PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

Algoritmo para el balanceo dinámico del grado de dificultad mediante aprendizaje de máquina en la implementación de un juego orientado a apoyar el desarrollo de la inteligencia espacial en niños de etapa pre-escolar

Tesis para optar por el Título de Ingeniero Informático, que presenta el bachiller:

Franco André Caballero Torres

ASESOR: Dr. César A. Beltrán Castañón

Lima, marzo de 2018

Dedicatoria

Al Creador, quien me dio la capacidad de conocer.

Al Redentor, quien me enseñó a usar lo que sé para el bien.

Al Santificador, quien me dio los dones para no darme por vencido.

A mis padres, cuya confianza fue mi fuerza y su futura alegría mi motivación.

A San José de Cupertino, cuya intercesión siempre me acompañó.

Y a Mary, la voz tangible que no dejó que me rinda.

"Dame el tiempo y la disposición, la fortaleza de mente y de corazón,
y así cumplir correctamente esta misión,
para alegría de mis padres, desarrollo mío
y gloria tuya, mi Señor".

Resumen

Dentro del ámbito educativo nacional, la Inteligencia espacial, a pesar de haber demostrado estar relacionada con una serie de habilidades que permiten y estimulan la creación y el desarrollo matemático y científico, no es muy reconocida y posee pocas herramientas que ayuden a su desarrollo en niños de edad preescolar, etapa en la que este se recomienda ampliamente. Sumado a esto encontramos la necesidad de herramientas que contribuyan en la enseñanza cuyos requerimientos han crecido en cantidad y complejidad en las últimas décadas, y que involucren modos innovadores de llevar el conocimiento aprovechando las tecnologías disponibles. La necesidad de contribuir con la educación también surge de los requerimientos del Aprendizaje Adaptativo, el cual es una metodología que, a través de la adecuación del nivel del contenido que se desea enseñar, permite al estudiante una experiencia de aprendizaje personalizada y más efectiva en resultados. Esta metodología aprovecha las posibilidades de interacción que proporcionan las tecnologías de información y la capacidad de procesamiento de los equipos informáticos para lograr su objetivo. La presente tesis describe el desarrollo de una aplicación educativa gamificada de apoyo en el desarrollo de la Inteligencia espacial en niños de etapa preescolar, e involucra el uso de tecnologías que permitan adaptar al estudiante la dificultad del juego presentado por el aplicativo. Para esto se hizo uso de métodos de Ajuste Dinámico de la Dificultad, a través de redes neuronales y aprendizaje supervisado. El entorno de juego está basado en el uso de representaciones virtuales de bloques lógicos, mediante los cuales se le presenta al alumno una figura la cual este debe imitar manipulando, mediante la pantalla táctil, otro conjunto bloques similares. Se evaluaron siete métricas en el desempeño del usuario relacionadas a cuan correcta es su respuesta en los siguientes conceptos: Encaje, ubicación, forma, tamaño, color, rotación y textura. Mediante estas métricas la aplicación elige el siguiente escenario a presentar al usuario ajustando diez atributos en dicho escenario. El proceso de adaptación busca introducir las métricas del usuario a un rango de acierto deseado y se realiza en dos pasos. Primero, se realiza sin presencia del usuario un entrenamiento de redes neuronales mediante propagación hacia atrás con información de casos base. Este primer paso permite obtener una versión inicial de la adaptabilidad. Y segundo, luego de cada ronda, se evalúa la respuesta del usuario mediante un conjunto de eventos que determinan la efectividad de la red neuronal para introducir a un usuario especifico al rango deseado, y se modifica la red usada para ese usuario con los resultados obtenidos. En los resultados del

proyecto se observó que la metodología empleada es efectiva para el caso propuesto, logrando introducir las métricas en el rango luego de un número de rondas jugadas. La evaluación de requerimientos computacionales (velocidad, efectividad, robustez y eficiencia) y funcionales (claridad, variedad, consistencia y escalabilidad) para una AI adaptativa también muestra resultados positivos. Sobre la rapidez de la solución, la respuesta para ambos modelos (solo entrenamiento inicial y modificación por eventos) es imperceptible para el usuario. En cuanto eficacia se logró resultados positivos, logrando mejorar las métricas respecto a un algoritmo manual en más del 70% de los casos y obteniendo un aumento promedio comparándola a un algoritmo manual de +0.012 para las redes neuronales y +0.02 para el aprendizaje supervisado. Estos valores representan el 13% y el 22% de la máxima mejora posible respectivamente. En cuanto a la robustez y eficacia, ambos modelos lograron adaptar la respuesta al usuario en la mayoría de casos y en un número similar de rondas, aunque el aprendizaje supervisado mostró ser más efectivo en el primer criterio, mejorando los resultados del algoritmo manual. Respecto a la variedad de los escenarios presentados se obtuvo, mediante la modificación por eventos, una menor variación entre estos, lo que se relaciona con la mejor adaptabilidad alcanzada. Y sobre la escalabilidad, ambos modelos mostraron resultados positivos para los tres niveles de desempeño evaluado, aunque el aprendizaje supervisado muestra ser más efectivo. Estos resultados permiten identificar beneficios en el uso de esta metodología específicamente para el ámbito evaluado, así como identificar en qué casos específicos es más efectiva. Los resultados positivos encontrados que en conjunto indican que se ha logrado realizar una aplicación que cumple en presentar al usuario un entorno adaptativo, hacen válido el seguir este camino para futuras investigaciones en la exploración de las aplicaciones gamificadas educativas de apoyo a la inteligencia espacial.

FACULTAD DE
CIENCIAS E INGENIERÍA
ESPECIALIDAD DE
INGENIERÍA INFORMÁTICA

PONTIFICIA **UNIVERSIDAD CATÓLICA** DEL PERÚ

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO INFORMÁTICO

TÍTULO: **Algoritmo para el balanceo dinámico del grado de dificultad mediante aprendizaje de máquina en la implementación de un juego orientado a apoyar el desarrollo de la inteligencia espacial en niños de etapa pre-escolar.**

ÁREA: Ciencias de la Computación.
ASESOR: Dr. César Armando BELTRÁN CASTAÑÓN
ALUMNO: Franco André CABALLERO TORRES
CÓDIGO: 20088915
TEMA N°: #669
FECHA: San Miguel, 16 de octubre de 2017

DESCRIPCIÓN

La inteligencia espacial es una de las siete inteligencias estudiadas por Gardner e implica la correcta percepción visual del entorno, la capacidad para crear y manipular imágenes mentales, y la orientación del cuerpo en el espacio. Está además relacionada con una gran serie de habilidades y capacidades que permiten y estimulan la creación, y estudios indican que su estimulación contribuye en el desarrollo del terreno matemático y científico. Sin embargo, en el ámbito nacional se promueve poco esta inteligencia, además el sistema de enseñanza tradicional es rígido y no lleva en consideración el conocimiento y habilidades del estudiante. En ese sentido, se requieren herramientas que contribuyan a su desarrollo, especialmente en la etapa inicial donde el impacto de desarrollar habilidades espaciales resulta más significativo. Las herramientas digitales pueden contribuir a este objetivo, ya que se ha demostrado que los juegos digitales usados en la educación impactan positivamente en la concentración y motivación del alumno desarrollando en el estudiante interés en la tecnología y habilidades para manejarlas. Por otro lado la mayor capacidad de procesamiento de los equipos informáticos, permiten implementar algoritmos especializados para analizar los datos de comportamiento obtenidos del alumno y adaptar el contenido a sus necesidades particulares. Esta tesis describe la implementación de una aplicación educativa gamificada de apoyo al desarrollo de la inteligencia espacial en niños de pre-escolar, e involucra el uso de tecnologías que permitan adaptar al estudiante la dificultad del juego presentado por el aplicativo. Para esto se hace uso de métodos de Balance Dinámico de la Dificultad, a través de redes neuronales y aprendizaje supervisado. El entorno de juego está basado en el uso de

Av. Universitaria 1801
San Miguel, Lima – Perú
Apartado Postal 1761
Lima 100 – Perú
Teléfono:
(511) 626 2000 Anexo 4801

representaciones virtuales de bloques lógicos, que son fichas con formas, tamaños y colores diferentes, mediante los cuales se le presenta al alumno una figura la cual este debe replicar manipulando, mediante la pantalla táctil, otro conjunto bloques similares. La elección de estos bloques para el aplicativo se debe a que estos son utilizados en la etapa preescolar para mejorar la abstracción espacial y reforzar los conceptos de forma, rotación, ubicación, tamaño y color entre otros, contribuyendo en el proceso a desarrollar habilidades espaciales. En este proyecto se evalúan siete métricas en el desempeño del usuario al replicar la figura mostrada: Encaje, ubicación, forma, tamaño, color, rotación y textura. Mediante estas métricas la aplicación elige el siguiente escenario que se le presenta al usuario ajustando los diez atributos de dicho escenario. Primero, se realizando un entrenamiento de redes neuronales mediante propagación hacia atrás con información de casos base y en un momento anterior a que el usuario utilice el aplicativo. Y segundo, mediante la evaluación de la respuesta del usuario ante un conjunto de eventos que determinan la efectividad de la red neuronal para introducir a un usuario especifico al rango de desempeño deseado, y que modifican la red de ese usuario con estos resultados tras cada ronda jugada. De este modo se consigue adaptar la dificultad de cada concepto involucrado de manera independiente y la dificultad del reto presentado al usuario será la adecuada para él.

OBJETIVO GENERAL

Desarrollar una aplicación de entorno gamificado que a través de aprendizaje de máquina balancee dinámicamente el grado de dificultad, a fin de apoyar al desarrollo de la inteligencia espacial en niños de educación pre-escolar.

OBJETIVOS ESPECÍFICOS

Los objetivos específicos son:

OE1. Definir que conceptos de la inteligencia espacial se evaluarán en el usuario a través de su interacción con los bloques lógicos; indicando para cada uno de estos conceptos la forma de evaluación y las métricas necesarias.

OE2. Diseñar un prototipo de software basado en entorno de juego que permita al usuario manipular bloques lógicos de manera virtual y posibilite evaluar los conceptos anteriormente definidos.

OE3. Implementar un mecanismo que permita evaluar la figura que presenta la aplicación y la respuesta del usuario.

OE4. Implementar un mecanismo que permita realizar un balance dinámico de la dificultad en cada uno de los conceptos evaluados

OE5. Integrar en una aplicación el entorno definido y el mecanismo de adaptabilidad desarrollado, y desarrollar pruebas que permitan evaluar la efectividad de la adaptabilidad.

ALCANCE

Este proyecto realizará una aplicación que permita trabajar con bloques lógicos en forma virtual, presentando al usuario una figura formada con estos, de modo que este pueda ser reproducida. La dificultad de la figura presentada se adaptará a las habilidades que el usuario presente en cada uno de los conceptos a evaluar. Estos conceptos han sido definidos en entrevistas con profesionales en psicología y educación teniendo en consideración la edad de los estudiantes y las limitaciones de la tecnología y recursos disponibles. Estos conceptos son la forma, tamaño, encaje, ubicación, rotación, color y textura del bloque. Además, el proyecto se centrará principalmente en lograr el balance dinámico de la dificultad que la aplicación presenta en cada uno de los conceptos, y esto se realizada a través de aprendizaje adaptativo aplicado a redes neuronales. Si bien se necesita un entorno de interacción con el estudiante (la aplicación), el proyecto no centra en el desarrollo de este ya que el objetivo principal es demostrar la adaptabilidad. El proyecto usará solo las características que pueden ser medidas en la interacción del alumno con la pantalla táctil y no se analizarán otras formas de interacción. Finalmente, este proyecto estará orientado a alumnos de educación inicial, restringiendo sus edades a 3, 4 y 5 años y no considerará a usuarios con habilidades especiales o diferentes.

Máximo: 100 páginas

Tabla de contenido

Índice de imágenes

Índice de tablas

1. DEFINICIÓN DEL PROBLEMA

El presente documento detalla la investigación y trabajo realizado para llevar a cabo este proyecto de fin de carrera. En este primer capítulo se explica cuál es la problemática que ha inspirado el proyecto, se detalla la solución propuesta y se describen las herramientas a utilizar, así como sus limitaciones.

1.1. Problemática

Las inteligencias múltiples es un concepto de la teoría educativa estudiado por Gardner y que propone la existencia de múltiples inteligencias, donde cada una depende de una realidad biológica y se desarrolla en forma independiente de las demás (Gardner, 1983). Aunque esta teoría ha cambiado las estrategias educativas alrededor del mundo, incluyendo al Perú, los esfuerzos realizados suelen colocan principal énfasis en algunas de ellas, como la matemática y la lingüística, trabajando las demás en mucho menor proporción, siendo este el caso de la inteligencia espacial (Ipanaqué & Rojas, 2012). Sin embargo, a pesar de que a lo largo del tiempo suele solo enfatizarse su rol en el terreno artístico, esta inteligencia también está relacionada con una serie de habilidades que permiten y estimulan la creación, desempeñan un rol importante en el terreno matemático y científico, y que además estudios indican que suele estar altamente desarrollada en personas de éxito académico (Diezmann & Watters, 2000). Ante la falta de reconocimiento de esta inteligencia en el ámbito nacional y debido a que se han encontrado muy pocas herramientas digitales que ayuden a profundizar en su desarrollo, es necesario proponer soluciones que permitan a los estudiantes ponerla en práctica; y, como mencionan Diezmann & Watters (2000), orientar estos esfuerzos principalmente en entornos de educación pre-escolar debido a que se ha observado una relación directa con el éxito en etapas posteriores.

Otro problema que se observa es el que tradicionalmente la educación se imparte siguiendo una estructura fija y rígida en cuanto a los temas y a los tiempos de enseñanza, sin considerar el estado actual de los conocimientos y habilidades del estudiante o la velocidad de aprendizaje del mismo (Skinner, 1970). Este problema podría verse disminuido con la aplicación de una técnica educativa llamada Aprendizaje Adaptativo, la cual tiene la premisa básica de “adaptar el proceso educativo a las fortalezas y debilidades de cada estudiante”, incluyendo retroalimentación y corrección, pues se ajusta de acuerdo con las interacciones del estudiante y al nivel de desempeño demostrado (OIE, 2014). Esta metodología de enseñanza se apoya principalmente en la tecnología, por un lado, debido a las

posibilidades de interacción que proporcionan las tecnologías de información, y por otro lado debido a la capacidad de procesamiento de los equipos informáticos, que mediante algoritmos especializados analizan los datos obtenidos del alumno para adaptar el contenido a sus necesidades particulares (OIE, 2014).

Un tercer problema observado es el incremento de las necesidades educativas en calidad y cantidad que requieren de herramientas que apoyen su enseñanza. Respecto a esto, se ha demostrado que el uso de la tecnología juega un rol importante, pues esta permite a los alumnos acceder a nuevos recursos e impactan positivamente en la concentración y motivación hacia lo aprendido, además de preparar, sobre todo a los más pequeños, en habilidades tecnológicas que son necesarias en la "Era de la Información" que vivimos (Sunkel & Trucco, 2010). Dentro de esta línea, se puede prestar principal atención al aprendizaje basado en juegos informáticos, donde se observa que el uso de un entorno gamificado mejora el aprendizaje de los conceptos enseñados (Garris, Ahlers, & Driske, 2002), influyen en el rendimiento escolar (Llorca, 2009), y tiene el potencial de atraer, motivar y retener a los usuarios (Kuo & Chuang, 2015). Adicionalmente, algunos estudios indican que el desarrollo de las habilidades espaciales a través de videojuegos obtiene resultados superiores, más amplios y durables (Terleck, Newcombe, & Little, 2008); y el uso de juegos de informáticos parece ser un factor que ya de por sí puede contribuir sustancialmente en este tipo de inteligencia (Newcombe & Frick, 2010).

Se observa entonces la necesidad de una herramienta que apoye el desarrollo de la inteligencia espacial, que sea adaptativa para trabajar junto con las fortalezas y debilidades del alumno, y que presente un entorno gamificado para potenciar sus capacidades. Teniendo en cuenta esto se desarrolló el presente proyecto donde se plantea una alternativa de solución a estos problemas a través de las tecnologías de la información y de inteligencia artificial. Primero, se realizó una aplicación educativa gamificada orientada a apoyar el desarrollo de la inteligencia espacial en los niños a través del uso de representaciones virtuales de bloques lógicos; y segundo, se otorgó a la aplicación la capacidad de adaptar la dificultad del reto que presenta al usuario, cambiando dinámicamente los parámetros de este reto dependiendo de las respuestas y capacidades que el niño demuestre al ir usándola; y de este modo adaptar la dificultad en cada uno de los conceptos involucrados.

Sin embargo, adaptar las características de un juego de acuerdo con las habilidades particulares del usuario conlleva retos que dependen del problema particular que se desea enfrentar. Podemos mencionar como uno de estos retos la

existencia de usuarios con habilidades radicalmente diferentes entre ellos, sumado al hecho que estos pueden mejorar sus habilidades a diferentes velocidades (Nielsen, Usability Engineerin, 1993); o la dificultad de predecir como impactará el cambiar una característica particular del juego en el desempeño futuro del usuario (Hunicke & Chapman, 2004). Estas relaciones, dependiendo de la complejidad del problema, pueden no ser triviales. Muchas veces realizar un balance de la dificultad del juego involucra analizar una cantidad considerable de elementos del juego, del comportamiento del usuario y como se relacionan entre sí, la cual es una tarea difícil de realizar, y de hacerse manualmente consume muchos recursos (Rollings & Adams, 2003). Es por esto que se optó por el uso de inteligencia artificial, ya que sus algoritmos especializados permiten relacionar las características involucradas, reconocer patrones en el comportamiento del usuario o predecir los efectos de un cambio en el juego (Andrade, Ramalho, & Santana, 2005). Se decidió utilizar una técnica informática conocida como Balance Dinámico de la Dificultad (Sampayo-Vargas, Cope, He, & Byrne, 2013), desarrollada aplicando redes neuronales y aprendizaje supervisado, debido a que el uso de estos elementos permiten procesar la gran cantidad de variables involucradas, reconocer patrones en diversos escenarios (Matich, 2001) y la técnica ha demostrado ser útil para descubrir relaciones y adaptar los diferentes elementos de un juego (Pfeifer, 2009). De este modo la dificultad del reto presentado a cada usuario será la adecuada para él en cada uno de los conceptos evaluados, contribuyendo así el desarrollo de sus capacidades espaciales en forma gradual.

1.2. Objetivo general

Desarrollar una aplicación de entorno gamificado que a través de aprendizaje de máquina balancee dinámicamente el grado de dificultad, a fin de apoyar al desarrollo de la inteligencia espacial en niños de educación pre-escolar.

1.3. Objetivos específicos

A continuación, se enumeran, en el orden cronológico de su desarrollo, los objetivos específicos del presente proyecto.

1. **Definir qué conceptos de la inteligencia espacial se evaluarán en el usuario a través de su interacción con los bloques lógicos; indicando para cada uno de estos conceptos la forma de evaluación y las métricas necesarias.**

Estas métricas deben permitir conocer el estado de las habilidades del niño en los conceptos evaluados de modo que luego se pueda realizar la adaptación de la

dificultad del reto presentado.

2. **Diseñar un prototipo de software basado en entorno de juego que permita al usuario manipular bloques lógicos de manera virtual y posibilite evaluar los conceptos anteriormente definidos.**

Se buscará que este entorno sea adecuado para alumnos de pre-escolar según las recomendaciones de profesionales del área, identificando en el proceso las necesidades arquitectónicas de la aplicación para desarrollar una propuesta satisfactoria.

3. **Implementar un algoritmo que permita evaluar la figura que presenta la aplicación y la respuesta del usuario.**

La eficacia de este algoritmo se verificará con un análisis estadístico, ya que es necesario obtener adecuadamente las métricas para luego poder usarlas en la adaptación de la dificultad.

4. **Implementar un módulo que permita realizar un balance dinámico de la dificultad en cada uno de los conceptos evaluados.**

La propuesta inicial se basa en aprendizaje de máquina usando redes neuronales y propagación hacia atrás, para luego ir refinando el aprendizaje usando técnicas de aprendizaje supervisado mediante eventos.

5. **Integrar en una aplicación el entorno definido y el módulo de adaptabilidad desarrollado, y desarrollar pruebas que permitan evaluar la efectividad de la adaptabilidad.**

Aquí se incluye la toma pruebas con usuarios virtuales que permitan evaluar la efectividad de la adaptabilidad y el análisis de los resultados.

1.4. Resultados esperados

En la Tabla 1 se muestra la lista de resultados esperados de acuerdo a los objetivos específicos planteados.

Tabla 1: Resultados esperados

Objetivo específico	Resultado esperado	Modo de verificación	Fuente verificación
1. Definir los conceptos a evaluar en los usuarios	1.1 Modelo de conceptos a evaluar y métricas a usar.	Juicio experto de un profesional en Educación Inicial.	Documento con conceptos y métricas a evaluar.
	1.2 Requerimientos del módulo de adaptabilidad que definirá el reto a presentar al usuario	Recolección de requerimientos	Documento con los requerimientos del módulo de adaptabilidad.

2. Diseñar el entorno gamificado	2.1 Diseño de la interfaz gráfica de la aplicación	Uso de lineamientos de usabilidad.	Documento de diseño de la interfaz.
	2.2 Documento de arquitectura: Diagrama de actividades y Diagrama de clases.	Seguimiento de estándares UML	Documento de arquitectura.
3. Implementar algoritmo de comparación	3.1 Diseño del algoritmo de comparación y descripción en seudocódigo	Recomendaciones del documento de Verdine (2014).	Documento con notas de desarrollo y seudocódigo.
	3.2 Implementación del módulo de comparación. Prueba de eficacia usando gráficos parcialmente modificados	Comparación estadística obtenido versus esperado, T Student	Código fuente y resultados de comparación estadística
4. Implementar un módulo de adaptabilidad	4.1 Módulo de adaptabilidad en base a redes neuronales con aprendizaje básico por propagación hacia atrás.	Recomendaciones del documento de Pfeifer (2009)	Archivos con código y documento con notas de desarrollo.
	4.2 Módulo de adaptabilidad mejorado a través de aprendizaje supervisado con eventos.		Documento con notas de pruebas y mejoras.
5. Integrar en una aplicación el entorno y el módulo de adaptabilidad desarrollado	5.1 Aplicación gamificada integrando interfaz y módulos de adaptabilidad	Juicio de experto en aplicaciones informáticas.	Archivos y capturas de pantalla de la aplicación
	5.2 Pruebas con usuarios virtuales para verificar adaptabilidad y Análisis de resultados	Requerimientos funcionales y computacionales de Spronck (2005).	Documento de pruebas de adaptabilidad con usuarios.

1.5. Herramientas, métodos, metodologías y procedimientos

En la Tabla 2 se indican cuáles serán las herramientas, métodos, metodologías y procedimientos a usar en el presente proyecto, de acuerdo con los resultados esperados definidos. Más adelante se explica cada uno a detalle.

Tabla 2: Herramientas, metodologías, métodos y procedimientos

Resultados esperados	Herramientas, metodologías, métodos y procedimientos
1.1 Modelo de conceptos a evaluar y métricas a usar.	Principios generales de gestión de proyectos y levantamiento de requerimientos Principios generales de diseño basado en escenarios
1.2 Requerimientos del módulo de adaptabilidad que definirá el reto a presentar al usuario	
2.2 Diseño de la interfaz gráfica de la aplicación	Lineamientos de usabilidad para el diseño de IU. Uso de diagramas UML Principios generales de diseño basado en escenarios
2.3 Documento de arquitectura: Diagrama de actividades y Diagrama de clases.	
3.1 Descripción de la solución en seudocódigo	Visual Studio C# Windows Forms para interfaz de prueba
3.2 Implementación de la solución en código fuente. Pruebas de valides.	

4.1 Módulo de redes neuronales con aprendizaje básico.	Redes neuronales artificiales Aprendizaje supervisado en redes neuronales Visual Studio C# Windows Forms para interfaz de prueba
4.2 Módulo mejorado a través de aprendizaje supervisado.	
4.3 Validación del funcionamiento de los módulos de aprendizaje.	
5.1 Aplicación gamificada integrando interfaz y módulos de adaptabilidad	Unity Visual Studio C#
5.2 Pruebas con usuarios virtuales para verificar adaptabilidad y Análisis de resultados	Requerimientos Funcionales y Computaciones para una IA adaptativa de Spronk (2005).

1.5.1. Métodos, Procedimientos, metodologías y herramientas

En este punto se detallan diversos métodos y herramientas usadas en el Proyecto.

A. Unity

Es una plataforma avanzada para el desarrollo de video juegos. Creada por Unity Technologies, está centrada principalmente en la creación de juegos 3D, pero también se puede utilizar en juegos 2D. Su principal característica es la capacidad de poder desarrollar juegos para múltiples plataformas, programando en un solo lenguaje y con mínimos cambios (Unity Technologies, 2015). Posee una interfaz gráfica que permite manipular objetos, efectos y animaciones directamente en ella, permitiendo también agregar scripts de código puro.

Algunas de sus características son (Academia Android, 2015):

- Permite programar en C# o J# aprovechando el potencial de estos lenguajes.
- Posee una tienda, Unity Asset Store, donde se puede descargar todo tipo de contenido útil para una aplicación.
- Posee una interfaz intuitiva y muy completa.
- Posee una gran documentación disponible y una comunidad activa grande.
- Permite la integración con Visual Studio para realizar depuración en tiempo real.
- Ediciones gratuitas con funcionalidad completa.

B. Visual Studio Community

Es un entorno de desarrollo integrado (IDE) de Microsoft que permite realizar programas principalmente para plataformas Windows y Web, aunque también puede utilizarse para desarrollar aplicaciones para Android y IOS. Pertenece a la familia de IDEs Visual Studio, siendo esta versión Community gratuita y con funcionalidad casi completa. Incluye un editor de código que soporta IntelliSense, una ayuda para la autocompletado de código y otras características de productividad. Posee capacidades de depuración en tiempo real, rastreo de variables, entre otros.

Algunas de sus características son (Visual Studio, 2016):

- Editor de código que permite trabajar con cualquier lenguaje a través de plug-in, entre ellas C++, C#, VB.
- Depurador tanto a nivel de código fuente como a nivel de máquina.
- IDE muy completo e intuitivo.
- Diseñador para Windows Forms y WPF.
- Integración con GitHub y otros proveedores Git.
- Otras herramientas e integración con varios productos (plug-in para trabajar con Unity)
- Ediciones gratuitas con funcionalidad casi completa.

C. Diseño basado en escenarios

Según Rivero (Rivero, 1998), los escenarios especifican las posibles formas de usar el sistema para obtener alguna función necesaria para el usuario. Son usados para describir el comportamiento externo del sistema centrado en el punto de vista del usuario; son adecuados para validar las especificaciones de requisitos; ayudan a involucrar tempranamente al usuario; facilitan la interacción y proveen criterios de aceptación para la validación basada en requisitos. También indica que se ha comprobado que el uso de escenarios ha adquirido un interés creciente debido a otras ventajas: facilitan el trabajo interdisciplinario con una continua interacción diseñador-cliente, diseñador-usuario; reducen la complejidad, puesto que obligan a la descomposición del problema del análisis de requisitos en tempranas etapas de diseño; tienen un rol importante en la identificación y manejo de excepciones y permiten alcanzar acuerdos parciales y consistencia sobre el sistema reduciendo el alcance de los procesos de discusión y acuerdo

D. Lineamientos de usabilidad para el diseño de IU.

Nielsen establece 10 principios generales para el diseño relacionado con la interacción, los cuales pueden usarse como “reglas generales” que se recomiendan seguir (Nielsen, Heuristic evaluation, 1994). Estos principios son los siguientes:

1. **Visibilidad del estado del Sistema**: El sistema siempre debe mantener a los usuarios informados acerca de lo que está pasando, a través de una retroalimentación adecuada en un tiempo razonable.
2. **Relación entre el sistema y el mundo real**: El sistema debe hablar el idioma de los usuarios, con palabras, frases y conceptos familiares para él, en lugar de términos orientados al sistema. Se debe seguir las convenciones del mundo real, haciendo que la información aparezca en un orden natural y lógico.

3. **Control del usuario y libertad**: Los usuarios suelen elegir las funciones del sistema por error y una "salida de emergencia" para salir del estado no deseado. Soporte deshacer y rehacer.
4. **Consistencia y estándares**: Los usuarios no deberían tener que preguntarse si diferentes palabras, situaciones o acciones significan lo mismo. Se debe seguir las convenciones de la plataforma.
5. **Prevención de errores**: Se requiere un diseño que evite que un problema se produzca en primer lugar. Se deben eliminar las condiciones comunes de error, en especial las causadas por el usuario.
6. **Reconocer en lugar de recordar**: Se debe reducir al mínimo la carga de memoria del usuario al hacer objetos, acciones y opciones visibles. El usuario no debería tener que recordar la información de una parte del diálogo a otro. Instrucciones de uso del sistema deben ser visibles o fácilmente recuperables cuando sea apropiado.
7. **Flexibilidad y eficiencia del uso**: Aceleradores - no vistas por el usuario principiante - a menudo pueden acelerar la interacción para el usuario experto de tal manera que el sistema puede servir tanto a los usuarios sin experiencia y con experiencia. Permitir a los usuarios adaptar las acciones frecuentes.
8. **Diseño estético y minimalista**: Los diálogos no deben contener información que es irrelevante o raramente necesaria. Cada unidad adicional de información en un diálogo compite con las unidades pertinentes de información y disminuye su visibilidad relativa.
9. **Ayuda para reconocer, diagnosticar y recuperarse de errores**: Los mensajes de error deben ser expresadas en un lenguaje sencillo (sin códigos), indica con precisión el problema, y constructivamente sugerir una solución.
10. **Ayuda y documentación**: A pesar de que es mejor si el sistema puede ser utilizado usar documentación, puede ser necesario proporcionar ayuda y documentación. Dicha información debe ser fácil de buscar, centrarse en la tarea del usuario listando pasos concretos, y no ser demasiado grande.

1.6. Alcance

A continuación, se especifica lo que abarca y lo que no abarca el presente proyecto:

- En este proyecto se realizó una aplicación que permita trabajar con bloques lógicos en forma virtual, presentando al usuario una figura formada con estos, de modo que esta pueda ser reproducida lo más exactamente posible

manipulando un segundo conjunto de bloques. Se eligió trabajar con bloques lógicos debido a que es una de las actividades más usadas para evaluar y contribuir al desarrollo de las habilidades espaciales en edades tempranas, especialmente las de transformación y rotación. Estos bloque lógicos son figuras de formas, tamaños y colores diferentes que pueden ser manipuladas libremente por el estudiante, reforzando así los conceptos de forma, rotación, ubicación, tamaño y color entre otros (Verdine, Irwin, Golinkoff, & Hirsh-Pasek, 2014) (Yilmaz, 2009) (Botero, 1990).

- La modalidad de juego está basada en rondas, en donde en cada una se muestra una figura en particular a imitar.
- El proyecto estará orientado a alumnos de etapa pre-escolar, restringiendo sus edades a 3, 4 y 5 años, ya que, como mencionan Diezmann & Watters (2000), las actividades específicas para el desarrollo de la inteligencia espacial deberían ser realizadas principalmente en entornos de educación temprana, dado que se observa experimentalmente que la estimulación en esta etapa proporcionan resultados eficaces en la mejora del aprendizaje posterior.
- La aplicación está diseñada para usarse en un dispositivo Tablet, y hace uso de la pantalla táctil. Esto es debido a la corta edad del usuario final, el cual interactúa de forma más fluida con estos dispositivos; y a la capacidad de la pantalla táctil de permitir y medir gestos y movimientos de modo más intuitivo que otros dispositivos informáticos de fácil acceso. El proyecto usará solo las características que pueden ser medidas en la interacción del alumno con la pantalla táctil y no se analizarán otras formas de interacción.
- La dificultad del reto se adapta a las habilidades que el usuario presenta en cada uno de los conceptos evaluados. Solo se utiliza para la adaptabilidad técnicas de aprendizaje supervisado en redes neuronales.
- Los conceptos evaluados están relacionados con la forma, tamaño, ubicación, rotación y color del bloque. Estos conceptos han sido definidos en entrevistas con profesionales en psicología y educación teniendo en consideración la edad de los estudiantes, y las limitaciones de la tecnología y recursos disponibles.
- El proyecto se centra en lograr el balance dinámico de la dificultad que la aplicación presenta en cada uno de los conceptos, la cual se realiza a través de aprendizaje adaptativo aplicado a redes neuronales. Si bien se necesita un entorno de interacción con el estudiante (la aplicación), el proyecto no se centra en el desarrollo de este ya que el objetivo principal es demostrar la adaptabilidad.

