Efecto del algoritmo de ajuste dinámico de dificultad en el compromiso y desempeño de los usuarios de un juego en realidad aumentada de disparo en primera persona

Ricardo Campo Cerón
Gilber Andrés Villaquiran Silva

Director: Carlos Felipe Rengifo Rodas, Ph.D
Codirector: Diego Enrique Guzmán Villamarín, Ph.D(c)

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Ingeniería en Automática Industrial
Popayán
2023

Efecto del algoritmo de ajuste dinámico de dificultad en el compromiso y desempeño de los usuarios de un juego en realidad aumentada de disparo en primera persona

Ricardo Campo Cerón
Gilber Andrés Villaquiran Silva

Trabajo de grado presentado a la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca para la obtención del título de

Ingeniero en:
Automática Industrial

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Ingeniería en Automática Industrial
Popayán
2023

Agradecimientos

... A Dios por cuidarnos, iluminarnos y brindarnos fortaleza a lo largo de nuestra trayectoria académica, tanto en los momentos difíciles como en los momentos de alegría.

... A nuestros padres, quienes con su amor, acompañamiento constante, apoyo incondicional y dedicación a nuestras vidas, nos brindaron la oportunidad, a través de su esfuerzo, de acceder a una educación de alta calidad y nos guiaron en cada paso que dimos.

... A nuestros hermanos por su constante acompañamiento, atención y ánimo en cada paso que dimos para llegar hasta aquí.

... A nuestros queridos abuelos, por su amor incondicional, sus sabios consejos y por estar siempre a nuestro lado.

... Al director de nuestra tesis, el Ph.D Carlos Felipe Rengifo Rodas, cuyos conocimientos y experiencia enriquecieron y guiaron con compromiso nuestro camino durante la realización de esta investigación.

... Al codirector de nuestra tesis, el Ph.D(c) Diego Enrique Guzmán Villamarín, por enriquecer nuestra comprensión en nuevas temáticas, por sus correcciones y su apoyo constante a lo largo de toda la investigación.

... A la Ph.D Mariela Muñoz Añasco por guiarnos con sus conocimientos en el modelado del comportamiento del juego a través de autómatas de estado finito y redes de Petri.

... A los miembros del grupo de semillero AR/VR, así como a los estudiantes del programa de Ingeniería en Automática Industrial y a todas las personas que, con su participación, voluntad y gran disposición, colaboraron en las pruebas, encuestas y uso del software propuestos para la realización de esta investigación.

... A todos los docentes de nuestra alma máter, quienes a lo largo de nuestra carrera han brindado valiosos conocimientos que enriquecen tanto nuestra vida profesional como personal, demostrando un constante esfuerzo y compromiso.

... A nuestra alma máter, la Universidad del Cauca, por proporcionarnos educación de alta calidad y por su constante lucha en defensa de nuestros derechos.

Resumen

En este trabajo de grado se analiza el efecto del algoritmo de Ajuste Dinámico de Dificultad en el compromiso y desempeño de los usuarios en un juego en realidad aumentada de disparo en primera persona. Esta investigación aporta al campo de estudio del control al aprovechar las tecnologías de la transformación digital para crear un entorno que combine realidad aumentada e inteligencia artificial, con el propósito de mejorar la aceptación y adaptabilidad de los usuarios a estas herramientas.

Se desarrolla el juego *Invasion Victory* para dispositivos *Android* con versión 8.0 en adelante y compatible con *ARCore*. El juego se presenta en dos versiones: una con ajuste manual de la dificultad para recopilar datos y entrenar algoritmos (*Random Forest*, *Support Vector Machine* y *Artificial Neural Network*). La segunda versión integra el algoritmo de ajuste para adaptar la dificultad a las habilidades de los jugadores.

Se recopilan datos adicionales de la segunda versión para analizar el efecto del algoritmo en el desempeño y motivación de los usuarios. Los resultados demuestran que los usuarios que interactúan con el algoritmo *Artificial Neural Network* muestran niveles más altos de compromiso. En términos de desempeño, el algoritmo *Artificial Neural Network* presenta diferencias significativas en comparación con los algoritmos *Random Forest* y *Support Vector Machine*; sin embargo, no se observan diferencias entre estos dos últimos. Respecto a la usabilidad del juego, no se aprecian variaciones notables en función del algoritmo empleado.

Palabras clave: ajuste dinámico de dificultad, realidad aumentada, algoritmo basado en aprendizaje estadístico, desempeño, compromiso, usabilidad.

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Capítulo 1

Generalidades

1.1. Planteamiento del Problema

El ajuste dinámico de dificultad (ADD) es un método que adapta la evolución de los juegos a las habilidades individuales del jugador, con el fin de evitar la frustración causada por un excesivo nivel de dificultad o el aburrimiento provocado por la ausencia de reto. El ADD mejora notoriamente la experiencia del jugador si lo mantiene dentro de lo que Nakamura y Csikszentmihalyi denominan ”canal de flujo” [1], en el cual el reto está en concordancia con el desafío. Además de los juegos, tanto serios como lúdicos, el ADD también ha sido utilizado para mejorar procesos de aprendizaje [2], incrementar la motivación de los pacientes para realizar sus terapias de rehabilitación [3], y promover estilos de vida saludables a través de la actividad física [4]. Estos ADDs, de acuerdo con Zohaib y colaboradores [5], utiliza principios tan diversos como los algoritmos probabilísticos, perceptrones de una y múltiples capas, secuencias dinámicas de comandos, sistema de aldea, aprendizaje reforzado, límites de confianza superiores para árboles y redes neuronales artificiales, y sistemas autoorganizativos con redes neuronales artificiales.

El ADD está estrechamente relacionado con la teoría de control realimentado, en la cual las variables de salida responden a los cambios efectuados en las variables de entrada, y estas a su vez son recalculadas por el controlador con base en la diferencia entre las variables medidas y sus valores deseados. En el caso de los juegos, se utilizan como variables de respuesta la destreza, el nivel de aprendizaje, las emociones del jugador y su percepción de dificultad [5]. Estas variables de salida se clasifican en fisiológicas y de desempeño. Entre las primeras, se pueden encontrar el ritmo cardiaco [6, 7], las expresiones faciales [8, 9], los movimientos corporales [10, 4] o la fuerza ejercida por los dedos [11], entre muchas otras. En cuanto a las variables de desempeño, frecuentemente se utilizan puntajes, tiempo requerido para culminar una tarea o número de vidas con que se finalizó el juego. Entre las variables manipuladas, las cuales definen el nivel de desafió, se tienen la velocidad, número y frecuencia de aparición de los adversarios [8, 12, 13], así como de cualquier otro elemento del juego que impacte en su dificultad.

El principal reto cuando se implementa un ADD consiste en seleccionar un algoritmo que mantenga a jugadores con diferentes niveles de destreza dentro del canal de flujo propio de cada uno. Para tal fin, se utilizan técnicas como aprendizaje por refuerzo [14], algoritmos evolutivos [15, 16, 17], métodos heurísticos [18, 9, 19, 20, 21], métodos probabilísticos [22, 2], y redes neuronales profundas [23, 24, 25, 26]; sin embargo, las ventajas de estos algoritmos se han sustentado en comparaciones con respecto a una versión del mismo juego en la que no se cuenta con ADD. Lo anterior ha generado un vacío de conocimiento en cuanto a criterios para la selección de algoritmos, dado que los enfoques anteriormente mencionados no han sido comparados entre sí.

Como consecuencia de lo anterior, se plantea evaluar el efecto del algoritmo de ajuste dinámico de dificultad en el compromiso y desempeño de los usuarios de un juego de realidad aumentada de disparo en primera persona. Para cumplir con este propósito, se partirá de la pregunta de investigación: ¿Genera el algoritmo de ADD una diferencia significativa en el desempeño y la motivación de los usuarios de un juego de realidad aumentada de disparo en primera persona? Esta pregunta conlleva a las siguientes hipótesis:

- *Hipótesis nula*: El algoritmo utilizado para el ADD no tiene un efecto significativo en el desempeño y la motivación de los usuarios de un juego de realidad aumentada de disparo en primera persona.
- *Hipótesis de investigación*: El algoritmo utilizado para el ADD tiene un efecto significativo en el desempeño y la motivación de los usuarios de un juego de realidad aumentada de disparo en primera persona.

1.2. Estado del Arte

Con el presente estado del arte, se busca dar respuesta a los siguientes interrogantes: ¿En qué juegos se ha implementado ADD?, ¿Qué mecanismos de ADD se han usado en estos juegos?, y ¿Qué comparaciones de juegos con ADD existen en la literatura? La búsqueda de referencias bibliográficas se realizó en las bases de datos *Scopus, Science Direct, SpringerLink* y *Sage Journals*. En total, se encontraron 44 artículos, de los cuales se excluyeron 12 que no estaban disponibles para descarga o que no contenían información relevante para este trabajo. Las investigaciones restantes fueron agrupadas en las siguientes categorías: ”Juegos con implementación de ADD”, la cual cuenta con 12 referencias, ”algoritmos empleados en juegos con ADD”, con 11 referencias, y ”comparación entre algoritmos utilizados para el ADD”, la cual comprende solo tres referencias.

1.2.1. Juegos con ADD

Entre los juegos que utilizan ADD, prevalecen los lúdicos sobre los serios. Un ejemplo de ello lo presentan Morosan y Poli [12], quienes adicionan un ADD al juego *Ms PacMan*. En este juego, el jugador debe guiar al personaje *Ms PacMan* a través de un laberinto 2D y obtener la mayor cantidad de puntos consumiendo elementos con forma de fruta, mientras evade a un grupo de fantasmas. Estos fantasmas, al entrar en contacto con *Ms PacMan*, disminuyen su número de vidas. Sin embargo, *Ms PacMan* puede eliminar a los fantasmas si consume una píldora energética y los choca. Empleando un ADD basado en un algoritmo genético que determina la velocidad con que se desplaza el jugador, así como las velocidades de acercamiento y huida de los fantasmas.

Nugraha y Chowanda [8] presentan un juego de horror y supervivencia denominado *Five Nights at Freddy's*, en el que el jugador debe permanecer dentro de la sala de seguridad de un edificio embrujado. La sala cuenta con dos puertas y dos lámparas que funcionan con energía eléctrica, cuyo uso debe ser administrado por el jugador. Este debe verificar con las lámparas si un fantasma se aproxima a la puerta de la sala, y de ser así, debe cerrarla; de lo contrario, el fantasma podrá entrar y asustar al jugador, caso en el cual el juego habrá terminado. Si el jugador sobrevive durante cinco minutos sin ser asustado, será proclamado como vencedor. En caso de que el jugador se quede sin energía eléctrica, las puertas permanecerán abiertas y el jugador podrá ser asustado por los fantasmas. En este trabajo realizaron un experimento en el cual intervinieron 31 participantes, quienes utilizaron dos versiones del juego: una con ADD y otra sin ADD. El juego con ADD modifica la frecuencia de aparición de las figuras aterradoras en función de las expresiones faciales del jugador. La comparación arrojó que el juego con ajuste dinámico mostró una mejora en el rendimiento del jugador y, además, le brindó una mayor satisfacción.

En la actualidad, el uso de ADD se ha expandido a los diferentes géneros de juegos; prueba de ello es su inclusión en los *exergames*, los cuales buscan mejorar el estilo de vida de las personas a través de la actividad física. Schwarz et al. [10] presentan *SmartLife*, un *exergame* para dispositivos móviles enfocado en la vida de los adolescentes. El juego se desarrolla en un mundo post-apocalíptico, donde la fuente de poder del jugador es su traje protector, el cual se carga al realizar actividades físicas y se utiliza para proteger el búnker donde habita el jugador. Este juego requiere que el usuario mantenga en movimiento la parte inferior de su cuerpo para continuar. Para la implementación del ADD, el juego adapta los niveles de actividad física requeridos en función de los valores entregados por un acelerómetro incorporado a la vestimenta del jugador y que se conecta a un teléfono inteligente a través de *Bluetooth*. Si el sensor detecta que el jugador camina lento, la narrativa se adecua y requerirá que el jugador se mueva con mayor intensidad. Al igual que en [8], se implementaron dos versiones del juego: una con ADD y otra sin ADD. En cuanto a condiciones de flujo, inmersión y disfrute, no se encontraron diferencias significativas; en

contraste, los jugadores sin ADD reflejaron mayor compromiso al jugar más tiempo y completar más misiones que el grupo que utilizó ADD.

Otros estudios que utilizan ADD en exergames son Ramasamy et al. [4], quienes desarrollan *Ski for Squat*, en el cual se requiere que el jugador haga sentadillas para mantener la fuerza en sus extremidades inferiores, y Huber et al. [13], donde el jugador debe atravesar un laberinto que cuenta con varias salas de ejercicios.

Por otra parte, el ADD también es utilizado para el aprendizaje y la rehabilitación mediante juegos serios, como el propuesto por Ninaus et al. [6]. En este juego de emergencia, se disponen de tres escenarios: accidente de tráfico, incendio en un edificio y choque de trenes. En estos, el jugador debe coordinar eficientemente al personal de emergencia, que consta de paramédicos, bomberos y ambulancias. En el escenario de accidente de tráfico, el personal de emergencia debe extraer a las víctimas de los autos, atenderlos en el lugar de los hechos y posteriormente llevarlos a un hospital. En el escenario de incendio en un edificio, el personal de emergencia debe evacuar el edificio en llamas, apagar las llamas para evitar su propagación, aplicar primeros auxilios a los heridos y eventualmente transportarlos a un hospital. En el escenario de choque de trenes, se debe extraer a las víctimas del siniestro, proporcionarles primeros auxilios para luego trasladarlos a un hospital y apagar los incendios causados por el choque. En este juego, tanto los transeúntes como el personal de emergencia pueden resultar heridos, y en caso de perecer las víctimas, deben ser transportadas igualmente a un hospital. Para este estudio, se compararon dos versiones del juego: una con ADD y otra sin ADD. El ADD se encarga de aumentar o disminuir la cantidad de eventos en cada escenario, volviendo el juego más o menos difícil de acuerdo a la frecuencia cardíaca del jugador. Los resultados señalan que la aplicación del ADD modificó el porcentaje de finalización de cada escenario; además, los usuarios percibieron que la versión adaptable tenía la dificultad correcta para ellos en comparación con la versión no adaptable, la cual encontraron más difícil.

En el contexto de la salud, Peng et al. [11] presentan un juego para el entrenamiento de la atención viso-háptica, denominado *FF-Dancing*. Este juego combina el control de la fuerza en la punta de los dedos índice y medio de ambas manos con una pantalla visual inmersiva. Toda la información visual es proporcionada por un *head-mounted display* (*HMD*). En el entorno virtual se muestran cuatro cilindros semitransparentes; en la base de cada cilindro hay un disco de diferente color, los cuales pueden moverse en forma vertical dentro del cilindro si la fuerza aplicada al sensor de fuerza excede un umbral predefinido. El objetivo del juego consiste en que el disco alcance una altura deseada. El ADD modifica la frecuencia de los eventos y el tiempo de duración para lograr una tasa de éxito constante del 75 % con una fluctuación del $\pm$ 5 %. Para las pruebas de validación se incluyeron 24 participantes, quienes fueron asignados en igual número al grupo de prueba y al grupo de control. Cada integrante de ambos grupos realizó 1200 ensayos, divididos en tres

sesiones de 400 ensayos con un descanso de cinco minutos entre cada sesión. Los resultados confirmaron que el modelo propuesto es capaz de mantener la tasa de éxito esperada, mientras que la tasa promedio de los participantes en el grupo de control fluctuó entre el 65 % y el 80 %.

También se han encontrado otros juegos serios en donde se utiliza el ADD como mecanismo para potenciar la experiencia del usuario. Gutiérrez et al. [27] proponen el juego *Music Therapy Brain Training*, cuyo objetivo es estimular la memoria a corto plazo. Atorf et al. [28] hacen uso del juego *Lost Earth 2307*, donde el jugador tiene que pasar de una escena a la siguiente y encontrar varios objetos ocultos. Kamkuimo et al. [7] implementan un simulador de conducción de camiones dedicado a las personas que sufren de trastorno de estrés pos-traumático. Fleming et al. [29] desarrollan un videojuego de conducción de ritmo rápido, en donde se busca que las experiencias de aprendizaje coincidan dinámicamente con la habilidad del alumno. Amiri et al. [30] presentan el juego *StepAR* para la rehabilitación de la marcha en personas con esclerosis múltiple.

Es posible observar que, en la búsqueda de reducir la frustración causada por una dificultad excesiva o el aburrimiento provocado por la ausencia del reto, los diseñadores de juegos han encontrado en el ADD una poderosa herramienta que permite adaptar el juego a las necesidades del jugador y mejorar la experiencia de juego. Esto ha llamado la atención de otras áreas como la salud, la rehabilitación, la educación y la actividad física, que encuentran en el ADD una alternativa viable para beneficiar, atraer e involucrar a los pacientes, aprendices y usuarios en general. Asimismo, mediante la revisión de estos artículos, fue posible establecer que el ADD es aplicado a una gran variedad de géneros de juegos sin presentar ningún tipo de inconveniente.

1.2.2. Algoritmos empleados en juegos con ADD

Al indagar sobre los algoritmos de ADD aplicados a juegos, se encontraron diferentes investigaciones. Cardia et al. [14] presentan un juego en 2D cuyo objetivo es que el usuario estalle los globos que aparecen en la parte inferior de la pantalla y ascienden para posteriormente desaparecer. El jugador debe estallar los globos con la representación virtual de sus manos, cuya posición es capturada por el dispositivo *Leap Motion*. En este juego, se implementa el ADD a través del algoritmo de aprendizaje por refuerzo denominado *Q-Learning*. Las variables de desempeño en este juego son el número de aciertos, el número de globos no alcanzados y el total de globos. La dificultad del juego se modifica a través de la cantidad de globos y de su velocidad de ascenso. Para evaluar la usabilidad de este juego, se utiliza el cuestionario *System Usability Scale* (SUS), donde las puntuaciones inferiores a 60 indican que el usuario estuvo insatisfecho, mientras que las puntuaciones superiores a 80 indican un alto nivel de satisfacción. Los resultados muestran un promedio de 74 puntos, lo que evidencia que el ambiente es usable y satisfactorio para el jugador.

Por su parte, García et al. [15] diseñan un algoritmo evolutivo (EA) para juegos similares a *Whac-a-Mole*. Este juego tiene como propósito la rehabilitación motora de la mano y su objetivo es alcanzar con un martillo a un objeto que va apareciendo aleatoriamente durante un determinado tiempo. Se manejan cinco perfiles de jugadores para simular diferentes habilidades: (1) alto rendimiento horizontal pero bajo a nivel vertical, (2) alto rendimiento de movimiento vertical pero bajo a nivel horizontal, (3) bajo rendimiento en las direcciones vertical y horizontal, (4) bajo rendimiento en las dos direcciones aunado a déficit de atención y (5) buen desempeño en ambos ejes pero con déficit de atención. Para cada perfil, se maneja la velocidad horizontal, la velocidad vertical y el tiempo que el objeto es visible en la pantalla. El EA se encarga de obtener las variables x y y para decidir la próxima posición donde aparecerá el objetivo y el tiempo de visibilidad del mismo de acuerdo al perfil previamente seleccionado. El experimento realizado consistió en simular jugadores con habilidades de acuerdo a cada perfil, poniendo a prueba el EA a medida que avanzaba el juego. Los resultados arrojados muestran que el EA se adapta correctamente a las habilidades del jugador.

También se encuentran otros investigadores que se apoyan en el uso de algoritmos evolutivos. Esto se puede notar en Weber y Notargiacomo [16], con el juego *The Empire Game*, que emplea un algoritmo genético para llevar a cabo el ADD y hacerlo más entretenido. Algo parecido sucede con Shakhova y Zagarskikh [17], quienes presentan un sistema ADD con red neuronal entrenada a través de algoritmos evolutivos para lograr una mejor adaptación de la dificultad y además reducirla de acuerdo a los cambios en el nivel de la habilidad del jugador.

Suaza et al. [18] presentan *The Forest of the Guardians*, un juego móvil en 2.5D de luchas contra un agente con tres niveles de dificultad predefinidos. Se emplea un ADD con una función heurística que mide la diferencia de ”puntos de salud” del personaje para determinar el nivel de dificultad que se le asignará al agente. Se realizó una prueba del juego con 19 estudiantes y al finalizar, cada uno de ellos respondió el cuestionario estandarizado *Player Experience Inventory* (*PXI*), el cual arrojó resultados positivos en los ítems relevantes para este estudio. El 79 % de los estudiantes estaban sumergidos en el juego, el 63.1 % sintió que el juego no era ni muy fácil ni muy difícil, y para el 73.8 % el juego fue desafiante pero no demasiado desafiante. Adicionalmente, se obtuvo una tasa de victorias del 74 % con un promedio de puntos de salud restante del 26 %.

De manera similar, en los juegos *Infinite Mario Bros* y *Pacman*, Blom et al. [9] utilizan selvas aleatorias para ajustar la dificultad del juego en función de la predicción de la dificultad que experimentará el usuario. Este método también es utilizado por Araujo et al. [19], donde emplean un controlador difuso que genera niveles personalizados de desafío. Adicionalmente, Demediuk et al. [20] y Demediuk et al. [21] desarrollan agentes IA que le brindan al jugador un nivel de dificultad adaptado a su habilidad en tiempo real al alterar la política de selección de acciones de *Monte Carlo Tree Search*.

Verma et al. [2] presentan *Chemo-o-crypt*, un juego serio en 2D donde el personaje recorre un mapa en búsqueda de elementos y moléculas para equilibrar una ecuación química mostrada al principio de cada nivel mientras evade a sus enemigos, quienes por contacto reducen sus puntos de salud. El mapa contiene monedas y corazones, los cuales otorgan puntos adicionales al jugador. Una vez el estudiante considere que tiene las moléculas suficientes y correctas para equilibrar la ecuación, debe buscar la palabra *GO* dentro del mapa para poder enviar su respuesta. El ADD incorpora una Red Bayesiana Dinámica que mide los rasgos faciales para determinar el estado emocional del jugador y variar el porcentaje de pérdida de salud por contacto con el enemigo, perdiendo desde un 25 % en el nivel más fácil hasta el 100 % en el nivel más difícil. Posteriormente, se realizó una comparación del juego con ADD vs sin ADD. Los resultados indican que los participantes que jugaron la versión con ADD presentaron una mejora significativa en el aprendizaje en comparación con los estudiantes que jugaron la versión sin ADD. La aplicación de estos métodos probabilísticos de ADD también se puede apreciar en Gonzales et al. [22], quienes presentan un método apoyado en un algoritmo inteligente de prueba y error, el cual es una forma de optimización Bayesiana que se basa en el proceso de regresión Gaussiana y el algoritmo *MAP-Elites*.

El análisis de estos artículos permite entrever que la selección del tipo de algoritmo a usar por los autores no posee una fórmula o una recomendación directa de cuál y cómo se debe implementar. Sin embargo, es posible encontrar una cierta tendencia a usar modelos heurísticos en los juegos de *player vs player*. Adicionalmente, la mayoría de los juegos utilizados son juegos en 2D y en muchas ocasiones juegos clásicos como *PacMan*, *Mario Bros*, *Tetris* o *Pong* para poner a prueba los algoritmos. Aunque no es el caso de los juegos serios o exergames, ya que estos presentan juegos con un fin específico y los cuales deben presentar ciertas características especiales en el diseño que ayuden al usuario. Se observa que el ADD presenta diferentes formas que permiten el uso variado de algoritmos, encontrándose entre estas los Algoritmos Evolutivos, Heurísticos, Probabilísticos y Redes Neuronales.

1.2.3. Comparación entre algoritmos utilizados para el ADD

Al profundizar en la búsqueda sobre artículos que enfaticen la comparación entre los métodos de ADD, se encuentran Darzi et al. [31], que se valen de una versión exergame de *Pong* para un solo jugador. El juego consta de dos paletas y una pelota sobre una mesa, donde la paleta inferior es controlada por el participante, mientras que la paleta superior es controlada por un agente. Se obtiene un punto cuando la pelota pasa la paleta del adversario. En este juego, la dificultad se puede ajustar usando dos parámetros: la velocidad de la pelota y el tamaño de las paletas. La comparación realizada en este artículo contrasta métodos de ajuste de dificultad enfocándose en los datos que usan como entrada. Los autores presentan tres tipos de ADD: uno

que hace uso de datos de rendimiento, un segundo que hace uso de datos de rendimiento y datos obtenidos a partir de un test de personalidad, y un tercero que hace uso de datos de rendimiento, personalidad y datos fisiológicos medidos a través de sensores. Adicionalmente, se implementa un ajuste de dificultad manual y uno completamente aleatorio. Los resultados muestran que el método ADD que utilizó todos los tipos de datos presentó mayor precisión para la velocidad de la pelota y el tamaño de las paletas de acuerdo a los valores deseados por los participantes, pero es posible que la diferencia entre las precisiones sea demasiado baja para mostrar un cambio en la experiencia del usuario. Por lo tanto, los autores indican que el uso de mediciones fisiológicas solo debe implementarse cuando impacten de forma evidente en el beneficio del jugador.

Por otra parte, en la investigación de Pfau et al. [32], se presenta *Eternal Challenge*, una mazmorra de un único jugador dentro del juego de rol multijugador masivo en línea *Aion*. El objetivo es adquirir puntos de experiencia, equipos, monedas y/u otros artículos deseables mientras se lucha contra diferentes oponentes controlados por agentes. Los autores realizan un estudio sobre los efectos del ADD a través del modelo profundo del comportamiento del jugador (DPBM) basado en redes neuronales, en comparación con el ajuste de parámetros heurísticos clásicos (HPT) de acuerdo a la tasa de finalización, enfocándose en la motivación intrínseca, el desafío percibido y la motivación del jugador. Los resultados obtenidos arrojan una preferencia significativa hacia el método DPBM.

En otra investigación, Moon y Seo [24] crean una versión de *Air Hockey* para computadora donde el usuario debe jugar contra un agente. El juego consta de una mesa con una superficie resbaladiza, un disco y dos delanteros, los cuales deben golpear el disco y empujarlo hasta la portería del oponente con el fin de conseguir puntos. Los autores proponen un método de adaptación rápida del usuario basado en redes neuronales profundas. Para su validación, este fue comparado con un método de red neuronal que incorpora capas de memoria a corto plazo (LSTM), y con un ADD convencional que modifica la dificultad según la victoria o derrota del jugador. Los resultados indican que el juego con adaptación rápida del usuario presentó mejores resultados en el porcentaje de victorias y en la posesión del disco que el método LSTM, pero equivalentes al método convencional. En términos de disfrute, inmersión y dificultad percibida, el método propuesto por los autores mostró mejores resultados que el método LSTM.

En cuanto a los artículos de comparaciones entre algoritmos aplicados a juegos para lograr un acertado ADD, se puede observar que una de las comparaciones maneja un enfoque en el cual el punto de interés son los datos de entrada para el ajuste. En la segunda existe una comparación de un ADD de redes neuronales vs un ADD en el que se tienen preestablecidos niveles de dificultad seleccionados de acuerdo a la victoria o derrota del usuario. En la tercera comparación intervienen dos algoritmos basados en redes neuronales y al igual que la segunda, un ADD de niveles preestablecidos, lo que muestra que la gama de métodos encontrados en el ítem de comparacio-

nes es limitada. La información es básicamente escasa, al solo encontrar tres artículos. Es, en definitiva, complejo no solo vislumbrar una diferencia entre varios de los métodos existentes, sino que también se dificulta establecer cuál de estos presenta mejores resultados en determinados juegos. Por otra parte, la mayoría de los artículos que aplican uno de los métodos, realiza la validación comparando el mismo juego en dos versiones: una con ADD y otra sin ADD, obteniendo generalmente mejores resultados en la implementación de este tipo de ajuste dinámico.

De la información obtenida a partir de los textos consultados, se utilizarán los aspectos relacionados con los tipos de algoritmos de ADD usados en juegos, la selección y medición de las variables de entrada para el ADD, al igual que las mecánicas que este ajuste modifica para adaptar la dificultad. Adicionalmente, se identificó que existen vacíos en la literatura relacionados con la comparación de la incidencia del ADD basado en aprendizaje de máquina. Así pues, esta investigación pretende realizar una contribución a la línea de investigación en el área del control, ya que permite aprovechar las tecnologías de la transformación digital en la generación de un entorno con RA e IA que mejore la aceptación y adaptabilidad de los usuarios a este tipo de herramientas.

1.3. Objetivos

1.3.1. Objetivo General

Evaluar el efecto del ajuste dinámico de dificultad basado en aprendizaje estadístico en el desempeño y motivación de los usuarios de un juego en realidad aumentada de disparo en primera persona.

1.3.2. Objetivos Específicos

- Concebir un juego en realidad aumentada de disparo en primera persona que almacene datos de comportamiento y desempeño.
- Implementar tres algoritmos de ajuste dinámico de dificultad basados en aprendizaje estadístico.
- Comparar la usabilidad, el compromiso y el desempeño de los usuarios en función del algoritmo de ajuste dinámico de dificultad.

1.4. Distribución de la monografía

Este trabajo de grado se divide en cinco capítulos que abordan diferentes aspectos de la investigación:

- Capítulo 1: Generalidades. En este capítulo se presenta el planteamiento del problema, el estado del arte y los objetivos de este trabajo de grado.
- Capítulo 2: Desarrollo del software. Aquí se especifican los requerimientos, las dinámicas, el diseño y el desarrollo del software, así como también las tecnologías utilizadas para la ejecución de este trabajo de grado.
- Capítulo 3: Desarrollo de algoritmos estadísticos. Este capítulo describe el desarrollo y la implementación de los algoritmos basados en aprendizaje estadístico utilizados para generar un ajuste automático de la dificultad de un juego en realidad aumentada de disparo en primera persona.
- Capítulo 4: Comprobación de Usabilidad, Compromiso y Desempeño. En este capítulo, se presenta el análisis de los resultados obtenidos para evaluar el efecto del algoritmo de ajuste dinámico de dificultad en el compromiso y desempeño de los usuarios de un juego en realidad aumentada de disparo en primera persona, así como también se examina la usabilidad del juego.
- Finalmente, Capítulo 5: Discusión, Conclusiones y Trabajos Futuros. En este capítulo, se discuten las conclusiones obtenidas a lo largo del desarrollo de este trabajo de grado, además de proponer posibles trabajos futuros a considerar.

Capítulo 2

Desarrollo Software

2.1. Requerimientos del Software

Las especificaciones del sistema responden al desarrollo de una prueba de concepto sobre el efecto del ADD en el compromiso y desempeño de un juego que emplea tecnología inmersiva. Este trabajo se enmarca dentro de la tesis de doctorado del codirector Diego Enrique Guzmán Villamarín, cuyo propósito es desarrollar un marco conceptual para el diseño de juegos serios. Por esta razón, se atiende a su orientación para identificar las condiciones necesarias en la elaboración del software.

De acuerdo con lo anterior, se realizan una serie de reuniones con el codirector y se aplica la metodología ágil *Scrum* [33] para obtener historias de usuario, de donde se logra determinar los elementos necesarios para llevar a cabo la prueba de concepto mencionada. Así pues, en primera medida, se obtienen los siguientes requerimientos generales:

- El juego debe implementar RA, puesto que esta tecnología presenta ventajas significativas con respecto a la RV en cuanto a la facilidad de despliegue. Dado que la RA se puede implementar en un dispositivo móvil, mientras que la RV requiere de un casco de inmersión [34].
- El juego es de disparo en primera persona debido a que es el género más popular, y por tanto, reporta una mayor cantidad de usuarios, lo cual es fundamental para la validación estadística de hipótesis entre el factor a estudiar (algoritmo de ADD) y las variables de respuesta: compromiso y desempeño. En el estudio hecho por [35], los géneros de juegos más jugados son acción con un 43 %, juegos de rol con un 29 %, estrategia con un 15 %, simulación con un 8 % y otros con un mínimo de 5 %. Dentro del género de acción, los subgéneros favoritos son disparo en primera persona con un 62 % y disparo en tercera persona con un 21 %.
- La información se debe almacenar en una base de datos en la nube para que pueda ser accedida por todos los dispositivos desde los que se juega, de manera tanto secuencial

como simultánea. Dicha información será el insumo para evaluar la veracidad y validez de las hipótesis formuladas en el planteamiento del problema.

- Se deben desarrollar dos versiones del juego: una para la recolección de información, que permita el entrenamiento de los algoritmos basados en aprendizaje estadístico, y otra con la implementación de ADD.

