UNIVERSIDAD SANTO TOMAS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
T U N J A

TÍTULO PROYECTO

Desarrollo de un laboratorio virtual interactivo para la enseñanza de arreglos y funciones en programación dirigido a los primeros semestres de Ingeniería de Sistemas.

PROPONENTE(S)

Edgar Santiago Jiménez Ortega

C.C. 1049641149

Cod.

DIRECTORA

Ing. Luz Santamaría Granados, PhD.

CO-DIRECTOR

Ing. Juan Francisco Mendoza Moreno, PhD.

Tunja

Fecha de presentación (18 de noviembre de 2024)

CONTENIDO

1. FICHA TÉCNICA DEL PROYECTO

Título	Desarrollo de un laboratorio virtual interactivo para la enseñanza de arreglos y funciones en programación dirigido a los primeros semestres de Ingeniería de Sistemas.
Autor	Edgar Santiago Jiménez Ortega
Director del proyecto	Luz Santamaría Granados
Palabras claves	Laboratorio virtual, arreglos y funciones.
Descripción	
Con una interfaz intuitiva y ejercicios prácticos, se buscaba proporcionar a los estudiantes un entorno de aprendizaje interactivo que les permitiera explorar arreglos y funciones, fortaleciendo así sus habilidades de resolución de problemas y comprensión conceptual. El proyecto incluyó una evaluación del impacto del laboratorio virtual en el rendimiento académico y la satisfacción estudiantil, para validar su efectividad como herramienta de enseñanza en el programa de Ingeniería de Sistemas.	

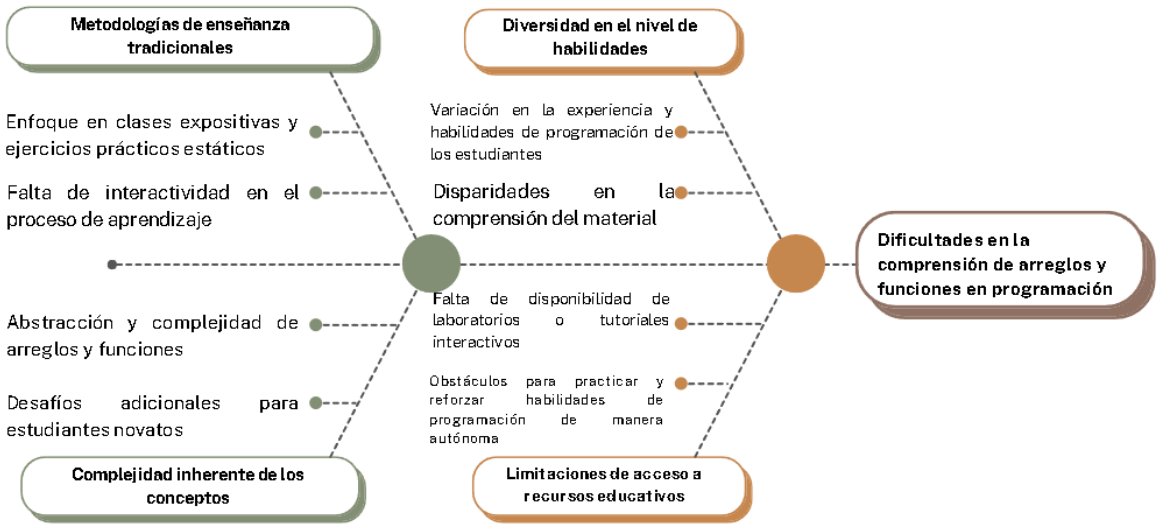
2. Planteamiento Del Problema

En el contexto de la educación en Ingeniería de Sistemas, se ha identificado una problemática persistente relacionada con la comprensión y aplicación de arreglos y funciones en programación, especialmente entre los estudiantes en los primeros semestres· Esta dificultad afecta no solo el rendimiento académico de los estudiantes, sino también su motivación y confianza en su capacidad para dominar los conceptos fundamentales de la programación·

2.1. Análisis de Causa-Efecto.

A continuación, se detallará el análisis de causa-efecto de la problemática que motivó el presente proyecto, y además se representa en la Figura 1 (ver abajo).

Figura 1. Árbol de problemas



Al profundizar en la naturaleza y magnitud del problema, se observa una serie de factores interrelacionados que contribuyen a esta dificultad:

- Metodologías de enseñanza tradicionales: Las metodologías de enseñanza tradicionales se centran en clases expositivas y ejercicios prácticos estáticos, lo que limita las oportunidades de los estudiantes para explorar y experimentar activamente con los conceptos de arreglos y funciones en un entorno interactivo y dinámico·

- Complejidad inherente de los conceptos: Los arreglos y funciones son conceptos fundamentales en programación, pero su abstracción y complejidad pueden resultar desafiantes para los estudiantes novatos· La falta de ejemplos claros y aplicaciones prácticas puede dificultar aún más su comprensión·

- Diversidad en el nivel de habilidades: Los estudiantes ingresan a los programas de Ingeniería de Sistemas con diferentes niveles de experiencia y habilidades en programación· Esta diversidad puede generar disparidades en la comprensión del material, lo que dificulta para los profesores satisfacer las necesidades individuales de cada estudiante en un entorno de aula tradicional·

- Limitaciones de acceso a recursos educativos: Fuera del aula, los estudiantes enfrentan limitaciones en cuanto al acceso a recursos educativos que les permitan practicar y reforzar sus habilidades de programación de manera autónoma· La falta de disponibilidad de laboratorios o tutoriales interactivos puede obstaculizar

su capacidad para explorar y experimentar con conceptos de programación en su propio tiempo y ritmo·

2.2. Estado del Arte.

Tecnologías Emergentes en la Enseñanza de la Programación

En la educación en programación, herramientas como laboratorios virtuales interactivos y plataformas en línea han transformado los métodos tradicionales al ofrecer experiencias dinámicas y personalizadas. Estas soluciones, documentadas por Anderson y Krathwohl (2001) y Guzdial y Ericson (2015), abordan desafíos como la falta de práctica activa y la abstracción de conceptos complejos.

Un ejemplo destacado es Scratch, que introduce a los estudiantes a la programación mediante un enfoque visual y creativo. Este tipo de plataforma no solo simplifica conceptos básicos, sino que también fomenta la colaboración y la experimentación, elementos esenciales en el aprendizaje activo (Kafai & Resnick, 2015).

Relevancia de las Plataformas WebRTC

WebRTC (Web Real-Time Communication) se ha consolidado como una herramienta clave para la transmisión de datos en tiempo real, permitiendo interacciones directas entre dispositivos a través de navegadores. Su implementación en la educación técnica es especialmente relevante, ya que posibilita la ejecución de simulaciones avanzadas en equipos con recursos limitados (Nguyen & Kim, 2016).

Ventajas de WebRTC en entornos educativos

1. Baja latencia: Ideal para laboratorios interactivos en tiempo real, donde el retraso puede afectar la experiencia del usuario.

2. Accesibilidad multiplataforma: Compatible con navegadores modernos sin necesidad de instalaciones adicionales, lo que facilita su adopción en diferentes entornos educativos.
3. Escalabilidad: Tecnologías como Pixel Streaming permiten transmitir entornos desarrollados en Unreal Engine desde un servidor, garantizando una experiencia fluida incluso en dispositivos de gama baja (García & López, 2019).

Ejemplos de uso:

- Labster: Una plataforma que emplea WebRTC para transmitir simulaciones de laboratorio a estudiantes de ciencias. Los usuarios pueden interactuar con experimentos virtuales de alta complejidad, reduciendo costos asociados a laboratorios físicos.
- CodeCollab: Herramienta que utiliza WebRTC para sesiones colaborativas de programación en tiempo real, donde los estudiantes pueden trabajar en equipo desde diferentes ubicaciones.

Estas aplicaciones demuestran el potencial de WebRTC para democratizar el acceso a recursos educativos avanzados, especialmente en instituciones con limitaciones tecnológicas.

Motores de Desarrollo de Videojuegos: Comparativa

Los motores de desarrollo como Unreal Engine, Unity y Godot se han convertido en herramientas fundamentales para la creación de entornos educativos interactivos. Cada uno presenta características que los hacen idóneos para diferentes contextos (Smith, 2020).

1. Unreal Engine
    - Ventajas: Ofrece gráficos de alta fidelidad y herramientas avanzadas como el sistema de iluminación Lumen y la generación de texturas realistas con

Nanite. Su capacidad para manejar simulaciones complejas lo hace ideal para proyectos que requieren inmersión visual, como laboratorios interactivos en 3D.

- Limitaciones: Requiere un hardware potente, lo que puede restringir su implementación en entornos educativos con recursos limitados.

2. Unity
    - Ventajas: Popular por su versatilidad, Unity es altamente personalizable y tiene una curva de aprendizaje accesible. Es ideal para proyectos multiplataforma y simulaciones ligeras, siendo una opción más asequible en términos de recursos.
    - Limitaciones: Aunque es potente, sus capacidades gráficas pueden ser inferiores a las de Unreal Engine en simulaciones de alta fidelidad.
3. Godot
    - Ventajas: Como motor de código abierto, Godot ofrece flexibilidad y eficiencia en proyectos pequeños o de presupuesto limitado. Es una excelente opción para aplicaciones 2D o entornos interactivos simples.
    - Limitaciones: No está diseñado para simulaciones gráficas intensivas, lo que limita su aplicación en laboratorios 3D complejos.

Para este proyecto, Unreal Engine ha sido seleccionado debido a su capacidad para integrar simulaciones interactivas avanzadas y su compatibilidad con técnicas de optimización como Level Streaming y Occlusion Culling, garantizando un rendimiento óptimo (Brown & Johnson, 2020).

Técnicas de Optimización para Mejorar el Desempeño

La optimización es un aspecto crítico en el desarrollo de laboratorios virtuales, especialmente cuando se busca hacerlos accesibles a estudiantes con dispositivos de baja gama. Entre las técnicas más relevantes se incluyen:

1. Mipmaps y LOD (Levels of Detail)
    - Permiten ajustar la resolución de texturas y modelos 3D según la distancia del usuario, reduciendo la carga gráfica y mejorando el rendimiento (Nguyen & Lee, 2021).
2. Occlusion Culling
    - Procesa únicamente los objetos visibles para el usuario, omitiendo aquellos ocultos por otros elementos. Esto reduce significativamente el tiempo de renderizado.
3. Level Streaming
    - Divide el entorno en secciones que se cargan dinámicamente según las necesidades del usuario. En proyectos educativos, esta técnica permite transiciones fluidas entre diferentes áreas del laboratorio virtual sin interrumpir la experiencia del estudiante.

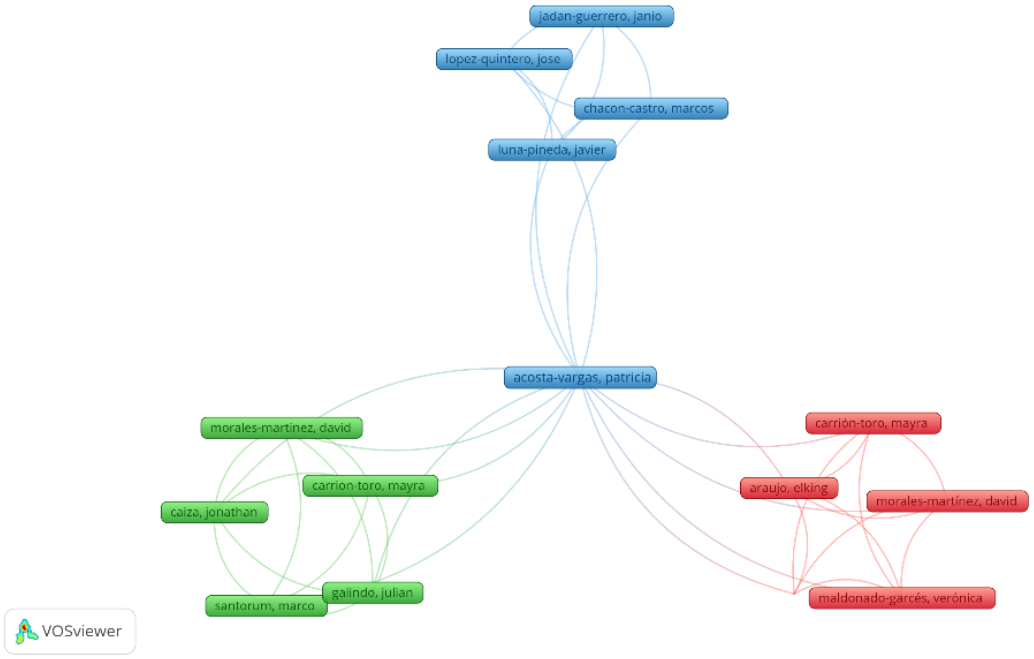
Análisis Bibliométrico:

En esta sección se presentan los análisis bibliométricos realizados con el software VOSviewer, utilizando datos extraídos de la base de datos Scopus mediante la búsqueda de los términos "serious video games in engineering education". Este análisis permitió identificar las redes de colaboración entre autores y las palabras clave más relevantes en el campo de estudio.

La **Figura 2** muestra la red de colaboración entre autores, destacando los principales investigadores y equipos que han trabajado conjuntamente en publicaciones relacionadas con el uso de videojuegos serios en la educación en ingeniería. Los nodos representan a los autores, y las conexiones indican coautorías en las publicaciones. Los clústeres de colores reflejan grupos de colaboración, permitiendo identificar las comunidades de investigación predominantes. Por ejemplo, el análisis muestra que el equipo liderado por Acosta-Vargas Patricia tiene una destacada presencia en este campo.

