



UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA EN DESARROLLO DE
VIDEOJUEGOS Y REALIDAD VIRTUAL

Ajuste de Dificultad Adaptativo para Juegos de
Puzzles

SAMUEL ANTONIO QUIROZ GONZÁLEZ

Profesor Guía: NICOLÁS BARRIGA

Profesor Co-guía: JOSÉ DEL SOLAR

Memoria para optar al título de
Ingeniero en Desarrollo de Videojuegos y Realidad Virtual

Talca – Chile
Julio, 2025



UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA EN DESARROLLO DE
VIDEOJUEGOS Y REALIDAD VIRTUAL

Ajuste de Dificultad Adaptativo para Juegos de
Puzzles

SAMUEL ANTONIO QUIROZ GONZÁLEZ

Profesor Guía: NICOLÁS BARRIGA

Profesor Co-guía: JOSÉ DEL SOLAR

Memoria para optar al título de
Ingeniero en Desarrollo de Videojuegos y Realidad Virtual

El presente documento fue calificado con nota: _____

Talca – Chile

Julio, 2025

Dedicado a mi novia, quien estuvo a mi lado durante toda la carrera, a mis padres por su apoyo incondicional en cada etapa, y a mis amigos por estar presentes y ayudarme a lo largo de este camino.

AGRADECIMIENTOS

Agradezco a mi profesor guía Nicolás Barriga y a mi profesor co-guía José del Solar por sus constantes revisiones, sus valiosos comentarios, su tiempo y compromiso durante todo el proceso.

TABLA DE CONTENIDOS

	página
Dedicatoria	I
Agradecimientos	II
Tabla de Contenidos	III
Índice de Figuras	V
Índice de Tablas	VI
Resumen	VII
 1. Introducción	 1
 2. Estado del arte y marco teórico	 3
2.1. Métricas de dificultad	4
2.2. Dificultad adaptativa dinámica	5
2.3. Descripción de las métricas utilizadas	9
2.4. Sistema de Evaluación TrueSkill	11
2.4.1. Modelo probabilístico	11
2.4.2. Actualización y aprendizaje	11
2.4.3. Ventajas y pertinencia en este proyecto	12
2.4.4. Diferencias entre TrueSkill y TrueSkill2	12
 3. Metodología	 13
3.1. Recopilación de información	13
3.1.1. Cuestionario para recopilar información inicial	14
3.2. Desarrollar el algoritmo	16
3.3. Evaluación del sistema	17
 4. Implementación	 20
4.1. Recolección de datos	20
4.2. Sistema de selección de niveles	22

4.2.1.	Fiabilidad de las escalas de percepción	23
4.2.2.	Matriz de correlación	24
4.2.3.	Predicción de dificultad esperada y su incertidumbre	25
4.2.4.	Cálculo de desempeño del jugador	26
4.2.5.	Actualización bayesiana del perfil de habilidad del jugador	27
5.	Resultados	29
5.1.	Descripción del experimento	29
5.1.1.	Comparación entre grupo experimental y control	31
5.1.2.	Niveles completados	33
5.1.3.	Análisis en jugadores que completaron 9 o 10 niveles	34
6.	Conclusiones	39
	Bibliografía	40
A:	Cuestionario player experience inventory (PXI)	45
B:	Selección de niveles	48

ÍNDICE DE FIGURAS

	página
2.1. Categorías de ajuste dinámico de dificultad	7
4.1. Flujo general del sistema implementado	22
4.2. Matriz de correlación entre variables normalizadas.	25
5.1. Comparación de la distribución de <i>Enjoyment</i> entre todos los jugadores y aquellos que completaron 9 o 10 niveles.	37
5.2. Comparación de la distribución de <i>Ease of control</i> entre todos los jugadores y aquellos que completaron 9 o 10 niveles.	37
5.3. Comparación de la distribución de <i>Autonomy</i> entre todos los jugadores y aquellos que completaron 9 o 10 niveles.	38

ÍNDICE DE TABLAS

	página
4.1. Alfa de Cronbach para agrupaciones de tres ítems	23
4.2. Alfa de Cronbach para combinaciones por pares	23
4.3. Alfa de Cronbach para todas las percepciones (Sin desafío percibido 1)	23
5.1. Alfas de cronbatch de los constructos	30
5.2. Alfas de cronbatch de los constructos con bajo valor	31
5.3. Resultados simplificados del Test de Levene y t-test para igualdad de medias	32
5.4. Estadísticos grupales por constructo para todos los jugadores	33
5.5. Promedio de niveles completados por grupo	34
5.6. Resultados de la prueba de Levene y t-test para niveles completados .	34
5.7. Resultados simplificados del Test de Levene y t-test para jugadores con 9 o 10 niveles completados	35
5.8. Estadísticos grupales por constructo para jugadores con 9 o 10 niveles completados	36

RESUMEN

Los juegos de puzzles, reconocidos por su capacidad de captar la atención de una amplia audiencia y su versatilidad, han demostrado ser un género prominente en la industria de los videojuegos. Sin embargo, uno de los problemas fundamentales en los juegos de puzzles es la dificultad estática, que en ocasiones puede no ajustarse adecuadamente a las habilidades individuales de los jugadores. Esto puede llevar a una experiencia frustrante o, por el contrario, demasiado fácil, lo que a menudo resulta en que los jugadores abandonen el juego de manera prematura.

Para abordar este desafío, este proyecto propone el desarrollo de un sistema de Ajuste Dinámico de la Dificultad (Dynamic Difficulty Adjustment, DDA), una técnica que ha ganado popularidad y atención en los últimos años. El proyecto utiliza una implementación digital del juego de puzzle Safari Rush Hour, y se centra únicamente en el sistema de DDA. Este sistema se encargará de seleccionar y asignar niveles que se adapten a las habilidades del jugador mediante la evaluación de cada uno de estos. Al final de cada nivel se guarda un valor para cada uno de estos parámetros y en base al historial de puntos obtenido se selecciona el que mejor se adapte a ellos. El objetivo es proporcionar desafíos que sean adecuadamente exigentes sin causar frustración, facilitando un proceso de aprendizaje continuo y efectivo.

Para llevar a cabo este proyecto, se sigue una metodología en tres fases: la primera consiste en una revisión bibliográfica para identificar las técnicas más adecuadas para el desarrollo del algoritmo de DDA. La segunda fase abarca la implementación del algoritmo en la versión digital del juego puzzle Safari Rush Hour, evaluando parámetros como la densidad de piezas y el número de movimientos óptimos. Finalmente, se realizan pruebas de usuario para medir el impacto del sistema en la experiencia de juego, evaluando métricas como tiempo de resolución, cantidad de movimientos y satisfacción del jugador. Los datos obtenidos permiten validar el sistema y garantizar su efectividad.

Este estudio es útil para desarrollar un sistema de DDA, ya sea en juegos de puzzles o de otros géneros, debido a que destaca aspectos clave a considerar en su creación y validación.

1. Introducción

En la industria de los videojuegos, uno de los principales desafíos es proporcionar una experiencia de juego equilibrada y personalizada para cada jugador. Un aspecto crucial en este equilibrio es el nivel de dificultad, el cual puede influir directamente en la satisfacción, retención y motivación del jugador. Un juego demasiado difícil puede generar frustración y abandono prematuro, mientras que uno demasiado fácil puede resultar aburrido y carente de estímulo [1]. Para abordar este problema, se han desarrollado diversas técnicas de DDA, que tienen como objetivo adaptar la dificultad del juego a las habilidades y progreso de cada jugador.

En los últimos años, las técnicas de DDA han ganado relevancia debido a su capacidad para mejorar la experiencia de juego de forma personalizada, ajustando el nivel de desafío a las habilidades del jugador [2]. Esta creciente atención refleja su importancia en el desarrollo de videojuegos, donde equilibrar la dificultad es un factor clave para mantener el interés y la satisfacción de los jugadores.

El problema específico que se aborda en esta memoria es el diseño e implementación de un sistema de Ajuste Dinámico de la Dificultad (Dynamic Difficulty Adjustment, DDA) que considere múltiples parámetros para seleccionar niveles de juego adecuados según el desempeño del jugador. Aunque existen investigaciones previas sobre sistemas adaptativos, este trabajo propone un enfoque que, además de basarse en métricas de desempeño, filtra y selecciona niveles de juego que se ajusten a las habilidades y preferencias individuales, mejorando así la experiencia de juego.

A continuación se presentará la hipótesis, objetivo general y específicos que esta memoria busca responder.

Hipótesis: Un sistema de dificultad adaptativa dinámica que selecciona los niveles

según el desempeño del jugador aumentaría la sensación de maestría y satisfacción, al ofrecer un desafío que se adapta a sus habilidades.

1. **Objetivo general:** Desarrollar un sistema de ajuste dinámico de la dificultad que considere múltiples parámetros y seleccione niveles de juego adecuados para cada jugador, basándose en su desempeño previo.

2. **Objetivos específicos:**

- a) Identificar las técnicas, metodologías, y métricas adecuadas para desarrollar un algoritmo de ajuste dinámico de la dificultad mediante una revisión del estado del arte.
- b) Implementar un algoritmo que, basándose en parámetros específicos, filtre y seleccione los niveles de juego más adecuados para cada jugador, garantizando que se adapten a sus habilidades y preferencias.
- c) Evaluar la efectividad del algoritmo de dificultad adaptativa dinámica a través de pruebas de usuario, analizando su impacto en la experiencia de juego y el desempeño de los jugadores

Este proyecto busca contribuir al desarrollo de sistemas de DDA al proporcionar un ejemplo de su aplicación en juegos de puzzles. Al realizar un estudio inicial para obtener información sobre los niveles y el perfil de los jugadores, seguido de una evaluación que valide la efectividad del sistema, se espera demostrar cómo el DDA influye en la experiencia de usuario y en qué medida la afecta.

El documento se estructura de la siguiente manera: en primer lugar, se presenta el marco teórico y una revisión del estado del arte sobre las técnicas y metodologías existentes de DDA. Posteriormente, se describe el diseño e implementación del sistema propuesto, detallando las métricas y parámetros utilizados. A continuación, se expone la metodología de evaluación mediante pruebas de usuario, las cuales consisten en dos estudios: el primero buscará recopilar información para correlacionar los elementos que determinan si un nivel de puzzle es difícil o fácil, el segundo estudio se enfocará en validar la efectividad de los algoritmos generados para asegurar que realmente cumplen con el ajuste dinámico esperado. Finalmente se presentan los resultados de las pruebas con usuarios realizadas, para finalmente dar las conclusiones de esta memoria de título.

2. Estado del arte y marco teórico

En esta sección se abordarán los estudios previos relacionados con la creación y resolución de juegos de puzzle. A continuación, se analizarán algunas métricas utilizadas para estimar la dificultad de estos juegos, y, por último, se explorará el concepto de dificultad adaptativa dinámica.

La creación de niveles en juegos de puzzles representa un reto que puede abordarse tanto a través del diseño manual como mediante la generación procedural de niveles. Históricamente, los juegos de puzzle han evolucionado significativamente, desde simples rompecabezas de papel hasta complejas experiencias digitales que desafían la mente del jugador. En los últimos años, la generación procedural ha ganado popularidad, no solo por su capacidad para optimizar el tiempo de diseño [3], sino también por su habilidad para crear experiencias personalizadas que se adaptan dinámicamente a las habilidades del jugador [4]. Esto no solo mejora la rejugabilidad, sino que también aumenta la satisfacción del usuario al enfrentar desafíos apropiados. Entre los métodos de generación procedural se encuentran enfoques como Monte Carlo Tree Search (MCTS) [5], aprendizaje profundo [6], algoritmos evolutivos [7] y algoritmos de búsqueda en profundidad [8].