En la Tabla 2.1 se presentan los requisitos funcionales del software, los cuales se obtuvieron a partir de la información del punto anterior. Estos requisitos indican que el juego debe aplicar la tecnología de Realidad Aumentada (RA) para superponer objetos tridimensionales en la realidad, permitiendo al usuario interactuar con ellos a través de un dispositivo móvil. Además, el juego debe pertenecer al género de disparo en primera persona, donde se simula el uso de un arma para destruir objetos enemigos desde la perspectiva del jugador.

Asimismo, se establece que el juego debe almacenar toda la información en una base de datos alojada en la nube, donde se guardarán los datos de compromiso y desempeño de los jugadores. Es importante destacar que el juego se presentará en dos versiones: una con ajuste manual de la dificultad y otra con el sistema de Ajuste Dinámico de Dificultad (ADD).

Titulo requisito	**Descripción del requisito**	**Condición de satisfacción**
Implementación de RA en el juego	Como *Product Owner*, quiero que el juego emplee la realidad aumentada, de tal manera que facilite su implementación en dispositivos móviles, permitiendo el acceso a un mayor número de personas.	El juego superpone objetos tridimensionales en el entorno, con los cuales el usuario puede interactuar a través de la pantalla de un dispositivo móvil.
Implementación del género de juego de disparo en primera persona.	Como *Product Owner*, quiero que el juego pertenezca al género de disparo en primera persona, de tal manera que permita atraer a una mayor cantidad de usuarios y obtener un número considerable de datos para la validación estadística de hipótesis.	El juego simula el uso de un arma de fuego para destruir objetos enemigos desde la perspectiva del usuario.

Continúa en la siguiente página.

Título requisito	Descripción del requisito	Condición de satisfacción
Almacenamiento de información en una base de datos en la nube	Como *Product Owner*, quiero que la aplicación guarde la información de todos los dispositivos en una base de datos, de tal manera que pueda acceder a ella en cualquier momento, a la vez que recopila los datos de juego sin importar el tiempo y lugar en que juegue el usuario.	El juego guarda información del compromiso y desempeño del jugador en la nube.
Desarrollo de dos versiones del juego	Como *Scrum Master*, quiero que el juego se presente en dos versiones: una inicial en donde el usuario ajuste los valores de dificultad, y una segunda con ADD, cuya base sea la información recolectada a través de la primera.	El juego presenta dos versiones: una con ajuste manual de dificultad y otra que emplea tres tipos diferentes de algoritmos aplicados en ADD.

Tabla 2.1: *Requisitos funcionales*

2.2. Definición de las dinámicas del juego

En función de los requerimientos principales de funcionalidad descritos en la sección anterior, se propone un juego en realidad aumentada de disparo en primera persona para dispositivos móviles de un único jugador, cuyo propósito es destruir a un grupo de adversarios. El juego consta de dos versiones: una inicial en la cual el usuario ajusta manualmente el nivel de dificultad del juego, y una segunda versión generada a partir de los datos obtenidos con la primera, en la cual la dificultad es ajustada por un algoritmo de aprendizaje.

Adicionalmente, se definen las dinámicas del juego, las cuales agrupan los detalles particulares, como lo son la temática, los datos del usuario, los parámetros modificables, las variables de desempeño y demás características específicas de la evolución del juego. Las dinámicas son planteadas y seleccionadas mediante la técnica de *Brainstorming* [36], en la que participaron los autores, los directores del presente proyecto y el Semillero AR-VR de la Universidad del Cauca.

La Tabla 2.2 presenta las posibilidades surgidas de la técnica de *Brainstorming* para el tipo de enemigo a enfrentar, su armamento y el arma con la que se le equipará al jugador. Los valores resaltados en color verde corresponden a las opciones seleccionadas. La idea a desarrollar gira en torno al ataque de naves extraterrestres, debido a que el uso de estos enemigos disminuye el

empleo de animaciones de movimiento y facilita emular lo que se supone sería su comportamiento en el entorno real. Para el armamento enemigo, son elegidos los misiles, ya que es posible generarlos de forma intermitente y una vez disparados, su existencia no depende de las naves. El jugador se equipa con una metralleta, debido a que esta permite la movilidad del usuario y no obstaculiza de forma importante la visión en la pantalla.

Tipo de enemigo	**Armas del enemigo**	**Armas del jugador**
Zombies	Garras	Arma corta
	Colmillos	Arma blanca
	Golpes	Motosierra
		Rifle de asalto
		Lanzallamas
		Ballesta
Naves extraterrestres	Misiles	Metralleta
	Rayos láser	Rayos láser
	Plasma	Carro tanque
		Lanzacohetes
Fantasmas	Posesión	Rayos
	Lanzamiento de objetos	Espada sagrada
		Malla electro-estática
Soldados	Pistola	Arma blanca
	Ametralladoras	Arma corta
	Rifle de asalto	Rifle de asalto
	Arma blanca	Fusil de francotirador
		Lanzacohetes

Tabla 2.2: *Temática del juego*

La Tabla 2.3 muestra, resaltados en verde, los datos de registro considerados relevantes para la identificación, individualización y seguridad del usuario. También, se incluyen una serie de preguntas poco frecuentes para la recuperación de la contraseña de la cuenta de usuario, con el objetivo de reducir la vulnerabilidad del acceso. Además, se registran los datos de sesión, los cuales serán evaluados posteriormente para determinar el compromiso del jugador. Es importante mencionar que estos datos no hacen referencia a información personal, lo cual facilita el despliegue de la aplicación en tiendas virtuales, ya que no entran en conflicto con reglamentos de recolección de datos personales.

Registro y validación de usuarios	Preguntas de recuperación	Datos de sesión del usuario
Nombre	Segundo nombre de tu madre	Fecha
Contraseña	Segundo nombre de tu hermana mayor	Hora
Apodo o seudónimo	Nombre del primer colegio al que fuiste	Duración
Edad	Tu apodo de niño	
Género	Segundo personaje favorito de películas	
Correo	Comida que más te desagrada.	
País	Nombre de tu futura mascota	
Pregunta de recuperación	Con quien te perderías en una isla desierta	
Experiencia en juegos RA	Cuantos hijos te gustaría tener	
	Nombre de tu actriz favorita	
	Equipo de fútbol favorito	
	Marca de celular favorita	
	Actividad favorita	
	Personaje de anime favorito	

Tabla 2.3: *Datos del usuario*

En la columna derecha de la Tabla 2.4 se muestran los indicadores de desempeño del jugador, de los cuales se seleccionaron seis, resaltados en color verde, por considerarse los más representativos de las habilidades de un jugador. En la columna izquierda del Cuadro 2.4 se muestran los parámetros que afectan la dificultad del juego. De estos, se seleccionaron solo tres, resaltados en color verde, con el fin de evitar la sobrecarga de información y minimizar los tiempos en el ajuste de valores. Para la selección de estas variables, se examinó cuáles de ellas suponían un mayor impacto en los indicadores de desempeño seleccionados y, por consiguiente, en la dificultad a la que se enfrenta el jugador. Los parámetros de la columna izquierda no elegidos en este apartado, igualmente, son implementados dentro del juego; sin embargo, se les asignan valores constantes.

Cambio de parámetros	Datos de desempeño
Cantidad de hordas por partida	Partida ganada o perdida
Cantidad de naves por horda	Tiempo de duración de la partida
Tiempo máximo de duración de la partida	Naves destruidas en la partida
Tiempo de aparición entre hordas	Hordas generadas
Velocidad de las naves	Puntos de vida restantes

Continúa en la siguiente página.

Cambio de parámetros	Datos de desempeño
Velocidad de los misiles de las naves	Tiempo de reacción
Velocidad de disparo del jugador	Cantidad de tiros disparados
Cadencia de disparo de las naves	Aciertos de balas del jugador
Cadencia de disparo del jugador	Daño total recibido por el jugador
Cantidad de puntos de vida del jugador	Puntaje de la partida
Cantidad de puntos de vida de las naves	Porcentaje de naves destruidas
Daño ocasionado por el misil de las naves	
Daño ocasionado por la bala del jugador	
Daño ocasionado por el choque de las naves con el jugador	
Munición del cargador del arma	
Radio de aparición de las naves	

Tabla 2.4: *Parámetros del juego*

En cuanto al comportamiento de las naves, se debe definir tanto la forma en que éstas aparecen en la escena como la manera en que se desplazan respecto al jugador. En relación al primer factor que se observa en la Tabla 2.5, se optó por hacer emerger las naves en un punto aleatorio del entorno real a través de pequeños portales temporales, lo cual está en concordancia con lo que podría ser una invasión extraterrestre. Por otra parte, se ha seleccionado un movimiento para las naves que esté dirigido constantemente hacia la ubicación del jugador, mientras realizan movimientos aleatorios evasivos que permiten resaltar los gráficos tridimensionales y emular lo que se supone es el comportamiento real de naves espaciales.

Aparición de naves	Movimiento de las naves
Desde la parte superior	Aleatorio
A través de portales temporales	Ataque a partir de cierta distancia del jugador
A través de neblina	Ataque directo
Escalamiento	Ataque mientras evaden

Tabla 2.5: *Comportamiento de las naves*

En la Tabla 2.6 se proponen una serie de ítems clave utilizados en la realidad aumentada, en los juegos de disparo en primera persona y en la implementación de los juegos en general, enfocados en el jugador. Estos ítems tienen el propósito de garantizar que el juego abarque todas las características que influyen en la forma en que el jugador interactúa con el mundo del juego.

Características del jugador
Trasladarse en el espacio real para cambiar su posición en un entorno virtual, mientras interactúa con diferentes objetos virtuales tridimensionales
Destruir el mayor número de enemigos disparándoles con el arma
Evitar las colisiones con los enemigos y sus misiles
Obtener puntos por desempeño
Perder y recuperar puntos de vida
Visualizar la posición de las naves
Obtener potenciadores
Recargar el arma
Pausar, reiniciar y/o abandonar la partida
Modificar parámetros del juego
Registrarse en la base de datos

Tabla 2.6: *Características del jugador*

Con el fin de brindar al usuario una experiencia más real e inmersiva, se formula la aplicación de efectos audiovisuales de la Tabla 2.7, conectados a dinámicas seleccionadas en las tablas anteriores. El uso de explosiones genera un impacto visual significativo en el usuario y acerca más a la realidad los eventos mostrados. Está restringido únicamente a las naves cuando son destruidas, para no limitar la visibilidad del entorno y reducir el espacio utilizado en la pantalla del dispositivo. Los sonidos seleccionados refuerzan la existencia de los acontecimientos mostrados en el juego, mientras se evita agregar sonidos de gritos y destrucción, los cuales no tienen elementos que los provoquen e innecesariamente generan aturdimiento o distracción en el jugador.

Explosiones	Sonidos
En enemigos al ser destruidos	Disparos
Impacto de misiles enemigos con el jugador	Impacto de balas
Aleatorias en el entorno	Impacto de misiles
Impacto de las naves con el jugador	Recarga de arma
	Misiles en movimiento
	Explosiones
	Movimiento de naves
	Aparición de naves
	Sonidos de fondo (gritos y destrucción)
	Música de fondo

Tabla 2.7: *Efectos audiovisuales*

En cuanto a los elementos adicionales propuestos para el juego en la Tabla 2.8, no se considera su implementación para la primera versión debido a que influyen directamente en los datos de desempeño que se buscan recolectar.

Objetos	Potenciadores
Árboles	Generar mayor daño a las naves
Escudos de protección	Recuperación total de vida del jugador
Potenciadores de juego	Anulación de daño sobre el jugador
Incentivos de movimiento (recompensas)	Todas las naves se ralentizan

Tabla 2.8: *Elementos adicionales*

Las dinámicas seleccionadas permiten definir de forma más clara la estructura de la trama, parte fundamental en el diseño de videojuegos. Según [37], está compuesta por la evolución narrativa, la estructura narrativa, los escenarios y los personajes; dicha estructura se explica a continuación:

- **Evolución narrativa:** Hace referencia a la generalidad de lo que ocurre dentro de la historia [37].
  En la historia del juego propuesto, los seres humanos nunca estuvieron seguros de la existencia de vida en otros planetas, hasta el día en que misteriosos portales aparecieron por todo el mundo, de donde surgieron extrañas naves espaciales con la única misión de destruirlos. En menos de un mes, aplastaron las fuerzas militares de la humanidad y desde entonces, cada persona defiende su propio hogar.

- **Estructura narrativa:** Instaura el orden en el que se desarrollan las situaciones en la historia y la influencia del jugador sobre las mismas [37].
  En este juego, el usuario podrá moverse en el entorno real con el dispositivo móvil para cambiar la posición del personaje dentro del juego, con el fin de sobrevivir a cinco hordas en un tiempo determinado. Las naves espaciales aparecerán cada cierto tiempo, y el jugador deberá dispararles para eliminarlas antes de que choquen con él y así evitar perder puntos de vida. El juego se ganará si el jugador logra sobrevivir a la aparición de todas las naves sin que los puntos de vida del personaje lleguen a cero.

- **Escenarios:** Fija el espacio de interacción del jugador, afianza la inmersión y precisa el contexto para los personajes [37].
  El escenario planteado presenta la superposición de objetos virtuales en el entorno del mundo real mediante el uso del dispositivo móvil. En la parte superior de la pantalla, se visualizará un botón de pausa, el número de hordas generadas, la cantidad de naves eliminadas, un temporizador y la barra de vida del personaje. En el centro de la pantalla, se encontrará un punto de mira que ayudará al jugador a apuntar a los objetivos. Además, en la parte inferior izquierda se mostrará un radar que indicará la ubicación de los enemigos alrededor del jugador, mientras que en la parte inferior derecha se ubicarán los botones de disparo y recarga.

- **Personajes:** Participan directamente en la evolución de la narrativa; su selección debe estar enlazada a la historia y ajustarse al público al que están dirigidos [37].
  Siendo un juego en primera persona, el personaje del jugador no es visible físicamente, ya que la cámara toma la perspectiva de dicho personaje. Por lo tanto, se seleccionan únicamente los personajes que cumplen la función de enemigos.

Tras seleccionar apropiadamente las dinámicas y la estructura de la trama del videojuego, los desarrolladores han propuesto un nombre que se ajusta perfectamente a su concepto: "*Invasion Victory*". Este hace referencia directamente a los elementos clave del juego y su objetivo principal, lo que lo convierte en una elección adecuada y coherente. De igual forma, en la Figura 2.1 se presenta el icono diseñado para la aplicación.

Figura 2.1: Icono de *Invasion Victory*

2.3. Diseño de interfaces de usuario

2.3.1. Diseño de interfaces iniciales

El diseño de los bosquejos de las interfaces se adaptan a las dinámicas seleccionadas en la sección anterior para presentar de forma ordenada la información necesaria al usuario.

- **Base:** La Figura 2.2 presenta una imagen de fondo, un margen a color, un botón para cerrar la aplicación y el recuadro *User data*, donde se intercambian los cuadros de registro, inicio de sesión y cambio de contraseña.

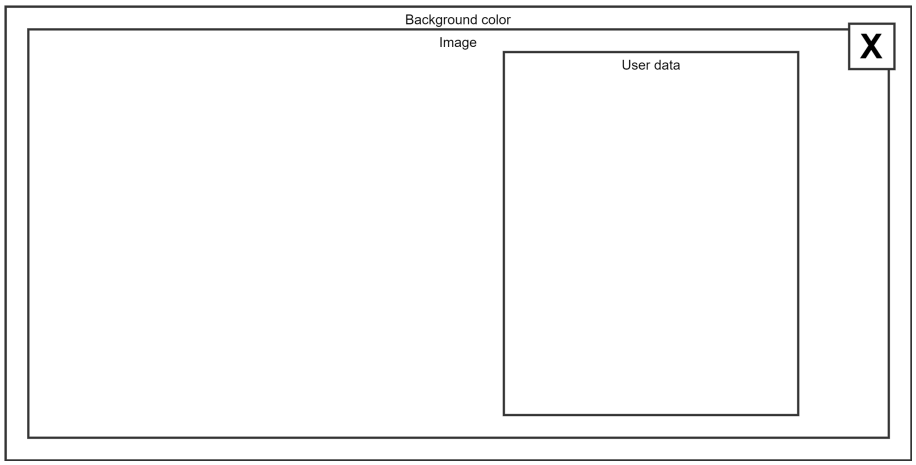


Figura 2.2: Plantilla base de datos de usuario

- **Ingreso al juego:** Cuenta con dos entradas de texto para el seudónimo y la contraseña, y tres botones. Dos de ellos redireccionan al usuario a las pantallas de cambio de contraseña o de registro de usuario, mientras que el tercero le permite ingresar al juego (Figura 2.3).

Figura 2.3: Pantalla de inicio de sesión (ver Tabla 2.3)

- **Registro:** La pantalla de Registro, presentada en la Figura 2.4, muestra dos entradas de texto: una para el seudónimo del usuario (*Nick*) y otra para la clave de acceso (*Password*). Además, la interfaz incluye dos menús desplegables, uno para seleccionar la respuesta a la pregunta *Have you played AR games before?* y otro con la pregunta para la recuperación de clave, junto con un cuadro de texto en la parte inferior para que el usuario escriba la correspondiente respuesta. En la parte inferior de esta pantalla, se encuentra una casilla de

verificación junto al elemento *Informed Consent*, para indicar el acuerdo con este. Igualmente, se incorpora una barra lateral a la derecha que permite desplazarse hacia arriba o hacia abajo en esta pantalla.

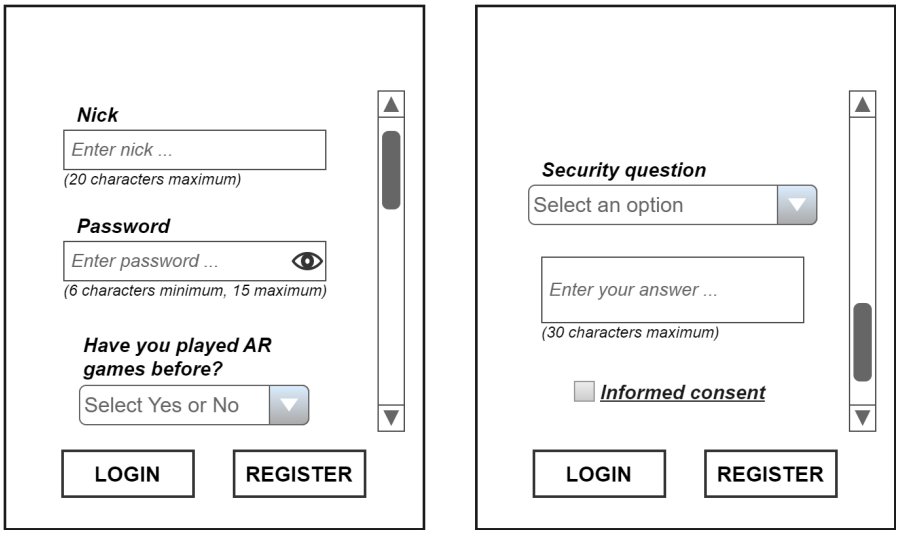


Figura 2.4: Pantalla de registro (ver Tabla 2.3)

- **Cambio de contraseña:** Presenta una entrada de texto para el seudónimo, un cuadro de opciones, otro de respuesta para la validación, una entrada de texto para la nueva contraseña y botones para cancelar o aceptar (Figura 2.5).

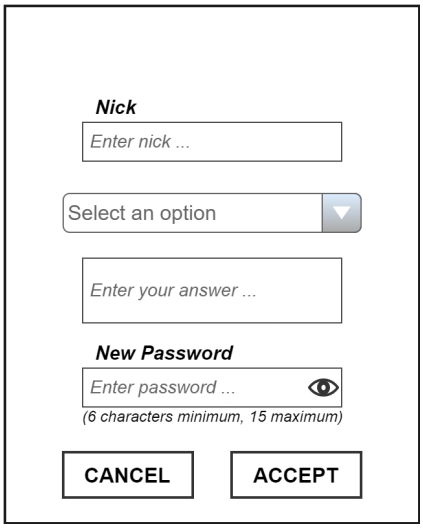


Figura 2.5: Pantalla de cambio de contraseña (ver Tabla 2.3)

- **Pantalla de vídeo:** Contiene un margen a color, un recuadro donde se reproduce un vídeo informativo sobre el juego (Figura 2.6). Adicionalmente, cuenta con un botón de omitir (*Skip*) para saltar a la pantalla siguiente.

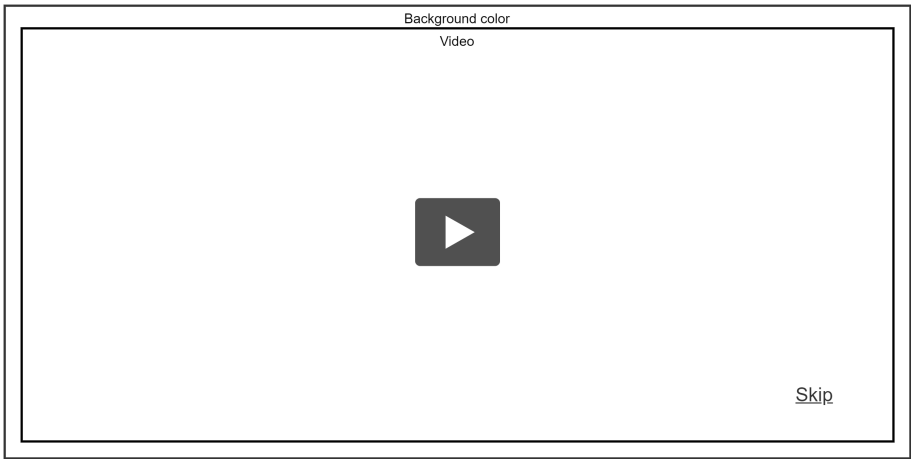


Figura 2.6: Pantalla de vídeo

- **Clasificación:** La pantalla presentada en la Figura 2.7, contiene un fondo con un recuadro superpuesto que contiene botones para acceder a las opciones: presentar créditos, acceder al manual del juego, cerrar sesión e iniciar el juego. También muestra un recuadro que presenta la clasificación de los mejores jugadores.

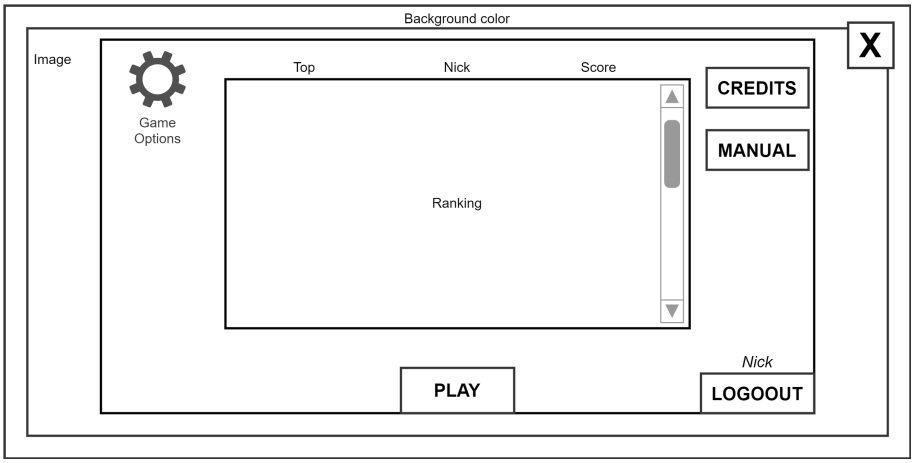


Figura 2.7: Pantalla de clasificación

- **Cambio de parámetros:** La pantalla presentada en la Figura 2.8, incluye un control deslizante corto para cambiar el modo de dificultad de aleatorio a manual y viceversa, tres controles

deslizantes con cuadros de texto para ajustar los parámetros que afectan la dificultad del juego: el número de naves a enfrentar por horda, el tiempo de aparición entre las hordas y la velocidad de las naves. Además, cuenta con dos botones, uno para cancelar y otro para aceptar los cambios en las variables.

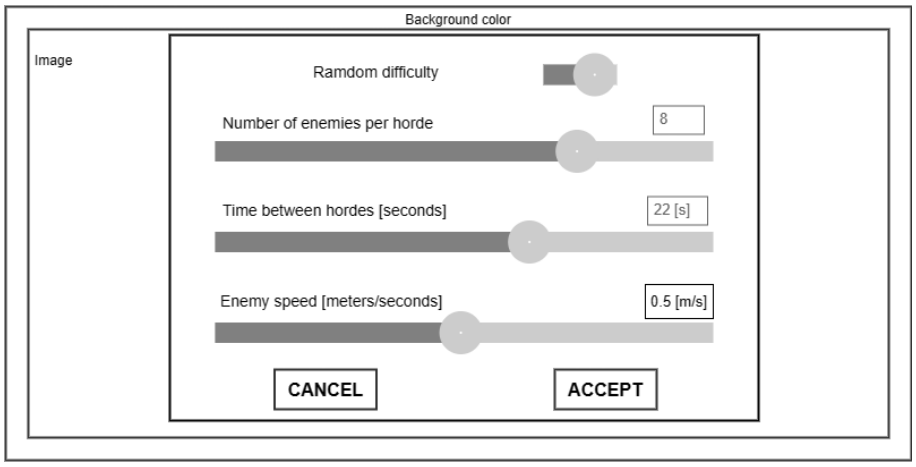


Figura 2.8: Pantalla de cambio de parámetros (ver Tabla 2.4)

- **Instrucciones de uso:** La pantalla presentada en la Figura 2.9, cuenta con un doble recuadro para mostrar imágenes explicativas del funcionamiento del juego y un botón para salir de esta opción.

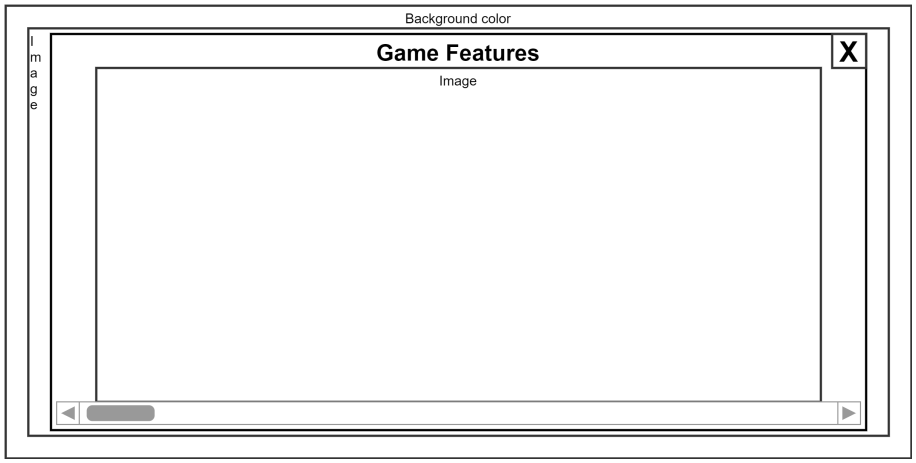


Figura 2.9: Pantalla de instrucciones

- **Créditos:** La pantalla presentada en la Figura 2.10 cuenta con un recuadro de texto con datos de componentes implementados en el juego y un botón para salir de esta opción.

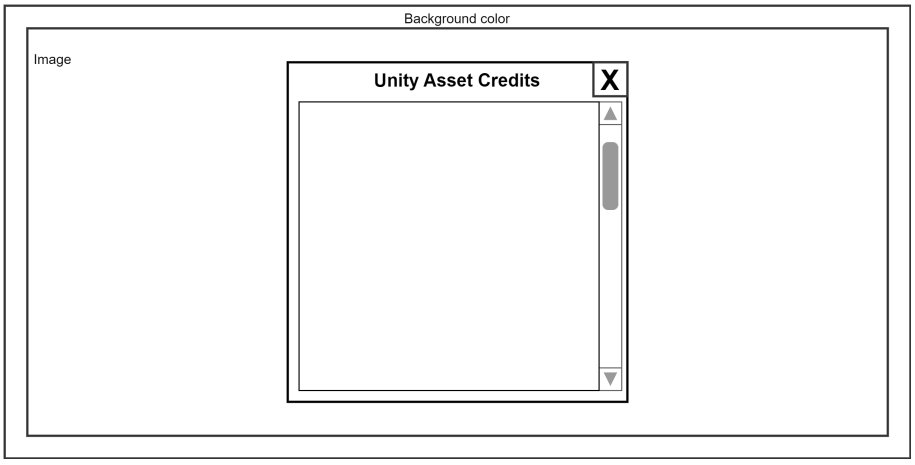


Figura 2.10: Pantalla de créditos

- **Inicio del juego:** En la parte superior de la Figura 2.11, se encuentra un botón de pausa, las cantidades de hordas generadas, la cantidad de naves eliminadas, un temporizador y la barra de vida del jugador. En la parte inferior se observa un radar y del otro lado los botones de disparar y recargar. Adicionalmente, se cuenta con un punto de mira en el centro de la pantalla.

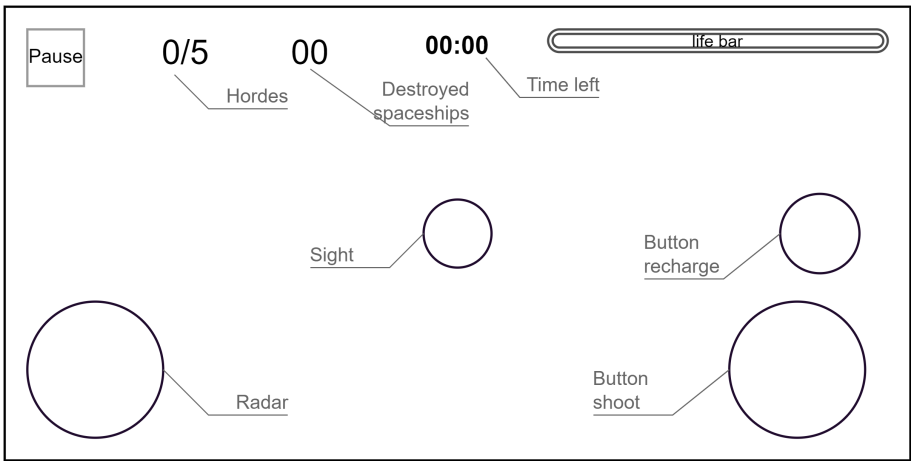


Figura 2.11: Pantalla de inicio del juego (ver Tabla 2.6)

- **Pausa del juego:** Sobrepone un menú en la pantalla que le permite al usuario detener el juego (Figura 2.12). En este menú se encuentran tres botones que permiten reanudar la partida, reiniciar la partida o salir de la partida.

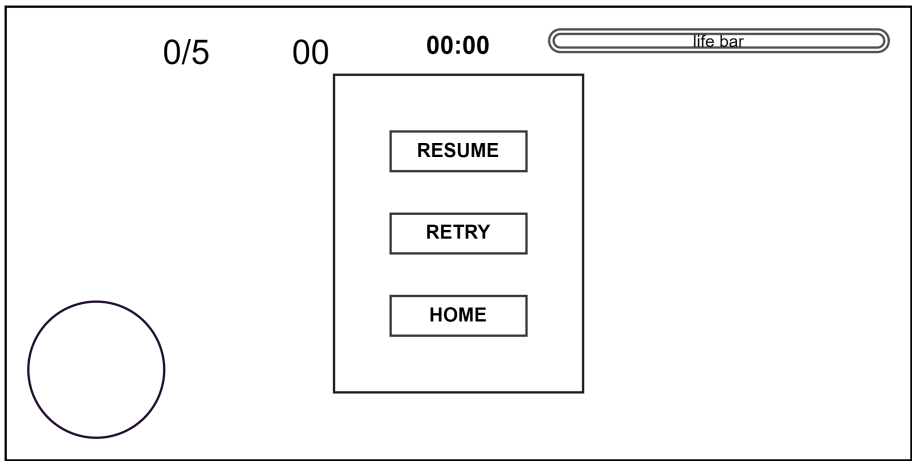


Figura 2.12: Pantalla de pausa

- **Juego terminado:** Presenta un mensaje de juego terminado en fracaso, un cuadro de texto para el puntaje y otro para el tiempo de juego. Además, cuenta con dos botones para redireccionar al jugador a otras pantallas y un botón para cerrar la aplicación (Figura 2.13).

Figura 2.13: Pantalla de juego terminado

- **Felicitaciones:** Presenta un mensaje de felicitaciones, un cuadro de texto para el puntaje y otro para el tiempo de juego. Además, cuenta con dos botones para redireccionar al jugador a otras pantallas y un botón para cerrar la aplicación (Figura 2.14).

