UNIVERSIDAD FINIS TERRAE

FACULTAD DE INGENIERÍA

INGENIERÍA CIVIL EN INFORMÁTICA Y TELECOMUNICACIONES

SERIOUS GAME PARA EL APRENDIZAJE DE LA PROGRAMACIÓN

BASTIAN ALONSO RIVAS SOTO

Trabajo de título presentado a la Facultad de Ingeniería de la Universidad Finis Terrae, para optar al Título de Ingeniero Civil en Informática y Telecomunicaciones

Profesor guía: Darío Rojas Diaz

Santiago, Chile

2025

INDICE DE CONTENIDOS

ÍNDICE DE TABLAS E ILUSTRACIONES

RESUMEN

Este proyecto desarrolla un Serious Game, diseñado para enseñar fundamentos de programación a estudiantes universitarios de primer año sin experiencia previa. El juego utiliza el framework MDA (Mecánicas, Dinámicas y Estéticas), que combina elementos lúdicos y pedagógicos para ofrecer una experiencia de aprendizaje interactiva y progresiva.

El prototipo permite a los usuarios aprender conceptos clave como secuencialidad, algoritmos básicos, condicionales y bucles mediante programación con bloques visuales. La estructura del juego se compone de niveles de dificultad creciente que aseguran la comprensión antes de avanzar. Las pruebas iterativas realizadas validaron su potencial para mejorar la adquisición de habilidades en programación.

A pesar de sus logros, el prototipo enfrenta limitaciones, como su enfoque exclusivo en fundamentos de programación, restricciones técnicas y la ausencia de métricas claras para evaluar su efectividad. Futuras iteraciones del proyecto se orientan a la incorporación de contenidos avanzados, mecanismos de evaluación formal y la integración con plataformas educativas.

Palabras clave: Serious Game, programación, enseñanza interactiva, gamificación.

ABSTRACT

This project presents a Serious Game, aimed at teaching programming fundamentals to first-year university students with no prior experience. The game leverages the MDA framework (Mechanics, Dynamics, and Aesthetics) to integrate engaging and pedagogical elements, offering an interactive and progressive learning experience.

The prototype allows users to grasp key programming concepts such as algorithms, conditionals, and loops through visual block-based programming. It features a level-based structure with increasing difficulty to ensure concept mastery before progression. Iterative testing confirmed its potential to enhance programming skill acquisition effectively.

Despite its achievements, the prototype faces limitations, including a focus on basic programming, technical constraints, and a lack of formal evaluation metrics. Future iterations aim to expand advanced content, introduce assessment mechanisms, and integrate with educational platforms to improve its educational impact.

Keywords: Serious Game, programming, interactive learning, gamification.

CAPÍTULO I: INTRODUCCIÓN

1.1 Descripción

Este documento presenta el desarrollo de un prototipo de *Serious Game* para el aprendizaje de los fundamentos de la programación, específicamente diseñado para estudiantes universitarios de primer año sin experiencia previa en el área. La programación, en el contexto de este trabajo, se refiere a los conceptos básicos que se imparten en los primeros semestres de las carreras de ingeniería o afines, tales como la secuencialidad, el uso de variables, los condicionales y los ciclos iterativos.

En el contexto actual de la educación superior, la programación se ha convertido en una habilidad importante para estudiantes de diversas disciplinas, especialmente en carreras relacionadas con la informática y la realidad virtual. Sin embargo, a pesar de su importancia, muchos estudiantes ingresan a la universidad sin una formación previa en programación, motivados por la alta demanda de profesionales en el área IT como indica (Huumit, 2024), lo que representa un desafío significativo en su primer año académico. Este fenómeno se debe, en gran parte, a que las bases curriculares de la educación media no incluyen de manera obligatoria la enseñanza de la programación, dejando esta formación como un contenido electivo que depende de la infraestructura y recursos disponibles en cada institución. También existe el caso de que no se imparte ninguna asignatura relacionada a esto, haciendo que la brecha sea aún más grande.

Ante esta situación, surge la necesidad de desarrollar métodos efectivos que permitan a los estudiantes aprender programación desde cero en un ambiente que les resulte accesible y motivador. La propuesta de este trabajo se centra en el desarrollo de un prototipo de Serious Game, ya que “La relación entre los juegos digitales y la movilización de estrategias de aprendizaje cognitivas y metacognitivas merece atención y requiere investigaciones que contribuyan a la comprensión de cómo estas estrategias pueden favorecer a los procesos de enseñanza y aprendizaje” (Pimentel et al., 2022). Este proyecto está orientado al aprendizaje de la programación para estudiantes de primer año de universidad. Este enfoque se fundamenta en la utilización de elementos lúdicos y la gamificación como estrategia educativa, aprovechando la

familiaridad de los estudiantes con los videojuegos y su potencial para facilitar el aprendizaje de conceptos complejos de manera progresiva y atractiva, ya que según los datos obtenidos por (Pimentel et al., 2022) el porcentaje de las personas jugadoras de entre 18 y 20 años es del 48,23%.

Este trabajo se basa en un enfoque ágil para el desarrollo del prototipo, con un énfasis en la iteración continua y la retroalimentación de usuarios reales. Además, se utilizarán principios de diseño de juegos serios para garantizar que el producto final no solo sea educativo, sino también atractivo y motivador para los estudiantes. La necesidad de que el software funcione de manera offline, así como la restricción de utilizar únicamente el motor Unity para su desarrollo limita la exploración de otros motores de juego.

La estructura del trabajo se organiza en varios capítulos. El primero presenta una introducción que contextualiza la importancia de la programación en la educación superior y revisa trabajos previos en el área de Serious Games y simulación de robots. A continuación, se describe el proceso de diseño y desarrollo del prototipo, seguido de los resultados obtenidos en las pruebas de testing con estudiantes. Finalmente, se discuten las conclusiones y posibles mejoras futuras, junto con una reflexión sobre el impacto educativo del proyecto.

1.2 Problemática

En el sistema educativo de enseñanza media, aunque existen bases curriculares que sugieren la inclusión del pensamiento computacional y la programación, estos contenidos no son obligatorios.

> “Del total de respuestas, el 54% de los estudiantes señala que tiene experiencia previa en programación. En la Fig. 9 se puede apreciar que el 26% tiene experiencia adquirida en la escuela primaria o secundaria, un 3% en otra carrera universitaria o terciaria y un 25% indica que ha estudiado programación por su cuenta.” (Dapozo et al., 2022)

En la práctica, la enseñanza de la programación suele quedar relegada a asignaturas electivas, dependiendo de la infraestructura y recursos de cada institución. Como resultado,

muchos estudiantes ingresan a la universidad sin haber recibido una formación formal en programación, lo que les genera dificultades al enfrentarse a cursos de programación desde el primer año.

La falta de experiencia previa en programación provoca una alta tasa de reprobación en estas asignaturas, especialmente en carreras que dependen fuertemente de estas habilidades, como informática y realidad virtual. Esto es preocupante, si tenemos en cuenta las estadísticas obtenidas por (Lázaro Alvarez, 2020), qué predice que el porcentaje de personas que se dan de baja es del 20% y los que repiten el primer año el 18% (p.20). Aunque existen diversas herramientas y métodos de apoyo, como talleres psicopedagógicos, tutorías y plataformas en línea, estas no siempre son efectivas para quienes inician desde cero, ya que muchas están orientadas a usuarios con conocimientos previos.

El problema central radica en la necesidad de un enfoque educativo que facilite el aprendizaje de la programación desde un nivel básico, utilizando métodos que conecten con las experiencias y habilidades previas de los estudiantes. La propuesta de solución se centra en el desarrollo de un "Serious Game", una plataforma gamificada que enseña programación a través de la simulación de escenarios robóticos, permitiendo a los estudiantes aprender de manera progresiva y adaptada a su nivel de conocimiento inicial.

1.3 Objetivos

Es importante contar con mecanismos adecuados para medir y evaluar el éxito de un proyecto, ya que esto permite verificar si se han alcanzado las metas establecidas y si el proyecto ha cumplido con sus objetivos. Para lograr esto, es fundamental definir claramente qué es lo que se espera lograr y establecer criterios específicos de éxito. Por eso es importante definir los siguientes puntos:

1.3.1 Objetivo general

Desarrollar un prototipo de Serious Game basado en el género de gestión de recursos para el aprendizaje de los fundamentos de la programación en primer año de universidad.

1.3.2 Objetivos específicos

1. Definir de unidades temáticas de aprendizaje de programación por especialistas para determinar diseño de niveles y forma de progresión.
2. Diseñar de juego mediante framework MDA para determinar fases de implementación y recursos visuales y sonoros.
3. Implementar de prototipo de juego en el motor Unity.
4. Realizar Testing con Estudiantes para comprobar el correcto funcionamiento del prototipo.

1.4 Alcances y limitaciones

Es fundamental establecer claramente los alcances y limitaciones que definirán el marco de trabajo y las expectativas sobre los resultados obtenidos. Estos elementos no solo permiten acotar el objeto de estudio, sino también comprender las restricciones inherentes que influyen en el proceso de investigación y desarrollo.

1.4.1 Alcances

El alcance de este proyecto se enfoca exclusivamente en la asignatura de programación para estudiantes de primer año de universidad, sin abordar contenidos más avanzados que puedan pertenecer a cursos de niveles superiores. El prototipo de Serious Game se diseñará para ser utilizado de forma individual, sin incorporar funcionalidades multijugador, lo que permitirá a los estudiantes aprender a su propio ritmo y sin la presión de la competencia con otros. Además, la investigación se centrará en el diseño de mecánicas y progresión del juego que sean pedagógicamente efectivas, dejando en un segundo plano aspectos como la estética y los gráficos avanzados, para garantizar que el foco esté en el aprendizaje de los conceptos de programación.

1.4.2 Limitaciones

Este trabajo se verá condicionado por varias limitaciones clave. En primer lugar, el software será desarrollado para funcionar de manera offline, lo que restringirá la implementación de características que dependen de la conectividad a internet, como actualizaciones en tiempo real o interacción en línea con otros usuarios. Además, se utilizará exclusivamente el motor de desarrollo Unity, lo que limita la exploración de otros motores o herramientas que podrían ofrecer ventajas técnicas o educativas, pero que no serán considerados debido a la experiencia previa en Unity. Por último, aunque se priorizará la funcionalidad y el diseño de la jugabilidad, la estética del juego no será un área central de estudio, lo que podría limitar la experiencia visual del usuario, enfocándose más en la efectividad educativa que en la presentación gráfica del prototipo.

CAPITULO II: ESTADO DEL ARTE

El estado del arte revela que tanto los Serious Games como los simuladores de robots son herramientas efectivas para la enseñanza de la programación, aunque se diferencian en su enfoque y aplicación. Los Serious Games se centran en el aprendizaje por acción, ofreciendo un contexto lúdico y narrativo para enseñar conceptos abstractos de programación de manera tangible (Gamelearn, 2024). La motivación del jugador se basa en recompensas, desafíos y una experiencia inmersiva, lo que reduce la frustración común entre principiantes (Treviño, 2025).

En contraste, las plataformas de simulación de robots priorizan la precisión técnica y la representación realista. Su principal objetivo es que el estudiante programe los movimientos exactos de un robot, lo que se vuelve una experiencia menos lúdica. La motivación en estos simuladores surge de la resolución lógica y exitosa de un problema técnico, en lugar de una narrativa de juego, lo que puede ser menos atractivo para estudiantes sin conocimientos previos (Sánchez et al, 2017). Si bien son excelentes para la enseñanza de la robótica y la ingeniería, su enfoque a menudo requiere un nivel de abstracción más alto, lo que puede ser un obstáculo para quienes se inician en la programación.

La enseñanza tradicional de la programación se ha centrado históricamente en un enfoque dirigido por el instructor, donde el énfasis principal está en la sintaxis del lenguaje y el diseño de algoritmos a través de ejercicios en papel o pizarrón (Trejos, 2018). Este método, si bien es efectivo para un público ya familiarizado con la lógica, presenta desafíos significativos para los estudiantes de primer año sin conocimientos previos. La alta abstracción de los conceptos, la falta de retroalimentación inmediata y la desconexión entre la teoría y la práctica en un entorno de clases expositivas suelen generar frustración y altas tasas de deserción (Checa, 2011).

Tabla 1

Comparación de los distintos métodos

Criterio de Comparación	Métodos Tradicionales	Otros Serious Games	Simulaciones de Robótica	Tu Propuesta
Público Objetivo	Todo público.	Amplio, niños a adultos, principiantes.	Personas interesadas en la robótica.	Principiantes universitarios sin experiencia.
Enfoque de Aprendizaje	Abstracto	Abstracto	Físico y abstracto	Visual y abstracto
Nivel de Abstracción	Alto	Intermedio	Alto	Bajo
Retroalimentación	Baja.	Inmediata y visual.	Inmediata y visual.	Inmediata, visual.
Justificación del Enfoque	Dar la capacidad de implementar código para un fin.	Ofrecer una experiencia entretenida.	Ayudar a simular los entornos físicos en los que se puede tener un robot.	Ofrece un contexto visual y educativo para que los estudiantes apliquen y vean el resultado de su código.

La tabla 1 es de tipo comparativa que muestra que este proyecto podría llenar un vacío en las herramientas de aprendizaje de la programación para estudiantes universitarios sin experiencia previa. Mientras que los Métodos Tradicionales pueden ser abstractos y tener una alta curva de aprendizaje, y las Simulaciones de Robótica se centran en la física y requieren conocimientos más avanzados, este proyecto se posiciona como una alternativa intermedia. Se diferencia de Otros Serious Games al utilizar una ambientación de robótica, lo que permite a los estudiantes aplicar conceptos abstractos a un contexto visual y tangible. De esta manera, el juego logra un bajo nivel de abstracción, ofreciendo una retroalimentación inmediata y un entorno de aprendizaje atractivo y accesible, diseñado específicamente para los desafíos que enfrentan los principiantes.

2.1 Videojuegos Educativos

Los videojuegos educativos son una herramienta efectiva para implementar pedagogías activas. De acuerdo con Rodriguez Jimenez y Garcia Pinilla (2020), estas metodologías, que exigen un seguimiento y una retroalimentación continua, se vuelven más efectivas gracias a las posibilidades inherentes a los videojuegos, como el aprendizaje interactivo. La gamificación es otro pilar clave, utilizando elementos de juego como recompensas, logros y niveles para motivar y mantener el interés del jugador, integrando el aprendizaje de manera atractiva. Además, estos

juegos se apoyan en la teoría de la zona de desarrollo próximo, proponiendo desafíos que están ligeramente por encima del nivel actual de competencia del jugador, pero aún dentro de sus capacidades, lo que promueve el aprendizaje continuo. Algunos de los *Serious Game* destacados son:

1.- Hacknet es un videojuego que simula el entorno de un hacker. Aunque su enfoque principal no es enseñar a programar desde cero, ofrece una experiencia inmersiva donde los jugadores deben resolver problemas usando comandos en una interfaz de línea de comandos. Esta aproximación a la programación resulta atractiva para aquellos interesados en aprender a programar de manera autodidacta, aunque requiere ciertos conocimientos previos.

2.- 7 Billion Humans y Human Resource Machine, desarrollados por Tomorrow Corporation, son dos juegos que enseñan conceptos de programación a través de la resolución de puzzles. En ambos juegos, los jugadores deben organizar secuencias de instrucciones para que los personajes en pantalla cumplan con ciertas tareas. Estos títulos son efectivos para introducir conceptos básicos de programación, como ciclos, condiciones, y manejo de memoria, en un entorno lúdico y accesible.

3.- While True: Learn () es un juego que explora el aprendizaje automático y la programación en un entorno de simulación. El jugador asume el papel de un programador que debe desarrollar y entrenar modelos de inteligencia artificial. Aunque el juego se centra más en la lógica y el aprendizaje automático, introduce a los jugadores en la programación y la resolución de problemas complejos, haciéndolo accesible para aquellos sin experiencia previa.

4.- Code Battle es un juego educativo diseñado para enseñar conceptos de programación a través de desafíos competitivos. En este juego, los jugadores escriben código para controlar unidades o personajes dentro de un entorno virtual y deben competir contra otros jugadores o la IA para lograr objetivos específicos, como recolectar recursos o derrotar enemigos.

2.2 Plataformas de Simulación de Robots

Las plataformas de simulación de robots son entornos valiosos para la enseñanza de la programación y la robótica, ya que permiten a los estudiantes experimentar con la programación en un entorno virtual antes de interactuar con robots físicos. Estas plataformas fomentan la colaboración, pero a menudo carecen de los elementos lúdicos que motivan a los estudiantes a seguir practicando y "se promueven otro tipo de habilidades muy importantes, y que además son de un carácter más social y comunicativo, como pueden ser el trabajo en equipo y la expresión oral" (Ángel-Díaz et al., 2020), por eso se describirán algunos de los más importantes:

1.- Open Roberta es una de las más destacadas. Desarrollada por Fraunhofer IAIS, esta plataforma permite a los usuarios programar robots virtuales y reales usando un entorno gráfico basado en bloques, similar a Scratch. Open Roberta es accesible para estudiantes de diferentes niveles, ofreciendo una introducción progresiva a la programación y robótica sin requerir experiencia previa.

2.- VEXcode VR es otra plataforma relevante que ofrece simulaciones de robots programables en un entorno virtual. Esta herramienta se utiliza ampliamente en entornos educativos para enseñar conceptos de robótica y programación a través de la resolución de problemas y la ejecución de tareas específicas con los robots simulados.

3.- Tinkerbots es una plataforma que combina kits de construcción robótica con una aplicación que permite a los usuarios programar los movimientos y comportamientos de sus robots. Aunque está más orientada hacia la robótica física, la simulación en su aplicación permite a los usuarios experimentar con la programación antes de ejecutar sus códigos en robots reales.

4.- RoboMind es una plataforma que enseña los conceptos básicos de programación y robótica utilizando un robot virtual. A través de esta herramienta, los estudiantes aprenden a programar en un lenguaje sencillo y orientado a objetos, permitiéndoles observar cómo sus códigos afectan el comportamiento del robot en un entorno simulado.

Estas plataformas y videojuegos destacan la importancia de integrar la programación en la educación de manera accesible y lúdica, ofreciendo a los estudiantes herramientas poderosas para aprender habilidades esenciales en el mundo actual.

Los Serious Game pueden ser efectivos en la enseñanza de la programación a través del juego, pero también presentan ciertas limitaciones. Muchos de ellos están orientados a usuarios con algún conocimiento previo o están más enfocados en la práctica y perfeccionamiento de habilidades que en la enseñanza desde cero. Además, su contenido puede ser limitado en cuanto a la variedad de conceptos que abordan, o bien, la interfaz y la mecánica del juego pueden no ser lo suficientemente intuitivas para todos los niveles de aprendizaje.