Resolver puzzles manualmente consume mucho tiempo y esfuerzo, por lo que se han desarrollado algoritmos para acelerar este proceso. Por ejemplo, Monte Carlo Tree Search se ha utilizado para resolver crucigramas [9], mientras que para puzzles de lógica como los Zebra puzzles, se han implementado Large Language Models asistidos por Satisfiability Modulo Theories (SMT) [10]. Otros enfoques incluyen algoritmos genéticos para resolver N-puzzles [11] y el algoritmo A* para sliding puzzles [12].

2.1. Métricas de dificultad

Las técnicas de medición de dificultad se pueden agrupar en tres enfoques generales. El primero está directamente relacionado con el modelo formal del juego, el segundo con características seleccionadas del juego, y el tercero involucra al jugador. Estos enfoques permiten al diseñador ajustar de manera precisa la dificultad de los niveles en función del rendimiento del jugador [3].

Modelo formal del juego: El primer enfoque utiliza un modelo formal del juego, generalmente definido en juegos de mesa como el ajedrez o el Conecta 4. Aquí, la dificultad se mide analizando el árbol de juego, donde se evalúan las posibles consecuencias de los movimientos del jugador. Algoritmos como MiniMax o Alpha-Beta Pruning son comunes en estos juegos. Esta técnica se basa en la estructura del árbol del juego, lo que permite que el sistema DDA ajuste la dificultad en función de la distancia más corta al éxito o al fracaso. Como ejemplo para este punto se pueden utilizar las estadísticas de algoritmos de búsqueda o la entropía de la solución del nivel [13], otra métrica puede ser la longitud de la solución, que usando un modelo estadístico, se puede estimar cuantos movimientos le va a tomar a un jugador resolver un puzzle [14], otra métrica que se puede tener en cuenta es la cantidad de intentos que le toma a un jugador completar un nivel, con esto se puede crear un modelo de regresión que sea capaz de predecir la dificultad de un nivel [15]. No obstante, la principal dificultad de estos modelos radica en que las métricas capturan la dificultad teórica, no necesariamente la percibida por el jugador. Un nivel puede tener una solución óptima corta, pero seguir siendo percibido como difícil si requiere intuiciones no triviales o patrones poco evidentes.

Modelo formal es desconocido o muy complejo: El segundo enfoque ocurre en juegos donde el modelo formal es desconocido o muy complejo para utilizarse, como en juegos de disparos en primera persona o plataformas. En estos casos, la dificultad se mide usando características seleccionadas del juego (como el número de enemigos) e indicadores de rendimiento del jugador (como enemigos eliminados por minuto). DDA utiliza estas métricas para ajustar la dificultad en intervalos predeterminados o en etapas específicas del juego. Como ejemplos para este punto se pueden utilizar métricas basadas en agentes,

en donde se simula un jugador real para estimar la dificultad [16]. La principal dificultad de estos modelos es que muchas de estas métricas dependen del contexto y estilo de juego, ya que la experiencia de juego puede variar mucho si por ejemplo un enemigo se encuentra posicionado en un lugar estratégico.

Centrado en la dificultad percibida por el jugador: El tercer enfoque está centrado en el jugador y mide la dificultad percibida. En este caso, el sistema DDA puede consultar directamente al jugador sobre su percepción de la dificultad o analizar sus emociones durante el juego. Este método proporciona un ajuste más personalizado, basado en la experiencia subjetiva del jugador. Como ejemplo se tiene un juego en donde pueden cambiar la características de este en tiempo real para poder ajustarlo al jugador [17].

En el contexto de los juegos de puzzle, estas técnicas de medición de dificultad son particularmente relevantes, ya que la dificultad se evalúa mediante diversas métricas que se agrupan en tres categorías principales: *características iniciales* (disposición de las piezas), *características de solución* (número mínimo de movimientos requeridos) y *características dinámicas* (tiempo necesario para resolver el puzzle) [18]. Estas métricas son esenciales para un sistema de DDA efectivo, ya que permiten ajustar los niveles de forma precisa, asegurando que el jugador enfrente un desafío adecuado según su historial de desempeño.

2.2. Dificultad adaptativa dinámica

Dynamic Difficulty Adjustment (DDA) es una técnica utilizada para ajustar la dificultad de los videojuegos en función del rendimiento del jugador, buscando un equilibrio entre desafío y accesibilidad. Este enfoque permite mantener al jugador inmerso, evitando tanto la frustración como el aburrimiento [4]. El DDA adapta el nivel de dificultad en tiempo real para proporcionar desafíos adecuados sin llegar a ser abrumadores, incrementando la satisfacción y retención del jugador.

La dificultad de un juego varía si se cambia alguno de estos tres elementos clave: atributos, comportamientos y eventos. Los atributos son las variables que definen las características de los personajes y enemigos dentro del juego, como la salud, el daño o la velocidad, los cuales pueden ajustarse en función del rendimiento del jugador. Los comportamientos, por otro lado, se refieren a las acciones y patrones de interacción de

los personajes no jugables (NPCs) y elementos del juego, que pueden ser modificados para aumentar o disminuir el desafío, como la velocidad de ataque de un enemigo o la frecuencia con la que un jefe cambia de fase. Finalmente, los eventos son ocurrencias predefinidas que se activan bajo ciertas condiciones, como proporcionar al jugador un ítem curativo cuando su salud es baja, lo que permite una adaptación más fluida y menos invasiva de la dificultad [19]. Si bien estos componentes no son universales y no se aplican a todos los juegos, establecen una base fundamental para los sistemas de DDA, permitiendo ajustes en tiempo real que aseguran una experiencia de juego equilibrada, adaptándose así a las habilidades y necesidades de cada jugador.

Los métodos de DDA pueden modificar los juegos utilizando diferentes enfoques, que se dividen en dos grandes clases: enfoques basados en reglas y enfoques basados en datos, los cuales especifican cómo deben cambiarse los parámetros según los estados de los jugadores.

Enfoques basados en reglas: Los sistemas basados en reglas dependen de la intuición de los diseñadores, definiendo reglas simples relacionadas con el rendimiento del jugador, como la puntuación o el tiempo, para ajustar dinámicamente el nivel de dificultad. Estas reglas activan el mecanismo de DDA cuando se cumplen ciertas condiciones, generalmente al comparar las métricas de rendimiento del jugador con umbrales predefinidos [20]. Este enfoque no requiere grandes volúmenes de datos y suele ser desarrollado directamente por diseñadores, lo que facilita su implementación en contextos donde se busca una solución más controlada y accesible. Por ejemplo, juegos como la herramienta de gamificación de matemáticas desarrollada por Jácome-Amores [21].

Enfoques basados en datos: Los enfoques basados en datos utilizan técnicas de aprendizaje automático y análisis estadístico para ajustar dinámicamente la dificultad del juego en función de datos del comportamiento o señales fisiológicas del jugador. Estos métodos tienen la capacidad de aprender y evolucionar a partir de las interacciones de los jugadores, permitiendo ajustes de dificultad más personalizados y precisos en tiempo real, superando las limitaciones de los sistemas basados en reglas [20]. No obstante, este tipo de enfoque suele requerir una mayor cantidad de datos, los cuales pueden ser difíciles de obtener en muchos casos, lo que representa un reto adicional en su implementación. Como ejemplo, un estudio propone un sistema basado en Razonamiento Basado en

Casos (Case-Based Reasoning, CBR) para evaluar dinámicamente el nivel de habilidad del jugador y adaptar la dificultad del juego según el último estado del juego haciendo uso del juego Tetris [22].

Entre las técnicas específicas de DDA que pueden implementarse (Ver figura 2.1) se incluyen algoritmos de búsqueda, teoría de control, modelos probabilísticos, aprendizaje por refuerzo, modelos de regresión y redes neuronales [20]. Cada uno de estos enfoques tiene características únicas y se describe en detalle a continuación.

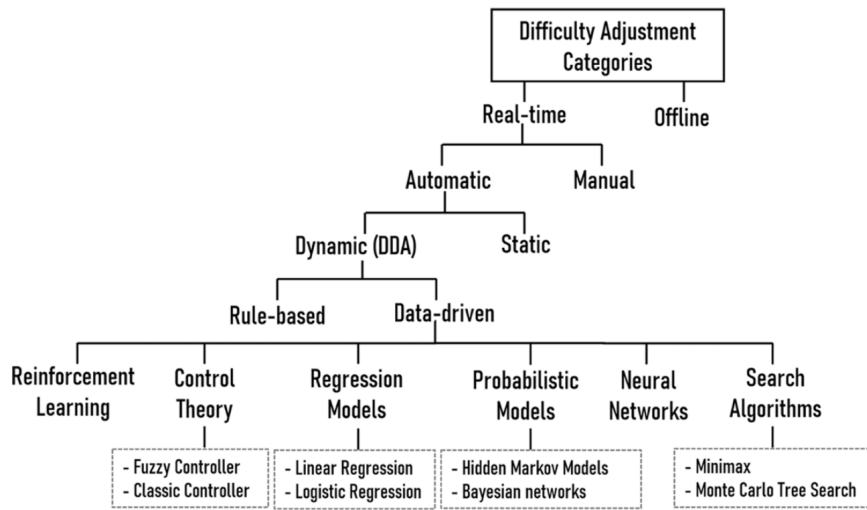


Figura 2.1: Diferentes categorías utilizadas en los sistemas de ajuste de dificultad dinámica en el área de aprendizaje automático [20].

Algoritmos de búsqueda: Minimax y Monte Carlo Tree Search (MCTS) son algoritmos de búsqueda que optimizan la toma de decisiones en juegos de estrategia y tablero. Minimax predice jugadas óptimas al evaluar escenarios posibles, ideal para juegos con movimientos controlados y finitos. MCTS, en cambio, usa búsquedas heurísticas, haciéndolo adecuado para juegos más complejos, donde explora sin necesidad de analizar cada opción exhaustivamente. La principal ventaja de ambos en DDA es su capacidad para responder dinámicamente al jugador ajustando la dificultad según sus movimientos y decisiones [23].

Teoría de control: La teoría de control adapta la dificultad de forma gradual y continua. Los controladores PID, por ejemplo, ajustan en tiempo real basándo-

se en el rendimiento del jugador, proporcionando una experiencia sin cambios abruptos. La lógica difusa utiliza variables emocionales, como frustración o entusiasmo, para adaptar la dificultad [24]. Estas técnicas son útiles para mantener al jugador en un estado de “flujo” (Flow) óptimo, haciendo ajustes pequeños y personalizados en la dificultad del juego.

Modelos probabilísticos Los modelos probabilísticos emplean técnicas estadísticas para inferir patrones de comportamiento del jugador a partir de sus acciones previas. Las redes bayesianas y los Modelos Ocultos de Markov (HMM) utilizan probabilidades condicionales para predecir las decisiones futuras del jugador [25], adaptando la dificultad del juego según el estilo y la tendencia de sus interacciones pasadas. Por otro lado, los N-gramas analizan secuencias de acciones, utilizando probabilidades para anticipar las decisiones inmediatas del jugador. Estas técnicas permiten ajustar dinámicamente la dificultad, anticipando las necesidades del jugador y ajustando el juego de acuerdo con las decisiones y comportamientos probables, lo que asegura una experiencia de juego más fluida y personalizada.

Aprendizaje por refuerzo: Este enfoque usa un agente que ajusta el juego en función del rendimiento del jugador mediante un proceso de prueba y error. A medida que el agente observa el éxito o fracaso del jugador, ajusta la dificultad para mejorar la experiencia. Es particularmente útil en juegos de acción y simulación, donde la dificultad debe reaccionar en tiempo real y adaptarse de manera flexible al estilo de juego [26].

Modelos de regresión: La regresión lineal y logística se emplean para predecir niveles de desafío óptimos en función de datos históricos del jugador. La regresión lineal ayuda a ajustar la dificultad basada en el tiempo o el progreso continuo del jugador, mientras que la logística clasifica su desempeño en términos de éxito o abandono. Estos modelos personalizan la dificultad en función de características individuales del jugador, maximizando así la retención y satisfacción del usuario [27].