Figura 2.14: Pantalla de felicitaciones

2.4. Modelado del comportamiento del juego

Un autómata de estado finito está compuesto por una serie de estados que cuentan con una señalización en su estado inicial y componentes de entrada que definen el paso de un estado a otro [38]. El uso de estos autómatas, a través de grafos de transición de estados, permite describir la interacción del usuario con las interfaces al activar los eventos mediante la pulsación de botones.

De acuerdo con la Figura 2.15, cuando el usuario ingresa a la aplicación se encuentra con la pantalla de *Login* (ver Figura 2.3). En el caso de tener un usuario y contraseña ya registradas, podrá ingresarlos por medio del botón *bt_login* y sus datos serán enviados a la base de datos para su validación. Si esta operación funciona correctamente, se despliega la pantalla *History* (ver Figura 2.6), con la narrativa del juego, desde donde se puede salir al presionar el botón *bt_skip* o esperar a que este termine su reproducción para pasar a la pantalla de *Ranking*.

Por otra parte, si el usuario no está registrado, podrá ingresar a la pantalla de *Register* (ver Figura 2.4) a través del botón *bt_registernow*. Allí, después de aportar la información necesaria y presionar el botón *bt_register*, sus datos serán guardados en la base de datos. Desde esta pantalla, el usuario tiene la opción de cerrar la aplicación pulsando el botón *bt_exit*, o leer el texto del consentimiento informado con el botón *bt_consent*, donde también es posible cerrar la aplicación.

Finalmente, en esta sección, podrá desde la pantalla *Login* ingresar a la pantalla *Recover* (ver Figura 2.5) pulsando el botón *bt_recover* para cambiar su contraseña y guardar los cambios en la base de datos con el botón *bt_accept*, o cerrar la aplicación. Cuando dos estados están conectados por el evento *time*, la transición de uno al otro se da sin necesidad de que el usuario ejecute

acción alguna. Los estados derivados de *Ranking* se describen en la Figura 2.16.

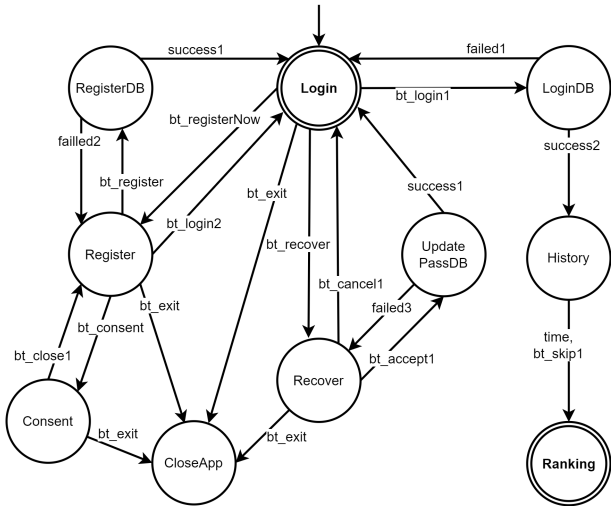


Figura 2.15: Autómata de estado finito para pantallas datos de usuario.

Una vez el usuario ingresa en la pantalla *Ranking* (ver Figura 2.7), podrá observar el listado de los 10 mejores jugadores. Adicionalmente, como se observa en la Figura 2.16, podrá pulsar el botón *bt_options*, que lo llevará a la pantalla *Parameters* (ver Figura 2.8), para modificar y guardar en un archivo de texto las variables del juego. Al presionar el botón *bt_credits*, aparecerá la pantalla *Credits* (ver Figura 2.10), que muestra los créditos de la aplicación. El botón *bt_manual* despliega instrucciones de uso de la aplicación en la pantalla *Manual* (ver Figura 2.9), y *bt_logout* cierra la sesión y vuelve a la pantalla de *Login*. El botón *bt_exit* cierra la aplicación y *bt_play* presenta el estado *Movement*, que le indica al usuario qué movimientos realizar para evadir las naves enemigas. Luego de mostrar esta información o presionar el botón *bt_skip2*, el usuario podrá pasar a la pantalla *Game*. Los estados derivados de *Game* se describen en la Figura 2.17.

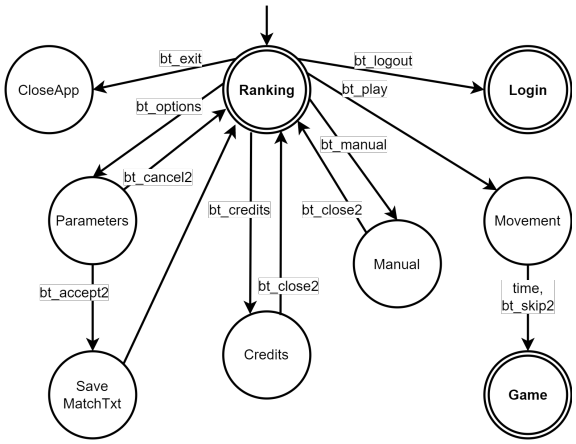


Figura 2.16: Autómata de estado finito para pantallas de menú de opciones.

Al desplegarse la pantalla *Game* (ver Figura 2.11), se presenta información necesaria para que el jugador interactúe y controle el juego. Luego de tres segundos, el jugador ingresa al estado *Playing*, el cual cuenta internamente con una red de Petri 2.21 que modela las interacciones del usuario y el juego. Desde aquí, es posible ingresar a la pantalla *Pause* (ver Figura 2.12) mediante el botón *bt_pause*, dónde se mostrará una ventana con los botones *bt_resume* para reanudar la partida, *bt_retry* para reiniciar la partida o *bt_home* para salir de la partida retornando a la pantalla de *Ranking*.

Una vez se termine la partida, bien sea por pérdida de puntos de vida, por tiempo agotado o porque se han generado y destruido todas las hordas, la aplicación almacena los datos de desempeño del jugador en un archivo de texto y después ingresa al estado *Connection*. Los estados derivados de *Connection* se describen en la Figura 2.18.

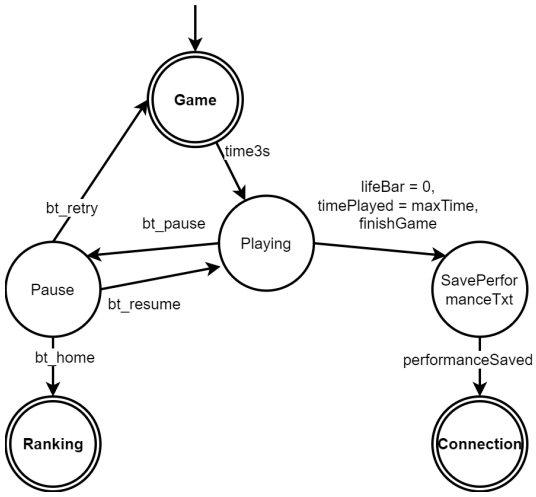


Figura 2.17: Autómata de estado finito para pantallas de juego.

En la Figura 2.18 se observa que al finalizar una partida se verifica la conexión a internet, puesto que es necesario almacenar en la base de datos el registro de la sesión, los valores de las variables que el usuario seleccionó y los valores de las variables de desempeño. Seguidamente, se obtiene de la base de datos el valor de una de las variables del *performance* que indica si fue ganada o no la partida. En función de este resultado, se despliega una de las dos pantallas: *Congratulations* (ver Figura 2.14) o *GameOver* (ver Figura 2.13). Junto con el puntaje obtenido, el tiempo de duración se muestran los botones *bt_home* para regresar a la pantalla *Ranking*, el botón *bt_retry* para jugar una partida con la misma selección de variables, y el botón *bt_exit* para cerrar la aplicación.

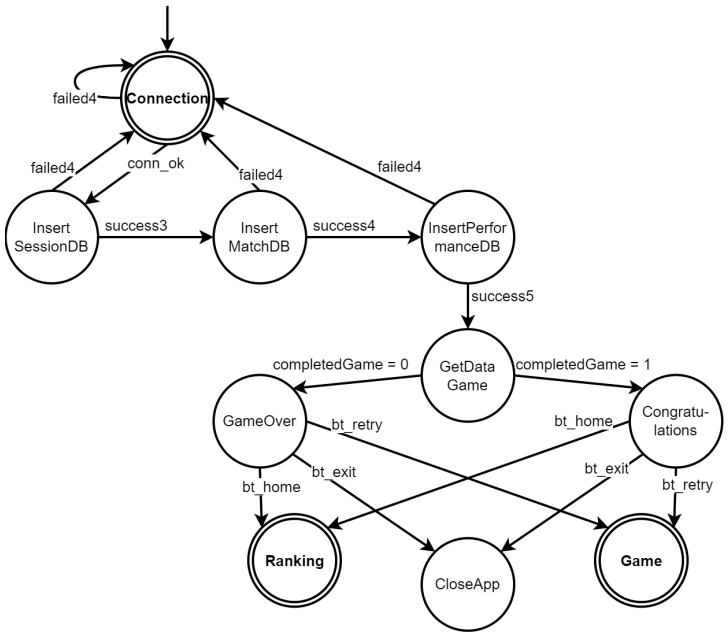


Figura 2.18: Autómata de estado finito para pantallas de final de juego.

2.4.1. Modelado del juego con Redes de Petri

Para modelar la interacción del usuario con los elementos dinámicos tridimensionales del juego, en los cuales este es libre de moverse, disparar y recargar para eliminar o esquivar las naves enemigas, se utilizan redes de Petri. Esto se debe a que la representación en autómatas de estados finitos no admite la posibilidad de alcanzar varios estados a la vez, lo cual ocurre durante la evolución del juego cuando se presentan múltiples procesos en ejecución cuya combinación habilita el disparo de las transiciones que activan nuevos estados también concurrentes. Adicionalmente, el uso de redes de Petri permite visualizar de forma más detallada la interconexión existente en esta etapa entre los elementos del juego y el jugador, así como el uso de arcos inhibidores que facilitan establecer las restricciones entre los posibles estados.

La Figura 2.19 modela los estados a los que puede acceder el jugador, los cuales, al tratarse de un juego de disparo en primera persona, están sujetos a las interacciones del usuario con la pantalla y a los recursos necesarios para el desarrollo del juego, como los puntos de vida del jugador y las balas. De igual forma, las transiciones representan los eventos propios de la evolución del juego que permiten el paso de un estado a otro.

Al ingresar en la red de Petri, el usuario se sitúa en un estado de inicio cuyo token habilita la ejecución de *StartGame*. Mediante este evento, se ubica un token en *enableGame*, indicando la activación del juego. Luego, se cargan los puntos de vida del jugador *lifePoints* con 100 tokens mediante el peso del arco que los conecta y, finalmente, se establece al jugador en un estado

disponible *Available*, donde monitorea la pantalla de juego sin pulsar ningún botón. Desde ahí, adquiere la posibilidad de disparar en ráfaga, disparar en automático o recargar. Estas acciones son descritas por las siguientes secuencias especificas:

- **Disparo en Ráfaga:** Esta interacción simula el disparo de tres balas al presionar y soltar el botón de disparo una vez. En otras palabras, desde el estado *Available*, el jugador presiona el botón *shootBurst*, lo que instaura un token en el estado *ShootingBurst*. Luego de un tiempo de espera, es activada la transición *wait*, que retorna al jugador al estado *Available*. Durante ese tiempo de espera, la misma transición *shootBurst* descuenta tres balas del cargador del arma al quitar tres tokens del estado *munition* y establecer tres tokens en el estado de balas disparadas *bulletsFired*. Esto, a su vez, habilita tres posibles eventos: *time1.5s*, que refiere al tiempo de vida de las balas donde no colisionan con ningún objeto, *PJColliderBullet*, que alude a la colisión de las balas con proyectiles enemigos, y *SSColliderBullets*, que indica la colisión de las balas con las naves espaciales enemigas, disminuyendo su número del estado *spaceships*. Cada bala destruida en cualquiera de estos eventos marca un token en el estado *BulletsRemoved*, acumulando las balas generadas necesarias para el proceso de recarga.
- **Disparo automático:** Esta secuencia replica el disparo de forma sostenida al presionar continuamente el botón disparar. Desde el estado *Available*, el jugador presiona el botón *autoShot*, el cual descuenta un token del estado *munition* y envía un token al estado *Shooting*, habilitando la ejecución del evento *btShoot_OnLonPress*, que reconoce la acción de botón sostenido y devuelve el token al estado *Shooting*, desde el cual es posible salir al activarse *btShoot_exit*, señalando que el botón de disparo no está presionado. Al mantenerse el marcaje en *Shooting*, es posible acceder en varias ocasiones a *btShoot_OnLonPress*, que a su vez descuenta tokens de *munition*, reduciendo el número de balas del cargador del arma e instaurándolas en *bulletsFired*, punto a partir del cual el comportamiento es igual al del disparo en ráfaga.
- **Recargar:** Para recargar la munición del arma, es necesario la activación del evento *recharge*. Para ello, el jugador debe estar en *Available* y el número de balas debe ser menor a 99; restricción dada por un arco inhibidor con peso de 99 desde *munition* hasta *recharge*. Luego de esta activación, es establecido un token en *Recharging*, que en conjunto con los tokens ubicados en *BulletsRemoved*, hacen posible ejecutar *fill*, restaurando *munition* a su valor inicial de 99. Para salir de esta operación, es necesario esperar un tiempo *wait* y así regresar a *Available*.

Paralelamente, los puntos de vida del jugador poseen una secuencia propia luego de ser cargados en el estado *lifePoints*, donde son disminuidos en dos tokens cuando el jugador es impactado por un proyectil. Para tal efecto, es activado *PJColliderPlayer*, que descuenta un token de *lifePoints*,

establece un token en los puntos de vida perdidos *lostLife* y marca un token en el estado perdiendo vida *LosingLife*. De forma similar, los puntos de vida del jugador son reducidos en 10 tokens cuando una nave colisiona con el jugador. Al ejecutarse *SSColliderPlayer*, se reduce un token de *lifePoints*, se elimina un token de naves espaciales *spaceships*, se incrementa un token en *lostLife* e instaura nueve tokens en *LosingLife*, donde es habilitado el evento *sub* que completa la reducción de los *lifePoints* según el número guardado en *LosingLife*. Por otro lado, los puntos de vida se incrementan en una unidad cada cuatro segundos, manteniendo un tope máximo de 100. Para este fin, debe existir un token en *eneableGame*, señalando que el juego está aún activo, y poseer tokens en *lostLife*.

El juego finaliza cuando el jugador obtiene la victoria (*completedGame=1*) o es derrotado (*completedGame=0*). En ambos casos, la red de Petri es bloqueada, informando que el juego ha terminado. Para ganar, se deben destruir todas las naves espaciales enemigas, reduciendo a cero los tokens del estado *spaceships*, lo que permite la activación de *finishGame* y alcanzar el estado *completedGame=1*. Por otra parte, el juego se pierde cuando los puntos de vida del jugador llegan a cero *lifeBar=0*, por lo cual *eneableGame* pierde su token y es marcado el estado *completedGame=0*.

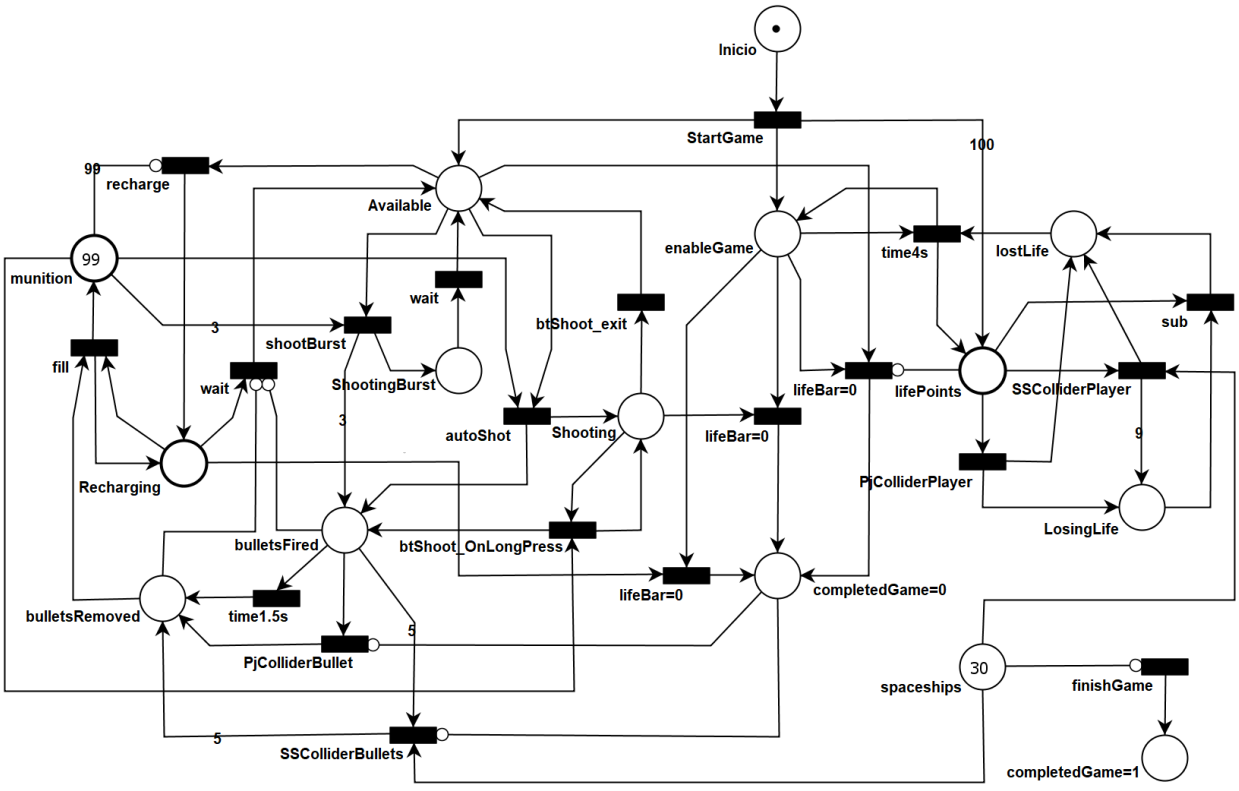


Figura 2.19: Red de Petri del usuario

La Figura 2.20 representa los estados alcanzables en el juego, de acuerdo a las acciones realizadas por las naves espaciales enemigas y los eventos de transición entre los estados. Para ello, se dispone de un token en el estado *inicio*, lo que hace posible la activación de *StartGame*. Este evento introduce un token en *eneableGame*, señalando que el juego se encuentra activo, y otro token en *GenerateHorde*, que en conjunto con el número de hordas definido en el estado *Hordes*, habilitan la ejecución de *GeneratedHorde* para la generación de las naves espaciales. Desde este evento, se origina un arco cuyo peso depende de la variable ”número de naves espaciales” previamente definido por el jugador y conecta con el estado *spaceships*, que guarda la cantidad de naves espaciales generadas. Estas naves pueden ser destruidas al colisionar con las balas del jugador a través de *SSColliderBullets*, lo que resta tokens de *spaceships* e introduce una cantidad igual de tokens en el estado naves destruidas *spaceshipsDestroyed* y en el estado *spaceshipsRemoved*. De forma similar, las naves desaparecen de la escena al colisionar con el jugador mediante *SSColliderPlayer*, lo que resta tokens de *spaceships* y los ubica en *spaceshipsRemoved*.

Adicionalmente, las naves disparan proyectiles hacia el jugador en un intervalo de cinco a ocho segundos con la activación de *time5-8s*, lo que introduce tokens al estado *projectilesFired*. Estos proyectiles pueden ser destruidos mediante tres eventos: *time3s*, en donde los proyectiles desaparecen luego de tres segundos al no impactar con el jugador, *PJColliderPlayer*, que simula el impacto del proyectil con el jugador, y *PJColliderBullet*, donde se replica el choque del proyectil con una bala disparada por el jugador.

La red de Petri cuenta igualmente con una representación del tiempo transcurrido en el juego, cuyo inicio es propiciado por el token en *eneableGame* y la ejecución del evento *time1s*, instalando cada segundo un token en el estado de tiempo jugado *timePlayed* y en el estado *TimeHordes*, que almacena tokens hasta alcanzar el peso del arco definido con anterioridad por el jugador en la variable ”tiempo entre hordas”, y lo conecta con el evento *timeComp*. Con este evento, es posible reincorporar un token en el estado *GeneratedHorde* y repetir el ciclo de generación de naves espaciales hasta remover todos los tokens de *Hordes*.

Por otra parte, el estado *TimeHordes* conecta también con el evento *maxTime* por medio de un arco con un peso residual predefinido, que hace referencia al tiempo de juego posterior a la generación de la última horda. En caso de generarse todas las hordas, es enviado un token al estado de juego ganado *completedGame=1*. Sin embargo, si se da el caso en que se active *maxTime* antes de lograrlo, el juego llega al estado de juego perdido *completedGame=0*.

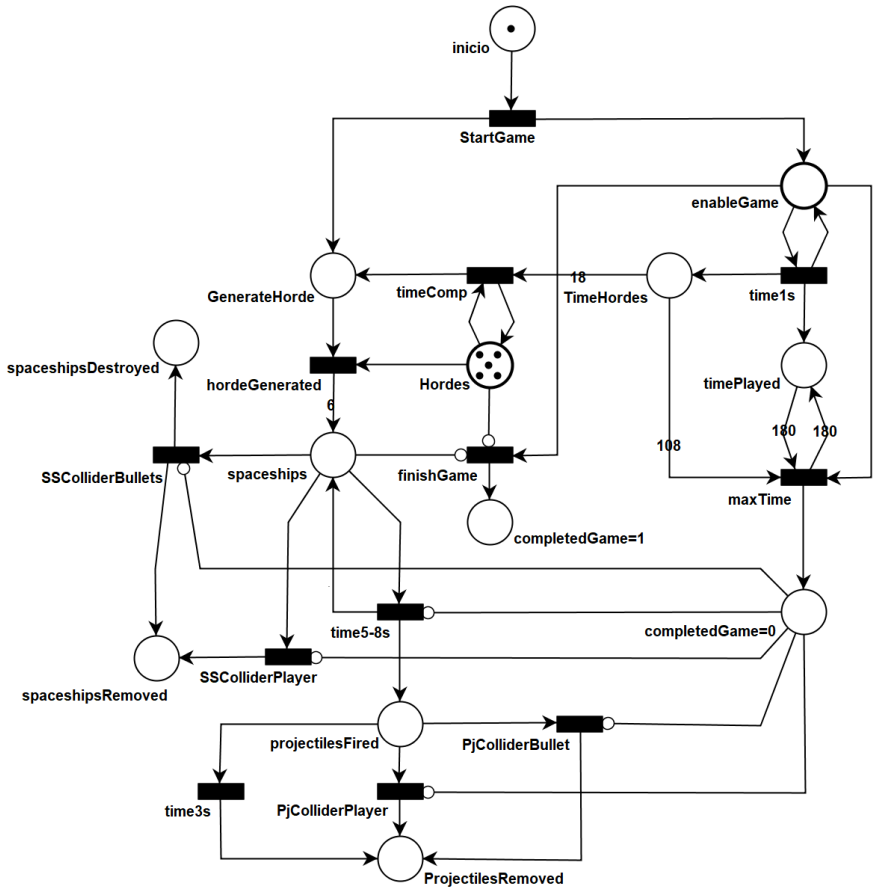


Figura 2.20: Red de Petri del las naves espaciales

La Figura 2.21 ilustra el acoplamiento entre las redes de Petri de las Figuras 2.19 y 2.20, donde se modela el comportamiento del jugador, el accionar de las naves espaciales enemigas, el uso de los recursos y el paso del tiempo en el juego. Para consolidar una única estructura representativa del juego en redes de Petri, se utilizan como puntos de interconexión los estados *inicio*, *eneableGame*, *spaceshpis*, *completedGame=0*, *completedGame=1*, así como los eventos *StartGame*, *PJColliderBullet*, *SSColliderBullets*, *PJColliderPlayer*, *SSColliderPlayer* y *finishGame*. Puesto que estos elementos son usados simultáneamente en las redes de Petri de las Figuras 2.19 y 2.20, se emplean todos sus elementos y conexiones. Sin embargo, los puntos de interconexión son implementados una sola vez, evitando así repeticiones innecesarias. Posteriormente, se realizan las correspondientes conexiones a través de arcos y sus respectivos pesos. Por último, es incorporado un nuevo estado llamado *TimeLife*, cuya función es guardar tokens hasta alcanzar el peso del arco de salida para sincronizar el tiempo de recuperación de vida con el tiempo de juego. Para ello, se realiza la vinculación por medio de un arco del evento *time1s* y le estado *TimeLife*, desde el cual se establece un arco de salida con peso igual a cuatro hasta *time4s*, para que este sea

activado cada cuatro segundos. De esta manera, se obtiene la red de Petri completa del juego.

A continuación, es presentada una de las posibles secuencias de activación de eventos que llevan al usuario a ganar el juego, representado en la Figura 2.21:

La sucesión de eventos es habilitada inicialmente por el token del estado *Inicio*. Con esto, el primer evento en dispararse es *StartGame*, permitiendo así la posterior activación de *hordeGenerated*, que instaura seis tokens en *spaceships*. Mientras tanto, el jugador pasa del estado disponible a disparo automático con *autoShot*, y activa en cinco ocasiones *btShoot_OnLongPress* seguidas de *SSColliderBullets* para destruir una de las naves establecidas en *spaceships*. Estas acciones deben repetirse hasta destruir todas las naves de la horda. Posteriormente, debe activarse el evento *time1s* en 18 oportunidades para lanzar el evento *timeComp* y poder generar una nueva horda con *hordeGenerated*, repetiendo el proceso de disparo con *btShoot_OnLongPress*. Estas secuencias se replican hasta agotar la munición, momento en el cual se debe ejecutar *btShoot_exit* seguido de *recharge*, habilitando así el evento *fill*: Sin embargo, este evento debe ejecutarse un máximo en 99 veces para completar la munición nuevamente y continuar con la destrucción de las naves espaciales mediante la serie de activaciones que permiten la generación y el disparo automático. Finalmente, al destruir todas las hordas de naves, se debe activar *btShoot_exit* y el evento *finishGame*, con lo cual se establece un token en *completedGame=1*, indicando que el juego ha sido ganado.

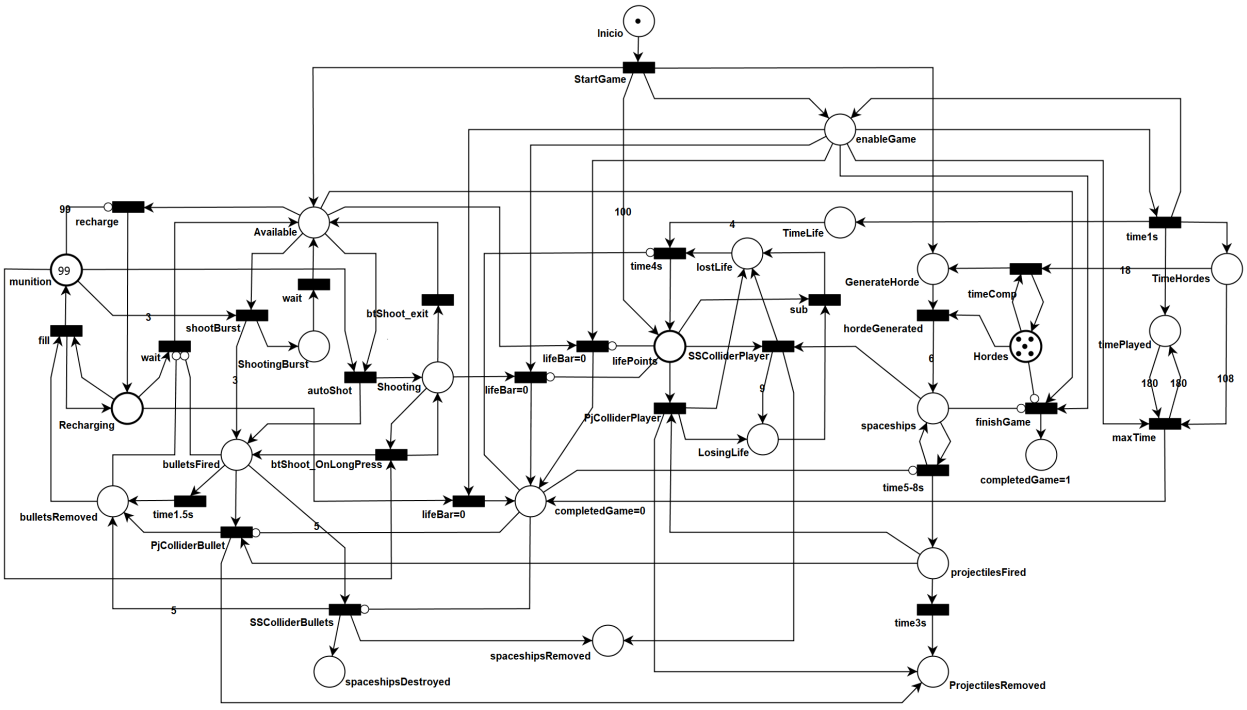


Figura 2.21: Red de Petri del juego

2.4.2. Flujos de información del juego

Los diagramas de secuencia UML (*Unified Modeling Language*) representan de forma gráfica las interacciones entre los objetos existentes dentro de un sistema al tiempo que proveen información para la comprensión del programa [39]. Se emplea el diagrama de la Figura 2.22 para modelar el envío de información del usuario entre los elementos que componen la aplicación. De esta forma, es posible observar el ingreso de los datos de registro por parte del usuario en la base de datos, haciendo uso del método *RegisterUser()* y la clase *:RegisterUserDB*. Luego, el usuario introduce sus datos para iniciar sesión, los cuales son guardados en la base de datos a través de *LoginUser()* y *:LoginUserDB*, para posteriormente guardarse en *:System*, el sistema del dispositivo móvil, usando *LoginDB*.

La selección de valores para ajustar la dificultad del juego es realizada por el usuario mediante *GameOptionMenu()* y guardada en *:SliderValues*, desde donde es enviada a *:System* con el método *SaveMatchTxt()*. Cuando inicia el juego, la clase *:GameControl* obtiene los valores ajustados por el jugador, guardados previamente en *:Sysem*, a través de *GetMatch()*, y son enviados los datos de desempeño del jugador a *:System* con *SavePerformanceTxt()*.

Al momento de finalizar la partida, se comprueba la conexión a internet con *Conection()*, y se obtiene la información de las variables seleccionadas y el desempeño del jugador, guardadas en *:System*, para almacenarlos en la base de datos según los datos de sesión también extraídos de *:System*.

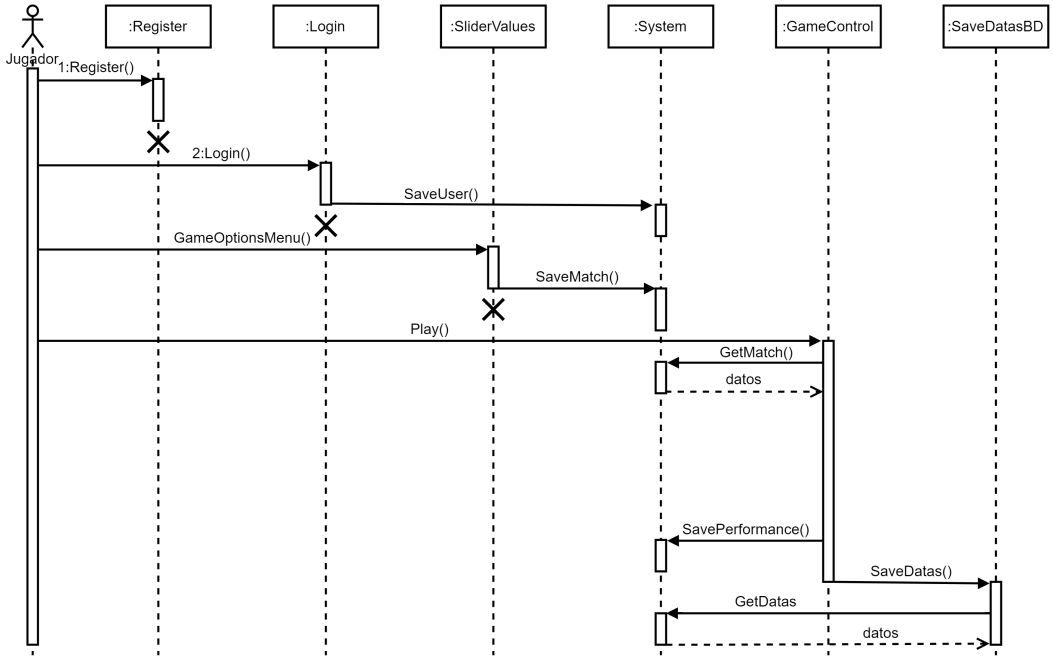


Figura 2.22: Flujos de información

2.5. Diseño y desarrollo de la Base de Datos

El juego emplea una base de datos relacional que parte de la definición de las dinámicas seleccionadas para el juego en la Tablas 2.3 y 2.4. De acuerdo a ellas, se realiza el modelo entidad-relación de la Figura 2.23, el cual permite observar cómo se relacionan entre sí los elementos de los que se requiere almacenar sus datos.