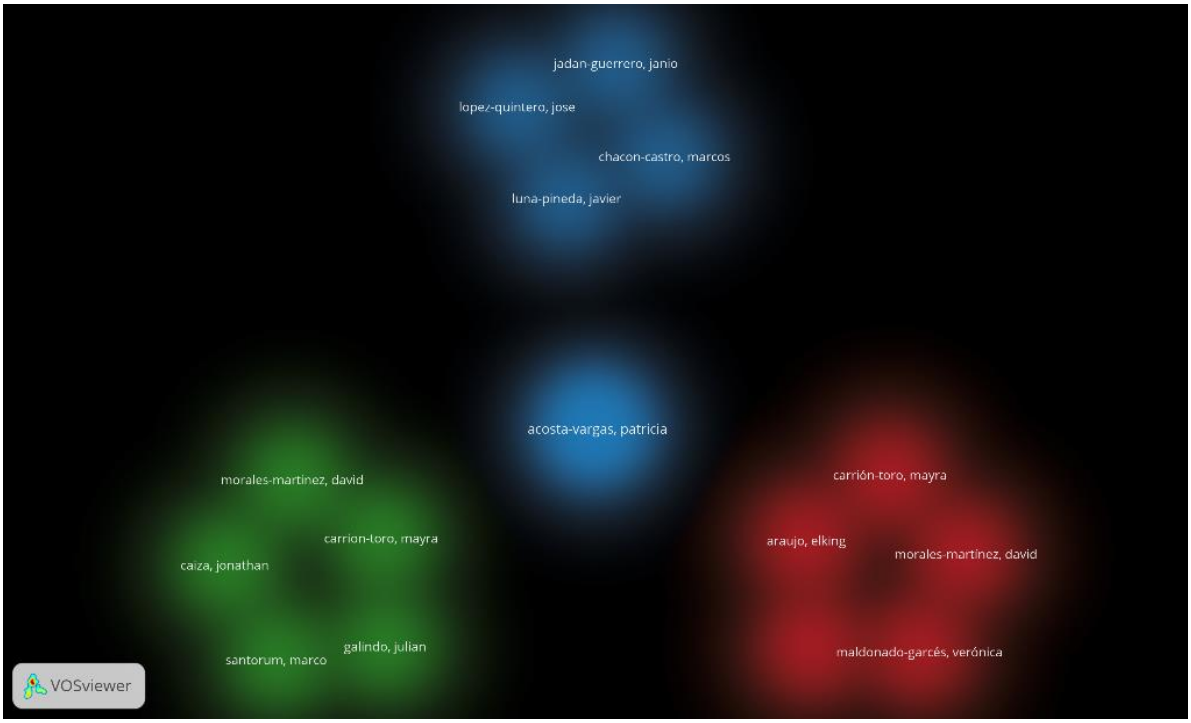
- Redes de colaboración entre autores

Figura 2. Red de colaboración entre autores en el tema de *"serious video games in engineering education"*. Elaboración propia basada en datos de Scopus con VOSviewer.



Por su parte, la **Figura 3** presenta la densidad de la red de colaboración. Las áreas con mayor densidad, representadas por colores cálidos, señalan regiones donde existe una intensa cooperación académica. Esto evidencia que ciertas instituciones o países tienen un mayor grado de interacción en este ámbito.

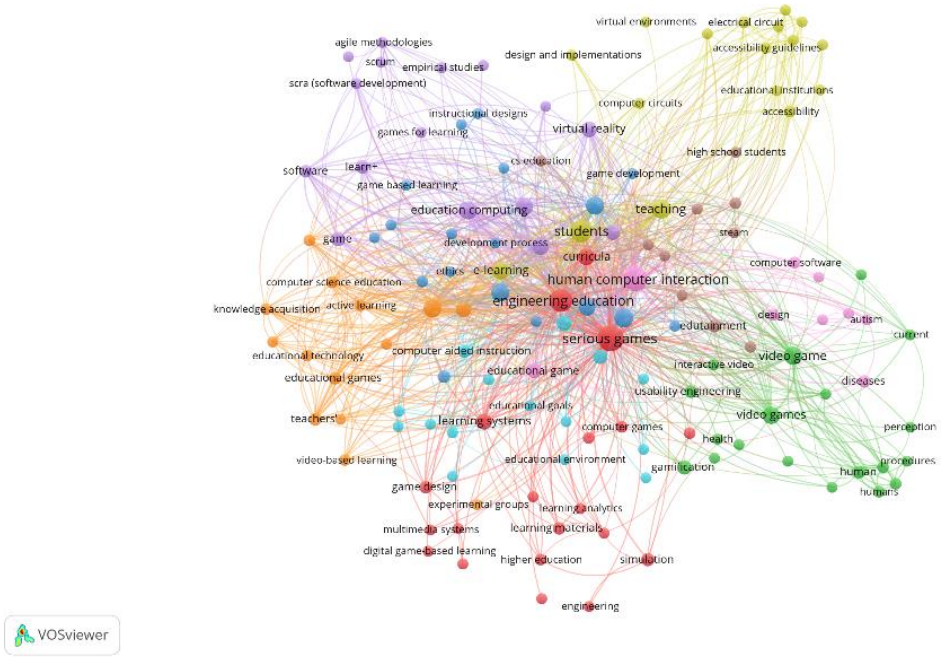
Figura 3. Densidad de la red de colaboración entre autores en el tema de *"serious video games in engineering education"*. Elaboración propia basada en datos de Scopus con VOSviewer.



- Análisis de palabras clave

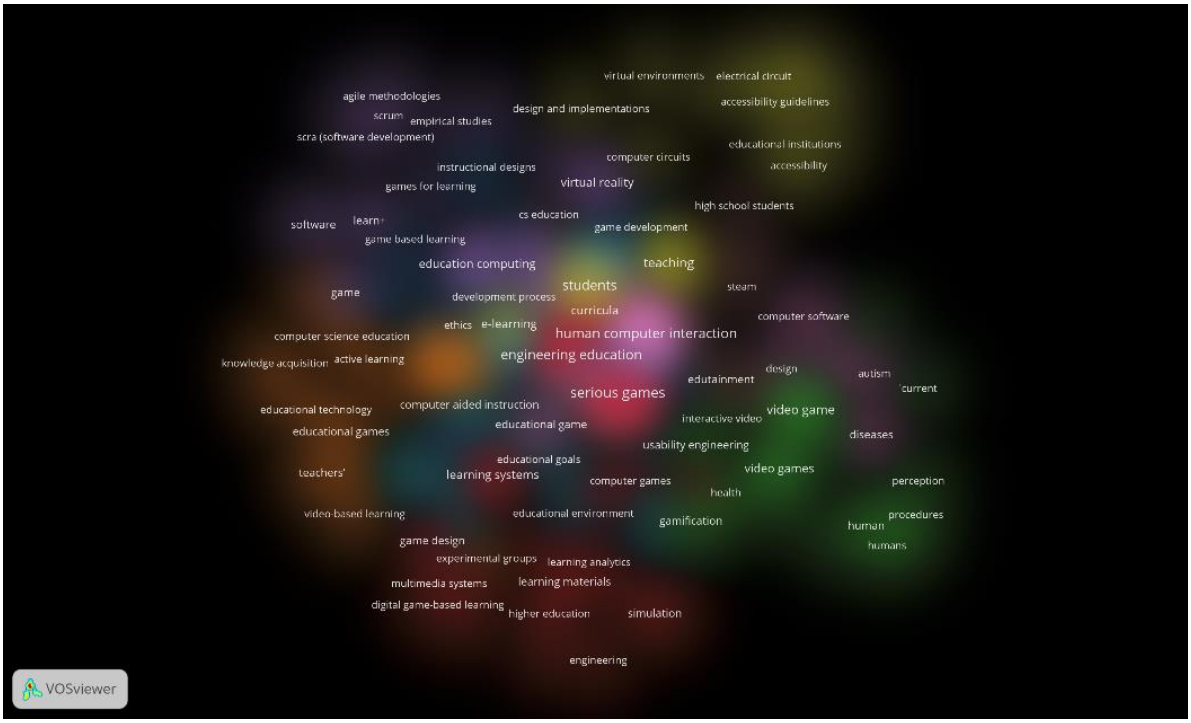
El análisis de palabras clave, representado en la **Figura 4**, revela las conexiones entre los conceptos más importantes del tema. Entre las palabras clave más frecuentes destacan *"educational games"*, *"serious games"*, y *"engineering education"*. Estas palabras clave se agrupan en torno a núcleos temáticos, mostrando las interrelaciones entre ellas y las tendencias de investigación predominantes.

Figura 4. Red de palabras clave relacionadas con *"serious video games in engineering education"*. Elaboración propia basada en datos de Scopus con VOSviewer.



Complementando esto, la **Figura 5** ilustra la densidad de las palabras clave, identificando áreas temáticas con mayor relevancia e investigación acumulada. Las regiones de mayor densidad sugieren que los aspectos relacionados con la implementación pedagógica y las evaluaciones de impacto son temas centrales en la investigación actual.

Figura 5. Densidad de palabras clave relacionadas con *"serious video games in engineering education"*. Elaboración propia basada en datos de Scopus con VOSviewer.



En conjunto, este análisis bibliométrico no solo destaca los patrones de colaboración y los temas clave, sino que también permite identificar posibles áreas para futuras investigaciones y colaboraciones interdisciplinarias.

3. Justificación

La creación del laboratorio virtual interactivo para la enseñanza de arreglos y funciones en programación responde a una problemática recurrente en los estudiantes de Ingeniería de Sistemas: la dificultad para comprender y aplicar conceptos fundamentales como los arreglos y las funciones en los primeros semestres de su formación. Estos conceptos no solo son pilares en el desarrollo de software, sino que también son fundamentales para adquirir competencias avanzadas en disciplinas relacionadas, como la inteligencia artificial, la ciencia de datos y el diseño de sistemas complejos.

La necesidad de abordar estas limitaciones es especialmente crítica dado el creciente protagonismo de las habilidades en programación en el ámbito profesional. En un contexto donde la automatización, la transformación digital y la innovación tecnológica están redefiniendo el mercado laboral, es indispensable que los estudiantes desarrollen una base sólida en lógica de programación y pensamiento algorítmico desde etapas tempranas de su formación académica. Estas habilidades no solo permiten resolver problemas de manera estructurada y eficiente, sino que también fomentan el pensamiento crítico, la creatividad y la capacidad de adaptarse a tecnologías emergentes.

Beneficios del desarrollo tecnológico

El laboratorio virtual interactivo propone ir más allá de los enfoques tradicionales de enseñanza al integrar herramientas tecnológicas que promueven un aprendizaje más dinámico, inmersivo y significativo. Este enfoque aporta los siguientes beneficios:

1. Fomento del pensamiento algorítmico: El diseño de actividades y simulaciones interactivas permitirá que los estudiantes experimenten con arreglos y funciones

de manera práctica, desarrollando su capacidad para descomponer problemas complejos en pasos más simples y estructurados.

2. Desarrollo de competencias de programación: El laboratorio proporcionará un entorno controlado donde los estudiantes puedan validar y depurar automáticamente su código, recibiendo retroalimentación inmediata que acelera su aprendizaje y les permite identificar errores comunes desde una etapa temprana.
3. Colaboración y aprendizaje social: Las funcionalidades colaborativas, como el trabajo en proyectos grupales o la participación en foros integrados, fomentarán el intercambio de ideas, la resolución conjunta de problemas y la creación de una comunidad de aprendizaje.
4. Seguridad y accesibilidad: La incorporación de tecnologías como WebRTC permitirá implementar funcionalidades avanzadas de protección de datos y control de identidad, garantizando que el laboratorio sea seguro y accesible para todos los estudiantes, independientemente de su nivel de habilidad o ubicación geográfica.

Desafíos tecnológicos

La implementación del laboratorio virtual interactivo presenta desafíos tecnológicos que requieren un enfoque riguroso y creativo. Entre estos se encuentran:

- Optimización del rendimiento: Diseñar una plataforma que pueda manejar múltiples usuarios simultáneamente sin comprometer la velocidad ni la experiencia del usuario.
- Interfaces intuitivas: Crear una interfaz gráfica de usuario amigable y accesible que facilite la navegación y el aprendizaje, incluso para aquellos con poca experiencia en programación.

- Validación automática de código: Desarrollar sistemas que analicen el código de los estudiantes en tiempo real, proporcionando retroalimentación detallada sobre errores de sintaxis, lógica y estilo de programación.

Trabajo futuro

El desarrollo de este proyecto no se limita a su implementación inicial, sino que busca establecer una base para futuras expansiones y mejoras. Algunas líneas de trabajo previstas incluyen:

- Ampliación de contenidos: Extender el alcance del laboratorio para cubrir otros conceptos clave en programación, como estructuras de datos, algoritmos avanzados y patrones de diseño.
- Integración de tecnologías emergentes: Incorporar elementos de realidad virtual y aumentada para crear simulaciones más inmersivas, mejorando la comprensión de conceptos abstractos.
- Herramientas de evaluación avanzadas: Implementar sistemas basados en inteligencia artificial que no solo validen el código de los estudiantes, sino que también adapten las actividades a sus necesidades individuales y estilos de aprendizaje.
- Colaboración interdisciplinaria: Trabajar con expertos en educación, diseño de interfaces y seguridad informática para garantizar que el laboratorio sea eficiente, inclusivo y relevante en el contexto educativo actual.

Este enfoque iterativo y colaborativo permitirá que el laboratorio evolucione continuamente, adaptándose a las necesidades cambiantes de los estudiantes y asegurando su relevancia en el dinámico ámbito de la educación en programación.

4. Objetivos

1. Objetivo General

Facilitar la comprensión y aplicación de arreglos y funciones en programación, mejorando el proceso de aprendizaje de estudiantes de Ingeniería de Sistemas en los primeros semestres, mediante el desarrollo e implementación de un laboratorio virtual interactivo, accesible desde cualquier lugar y en cualquier momento.

2. Objetivos específicos

A continuación, se presentan los objetivos específicos.

Tabla 2. Objetivos específicos

Nro.	Objetivo específico
1	Diseñar una interfaz intuitiva para el laboratorio virtual, permitiendo a los estudiantes de primer semestre de Ingeniería de Sistemas el acceso a los recursos educativos simulados sobre arreglos y funciones de programación de manera autónoma.
2	Desarrollar ejercicios prácticos y casos de estudio que aborden diferentes aspectos y aplicaciones de los arreglos y funciones en programación, para el fortalecimiento de las habilidades de resolución de problemas y la comprensión conceptual de los estudiantes, y su implementación dentro del laboratorio virtual.

3	Evaluar el impacto del laboratorio virtual en el rendimiento académico y la satisfacción estudiantil, mediante la recopilación de datos cuantitativos y cualitativos, en colaboración con docentes y estudiantes del programa de Ingeniería de Sistemas.