Redes neuronales: Este enfoque es realmente útil para interpretar datos complejos del jugador y adaptar la dificultad en función de patrones de comportamiento o emociones, como ansiedad o aburrimiento. Al identificar y prever patrones

de frustración o aburrimiento en el jugador, estas redes permiten ajustar la dificultad de forma continua, manteniendo un balance dinámico entre reto y disfrute, especialmente en juegos de tiempo real [28].

Bosques Aleatorios (Random Forest): Son modelos de aprendizaje supervisado que combinan múltiples árboles de decisión para realizar predicciones robustas y precisas. En el contexto de DDA, se utilizan para predecir la habilidad del jugador basándose en características como el tiempo de reacción, la precisión o patrones de interacción con el juego [29].

En conjunto, estas técnicas de aprendizaje automático permiten que los videojuegos se adapten a las habilidades y preferencias de cada jugador, mejorando la jugabilidad y el compromiso, y ofreciendo una experiencia personalizada que puede ajustarse a medida que el jugador avanza en el juego.

2.3. Descripción de las métricas utilizadas

Para evaluar la efectividad del sistema de ajuste dinámico de dificultad (DDA) propuesto, se utilizarán diversas métricas cuantitativas provenientes de la estadística y el aprendizaje automático. Estas permiten validar tanto la confiabilidad del instrumento de medición como el rendimiento predictivo de los modelos implementados.

- **Alfa de Cronbach (α):** Mide la consistencia interna de un conjunto de ítems que buscan evaluar un mismo constructo (por ejemplo, dificultad percibida o desafío). Se calcula mediante la siguiente fórmula:

$$\alpha = \frac{k}{k-1} \left(1 - \frac{\sum_{i=1}^k \sigma_i^2}{\sigma_T^2} \right) \quad (2.1)$$

Donde k es el número de ítems, σ_i^2 es la varianza de cada ítem individual y σ_T^2 es la varianza total del puntaje combinado. Valores de $\alpha > 0,7$ suelen ser considerados aceptables [30].

- **Mean Absolute Error (MAE):** Representa la media de las diferencias absolutas entre los valores reales y las predicciones del modelo:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (2.2)$$

Esta métrica es robusta frente a valores atípicos y útil en contextos donde los errores pueden presentar colas pesadas o distribución no normal.

- **Root Mean Squared Error (RMSE)**: Representa la raíz cuadrada del error cuadrático medio, dando mayor peso a errores grandes:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (2.3)$$

Es apropiado cuando se asume que los errores siguen una distribución normal e independiente.

- **Importancia de variables**: En modelos de *Machine learning*, se calcula el aporte relativo de cada variable a la predicción, permitiendo identificar qué métricas del nivel (por ejemplo, entropía, tiempo o reinicios) influyen más en la dificultad percibida.
- **Exactitud (Accuracy)**: se refiere al porcentaje de casos correctamente clasificados en modelos que predicen si un jugador completará o no un nivel:

$$\text{Accuracy} = \frac{\text{Nº Predicciones correctas}}{\text{Nº Total predicciones}} \quad (2.4)$$

En este proyecto se utilizará como métrica principal el MAE, dado que la variable objetivo (dificultad percibida) está influenciada por factores subjetivos, y los errores de predicción probablemente no siguen una distribución normal. Al ser menos sensible a valores atípicos y más robusta ante distribuciones con colas pesadas, el MAE permite obtener estimaciones más estables y representativas en este contexto. El RMSE se mantendrá como métrica complementaria para detectar la presencia de errores extremos [31].

2.4. Sistema de Evaluación TrueSkill

TrueSkill es un sistema de evaluación bayesiano desarrollado por Microsoft para determinar la habilidad de jugadores en videojuegos multijugador [32]. Es una extensión del sistema ELO, ampliamente utilizado en ajedrez, pero diseñado para superar sus limitaciones: permite manejar partidas entre múltiples jugadores o equipos, modelar empates y estimar la incertidumbre sobre las habilidades de los jugadores.

2.4.1. Modelo probabilístico

Cada jugador i tiene una habilidad latente s_i modelada como una variable aleatoria con distribución normal:

$$s_i \sim \mathcal{N}(\mu_i, \sigma_i^2) \quad (2.5)$$

donde μ_i representa la estimación de habilidad del jugador y σ_i^2 representa la incertidumbre asociada.

En una partida, la habilidad se traduce en un rendimiento observado p_i :

$$p_i \sim \mathcal{N}(s_i, \beta^2) \quad (2.6)$$

donde β es una varianza fija que modela la variabilidad del desempeño en cada juego.

2.4.2. Actualización y aprendizaje

La actualización de las habilidades en el sistema TrueSkill se basa en los resultados de las partidas, definidos como ganar, perder o empatar. Tras cada encuentro, el sistema ajusta las distribuciones de habilidad $\mathcal{N}(\mu_i, \sigma_i^2)$ de los jugadores involucrados mediante inferencia bayesiana.

Cuando un jugador gana, su valor de μ (habilidad estimada) aumenta y su incertidumbre σ disminuye. En caso de perder, μ disminuye. Si hay empate, el sistema interpreta que las habilidades eran similares y ajusta los valores de μ y σ en menor magnitud.

Este mecanismo permite capturar gradualmente el progreso del jugador y reducir la incertidumbre con el tiempo. La actualización se realiza con pocas partidas, permitiendo que el sistema responda con rapidez a los cambios de rendimiento. Esta

retroalimentación dinámica es clave para que el sistema de DDA seleccione niveles acordes al perfil actual de habilidad del jugador.

2.4.3. Ventajas y pertinencia en este proyecto

En el contexto de este proyecto, donde se busca adaptar la dificultad de los niveles en función del rendimiento del jugador, el uso de TrueSkill permite:

- Modelar el progreso del jugador como una distribución de probabilidad que se actualiza dinámicamente.
- Establecer un sistema de matchmaking entre niveles y jugadores, seleccionando desafíos cuya dificultad se acerque a la habilidad esperada del jugador.
- Reemplazar una simple definición binaria de “victoria” por una estimación continua de rendimiento [33], que puede considerar métricas como número de movimientos, tiempo total o reinicios.

2.4.4. Diferencias entre TrueSkill y TrueSkill2

TrueSkill2 es una extensión del modelo original de TrueSkill que mejora significativamente la precisión de las evaluaciones de habilidad al incorporar más información observada y realizar suposiciones más realistas [33]. A diferencia de TrueSkill, que sólo considera el resultado de la partida (ganar, perder o empatar), TrueSkill2 también utiliza estadísticas individuales como número de muertes, bajas, tendencia a abandonar y pertenencia a escuadras. Además, TrueSkill2 asume que las habilidades en diferentes modos de juego están correlacionadas, y modela el progreso del jugador con un sesgo hacia la mejora, especialmente en sus primeras partidas. Otra diferencia clave es que penaliza a los jugadores que abandonan partidas, ajustando su habilidad como si hubieran perdido. TrueSkill2 puede operar en modo en línea o en lote, lo que permite tanto actualizaciones rápidas como análisis históricos más precisos. Estas mejoras permiten al sistema adaptarse mejor a escenarios complejos y datos ruidosos, como los encontrados en juegos multijugador modernos.

Al comparar el rendimiento esperado (derivado de la habilidad estimada por TrueSkill) con el rendimiento real obtenido por el jugador en un nivel específico, es posible calcular un delta de rendimiento que permite ajustar dinámicamente la dificultad del siguiente nivel a mostrar.

3. Metodología

En esta sección se describe la adaptación del juego de puzzle “Safari Rush Hour”, seguido por la metodología utilizada en el proyecto para desarrollar esta memoria. La estructura de la metodología está dividida en tres fases principales, que responden a los objetivos planteados en la sección anterior. Estas fases son recopilación de información, desarrollo del algoritmo y evaluación del sistema DDA [19], permitiendo un desarrollo progresivo y enfocado en lograr una experiencia de juego adaptativa y personalizada para los usuarios.

Safari Rush Hour es un juego de puzzle de bloques deslizantes que se desarrolla en un tablero cuadrado de 7 x 7. El objetivo principal del juego es mover las piezas en el tablero para despejar el camino y permitir que un rover explorador de 2 x 2 llegue a la salida, que se encuentra al lado derecho del tablero en la tercera y cuarta fila.

El juego cuenta con 19 piezas de rompecabezas divididas en 7 tipos diferentes, cada uno representando un animal. Estas piezas se colocan en el tablero para formar una configuración inicial. Una vez en el tablero, las piezas se pueden desplazar horizontal o verticalmente dependiendo de su orientación inicial. Entre estas piezas, hay algunas de forma cuadrada de 2 x 2, que al igual que el rover, tienen la capacidad de moverse en ambas direcciones.

3.1. Recopilación de información

En esta primera fase, se llevará a cabo una recolección de datos mediante pruebas con usuarios, diseñadas para recopilar información relevante sobre el desempeño de los jugadores en diferentes niveles del juego. Este análisis preliminar tiene como objetivo identificar los parámetros que determinan cuándo un nivel puede considerarse

difícil o cuándo un jugador demuestra una habilidad elevada, permitiendo así definir los criterios necesarios para ajustar dinámicamente la dificultad del juego.

A continuación, se detalla el cuestionario que se utilizará en esta etapa, el cual está estructurado en secciones específicas para evaluar tres aspectos principales: la experiencia previa de los jugadores con videojuegos y juegos de puzzles, la dificultad percibida en cada nivel y la satisfacción general de los participantes con los niveles jugados. Las preguntas se encuentran en escala Likert de 7 opciones [34], en donde se señalan los extremos y el usuario debe elegir cual de las opciones lo representa mejor.

Las preguntas se diseñaron a partir de constructos, lo que implica que, para medir un aspecto específico, se formularon varias preguntas similares con el objetivo de reducir el ruido en las respuestas. Cada constructo está compuesto por tres preguntas, las cuales se mezclan con el resto de las preguntas del cuestionario antes de ser presentadas al usuario [34].

3.1.1. Cuestionario para recopilar información inicial

Preguntas de Trasfondo

Estas preguntas buscan conocer el perfil y la experiencia previa de los participantes con videojuegos y juegos de puzzles [35].

1. ¿Qué edad tienes?

(Respuesta abierta)

2. ¿Cuántas horas a la semana dedicas a jugar videojuegos?

Opciones: Menos de 1 hora, 1-2 horas, 2-4 horas, 4 - 7 horas, 7-12 horas, 12-20 horas, Más de 20 horas.

3. Soy muy bueno en juegos de puzzle

Opciones: Totalmente en desacuerdo, ..., Totalmente de acuerdo.

Preguntas Específicas del Nivel

Estas preguntas se realizarán inmediatamente después de que el jugador complete un nivel.

Dificultad percibida Este constructo incluye preguntas diseñadas para evaluar cómo los jugadores perciben la dificultad de cada nivel. Medir esta percepción es crucial para facilitar la entrada al estado de flujo (Flow) [1], convirtiéndola en una métrica clave para lograr dicho estado óptimo de experiencia.

1. Este nivel fue muy difícil

Opciones: Totalmente en desacuerdo, ..., Totalmente de acuerdo.

2. Estuve cerca de abandonar este nivel debido a su dificultad.

Opciones: Totalmente en desacuerdo, ..., Totalmente de acuerdo.

3. Tuve que esforzarme mucho para superar este nivel.

Opciones: Totalmente en desacuerdo, ..., Totalmente de acuerdo.

Desafío percibido Este constructo contiene preguntas que tienen como finalidad estimar el desafío percibido del jugador en cada nivel, ya que otros estudios lo indican como un factor importante [36].