- El proyecto no considerará a usuarios con habilidades especiales o diferentes.

1.7. Limitaciones

A continuación, se presentan las limitaciones y obstáculos del presente proyecto.

1. Las dificultades de realizar extensas pruebas con usuarios reales, ya que estos son alumnos de muy corta edad y se requiere autorización de los padres. Además, las pruebas requerirán repetición y uso intensivo del juego, lo cual no puede ser solicitado a usuarios reales
2. La compleja relación que puede existir entre las diversas capacidades a medir en el usuario y los múltiples conceptos de inteligencia espacial a evaluar, lo que hace en cierto grado impredecible los resultados.

1.8. Justificación

En los siguientes puntos se detalla las razones que justifican el desarrollo del presente proyecto.

- Es conveniente para los educadores contar con una herramienta que permitan enfrentar de manera personalizada las necesidades de aprendizaje de sus alumnos en cuanto a inteligencia espacial y percepción visual, ya que la adaptación al alumno mejora la comprensión de lo que se enseña y acelera el aprendizaje.
- Es conveniente para los estudiantes contar con una herramienta educativa gamificada ya que estas atraen y motivan a los estudiantes, lo cual se traduce indirectamente en un mayor interés en lo enseñado.
- Es necesario para los estudiantes de nivel inicial tener una herramienta que ayude a mejorar sus capacidades en inteligencia espacial y percepción visual, ya que esto se traduce en un mejor desempeño académico en el área de ciencias y matemática, lo cual tendría un impacto social.

1.9. Gestión de proyecto - Análisis de Viabilidad

Se realizó un apartado donde se analizó si era posible realizar este proyecto antes de iniciarlo, desde un punto de vista temporal, tecnológico y económico, demostrando que se contaba con los recursos adecuado. Este puede observarse en el Anexo 1.

1.10. Metodología para el desarrollo del proyecto

Para el desarrollo de este proyecto de utilizó una metodología propia inspirada principalmente en el Proceso Unificado Ágil (AUP), ya que mezcla la formalidad en

el proceso de desarrollo, necesaria para realizar un seguimiento objetivo del mismo, y la adaptabilidad del proceso, necesario en una aplicación que, como en esta, no se tiene todos los requerimientos completamente definidos inicialmente. La metodología AUP presenta fases realizadas en forma lineal que guían el desarrollo del proyecto a gran escala, y disciplinas que representan la labor necesaria a realizar en cada una de las iteraciones o ciclos de menor duración realizados a lo largo del proyecto (Ambler, 2006).

Debido a que el proyecto involucra no solo desarrollo de software sino principalmente el desarrollo de un algoritmo, se ha tenido que ajustar la metodología a las necesidades particulares. Cabe resaltar que, para el presente proyecto, el aplicativo (que permite interactuar con el usuario y recopilar información) y el algoritmo (que usa esta información y responde al usuario), no se ha desarrollado de manera separada, sino que se desarrollan de manera conjunta en todas las fases del proyecto. Esto es porque se busca que el proyecto presente una solución **integrada en todas sus partes**, y no desarrolladas por separado. Se plantea como parte de la tesis de este proyecto que, para un máximo aprovechamiento del desarrollo de debe de considerar que las características de un elemento afectan al otro: el algoritmo define lo que el aplicativo debe medir, pero el aplicativo limita lo que el algoritmo puede recibir.

Considerando esto se han adaptado las fases de desarrollo para el problema presentado, manteniendo el modelo de lineal original de AUP. Estas fases son las siguientes:

- **Concepción**: El objetivo es definir la solución en forma más concreta, las características que incluirá y los elementos que interactuaran con ella, tanto a nivel de aplicativo como de algoritmo.
- **Elaboración**: El objetivo es definir la arquitectura de la solución.
- **Prototipado**: El objetivo es construir el software inicial con los elementos principales necesarios para la experimentación.
- **Experimentación**: El objetivo es mejorar el software a través de incrementos iterativos, donde se analiza tanto las capacidades del aplicativo como la efectividad del algoritmo, realizando cambios de acuerdo a los resultados obtenidos. En proceso iterativo implica los cambios propios del enfoque incremental (basado en la prioridad de las características) y los cambios propios del análisis (basado en las pruebas sobre el algoritmo).
- **Integración**: El objetivo es completar la integración de algoritmo y el aplicativo, realizar ajustes finales y resolver problemas relacionados al despliegue.

Las disciplinas involucradas en cada iteración también se han adaptado a las necesidades del proyecto, siendo estas las siguientes.

- **Modelado de áreas, métricas y diseño**: EL objetivo es entender el problema (ya que este por su naturaleza no está bien definido) y los requerimientos de este, y como este afecta el diseño tanto del algoritmo como de la aplicación.
- **Implementación de interfaz**: El objetivo transformar la parte del modelado relacionada a la interfaz en código ejecutable y realizar pruebas unitarias.
- **Implementación de algoritmos**: El objetivo transformar la parte del modelado relacionada al algoritmo en código ejecutable.
- **Entrenamiento y pruebas**: El objetivo es realizar el entrenamiento del algoritmo y las pruebas que permitan analizar su efectividad, tanto relacionadas con la interfaz como solo desde la parte algorítmica.
- **Gestión del cambio**: El objetivo es controlar los cambios en el proyecto de manera ordenada, manejando el avance en los requerimientos añadidos y las versiones de la solución.

Se realizó una iteración para cada una de las tres primeras fases, así como para la última, mientras que para la fase de Experimentación si se realizó un número mayor de iteraciones. En la Ilustración 1 se observa gráficamente la aplicación de la metodología para este proyecto.

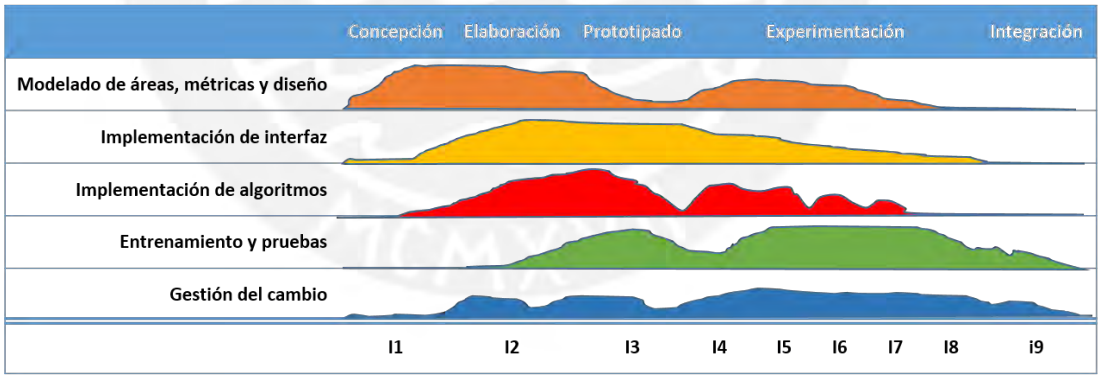

Ilustración 1: Metodología implementada y su ejecución a lo largo del desarrollo

La gestión del cambio de realizo aprovechando las características de versionamiento de la herramienta de almacenamiento usada. También se controló cambios realizados al proyecto y los requerimientos que surgieron durante el desarrollo a través de un registro de requerimientos y cambios implementado en OneNote, aprovechando su flexibilidad, capacidad de crear jerarquías y uso de iconos visuales que facilitan la comprensión.

2. MARCO CONCEPTUAL

Esta sección explica los conceptos que han sido mencionados en la problemática y otros asociados a estos.

2.1. Objetivo del marco conceptual

Explicar los principales conceptos relacionados al Proyecto, de modo que se puedan comprender los temas tratados en y sus implicaciones.

2.2. Definiciones

Se presentan a continuación algunos términos importantes.

2.2.1. Balance dinámico de la dificultad - BDD (*Dynamic game difficulty balancing*)

También conocido como ajuste dinámico de la dificultad (ADD), es un proceso en el que automáticamente se modifica los parámetros, comportamientos y escenarios de un juego en base al desempeño y decisiones que toma el usuario. El objetivo principal es que la dificultad del juego sea la indicada para el usuario y así evitar que este sea muy complicado, ya que el usuario se sentiría frustrado, o muy fácil, ya que el usuario se sentiría aburrido. Presentándole al usuario un reto adecuado se logra que este valore el juego como útil durante mucho más tiempo, manteniendo además un nivel de interés constante (Hunicke & Chapman, 2004). En la Ilustración 2 se muestra el flujo que se debe seguir para mantener un balance dinámico de la dificultad.

Ilustración 2: Flujo que sigue la relación entre el reto que presenta el juego y la habilidad del usuario. Adaptado de (Hunicke & Chapman, 2004)

Se ha buscado integrar este proceso en una gran variedad de juegos, con resultados diferentes. En algunos casos se ha logrado una integración exitosa que mantiene un flujo adecuado de cambios sin alterar otra característica del juego; sin

embargo, en la mayoría de casos esto no se ha logrado, principalmente debido a la dificultad y complejidad de tomar decisiones en tiempo real sin afectar el rendimiento o de confundir al usuario con decisiones drásticamente diferentes a las que él esperaba (Spronck, Ponsen, & Sprinkhuizen-Kuyper, 2005). Sin embargo, las investigaciones es este campo continúan ya que una inteligencia artificial que adapte la dificultad de un juego eficientemente implica una ventaja significativa sobre otros.

2.2.2. Inteligencia espacial

La Inteligencia Espacial implica la correcta percepción visual del entorno, la capacidad para crear y manipular imágenes mentales, y la orientación del cuerpo en el espacio (Dickinson, 1996). Está además relacionada con una gran serie de habilidades y capacidades que permiten y estimulan la creación. demostrando también su desarrollo ser muy importante en el terreno matemático y científico; un análisis cuidadoso de personas altamente creativas en estos dos últimos terrenos encuentra índices de inteligencia espacial muy elevados en ellas, y sugiere que esta es un área muy importante de la capacidad humana (Diezmann & Watters, 2000). También se refiere a ella como habilidad espacial. La inteligencia espacial es una de las siete inteligencias propuestas por Howard Gardner (1983) quien la describe como una inteligencia que incluye la habilidad para percibir y visualizar el mundo espacio-visual con exactitud y para formar y manipular imágenes mentales. Está relacionada con la manipulación de información presentada en forma visual, esquemática o simbólica, en contraste con la modalidad verbal basada en el lenguaje (Lohman, D. F., Pellegrino, J. W., Alderton, D. L., & Regian, J. W., 1987) La inteligencia espacial se puede inferir de la habilidad para invocar y usar una forma particular de representación y razonamiento. Razonar con representaciones espaciales (incluyendo diagramas, dibujos, mapas y modelos) difiere sustancialmente del razonamiento secuencial basado en las representaciones lingüísticas como el texto (Rogers, 1995).

2.2.3. Percepción visual

Se refiere a la habilidad del cerebro para dar sentido a lo que observa a través de los ojos. No se refiere a ver perfectamente lo que se presenta, sino a la capacidad de comprenderlo adecuadamente. En niños pequeños, esta habilidad es importante para realizar actividades cotidianas como leer, escribir, vestirse, completar rompecabezas, cortar, dibujar y completar problemas, entre otras. Sin la habilidad de completar estas tareas diarias, la autoestima del niño puede verse disminuida y su capacidad académica comprometida. El centro de terapia ocupacional Kid Sense

(2013) establece los siguientes “bloques de construcción” necesarios para el adecuado desarrollo de la percepción visual:

- Procesamiento sensorial: El adecuado registro, interpretación y respuesta a estimulaciones sensoriales en el ambiente y en el propio cuerpo.
- Atención visual: La habilidad de enfocarse en información visual importante y filtrarla de la información no importante de fondo.
- Discriminación visual: La habilidad de determinar diferencias o similitudes entre objetos basados en su forma, color, tamaño y otros.
- Memoria visual: La habilidad de recordar rasgos de una forma u objeto.
- Relaciones visuales-espaciales: Entender la relación del objeto con el entorno.
- Memoria visual-secuencial: La habilidad de recordar una secuencia de objetos en el orden correcto.
- Diferencia visual fondo-figura: La habilidad de encontrar algo en un fondo complejo.
- Constancia visual de forma: La habilidad para reconocer que la forma de la figura es la misma, incluso si ha cambiado su tamaño o se ha girado.
- Cierre visual: La habilidad para reconocer un objeto o forma cuando parte de la figura está oculta.

2.2.4. Habilidades de inteligencia espacial - percepción visual

En una entrevista realizada con la profesional en psicología Angélica Aedo (Aedo, Primera entrevista, 2015) (Aedo, Segunda entrevista, 2015) se identificaron los siguientes conceptos básicos relacionados a la inteligencia espacial y percepción visual. En una entrevista posterior con la licenciada en educación Susana Torres (Torres, 2015) se corroboró que estos conceptos básicos serían muy adecuados para una aplicación educativa que incentive el desarrollo de la inteligencia espacial en el estudiante de etapa preescolar.

- Reconocimiento de figuras: Capacidad para diferenciar una figura o forma de otra.
- Traslación de figuras: Capacidad para poder ordenar un conjunto de figuras para igualar con exactitud un modelo ya ordenado.
- Rotación de figuras: Capacidad para mentalmente rotar figura y reconocerlas así se encuentren en una posición diferente.
- Escalamiento de figuras: Capacidad para mentalmente aumentar o disminuir el tamaño de una figura en forma proporcional, y reconocerla así se encuentre en otra escala.

- Ordenamiento de figuras: Capacidad para comprender los conceptos de delante/detrás y reconocer una figura en el orden adecuado.
- Percepción de color: Capacidad para poder diferenciar los colores adecuadamente.
- Percepción de textura: Capacidad para reconocer y diferenciar adecuadamente patrones o texturas de fondo, aun si la forma usada para contenerlos es la misma.
- Percepción de simetría: Capacidad para reconocer y diferenciar objetos aun cuando estén invertidos.

2.2.5. Aprendizaje adaptativo educacional

La enseñanza tradicional se basa principalmente en un modelo de enseñanza lineal. Esto se refiere a presentar la enseñanza con una secuencialidad preestablecida, donde se dan a conocer los conceptos uno tras otro sin asegurar su correcto aprendizaje (Skinner, 1970). Ante esta realidad Skinner propone una mejora en el aprendizaje que derivó en el desarrollo de su "Máquina de enseñar". Se dice que él "tuvo esta idea al visitar la escuela de uno de sus hijos, en la clase de matemáticas, donde se dio cuenta que el profesor hacía todo lo contrario a los principios de aprendizaje, los niños hacían problemas de matemáticas uno tras otro, sin recibir *feedback* alguno, hacían cuentas una tras otra sin saber si los resultados estaban bien o no" (Aguayo). Skinner observó que el proceso de aprendizaje debe ser dividido en un "gran número de pasos muy pequeños" y el reforzamiento debe depender de la realización de cada paso (Wleklinski, 2011). Esta nueva forma de enseñanza derivó en la Teoría del Aprendizaje Programado, la cual ha resultado en la metodología de enseñanza que se conoce actualmente como Aprendizaje Adaptativo

Esta metodología de enseñanza tiene la premisa básica de "adaptar el proceso educativo a las fortalezas y debilidades de cada estudiante" (OIE, 2014). Este aprendizaje tiene una aproximación no-lineal a la instrucción, retroalimentación y corrección, pues se ajusta de acuerdo a las interacciones del estudiante y al nivel de desempeño demostrado. Consecuentemente, se adapta y anticipa el tipo de contenidos y recursos que este necesitará en un momento específico para progresar en el curso que se desea enseñar (OIE, 2014). Usa el potencial de tres elementos: El acceso a la tecnología desarrollada en la última década; la existencia de sofisticada analítica de datos y técnicas de aprendizaje; y la investigación emergente sobre cómo aprenden las personas. Con la convergencia de estos tres elementos es posible generar sistemas inteligentes de aprendizaje adaptativo

(Lemke, 2014). En la Ilustración 3 se observa esta convergencia.

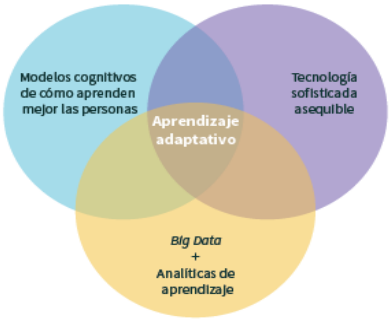

Ilustración 3: Fenómenos convergentes de apoyo al aprendizaje adaptativo. Adaptación de Lemke, 2014. Recuperado de (OIE, 2014).

En los últimos años esta metodología de enseñanza ha tenido una especial mención en los ambientes educativos internacionales. En el 2012, de acuerdo al reporte de adopción de tendencias del grupo consultor Gartner, el aprendizaje adaptativo se encontraba cerca del punto más alto del pico de expectativas sobredimensionadas, y en el 2013, el grupo ubicó a la tendencia justo atravesando la etapa del abismo de desilusión para luego volver a subir indicando un alto potencial de crecimiento durante los siguientes años al mismo tiempo que se comenzó a observar cada vez más implementaciones en el ámbito educativo (OIE, 2014).

2.2.6. *Game-based learning* (Aprendizaje basado en el juego)

Este concepto se refiere al uso de los juegos como una herramienta educativa. Cuando se usa juegos de computadora y juegos en general para propósitos educativos, muchos aspectos de proceso de aprendizaje son soportados: Los estudiantes son motivados a combinar conocimiento de diferentes áreas para encontrar una solución o para tomar una decisión en un cierto punto, los estudiantes pueden probar como el resultado del juego cambia basados en sus decisiones y sus acciones. Son también motivados a interactuar con otros miembros del equipo y discutir y negociar subsecuentes pasos, mejorando, entre otras cosas, sus habilidades sociales (Pivec, 2003, julio)

El modelo que se sigue en el aprendizaje basado en juegos es el siguiente (Garris, Ahlers, & Driske, 2002): Se debe tener un programa con contenido instruccional (un programa que busca enseñar algo al usuario) que incorpore características de un juego. Estas características desatarán una serie de “ciclos” en los que el usuario emite juicios, reacciona con interés y disfrute, muestra persistencia en la actividad y obtiene mayor retroalimentación del sistema. De este modo cuando se alinean adecuadamente contenido educativo y características de un juego se tiene como

resultado una recurrente automotivación para jugar, lo que resulta en el cumplimiento de los objetivos de aprendizaje planteados. Este ciclo se puede apreciar en la Ilustración 4.

Ilustración 4: Modelo del ciclo de juego. Traducido de (Garris, Ahlers, & Driske, 2002)

2.2.7. Redes neuronales artificiales

Desacuerdo a Pfeifer (Pfeifer, 2009) las redes neuronales artificiales tratan de imitar el modelo de interconexión de las neuronas del cerebro humano, pero a una mucho menor escala. Cuenta con entradas que representan variables independientes y salidas que representan los resultados independientes de las variables de entrada. De este modo la red en sí es una función matemática compleja que da un conjunto único de salida para un conjunto de entrada. Las redes neuronales son usadas por la comunidad desarrolladora de inteligencia artificial para solucionar todo tipo de problemas generalmente relacionados con el reconocimiento de patrones, incluyendo imágenes manuscritos y secuencias de tiempo como las tendencias financieras (Matich, 2001). Las redes neurales a menudo se combinan con otras técnicas tales como sistemas *fuzzy*, algoritmos genéticos y métodos probabilísticos.

También existen las llamadas “redes neuronales artificiales evolutivas” que en lugar de aprendizaje por refuerzo se usan algoritmos genéticos que crean permutaciones de la red y juzgando los resultados de una función de finura. Las mejores redes se utilizan entonces para crear una nueva generación en la que se repite el proceso. Este ha demostrado ser un método de trabajo adecuado para la evolución de las redes neuronales con posibles mejores resultados, pero se considera que es demasiado lento para un aprendizaje supervisado (Spronck, Ponsen, & Sprinkhuizen-Kuyper, 2005).

2.2.8. Aprendizaje supervisado

Aunque diferentes autores varían la definición del término, se puede entender como un modo de aprendizaje de maquina en la que se cuenta con un información para

casos específicos conocidos, y se busca a través de esta información inicial formular un conjunto de reglas que permita cubrir y dar respuesta también a casos desconocidos (Cassis, 2015).

Según Pfeifer (Pfeifer, 2009), este término puede significar dos cosas ligeramente diferentes en la literatura:

- El entrenamiento tiene lugar mientras que un jugador humano está jugando el juego. Un tipo de retroalimentación es dada directamente por el jugador, que se utiliza para realizar ajustes en la inteligencia artificial del juego mientras se está jugando.
- El entrenamiento que utiliza datos de entrada los cuales son asociados con los datos de salida que representan resultados correctos. El usar un algoritmo de entrenamiento para inicializar la red neuronal entra en esta categoría, ya que se utiliza pares de estado/acción predefinidos.

Para el presente proyecto se usó la segunda forma del aprendizaje supervisado para el entrenamiento de la red neuronal inicial, y la primera forma para el ciclo de mejora de la red neuronal.

- Conclusiones al marco conceptual

Como se puede observar, los conceptos relacionados con el presente trabajo son muy variados, desde tipos de inteligencia y métodos de aprendizaje, hasta modelos para tener en cuenta para desarrollar una aplicación y algoritmo adaptativo. Todos estos conceptos son necesarios para entender a profundidad el problema presentado y así poder tener una mayor consciencia del alcance del mismo. Entender a profundidad estos conceptos, y otros más no incluidos por ser ya conocidos en el ambiente académico, será de vital importancia al momento de proponer la alternativa de solución.

3. ESTADO DEL ARTE

En la presente sección se revisan los esfuerzos que se han realizado para resolver el problema planteado, enfocándose principalmente en las soluciones sobre aprendizaje adaptativo existentes.

3.1. Metodología de revisión

Se realizó una revisión simple de la literatura existente guiada por los objetivos descritos a continuación. De los resultados encontrados se exponen los de mayor aporte.

3.2. Objetivos de la revisión del estado del arte

Los objetivos planteados para realizar esta revisión son los siguientes:

- Conocer que conceptos involucra para la Inteligencia espacias en el uso de bloques lógicos.
- Conocer que conceptos involucra el aprendizaje adaptativo y el balance dinámico de la dificultad.
- Conocer las estructuras que han sido propuestas para un sistema que involucre balance dinámico de la dificultad.
- Conocer los algoritmos que se han hecho posible la implementación de balance dinámico de la dificultad en otros sistemas.
- Conocer las métricas usadas para medir las habilidades en cuanto a inteligencia espacial.
- Conocer los conceptos que involucra una aplicación de tipo educativo.

3.3. Documentos académicos

Los siguientes fueron los documentos académicos más representativos encontrados en la revisión acorde a los objetivos establecidos.

3.3.1. Contributions of executive function and spatial skills to preschool mathematics achievement (Verdine, Irwin, Golinkoff, & Hirsh-Pasek, 2014)

Este trabajo investiga la contribución que tiene el desarrollo de la función ejecutiva y de las habilidades espaciales en ayudar al estudiante a alcanzar los logros propuestos en el área de matemáticas en la etapa preescolar. Se realizaron una serie de test sobre un conjunto de 44 estudiantes a la edad de 3 años para evaluar su estado en cuanto a la función ejecutiva (FE, habilidades cognitivas elevadas usadas para planear, procesar información y resolver problemas orientados al alcance de metas en situaciones novedosas) y a las habilidades espaciales. Luego

se realizó otra serie de test a los cuatro años sobre los mismos alumnos. Los resultados del test indicaron que el estado de la FE y las habilidades espaciales pueden predecir en un 70% el desempeño matemático en años posteriores.

Para la medición de las habilidades espaciales se realizó un test 2D y uno 3D. Ambos test se presentaron de modo idéntico con la diferencia de la naturaleza de las piezas entregadas a los niños y la dimensión a trabajar. En el caso del test 2D, se le presentó a los estudiantes una serie de figuras, las cuales ellos debían imitar con las fichas que les serían entregadas. La orden dada para resolver el test fue "has que tus fichas luzcan igual que la figura". En la Ilustración 5 se presentan las figuras que los estudiantes debían imitar.

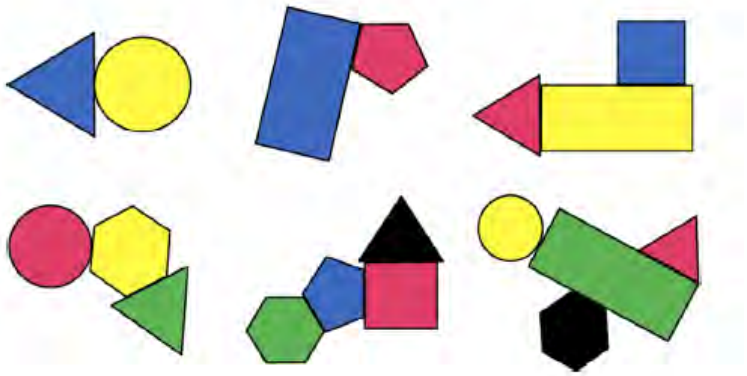

Ilustración 5: Modelos a imitar en el test 2D. Recuperado del documento citado.

Para este test, cada uno de las fichas fue evaluada en cada uno de las tres siguientes dimensiones:

1. Piezas adyacentes: Si la pieza fue colocada al costado de su vecina correspondiente, (dentro de 1 centímetro) 1 punto era otorgado.
2. Dirección horizontal y vertical: Se evaluó si la pieza estaba correctamente arriba, abajo, derecha o izquierda de su pieza vecina. Si al menos el 50% de la pieza estaba dentro del cuadrante que mostraba la pieza correcta se otorga 1 punto.
3. Posición relativa: Usando una transparencia se verificó si la ficha se encontraba al menos 50% dentro de la posición relativa correcta, otorgando 1 punto en caso fuera cierto.

La investigación concluyó que debido a que las habilidades espaciales pueden ser alteradas (Uttal, 2013), y a que la data del experimento muestra una relación entre las habilidades espaciales y matemáticas, sería posible mejorar las habilidades matemáticas mejorando las habilidades espaciales y geométricas. Estas

conclusiones reciben apoyo de estudios que muestran mejoras matemáticas en colegios iniciales debido a intervenciones usando actividades espaciales (Grissmer, 2013).

3.3.2. Creating Adaptive Game AI in a Real Time Continuous Environment using Neural Networks (Pfeifer, 2009)

En este artículo se presenta un modelo de cómo realizar inteligencias artificiales adaptativas para juegos que permitan adaptarse al usuario en tiempo real. Usa principalmente técnicas de aprendizaje por refuerzo y aprendizaje supervisado en redes neuronales, y establece una metodología a seguir que permita realizar una inteligencia artificial que realice efectivamente su labor. Trabaja en base a agentes, elementos controlados por computadora a los que se enfrentará el usuario. Estos alimentan las redes neuronales con sus estados actuales y reciben de ella en comportamiento que deben seguir para adaptarse al comportamiento del usuario.

3.3.3. The effectiveness of adaptive difficulty adjustments on student's motivation and learning (Sampayo-Vargas, Cope, He, & Byrne, 2013)

En este artículo se especifican los detalles y los requerimientos necesarios en el desarrollo de la parte experimental y de validación de resultados de un proyecto similar. En este proyecto se trabaja con *Adaptive Difficulty*, un concepto que implica adaptar la dificultad del juego al jugador teniendo en cuenta el rendimiento de este en los problemas que se le presenta. Usa tres modelos de juego, dos modelos virtuales idénticos (es decir, que se evalúan mediante una aplicación interactiva) que solo se diferencian en el manejo de la dificultad (uno es adaptativo y el otro incremental) y el otro modelo por escrito (sin aplicación interactiva). Los resultados estadísticos muestran que no hay mucha diferencia en el componente "motivacional" de los juegos, pero si en el componente de "aprendizaje", es cual es significativamente mayor en el modelo adaptativo.

3.3.4. Challenge-Sensitive Action Selection, an Application to Game Balancing (Andrade, Ramalho, & Santana, 2005)

Este artículo describe detalladamente el tema *Dynamic game difficulty balacing*. Para esto aplica el método *Challenge-sensitive action selection*, el cual divide el aprendizaje en dos partes. Primero, crear una inteligencia artificial fuera de línea con aprendizaje por refuerzo tradicional, de modo que se puedan aprender las acciones optimas ante determinadas situaciones y mapear sus valores. Aquí también se incluye el mapear las soluciones sub-optimas. Segundo, realizar la adaptabilidad de la dificultad en forma online mediante una función que selecciona la acción mínima necesaria de las aprendidas anteriormente (optima o sub-optima)

para igualar el desempeño del usuario.

3.3.5. Generación de Juegos Educativos Adaptativos (Carro, Breda, & Castillo, 2002, Mayo)

Este artículo habla sobre las características que se deben tener en cuenta al desarrollar un juego educativo adaptativo, poniendo especial detalle en los requerimientos en los usuarios, en las actividades que el juego desea dar a conocer y en el juego informático mismo. También describe la metodología necesaria a seguir para llevar a cabo esta tarea.

3.3.6. AI for Dynamic Difficulty Adjustment in Games (Hunicke & Chapman, 2004)

En este artículo se propone ajustar la dificultad del juego estableciendo un mapa de transición de los estados de un jugador, para de este modo poder determinar en qué momentos intervenir cambiando las variables del entorno. Se pone especial atención en que cada posible modificación tiene un costo, y se definen políticas a usar estas modificaciones dependiendo de la experiencia que demuestra el jugador para poder adecuarse adecuadamente a él.

3.3.7. An application of adaptive games-based learning based on learning style to teach SQL (Soflano, Connolly, & Hainey, 2015)

Este artículo desarrollado por estudiantes de la University of the West of Scotland discute el término adaptabilidad en un contexto Game-based Learning (aprendizaje basado en el juego) y presenta los resultados de un estudio experimental que investiga las diferencias en la eficacia del aprendizaje de los diferentes modos de juego en comparación a un aprendizaje basado en papel. El estudio se realizó con 120 estudiantes de Educación Superior aprender el lenguaje SQL de base de datos.

El proyecto desarrolla tres modos de juego y busca comparar los resultados:

- Un modo no-adaptativo del juego. Este modo trata a todos los estudiantes de la misma manera y no tiene en cuenta el estilo de aprendizaje del estudiante.
- Un modo de adaptación fuera del juego. Las características de un estudiante se identifican por medio de un cuestionario que lo evalúa de acuerdo a los estilos de aprendizaje Felder-Silverman, completado antes de usar el juego y a continuación la jugabilidad es personalizada de acuerdo con el estilo de aprendizaje del estudiante.
- Un modo de adaptación dentro del juego. En este modo, las características de los estudiantes se identifican durante el juego. Cómo es posible que el estudiante cambie el estilo de aprendizaje en el transcurso del juego, el juego tiene un sistema adaptativo que puede personalizar de forma automática el

juego en tiempo real, de acuerdo con el estilo de aprendizaje actual del estudiante. El estilo de aprendizaje actual del estudiante se identifica mediante el análisis de las interacciones históricas entre el estudiante y el juego. En la Ilustración 6 se puede ver una captura de pantalla de la aplicación.