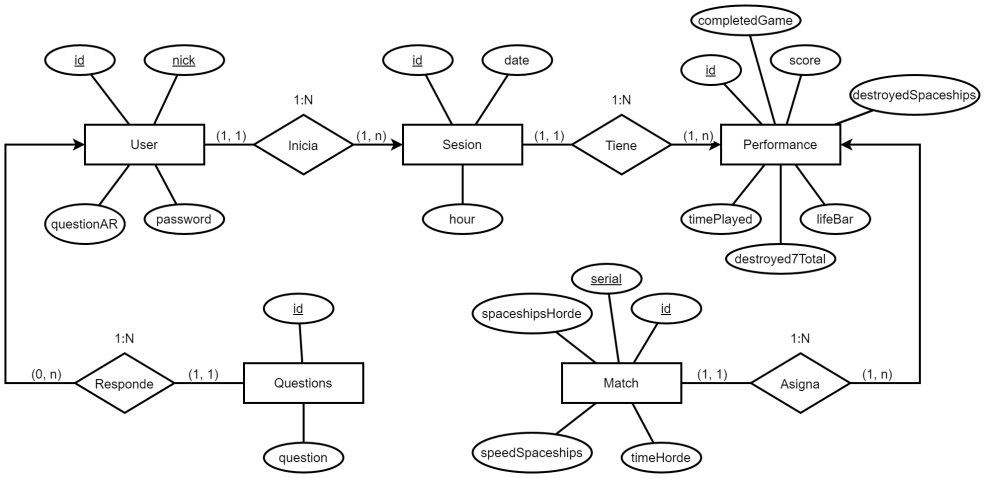


Figura 2.23: Modelo entidad-relación

El modelo lógico implementa los conceptos obtenidos del usuario que tienen la posibilidad de incrementar en el tiempo y que aporten estructuralmente a las transacciones necesarias [40]. De acuerdo al esquema de entidad-relación obtenido en la Figura 2.23, se construye el modelo lógico presentado en la Figura 2.24:

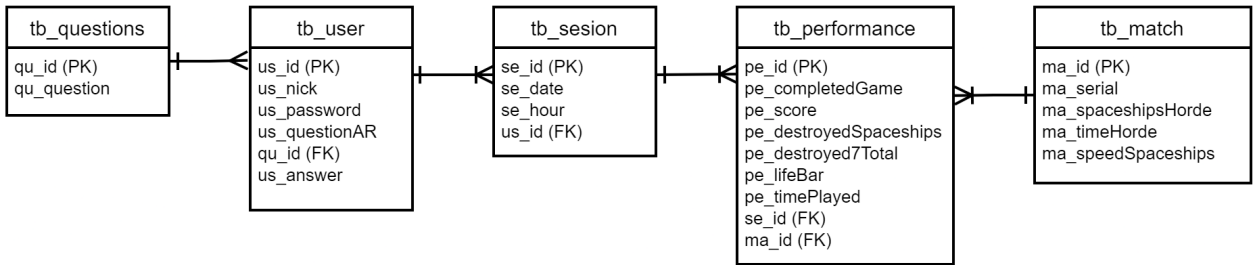


Figura 2.24: Modelo lógico de la base de datos relacional

El diseño del modelo físico de la base de datos es implementado en un sistema gestor de base de datos y toma como base la representación e implementación de los elementos del modelo lógico, considerando los datos a almacenar y las transacciones requeridas [40]. A continuación, se presenta el modelo físico obtenido:

```
CREATE TABLE tb_questions (
    qu_id INT NOT NULL AUTO_INCREMENT,
    qu_question VARCHAR(50) NOT NULL,
    PRIMARY KEY (qu_id)
);
CREATE TABLE tb_match (
    ma_id INT NOT NULL AUTO_INCREMENT,
    ma_serial VARCHAR(25) NOT NULL,
    ma_spaceshipsHorde INT NOT NULL,
    ma_timeHorde INT NOT NULL,
    ma_speedSpaceships FLOAT NOT NULL,
    PRIMARY KEY (ma_id)
);
CREATE TABLE tb_user (
    us_id INT NOT NULL AUTO_INCREMENT,
    us_nick VARCHAR(50) NOT NULL,
    us_password VARCHAR(50) NOT NULL,
    us_questionAR VARCHAR(10) UNSIGNED NOT NULL,
    qu_id INT NOT NULL,
    us_answer VARCHAR(50) NOT NULL,
    PRIMARY KEY (us_id),
    UNIQUE KEY us_nick (us_nick),
    FOREIGN KEY (qu_id) REFERENCES tb_questions (qu_id)
);
CREATE TABLE tb_sesion (
    se_id INT NOT NULL AUTO_INCREMENT,
    se_date DATE NOT NULL,
    se_hour TIME NOT NULL,
    us_id INT NOT NULL,
    PRIMARY KEY (se_id),
    FOREIGN KEY (us_id) REFERENCES tb_user (us_id)
);
CREATE TABLE tb_performance (
    pe_id INT NOT NULL AUTO_INCREMENT,
    pe_completedGame TINYINT(1) NOT NULL,
    pe_score INT NOT NULL,
    pe_destroyedSpaceships INT NOT NULL,
    pe_destroyed7Total FLOAT NOT NULL,
    pe_lifeBar INT NOT NULL,
    pe_timePlayed FLOAT NOT NULL,
    se_id INT NOT NULL,
    ma_id INT NOT NULL,
    PRIMARY KEY (pe_id),
    FOREIGN KEY (se_id) REFERENCES tb_sesion (se_id),
    FOREIGN KEY (ma_id) REFERENCES tb_match (ma_id)
);
```

La Tabla 2.9 permite detallar de forma más clara la implementación de la base de datos en el juego al relacionar los nombres asignados en la Figura 2.24 con su respectiva variable previamente definida en las Tablas 2.3 y 2.4. Los campos que contienen ”id” dentro del modelo lógico de la base de datos son identificadores de cada tabla que funcionan como llaves primarias (PK) o llaves foráneas (FK).

Variables modificables por el jugador (*match*)	**Descripción**
ma_spaceshipsHorde	Naves seleccionadas por horda
ma_timeHorde	Tiempo de aparición entre hordas
ma_speedSpaceships	Rango de Velocidad de las naves
Datos de desempeño del jugador (*performance*)	
pe_completedGame	Juego ganado o juego perdido
pe_destroyedSpaceships	Número de naves destruidas
pe_score	Puntaje de acuerdo a la partida
pe_destroyed7Total	Naves destruidas sobre (naves seleccionadas * número de hordas)
pe_lifeBar	Barra de vida
pe_timePlayed	Tiempo de duración de la partida en segundos
Datos del usuario (*user*)	
us_nick	Seudónimo elegido por el usuario para identificarse
us_password	Clave de seguridad creada por el usuario
us_questionAR	Experiencia previa en juegos RA (Si/No)
us_answer	Respuesta del usuario a la pregunta de seguridad para cambio de contraseña
Pregunta de seguridad (*question*)	
qu_question	Pregunta de seguridad para cambio de contraseña
Datos de sesión de la partida (*sesion*)	
se_date	Fecha de inicio de una sesión
se_hour	Hora de inicio de una sesión

Tabla 2.9: *Variables de la base de datos*

2.6. Desarrollo del juego en RA de disparo en primera persona

Para la elaboración de *Invasion Victory* se utilizó el entorno de desarrollo *Unity Game Engine*[1], el cual emplea componentes dinámicos 3D para la generación de contenidos virtuales interactivos [41]. El juego además incorpora la plataforma de desarrollo de aplicaciones de realidad

[1] https://unity.com/

aumentada *Vuforia Engine*[2] y la librería *ARCore* de *Unity*, con el fin de que el usuario interactúe con objetos tridimensionales dentro del área de juego.

2.6.1. Interfaces

Invasion Victory consta de cuatro escenas principales. En la primera se despliegan las pantallas relacionadas con los datos del usuario (Figuras 2.3 a 2.6). En la segunda escena se muestran las pantallas de las Figuras 2.6 a 2.10 que contienen la configuración del juego y sus características. En la tercera escena se encuentra la partida del juego en realidad aumentada con los objetos de las Figuras 2.11 y 2.12. La última escena incluye los resultados de la partida jugada (Figuras 2.13 y 2.14). A continuación se muestran las interfaces implementadas:

- La interfaz de inicio de sesión, presentada en la Figura 2.25, permite ingresar el seudónimo de usuario (*nickname*) y contraseña, los cuales son enviados a la base de datos por medio del botón *Login*. Estos datos son verificados para permitir o no el ingreso a la siguiente interfaz.

Figura 2.25: Interfaz de inicio de Sesión

- La interfaz de registro, presentada en la Figura 2.26, permite guardar seudónimos de usuario que posean un máximo de 20 caracteres entre números y letras, una contraseña que contenga de seis a 12 caracteres entre números y letras, la respuesta sobre experiencia previa en juegos RA del jugador (Si/No), así como una pregunta y una respuesta de seguridad para el cambio de contraseña. También presenta una casilla de verificación para aceptar el acuerdo de consentimiento informado. La información es almacenada en la base datos a través del

[2] https://www.ptc.com/en

botón *Register* para crear un nuevo usuario en el juego.

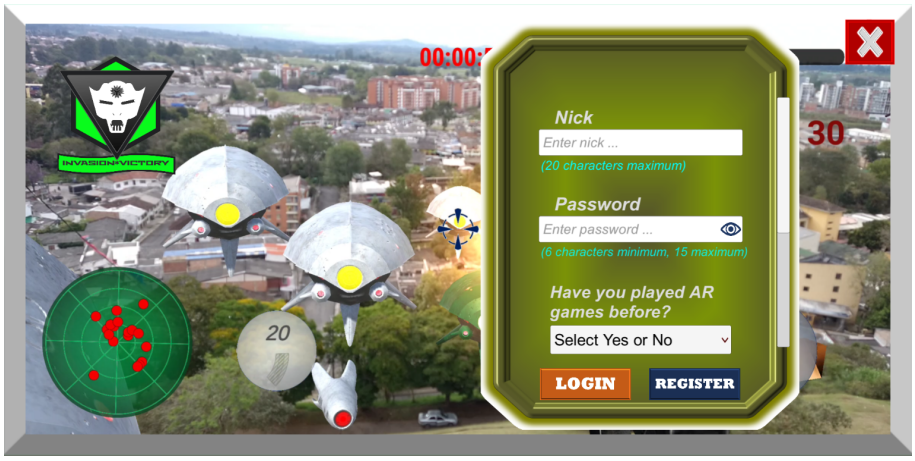


Figura 2.26: Interfaz de registro de usuario

- La interfaz de cambio de contraseña, presentada en la Figura 2.27, permite al usuario actualizar su contraseña de juego tras ingresar el *nickname*, seleccionar y responder la pregunta elegida durante la etapa de registro, ingresar una nueva contraseña con iguales características a la ingresada en el momento del registro y presionar el botón *Accept*.

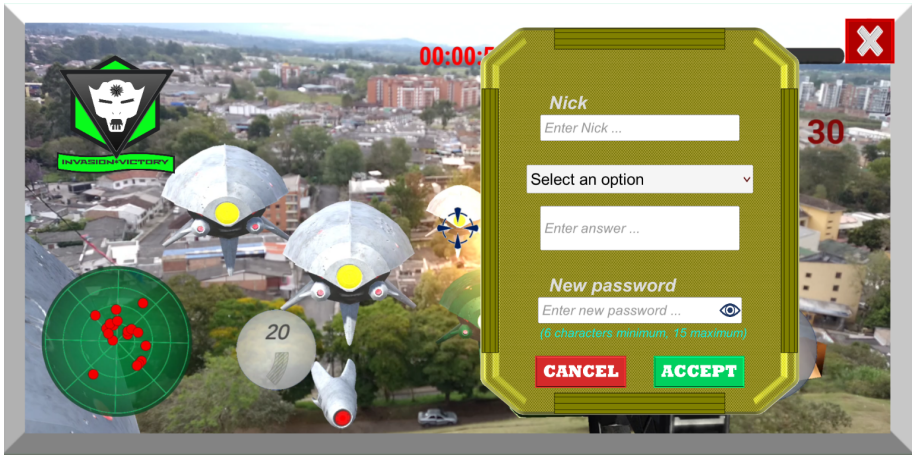


Figura 2.27: Interfaz de cambio de contraseña

- La interfaz de consentimiento informado, presentada en la Figura 2.28, comunica al usuario de forma escrita como se realizará el manejo de la información suministrada y generada durante el uso de la aplicación.

Figura 2.28: Interfaz de consentimiento informado

- La interfaz de ambientación, presentada en la Figura 2.29, ilustra a través de un vídeo la narrativa del juego.

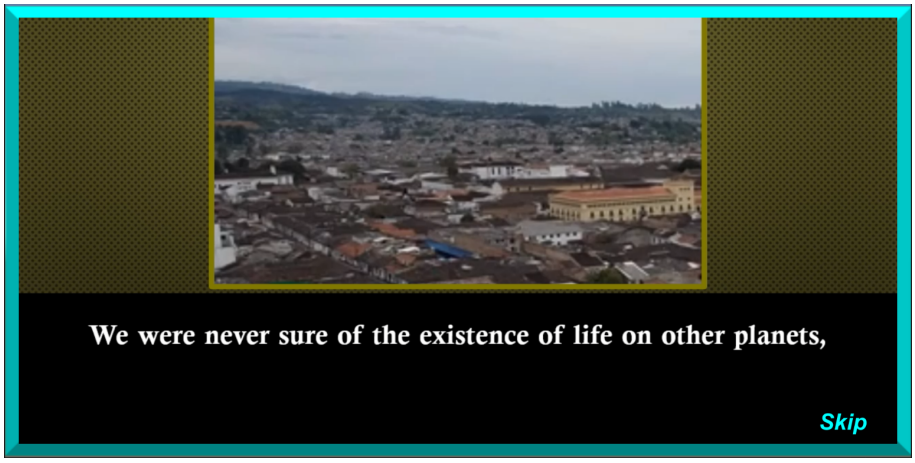


Figura 2.29: Interfaz de la historia

- La interfaz de clasificación de jugadores, presentada en la Figura 2.30, muestra una tabla en donde aparecen los 10 mejores jugadores de acuerdo con el puntaje obtenido y la posición general del jugador en el *ranking* general junto con su mejor puntuación. Adicionalmente, esta interfaz facilita el acceso a otras escenas.

Figura 2.30: Interfaz de clasificación de jugadores

- La interfaz de cambio de parámetros, presentada en la Figura 2.31, ofrece la posibilidad de escoger entre los modos de dificultad aleatorio o manual a través de un botón deslizante corto. Al aplicar el modo manual son activados tres botones deslizantes que modifican: (1) el número de naves por horda a los que se enfrenta el jugador, (2) el tiempo en segundos entre la aparición de dos hordas de naves, y (3) la velocidad en metros por segundo de las naves enemigas. Estas tres variables definen la dificultad del juego.

Figura 2.31: Interfaz de cambio de parámetros

- La interfaz de créditos de la aplicación, presentada en la Figura 2.32, muestra en texto la información de los creadores y paginas web de acceso de algunos elementos audio visuales utilizados en el juego.

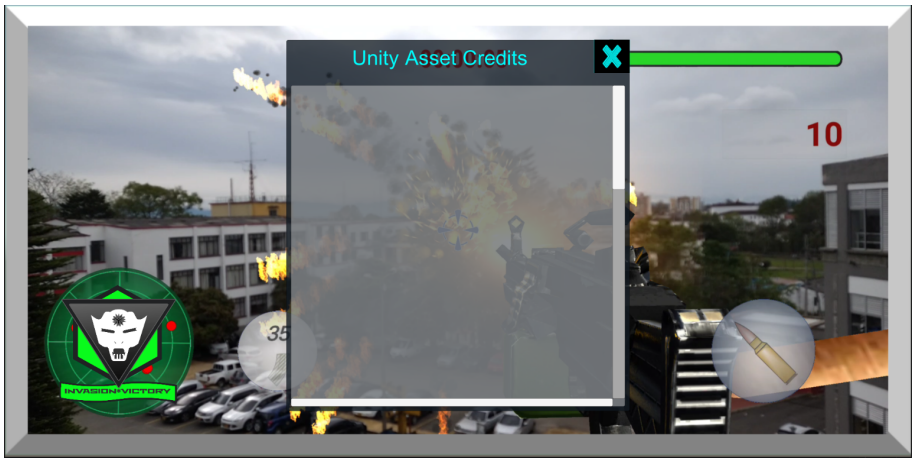


Figura 2.32: Interfaz de créditos de la aplicación

- La interfaz de instrucciones del juego, presentada en la Figura 2.33, presenta una serie de imágenes acompañadas de texto que especifican el propósito de cada elemento visual encontrado en las interfaces de *Invasion Victory*. (Ver Anexo B)

Figura 2.33: Interfaz de instrucciones del juego

- La interfaz de movimientos del jugador 2.34 presenta en vídeo el tutorial sobre los movimientos que puede realizar el usuario para interactuar con las naves enemigas y el entorno dentro juego.

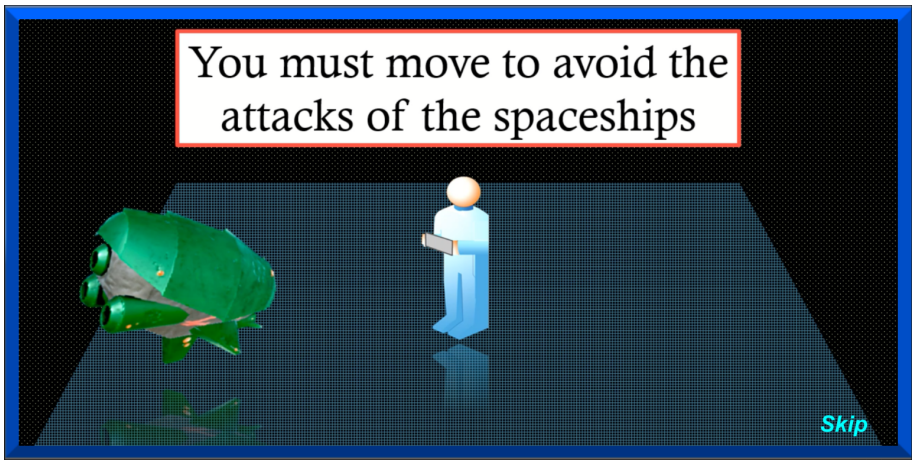


Figura 2.34: Interfaz de movimientos del jugador

- La interfaz de la partida del juego, presentada en la Figura 2.35, es la pantalla de juego donde el usuario participa activamente del disparo en primera persona. Esta cuenta con información visual sobre la cantidad de hordas que se han generado hasta el momento, las naves que ha destruido, el tiempo restante para completar el juego, la cantidad de vida que posee representada en una barra, un radar que le proporciona la posición de las naves enemigas a su alrededor, una señalización del punto hacia donde se dirigen las balas que dispara, el botón de recargar munición y el botón que le permite disparar en el entorno de realidad aumentada.

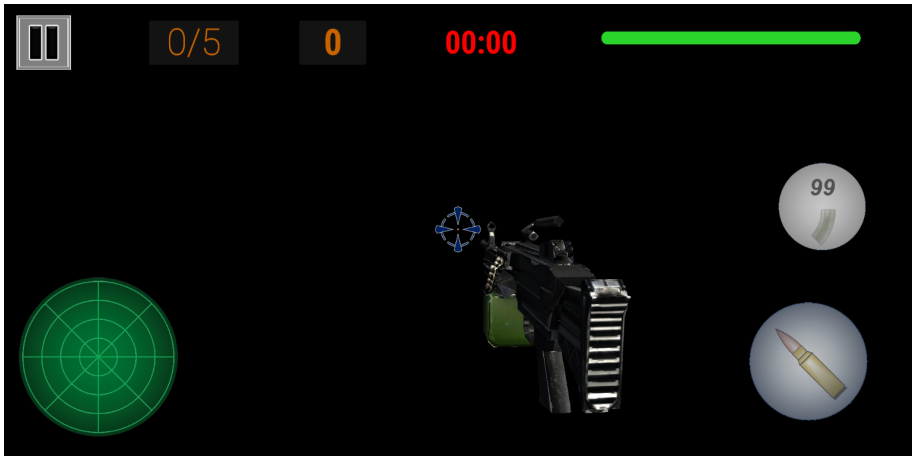


Figura 2.35: Interfaz de la partida del juego

- La interfaz de la partida en pausa, presentada en la Figura 2.36, es un menú que al desplegarse detiene por completo el juego y presenta tres opciones: el botón *resume* retoma la

partida en el momento exacto en que se encontraba antes de pulsar el botón *pause*, el botón *retry* permite reiniciar la partida con los mismo parámetros pero como un nuevo juego, y el botón *home* envía nuevamente al jugador a la interfaz de clasificación de jugadores (Figura 2.30) para que este pueda realizar otros ajustes o explorar otras interfaces.

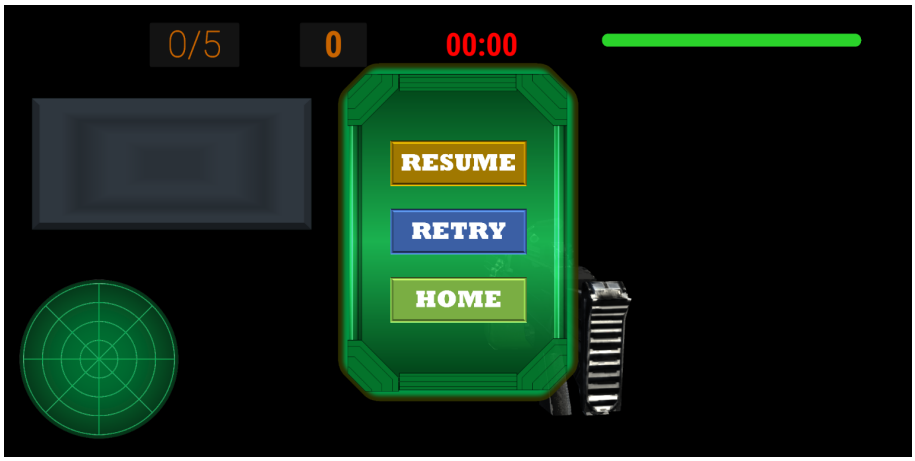


Figura 2.36: Interfaz de la partida en pausa

- La interfaz de partida perdida, presentada en la Figura 2.37, se despliega únicamente si el usuario ha perdido el juego, y presenta el puntaje obtenido en la partida junto con la duración de esta. Además, cuenta con tres botones: uno para dirigir al usuario a la pantalla de clasificación, otro para iniciar una nueva partida y el tercero permite cerrar por completo la aplicación.

Figura 2.37: Interfaz de partida perdida

- La interfaz de partida ganada, presentada en la Figura 2.38, se despliega únicamente si el usuario gana el juego. Esta interfaz muestra el puntaje obtenido en la partida junto con la duración de esta. Además, consta de los mismos tres botones que la interfaz de partida perdida (Figura 2.37).

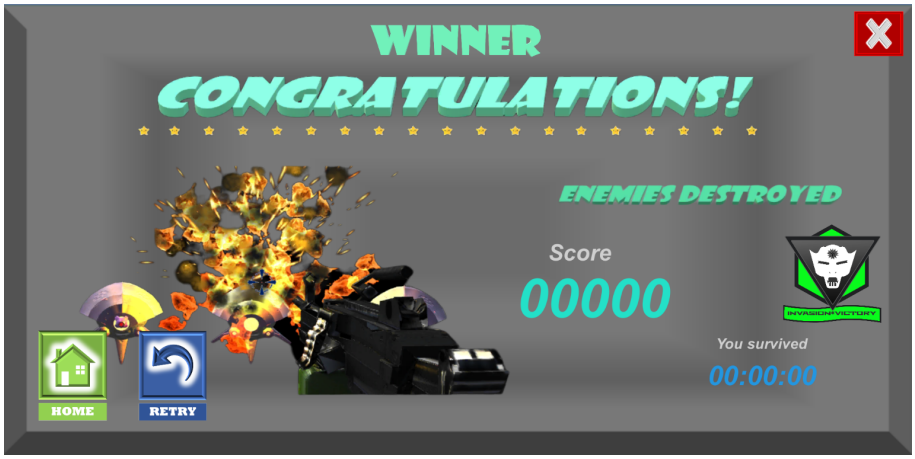


Figura 2.38: Interfaz de partida ganada

2.6.2. Desarrollo de código

Para establecer el despliegue de las interfaces de acuerdo a los autómatas de estado finito de las Figuras 2.15 a 2.18, se utilizan los estados como una serie de instrucciones en forma de métodos que ejecutan operaciones especificas. En este caso, cada método se encarga del despliegue de una de las pantallas establecidas. El *token* hace referencia a la pantalla actual en la que se encuentra el usuario. Los arcos establecen las secuencias disponibles para pasar de una pantalla a otra, y los eventos corresponden a los botones que activan el cambio de pantalla. Las funcionalidades de los botones son creadas a través del uso del lenguaje de programación C# en *Unity Game Engine*, en conjunto con el entorno de desarrollo integrado *Visual Studio*.

Las cuatro autómatas de estado finito se implementan en un solo archivo *script* de control, donde se tiene una lista definida con todos los estados existentes que serán activados individualmente por un método principal a través del accionamiento de los eventos. Cada estado de la lista ejecuta en un *script* aparte una secuencia de instrucciones asociadas al manejo de cada interfaz. La activación de cada uno de los eventos se encuentra vinculado a un único método que cambia el estado actual, llamando al método principal para ejecutar el estado siguiente.

Para la creación del código fuente que pone en funcionamiento las dinámicas de disparo en primera persona del juego, es implementada, además de las herramientas antes mencionadas, la

red de Petri de la Figura 2.21. Su funcionamiento es activado únicamente en el estado *Playing* de la Figura 2.17, en el cual:

- Los estados, al igual que en los de los autómatas, se representan como métodos que ejecutan operaciones específicas.
- Los tokens indican la activación de los estados en los que se encuentran, al igual que la cantidad de recursos disponibles y utilizados dentro del juego, como lo son los puntos de vida del jugador, la munición del arma, el número de naves, el tiempo transcurrido en segundos, el número de misiles disparados por las naves y el número de hordas.
- Los arcos o flechas corresponden a las posibles secuencias de paso entre los métodos y al uso de los recursos durante el juego.
- El peso en los arcos representa la cantidad de tokens que se adicionan o se restan de los recursos del juego. Un arco de peso unitario indica únicamente el paso de un estado a otro.
- Las transiciones son las condiciones necesarias para habilitar los métodos y hacen alusión a eventos del juego como la detección de colisiones entre objetos, condiciones en temporizadores o contadores, y pulsación de botones.

Asimismo durante el desarrollo del código fuente de *Invasion Victory*, fueron implementados los métodos de desarrollo de software *Test Driven Development* (*TDD*), encaminado a establecer el código a partir del análisis y solución de las posibles fallas para la posterior eliminación de duplicaciones [42], y *Behavior Driven Development* (*BDD*), orientado a preparar la prueba conforme a los posibles escenarios en los que funcionará el código [43]. Para más información, ver Anexo A

2.6.3. Parametrización del juego

Las tres variables que definen la dificultad del juego (número de naves espaciales por horda, tiempo de aparición entre hordas y velocidad de las naves espaciales) son ajustadas por el jugador a través de los tres controles deslizantes en la Figura 2.31. Los límites mínimo y máximo de estas variables, que equivalen al dimensionamiento del actuador en un lazo de control industrial, se definieron para asegurar que las variables controladas pudieran variar entre cero y 100 %. Una vez definidos estos rangos, se hizo necesario discretizar las amplitudes de las variables manipuladas para facilitar, en etapas posteriores, el entrenamiento del algoritmo de aprendizaje estadístico que predice el cuantil en el que se ubicará el jugador. Para obtener estos valores, cada variable fue incrementada progresivamente mientras las otras dos permanecían estáticas. De los resultados experimentales se observó que, por ejemplo, los cambios en el tiempo de aparición entre hordas consecutivas de menos de dos segundos no generaban diferencias significativas en el rendimiento. La Tabla 2.10 resume los valores obtenidos.

Variable	Límite inferior	Límite superior	Variación
Naves por horda	3	10	1
Tiempo de aparición entre hordas	6 s	30 s	2 s
Velocidad de las naves	0.4 m/s	1.2 m/s	0.1 m/s

Tabla 2.10: *Parametrización de variables*

2.6.4. Pruebas de estrés

El juego *Invasion Victory* envía los datos generados durante la partida a una base de datos relacional alojada en los servidores de *Hostinger*. Además, accede a esta base de datos para mostrar información como el registro de los mejores jugadores. Para habilitar este flujo bidireccional de información, se implementó el uso del lenguaje de programación *PHP* y el gestor de base de datos *MySQL*. Debido a la importancia crucial de la base de datos, fue necesario llevar a cabo pruebas exhaustivas para asegurar el correcto funcionamiento del servicio de *hosting*, especialmente considerando la interacción simultánea de múltiples usuarios. Además, se consideraba fundamental evaluar la capacidad de almacenamiento disponible para garantizar un rendimiento óptimo.

Con el objetivo de garantizar el correcto funcionamiento y rendimiento del servicio de *hosting*, se realizó una primera prueba de estrés para la base de datos y validación del juego con el grupo del Semillero de Realidad Virtual y Aumentada de la Universidad del Cauca. En esta prueba participaron 12 personas a quienes se les compartió el juego en sus dispositivos móviles y se les explicó el tema de investigación. Posteriormente, jugaron con la aplicación sin límite de tiempo ni número de intentos. Durante esta prueba, se encontraron fallas de funcionalidad al momento del registro y en dispositivos en los que la aplicación no funcionó correctamente debido a la falta de compatibilidad con *ARCore*. También se observaron errores intermitentes en los que el juego dejaba de funcionar y luego se corregía automáticamente. Además, se tomaron en cuenta las recomendaciones de los *testers*.

Esta información se utilizó para realizar correcciones, lo que determinó la necesidad de una segunda prueba de validación. En esta segunda prueba, participaron 10 estudiantes del mismo grupo del semillero, en las mismas condiciones que la prueba anterior. Los resultados mostraron correcciones en las fallas, excepto en lo que respecta a la incompatibilidad con *ARCore*, ya que es un elemento necesario para la configuración del juego en realidad aumentada. Además, se encontraron nuevas fallas relacionadas con el sistema de puntuación implementado, en el cual los puntos solo se otorgaban por el número de naves destruidas. Esto ocasionaba que los jugadores seleccionaran un número alto de naves, pero con valores que reducían la dificultad en las otras dos variables. Por esta razón, se vio la necesidad de establecer un nuevo sistema de puntuación utilizando los métodos TDD y BDD.

Finalmente, se realizó una tercera prueba en la que participaron seis estudiantes del grupo de semillero, en las mismas condiciones que las pruebas anteriores. En esta última prueba, no se encontraron fallas. Cabe destacar que desde el momento en que inició cada prueba de estrés hasta el inicio de la siguiente, algunos de los *testers* continuaron jugando desde sus casas.

Después de descartar los datos generados durante el chequeo de funcionalidades específicas por parte de los diseñadores y los datos erróneos originados por usuarios cuyos dispositivos móviles no eran compatibles con *ARCore*, se obtuvieron un total de 111 datos a partir de las tres pruebas.

2.6.5. Análisis estadístico de las variables de entrada y de salida

Se propone un análisis estadístico de las variables de entrada y de salida mediante la identificación de lazo abierto no lineal de la interacción entre el usuario y el juego, a partir de los datos recolectados durante las tres pruebas de estrés, para evitar una selección subóptima de variables para el posterior entrenamiento de los algoritmos de ADD. En este análisis, se tiene en cuenta únicamente el efecto de las tres variables manipuladas sobre el número de naves destruidas, debido a que esta variable de salida presenta una mayor dependencia de las entradas y, a la vez, es el objetivo principal del juego.