1. Este nivel me ofreció un desafío justo, ni demasiado fácil ni demasiado difícil.

Opciones: Totalmente en desacuerdo, ..., Totalmente de acuerdo

2. Este nivel representó un desafío interesante para mí.

Opciones: Totalmente en desacuerdo, ..., Totalmente de acuerdo

3. El nivel me hizo pensar y esforzarme para encontrar una solución.

Opciones: Totalmente en desacuerdo, ..., Totalmente de acuerdo

Datos adicionales recogidos

Además del cuestionario, el juego recogerá las siguientes métricas automáticamente para cada nivel:

- ¿Fue capaz de terminar el nivel? (*Sí/No*)
- Cantidad de reinicios realizados.
- Longitud del camino encontrado (en movimientos).
- Tiempo total empleado para completar el nivel (en segundos).

- Nodos expandidos por A*.
- Entropía del nivel.

Preparación de datos

Los resultados del cuestionario y las métricas del juego se integran en un dataset estructurado. Con este dataset, se buscan patrones que correlacionen las percepciones subjetivas (dificultad y desafío) con los datos objetivos (tiempo, nodos expandidos, entropía, etc.). Esto permite definir parámetros más precisos para implementar el sistema de DDA en el juego.

3.2. Desarrollar el algoritmo

A partir de los datos obtenidos en la fase anterior, se procederá a diseñar y desarrollar el algoritmo de DDA. El objetivo de esta etapa es crear un sistema capaz de ajustar dinámicamente la dificultad de los niveles del juego, adaptándolos a la habilidad individual de cada jugador. Para ello, el algoritmo analizará y clasificará los niveles, buscando lograr un equilibrio entre el desafío y la accesibilidad.

Para este desarrollo, se han identificado varias métricas que capturan aspectos clave del nivel y de la solución requerida:

Elementos iniciales del nivel Se considerará la proporción entre piezas ocupadas y el espacio total disponible en el tablero y el número de piezas que bloquean la salida.

Elementos de la solución Se evalúa la cantidad de nodos expandidos por A* [37] y cantidad de entropía del nivel [13].

Con los datos recopilados, se emplean técnicas de aprendizaje automático (Machine Learning) para analizar la relación entre las variables del sistema. En este contexto, la dificultad percibida por los usuarios se define como la variable dependiente, mientras que el resto de las métricas se consideran variables independientes. El objetivo es desarrollar un modelo capaz de predecir la dificultad percibida en función de los datos disponibles.

Como modelos de aprendizaje automático, se puede utilizar Random Forest, una técnica que ha demostrado buenos resultados al estimar la dificultad de niveles en

estudios previos [29]. Los resultados obtenidos mediante se compararán con los otros métodos de aprendizaje automático, como lo puede ser modelos basados en Redes Bayesianas (Bayesian Networks), que también han mostrado eficacia al estimar la dificultad de niveles [21]. Esta comparación busca determinar cuál de los modelos proporciona una predicción más precisa para la dificultad percibida de cada nivel. Además, para niveles que aún no han sido jugados, los modelos presentados permiten predecir la dificultad sin que los jugadores tengan que probarlos.

El sistema también debe ser capaz de evaluar la habilidad del jugador para asignarle un nivel adecuado. Para ello, se empleará un modelo random forest, donde la variable dependiente es la habilidad del jugador, y las variables independientes incluyen resultados de niveles anteriores, como si el jugador logró completarlos, la cantidad de reinicios, la longitud del camino encontrado y el tiempo requerido. La elección de un modelo de random forest se basa en estudios previos muestran que este modelo fue capaz de estimar la habilidad de un jugador con un 70 % de Exactitud (accuracy) en los primeros 30 segundos de un juego [38].

3.3. Evaluación del sistema

En esta etapa final, se realizan pruebas con usuarios para evaluar la eficacia del sistema de DDA. Para ello, se forman dos grupos: uno en el que el sistema de DDA estará activado y otro en el que estará desactivado. El objetivo es validar si el algoritmo asigna correctamente los niveles de dificultad en función del desempeño del jugador y si esto mejora la experiencia de juego. Los resultados obtenidos no solo permitirán medir la efectividad del sistema, sino también identificar áreas de mejora y realizar ajustes en el algoritmo según sea necesario, garantizando que el sistema cumpla con los objetivos de personalización y adaptación en la experiencia de juego.

Inicialmente, se consideró utilizar el Game Experience Questionnaire (GEQ) como herramienta para la evaluación. Sin embargo, siguiendo los estudios de Chong, Brühlmann y Mekler [39], se descartó esta opción, ya que estos autores advierten sobre las deficiencias de este cuestionario, destacando que, en su contexto de aplicación, sus resultados no son confiables y que no refleja adecuadamente la experiencia del jugador. La crítica a esta escala fue tan contundente que se hizo un llamado de atención a los investigadores en la sección de conclusiones.

Por otro lado, se optó por emplear el “Player Experience Inventory” (PXI) [34], ya

que este cuestionario ha demostrado ser una herramienta robusta en la evaluación de la experiencia del jugador. Según un estudio de validación con una amplia muestra de participantes, el PXI muestra buenos resultados en la medición de las consecuencias psicosociales y funcionales del juego digital.

El modelo PXI cuenta con 6 pasos para poder construir un cuestionario, estos son:

Familiarízate con PX y sus constructos El PXI mide la experiencia del jugador como un constructo multidimensional con 11 aspectos: 5 funcionales (control, retroalimentación, estética, claridad, desafío) y 6 psicosociales (dominio, curiosidad, inmersión, autonomía, significado y disfrute). Cada uno tiene tres preguntas similares para mayor confiabilidad en los resultados [40].

Respetar la validación El PXI es una escala validada, modificar constructos o preguntas invalida los resultados. Puedes agregar preguntas adicionales, como disfrute o demografía, pero no cambiar las originales.

Usar la medición correcta El PXI utiliza una escala Likert de 7 puntos (-3 a +3) con etiquetas de “Totalmente en desacuerdo”, hasta “Totalmente de acuerdo”. Cambiar este formato invalida la escala. El puntaje -3 a +3 facilita la interpretación.

Definir la plataforma de distribución Elegir el medio de distribución adecuado (papel, online, móvil o dentro del juego). Hay herramientas disponibles para simplificar su implementación.

Definir el momento de medición Aplicar el PXI cuando la experiencia del jugador esté reciente, preferiblemente justo después de jugar o mientras el recuerdo sigue fresco.

Aleatorizar el orden de los ítems Mezcla las preguntas de los constructos para evitar agrupamientos y sesgos, garantizando respuestas imparciales.

Dado que el PXI ha demostrado ser una herramienta eficaz y fiable para medir la experiencia del jugador, se utilizará la versión estándar del cuestionario en este estudio. Como se ha mencionado en los resultados del estudio de validación, el PXI

muestra buenos resultados en la evaluación de las consecuencias psicosociales y funcionales del juego digital, lo que lo convierte en una opción robusta para evaluar el impacto del sistema de DDA en la experiencia del jugador [40].

Para cerrar, se realiza un análisis comparativo entre los dos grupos. Esto permite identificar diferencias en las métricas evaluadas, destacando si el sistema de DDA tiene un impacto positivo en la experiencia de juego y en la percepción de los usuarios. Este análisis es fundamental para validar la efectividad del sistema y respaldar los ajustes necesarios en su implementación.

4. Implementación

En este capítulo se describe la implementación del sistema propuesto, abordando cada una de sus etapas: desde la recolección de datos de usuarios reales, la construcción de un sistema de evaluación de habilidad inspirado en TrueSkill2, hasta la asignación dinámica de niveles que se ajusten a las capacidades estimadas de cada jugador.

4.1. Recolección de datos

Para la recolección de datos, se modificó una versión ya existente del videojuego, adaptándola a una plataforma web conectada a una base de datos en tiempo real mediante Firebase. Esta transformación tuvo como principal objetivo facilitar la distribución del experimento, permitiendo el acceso remoto desde diversos dispositivos y automatizando el almacenamiento de datos generados durante las sesiones de juego. El software creado está públicamente disponible ¹.

Antes de que el usuario pudiera comenzar a interactuar con el sistema, se le presentaba un consentimiento informado, conforme a los principios éticos establecidos por la Universidad de Talca. Dicho consentimiento incluía una descripción detallada del propósito del estudio, el tipo de datos recolectados, su uso exclusivo con fines académicos, y el derecho del participante a retirarse en cualquier momento. Para continuar, era obligatorio que el participante completara un breve formulario con información básica como edad, horas estimadas de juego a la semana y experiencia previa en puzzles. Solo una vez aceptado el consentimiento, el jugador podía acceder al juego.

¹<https://samu385.github.io/Safari-rush-hour-build/>

Durante la experiencia, después de completar cada nivel, se desplegaba automáticamente un cuestionario breve con preguntas diseñadas para evaluar la percepción del jugador respecto a la dificultad del nivel. Estas preguntas, descritas previamente en el capítulo de la metodología, permitieron construir una medida subjetiva de dificultad percibida que sería utilizada posteriormente en el entrenamiento del sistema predictivo.

Cada sesión almacenó automáticamente los siguientes datos por nivel: número de movimientos, número de reinicios, tiempo utilizado, tiempo de resolución óptimo, profundidad promedio del árbol de búsqueda, entropía del nivel, cantidad de nodos expandidos, y respuestas a seis preguntas de percepción. Esta información fue crucial tanto para entrenar los modelos de predicción como para alimentar el sistema adaptativo de dificultad descrito más adelante.

La elección de una arquitectura web con almacenamiento en la nube permitió escalar el experimento a un número mayor de participantes y simplificó el proceso de recolección y limpieza de los datos, asegurando trazabilidad y consistencia en los registros obtenidos.

En la Figura 4.1 se presenta un resumen visual del flujo completo seguido durante la implementación del sistema.



Figura 4.1: Flujo general del sistema implementado

4.2. Sistema de selección de niveles

Para lograr una asignación precisa de niveles según la habilidad del jugador, se implementó un sistema basado en el modelo TrueSkill2, extendido con elementos personalizados para juegos de tipo puzzle. Este sistema considera tanto el rendimiento

del jugador como la dificultad esperada del nivel, integrando un modelo de predicción de dificultad y un mecanismo de actualización bayesiana que permite ajustar dinámicamente la estimación de habilidad del jugador.

4.2.1. Fiabilidad de las escalas de percepción

Se evaluó la consistencia interna de los grupos de ítems que componían los constructos medidos mediante escalas Likert, utilizando el coeficiente alfa de Cronbach. Los resultados obtenidos se presentan en la Tabla 4.1 y la Tabla 4.2.

Tabla 4.1: Alfa de Cronbach para agrupaciones de tres ítems

Constructo	Alfa de Cronbach
Desafío Percibido 1, 2 y 3	0.5205
Dificultad Percibida 1, 2 y 3	0.7844

Tabla 4.2: Alfa de Cronbach para combinaciones por pares

Pares de ítems	Alfa de Cronbach
Desafío Percibido 1 y 2	0.4099
Desafío Percibido 1 y 3	-0.1371
Desafío Percibido 2 y 3	0.7530

Tabla 4.3: Alfa de Cronbach para todas las percepciones (Sin desafío percibido 1)

Ítems	Alfa de Cronbach
Percepciones	0.887

Los resultados muestran que el constructo *Desafío Percibido*, compuesto por tres ítems, presenta una fiabilidad interna insuficiente según el umbral comúnmente aceptado de $\alpha > 0,7$. En particular, el ítem **Desafío Percibido 1** mostró una correlación baja e incluso negativa con los ítems restantes (ver Tabla 4.2), indicando que no mide coherentemente el mismo concepto.

