., Jefferson, NC, 2012.
- [55] J. Woolford. *How Musicals Work: And How to Write Your Own*. Nick Hern Books, London, 2012.
- [56] World Rock Paper Scissors Association. Pokémon is Just Glorified Rock Paper Scissors, 2020. Consultado el 9 de octubre de 2025.
- [57] World Rock Paper Scissors Association. How the Rock Paper Scissors Framework Underpins Most Modern Games, 2021. Consultado el 9 de octubre de 2025.
- [58] J. Yorke. *Into the Woods: How Stories Work and Why We Tell Them*. Penguin Books, London, 2013.