Las plataformas de simulación de robots ofrecen un entorno valioso para la enseñanza de la programación y la robótica, permitiendo a los estudiantes experimentar con la programación de robots en un entorno virtual antes de interactuar con robots físicos. Estas plataformas permiten la simulación de diversas acciones y comportamientos de robots, proporcionando una forma accesible y flexible de aprender conceptos complejos sin los costos y limitaciones asociados con el hardware real. Sin embargo, también presentan algunas restricciones. La principal desventaja es que, a pesar de ofrecer una simulación realista, no siempre pueden replicar la complejidad y los problemas imprevistos que surgen al trabajar con robots físicos. Además, la experiencia puede ser menos tangible, lo que podría afectar la comprensión profunda de los aspectos prácticos de la robótica y carece de incentivos para la motivación de los estudiantes.

A diferencia de los ejemplos mencionados, el "Serious Game" propuesto buscará llenar el vacío existente al enfocarse específicamente en estudiantes de primer año de universidad que no tienen experiencia previa en programación. Este juego tendrá un enfoque altamente progresivo y gamificado, diseñado para enseñar desde los fundamentos básicos de la programación, utilizando robots virtuales que los estudiantes programarán en un entorno de simulación. El objetivo es crear una herramienta educativa que combine los elementos exitosos de los juegos anteriores, pero con una accesibilidad y enfoque didáctico que permita a los principiantes absolutos avanzar en su aprendizaje de manera efectiva y motivadora.

CAPITULO III: MARCO TEORICO

El desarrollo de este Serious Game se fundamenta en un marco teórico que aborda la problemática educativa de la enseñanza de la programación y evalúa la viabilidad de una solución lúdica e interactiva. Este apartado se centra en los conceptos clave que justifican el proyecto, a saber, el uso de la gamificación para la educación superior, la relevancia del pensamiento computacional y los desafíos específicos del aprendizaje de la programación para estudiantes principiantes.

3.1 La Gamificación

La gamificación consiste en la aplicación de elementos de juego (como puntos, insignias o tablas de clasificación) en contextos que no son juegos (como un salón de clases) para aumentar la motivación y el compromiso de los participantes (Viñas, 2022). Esta se ha consolidado como una estrategia didáctica innovadora para el entorno educativo. Su objetivo es aplicar elementos y mecánicas de juegos en contextos no lúdicos para mejorar la motivación, el compromiso y el aprendizaje (Viñas, 2022). En la educación superior, la gamificación ha demostrado ser especialmente efectiva en campos como la ingeniería y la arquitectura, donde las simulaciones y la práctica activa son fundamentales (Ojeda y Zaldívar, 2023).

3.2 Los Serious Games en la Educación Superior

El uso de Serious Games en la educación superior representa una evolución significativa de las metodologías de enseñanza tradicionales. A diferencia de la gamificación, que simplemente aplica elementos de juego en el aula, un Serious Game es un videojuego completo diseñado con un propósito educativo explícito. Estos juegos han demostrado ser herramientas valiosas para el aprendizaje de habilidades técnicas y la adquisición de conocimientos complejos en un entorno seguro y atractivo (Gamelearn, 2024).

En el contexto de la programación, los Serious Games son particularmente efectivos porque abordan la alta abstracción inherente al código. Permiten a los estudiantes interactuar con conceptos teóricos de manera visual y tangible, como se evidencia en tu prototipo con los bloques

de programación y la representación de los robots. Las principales ventajas de este enfoque incluyen:

1. Motivación y compromiso: Los Serious Games captan y mantienen el interés de los estudiantes a través de una experiencia lúdica y envolvente. La sensación de progreso, el desafío y la recompensa incentivan a los jugadores a seguir aprendiendo.
2. Aprendizaje activo y práctico: En lugar de memorizar, los estudiantes aprenden haciendo. Al enfrentarse a problemas en el juego, deben aplicar conceptos como algoritmos y condicionales para encontrar soluciones, lo que fomenta el pensamiento crítico y la resolución de problemas en tiempo real.
3. Retroalimentación inmediata: El juego proporciona un *feedback* instantáneo y visual sobre la validez de las instrucciones programadas. Si un bloque de código funciona, el robot lo ejecuta; si hay un error, el jugador lo ve de inmediato y puede corregirlo, un proceso iterativo que es crucial para el aprendizaje de la programación (Pimentel et al., 2022).
4. Entorno seguro: Los jugadores pueden experimentar con diferentes estrategias y cometer errores sin las consecuencias negativas que tendrían en una clase tradicional, como una calificación baja. Esto reduce la ansiedad y la frustración que a menudo se asocian con el aprendizaje de la programación para principiantes.

Este enfoque no solo ayuda a los estudiantes a comprender los fundamentos de la programación, sino que también los prepara para los desafíos del mundo real, mejorando su capacidad para aplicar la lógica en contextos complejos.

3.3 El Pensamiento Computacional y su Relevancia

El pensamiento computacional es un proceso mental esencial para la era digital, que implica la resolución de problemas de manera estructurada y eficiente. Se basa en cuatro pilares fundamentales (Educo, 2024):

1. **Descomposición:** Dividir un problema complejo en partes más pequeñas y manejables.
2. **Reconocimiento de patrones:** Identificar similitudes y tendencias para aplicar soluciones conocidas.

3. **Abstracción:** Centrarse en los detalles importantes e ignorar los irrelevantes.
4. **Algoritmos:** Crear una secuencia de pasos lógicos para resolver el problema.

Las bases curriculares de la educación media no incluyen la enseñanza obligatoria de la programación, por lo que muchos estudiantes ingresan a la universidad sin haber desarrollado estas habilidades. Por lo tanto, una herramienta que fomente el pensamiento computacional desde cero es de gran valor. El juego, al requerir que los jugadores descompongan tareas y construyan algoritmos para resolver desafíos, actúa como una plataforma ideal para este fin.

CAPITULO IV: METODOLOGÍA

Para el desarrollo del Serious Game, se han considerado varias metodologías antes de decidir la metodología ágil. A continuación, se presenta un análisis de las alternativas evaluadas y la justificación de la elección final.

4.1 Alternativas consideradas

Es importante tener en cuenta las diferentes opciones que se tienen para ver si algunas de ellas se ajustan de mejor manera a la forma que se tendrá de trabajar. por eso se presentarán las diferentes metodologías que se discutieron:

1.- La metodología en cascada es un enfoque secuencial donde cada fase del desarrollo debe completarse antes de pasar a la siguiente. Se sigue un orden estricto de etapas, desde la definición de requisitos hasta la implementación y pruebas. Algunas de las ventajas son la estructura clara y fácil de gestionar, ideal para proyectos con requisitos bien definidos y estables. Por otro lado, las desventajas son la falta de flexibilidad para adaptarse a cambios durante el desarrollo, lo que puede ser problemático si los requisitos evolucionan.

2.- La metodología iterativa, el desarrollo se divide en pequeños ciclos (iteraciones), donde se construye y mejora el producto en cada ciclo basado en la retroalimentación. Las ventajas son que permite realizar ajustes basados en la retroalimentación temprana y continua, mejorando la calidad del producto final. Alguna desventaja es que puede requerir más tiempo y recursos para gestionar las iteraciones y las revisiones frecuentes.

3.- La metodología ágil se centra en la entrega incremental y continua de funcionalidades. Prioriza la colaboración con el cliente y la capacidad de adaptarse a los cambios rápidamente. Las ventajas son alta flexibilidad, enfoque en la colaboración y la retroalimentación continua, permite adaptarse a cambios en los requisitos y las desventajas son que requiere un alto grado de comunicación y colaboración, puede ser menos estructurada en comparación con otros enfoques.

Se ha decidido utilizar la metodología ágil SCRUM para el desarrollo del prototipo educativo debido a las siguientes razones, explicadas con más detalle en (Atlassian, 2025.):

1.- Flexibilidad y Adaptabilidad: La naturaleza del proyecto implica que los requisitos pueden evolucionar a medida que se recibe retroalimentación de los estudiantes y profesores. La metodología ágil permite adaptarse rápidamente a estos cambios, asegurando que el prototipo evolucione de acuerdo con las necesidades de los usuarios.

2.- Entrega Incremental: La entrega continua de funcionalidades permite que los estudiantes comiencen a interactuar con el prototipo desde las primeras etapas del desarrollo, lo cual es crucial para validar las hipótesis y hacer ajustes necesarios.

3.- Retroalimentación Continua: Al final de cada iteración, se obtiene retroalimentación de los usuarios, lo que permite realizar mejoras continuas y asegurar que el prototipo cumpla con las expectativas de los usuarios.

Por eso, para el desarrollo del prototipo de Serious Game basado en simulación de robots, se emplea la metodología ágil SCRUM, la cual facilita una gestión eficiente del proyecto mediante ciclos iterativos y la adaptación constante a los requerimientos y cambios. Esta metodología es adecuada para proyectos de desarrollo de software en los que es crucial ajustar y mejorar continuamente el producto según el feedback recibido y las necesidades emergentes.

4.2 Herramientas

Es importante tener en cuenta las herramientas que se utilizaran para el desarrollo e intentar dejar en claro en que pueden ayudar para el proceso, aquí una lista de las plataformas que se usarán:

1.- Unity se utilizará como la herramienta principal para el desarrollo del prototipo de Serious Game. Esta plataforma es ampliamente conocida en la industria de los videojuegos por su flexibilidad y robustez en la creación de experiencias interactivas en 2D y 3D. Unity ofrece una amplia gama de características que facilitan la implementación de mecánicas de juego,

simulaciones y gráficos de alta calidad, además de ser compatible con múltiples dispositivos. Su motor gráfico y capacidad para integrar scripts en C# permiten un desarrollo ágil y eficiente, especialmente en la creación de entornos interactivos y simulaciones, como el diseño de un juego educativo basado en robots. Además, al tratarse de un entorno conocido y que se maneja con facilidad, permite concentrar los esfuerzos en la creación del contenido educativo y en la optimización de la jugabilidad, en lugar de aprender un nuevo motor, lo que garantiza un desarrollo más rápido y eficiente.

2.- Notion es utilizado para la planificación y seguimiento de las tareas dentro de cada sprint. Trello permitirá organizar el trabajo en tableros visuales, facilitando la gestión de tareas, la asignación de prioridades y la monitorización del progreso del proyecto. Cada objetivo específico se desglosa en tareas menores que se organizan en tarjetas dentro de Trello, lo que permitirá un seguimiento claro y efectivo del avance.

4.3 Aplicación del Framework MDA

Según Putra y Yasin (2021), el framework MDA es clave en el desarrollo de experiencias de juego significativas. Este será fundamental en el diseño y desarrollo del Serious Game. Este enfoque se basa en tres componentes interrelacionados que influyen en la experiencia del jugador.

1.-Mechanics (Mecánicas): Se definirán las reglas y sistemas que componen el juego. En el caso del prototipo, esto incluirá el diseño de las mecánicas de programación y la interacción del usuario con el simulador de robots. Las mecánicas se centrarán en cómo los jugadores interactúan con el entorno y los robots, incluyendo la resolución de puzzles y la ejecución de comandos de programación.

2.-Dynamics (Dinámicas): Se abordará cómo las mecánicas afectan el comportamiento del juego, cómo los jugadores interactúan entre sí y con el entorno del juego. Se evaluará cómo las reglas y sistemas se manifiestan en la experiencia del jugador, lo que ayudará a ajustar la jugabilidad y la progresión del aprendizaje.

3.-Aesthetics (Estética): Aunque el enfoque principal está en la jugabilidad, también se prestará atención a la presentación visual y la experiencia general del juego. Se asegurará que la estética apoye la mecánica y dinámica del juego, facilitando un entorno de aprendizaje atractivo y efectivo.

CAPÍTULO V: DESARROLLO

Para poder desarrollar este proyecto es necesario tener en cuenta las diferentes necesidades que lo hacen relevante. Desde a quién está orientado el producto hasta cómo serán las representaciones visuales de lo que se quiere enseñar, para partir se verá el siguiente punto.

5.1 Público Objetivo

En el desarrollo de un Serious Game orientado a la enseñanza de programación, es importante definir claramente el público objetivo. Este juego está dirigido principalmente a estudiantes de primer año de universidad, específicamente aquellos que no tienen experiencia previa. Esto incluye a aquellos que están cursando su primer curso de programación o fundamentos computacionales, donde se enseñan los conceptos básicos para resolver problemas de manera algorítmica.

La falta de formación en programación en la educación media a menudo crea una brecha en las habilidades que se requieren en la educación superior, por lo que este video juego busca cerrar esa brecha de una manera interactiva y accesible. El enfoque del prototipo es enseñar principios fundamentales de la programación de una manera atractiva, usando mecánicas de juego que promuevan el aprendizaje práctico.

5.1.1 Necesidades del Público

El público objetivo necesita un enfoque que despierte su curiosidad y haga que aprender programación sea atractivo. El juego debe ser lo suficientemente entretenido como para mantener su interés, pero también educativo para que adquieran las habilidades necesarias. Sumado a esto, la mayoría de los estudiantes de este nivel son jugadores que puedan verse atraídos a la idea de poder complementar las clases con una experiencia que sea más llamativa. El hecho de que este sea interactivo ya es un avance de una clase en la que solo se reparten los conocimientos de manera teórica.

De acuerdo con un estudio sobre el futuro de la educación en Chile se destaca la importancia de todo lo que involucra la programación. La incorporación de la tecnología en el aula es crucial para el desarrollo de las habilidades del siglo XXI, ya que prepara a los estudiantes para enfrentar los desafíos del futuro. El enfoque educativo debe adaptarse a estos cambios, promoviendo la enseñanza de competencias como la programación, la robótica y el pensamiento computacional, esenciales para los trabajos del mañana. Iniciativas como las asignaturas que incorporan tecnologías educativas permiten a los estudiantes no solo desarrollar conocimientos, sino también habilidades prácticas que les serán útiles a lo largo de su vida profesional (Pais Digital, 2023).

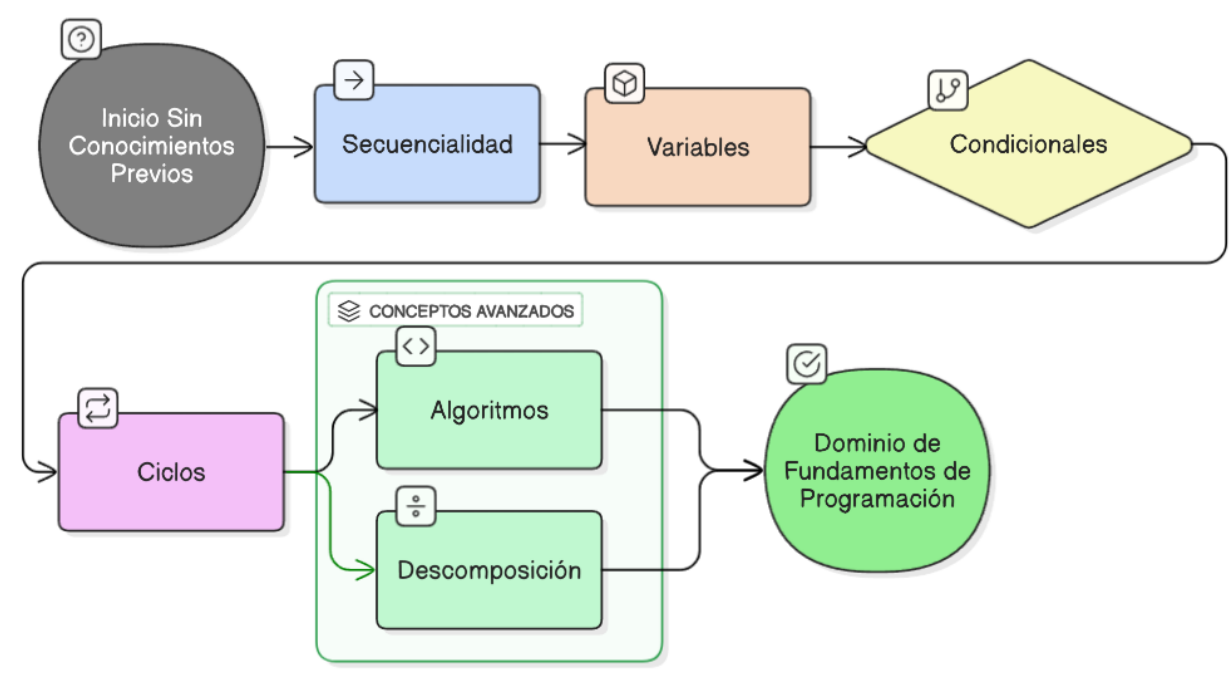
"Sin embargo, aprender a programar no es un ejercicio fácil, más aún si se tiene en cuenta que se involucran niveles de organización y sistematización de procesos cognitivos ligados a la organización y atención, así como también un alto grado de creatividad a la hora de resolver un problema." (Jones et al., 2022). El juego debe ayudar a reducir esta barrera al presentar la programación de forma visual y práctica, vinculando las acciones dentro del juego con los resultados del código. Para solucionar este problema lo que se busca es colocar al estudiante en un entorno donde se pueda ver de forma simple cómo funcionan los conceptos más complejos de razonar en la programación, esto por medio de recursos bien definidos y de problemas que le crean una necesidad al jugador a seguir mejorando y optimizando sus sistemas.

5.2 Contenidos de Aprendizaje

Es importante identificar el foco y estructurar los conocimientos y habilidades que los estudiantes deben adquirir a través del Serious Game. Estos contenidos deben estar alineados con los objetivos pedagógicos, y el diseño del juego debe integrarlos de forma práctica y atractiva.

Figura 1

Diagrama de los contenidos de aprendizaje



En la figura 1 se presenta una visión holística de los contenidos de aprendizaje que el Serious Game busca impartir. El diagrama ilustra la progresión lógica de los conceptos, comenzando por los más fundamentales y avanzando hacia habilidades más complejas. La secuencialidad es la base, seguida por el uso de variables para gestionar recursos. Posteriormente, se introducen las condicionales, que permiten al robot tomar decisiones, y los ciclos iterativos, que facilitan la automatización de tareas. Todos estos conceptos se aplican en el desarrollo de algoritmos y la descomposición de problemas, que son las habilidades clave para la resolución de desafíos. Esta estructura de aprendizaje se integra de manera orgánica en el entorno del juego, donde el robot sirve como una representación visual y tangible de la lógica que el estudiante está construyendo.

5.2.1 Fundamentos de la Programación

Los contenidos de aprendizaje se enfocan en enseñar los conceptos básicos de programación mediante un enfoque basado en la simulación y el uso de robots. Los conceptos clave incluyen:

1- La secuencialidad hará que los estudiantes aprendan a organizar instrucciones de manera lógica y secuencial, una habilidad esencial en la programación.

2- Uso de variables para almacenar y gestionar información dentro del juego. El jugador aprenderá a manipular y controlar valores, como recursos y parámetros del robot.

3- Las condicionales que se enseñarán decisiones lógicas mediante la implementación de estructuras condicionales que permiten que el comportamiento del robot cambie según diferentes situaciones.