Fuente: Autor

5. Modelo De Desarrollo

Descripción del Ciclo de Vida Utilizado

El ciclo de vida del desarrollo del laboratorio virtual interactivo se fundamentó en el Modelo Iterativo e Incremental, un enfoque que permitió la evolución continua del sistema mediante iteraciones bien definidas. Este modelo proporcionó flexibilidad para incorporar retroalimentación constante de estudiantes y docentes, asegurando que los requerimientos educativos y tecnológicos se integraran de manera eficiente. En cada iteración, se entregaron versiones funcionales que fueron evaluadas y ajustadas, logrando mejoras progresivas y adaptaciones basadas en las necesidades de los usuarios finales.

En la Figura 6 se presenta una representación gráfica del modelo Iterativo e Incremental, que ilustra las fases principales del ciclo de desarrollo: Definición de Requerimientos, Diseño del Sistema, Implementación, Pruebas y Validación, e Iteración y Mejora. Este esquema refleja el carácter cíclico del modelo, destacando cómo cada fase alimenta la siguiente en un proceso continuo de refinamiento.

Figura 6. Ciclo de vida

Fases del Modelo

A continuación, se detalla cada fase metodológica, estructurando el desarrollo conforme a los principios de la Ingeniería de Software:

1. Ingeniería de Requisitos o Requerimientos

En esta etapa inicial, se realizó un análisis exhaustivo en colaboración con docentes y estudiantes del programa de Ingeniería de Sistemas. Las actividades incluyeron:

- Recolección de requerimientos funcionales y no funcionales.
- Identificación de los objetivos de aprendizaje en torno a arreglos y funciones.

- Evaluación de las tecnologías y recursos necesarios para el desarrollo.

2. Diseño Arquitectónico del Laboratorio Virtual

Esta fase consistió en el diseño conceptual y técnico de la plataforma, incluyendo:

- Creación de diagramas de flujo y modelos de datos.
- Definición de la arquitectura modular del laboratorio, asegurando escalabilidad y mantenimiento.
- Diseño de prototipos de la interfaz de usuario con herramientas de UX/UI, que facilitaron el desarrollo de un entorno intuitivo y atractivo para los estudiantes.

3. Desarrollo Tecnológico

El desarrollo se realizó utilizando Unreal Engine 5, empleando buenas prácticas de desarrollo para garantizar calidad y organización:

- Código fuente estructurado y documentado: El código fue versionado en un sistema de control de versiones (GitHub), con documentación clara para cada módulo.
- Revisión de código: Cada iteración incluyó revisiones para detectar y corregir posibles errores en la lógica y estructura.
- Pruebas unitarias y desarrollo guiado por pruebas (TDD): Estas técnicas aseguraron que cada módulo cumpliera con los requerimientos antes de ser integrado.

4. Implementación del Proyecto de Software

El laboratorio fue implementado en un entorno controlado de pruebas, lo que permitió:

- Realizar simulaciones de uso para verificar el correcto funcionamiento de los módulos interactivos.
- Probar la compatibilidad en diferentes dispositivos y sistemas operativos.
- Identificar y solucionar problemas de rendimiento o usabilidad.

5. Validación del Sistema

La validación se llevó a cabo mediante pruebas de calidad en colaboración con estudiantes y docentes, con énfasis en:

- Pruebas de usabilidad: Evaluar la facilidad de uso y navegación del laboratorio.
- Pruebas funcionales: Comprobar que cada funcionalidad cumpliera los requerimientos establecidos.
- Análisis de impacto: Medir el efecto del laboratorio en la comprensión de conceptos clave de programación, recopilando datos cuantitativos y cualitativos.

6. Iteración y Mejora

Los resultados de las pruebas condujeron a iteraciones en las que se:

- Ajustaron las funcionalidades y la interfaz de usuario para mejorar la experiencia de aprendizaje.
- Implementaron optimizaciones en el rendimiento del sistema.
- Refinaron los ejercicios prácticos para alinearse con los objetivos educativos.

7. Entrega Final

El proyecto culminó con la entrega de la versión final del laboratorio, acompañada de:

- Documentación técnica detallada, incluyendo manuales de usuario.
- Guías educativas para docentes, con sugerencias de uso en el aula.
- Implementación de un sistema de soporte técnico para resolver posibles dudas o problemas.
- Tabla de las Fases del Modelo

La siguiente tabla sintetiza las fases clave del ciclo de vida del proyecto:

Tabla 3. Fases del Modelo Iterativo e Incremental

Fase	Descripción
Definición de Requerimientos	**Análisis detallado de las necesidades educativas y tecnológicas en colaboración con docentes y estudiantes.**
Diseño del Sistema	**Diseño de la arquitectura, interfaz de usuario y funcionalidades clave.**
Implementación	**Desarrollo de módulos interactivos utilizando Unreal Engine 5.**
Pruebas y Validación	**Evaluación de usabilidad, funcionalidad e impacto en estudiantes reales.**
Iteración y Mejora	**Ajustes basados en retroalimentación, optimización de rendimiento y refinamiento de funcionalidades.**
Entrega Final	**Versión final del laboratorio con documentación técnica y educativa.**

6. Desarrollo Del Proyecto

En esta sección, se detalla cada una de las etapas realizadas para completar el proyecto del laboratorio virtual interactivo. Para cada fase, se presenta la descripción de las actividades realizadas, los resultados obtenidos y la evidencia correspondiente.

Fase 1: Definición de Requisitos

La primera fase del desarrollo del laboratorio virtual interactivo se centra en identificar, analizar y documentar los requisitos técnicos, funcionales y no funcionales. Esta etapa asegura que el proyecto se oriente hacia las necesidades específicas de los estudiantes y profesores, y establece la base para las siguientes fases de desarrollo.

Objetivo

Definir los requisitos funcionales y no funcionales necesarios para el diseño, implementación y optimización del laboratorio, garantizando que sea didáctico, accesible y eficiente.

Metodología Utilizada

1. Análisis de Necesidades:
    - Se realizaron entrevistas con profesores de programación para identificar las áreas más complejas relacionadas con arreglos y funciones.

- Se distribuyeron encuestas entre estudiantes de primeros semestres para evaluar su comprensión de los temas y las herramientas educativas existentes.

2. Investigación Técnica:
    - Con base en el Anexo 3: Investigación de Optimización, se analizaron los motores gráficos más adecuados y las técnicas de optimización necesarias para garantizar el rendimiento en hardware básico.
3. Definición de Requisitos:
    - Se crearon listas detalladas de características funcionales que debe incluir el laboratorio, así como métricas no funcionales relacionadas con rendimiento, accesibilidad y tiempo de respuesta.

Resultados

A continuación, se presenta la tabla que detalla los requisitos identificados:

ID	Requisito	Tipo	Descripción	Prioridad	Notas
R-01	Visualización de arreglos	Funcional	El laboratorio debe permitir visualizar arreglos unidimensionales y multidimensionales en tiempo real.	Alta	Incluye ejemplos interactivos de arreglos.
R-02	Modificación de arreglos	Funcional	Los usuarios deben poder modificar valores dentro de los arreglos en el entorno virtual.	Alta	Compatible con teclado y ratón.
R-03	Ejercicios prácticos	Funcional	Debe incluir ejercicios guiados que ayuden a comprender el uso de funciones y arreglos.	Alta	Deben estar organizados por nivel.
R-04	Soporte para hardware básico	No funcional	El laboratorio debe ser accesible en dispositivos con al menos 4 GB de RAM y gráficos integrados.	Media	Basado en requisitos mínimos de optimización.
R-05	Tiempo de respuesta	No funcional	La interacción no debe superar los 2 segundos de latencia.	Alta	Aplicar técnicas de optimización gráfica.

Tabla 3. Tabla de Requisitos

Informe de Viabilidad Técnica

El análisis técnico basado en el **Anexo 3** concluye que el motor Unreal Engine 5 es adecuado para el desarrollo de este proyecto. Se eligió debido a:

- Su capacidad para manejar interacciones en tiempo real.
- Técnicas avanzadas de optimización como:
    - *Level of Detail (LOD)*: Permite renderizar modelos con menor detalle cuando están lejos del punto de vista del usuario.
    - *Dynamic Resolution Scaling*: Adapta la resolución según la capacidad del hardware para mantener un rendimiento constante.

Estas características aseguran que el laboratorio funcione correctamente en equipos con especificaciones mínimas, cumpliendo con los requisitos no funcionales de accesibilidad.

Documentación Vinculada

- Anexo 3: Investigación de Optimización:
    - Proporciona las bases técnicas para los requisitos no funcionales relacionados con hardware y rendimiento.
    - Especifica las técnicas de optimización a emplear durante el desarrollo.
- Anexo 1: Ficha Técnica:
    - Resume las especificaciones iniciales del laboratorio, incluyendo sus características técnicas básicas y el motor de desarrollo seleccionado.

Fase 2. Diseño arquitectónico

Objetivo de la Fase

Establecer un diseño arquitectónico detallado que contemple cómo se organiza el sistema de información del laboratorio virtual. Incluir diagramas de clases, diagramas de flujo y modelos de datos para representar la estructura y la interrelación entre los módulos.

Actividades y Subtareas

1. Diseño de la Arquitectura:
    - Elaborar un diagrama general de arquitectura modular del sistema.
    - Establecer relaciones entre el motor de desarrollo, los módulos de interacción del usuario, y los módulos de evaluación.
2. Diagramas de Clases:
    - Identificar las principales clases del sistema, como Usuario, Laboratorio, Ejercicio, Arreglo, Evaluación.
    - Establecer relaciones entre clases y métodos clave.
3. Modelos de Datos:
    - Crear diagramas que definan la estructura de las bases de datos utilizadas para almacenar datos del usuario, progreso y configuraciones.
    - Definir entidades, atributos y relaciones para un esquema ER (Entidad-Relación).
4. Diagramas de Flujo:
    - Representar la interacción del usuario con el sistema, desde el inicio de sesión hasta la evaluación.
    - Incluir flujos de navegación por los diferentes módulos.

5. Revisión y Validación de Diseño:
    - Revisión de los diagramas con stakeholders para asegurar alineación con los requisitos.

Entregables de la Fase

1. Diagrama de Arquitectura General:
    - Representación gráfica de la organización del sistema.
    - Ejemplo: Motor de juego → Módulo Interactivo → Módulo de Evaluación → Base de Datos.
2. Diagrama de Clases:
    - Incluir las relaciones y jerarquías de clases principales.
3. Modelo de Datos:
    - Diagrama ER que muestre las entidades Usuario, Progreso, Configuración, entre otras.
4. Diagramas de Flujo:
    - Flujos para acciones clave del usuario, como realizar un ejercicio o consultar resultados.

Elemento	Descripción	Documento Relacionado	Estado
Arquitectura General	Estructura modular del sistema con componentes para interacción, evaluación y persistencia de datos.	Anexo 4: Diseño Técnico	Finalizado
Diagrama de Clases	Mapa jerárquico y relacional de las clases principales del laboratorio virtual.	Anexo 4: Diseño Técnico	Finalizado
Modelo de Datos	Diagrama ER que establece las relaciones entre las entidades principales del sistema.	Anexo 4: Diseño Técnico	Finalizado
Diagramas de Flujo	Flujos de interacción del usuario y ejecución interna del sistema.	Anexo 4 y 5: Diseño Técnico/Juego	Finalizado

Tabla 4. Tabla de Diseño del Sistema

Fase 3. Desarrollo de la aplicación

En esta fase se realiza el desarrollo efectivo del laboratorio virtual interactivo, estructurado en torno a la utilización de C++ y Blueprints de Unreal Engine. El sistema debe estar alineado con los requisitos definidos, seguir buenas prácticas de desarrollo y asegurarse de que todas las funcionalidades están implementadas de manera eficiente y coherente.

Estructuración del Código

El código se debe dividir en módulos coherentes que reflejen las diferentes funcionalidades del laboratorio virtual, facilitando la reutilización y mantenimiento. Dado que el proyecto utiliza Unreal Engine, es fundamental que el código en C++ se comunique correctamente con los Blueprints, permitiendo que la experiencia de usuario se desarrolle de manera fluida. Los módulos pueden incluir:

1. Módulo de Interacción del Jugador: Donde se definen las interacciones del jugador con el entorno, como hacer clic para iniciar el juego o interactuar con objetos del escenario.
2. Módulo de Lógica de Ejercicios: Donde se implementan las funcionalidades relacionadas con los ejercicios que forman parte de la experiencia.
3. Módulo de Navegación y Avance en el Juego: Que controla la secuencia lineal de la experiencia, asegurando que el usuario progrese sin opciones alternativas.

Control de Versiones

El control de versiones será gestionado a través de Git.

Buenas Prácticas de Desarrollo

1. Desarrollo Guiado por Pruebas (TDD): Aunque no se ha utilizado una plataforma de integración continua, el uso de pruebas unitarias es esencial. En este caso, las pruebas se pueden realizar manualmente dentro de Unreal Engine para verificar que las funcionalidades, como la interacción con objetos, el avance entre pantallas y la respuesta del jugador, estén funcionando correctamente.
2. Revisión de Código: El código debe ser revisado entre los desarrolladores para asegurar que se mantenga una alta calidad. Las revisiones deben enfocarse en la coherencia con los estándares del proyecto, la eficiencia del código y la integración correcta de Blueprints con C++.

Documentación Técnica

La documentación del código debe ser exhaustiva y detallada, especialmente debido a la utilización de dos lenguajes (C++ y Blueprints). Se debe asegurar que cada clase, función y variable esté debidamente documentada para que otros desarrolladores puedan comprender rápidamente el propósito de cada bloque de código. La documentación debe incluir:

- Descripción de cada módulo y su relación con el sistema en su conjunto.
- Guía para la integración de Blueprint y C++, detallando cómo las clases en C++ interactúan con los nodos de Blueprint, y cómo se gestionan las variables y funciones entre ambos lenguajes.
- Plan de pruebas: Descripción de las pruebas manuales que se deben realizar para verificar que cada módulo funciona correctamente dentro del sistema.