El análisis de consistencia interna para los ítems **Perception2** a **Perception6**, excluyendo el ítem conflictivo, arrojó un alfa de Cronbach de 0.887 (ver Tabla 4.3). Este valor sugiere una excelente fiabilidad interna y respalda la validez del constructo

medido por estos ítems. Por tanto, se decidió emplear el promedio de estas cinco variables como una nueva medida compuesta de dificultad percibida. Esta decisión mejora la robustez estadística del análisis posterior y garantiza una representación más coherente de la percepción de dificultad por parte de los jugadores.

4.2.2. Matriz de correlación

Se generó una matriz de correlación de Pearson sobre las variables numéricas normalizadas para visualizar relaciones internas del dataset. Esta permitió observar asociaciones clave entre métricas del nivel como entropía, nodos expandidos y movimientos óptimos, y cómo estas se relacionan con la percepción de la dificultad (Ver figura 4.2).

De estos resultados podemos ver que “MediaDificultad” presenta correlaciones moderadas a altas con múltiples métricas del sistema, destacando su asociación con los movimientos óptimos (*OptimalMoves*, 0.62), nivel de entropía (*LevelEntropy*, 0.60) y Altura promedio del árbol (*AverageTreeHeight*, 0.60). Esto indica que la percepción subjetiva de dificultad se alinea con propiedades estructurales del nivel.

También se correlaciona con tiempo total (*TotalTime*, 0.52) y movimientos totales (*TotalMovements*, 0.43), lo que sugiere que los jugadores perciben mayor dificultad en niveles que requieren más tiempo y acciones para ser completados.

En contraste, variables personales como horas de juego a la semana (*HoursPlayed*) o la habilidad en los juegos de puzzle (*PuzzleHabilities*) muestran correlaciones débiles o nulas, evidenciando que la dificultad percibida está más influenciada por las características del nivel que por los atributos del jugador.

Estos resultados también permiten observar que el tiempo total invertido en el nivel presenta una correlación más alta con la dificultad percibida (0.52) que el tiempo de la última solución o (*Time*, 0.35), lo que sugiere que considerar el proceso completo del jugador ofrece una mejor estimación de su experiencia. De forma similar, los movimientos totales (*TotalMovements*, 0.43) presentan una asociación más fuerte con la dificultad que los movimientos de la última solución, respaldando su uso como métrica más representativa del esfuerzo real percibido.

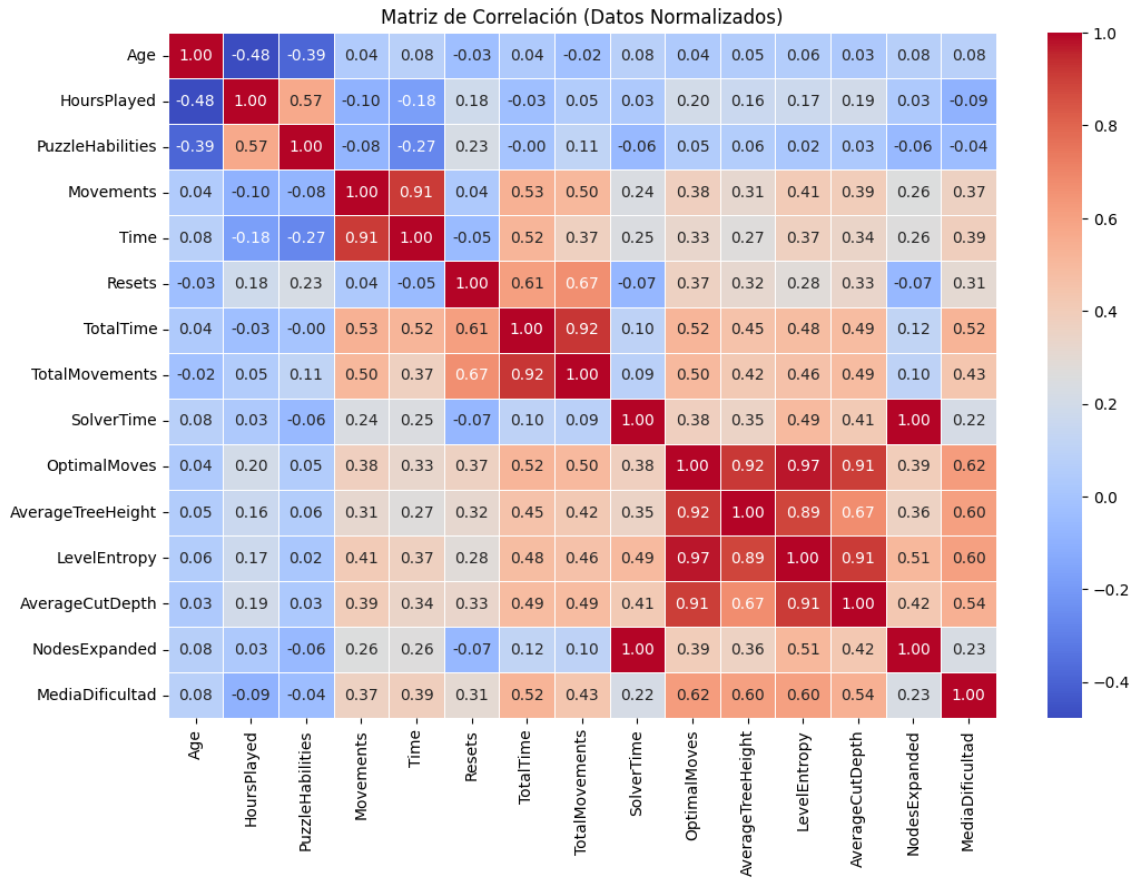


Figura 4.2: Matriz de correlación entre variables normalizadas.

4.2.3. Predicción de dificultad esperada y su incertidumbre

Dado que muchos niveles no habían sido jugados previamente, se entrenó un modelo de *Random Forest* para predecir la dificultad percibida (μ) de cada nivel en función de variables estructurales: `SolverTime`, `OptimalMoves`, `AverageTreeHeight`, `LevelEntropy`, `AverageCutDepth` y `NodesExpanded`. La salida de este modelo corresponde a la dificultad percibida por los jugadores, originalmente registrada en una escala de Likert de 1 a 7. Para mejorar la visualización e interpretación de los resultados, estos valores fueron escalados multiplicativamente al rango de 1000 a 6000.

Para estimar la incertidumbre asociada a la dificultad predicha (σ^2), se entrenó un segundo modelo que predice la varianza de percepción entre los usuarios utilizando

los mismos atributos. Esta varianza permite modelar la confianza en la estimación de dificultad para cada nivel:

$$\mu_{\text{nivel}} = f_{\text{RF}}(x_{\text{nivel}}), \quad \sigma_{\text{nivel}}^2 = g_{\text{RF}}(x_{\text{nivel}}) \quad (4.1)$$

donde f_{RF} y g_{RF} son modelos Random Forest distintos para la media y varianza respectivamente.

4.2.4. Cálculo de desempeño del jugador

A diferencia de la formulación original de TrueSkill basada solo en victoria o derrota, se incorporó una métrica continua de desempeño observada por nivel. Para ello, se definió una función de desempeño normalizado que utiliza los valores de **Time**, **Resets**, **Movements** y **OptimalMoves** del jugador, ponderados y escalados respecto al desempeño óptimo:

$$\text{score}_{\text{norm}} = 0,4 \cdot \frac{OM}{M} + 0,4 \cdot \frac{300}{T} + 0,2 \cdot \frac{R}{M} \quad (4.2)$$

donde:

- OM : movimientos óptimos.
- M : movimientos reales.
- T : tiempo usado (en segundos).
- R : número de reinicios.

Los pesos utilizados en esta función fueron asignados de manera arbitraria, pero informada a partir del análisis previo de la matriz de correlación. Dicho análisis permitió identificar qué variables presentaban mayor relación con la percepción de dificultad y el desempeño del jugador, sirviendo como guía para una asignación razonada de los coeficientes.

Este score se multiplica por la dificultad esperada del nivel para estimar el rendimiento efectivo:

$$\text{perf}_{\text{real}} = \text{score}_{\text{norm}} \cdot \mu_{\text{nivel}} \quad (4.3)$$

A partir de esta métrica, se puede cuantificar cuán cerca estuvo el jugador del desempeño esperado, permitiendo una actualización fina de su perfil.

4.2.5. Actualización bayesiana del perfil de habilidad del jugador

Cada jugador es modelado como una distribución normal $N(\mu_j, \sigma_j^2)$. Luego de cada partida, su habilidad se actualiza según el rendimiento observado frente al esperado. Primero se calcula la probabilidad de completar el nivel dada la diferencia entre habilidad y dificultad estimadas:

$$P_{\text{completar}} = \Phi \left(\frac{\mu_j - \mu_{\text{nivel}}}{\sqrt{\sigma_j^2 + \sigma_{\text{nivel}}^2 + 2\beta^2}} \right) \quad (4.4)$$

donde:

- Φ es la función de distribución acumulada (CDF) de la normal,
- β es la varianza del ruido del desempeño (hiperparámetro del sistema).

Luego se calcula la diferencia entre el resultado observado (1 si completó, 0 si no) y esta probabilidad:

$$\text{error} = \text{resultado} - P_{\text{completar}} \quad (4.5)$$

Este error se pondera por un factor tipo Kalman:

$$K = \frac{\sigma_j^2}{\sigma_j^2 + \beta^2} \quad (4.6)$$

Finalmente, se actualiza la habilidad del jugador con un término modulado por el desempeño observado:

$$\mu'_j = \mu_j + K \cdot \left(\frac{\text{perf}_{\text{real}}}{\mu_{\text{nivel}}} \right) \cdot \text{error} \cdot \beta \cdot \lambda \quad (4.7)$$

$$\sigma_j^{2'} = \sigma_j^2 \cdot (1 - K) \quad (4.8)$$

donde λ es un factor de escala ajustable que permite controlar la magnitud del cambio. Este enfoque permite capturar tanto mejoras consistentes como inconsistencias, ajustando el grado de confianza (varianza) y la habilidad estimada.

Selección del nivel adecuado

Una vez actualizada la habilidad del jugador, el sistema selecciona el siguiente nivel con base en la *match quality*, una medida que compara la habilidad del jugador con la dificultad esperada del nivel, considerando ambas incertidumbres:

$$Q = \exp \left(-\frac{(\mu_j - \mu_{\text{nivel}})^2}{2(\sigma_j^2 + \sigma_{\text{nivel}}^2 + 2\beta^2)} \right) \quad (4.9)$$

Se elige el nivel que maximiza esta calidad de emparejamiento Q , garantizando que el siguiente desafío se encuentre dentro de un rango óptimo de dificultad adaptada a la habilidad estimada.

5. Resultados

En esta sección se presentan los resultados obtenidos durante el desarrollo de esta memoria de título, enfocados en el análisis del sistema de dificultad adaptativa sobre la experiencia de los jugadores.

5.1. Descripción del experimento

El experimento fue realizado con un total de 51 participantes, quienes fueron divididos en dos grupos. El primero fue el grupo de control ($n=28$), que jugó una versión del videojuego sin dificultad adaptativa, y el segundo fue el grupo experimental ($n=23$), que jugó una versión con dificultad adaptativa, donde el uso del sistema fue dado al azar para cada uno de los participantes.

Cuestionario y constructos

Para evaluar la experiencia de los jugadores se utilizó el cuestionario Player Experience Inventory (PXI) [34], el cual mide 11 dimensiones: Facilidad de Control (*Ease of Control*), Objetivos y Reglas (*Goals and Rules*), Desafío (*Challenge*), Retroalimentación del Progreso (*Progress Feedback*), Atracción Audiovisual (*Audiovisual Appeal*), Significado (*Meaning*), Curiosidad (*Curiosity*), Maestría (*Mastery*), Inmersión (*Immersion*), Autonomía (*Autonomy*) y Disfrute (*Enjoyment*). Cada una de estas dimensiones fue evaluada mediante constructos compuestos por tres preguntas, las cuales fueron presentadas en orden aleatorio dentro del cuestionario.