Para esta evaluación, los 111 datos recolectados de las anteriores pruebas de estrés fueron utilizados en el entrenamiento de un *Random Forest* (RF) en lenguaje R para la regresión. El proceso de entrenamiento integró las siguientes fases:

1. Los datos recolectados fueron repartidos aleatoriamente en dos conjuntos: el conjunto de entrenamiento, que consistió en el 70 % de los datos, y el conjunto de prueba, que comprendió el 30 % restante.
2. Se implementó un modelo *Random Forest* (RF) con el objetivo de predecir el número total de naves espaciales destruidas, el cual es un valor entero comprendido entre uno y 50. El modelo se entrenó utilizando un valor de mediana de 13 y un valor de rango intercuartílico de 14.
3. Se calculó el coeficiente de determinación (R^2) a partir de los valores medidos (y_i) y pronosticados ($\hat{y}_i$) para el número total de naves espaciales destruidas. (R^2) es una medida que varía entre cero y uno y se utiliza para evaluar el grado de concordancia entre y_i y $\hat{y}_i$.

$$R^2 = \text{máx}\left(0, 1 - \frac{\sum_{i=1}^{N_T}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N_T}(y_i - \bar{y})^2}\right) \tag{2.1}$$

Donde N_T es el número de muestras del conjunto de prueba, $\bar{y}$ es la media de ese conjunto, y la función $\max$ asegura que el valor mínimo de R^2 sea 0.

4. Mediante la implementación de la función `varImp` del paquete *caret* del lenguaje R, se calculó el poder predictivo de cada variable manipulada. Este método es conocido como evaluación de importancia variable [44, 45]

Debido a que el coeficiente de determinación y la importancia de los predictores dependen de la selección de los conjuntos de datos de entrenamiento y prueba, se hizo necesario repetir los pasos anteriores 1000 veces a fin de evitar sesgos en los resultados. Es posible observar en las Figuras 2.39 y 2.40 la variabilidad del coeficiente de determinación y la importancia de la variable para los 1000 experimentos. En la Figura 2.40 destaca la velocidad de las naves espaciales por poseer el mayor poder predictivo, sosteniéndose cerca del 100 % en las 1000 pruebas. Por otra parte, el número de naves espaciales por horda presentó un valor medio de cero y el poder predictivo más bajo, con algunos valores atípicos llegando a valores de hasta el 50 %. Finalmente, el tiempo entre hordas mostró la mayor variabilidad en todos los experimentos. La Tabla 2.11 resume la importancia de cada variable manipulada.

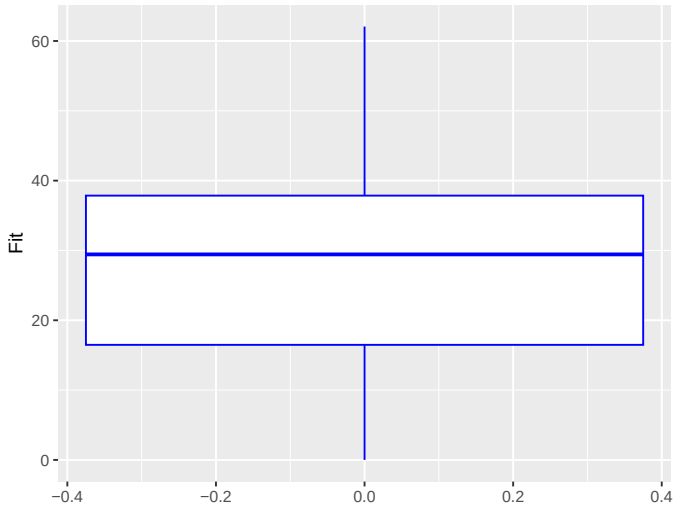


Figura 2.39: Variabilidad de R^2 para 1000 ensayos de entrenamiento y prueba.

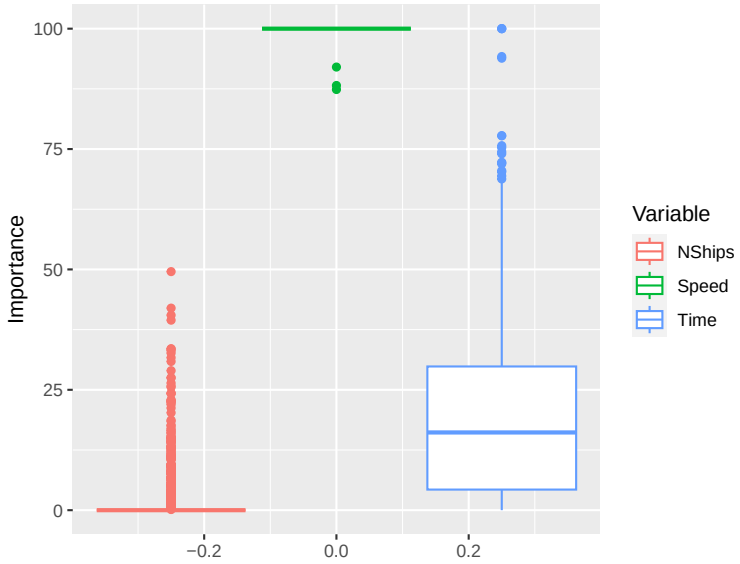


Figura 2.40: Importancia de los predictores en 1000 ensayos de entrenamiento y prueba

Donde *NShips* es el número de naves por horda, *Speed* es la velocidad de las naves, y *Time* es el tiempo entre hordas.

Variable	Min (%)	Max (%)	Mediana (%)	IQR (%)
Naves por horda	0	49.54	0	0
Velocidad de las naves	87.36	100	100	0
Tiempo de aparición entre hordas	0	100	16.14	25.56

Tabla 2.11: *Estadísticas de la importancia de los predictores en 1000 ensayos de entrenamiento y prueba.*

Los resultados experimentales obtenidos en el análisis propuesto están ligados a la relación entre cada variable y el desarrollo del juego. Por ejemplo, se observó que al aumentar la velocidad de las naves, aumenta la posibilidad de que colisionen con el jugador, pero disminuye la probabilidad de que sean destruidas. Además, debido a que el número máximo de naves a destruir es independiente de la velocidad de las naves espaciales, el incremento en esta variable solo puede afectar negativamente al usuario. A su vez, un incremento en la cantidad de naves espaciales por horda puede aumentar la posibilidad de obtener una mayor puntuación, pero así mismo esta elección puede ser contraproducente, ya que eleva la probabilidad de colisionar con una nave espacial o ser alcanzado por una bala enemiga.

Por otro lado, el tiempo entre hordas demostró tener un poder predictivo bajo (mediana = 16.4 %), ya que este valor no tiene un impacto significativo en el número máximo de naves espaciales que pueden ser destruidas; en cambio, su principal efecto es limitar la duración del juego. Los resultados obtenidos cuestionaron la selección previa de las variables manipuladas y se deter-

minó que solamente la velocidad de las naves espaciales debe ser tomada en cuenta en el futuro entrenamiento de los algoritmos de aprendizaje estadístico utilizados en esta investigación sobre el ADD.

En la Figura 2.39 se observa una falta de concordancia entre los resultados medidos y los predichos. Este fenómeno podría ser explicado por dos factores. El primero es el nivel de atención y compromiso del usuario durante el juego, que puede ser influenciado por factores externos y estados emocionales. El segundo factor es el aprendizaje, que permite la mejora de las habilidades de los usuarios de una sesión a otra. Para minimizar la influencia de estos factores en los resultados, es necesario llevar a cabo más experimentos bajo condiciones controladas y supervisadas.

Uno de los inconvenientes presentados durante el análisis estadístico de las variables de entrada y de salida fue la reducida muestra utilizada, lo cual generó una alta variabilidad en dos de los cuatro estadísticos considerados. En particular, el poder predictivo del tiempo entre hordas que fluctuó entre 0 % y 100 % en los 1000 ensayos, y el coeficiente de determinación osciló del 0 % al 62 % en los mismos ensayos. Se pretende superar esta limitación realizando una nueva verificación con una muestra de datos más amplia luego de distribuir *Invasion Victory* gratuitamente a través de *Google Play Store*.

Los resultados obtenidos durante este análisis se presentan en el artículo titulado ”*A Statistical Approach to Select the Manipulated Variables of a Difficulty Adjustment Closed-loop Control for an Augmented Reality Video Game*”(ver Anexo C). En dicho artículo, se implementa esta metodología con el propósito de eliminar variables manipuladas que tengan un efecto reducido o nulo en el rendimiento de los jugadores. Además, este enfoque representa una dirección prometedora para el diseño del ADD que incorpora principios de control automático y aprendizaje estadístico.

Capítulo 3

Desarrollo de Algoritmos Estadísticos

3.1. Protocolos para recolección de datos

El presente protocolo fue desarrollado de acuerdo a los requerimientos de la investigación.

- Recursos humanos: Investigadores y usuarios para la toma de datos.
- Recursos materiales de usuarios: Dispositivo *Android 8.0* o superior compatible con la tecnología *ARCore* y con conexión a Internet.
- Recursos materiales de investigadores: Computador con conexión a Internet.
- Recursos físicos: La aplicación se puede utilizar tanto sentado como de píe; no obstante, se sugiere contar con un área de dos metros de largo por dos de ancho y libre de obstáculos.

3.1.1. Recolección de datos

Antes de la recolección de datos:

- Los investigadores publican la aplicación en *Google Play Store* y en *Uptodown*, disponible únicamente para descarga en Colombia.
- Los usuarios descargan de forma gratuita la aplicación de las tiendas virtuales *Google Play Store* o *Uptodown* en su dispositivo *Android*.

Durante la recolección de datos:

- Los usuarios se registran en la aplicación ingresando un nombre de usuario, una contraseña, reportando su experiencia previa en juegos con RA, seleccionando una pregunta e introduciendo su respuesta para la recuperación de la cuenta.
- Los usuarios aceptan el consentimiento informado durante su registro en la aplicación. *"Entiendo que se hará uso de un 'nombre de usuario' elegido por mi, para proteger mi identidad, la confidencialidad de los datos y el derecho del anonimato. Del mismo modo, compren-*

do que los datos y resultados no serán usados para ninguna discriminación étnica, política, social, religiosa, económica ni de ninguna índole, y que podré retirarme de la investigación sin previo consentimiento ni con ningún tipo de detrimento para mí. Entiendo que los datos serán utilizados para análisis y socialización en forma grupal y que no se harán apreciaciones particulares que identifiquen a ningún tipo de participante.
Comprendo que los datos obtenidos en el estudio pueden ser publicados o difundidos con fines científicos.
Al marcar la casilla ubicada al lado izquierdo de la opción que despliega el presente consentimiento, convengo en participar en este estudio de investigación."

- Los usuarios juegan *Invasion Victory* de acuerdo a su criterio y capacidades.
- Los usuarios visualizan su entorno a través de la pantalla del teléfono móvil, a dicho flujo de imágenes se le superponen las naves que se deben destruir, las cuales aparecen en cualquier punto de los 360 grados de la periferia. En este sistema de realidad aumentada, los jugadores perciben en todo momento tanto el suelo como los objetos que los rodean, aún así, cuando el juego se utiliza de pie conlleva a un ejercicio físico moderado.
- Los investigadores revisan periódicamente la base de datos ubicada en *Hostinger*.
- Los investigadores revisan periódicamente los comentarios sobre posibles fallas del juego en el correo electrónico enlazado a la cuenta de *Google Play Console* y en la página de descarga de *Uptodown*.

Después de la recolección de datos:

- Los investigadores realizan la limpieza de los datos para el entrenamiento de los algoritmos.

Uso de los datos recolectados

Los datos recolectados serán usados para el entrenamiento de un modelo de aprendizaje supervisado basado en los algoritmos *Random Forest*, *Support Vector Machine* y *Neuronal Network* con el objetivo de integrarlos como ADD en una segunda versión del juego *Invasion Victory*.

3.1.2. Aspectos Ético Legales

En el presente proyecto, el cual se regirá por las disposiciones establecidas por el Ministerio de Salud y Protección en la Resolución 8430 de 1993, no se recolectarán datos personales del usuario, toda vez que para utilizar la aplicación *Invasion Victory* solo se requiere un nombre de usuario, el cual puede ser cualquier cadena alfanumérica de hasta 20 caracteres, como por ejemplo *abcd1234*, además de una palabra clave. Al usuario NO se le solicitará ni su nombre, ni su número de cédula, ni su teléfono, ni su edad, ni su sexo. La información que se capturará, cada

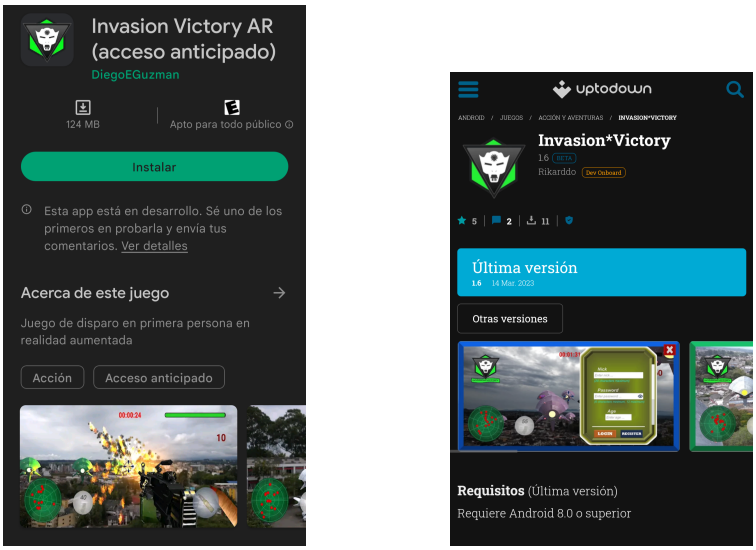
vez que el usuario utilice el juego, se limita, de una parte, a la forma en que se parametriza el nivel de dificultad (velocidad de las naves enemigas, número de naves por horda, y tiempo entre dos hordas consecutivas); y por otra parte, al desempeño (número de naves destruidas, duración de la partida, y porcentaje de vida restante). La información recolectada se utilizará para entrenar algoritmos de aprendizaje estadístico que, en la segunda versión del juego, se encargarán de adaptar automáticamente el nivel de dificultad al desempeño del usuario, con el fin de que este no se frustre o se aburra por un excesivo o por un muy reducido nivel de desafío, respectivamente. Para la segunda versión del juego, se recolectarán los mismos datos que en la primera versión, excepto que esta vez no existirán datos de parametrización definidos por el usuario.

3.2. Distribución Digital

El lanzamiento del juego se realiza en la tienda virtual de *Google Play Store*, en la cual los usuarios pueden buscar, visualizar, publicar o descargar contenido para dispositivos compatibles a través de internet. Para ello, en primer lugar, se revisaron los requisitos y condiciones dados por *Google LLC*[1] para la publicación de aplicaciones en su tienda virtual. Posteriormente, se generaron los archivos necesarios para el llenado del formulario de publicación dispuesto en *Google Play Console*. Luego, se acordó una reunión con el *Scrum Master* para el llenado del formulario, quien es poseedor de una cuenta paga de desarrollador de *Google Play Console*, y, por último, se envió al grupo de *Google LLC* encargado de la verificación de los requisitos para la publicación.

De igual forma, los investigadores publicaron la aplicación en *Uptodown*[2], un distribuidor digital de aplicaciones multiplataforma dirigidas a *Android*. Para ello, fueron revisados los requisitos y condiciones de publicación establecidas por *Uptodown*. A continuación, se creó una cuenta gratuita de desarrollador para esta tienda virtual, luego fueron generados los archivos necesarios para rellenar el formulario dispuesto en la cuenta de desarrollador, y finalmente se completó y envió el formulario para su revisión por parte del equipo de verificación de *Uptodown*.

[1] `https://play.google.com/intl/es_co/about/play-terms/index.html`
[2] `https://en.uptodown.com/aboutus/uptodown`



(a) *Google Play Store* (b) *Uptodown*

Figura 3.1: Plataformas de descarga de *Invasion Victory*

Con el objetivo de promocionar el juego, se planteó la implementación de una estrategia de marketing digital que incluyó el desarrollo de contenido visual atractivo, como vídeos y *flyers*, dirigidos a las plataformas de redes sociales más populares, como *Facebook* e *Instagram*. Asimismo, se creó una página web[3] dedicada exclusivamente al juego para brindar información detallada y una experiencia interactiva a los usuarios. Además, se buscó llegar a una audiencia más amplia mediante la distribución de volantes impresos en distintos puntos estratégicos de la Ciudad. Este tipo de publicidad se realizó desde el lanzamiento de la primera versión del juego hasta finalizar la recolección de datos de la segunda versión. Adicionalmente, dado que *Invasion Victory* es una nueva aplicación dirigida a un mercado altamente competitivo, como el sector de juegos dedicados al entretenimiento, resultó crucial impulsar su presencia para asegurar una amplia base de usuarios. Por esta razón, se implementó un torneo, con el objetivo de obtener una mayor cantidad de información y agilizar el proceso de recopilación para la primera versión del juego.

3.2.1. Actualizaciones de la Aplicación en plataformas de distribución digital.

Se realizan actualizaciones de las versiones del juego de acuerdo con las recomendaciones de los usuarios y bajo el criterio de los investigadores, con el objetivo de agregar mejoras y facilitar el uso de la aplicación para los usuarios. Para ello, se organizan reuniones con el *Scrum Master*, que cuenta con la suscripción de desarrollador en *Google Play Console*, a fin de actualizar el archivo que contiene la aplicación. Por último, se envía el archivo al equipo encargado de verificar

[3] `https://invasionvictory.semilleroarvrunicauca.com/`

los requisitos de *Google LLC* para su revisión.

De manera similar, los investigadores actualizan la aplicación en *Uptodown*, un distribuidor digital de aplicaciones multiplataforma para *Android*. Utilizando una cuenta gratuita de desarrollador para esta tienda virtual, se realiza la actualización del archivo que contiene la aplicación y se envía para su revisión por parte del equipo de verificación de *Uptodown*.

3.3. Desarrollo de Ajustes Dinámicos de Dificultad

El ajuste dinámico de dificultad (ADD) implica la capacidad de ajustar automáticamente las características, comportamientos y escenarios que determinan la dificultad de un juego a medida que se juega. Esto se logra mediante una monitorización constante del rendimiento del jugador, lo que permite evaluar su habilidad y realizar ajustes en función de su desempeño. El objetivo es adaptar el juego para proporcionar desafíos adecuados y mantener un equilibrio entre diversión y desafío, brindando así una experiencia más satisfactoria y personalizada al jugador [5].

Los pasos implementados en el presente trabajo para el desarrollo e implementación de un Ajuste Dinámico de Dificultad (ADD) comprenden: recopilación de la información; análisis estadístico de las variables de entrada y de salida; el entrenamiento de los algoritmos *Random Forest*, *Support Vector Machine* y *Artificial Neural Network*; y el despliegue de los algoritmos en el juego *Invasion Victory*.

3.3.1. Recopilación de la información

Al finalizar la estrategia de torneo aplicada para la recolección de datos, se obtuvieron un total de 1287 registros de partidas jugadas. Se realizó un filtro inicial para excluir aquellas partidas con errores de funcionamiento y las partidas de prueba efectuadas por los investigadores. Esto resultó en un total de 1157 datos correspondientes a 83 usuarios.

Posteriormente, se aplicó un segundo filtro a las partidas restantes, considerando únicamente aquellas en las que los usuarios lograron completar el juego. Esto se debe a que el torneo es un método de recopilación de datos no controlado, en el cual los participantes juegan desde cualquier ubicación con acceso a Internet, sin una supervisión directa de los investigadores. Durante el desarrollo de cada partida, pueden surgir diversos motivos que interrumpan el juego, como problemas de conexión o distracciones de cualquier tipo. Después de aplicar el segundo filtro mencionado, se obtuvo un total de 429 datos seleccionados para la siguiente fase de análisis estadístico. Estos datos cumplen con los criterios establecidos, lo cual proporciona una muestra más precisa y confiable para realizar las evaluaciones pertinentes.

3.3.2. Segundo análisis estadístico de las variables de entrada y de salida

Dado que uno de los inconvenientes presentados durante el primer análisis estadístico de variables de entrada y salida fue la reducida muestra utilizada, lo cual generó una alta variabilidad en algunos de los estadísticos considerados, fue necesario realizar un segundo análisis para ratificar o contradecir los resultados del primero y solventar el problema causado por el número de muestras.

En la Figura 3.2, se puede apreciar que los jugadores que no lograron terminar sus partidas tendieron a sobrevalorar sus habilidades al elegir valores que incrementaban la dificultad del juego, sin corresponder realmente con su nivel de destreza. Por ejemplo, en el recuadro de la esquina superior izquierda que muestra el número de naves seleccionadas, aquellos usuarios que no completaron el juego (”0”) presentan una mediana de ocho naves elegidas, en contraste con los que sí terminaron las partidas (”1”), quienes poseen una mediana de seis naves seleccionadas. Esta tendencia se refleja en otros aspectos destacados en la misma figura. En el recuadro de la esquina superior derecha, que muestra el tiempo entre hordas, se observa que los jugadores que no finalizaron el juego tienen una mediana de 14 segundos, mientras que aquellos que sí lo completaron tienen una mediana de 20 segundos. Además, en el recuadro de la esquina inferior izquierda que muestra la velocidad, se evidencia que los jugadores que no terminaron el juego presentan una mediana de 0.7 metros por segundo, mientras que aquellos que lo terminaron tienen una mediana de 0.6 metros por segundo. Por último, se destaca que los usuarios que no lograron completar el juego destruyeron menos naves en comparación con aquellos que sí finalizaron sus partidas.

En resumen, los jugadores que no completaron el juego mostraron patrones consistentes en la elección de un número mayor de naves por horda, un menor tiempo entre hordas, una velocidad más alta y una menor cantidad de naves destruidas en comparación con aquellos que lograron finalizar el juego. Esta última variable se considera el indicador de desempeño a medir, ya que representa el objetivo principal del juego.

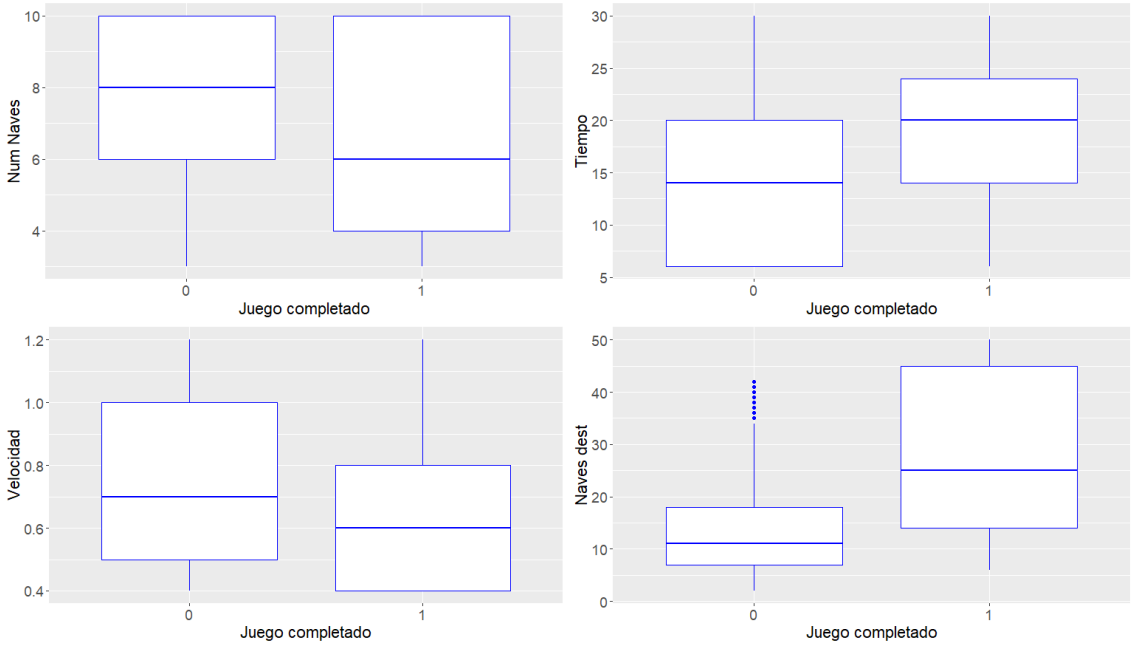


Figura 3.2: Juego completado Vs. variables de entrada y salida

Se realizaron pruebas ANOVA para determinar la significancia de los valores presentados en la Figura 3.2. Los resultados indican que en cada comparación efectuada existen diferencias significativas entre los usuarios que completaron las partidas y aquellos que no lo hicieron. Los resultados de las pruebas ANOVA se presentan en la Tabla 3.1:

Variables	**Significancia**
Naves por horda	$< \ 2 * e^{-16}$
Tiempo entre hordas	$< \ 2 * e^{-16}$
Velocidad	$4.45 * e^{-09}$
Naves destruidas	$< \ 2 * e^{-16}$

Tabla 3.1: *Resultados de significancia*

En la Figura 3.3, se puede observar en la gráfica ubicada en la parte superior una relación que se asemeja a una línea recta entre el número de naves por horda y la cantidad de naves destruidas por el usuario. Además, los diagramas de caja revelan una dispersión mínima de los datos, sin presentar valores atípicos. Por lo tanto, esta estrecha relación sugiere que el número de naves por horda es una variable adecuada para controlar la variable de salida.

Por otro lado, en la gráfica del centro, se muestra la relación entre el tiempo entre hordas y las naves destruidas. En este caso, no se observa una tendencia clara en el cambio de naves destruidas en función del tiempo entre hordas. Además, se pueden identificar valores atípicos en los tiempos de seis, 10 y 14 segundos, mientras que a partir de los 16 segundos en adelante, los

valores presentan una alta dispersión. Estos hallazgos indican que la variable del tiempo entre hordas no es apropiada para controlar la variable de salida.

En la gráfica ubicada en la parte inferior, se muestra la relación entre la velocidad de las naves y el número de naves destruidas. Aquí se puede observar una alta dispersión en el número de naves destruidas para velocidades que van desde 0.4 metros por segundo hasta 0.8 metros por segundo. Además, se presentan valores atípicos para velocidades de 0.9, 1.1 y 1.2 metros por segundo. Por lo tanto, según este análisis, la velocidad de las naves también se descarta como una variable adecuada para controlar la variable de salida.

Con base en el análisis anterior, se determina que los algoritmos de ajuste dinámico de dificultad se entrenarán utilizando como variable manipulada el número de naves por horda seleccionadas por el usuario y como variable de respuesta el número de naves enemigas destruidas por el usuario. En cuanto a las variables de tiempo entre hordas y velocidad de las naves, se les asignarán valores constantes de 20 segundos y 0.6 metros por segundo, respectivamente. Estos valores representan la mediana de las partidas finalizadas, como se evidencia en la Figura 3.2.

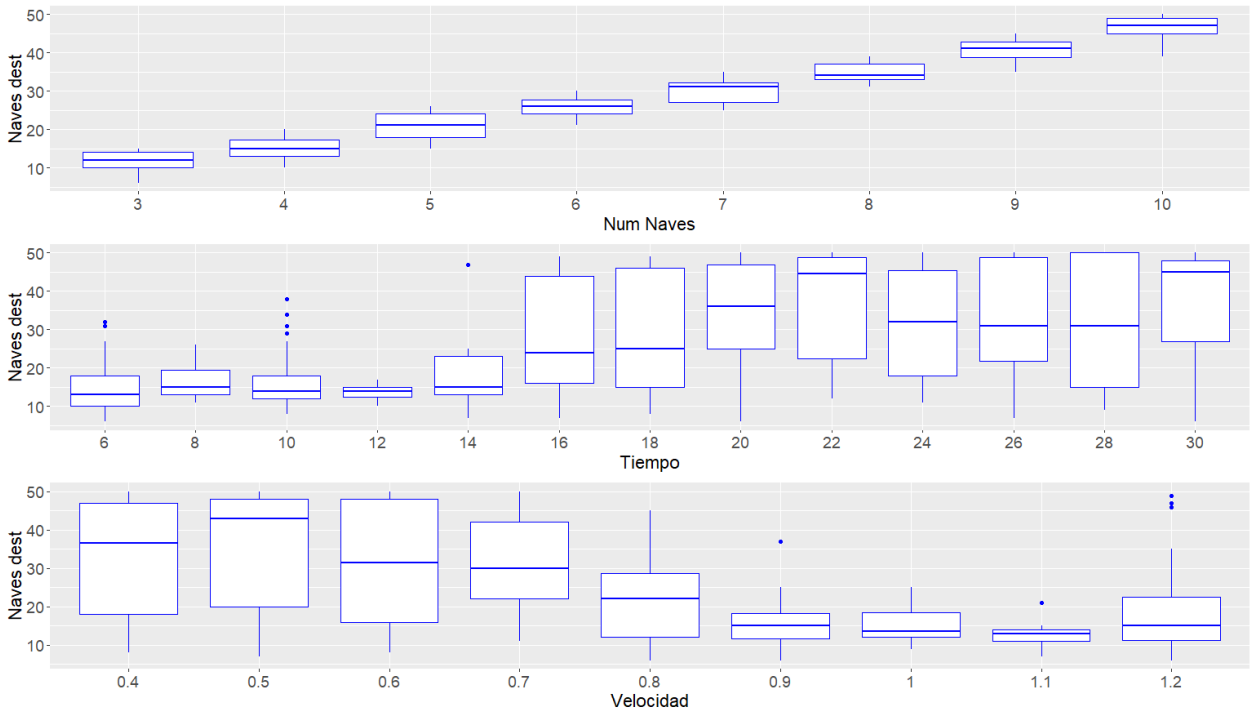


Figura 3.3: Variables de entrada Vs. naves destruidas

En conclusión, tras revisar y comparar ambos análisis estadísticos realizados a las variables de entrada y salida, se puede inferir que en el primer análisis se consideraron todos los datos, sin importar si el jugador completó o no su partida debido a distracciones o fallos de conexión. Dado que ambos experimentos no son completamente controlados, el filtro aplicado en el segundo

análisis garantiza una mayor fiabilidad de los datos. Por lo tanto, se decide descartar el análisis inicial y efectuar el ajuste dinámico de dificultad de acuerdo con los hallazgos del segundo análisis.

3.3.3. Entrenamiento de Algoritmos

Una vez identificada la variable manipulada que permitirá realizar el ajuste de la variable de salida, se procede a realizar el entrenamiento de los algoritmos a implementar. En este caso específico, se detalla a continuación el tipo y la descripción de cada algoritmo:

- Bosque Aleatorio (RF, por sus siglas en inglés): es una colección de varios árboles de decisión. Cada árbol se construye de forma independiente utilizando diferentes subconjuntos de datos de entrenamiento y características. Luego, las predicciones de todos los árboles se combinan para obtener una predicción final. Esta combinación de múltiples árboles de decisión mejora la precisión de las predicciones en comparación con un solo árbol de decisión, ya que cada árbol puede capturar diferentes aspectos o patrones de los datos.
  Un árbol de decisión, por su parte, es un modelo de aprendizaje automático que se construye mediante una serie de decisiones basadas en valores variables. Cada decisión en el árbol representa una pregunta o una condición sobre las características del problema. Estas decisiones conducen a diferentes caminos en el árbol, hasta llegar a una predicción o resultado final [46].
- Máquina de Soporte Vectorial (SVM, por sus siglas en inglés): es un algoritmo de aprendizaje automático utilizado para crear modelos de clasificación y regresión. En el caso de la clasificación, si se tiene un conjunto de datos con dos clases que pueden ser separadas linealmente, hay múltiples líneas que pueden usarse para distinguir entre ambas clases. El objetivo de la SVM es encontrar la línea que mejor separe las dos clases, la cual se llama hiperplano en dimensiones superiores [47].
- Red Neuronal Artificial (ANN, por sus siglas en inglés): es una red compuesta por nodos que imitan a las neuronas biológicas del cerebro humano. Estos nodos están interconectados en diferentes capas, como la capa de entrada, la capa oculta y la capa de salida. Durante el proceso de aprendizaje, los pesos y sesgos de las conexiones entre los nodos se ajustan iterativamente para mejorar la capacidad de la red para clasificar y tomar decisiones. Una red neuronal típica tiene muchas neuronas artificiales y aprende mediante el ajuste de pesos y sesgos en cada capa para obtener resultados óptimos [48].