4- Los ciclos iterativos con ellos el juego presentará situaciones donde los estudiantes deberán implementar ciclos o bucles para repetir acciones hasta que se cumplan ciertas condiciones.

5.2.2 Pensamiento Lógico y Resolución de Problemas

Se debe tener en cuenta que no solo se tiene que entender los conceptos de la programación, sino también el poder entender los pasos que conlleva un proceso grande.

Descomposición

Es una habilidad clave en la resolución de problemas y uno de los aspectos más importantes en la enseñanza de programación. En el contexto del juego, esta habilidad será fomentada de varias maneras:

1.- La división del objetivo principal, los jugadores tendrán un objetivo complejo, como la construcción de una casa. En lugar de abordarlo como una sola tarea, el juego guiará al jugador a dividirlo en varias subtareas manejables, como la recolección de recursos, tales como madera y piedra, la preparación de terrenos y la construcción de las paredes y el techo.

2.- Las subtareas programables hacen que cada subtarea sea planificable de manera independiente, lo que permitirá que los jugadores definan acciones específicas para los robots. Por ejemplo:

Recolectar madera podría ser dividido en subtareas como "ir al bosque", "cortar árbol" y "transportar madera".

Construir una pared podría involucrar "transportar materiales", "posicionar bloques" y "asegurar la estructura".

3.- La planificación estratégica hace que el jugador aprenda a planificar sus movimientos y los de sus robots, priorizando las acciones de manera lógica. Para construir el techo de una casa primero se necesita completar las paredes o que, para conseguir una plancha de madera refinada, primero se tiene que conseguir un tronco de madera. Este enfoque enseña a los estudiantes la importancia de pensar en etapas y dependencias entre tareas.

Algoritmos

La enseñanza de algoritmos será uno de los pilares de la resolución de problemas en el juego. Los estudiantes aprenderán a crear y ejecutar secuencias lógicas de instrucciones para lograr los objetivos dentro del juego. Algunos ejemplos de cómo se abordarán los algoritmos incluyen:

1.- Las secuencias de pasos donde los jugadores crearán algoritmos que guíen a los robots en acciones específicas. Por ejemplo, un algoritmo para talar árboles y recolectar madera puede consistir en:

- Moverse hacia el área de árboles.
- Seleccionar el árbol más cercano.
- Cortar el árbol.
- Recolectar madera caída.
- Transportar la madera al almacén.

2.- Las condiciones y bucles harán que a medida que los jugadores avancen, el juego introducirá estructuras algorítmicas más complejas, como condicionales (if/else) y bucles (for/while). Por ejemplo, un robot podría recolectar madera hasta que haya una cantidad suficiente en el almacén, o podría detenerse si no hay más árboles cercanos disponibles:

- If/else: “Si el almacén tiene menos de 20 unidades de madera, recolectar más; de lo contrario, detener la recolección”.
- While: “Mientras haya árboles disponibles, seguir cortando y recolectando madera”.

Optimización

La optimización será una habilidad que los jugadores desarrollarán al mejorar la eficiencia de sus robots y sus estrategias a lo largo del juego. Este subcomponente se abordará de las siguientes formas:

1.- Optimización de recursos: El jugador aprenderá a programar sus robots para minimizar el uso de recursos o maximizar la cantidad de materiales recolectados en el menor tiempo posible. Esto podría involucrar optimizar las rutas que toman los robots o crear algoritmos que recolecten múltiples tipos de recursos de forma simultánea.

2.- Los jugadores aprenderán a optimizar el tiempo que toma completar una tarea. En lugar de que un robot complete una tarea antes de comenzar otra, los jugadores podrán distribuir tareas entre múltiples robots para trabajar en paralelo. Por ejemplo, un robot podría estar recolectando madera mientras otro prepara el terreno para la construcción, lo que aceleraría significativamente el progreso del juego.

3.- A través de la retroalimentación visual y el análisis del comportamiento de los robots, los jugadores podrán detectar cuándo sus algoritmos son ineficientes. El juego podría mostrar cómo los robots repiten pasos innecesarios, se quedan inactivos o recorren largas distancias sin

necesidad. Los jugadores serán alentados a modificar sus algoritmos para eliminar estos comportamientos no óptimos.

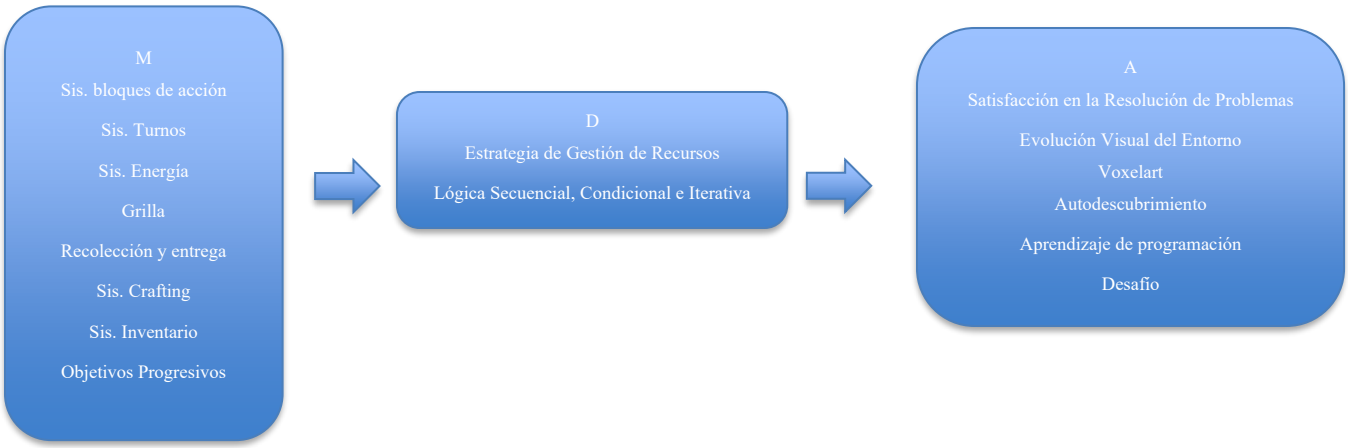
4.- A medida que los jugadores se familiaricen con la lógica de programación, se enfrentarán a desafíos más complejos que requerirán la aplicación de algoritmos más avanzados. Aquí es donde aprenderán a reducir los tiempos de ejecución y a distribuir los recursos de manera más efectiva, cómo usar bucles más eficientes o condicionales más precisos.

5.3 Implementación del framework MDA

El Framework MDA (Mecánicas, Dinámicas y Estéticas) es un enfoque de diseño de juegos que descompone la experiencia del jugador en tres componentes clave. A continuación, se describe cómo este marco será aplicado para el diseño del Serious Game para el aprendizaje de la programación.

Figura 2

Diagrama de implementación de framework MDA en prototipo



En la figura 2 se puede apreciar de manera visual como está conformada la implementación del framework MDA, a continuación, se detalla cada apartado.

5.3.1 Mecánicas (Mechanics)

La implementación de un Serious Game requiere el diseño detallado de las mecánicas para la progresión del jugador a través del contenido y los niveles del juego. Este apartado aborda cómo se organizan estos elementos para configurar un escenario de interacción que permita el aprendizaje de la programación en forma progresiva, atractiva y educativa.

Las mecánicas del juego son las reglas y sistemas de reglas que dictan cómo los jugadores interactúan con el juego. En el Serious Game diseñado, las mecánicas estarán directamente ligadas al proceso de aprendizaje de la programación. Estas son las principales mecánicas a desarrollar:

1.- **Un sistema de programación basado en bloques de acciones**. Los jugadores programarán las acciones de sus robots utilizando un sistema de bloques que representarán diferentes conceptos de programación a través de acciones, tales como actuadores, condicionales, bucles, etc. Este sistema facilita la comprensión de los conceptos de lógica sin necesidad de escribir código de manera explícita. En este sentido, los bloques son un conjunto de acciones preestablecidos con tal de evitar la memorización de sintaxis de código tradicional y mantener siempre disponible visualmente las posibilidades de programación según el nivel del juego, es decir, a medida que avanza en el juego, se desbloquean más bloques para que el jugador pueda usar.

Figura 3

Panel de bloque de acciones que puede realizar el jugador

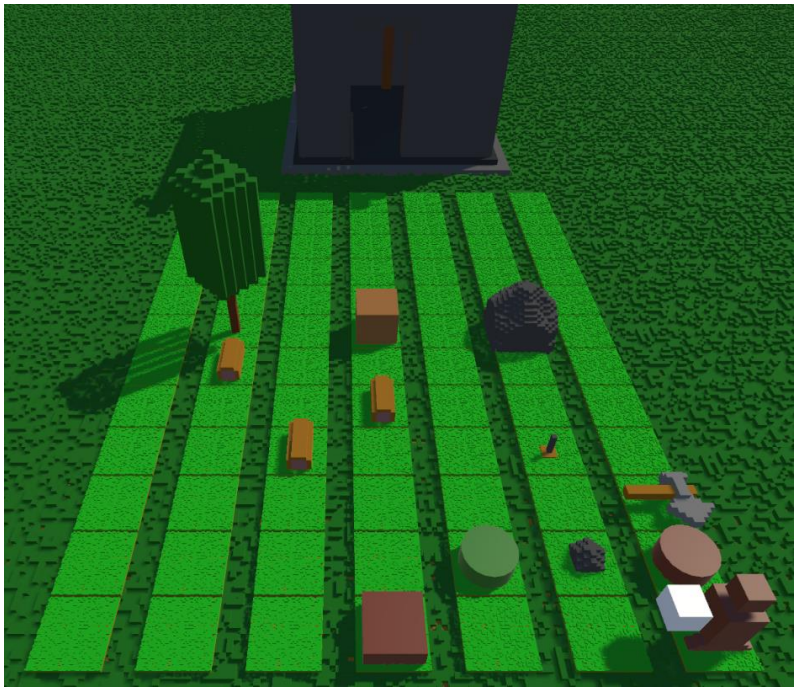
Como se aprecia en la Figura 3, las acciones del robot hacen que los jugadores puedan programar a las máquinas para realizar diferentes tareas en el entorno, llegar a lugares concretos

para recolectar materiales, refinarlos, entregarlos en puntos para poder crear una cadena de producción y desplazarse en función de las condiciones programadas.

Esto se logra en mayor medida con puntos definidos por zonas que tienen recursos, mesas de trabajo y zonas de entrega. Es importante tener en consideración que las divisiones tienen que estar a una distancia que haga factible el hecho de transportar objetos entre ellas.

Figura 4

Distribución de zonas en el juego



Como se puede ver en la figura 4, se debe tener una zona de recolección de madera y otra de metales a la derecha para que en el centro se tenga la fabricación de los materiales refinados que tengan camino directo hacia el área de entrega de materiales. Para que con todo eso se deje una zona de la pantalla para mostrar el objetivo final, que es la construcción de una casa, para que se pueda ir viendo el progreso que se ha conseguido y dar la satisfacción al jugador de cómo poco a poco se va logrando el objetivo que en un principio parecía lejano.

Los sistemas de condiciones y bucles son importantes son la base para que los jugadores aprenderán a implementar estructuras de control las instrucciones if, while y for, que permitirán

que los robots tomen decisiones o repitan tareas de manera automática según las condiciones establecidas por el estudiante.

Para este apartado, se tiene que implementar una interacción con un objeto que no tenga información dada con certeza, es decir, que tiene que depender de un componente de aleatoriedad. El medio por el que se llevara a cabo es un objeto que al interactuar con él tenga posibilidad de que aparezca en más de un lugar. Para implementar los bucles, se hará que un objeto tenga que cumplir una tarea repetitiva hasta que se cumpla una condición dada por el usuario.

2- **Un sistema de turnos.** El sistema de turnos es el que organiza el flujo del juego. Cada turno representa una unidad de tiempo durante la cual el jugador puede planificar y ejecutar una serie de acciones utilizando su robot. A diferencia de los juegos en tiempo real, este sistema otorga al jugador un espacio para reflexionar sin presión, facilitando la comprensión lógica y estratégica de las decisiones que toma. Durante su turno, el jugador puede programar al robot, evaluar su entorno, revisar objetivos y gestionar recursos, todo dentro de una fase estructurada que finaliza una vez que se ejecutan las acciones programadas. El juego mantiene en todo momento el control sobre el tiempo, permitiendo que se aprenda a gestionar sus acciones dentro de este marco, sin sentirse abrumado por la presión del tiempo real. Se tiene una fase de programación, donde se reestablece la energía y el jugador define las acciones del robot y fase de ejecución, donde se ponen en práctica dichas acciones y se observa el resultado.

Al estructurar la experiencia en segmentos ordenados, el jugador puede detenerse a analizar causas y efectos, planificar a futuro, e iterar soluciones en un entorno seguro. Este enfoque permite que cada decisión tenga peso, y que el jugador vea de forma clara las consecuencias de su programación. Además, es un sistema que se adapta fácilmente a niveles progresivos de dificultad, permitiendo que el aprendizaje ocurra por etapas y con retroalimentación constante.

3- **El sistema de energía.** El sistema de energía regula la cantidad de acciones que un robot puede ejecutar durante su turno. Está diseñado para representar el concepto de recursos limitados en la

programación y gestión de tareas, enseñando al jugador a priorizar y optimizar decisiones. Este sistema asigna a cada robot una cantidad fija de energía por turno, que se recarga automáticamente al inicio de cada nuevo turno. Cada acción del robot, como moverse, recolectar o interactuar con un objeto, consume una cierta cantidad de energía, lo que obliga al jugador a planificar con antelación y considerar el costo de cada acción. Se tiene un contador de energía visible, cantidad de energía que consume cada acción, y un sistema de validación que impide ejecutar acciones si no hay suficiente energía disponible.

Todo esto busca que el jugador internalice el valor de la optimización, la planificación y el aprendizaje a través del error, ya que un uso ineficiente de la energía puede significar no cumplir los objetivos del turno. No solo es limitar arbitrariamente las acciones, sino más bien servir como un marco lógico que desafíe al jugador a resolver problemas con restricciones, promoviendo un pensamiento estratégico. Este sistema permite reforzar conceptos como la economía de recursos, la toma de decisiones basadas en consecuencias y la importancia de secuenciar acciones de forma lógica y eficiente. Con él, los jugadores pueden explorar distintas soluciones a un problema, adaptarse a contextos cambiantes y mejorar progresivamente su desempeño mediante la experimentación dentro de un entorno controlado y educativo.

4- **Grilla.** La grilla corresponde a casillas que define la estructura del entorno en el que los robots se mueven, interactúan y ejecutan sus acciones. El mundo del juego está dividido en una cuadrícula, cada una representando una unidad de espacio. Este diseño facilita el control preciso del desplazamiento, la ubicación de recursos, zonas de trabajo, obstáculos y objetivos, asegurando que el jugador comprenda visualmente los límites y posibilidades del entorno. Las casillas actúan como nodos en los que se pueden llevar a cabo acciones. Esto proporciona una base clara para el mundo y la ejecución de las acciones programadas por el jugador. Permite estructurar mecánicas como el movimiento por turnos, el uso de energía al desplazarse, o la planificación de rutas eficientes para recolectar y entregar ítems. Además, contribuye a que el jugador tenga un entorno controlado y comprensible.

5- **Recolección y Deposición**: permite a los jugadores interactuar con el entorno para obtener recursos. Se basa en la programación de acciones del robot, quien debe desplazarse hasta un lugar específico que contienen materiales y ejecutar una acción de recoger o dejar interactuando con las casillas y el inventario del robot. Se tiene que considerar el camino hacia el recurso, los tipos de materiales y las condiciones para que un robot pueda recolectar (como tener suficiente energía o estar correctamente posicionado. Además, existen límites de capacidad de carga que conectan directamente con el sistema de inventario, lo que obliga al jugador a tomar decisiones sobre qué recolectar, cuándo hacerlo y cómo distribuir sus acciones eficientemente en cada turno. Se busca que el jugador planifique rutas, priorice recursos, gestione su energía y tome decisiones.

6- **Un sistema de crafting.** El sistema de crafting permite combinar recursos recolectados para crear nuevos objetos. Está diseñado para ser accesible y visual, mediante bloques que representan recetas específicas que pueden ser vistas en la mesa de creación y que la acción es válida cuando se tiene en su inventario los materiales necesarios y que el robot este posicionado correctamente. El crafting no solo amplía la utilidad de los recursos, sino que también introduce conceptos de procesos, transformación y dependencia lógica entre acciones, lo cual fortalece el aprendizaje por medio de la programación estructurada. Se debe tener una interfaz de creación, un conjunto de recetas predefinidas y los materiales base que el jugador ha recolectado previamente. Al ejecutar una receta, los materiales son consumidos del inventario del jugador y se genera un nuevo ítem que se puede utilizar cumplir objetivos. Además, se debe introducir al jugador en procesos secuenciales más complejos, donde cada acción tiene consecuencias en etapas futuras. Es una herramienta para enseñar cómo pequeñas decisiones acumuladas impactan en sistemas más grandes, y cómo la organización de recursos y tiempo es clave para alcanzar un objetivo. A través del crafting, el jugador no solo aplica conocimientos adquiridos en otros sistemas, sino que también se enfrenta al desafío de optimizar su cadena de producción, entendiendo que la programación y la estrategia están profundamente entrelazadas.

7- **Un sistema de inventario** para gestionar objetos. Esto permite almacenar, organizar y visualizar los ítems obtenidos por el robot. Fomenta la planificación, ya que obliga al jugador a tomar decisiones sobre qué llevar, usar y guardar para más adelante. El inventario se presenta de manera visual y accesible, que responde a la limitación de capacidad y tipo. Además, se vincula con el

sistema de recolección y el de crafting, sirviendo de puente entre lo que el jugador obtiene y lo que puede construir. Este sistema se actualiza automáticamente cada vez que se utiliza una acción correspondiente, asegurando consistencia entre lo que se ve en pantalla y el estado real del inventario.

Se da una estructura y significado a los recursos del juego, permitiendo que el jugador trace estrategias a partir de la administración de objetos disponibles. También promueve la toma de decisiones y el pensamiento a largo plazo, ya que ciertos materiales pueden ser escasos o requerir múltiples pasos para obtener. El jugador aprende, a través de la interacción con el inventario, a organizar, priorizar y planificar en función de sus objetivos. En este sentido, el inventario no es solo un contenedor de ítems, sino una herramienta pedagógica que refuerza habilidades fundamentales como la lógica de procesos, el análisis de recursos y la optimización.