Alfa de Cronbach

Antes de realizar el análisis comparativo, se calculó el coeficiente alfa de Cronbach para cada constructo del PXI, con el fin de evaluar la consistencia interna de las respuestas. Este coeficiente varía entre 0 y 1, donde valores más altos indican mayor correlación entre los ítems. Los resultados se muestran en la Tabla 5.1.

Tabla 5.1: Alfas de cronbatch de los constructos

Constructo	Elementos	Alfa de cronbatch
Audiovisual Appeal	AA_1, AA_2, AA_3	0.84
Autonomy	ATM_1, ATM_2, ATM_3	0.89
Challenge	CH_1, CH_2, CH_3	0.64
Curiosity	CUR_1, CUR_2, CUR_3	0.81
Ease of Control	EC_1, EC_2, EC_3	0.65
Enjoyment	ENJ_1, ENJ_2, ENJ_3	0.84
Goals and Rules	GR_1, GR_2, GR_3	0.77
Immersion	INM_1, INM_2, INM_3	0.79
Mastery	MT_1, MT_2, MT_3	0.84
Meaning	MNG_1, MNG_2, MNG_3	0.89
Progress Feedback	PF_1, PF_2, PF_3	0.64

Dado que los valores de *Challenge* (0.64), *Ease of Control* (0.65) y *Progress Feedback* (0.64) fueron relativamente bajos, se evaluaron combinaciones alternativas de ítems para intentar mejorar su confiabilidad interna. En los dos primeros casos, algunas combinaciones superaron el valor original, aunque de forma moderada. En *Progress Feedback*, ninguna combinación logró mejorar el alfa inicial. Los resultados detallados se muestran en la Tabla 5.2.

Tabla 5.2: Alfas de cronbatch de los constructos con bajo valor

Constructo	Elementos	Alfa de cronbatch
Challenge	CH_1, CH_2	0.47
Challenge	CH_1, CH_3	0.47
Challenge	CH_2, CH_3	0.68
Ease of Control	EC_1, EC_2	0.68
Ease of Control	EC_1, EC_3	0.53
Ease of Control	EC_2, EC_3	0.41
Progress Feedback	PF_1, PF_2	0.64
Progress Feedback	PF_1, PF_3	0.64
Progress Feedback	PF_2, PF_3	0.32

5.1.1. Comparación entre grupo experimental y control

Antes de comparar los resultados entre grupos, es necesario mencionar el uso del Test de Levene, que permite determinar si dos grupos presentan varianzas similares. Este análisis es importante porque define qué versión del t-test se debe interpretar. Si el valor de significancia (p) del Test de Levene es menor a 0.05, se asume que las varianzas son diferentes y se debe interpretar la fila que no asume igualdad de varianzas. En cambio, si p es mayor a 0.05, se considera que las varianzas son iguales y se interpreta la fila que asume igualdad de varianzas. Esta validación se aplica en las tablas de comparación que se presentan a continuación [41].

Una vez verificada la fiabilidad de los datos, se procedió a analizar los resultados obtenidos mediante la prueba t (t - test) para muestras independientes. En las tablas 5.3 y 5.4 se presentan los resultados. En ningún caso se obtuvo una diferencia estadísticamente significativa ($p > 0,05$), aunque la dimensión *Ease of control* fue la más cercana al umbral ($p = 0,139$).

En las tablas de resultados, la columna F corresponde al valor estadístico calculado por el Test de Levene, el cual compara cuánta dispersión (varianza) existe entre los grupos. Un valor de F más alto indica una mayor diferencia relativa entre las varianzas. La columna Sig. muestra el valor de significancia asociado a este test; si es menor a 0.05, se considera que las varianzas son significativamente distintas. Por último, la columna Sig. (2-tailed) corresponde al valor de p del t-test, el cual indica si existe una diferencia estadísticamente significativa entre las medias de los grupos

en la dimensión evaluada.

Tabla 5.3: Resultados simplificados del Test de Levene y t-test para igualdad de medias

Constructo	Var. Igualdad	F	Sig.	Sig. (2-tailed)
Audiovisual Appeal	Asumida	0.07	.795	.813
	No asumida			.813
Autonomy	Asumida	0.07	.791	.421
	No asumida			.420
Challenge	Asumida	0.82	.368	.692
	No asumida			.697
Curiosity	Asumida	0.01	.936	.687
	No asumida			.691
Ease of Control	Asumida	2.87	.097	.139
	No asumida			.124
Enjoyment	Asumida	0.21	.652	.342
	No asumida			.344
Goals and Rules	Asumida	1.14	.291	.422
	No asumida			.407
Immersion	Asumida	0.16	.695	.628
	No asumida			.629
Mastery	Asumida	4.46	.040	.157
	No asumida			.141
Meaning	Asumida	2.25	.140	.884
	No asumida			.888
Progress Feedback	Asumida	0.98	.326	.872
	No asumida			.872

Tabla 5.4: Estadísticos grupales por constructo para todos los jugadores

Constructo	Grupo	N	Media	Desv. Estándar	Error Estándar
Audiovisual Appeal	Exp.	23	1.41	1.38	0.29
	Control	28	1.50	1.43	0.27
Autonomy	Exp.	23	1.06	1.58	0.33
	Control	28	0.69	1.63	0.31
Challenge	Exp.	23	1.20	1.58	0.33
	Control	28	1.36	1.31	0.25
Curiosity	Exp.	23	1.49	1.42	0.30
	Control	28	1.64	1.22	0.23
Ease of Control	Exp.	23	1.80	1.08	0.23
	Control	28	1.20	1.67	0.32
Enjoyment	Exp.	23	2.35	0.91	0.19
	Control	28	2.11	0.88	0.17
Goals and Rules	Exp.	23	2.55	0.61	0.13
	Control	28	2.38	0.84	0.16
Immersion	Exp.	23	1.39	1.31	0.27
	Control	28	1.21	1.28	0.24
Mastery	Exp.	23	1.75	0.95	0.20
	Control	28	1.24	1.48	0.28
Meaning	Exp.	23	1.25	1.57	0.33
	Control	28	1.19	1.15	0.22
Progress Feedback	Exp.	23	1.54	1.04	0.22
	Control	28	1.49	1.07	0.20

5.1.2. Niveles completados

Se comparó la cantidad de niveles completados por cada grupo. La media fue mayor en el grupo experimental ($M = 9.52$) que en el grupo control ($M = 8.36$) ver tabla 5.5, y esta diferencia fue estadísticamente significativa ($p = 0.002$) ver tabla 5.6, lo que sugiere que el sistema DDA tuvo un efecto positivo en la retención.

Tabla 5.5: Promedio de niveles completados por grupo

Nombre	Grupo	N	Media	Desviación Estándar
Niveles completados	Experimental	23	9.52	.079
	Control	28	8.36	1.68

Tabla 5.6: Resultados de la prueba de Levene y t-test para niveles completados

Nombre	Var. Igualdad	F	Sig.	Sig. (2-tailed)
Niveles completados	Asumida	10.42	.002	.004
	No asumida			.002

5.1.3. Análisis en jugadores que completaron 9 o 10 niveles

Como análisis complementario, se consideró únicamente a los jugadores que completaron 9 o 10 niveles, con el objetivo de observar posibles diferencias en la experiencia reportada por quienes tuvieron una mayor exposición al sistema. Este subconjunto incluyó a 19 participantes en el grupo experimental y 14 en el grupo control.

Los resultados se presentan en la Tabla 5.7. Si bien no se encontraron diferencias estadísticamente significativas entre grupos, algunos constructos mostraron tendencias que merecen atención. En particular, Enjoyment arrojó un valor de significancia $p = 0.051$ (ver figura 5.1) bajo la suposición de igualdad de varianzas, lo que se considera marginalmente significativo y resulta especialmente interesante en el contexto de este estudio, ya que sugiere una posible diferencia en el disfrute experimentado entre ambos grupos. Además, se observó una diferencia notable en Ease of Control $p = 0.086$ (ver figura 5.2), donde el grupo expuesto al sistema adaptativo obtuvo una media más alta (1.95 vs 1.11), lo que podría indicar una mayor sensación de control durante la interacción con el juego. Por otro lado, el constructo Autonomy también mostró una diferencia cercana a la significancia $p = 0.053$ (ver figura 5.3), con una media más elevada en el grupo experimental (1.39 vs 0.38), lo cual sugiere que los participantes podrían haber percibido una mayor libertad de acción al interactuar con niveles adaptativos.

Tabla 5.7: Resultados simplificados del Test de Levene y t-test para jugadores con 9 o 10 niveles completados

Constructo	Var. Igualdad	F	Sig.	Sig. (2-tailed)
Audiovisual Appeal	Asumida	0.61	.441	.291
	No asumida			.308
Autonomy	Asumida	0.90	.351	.053
	No asumida			.068
Challenge	Asumida	2.09	.158	.816
	No asumida			.803
Curiosity	Asumida	4.55	.041	.123
	No asumida			.152
Ease of Control	Asumida	1.69	.203	.086
	No asumida			.108
Enjoyment	Asumida	0.92	.346	.051
	No asumida			.070
Goals and Rules	Asumida	0.79	.380	.551
	No asumida			.578
Immersion	Asumida	0.72	.402	.185
	No asumida			.211
Mastery	Asumida	2.42	.130	.292
	No asumida			.324
Meaning	Asumida	0.01	.943	.364
	No asumida			.384
Progress Feedback	Asumida	1.52	.227	.949
	No asumida			.950

Tabla 5.8: Estadísticos grupales por constructo para jugadores con 9 o 10 niveles completados

Constructo	Grupo	N	Media	Desv. Estándar	Error Estándar
Audiovisual Appeal	Exp.	19	1.58	1.16	0.27
	Control	14	1.10	1.42	0.38
Autonomy	Exp.	19	1.39	1.22	0.28
	Control	14	0.38	1.66	0.44
Challenge	Exp.	19	1.47	1.37	0.31
	Control	14	1.57	0.85	0.23
Curiosity	Exp.	19	1.86	0.88	0.20
	Control	14	1.24	1.37	0.37
Ease of Control	Exp.	19	1.95	1.10	0.25
	Control	14	1.11	1.62	0.43
Enjoyment	Exp.	19	2.54	0.56	0.13
	Control	14	2.05	0.85	0.23
Goals and Rules	Exp.	19	2.58	0.66	0.15
	Control	14	2.40	1.01	0.27
Immersion	Exp.	19	1.60	1.09	0.25
	Control	14	0.98	1.54	0.41
Mastery	Exp.	19	2.04	0.69	0.16
	Control	14	1.71	1.03	0.27
Meaning	Exp.	19	1.58	1.13	0.26
	Control	14	1.17	1.44	0.39
Progress Feedback	Exp.	19	1.72	0.78	0.18
	Control	14	1.74	0.90	0.24

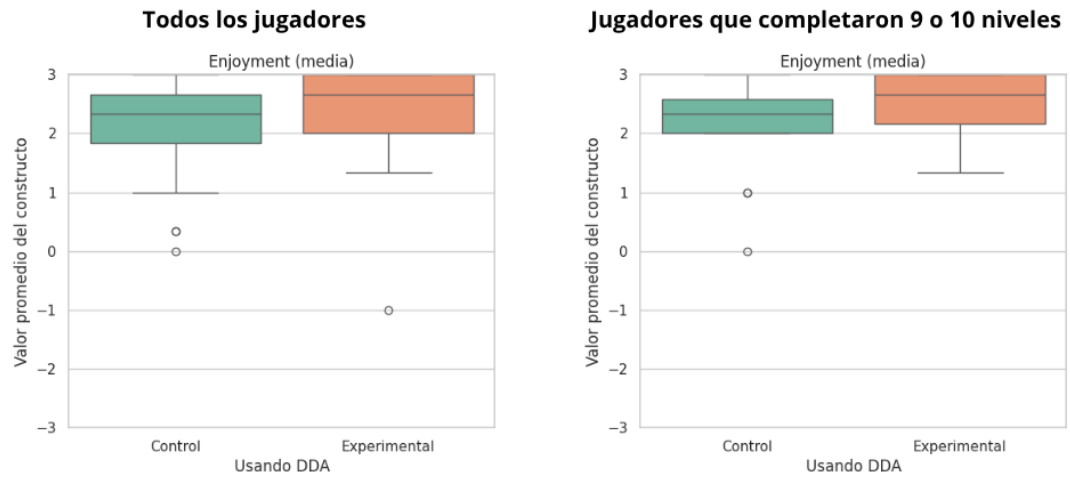


Figura 5.1: Comparación de la distribución de *Enjoyment* entre todos los jugadores y aquellos que completaron 9 o 10 niveles.