Ilustración 6: Captura de pantalla de la aplicación. Recuperado de (Soflano, Connolly, & Hainey, 2015).

Los resultados muestran que el juego desarrollado, sin importar el modo, produce mejores resultados de aprendizaje que los que aprendieron de un libro de texto, mientras que el modo adaptativo era mejor en términos de permitir a los estudiantes completar las tareas más rápido que las otras dos versiones del juego.

3.3.8. Development of an Adaptive Learning System with Multiple Perspectives based on Students' Learning Styles and Cognitive Styles (Yang, Hwang, & Jen-Hwa , 2012)

En este estudio de la Universidad Nacional de Ciencia y Tecnología de Taiwán se desarrolló un sistema de aprendizaje adaptativo tomando en cuenta múltiples dimensiones de características personales. Se propuso un módulo de presentación personalizada basado en el modelo de estilo cognitivo dependiente/independiente y las cuatro dimensiones del estilo de aprendizaje de Felder-Silverman. Se llevó a cabo un experimento para evaluar el rendimiento del enfoque propuesto en un curso de la informática. Cincuenta y cuatro participantes fueron asignados al azar a un grupo experimental que aprendió con un sistema de aprendizaje adaptativo desarrollado basado en el módulo de presentación personalizada, y un grupo de control que aprendió con el sistema de aprendizaje convencional sin presentación personalizada.

Este trabajo usa el concepto de módulo generador de contenido, el cual es usado para extraer contenido de los materiales de enseñanza en bruto y generar “trozos” de información para componer materiales personalizados basados en la forma de

presentación desarrollada para un usuario en particular. Cada unidad de una materia a enseñar contiene una serie de componente, los cuales pueden clasificarse entre identificador de unidad, texto, ejemplo, figura, contenido fundamental, contenido suplementario. El sistema organiza los componentes y seleccionar los apropiados para cada presentación particular. La Ilustración 7 representa el módulo generador de contenido.

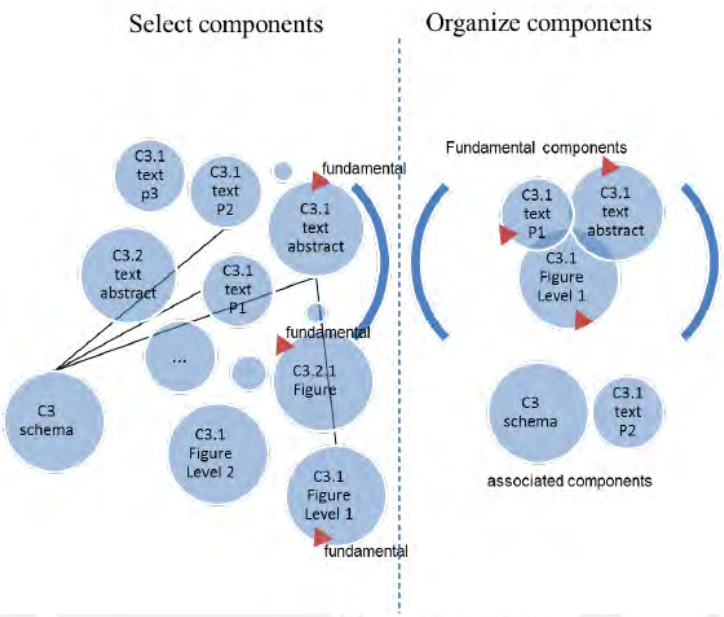

Ilustración 7: Módulo generador de contenido. Recuperado de (Yang, Hwang, & Jen-Hwa , 2012).

Los resultados generados dan como resultado la personalización multidimensional de material educativo y su presentación, lo que pretende ofrecer al estudiante mayor eficiencia al momento de aprender los conceptos. Estos son mostrados a través del módulo de aprendizaje, como se puede observar en la Ilustración 8.

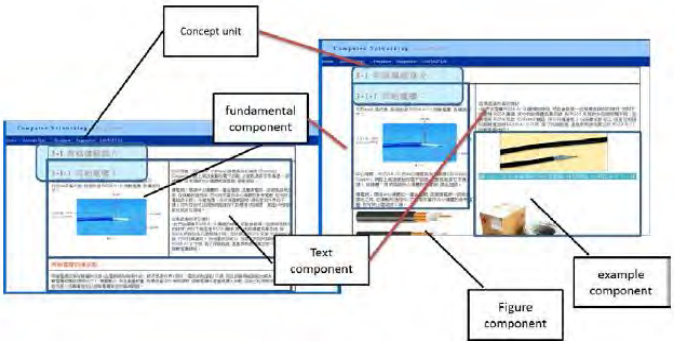

Ilustración 8: Módulos de aprendizaje para estudiantes campo-dependientes con estilos de aprendizaje diferentes. Recuperado de (Yang, Hwang, & Jen-Hwa , 2012).

3.4. Juegos y aplicaciones comerciales

Los siguientes son juegos y aplicaciones relacionadas con los objetivos planteados en el presente proyecto.

3.4.1. DreamBox (DreamBox, 2015)

Es una plataforma educativa especializada en la enseñanza de matemática, enfocada en alumnos de escuelas primarias y primeros grados de secundaria. Entre sus características se encuentran las de adaptarse al alumno individualmente, indicando la siguiente lección adecuada en el momento adecuado, usando métodos de aprendizaje adaptativo. DreamBox analiza cerca de 48,000 datos por estudiante a medida que este usa la aplicación para así ofrecerle la forma más eficaz de aprendizaje.

Mediante este método, DreamBox busca que los estudiantes adquieren la comprensión conceptual y habilidades necesarias para resolver problemas del mundo real, ya sea que necesiten intervención para alcanzar las expectativas o enriquecimiento de sus habilidades. La aplicación se asegura que cada concepto es entendido antes de continuar con la siguiente lección. Adicional a esto, se pueden generar reportes y datos que permitan a los maestros ayudar a los estudiantes a alcanzar su potencial. Un ejemplo de la pantalla de esta aplicación está en la Ilustración 9.

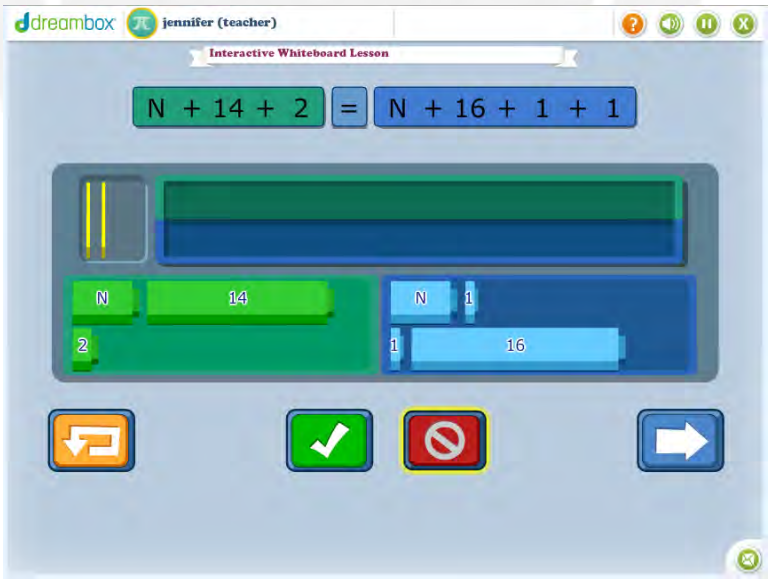

Ilustración 9: Ejemplo de la aplicación DreamBox. Obtenido en la página web https://play.dreambox.com

En la página web de esta compañía se pueden apreciar numerosos casos de estudio que indican que se han obtenido excelentes resultados con la aplicación de esta tecnología en diversas instituciones educativas. Estos resultados parecen

indicar el éxito de esta aplicación, por lo que se puede considerar una de las más eficaces en el uso del aprendizaje adaptativo.

3.4.2. Knewton Adaptive Learning Platform (Knewton, 2014)

Knewton es una plataforma estructurada que permite a sus usuarios construir aplicaciones que usan aprendizaje adaptativo. Combina y consolida datos estadísticos y psicométricos de todos sus usuarios para permitir personalización a escala masiva. Además de esto, provee análisis a nivel de conceptos, seguimiento del avance y mejora de los alumnos, entre otras características. Mediante técnicas de *machine-learning*, esta aplicación busca determinar en tiempo real las fortalezas y debilidades de los alumnos, así como sus patrones de aprendizaje. La compañía fue fundada en 2008 y actualmente se encuentra en 21 países alrededor del mundo.

La razón principal por la que Knewton se dice diferenciar de otras aplicaciones que usan aprendizaje adaptativo es que no trabaja con un solo punto de adaptabilidad, sino que busca "capturar" datos de la competencia del alumno a nivel de conceptos. De este modo se busca estimar no solo lo que el estudiante "hizo", sino lo que el estudiante "sabe" y a un nivel granular. De este modo se puede saber cuán preparado está un estudiante para futuras instrucciones y estimar cómo evolucionan sus habilidades con el tiempo. La Ilustración 10 muestra ejemplos de los caminos de aprendizaje personalizados.

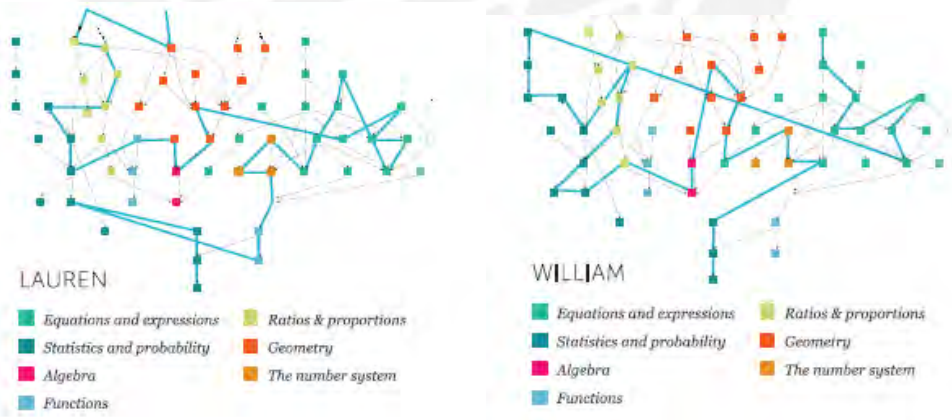

Ilustración 10: Ejemplos de caminos de aprendizaje personalizados. Recuperado de (Knewton, 2014).

3.4.3. Mario Kart

Mario Kart es básicamente un juego de carreras donde los jugadores pueden obtener objetos en forma aleatoria que les permiten mejorar sus características (mayor velocidad) o retrasar a otros jugadores (atacándolos de algún modo). Este es uno de los juegos que más aparece como ejemplo en los artículos relacionados

al balance dinámico de la dificultad debido a que el sistema que presenta para realizar esta tarea es fácilmente identificable, así como explotable para los jugadores. Este sistema es conocido como *rubber banding* o banda elástica y significa que el juego buscará restringir al máximo a distancia existente entre el jugador en primer lugar y el jugador en el último. Para esto el juego se vale de algunas modificaciones tanto a los jugadores CPU como a las características del escenario. Si un jugador está en primer o último lugar, por ejemplo, el juego incrementará o reducirá, respectivamente, la velocidad de los oponentes CPU para acercarse al jugador. Del mismo modo, el juego cambiara las probabilidades de obtener objetos dependiendo de posición del jugador, dando mejores objetos a los jugadores que se encuentran al final, y los más sencillos a los jugadores que están encabezando la carrera (Saltsman, 2009).

El sistema tiene una ventaja significativa: el juego permite ganar casi a cualquier jugador en cualquier punto de la carrera, igualando las "probabilidades de ganar" con los demás jugadores. Es además fácil de implementar y probar. El problema radica en que el sistema es fácilmente explotable, haciendo sencillo para cualquier jugador generar una estrategia que le permita ganar "aprovechando el sistema". Por ejemplo, el jugador que se encuentre en el primer lugar al comenzar la última vuelta, tendrá muchísimas probabilidades de ser atacado por los objetos que el sistema de a los demás jugadores, perdiendo debido a esto el primer lugar. Una estrategia que aprovecha el sistema sería quedarse en el segundo lugar al cruzar la segunda vuelta y esperar a que el sistema "lo haga ganar a uno" (Saltsman, 2009). La Ilustración 11 ejemplifica este conocido efecto.

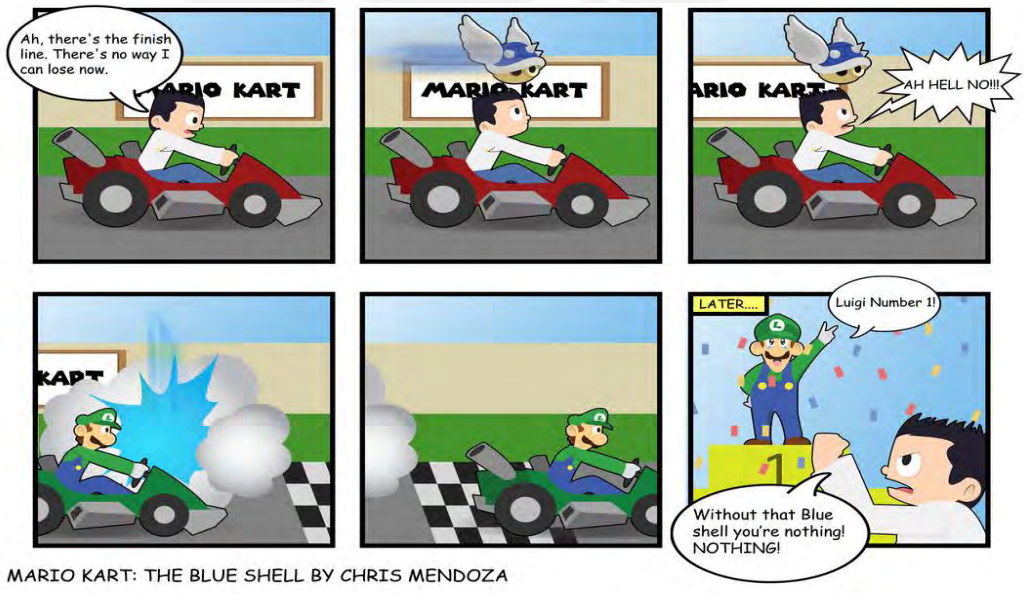

Ilustración 11: Caricatura que ejemplifica los efectos no deseados del sistema "banda elástica". Recuperado de (Mendoza)

3.4.4. Flow (Chen, Flow in games, 2006)

Este juego es resultado de la tesis de Jenova Chen llamada “Flow in games”, en donde se establece una metodología de diseño de juegos para realizar el ajuste dinámico de la dificultad (ADD), centrado en el usuario; el juego fue creado para poner a prueba este concepto. El juego está diseñado para ser extremadamente minimalista, de modo que se pueda evaluar la eficiencia del sistema de ADD. Consiste en usar el mouse para hacer mover un organismo en una biosfera donde consume otros organismos, evoluciona y avanza cada vez a niveles más profundos. En la Ilustración 12 se observa una captura de pantalla de este juego.

Ilustración 12: Captura de pantalla del juego Flow. Recuperado de (Chen, Flow in Games)

El juego está dividido en 20 niveles, donde cada nivel introduce nuevas criaturas y desafíos. Ofrece a los jugadores puedes escoger avanzar o retroceder a voluntad entre los niveles dependiendo del organismo que consuman, de este modo ellos pueden decidir enfrentar el desafío o evadirlo y regresar después. Al morir el jugador en un nivel, este es regresado al nivel anterior, es cual es relativamente más sencillo. Las opciones que se le brindan al jugador están diseñadas para que cree una experiencia de juego basada en su propio comportamiento, de modo que es él mismo quien dirige inconscientemente la adaptabilidad. Esto está orientado tanto para jugadores novatos como para experimentados (Chen, Flow in games, 2006). Las criticas indican un éxito en este sistema, manteniendo al jugador concentrado en el juego y con un reto constante.

3.4.5. God Hand

Este es, de acuerdo a los comentarios encontrados, uno de los juegos que más exitosamente implementa el concepto de ajuste dinámico de la dificultad. Se trata de un juego del genero *Beat'em up,* donde básicamente el jugador se enfrenta cuerpo a cuerpo contra hordas de enemigos a través de diferentes escenarios. Lo que hace el sistema de ADD de este juego es seguir el desempeño del jugador y, a

medida que derrota más enemigos, va incrementando el nivel de dificultad desde el nivel 1 hasta el nivel 4. El éxito de este sistema reside en que es completamente visible para el jugador y fue diseñado para que sea parte de la estrategia del mismo. La pantalla del juego presenta un indicador de en qué nivel de dificultad se encuentra el jugador y un indicador de cómo se encuentra su progreso para pasar al siguiente nivel. El jugador puede ver cómo, a medida que elimina más enemigos, el indicador de progreso aumenta, hasta que al llenarse el nivel de dificultad aumenta y el indicador de progreso se reinicia. En la Ilustración 13 Ilustración 13se puede observar el contador de dificultad mencionado.

Ilustración 13: Captura de pantalla del juego God Hand donde se aprecia el contador de dificultad (esquina inferior izquierda) Recuperado de (Kuomon, 2012)

Una mayor dificultad significa que los enemigos usarán mejores estrategias y tendrán un poder de ataque mayor. El juego además motiva al usuario a buscar un mayor nivel de dificultad entregando mayores recompensas al derrotar a los enemigos en estos niveles. Estas recompensas en forma de monedas son usadas luego para que el jugador aprenda nuevos movimientos que le permitan mejorar las habilidades de combate de su personaje (Saltsman, 2009).

De este modo, el jugador sabe en todo momento a que se enfrenta y como cambiará el escenario en respuesta a su comportamiento; y es motivado a enfrentar cada vez mayores desafíos como parte del juego. Los jugadores novatos pueden enfrentar el juego consientes del reto que enfrentan, mientras que, sin necesidad de realizar cambio alguno en la configuración del juego, los jugadores experimentados enfrentarán cada vez un mayor desafío de acuerdo a sus habilidades. De este modo el jugador puede usar el “contador” de dificultad como parte de su estrategia de juego, sin experimentar la sensación de que el juego “lo ha traicionado” al variar

la dificultad sin informarle.

3.5. Conclusiones sobre el estado del arte

Como podemos observar, los de balance dinámico de la dificultad y aprendizaje adaptativo se conceptos encuentran en un periodo de expansión y explotación. Los trabajos realizados motivan a seguir explorando este campo al presentar resultados positivos, sobre todo aquellos que concluyen lo significativo que es el concepto de "adaptación" para la mejora del aprendizaje. Los trabajos relacionados con balance dinámico nos informan de retos importantes en la elaboración de estos sistemas, pero también nos demuestran la factibilidad de los mismos.

En cuanto a las aplicaciones comerciales podemos observar como el sector educativo está empezando a sacar provecho del aprendizaje adaptativo, integrando a plataformas y aplicaciones educativas elementos que permitan la adaptación de sus contenidos para generar propuestas personalizadas a los estudiantes. Podemos resaltar el notorio éxito de las herramientas arriba mencionadas, lo que motiva a continuar desarrollando aplicaciones similares. También podemos observar la doble naturaleza del balance dinámico de la dificultad al aplicarlo a un entorno gamificado. Por un lado, se logra con éxito el objetivo de balancear la dificultad, entregando a jugadores con diferentes niveles de habilidad la oportunidad de disfrutar y engancharse con el juego, permitiendo con esto un desarrollo progresivo de sus habilidades. Por otro lado, vemos también que pueden existir efectos no deseados como que el jugador se sienta "engañado" por el juego al percibir un cambio drástico en la dificultad. Vemos también que esto no depende del sistema de ADD en sí, sino de cuan adecuadamente se aplique el concepto de balance y como este es presentado al usuario. Estos resultados servirán para el desarrollo del proyecto y son una motivación para continuar con la investigación.

4. CONCEPTOS A EVALUAR Y MÉTRICAS

En esta parte se muestra como se realizó el proceso de identificación y definición de los conceptos evaluados, así como las métricas que permitirán la adaptabilidad.

4.1. Definición de conceptos a evaluar

El alcance del proyecto define presentar al usuario un conjunto de bloques lógicos formando una figura definida y permitir al usuario responder imitando dicha figura manipulando otro conjunto de bloques lógicos a su disposición. A cada actividad que el usuario debe resolver se le denominó “reto”; y cada reto presenta un conjunto de características específicas que la aplicación se debe controlar. La Ilustración 14 permite comprender mejor este punto.

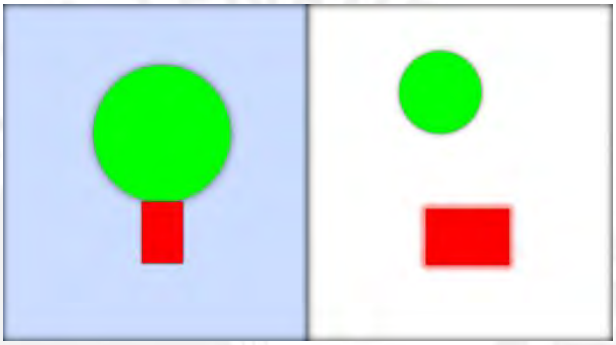

Ilustración 14: Ejemplo del reto que se le pide al usuario resolver. Elaboración propia.

Teniendo en cuenta esto, el primer objetivo del Proyecto fue definir los conceptos relacionados a inteligencia espacial y percepción visual que serían evaluados en la aplicación. Esta selección se realizó con la colaboración de una licenciada en psicología (Aedo, Segunda entrevista, 2015) y una licenciada en educación (Torres, 2015).

Para realizar esta selección se establecieron primero algunos criterios importantes producto de la colaboración realizada:

- Los conceptos deben ser relevantes en cuando a poner en uso la inteligencia espacial y percepción visual del usuario.
- Los conceptos deben estar relacionados con el tipo de juego que se desea presentar al usuario, en este caso, relacionado a los bloques lógicos.
- Los conceptos deben ser fácilmente identificables por los usuarios.
- El nivel de dificultad que cada uno de los conceptos admita debe estar dentro de la edad considerada para los usuarios.
- Los conceptos seleccionados deben poder ser representados de forma práctica por la aplicación.

- Los conceptos deben poder ser medidos por la aplicación.
- Se debe considerar un aproximado de la dificultad de cada concepto a evaluar de modo que se seleccione aquellos que sea factible desarrollar dentro del tiempo planificado del proyecto.

Teniendo en cuenta estos criterios, las habilidades que se consideraron el estudiante pondría en práctica son las siguientes, cada una de estas habilidades se encuentra definida en el punto 2.2.4 de este documento:

- Reconocimiento de figuras
- Traslación de figuras
- Rotación de figuras
- Escalamiento de figuras
- Percepción de color
- Percepción de textura
- Ordenamiento de figuras
- Percepción de simetría

La aplicación, sin embargo, no puede realizar un juicio objetivo sobre el nivel de la habilidad de usuario en cada uno de estos conceptos. Por ende, se seleccionó los siguientes conceptos que podrán ser evaluados directamente sobre la respuesta que el usuario de a la figura mostrada. Estos conceptos son:

- Distribución de los bloques usados
- Ubicación de los bloques usados
- Angulo de rotación de los bloques usados
- Tamaño de los bloques usados
- Textura de los bloques usados
- Orden de los bloques usados (atrás-delante de otro)

Cada uno de estos conceptos, sin embargo, se debió aterrizar en forma de métricas que puedan ser medidas por la aplicación, lo cual se explica en el siguiente apartado.

4.2. Requerimientos del módulo de adaptabilidad

El módulo de adaptabilidad será el responsable de adaptar la dificultad de los atributos del juego en respuesta al desempeño del usuario. Para poder realizar esta adaptación en los diferentes conceptos evaluados es necesario conocer el nivel de la habilidad del jugador en cada uno de esos conceptos. Esto se logra a través del establecimiento de métricas que permitan medir cuantitativamente el estado de estas habilidades. El módulo de adaptabilidad entonces debe recibir estas métricas

y en base a ellas dar como respuesta información que permita modificar los atributos del juego, de modo que el siguiente escenario que se le presente al usuario este mejor adaptado a sus habilidades. En la Ilustración 15 se diagrama este proceso.

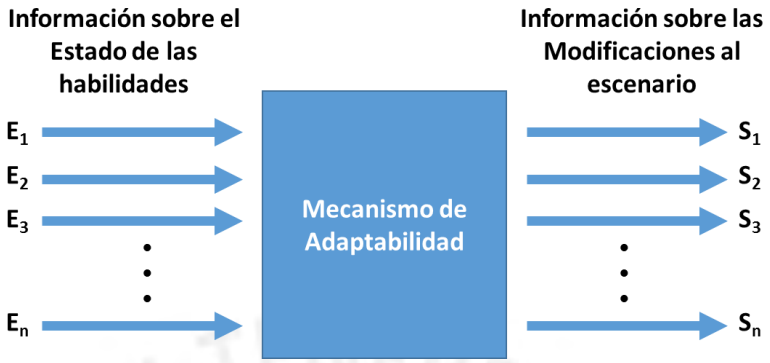

Ilustración 15: Entradas y salidas del Módulo de Adaptabilidad

En esta primera parte del desarrollo de la aplicación, el módulo de adaptabilidad se considera una "caja negra", de modo que su funcionamiento interno sea independiente del resto del proyecto. Sin embargo, es necesario definir de forma detallada las entradas y salidas con las que trabajará este módulo, de modo que no se presenten problemas de incompatibilidad.

4.3. Métricas de los conceptos a evaluar

El punto central del Proyecto es adaptar el reto que se le presenta al usuario al nivel particular de habilidades que presente durante la interacción con la aplicación. El Proyecto usará redes neuronales para adaptar las características del reto en función a las habilidades del usuario.

4.3.1. Tipos de métricas

Se resolvió que para adaptar el reto presentado a las habilidades del usuario es necesario definir dos conjuntos de métricas:

- Métricas de entrada: Son las métricas que permiten identificar las habilidades del usuario relacionadas a los conceptos de inteligencia espacial y percepción visual anteriormente definidos. Se les denominó de entrada porque se las usará luego como valores de entrada en el sistema de redes neuronales. Responden a la pregunta "¿Cuáles son las características que se pueden medir en la respuesta o interacción de los usuarios?"
- Métricas de salida: Son las métricas que permitirán identificar la dificultad y las diferentes características que presenta un reto en particular. Estas métricas están relacionadas a las características de la imagen a imitar o de las opciones

y restricciones que se le brinde al usuario para resolver el reto. Se les denominó de salida porque están relacionadas con los valores se espera recibir del sistema de redes neuronales luego del procesamiento. Responden a la pregunta "¿Cuáles son las características del juego que se pueden cambiar?"

4.3.2. El problema de la relación entre métricas

Parte del problema que el Proyecto busca resolver es el de la complejidad existente en realizar una relación exacta entre una métrica de entrada y una métrica de salida en particular. La complejidad del sistema de métricas reside en que una métrica de entrada puede afectar de manera simultánea múltiples métricas de salida, mientras que una métrica de salida puede verse afectada por múltiples métricas de entrada. Este problema será manejado a través del sistema de redes neuronales explicado más adelante.

4.3.3. Métricas definidas

Aunque definir una relación exacta entre las métricas resulta compleja, se decidió analizar los conceptos anteriormente definidos y realizar una búsqueda subjetiva de las posibles métricas existentes, de modo que se pueda contar con un conjunto inicial de métricas de las cuales se seleccionara las más adecuadas. En el Anexo 2, se puede observar las posibles métricas definidas para cada concepto.

Entre las métricas posibles se puede observar que varias métricas se presentan iguales en diferentes conceptos. Si se tiene en cuenta que esta selección es preliminar y engloba un alto grado de subjetividad, se vuelve más importante que el módulo de adaptabilidad del reto no clasifique las métricas, sino que trabaje con todas juntas y calcule internamente como debe influir cada una de ellas en el resultado. También se puede observar que para las métricas de salida se considera tanto aquellas métricas inherentes al grafico mostrado como aquellas relacionadas con características y limitaciones que establece la aplicación. De igual modo para las métricas de entrada se consideran aquellas que se obtienen de la figura/respuesta final del usuario como de las características mostradas por el durante la interacción.

En base a esta lista preliminar, se seleccionó un grupo de métricas con las que se trabajó en el Proyecto. Esto se realizó porque reducir las métricas a un número menor permite trabajar con mayor precisión y una mejor observación y control de los resultados. Para realizar esta selección de métricas definitivas se establecieron los siguientes criterios que buscan ayudar al logró de los objetivos del proyecto:

- Deben ser claras de entender y observar a simple vista.

- Se debe tener métricas representativas de cada uno de los conceptos definidos.
- Se deben considerar tiempos estimados de implementación que permitan trabajar dentro del tiempo establecido para el proyecto.
- Se debe tener en cuenta la posible interfaz gráfica al momento de la selección, ya que resulta importante considerar las posibilidades y limitaciones que presentará la aplicación para su medición.

Como resultado de la selección se tuvieron las siguientes métricas definitivas. Sus valores están normalizados, de modo que su rango este entre 0 y 1.

Métricas de entrada

El detalle de estas métricas se aprecia en la Tabla 3.

Tabla 3: Métricas de entrada basadas en la respuesta del usuario

N°	Métrica	Nombre en código	Descripción
1	Ratio de encaje	MatchingRate	Indica la razón de bloques que están en una posición acertada en relación al total
2	Ratio de desfase de ubicación	LocationGapRate	Indica la razón de la suma de los desfases de los bloques acertados respecto a su posición correcta en relación al total
3	Ratio de forma correcta	CorrectShapeRate	Indica la razón de bloques que poseen la forma correcta en relación al total.
4	Ratio de tamaño correcto	CorrectSizeRate	Indica la razón de bloques que poseen el tamaño correcto en relación al total.
5	Ratio de color correcto	CorrectColorRate	Indica la razón de bloques que poseen e color correcto en relación al total.
6	Ratio de rotación correcta	CorrectRotationRate	Indica la razón de bloques que poseen la rotación correcta en relación al total.
7	Ratio de textura correcta	CorrectTextureRate	Indica la razón de bloques que poseen la textura correcta en relación al total.

Métricas de salida

El detalle de estas métricas se aprecia en la Tabla 4.

Tabla 4: Métricas de salida que indican los cambios en los atributos

N°	Métrica	Nombre en código	Descripción
1	Numero de bloques	NumBlocksN	Variación en el número máximo de bloques en el gráfico, entre el número máximo posible de bloques.
2	Numero de formas	NumShapesN	Variación en el número máximo de formas diferentes en el gráfico, entre el número máximo posible de formas.
3	Máxima complejidad de formas	MaxShapeComplexityN	Variación en la complejidad máxima de las formas, entre la complejidad máxima posible.
4	Suma de todas las	SumNumPossibleRotationsN	Variación en la suma del número

	posibles rotaciones		posible de rotaciones de cada bloque, sobre el máximo posible de rotaciones para el máximo de fichas.
5	Numero de rotaciones máximo	MaxNumPossibleRotationsN	Variación en el máximo valor permitido para el numero de rotaciones por bloque, sobre el máximo posible de rotaciones.
6	Numero de tamaños de bloque	NumBlockSizesN	Variación en el número de tamaños existentes en la figura, sobre el número máximo posible de tamaños.
7	Variación en el tamaño de los bloques	VarBlockSizesN	Variación en el número de tamaños existentes en la figura, sobre el número de bloques en la figura.
8	Numero de bloques con misma forma y diferente textura	NumBlockSameShape_ DifferentTextureN	Variación en el número de bloques que son iguales en forma, pero diferentes en textura, sobre la diferencia entre el número de bloque en la figura y el número de formas.
9	Numero de texturas	NumBlockTexturesN	Variación en el número de texturas existentes en la figura, sobre el número máximo posibles de texturas.
10	Numero de colores	NumColorsN	Variación en el número de colores existentes en la figura, sobre el número máximo posibles de texturas.