8- **Objetivos progresivos** estructura el avance del jugador a través de metas. Este sistema guía el aprendizaje del jugador y le proporciona una dirección clara dentro del juego, asegurando que los conceptos de programación y mecánicas del juego se introduzcan de manera gradual y contextualizada. En este sentido, cada objetivo funciona como un peldaño que, al superarse, amplía las capacidades del jugador y la complejidad de los retos a enfrentar. Teniendo una lista de objetivos específicos, un sistema de seguimiento que verifica su cumplimiento, y recompensas o desbloqueos que se otorgan tras su finalización. Estos pueden variar desde tareas simples hasta desafíos más complejos. A medida que el jugador avanza, se presentan nuevas metas que requieren aplicar conocimientos adquiridos.

5.3.2 Dinámicas (Dynamics)

Las dinámicas son el comportamiento emergente que surge cuando los jugadores interactúan con las mecánicas del juego. Son el puente entre las reglas establecidas y la experiencia que los jugadores viven a medida que avanzan. En el Serious Game, las dinámicas permitirán a los jugadores experimentar cómo sus decisiones afectan el progreso del juego y cómo pueden optimizar sus estrategias. Algunas de las dinámicas clave que definirán la experiencia del jugador son:

Esto hace que a medida que los robots realizan tareas, los jugadores tendrán que gestionar los recursos que recolecten y los procesos para cumplir su objetivo. Esto emerge de la interacción entre varios sistemas centrales del juego. Cada uno aporta una pieza clave que, al entrelazarse, obliga al jugador a tomar decisiones estratégicas sobre cómo administrar los recursos disponibles, tanto materiales, energía, rutas, etc. Las acciones limitadas por turno, la necesidad de recolectar ciertos objetos en lugares específicos, y la posibilidad de fabricar nuevos, generan un entorno en el que la eficiencia y la planificación son importante.

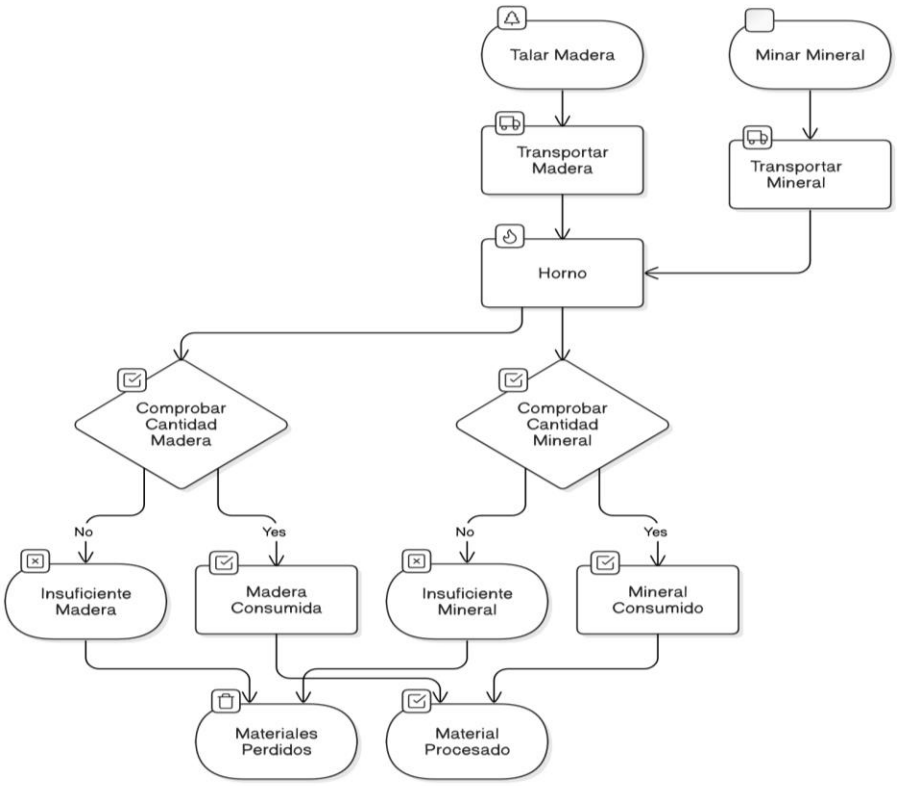
Esta dinámica nace de la limitación deliberada impuesta por el sistema de energía, que impide que el jugador actúe indefinidamente en un turno, de la necesidad de recolectar y transportar recursos para fabricar objetos clave mediante el sistema de crafting, incentivada por los objetivos. El jugador debe considerar, por ejemplo, si le conviene usar su energía para moverse hacia una fuente de recursos o si es mejor recolectar lo que tiene cerca y usar el turno siguiente para avanzar. A esto se suma la restricción del inventario, que obliga a priorizar qué objetos transportar y en qué momento.

No basta con ejecutar acciones por inercia: el jugador debe anticipar futuras necesidades, considerar distancias entre zonas de recolección y estaciones de trabajo, y administrar su inventario para no malgastar esfuerzos.

Sumado a esto, el dividir el trabajo entre diferentes robots, asignando tareas específicas para optimizar. Eso hace que el jugador pueda crear secuencias de trabajo en equipo, donde un robot recolecta materiales, otro los refine y un tercero los transporte hasta la zona de construcción. Eso también conlleva a tener que evitar colisiones y conflictos en la ejecución de comandos, ya que los jugadores deberán programar las rutas de cada robot.

Figura 5

Diagrama de proceso de fundir metal



Como se aprecia en la figura 5, teniendo un proceso así, hace que el usuario tenga múltiples maneras de cumplir el objetivo, dando opciones para organizar sus recursos de manera que se cumpla un objetivo y que se generen dudas acerca de cómo afrontar el desafío. Si bien se tiene el ejemplo de como hacer un material en específico, se puede llegar a este objetivo de diferentes maneras. Sumado a esto, en la imagen se puede ver un camino posible, pero eso no quiere decir que sea el único, ya que dependiendo de cómo se interactúa con las diferentes mecánicas del juego se le pueden agregar pasos y modificar el flujo dándole al jugador libertad de organización.

Lógica Secuencial, Condicional e Iterativa

El jugador debe programar las acciones de su robot utilizando bloques que representan instrucciones secuenciales (una acción tras otra), condicionales (acciones que se ejecutan solo si se cumple cierta condición) e iterativas (repetición de acciones mediante bucles). Esta lógica forma la base del pensamiento computacional, y es esencial en el aprendizaje de programación. En este

juego, dichas estructuras se convierten en una herramienta visual con la que los jugadores pueden experimentar activamente, aplicándolas para resolver situaciones concretas en el entorno del juego.

El jugador planifica previamente las acciones del robot antes de que se ejecuten, teniendo en cuenta el estado del entorno (como obstáculos, recursos, o condiciones aleatorias). El sistema de turnos refuerza esta lógica, ya que cada turno representa un momento en el que se evalúa el bloque de instrucciones programado. Al hacerlo, el juego simula el comportamiento de un programa real, ejecutando bloque por bloque las instrucciones dadas, tal como lo haría un intérprete o compilador en un lenguaje de programación tradicional.

Se debe obliga al jugador a ser preciso en las instrucciones: por ejemplo, para que un robot llegue a una zona de recolección, no basta con "moverse", sino que debe hacerlo en la secuencia exacta de casillas, con instrucciones bien ordenadas. Si el jugador desea que el robot recoja un objeto solo si hay uno disponible, deberá usar una estructura condicional. Si quiere que el robot atraviese varias casillas similares, puede usar un bucle. Así, cada sistema colabora para que el jugador entienda cómo aplicar los principios fundamentales de la lógica computacional en un contexto interactivo. No confundir con que solo hay un camino para realizar una tarea, solo que se debe ser preciso con las instrucciones que se les entregan a los diferentes robots.

5.3.3 Estéticas (Aesthetics)

Las estéticas se refieren a la experiencia emocional que el jugador obtiene al interactuar con el juego. En este caso, las estéticas estarán enfocadas en crear una experiencia educativa motivante y accesible para estudiantes que están aprendiendo a programar. Algunas de las estéticas clave que se pretenden alcanzar incluyen.

Satisfacción en la Resolución de Problemas

Uno de los aspectos más relevantes es la satisfacción en la resolución de problemas. Se busca que el jugador experimente una sensación de logro al enfrentarse a desafíos de

programación, superarlos mediante la aplicación lógica y estructurada de soluciones. Para lograrlo, cada nivel presenta un problema claro y bien definido, asegurando que se tenga una comprensión precisa de lo que debe resolver.

Una de las estrategias fundamentales es proporcionar retroalimentación positiva inmediata al jugador cuando encuentra una solución. Esta respuesta puede manifestarse a través de animaciones visuales atractivas y mensajes motivadores que refuercen la sensación de éxito. De esta manera, se genera una experiencia de juego más gratificante y se mantiene el interés del jugador en seguir avanzando.

Además de la gestión del error dentro del juego. En lugar de penalizar al jugador por fallar, se le anima a iterar sobre su solución, explorando diferentes enfoques hasta encontrar la respuesta correcta. Esta metodología no solo reduce la frustración, sino que fomenta un aprendizaje basado en la experimentación y la mejora continua. Así, el jugador desarrolla confianza en sus habilidades, entendiendo que la programación implica un proceso de prueba y error que forma parte natural del aprendizaje.

Evolución Visual del Entorno

La progresión y evolución visual del entorno desempeñan un papel clave para reforzar la sensación de avance del jugador. Uno de los principales objetivos es que el usuario no solo perciba su progreso a través de la superación de niveles o la adquisición de nuevas habilidades, sino también mediante transformaciones visibles en el mundo del juego.

Se implementan transformaciones visuales dinámicas que reflejan los logros del jugador. A medida que completa tareas y resuelve desafíos de programación, el entorno evoluciona y mejora de manera perceptible. El juego está ambientado en una ciudad en construcción, cada nivel superado podría representar la finalización de una parte de la casa, la activación de nuevas estructuras o el embellecimiento del entorno. De esta manera, el jugador ve el impacto directo de sus acciones, lo que refuerza su motivación para seguir avanzando.

Además, cada nivel está diseñado con una estética diferenciada que no solo proporciona variedad visual, sino que también refuerza la idea de progreso. Esto puede manifestarse en cambios de iluminación, paletas de colores, elementos interactivos o detalles adicionales que hacen que el entorno se sienta vivo y en constante evolución.

Esta progresión visual también funciona como una motivación extrínseca, ya que, aunque estos cambios no afectan directamente la jugabilidad, sí sirven como un indicador tangible del progreso del jugador. Saber que cada acción contribuye a la transformación del mundo del juego genera una conexión emocional con el entorno y refuerza el sentido de logro. Así, la evolución visual no es solo un aspecto estético, sino una herramienta fundamental para mejorar la experiencia de aprendizaje y mantener al jugador comprometido con su avance.

Voxelart

El estilo visual basado en Voxel Art aporta una identidad clara y accesible al juego, combinando simplicidad geométrica con un atractivo estético nostálgico. Este enfoque permite representar elementos del mundo del juego, ya sean los robots, recursos, casillas y obstáculos, de forma clara y coherente, facilitando la comprensión visual de las acciones sin distraer al jugador con detalles innecesarios.

Este lenguaje visual también favorece la inmersión y la empatía, ya que incluso con una estética minimalista, los personajes y objetos poseen personalidad. El jugador puede interpretar rápidamente cual es el recurso que se requiere, donde está un robot, o cómo interactúan entre sí los distintos elementos del escenario. Este nivel de claridad visual se convierte en una herramienta poderosa para el aprendizaje, ya que minimiza la carga cognitiva y centra la atención en la lógica de programación.

Además, el uso del Voxel Art facilita la escalabilidad del proyecto. Dado que los activos se construyen con unidades modulares, resulta sencillo añadir nuevos elementos visuales, niveles o mecánicas sin comprometer la coherencia estética del juego. Esto también permite a los

desarrolladores y, eventualmente, a los propios jugadores, experimentar con la creación de contenido nuevo dentro de un estilo visual uniforme.

Autodescubrimiento

El autodescubrimiento y el aprendizaje activo son pilares fundamentales en el diseño de este Serious Game, ya que el objetivo no es solo enseñar programación, sino fomentar una mentalidad exploratoria y analítica en los jugadores. En lugar de proporcionar instrucciones rígidas y lineales, el juego está diseñado para que los jugadores aprendan por sí mismos a través de la experimentación y la resolución de problemas.

Para lograr esto es la exploración sin restricciones. A diferencia de otros juegos educativos que imponen un camino único para resolver cada desafío, este permite que los jugadores experimenten diferentes enfoques y descubran soluciones a su propio ritmo. Esta libertad fomenta la creatividad y refuerza la autonomía del jugador, permitiéndole aprender de manera más significativa.

Aprendizaje de programación

El juego está diseñado para ofrecer múltiples soluciones a cada problema. En programación, rara vez hay una única manera correcta de resolver un desafío, y este principio se refleja en la jugabilidad. Los jugadores pueden encontrar diversas formas de lograr un objetivo, lo que fortalece su pensamiento computacional y su capacidad para analizar distintos escenarios. Esto no solo mejora sus habilidades lógicas, sino que también los motiva a probar nuevas estrategias sin miedo al error.

El aprendizaje también se ve impulsado por desafíos progresivos, en los que cada nuevo nivel introduce mecánicas sin explicaciones extensas ni tutoriales excesivos. En lugar de explicaciones detalladas, el juego sugiere de manera sutil cómo funciona una nueva mecánica,

incentivando al jugador a descubrirla por sí mismo. Este enfoque promueve la experimentación y la intuición, dos habilidades clave en la resolución de problemas y la programación.

Estos elementos convierten al juego en una herramienta poderosa para el aprendizaje activo. No solo enseña conceptos de programación, sino que también desarrolla habilidades de resolución de problemas, fomenta la autonomía y refuerza el pensamiento lógico, todo a través de la exploración y la experimentación.

Desafío

El desafío en este proyecto se presenta como el núcleo motivador del aprendizaje, donde cada nivel exige al jugador resolver problemas mediante la programación lógica de robots. Desde el inicio, se introducen mecánicas simples como el movimiento secuencial y la recolección de objetos, las cuales se van complejizando al incorporar elementos como condicionales, interacción con objetos del entorno y limitaciones de energía por turno. Estas restricciones obligan al jugador a pensar antes de actuar, planificar sus movimientos y optimizar los recursos disponibles, lo que convierte cada misión en un pequeño rompecabezas lógico.

A medida que el jugador avanza, el desafío no solo se incrementa en dificultad, sino que también en variedad. Se introducen nuevas reglas, mecánicas y combinaciones de objetivos que requieren mayor atención y dominio de la lógica programática. Por ejemplo, superar un nivel puede implicar calcular el orden correcto de acciones, responder a condiciones aleatorias en el entorno, o coordinar múltiples robots para cumplir un objetivo común. Este crecimiento paulatino del reto permite que el jugador no se sienta abrumado, manteniendo su interés y favoreciendo el aprendizaje significativo a través de la práctica.

Este enfoque al desafío está diseñado para que el error forme parte natural del proceso de aprendizaje. El jugador puede equivocarse, observar el comportamiento del robot y ajustar su lógica hasta encontrar la solución óptima. Esto promueve la resiliencia y la perseverancia, habilidades clave tanto en la programación como en la vida real. El juego no castiga el fallo, sino

que lo convierte en una oportunidad de experimentación, lo cual refuerza su valor como una herramienta educativa que enseña desde la experiencia, no solo desde la teoría.

5.4 Diseño de Niveles

Para afrontar lo implementado en el MDA, se tiene que definir cuál será la progresión que hará el jugador para tener una experiencia gratificante y útil para la educación.

5.4.1 Progresión

El juego debe ofrecer una progresión gradual que permita a los jugadores asimilar los conceptos de programación de manera progresiva y práctica. La progresión se organizará en niveles o fases de dificultad creciente, asegurando que los jugadores puedan dominar cada concepto antes de avanzar al siguiente. La estructura de progresión es la siguiente.

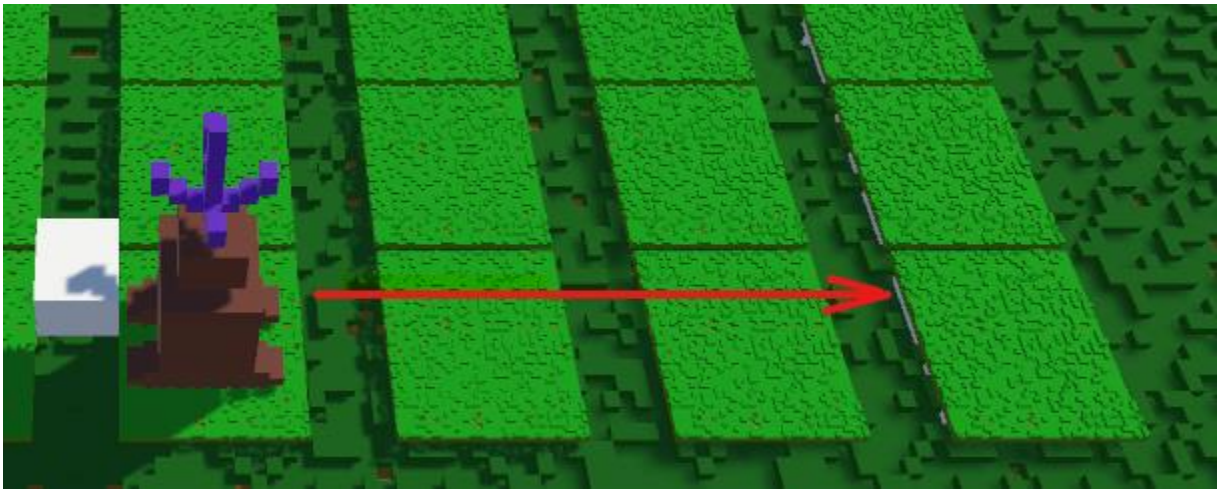
Nivel 1: Iniciación al Movimiento

Aquí se tiene que enseñar las instrucciones básicas de movimiento mover adelante, mover atrás y rotaciones. El robot se desplazará por el entorno para alcanzar un objetivo, recolectar un recurso o llegar a un destino. Los jugadores tendrán que crear secuencias de comandos simples para que el robot se mueva correctamente.

En este nivel el proceso que se va a realizar es la petición de que el usuario llegue a una casilla en específico del mapa para hacer la primera comprobación de que se entiende el sistema de ingresar y ejecutar las instrucciones.

Figura 6

Petición de objetivo de movimiento al jugador



Como se puede ver en la figura 6, el jugador tendrá que colocar una serie de instrucciones que lo lleven hacia la casilla en especifico para poder continuar con el juego.

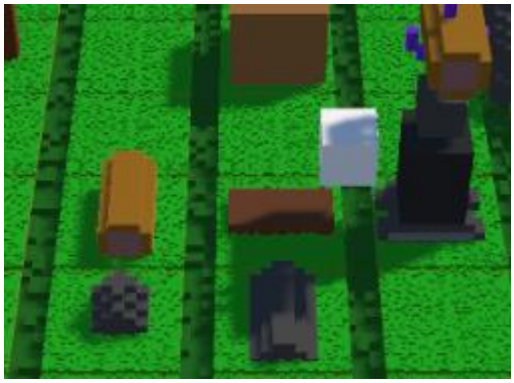
Nivel 2: Recolección de Recursos

Se tiene que introducir acciones adicionales, como "recoger" o "colocar" recursos. El robot será programado para recoger materiales como madera o piedra y llevarlos a un lugar de almacenamiento. Además, los jugadores deberán programar secuencias de acciones y optimizarlas para mejorar la eficiencia.