El entrenamiento de los algoritmos se lleva a cabo en el entorno de desarrollo *R Studio*, utilizando el Lenguaje R. El primer paso implica la división de los datos filtrados durante el segundo análisis estadístico en dos conjuntos: el conjunto de entrenamiento, que representa el 70 % de los datos, y el conjunto de prueba, que abarca el 30 % restante. Se aplica validación cruzada en 10 ocasiones

para identificar la interacción entre el juego y el jugador a través de tres algoritmos: *Random Forest*, *Support Vector Machine* y *Artificial Neural Network*. Estos algoritmos se emplean para predecir el número de naves destruidas en función del tiempo entre hordas, la velocidad de las naves y la cantidad de naves por horda. Posteriormente, se desarrolla una función inversa que determina el controlador. En esta función se extraen los datos correspondientes al número de naves destruidas, y se fijan valores constantes de 20 segundos para el intervalo de tiempo entre hordas y 0.6 metros por segundo para la velocidad de las naves, sobre los cuales se realiza el cálculo del número de naves por hordas. Los porcentajes de ajuste de los modelos obtenidos a partir del entrenamiento para predecir los valores de naves por horda, se encuentran detallados en la Tabla 3.2:

Algoritmo	**Ajuste (%)**
Random Forest	96.7 %
Support Vector Machine	97.0 %
Neuronal Network	96.5 %

Tabla 3.2: *Porcentaje de ajuste de los algoritmos*

Después de entrenar y realizar pruebas para evaluar los resultados, se observa que el valor de las naves destruidas por el usuario oscila entre 10 y 50, aumentando en incrementos enteros. Estos valores de entrada y las salidas correspondientes son finitos, lo cual representa una ventaja al momento de implementar el ajuste de la dificultad en el juego, ya que es posible determinar las salidas esperadas para cada valor de entrada en función de los resultados proporcionados por cada algoritmo. Por consiguiente, se ha tomado la decisión de generar e implementar tablas que relacionen cada entrada con su respectiva salida. Esto permite reducir el uso de recursos en el juego, ya que en lugar de agregar y ejecutar un algoritmo, el juego simplemente debe realizar una consulta en una tabla predefinida.

Una vez obtenidas las tablas que permiten definir el número de naves por horda en función del número de naves destruidas, se procede a establecer intervalos regulares de naves destruidas basados en la diferencia mínima posible entre el número de naves destruidas para cada cantidad de naves por horda, teniendo en cuenta que el número de hordas se mantiene constante en cinco. Por lo tanto, para cada incremento de cinco naves destruidas, se busca el valor correspondiente de naves por horda seleccionadas.

Se muestra a continuación la Tabla 3.3 con los valores de naves por horda entregados por los diferentes algoritmos, la velocidad, el tiempo y valores de naves destruidas.

Naves dest.	Tiempo	Velocidad	Num Naves RF	Num Naves SVM	Num Naves ANN
10	20	0.6	3	3	3
15	20	0.6	3	4	4
20	20	0.6	4	5	5
25	20	0.6	5	6	6
30	20	0.6	6	7	7
35	20	0.6	8	8	8
40	20	0.6	10	9	9
45	20	0.6	10	10	10
50	20	0.6	10	10	10

Tabla 3.3: *Tabla resultante del entrenamiento*

3.4. Despliegue de tablas obtenidas a partir de los algoritmos

A partir de la Tabla 3.3, se seleccionan los valores de las columnas correspondientes al número de naves por horda, y se eliminan los datos duplicados para obtener los distintos niveles de dificultad para cada uno de los tres algoritmos. El resultado de este proceso se muestra en la Tabla 3.4. Por otra parte, las variables de tiempo entre hordas y velocidad de las naves espaciales se mantienen en los valores de 20 segundos y 0.6 metros por segundo, respectivamente, para cada uno de los niveles obtenidos.

Random Forest	***Support Vector Machine***	***Neuronal Network***
3	3	3
4	4	4
5	5	5
6	6	6
8	7	7
10	8	8
	9	9
	10	10

Tabla 3.4: *Niveles de dificultad de los ADDs*

Una vez obtenida la tabla con los niveles de dificultad a partir de los resultados entregados por cada algoritmo, se procede a su incorporación en el juego *Invasion Victory*. Inicialmente, el usuario juega con una configuración predeterminada en el nivel inicial, con un número de tres naves por horda. A partir de este punto, para comprobar que el ajuste hecho por el ADD funciona correctamente, los investigadores proponen dos límites en porcentaje, dentro de los cuales se pretende mantener al jugador: el 60 % y el 80 %, acuerdo al número de naves enemigas destruidas. Por lo cual, al finalizar cada partida, se verifica si el usuario destruyó más del 80 % del total de las naves

enemigas. En caso afirmativo, su nivel de dificultad se incrementa al valor inmediatamente superior de la tabla en la siguiente partida. Por otro lado, si al finalizar la partida el usuario destruyó menos del 60 % del total de las naves, el nivel de dificultad disminuye al valor inmediatamente inferior de la tabla en la siguiente partida. Por último, si el usuario destruye entre el 60 % y el 80 % del total de las naves, el nivel de dificultad se mantiene.

Debido a que las tablas presentan valores límites de tres naves por horda y 10 naves por horda, si un usuario llega a alguno de estos límites y, según su rendimiento, es necesario disminuir el nivel de dificultad mientras se encuentra en tres naves o, por el contrario, es necesario aumentar el nivel de dificultad estando en 10 naves por horda, el nivel de dificultad se mantendrán en esos valores, ya que no es posible disminuir o incrementar más allá de estos límites. Lo anterior se puede visualizar en la Figura 3.4.

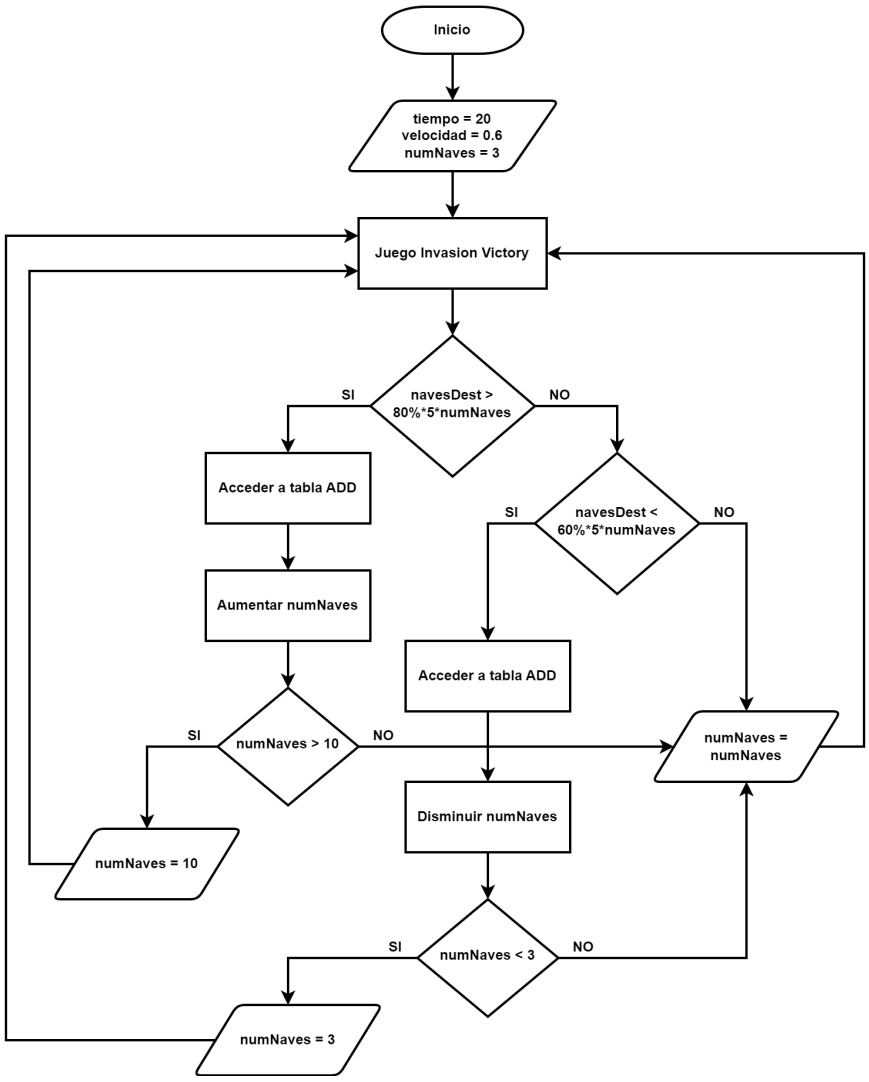


Figura 3.4: Funcionamiento general de los ADDs

Asimismo, se realizan ajustes al juego en cuanto a la modificación de pantallas, ya que ahora no es necesario que el jugador ajuste parámetros de dificultad. Además, se agregan elementos para hacer el juego más llamativo, como la división en ocho niveles según el rendimiento del usuario. También se incorpora nuevo armamento adicional. Por último, se incluye una sección dedicada a realizar una encuesta de usabilidad, la cual aparecerá a cada jugador al ingresar por segunda vez en la aplicación.

Capítulo 4

Comprobación de usabilidad, Compromiso y Desempeño

Después de haber integrado el ADD en el juego *Invasion Victory* y haber realizado los ajustes necesarios para las mejoras, se procede a actualizarlo en las plataformas de distribución digital, siguiendo el mismo procedimiento que se ha implementado en actualizaciones anteriores 3.2.1. Posteriormente, se continúa con la estrategia de marketing digital, ahora enfocada en la segunda versión del juego con ADD. Sin embargo, para esta segunda versión, no se aplicará la estrategia de recolección de datos a través de torneo, ya que la motivación basada en premios podría afectar el nivel de compromiso, sesgando así los resultados.

Luego de aplicar la estrategia de marketing digital, se obtuvieron 104 datos de rendimiento referentes a 44 usuarios registrados. De estos datos, se eliminaron las partidas iniciales de cada usuario, ya que el algoritmo de ADD realiza los ajustes a partir de la segunda partida, mientras que la primera tiene una configuración predeterminada. Luego, se aplicó un segundo filtro en el que se eliminaron las partidas de jugadores que presentaban un número de naves destruidas igual a cero, ya que dicho resultado implica que estos usuarios se presentan como espectadores casuales y no participan en el juego.

En el siguiente análisis estadístico, se explora el compromiso y el desempeño de un grupo de usuarios en el juego *Invasion Victory*, un juego en realidad aumentada de disparo en primera persona. En particular, se investiga cómo el número de partidas jugadas y el número de naves destruidas están relacionados con el algoritmo de ajuste dinámico asignado a cada jugador. Los algoritmos de ajuste dinámico en cuestión son *Random Forest*, *Support Vector Machine* y *Artificial Neural Network*. A cada usuario se le ha asignado aleatoriamente uno de estos algoritmos, sin que conozca su especificidad.

El objetivo de este estudio es examinar si el algoritmo de ajuste dinámico tiene un impacto significativo en el compromiso del jugador, medido por el número de partidas jugadas, y en su desempeño, medido por el número de naves destruidas. Además, se realiza una encuesta para evaluar

la usabilidad del juego, lo que permite obtener información adicional sobre la experiencia general de los usuarios.

4.1. Compromiso

Con el propósito de evaluar el compromiso de los usuarios que aportan datos relevantes para esta investigación, se presenta a continuación la Tabla 4.1, donde se encuentran los usuarios con su número de identificación dentro de la base de datos ID. Los usuarios son agrupados según el algoritmo asignado, así como el número de sesiones (es decir, el número de veces que ingresó a la aplicación) y partidas jugadas por cada uno de ellos. Estos datos permiten determinar el grado de participación y dedicación que un jugador tiene hacia el juego, además de cuán involucrado y motivado está para seguir jugando. La tabla muestra que los jugadores asignados a los algoritmos *Random Forest* y *Support Vector Machine* ingresaron a la aplicación una o dos veces como máximo, mientras que aquellos que utilizaron *Artificial Neural Network* lo hicieron entre una y siete veces. En cuanto al número de partidas, los usuarios que utilizaron *Random Forest* jugaron entre una y siete partidas, los que emplearon *Support Vector Machine* jugaron entre una y nueve partidas, y los usuarios de *Artificial Neural Network* jugaron entre tres y quince partidas en sus sesiones.

ID Usuario	Sesiones	Partidas
Random Forest		
17	1	3
23	1	2
29	1	3
41	1	1
51	2	5
53	1	7
Support Vector Machine		
18	1	3
20	1	1
26	1	1
55	2	9

Continúa en la siguiente página.

ID Usuario	Sesiones	Partidas
Artificial Neuronal Network		
21	6	12
48	7	15
50	1	3
52	3	5

Tabla 4.1: *Datos de compromiso*

Para observar de forma más clara el compromiso según cada algoritmo asignado, se presenta en la Tabla 4.2 el total de usuarios, el total de sesiones, el promedio de sesiones, el total de partidas y el promedio de partidas para cada tipo de algoritmo.

Algoritmo	Usuarios	Sesiones	$\frac{\text{Sesiones}}{\text{Usuarios}}$	Partidas	$\frac{\text{Partidas}}{\text{Usuarios}}$
Random Forest	6	7	1.2	21	3.5
Support Vector Machine	4	5	1.3	14	3.5
Neuronal Network	4	17	4.3	35	8.8

Tabla 4.2: *Promedios de compromiso por algoritmo*

4.2. Desempeño

Para determinar el efecto del algoritmo en el desempeño del grupo de usuarios, se evalúa el número de naves espaciales destruidas según el algoritmo asignado. La variable ”número de naves destruidas” es la establecida para medir el desempeño, dado que el objetivo principal del juego es destruir la mayor cantidad de naves posibles. En la Figura 4.1, se puede observar que los usuarios que emplearon el algoritmo *Random Forest* presentan un valor central de la distribución de sus datos en 18 naves destruidas, abarcando un rango de entre una y 33 naves. En contraste, los usuarios a los que se les asignó el algoritmo *Support Vector Machine* muestran una mediana de 17 naves destruidas, con un margen que oscila entre ocho y 26 naves. Por otra parte, los usuarios que utilizaron la *Neural Network* tienen una mediana de 20 naves destruidas y abarcan un rango que va de 10 a 42 naves destruidas, destacando un valor atípico de 45 naves destruidas.

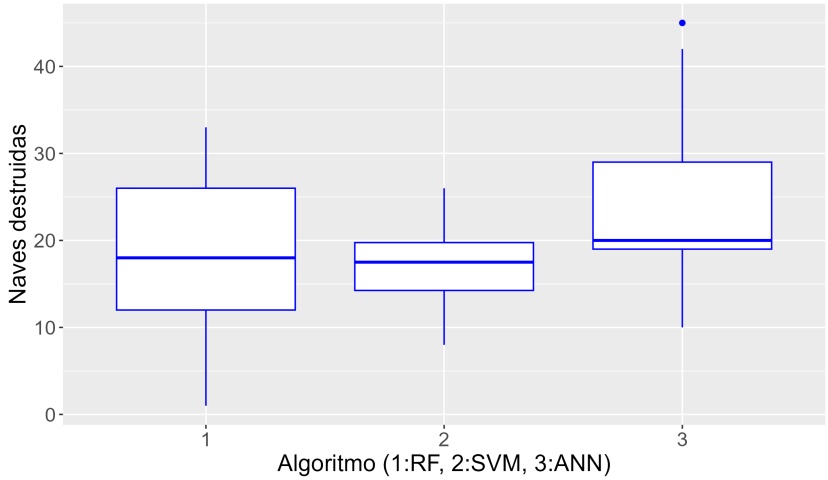


Figura 4.1: ADD vs naves destruidas

Al aplicar una prueba ANOVA a los datos anteriores se obtuvo una significancia de 0.00798, lo que indica que existen diferencias significativas entre al menos dos de los algoritmos de ajuste dinámico de dificultad. Para determinar entre que algoritmos se presentaron dichas diferencias se aplicó la prueba de la diferencia significativa honesta de Tukey, a partir de la cual se obtuvo la Tabla 4.3:

Algoritmo 1	**Algoritmo 2**	**Significancia**
Support Vector Machine	*Random Forest*	0.99
Neuronal Network	*Random Forest*	0.02
Neuronal Network	*Support Vector Machine*	0.04

Tabla 4.3: *Resultados de la prueba Tukey*

La implementación del ADD en los juegos consiste en ajustar dinámicamente la dificultad del juego en función de la habilidad y desempeño del jugador para mantenerlo dentro de canal de flujo [1]. Por esta razón, resulta imprescindible evaluar cómo el algoritmo ha mantenido a los usuarios dentro de los porcentajes estipulados en la Figura 3.4, que oscilan entre el 60 % y 80 %. Estos valores de dificultad determinan el canal de flujo en el juego *Invasion Victory*, cuyo objetivo es asegurar que el jugador no experimente aburrimiento debido a la falta de desafío o se sienta abrumado por una dificultad demasiado alta en relación a sus habilidades. La Figura 4.2 muestra de manera gráfica el canal de flujo de *Invasion Victory*.

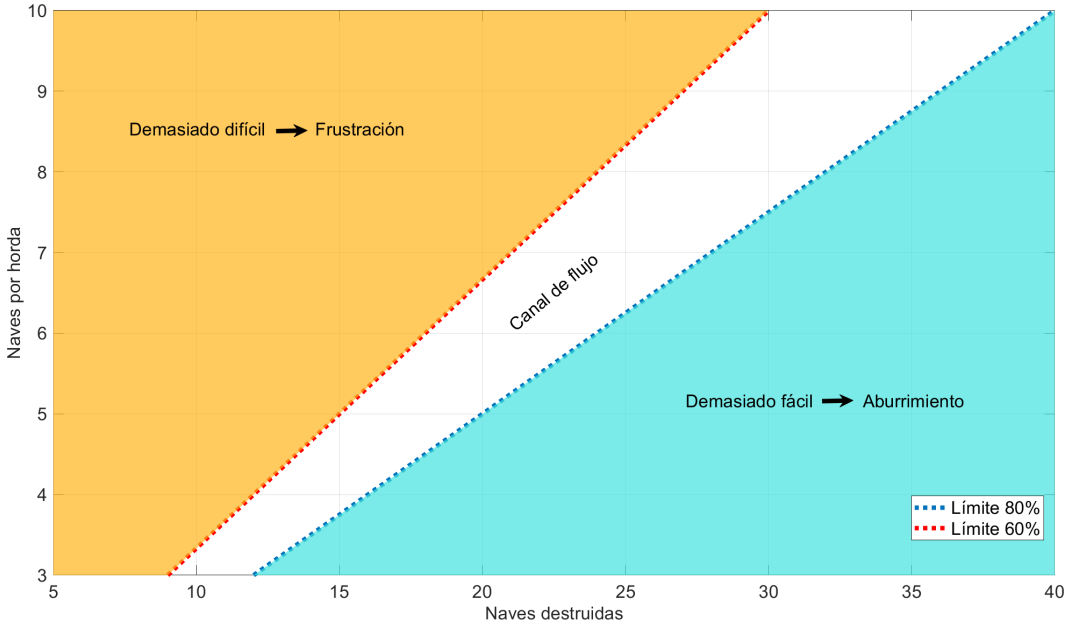


Figura 4.2: Canal de flujo de *Invasion Victory*

A continuación, se presenta la evolución de los jugadores en relación con el canal de flujo durante las partidas para cada algoritmo implementado. Cada punto en las figuras representa una partida con un número específico de naves destruidas. Las Figuras 4.3, 4.4 y 4.5 corresponden a los algoritmos *Random Forest*, *Support Vector Machine* y *Neural Network*, respectivamente.

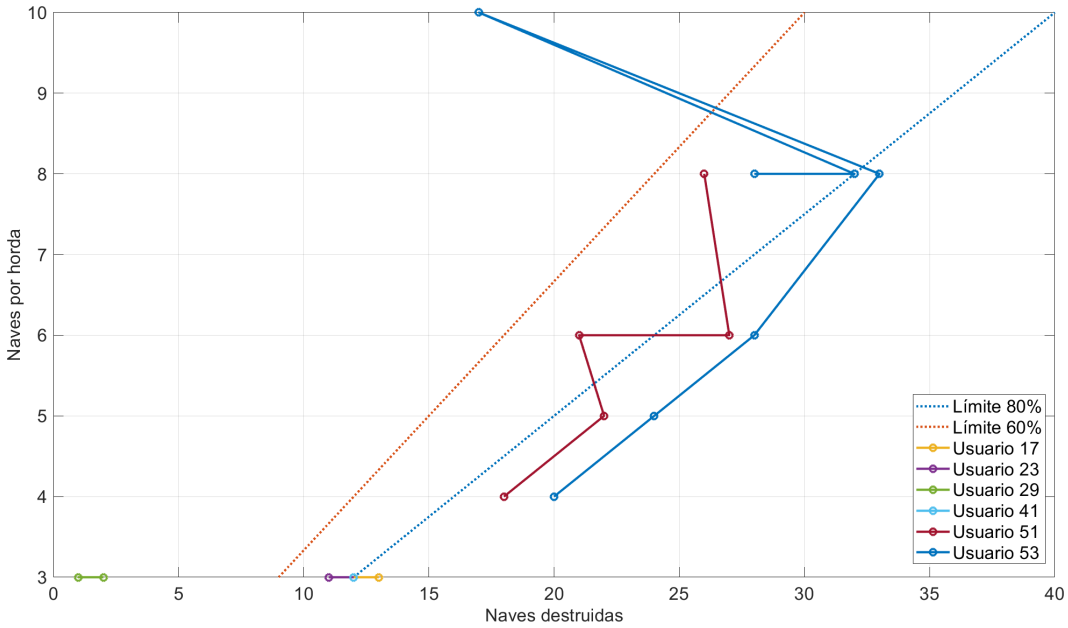


Figura 4.3: Canal de flujo - RF

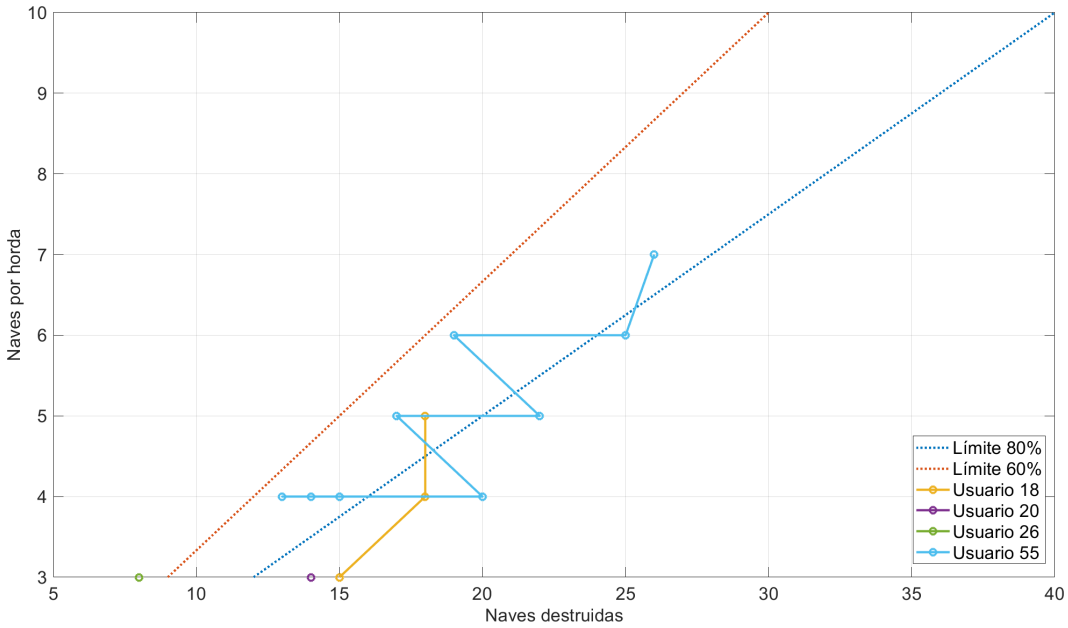


Figura 4.4: Canal de flujo - SVM

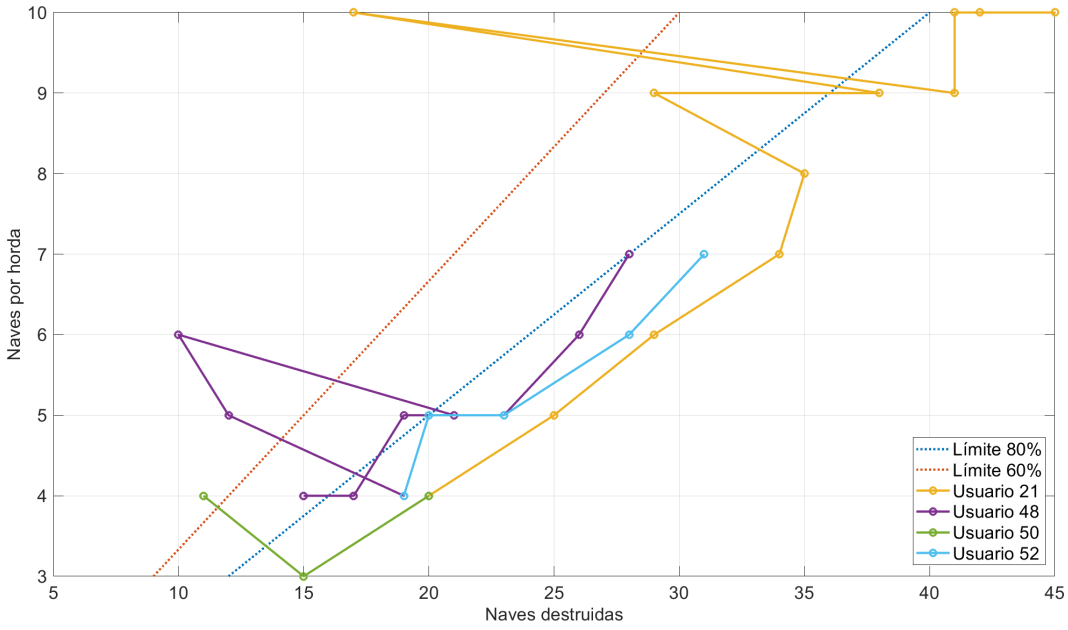


Figura 4.5: Canal de flujo - ANN

4.3. Usabilidad

Con el objetivo de evaluar la usabilidad del juego *Invasion Victory*, se ha llevado a cabo una encuesta, siguiendo el enfoque presentado por Abdellatif et al. [49]. Este enfoque se centra en diversas características de calidad utilizadas para la evaluación de juegos serios. Además, se han incorporado algunas afirmaciones relevantes de la escala estandarizada *Player Experience Inventory*, diseñada para medir la experiencia del jugador tanto a nivel funcional como psicosocial, en combinación con las afirmaciones propuestas en [50]. La usabilidad se define según la norma internacional ISO 9241-11 [51] como la capacidad de un producto para ser utilizado por usuarios específicos con eficacia, eficiencia y satisfacción en un contexto de uso particular.

El propósito de esta encuesta es estimar la usabilidad a través de resultados estadísticos inferenciales, los cuales se obtendrán al medir distintas características cualitativas, como la navegación de la aplicación, la jugabilidad, el diseño del juego, la comprensión y la opinión del usuario. Para ello, se han formulado una serie de afirmaciones pertinentes. A continuación, se exponen las respuestas brindadas por los usuarios para cada uno de los componentes de la encuesta. Es importante destacar que la participación en la encuesta mencionada fue completamente voluntaria, siendo completada únicamente por aquellos que optaron por participar. En total, se obtuvieron 17 encuestas. Dado que el objetivo principal de este estudio es verificar la influencia del algoritmo asignado a los usuarios, se impuso la restricción de eliminar las respuestas de aquellos usuarios que solo jugaron una partida o que participaron únicamente en la primera versión del juego. Siguiendo esta restricción, se realizó el análisis con siete encuestas que cumplían con dichos requisitos.

- **Afirmación 2.** El proceso de registro e inicio de sesión en la aplicación es sencillo. Ver Figura 4.6

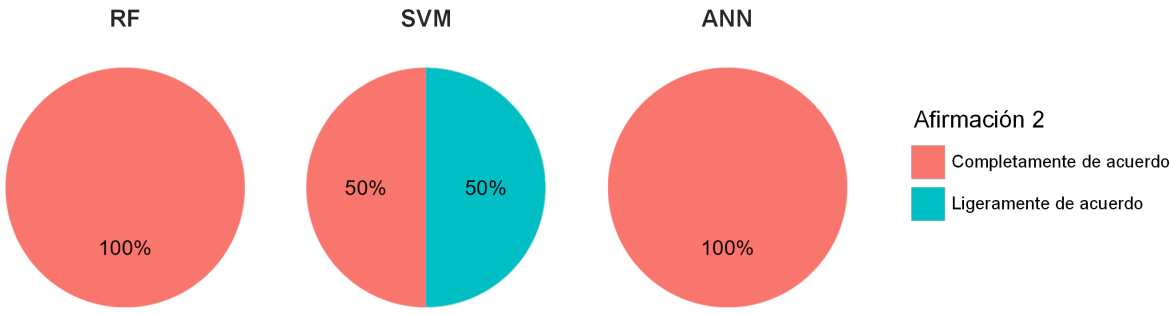


Figura 4.6: Afirmación 2 de usabilidad

- **Afirmación 3.** La ejecución de la aplicación en términos de visualización de la pantalla, alcance y manejo de botones, así como el acceso a diferentes funciones, es cómoda. Ver

Figura 4.7

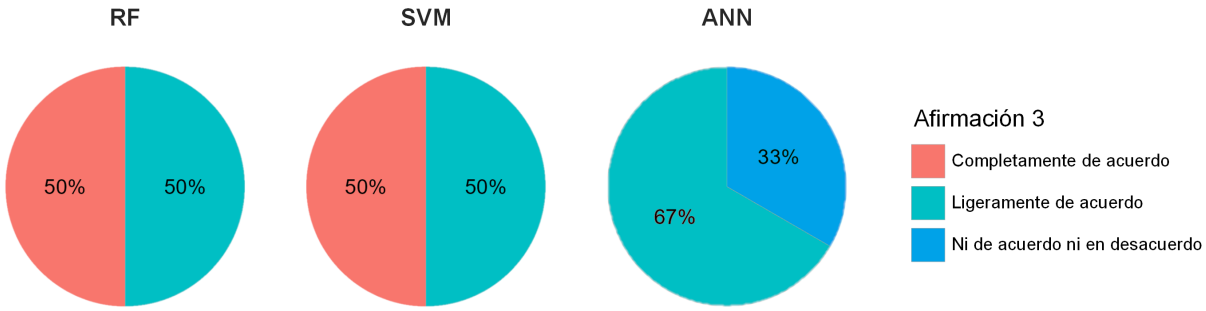


Figura 4.7: Afirmación 3 de usabilidad

- **Afirmación 4.** Fue fácil saber cómo realizar acciones en el juego. Ver Figura 4.8

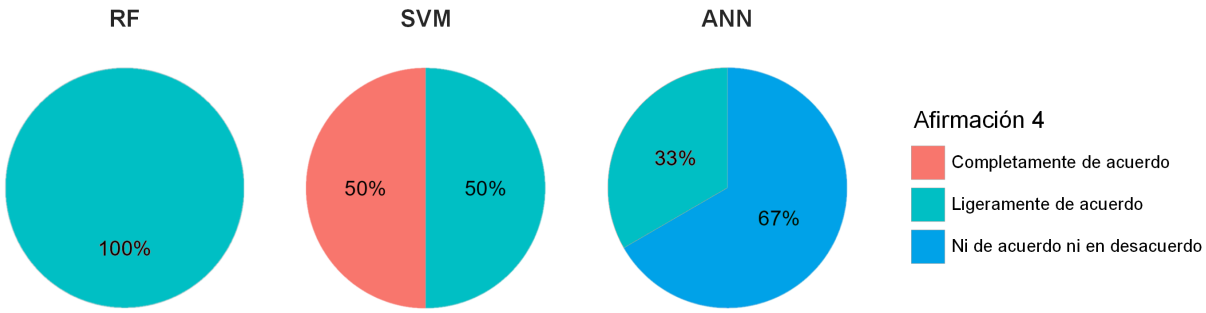


Figura 4.8: Afirmación 4 de usabilidad

- **Afirmación 5.** Me sentí libre de tomar decisiones dentro del juego. Ver Figura 4.9