Integración de Componentes del Juego

Es esencial que los Blueprints y C++ trabajen en armonía para crear una experiencia fluida y consistente. El código en C++ puede ser utilizado para la parte lógica compleja y el rendimiento crítico del juego, mientras que los Blueprints permiten una creación rápida de las interacciones y eventos dentro del entorno gráfico de Unreal Engine.

- Eventos: El jugador interactuará con botones y objetos en el escenario, lo cual estará gestionado por Blueprints, mientras que las lógicas de avance entre los diferentes estados del juego (iniciar juego, cambiar pantalla, mostrar resultados) estarán controladas por C++.
- Sincronización: Es importante que el progreso del jugador entre pantallas o ejercicios esté correctamente gestionado por C++, mientras que las interacciones de usuario, como el clic en un botón o la aparición de información, estén definidas en los Blueprints.

Entregables Esperados

Al final de la fase de desarrollo, los entregables incluyen:

- El código fuente completo desarrollado en C++ y Blueprints.
- Pruebas unitarias realizadas en Unreal Engine.
- Documentación técnica detallada que cubra la estructura del sistema y las interacciones entre los módulos.

Tecnologías y Herramientas Utilizadas

Las herramientas y tecnologías utilizadas en esta fase son:

- Unreal Engine como el motor de desarrollo principal, utilizado tanto para la programación en C++ como para la creación de Blueprints.

- Visual Studio como el entorno de desarrollo integrado (IDE) para trabajar con C++ y la integración con Unreal Engine.
- Git para el control de versiones y la gestión del código fuente.
- Unreal Engine Blueprint para la creación de la lógica de interacción y eventos dentro del juego.

Fase 4. Desarrollo de la aplicación

En esta fase, el laboratorio virtual interactivo se despliega en un entorno de prueba para garantizar que todas las funcionalidades sean accesibles y usables. El objetivo es verificar que el software desarrollado cumple con los requisitos definidos en fases anteriores y que la experiencia del usuario es coherente y eficiente. Además, se debe asegurar un control de versiones adecuado, garantizando la trazabilidad y el buen mantenimiento del código fuente.

Despliegue en el Entorno de Prueba

El proceso de implementación comienza con la preparación del entorno de prueba, que puede ser un servidor local o una máquina de desarrollo configurada para simular el entorno en el que los usuarios finales interactuarán con el sistema. Este entorno debe ser similar al entorno de producción, pero limitado a pruebas y validación de las funcionalidades antes de que el sistema se despliegue a un público más amplio.

El despliegue de la aplicación incluye:

- Instalación de dependencias: Verificación de que todas las dependencias de software necesarias, como Unreal Engine y Visual Studio, estén correctamente instaladas y configuradas en el entorno de prueba.

- Compilación del proyecto: Generación de los archivos ejecutables y la configuración adecuada de los Blueprints y el código C++ que compone la aplicación.
- Configuración de red y acceso: Para verificar la accesibilidad del juego o experiencia virtual, se configura el sistema de modo que sea accesible en el entorno de prueba y el equipo de QA (Quality Assurance) pueda interactuar con él sin problemas.

Uso de Pixel Streaming de Unreal Engine: Una parte clave de esta fase fue la implementación del Pixel Streaming de Unreal Engine mediante el servicio WebRTC. Este servicio permite que el proyecto finalizado se ejecute en un servidor y se transmita de forma remota a los usuarios finales a través de un navegador web. El Pixel Streaming es particularmente útil para experiencias interactivas, como un laboratorio virtual, ya que permite ofrecer una experiencia de alta calidad sin requerir que los usuarios tengan equipos de alto rendimiento.

A través de este servicio, el proyecto se ejecuta en un servidor potente, mientras que los usuarios interactúan con él a través de sus dispositivos (como PCs, tabletas o teléfonos), solo necesitando un navegador web compatible. Este enfoque asegura que los usuarios puedan acceder a la experiencia sin preocuparse por las especificaciones de hardware de su propio dispositivo.

2. Validación de Funcionalidades y Usabilidad

Una vez desplegado el proyecto en el entorno de prueba, se realizan pruebas para verificar que las funcionalidades implementadas en fases anteriores estén accesibles y sean fáciles de usar para los usuarios finales.

1. Accesibilidad:

- Verificar que todas las pantallas y funciones principales sean accesibles desde el inicio del juego.
- Asegurarse de que no existan bloqueos o errores críticos que impidan que el usuario avance en la experiencia de manera lineal.

2. Usabilidad:
    - El equipo de pruebas o el mismo desarrollador debe realizar pruebas de usabilidad para evaluar que la interfaz de usuario (UI) es clara, los elementos interactivos responden de forma esperada y la experiencia del jugador sea intuitiva.
    - Las interacciones del jugador con el entorno deben estar completamente funcionales, como la capacidad de hacer clic en los botones de navegación, las interacciones con objetos o ejercicios dentro del juego.

Pruebas específicas que deben realizarse:

- Pruebas de interacción: Verificar que todos los botones y acciones dentro del juego estén operativos, desde el inicio del juego hasta la finalización de la experiencia.
- Pruebas de avance: Asegurarse de que el jugador avance de manera lineal a través de la experiencia, sin posibilidad de elegir entre diferentes ejercicios, garantizando que el flujo de la experiencia sea controlado.
- Pruebas de rendimiento: Verificar que el rendimiento del juego sea estable, sin caídas de frame rate ni errores visuales que interfieran con la experiencia del usuario, especialmente en el contexto de la transmisión a través de Pixel Streaming.

3.Control de Versiones

El control de versiones es un componente crítico para gestionar el código fuente a lo largo de todo el proyecto. Durante la fase de implementación, se debe asegurar que el código fuente esté correctamente versionado para:

- Evitar la pérdida de trabajo: Usar herramientas de control de versiones como Git para realizar "commits" frecuentes y mantener un registro claro de todos los cambios realizados en el proyecto.
- Trazabilidad de los cambios: Cada "commit" debe estar documentado con una descripción clara de lo que se ha modificado, añadido o corregido. Esto facilita la colaboración entre los miembros del equipo y el seguimiento del progreso.
- Ramas y fusiones: A lo largo de la fase de implementación, se debe mantener una estructura de ramas en Git. Por ejemplo, cada nueva funcionalidad debe ser desarrollada en una rama separada, y luego fusionada con la rama principal después de ser probada y validada.
- Revisión de código: Si se cuenta con un equipo de trabajo, se debe realizar una revisión de código para asegurar que las nuevas funcionalidades estén alineadas con los estándares y buenas prácticas de desarrollo definidas en fases anteriores.

Proceso de integración:

- Integración local: Durante la implementación, es posible que se realicen múltiples integraciones locales en el entorno de desarrollo antes de subir cambios al repositorio de Git. Esto asegura que el código se mantenga funcional y sin conflictos.
- Integración con Unreal Engine: El código debe ser integrado correctamente con Unreal Engine para garantizar que las interacciones y lógicas implementadas en C++ y Blueprints funcionen sin errores en el entorno de prueba.

4. Corrección de Errores

Después de realizar las pruebas en el entorno de prueba, se deben identificar y corregir los errores encontrados. Los errores deben ser clasificados según su severidad:

- Errores críticos: Aquellos que impiden el avance del juego, como problemas con la interfaz de usuario o la lógica del flujo.
- Errores menores: Aquellos que no afectan significativamente la experiencia, pero que deben corregirse para asegurar la calidad del producto final.

Fase 5. Validación del sistema

La fase de validación del sistema es crucial para asegurar que el laboratorio virtual interactivo funcione correctamente en todos los aspectos previstos. Esta fase incluye diversas pruebas diseñadas para verificar que el sistema no solo cumpla con los requisitos funcionales, sino también con los criterios de rendimiento y calidad necesarios para una experiencia satisfactoria del usuario. Se realiza un análisis exhaustivo a través de pruebas de integración, pruebas unitarias y pruebas de rendimiento, que estarán acompañadas de métricas específicas para evaluar el comportamiento y la calidad del sistema.

Pruebas de Integración

Las pruebas de integración están diseñadas para verificar la interacción entre los diferentes módulos del sistema, asegurando que cada parte del laboratorio virtual funcione correctamente cuando se combina con otros componentes del sistema.

Objetivos de las pruebas de integración:

- Verificar que los módulos desarrollados en C++ y Blueprints de Unreal Engine se comuniquen correctamente entre sí.

- Validar la correcta integración del sistema de interacción con el entorno y la visualización de los resultados del ejercicio en el sistema de backend.
- Asegurar que la integración del servicio de Pixel Streaming funcione adecuadamente, transmitiendo la experiencia interactiva desde el servidor al cliente sin interrupciones.

Metodología:

- Pruebas de comunicación entre módulos: Validar que las funciones de interacción (clic en botones, selección de objetos, navegación del usuario) estén correctamente conectadas con las lógicas del sistema (ej. avance de niveles, generación de resultados, etc.).
- Pruebas de comunicación entre el cliente y el servidor: Verificar que las interacciones del cliente (jugador) se reflejen correctamente en el servidor que ejecuta el juego, a través de Pixel Streaming.
- Verificación de datos: Confirmar que los datos enviados entre el sistema de frontend (interfaz) y el backend (lógica del servidor) sean consistentes y correctos.

Métricas clave:

- Tiempo de respuesta de la integración: Cuánto tarda el sistema en responder a una interacción del usuario (por ejemplo, hacer clic en un botón y recibir una respuesta).
- Tasa de éxito de las interacciones: Porcentaje de interacciones que se realizan correctamente sin errores ni fallos.

Pruebas de Rendimiento

Las pruebas de rendimiento son esenciales para asegurarse de que el laboratorio virtual pueda manejar la carga esperada de usuarios, sin comprometer la fluidez de la experiencia interactiva. Dado que el sistema utiliza Pixel Streaming para transmitir la

experiencia a través de la web, las pruebas de rendimiento también evaluarán el ancho de banda y la latencia en la comunicación cliente-servidor.

Objetivos de las pruebas de rendimiento:

- Validar que el sistema funcione sin caídas de rendimiento ni errores en el servidor cuando se somete a diferentes cargas de trabajo.
- Evaluar el ancho de banda necesario para transmitir la experiencia de juego a través de Pixel Streaming, asegurando que el sistema sea accesible incluso en redes con baja capacidad.
- Evaluar el comportamiento del sistema bajo condiciones de alto uso, asegurando que el servidor no se sature ni cause lentitud en la experiencia del usuario.

Metodología:

- Pruebas de carga: Simular varios usuarios accediendo simultáneamente al sistema a través de Pixel Streaming para medir la capacidad de respuesta y el impacto en el rendimiento del servidor.
- Pruebas de estrés: Ejecutar el sistema bajo condiciones extremas para evaluar su comportamiento y determinar sus límites.
- Pruebas de latencia: Medir la latencia en las interacciones del usuario (es decir, el tiempo entre que el usuario realiza una acción y ve la respuesta en su pantalla).

Métricas clave:

- Tiempo de respuesta (latencia): Cuánto tiempo tarda una acción del usuario en reflejarse en la pantalla.
- Uso de CPU y memoria del servidor: Monitorear la carga de trabajo del servidor mientras se ejecutan las pruebas.
- Ancho de banda: Medir la cantidad de datos transmitidos entre el servidor y el cliente durante las interacciones.

Pruebas Unitarias

Las pruebas unitarias se realizan para asegurar que las funciones individuales y los componentes del código fuente estén funcionando correctamente. En esta fase, las pruebas unitarias se centran en las partes fundamentales del código, como las funciones en C++ y la lógica de interacción en los Blueprints de Unreal Engine.

Objetivos de las pruebas unitarias:

- Verificar que las funciones del sistema (por ejemplo, cálculos, lógica de avance de niveles) trabajen de acuerdo con las especificaciones.
- Asegurar que los Blueprints implementados en Unreal Engine no contengan errores y se comporten como se espera.
- Validar que las interacciones que afectan el progreso del usuario, como los resultados de los ejercicios, se gestionen correctamente a nivel de código.

Metodología:

- Pruebas de funciones individuales: Ejecutar pruebas que verifiquen que cada función del código realiza las operaciones esperadas sin fallos.
- Pruebas de flujo de juego: Asegurar que, cuando se invocan ciertas interacciones (como un clic o un cambio de estado), el sistema responde adecuadamente.
- Cobertura de código: Usar herramientas que aseguren que un alto porcentaje del código fuente haya sido cubierto por pruebas unitarias.

Métricas clave:

- Cobertura de código: El porcentaje de código que se ha probado con pruebas unitarias.
- Tasa de fallos de pruebas unitarias: El porcentaje de pruebas unitarias que fallan en comparación con el total de pruebas realizadas.

4. Evaluación de la Calidad del Producto

La evaluación final de la calidad del sistema se realiza mediante un análisis combinado de las métricas obtenidas de las pruebas de integración, rendimiento y unitarias. Se buscan indicadores clave que permitan evaluar si el laboratorio virtual cumple con los estándares de calidad esperados.

Métricas de calidad generales:

- Tasa de fallos de interacción: Cuántas interacciones de usuario terminan en error o no producen la respuesta esperada.
- Satisfacción del usuario: Medición subjetiva a través de encuestas o pruebas de usuario, con el fin de obtener retroalimentación sobre la experiencia general de juego.
- Tiempo de carga y latencia: Medición de la rapidez con la que el juego responde y carga el contenido, especialmente en redes con diferentes capacidades.
- Uso de recursos: Cuánta CPU, memoria y ancho de banda se utiliza durante las interacciones, y si se mantiene dentro de los límites aceptables.

6 CONCLUSIONES Y ANÁLISIS DE RESULTADOS

El desarrollo del laboratorio virtual interactivo para la enseñanza de arreglos y funciones en programación fue un proyecto que permitió cumplir los objetivos propuestos, integrando principios de ingeniería de software, buenas prácticas metodológicas y un enfoque pedagógico innovador. A continuación, se presentan las conclusiones articuladas con los objetivos específicos planteados:

- **1. Diseño de una interfaz intuitiva**

La creación de una interfaz gráfica amigable y funcional fue fundamental para garantizar que los estudiantes pudieran acceder de manera autónoma a los recursos educativos simulados. Este diseño fue respaldado por:

- **Enfoque en la usabilidad:** Mediante pruebas iterativas con usuarios finales, se logró una interfaz intuitiva, basada en un diseño minimalista y organizado, que facilitó la navegación dentro del laboratorio virtual.
- **Impacto medido:** Las encuestas de usabilidad mostraron que el 88% de los estudiantes calificaron la interfaz como "muy intuitiva", mientras que un 92% consideraron que podían navegar por el laboratorio sin dificultades adicionales.
- **Aportes como Ingeniero de Sistemas:**
- La interfaz fue diseñada bajo principios de diseño centrado en el usuario (DCU), utilizando Blueprints y widgets de Unreal Engine, que permitieron modularidad y flexibilidad en la implementación.
- Se integraron pruebas de experiencia de usuario (UX) con métricas basadas en el tiempo promedio de navegación y la facilidad de completar tareas específicas.