Aquí se espera que el usuario sea capaz de planificar una ruta para recoger los materiales que aparecen en el mapa y llevarlo a la zona de entrega para lograr el objetivo.

Figura 7

Presentación de diferentes objetos del prototipo



Como se puede ver en la figura 7, estos son algunos de los materiales que contiene el prototipo que sirven para que el jugador tenga una referencia visual. Además, le suma variedad a lo que se puede conseguir.

Nivel 3: Introducción a Condicionales

Enseñar el uso de condicionales. El robot tomará decisiones en función de condiciones del entorno, por ejemplo, si hay un obstáculo, tomar una ruta alternativa. Aquí se quiere que los jugadores deben programar robots que puedan adaptarse a cambios en el entorno de forma autónoma.

Figura 8

Ejemplo de problema condicional "if" y resolución

Como se puede ver en la figura 8, se tiene un problema donde el usuario tiene que recoger un material de una de las mesas de la parte de abajo, que cuando se utiliza la acción "PULSAR" del juego el objeto, en este caso una roca, se utiliza la instrucción IF para hacer la comprobación de que si la mesa blanca contiene el material roca. Haciendo esa comprobación se tiene dos posibilidades, que este o no, en cada caso lo que tiene que hacer el robot son una serie de instrucciones que solo son validas para recoger el material si se planteo de manera correcto el condicional.

Nivel 4: Uso de Bucles

Se debe introducir el concepto de bucles "while". Los estudiantes aprenderán a hacer que los robots repitan tareas automáticamente, como recolectar un conjunto de recursos de forma continua. Esto representa un desafío a la hora de programar trabajos repetitivos con eficiencia utilizando bucles.

Luego completar el objetivo del nivel anterior se le dará una recompensa al usuario para que pueda continuar con el aprendizaje, pero para conseguir la recompensa tendrá que superar una prueba lógica de la condición “if” en la cual el objeto puede aparecer de manera aleatoria en cualquiera de las dos cajas para así que el usuario no tenga la información de donde se encuentra el objeto hasta que no se ejecute el código y se realice la comparación.

Nivel 5: Gestión de Múltiples Robots

Enseñar a los jugadores cómo coordinar varios robots trabajando simultáneamente. Los jugadores deben programar diferentes robots para colaborar en tareas como la construcción de una estructura o la recolección de grandes cantidades de recursos. Tienen que gestionar la interacción entre robots y optimizar la distribución de tareas entre ellos.

Nivel 6: Construcción de una Estructura Compleja

Desafiar al jugador a usar todos los conceptos aprendidos para construir una estructura compleja. Los jugadores deberán programar a los robots para coordinar la recolección de recursos, la fabricación de materiales y la construcción de una casa. Además de optimizar el uso de los suministros y coordinar múltiples robots para completar el proyecto de construcción.

Cada nivel estará diseñado de manera que el jugador adquiera nuevas habilidades de programación mientras supera desafíos gradualmente más difíciles. Esta estructura asegurará que los jugadores no solo aprendan los conceptos teóricos, sino que también desarrollen habilidades prácticas para resolver problemas mediante la programación.

5.5 Arquitectura

La arquitectura del proyecto está basada en principios de modularidad, escalabilidad y buenas prácticas de programación. Para estructurar el código se aplicó el patrón Mediador, lo que permite mantener las mecánicas del juego separadas y fácilmente ampliables. Esta decisión fue tomada para facilitar el crecimiento del prototipo sin necesidad de reescribir grandes partes del código.

Uno de los principales desafíos fue aprender a aplicar estos patrones en Unity, lo que implicó una etapa de exploración profunda del motor y reestructuración del código inicial. Como resultado, se logró una base sólida y mantenible, donde cada sistema funciona de manera independiente pero coordinada a través de controladores centrales.

El sistema de turnos es gestionado por un ControladorTurnos, que coordina la ejecución de las acciones de robots, objetos interactivos y elementos del entorno. Estas acciones están organizadas en clases hijas que heredan de una clase base, siguiendo una estructura clara y flexible, ver más detalles en el siguiente apartado de scripts en Implementación de Prototipo en Unity.

Además, se utilizan herramientas de Unity como los ScriptableObjects para definir datos reutilizables como objetivos, materiales, recetas o tipos de acciones. Esto mejora la organización y la eficiencia del proyecto, teniendo también información que puede ser llamada y utilizada en diversos objetos dando más independencia.

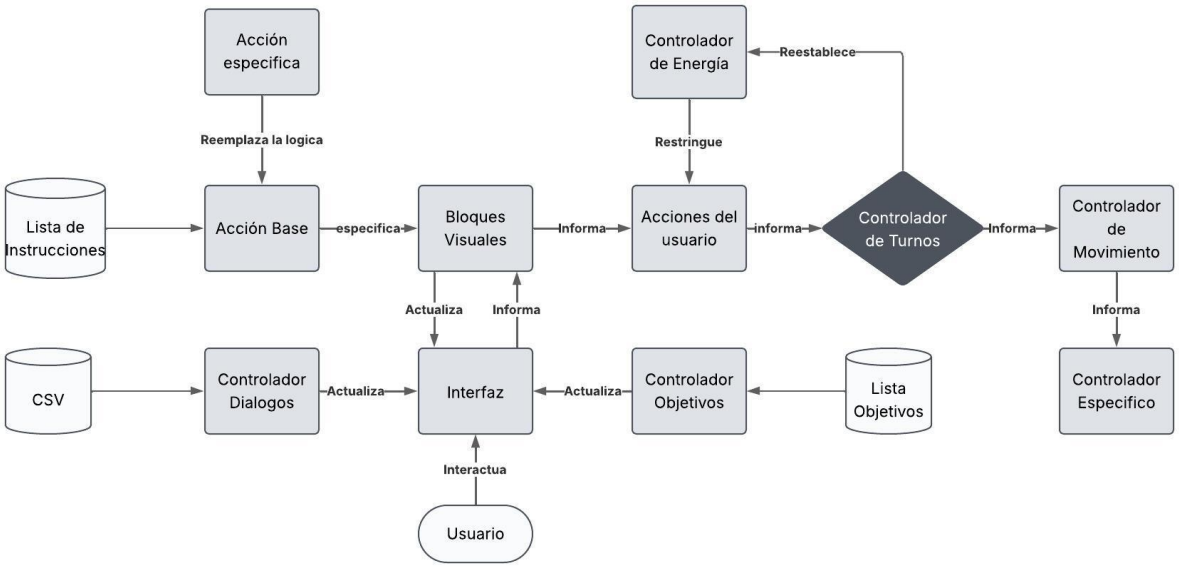
Se tiene una lista para identificar los tipos posibles y evitar errores comunes, y está integrado con un sistema de programación por bloques que almacena las instrucciones que debe seguir cada robot. Estas instrucciones son ejecutadas y validadas por el Controlador Movimiento, que también gestiona recursos como la energía y evita conflictos entre robots durante los turnos.

También se desarrolló un sistema de diálogos que guía al jugador dentro del juego. Está basado en archivos CSV y permite mostrar mensajes personalizados, facilitando la comprensión

de las mecánicas sin depender de explicaciones externas. Los diálogos se integran con la interfaz del juego y pueden activar animaciones o efectos para reforzar la experiencia del jugador.

Figura 9

Diagrama de funcionamiento de arquitectura



En la figura 9 se puede apreciar de manera visual el esquema que fue implementado en el proyecto. Esto permite comprender con mayor claridad la estructura y las relaciones entre sus distintos componentes. Así es posible identificar cómo se integran las diferentes partes del proyecto y cómo interactúan entre.

5.6 Implementación de Prototipo en Unity

Este apartado detalla el proceso de desarrollo del Serious Game utilizando Unity como motor de creación. La implementación del prototipo será clave para materializar las mecánicas, la interacción y la experiencia de aprendizaje planeadas. A continuación, se describe cada fase de la implementación.

5.6.1 Estructura del Proyecto en Unity

El desarrollo del prototipo comenzará con la creación de la estructura básica del proyecto en Unity, organizando los principales componentes del juego, como escenas, scripts y prefabricados. Esto permitirá una gestión eficiente del código y los recursos gráficos.

Escenas

En cuanto a las escenas solo va a ver una que hará que se vaya ampliando en espacio de juego y desbloqueando las mecánicas, permitiendo la carga y gestión independientes de cada etapa del proceso educativo. Se estructurará de manera modular para facilitar la iteración rápida durante el desarrollo. Implementando el concepto de mapa por zonas, descrito en las mecánicas, que harán que el usuario tenga que moverse por todo el escenario

Scripts

Los scripts son el código que hace que el prototipo funcione, toda la lógica para cada mecánica del video juego. Se implementarán para controlar las mecánicas y comportamientos que tiene que hacer el prototipo, la lógica de programación de los robots y la interacción del jugador con el entorno. Los scripts estarán organizados de manera clara y modular, facilitando la expansión del prototipo y el ajuste de las funcionalidades.

Se aplicó un patrón de diseño, esta decisión tomada por la formación recibida hace que a la hora de escalar y desarrollar nuevo contenido para el serious game se trabaje de forma modular e independiente entre cada mecánica. Sin embargo, su implementación en Unity requirió aprender a utilizar el motor con mayor profundidad. Aunque ya se tenía una comprensión general sobre Unity, fue necesario explorar cómo gestionar mejor los controladores y la comunicación entre ellos para aplicar estos principios de diseño de manera efectiva. Esta adaptación no solo mejora la organización del código, sino que también facilita la adición de nuevos participantes en el turno y hace que el sistema sea más modular, flexible y mantenible a largo plazo. El tiempo que se dedicó a aprender el funcionamiento del motor va desde 5 meses hasta llegar a la estructura que se tiene

en la versión final del proyecto, ya que en un principio se experimentó con la programación y desarrollo de las mecánicas básicas para probar si la idea principal que se tiene del videojuego era interesante sin tener ningún tipo de estructura de código para un mejor mantenimiento del mismo, haciendo que se haga tenido que reestructurar gran parte del contenido realizado en las fases temprana, siendo utilizadas mayormente para familiarizarse con Unity.

Se decidió optar por el patrón mediador para estructurar el "ControladorTurnos", que es el que conecta a todos los estados, ya que permite centralizar la coordinación de los distintos controladores sin generar dependencias directas entre ellos. Actualmente, este script gestiona la ejecución de acciones de robots, recolectables y palancas, lo que hace que se gestione de manera correcta las distintas fases de cada turno, ya que se tiene la necesidad de esperar a que se completen primero las acciones de algunos objetos para poder continuar con el proceso.

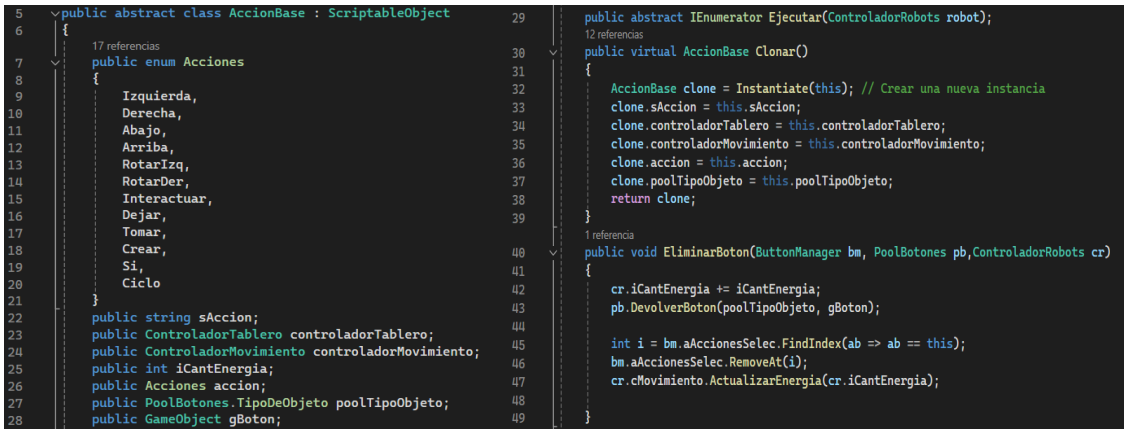
Además, Unity ofrece diversas herramientas diseñadas para simplificar la gestión de información, entre las cuales destacan los "ScriptableObjects". Estas permiten almacenar datos que servirán como base para acciones específicas dentro del juego. Un ejemplo práctico es su aplicación en la definición de objetivos del juego, al tratarse de información previamente establecida, esta herramienta permite gestionar de manera eficiente y reutilizable a lo largo del desarrollo. Con esto se tiene que definir toda aquella información que tiene estas características creando así el apartado de acciones, materiales, objetivos, recetas y robots.

Se tiene una gran diversidad de scripts en el proyecto, ya se desde la ejecución de las acciones hasta la actualización de elementos visuales. Para partir, uno de los sistemas más importantes que se tienen es la forma en la que se manejan las acciones, las buenas prácticas dictan que se debe tener los procesos complejos divididos en procesos sencillos facilitando la escalabilidad, legibilidad y flexibilidad. Por eso se tiene una clase de acción base. La cual la heredan los subtipos de instrucciones, ya sean mover, interactuar, rotar y condicional, todas ellas con sus campos diferenciadores que buscan tener lo necesario para llevar a cabo su funcionamiento.

Como implementación, se hizo un enum para tener definidas las acciones que se pueden realizar y así evitar el uso de comparaciones de cadenas de texto, siendo así más fácil de trabajar con las asignaciones a las instrucciones. Sumado a esto más información útil para otros sistemas del Serious Game. También este script tiene funciones propias para el funcionamiento de las acciones, tales como “Ejecutar”, “EliminarBoton” y sus variantes.

Figura 10

Parte de código de AccionBase



```
public abstract class AccionBase : ScriptableObject
{
    17 referencias
    public enum Acciones
    {
        Izquierda,
        Derecha,
        Abajo,
        Arriba,
        RotarIzq,
        RotarDer,
        Interactuar,
        Dejar,
        Tomar,
        Crear,
        Si,
        Ciclo
    }
    public string sAccion;
    public ControladorTablero controladorTablero;
    public ControladorMovimiento controladorMovimiento;
    public int iCantEnergia;
    public Acciones accion;
    public PoolBotones.TipoDeObjeto poolTipoObjeto;
    public GameObject gBoton;

    public abstract IEnumerator Ejecutar(ControladorRobots robot);
    12 referencias
    public virtual AccionBase Clonar()
    {
        AccionBase clone = Instantiate(this); // Crear una nueva instancia
        clone.sAccion = this.sAccion;
        clone.controladorTablero = this.controladorTablero;
        clone.controladorMovimiento = this.controladorMovimiento;
        clone.accion = this.accion;
        clone.poolTipoObjeto = this.poolTipoObjeto;
        return clone;
    }
    1 referencia
    public void EliminarBoton(ButtonManager bm, PoolBotones pb,ControladorRobots cr)
    {
        cr.iCantEnergia += iCantEnergia;
        pb.DevolverBoton(poolTipoObjeto, gBoton);

        int i = bm.aAccionesSelec.FindIndex(ab => ab == this);
        bm.aAccionesSelec.RemoveAt(i);
        cr.cMovimiento.ActualizarEnergia(cr.iCantEnergia);

    }
```

En la figura 10 se muestra el código de las acciones que interactúan con diversos sistemas, donde las acciones están integradas al método de programación por bloques. Estos contienen la información necesaria para cada robot, que posteriormente se utiliza en la gestión de turnos. Este sistema es responsable de coordinar los movimientos y acciones de los diferentes objetos en el juego. Todas las acciones descritas representan una simplificación destinada a ofrecer al lector una idea general del funcionamiento básico. Sin embargo, cada instrucción cuenta con un sistema de comportamiento complejo que opera detrás de escena para garantizar su correcta ejecución.

El sistema central denominado "Controlador Movimiento" actúa como el núcleo principal para gestionar las diferentes acciones que realizan los robots en el juego. Este controlador no solo se encarga de coordinar y ejecutar las instrucciones, sino también de garantizar que se sigan de manera ordenada y eficiente. En este sistema, las acciones programadas por el usuario son almacenadas previamente en una lista asociada a cada robot. Cuando un robot desea iniciar una

actividad, el "Controlador Movimiento" accede a esta lista para extraer y ejecutar las instrucciones. Este proceso incluye verificar la validez de las acciones, gestionar el consumo de recursos como la energía disponible, y coordinar las interacciones con otros sistemas del juego, como la detección de obstáculos o la recogida de objetos. Además, el "Controlador Movimiento" asegura la sincronización entre los diferentes robots, evitando conflictos en los turnos y asegurando que las acciones programadas se ejecuten sin interrupciones, respetando las limitaciones del entorno y las reglas del juego. Este enfoque permite mantener un control centralizado, mientras se garantiza la flexibilidad y adaptabilidad de las mecánicas del juego.

El proyecto incluye una arquitectura robusta compuesta por más de 30 scripts, cada uno diseñado con un propósito específico para abordar casos particulares dentro del sistema. Estos scripts están estructurados para garantizar que cada componente del juego cumpla su función de manera eficiente, modular y escalable. Algunos scripts están dedicados a gestionar elementos fundamentales, como la lógica principal de los robots, el procesamiento de las acciones programadas por el jugador y la interacción con el entorno de juego. Otros se centran en aspectos más específicos, como la detección de colisiones, la administración de recursos (como energía o recolectables), y la generación de objetivos dinámicos.

Además, existen scripts especializados en manejar el flujo general del juego, como la gestión de turnos, la sincronización entre robots y la verificación del cumplimiento de condiciones de los distintos objetivos. También se incluyen scripts para personalizar elementos visuales y efectos, garantizando una experiencia de usuario inmersiva y atractiva. Esta especialización permite no solo abordar los requerimientos únicos de cada caso dentro del juego, sino también facilitar futuras expansiones o modificaciones, manteniendo el código organizado y eficiente. La estructura modular asegura que los cambios en un script no afecten negativamente al resto del sistema, promoviendo un desarrollo ágil y sostenible.

Sumado a esto, una parte del sistema está compuesta por scripts dedicados a gestionar la interacción con los diversos tipos de menús que existen en el juego. Estos menús abarcan una amplia variedad de funciones esenciales para la experiencia del jugador, desde la visualización y

administración de los inventarios de objetos recolectados hasta el control del sistema de diálogos que guía la narrativa y proporciona contexto adicional a las misiones.