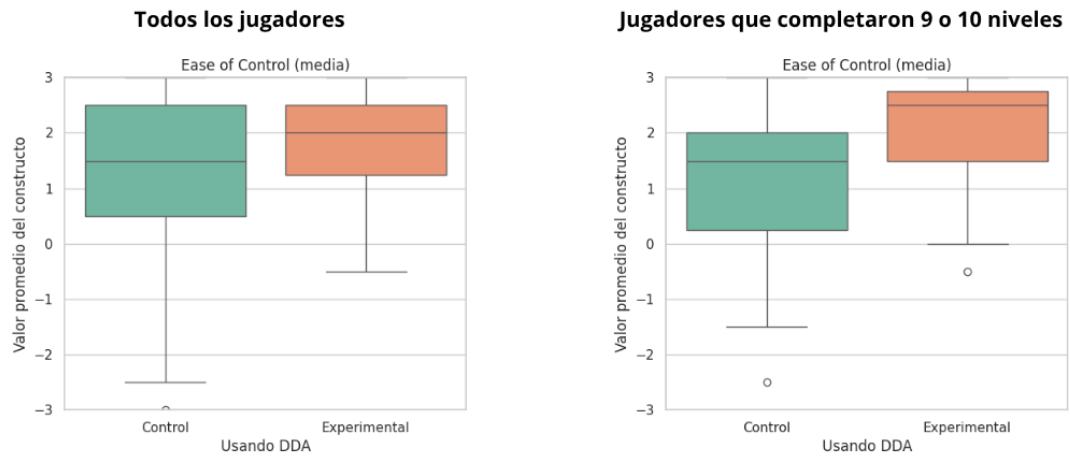


Figura 5.2: Comparación de la distribución de *Ease of control* entre todos los jugadores y aquellos que completaron 9 o 10 niveles.

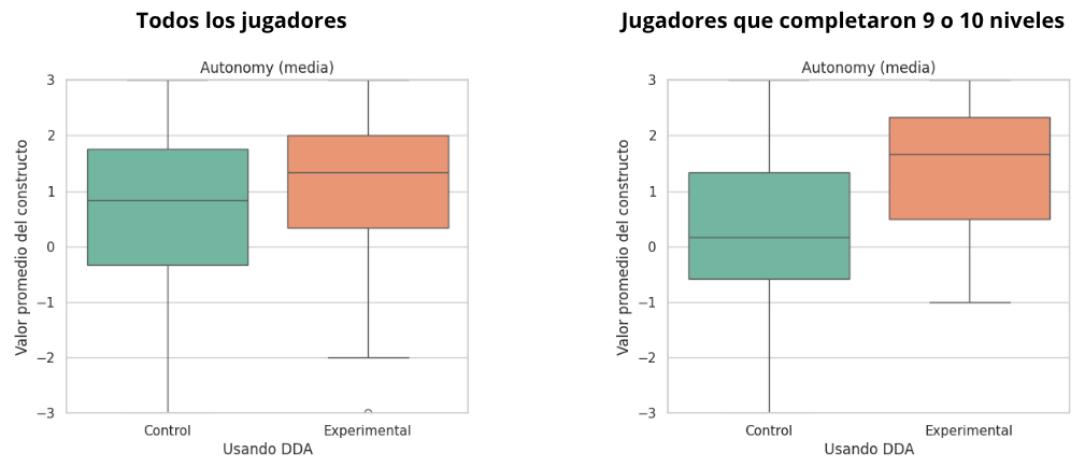


Figura 5.3: Comparación de la distribución de *Autonomy* entre todos los jugadores y aquellos que completaron 9 o 10 niveles.

6. Conclusiones

Esta memoria de título abordó el diseño, implementación y evaluación de un sistema de Ajuste Dinámico de la Dificultad (DDA) para un juego de puzzles tipo *Safari Rush Hour*. El sistema fue desarrollado utilizando métricas del nivel (como entropía, movimientos óptimos, tiempo de solución y reinicios) para predecir la dificultad percibida mediante modelos de *Random Forest*, y se complementó con una estimación de la habilidad del jugador inspirada en el modelo TrueSkill2. La implementación fue evaluada con un experimento de usuario dividido en dos grupos: uno con dificultad adaptativa y otro con dificultad estática, donde se aplicó el cuestionario PXI para medir la experiencia subjetiva de juego.

Ya que el experimento no arrojó diferencias estadísticamente significativas entre el grupo de control y el grupo experimental en la mayoría de los constructos del PXI, no se puede afirmar que la hipótesis planteada de que el sistema DDA aumentaría la sensación de maestría y satisfacción haya sido validada en su totalidad. Sin embargo, se destaca que existieron diferencias interesantes al analizar específicamente a los jugadores que completaron 9 o 10 niveles. En este subgrupo, se observó un mayor nivel de *Enjoyment* con un $p = 0.051$, lo que sugiere que el sistema de dificultad adaptativa podría tener un efecto más significativo en jugadores que alcanzan una mayor exposición al sistema y enfrentan desafíos sostenidos. Asimismo, el constructo *Autonomy* arrojó un $p = 0.053$, con una media más alta en el grupo con DDA, lo que podría reflejar una mayor sensación de libertad de acción al interactuar con niveles adaptados a sus habilidades.

Estos hallazgos abren las puertas a futuras investigaciones enfocadas en sesiones de juego más largas o con mayor cantidad de niveles para detectar efectos significativos en la experiencia de usuario.

Bibliografia

- [1] Mihaly Csikszentmihalyi. *Flow: The psychology of optimal experience*. New York: Harper & Row, 1990.
- [2] Mohammad Zohaib. Dynamic difficulty adjustment (DDA) in computer games: A review. *Advances in Human-Computer Interaction*, 2018(1):5681652, 2018.
- [3] Dagmara Dziedzic and Wojciech Włodarczyk. Approaches to measuring the difficulty of games in dynamic difficulty adjustment systems. *International Journal of Human-Computer Interaction*, 34(8):707–715, 2018.
- [4] Qian Lin. The impact of dynamic game difficulty balancing on player user experience in puzzle games : A case study. Master’s thesis, University of Skövde, School of Informatics, 2023.
- [5] Bilal Kartal, Nick Sohre, and Stephen Guy. Generating sokoban puzzle game levels with monte carlo tree search. Technical report, University Digital Conservancy, 2016.
- [6] Yahia Zakaria, Magda Fayek, and Mayada Hadhoud. Procedural level generation for sokoban via deep learning: An experimental study. *IEEE Transactions on Games*, 15(1):108–120, 2022.
- [7] Hans Schaa and Nicolas A Barriga. Generating entertaining human-like sokoban initial states. In *2021 40th International Conference of the Chilean Computer Science Society (SCCC)*, pages 1–6. IEEE, 2021.
- [8] Łukasz Spierewka, Rafał Szrajber, and Dominik Szajerman. Procedural level generation with difficulty level estimation for puzzle games. In *International Conference on Computational Science*, pages 106–119. Springer, 2021.

- [9] Jingping Liu, Lihan Chen, Sihang Jiang, Chao Wang, Sheng Zhang, Jiaqing Liang, Yanghua Xiao, and Rui Song. A crossword solving system based on monte carlo tree search. *Artificial Intelligence*, 335:104192, 2024.
- [10] Shmuel Berman, Baishakhi Ray, and Kathleen McKeown. Solving zebra puzzles using constraint-guided multi-agent systems. *arXiv preprint arXiv:2407.03956*, 2024.
- [11] Harsh Bhasin and Neha Singla. Genetic based algorithm for n-puzzle problem. *International Journal of Computer Applications*, 51(22), 2012.
- [12] Chang Liu, Shunqi Huang, Mohd Nor Akmal Khalid, and Hiroyuki Iida. Attractiveness of single-agent game: case study using sliding puzzle. In *2020 International Conference on Advanced Information Technologies (ICAIT)*, pages 76–81. IEEE, 2020.
- [13] Eugene You Chen Chen, Adam White, and Nathan R Sturtevant. Entropy as a measure of puzzle difficulty. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, pages 34–42, 2023.
- [14] Jeppe Theiss Kristensen, Arturo Valdivia, and Paolo Burelli. Statistical modelling of level difficulty in puzzle games. In *2021 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2021.
- [15] Jeppe Theiss Kristensen and Paolo Burelli. Difficulty modelling in mobile puzzle games: An empirical study on different methods to combine player analytics and simulated data. *International Journal of Computer Games Technology*, 2024(1):5592373, 2024.
- [16] Matthew Stephenson and Jochen Renz. Agent-based adaptive level generation for dynamic difficulty adjustment in angry birds. *arXiv preprint arXiv:1902.02518*, 2019.
- [17] Rubem Jose Vasconcelos De Medeiros and Tacio Filipe Vasconcelos De Medeiros. Procedural level balancing in runner games. In *2014 Brazilian Symposium on Computer Games and Digital Entertainment*, pages 109–114. IEEE, 2014.
- [18] Marc Van Kreveld, Maarten Löffler, and Paul Mutser. Automated puzzle difficulty estimation. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 415–422. IEEE, 2015.

- [19] Gabriel K Sepulveda, Felipe Besoain, and Nicolas A Barriga. Exploring dynamic difficulty adjustment in videogames. In *2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, pages 1–6. IEEE, 2019.
- [20] Fatemeh Mortazavi, Hadi Moradi, and Abdol-Hossein Vahabie. Dynamic difficulty adjustment approaches in video games: a systematic literature review. *Multimedia Tools and Applications*, pages 1–48, 2024.
- [21] Ligia Jácome-Amores, Wimper Rivera Freire, and Richard Sánchez Sánchez. Gamification as an educational strategy to strengthen cognitive abilities of mathematics in school children. In *Innovation and Research: A Driving Force for Socio-Economic-Technological Development 1st*, pages 142–150. Springer, 2021.
- [22] Diana Sofía Lora-Ariza, Antonio A Sánchez-Ruiz, Pedro Antonio González-Calero, and Irene Camps-Ortueta. Measuring control to dynamically induce flow in tetris. *IEEE Transactions on Games*, 14(4):579–588, 2022.
- [23] Kuan Liang Tan, Chin Hiong Tan, Kay Chen Tan, and Arthur Tay. Adaptive game ai for gomoku. In *2009 4th International Conference on Autonomous Robots and Agents*, pages 507–512. IEEE, 2009.
- [24] Carlos Lara-Alvarez, Hugo Mitre-Hernandez, Juan J Flores, and Humberto Pérez-Espinosa. Induction of emotional states in educational video games through a fuzzy control system. *IEEE Transactions on Affective Computing*, 12(1):66–77, 2018.
- [25] Marcelo Zamith, Jose Ricardo da Silva Junior, Esteban WG Clua, and Mark Joselli. Applying hidden markov model for dynamic game balancing. In *2020 19th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 38–46. IEEE, 2020.
- [26] Elinga Pagalyte, Maurizio Mancini, and Laura Climent. Go with the flow: Reinforcement learning in turn-based battle video games. In *Proceedings of the 20th ACM international conference on intelligent virtual agents*, pages 1–8, 2020.
- [27] William Rao Fernandes and Guillaume Levieux. -logit: Dynamic difficulty adjustment using few data points. In *Joint International Conference on Entertainment Computing and Serious Games*, pages 158–171. Springer, 2019.