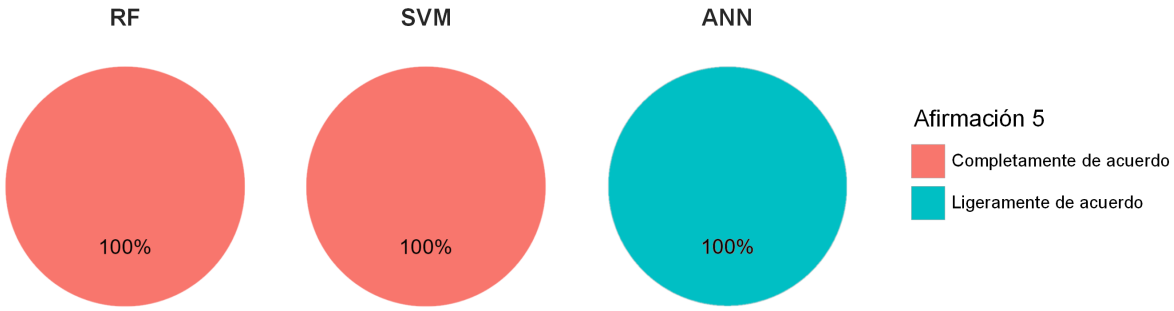


Figura 4.9: Afirmación 5 de usabilidad

- **Afirmación 6.** Disfruté mucho jugar el juego debido a su excelente jugabilidad. En donde la jugabilidad aborda todos los aspectos relacionados con la interacción del jugador con el juego, incluyendo los controles, la mecánica del juego, el diseño de niveles, las metas

y objetivos, la dificultad, la respuesta del juego a las acciones del jugador y la sensación general de diversión y satisfacción que se obtiene al jugarlo. Ver Figura 4.10

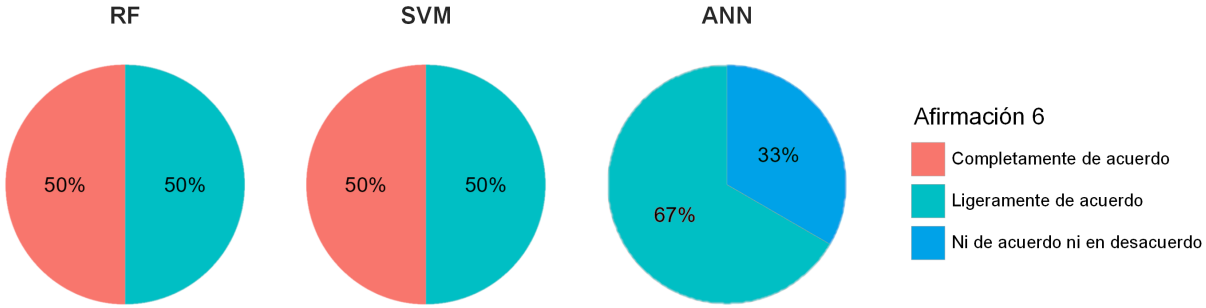


Figura 4.10: Afirmación 6 de usabilidad

- **Afirmación 7.** Me gustó la apariencia del menú de opciones del juego. Ver Figura 4.11

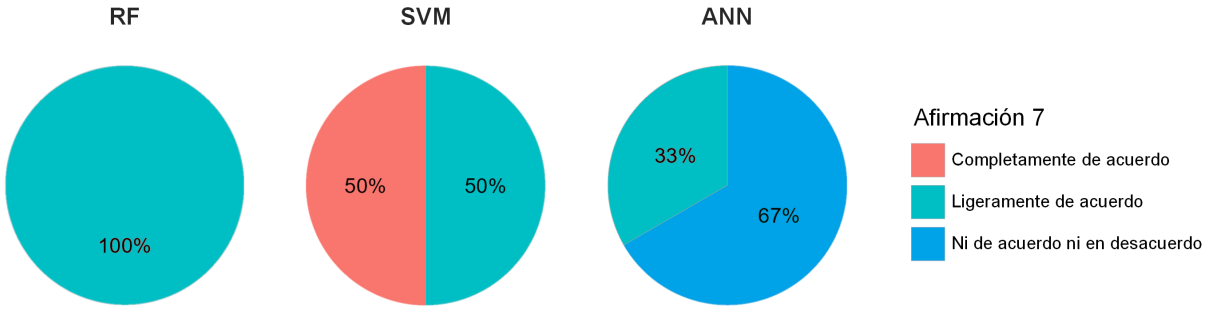


Figura 4.11: Afirmación 7 de usabilidad

- **Afirmación 8.** Disfruté la forma en que fue diseñada la pantalla principal de juego. Ver Figura 4.12

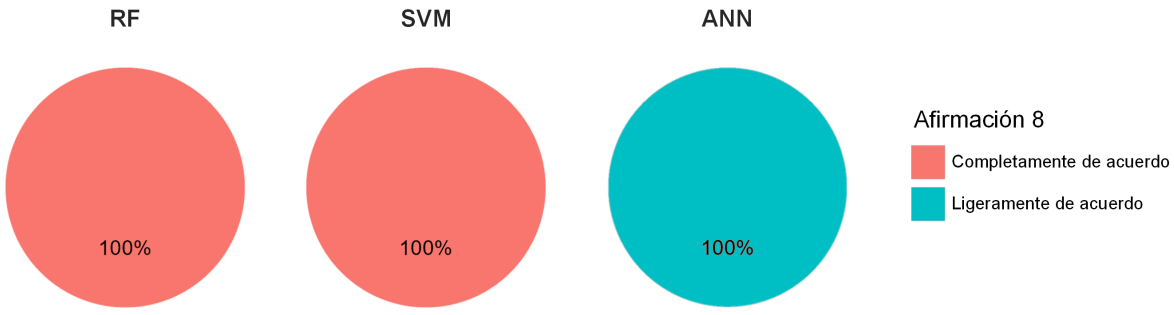


Figura 4.12: Afirmación 8 de usabilidad

- **Afirmación 9.** El juego me informó de mi progreso en el juego. Ver Figura 4.13

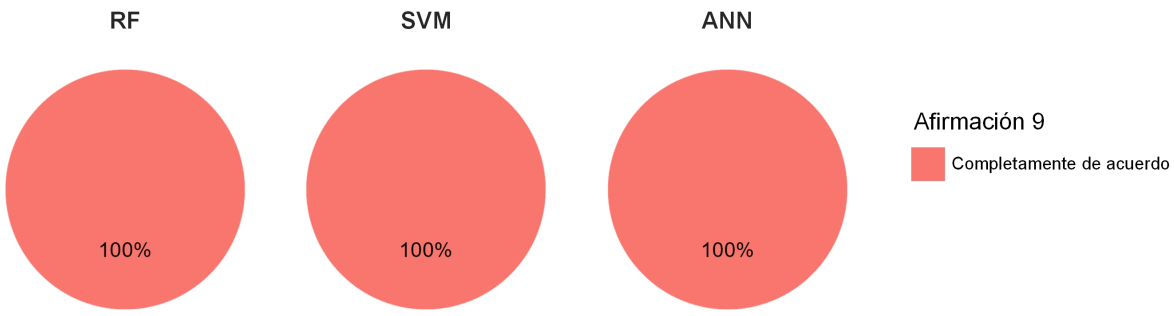


Figura 4.13: Afirmación 9 de usabilidad

- **Afirmación 10.** Los objetivos del juego fueron claros para mí. Ver Figura 4.14

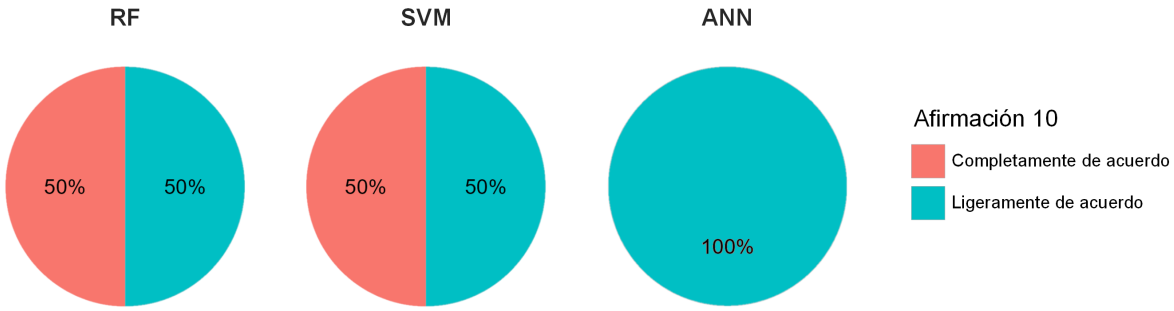


Figura 4.14: Afirmación 10 de usabilidad

- **Pregunta abierta.** Por favor, comparte tus opiniones y/o aportes sobre el juego en el espacio a continuación.

Capítulo 5

Discusión, Conclusiones y Trabajos futuros

5.1. Discusión

Los resultados muestran el grado de compromiso de los usuarios con la aplicación, medido por el número de partidas y la frecuencia de ingreso a la misma (ver Tabla 4.2). Se observa que los jugadores a quienes se les asignó el algoritmo *Artificial Neural Network* jugaron en promedio 8.8 partidas, mientras que los usuarios de *Random Forest* y *Support Vector Machine* jugaron menos de la mitad de ese valor, con un promedio de 3.5 partidas con cada algoritmo. Esta misma tendencia se refleja en el número de sesiones, donde aquellos usuarios que jugaron *Invasion Victory* con el algoritmo *Artificial Neural Network* ingresaron a la aplicación en promedio una mayor cantidad de ocasiones que los jugadores que usaron *Random Forest* y *Support Vector Machine*.

A partir de los resultados de desempeño presentados en la Figura 4.1, y mediante la prueba ANOVA seguida del análisis de la prueba de *Tukey*, se revela la existencia de una diferencia significativa en el desempeño entre los jugadores a quienes se les asignó el algoritmo *Artificial Neural Network*, y aquellos que utilizaron *Random Forest* y *Support Vector Machine*. Sin embargo, no se encontró una diferencia significativa entre los usuarios de *Random Forest* y *Support Vector Machine*.

Al evaluar el rendimiento del Ajuste Dinámico de Dificultad en el juego *Invasion Victory*, con el propósito de determinar su efectividad en mantener a los jugadores inmersos en el flujo de juego, es evidente desde el principio observar que algunos jugadores únicamente participaron en una o dos partidas. No obstante, esta cantidad resulta insuficiente para apreciar un progreso significativo en el juego en relación a los límites previamente establecidos.

En relación a cada algoritmo, es relevante destacar los siguientes hallazgos:

- En el caso específico del algoritmo *Random Forest*, como se muestra en la Figura 4.3, se puede observar que para aquellos usuarios que demuestran un progreso en el juego, los valores proporcionados por el algoritmo los mantienen dentro del canal de flujo, incluso cuando

superan nuevamente los umbrales. En estas situaciones, se lleva a cabo una redirección hacia el canal mediante el ajuste del número de naves por horda. Es relevante destacar que este algoritmo no contempla valores de ajuste para siete y nueve naves por horda, lo que da lugar a discontinuidades en los niveles de dificultad. A pesar de ello, los resultados muestran que, al menos para este grupo de usuarios, los valores proporcionados por el algoritmo resultan efectivos para mantenerlos en el canal de flujo.

- De igual forma, en la Figura 4.4, se ilustra cómo el algoritmo *Support Vector Machine* exhibe resultados que revelan que, para los jugadores que experimentaron una progresión en el juego, los valores ofrecidos por el algoritmo guían efectivamente al jugador hacia el canal de flujo. A pesar de que puedan surgir ocasiones en las que el jugador supere nuevamente el límite del 80 %, es redirigido de vuelta al canal.
- A su vez, en la Figura 4.5, se puede observar cómo el algoritmo *Artificial Neural Network* presenta un comportamiento peculiar en el que se evidencian variaciones en el número de naves destruidas en función del cambio en la cantidad de naves por horda. Sin embargo, a pesar de estas variaciones, no se logra llevar al usuario dentro del canal de flujo. Además, se observa que algunos jugadores logran destruir cantidades por debajo del 60 % y por encima del 80 % para un número igual de naves por horda.

El análisis de usabilidad revela que tanto los usuarios de *Random Forest* como los de *Support Vector Machine* muestran estar de acuerdo en diversos niveles con las afirmaciones presentadas en la encuesta. Por otro lado, los usuarios de *Artificial Neural Network* carecen de una opinión clara en algunos casos, ya que seleccionaron la opción ”ni de acuerdo ni en desacuerdo”, especialmente en relación con las afirmaciones tres, cuatro, seis y siete. La afirmación tres se enfoca en la experiencia de manejo de la aplicación, la cuatro y la cinco abordan la jugabilidad, y la número siete se centra en el diseño. Estos usuarios no poseen una opinión firme o definitiva acerca de estos enunciados.

En relación a la pregunta abierta, las respuestas proporcionan información valiosa sobre la experiencia de juego. Los jugadores, independientemente del algoritmo empleado, utilizaron adjetivos positivos para calificar el juego, como ”llamativo”, ”bien entretenido”, ”excelente”, ”bien logrado” y ”bueno”. No se encontraron respuestas de desaprobación, pero sí hubo recomendaciones en cuanto al diseño por parte de cuatro usuarios: dos de *Artificial Neural Network*, uno de *Random Forest* y otro de *Support Vector Machine*. Estas recomendaciones se centraron en el posicionamiento de algunos elementos y en la sugerencia de agregar elementos para hacer el juego más atractivo. Adicionalmente, se reportaron problemas de despliegue de la aplicación al utilizar la realidad aumentada por parte de un usuario de *Random Forest* y otro de *Artificial Neural Network*. Estos problemas podrían estar relacionados con el uso de la aplicación en lugares con poca iluminación o fondos monocromáticos, lo que limita el reconocimiento del fondo por parte de

la tecnología *ARCore*. Estos hallazgos resaltan la importancia de considerar las condiciones de entorno al utilizar la realidad aumentada en el juego.

5.2. Conclusiones

Los niveles de compromiso exhibidos por los usuarios de Invasion Victory muestran una tendencia a ser mayores en los jugadores asignados al algoritmo de Ajuste Dinámico de Dificultad basado en *Artificial Neural Network*, en comparación con los usuarios de *Random Forest* y *Support Vector Machine*. Por lo tanto, es posible establecer de manera concluyente, a partir de la información disponible, que el algoritmo empleado para el Ajuste Dinámico de Dificultad tiene un impacto significativo en el grado de compromiso de los usuarios en un juego en realidad aumentada de disparo en primera persona.

Al comparar los niveles de desempeño entre los usuarios de los tres tipos de algoritmo utilizados *Random Forest*, *Support Vector Machine* y *Artificial Neural Network* en relación con la cantidad de naves enemigas destruidas en el juego *Invasion Victory*, se constata que el tipo de algoritmo empleado para el Ajuste Dinámico de Dificultad influye de manera significativa en el nivel de desempeño de los usuarios en un juego en realidad aumentada de disparo en primera persona.

El análisis realizado con el objetivo de evaluar la eficacia del Ajuste Dinámico de Dificultad revela que, en el caso de los usuarios que evidenciaron progreso en el juego, se logró un seguimiento exitoso de los niveles de dificultad basado en el número de naves por horda proporcionado por los algoritmos. Específicamente, el algoritmo *Random Forest* demostró ser capaz de mantener a los jugadores dentro del canal de flujo de manera efectiva. Asimismo, a partir de los resultados obtenidos, se puede concluir que el algoritmo *Support Vector Machine* también logra mantener a los jugadores en este mismo canal de flujo. En contraste, el algoritmo *Artificial Neural Network* no logró guiar de manera efectiva a los jugadores dentro de los límites establecidos del canal de flujo.

La opinión sobre la usabilidad del juego *Invasion Victory*, a partir de los resultados obtenidos, no presenta una variación significativa. A pesar de que existen respuestas en las que los usuarios de *Artificial Neural Network* no están ni completamente de acuerdo ni en desacuerdo con algunos enunciados, estas indican que los usuarios de dicho algoritmo mantienen una posición moderada hacia el acuerdo. En consecuencia, se puede afirmar que el algoritmo utilizado para el Ajuste Dinámico de Dificultad no tiene un efecto significativo en la usabilidad en el contexto de un juego en realidad aumentada de disparo en primera persona.

5.3. Trabajos Futuros

- Desarrollar el juego *Invasion Victory* para dispositivos que utilicen el sistema operativo móvil *iOS*, con el objetivo de ampliar el número de dispositivos compatibles con la aplicación.
- Desarrollar un juego de realidad aumentada cuya temática se alinee con un análisis exhaustivo de las preferencias del mercado en el ámbito de los juegos destinados al entretenimiento.
- Desarrollar y comparar Agentes Artificiales (IA) emuladores del comportamiento de jugadores reales, tomando como referencia inicial los datos de usuarios recopilados en *Invasion Victory*.

Bibliografía

[1] J. Nakamura and M. Csikszentmihalyi, *The Concept of Flow*. Dordrecht: Springer Netherlands, 2014, pp. 239–263. [Online]. Available: https://doi.org/10.1007/978-94-017-9088-8_16

[2] V. Verma, S. D. Craig, R. Levy, A. Bansal, and A. Amresh, "Domain knowledge and adaptive serious games: Exploring the relationship of learner ability and affect adaptability," *Journal of Educational Computing Research*, vol. 60, pp. 406–432, 2021. [Online]. Available: https://doi.org/10.1177/07356331211031287

[3] S. Willwacher and O. Korn, "Gamification of movement exercises in rehabilitation and prevention: A framework for smart training in ai-based exergames," in *Advances in Industrial Design*, vol. 260. Cham: Springer International Publishing, 2021, pp. 855–862. [Online]. Available: https://doi.org/10.1007/978-3-030-80829-7_104

[4] P. Ramasamy, S. Das, and Y. Kurita, "Ski for squat: A squat exergame with pneumatic gel muscle-based dynamic difficulty adjustment," in *Universal Access in Human-Computer Interaction. Access to Media, Learning and Assistive Environments*, vol. 12769. Cham: Springer International Publishing, 2021, pp. 449–467. [Online]. Available: https://doi.org/10.1007/978-3-030-78095-1_33

[5] M. Zohaib, "Dynamic difficulty adjustment (dda) in computer games: A review," *Advances in Human-Computer Interaction*, vol. 2018, 2018. [Online]. Available: https://doi.org/10.1155/2018/5681652

[6] M. Ninaus, K. Tsarava, and K. Moeller, "A pilot study on the feasibility of dynamic difficulty adjustment in game-based learning using heart-rate," in *Games and Learning Alliance*, vol. 11899. Cham: Springer International Publishing, 2019, pp. 117–128. [Online]. Available: https://doi.org/10.1007/978-3-030-34350-7_12

[7] S. A. Kamkuimo K., B. Girard, and B.-A. J. Menelas, "Dynamic difficulty adjustment through real-time physiological feedback for a more adapted virtual reality exposure therapy," in *Games and Learning Alliance*, vol. 12517. Cham: Springer International Publishing, 2020, pp. 102–111. [Online]. Available: http://doi.org/10.1007/978-3-030-63464-3_10

[8] Andrew, A. N. Tjokrosetio, and A. Chowanda, "Dynamic difficulty adjustment with facial expression recognition for improving player satisfaction in a survival horror

game," *ICIC Express Letters*, vol. 14, pp. 1097–1104, 2020. [Online]. Available: https://doi.org/10.24507/icicel.14.11.1097

[9] P. M. Blom, S. Bakkes, and P. Spronck, "Modeling and adjusting in-game difficulty based on facial expression analysis," *Entertainment Computing*, vol. 31, 2019. [Online]. Available: https://doi.org/10.1016/j.entcom.2019.100307

[10] A. Schwarz, G. Cardon, S. Chastin, J. Stragier, L. De Marez, and A. DeSmet, "Does dynamic tailoring of a narrative-driven exergame result in higher user engagement among adolescents? results from a cluster-randomized controlled trial," *International Journal of Environmental Research and Public Health*, vol. 18, no. 14, p. 7444, 2021. [Online]. Available: http://doi.org/10.3390/ijerph18147444

[11] C. Peng, D. Wang, Y. Zhang, and J. Xiao, "A visuo-haptic attention training game with dynamic adjustment of difficulty," *IEEE Access*, vol. 7, pp. 68 878–68 891, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2918846

[12] M. Morosan and R. Poli, "Automated game balancing in ms pacman and starcraft using evolutionary algorithms," in *Applications of Evolutionary Computation*, vol. 10199. Cham: Springer International Publishing, 2017, pp. 377–392. [Online]. Available: https://doi.org/10.1007/978-3-319-55849-3_25

[13] T. Huber, S. Mertes, S. Rangelova, S. Flutura, and E. André, "Dynamic difficulty adjustment in virtual reality exergames through experience-driven procedural content generation," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE Computer Society, 2021. [Online]. Available: https://doi.org/10.1109/SSCI50451.2021.9660086

[14] L. Cardia da Cruz, C. A. Sierra-Franco, G. F. M. Silva-Calpa, and A. Barbosa Raposo, "A self-adaptive serious game for eye-hand coordination training," in *HCI in Games*, vol. 12211. Cham: Springer International Publishing, 2020, pp. 385–397. [Online]. Available: https://doi.org/10.1007/978-3-030-50164-8_28

[15] B. E. Garcia, M. K. Crocomo, and K. O. Andrade, "Dynamic difficulty adjustment in a whac-a-mole like game," in *2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, 2018, pp. 439–447. [Online]. Available: https://doi.org/10.1109/SBGAMES.2018.00020

[16] M. Weber and P. Notargiacomo, "Dynamic difficulty adjustment in digital games using genetic algorithms," in *2020 19th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE Computer Society, 2020, pp. 62–70. [Online]. Available: https://doi.org/10.1109/SBGames51465.2020.00019

[17] M. Shakhova and A. Zagarskikh, "Dynamic difficulty adjustment with a simplification ability using neuroevolution," *Procedia Computer Science*, vol. 156, pp. 395–403, 2019, 8th International Young Scientists Conference on Computational Science. [Online]. Available: https://doi.org/10.1016/j.procs.2019.08.219

[18] J. Suaza, E. Gamboa, and M. Trujillo, "A health point-based dynamic difficulty adjustment strategy for video games," in *Entertainment Computing and Serious Games*. Cham: Springer International Publishing, 2019, pp. 436–440. [Online]. Available: https://doi.org/10.1007/978-3-030-34644-7_42

[19] V. Araujo, D. Mendez, and A. Gonzalez, "A novel approach to working memory training based on robotics and ai," *Information*, vol. 10, no. 11, 2019. [Online]. Available: https://doi.org/10.3390/info10110350

[20] S. Demediuk, M. Tamassia, W. L. Raffe, F. Zambetta, X. Li, and F. Mueller, "Monte carlo tree search based algorithms for dynamic difficulty adjustment," in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, 2017, pp. 53–59. [Online]. Available: https://doi.org/10.1109/CIG.2017.8080415

[21] S. Demediuk, M. Tamassia, X. Li, and W. L. Raffe, "Challenging ai: Evaluating the effect of mcts-driven dynamic difficulty adjustment on player enjoyment," in *Proceedings of the Australasian Computer Science Week Multiconference*, ser. ACSW 2019. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3290688.3290748

[22] M. González-Duque, R. B. Palm, D. Ha, and S. Risi, "Finding game levels with the right difficulty in a few trials through intelligent trial-and-error," in *2020 IEEE Conference on Games (CoG)*. IEEE Computer Society, 2020, pp. 503–510. [Online]. Available: https://doi.org/10.1109/CoG47356.2020.9231548

[23] M. Rajabi, M. Ashtiani, B. Minaei-Bidgoli, and O. Davoodi, "A dynamic balanced level generator for video games based on deep convolutional generative adversarial networks," *Scientia Iranica*, vol. 28, no. 3, pp. 1497–1514, 2021. [Online]. Available: https://doi.org/10.24200/sci.2020.54747.3897

[24] H.-S. Moon and J. Seo, "Dynamic difficulty adjustment via fast user adaptation," in *Adjunct Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '20 Adjunct. New York, NY, USA: Association for Computing Machinery, 2020, p. 13–15. [Online]. Available: https://doi.org/10.1145/3379350.3418578

[25] D. B. Or, M. Kolomenkin, and G. Shabat, "Dl-dda - deep learning based dynamic difficulty adjustment with ux and gameplay constraints," in *2021 IEEE Conference on Games (CoG)*.

IEEE Computer Society, 2021. [Online]. Available: https://doi.org/10.1109/CoG52621.2021.9619162

[26] W. Rao Fernandes and G. Levieux, "δ-logit : Dynamic difficulty adjustment using few data points," in *Entertainment Computing and Serious Games*. Cham: Springer International Publishing, 2019, pp. 158–171. [Online]. Available: https://doi.org/10.1007/978-3-030-34644-7_13

[27] M. A. Gutiérrez, J. J. Rosero, D. E. Guzmán, and C. F. Rengifo, "A music therapy serious game with dynamic difficulty adjustment for stimulating short-term memory," in *Converging Clinical and Engineering Research on Neurorehabilitation IV*, vol. 28. Cham: Springer International Publishing, 2021, pp. 723–734. [Online]. Available: https://doi.org/10.1007/978-3-030-70316-5_116

[28] D. Atorf, S. Dyck, and J. Steinbach, "Towards a concept for a hidden object game with dynamic difficulty adjustment," *18th International Conference on Cognition and Exploratory Learning in Digital Age, CELDA 2021*, p. 311–314, 2021.

[29] R. Flemming, E. Schmück, D. Mussack, P. Cardoso-Leite, and P. Schrater, "A generalizable performance evaluation model of driving games via risk-weighted trajectories," in *EDM 2019 - Proceedings of the 12th International Conference on Educational Data Mining*. International Educational Data Mining Society, 2019, Conference paper, p. 548–551.

[30] Z. Amiri, Y. A. Sekhavat, and S. Goljaryan, "Stepar: A personalized exergame for people with multiple sclerosis based on video-mapping," *Entertainment Computing*, vol. 42, p. 100487, 2022. [Online]. Available: https://doi.org/10.1016/j.entcom.2022.100487

[31] A. Darzi, S. M. McCrea, and D. Novak, "User experience with dynamic difficulty adjustment methods for an affective exergame: Comparative laboratory-based study," *JMIR Serious Games*, vol. 9, no. 2, 2021. [Online]. Available: https://doi.org/10.2196/25771

[32] J. Pfau, J. D. Smeddinck, and R. Malaka, "Enemy within: Long-term motivation effects of deep player behavior models for dynamic difficulty adjustment," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ser. CHI '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3313831.3376423

[33] A. Menzinsky, G. López, J. Palacio, M. A. Sobrino, R. Álvarez, and V. Rivas, *Historias de usuario*. Scrum Manager, 2022, no. 3.01, ingeniería de requisitos Ágil.

[34] A. Ambika, H. Shin, and V. Jain, "Immersive technologies and consumer behavior: A systematic review of two decades of research," *Australian Journal of Management*, vol. 0, no. 0, 2023. [Online]. Available: https://doi.org/10.1177/03128962231181429

[35] B. Braun, J. M. Stopfer, K. W. Müller, M. E. Beutel, and B. Egloff, "Personality and video gaming: Comparing regular gamers, non-gamers, and gaming addicts and differentiating between game genres," *Computers in Human Behavior*, vol. 55, pp. 406–412, 2016. [Online]. Available: https://doi.org/10.1016/j.chb.2015.09.041

[36] "Brainstorming," in *Encyclopedia of Creativity, Invention, Innovation and Entrepreneurship*, E. G. Carayannis, Ed. Cham: Springer International Publishing, 2020, pp. 204–204. [Online]. Available: http://doi.org/10.1007/978-3-319-15347-6_300104

[37] J. R. López-Arcos, N. Padilla-Zea, P. Paderewski, F. L. Gutiérrez, and A. Abad-Arranz, "Designing stories for educational video games: A player-centered approach," in *Proceedings of the 2014 Workshop on Interaction Design in Educational Environments*, ser. IDEE '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 33–40. [Online]. Available: https://doi.org/10.1145/2643604.2643611

[38] A. Grami, "Chapter 20 - finite-state machines," in *Discrete Mathematics*. Academic Press, 2022, pp. 373–388. [Online]. Available: https://doi.org/10.1016/B978-0-12-820656-0.00020-4

[39] C. Alvin, B. Peterson, and S. Mukhopadhyay, "Static generation of uml sequence diagrams," *International Journal on Software Tools for Technology Transfer*, vol. 23, no. 1, pp. 31–53, 2021. [Online]. Available: https://doi.org/10.1007/s10009-019-00545-z

[40] H. Cardona, J. E. Masso, M. F. Mera, S. M. Roa, E. F. Ruano, M. D. Torres, and M. I. Vidal, *Diseno e Implementación de Bases de Datos desde una Perspectiva Práctica*. LATIn Project, 2014. [Online]. Available: https://hdl.handle.net/20.500.11785/586

[41] Unity Technologies. Unity real-time development platform | 3d, 2d vr & ar engine. [Online]. Available: https://unity.com

[42] H. Kou, P. M. Johnson, and H. Erdogmus, "Operational definition and automated inference of test-driven development with zorro," *Automated Software Engineering 2009 17:1*, vol. 17, pp. 57–85, 2009. [Online]. Available: https://doi.org/10.1007/s10515-009-0058-8

[43] I. B. Kerthyayana, "Combination of test-driven development and behavior-driven development for improving backend testing performance," *Procedia Computer Science*, vol. 157, pp. 79–86, 2019. [Online]. Available: https://doi.org/10.1016/j.procs.2019.08.144

[44] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[45] A. Mizumoto, “Calculating the relative importance of multiple regression predictor variables using dominance analysis and random forests,” *Language Learning*, vol. 73, no. 1, pp. 161–196, 2023. [Online]. Available: https://doi.org/10.1111/lang.12518

[46] G. Rebala, A. Ravi, and S. Churiwala, *Random Forests*. Cham: Springer International Publishing, 2019, pp. 77–94. [Online]. Available: https://doi.org/10.1007/978-3-030-15729-6_7

[47] E. Bisong, *Support Vector Machines*. Berkeley, CA: Apress, 2019, pp. 255–268. [Online]. Available: https://doi.org/10.1007/978-1-4842-4470-8_22

[48] N. K. Manaswi, *Multilayer Perceptron*. Berkeley, CA: Apress, 2018, pp. 45–56. [Online]. Available: https://doi.org/10.1007/978-1-4842-3516-4_3

[49] A. J. Abdellatif, B. McCollum, and P. McMullan, “Serious games: Quality characteristics evaluation framework and case study,” in *2018 IEEE Integrated STEM Education Conference (ISEC)*, 2018, pp. 112–119. [Online]. Available: https://doi.org/10.1109/ISECon.2018.8340460

[50] D. Mariaca, D. Guzman, and J. Londono, “A serious game for cognitive and fine motor skill rehabilitation: a qualitative evaluation,” in *2022 IEEE 40th Central America and Panama Convention (CONCAPAN)*, 2022, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CONCAPAN48024.2022.9997622

[51] “Iso/iec 9126: 1998 ergonomic requirements for office work with visual display terminals-part 11: Guidance on usability,” 2018.

Anexos

Anexo A

Documentación Software

Ingresa el siguiente enlace de *GitHub* en tu navegador:
`https://github.com/carlosfeliperengifo/invasion-victory.git`

Se abrirá el siguiente repositorio de *GitHub*, el cual contiene todos los archivos usados en el proyecto *Invasion Victory*:

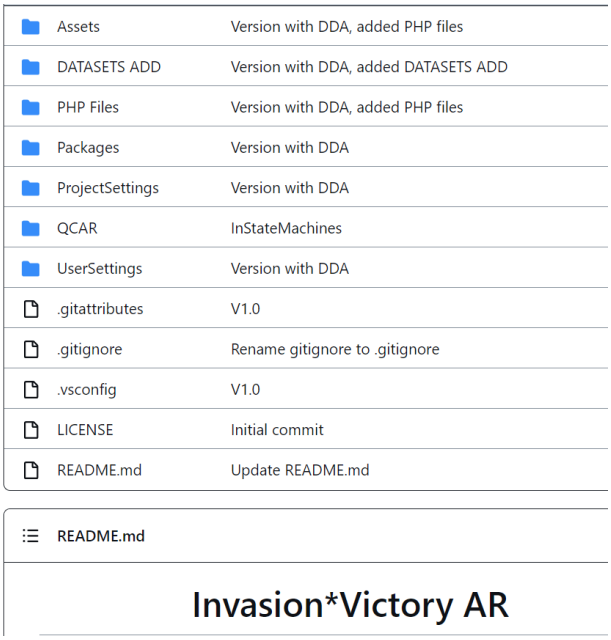


Figura A.1: Repositorio del proyecto *Invasion Victory*

Al hacer clic en la carpeta *Assets*, se mostrarán los recursos empleados en el proyecto de *Unity*, tales como elementos tridimensionales, escenas, *scripts*, música, audios, videos, tablas de algoritmos y otros componentes utilizados para el desarrollo del juego.