La implementación de un sistema de diálogos en el videojuego juega un papel crucial en proporcionar una guía clara y accesible para el jugador, especialmente en los momentos en los que se introducen nuevas mecánicas o se deben realizar acciones específicas. En versiones anteriores del juego, la explicación de cómo funcionaban las distintas características del juego recae completamente sobre la persona que ya conocía el funcionamiento, lo que generaba una fuerte dependencia de esta figura. Esto resultaba en una experiencia menos autónoma para el jugador, ya que el aprendizaje del juego dependía de la presencia y explicaciones de otros.

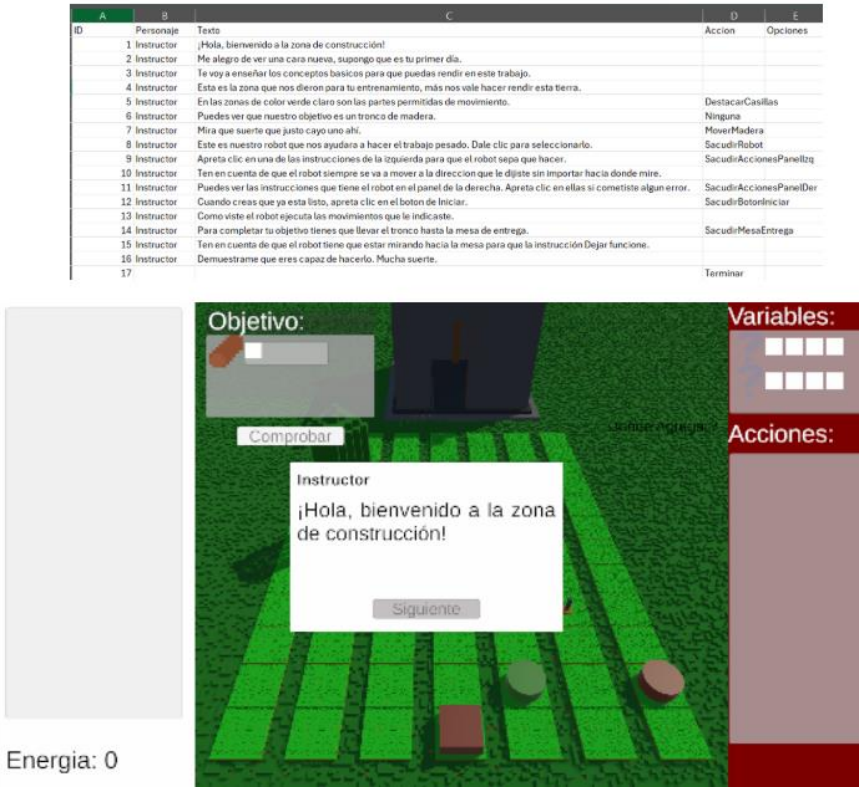
Al incorporar el sistema de diálogos, se elimina esta dependencia, ya que el propio juego es capaz de comunicar de manera estructurada y comprensible las instrucciones y reglas necesarias para progresar. Este sistema no solo facilita la comprensión de los jugadores novatos, sino que también mejora la accesibilidad del juego, permitiendo que los jugadores puedan interactuar con el entorno y aprender a través de la narrativa. Además, los diálogos pueden ser usados para guiar al jugador en momentos críticos o al presentar nuevas mecánicas, creando una experiencia de aprendizaje más fluida y menos fragmentada. En resumen, el sistema de diálogos hace que el juego sea más independiente, accesible y fácil de entender para los jugadores, lo que mejora significativamente la experiencia general.

El funcionamiento de este sistema comienza con la implementación de un archivo CSV (valores separados por coma), que se utiliza para almacenar y organizar los diálogos del juego de manera estructurada. Este archivo actúa como una base de datos de texto, donde se especifican los diferentes mensajes y acciones que el juego debe ejecutar durante las interacciones con el jugador. A partir de este archivo, el script "GestorDialogos" se encarga de procesar la información contenida en el CSV y generar un objeto de diálogo que contiene todos los atributos necesarios, como el texto que debe mostrarse al jugador y las condiciones de las interacciones.

Una vez que el objeto diálogo es creado, el "ControladorDialogos" se encarga de integrarlo en la interfaz de usuario (UI), asegurándose de que se muestre de manera adecuada en pantalla. Este controlador también gestiona las diferentes acciones que deben ejecutarse en respuesta a los diálogos, tales como animaciones o efectos visuales adicionales. Un ejemplo de estas acciones es el "sacudir" a un objeto específico en el juego, lo que se utiliza como una técnica visual para captar la atención del jugador o aclarar un punto importante.

Figura 11

Ejemplo de diálogos en CSV e implementación en juego.



En la figura 11 se muestra cómo funciona el sistema de diálogos y el resultado de como se muestra al jugador.

Los prefabs (objetos predefinidos reutilizables) se utilizarán para representar los robots, objetos interactivos y otros elementos importantes del juego. Esto permitirá la creación rápida de nuevos objetos sin tener que diseñarlos desde cero, también para tener almacenada la información

de los propios recursos. Como los scripts, son una gran variedad y de distinta naturaleza. Haciendo que se tenga uno para cada objeto con la lógica de programación propia para cada uno

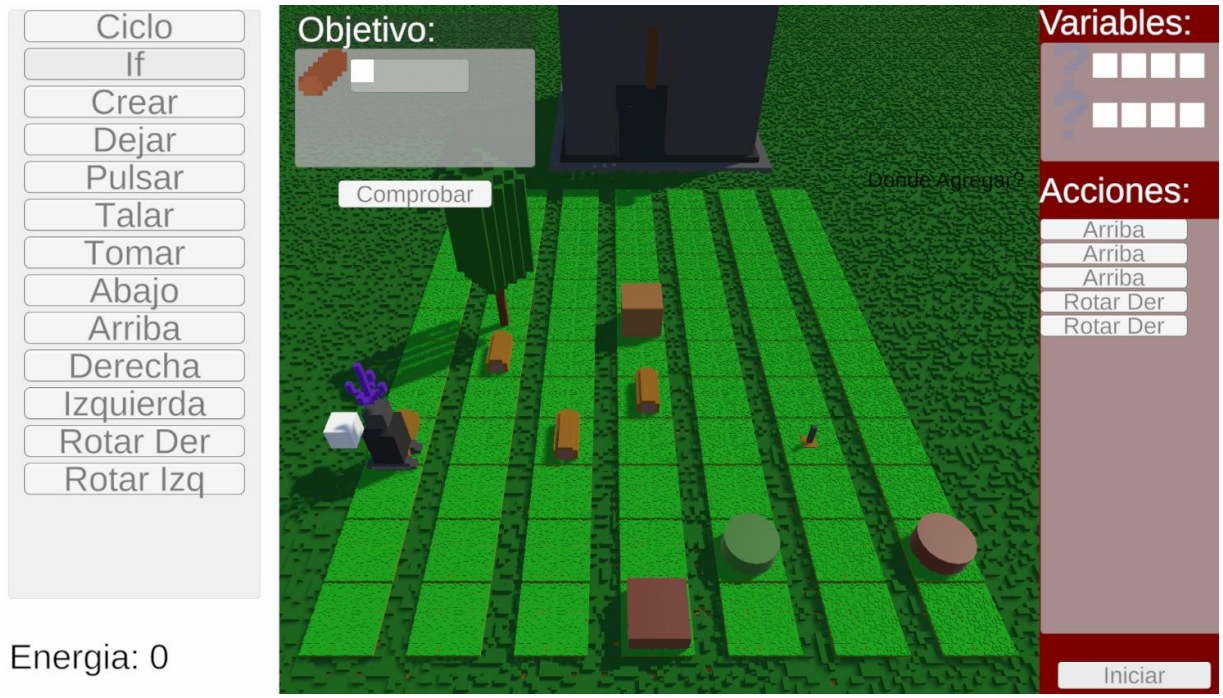
5.6.2 Sistema de Programación Visual

El sistema de programación por bloques se compone de dos partes fundamentales que permiten a los jugadores interactuar de manera intuitiva con el juego. La primera parte es el panel de la izquierda, donde se encuentran todas las acciones disponibles que el jugador puede seleccionar para incluir en su programación. Estas acciones están representadas por bloques visuales, cada uno de los cuales corresponde a una instrucción o comando que el robot podrá ejecutar durante el juego. El jugador puede arrastrar y soltar estos bloques para construir una secuencia lógica de acciones.

La segunda parte del sistema es el panel de la derecha, que funciona como un resumen o secuencia de las acciones seleccionadas por el jugador. En este panel, los bloques se ordenan en el orden en que el jugador desea que se ejecuten, creando una programación que luego se podrá ejecutar al presionar el botón de inicio. Este panel muestra claramente la secuencia completa de instrucciones que el robot debe seguir durante su turno, permitiendo al jugador revisar y ajustar la programación antes de que se lleve a cabo. En conjunto, estos dos paneles facilitan el proceso de programación al ofrecer una interfaz visual que hace el diseño de programas más accesible y comprensible para los jugadores.

Figura 12

Demostración de paneles de acciones en juego



Como se puede apreciar en la figura 12 se tienen diversos paneles para la jugabilidad e interacción con los objetos del juego.

5.6.3 Gráficos y Animaciones

El prototipo incluirá gráficos y animaciones sencillas para representar de manera clara y funcional las acciones realizadas por los robots y los cambios en el entorno del juego. Si bien no se buscará alcanzar un nivel de detalle visual de alta gama, se emplearán elementos gráficos básicos pero efectivos para asegurar que el jugador pueda entender y disfrutar de la experiencia de juego. El objetivo es mantener la simplicidad visual mientras se garantiza una jugabilidad fluida y comprensible.

Esto también se debe a la falta de un equipo especializado en el área estética del juego, lo que limita los recursos disponibles para desarrollar gráficos y diseño visual de alta calidad. Dado que el proyecto es un prototipo, el enfoque principal debe ser validar si la idea del juego es comprensible y funcional, en lugar de centrarse en la perfección visual. Los esfuerzos deben orientarse a garantizar que los jugadores comprendan el propósito y las mecánicas del juego, asegurando que la jugabilidad sea coherente y fluida, mientras se dejan los aspectos estéticos más elaborados para fases de desarrollo posteriores.

En cuanto a las animaciones, se implementarán acciones esenciales como el movimiento de los robots, la recolección de objetos y la construcción de estructuras, las cuales se gestionan utilizando el sistema de animaciones de Unity junto con el asset Dotween. Este asset facilitará la creación de animaciones básicas, particularmente aquellas relacionadas con el movimiento de los robots y la interacción con los objetos del entorno, lo que permitirá una experiencia dinámica y atractiva.

Cada acción en el juego estará acompañada de una animación específica que facilitará la comprensión de los procesos que están ocurriendo en el juego, mejorando la inmersión y el disfrute del jugador. Además, se integrarán efectos visuales como partículas y otras señales visuales para proporcionar retroalimentación inmediata cuando un robot complete una acción. Estos efectos no solo servirán para mantener el interés visual, sino también para comunicar claramente los resultados de las acciones realizadas, ayudando a los jugadores a entender las consecuencias de sus comandos y mejorando su experiencia de aprendizaje y juego.

5.7 Testing

El enfoque de las pruebas en este caso estará centrado en verificar el funcionamiento técnico y la jugabilidad del prototipo, asegurando que los jugadores puedan interactuar con el juego

de principio a fin sin encontrar errores críticos que impidan el progreso. A continuación, se detallan los aspectos clave que se evaluarán:

5.7.1 Prueba de jugabilidad

El objetivo central de esta prueba es asegurar que la experiencia de juego sea fluida y sin interrupciones técnicas. Para lograrlo, se llevó a cabo una serie de verificaciones en las que se valoraron distintos aspectos clave de la jugabilidad.

Se comprobó que los robots puedan ejecutar correctamente las acciones programadas por los jugadores, como moverse, recolectar objetos, interactuar con otros robots y cumplir con los objetivos establecidos en el juego. Esto incluye verificar que las instrucciones dadas por el jugador se traduzcan de manera adecuada en el comportamiento de los robots dentro del entorno del juego. Se evaluará también que las respuestas de los robots sean coherentes con las acciones programadas, como seguir una ruta específica o realizar tareas en un orden determinado.

Es importante que el sistema de gestión de turnos funcione sin problemas. En este sentido, se comprobó que los turnos se respeten correctamente, es decir, que después de que un robot realice sus acciones, el siguiente turno comience de manera fluida, sin errores ni bloqueos. Esto incluye garantizar que el sistema recargue correctamente la energía de los robots al inicio de cada turno, y que se mantenga un orden lógico en las interacciones entre robots y objetos en el entorno.

Las instrucciones básicas de programación que el jugador introduce, como los condicionales (if-else) y los bucles (for, while), deben ejecutarse correctamente durante el transcurso del juego. Se verificó que las condiciones asignadas a los robots sean evaluadas de manera precisa y que las acciones asociadas a esas condiciones se realicen sin fallos. Por ejemplo, si un jugador programa un robot para que recoja un objeto solo si tiene suficiente energía, se debe comprobar que el robot solo realice la acción si se cumple esa condición. De igual manera, se revisó que los bucles sean capaces de repetir las acciones esperadas sin que se detengan o causen errores en la lógica del juego.

Se validó que el jugador pueda acceder de manera fácil y clara a las diferentes opciones de programación de los robots. Esto incluirá verificar que el panel de acciones (como los bloques de programación) sea accesible, que las instrucciones puedan ser seleccionadas correctamente al espacio de trabajo, y que el orden de las instrucciones en el flujo de trabajo sea claro y comprensible. Además de que el jugador pueda aplicar correctamente condiciones, bucles y otras estructuras de control dentro del sistema de programación, sin que el juego se vuelva confuso o difícil de usar.

La secuencia de interacción debe permitir que el jugador avance de manera coherente a través de los diferentes niveles o etapas del juego. Se comprobó que el progreso del jugador se mantiene bien orientado en términos de dificultad y que los objetivos del juego evolucionen de acuerdo con el avance del jugador. Si el jugador completa un nivel correctamente, se validará que el juego pase a la siguiente fase sin errores, manteniendo la lógica de la dificultad progresiva y la adquisición de nuevas habilidades o conocimientos.

Se comprobó que la dificultad del juego se ajusta de manera adecuada a medida que los jugadores avanzan. Esto incluye la revisión de cómo los robots enfrentan desafíos cada vez más complejos en función de las habilidades que van desarrollando a lo largo del juego. La prueba de jugabilidad también se enfoca en asegurar que el diseño del juego permite a los jugadores progresar de manera lógica y que la curva de aprendizaje sea equilibrada, sin que el juego se vuelva demasiado difícil o fácil.

5.7.2 Prueba de rendimiento y estabilidad

Esta prueba tiene como objetivo garantizar que el juego pueda funcionar de manera constante y fluida bajo diversas condiciones, incluso durante sesiones largas de juego. La estabilidad y el rendimiento son factores clave que pueden influir directamente en la experiencia del jugador, ya que un rendimiento deficiente o errores frecuentes pueden interrumpir la jugabilidad y disminuir la calidad general del juego. A continuación, se detallan los aspectos que se evaluarán en esta prueba:

Durante las sesiones de prueba, se ejecutará el juego para evaluar cómo está el rendimiento general. En un escenario con varios robots programados para ejecutar acciones, el sistema debe ser capaz de gestionar y procesar todas las instrucciones de manera eficiente, sin que se produzcan caídas notorias en el rendimiento.

Las secuencias de acciones programadas, que pueden incluir movimientos complejos o interacciones entre robots y el entorno, también se someterán a pruebas de rendimiento. Se evaluará cómo el juego maneja la ejecución de comandos complejos.

La estabilidad del juego también se pone a prueba al simular diversas condiciones del sistema, como diferentes configuraciones de hardware o variaciones en los recursos del sistema (como la memoria y el procesamiento). Se prueba el comportamiento del juego en dispositivos con especificaciones más bajas, así como en aquellos con mayor capacidad, para asegurarse de que la experiencia de juego no se vea comprometida en ningún caso. La idea es que el juego sea accesible para una amplia gama de usuarios, sin depender de equipos de alto rendimiento.

En la prueba de rendimiento, también se examinará cómo el juego maneja posibles errores o caídas inesperadas del sistema, como un mal funcionamiento durante la ejecución de acciones programadas o problemas de sincronización entre los robots y el entorno. Se verifica que, en caso de que ocurra un error, el juego sea capaz de recuperarse sin necesidad de reiniciar o perder el progreso del jugador. Además, se evaluará si los errores que ocurren durante el juego son claros y se comunican de manera efectiva al jugador, con opciones para solucionar o evitar que ocurran nuevamente.

Se realizan pruebas de rendimiento en sesiones prolongadas de juego, para asegurarse de que el juego mantenga una experiencia fluida sin caídas o errores técnicos a medida que avanza el tiempo de juego. Estas pruebas ayudarán a identificar posibles fugas de memoria o problemas que

surgen después de varias horas de juego, asegurando que el juego siga siendo estable y jugable en sesiones largas, lo cual es crucial en un prototipo que pueda ser jugado durante tiempos extendidos de prueba.

Otro aspecto clave que se evalúa en esta prueba es el rendimiento de la interfaz de usuario (UI) bajo condiciones de carga. La UI, que incluye menús, paneles de acciones, inventarios y retroalimentación visual, debe mantenerse estable y fluida durante toda la interacción del jugador. Se verificará que los elementos visuales y de interacción, como los menús y botones, no afecten negativamente el rendimiento ni causen retrasos al cargarse o al procesarse las interacciones.

5.7.3 Prueba de interfaz de usuario (UI)

La interfaz de usuario (UI) debe ser diseñada de manera que sea intuitiva, clara y fácil de navegar, asegurando que los jugadores puedan interactuar con el juego sin obstáculos ni confusión. Esta prueba abarca los siguientes aspectos clave:

Se evalúa la estructura de los menús principales y secundarios para garantizar que los jugadores puedan localizar fácilmente las opciones y funciones necesarias. Esto incluye la organización de menús, la lógica de navegación entre pantallas y la disposición de elementos como botones, paneles y desplegables.

Se revisa que los elementos de la interfaz, como íconos, etiquetas, textos y colores, sean visualmente claros y fácilmente identificables. Se pondrá énfasis en el contraste y la legibilidad, verificando que los textos sean comprensibles en todo tipo de pantallas y resoluciones, especialmente en dispositivos con características técnicas variadas. Todos los botones, controles deslizantes, casillas de selección y otros elementos interactivos serán probados para garantizar que respondan correctamente a las acciones del usuario. Se verificará que no existan botones inactivos, acciones que no se ejecuten o elementos que funcionen de manera incorrecta.

Durante la prueba, se evalúa la fluidez de las transiciones y animaciones dentro de la interfaz, asegurando que no haya retrasos perceptibles que puedan interrumpir la experiencia del usuario. Esto incluye animaciones al abrir menús, cambiar configuraciones o realizar acciones en el juego.