Respecto a la complejidad de una forma, esta está relacionada con su simetría y el número de lados que presenta (1 para circulo, cuadrado y triangulo, 2 para rectángulo, 3 para hexágono y pentágono, y 4 para semicírculo). Respecto al número de rotaciones, este indica la cantidad de posibles variaciones que se presentan al rotar un bloque (rotación mínima de 45°), descartando aquellas que son iguales (1 para circulo, 2 para cuadrado, 4 para rectángulo y 8 para el resto de formas). Los bloques con mayor número de rotaciones posibles se consideran más difíciles de acertar en cuanto a su rotación. Respecto al tamaño se consideran 5 tamaños posibles, los cuales guardan una proporción determinada (0.75 para diminuto, 1 para pequeño, 1.5 para mediano. 2 para grande y 2.5 para enorme). Cabe resaltar que la forma en que se ha implementado la solución de adaptación del reto permite ingresar nuevas métricas de salida con facilidad, lo que hace más fácil la escalabilidad de métricas.

5. DESCRIPCIÓN DEL JUEGO Y ENTORNO GRÁFICO

El Proyecto busca demostrar que es posible realizar una aplicación con características educativas donde la dificultad se puede adaptar a las habilidades del usuario durante la ejecución. Es importante indicar que el objetivo principal de este Proyecto no es realizar a detalle la aplicación completa, sino sentar las bases para el desarrollo de aplicaciones con capacidades similares y poder realizar un análisis de los principales problemas y las consideraciones a tener en cuenta en este desarrollo. Es por esto que la interfaz gráfica fue desarrollada solo para poner a prueba el concepto de la aplicación y servir como entorno de pruebas para la adaptabilidad, no con la intención de ser una versión final. En este proceso se han encontrado numerosas consideraciones a tener en una aplicación de este tipo, las cuales se detallan es el presente capitulo.

5.1. Características educativas de la aplicación

Para que una aplicación se pueda considerar educativa, se debe tener objetivos de aprendizaje, los cuales son las habilidades que se pretende desarrollar o potenciar mediante el uso de la aplicación (Carro, Breda, & Castillo, 2002, Mayo). En relación al nivel educativo, dentro del alcance, se pueden identificar las siguientes habilidades como objetivos de aprendizaje, los cuales son trabajados por maestros dentro de la etapa pre-escolar (Torres, 2015):

- Traslación de figuras
- Imitación de formas
- Ubicación espacial
- Igualdad-diferencia
- Atención y concentración

Estas actividades están alineadas con las habilidades sobre inteligencia espacial y percepción visual identificadas en el estudiante y los conceptos definidos para trabajar en la aplicación.

5.2. Consideraciones previas

A continuación, se consideran algunas características importantes a tener en cuenta en la realización de la secuencia de juego.

Límite de tiempo

Como se menciona en Sampayo-Vargas, Cope, He, & Byrne (2013), el tiempo tomado para resolver el problema presentado puede ser un indicador importante de

la habilidad del jugador. Sin embargo, también se puede presentar un tiempo máximo de juego con el objetivo de limitar este recurso y agregar un componente de riesgo y emoción que motive al jugador. Para el Proyecto se eligió el segundo caso, ya que se están midiendo múltiples conceptos en el usuario y existen otros indicadores más significativos que el tiempo. La primera opción posibilitaría que un jugador saque buenos resultados a costa de un muy largo tiempo de juego. Poner a todos los jugadores ante un mismo límite de tiempo permite evitar esta circunstancia.

Necesidad de guía

Debido al carácter experimental de la aplicación a desarrollar, ya que incluirá el componente adaptativo no común en aplicaciones para pre-escolar, y también debido a que la aplicación presentará al maestro mediciones sobre la respuesta del alumno y busca ser un apoyo en la evaluación que el hará sobre el mismo; y considerando la edad del usuario, la aplicación a desarrollar está pensada a usarse en compañía del maestro. Es por esto que las interfaces presentan componentes que requieren una ligera guía en su uso que el maestro debe dar al alumno. Sin embargo, se demostró al finalizar el proyecto, que aun los alumnos de menor edad pudieron manejar el aplicativo con solo una ayuda la primera vez y luego casi ninguna intervención.

5.3. Secuencia de juego

Para entender mejor el proceso de interacción que se realiza entre la aplicación y el usuario se ha realizado el flujo del proceso, el cual se puede observar en la Ilustración 16 a través del diagrama de actividades. La definición detallada de algunos de elementos mencionados como sesión, ronda y escenario se encuentra en la sección 5.5.1.

A continuación, se explica el detalle de algunas actividades:

- **Interactuar con ronda**: Aquí el usuario interactúa con la aplicación. Para esto puede realizar cualquiera de estas actividades:
  - Ingresar o sacar un bloque de la zona de dibujo
  - Mover un bloque
  - Rotar un bloque
  - Escalar un bloque
  - Cambiar el color a un bloque
  - Cambiar la posición (encima-debajo) de un bloque respecto a otro.
  - Dar por terminada la ronda

- **Evaluar solución del usuario**: Aplicación evalúa la figura conformada por los bloques usados por el usuario y determina el valor de las métricas de entrada.
- **Calcular variación en los atributos del escenario**: Si el usuario desea continuar con la siguiente ronda la aplicación en forma interna envía las métricas de entrada al sistema de redes neuronales y obtiene las métricas de salida.
- **Seleccionar nuevo escenario**: La aplicación selecciona un nuevo gráfico y escenario de acuerdo a las métricas de salida obtenidas.

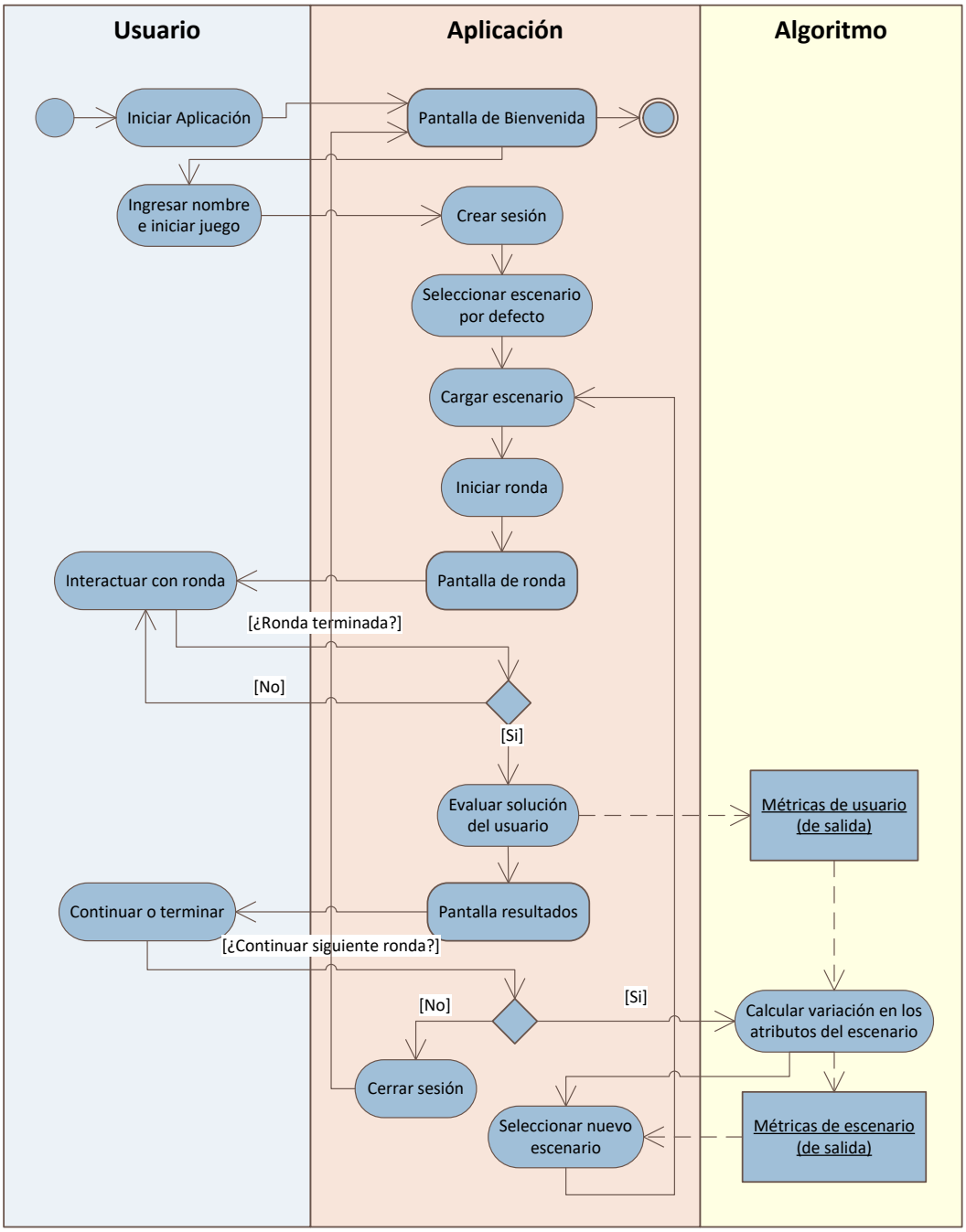

Ilustración 16: Diagrama de actividades, flujo principal de la aplicación. Elaboración propia.

En cada ciclo la aplicación envía las métricas obtenidas al sistema de redes neuronales. De ese modo se logra que la aplicación se adecue al usuario en cada uno de estos ciclos. Esta adecuación tiene dos características. La primera es propia de la adaptabilidad incorporada. La segunda sucede naturalmente en los juegos educativos gamificados gracias al ciclo de juego (Garris, Ahlers, & Driske, 2002):

- La aplicación se adecua a las habilidades actuales del usuario subiendo o bajando el nivel de dificultad en cada uno de los conceptos definidos a través de la manipulación de las métricas de salida.
- En cada interacción el usuario refuerza sus habilidades a través del juego, adquiriendo poco a como las habilidades propias para resolver el reto presentado.

5.4. Diseño de la interfaz grafica

Como se puede observar de los puntos anteriores, la interfaz gráfica necesaria para el juego soporta un proceso principal que es la interacción entre el jugador y la aplicación para resolver el reto que se propone. Las vistas adicionales son para actividades complementarias como el mensaje de bienvenida o la muestra de los resultados de juego. El proceso de desarrollo de la interfaz gráfica se detalla en los siguientes puntos.

5.4.1. Bosquejo de las vistas

Se desarrollaron tres vistas para la interfaz. De estas la principal es la Pantalla de Ronda que es donde el usuario interactuará para resolver el reto planteado. El detalle de cada una de las pantallas se muestra a continuación.

Pantalla de Bienvenida

Esta es la pantalla inicial de la aplicación. Da la bienvenida al usuario y permite que este decida el momento para iniciar el juego. Al presionar el botón comenzar aparece un cuadro de dialogo en el cual se le solicita el nombre al usuario. En la Ilustración 17 y la Ilustración 18 se puede observar capturas de pantallas hechas en la aplicación final.

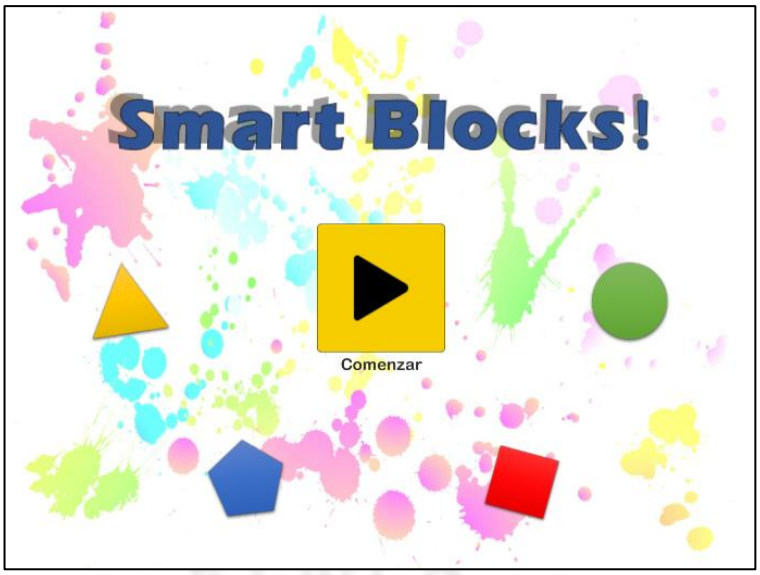

Ilustración 17: Pantalla de Bienvenida. Elaboración propia.

Ilustración 18: Dialogo en la Pantalla de Bienvenida. Elaboración propia.

El botón “¡A jugar!” marca el comienzo de una nueva sesión que contendrá varios ciclos de juego. El nombre ingresado por el usuario, guiado por el maestro, será usado para identificar la sesión y aparecerá más adelante en el título de la pantalla de Ronda.

Pantalla de Ronda

Esta es la pantalla principal. Aquí el usuario interactúa con la aplicación para resolver el reto presentado manipulando los bloques existentes mediante el movimiento de arrastre o usando los controles disponibles.

Las características de la pantalla se han definido de modo que se puedan manipular fácilmente los bloques de acuerdo a lo establecido en la definición de conceptos. En

la Ilustración 19 se puede observar esta pantalla.

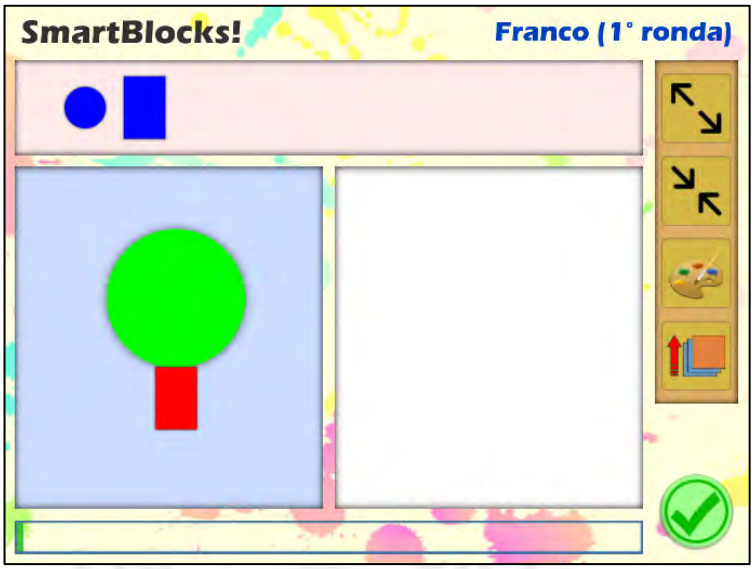

Ilustración 19: Pantalla de Ronda con juego en proceso. Elaboración propia.

En esta pantalla se pueden identificar los siguientes elementos:

- **Título**: Muestra a la izquierda el nombre de la aplicación y a la derecha el nombre del usuario y número de la ronda en la que se encuentra.
- **Tira de bloques no usados (rosa)**: Muestra los bloques que aún no han sido utilizados por el usuario y no forman parte de la figura mostrada por él.
- **Figura a imitar (celeste)**: Es la figura que el usuario debe reproducir con los bloques que se le presentan.
- **Zona de dibujo (blanco)**: Es el área donde el usuario reproduce la figura a imitar.
- **Controles**: Permiten al usuario manipular diversas características de los bloques que se le presenta, permitiéndole realizar la reproducción de la figura.
- **Posicionar bloque**: Permite cambiar la ubicación del bloque arrastrándolo a través de la pantalla. Esto se realiza colocando un dedo sobre el bloque y moviéndolo a la posición deseada.
- **Rotar bloque**: Permite cambiar el ángulo del bloque. Esto se realiza colocando dos dedos a rededor del boque y rotándolo imitando el movimiento en el mundo real.
- **Botones escalar bloque**: Permite aumentar o reducir el tamaño del bloque en función a escalas predefinidas.
- **Botones cambiar color a bloque**: Permite modificar el color actual de bloque por alguno de los permitidos.
- **Botón Subir bloque de capa**: En caso el bloque seleccionado se encuentre

debajo de otro (tenga un bloque superpuesto), permite poner el bloque encima de los demás.

- **Barra de tiempo**: Permite estimar el tiempo restante para completar la ronda actual
- **Botón de finalizar**: Permite dar como terminada la ronda antes de que la barra de tiempo llegue a su límite.

La pantalla permite que los bloques se puedan mover libremente y los controles permiten el cambio de sus características permitiéndole al usuario imitar la figura mostrada. En la Ilustración 20 se puede ver un ejemplo de un bloque donde el color está siendo manipulado.

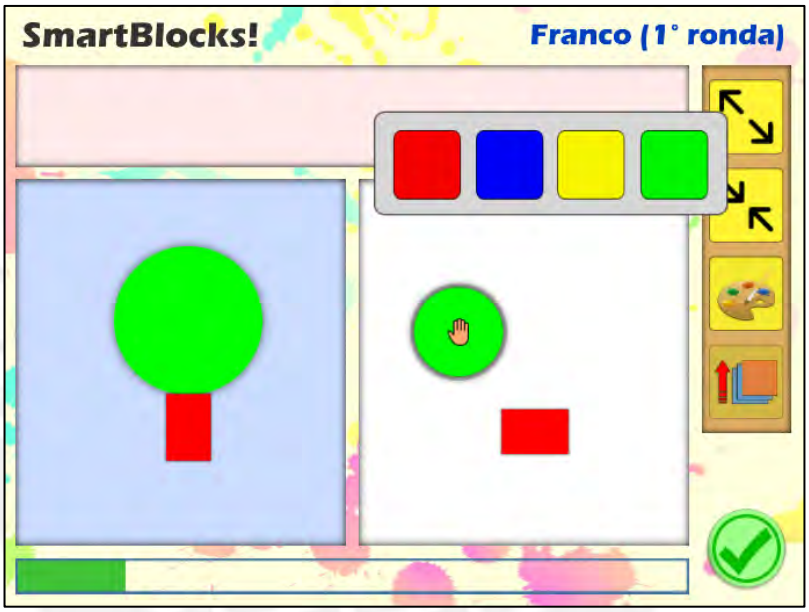

Ilustración 20: Pantalla de Ronda mientras se manipula el color de un bloque. Elaboración propia.

Algunas características adicionales de esta pantalla son:

- Los bloques pueden salir o ingresar de la Zona de bloques no usados y la aplicación se encargará de mantener esta área ordenada.
- Los bloques que forman la Figura a imitar no pueden ser movidos.
- Se puede manipular un bloque a la vez.
- Los bloques pueden superponerse unos sobre otros.
- El botón Finalizar permite terminar el juego antes de que el tiempo límite sea alcanzado.

Al terminar de editar la imagen, se le muestra al usuario un diálogo de confirmación del fin de la ronda que le da la oportunidad de comparar mejor los dos gráficos. En la Ilustración 21 se puede observar este diálogo. Recordar que, como se indica en

el punto 5.2, la aplicación está pensada para ser usada junto con el maestro.

Ilustración 21: Diálogo de confirmación de fin de ronda. Elaboración propia.

Pantalla de Resultados

En esta pantalla se muestran los resultados del juego. Mostrar los resultados tiene como objetivo el que el estudiante reciba cierta retroalimentación que le ayude a mejorar sus resultados. Para el Proyecto se consideró necesario desarrollar dos pantallas de este tipo: una orientada al usuario que le muestre en forma sencilla el nivel de éxito alcanzado, y otra orientada al educador o padre que le permita conocer el éxito alcanzado a detalle en cada concepto evaluado.

La pantalla que visualiza el usuario final presenta los resultados del juego de un modo más adecuado para la edad objetivo. Aquí se resaltan los errores cometidos de modo que el usuario los pueda identificar y corregir en el siguiente ciclo de juego. En la Ilustración 22 se puede observar esta pantalla. Recordar que, como se indica en el punto 5.2, la aplicación está pensada para ser usada junto con el maestro.

Ilustración 22: Pantalla de resultados orientada al estudiante. Elaboración propia.

La pantalla orientada al educador, además de permitir visualizar el éxito alcanzado en cada concepto, también permite evaluar mejor la función de adaptabilidad de la aplicación. En la Ilustración 23 se puede observar un ejemplo de esta pantalla.

Ilustración 23: Pantalla de resultados con resultados por habilidad. Elaboración Propia.

Luego de observar los resultados el usuario puede continuar con la siguiente ronda o cerrar la sesión.

5.5. Desarrollo de la aplicación

En este apartado se describen los diferentes componentes de la aplicación desarrollados y el papel que juegan, así como sus interacciones.

5.5.1. Definición de componentes importantes

El desarrollo de la aplicación requirió de definir componentes adecuados para cada labor. Estos componentes definidos son, en primera instancia, conceptos que engloban características y actividades, pero también tienen una representación física dentro de la aplicación, ya sea como clases programadas o como objetos del entorno de desarrollo Unity.

Para el presente proyecto se decidió separar la lógica de juego de la que maneja el entorno grafico para un mayor orden y control de la aplicación. Además, en proyectos anteriores al actual se observó algunos efectos no deseados al programar en la plataforma Unity, y de este modo se minimiza este riesgo. La plataforma Unity funciona en base a GameObjects, cada uno de los cuales representa un objeto "físico" en la pantalla (sea visible o no). El fondo, los bloques y los botones en la aplicación son todos GameObjects cada uno con su respectiva estructura interna en el motor del juego. El modelo y lógica del juego (rondas, escenarios y bloques activos entre otros) son clases definidas por el usuario, y sin relación directa con el entorno de juego o representación visible en él. El Controlador de Entorno y el Controlador de Juego son los únicos que comparten información entre si y se encargan de actualizar los cambios ocurridos en uno de los ambientes en el otro. Por ejemplo, si un bloque de entorno (con interfaz gráfica) es cambiado de posición, el controlador de juego se encarga de actualizar estos cambios en el bloque virtual que contiene. Teniendo en cuenta esto, los componentes son los siguientes:

Controlador de entorno `(GameControllerScript)`

Este componente engloba todos los campos y funciones relacionadas con el manejo del entorno grafico el juego. Está definido como un *script* de Unity, y por ende derivado de la clase MonoBehaivour, la cual permite la interacción con el motor de juego del entorno. Las características de este componente son:

- Controla el bucle del juego ejecutando la lógica en cada cuadro.
- Controla la interfaz y efectos visuales de la aplicación.
- Actualiza el estado de los demás componentes del entorno para reflejar los cambios en la lógica interna y ordena al Controlador de Juego actualizar sus datos internos ante algún cambio del entorno.

- Carga la base de datos de gráficos y los valores iniciales del módulo de adaptabilidad, ya que estos se guardan como recursos en el entorno del juego.

Bloque de entorno `(BlockScript)`

Es una representación gráfica del bloque que controla el usuario. También está definido como un *script* de Unity. Sus funciones son:

- Permite rastrear el movimiento de los bloques en el entorno realizados por el usuario.
- Permite transformar los valores del bloque en el entorno (posición, ángulo, color, capa) que se encuentran en formato Unity a valores en formato permitido en los bloques activos de la lógica.

Controlador de juego `(GameController)`

Este componente engloba todo el modelo y la lógica del juego, las cuales solo existen en clases programadas externas al entorno. Sus funciones son:

- Permitir el acceso a métodos de la lógica de juego.
- Manejar el estado del juego (Juego listo, ronda lista, jugando…)
- Crear sesiones y rondas.
- Mantener la jerarquía Sesión\Ronda\Escenario\Grafico\BloqueActivo.
- Conectar el módulo de adaptabilidad con el entorno.
- Controlar las pruebas con usuarios virtuales.

Agente de juego `(GameAgent)`

Este componente engloba el modelo y la lógica del módulo de adaptabilidad. Está desarrollado para que este módulo sea independiente del resto de los componentes y sirve de puente entre ellos. En secciones posteriores se explicará a detalle el funcionamiento de este módulo. Sus funciones son:

- Procesar las rondas ya terminadas para obtener información referente a el siguiente escenario que debe jugar el usuario.
- Obtener el siguiente escenario y entregarlo al controlador de juego.
- Acceder a las redes neuronales para realizar el aprendizaje supervisado.

Sesión

Representa una sesión, la cual es un grupo de rondas que se juegan en forma continua y con un mismo usuario. Almacena información del usuario y de las rondas jugadas por él, por lo que conceptualmente se interpreta como un conjunto de

información perteneciente a un usuario en particular.

Ronda

Representa una de las rondas o “uno de los juegos” que realiza el usuario durante su sesión, donde se le presenta un reto con características particulares. Almacena el escenario al cual se enfrenta el usuario, su respuesta y las métricas obtenidas de esta, así como el número de la ronda y el estado de la misma.

Escenario

Representa el conjunto de características que identifica el reto que se le presenta al usuario. En ella se encuentra el grafico que el usuario debe imitar, así como otras características que pueden influir en su desempeño, como el tiempo de juego máximo. También almacena el valor de las métricas que representan la dificultad de ese escenario en particular.

Gráfico

Representa un conjunto de bloques que guardan una relación entre sí (ubicación) y que conforman un todo ordenado. En ella se encuentran bloques en diferentes posiciones y ángulos, con texturas y colores diferentes cada uno. Algunas de sus funciones son:

- Calcula las métricas de escenario relacionadas al gráfico.
- Serializar o deserializar el grafico para poder almacenarlo.
- Relacionar un conjunto de bloques en una sola unidad, administrar los bloques y el árbol de bloques.
- Puede ser estático (los bloques no se pueden manipular) o dinámico (los bloques se pueden manipular).

Cada ronda contiene dos gráficos. Uno de ellos está dentro del escenario utilizado y representa la imagen que se le pedirá imitar al usuario. Este grafico es cargado al iniciar la ronda y es estático ya que no se puede manipular. El segundo grafico es el “gráfico de usuario”, el cual es miembro directo de la ronda y representa la imagen que va formando el jugador. Es dinámico ya que se modifica y actualiza para reflejar el estado de los bloques de entorno de acuerdo al movimiento del jugador.

Bloque activo

Representa un bloque lógico con una forma, tamaño, posición, rotación, color y textura definida en un momento del tiempo. Puede también ser estático si es parte

de un gráfico estático y no se puede modificar, o dinámico si puede cambiar. Este último caso ocurre para reflejar las acciones del usuario al intentar reproducir la figura que se le presenta.

Bloque

Es una característica del Bloque Activo que indica la forma y textura del bloque. Se maneja por separado porque estas características son estáticas, es decir no pueden ser cambiadas por el jugador durante el tiempo de juego.

Árbol de bloques activos

Es un bloque activo adaptado para poder manejar jerarquías, pudiendo agregárseles "bloques hijos" a un bloque activo raíz. Hace referencia a clases de su mismo tipo, por lo que se puede manejar como un elemento o un conjunto de elementos simultáneamente. El "árbol de bloques" que contiene uno de estos elementos busca ser una representación jerárquica de los bloques activos presente en el gráfico, y hace referencia directa a ellos.

5.5.2. Relación entre los componentes

La interacción entre los diferentes componentes de la aplicación se puede apreciar a través del diagraman de clases. En la Ilustración 24 se presenta un diagrama de clases simplificado para tener una visión general de la interacción. En el Anexo 3 se puede observar el diagrama completo.

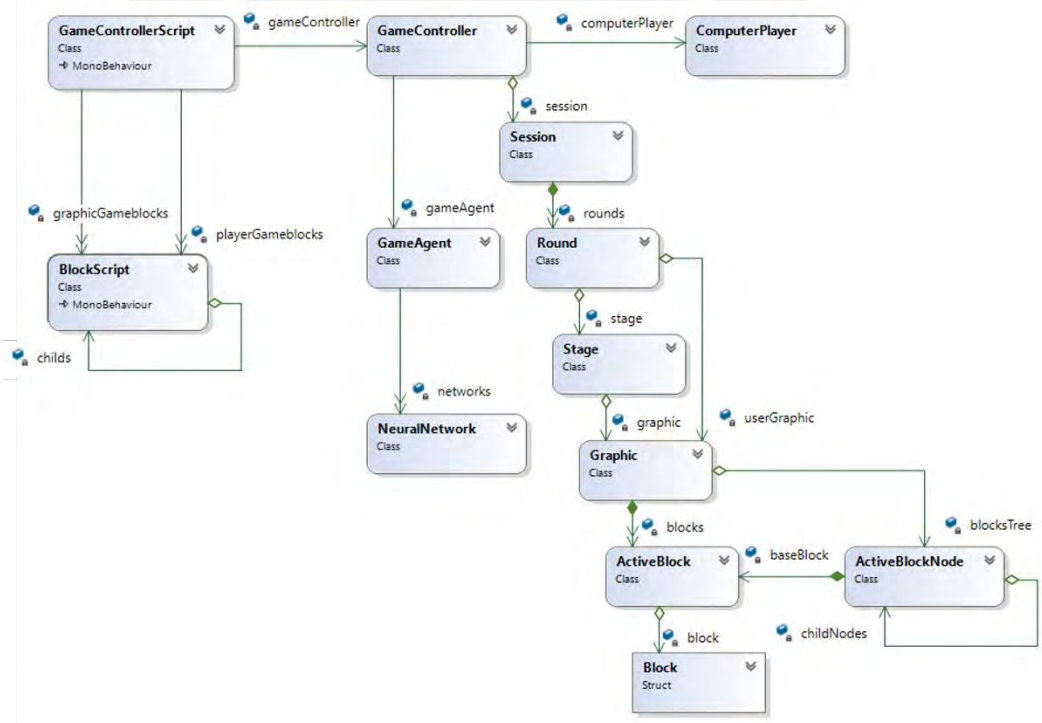

Ilustración** 24**: Diagrama de clases simplificado. Elaboración propia.

6. ALGORITMO DE EVALUACIÓN DE FIGURAS

Parte del proceso que realiza la aplicación es el evaluar la respuesta del usuario comparándola con la figura base que debía imitar. Al hacerlo debe convertir los resultados de esta evaluación en métricas que alimentarán módulo de adaptabilidad. En este capítulo se presenta la solución planteada a este problema de evaluación de figuras.

6.1. Escenario referencial

El algoritmo de evaluación de figuras está inspirado en el trabajo *Contributions of executive function and spatial skills to preschool mathematics achievement* (Verdine, Irwin, Golinkoff, & Hirsh-Pasek, 2014) en el cual se presenta a los estudiantes un escenario similar al que el Proyecto establece. En este trabajo se busca descubrir si existe una relación entre la inteligencia espacial y funciones ejecutivas con el futuro desempeño del estudiante en matemáticas. Para esto se realiza un conjunto de evaluaciones que tienen la siguiente metodología. Se muestra a los estudiantes una figura formada por bloques lógicos y se les pide "hacer que la figura luzca como el modelo" manipulando un conjunto de bloques lógicos reales. En la Ilustración 25 se puede observar un ejemplo de las figuras usadas en la evaluación.