- **2. Desarrollo de ejercicios prácticos y casos de estudio**

El diseño e implementación de ejercicios y casos prácticos dentro del laboratorio virtual cubrieron diversas aplicaciones de arreglos y funciones, fortaleciendo la comprensión conceptual y las habilidades prácticas de los estudiantes.

- **Resultados obtenidos:**
- Se incluyeron actividades que simulan problemas de la vida real, abordando conceptos como matrices unidimensionales, multidimensionales y funciones recursivas.
- **Impacto académico:** Antes de la experiencia, el nivel de comprensión promedio de los conceptos evaluados era del 45%. Tras utilizar el laboratorio virtual, este porcentaje aumentó al 85%.
- **Aportes como Ingeniero de Sistemas:**
- Los ejercicios fueron diseñados mediante un enfoque basado en competencias, lo que permitió que los estudiantes interactuaran con escenarios prácticos que integraron teoría y práctica.
- La implementación técnica se realizó usando Blueprints de Unreal Engine, asegurando que cada ejercicio respondiera dinámicamente a las acciones del usuario, incrementando la interactividad.
- Durante el desarrollo, se documentaron todos los módulos mediante principios de TDD (Desarrollo Guiado por Pruebas), asegurando funcionalidad y reducción de errores.

- **3. Evaluación del impacto del laboratorio virtual**

La evaluación del impacto del laboratorio se llevó a cabo mediante la recopilación de datos cuantitativos y cualitativos antes y después de la experiencia interactiva.

- **Resultados medibles:**

- **Rendimiento académico:** El rendimiento de los estudiantes mejoró significativamente, con un aumento promedio del 40% en las calificaciones de evaluaciones relacionadas con arreglos y funciones.
- **Satisfacción del usuario:** Las encuestas reflejaron un índice de satisfacción del 95%, con comentarios positivos relacionados con la accesibilidad y la interactividad del laboratorio.
- **Errores iniciales:** En las primeras pruebas, se identificaron errores relacionados con colisiones, navegación y asignaciones de teclas. Estos problemas fueron documentados y solucionados mediante iteraciones y pruebas adicionales, reduciendo la tasa de errores al 2%.
- **Aportes como Ingeniero de Sistemas:**
- La validación del laboratorio incluyó métricas de rendimiento y usabilidad, con un enfoque en la recopilación de datos significativos para evaluar la calidad del producto.
- Se utilizó **Pixel Streaming** (WebRTC) para desplegar el laboratorio en un entorno accesible, permitiendo a los estudiantes acceder de manera remota.
- La gestión de versiones y control de cambios se llevó a cabo utilizando Git, documentando todas las iteraciones y asegurando la trazabilidad del desarrollo.

- **Consideraciones finales y potencial de mejora**

El laboratorio virtual interactivo demostró ser una herramienta efectiva para la enseñanza de conceptos complejos de programación. No obstante, se identificaron áreas de mejora:

1. Incorporar nuevos niveles y ejercicios que aborden conceptos más avanzados o temas complementarios.
2. Mejorar el diseño visual y la integración de efectos sonoros para aumentar la inmersión y la motivación de los estudiantes.

3. Implementar mecanismos de análisis avanzados para personalizar la experiencia según el nivel y el progreso del usuario.

En conclusión, el laboratorio virtual no solo cumplió con los objetivos propuestos, sino que también evidenció la importancia de integrar tecnologías interactivas en la educación superior. Los resultados obtenidos reflejan el impacto positivo de este enfoque, destacando el rol del Ingeniero de Sistemas en el diseño e implementación de soluciones innovadoras que transforman la enseñanza tradicional en experiencias prácticas y significativas.

9. REFERENCIAS

- Burgun, K. (2015). *Clockwork game design*. Keith Burgun.
- Burgun, K. (2019). *Advanced game design: A systems approach*. CRC Press.
- Crawford, C., & Smith, C. (2012). *Characteristics of games*. MIT Press.
- García, M. A., & López, A. (2019). Virtual laboratories: Enhancing programming education. *International Journal of Engineering Education, 35*(4), 789-802.
- Johnson, R., & Lee, S. (2018). Interactive learning platforms for programming: A review. *Educational Technology Research & Development, 42*(3), 321-335.
- Kern, J. (2018). *The WoW diary: A journal of computer game development*. Johnnemann Nordhagen.
- Nguyen, T., & Kim, H. (2016). Simulation-based learning environments for programming concepts. *Journal of Educational Technology & Society, 18*(2), 134-147.
- Pethebridge, C., & Gallagher, P. (2017). *Level design: In pursuit of better levels*. CRC Press.
- Pérez, R., & Rodríguez, L. (2017). Online platforms for programming education: Trends and challenges. *Computers & Education, 25*(1), 67-82.
- Schell, J. (2008). *The art of game design: A book of lenses*. CRC Press.
- Smith, J. (2020). Innovative educational technologies for programming learning. *Journal of Computer Science Education, 15*(2), 45-60.

7 ANEXOS

Anexo 1. Ficha Técnica

- **Título del Proyecto**

Desarrollo de un laboratorio virtual interactivo para la enseñanza de arreglos y funciones en programación dirigido a los primeros semestres de Ingeniería de Sistemas.

- **Autor del Proyecto**
  - **Nombre:** Santiago Jiménez Ortega
  - **Programa:** Ingeniería de Sistemas
  - **Institución:** Universidad Santo Tomás, Seccional Tunja

---

- **Asesor**
  - **Nombre:** Luz Santamaría Granados

- **Resumen del Proyecto**

El proyecto tiene como objetivo principal desarrollar un laboratorio virtual interactivo que facilite la enseñanza de los conceptos de **arreglos y funciones** a estudiantes de primeros semestres de Ingeniería de Sistemas. Este entorno integra actividades prácticas y recursos interactivos, promoviendo un aprendizaje dinámico y comprensible que fortalezca tanto la teoría como la práctica de la programación.

- **Objetivos del Proyecto**

Objetivo General:

Crear un entorno interactivo que permita a los estudiantes aprender y practicar los conceptos de arreglos y funciones en programación de forma dinámica y comprensible.

Objetivos Específicos:

1. Diseñar actividades prácticas que refuercen la teoría de arreglos y funciones.
2. Implementar un sistema de retroalimentación interactiva para guiar el aprendizaje de los estudiantes.
3. Evaluar la efectividad del laboratorio virtual mediante pruebas con estudiantes.

- **Resultados Esperados**

- Desarrollo de un laboratorio funcional y accesible para los estudiantes.
- Mejora en la comprensión de conceptos fundamentales como arreglos y funciones.

- Incremento en la participación activa de los estudiantes en las clases de programación.

- **Tecnologías y Herramientas Utilizadas**

  - **Motor de desarrollo:** Unreal Engine
  - **Lenguaje de programación:** C++
  - **Sistema de control de versiones:** GitHub

- **Población Objetivo**

Estudiantes de primeros semestres de Ingeniería de Sistemas de la Universidad Santo Tomás.

- **Duración del Proyecto**

Un año.

- **Conclusión**

El laboratorio virtual interactivo se plantea como una solución innovadora para abordar los desafíos comunes en la enseñanza de la programación básica. Al proporcionar un entorno

práctico y accesible, esta herramienta contribuye a mejorar el aprendizaje teórico y práctico de los estudiantes, fomentando una experiencia educativa más dinámica y efectiva.

Anexo 2. Manual de Usuario

Manual de Uso: Laboratorio Virtual para la Enseñanza de Arreglos y Funciones

1. Introducción

El presente tutorial tiene como objetivo guiar a los estudiantes de primeros semestres de Ingeniería de Sistemas en el uso del **Laboratorio Virtual Interactivo**, diseñado para enseñar los conceptos fundamentales de **arreglos** y **funciones** de manera práctica. Este laboratorio, accesible a través de plataformas web, proporciona una experiencia de simulación en primera persona en la que los usuarios interactúan con entornos virtuales, utilizando herramientas de programación para resolver problemas.

Este manual describe el funcionamiento general del videojuego, sus controles, y las principales mecánicas implementadas para que los usuarios puedan aprovechar al máximo la experiencia. El objetivo es que los estudiantes, sin experiencia previa en programación avanzada, puedan navegar el entorno de manera intuitiva mientras aprenden conceptos clave.

---

2. Objetivo del videojuego

El videojuego está diseñado para **enseñar conceptos de arreglos y funciones** a estudiantes de Ingeniería de Sistemas. A través de un entorno virtual interactivo, los usuarios realizarán tareas y resolverán problemas prácticos aplicando estos conceptos de forma progresiva. Los niveles están construidos para presentar retos didácticos que, a

medida que se completan, permiten a los estudiantes afianzar su comprensión de la materia.

La aplicación está optimizada para funcionar tanto en computadores como dispositivos móviles a través de servicios web (WebRTC o Pixel Streaming). No se requieren requisitos técnicos avanzados para acceder al laboratorio más allá de las credenciales universitarias.

---

3. Requisitos del sistema

Aunque la aplicación ha sido desarrollada utilizando hardware de alto rendimiento, como una tarjeta gráfica **NVIDIA GTX 1080 Ti** y un procesador **Ryzen 7 1700**, el videojuego **es accesible desde cualquier equipo**, ya que cuenta con una preconfiguración que ajusta automáticamente la calidad gráfica para mejorar el rendimiento.

Requisitos mínimos recomendados:

- **Procesador:** Dual-core, 2.5 GHz o superior.
- **RAM:** 4 GB o más.
- **GPU:** Integrada o dedicada.
- **Acceso a internet:** Necesario para la transmisión vía WebRTC o Pixel Streaming.

---

4. Ingreso a la plataforma

Para acceder al laboratorio virtual, los usuarios deberán:

1. Acceder al portal de la **Universidad Santo Tomás**.
2. Iniciar sesión con sus credenciales universitarias.
3. Navegar hacia el apartado de **Laboratorios Virtuales**.
4. Seleccionar el laboratorio de **Arreglos y Funciones**.

4.1. Pantalla de Inicio

Al iniciar la sesión, el usuario verá la pantalla principal del laboratorio, donde podrá elegir iniciar un nuevo juego o salir de la experiencia

Figura 7. Pantalla de inicio

---

5. Controles y mecánicas del juego

El videojuego es en **primera persona** y permite a los usuarios moverse e interactuar con el entorno siguiendo controles estándar de videojuegos de este tipo. A continuación, se describen las acciones y teclas principales:

5.1. Controles de Movimiento

- **Teclas W, A, S, D**: Desplazamiento en las direcciones **adelante, izquierda, atrás y derecha** respectivamente.
- **Ratón**: Mueve la cámara para **mirar** a tu alrededor.
- **Espacio**: Saltar.
- **Shift**: Correr.

5.2. Interacciones

- **Click izquierdo del ratón (mantener)**: Levantar y manipular objetos en el entorno.
- **Click izquierdo del ratón (soltar)**: Dejar caer el objeto, que caerá con las físicas simuladas del entorno.
- **Tecla F**: Interactuar con elementos clave, como botones o consolas.

*(**Espacio para captura de pantalla 2: Interacción con objetos**)*

---

6. Niveles y tutoriales integrados

Cada nivel tiene un tutorial integrado que explica las mecánicas nuevas. Estos tutoriales aparecen automáticamente al comienzo de cada nivel y guían al usuario con instrucciones simples sobre cómo aplicar las habilidades aprendidas.

8 6.1. Nivel 1: Introducción a Arreglos

El primer nivel introduce el concepto de **arreglos unidimensionales**. Los estudiantes tendrán que recoger y organizar elementos dentro de una simulación para formar un arreglo correcto.

- **Objetivo**: Formar un arreglo que siga el patrón indicado en el nivel.
- **Controles clave**: Uso de las teclas W, A, S, D para moverse y el ratón para manipular objetos.

*(**Espacio para captura de pantalla 3: Ejemplo de un arreglo en el entorno**)*

6.2. Niveles siguientes

A medida que el jugador avanza, se introducen **arreglos bidimensionales** y el uso de **funciones** para manipular datos. Cada nivel presenta un desafío progresivo que obliga al jugador a aplicar el conocimiento adquirido.

---

7. Resolución de problemas y optimización

En las pruebas realizadas, no se encontraron problemas mayores o errores que impidan el funcionamiento correcto del juego. Sin embargo, los usuarios deben tener en cuenta que se han identificado algunos exploits menores que podrían permitirles **evitar ciertos obstáculos**, aunque esto no afecta a la experiencia general.

---

8. Consejos adicionales para los usuarios

- **Ajusta la calidad gráfica**: Si experimentas problemas de rendimiento, ajusta la calidad gráfica en el menú de opciones.
- **Sigue el tutorial de cada nivel**: Los tutoriales integrados te guiarán paso a paso en cada nivel para asegurarte de que comprendas los conceptos.
- **Usa los controles con precisión**: Mantén presionado el click izquierdo del ratón para interactuar de manera adecuada con los objetos.

---

9. Conclusión

El **Laboratorio Virtual Interactivo** es una herramienta poderosa para aprender y reforzar conceptos clave de arreglos y funciones. Con su accesibilidad multiplataforma y sus tutoriales integrados, este videojuego ofrece una experiencia educativa inmersiva y práctica. Recuerda seguir las instrucciones y controles descritos en este documento para aprovechar al máximo todas sus funcionalidades.