- [28] Xinyu Li, Suoju He, Yue Dong, Qing Liu, Xiao Liu, Yiwen Fu, Zhiyuan Shi, and Wan Huang. To create dda by the approach of ann from uct-created data. In *2010 international Conference on computer application and system modeling (ICCASM 2010)*, volume 8, pages V8–475. IEEE, 2010.
- [29] David Buckley, Ke Chen, and Joshua Knowles. Rapid skill capture in a first-person shooter. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(1):63–75, 2015.
- [30] Jorge González Alonso and Mauro Pazmiño Santacruz. Cálculo e interpretación del alfa de cronbach para el caso de validación de la consistencia interna de un cuestionario, con dos posibles escalas tipo likert. *Revista publicando*, 2(1):62–67, 2015.
- [31] Timothy O Hodson. Root mean square error (rmse) or mean absolute error (mae): When to use them or not. *Geoscientific Model Development Discussions*, 2022:1–10, 2022.
- [32] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill™: a bayesian skill rating system. *Advances in neural information processing systems*, 19, 2006.
- [33] Tom Minka, Ryan Cleven, and Yordan Zaykov. Trueskill 2: An improved bayesian skill rating system. Technical report, Microsoft Research, 2018.
- [34] Vero Vanden Abeele, Katta Spiel, Lennart Nacke, Daniel Johnson, and Kathrin Gerling. Development and validation of the player experience inventory: A scale to measure player experiences at the level of functional and psychosocial consequences. *International Journal of Human-Computer Studies*, 135:102370, 2020.
- [35] Roderick Lottering, Robert Hans, and Manoj Lall. The impact of crossword puzzles on students’ performance: Does pre-exposure to puzzles matter? In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pages 545–550. IEEE, 2018.
- [36] Kristen Lucas and John L Sherry. Sex differences in video game play: A communication-based explanation. *Communication research*, 31(5):499–523, 2004.
- [37] Elio Valenzuelab, Hans Schaab, Nicolas A Barrigab, and Gustavo Patow. Using search algorithm statistics for predicting maze and puzzle difficulty. 2024.

- [38] David Buckley, Ke Chen, and Joshua Knowles. Predicting skill from gameplay input to a first-person shooter. In *2013 IEEE Conference on Computational Intelligence in Games (CIG)*, pages 1–8. IEEE, 2013.
- [39] Effie L-C Law, Florian Brühlmann, and Elisa D Mekler. Systematic review and validation of the game experience questionnaire (geq)-implications for citation and reporting practice. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*, pages 257–270, 2018.
- [40] Sebastian AC Perrig, Nicolas Scharowski, Florian Brühlmann, Nick von Felten, Klaus Opwis, and Lena Fanya Aeschbach. Independent validation of the player experience inventory: Findings from a large set of video game players. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2024.
- [41] Howard Levene. Robust tests for equality of variances. In *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*, pages 278–292. Stanford University Press, 1960.

A. Cuestionario player experience inventory (PXI)

Este apéndice presenta todas las preguntas del cuestionario Player Experience Inventory (PXI) que fueron respondidas por los participantes del estudio. Es importante señalar que los ítems fueron mostrados en un orden aleatorio durante la aplicación del instrumento, con el fin de reducir posibles sesgos en las respuestas.

Cuestionario Player Experience Inventory (PXI)

Audiovisual Appeal (AA)

- AA_1: Disfruté la forma en que se diseñó el juego.
- AA_2: Me gustó la apariencia y sensación del juego.
- AA_3: Aprecié la estética del juego.

Autonomy (ATM)

- ATM_1: Me sentí libre de jugar el juego a mi manera.
- ATM_2: Sentí que tenía opciones sobre cómo quería jugar este juego.
- ATM_3: Tuve una sensación de libertad sobre cómo quería jugar este juego.

Challenge (CH)

- CH_1: El juego no fue demasiado fácil ni demasiado difícil de jugar.

- **CH_2:** El juego fue desafiante pero no demasiado desafiante.
- **CH_3:** Los desafíos del juego tenían el nivel de dificultad adecuado para mí.

Curiosity (CUR)

- **CUR_1:** Quería explorar cómo el juego evolucionaba.
- **CUR_2:** Quería descubrir cómo el juego progresaba.
- **CUR_3:** Me sentí ansioso por descubrir cómo continuaba el juego.

Ease of Control (EC)

- **EC_1:** Fue fácil saber cómo realizar acciones en el juego.
- **EC_2:** Las acciones para controlar el juego estaban claras para mí.
- **EC_3:** Pensé que el juego era fácil de controlar.

Enjoyment (ENJ)

- **ENJ_1:** Me gustó jugar el juego.
- **ENJ_2:** El juego fue entretenido.
- **ENJ_3:** Me lo pasé muy bien jugando a este juego.

Goals and Rules (GR)

- **GR_1:** Comprendí el objetivo general del juego.
- **GR_2:** Las metas del juego estaban claras para mí.
- **GR_3:** Entendí los objetivos del juego.

Immersion (INM)

- **INM_1:** Ya no era consciente de lo que me rodeaba mientras jugaba.
- **INM_2:** Estaba inmerso en el juego.
- **INM_3:** Estaba completamente enfocado en el juego.

Meaning (MNG)

- MNG_1: Jugar el juego fue significativo para mí.
- MNG_2: El juego se sintió relevante para mí.
- MNG_3: Jugar este juego fue valioso para mí.

Mastery (MT)

- MT_1: Sentí que era bueno jugando este juego.
- MT_2: Me sentí capaz mientras jugaba el juego.
- MT_3: Tuve una sensación de maestría jugando este juego.

Progress Feedback (PF)

- PF_1: El juego me informó de mi progreso en el juego.
- PF_2: Pude evaluar fácilmente cómo me estaba desempeñando en el juego.
- PF_3: El juego brindó información clara sobre mi progreso hacia los objetivos.

B. Selección de niveles

En este anexo se presenta la lógica utilizada para la selección de niveles en el sistema de dificultad adaptativa. Esta selección depende de si el sistema está utilizando o no el ajuste dinámico de dificultad (DDA). Cuando DDA está desactivado, los niveles son seleccionados desde un conjunto predefinido agrupados por bloques. En cambio, cuando DDA está activado, se recomienda el nivel que mejor se ajusta al perfil de habilidad del jugador.

Ambas funciones se encargan de retornar un nivel no jugado por el jugador. En el caso de DDA desactivado, el nivel se elige aleatoriamente desde un conjunto acotado por índice. En el caso activado, se utiliza una fórmula basada en la similitud entre la dificultad esperada del nivel y la habilidad del jugador para encontrar el nivel más adecuado.

Listing B.1: Actualizar la habilidad del jugador

```
public void UpdateSkill(string playerID, string
    levelID, bool wasCompleted, float time, int resets,
    int movements, int optimalMoves)
{

    levelsPlayedIds.Add(levelID);
    float beta = 300f;

    LevelData_csv level = levels.Find(l => l.LevelID ==
        levelID);
    if (level == null)
    {
        return;
```

```

    }

    if (!playerSkills.ContainsKey(playerID))
        playerSkills[playerID] = new PlayerSkill();

    PlayerSkill skill = playerSkills[playerID];
    float muPlayer = skill.Mu;
    float varPlayer = skill.Sigma2;

    float muLevel = level.ExpectedDifficulty;
    float varLevel = level.ExpectedVariance;

    double denom = Math.Sqrt(varPlayer + varLevel + 2 *
        beta * beta);
    double z = (muPlayer - muLevel) / denom;
    double pCompletar = 0.5 * (1.0 + Erf(z / Math.Sqrt(2)))
        ); // (z)

    float resultado = wasCompleted ? 1f : 0f;
    float prediction = resultado - (float)pCompletar;

    float perfReal = CalculatePerformance(time, resets,
        movements, optimalMoves, muLevel);
    float preformance = perfReal / muLevel;

    float K = varPlayer / (varPlayer + beta * beta);
    float deltaMu = K * preformance * prediction * beta *
        300f;

    skill.Mu += deltaMu;
    skill.Sigma2 *= (1f - K);

    if (!usingDDA && wasCompleted && fixedLevelPaths.
        ContainsKey(fixedLevelIndex) && fixedLevelPaths[
            fixedLevelIndex].Contains(levelID))

```

```
{
    fixedLevelIndex++;
}

}

public static double Erf(double x)
{
    double sign = Math.Sign(x);
    x = Math.Abs(x);

    double a1 = 0.254829592;
    double a2 = -0.284496736;
    double a3 = 1.421413741;
    double a4 = -1.453152027;
    double a5 = 1.061405429;
    double p = 0.3275911;

    double t = 1.0 / (1.0 + p * x);
    double y = 1.0 - (((((a5 * t + a4) * t + a3) * t + a2)
        * t + a1) * t) * Math.Exp(-x * x);

    return sign * y;
}

public float CalculatePerformance(float time, int resets,
    int movements, int optimalMoves, float
    dificultadEsperada)
{
    float movScore = 1f;
    float timeScore = 1f;
    float resetScore = 1f;

    if (optimalMoves > 0 && movements > 0)
        movScore = Mathf.Clamp01((float)optimalMoves /
            movements);
}
```

```
    if (time > 0)
        timeScore = Mathf.Clamp01(300f / time);

    if (resets > 0 && movements > 0)
        resetScore = Mathf.Clamp01((float)resets /
            movements);

    float scoreNormalizado = 0.4f * movScore + 0.4f *
        timeScore + 0.2f * resetScore;

    return scoreNormalizado * dificultadEsperada;
}
```

Listing B.2: Elegir el nivel adecuado

```
public LevelData GetRecomendedlevel()
{
    if (usingDDA)
    {
        return GetRecomendedlevelDDA();
    }
    else
    {
        return GetRecomendedlevelFixed();
    }
}

private LevelData GetRecomendedlevelDDA()
{
    if (!playerSkills.ContainsKey(playerID)) return null;

    PlayerSkill skill = playerSkills[playerID];
    float bestScore = float.MinValue;
    LevelData_csv recommendedLevel = null;

    string notPlayedLevelsString = "Niveles no jugados:";
    var notPlayedLevels = new List<LevelData_csv>();
}
```

```

        foreach (var level in levels)
        {
            if (!levelsPlayedIds.Contains(level.LevelID))
            {
                notPlayedLevels.Add(level);
                notPlayedLevelsString += $" {level.LevelID},";
            }
        }

        foreach (var level in notPlayedLevels)
        {
            float sigma2 = skill.Sigma2 + level.
                ExpectedVariance + 2f * beta * beta;
            float q = Mathf.Exp(-Mathf.Pow(skill.Mu - level.
                ExpectedDifficulty, 2) / (2f * sigma2));

            if (q > bestScore)
            {
                bestScore = q;
                recomendLevel = nivel;
            }
        }

        return recomendLevel;
    }

    private LevelData GetRecomendedlevelFixed()
    {
        if (fixedLevelPaths.ContainsKey(fixedLevelIndex))
        {
            List<string> levelIDs = fixedLevelPaths[
                fixedLevelIndex];
            if (levelIDs.Count > 0)
            {
                string levelID = levelIDs[UnityEngine.Random.
                    Range(0, levelIDs.Count)];
            }
        }
    }

```

```
        LevelData_csv level = levels.Find(l => l.
            LevelID == levelID);
        if (level != null)
        {
            return level.TransformToLevelData();
        }
    }
    return null;
}
```