Autarca	Version with DDA
Editor	Version with DDA
EffectExamples	Version with DDA
Free Game Music Collection	Version with DDA
Free Pack	Version with DDA
FreeMachineGun	Version with DDA
Icons	Version with DDA
Low Poly Guns	Version with DDA
MAC_SciFiSniperRifle	Version with DDA
Materials	InStateMachines
Plugins	Version with DDA
Prefabs	Version with DDA
Resources	Version with DDA
Scenes	Version with DDA
Scripts	Version with DDA, added PHP files
Weapons of Choice FREE - Komposite Sound	Version with DDA
XR	Version with DDA
m31_teleporter_FX	V1.0

Figura A.2: Recursos usados en el proyecto *Invasion Victory*

Dentro de la opción *DATASETS ADD*, descubrirás dos archivos. El archivo con el nombre *InvasionAR1_V4* contiene los datos utilizados en el entrenamiento de los algoritmos implementados en el Ajuste Dinámico de Dificultad (ADD), los cuales fueron recopilados en la primera versión del juego. Además, encontrarás el archivo *InvasionAR2_V2*, que contiene los datos empleados para analizar el impacto del algoritmo de ajuste dinámico de dificultad en el compromiso y rendimiento de los usuarios de *Invasion Victory*. Estos datos fueron recolectados a partir de la segunda versión del juego.

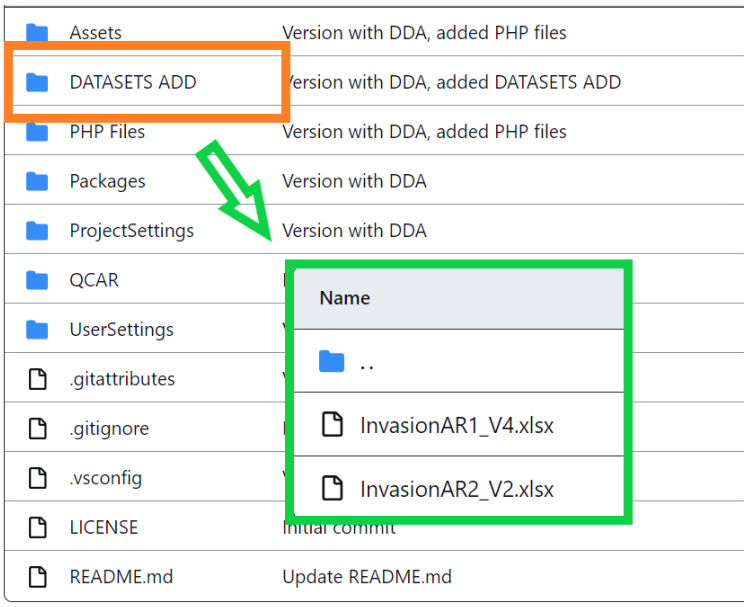


Figura A.3: Carpeta del Conjunto de datos del proyecto

Dentro de la carpeta *PHP Files*, se localizan las carpetas *invasion-victory1_PHP* y *invasion-victory2_PHP*, las cuales albergan los archivos *.php* implementados para establecer la conexión entre la primera y segunda versión del juego con la Base de Datos en la nube. Asimismo, se encuentra la carpeta *invasion-victory_docs*, que contiene la política de privacidad del juego y el reglamento del torneo organizado para la primera recolección de datos.

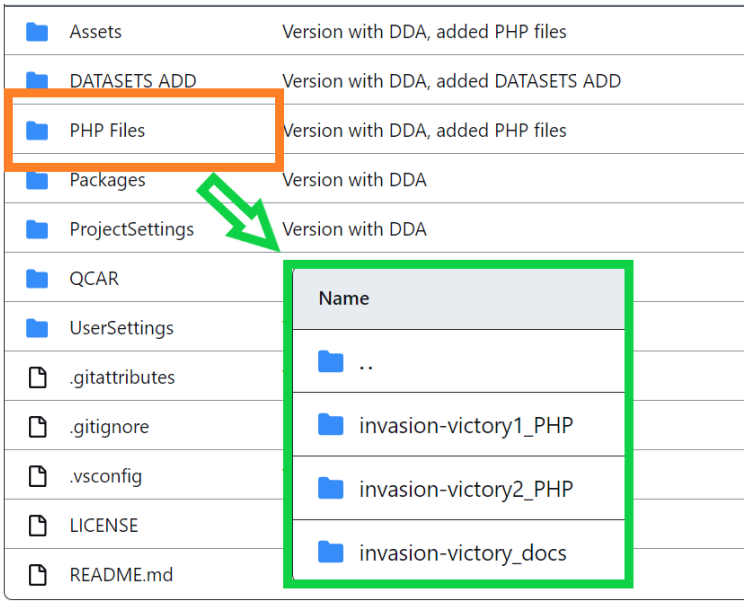


Figura A.4: Carpeta de archivos PHP del proyecto

Anexo B

Manual de usuario de *Invasion Victory*

El siguiente manual presenta las funciones de los diversos botones y la información de las interfaces de usuario:

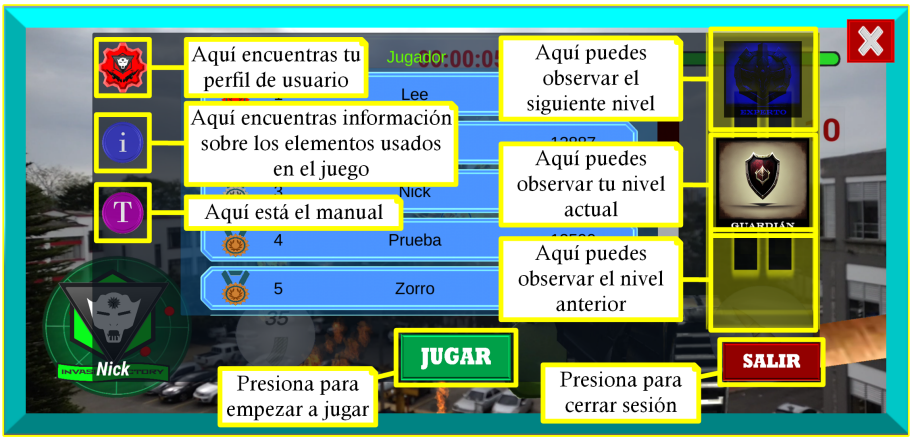


Figura B.1: Manual - pantalla de opciones del usuario

Figura B.2: Manual - pantalla de clasificación

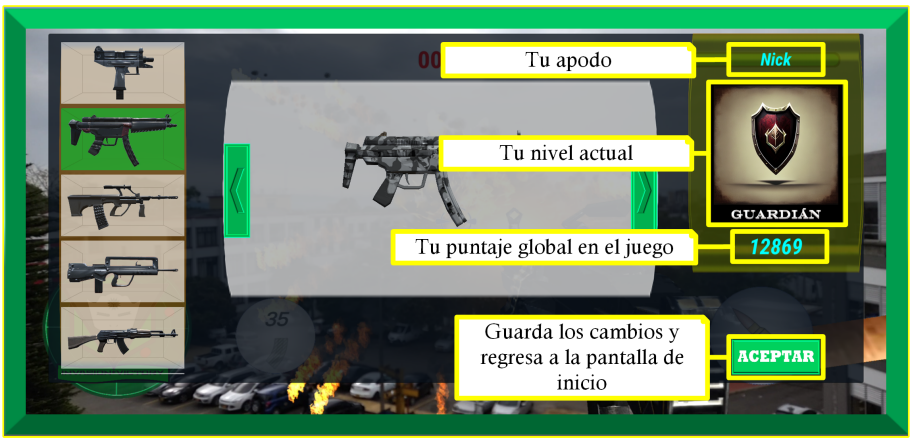


Figura B.3: Manual - pantalla de parámetros del juego

Figura B.4: Manual - pantalla de nivel del jugador

Figura B.5: Manual - pantalla de movimiento del jugador

Figura B.6: Manual - pantalla del juego

Figura B.7: Manual - pantalla de control del juego

Figura B.8: Manual - pantalla de pausa del juego

Figura B.9: Manual - pantalla resumen de la partida

Anexo C

Artículo científico IEEE

El siguiente artículo científico en inglés ha sido aprobado para ser presentado en la conferencia en Popayán, Colombia, del 17 al 20 de octubre de 2023.

IEEE CCAC 2023 notification for paper 3

Externo Recibidos ×

IEEE CCAC 2023 <ieeeccac2023@easychair.org> 14 ago 2023, 23:53

para mí

Dear Gilber Andres Villaquiran,

It is our pleasure to announce you that the submission # 3 entitled "A statistical approach to select the manipulated variables of a difficulty adjustment closed-loop control for an augmented reality video game" has been approved to be presented at the conference in Popayan, Colombia, on October 17 to 20, 2023.

Please address all the comments done by the reviewers in your camera-ready version.

In a few days, we will send you directions for your camera-ready version, and also the registration information to participate in the conference.

Congratulations! we expect to see you in Popayan!

Best Regards,

IEEE CCAC 2023 - Organizing Committee

Figura C.1: Notificación de artículo aprobado por IEEE

A statistical approach to select the manipulated variables of a difficulty adjustment closed-loop control for an augmented reality video game

1st Ricardo Campo
Electronic, Instrumentation and Control
University of Cauca
Popayán, Colombia
ricampo@unicauca.edu.co

2nd Gilber Andres Villaquiran
Electronic, Instrumentation and Control
University of Cauca
Popayán, Colombia
gilvilla1@unicauca.edu.co

3rd Carlos Felipe Rengifo
Electronic, Instrumentation and Control
University of Cauca
Popayán, Colombia
caferen@unicauca.edu.co

4th Diego Enrique Guzmán
Electronic, Instrumentation and Control
University of Cauca
Popayán, Colombia
diegoguzman@unicauca.edu.co

***Abstract*—Dynamic difficulty adjustment (DDA) has become a popular method for enhancing the gaming experience in both serious and recreational video games. However, most DDAs are designed using heuristic rules, which can lead to suboptimal selection of the variables manipulated to affect the game's challenge. To address this issue, we propose to perform a non linear open-loop identification of the interaction between the user and the game, in which the user parameterizes the difficulty manually, and the resulting data, along with the performance data from multiple game sessions, are used to train a machine learning model. Then, the predictive power statistical analysis is used to remove manipulated variables with reduced or null effect on players performance. In this paper, we report the experimental results obtained using a first-person shooter game (*Invasion Victory*) that we developed. Our findings demonstrate that this approach can successfully identify meaningful correlations between manipulated and controlled variables. The game is available for download from the *Google Play Store*. The proposed approach represents a promising direction for DDA design that incorporates principles of automatic control and statistical learning.**

***Index Terms*—dynamic difficulty adjustment, open-loop identification, coefficient of determination, gaming experience, first-person shooter game.**

I. Introduction

Dynamic difficulty adjustment (DDA) is a technique used to adapt the evolution of games to the abilities of individual players, with the aim of preventing frustration due to excessive difficulty or boredom from a lack of challenge [1]. When implemented effectively, DDA can greatly enhance the player's experience by keeping them within the "flow channel" identified by Nakamura and Csikszentmihalyi [2]. This channel refers to the optimal state in which a player is fully immersed in the game and experiencing a balance between the level of challenge and their capabilities. By continuously adjusting the game's difficulty to match the player's skill level, DDA can help maintain this state of flow and improve the overall gaming experience.

979-8-3503-2472-3/23/$31.00 ©2023 IEEE

DDA systems are closed-loop control systems, in which changes in the manipulated variables lead to corresponding responses in the controlled variables. The controller then recalculates the manipulated variables for the next scene or within the same scene [3]. While some DDA, such as *Invasion Victory*, only use performance variables, others also incorporate physiological variables to infer emotional states such as concentration, frustration, or fatigue. The manipulated variables correspond to the elements that can affect the game's difficulty, such as the speed of moving objects, the number of elements to memorize or rearrange, or the number of elements to capture.

DDA has been used in a variety of games, including those designed for entertainment [4], education [5], rehabilitation [6], and promoting healthy lifestyles through physical activity [7]. According to Zohaib et al [3], a range of principles has been employed in these applications, including probabilistic algorithms [8], single and multilayer perceptrons [9], dynamic scripting [10], village systems [11], reinforced learning [12], artificial neural networks trained with data generated by the upper confidence bound for trees method [13], [14] and self-organizing systems with artificial neural networks [15]. As previously indicated, these algorithms modify the manipulated variables of the game to adjust its difficulty level, with the aim of keeping players within the flow channel and enhancing their experience.

II. Related Work

The game proposed by Ninaus et al [16] consists of three scenarios: a traffic accident, a building fire, and a train crash. The goal is to manage teams of paramedics, firefighters, and

ambulances to respond to each emergency effectively. In this game, DDA uses the player's heart rate as a measure of their level of arousal to determine the appropriate difficulty level. This approach enables the game to adjust to the player's emotional state in real-time, providing a more personalized and engaging experience. Nugraha and Chowanda [17] present a survival game called *Five Nights at Freddy's*, in which the player must avoid being scared by ghosts for five minutes. In this game, the DDA system employs the recognition of the player's facial expressions through Affectiva's AFFDEX SDK system to increase or decrease the difficulty. Schwarz and colleagues [18] introduce a mobile exergame named *SmartLife*, wherein the players are required to engage in physical activities to restore the power source that is necessary for their survival in a post-apocalyptic world. In this game, the DDA measures the lower body movements of the player using an accelerometer embedded in their attire and transmits this data to the player's smartphone via Bluetooth. Peng et al, [19] present a game for visuo-haptic attention training, called *FF-Dancing*, which consists of moving colored discs from an initial position to reach a set height. In this game, the DDA measures the strength of the tips of the index and middle fingers of both hands.

Regarding the performance variables used for DDA implementation, scoring is often used as the main indicator. An example of this is presented by Cardia et al [20], who propose a 2D game whose objective is to pop balloons that ascend from the bottom of the screen and then disappear at the top of the screen. In this game, the performance variable is the number of balloons popped. The health status of the avatar representing the player is also commonly used as a performance variable. Such is the case of the game *The Forest of the Guardians* [21], a mobile phone fighting game with three predefined difficulty levels, in which the health status of the avatar at the end of each combat defines the difficulty level of the next encounter. The response time of the player to in-game events can be a valuable performance variable. For instance, Gutierrez et al [22] developed *Music Therapy Brain Training*, a serious game designed to improve short-term memory through one-minute videos sourced from YouTube. After watching the videos, the game prompts players with questions to test their memory skills. Performance in this game is measured by both the player's response time and the number of correct answers provided.

Manipulated variables in games refer to those elements that affect the difficulty level. For instance, in the game proposed by Morosan and Poli [23], in which the player must guide the character PacMan through a 2D maze and obtain the most points by consuming fruit-shaped elements, while avoiding a group of ghosts, who if they come into contact with PacMan decrease their number of lives. These ghosts can be eliminated if PacMan consumes an energy pill and collides with them. In this game, the DDA determines the speed with which the player moves, and the speed of the ghosts. Similarly, the number and frequency of appearance of adversaries are used as manipulated variables by Blom et al [24] in the game *Infinite Mario Bros*, which is an open source version of *Super Mario Bros*, where the DDA varies the difficulty through the number of adversaries and the number of obstacles to evade. In the physical activity game proposed by Huber et al [25], the player must go through a maze made up of several exercise rooms, and the DDA adapts the structure of the rooms to increase or decrease the required level of physical activity.

Despite the increasing use of DDA, it is rare to find an analysis that verifies the dependence relationship between the controlled variables and the manipulated variables. This research presents a variable importance analysis based on Random Forest (RF) for an augmented reality (AR) game called *Invasion Victory*. Nineteen volunteers from the *Semillero AR/VR en la Industria 4.0* of the *Universidad del Cauca* participated in the data collection.

III. Methods

A. Game development

Invasion Victory is a first-person shooter game that tells the story of an alien invasion. The players are surrounded by enemy spaceships that aim to destroy them. The objective is to earn as many points as possible and reach the top of the game's leaderboard (Figure 1). The players must destroy the maximum number of spaceships within three minutes while avoiding missile hits and collisions with the spaceships to protect their life points. The game generates five hordes of enemy spaceships, which the player must destroy with five machine gun hits. A radar displays the position of the spaceships, and the player can activate their weaponry to eliminate them. The player starts with a total of 100 life points, which decrease by two points for each missile hit and ten points for each collision with a spaceship. The life points increase by one point every four seconds of gaming. The game ends in one of the following scenarios: (1) all the spaceships of the five hordes are destroyed, (2) the player loses all their life points, or (3) the maximum game time of 3 minutes is reached.

Invasion Victory was developed using *Unity Game Engine* [26], which employs 3D dynamic components for the generation of interactive virtual content. The game incorporates the augmented reality application development platform *Vuforia Engine* and the *ARCore* library of *Unity Game Engine* to allow the user to interact with three-dimensional objects within the game area. The game was programmed in the C# language coupled with *Unity Game Engine* and the *Visual Studio Integrated Development Environment*. The game theme and its dynamics were designed using the Brainstorming methodology [27] and the agile Scrum methodology [28] was used for software development. *Invasion Victory* sends the data generated during the game to a relational database hosted on Hostinger's servers. The game also accesses this database to display, for example, the players with the highest scores. This bidirectional flow of information was implemented using the PHP programming language and the MySQL database handler. To ensure proper game functionality, the application requires a smartphone running *Android 8.0* or higher and supporting *ARCore* technology.

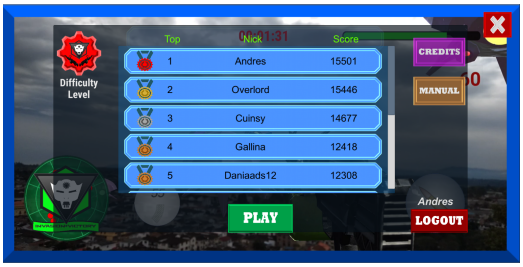


Fig. 1. Player ranking interface

Fig. 3. Parameter change interface

B. Game playing

Invasion Victory can be used seated or standing; in the latter case it is suggested to have an area of two meters long by two meters wide and free of obstacles. During the game, the users visualizes their environment through the cell phone screen, and the spaceships to be destroyed are superimposed on the flow of images, which appear at any point of the 360-degree periphery (Figure 2). In this augmented reality system, the player perceives both the ground and the surrounding objects at all times, even so, when the game is used while standing up, it involves moderate physical exercise.

Fig. 2. Spaceships appearing as augmented reality objects.

C. Open loop identification

In the open-loop identification phase of the user-game interaction dynamics, the variables manipulated are (i) the number of enemy spaceships per horde, (ii) the time of appearance between hordes, and (iii) the speed at which enemy spaceships move. To configure these variables the user has two options. In the first one, the player enters by default to the *Home* screen (Figure 1) and then enters the game screen automatically activating the random difficulty mode. In the second configuration option, the player enters the *Home* screen and then enters the *Parameters* screen (Figure 3) where the player can switch from the random difficulty mode to the manual difficulty setting mode.

The three variables that define the difficulty of the game (number of spaceships per horde, time of appearance between hordes, and speed of the spaceships) are adjusted by the player through the three sliders in Figure 3. The minimum and maximum limits of these variables, which is equivalent to the sizing of the actuator in an industrial control loop, were defined to ensure that the controlled variables could vary between 0 and 100%. Once these ranges were defined, it became necessary to discretize the amplitudes of the manipulated variables to facilitate, in later stages, the training of the statistical learning algorithm that predicts the quantile in which the player will be located. To obtain these values, each variable was progressively increased while the other two remained static. From the experimental results, it was observed that, for example, changes in the appearance time between consecutive hordes of less than two seconds did not generate significant differences in performance. Table I summarizes the values obtained.

TABLE I
RANGES AND QUANTIFICATION STEPS FOR MANIPULATED VARIABLES

Variable	Lower limit	Upper limit	Step
Spaceships per horde	3	10	1
Time between hordes	6 s	30 s	2 s
Speed of spaceships	0.4 m/s	1.2 m/s	0.1 m/s

D. Data collection

The game underwent three rounds of testing. In the first test, 12 members of the *Semillero AR/VR en la Industria 4.0* used the application without limitations in the number of sessions or in the type of difficulty. However, there were issues with participant registration and intermittent errors in the game's operation. When these problems were addressed, a second test was conducted with ten students from the same group, which revealed that the most of the initial issues were resolved. A third pilot was conducted, in which no flaws were detected. As expected, the test showed that *Invasion Victory* runs only on mobile phones compatible with the *ARCore* library.

After completing the three trials, erroneous information from matches with intermittent issues and from users with devices not compatible with *ARCore* was eliminated.

IV. RESULTS

To analyze the effect of the three manipulated variables on the total number of destroyed spaceships, the 111 samples obtained after the tests were processed in R Language to train

a RF for regression. The training procedure comprised the following steps:

1) The data set (111 samples) was randomly divided into training and test sets. The former comprised 70% of the samples, and the remaining 30% constituted the latter.
2) A RF was trained to predict the total number of spaceships destroyed, which was an integer number between 1 and 50, with median and interquartile range values equal to 13 and 14, respectively.
3) The measured (y_i) and predicted values ($\hat{y}_i$) for the total number of destroyed spaceships were used to calculate the coefficient of determination, which is a measure of the agreement between y_i and $\hat{y}_i$ and ranges between 0 and 1:

$$R^2 = \max\left(0, 1 - \frac{\sum_{i=1}^{N_T}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N_T}(y_i - \bar{y})^2}\right) \quad (1)$$

With N_T being the number of samples of the test set, and $\bar{y}$ the mean of that set. The function max is used to ensure that the minimum value of R^2 is 0.
4) The predictive power of each manipulated variable was calculated using the `varImp` function of the R Language *caret* package. This method is commonly referred to as variable importance assessment in the machine learning community [29], [30].

To avoid bias in the results, we repeated the previous steps 1000 times since both the coefficient of determination and the predictors' importance depend on the selection of the training and test datasets. Figures 4 and 5 show the variability of the coefficient of determination and the variable importance for the 1000 experiments. Figure 5 demonstrates that the speed of the spaceships had the highest predictive power, remaining close to 100% in all 1000 trials. In contrast, the number of spaceships per horde, whose median value was zero, had the lowest predictive power, with some outliers reaching values as high as 50%. Additionally, the time between hordes showed the greatest variability across all experiments. Table II summarizes the importance of each manipulated variable.

TABLE II
STATISTICS OF THE PREDICTOR'S IMPORTANCE ACROSS 1000 TRAINING AND TEST TRIALS. *NShips* IS THE NUMBER OF SPACESHIPS PER HORDE, *Speed* IS THE SPEED OF SPACESHIPS, AND *Time* IS THE TIME BETWEEN HORDES

Variable	Min (%)	Max (%)	Median (%)	IQR (%)
NShips	0	49.54	0	0
Speed	87.36	100	100	0
Time	0	100	16.14	25.56

V. DISCUSSION AND CONCLUSION

In this work, we proposed an evaluation of the importance of predictors to determine the effect of manipulated variables of an AR game on player performance. Our experimental results

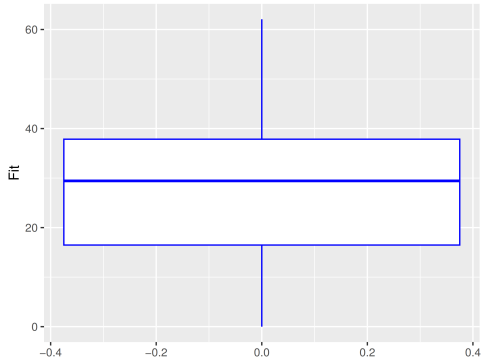


Fig. 4. Variability of R^2 for 1000 training and test trials.

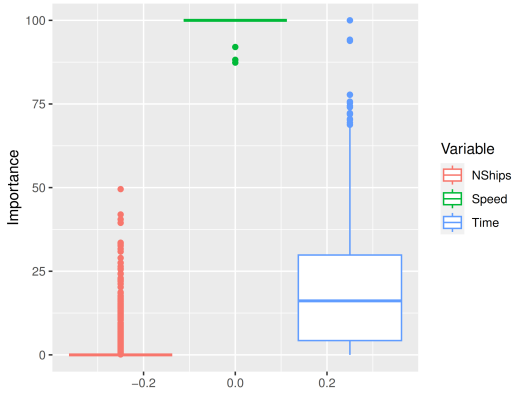


Fig. 5. Predictor's importance across 1000 training and test trials. *NShips* is the number of spaceships per horde, *Speed* is the speed of spaceships, and *Time* is the time between hordes.

showed that, for *Invasion Victory*, the speed of the spaceships had higher predictive power than the number of spaceships per horde and the time elapsed between the appearance of two consecutive hordes. This result was due to the fact that the faster the spaceships, the higher the probability of collision with the user and the lower the possibility of destroying them. Furthermore, since the maximum number of spaceships that a user can destroy is 50, which is independent of the selected speed, faster spaceships can only be detrimental to the player. On the other hand, a higher number of spaceships per horde can be beneficial because it increases the highest possible score a user can achieve, but this choice is also risky as it increases the probability of being hit by an enemy bullet or collided by a spaceship. The time between hordes showed a low predictive power (median = 16.4%) because this even integer number between 6 and 30 seconds does not condition the maximum number of spaceships that can be destroyed. Its main effect is to determine the game duration. These findings challenged our initial selection of manipulated variables, and we concluded that only the speed of the spaceships should be considered.

The low agreement between the measured and predicted outcomes shown in Figure 4 can be explained by two factors.

The first one is the level of attention and engagement of the user during gameplay, which is affected by emotional states and external factors. The second one is the learning effect, which means that users may improve their skills from one session to another. These aspects suggest that further experiments should be conducted under controlled and supervised conditions to minimize their influence on the results.

One limitation of our study was the small sample size, which led to a wide dispersion of two of the four statistics considered. One of them was the predictive power of time between hordes, which varied between 0% and 100% across the 1000 trials, and the other one was the coefficient of determination, which varied from 0% to 62% during the 1000 trials. To overcome this issue, *Invasion Victory* will be distributed for free in *Google Play Store* expecting to gather data from at least 100 users playing an average of 10 times each.

Our proposed approach aims to simplify dynamic difficulty adjustment design by eliminating those variables whose positive and negative effects on player performance counterbalance each other. In the case of *Invasion Victory*, two of the three selected variables will not be considered to control the difficulty of the game. In subsequent studies, we will determine if the variables with the highest predictive power remain the same when different machine learning models are trained.

REFERENCES

[1] J. C. Lopes and R. P. Lopes, "A review of dynamic difficulty adjustment methods for serious games," in *Optimization, Learning Algorithms and Applications*, A. I. Pereira, A. Košir, F. P. Fernandes, M. F. Pacheco, J. P. Teixeira, and R. P. Lopes, Eds. Cham: Springer International Publishing, 2022, pp. 144–159.

[2] J. Nakamura and M. Csikszentmihalyi, "The concept of flow," in *Flow and the Foundations of Positive Psychology: The Collected Works of Mihaly Csikszentmihalyi*. Dordrecht: Springer Netherlands, 2014, pp. 239–263.

[3] M. Zohaib, "Dynamic difficulty adjustment (DDA) in computer games: A review," *Advances in Human-Computer Interaction*, vol. 2018, 2018.

[4] M. Ishihara, S. Ito, R. Ishii, T. Harada, and R. Thawonmas, "Monte-Carlo tree search for implementation of dynamic difficulty adjustment fighting game AIs having believable behaviors," in *IEEE Conference on Computational Intelligence and Games (CIG)*, 2018, pp. 1–8.

[5] V. Verma, S. D. Craig, R. Levy, A. Bansal, and A. Amresh, "Domain knowledge and adaptive serious games: Exploring the relationship of learner ability and affect adaptability," *Journal of Educational Computing Research*, vol. 60, pp. 406–432, 2021.

[6] S. Willwacher and O. Korn, "Gamification of movement exercises in rehabilitation and prevention: A framework for smart training in AI-based exergames," in *Advances in Industrial Design*, vol. 260. Cham: Springer International Publishing, 2021, pp. 855–862.

[7] P. Ramasamy, S. Das, and Y. Kurita, "Ski for squat: A squat exergame with pneumatic gel muscle-based dynamic difficulty adjustment," in *Universal Access in Human-Computer Interaction. Access to Media, Learning and Assistive Environments*, vol. 12769. Cham: Springer International Publishing, 2021, pp. 449–467.

[8] H. Hsieh, "Generation of adaptive opponents for a predator-prey game," *Asia University*, 2008.

[9] C. Pedersen, J. Togelius, and G. Yannakakis, "Modeling player experience for content creation," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 2, no. 1, pp. 54 – 67, 2010.

[10] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, "On-line adaptation of game opponent AI with dynamic scripting," *International Journal of Intelligent Games and Simulation*, vol. 3, no. 1, pp. 45 – 53, 2004.

[11] R. Hunicke and V. Chapman, "AI for dynamic difficulty adjustment in games," *Challenges in game artificial intelligence AAAI workshop*, vol. 2, pp. 91 – 96, 2004.

[12] C. H. Tan, K. C. Tan, and A. Tay, "Dynamic game difficulty scaling using adaptive behavior-based AI," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 4, pp. 289–301, 2011.

[13] X. Li, S. He, Y. Dong, Q. Liu, X. Liu, Y. Fu, Z. Shi, and W. 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)*, vol. 8. IEEE, 2010, pp. 475–478.

[14] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *Machine Learning: ECML 2006*, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 282–293.

[15] A. Ebrahimi and M.-R. Akbarzadeh-T, "Dynamic difficulty adjustment in games by using an interactive self-organizing architecture," in *Iranian Conference on Intelligent Systems (ICIS)*. IEEE, 2014, pp. 1–6.

[16] M. Ninaus, K. Tsarava, and K. Moeller, "A pilot study on the feasibility of dynamic difficulty adjustment in game-based learning using heart-rate," in *Games and Learning Alliance*, vol. 11899. Cham: Springer International Publishing, 2019, pp. 117–128.

[17] Andrew, A. N. Tjokrosetio, and A. Chowanda, "Dynamic difficulty adjustment with facial expression recognition for improving player satisfaction in a survival horror game," *ICIC Express Letters*, vol. 14, pp. 1097–1104, 2020.

[18] A. Schwarz, G. Cardon, S. Chastin, J. Stragier, L. De Marez, and A. DeSmet, "Does dynamic tailoring of a narrative-driven exergame result in higher user engagement among adolescents? results from a cluster-randomized controlled trial," *International Journal of Environmental Research and Public Health*, vol. 18, no. 14, p. 7444, 2021.

[19] C. Peng, D. Wang, Y. Zhang, and J. Xiao, "A visuo-haptic attention training game with dynamic adjustment of difficulty," *IEEE Access*, vol. 7, pp. 68 878–68 891, 2019.

[20] L. Cardia da Cruz, C. A. Sierra-Franco, G. F. M. Silva-Calpa, and A. Barbosa Raposo, "A self-adaptive serious game for eye-hand coordination training," in *HCI in Games*, vol. 12211. Cham: Springer International Publishing, 2020, pp. 385–397.

[21] J. Suaza, E. Gamboa, and M. Trujillo, "A health point-based dynamic difficulty adjustment strategy for video games," in *Entertainment Computing and Serious Games*. Cham: Springer International Publishing, 2019, pp. 436–440.

[22] M. A. Gutiérrez, J. J. Rosero, D. E. Guzmán, and C. F. Rengifo, "A music therapy serious game with dynamic difficulty adjustment for stimulating short-term memory," in *Converging Clinical and Engineering Research on Neurorehabilitation IV*, vol. 28. Cham: Springer International Publishing, 2021, pp. 723–734.

[23] M. Morosan and R. Poli, "Automated game balancing in Ms PacMan and StarCraft using evolutionary algorithms," in *Applications of Evolutionary Computation*, vol. 10199. Cham: Springer International Publishing, 2017, pp. 377–392.

[24] P. M. Blom, S. Bakkes, and P. Spronck, "Modeling and adjusting in-game difficulty based on facial expression analysis," *Entertainment Computing*, vol. 31, 2019.

[25] T. Huber, S. Mertes, S. Rangelova, S. Flutura, and E. André, "Dynamic difficulty adjustment in virtual reality exergames through experience-driven procedural content generation," in *IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE Computer Society, 2021.

[26] Unity Technologies, "Unity real-time development platform | 3d, 2d vr & ar engine." [Online]. Available: https://unity.com

[27] "Brainstorming," in *Encyclopedia of Creativity, Invention, Innovation and Entrepreneurship*, E. G. Carayannis, Ed. Cham: Springer International Publishing, 2020, pp. 204–204.

[28] A. Menzinsky, G. López, J. Palacio, M. A. Sobrino, R. Álvarez, and V. Rivas, *Historias de usuario*. Scrum Manager, 2022, no. 3.01, ingeniería de requisitos Ágil.

[29] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[30] A. Mizumoto, "Calculating the relative importance of multiple regression predictor variables using dominance analysis and random forests," *Language Learning*, vol. 73, no. 1, pp. 161–196, 2023.