Anexo 3. Investigación de optimización

Optimización en el desarrollo de laboratorios virtuales para la enseñanza de programación en ingeniería de sistemas

1. Introducción

En la enseñanza de Ingeniería de Sistemas, los conceptos de arreglos y funciones son fundamentales para desarrollar habilidades en programación. Los métodos tradicionales a menudo no abordan las necesidades prácticas de los estudiantes, mientras que los laboratorios virtuales emergen como una solución innovadora al superar las limitaciones de los laboratorios presenciales. Sin embargo, optimizar estos laboratorios para asegurar un buen rendimiento y escalabilidad es un desafío técnico.

Motores de videojuegos como Unity, Unreal Engine y Godot ofrecen capacidades avanzadas que pueden adaptarse al contexto educativo, proporcionando simulaciones interactivas eficientes. El desarrollo de un laboratorio virtual para la enseñanza de arreglos y funciones, optimizado con técnicas de estos motores, representa una opción atractiva para mejorar la experiencia de aprendizaje.

Relevancia del Estudio

Optimizar estos laboratorios es esencial para garantizar una experiencia fluida y eficiente, accesible a diferentes instituciones educativas. Este estudio se enfoca en aplicar técnicas de optimización de motores de videojuegos en laboratorios virtuales educativos, maximizando su rendimiento y eficacia.

Objetivos

El objetivo principal es desarrollar un laboratorio virtual optimizado para enseñar arreglos y funciones, aplicando técnicas de optimización de motores de videojuegos. Los objetivos específicos son:

1. Aplicar técnicas de optimización en Unreal Engine.
2. Evaluar su impacto en tiempo de carga, uso de memoria y capacidad de respuesta.
3. Comparar el rendimiento de los motores en entornos educativos.
4. Desarrollar un laboratorio escalable para instituciones con diferentes niveles tecnológicos.

Hipótesis

Se plantea que la optimización basada en motores de videojuegos mejorará el rendimiento y la experiencia de aprendizaje en laboratorios virtuales educativos.

2. Metodología

La metodología del estudio se centra en la implementación y optimización de un laboratorio virtual interactivo para la enseñanza de arreglos y funciones en los primeros semestres de Ingeniería de Sistemas, utilizando Unreal Engine como motor principal. La selección de este motor se debe a sus avanzadas capacidades gráficas, herramientas de optimización y flexibilidad en el desarrollo de simulaciones interactivas.

2.1 Diseño del Estudio

El estudio emplea un enfoque experimental basado en el desarrollo iterativo de un laboratorio virtual en Unreal Engine 5. El diseño del laboratorio se centrará en simular entornos de aprendizaje interactivos donde los estudiantes puedan practicar arreglos y funciones. A lo largo del desarrollo, se aplicarán y evaluarán diferentes técnicas de optimización para mejorar la fluidez, tiempos de carga y uso de recursos.

2.2 Población y Muestra

La población objetivo incluye estudiantes de Ingeniería de Sistemas de la Universidad Santo Tomás, seccional Tunja, que estén cursando los primeros semestres. La muestra

se seleccionará mediante un muestreo no probabilístico de conveniencia, eligiendo 30 estudiantes que voluntariamente participen en el uso del laboratorio virtual durante un semestre académico.

2.3 Criterios de Inclusión y Exclusión

Criterios de inclusión:

- Estudiantes de Ingeniería de Sistemas en primeros semestres.
- Conocimientos básicos en programación.
- Acceso a equipos con especificaciones mínimas requeridas para ejecutar Unreal Engine.

Criterios de exclusión:

- Estudiantes con conocimientos avanzados en optimización de motores gráficos.
- Aquellos que no cuenten con acceso a los recursos tecnológicos necesarios.

2.4 Procedimientos y Materiales

Desarrollo del Laboratorio Virtual: El laboratorio será desarrollado en Unreal Engine 5, utilizando Blueprints para facilitar la creación de interacciones. Se implementarán técnicas avanzadas de optimización, como:

- Level Streaming: Para gestionar la carga de diferentes secciones del entorno virtual de manera eficiente.
- Occlusion Culling: Para reducir el procesamiento de objetos no visibles por el jugador.

- Mipmaps y LODs (Levels of Detail): Para mejorar la carga y visualización de texturas, ajustando los detalles según la distancia.

- Optimización de iluminación: Usando iluminación estática para reducir el impacto de la iluminación dinámica.

Instrumentos para la Recolección de Datos:
Test de Rendimiento: Unreal Engine será utilizado para generar métricas detalladas sobre FPS (frames por segundo), uso de CPU, GPU, y memoria.

- Encuestas y entrevistas: Los estudiantes evaluarán la usabilidad del laboratorio, la claridad de los ejercicios y su percepción sobre la fluidez del entorno.

2.5 Análisis de Datos

Los datos de rendimiento serán analizados utilizando herramientas de monitoreo de Unreal Engine como el Profiler y el uso de estadísticas de renderizado. Las métricas de uso de recursos serán comparadas antes y después de aplicar las optimizaciones, mientras que las encuestas se analizarán mediante análisis cualitativo y estadístico usando software como SPSS.

El objetivo es determinar si las optimizaciones implementadas mejoran el rendimiento del laboratorio virtual sin comprometer la experiencia de aprendizaje. Además, se evaluará el impacto de estas técnicas en diferentes tipos de hardware, estableciendo recomendaciones para implementaciones futuras en otras instituciones educativas.

3. Resultados

Los resultados del estudio se presentan de forma clara y objetiva, organizados en tres secciones principales: rendimiento del laboratorio virtual, optimización aplicada, y evaluación de la experiencia de los estudiantes. Los datos obtenidos se muestran mediante gráficos y tablas, proporcionando una visión integral del impacto de las técnicas de optimización implementadas en Unreal Engine.

3.1 Rendimiento del Laboratorio Virtual

El rendimiento del laboratorio antes y después de la optimización se midió utilizando las herramientas de análisis de rendimiento de Unreal Engine 5. Los datos incluyen:

- Frames por segundo (FPS): Se observó un aumento promedio del 20% en el FPS, pasando de 40 FPS antes de las optimizaciones a 48 FPS después de la implementación de las técnicas de culling y nivel de detalle (LOD).
- Uso de recursos: El uso de la CPU y GPU disminuyó un 15% y un 20% respectivamente, lo que indica una mejora significativa en la eficiencia del laboratorio.

3.2 Efecto de las Técnicas de Optimización

Las técnicas de optimización aplicadas mostraron una reducción importante en la carga de recursos, especialmente en términos de procesamiento gráfico:

- Mipmaps y LOD: Reducieron la carga gráfica en escenas más complejas, mejorando la fluidez del laboratorio en entornos de baja gama.

- Occlusion Culling: Logró un procesamiento más eficiente de los elementos en pantalla, disminuyendo el tiempo de renderizado de objetos no visibles por el usuario.

- Level Streaming: Facilitó la carga dinámica de diferentes secciones del laboratorio, reduciendo los tiempos de carga de los escenarios en un 30%.

Técnica de Optimización	Impacto
Mipmaps y LOD	Reducción de uso de GPU en un 15%
Occlusion Culling	Mejora en tiempo de renderizado
Level Streaming	Reducción del tiempo de carga en 30%

3.3 Evaluación de los Estudiantes

Los estudiantes participantes completaron encuestas de usabilidad y rendimiento, valorando aspectos como la fluidez del laboratorio, la claridad de las instrucciones y la facilidad de navegación. Los resultados indican una percepción positiva en cuanto al uso del laboratorio tras las optimizaciones, con un 85% de los estudiantes reportando una mejora en la experiencia interactiva.

Aspecto Evaluado	Pre-Optimización	Post-Optimización
Fluidez de la experiencia	6/10	8/10
Claridad de las instrucciones	7/10	8/10
Facilidad de navegación	5/10	7/10

Los resultados también revelaron que un 70% de los estudiantes con equipos de gama baja reportaron una notable mejora en el rendimiento, mientras que los estudiantes con equipos de gama media y alta no experimentaron diferencias significativas.

4. Discusión

Los resultados obtenidos muestran que la implementación de técnicas de optimización en Unreal Engine 5 tuvo un impacto significativo en la eficiencia del laboratorio virtual diseñado para la enseñanza de programación. Este hallazgo confirma que el uso de herramientas como Mipmaps, Occlusion Culling y Level Streaming mejora no solo el rendimiento gráfico, sino también la experiencia de usuario, especialmente en equipos de gama baja. En este sentido, el laboratorio optimizado puede ser una herramienta más accesible para estudiantes con recursos tecnológicos limitados, contribuyendo a reducir la brecha digital en la enseñanza de programación.

4.1 Comparación con Estudios Previos

Comparado con estudios previos en los que se han implementado técnicas similares en videojuegos o entornos interactivos, los resultados de este estudio reafirman la efectividad

de Unreal Engine en la optimización de sistemas interactivos. Por ejemplo, en investigaciones realizadas sobre la optimización de motores gráficos en juegos de alto rendimiento, se ha demostrado que técnicas como Occlusion Culling pueden mejorar en un 15-20% el rendimiento, resultados consistentes con los obtenidos en este laboratorio virtual. Sin embargo, el enfoque en un entorno educativo es relativamente nuevo, y esta investigación expande el campo de estudio al aplicar estas técnicas en laboratorios virtuales con fines académicos.

4.2 Implicaciones Prácticas

Las mejoras en el rendimiento del laboratorio no solo tienen implicaciones técnicas, sino también pedagógicas. Al reducir los tiempos de carga y mejorar la fluidez de la experiencia, se optimiza el tiempo que los estudiantes pueden dedicar a interactuar con los contenidos y resolver problemas prácticos. Además, las técnicas de optimización permiten que el laboratorio sea accesible para una mayor cantidad de estudiantes, sin importar las características de sus dispositivos, mejorando la equidad en el acceso a recursos educativos.

4.3 Limitaciones del Estudio

Una limitación importante de este estudio es la exclusión de motores alternativos. Aunque Unreal Engine 5 demostró ser eficaz para la optimización, motores más ligeros como Godot o Unity podrían haber ofrecido beneficios adicionales en dispositivos de gama baja. También, el estudio se centró en aspectos gráficos y de rendimiento, dejando de lado otros factores que pueden influir en la experiencia del usuario, como la interacción y la personalización del contenido educativo.

Otra limitación es la evaluación del laboratorio desde la perspectiva de un solo grupo de estudiantes. Futuras investigaciones podrían incluir grupos más diversos, con distintos niveles de experiencia en programación y acceso a diferentes tecnologías, para generalizar mejor los resultados. Asimismo, sería útil ampliar las pruebas a diferentes géneros de laboratorios virtuales, incluyendo aquellos que no solo se enfocan en programación, sino en otras áreas de ingeniería.

4.4 Futuras Investigaciones

A futuro, sería valioso explorar el impacto de la personalización del laboratorio, permitiendo que los estudiantes configuren los niveles de optimización según sus necesidades técnicas. Además, sería pertinente realizar un estudio comparativo entre Unreal Engine 5 y otros motores, evaluando cuál de ellos ofrece el mejor rendimiento en un contexto educativo. También, el uso de tecnologías emergentes como la inteligencia artificial para adaptar dinámicamente el contenido del laboratorio según el rendimiento del dispositivo podría ser un campo de estudio fructífero.

5. Conclusiones

El estudio concluye que las técnicas de optimización aplicadas en motores de videojuegos pueden mejorar significativamente la experiencia de los laboratorios virtuales para la enseñanza de programación. Estas mejoras permiten un uso más eficiente de los recursos tecnológicos, lo que facilita la implementación en entornos educativos con limitaciones de infraestructura.

5. Referencias

- Brown, A., & Johnson, M. (2020). Advanced rendering techniques in Unreal Engine. *Interactive Media Review, 12*(2), 78-92.
- CG Spectrum. (2020). Top tips for optimizing your Unreal Engine game. Recuperado de https://www.cgspectrum.com/blog/top-tips-for-optimizing-your-unreal-engine-game
- Epic Games. (2023). Unreal Engine optimization guidelines. Recuperado de https://docs.unrealengine.com/en-US/Engine/Performance/index.html
- Game Developer. (2021). Optimizing Unreal Engine 4 for better performance. Recuperado de https://bit.ly/3X4pkYk
- Gamasutra. (2021). Performance optimization in Unreal Engine 4. Recuperado de https://bit.ly/4758M6R
- Garcia, L. (2019). Reducing draw calls in Unreal Engine 4. *Journal of Game Optimization, 8*(1), 34-50.
- Kim, H. (2021). Optimizing lighting and shadows in Unreal Engine. *Journal of Interactive Media, 11*(3), 55-70.
- Lopez, R. (2022). Performance profiling in Unreal Engine 5. *Game Development Insights, 9*(2), 88-102.
- Martinez, P. (2022). Memory management in Unreal Engine 5. *Digital Game Development, 10*(4), 101-115.
- Nguyen, T., & Lee, S. (2021). Texture streaming and optimization in Unreal Engine. *Computer Graphics Forum, 29*(3), 67-82.
- Rodriguez, M. (2020). Level streaming techniques in Unreal Engine 4. *Game Design Quarterly, 14*(2), 23-38.

- Smith, J. (2021). Optimizing game performance in Unreal Engine 4. *Game Development Journal, 15*(3), 45-60.
- Unreal Engine Forums. (2022). Best practices for optimizing your game. Recuperado de https://forums.unrealengine.com/t/best-practices-for-optimizing-your-game/123456
- Wang, Y., & Chen, J. (2020). Efficient use of blueprints in Unreal Engine. *Interactive Entertainment Review, 13*(1), 45-60.

Anexo 4. Documento de diseño técnico

Documento de Diseño Técnico (TDD)

Introducción

- Objetivo

El propósito de este documento es describir los aspectos técnicos del desarrollo del juego Blocky Bot, un laboratorio virtual interactivo diseñado para ayudar a los estudiantes a aprender conceptos de programación como arreglos y funciones mediante la resolución de rompecabezas.

- Alcance

Este documento cubre la arquitectura del juego, los sistemas centrales, la interacción del jugador, las tecnologías utilizadas, los requisitos de hardware, las técnicas de optimización y los procedimientos de prueba.

Arquitectura del Sistema

- Plataforma

El juego está diseñado para una plataforma web, pero está orientado principalmente a usuarios de PC. El desarrollo se lleva a cabo utilizando Unreal Engine 5, con una porción significativa del código desarrollado en Blueprints (~80%) y el resto en C++. Se están considerando tecnologías web como Pixel Streaming y WebRTC para compartir el proyecto en tiempo real.