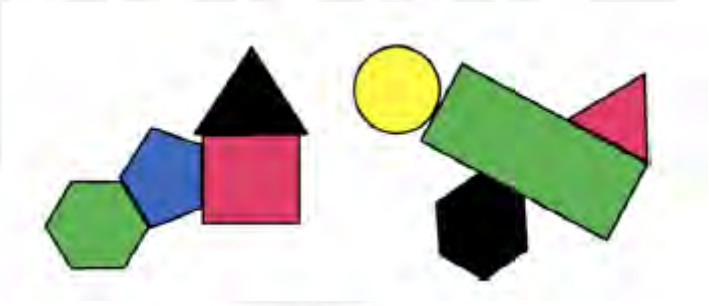

Ilustración 25: Dos ejemplos de figuras usadas para la evaluación de habilidades espaciales. Recuperado de (Verdine, Irwin, Golinkoff, & Hirsh-Pasek, 2014)

Para la evaluación de la respuesta del usuario se usa un criterio definido, el cual se resume en los siguientes puntos:

- Cada figura tiene una pieza base que se usará como pieza de referencia para realizar las mediciones. Esta pieza la más grande, la conectada a más elementos o ambas. Esta pieza no es evaluada por ser de referencia.
- **Piezas adyacentes**: Se verificó que cada pieza de la respuesta esté al costado de su vecina correspondiente (dentro de 1 centímetro). Se ser correcto se otorga un punto.
- **Dirección horizontal y vertical**: Se evaluó si la pieza estaba correctamente

arriba, abajo, derecha o izquierda de su pieza vecina. Si al menos el 50% de la pieza estaba dentro del cuadrante que mostraba la pieza correcta se otorga 1 punto.

- **Posición relativa**: Se verificó si la ficha se encontraba al menos 50% dentro de la posición relativa correcta, otorgando 1 punto en caso fuera cierto.
- Los puntos obtenidos son sumados y comparados con el máximo posible para obtener un porcentaje de acierto.

Estos criterios son usados como base para el algoritmo de evaluación de figuras desarrollado. Es necesario resaltar que este criterio está definido sobre un escenario modelo, el cual difiere del abarcado en el proyecto

6.2. Diferencias entre el escenario referencial y el usado en el Proyecto

El escenario referencial usado en el trabajo de Verdine (2014) y el usado en el proyecto presentan varias similitudes y diferencias que es necesario de tener en cuenta al desarrollar un criterio de evaluación. Como se explicó en un punto anterior, el escenario es el conjunto de características que tiene el reto presentado al usuario. Estas características incluyen la cantidad de bloques usados, las formas de los bloques, las distribuciones de los mismos entre otros. El escenario referencial mostrado anteriormente está definido para centrarse en un conjunto reducido de características, mientras que el usado en el Proyecto abarca una mayor cantidad de ellas. En la Tabla 5 se detallan las similitudes existentes.

Tabla 5: Similitudes entre escenario referencial y real

Escenario referencial y Escenario proyecto
Ambos trabajan con una distribución ordenada de bloques lógicos.
Ambos contemplan la posibilidad de bloques alrededor de otros o en secuencia
Ambos contemplan rotación de bloques
Ambos consideran una figura como bloques “pegados” unos a otros.
Ambos contemplan formas y colores definidos

En la Tabla 6 se detallan las diferencias existentes.

Tabla 6: Diferencias entre escenario referencial y real

Escenario referencial	Escenario proyecto
Se tiene un número relativamente bajo de bloques (2 a 4)	Se puede llegar a un número alto de bloques (2 a 20)
Todas las formas presentan colores diferentes.	Las formas pueden presentar colores iguales
Todas las formas son diferentes	Las formas pueden ser iguales
No se considera textura o superposición de bloques	Si se considera textura y superposición de boques.

6.3. Criterio de evaluación para el Proyecto

Teniendo en cuenta estas diferencias y siguiendo la línea del trabajo mencionado, se estableció el siguiente criterio para la evaluación de las figuras en el Proyecto:

- Todas las figuras cuentan con un bloque base el cual servirá de referencia. El bloque base es el más grande o el que tiene más bloques a su alrededor.
- Todas las figuras usas en la aplicación tiene una organización jerárquica de los bloques que la componen. Cada bloque que compone la figura, con excepción del bloque base, debe tener un bloque padre. Un bloque padre pude tener muchos hijos. En la Ilustración 26: Creación de un gráfico en Modo Dibujo, se pueden apreciar las líneas de jerarquía. Elaboración propia.Ilustración 26 se puede apreciar la aplicación en Modo Dibujo donde se está creando una figura con jerarquías.

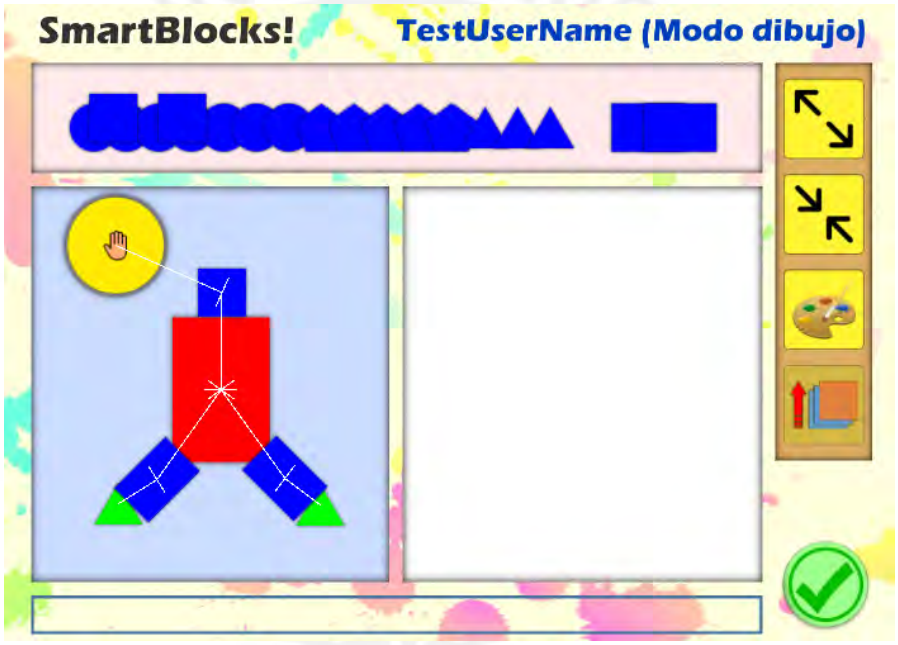

Ilustración 26: Creación de un gráfico en Modo Dibujo, se pueden apreciar las líneas de jerarquía. Elaboración propia.

- Los bloques de una figura deben permanecer cercanos unos a otros de modo que toda la figura pueda ser identificada como un solo objeto. Por ejemplo, una figura no puede tener un árbol y una casa a la vez porque serían dos objetos en la misma figura.
- Para identificar los bloques en la distribución se hace en base a dos criterios
- **Posición relativa adyacente**: El vecino de un bloque (su hijo) se identifica como posible correcto si se encuentra a una distancia establecida identificada como *MIN_MATCH_DISTANCE*. Para las pruebas realizadas esta distancia de se definió en 1.5 unidades.

- **Cuadrante relativo**: Se estableció un cuadrante relativo a la dirección del bloque hijo respecto del padre. El cuadrante va desde menos 45 grados hasta más 45 grados la dirección del bloque hijo respecto del padre, formando así un cuadrante relativo a ellos. Un bloque se identifica como posible correcto si se encuentra dentro del cuadrante relativo establecido.

6.4. Proceso de evaluación

La evaluación de la figura respuesta del usuario es el proceso que determina cuan acertada es la figura del usuario respecto a la mostrada e incluye el cálculo de métricas que indiquen los valores de similitud. Se puede dividir en dos etapas explicadas a continuación.

6.4.1. Etapa de selección de distribución

La primera etapa consiste en establecer cuan parecida es la distribución de los bloques en la respuesta del usuario respecto a la distribución de la figura mostrada. Se entiende como distribución a la ubicación relativa de los bloques uno respecto a otro que forman un conjunto ordenado reconocible. Identificar si la distribución es correcta representa un reto importante porque, si bien a simple vista el cerebro humano puede encontrar similitudes entre los bloques (sobre todo si se asocia la figura a un objeto de mundo real), para el Proyecto se requiere un método objetivo y automatizado de evaluación. En la Ilustración 27 se pueden observar a modo de ejemplo diferentes distribuciones que buscan representar la misma figura. El responder a la figura mostrada, el usuario busca representar cada bloque de la figura con un bloque elegido por él. Sin embargo, los “errores” en la distribución del usuario hacen difícil conocer que bloque del usuario representa cada bloque de la figura.

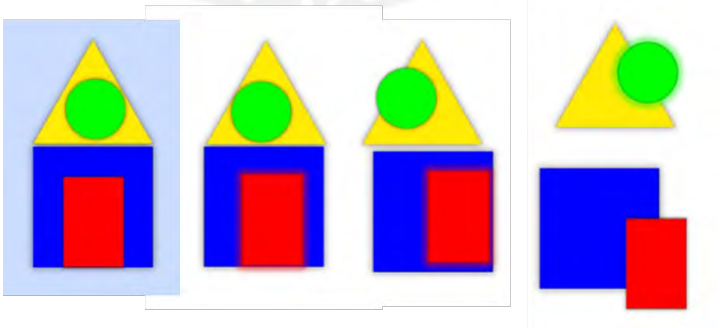

Ilustración 27: Diferentes distribuciones que buscan representar la misma figura. Elaboración propia.

El programa debe analizar los bloques en la respuesta del usuario y reconocer en ella la distribución de la figura mostrada, intentando descubrir que bloque de la figura representa cada bloque del usuario. Para esto el programa se apoya en las

siguientes funciones.

```
Procedimiento EncontrarHijosAdecuados (arbolUsuario, nodoUsuario, nodoDeseado, listaBloques)
    bloquePadre ← nodoUsuario.bloque;
    valorEncajeAcumulado ← 0;
    Para i ← 0 Hasta nodoDeseado.numeroDeHijos Hacer
        bloqueRequerido ← nodoDeseado.hijos[i].bloque;
        anguloRelativoRequerido, distanciaRelativaRequerida ← UnidadesPolares
                                                              (bloqueRequerido.posiciónRelativa);
        Para j ← 0 Hasta listaBloques.numeroDeBloques Hacer
            bloqueEvaluado ← listaBloques[j];
            Si bloqueEvaluado ϵ arbolUsuario Entonces SiguientePara;
            posiciónRelativa ← bloqueEvaluado.posición - bloquePadre.posición;
            anguloRelativo, distanciaRelativa ← UnidadesPolares (posiciónRelativa);
            cambioAngulo ← anguloRelativo - anguloRelativoRequerido;
            cambioDistancia ← distanciaRelativa - distanciaRelativaRequerida;
            Si DentroRangoEncaje (cambioAngulo, cambioDistancia) Entonces
                valorEncajeInterno ← ValorarEncaje (cambioAngulo, cambioDistancia);
                valorSelección ← valorEncajeInterno + Penalidades (bloqueEvaluado, bloqueRequerido);
                Si valorSelección < mejorValorSelección Entonces
                    mejorvalorEncajeInterno ← valorEncajeInterno;
                    mejorValorSeleccion ← valorSelección;
                    mejorBloque ← bloqueEvaluado;
                FinSi
            FinSi
        FinPara
        Si mejorBloque != NULO Entonces
            nodoUsuario.hijos[i] ← NuevoNodo (mejorBloque, nodoModelo.hijos[i].numeroDeHijos);
            valorEncajeAcumulado ← valorEncajeAcumulado + Convertir (valorEncajeInterno);
        SiNo
            nodoUsuario.hijos[i] ← NULO;
        FinSi
    FinPara
    Para i ← 0 Hasta nodoDeseado.numeroDeHijos Hacer
        Si nodoUsuario.hijos[i] != NULO Entonces
            valorEncajeAcumulado ← valorEncajeAcumulado + EncontrarHijosAdecuados (arbolUsuario,
                                                  nodoUsuario.hijos[i], nodoDeseado.hijos[i], listaBloques);
        FinSi
    FinPara
    Devolver valorEncajeAcumulado;
FinProcedimiento


Procedimiento Grafico.ObtenerMejorArbolEncaje (arbolDeseado)
    Para Cada bloque En listaBloques Hacer
        nodoUsuario ← NuevoNodo (bloque, arbolDeseado.numeroDeHijos);
        valorEncajeAcumulado ← EncontrarHijosAdecuados (nodoUsuario, arbolDeseado,
                                                        nodoUsuario, listaBloques);
        Si (valorEncajeAcumulado > mejorValorEncajeAcumulado Entonces
            mejorArbol ← nodoUsuario;
            mejorValorEncajeAcumulado ← valorEncajeAcumulado;
        FinSi
    FinPara
    ratioEncaje ← mejorValorEncajeAcumulado / arbolDeseado.valorEncajeAcumulado;
    Devolver mejorArbol, ratioEncaje;
FinProcedimiento
```

La primera función permite encontrar la distribución más adecuada dado un "nodo deseado" que indica la distribución que se desea encontrar. La distribución encontrada se valoriza en base a los resultados acumulados de la función *ValorarEncaje*, la cual valora cada uno de los bloques pertenecientes a la distribución encontrada de acuerdo a su posición respecto a la deseada. La comparación de la posición se realiza en forma relativa al bloque padre (el anterior en la jerarquía) y teniendo en cuenta dos variables, ambas dependientes de la posición relativa: el ángulo respecto al bloque padre y la distancia respecto al bloque padre (dirección y modulo en coordenadas polares).

```
Función ValorarEncaje (diferenciaAngulo, diferenciaDistancia)
    Devolver | diferenciaAngulo | / Max_Dif_Angular + | diferenciaDistancia | / Max_Dif_Distancia;
FinFunción
```

Donde se define:

- **diferenciaAngulo**: La diferencia en radianes entre la dirección del bloque y la dirección deseada.
- **diferenciaDistancia**: La diferencia en unidades entre la distancia del bloque a su padre y la distancia deseada.
- **Max_Dif_Angular**: La máxima diferencia permitida para la dirección para que se considere un acierto en el encaje.
- **Max_Dif_Distancia**: La máxima diferencia permitida para la distancia para que se considere un acierto en el encaje.

Se eligió esta función de valoración porque de este modo la distancia y el ángulo intervienen de modo equivalente en la valoración del encaje. También, según la variación en el ángulo y distancia al punto deseado se establece si la ficha evaluada está dentro del rango aceptable, y de ser así se le asigna un valor de encaje que indica su exactitud. Además, para decidir si se elige ese bloque como representante del nodo hijo deseado, se penaliza la selección en base a la forma, tamaño y color del bloque. Esto es porque, en caso estas características sean diferentes, es menor probable que el usuario haya pretendido que dicho bloque represente el bloque deseado.

```
Función Penalidades (bloqueEvaluado, bloqueRequerido)
    penalidadEncaje ← 0;
    Si bloqueEvaluado.forma != bloqueRequerido.forma Entonces
        penalidadEncaje ← penalidadEncaje + Penalidad_Forma;
    Si bloqueEvaluado.color != bloqueRequerido.color Entonces
        penalidadEncaje ← penalidadEncaje + Penalidad_Color;
    Si bloqueEvaluado.tamaño != bloqueRequerido.tamaño Entonces
        penalidadEncaje ← penalidadEncaje + Penalidad_Tamaño
    Devolver penalidadEncaje;
```

```
FinFunción
```

La segunda función es quien llama a la primera, y en ella se comparan las mejores distribuciones resultantes de elegir cada uno de los bloques del usuario como bloque base. Se elige entonces como bloque base aquel cuya mejor distribución sea la que posea el mejor valor de encaje acumulado. Este valor de encaje acumulado se usa luego para calcular la métrica **Ratio de encaje** (*MatchingRate*), la cual representa en un solo valor cuan similar es la respuesta del usuario respecto a la figura mostrada, en cuanto a la distribución.

6.4.2. Etapa de comparación individual de bloques

Una vez se ha identificado la mejor distribución para la respuesta del usuario, se compara bloque a bloque con la figura mostrada, determinando para cada bloque si se ha acertado en la forma, el color, el tamaño, la rotación y la textura.

Como una figura puede contar con varios niveles de jerarquía, si al comparar un bloque se observa que este tiene hijos, se compara primero cada uno de sus hijos. El algoritmo entonces evaluará el último nivel en la jerarquía e ira subiendo en ella hasta llegar al bloque base.

Para cada uno de los conceptos evaluados (forma, color, tamaño, rotación, textura), si el valor de un bloque es acertado, se le otorga a ese bloque el valor de 1 en ese concepto. En cambio, si el valor no es acertado, se le entrega a ese bloque el valor de 0. Este valor se transfiere a los bloques padres, es decir, cada bloque tiene la suma de los valores de sus bloques hijos más es suyo propio. De ese modo el bloque base tendrá el acumulado de aciertos de todos los demás bloques en cada concepto.

Con el acumulado de los valores en cada concepto se obtiene las 6 métricas restantes que identifican cuan acertada es la respuesta del usuario.

```
Procedimiento EvaluarGrafico (gráficoUsuario, gráficoDeseado)
    arbolDeseado ← gráficoDeseado.árbol;
    numBloquesDeseado ← gráficoDeseado.numeroDeBloques;
    Si gráficoUsuario.bloquesActivos > 0 Entonces
        arbolUsuario, ratioEncaje ←
                gráficoUsuario.ObtenerMejorArbolEncaje (arbolDeseado);
        // LocationGapRate
        brechaUbicaciónProm ← arbolUsuario.CalcularBrechaUbicaciónPromedio (arbolDeseado);
        holguraUbicación ← Max_Dif_Distancia - brechaUbicaciónProm;
        ratioBrechaUbicación ← (holguraUbicación * arbolUsuario.numeroDeBloques) /
                (Max_Dif_Distancia * (numBloquesDeseado));
        // CorrectShapeRate
        ratioFormaCorrecta ← arbolUsuario.ContarFormasCorrectas (arbolDeseado) /
                numBloquesDeseado;
        // CorrectRotationRate
```

```
            ratioRotaciónCorrecta ← arbolUsuario.ContarRotaionesCorrectas (arbolDeseado) /
                    numBloquesDeseado;
            // CorrectSizeRate
            ratioTamañoCorrecto ← arbolUsuario.ContarTamañosCorrectos (arbolDeseado) /
                    numBloquesDeseado;
            // CorrectColorRate
            ratioColorCorrecto ← arbolUsuario.ContarColoresCorrectos (arbolDeseado) /
                    numBloquesDeseado;
            // CorrectTextureRate
            ratioTexturaCorrecta ← arbolUsuario.ContarTexturasCorrectas (arbolDeseado) /
                    numBloquesDeseado;
        SiNo
            ratioEncaje ← 0;
            ratioBrechaUbicación ← 0;
            ratioRotaciónCorrecta ← 0;
            ratioFormaCorrecta ← 0;
            ratioTamañoCorrecto ← 0;
            ratioColorCorrecto ← 0;
            ratioTexturaCorrecta ← 0;
        FinSi
        métricas.ratioEncaje ← ratioEncaje;
        métricas.ratioBrechaUbicación ← ratioBrechaUbicación;
        métricas.ratioRotaciónCorrecta ← ratioRotaciónCorrecta;
        métricas.ratioFormaCorrecta ← ratioFormaCorrecta;
        métricas.ratioTamañoCorrecto ← ratioTamañoCorrecto;
        métricas.ratioColorCorrecto ← ratioColorCorrecto;
        métricas.ratioTexturaCorrecta ← ratioTexturaCorrecta;
        devolver métricas;
    FinProcedimiento
```

Con las métricas ya definidas para cada uno de los conceptos se puede ejecutar el siguiente paso, el cual es enviar estas métricas al algoritmo de adaptabilidad que devolverá los valores con los que se podrá definir el siguiente escenario para el usuario.

6.5. Verificación de efectividad

Para verificar la efectividad del algoritmo se realizó una comparación estadística entre el resultado del algoritmo y el resultado esperado, ambos sobre un conjunto de gráficos alterados controladamente. Al ser controladas las alteraciones se puede conocer también las métricas del grafico resultante, lo que vendría a ser el resultado esperado. El módulo de evaluación de gráficos, en cambio, no conoce la distribución de la respuesta del usuario y es su labor encontrarla entre los bloques. La comparación estadística permitirá verificar a cuantos de los gráficos modificados el módulo de evaluación les reconoce correctamente la distribución, indicando si este método resulta estadísticamente adecuado para su uso.

Se implementó la clase *GraphicTester* que engloba los procedimientos necesarios para la prueba, además de una aplicación de *Windows Forms* que permite analizar

a detalle la ejecución de la prueba. En la Ilustración 28 se puede observar esta herramienta durante la ejecución.

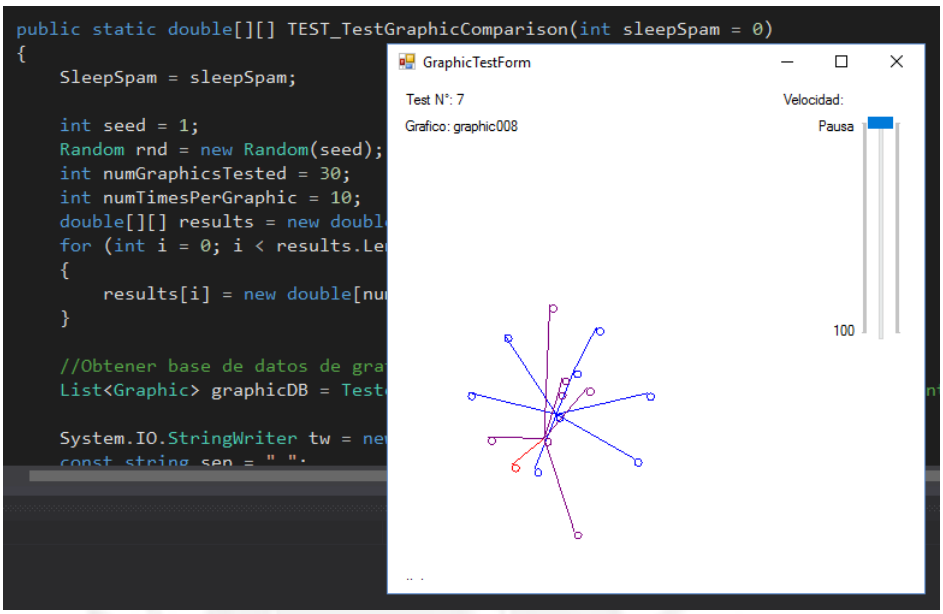

Ilustración 28: Captura de la herramienta desarrollada para el análisis de la prueba de efectividad.

A medida que se realiza la prueba, la aplicación en Windows Forms permite observar lo que sucede internamente. Un *slider* permite controlar la velocidad de la prueba y las etiquetas permiten identificar el grafico evaluado. El grafico de medio es una representación de la distribución de los gráficos involucrados en la evaluación. En azul, la distribución del grafico original sin modificaciones. En rojo, la distribución del grafico modificado. En morado, la distribución reconocida por el módulo de evaluación. Si se observa que el color morado oculta totalmente al rojo, significa que la distribución modificada es totalmente reconocida. En caso se muestren partes rojas, estas son secciones del grafico modificado que no fueron reconocidas por el módulo evaluador. Mediante esta herramienta se pudo identificar correcciones necesarias al módulo evaluador, entre ellas la necesidad de agregar las penalidades anteriormente mencionadas.

Para la prueba general se eligen al azar 30 gráficos, y se prueba cada uno de ellos 10 veces con distintas modificaciones. Se establecen límites de modificación para la dirección y distancia a modificar en la posición de los bloques, los cuales son iguales a la máxima diferencia angular y la máxima diferencia de distancia respectivamente. La variación real en cada bloque se realiza en forma aleatoria respetando el valor elegido como máximo. La función *AlterGraphic* se encargará de transmitir los criterios de modificación a cada uno de los bloques y regresar las métricas relacionadas a las modificaciones luego de realizarlas. Por último, se

llamará al procedimiento *EvaluateGraphic* (EvaluarGráfico en el seudocódigo anterior) para obtener las métricas según esta función y compararlas con las obtenidas por *AlterGraphic*. Los resultados son almacenados en memoria (para la prueba T-Student) y en archivo (para la prueba en hoja de cálculo).

La prueba T-Student se realizó a través de una aplicación de consola y usando el código indicado en el artículo *Ejecución de prueba: prueba T con C#* (Dr. McCaffrey, 2015), el cual, de acuerdo al autor, es público y de libre uso. La prueba compara dos conjuntos de datos y determina cuan probable es que ambas medidas estadísticamente no demuestren diferencias. Para obtener una conclusión se analiza el valor *p-value*, y en caso este sea mayor a 0.05, se considera que no existe suficiente información para indicar que los conjuntos son estadísticamente diferentes. En la Ilustración 29 se observa el resultado de la prueba T-Student para los datos de comparación obtenidos.

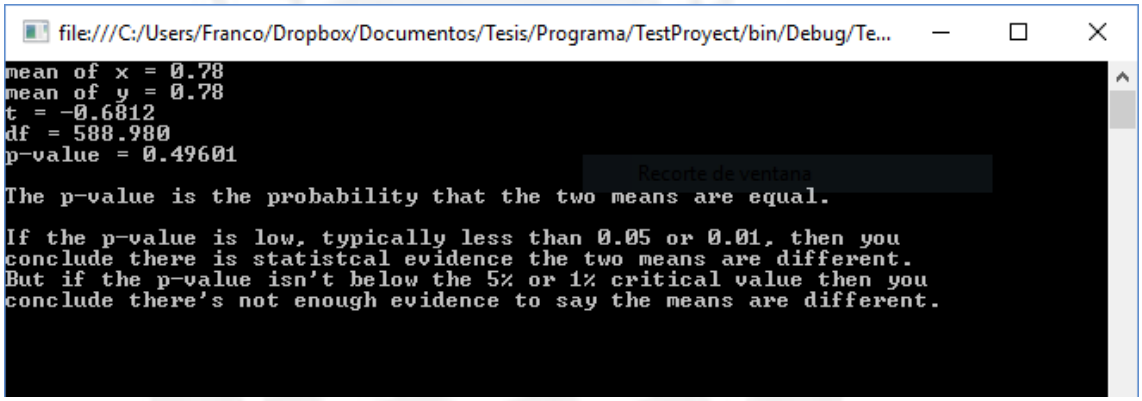

Ilustración 29: Captura del resultado de la prueba T-Student

Siendo el valor *p-value* mayor a 0.05, se concluye que es muy probable que no haya diferencia estadística suficiente entre los resultados esperados y los obtenidos por el módulo evaluador.

Adicional a esta prueba se realizó una prueba en hoja de cálculo, con el objetivo de calcular cuantitativamente la efectividad del módulo. Con la diferencia entre los ratios de encaje y el ratio de encaje esperado se obtuvo el Porcentaje de Acierto. En caso este valor fuera mayor a una constante establecida en 95%, se considera al resultado aceptable. El resultado general de la prueba dio un total de 290 muestras aceptables sobre un total de 300, indicando un 97% de efectividad. La Ilustración 30 muestra parte de los datos analizados y los resultados de la prueba.

N°T	Gname	BC	Ang	Dis	AlterMetrics							UserMetrics							Comparation							%Acier	Acepta
143	graphic001	2	0.74	1.16	0.55	0.43	1.00	1.00	1.00	1.00	1.00	0.55	0.43	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100%	TRUE
151	graphic012	10	0.78	0.87	0.55	0.39	1.00	1.00	1.00	1.00	1.00	0.55	0.39	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100%	TRUE
214	graphic024	10	0.76	1.19	0.51	0.51	1.00	1.00	1.00	1.00	1.00	0.51	0.51	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100%	TRUE
233	graphic012	10	0.71	1.24	0.51	0.35	1.00	1.00	1.00	1.00	1.00	0.51	0.35	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100%	TRUE
108	graphic024	10	0.74	1.42	0.51	0.48	1.00	1.00	1.00	1.00	1.00	0.58	0.53	0.70	0.80	0.80	1.00	1.00	-0.08	-0.06	0.30	0.20	0.20	0.00	0.00	85%	FALSE
180	graphic008	8	0.77	1.40	0.50	0.31	1.00	1.00	1.00	1.00	1.00	0.40	0.24	0.75	0.75	0.75	0.75	0.88	0.10	0.07	0.25	0.25	0.25	0.25	0.13	80%	FALSE
	Media				0.78							0.78							0.01	0.01	0.01	0.01	0.01	0.01	0.01	99%	TRUE
	Mediana				0.78							0.78							0.00	0.00	0.00	0.00	0.00	0.00	0.00	100%	TRUE
	Desviación				0.10							0.11							0.04	0.03	0.05	0.06	0.06	0.06	0.04	6%	
																			Constante Porcentaje Aceptable:								95%
																			N° Aceptables:								290
																			Tamaño Muestra:								300
																			% Aceptables en la muestra:								97%

Ilustración 30: Captura del resultado de la prueba estadística en hoja de cálculo.

7. MÓDULO DE ADAPTABILIDAD

El módulo de adaptabilidad es el núcleo de la aplicación ya que permite adaptar la dificultad de cada escenario de acuerdo al nivel de las habilidades del usuario. De este modo el usuario recibe un escenario ni muy complicado ni muy fácil. En el Proyecto se buscó que esta personalización de la dificultad sea independiente en cada uno de los conceptos evaluados, es decir, pudiendo bajar la dificultad en uno de los conceptos mientras que se sube en otro. En este capítulo se explica detalladamente el diseño de este módulo y los procesos usados para su implementación.

7.1. Requerimientos

Para que una inteligencia artificial que adapte la dificultad del juego sea eficiente se requieren las siguientes características (Hunicke & Chapman, 2004).

1. El juego debe identificar tan rápido como pueda el nivel inicial del usuario
2. Debe rastrear exacta y rápidamente la evolución o regresión del usuario.
3. Debe ser creíble, la adaptación debe ser imperceptible

7.2. Conceptos relevantes

En los siguientes puntos se describen conceptos relevantes para desarrollo del módulo de adaptabilidad y de la Inteligencia Artificial desarrollar.

7.2.1. Aprendizaje por refuerzo

En los algoritmos de aprendizaje por refuerzo el programa no recibe información del entrenamiento desde el exterior como en el aprendizaje supervisado; en su lugar, recibe retroalimentación del medio ambiente a modo de refuerzo o recompensa por sus acciones propias dentro del juego (Pfeifer, 2009). Los agentes pueden ser programados sin decirles explícitamente cómo lograr sus objetivos, sino que son impartidos por la recompensa y el castigo.

Se pueden identificar dos formas de realizar esto (Kaelbling, 1996):

- Buscar en el espacio de comportamientos hasta que uno ha encontrado una estrategia de éxito.
- Encontrar formas de estimar los estados y las acciones en el mundo y asignarles un valor.

El primero se hace a menudo a través de algoritmos genéticos y programación, mientras que el segundo se basa en la programación dinámica y en técnicas estadísticas.

7.2.2. Requerimientos computacionales para una IA adaptativa

Estos cuatro requerimientos computacionales que una inteligencia artificial adaptativa necesita, ya que sin ella sería inútil en la práctica. Los cuatro requisitos computacionales son los siguientes (Spronck, Ponsen, & Sprinkhuizen-Kuyper, 2005):

- Velocidad: La IA de un juego adaptativo debe ser computacionalmente rápida, ya que el aprendizaje se lleva a cabo mientras el juego se realiza.
- Eficacia: La IA de un juego adaptativo debe ser eficaz durante todo el proceso de aprendizaje, para evitar que se convierta inferior a la diseñada manualmente. Cuando es eficaz, la IA de un juego adaptable produce comportamiento exitoso razonable en todo momento.
- Robustez: La IA de un juego adaptativo tiene que ser robusta con respecto a la aleatoriedad inherente a la mayoría de los juegos.
- Eficiencia: La IA de un juego adaptativo debe ser eficiente en relación con el número de oportunidades de aprendizaje necesarias para tener éxito, ya que en un solo juego, un jugador solo experimenta un número limitado de encuentros con situaciones similares.

Para cumplir los requisitos el algoritmo debe ser de "alto rendimiento". Los dos principales factores de importancia cuando se intenta lograr un alto rendimiento para un algoritmo de aprendizaje son la exclusión de aleatoriedad y la adición de conocimiento específico al tema (Fogel, 2000) por lo que será necesario incorporar estos dos elementos a la IA a desarrollar.