Se analiza la efectividad de la retroalimentación visual y sonora al interactuar con la UI. Esto incluye la confirmación de acciones (como clics en botones o selección de opciones) y la notificación al usuario sobre errores, advertencias o estados del juego.

Se considera la inclusión de opciones que mejoren la accesibilidad, como la posibilidad de ajustar el tamaño del texto, habilitar un modo de alto contraste o incluir alternativas para jugadores con discapacidades.

5.7.4 Prueba de animaciones y efectos visuales

El objetivo de esta prueba es garantizar que las animaciones y los efectos visuales del juego se ejecuten de manera correcta, fluida y sincronizada con las acciones del jugador, mejorando así la inmersión y la experiencia general, teniendo en cuenta de que al ser un prototipo este apartado no se destaca. A continuación, se detallan los aspectos clave que serán evaluados:

Se comprueba que las animaciones de movimiento, recolección de objetos y construcción de estructuras se ejecuten de manera fluida y acorde con las acciones realizadas por los robots. Esto incluye verificar transiciones suaves entre estados, como caminar, detenerse o realizar una acción específica.

Se evalúa que las animaciones respondan inmediatamente a los comandos del jugador, asegurando que no haya retrasos perceptibles entre la acción ejecutada y la animación correspondiente.

Se valida la calidad y correcta implementación de los efectos visuales, como partículas para indicar la recolección de objetos, el uso de energía, o cambios en el entorno. Esto incluye efectos de polvo, chispas o destellos que acompañen las acciones de los robots.

Se evalúa que los efectos visuales proporcionan una retroalimentación clara e intuitiva al jugador sobre el resultado de sus acciones. Por ejemplo, un efecto de partículas puede señalar que un objeto fue recolectado exitosamente o que una estructura fue completada.

Se observa cómo las animaciones y los efectos visuales impactan el rendimiento del juego en diferentes dispositivos. Esto incluye medir la tasa de cuadros por segundo (FPS) y evitar problemas como caídas de rendimiento o retrasos debido a una sobrecarga de efectos visuales.

Se evalúa cómo las animaciones y efectos visuales interactúan con el entorno del juego, asegurando que no se generen errores visuales, como superposiciones incorrectas, objetos flotantes o elementos que desaparecen de manera inesperada.

Se verifica que las animaciones y efectos visuales se mantengan consistentes y bien renderizados en diferentes resoluciones de pantalla, asegurando una experiencia uniforme para todos los jugadores.

5.7.6 Identificación de errores críticos

Se priorizo la detección de errores que puedan interrumpir o bloquear el progreso del jugador durante el desarrollo del juego. Este enfoque es fundamental para garantizar una experiencia fluida y sin interrupciones. Los principales aspectos que se evaluarán incluyen:

Se analizó si los jugadores pueden interactuar correctamente con todos los elementos del entorno, como recolectar objetos, activar mecanismos o construir estructuras. Los errores que impidan o dificulten estas interacciones serán identificados y registrados.

Se evalúa si las instrucciones programadas por los jugadores, como condicionales, bucles y movimientos, se ejecutan de manera correcta y consistente. Esto incluye verificar que no haya discrepancias entre la lógica de programación del jugador y el comportamiento esperado en el juego.

Se revisa que los sistemas subyacentes, como el controlador de movimiento, la gestión de turnos y el sistema de energía, funcionen sin inconsistencias que puedan generar comportamientos impredecibles o resultados incorrectos.

Se identifican situaciones en las que el jugador no pueda avanzar en el juego debido a errores críticos, como niveles que no se completan a pesar de cumplir los requisitos, fallos en la transición entre niveles o sistemas que no reaccionen como deberían.

Se evalúa si los jugadores pueden navegar y utilizar la interfaz de manera intuitiva. Esto incluye la detección de elementos que no respondan al clic, menús que no se desplieguen correctamente o botones que no cumplan su función asignada.

Se prueba el juego en diferentes configuraciones de hardware y software para identificar posibles errores técnicos, como cuelgues, cierres inesperados o incompatibilidades con ciertos dispositivos.

Se verifica que las animaciones, efectos visuales y elementos interactivos del entorno estén sincronizados con las acciones del jugador. Esto incluye problemas como movimientos que no coincidan con las instrucciones o efectos visuales que no se activen en el momento adecuado.

5.8 Resultados de implementación

Los resultados son de gran relevancia, ya que permiten validar la efectividad del prototipo como herramienta de apoyo al aprendizaje de la programación. Evaluar la experiencia de los jugadores proporciona evidencia sobre el grado de comprensión, usabilidad y motivación que genera el sistema, elementos fundamentales en el diseño de entornos educativos interactivos. Además, estos resultados permiten identificar aspectos del diseño que requieren ajustes o mejoras, asegurando que el desarrollo futuro del prototipo esté alineado con las necesidades pedagógicas y cognitivas de los jugadores. En este sentido, la retroalimentación recogida no solo orienta el perfeccionamiento del producto, sino que también aporta información importante para investigaciones posteriores en el ámbito del aprendizaje basado en juegos.

5.8.1 Información técnica del prototipo

Es importante saber que requisitos mínimos son necesarios para poder ejecutar el serious game, para eso a continuación se presentan los aspectos más importantes.

Tabla 2

Requerimientos mínimos de prototipo.

	Mínimo
Sistema operativo	Windows 10
Procesador	i5-1135G7 2.40GHz
Memoria	8.00 GB
Gráficos	Intel Iris Xe Graphics
Almacenamiento	100 mb

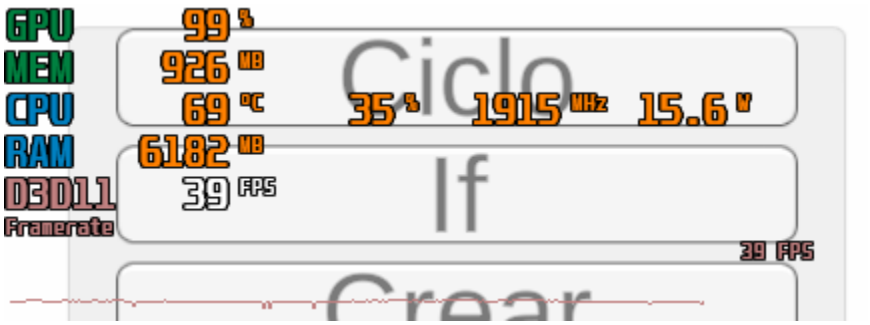
Nota: Este es el equipo de menor capacidad en el que se probó.

En la tabla 2 se muestra información técnica que ayuda a tener una idea general, de cómo es el rendimiento del prototipo y que equipo se debe tener para una experiencia mínima, la información se debe actualizar con nuevos requisitos a medida que se pruebe con más computadores, además da información del rendimiento del prototipo. Para poder medir las estadísticas del prototipo se utilizó el programa MSI Afterburner junto a RivaTuner statistics

server. La forma que tiene de mostrar los datos es por medio de valores en la pantalla mientras se ejecuta el juego.

Figura 13

Ilustración de rendimiento del prototipo dado por MSI Afterburner



En la figura 13 se muestran los datos que se obtuvieron durante la ejecución del prototipo y se pueden obtener los resultados de las pruebas realizadas con el equipo con las especificaciones establecidas como mínimas en la tabla 2 fueron de 40 fps de media. Lo que indica un rendimiento aceptable, pero mejorable, más aun teniendo en cuenta de que el prototipo cuenta con gráficos y animaciones básicas para probar el funcionamiento de este.

El prototipo fue pensado para ser jugado en un computador de sobremesa, también compatible con notebook con la potencia mínima requerida en la tabla 2, esto excluye a celulares, tables y consolas.

La forma de distribución del prototipo por ahora está limitada, ya que al no contar con una versión estable y testeada por una mayor cantidad de usuarios para identificar y pulir detalles en implementación que no se ven a simple vista. Además, por la propia naturaleza de un prototipo se presupone que no es un producto que se entregara al público. Esto hace que si se quiere obtener se tenga que contactar con el creador de este.

5.8.2 Resultados del Testing

Con el propósito de evaluar la experiencia de los usuarios en relación con la propuesta lúdico-educativa desarrollada, se diseñó y aplicó un instrumento de recolección de datos estructurado en torno a una escala de tipo Likert (valores de 1 a 5). Esta encuesta contempló diversos aspectos vinculados al uso del sistema, tales como la claridad de las mecánicas implementadas, la percepción de dificultad, la comprensión de conceptos clave de programación, así como la valoración general del entorno interactivo. Los resultados obtenidos, presentados a continuación en forma de promedios por ítem, permiten realizar un análisis integral sobre la efectividad y usabilidad del prototipo, identificando tanto sus fortalezas como las oportunidades de mejora en su diseño pedagógico y funcional.

Las variables consideradas en la evaluación incluyeron la facilidad para programar instrucciones, la comprensión de los sistemas implementados en el juego (movimiento, rotación, recolección, creación de objetos y gestión de variables), la claridad de los elementos visuales, la dificultad percibida de las tareas, la progresión del desafío, la motivación del jugador y la percepción del aprendizaje en programación. Se presentan en el siguiente grafico de manera visual.

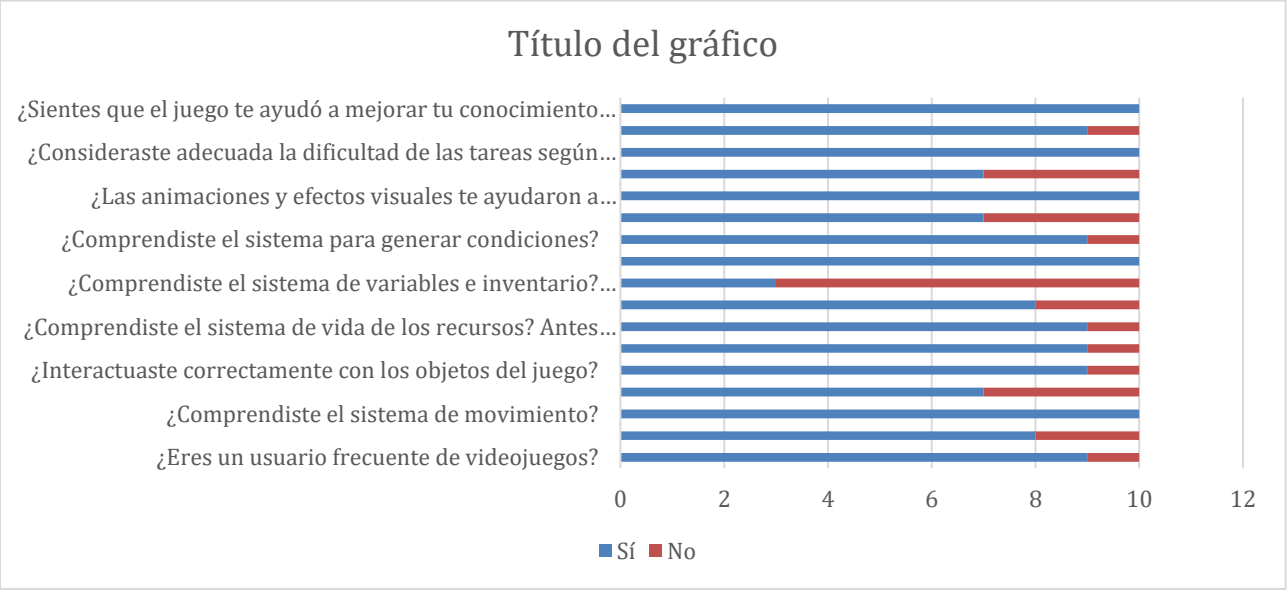
Figura 14

Gráfico de resultados apartados de encuestas de 1/5

Como se puede ver en el gráfico de la figura 14, los resultados reflejan una valoración positiva del prototipo por parte de los participantes. La mayoría de las variables evaluadas presentan promedios elevados, lo que indica una buena comprensión de las mecánicas del juego, una percepción adecuada de la dificultad y una experiencia de usuario satisfactoria. Destacan especialmente las altas puntuaciones en motivación, calidad visual y percepción de aprendizaje, lo que sugiere que el entorno interactivo logró involucrar al usuario y facilitar la asimilación de conceptos básicos de programación.

Figura 15

Gráfico de resultados apartados de encuestas de Si/No

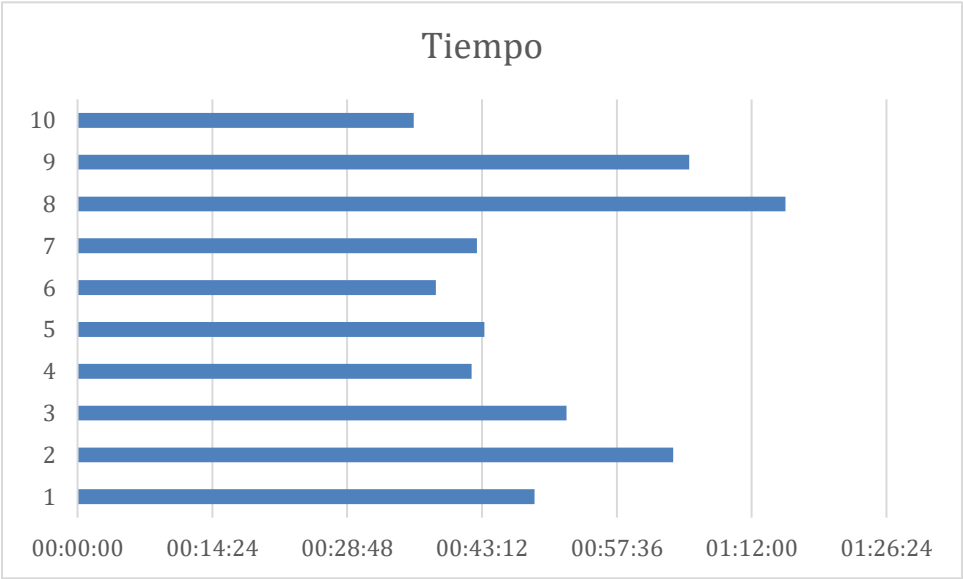


En la figura 15 se puede evidenciar que la recepción del prototipo por parte del grupo de participantes fue mayoritariamente positiva. Los resultados reflejan una buena aceptación general del sistema, lo cual sugiere que el diseño y la implementación inicial del prototipo lograron cumplir con los objetivos propuestos en cuanto a experiencia del jugador y comprensión de las mecánicas principales. Esta entrega una primera validación del enfoque adoptado, indicando que el prototipo tiene el potencial de ser una herramienta eficaz para el propósito educativo planteado.

Como ultimo apartado importante a destacar es el tiempo promedio de prueba del usuario, el siguiente grafico lo muestra visualmente.

Figura 16

Gráfico de resultados apartados de encuestas de Si/No



Con los datos que se muestran en el gráfico de la figura 16, el tiempo promedio de juego fue de aproximadamente 53 minutos por jugador, con una duración mínima de 35:54 minutos y una máxima de 1:15:37 horas. Estos datos permiten establecer una estimación del tiempo requerido para completar la experiencia propuesta y evaluar la extensión y complejidad del prototipo en función del tiempo de interacción, a continuación, se detallan los aspectos más importantes.

Estabilidad y errores críticos

Durante la prueba, ningún jugador pudo completar todas las tareas sin errores críticos, lo que indica que hay fallos que afectan la experiencia de juego. La estabilidad del sistema recibió una calificación promedio de 3.4/5, con una calificación mínima de 1/5, lo que sugiere que algunos errores son más severos que otros y pueden estar impidiendo el progreso de los jugadores.

Comprensión de mecánicas clave

La mayoría de los jugadores lograron comprender el sistema de movimiento, con 9 de 10 jugadores entendiéndolo correctamente. Sin embargo, el sistema de rotación presentó más dificultades, con 3 jugadores que no lo comprendieron inicialmente. Algo similar ocurrió con el sistema de energía, donde 4 de 10 jugadores no entendieron su funcionamiento.

El sistema de interacción con objetos también presentó problemas, ya que solo 4 de 10 jugadores lograron interactuar correctamente sin asistencia. Además, el uso del botón "Comprobar" no fue completamente intuitivo, ya que 3 jugadores no lo presionaron en ningún momento.

Uno de los sistemas que más dificultades presentó fue el de recolección de recompensas mediante la palanca, ya que solo 2 de 10 jugadores lo comprendieron adecuadamente. Por otro lado, el sistema de crafteo tuvo una mejor recepción, con 8 de 10 jugadores entendiéndolo correctamente, y 9 de 10 lograron crear el hacha con éxito.

El sistema de vida de los recursos fue comprendido por 8 de 10 jugadores, lo que indica que su funcionamiento es bastante claro. Sin embargo, el sistema de variables e inventario fue uno de los aspectos más problemáticos, ya que solo 1 jugador lo comprendió completamente. En cuanto a la generación de condiciones con la instrucción if, 9 de 10 jugadores entendieron la teoría, pero 7 de ellos tuvieron dificultades al aplicarla correctamente en la programación.

Dificultades y necesidad de intervención

Todos los jugadores necesitaron ayuda del instructor en algún momento, lo que refleja que varias mecánicas no son lo suficientemente intuitivas. En promedio, se realizaron 19

intervenciones por jugador, con un caso extremo de 38 intervenciones, lo que sugiere que ciertos aspectos del juego pueden requerir ajustes en su presentación o explicación.

Además, 9 de 10 jugadores reportaron dificultades para completar algún objetivo, lo que refuerza la idea de que algunas mecánicas no son completamente claras. Uno de los puntos más confusos fue el hecho de que los robots pueden moverse simultáneamente, ya que solo 3 de 10 jugadores comprendieron esta mecánica, aunque 9 de 10 lograron entender el sistema de movimiento para ambos robots una vez explicado.

Encuesta para el Jugador

Es la prueba para la persona que no conoce el funcionamiento del prototipo (véase anexo 2) y que será la persona que entregue la retroalimentación necesaria para comprobar si los conceptos son válidos y entendibles. Aquí tienes el análisis estructurado de los resultados de los jugadores:

Perfil de los jugadores

Los participantes en la prueba tenían distintos niveles de conocimiento en programación, con un promedio de 2.3 en una escala del 1 al 5, lo que indica un nivel de principiante a intermedio. La mayoría de los jugadores eran usuarios frecuentes de videojuegos (9 de 10), aunque solo 6 de 10 habían jugado previamente videojuegos de automatización o gestión de recursos, lo que podría influir en su familiaridad con ciertas mecánicas del juego.

Facilidad de uso y comprensión de mecánicas

En general, la programación de instrucciones fue considerada relativamente fácil, con una calificación promedio de 4.2/5. Todos los jugadores comprendieron el sistema de movimiento, pero el sistema de rotación presentó algunas dificultades, ya que 3 jugadores no lo comprendieron correctamente.

La interacción con los objetos del juego fue clara para la mayoría, aunque uno de los jugadores tuvo problemas. Sin embargo, uno de los sistemas más confusos fue el de recolección de recompensas mediante la palanca, que antes de una explicación obtuvo una calificación baja de 2.8/5, aunque después de explicarlo, mejoró a 4.6/5, lo que indica que la mecánica no es intuitiva por sí sola.