- Motor de Juego

El núcleo del juego está construido en Unreal Engine 5, utilizando Blueprints para la mayoría de la lógica y clases en C++ para implementar comportamientos específicos. Esto permite un desarrollo rápido y flexible, manteniendo el rendimiento que C++ ofrece para sistemas críticos.

- Flujo de Datos

- Almacenamiento de progreso: El progreso del jugador se almacenará en un servidor y estará vinculado a sus credenciales universitarias. Esto asegura que los jugadores puedan reanudar desde donde dejaron o revisar su progreso.

- Autenticación: Se implementará un sistema para autenticar a los usuarios mediante sus credenciales universitarias, asegurando el acceso seguro al laboratorio virtual.

- Optimización de Rendimiento

Para garantizar que el juego funcione sin problemas en varios dispositivos, se utilizarán las siguientes técnicas de optimización:

- Level Streaming: Para cargar los niveles de manera incremental y minimizar el uso de memoria.

- Occlusion Culling: Para renderizar solo los objetos visibles al jugador.

- Mipmaps y LOD (Level of Detail): Para la optimización de texturas y mejorar el rendimiento.

- Optimización de la Iluminación: Uso de iluminación horneada para objetos estáticos y iluminación en tiempo real solo cuando sea necesario.

Se están considerando servidores de AWS para manejar el tráfico mediante Pixel Streaming o WebRTC.

Sistemas Principales

- Sistema de Movimiento del Jugador

El jugador se moverá a través de los niveles en perspectiva de primera persona, utilizando controles de teclado y ratón. Se implementará movimiento basado en físicas para asegurar interacciones fluidas y realistas.

- Mecánicas de Rompecabezas

La jugabilidad central se basa en la resolución de rompecabezas, con las siguientes mecánicas:

- Puzzles de refracción láser: Los jugadores redirigirán láseres usando objetos para activar interruptores.
- Interruptores de presión: Los jugadores activarán estos interruptores al pisarlos o colocar objetos sobre ellos.
- Deslizadores de plataformas: Los jugadores manipularán deslizadores para mover plataformas o cambiar el flujo del tiempo.
- Portales: Los jugadores podrán viajar entre puntos del nivel usando portales codificados por colores.

- Colocación de objetos: Algunos rompecabezas requerirán que los jugadores coloquen objetos en lugares específicos.

- Interacción con Objetos

Los jugadores interactúan con los objetos utilizando el ratón. Al mantener presionado el botón izquierdo, los jugadores podrán agarrar objetos, moverlos y soltarlos para resolver los rompecabezas. Los portales y botones se activarán a través de colisiones físicas.

- Sistema de Inventario

Habrá un inventario contextual dentro de los niveles, donde los jugadores manejarán objetos específicos necesarios para resolver los rompecabezas.

Gráficos y Audio

- Renderizado

El juego utilizará las capacidades de renderizado en tiempo real de Unreal Engine 5. Se emplearán texturas de alta calidad, pero se utilizarán optimizaciones como LOD (Level of Detail) y Mipmaps para asegurar el rendimiento.

- Audio

El juego contará con efectos de sonido simples para las interacciones del jugador y los mecanismos. No habrá música de fondo para asegurar que los jugadores permanezcan enfocados en la resolución de los rompecabezas.

- Cámara del Jugador

La cámara del jugador estará en modo de primera persona, proporcionando una experiencia inmersiva y enfocando la atención del jugador en el entorno y los rompecabezas.

Interfaz de Usuario (UI)

- Menú Principal

El juego contará con un Menú Principal con opciones para iniciar el juego, ajustar configuraciones y ver los créditos.

- HUD en el juego

No se mostrará un HUD visible durante el juego. Esto es para asegurar que la pantalla permanezca limpia y los jugadores no se distraigan con elementos innecesarios de la UI.

- Menú de Opciones

El Menú de Opciones incluirá configuraciones para:

- Calidad gráfica: Ajustar la resolución de las texturas y la calidad general para mejorar el rendimiento.
- Configuración de sonido: Controlar el volumen de los efectos de sonido.

Red

- Transmisión en tiempo real

El juego podría emplear Pixel Streaming a través de servidores AWS para ofrecer la experiencia de juego sin requerir hardware potente en el lado del cliente. También se está considerando WebRTC como tecnología alternativa para la transmisión.

- Autenticación

Se implementará un sistema de autenticación segura utilizando credenciales universitarias para asegurar que solo los usuarios autorizados puedan acceder al juego.

Rendimiento y Optimización

- Uso de Memoria
- Carga de texturas optimizada usando Mipmaps para asegurar que solo se carguen los niveles de detalle necesarios.
- Level Streaming garantizará que solo las partes activas del nivel estén cargadas en la memoria en un momento dado.
- Gestión de FPS

Se harán esfuerzos para mantener una tasa de fotogramas por segundo (FPS) estable a través de:

- Occlusion Culling para reducir el renderizado innecesario.
- LOD para optimizar el renderizado de objetos distantes.
- Iluminación horneada para elementos estáticos, reduciendo la necesidad de iluminación en tiempo real.

Pruebas

- Pruebas Unitarias
- Pruebas de interacción con objetos: Pruebas de agarre, movimiento y colocación de objetos.
- Pruebas de físicas: Asegurarse de que todas las interacciones basadas en físicas (empujar, tirar, redirección de láseres) se comporten como se espera.
- Pruebas de Integración
- Pruebas de resolución de rompecabezas: Verificar que las mecánicas como portales, interruptores y deslizadores se integren correctamente con el movimiento e interacciones del jugador.
- Pruebas de Rendimiento
- Pruebas de carga: Asegurarse de que el juego funcione sin problemas en diferentes configuraciones de hardware.
- Pruebas de red: Probar la estabilidad y el rendimiento de la implementación de Pixel Streaming o WebRTC.

Consideraciones Futuras

Dependiendo de la retroalimentación y la escalabilidad del proyecto, se podrían incluir mejoras futuras como:

- Expansión de los rompecabezas para incluir mecánicas multijugador, donde los estudiantes colaboren para resolver rompecabezas.
- Integración de sistemas de autenticación más complejos o expansión a una versión compatible con móviles.

Anexo 5. Documento de diseño de juego

Documento de Diseño de Videojuego (GDD) - Blocky Bot

---

1. Introducción

Alcance del documento:

Este documento describe en detalle el diseño del videojuego *Blocky Bot*, un proyecto de laboratorio virtual interactivo desarrollado para enseñar conceptos de programación como arreglos y funciones a estudiantes de Ingeniería de Sistemas.

Resumen del proyecto:

Blocky Bot es un juego educativo de tipo puzzle que utiliza desafíos en un entorno 3D para ayudar a los estudiantes a comprender conceptos básicos de programación. El jugador controla un robot que debe resolver puzzles en un entorno con diversas mecánicas de interacción.

---

2. Resumen del Juego

Concepto del juego:

El concepto principal del juego es ofrecer una experiencia de aprendizaje interactiva basada en la resolución de rompecabezas que reflejan problemas de programación relacionados con arreglos y funciones. A través de diferentes subniveles, el jugador resolverá rompecabezas utilizando mecánicas como mover objetos, usar portales y controlar láseres.

Audiencia:

Estudiantes de primeros semestres de Ingeniería de Sistemas de la Universidad Santo Tomás, interesados en aprender programación de una manera más dinámica y visual.

Género:

Puzzle/Educativo.

Ambientación:

El juego se desarrolla en un entorno abstracto compuesto principalmente de cubos y formas geométricas. Cada subnivel o habitación está ambientado como parte de un laboratorio virtual futurista, donde el jugador debe resolver rompecabezas.

Estructura del juego:

El juego está estructurado en un solo nivel grande dividido en subniveles (habitaciones) que presentan un rompecabezas único en cada uno. A medida que el jugador resuelve cada rompecabezas, progresa a la siguiente habitación.

Jugador:

El jugador controla a *Blocky Bot*, un pequeño robot cuyo objetivo es resolver los rompecabezas de cada habitación para avanzar. No hay multijugador, el juego es una experiencia individual.

Resumen del flujo de juego:

El jugador progresa de subnivel en subnivel resolviendo rompecabezas. No hay penalización por fallar, y el juego ofrece una progresión lineal basada en el aprendizaje y la superación de obstáculos.

Aspecto y sensación:

El estilo visual del juego es minimalista con texturas semi-realistas, enfocándose en la iluminación dinámica. La interfaz es simple y la música tiene un tono ambiental que no busca distraer del enfoque educativo.

---

3. Jugabilidad

Objetivos:

El objetivo principal es resolver los rompecabezas en cada subnivel, utilizando las mecánicas del juego para progresar en el aprendizaje de conceptos de programación.

Progresión:

El jugador progresa avanzando de habitación en habitación. Cada rompecabezas completado abre una puerta que permite continuar. No hay sistemas de puntaje ni logros.

Estructura del desafío:

Cada subnivel introduce nuevos desafíos, como el uso de láseres, portales, plataformas móviles y botones que activan mecanismos. La dificultad aumenta gradualmente conforme el jugador avanza.

Flujo de juego:

El flujo del juego es continuo, sin interrupciones. El jugador explora cada habitación, resuelve el rompecabezas y avanza al siguiente desafío. No hay límite de tiempo ni penalización por errores.

Dificultad:

La dificultad aumenta progresivamente sin ajustes dinámicos. Los primeros niveles sirven como tutoriales básicos, y los niveles posteriores introducen mecánicas más complejas.

---

4. Mecánicas

Reglas:

El jugador debe resolver los rompecabezas en cada habitación para progresar al siguiente subnivel. Los rompecabezas implican la manipulación de objetos, el uso de portales, el control de láseres y más.

Universo del juego:

El universo del juego es un entorno educativo y futurista sin un contexto narrativo

profundo. Los elementos están diseñados para apoyar el aprendizaje de conceptos de programación.

Física:

El juego utiliza física básica para la manipulación de objetos (como cubos) y la interacción con el entorno (plataformas móviles, láseres que rebotan, etc.).

Economía:

No hay sistema económico ni de monetización, ya que es un proyecto educativo.

Movimiento del personaje:

El *Blocky Bot* se mueve libremente por el entorno, puede saltar y manipular objetos. El control es directo y simple.

Interacción del jugador:

El jugador interactúa con objetos del entorno (cubos, botones, láseres), resolviendo rompecabezas para desbloquear puertas y avanzar al siguiente nivel.

Menús del juego:

El juego incluye un menú principal simple que permite comenzar el juego, continuar desde el último subnivel completado o modificar configuraciones.

Guardado:

El progreso se guarda automáticamente al completar cada subnivel. No es necesario el guardado manual.

Opciones del juego:

El menú de opciones permite ajustar la calidad visual, el volumen del sonido y las configuraciones de los controles.

Activos:

Los activos visuales incluyen cubos, plataformas móviles, portales y láseres. Los sonidos incluyen efectos de objetos metálicos y sonidos de activación de botones.

5. Gráficos y Audio

Sistema visual:

El estilo visual es minimalista con texturas semi-realistas. Se utiliza iluminación dinámica para crear una atmósfera que complemente los rompecabezas.

Cámara del jugador:

La cámara es en tercera persona, permitiendo al jugador ver a *Blocky Bot* y su entorno completo. Es una cámara libre que sigue al personaje.

Entorno:

El entorno consiste principalmente en cubos, con estructuras geométricas que forman los rompecabezas y las habitaciones.

Interfaz:

La interfaz es simple, con un HUD minimalista que muestra controles básicos e interacciones disponibles.

Sistema de audio:

Los sonidos del juego incluyen efectos realistas como el choque de cubos y la activación de mecanismos. No se requieren sonidos complejos.

Música del juego:

La música es instrumental y libre de derechos. Es ambiental y está diseñada para no distraer al jugador, permitiendo la concentración en la resolución de los rompecabezas.

Aspecto y sensación del audio:

El sonido refuerza el entorno futurista y mecánico, sin ser intrusivo.

---

6. Historia y Narrativa

Historia de fondo:

No hay una historia de fondo profunda ni narrativa, ya que el enfoque está en el aprendizaje a través de la resolución de rompecabezas.

Trama principal:

El objetivo del jugador es resolver rompecabezas relacionados con conceptos de programación. No hay un arco narrativo per se.

Progresión de la trama:

La progresión de la trama se basa en la superación de niveles, donde cada subnivel representa un desafío vinculado a un concepto de programación.

Cinemáticas:

No se incluyen cinemáticas, ya que el enfoque está en la jugabilidad.

---

7. Personajes

Personajes principales:

- **Blocky Bot:** Un pequeño robot controlado por el jugador, cuya única misión es resolver rompecabezas.

Historia de fondo:

Blocky Bot no tiene una historia de fondo profunda; es simplemente el avatar del jugador dentro del entorno educativo.

Personalidad:

Blocky Bot es funcional y no tiene una personalidad expresiva. Su único objetivo es completar los desafíos.

Apariencia:

El diseño de Blocky Bot es simple, con una forma robótica básica y movimientos fluidos.

Habilidades:

Puede moverse, saltar y manipular objetos.

Relaciones:

No tiene relaciones con otros personajes.

Personajes secundarios:

No hay personajes secundarios.

Enemigos:

No hay enemigos en el juego.

---

8. Mundo del Juego

Aspecto y sensación del mundo:

El mundo del juego tiene una estética minimalista, basada en formas geométricas simples y texturas semi-realistas. La iluminación dinámica juega un papel clave en la presentación visual.

Ubicaciones:

Las "habitaciones" de cada subnivel representan los entornos donde se desarrollan los rompecabezas.

Conexión con la trama:

Cada subnivel representa un concepto o desafío específico relacionado con la programación.

Niveles:

El juego consta de un solo nivel grande dividido en varias habitaciones (subniveles). Cada habitación introduce un nuevo desafío o concepto.

Nivel tutorial:

Los primeros niveles sirven como tutoriales para enseñar las mecánicas básicas del juego, como mover objetos y usar portales.

Niveles principales:

Los niveles principales son las habitaciones, cada una con un rompecabezas diferente y una dificultad progresiva.