7.2.3. Requerimientos funcionales para una IA adaptativa

Los requisitos funcionales no son en sí requerimientos, sino son más bien fuertes preferencias indicadas por los desarrolladores de juegos (Spronck, Ponsen, & Sprinkhuizen-Kuyper, 2005). El no cumplir con los requisitos funcionales significa que muchos desarrolladores de juegos no estarán dispuestos a incluir la técnica en sus juegos, aunque la técnica de buenos resultados y cumpla todos los requisitos computacionales. Los requisitos funcionales son los siguientes (Spronck, Ponsen, & Sprinkhuizen-Kuyper, 2005):

- Claridad: La IA de un juego adaptativo debe producir resultados fácilmente interpretables. Los desarrolladores de juegos suelen desconfiar de las técnicas de aprendizaje ya que los resultados son difíciles de entender.
- Variedad: La IA de un juego adaptativo debe producir una variedad de diferentes comportamientos, porque el comportamiento predecible es menos

entretenido que el comportamiento impredecible.

- Consistencia: El número medio de oportunidades de aprendizaje necesaria para la IA de un juego adaptativo para producir resultados exitosos deben tener una alta consistencia, es decir, una variación baja, para asegurarse de que su logro es independiente tanto del comportamiento del jugador humano y de las fluctuaciones aleatorias en el proceso de aprendizaje.
- Escalabilidad: La IA de un juego adaptativo debe ser capaz de escalar el nivel de dificultad de sus resultados al nivel de habilidad del jugador humano.

7.3. Modelo

En esta parte del documento se explica cómo fue modelado el módulo de adaptabilidad que permitirá realizar la adaptación dinámica de la dificultad.

7.3.1. Metodología de aprendizaje para el presente proyecto

El presente proyecto toma como base el trabajo realizado por Andreas Pfeifer (2009) adaptado al caso particular. Se trabajará con una red neuronal cuya labor es definir las variaciones necesarias de los atributos del escenario con el que interactuará el usuario. Esta red neuronal se entrenará inicialmente con un conjunto de datos de casos ejemplo usando la técnica denominada *backpropagation training* o entrenamiento de propagación hacia atrás, y luego, en tiempo real de juego, el modelo generado atravesará varios ciclos de análisis y mejoras definidos como Aprendizaje Supervisado, de modo que, a medida que el número jugado de rondas aumente, la red neuronal estará más adecuada al usuario particular. La Ilustración 31 muestra la metodología general de trabajo a usar en el presente proyecto.

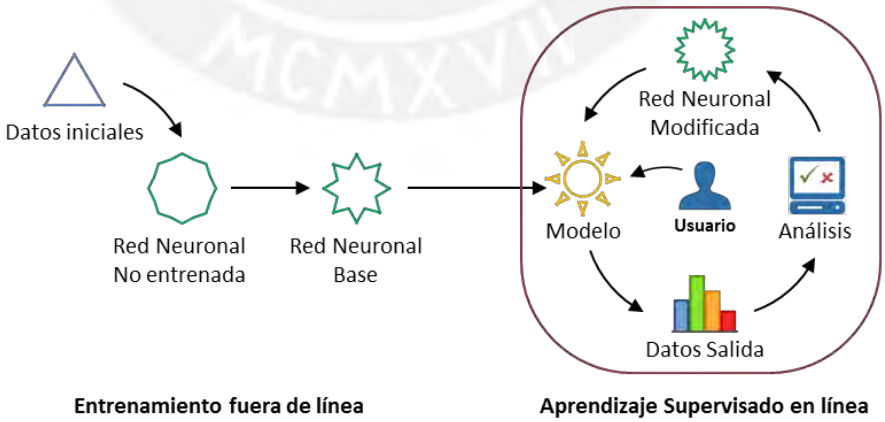

Ilustración 31: Metodología de aprendizaje para el presente proyecto. Elaboración propia.

El módulo está basado en redes neuronales debido a que se busca encontrar una relación entre las métricas de entrada y las métricas de salida, la cual puede ser matemáticamente muy compleja y presentar elementos que no son reconocibles

fácilmente por el desarrollador. Las redes neuronales permiten una solución relativamente simplificada al problema; los cálculos realizados por las capas internas permiten definir patrones y encontrar la relación entre las métricas de entrada y de salida.

El objetivo del algoritmo de adaptabilidad es mantener al usuario dentro de un canal de flujo establecido. Esto se obtiene manteniendo sus métricas dentro de un rango dado que para el proyecto se encuentra entre el 0.4 y 0.7. Un ejemplo grafico de esto se puede observar en la Ilustración 32.

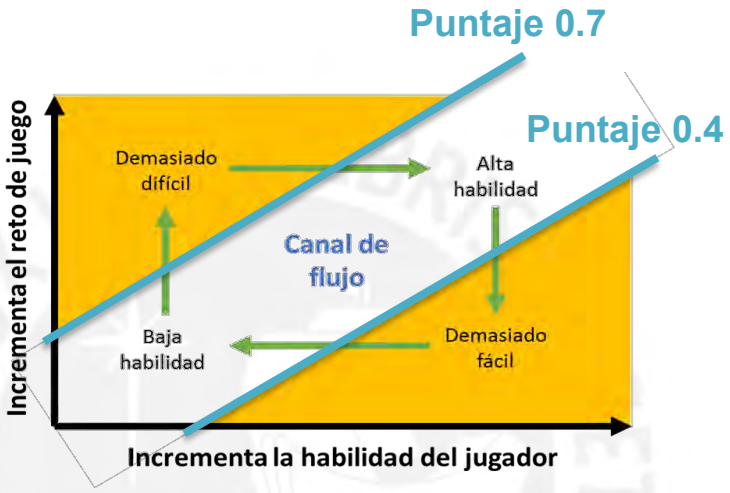

Ilustración 32: Canal de flujo definido para el proyecto. Adaptado de (Hunicke & Chapman, 2004)

Para comprender mejor los valores obtenidos de las métricas se ha decidido clasificarlos en tres posibles categorías:

- **Resultado alto**: Cuando el valor de la métrica es mayor a 0,7 (70% de acierto) Se espera que el algoritmo busque reducir el valor de la métrica, es decir, aumente la dificultad en ese concepto.
- **Resultado medio**: Cuando el resultado es mayor a 0,4 y menor a 0.7 (entre 40% y 70% de acierto) y menor que un resultado alto. Se espera que el algoritmo no modifique este valor, es decir mantenga el nivel actual de dificultad en él.
- **Resultado bajo**: Cuando el resultado es menor a 0,4 (menos del 40% de acierto). Se espera que el algoritmo busque aumentar el valor de la métrica, es decir, reduzca el nivel de dificultad para ese concepto.

Momentos de aprendizaje

Como se mencionó anteriormente, hay dos momentos de aprendizaje:

1. Entrenamiento inicial en base a *BackPropagation Training*, en el cual se usa un

conjunto de valores ejemplo del comportamiento deseado que permite a las redes neuronales dar una respuesta inicial básica. A este momento también le denominamos aprendizaje básico o fuera de línea.

2. Entrenamiento aplicando aprendizaje supervisado durante el juego, el cual evalúa los cambios que sugiere la red y refuerza aquellos que demuestran ser más adecuados.

El uso de estos dos momentos se basa en el trabajo realizado por Andreas Pfeifer sobre juegos adaptativos en tiempo real (Pfeifer, 2009). Este método permite no comenzar el entrenamiento supervisado desde cero, sino contar con una base que luego pueda ser perfeccionada. De este modo se tiene parte de trabajo ya avanzado y se puede tener un mayor detalle en el análisis de los resultados.

7.3.2. Sistema de redes neuronales

Como se ha mencionado, se tienen dos grupos de métricas que interactúan con el sistema: las de entrada y las de salida. Las métricas de entrada indican el nivel de acierto del jugador en cada uno de los conceptos evaluados, mientras que las de salida indican los cambios que deben hacerse en los atributos del escenario. Cada métrica de salida representa un atributo configurable del escenario (cantidad de piezas, numeró máximo de colores), y lo que se busca es definir como debe ser afectado ese atributo para que el siguiente escenario esté mejor adaptado. Estos atributos son los siguientes:

1. Numero de bloques
2. Numero de formas
3. Máxima complejidad de formas
4. Suma de todas las posibles rotaciones
5. Numero de rotaciones máximo
6. Numero de tamaños de bloque
7. Variación en el tamaño de los bloques
8. Numero de bloques con misma forma y diferente textura
9. Numero de texturas
10. Numero de colores

Se decidió entonces crear una red neuronal para cada uno de los atributos, cada una alimentada por las métricas de entrada y que definiría si la métrica de salida

(atributo de escenario) debería subir, bajar, o mantenerse. Se tienen entonces 10 redes neuronales.

Red Neuronal

Cada red neuronal presenta 3 capas: La capa de entrada formada por 7 nodos de entrada, la capa intermedia formada por 5 nodos y la capa de salida formada por 3 nodos de salida.

Nodos de entrada

Las siguientes son las entradas de la red, cada una contiene el valor de una métrica y están dentro del rango 0 a 1.

- **Entrada 1**: Ratio de encaje
- **Entrada 2**: Ratio de desfase de ubicación
- **Entrada 3**: Ratio de forma correcta
- **Entrada 4**: Ratio de tamaño correcto
- **Entrada 5**: Ratio de color correcto
- **Entrada 6**: Ratio de rotación correcta
- **Entrada 7**: Ratio de textura correcta

Mantener los elementos dentro el rango 0 a 1 permite evitar desbordes o resultados muy altos durante los cálculos (Pfeifer, 2009). Todos los números involucrados pertenecen a ese rango.

Nodos medios

Los nodos intermedios permiten realizar cálculos internos que configuran la red para responder de acuerdo a las necesidades. Aunque no existe forma segura de determinar cuál es el número más adecuado de nodos, más que con ensayo y error, se eligió 5 por ser un numero intermedio entre el número de nodos de entrada y de salida de la red, criterio similar al usado en otros trabajos (Pfeifer, 2009).

Salidas

Los nodos de salida de cara red están asociados a las tres acciones posibles a tomar para ese atributo.

- **Salida 1**: Bajar dificultad de atributo
- **Salida 2**: No hacer cambios
- **Salida 3**: Subir dificultad de atributo

Se elegirá la acción dependiendo de la salida con el mayor valor. Por ejemplo, si la

respuesta de la red resulta S1=0.9, S2=0.2 S3=0.1, se elegirá la acción “Bajar dificultad de atributo.

En la Ilustración 33 se muestra un gráfico que representa la red neuronal para uno solo de los atributos.

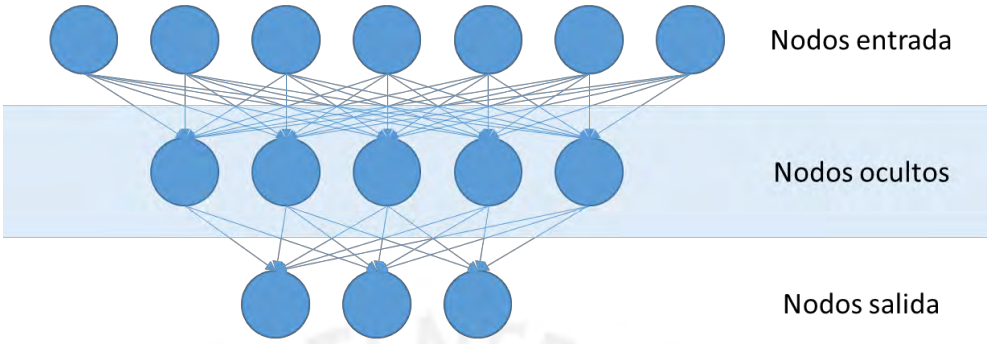

Ilustración 33: Estructura de una de las redes neuronales usada para un atributo. Elaboración propia

El sistema o conjunto de redes neuronales puede graficarse como se aprecia en la Ilustración 34, donde cada **métrica** de entrada es entregada a la **red** de cada atributo, y cada red regresa 3 números que indican cada uno la preferencia que se tiene por cada una de las acciones para ese atributo. Esos 3 valores luego son transformados a un solo valor **salida** que representa la acción a tomar (-1=bajar dificultad, 0 = no modificar, 1 = subir dificultad).

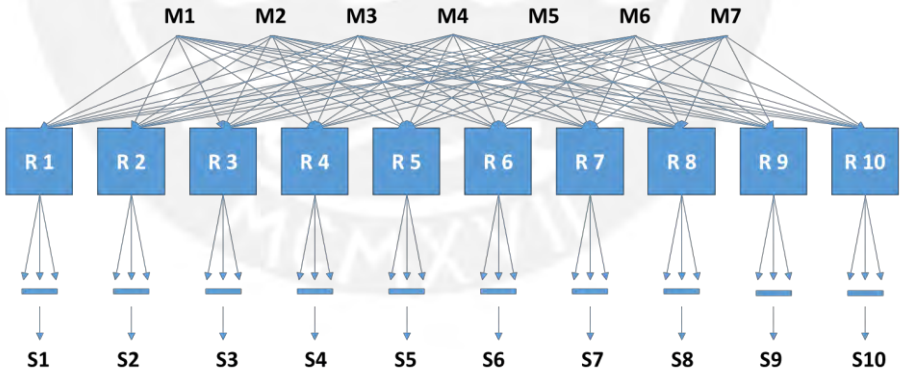

Ilustración 34: Sistema de redes neuronales. Elaboración propia.

7.3.3. Implementación de las redes neuronales

Se implementaron las redes neuronales a través de perceptrones multicapa. Para esto se tomó como referencia la información y el código discutido en el artículo *Coding Neural Network BackPropagation Using C#* (McCaffrey, 2015).

Se definió para el proyecto la clase *NeuralNetwork*, la cual almacena los componentes internos de la red y todas las funciones que permiten realizar el entrenamiento y procesamiento de los datos. Cada una de las redes neuronales del

sistema anteriormente descrito es una instancia de esta clase.

Para este proyecto se utilizó pesos para los arcos que unen los nodos de las capas entrada-oculta y oculta-salida; así como sesgos o *biases* para los nodos de las capas oculta y de salida. Cada uno de estos valores interviene en la generación de los resultados de salida a través de operaciones matemáticas sencillas. Inicialmente estos valores son generados aleatoriamente; es a través del entrenamiento que estos valores se van definiendo de modo que la red pueda responder de manera esperada con una probabilidad considerable a información específica que se le ingrese.

El proceso interno realizado por la red neuronal no se explicará a detalle debido a que este es común y relativamente sencillo; sin embargo, es necesario especificar las funciones internas y valores usados en su funcionamiento, lo que se describe a continuación:

- La red recibe los datos de entrada, y calcula los valores entrantes a los nodos ocultos, multiplicando cada nodo por el arco respectivo y sumando los valores de los sesgos correspondientes.
- A estos valores resultantes se les aplica la función de activación *Hypertan*, la cual restringe los valores al rango [-1,1] pero los mantiene casi iguales a medida que son cercanos al 0.
- Los valores modificados luego son ingresados al cálculo de los nodos de salida, multiplicando cada nodo por el arco respectivo y sumando los valores de los sesgos correspondientes.
- Una vez obtenidos los valores de salida se les aplica la función *Softmax* para que así estos se escalen de modo que la suma de ellos sea 1. Estos son los valores devueltos por la red.

El entrenamiento por propagación hacia atrás para un conjunto especifico de entrada se realiza básicamente calculando la diferencia entre el valor devuelve la red y el valor esperado, aplicándolo a la derivada de las funciones involucradas, comenzado desde la capa de salida y continuando en la capa oculta. Con los valores resultantes se actualizan los pesos y sesgos en función a tres parámetros de aprendizaje:

- La ratio de aprendizaje, que determina que porcentaje del valor de actualización calculado para un peso se termina por agregar realmente a ese peso para modificarlo.
- El momento, que determina que porcentaje del valor de actualización del ciclo

anterior afecta a ese peso en el ciclo actual. La presencia de esta variable previene la oscilación de los valores cuando el entrenamiento no converge a un valor estable, y además contribuye a no quedarse en óptimos locales (McCaffrey, 2015).

- El máximo de épocas, indica la cantidad de veces que se repite el entrenamiento con todos los datos enviados a la función. En cada iteración los datos son reordenados para ser visitados en forma aleatoria.

Los valores utilizados para el entrenamiento en este proyecto se observan en la Tabla 7.

Tabla 7: Variables de aprendizaje para el entrenamiento inicial

Variable	Valor
Numero de épocas	1000;
Ratio de aprendizaje	0.05;
Momento	0.01;

También se implementaron funciones que permitieron guardar y cargar desde texto y archivo la información interna de un conjunto de redes neuronales ya entrenadas, a través de su serialización en formato XML, lo que permite almacenar la red neuronal de cada uno de los usuarios por separado.

7.4. Implementación del entrenamiento básico

La primera fase de configuración se realizó a través del método de propagación hacia atrás o *Backpropagation Training*. Para este proceso se usó un conjunto de casos representativos que indican el comportamiento genérico que se espera recibir de las redes neuronales ante determinadas entradas. Estos casos representativos son pares “estado/acción” que representan el comportamiento básico que se desea lograr: ante determinadas entradas, determinada acción sobre determinado atributo. Los valores definidos para cada uno de los casos representativos elegidos fueron definidos de forma intuitiva ya que representa un método heurístico, pero basados en el trabajo realizado en el proceso de selección de métricas, y teniendo en cuenta la información recopilada sobre los conceptos evaluados. Por lo que se pueden considerar un punto de partida razonable.

Dentro de estos estados representativos a los que se ha definido una acción están los siguientes:

- Cuando uno de los conceptos tiene un bajo resultado mientras que todos los demás tienen alto resultado

- Cuando uno de los conceptos tiene un bajo resultado mientras todos los demás tienen resultado medio.
- Cuando uno de los conceptos tiene alto resultado mientras que todos los demás tienen un bajo resultado.
- Cuando dos conceptos tienen un resultado (alto o bajo) mientras que los demás tienen un resultado opuesto.
- En los casos extremos como todos altos, todos bajos, o valores intercalados.

Los datos de entrenamiento iniciales se pueden apreciar en Anexo 4**¡Error! No se encuentra el origen de la referencia.**.

El entrenamiento básico se realizó desde una aplicación de línea de comandos separada de la aplicación final. Se comprobó, mediante el ingreso de datos de entrada de ejemplo, que las redes devolvían valores coherentes y correspondiente con el comportamiento deseado. La red neuronal resultante fue luego serializada con las funciones previamente definidas y trasladada a la aplicación final como red neuronal base para la siguiente fase de entrenamiento.

Cabe resaltar que tanto la aplicación de línea de comandos como la aplicación final hacen referencia al mismo código y librerías desarrolladas, siendo diferente solo la interfaz y el controlador que integra esta al resto de las funciones. Esta forma de manejar el entrenamiento se eligió porque permite realizar pruebas en un entorno separado, sin la necesidad de cargar toda la interfaz gráfica y reduciendo las complicaciones que podría generar el entorno de desarrollo Unity. El entorno de programación es el mismo para la interfaz y para la programación (Visual estudio C#), por lo que no existieron complicaciones o incompatibilidades en cuando a la separación de entornos.

7.5. Implementación del Aprendizaje supervisado

En esta parte del Proyecto se buscó mejorar y personalizar los resultados obtenidos en el entrenamiento básico al incorporar un tipo de entrenamiento supervisado, es decir en presencia de usuario. Para realizar esto, se definió un conjunto de criterios que permiten valorar la salida obtenida de la red. Luego de realizar esta valoración se puede reforzar aquellas respuestas que demuestran un valor adecuado o tomar acciones ante aquellas que no presentan resultados satisfactorios.

La implementación de esta técnica puede dividirse en dos etapas: la valoración de las respuestas obtenidas de la red y la retroalimentación realizada con estas respuestas.

Valoración de respuestas

Definir si una respuesta de la red tiene un valor positivo o negativo depende de cuan adecuadamente esta respuesta cumple con los objetivos definidos. Estos objetivos se obtienen de dos fuentes: los requerimientos funcionales y computacionales anteriormente especificados, y los objetivos propios del juego.

Los objetivos de la primera fuente son analizados en la ejecución de las pruebas virtuales y no se considerarán en este proceso. En cuanto a los objetivos propios del juego, el principal criterio para tener en cuenta es la adaptabilidad, es decir, mantener al usuario dentro de un rango adecuado de dificultad. Como se ha mencionado anteriormente, la aplicación debe presentar al usuario retos que se encuentren dentro del "canal de flujo", mostrando retos ni muy fáciles ni muy difíciles. Valorar el resultado de las redes neuronales es entonces valorar **la eficacia del algoritmo para lograr que el usuario suba o baje el valor de sus métricas en relación con el resultado anterior**; de modo que ingrese o se mantenga dentro del canal de flujo.

Para esto, se necesita una serie de criterios que comparen los resultados de una ronda respecto a la ronda anterior, y así reforzar los puntos que colaboren con el objetivo deseado. Cada uno de estos criterios involucra situaciones en el estado juego, a las cuales en texto de Andreas Pfeifer (2009) se refiere como **eventos**. Para cada uno de estos eventos se ha definido una **acción** a tomar, la cual está definida en base al conocimiento que se tiene de la dinámica interna del juego. Es por eso que este conjunto de eventos-reglas se considera como **aprendizaje supervisado**, pues, a diferencia del aprendizaje por refuerzo tradicional, el refuerzo no se aplica directamente sobre la acción tomada por la red, sino que se puede reforzar una acción que se considera más adecuada a esa situación. De este modo se logra tener un mejor control sobre el comportamiento resultante de la red (Pfeifer, 2009).

Se tiene los siguientes criterios que permitirán guiar la selección de eventos.

Respecto a la respuesta de usuario se tiene, para cada métrica de usuario:

- Cuando el valor de la métrica entra al rango o se mantiene dentro, se puede decir que el efecto es positivo.
- Cuando el valor de la métrica estaba adentro y sale del rango o se mantiene fuera, se puede decir que el efecto es negativo.

Respecto a la modificación de los atributos se pueden realizar algunos supuestos que permitirán también guiar la selección de eventos. Para cada atributo:

- Cuando el valor de un atributo suba, es imposible que tenga relación con que el valor de una métrica de usuario suba.
- Cuando el valor de un atributo baje, es imposible que tenga relación con que el valor de una métrica de usuario baje.
- Cuando el valor de un atributo se mantenga, es imposible que tenga relación con la variación del valor en algunas métricas de usuario.

Cabe recordar que el rango definido como adecuado para cada métrica de usuario es el que representa un "resultado medio", como se explica en el punto 7.3.1 y que se obtiene cuando el valor numérico del resultado se encuentra entre 0.4 y 0.7.

Eventos

Siguiendo esta línea se definieron los siguientes eventos y su respectiva acción a tomar en cada uno de ellos. En la Tabla 8 podemos ver los eventos relacionados a un refuerzo directo de las respuestas del usuario.

Tabla 8: Reglas de aprendizaje relacionadas a un refuerzo directo

N°	Evento	Acción
1a	Cuando todas las métricas fuera del rango en el resultado anterior ingresan al rango y las que estaban dentro se mantienen	Se refuerzan la acción tomada en todos los atributos.
2a	Cuando todas las métricas dentro del rango en el resultado anterior salen del rango y las que estaban fuera se mantienen	Se refuerza la acción "mantener" en aquellos atributos que subieron o bajaron, y se refuerza aleatoriamente "subir" o "bajar" en los que indicaban mantener.

También se definieron eventos relacionados a un refuerzo directo, pero de forma parcial, reforzando solo de aquellos atributos que se consideran tuvieron influencia directa en el cambio de la respuesta. En la Tabla 9 se aprecian estos eventos.

Tabla 9: Reglas de aprendizaje relacionadas a un refuerzo parcial

N°	Evento	Acción
1b	Cuando todos los conceptos fuera del rango ingresan al rango o se mantienen afuera, y los que estaban dentro se mantienen	Se refuerzan solo las acciones "subir" o "bajar" en los atributos que las indicaban
2b	Cuando todos los conceptos dentro del rango salen del rango o se mantienen, y los que estaban fuera se mantienen	Se refuerza la acción "mantener" solo en aquellos atributos que subieron o bajaron

Existen situaciones en las que se puede saber con seguridad que los conceptos afectados tienen relación directa con los atributos modificados. Estos eventos buscan relacionar cambios en los atributos con cambios en la respuesta del usuario

para reforzarlos positiva o negativamente. En la Tabla 10 se tiene los eventos relacionados a estas situaciones.

Tabla 10: Reglas de aprendizaje relacionadas a cambios en respuestas y atributos

N°	Evento	Acción
3	Cuando solo hay métricas altas que han bajado y las demás se han mantenido	Se refuerza la acción “subir” solo en aquellos atributos que la indiquen
4	Cuando solo hay métricas bajas que han subido y las demás se han mantenido	Se refuerza la acción “bajar” solo en aquellos atributos que la indiquen

También se pueden identificar situaciones en las que algunos valores de la respuesta del usuario no han cambiado, y se desea activar un cambio en estos valores de modo que el usuario pueda experimentar escenarios diferentes. En la Tabla 11 se describen estas situaciones.

Tabla 11: Reglas de aprendizaje relacionadas a ausencia de cambios

N°	Evento	Regla/Acción
5	Cuando hay métricas que no cambian, si estos solo están sobre el rango.	Se refuerza la acción “subir” en aquellos atributos que indicaban “mantener”.
6	Cuando hay métricas que no cambian, si estos están solo bajo el rango.	Se refuerza la acción “bajar” en aquellos atributos que indicaban “mantener”

Retroalimentación supervisada

Luego de que una ronda es jugada por el usuario y antes de generar el siguiente escenario, esta es analizada para verificar si cumple con alguna de las reglas anteriormente descritas. Una vez se han identificado un evento y acción a tomar, se lo establece en forma numérica como un arreglo de arreglos de decimales. Debido a que cada evento puede afectar a varios atributos (lo que determina un arreglo) y para el refuerzo cada atributo se debe indicar información de la respuesta del usuario y de la respuesta esperada que se desea entrenar. En la Ilustración 35 se ejemplifica esto.

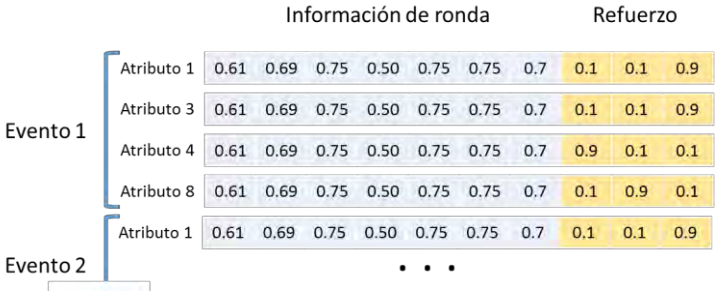

Ilustración 35: Ejemplo de información de refuerzo

Cada uno de estos arreglos representa entonces un “elemento de entrenamiento”, los cuales serán ingresados al sistema de redes neuronales a través del mismo método *BackPropagation* utilizado en el entrenamiento inicial. Cada arreglo es ingresado en el método de entrenamiento del atributo respectivo al que se desea modificar. Las variables de aprendizaje utilizadas en este proceso se aprecian en la Tabla 12.

Tabla 12: Variables de aprendizaje para el entrenamiento supervisado

Variable	Valor
Numero de épocas	10
Ratio de aprendizaje	0.1
Momento	0.01

7.6. Función selectora de escenarios

Esta es una parte importante del módulo de adaptabilidad, ya que explica cómo son utilizados los resultados del sistema de redes neuronales para seleccionar el siguiente escenario. El sistema de redes neuronales regresa la acción necesaria para cada atributo, es decir, si el atributo debe subir, mantener o bajar su dificultad. Al estar valorada la dificultad de cada atributo como un numero entre el rango de 0 y 1, las variaciones se expresan también como valores numéricos, negativos o positivos, que se suman al valor actual de los atributos para obtener los nuevos atributos recomendados.

Lo ideal en este proceso seria que se eligiera un gráfico que cumpla todos los valores de los nuevos atributos. Sin embargo, en el presente proyecto, la cantidad de gráficos es limitada y, si bien consideran una gran variedad de valores para los atributos y combinaciones posibles de los mismos, no siempre se puede obtener un gráfico que cumpla totalmente lo deseado. Esta función debe elegir en función a los nuevos atributos el grafico de la base de datos que más se acerque a lo requerido.

Transformación de resultados de red a valores de variación y obtención de nuevos atributos

En primer lugar, el resultado de la red para cada atributo (-1, 0 o 1) se transforma a variaciones en valor numérico (ej.: 0.25 o -0,15). Este valor de variación es independiente a cada atributo, y especifico de las posibilidades de variación reales de los mismos. Un ejemplo de la obtención de los nuevos atributos se observa en la Ilustración 36.

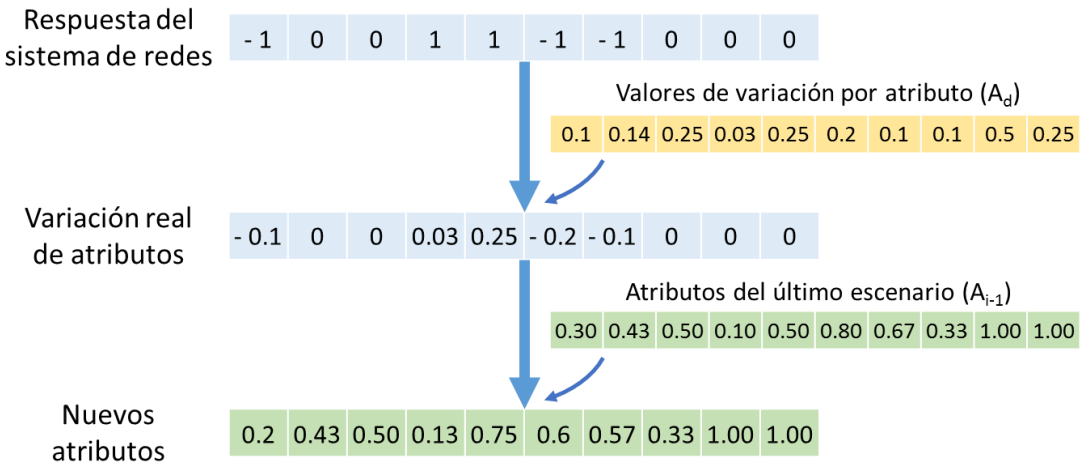

Ilustración 36: Transformación y obtención de nuevos atributos

Una vez obtenidos los nuevos atributos se procede a buscar en la base de datos de gráficos aquel que encaje mejor con ellos. La base de datos de gráficos almacena una serie de gráficos diferentes, cada uno con atributos particulares. El objetivo de la función selectora es buscar entre ellos el siguiente grafico a usar en base a los nuevos atributos.

Modelo de selección

El primer paso es seleccionar de la base de datos los gráficos que mejor cumplan con cada uno de los atributos de forma independiente. Para esto cada atributo posee un rango de aceptación de +- el valor de variación por atributo, el cual permite explorar una cantidad mayor de gráficos sin afectar significativamente los requerimientos. Si el grafico se encuentra dentro del rango de aceptación de ese atributo, una referencia a ese grafico es almacenada y relacionada con dicho atributo. Si ningún grafico está dentro del rango, se almacena una relación al grafico que más se acercó a dicho rango. En la Ilustración 37 se observa este paso.