Otro punto de dificultad fue el sistema de vida de los recursos, donde uno de los jugadores no lo comprendió ni antes ni después de la explicación. Además, el sistema de variables e inventario resultó ser el más problemático, ya que solo 3 jugadores lo comprendieron antes de la explicación, aunque tras recibirla, todos lograron entenderlo.

El sistema para generar condiciones fue bien comprendido por la mayoría, aunque 4 jugadores tuvieron dificultades con la instrucción if, y la configuración de condicionales y bucles fue calificada en promedio con 4.1/5, lo que indica que algunos usuarios encontraron el proceso algo complicado.

Aspectos visuales y usabilidad

El juego recibió una respuesta positiva en términos de animaciones y efectos visuales, ya que todos los jugadores coincidieron en que ayudaban a comprender lo que ocurría en el juego. La calidad visual de las animaciones fue calificada en promedio con 4.3/5. Sin embargo, la percepción del progreso en la construcción de la casa fue menos clara, ya que 3 de los jugadores no sintieron que avanzaban conforme completaron los objetivos, posiblemente debido a que no se le daba la importancia que se merece, ya que la idea principal y objetivo final es la construcción de dicha estructura.

En cuanto a la colocación de acciones, la mayoría no encontró dificultades significativas, con un promedio de 1.6/5 en la escala de confusión, donde 5 indica mucha confusión. Sin embargo, algunos jugadores mencionaron que los sprites de ciertos objetos, como la roca y la losa, no eran suficientemente representativos, lo que podría afectar la claridad visual del juego. Aspectos que al

tratarse de un prototipo y no contar con un área dedicada puramente al estilo artístico hace que este campo se valore de manera negativa.

El menú de creación de objetos fue, en general, comprensible, con calificaciones de 4 a 5, salvo por un jugador que mencionó que algunos objetos no eran claros.

Dificultad y progresión

La dificultad del juego fue considerada adecuada para el nivel de conocimientos de los jugadores, ya que todos respondieron afirmativamente a esta pregunta. La progresión de dificultad recibió una calificación positiva de 4.6/5, lo que sugiere que el ritmo del juego es bien recibido.

Sin embargo, la instrucción if fue considerada más desafiante, con una calificación promedio de 4.1/5 en términos de dificultad. Esto refuerza la idea de que los conceptos de condicionales necesitan más apoyo o mejoras en su implementación dentro del juego. Aquí se ve realmente el potencial del serious game, ya que las personas que no estaban familiarizadas con este apartado, luego de realizar una breve explicación del funcionamiento y darle un ejemplo práctico. Todos lograron completar de manera satisfactoria el reto planteado para este tema.

Motivación y opinión general

El 90% de los jugadores se sintieron motivados para completar todas las tareas del juego, lo que indica que el diseño y la mecánica del juego generan interés.

Las opiniones sobre mejoras incluyen aspectos como mayor feedback visual, mejor presentación de la información en el panel de la izquierda, mejoras en los diálogos y adición de música y sonidos. También se mencionó la necesidad de un sistema para guardar instrucciones, lo que sugiere que los jugadores podrían querer experimentar más con su programación sin perder progreso o poder volver a utilizar cadenas de acciones ocupadas anteriormente.

La calificación promedio del juego fue de 4.6/5, lo que indica una valoración positiva. Además, todos los jugadores sintieron que el juego les ayudó a mejorar su conocimiento en programación, con un promedio de 4.6/5 en cuanto al impacto educativo. Esta estadística puede estar alterada por el hecho de que la forma de recibir la puntuación de los jugadores, ya que el evaluador le pregunta directamente al usuario, lo que hace que sea más complicado el dar notas negativas.

CAPÍTULO IV: CONCLUSIONES Y RECOMENDACIONES

El desarrollo del prototipo de Serious Game representa una herramienta alternativa al aprendizaje de programación en estudiantes universitarios de primer año sin experiencia previa en el área, además de ser una solución innovadora para abordar las necesidades educativas. De acuerdo con la progresión diseñada para el jugador, el estudiante avanzará desde el nivel 1 hasta el nivel 6, lo que permite aumentar las oportunidades que comprenda y domine cada concepto antes de proceder al siguiente. Conforme a lo anterior, se logra evidenciar que este enfoque combina elementos lúdicos y pedagógicos para abordar la brecha educativa identificada en la problemática y se ven aspectos para poder seguir desarrollando esta herramienta.

La implementación bajo el framework MDA permitió estructurar mecánicas, tales como la programación basada en bloques visuales, dinámicas que promueven la resolución de problemas en tiempo real, estéticas diseñadas para mantener el interés y la motivación del estudiante, se logró un entorno propicio para el aprendizaje activo. También incluye dinámicas y estéticas enfocadas en un aprendizaje progresivo, práctico y motivador. A través de pruebas iterativas, se confirmó la validez del diseño y su potencial para mejorar la comprensión de conceptos clave de programación, como algoritmos, condicionales y bucles, mediante tareas interactivas y visuales. Los resultados obtenidos durante el diseño e implementación demuestran que la aplicación del framework MDA (Mecánicas, Dinámicas y Estéticas) permite crear una experiencia educativa interactiva y atractiva desde el punto de vista jugable, ya que con esta se tiene un enfoque claro de qué es lo que se busca a la hora de implementar las diversas partes que lo componen.

El proceso iterativo utilizado permitió no solo ajustar y optimizar las funcionalidades del juego, sino también validar su el correcto funcionamiento mediante pruebas preliminares con usuarios, destacando el impacto positivo en la percepción de conceptos como algoritmos, condicionales y bucles. Asimismo, la estructura de niveles con dificultad progresiva indica que los estudiantes puedan avanzar de manera gradual y sólida, pero falta comprobar si se puede adaptar para evitar la necesidad de un instructor para que el prototipo se adapte al ritmo de aprendizaje y consolidando las bases necesarias para enfrentar desafíos más avanzados.

A pesar de los logros alcanzados con este proyecto, se reconocen varias limitaciones significativas que afectan su alcance y su potencial como herramienta educativa integral. En primer lugar, el prototipo está diseñado exclusivamente para abordar los fundamentos de la programación, lo que implica que no incluye contenidos más avanzados como estructuras de datos complejas, algoritmos avanzados o programación orientada a objetos. Esto restringe su utilidad a etapas iniciales del aprendizaje y limita el interés de estudiantes con conocimientos intermedios o avanzados.

Los tiempos ajustados de desarrollo afectaron la calidad general del prototipo. Algunas mecánicas y elementos visuales no alcanzaron el nivel de pulido deseado, lo que podría impactar negativamente la experiencia del usuario y la percepción general del juego como una herramienta bien elaborada. Estas restricciones hacen evidente la necesidad de dedicar recursos adicionales en futuras versiones para mejorar la calidad, ampliar el alcance de los contenidos y garantizar que el producto cumpla con altos estándares tanto educativos como técnicos. Estas limitaciones, si bien acotan el impacto inmediato del proyecto, también abren oportunidades para futuras investigaciones y desarrollos que integran funcionalidades más sofisticadas, ampliando así el alcance del Serious Game a un público más especializado como mejorando el rendimiento y arreglando errores que son inevitables que aparezcan con la naturaleza de un proyecto tan grande y desarrollado por una persona, junto con el poco tiempo que se tiene para completarlo.

La principal limitación que presenta este proyecto radica en la ausencia de un sistema formal y estructurado para evaluar la efectividad del método propuesto en términos de impacto real en el aprendizaje de los alumnos. Si bien el diseño del juego está orientado hacia la enseñanza de conceptos fundamentales de programación, aún no se han construido métricas claras ni estudios que respalden que los estudiantes están adquiriendo y reteniendo estas habilidades de manera efectiva. Tampoco se han realizado pruebas piloto o recopilado datos concretos que permitan analizar el progreso educativo de los usuarios, identificar áreas de mejora o ajustar las mecánicas del juego para optimizar los resultados educativos.

Los principales problemas identificados durante el desarrollo, incluyen la presencia de errores críticos que impiden completar el juego, una alta dependencia del instructor para avanzar y dificultades específicas con el sistema de recolección de recompensas, así como con el uso de variables e inventario. Sin embargo, todos estos problemas fueron solucionados en la etapa de prueba.

Para mejorar la experiencia de los jugadores, según los resultados de la etapa de prueba, se deben proporcionar instrucciones más claras y tutoriales interactivos para reducir la confusión en el uso de mecánicas claves. También, se detectó una necesidad significativa de la intervención del instructor, por lo que existen oportunidades de mejora en la implementación de la interfaz asociada al sistema de recolección de recompensas y simplificar la gestión de variables e inventario. Aún así, el juego recibió una evaluación positiva en términos de experiencia de usuario, aprendizaje y motivación.

En este sentido, para mejorar la experiencia, se recomienda hacer más intuitivos los sistemas de recolección e inventario, añadir más feedback visual, y optimizar la interfaz para mostrar información clave de manera más clara. Adicionalmente, se podría implementar un sistema para guardar instrucciones, lo que mejoraría la experiencia de los jugadores que desean experimentar con diferentes estrategias de programación.

Las proyecciones futuras se orientan hacia la mejora y expansión del prototipo, explorando la incorporación de nuevas temáticas y funcionalidades que permitan abarcar niveles más avanzados de programación, así como la adaptación a diferentes contextos educativos. También se plantea la posibilidad de realizar estudios longitudinales para evaluar el impacto del juego en el desempeño académico de los estudiantes a largo plazo, así como su integración con otras herramientas y plataformas educativas

Este trabajo no solo quiere reforzar la importancia de los juegos serios como herramientas educativas, sino que también destaca el valor de enfoques innovadores en la enseñanza de

habilidades técnicas. El prototipo desarrollado sienta las bases para la implementación de una estrategia pedagógica basadas en la gamificación, contribuyendo a cerrar la brecha en la educación en programación y ofreciendo una experiencia significativa para los estudiantes en su transición hacia el mundo de las ciencias de la computación.

REFERENCIAS

1. Ojeda-Lara, Oscar Gilberto, & Zaldívar-Acosta, Marisa del Socorro. (2023). Gamificación como Metodología Innovadora para Estudiantes de Educación Superior. *Revista Tecnológica-Educativa Docentes 2.0*, *16*(1), 5-11. Epub 25 de enero de 2024. https://doi.org/10.37843/rted.v16i1.332
2. Sánchez-Alonso, R. E., Ortega-Moody, J., González-Barbosa, J. J., & Reyes-Morales, G. (2017). Uso de Plataformas para el Desarrollo de Aplicaciones Virtuales en el Modelado de Robot Manipuladores. Revista Iberoamericana De Automática E Informática Industrial, 14(3), 279–287. https://doi.org/10.1016/j.riai.2017.04.001
3. Ángel-Díaz, C. M., Segredo, E., Arnay, R., & León, C. (2020). *Simulador de Robótica Educativa para la promoción del Pensamiento Computacional | Revista de Educación a Distancia (RED).* Obtenido de Revistas UM: https://revistas.um.es/red/article/view/410191
4. Atlassian. (s.f.). *Qué es scrum y cómo empezar.* Obtenido de Atlassian: https://www.atlassian.com/es/agile/scrum
5. Checa, R. d. (2011). *LA INNOVACIÓN METODOLÓGICA EN LA ENSEÑANZA DE LA.* Obtenido de https://dialnet.unirioja.es/descarga/articulo/6043076.pdf
6. Dapozo, G., Greiner, C., Petris, R., Irrazabal, E., Company, A., Espíndola, M., & Medina, Y. (15 de 06 de 2022). *Evaluación de habilidades de pensamiento computacional al inicio de una asignatura de programación en una carrera de informática.* Obtenido de RIUNNE: http://repositorio.unne.edu.ar/handle/123456789/50684
7. *Educo*. (28 de junio de 2024). Obtenido de El blog de educo: https://www.educo.org/blog/como-aplicar-el-pensamiento-computacional-aula
8. Gamelearn. (2024). Obtenido de Gamestrategies: https://gamestrategies.io/es/blog/la-teoria-del-game-based-learning/

9. Huumit. (23 de febrero de 2024). *Las Habilidades más Demandadas en la Industria IT: Perspectivas para el 2024.* Obtenido de linkedin: https://es.linkedin.com/pulse/las-habilidades-m%C3%A1s-demandadas-en-la-industria-perspectivas-para-wbr7f
10. Jones, E., Jimenez, C., Ormeño, P., & Poblete, N. (Junio de 2022). *Metodologías activas para la enseñanza de programación a estudiantes de ingeniería civil informática.* Obtenido de Scielo: https://www.scielo.cl/scielo.php?pid=S0718-50062022000300053&script=sci_arttext
11. Lázaro Alvarez, N. (2020). *Acciones Tutoriales con TIC atendiendo a factores predictivos de la deserción estudiantil en carreras de Ingeniería Informática.* Obtenido de Repositorio Digital: https://repositorio.uci.cu/handle/123456789/10115
12. Pimentel, F., Marques, M., & Sales Junior, V. (2022). *Estrategias de aprendizaje a través de los juegos digitales en un contexto universitario.* Obtenido de Revista Comunicar: https://www.revistacomunicar.com/index.php?contenido=detalles&numero=73&articulo=73-2022-07
13. Putra, S. D., & Yasin, V. (3 de Julio de 2021). *MDA Framework Approach for Gamification-Based Elementary Mathematics Learning Design.* Obtenido de IJESTY: https://digilib.esaunggul.ac.id/public/UEU-Journal-21487-11_1830.pdf
14. Rodriguez Jimenez, O. R., & Garcia Pinilla, J. I. (2020). *Una guía para el desarrollo de un videojuego educativo en educación superior.* Obtenido de Cultura Educación Sociedad: https://ojstest.certika.co/culturaeducacionysociedad/article/view/2759
15. Trejos Buriticá, O. I. (2018). *Ejercicios en computador vs. ejercicios en papel para enseñar a programar: un estudio comparativo.* Obtenido de https://dialnet.unirioja.es/descarga/articulo/7023011.pdf
16. Treviño, R. (2025). *Jugar sin miedo a equivocarse: el poder educativo de los serious games.* Obtenido de https://tecscience.tec.mx/es/humano-social/serious-games-aprendizaje/
17. Viñas, M. (2022). *Nueva estrategia educativa*. Obtenido de https://sedici.unlp.edu.ar/bitstream/handle/10915/147803/Documento_completo.+Vi%C3%B1as+(final).pdf-PDFA.pdf?sequence=1&isAllowed=y

ANEXOS

Anexo 1.

Test para evaluador.

Estas son las preguntas que el evaluador tiene que responder mientras observa el funcionamiento del prototipo.

Fluidez del juego	
	¿El jugador pudo completar todas las tareas sin interrupciones por errores críticos? (Sí/No)
	En una escala del 1 al 5, ¿cómo calificaría la estabilidad del sistema durante la prueba?
Interacciones del jugador	
	¿El usuario comprendió el sistema de movimiento?
	¿El usuario comprendió el sistema de rotación?
	¿Comprendió el funcionamiento del sistema de energía?
	¿Interactuó correctamente con los objetos del juego? (Sí/No)
	¿Fue necesaria la intervención del instructor?
	En caso afirmativo, ¿cuántas veces?
	¿Presionó el botón de comprobar? (Sí/No)
	¿Se encontró con dificultades para completar algún objetivo? (Sí/No)
	¿Comprendió el sistema de recolección de recompensas mediante la palanca?
	¿Comprendió el sistema de crafteo?
	¿Logró crear el hacha?
	¿Comprendió el sistema de vida de los recursos?
	¿Comprendió el sistema de variables e inventario?
	¿Comprendió el sistema para generar condiciones?
	¿Comprendió que es posible colocar acciones antes de la

	instrucción if?
	¿Presentó dificultades con la instrucción if? (Sí/No)
	¿Comprendió que los robots pueden moverse simultáneamente?
	¿Comprendió el sistema de movimiento para ambos robots?
	Tiempo

Anexo 2

Test de jugador

Estas son las preguntas que se responden al finalizar el prototipo.

Experiencia de previa	
	¿En qué nivel de conocimientos de programación te encuentras? (1 al 5)
	¿Eres un usuario frecuente de videojuegos? (Sí/No)
	¿Has jugado previamente videojuegos de automatización o gestión de recursos? (Sí/No)
Sistema de programación	
	¿Qué tan fácil te resultó programar las instrucciones? (1 al 5)
	¿Comprendiste el sistema de movimiento? (Sí/No)
	¿Comprendiste el sistema de rotación? (Sí/No)
	¿Interactuaste correctamente con los objetos del juego? (Sí/No)
	¿Qué tan claro te resultó el sistema de recolección de recompensas mediante la palanca? Antes de la explicación (1 al 5)
	¿Qué tan claro te resultó el sistema de recolección de recompensas mediante la palanca? Después de la explicación (1 al 5)
	¿Comprendiste el sistema de creación de objetos?
	¿Comprendiste el sistema de vida de los recursos? Antes de la explicación
	¿Comprendiste el sistema de vida de los recursos? Después de la explicación

	¿Comprendiste el sistema de variables e inventario? Antes de la explicación
	¿Comprendiste el sistema de variables e inventario? Después de la explicación
	¿Comprendiste el sistema para generar condiciones?
	¿Tuviste dificultades con las instrucciones if en la programación? (Sí/No)
	¿Qué tan claro fue el proceso de configurar condicionales y bucles? (1 al 5)
Retroalimentación visual y efectos	
	¿Las animaciones y efectos visuales te ayudaron a comprender lo que ocurría en el juego? (Sí/No)
	¿Cómo calificarías la calidad visual de las animaciones? (1 al 5)
	¿Percibiste el progreso de la construcción de la casa a medida que completabas los objetivos? (Sí/No)
	¿Tuviste dificultades al colocar las acciones? (1 al 5, donde 5 indica mucha confusión)
	¿Consideras que los sprites de los objetos son representativos? En caso contrario, ¿cuáles no?
	¿Los objetos del menú de crafteo eran comprensibles? En caso contrario, ¿hubo alguno que no se entendiera?
Dificultad percibida	
	¿Consideraste adecuada la dificultad de las tareas según tu nivel de conocimientos? (Sí/No)
	¿Cómo calificarías la progresión de dificultad en el juego? (1 al 5)
	¿Cómo calificarías la dificultad de las instrucciones if? (1 al 5)
Experiencia de juego	
	¿Te sentiste motivado a completar todas las tareas del juego? (Sí/No)
	¿Qué aspectos mejorarías o cambiarías en el juego? (Respuesta abierta)
	¿Cómo calificarías tu opinión general sobre el juego? (1 al 5)
	¿Sientes que el juego te ayudó a mejorar tu conocimiento sobre

	programación? (Sí/No)
	En caso afirmativo, en una escala de 1 al 5.