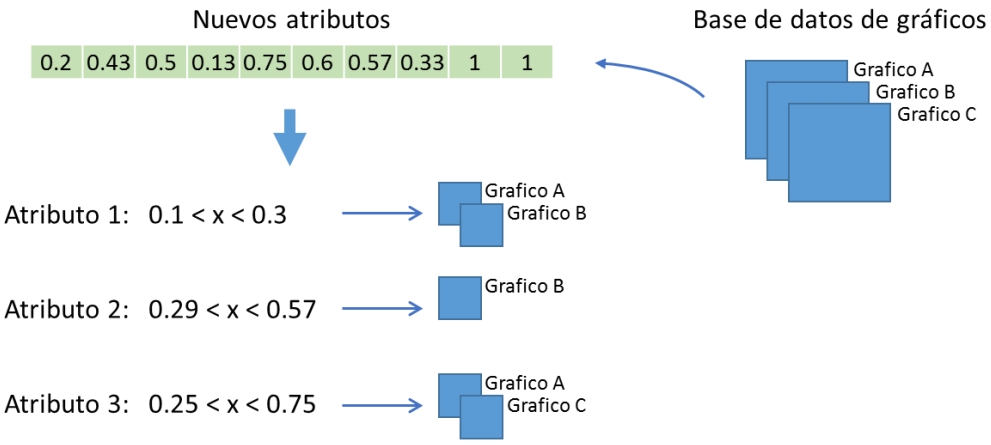

Ilustración 37: Selección de los mejores gráficos en base a rangos del atributo

Segundo, una vez obtenidos los mejores gráficos para cada atributo, se procede a valorar cada uno de esos gráficos en función a cuantos atributos cumplen. Si el grafico cumple con más atributos dentro de sus rangos, mayor puntaje obtiene. Para evitar empates se definieron pesos para cada uno de los atributos. Esto también permite dar preferencia a ciertos atributos que se pueden considerar más significativos, evitando por ejemplo que se seleccione un gráfico con cantidad de bloques mucho menor que la requerida en pro de un encaje más adecuado a los atributos de rotación o tamaño. La Ilustración 38 ejemplifica esto.

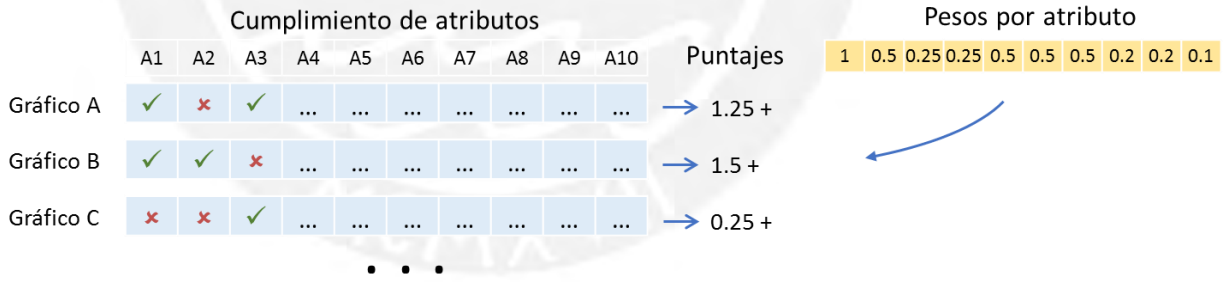

Ilustración 38: Valoración de gráficos en base a cumplimiento de atributos

Por último, se selecciona el grafico con mayor puntaje, y este es el que se envía al siguiente escenario.

7.7. Interfaz de pruebas

Se creó la clase *SessionTester* para englobar los procedimientos necesarios para la prueba, y se integró con una aplicación de *Windows Forms* que permitió analizar a detalle la ejecución de la prueba. En la Ilustración 39 se puede observar esta herramienta durante la ejecución.

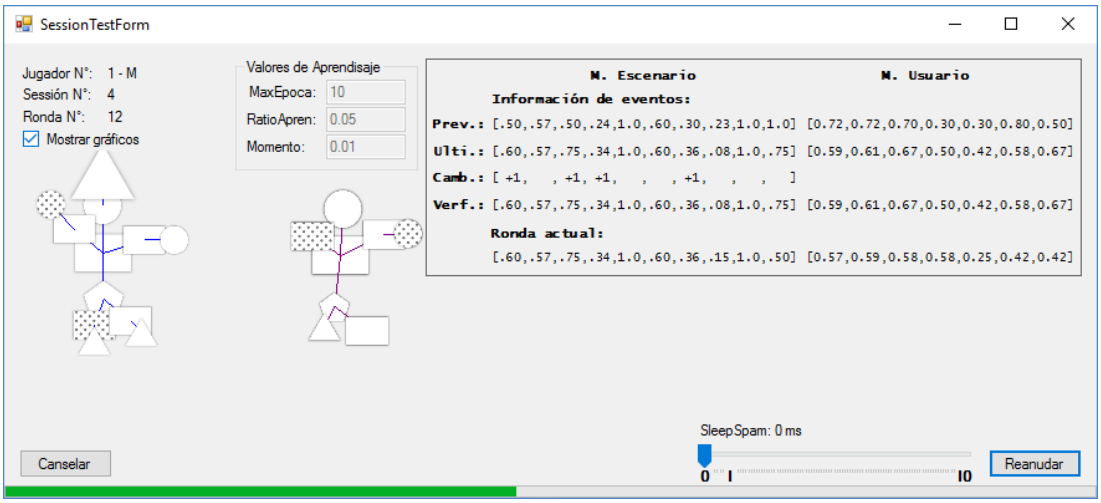

Ilustración 39: Captura de la herramienta desarrollada para el análisis de la prueba de adaptabilidad.

A medida que se realiza la prueba, la aplicación en *Windows Forms* permite observar lo que sucede internamente. La interfaz cuenta con una zona donde se puede observar el grafico evaluado y la respuesta del usuario virtual, y otra zona donde se observan las respuestas de las ultimas 3 rondas y su análisis para determinar la existencia de eventos. Un control de desplazamiento permite controlar la velocidad de la prueba. También existe una opción para detenerse al detectar un evento y poder evaluarlo a detalle con la información mostrada. Mediante esta herramienta se pudieron identificar las mejores opciones en cuanto a los eventos a utilizar y se comprendió mejor como estos afectan los resultados.

8. PRUEBAS AL MÓDULO DE ADAPTABILIDAD

En este capítulo se explica cómo se realizó la evaluación del módulo de adaptabilidad tanto antes como después del aprendizaje supervisado. Se explica también la creación de usuarios virtuales para la evaluación y el análisis del módulo de adaptabilidad tanto a juicio del desarrollador como en base a los requerimientos computacionales y funcionales definidos por Spronck (2005).

8.1. Definición de usuarios virtuales

Para el análisis de la efectividad del sistema de redes neuronales fue necesario realizar pruebas a gran escala que permitan revisar la evolución de las métricas del usuario a lo largo del juego. Para esto es necesario contar con usuarios de prueba que puedan ejecutar la aplicación un gran número de veces. Siendo esto impráctico en la vida real y con una alta demanda de tiempo y recursos, se optó por crear un "usuario virtual", el cual emula el comportamiento del jugador durante el desarrollo de las pruebas. Esta es una técnica recurrente durante el análisis de juegos y especialmente para la aplicación del balance dinámico de la dificultad (Andrade, Ramalho, & Santana, 2005).

Usuario virtual

Denominado también dentro del proyecto *ComputerPlayer* debido a que es un agente de juego controlado por la computadora. Este agente imita la respuesta del usuario al tener una **capacidad de respuesta limitada**, la cual se establece al iniciar la sesión y es independiente para cada uno de los conceptos evaluados.

Este usuario virtual posee una serie de **habilidades** que representan capacidades de respuesta relacionadas a las acciones que permite el juego (por ejemplo, mover, rotar, colorear) y en cada una de esas habilidades posee **puntos de acierto**, los cuales permiten responder positivamente pero solo hasta cierto nivel. Estas habilidades y puntos de acierto están inspirados en una observación previa de usuarios utilizando la aplicación (sin adaptabilidad) donde se encontró que estos solían cometer los mismos errores y en cantidad similar a través de los diferentes escenarios. Las habilidades definidas para el usuario virtual se observan en la Tabla 13.

Tabla 13: Habilidades para los puntos de acierto del usuario virtual

N°	Habilidad	Nombre en código	Función
1	Bloques usados	numUsedBlocksPoints	Número máximo de bloques que el usuario saca de la zona de no usados.
2	Puntos Forma	numCorrectShapePoints	Número de bloques que en promedio el usuario acierta en la forma.
3	Puntos Posición	numCorrectLocationPoints	Número de bloques que en promedio el usuario coloca a una distancia valida de su ubicación correcta.
4	Rango Posición	correctLocactionRange	Error promedio en distancia de un bloque con ubicación correcta.
5	Puntos Dirección	numCorrectDirectionPoints	Número de bloques que en promedio el usuario coloca a un ángulo valido de su dirección correcta.
6	Rango Dirección	correctDirectionRange	Error promedio en ángulo de un bloque con dirección correcta.
7	Puntos Rotación	numCorrectRotationPoints	Número de bloques que en promedio el usuario acierta en la rotación.
8	Puntos Color	numCorrectColorPoints	Número de bloques que en promedio el usuario acierta en el color.
9	Puntos Tamaño	numCorrectSizePoints	Número de bloques que en promedio el usuario acierta en el tamaño.
10	Puntos Textura	numCorrectTexturePoints	Número de bloques que en promedio el usuario acierta en la textura.

Se integró en la aplicación la capacidad de guardar registro de cada una de las jugadas realizadas, almacenando entre otras cosas información de usuario, de su respuesta y del escenario. Esta información se usó más adelante para establecer los valores de cada habilidad del usuario. Se crearon en base a esto varios perfiles de jugador con capacidades diferentes.

Perfiles obtenidos de usuarios reales

Se registraron los resultados de doce alumnos, usuarios de entre 4 y 5 años, cada uno de los cuales jugó un promedio de 4 rondas. Luego se analizaron estos resultados para definir los puntos de acción de los usuarios virtuales. De nueve de estos alumnos se obtuvo data utilizable. Los resultados de estos usuarios pueden observares en el Anexo 5.

Como resultado del análisis se descubrió que usuarios de la misma edad pueden tener resultados muy diferentes, es por esto que se decidió clasificarlos no en base a esta sino en base al desempeño. Producto de esto se definieron los siguientes usuarios virtuales:

- **Usuario desempeño bajo**: En este usuario los puntos de acierto son bajos, limitándose por ejemplo a un uso máximo de 5 bloques y solo 3 bloques en distribución correcta

- **Usuario desempeño medio**: En este usuario los puntos de acierto son mayores, llegando a usar hasta 8 bloques, con alrededor de 7 en distribución correcta.
- **Usuario desempeño alto**: En este usuario los puntos de acción son muy altos, usando todos los bloques (17) y acertando casi todos ellos (15) con un rango de error pequeño.

En la Tabla 14 se pueden observar los puntos de acierto para cada una de las habilidades en los tres tipos de usuario.

Tabla 14: Puntos de acierto para cada uno de los usuarios virtuales definidos

N°	Habilidad	Puntos		
		Desemp. Bajo	Desemp. Medio	Desemp. Alto
1	Cantidad Bloques usados	4	8	15
2	Cantidad Forma correcta	3	7	12
3	Cantidad Ubicación correcta	3	8	12
4	Rango Ubicación correcta	0,61	0.56	0,41
5	Cantidad Dirección correcta	3	8	12
6	Rango Dirección correcta	17,4	17,5	23,1
7	Cantidad Rotación correcta	3	4	7
8	Cantidad Color correcto	2	5	10
9	Cantidad Tamaño correcto	2	3	7
10	Cantidad Textura correcta	2	5	7

8.2. Prueba de funcionamiento y modificaciones Iniciales

Para realizar una prueba preliminar del correcto funcionamiento de los algoritmos, y previo a la definición de los usuarios virtuales, se definió un usuario virtual con los puntos de acción establecidos a juicio de desarrollador, teniendo en cuenta las experiencias de juego observadas y su conocimiento sobre el funcionamiento interno de la aplicación. A este usuario se le denominó **Usuario X**. En el Anexo 6 se exponen las modificaciones las resaltantes realizadas luego de una prueba preliminar con el Usuario X, las cuales corrigieron algunos problemas encontrados en el módulo.

8.3. Definición de prueba de los algoritmos

Para el desarrollo de las pruebas se utilizaron dos conceptos tomados de uno de los trabajos investigados sobre desarrollo de inteligencia artificial adaptativa en juegos (Pfeifer, 2009).

- ***GameRounds***: Es cada una de las rondas o retos individuales que se le presenta al usuario, en donde este se enfrenta a un escenario diferente cada vez y desde donde se obtienen las métricas de la respuesta del usuario.

- ***TestRun***: Conjunto de *GameRounds,* realizados con un mismo usuario virtual. Para el presente proyecto, cada *TestRun* consta de varias sesiones, en cada una de las cuales se ejecutan varios *GameRounds*. Permite analizar los resultados de adaptación para ese usuario en particular.

Para este proyecto se realizaron **tres *TestRuns***, uno para cada tipo de usuario virtual. En cada TestRun se ejecutan **diez sesiones**, y en cada una de estas **cincuenta *GameRounds***, número que representa una cantidad alta pero dentro del rango de lo que un jugador podría realizar por sesión.

Los resultados de la primera sesión de cada una de estas pruebas se muestran como tablas en la sección anexos, desde el Anexo 7 hasta el Anexo 12. En estas tablas cada fila representa cada uno de los *GameRound* o rondas jugadas. Los primeros siete valores (M1 a M7) representan el acierto del jugador en dicha ronda expresado en las métricas de usuario. Los diez valores siguientes representan la decisión tomada por las redes neuronales en cada atributo del escenario en base a la respuesta del usuario, donde 1 significa un aumento de la dificultad en ese atributo, 0 implica no cambio y -1 una disminución del mismo. Los últimos 10 valores representan el valor de los atributos del escenario elegido. Es importante recordar que la salida de las redes neuronales es en respuesta a las métricas de usuario ingresadas, pero no se puede asegurar que una salida en particular responde a una métrica en particular, pues múltiples atributos pueden afectar múltiples métricas y viceversa. Las demás sesiones no se han incluido en el documento debido a su extenso tamaño.

Para cada modelo evaluado se ha realizado un resumen de los resultados con los datos promedios obtenidos en las 10 sesiones, los cuales se muestran en las siguientes secciones.

8.4. Resultados y análisis del modelo con IA manual

Para poder realizar una comparación cuantitativa y cualitativa de los resultados se requiere un punto de referencia, el cual, en casos semejantes, suele ser una IA desarrollada manualmente. Ésta es una versión sencilla de la solución, sin ningún tipo de aprendizaje, pero que devuelve un comportamiento básico requerido. Para este proyecto se desarrolló una función en la que se ha establecido, rudimentariamente y en forma predefinida, relaciones entre métricas y atributos, indicando cuales de estos últimos deberían subir o bajar ante los cambios en los primeros. Igual que en caso entrenado, se le entrega a esta función las métricas de usuario y esta devuelve los cambios a realizar en los atributos. En la Tabla 15 se

pueden observas los resultados promedio obtenidos en cada una de las métricas de usuario al utilizar este modelo.

Tabla 15: Resultados promedio con IA manual

Usuario	Criterio	M1	M2	M3	M4	M5	M6	M7
Desempeño bajo	Promedio	0.51	0.53	0.62	0.40	0.44	0.64	0.65
	Desv. Estand.	0.13	0.12	0.16	0.13	0.15	0.15	0.15
Desempeño medio	Promedio	0.61	0.61	0.65	0.49	0.40	0.58	0.65
	Desv. Estand.	0.13	0.12	0.16	0.13	0.15	0.15	0.15
Desempeño alto	Promedio	0.63	0.62	0.70	0.66	0.62	0.61	0.73
	Desv. Estand.	0.11	0.10	0.12	0.09	0.12	0.15	0.12

Analizando los datos encontrados se obtuvieron las siguientes observaciones:

- Los resultados indican que con la IA manual ya existe una cierta adaptabilidad. Esto se puede afirmar debido a que se observa que la mayor parte de los resultados se encuentran dentro del rango deseado, aunque muchos de ellos aun cercanos a los extremos de este.
- Se puede observar una especial dificultad de adaptación para las métricas M3, M4, M5 y M7. Al contrastar esta información con los datos de rondas jugadas por usuarios reales, se observa que esas métricas son las que obtuvieron en promedio menor puntaje.
- Se puede establecer una “mejora ideal deseada promedio” para alcanzar el centro del rango. Esta mejora ideal se calcula promediando el valor absoluto de la diferencia entre el valor obtenido (cada uno de los 21 casos) y el valor ideal deseado (0.55). Para los resultados obtenidos este valor es de +0.089.

8.5. Resultados y análisis del modelo solo con entrenamiento inicial

En esta sección se presentan las pruebas realizadas con los usuarios virtuales sobre el modelo sin el aprendizaje supervisado, es decir, solo con el entrenamiento por *Back-propagation training* realizado sin interacción con el usuario. Se realizaron tres *TestRun*, cada uno de los cuales con un usuario virtual diferente y ejecutando diez sesiones de 50 rondas.

Para realizar un análisis objetivo de estos resultados se calculó un resumen estadístico donde se encuentra el promedio y desviación estándar para cada métrica de usuario evaluada. Se agrega también el concepto Ronda Estable, el cual se refiere al número de ronda aproximada en la que valor de la métrica dejo de variar significativamente o empieza a mostrar un comportamiento recurrente. En la

Tabla 16 se pueden observar estos resultados.

Tabla 16: Resumen estadístico de resultados solo con entrenamiento inicial

Usuario	Criterio	M1	M2	M3	M4	M5	M6	M7
Desempeño bajo	Promedio	0.57	0.58	0.67	0.44	0.51	0.68	0.70
	Desv. Estand.	0.19	0.18	0.22	0.19	0.20	0.20	0.20
	Ronda estable	2	2	2	2	2	2	2
Desempeño medio	Promedio	0.57	0.58	0.60	0.50	0.35	0.55	0.61
	Desv. Estand.	0.08	0.07	0.10	0.13	0.10	0.11	0.11
	Ronda estable	6	6	6	6	5	6	4
Desempeño alto	Promedio	0.61	0.60	0.68	0.64	0.63	0.59	0.71
	Desv. Estand.	0.12	0.11	0.13	0.09	0.12	0.16	0.12
	Ronda estable	7	7	10	7	7	7	10

TestRun 1: Usuario de Desempeño Bajo

El resultado completo de la primera sesión se puede observar en el Anexo 7.

Analizando los datos encontrados se obtuvieron las siguientes observaciones:

- En la tabla de resultados se puede observar que el usuario comenzó acertando en todos los conceptos, pero rápidamente la aplicación se adaptó y le presento en la segunda ronda un reto más adecuado a sus capacidades. Este comportamiento se repite en las demás sesiones.
- En las siguientes rondas se ve como la aplicación trata de mantener los resultados del usuario dentro del canal de flujo, observándose que las respuestas del usuario se intercalan principalmente entre resultados medios y altos, lo que se percibiría como retos adecuados y retos sencillos respectivamente.
- Los datos estadísticos indican que los resultados, en todas las métricas, se mantuvieron dentro del rango deseado, con una tendencia hacia un resultado alto en las métricas M3, M6 y M7.
- La desviación estándar, similar en todas las métricas nos indican que los resultados no fueron tan homogéneos.

TestRun 2: Usuario de Desempeño Medio

El resultado completo de la primera sesión se puede observar en el Anexo 8.

Analizando los datos encontrados se obtuvieron las siguientes observaciones:

- Para este usuario se observa un descenso de los valores a partir de la ronda 6, demorando ligeramente más que para el usuario anterior. Esto se debe al mejor

desempeño de este usuario que requiere un mayor número de rondas para alcanzar el equilibrio.

- Todas las métricas, excepto M5, están dentro del canal de flujo. Se puede observar que en general los valores son menores que para el usuario anterior, lo cual indica que la aplicación se está adaptando más adecuadamente para este usuario.
- La desviación estándar indica una variación de las métricas menor en comparación a las de usuario anterior, lo cual significa que este usuario se ha enfrentado a retos más uniformes.

TestRun 3: Usuario de Desempeño Alto

El resultado completo de la primera sesión se puede observar en el Anexo 9. Analizando los datos encontrados se obtuvieron las siguientes observaciones:

- Los resultados de este usuario son más uniformes, y tienden a ser altos, aunque sin salirse del rango en 6 de 7 métricas. Solo M7 presenta un valor superior al deseado, indicando que en este concepto no se logró completamente la adaptabilidad deseada.
- Para este usuario se observa un descenso de los valores a partir de la ronda 7, una más que el test anterior, y existe un segundo punto, cerca de la ronda 13, donde los valores vuelven a bajar. Esto es coherente con el aumento de la performance del usuario, lo ya que requiere más rondas para el equilibrio.

Variación de resultados respecto a la IA manual

Los valores que se observan en la Tabla 17 se han conseguido comparando los resultados obtenidos con el entrenamiento inicial de las redes neuronales y los obtenidos con la IA manual. Para la comparación se tomó como objetivo deseado el centro del rango (0.55) y se calculó cuanto se acerca o se aleja la métrica respecto a este punto.

Tabla 17: Variación del modelo con entrenamiento inicial respecto a la IA manual

Usuario	Crit. de variación	M1	M2	M3	M4	M5	M6	M7
Desempeño bajo	Var. Promedio	+0.02	-0.01	-0.05	+0.05	+0.07	-0.04	-0.05
	Var. Desv. Estan.	-0.07	-0.06	-0.06	-0.06	-0.05	-0.05	-0.05
Desempeño medio	Var. Promedio	+0.04	+0.03	+0.04	+0.01	-0.05	+0.03	+0.04
	Var. Desv. Estan.	-0.01	-0.01	-0.01	0.02	0.00	0.00	-0.01
Desempeño alto	Var. Promedio	+0.02	+0.02	+0.02	+0.02	-0.01	+0.03	+0.02
	Var. Desv. Estan.	-0.01	-0.01	-0.01	-0.01	0.00	-0.01	-0.01

Requerimientos computacionales y funcionales del algoritmo

Como se explica en el punto 7.1, se debe verificar cuan adecuadamente se cumplen los requerimientos definidos por Spronk (2005) para que la adaptación dinámica se considere efectiva.

Respecto a los requerimientos computacionales:

- **Velocidad**: La selección del siguiente escenario se realizado muy rápidamente y es imperceptible para el usuario. Esto se debe a que en este modelo solo se ha requerido procesar las métricas de usuario a través de las redes neuronales y no ha habido entrenamiento supervisado, el cual demanda más recursos.
- **Efectividad**: El algoritmo muestra una reacción positiva a las respuestas del usuario logrando ingresar y mantenerse las métricas dentro del rango deseado la mayoría de los casos (Tabla 16). Hubo una mejora en el promedio de las métricas en la mayoría de los casos (71% de los 21 casos), con un acercamiento promedio de +0.023; mientras que el alejamiento se dio en los casos restantes (29%) con un promedio de -0.035 (Tabla 17). Se observa un alejamiento significativo en las métricas de desempeño bajo, por lo que se puede concluir que este modelo no es beneficioso para él, aunque la mejora si es significativa para los otros dos usuarios en 6 de las 7 métricas evaluadas. Además, el promedio general de la variación es positivo, indicando un acercamiento de +0.012; y considerando que el acercamiento promedio ideal necesario de la IA manual es de +0.089, se está cubriendo el 13% del deseado.
- **Robustez**: En cuando a la capacidad de introducir las métricas al rango, se encontró éxito en 19 de los 21 casos (90%) (Tabla 16). Aunque en la mayoría de los casos se cumple con el objetivo propuesto, este modelo muestra una robustez menor a la de la IA manual.
- **Eficiencia**: Se observa que la generación de escenarios adecuados se logra relativamente rápido (Tabla 16), alcanzando la estabilidad aproximadamente en la 2ra ronda para el usuario de desempeño bajo, aproximadamente en la 6ta ronda para el usuario de desempeño medio y aproximadamente en la 7ma ronda para el de desempeño alto, por lo que se considera cumple con el requerimiento.

Respecto a los requerimientos funcionales:

- **Claridad**: Se observa que las salidas de la rede neuronal es coherente, relacionada con el desempeño del usuario y no presentan problemas al ser transformadas en los nuevos valores de atributos.

- **Variedad**: Se observó de acuerdo con la variación en la desviación estándar (Tabla 17) que este modelo redujo la variación en los escenarios presentados al usuario, especialmente para el usuario de bajo desempeño. Analizando los resultados directos de cada ronda (ejemplos en Anexo 7, Anexo 8 y Anexo 9) se encontró que, si bien los resultados de las redes buscaban atraer al usuario a un cierto rango, este por momentos salía de él y podía experimentar resultados variados. La transformación de los resultados de la red a escenarios definidos también jugó un papel significativo ya que no siempre se pueden encontrar escenarios en la base de datos que cumplan con los cambios en los atributos deseados. En resumen, aun en la adaptabilidad se observa variedad en los resultados.
- **Consistencia**: Observando los resultados directos de cada ronda (ejemplos en Anexo 7, Anexo 8 y Anexo 9) se observa que la ronda en la que algoritmo devuelve resultados adaptados es similar para cada métrica dentro de un mismo usuario. Para usuarios diferentes esta "ronda estable" varia; sin embargo, este resultado es parte de la lógica del juego ya que desea que la dificultad suba gradualmente y no de golpe.
- **Escalabilidad**: Al analizar independientemente los promedios de cada métrica para los diferentes desempeños (Tabla 16) se observan valores variados y no se logra identificar un patrón entre estos. Se observa que para el usuario de desempeño medio presenta dificultades para adaptar la métrica M5, mientras que para los otros dos la dificultad está en la métrica M7. Sin embargo, en promedio se logra resultados positivos y similares para los tres niveles de desempeño de usuario.

8.6. Resultados y análisis del modelo con aprendizaje supervisado

En esta sección se presentan las pruebas realizadas utilizando el modelo de aprendizaje supervisado sobre el entrenamiento inicial. Siguiendo la misma metodología usada en la prueba anterior, se realizaron tres *TestRun*, uno con cada usuario virtual y ejecutando 10 sesiones de 50 rondas. En la Tabla 18 se presenta el resumen estadístico de los resultados para cada usuario.

Tabla 18: Resumen estadístico de resultados con aprendizaje supervisado

Usuario	Criterio	M1	M2	M3	M4	M5	M6	M7
Desempeño bajo	Promedio	0.53	0.54	0.63	0.42	0.47	0.64	0.66
	Desv. Estand.	0.19	0.18	0.23	0.18	0.18	0.19	0.21

	Ronda estable	3	3	3	2	3	3	3
Desempeño medio	Promedio	0.54	0.55	0.56	0.46	0.38	0.53	0.59
	Desv. Estand.	0.10	0.10	0.12	0.14	0.13	0.12	0.11
	Ronda estable	6	6	5	6	5	6	6
Desempeño alto	Promedio	0.60	0.60	0.67	0.64	0.64	0.57	0.70
	Desv. Estand.	0.12	0.11	0.14	0.09	0.12	0.17	0.12
	Ronda estable	7	7	7	7	7	7	7

La Tabla 19 muestra la variación respeto al modelo de solo entrenamiento inicial.

Tabla 19: Variación del modelo con aprendizaje supervisado respecto al entrenamiento inicial

Usuario	Crit. de variación	M1	M2	M3	M4	M5	M6	M7
Desempeño bajo	Var. Promedio	**0.00**	**+0.02**	**+0.04**	**-0.02**	**-0.04**	**+0.04**	**+0.04**
	Var. Desv. Estan.	0.01	0.00	-0.01	0.01	0.01	0.00	0.00
Desempeño medio	Var. Promedio	**+0.01**	**+0.03**	**+0.04**	**-0.04**	**+0.03**	**-0.02**	**+0.02**
	Var. Desv. Estan.	-0.02	-0.03	-0.02	-0.01	-0.02	-0.01	0.00
Desempeño alto	Var. Promedio	**+0.01**	**+0.01**	**+0.01**	**+0.01**	**-0.01**	**+0.01**	0.00
	Var. Desv. Estan.	0.00	0.00	-0.01	0.00	0.00	-0.01	0.00

TestRun 1: Usuario de Desempeño Bajo y aprendizaje supervisado

El resultado completo de la primera sesión se puede observar en el Anexo 10.

Analizando los datos encontrados se obtuvieron las siguientes observaciones:

- En todas las métricas se observan resultados menores que los hallados sin el aprendizaje supervisado, con una disminución de aproximadamente 0.04 y logrado mantenerse dentro del rango ideal o canal de flujo. La métrica M7, que se encontraba en el límite del rango, está ahora dentro de él, lo cual es una mejora en comparación a el algoritmo sin aprendizaje supervisado.
- El cambio de las métricas es variado, aunque en promedio positivo. Cuatro de las métricas se han acercado al centro de rango, tres de ellas significativamente (+0.04). Otras dos métricas se han alejado (-0.02, -0.04).
- La desviación presenta valores similares al experimento inicia, lo cual indica que la variación del acierto es similar en ambas pruebas.
- La tabla completa de resultados muestra variación en los atributos del escenario en todas las rondas jugadas indicando escenarios variados.

TestRun 2: Usuario de Desempeño Medio y aprendizaje supervisado

El resultado completo de la primera sesión se puede observar en el Anexo 11.

Analizando los datos encontrados se obtuvieron las siguientes observaciones:

- Para el usuario de desempeño medio también se aprecia reducción (aproximadamente 0.03) en los valores promedio en 6 de las 7 métricas. La excepción es M5, donde hay un incremento de 0.03, el cual resulta positivo ya que acerca la métrica al rango.
- Los resultados son positivos para M1, M2, M3, M5 y M7, ya que en promedio lograron acercarse al centro del rango deseado; mientras que son contrarios para M4 y M6 quienes se alejaron del rango. Sin embargo, todas las métricas que estaban dentro del canal de flujo continúan dentro de él, mientras que M5 aún permanece afuera, aunque más cercana. El resultado, en general, se puede valorar positivamente.
- Observando la tabla completa de resultados se descubrió que, a partir de la ronda nueve, los atributos de los escenarios variaron muy poco unos de otros, lo cual se traduce en escenarios similares para el usuario. Este es un efecto esperado del algoritmo de adaptabilidad, aunque debe ser contrarrestado para mostrar variabilidad al usuario.

TestRun 3: Usuario de Desempeño Alto y aprendizaje supervisado

El resultado completo de la primera sesión se puede observar en el Anexo 12

Analizando los datos encontrados se obtuvieron las siguientes observaciones:

- Es este caso también se observó una disminución en los valores de las métricas, aunque en mucho menor medida (0.01). Todas las métricas que se encontraban en el rango se mantuvieron en él, y M7 que se encontraba fuera logró acercarse al límite superior, lo cual es un punto positivo para el algoritmo.
- Se observa que 5 de las 7 métricas se acercaron ligeramente al centro del rango (0.01) mientras que solo una se alejó en igual proporción.
- La estabilidad se alcanzó visiblemente alrededor de la ronda 7, mientras que los resultados completos muestran un segundo momento de estabilidad entre las rondas 13 y 20.
- Al igual que con el usuario de desempeño medio, se observó en algunos casos que en las últimas rondas se repitieron los mismos atributos, generando escenarios similares en diferentes rondas consecutivas.

Variación de resultados respecto a la IA manual

Los valores que se observan en la Tabla 20 se han conseguido comparando los resultados obtenidos con el aprendizaje supervisado y los obtenidos con la IA

manual. Para calcular la variación se tomó como objetivo deseado el centro del rango (0.55) y se calculó cuanto se acerca o se aleja la métrica respecto a este punto.

Tabla 20: Variación del modelo con aprendizaje supervisado respecto a la IA manual

Usuario	Crit. de variación	M1	M2	M3	M4	M5	M6	M7
Desempeño bajo	Var. Promedio	0.02	0.01	-0.02	0.02	0.03	0.00	-0.01
	Var. Desv. Estan.	-0.06	-0.06	-0.07	-0.05	-0.04	-0.04	-0.05
Desempeño medio	Var. Promedio	0.04	0.06	0.09	-0.03	-0.02	0.01	0.06
	Var. Desv. Estan.	-0.03	-0.03	-0.03	0.01	-0.03	-0.01	-0.01
Desempeño alto	Var. Promedio	0.03	0.02	0.03	0.03	-0.02	0.04	0.02
	Var. Desv. Estan.	-0.01	-0.01	-0.01	-0.01	0.00	-0.02	-0.01

Requerimientos computacionales y funcionales

Al igual que en el aprendizaje fuera de línea se analizaron los requerimientos definidos.

Respecto a los requerimientos computacionales:

- **Velocidad**: Las pruebas mostraron que el aprendizaje supervisado se realiza con mucha rapidez, de modo que resulta imperceptible para el usuario y no afecta el rendimiento de la aplicación. Se considera que cumple adecuadamente este requerimiento.
- **Efectividad**: Respecto al solo entrenamiento inicial, se observó (Tabla 19) como mejoraron los índices de ingreso al rango deseado en la mayoría de los casos (67% de los 21 evaluados), con un acercamiento promedio de +0.023. El alejamiento de las métricas del centro se dio en una proporción menor (24%), pero sin que ningún caso afectado saliera del rango (Tabla 18). La mejora es más significativa para los usuarios bajo y medio, donde se puede calcular un acercamiento promedio de +0.03 en las métricas. El promedio general de acercamiento es de +0.01.

  Respecto a la IA manual (Tabla 20) mejoraron los índices de ingreso al rango deseado en la mayoría de los casos (71%), con un acercamiento promedio de +0.034; mientras que el alejamiento de las métricas del centro se dio en una proporción menor (24%) con un promedio de -0.02. El promedio general de acercamiento es de +0.02, y considerando que el acercamiento ideal promedio necesario de la IA manual es de +0.089, se consideran como positivos estos resultados ya que cubren el 22% del ideal.
- **Robustez**: En cuando a la capacidad de introducir las métricas al rango, se encontró éxito en 20 de los 21 casos (95%). El resultado se puede considerar

positivo por cumplir generalmente con el objetivo propuesto y elevar el resultado anterior sin aprendizaje supervisado. En comparación a la AI manual también hay mejora ya que, aunque se mantiene un caso fuera de rango (M5 medio), los casos en el límite del rango se han reducido de 3 a solamente 1.

- **Eficiencia**: Se observa que la generación de escenarios adecuados se logra relativamente rápido (Tabla 18), alcanzando la estabilidad aproximadamente en la 3ra ronda para el usuario de desempeño bajo, aproximadamente en la 6ta ronda para el usuario de desempeño medio y aproximadamente en la 7ma ronda para el de desempeño alto, por lo que se considera cumple con el requerimiento.

Respecto a los requerimientos funcionales:

- **Claridad**: Se observa que las salidas de la rede neuronal es coherente, relacionada con el desempeño del usuario y no presentan problemas al ser transformadas en los nuevos valores de atributos.
- **Variedad:** Se puede observar (Tabla 19) que el aprendizaje supervisado no tuvo un impacto significativo en la variación de resultados. Revisando los datos directos ronda (ejemplos en Anexo 10, Anexo 11 y Anexo 12) solo en algunos casos se observan varias rondas con poca variación entre ellas, pero en general la distribución de cambios es similar a la del módulo no supervisado. Este resultado, aunque reduce la variedad, tiene mucho sentido ya que la busca ingresar los resultados del usuario al rango y mantenerlos en él, por lo que al encontrar un escenario que cumpla este objetivo no necesita cambiarlo.
- **Consistencia**: Al igual que en el modelo anterior, los resultados directos de cada ronda (ejemplos en Anexo 10, Anexo 11 y Anexo 12) se observa que la ronda en la que algoritmo devuelve resultados adaptados es similar para cada métrica dentro de un mismo usuario.
- **Escalabilidad**: Al igual que en el caso anterior, se observan valores variados para los diferentes desempeños y no se logra identificar un patrón entre estos (Tabla 18). Sin embargo, aunque la dificultad para adaptar M5 del usuario medio se mantiene, la de M7 ya no está presente en para el desempeño bajo y se mantiene solo para el alto. En promedio también se logra resultados positivos y similares para los tres niveles de desempeño.

8.7. Análisis comparativo de requerimientos para aprendizaje supervisado y por redes neuronales.

A continuación en la Tabla 21 se muestra un análisis comparativo entre los resultados encontrados en cada uno de los modelos implementados.

Tabla 21: Análisis comparativo de requerimientos de los modelos implementados

Requerimiento	Modelo solo Red Neuronal	Modelo aprendizaje supervisado
Velocidad	Obtención inmediata de respuesta, imperceptible para el usuario.	Obtención muy rápida de respuesta, imperceptible para el usuario.
Efectividad	Mejora en el 71% de los casos con +0.023. Baja en el 29% de los casos con -0.035. Promedio general positivo con +0.012 o 13% del deseado. El modelo no es beneficioso para el usuario de desempeño bajo.	Mejora en el 71% de los casos con +0.034. Baja en el 21% de los casos con -0.02. Promedio general positivo con +0.02 o 22% del deseado. El modelo afecta a los 3 usuarios de modo similar.
Robustez	Éxito en introducir las métricas al rango en 90% casos. En este requerimiento no mejora respecto a la IA manual.	Éxito en introducir las métricas al rango en 95% casos. Mejora reduciendo los casos límite de 3 a 1 respecto a la IA manual
Eficiencia	Rápida adaptación dependiendo del desempeño del usuario (2da, 6ta y 7ma ronda).	Rápida adaptación dependiendo del desempeño del usuario (3ra, 6ta y 7ma ronda).
Claridad	Salida coherente, relacionada e adecuadamente interpretable.	Salida coherente, relacionada e adecuadamente interpretable.
Variedad	Se observa variedad en los resultados, aunque se conserva en promedio la adaptabilidad. Reducción de variedad en el usuario de desempeño bajo. La variedad depende significativamente de las limitaciones para convertir atributos a escenarios.	En algunos casos se reduce la variedad como efecto secundario de mejorar el ingreso de métricas al rango. La variedad depende significativamente de las limitaciones para convertir atributos a escenarios.
Consistencia	La ronda en la que se logra la adaptabilidad es similar entre cada prueba del mismo usuario. Para usuarios diferentes varia, pero esto es parte del requerimiento de juego de cambiar gradualmente la dificultad.	Al igual que en el otro modelo, la ronda en la que se logra la adaptabilidad es similar entre cada prueba del mismo usuario y varia para usuarios diferentes.
Escalabilidad	Se observan valores variados sin un patrón definido, con dificultad en M5-medio, M7-bajo y M7-alto. Valores en promedio si son similares y positivos en los tres niveles.	Se observan valores variados sin un patrón definido, con dificultad solo en M5-medio, M7-alto. Valores en promedio si son similares y positivos en los tres niveles.

9. CONCLUSIONES Y TRABAJOS FUTUROS

En este apartado final se exponen las conclusiones al Proyecto, así como las recomendaciones para futuros desarrolladores.

9.1. Conclusiones

Teniendo en cuenta los resultados del proyecto, se presentan a continuación las conclusiones para cada uno de los objetivos específicos definidos inicialmente.

9.1.1. Conceptos a evaluar y métricas

Se logró definir los conceptos a evaluar realizando una investigación previa sobre los componentes importantes involucrados, tanto a nivel educativo con ayuda de los profesionales respectivos, como a nivel informático. La definición de estos conceptos se realizó definiendo claramente criterios tanto educativos como informáticos que permitan su elección. También fue importante circunscribirse al escenario de juego elegido (bloques lógicos), sus limitaciones y el requerimiento de adaptabilidad. Realizar esta definición de forma previa al desarrollo del juego en sí mismo demostró ser muy útil, ya que permitió conocer limitaciones y consideraciones importantes al mismo. Se concluye de esto que **la capacidad de adaptabilidad de un juego no debe considerarse solo como un agregado posterior a la definición del mismo**, **sino que, la consideración de esta capacidad previa a la definición del juego, así como la definición clara de los conceptos a evaluar y las métricas implicadas, puede ayudar significativamente a tener una visión más clara de la aplicación a diseñar y explotar así mejor sus capacidades**.

Este trabajo previo al desarrollo también puede considerarse más ordenado y efectivo al momento de realizar una aplicación educativa, pues una aplicación de este tipo requiere tener claramente definido los conceptos que se desea enseñar o reforzar, para un mayor control por parte del educador. Permite también tener una guía clara de lo que se quiere lograr (enseñar), de modo que este objetivo educativo no se pierda a lo largo de las siguientes etapas del desarrollo. Si bien este requisito educativo se puede considerar un requerimiento más, la complejidad del mismo hace necesario un trato especial al cual el trabajo previo parece contribuir.

Se concluye también que **trabajar en base a conceptos a evaluar permite un desarrollo más detallado y profundo de las métricas**, como se observa en el Anexo 2. Comprender claramente lo que se quiere medir resulta crucial para este proceso. Más aun, considerando el problema de relación entre las métricas

expuesto en el punto 4.3.2, la necesidad de hacer objetiva y en cierto modo sistematizar la definición de las métricas resulta evidente. Se considera que la definición de criterios de selección realizada también contribuyó a esto.

9.1.2. Descripción del juego y entorno gráfico

Se logró realizar la interfaz gráfica que permite al usuario interactuar con los bloques lógicos, cumpliendo de este modo con el objetivo propuesto. Al haber considerado las posibles limitaciones de una interfaz en la definición de las métricas no se tuvo problemas en la implementación de las mismas al realizar su desarrollo. Además, se mantuvo presente los requerimientos educativos durante esta parte del desarrollo. Estos dos últimos puntos fueron importantes para el complimiento del objetivo. Se considera entonces que **en una aplicación adaptativa la interfaz influye considerablemente en el módulo de adaptabilidad ya que determina las limitaciones de las métricas a evaluar**. **Tener en cuenta la adaptabilidad previa al desarrollo de la interfaz permite aprovechar al máximo sus capacidades.**

Se desarrolló también paso a paso la secuencia del juego, y se definieron una serie de componentes cuyo objetivo fue manejar los datos y la lógica del juego de forma ordenada. Se presentan en el capítulo respectivo el diagrama de clases, las pantallas de la aplicación y el diagrama de actividades. A través de estos elementos, considerados también resultados del presente objetivo, se puede observar las relaciones entre los diferentes componentes y una mayor comprensión de la tarea de cada uno. A través de los bosquejos realizados antes de implementar la interfaz gráfica se pudo detallar características importantes, como las capacidades de escalamiento y rotación, las cuales dependen en parte de la interfaz realizada. También resultaron importantes al momento de definir la forma de presentación de los resultados, ya que la interfaz, en este punto, influye mucho en la comprensión de los mismos.

Se considera que la secuencia establecida de Sesión, Ronda, Escenario, Grafico, Bloque Activo y Bloque resulto crucial para el buen funcionamiento de la aplicación, ya que conforma una jerarquía clara donde cada componente está encargado de administrar los componentes del cual es contenedor. Permite rastrear errores y realizar cambios de modo más ordenado y simple. Se concluye entonces que **la definición y organización de los componentes de forma modulada y estructurada en una aplicación adaptativa puede contribuir significativamente al éxito de su realización.**

9.1.3. Algoritmo de Evaluación de Figuras

Para cumplir este objetivo se tomó como base un estudio en el que se le presenta a niños un escenario similar al del Proyecto, el cual ayudó a definir los criterios de comparación a usar y los requerimientos previos que los gráficos usados debían cumplir ya que establece parámetros definidos por educadores que permiten realizar una evaluación más adecuada al caso. Se puede concluir que **contribuye positivamente el respetar las áreas de conocimiento involucradas en forma previa a la definición de algoritmos, y se rescata principalmente la importancia que tiene incluir información específica al juego al momento de definir criterios de evaluación**.

Se observó que crear un algoritmo para la evaluación de figuras resulta una tarea en parte compleja pues incluye un componente subjetivo: la figura representa un objeto real (una casa, un árbol) que puede tener diferentes interpretaciones por diferentes usuarios. Teniendo en cuenta esto, se considera necesario **resaltar el carácter objetivo de la prueba** (por ejemplo, indicar claramente al usuario que lo que se busca es igualdad absoluta) **y al mismo tiempo incluir en lo posible este componente subjetivo en el algoritmo**. Esta última parte se realizó en el Proyecto estableciendo una jerarquía de bloques, jerarquía que guarda en cierto modo sentido con el objeto del mundo real (por ejemplo, en el dibujo de un hombre, el tronco es el padre, los brazos son los hijos). De este modo el algoritmo puede realizar una evaluación más “permisiva” y al mismo tiempo objetiva. Se recomienda la **presencia del profesional experto durante la definición de estas características**.

Al tener el usuario libertad total para posicionar o manipular los bloques, no es posible inicialmente saber dónde comienza la estructura que él ha creado, es decir “cuál es el bloque inicial”. Lo que hace el algoritmo usado es **crear jerarquías eligiendo bloques de forma aleatoria. En cada selección realizada, se revisa cuan correcta es la ubicación de cada bloque respecto a su padre, comparándolo con las ubicaciones en la figura original. El acierto o error tiene un valor continuo que se va acumulando de hijos a padres, de modo que al finalizar la evaluación de la distribución se tiene un valor único de error para todo el grafico**. Realizando esto con todas las combinaciones jerárquicas posibles en los bloques se puede determinar cuál de estas es la más apropiada en base al valor resultante obtenido. Este proceso dio resultados positivos con un éxito del 97%, sirviendo adecuadamente como algoritmo de comparación para la distribución. **Es posible entonces implementar un algoritmo de evaluación que**

realice un juicio objetivo sobre la figura resultado del usuario, para lo cual hay que considerar la propia naturaleza de problema presentado, la cual es, en este caso, la presencia de una jerarquía en la figura.

9.1.4. Módulo de Adaptabilidad

Para esto se definió un módulo basado en un conjunto de redes neuronales que reciben como entrada información del desempeño del usuario en cada concepto (métricas) y entrega como salida la acción que debe tomarse en cada uno de los atributos del escenario, ya sea subir, mantener o bajar su dificultad. Las salidas son transformadas a valores de atributos para el escenario siguiente, los cuales luego son buscados en una colección de gráficos para elegir el que más se adapte a estos valores para presentársele al usuario. El módulo planteado demostró ser una forma adecuada de manejar la adaptabilidad ya que, **habiendo definido las métricas de entrada y salida, se puede trabajar independientemente del resto del juego; al mismo tiempo que cumple los requisitos establecidos previamente en la definición de conceptos y diseño de la aplicación**.

Se definió una metodología de aprendizaje que incluye dos partes: un entrenamiento básico usando *Backpropagation* y un entrenamiento en presencia del usuario usando técnicas de refuerzo y aprendizaje supervisado. Para la primera parte, se logró entrenar las redes neuronales usando un conjunto de entrenamiento de 50 casos ejemplo que describen en forma genérica el comportamiento deseado. El entrenamiento a través de *Backpropagation* se realizó de modo exitoso y permitió obtener resultados coherentes con los datos ingresados. Esto permite afirmar que **el entrenamiento por *backpropagation* es posible en escenarios de este tipo, donde las entradas están limitadas dentro de un rango y las salidas representan opciones excluyentes de acciones a tomar**.

Para la segunda parte, se definieron un conjunto de pares evento-acción, que permitió hacer un refuerzo de modo supervisado, es decir, controlando características específicas en los eventos que se toman en cuenta. Previo a esto se establecieron criterios que permitieron definir los eventos con mayor claridad. **Estos criterios fueron obtenidos realizando un análisis minucioso del funcionamiento del juego, de modo que se puedan encontrar relaciones entre los cambios que realizan los datos de entrada y de salida a nivel teórico**. De esto se concluye la importancia de tener clara la lógica de juego y los elementos involucrados al momento de definir los eventos que guiarán el aprendizaje supervisado.

Otro de los resultados de este objetivo es la función selectora de escenarios.

Debido a que la cantidad de escenarios es limitada y en parte predefinida, se necesita una función que determine cuál de los gráficos es más compatible con los valores deseados. Esto se logró con la valoración individual de los atributos de cada grafico dentro de un rango alrededor del valor deseado, aplicando pesos, y eligiendo el de mejor valor acumulado. **La aplicación de este método resulto en propuestas coherentes que indican resulta ser un método eficaz de selección**.

Para demostrar la efectividad de los métodos de aprendizaje establecidos se requiere de realizar pruebas a los mismos, como se verá en el siguiente objetivo. Sin embargo, en esta parte **se ha podido demostrar que es posible implementar una estructura y una metodología de aprendizaje para realizar en una aplicación educativa la adaptación dinámica de la dificultad.**

9.1.5. Integración de la aplicación y pruebas al módulo

La interfaz de usuario, la lógica interna y el módulo de adaptabilidad se integraron sin ninguna complicación, funcionando adecuadamente como conjunto y dando como resultado una aplicación estable, la cual puede ejecutarse en un dispositivo móvil Android a través de la instalación del archivo APK respectivo. Los valores obtenidos en el aprendizaje inicial son obtenidos previamente, guardados y se cargan al iniciar la aplicación. El aprendizaje supervisado si se realiza durante el uso de la aplicación. La aplicación fue instalada un una Tablet Samsung Galaxy A (elegida por su tamaño para facilitar su uso al usuario) y **se pudo apreciar que su funcionamiento exitoso. Con esto se puede concluir que se logró realizar una aplicación educativa con las características requeridas**.

Para verificar la eficacia de la adaptabilidad se definieron usuarios virtuales basados en el desempeño de usuarios reales. Para esto se tuvo que definir las variables que debe tener un usuario y que influyen en su desempeño; obtenidas en concordancia con los componentes evaluados y las posibilidades del juego. **No es suficiente solo obtener el valor final del desempeño, sino valores que describan al usuario y que, ante diferentes dificultades de escenario, reaccionen de manera particular como un usuario real lo haría.** Se registró luego el desempeño de los usuarios reales en base a estas variables y se realizaron promedios para obtener los valores finales de los usuarios virtuales. Del análisis realizado para la creación de usuarios virtuales se pude deducir **la importancia de definir variables para describir las capacidades del usuario, de modo que los usuarios virtuales ofrezcan una respuesta más realista en las pruebas**.

Las pruebas realizadas nos muestran los resultados a lo largo de 10 sesiones de 50

rondas cada una para cada uno de los 3 usuarios virtuales definidos. Las pruebas fueron analizadas en base a dos criterios: el juicio del desarrollador debido a su conocimiento de los requerimientos de la aplicación, y en base a los requerimientos funcionales y computacionales para el balance dinámico de la dificultad de Spronck. Gracias a lo primero se pudo evaluar los resultados en forma cuantitativa, verificando su cercanía al rango deseado de balance o canal de flujo. **Aplicando solo el entrenamiento básico por *Backpropagation* se observaron ya resultados positivos.** Mediante el uso de usuarios virtuales, es posible la adaptabilidad ingresando al rango deseado en el 90% de los casos. Los índices de ingreso al rango respecto a la IA manual mejoraron en la mayoría de los casos (71%), con un acercamiento promedio de +0.012 que representa el 13% de la máxima mejora posible; **por lo que se concluye que este método si permite ser usado como base inicial para un módulo de adaptación de dificultad**.

Se observó también que, luego de realizar el aprendizaje supervisado, los valores obtenidos en las pruebas mejoraron en su mayoría y para todos los usuarios respecto al caso no supervisado. Mediante el uso de usuarios virtuales, es posible obtener un resultado positivo en la adaptabilidad, ingresando al rango deseado en el 95% de los casos. Los índices de ingreso al rango respecto a la IA manual mejoraron en la mayoría de los casos (71%), con un acercamiento promedio de +0.02 que representa el 22% de la máxima mejora posible. **Se concluye también de esto que el aprendizaje supervisado basado en eventos puede ser cuantitativamente eficaz en un módulo de adaptación de la dificultad**.

En cuanto al análisis basado en los requerimientos computacionales y funciones de Spronck, la IA cumplió con la mayoría de ellos, resaltando principalmente la eficacia (el cumplimento de la adaptabilidad), la robustez (eficacia en diferentes escenarios) y la escalabilidad (estar a nivel de desempeño del usuario) en los tres niveles evaluados. Se observó un efecto mixto en la variabilidad; para el desempeño bajo, los escenarios seleccionados, aun luego de alcanzar el equilibro, suelen presentar diferencia uno de otros. Para los desempeños medio y alto, luego de alcanzar el equilibrio, el módulo puede elegir escenarios similares para rondas consecutivas. **Se explica este comportamiento porque este último escenario está totalmente adaptado al usuario y, por ende, como el usuario virtual no varía mucho sus habilidades en la prueba, el escenario tampoco necesita variar**. Este comportamiento es, desde el punto de vista algorítmico, esperado y deseado, pero dentro de los requerimientos del balance dinámico de la dificultad, se necesita que

la IA otorgue variabilidad a sus respuestas para mayor entretenimiento del usuario. Sin embargo, esto puede solucionarse fácilmente estableciendo un rango de aleatoriedad durante la selección del gráfico. En conclusión, **se puede afirmar que el sistema de redes y aprendizaje realiza su trabajo adecuadamente, y que es mediante métodos durante la selección de los escenarios donde se debe manejar este efecto.**

9.2. Trabajos futuros

Un tema de investigación futura importante sería validar si los resultados y recomendaciones propuestas en este proyecto pueden replicarse incluyendo conceptos de aprendizaje más variados dentro del ámbito de la inteligencia espacial o incluso a otros entornos educativos. Sería un importante aporte verificar los resultados en entornos más extensos y complejos. Un primer paso para esto podría ser agregar los conceptos de profundidad y rotación 3D a las características evaluadas.

También se recomienda la creación de usuarios virtuales más detallados, recopilando información de una muestra mayor de usuarios reales y adicionando variables como la cantidad de correcciones hechas o el tiempo entre movimiento y movimiento que no aparentan estar directamente relacionada con los conceptos evaluados pero que quizás podrían refinar el proceso. El módulo de adaptabilidad también podría desarrollarse considerando atributos adicionales que no dependan del gráfico, como son bloques distractores y ratios sobre cuán diferente es un bloque al inicio del juego respecto al que debería ser su estado final. La interfaz y parte de la lógica de la aplicación debe adaptarse para incluir estas características.

Otra área interesante de investigación seria evaluar el impacto que tendría en la eficacia el uso de una función creadora de gráficos, en lugar de una seleccionadora que es la que se ha desarrollado en este proyecto. Prescindir de una base de datos estática de gráficos y pasar a la generación de gráficos en forma dinámica en base a los atributos deseados podría mejorar significativamente los resultados de adaptabilidad, además de brindar un mayor control sobre la evaluación de los mismos. Incluir pruebas individuales a cada uno de los conceptos evaluados también brindaría un mejor entendimiento sobre la efectividad de la adaptabilidad y su impacto en la performance del jugador.

10. BIBLIOGRAFÍA

Academia Android. (16 de abril de 2015). *Presentación del motor de juegos Unity 3D*. Obtenido de Academia Android: http://academiaandroid.com/motor-de-juegos-unity-3d/

Aedo, A. (27 de agosto de 2015). Primera entrevista. (F. Caballero, Entrevistador)

Aedo, A. (27 de Agosto de 2015). Segunda entrevista. (F. Caballero, Entrevistador)

Aguayo, L. V. (s.f.). *Máquinas de enseñanza de Skinner.* Obtenido de http://www.conducta.org/assets/pdf/Valero_Maquinas_ensen%CC%83anza_Skinner.pdf

Ambler, S. W. (13 de 05 de 2006). *The Agile Unified Process (AUP)*. Obtenido de Ambysoft: http://www.ambysoft.com/unifiedprocess/agileUP.html

Andrade, G., Ramalho, G., & Santana, H. (2005). *Challenge-Sensitive Action Selection: an Application to Game Balancing.*

Botero, M. M. (1990). Desarrollo de habilidades de percepción: Una estrategia de investigación en la cátedra. *Investigación y Desarrollo. Universidad del Norte. 1 (1)*, 127-137.

Carro, R. M., Breda, A. M., & Castillo, G. (2002, Mayo). A Methodology for Developing Adaptive Educational Game Environments. *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems* (págs. (pp. 90-99)). Springer Berlin Heidelberg.

Cassis, A. (20 de 10 de 2015). *Aprendizaje Supervisado.* Obtenido de Inteligencia Artificial 101: https://inteligenciaartificial101.wordpress.com/author/alejandrocassis/

Centro de terapia ocupacional Kid Sense. (2013).

Chen, J. (2006). Flow in games. Obtenido de http://www.jenovachen.com/flowingames/thesis.htm

Chen, J. (s.f.). *Flow in Games*. Obtenido de http://interactive.usc.edu/projects/cloud/flowing/

Dickinson, D. (1996). Learning through many kinds of intelligence.

Diezmann, C. M., & Watters, J. J. (2000). Identifying and supporting spatial intelligence in young children. *Contemporary Issues in Early Childhood 1(3)*, págs. 299-313.

Dr. McCaffrey, J. (noviembre de 2015). *Ejecución de prueba: prueba T con C#.* Obtenido de MIcrosoft MSDN Magazine: https://msdn.microsoft.com/es-es/magazine/mt620016.aspx

DreamBox. (2015). *DreamBox*. Obtenido de www.play.dreambox.com

Fogel, D. &. (2000). *How to solve it: Modern Heuristics.* Springer.

Gardner, H. (1983). *Frames of Mind: The theory of Multiple intelligences.* New York: Basic Books.

Garris, R., Ahlers, R., & Driske, J. E. (2002). Games, Motivation, and Learning: A Research and Practice Model. *Simulation & Gaming, 33, 4*, 441-467.

Grissmer, D. W. (2013). Play-based after-school curriculum improves measures of executive function, visuospatial and math skills, and classroom behavior for high risk K1 children. *Paper presented at the meeting of the Society for Research in Child Development.* Seattle.

Hunicke, R., & Chapman, V. (2004). AI for Dynamic Difficulty Adjustment in Games.

Ipanaqué, C. R., & Rojas, M. (2012). *INTELIGENCIAS MÚLTIPLES PERCIBIDAS POR LAS DOCENTES DE LOS ESTUDIANTES DE CINCO AÑOS DE LA RED 01 REGIÓN CALLAO.* Lima, Perú: Universidad San Ignacio de Loyola.

Kaelbling, L. P. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, págs. 237-285.

Knewton. (2014). *Heavy Duty Infrastructure for the Adaptive World.*

Kuo, M.-S., & Chuang, T.-Y. (2015). How gamification motivates visits and engagement for online academic dissemination - An empirical study. *Computers in Human Behavior 55 (2016)*, págs. 16-27.

Kuomon. (2012). *10 REASONS WHY I LOVE GOD HAND*. Obtenido de Screw Attack: http://www.screwattack.com/post/51216735

Lemke, C. (2014). *Intelligent Adaptive Learning: An Essential Element of 21st Century Teaching and Learning.* Obtenido de Dreambox Learning: http://www.dreambox.com/white-papers/intelligent-adaptive-learning-an-essential-element-of-21st-century-teaching-and-learning

Llorca, M. (2009). Hábitos y uso de los videojuegos en la comunicación visual: influencia en la inteligencia espacial y el rendimiento.

Lohman, D. F., Pellegrino, J. W., Alderton, D. L., & Regian, J. W. (1987). Individual

differences in spatial abilities. En S. H. Newstead, *Intelligence and Cognition.* Dordrich:: Kluwer.

Matich, D. J. (2001). Redes Neuronales: Conceptos básicos y aplicaciones. *Cátedra de Informática Aplicada a la Ingeniería de Procesos – Orientación I*.

McCaffrey, J. (14 de abril de 2015). Coding Neural Network Back-Propagation Using C#. VisualStudio magazine.

Mendoza, C. (s.f.). *Mario Kart Comic: The Blue Shell.* Deviant Art, United Kingdom. Obtenido de http://www.deviantart.com/morelikethis/349415150

Newcombe, N. S., & Frick, A. (2010). Early Education for Spatial Intelligence: Why, What, and How. *MIND, BRAIN, AND EDUCATION*, 102-111.

Nielsen, J. (1993). *Usability Engineerin.* San Francisco: Morgan Kaufmann.

Nielsen, J. (1994). Heuristic evaluation. *Usability inspection methods, 17(1)*, págs. 25-62.

OIE, O. d. (2014). *Reporte EduTrends: Aprendizaje y evaluación adaptativos.* Monterrey, Nuevo Leon, Mexico: Tecnologico de Monterrey.

Pfeifer, A. (Marzo de 2009). Creating Adaptive Game AI in a Real Time Continuous Environment using Neural Networks.

Pivec, M. D. (2003, julio). Aspects of game-based learning. *In 3rd International Conference on Knowledge Management*, (págs. 216-225). Graz, Austria.

Rivero, L. C. (1998). *Una Estrategia de Análisis Orientada a Objetos basada en Escenarios: Aplicación en un Caso Real.*

Rogers, E. (1995). A cognitive theory of visual interaction. En N. H. J. Glasgow, *Diagrammatic Reasoning: Cognitive and Computational Perspectives* (págs. 481-500). Cambridge UK: MIT Press.

Rollings, A., & Adams, E. (2003). *On game design.* Indianapolis: New Riders.

Saltsman, A. (05 de 07 de 2009). *Game Changers: Dynamic Difficulty*. Obtenido de Gamasutra: http://www.gamasutra.com/blogs/AdamSaltsman/20090507/83913/Game_Changers_Dynamic_Difficulty.php

Sampayo-Vargas, S., Cope, C. J., He, Z., & Byrne, G. J. (2013). The effectiveness of adaptive difficulty adjustments on students motivation and learning. *Computers & Education 69*, págs. 452–462.

Skinner, B. F. (1970). *Tecnología de la Enseñanza.*

Soflano, M., Connolly, T. M., & Hainey, T. (2015). An application of adaptive games-based learning based on learning style to teach SQL. *Elsevier: Computers & Education 86*, 192e211.

Spronck, P., Ponsen, M., & Sprinkhuizen-Kuyper, I. (2005). Adaptive game AI with dynamic scripting. *Machine Learning, 63(3)*, 217-248.

Sunkel, G., & Trucco, D. (2010). TIC para la educación en América Latina. Riesgos y oportunidades. *Serie Políticas Sociales, (167)*.

Terleck, M. S., Newcombe, N. S., & Little, M. (2008). Durable and Generalized Effects of Spatial Experience on Mental Rotation: Gender Differences in Growth Patterns. *APPLIED COGNITIVE PSYCHOLOGY*, 996–1013.

Torres, S. D. (19 de Setiembre de 2015). Primera entrevista: Temas educativos en relación al uso de bloques lógicos. (F. A. Caballero, Entrevistador)

Unity Technologies. (24 de Octubre de 2015). *Unity*. Obtenido de Unity3D: https://unity3d.com/es/unity

Uttal, D. H. (2013). The malleability of spatial skills: A metaanalysis analysis of training studies. *Psychological Bulletin*, 139, 352–402.

Verdine, B. N., Irwin, C. M., Golinkoff, R. M., & Hirsh-Pasek, K. (2014). Contributions of executive function and spatial skills to preschool mathematics achievement. *Journal of Experimental Child Psychology 126*, 37–51.

Visual Studio. (25 de Agosto de 2016). *Visual Studio Community*. Obtenido de Visual Studio: https://beta.visualstudio.com/vs/community/

Wleklinski, N. (2011). *Skinner's Teaching Machine and Programmed Learning Theory.* University of Illinois Board of Trustees.

Yang, T.-C., Hwang, G.-J., & Jen-Hwa , S. (2012). Development of an Adaptive Learning System with Multiple Perspectives based on Students' Learning Styles and Cognitive Styles. *Educational Technology & Society, 16 (4)*, 185–200.

Yilmaz, H. B. (2009). On the development and measurement of spatial. *International Electronic Journal of Elementary Education, Vol.1, Issue 2*, (83-96).