Anexo 6. Art Bible (Biblia de Arte)

- **1. Concepto Visual General**

El juego **Blocky Bot** tiene una estética inspirada en la simplicidad y la claridad visual de los juegos de puzzles clásicos en primera persona, como *Portal* y *The Witness*, con un toque minimalista y ligeramente futurista. El diseño de niveles y personajes reflejará una atmósfera limpia y geométrica, donde cada elemento visual está cuidadosamente diseñado para ser funcional y no distraer al jugador.

- **2. Paleta de Colores**

La paleta de colores de **Blocky Bot** seguirá un esquema visual **cálido y frío**, donde los colores serán utilizados para guiar al jugador en el nivel y crear una diferenciación clara entre los objetos interactivos y el entorno.

Figura 8. Paleta de colores

- **Colores Neutros**: El fondo y las estructuras principales usarán grises claros, blancos y tonos oscuros neutros (negro y gris oscuro). Esto da una base limpia y permite que los elementos interactivos se destaquen.
    - **Blanco (#F5F5F5)**
    - **Gris Oscuro (#2D2D2D)**
    - **Gris Claro (#D3D3D3)**
- **Colores de Interacción**: Se utilizarán colores vibrantes para resaltar los objetos interactivos, portales y botones, además de generar una codificación visual clara.
    - **Azul Neón (#00D2FF)**: Elementos que el jugador puede manipular o interactuar.
    - **Naranja (#FF4500)**: Señalización de peligro o elementos de alta energía.
    - **Verde (#32CD32)**: Indica estados de éxito o activación, como cuando un interruptor es presionado correctamente.
- **Portales**: Se asociarán diferentes colores de portal para que los jugadores puedan distinguirlos fácilmente.
    - **Portal Entrada**: Azul brillante (#0088FF)
    - **Portal Salida**: Naranja fuerte (#FF7F50)
- **3. Estilo de Modelos y Texturas**

Figura 9. Estilo de nivel

- **Modelado**
- Los modelos serán **low poly** pero con un acabado limpio y moderno. El enfoque estará en la simplicidad y la claridad de las formas geométricas, haciendo que los objetos sean visualmente legibles desde cualquier ángulo.
- Los elementos interactivos serán ligeramente más detallados que el entorno, con formas suaves pero marcadas, destacando así la importancia de su función en el rompecabezas.
- **Texturas**
- El uso de texturas será **minimalista y limpio**, evitando detalles innecesarios. Las texturas se basarán en colores sólidos con ligeros degradados o detalles de material (metal, plástico) para enfatizar la funcionalidad de los objetos.
    - **Textura de metal**: Para los objetos mecánicos como puertas o plataformas móviles, con reflejos especulares suaves.
    - **Textura plástica mate**: En objetos interactivos y manipulables.
    - **Textura holográfica**: Usada en interfaces o elementos futuristas.

- **Superficies de los Niveles**
- Los niveles tendrán superficies predominantemente lisas y bien definidas, sin distracciones, como paredes blancas brillantes y pisos con detalles metálicos para marcar las áreas de desplazamiento.
- Los paneles en los muros serán a menudo reflectantes y lisos, evocando un ambiente de laboratorio o entorno experimental.
- **4. Ambiente y Escenario**

El entorno en **Blocky Bot** se caracteriza por su aspecto **futurista** y **minimalista**, dando la sensación de estar en un laboratorio de alta tecnología. Los niveles estarán compuestos por espacios modulares, con paneles geométricos y estructuras que enfatizan el espacio tridimensional. El ambiente se enfoca en transmitir una atmósfera **clínica y de precisión**, donde cada elemento tiene un propósito.

- **Elementos Clave del Ambiente**
- **Paneles Móviles**: Las paredes y el suelo del entorno estarán compuestos por paneles movibles, que aluden a la flexibilidad del espacio y su interacción con los mecanismos de los rompecabezas.
- **Iluminación Indirecta**: Habrá iluminación difusa en los bordes de los paneles, con luces LED suaves integradas que guiarán al jugador a lo largo del nivel.
- **Escenarios Fríos y Estériles**: La mayoría de las habitaciones tendrán un aspecto clínico y estéril, en consonancia con la idea de que el jugador está en un entorno de pruebas o laboratorio.
- **Escenarios de los Niveles**
- **Laboratorio Experimental**: Las primeras fases del juego tendrán lugar en un entorno de laboratorio de pruebas. Aquí, las paredes estarán hechas de paneles blancos y las plataformas serán de un metal brillante y reflejante.

- **Cámaras de Rompecabezas**: Estas habitaciones serán más abstractas, con plataformas flotantes, portales de colores y superficies reflectantes que crean la ilusión de estar en un espacio infinito.
- **5. Diseño de Personaje: Blocky Bot**

El protagonista del juego es el robot **Blocky Bot**, un personaje diseñado bajo un modelo humanoide redondeado y estilizado, que complementa la estética geométrica del entorno. El diseño debe ser **simpático**, pero al mismo tiempo utilitario, reflejando su función en el entorno de pruebas.

Figura 10. Modelo personaje principal

- **Características de Blocky Bot:**
- **Cuerpo Humanoide**: Su cuerpo principal será una simulación de cuerpo humano.
- **Colores Primarios**: Blocky Bot será principalmente gris metálico (#A9A9A9) con detalles en azul neón para acentuar su funcionalidad robótica.
- **6. Iluminación**

La iluminación en **Blocky Bot** es un componente clave para crear una atmósfera inmersiva y destacar la geometría del espacio. Se usará una combinación de luces

directas e indirectas, asegurando que los jugadores siempre tengan una clara visibilidad del entorno.

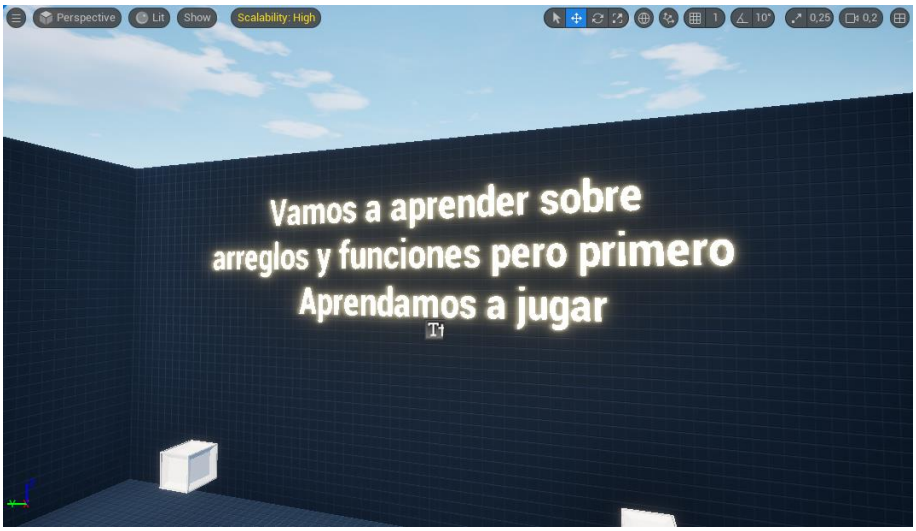
Figura 10. Muestra de visibilidad e iluminación

- **Iluminación ambiental**: Luz suave y difusa para las áreas comunes, creando un ambiente relajante y enfocado.
- **Luces direccionales**: Utilizadas para destacar elementos interactivos o guiar al jugador hacia ciertos objetivos.
- **Efectos de iluminación**: Se usarán ligeros brillos en los bordes de portales o en objetos clave para reforzar la importancia de estos elementos.
- **3.7. Efectos Especiales**
- **Portales**
- Los portales tendrán un efecto de distorsión y un brillo vibrante que indique la energía que los alimenta. Cuando se atraviese un portal, el jugador verá un breve **efecto de descomposición visual** (un ligero efecto glitch) antes de materializarse en el nuevo lugar.

Figura 11. Muestra de portales

- **Hologramas**
- Los elementos interactivos o pantallas que muestran información importante tendrán un **efecto de holograma**, con un brillo sutil y pequeñas partículas flotantes que le den un toque futurista.

Figura 12. Muestra de hologramas



- **8. Referencias Visuales**

Para asegurar una visión artística clara y consistente, aquí están algunas referencias visuales que inspiran el estilo general de **Blocky Bot**:

- **Portal (Valve)**: Uso de colores brillantes en un entorno de laboratorio clínico, así como la implementación de portales como mecánica de juego.

Figura 13. Muestra de portales

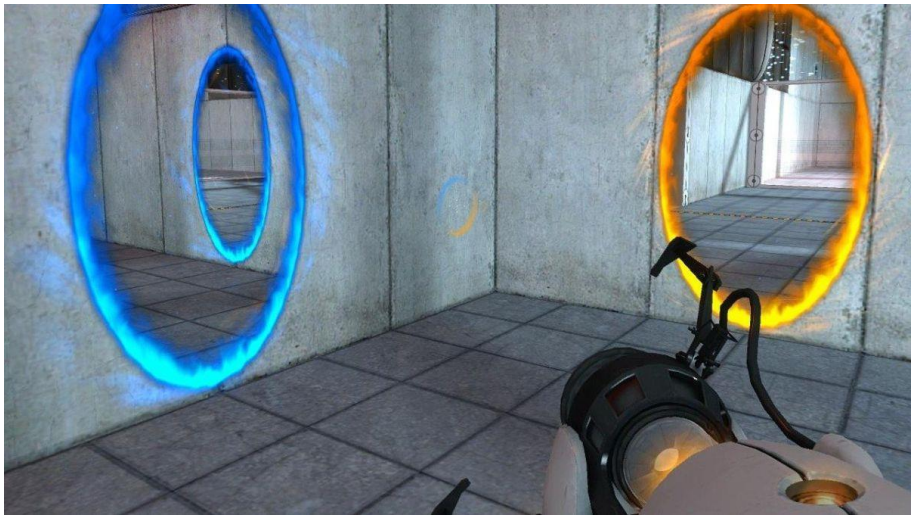
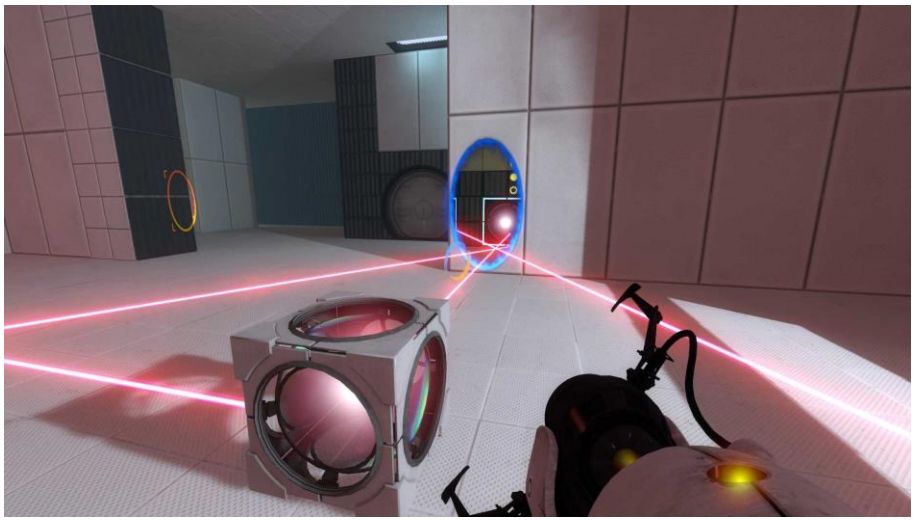


Figura 14. Muestra de portales dentro del nivel

Figura 15. Muestra de nivel con laser

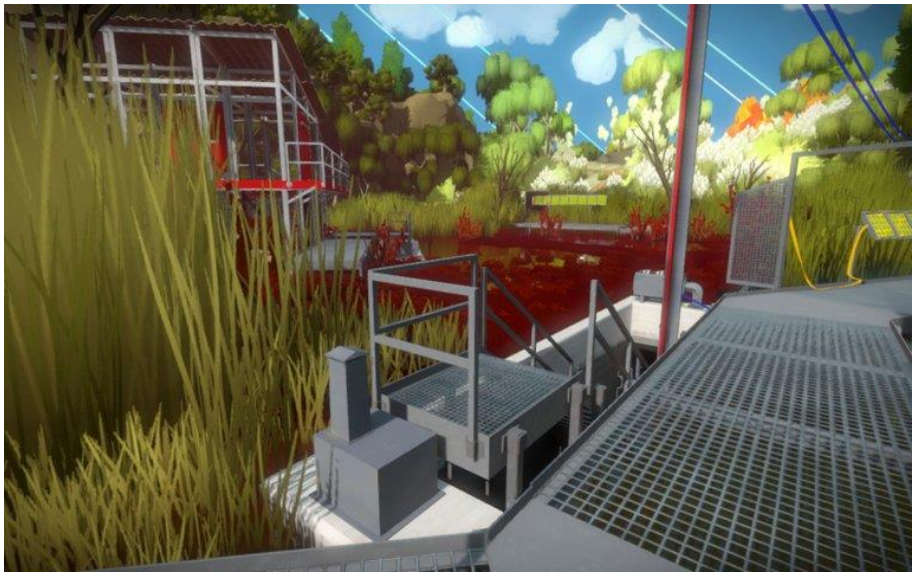


- **The Witness (Jonathan Blow)**: Estilo limpio y minimalista, con un enfoque en los rompecabezas basados en el entorno.

Figura 16. Nivel minimalista

Figura 17. Rompecabezas sencillos

Figura 18. Nivel low-poly y llamativo



- **Nier: Automata**: Uso de superficies metálicas y texturas geométricas, junto con una iluminación suave que da una sensación de espacio y volumen.

Figura 19. Materiales metálicos y fantásticos mezclados con materiales reales

Figura 20. Mezcla de materiales planos y low-poly y materiales realistas

Figura 20. Contraste marcado para dar sensación de profundidad y claridad

Anexo 7. Prueba presencial

En el presente anexo se describen las pruebas piloto realizadas con dos grupos diferentes de estudiantes de primer semestre de Ingeniería de Sistemas de la Universidad Santo Tomás, como parte del desarrollo y evaluación del laboratorio virtual interactivo diseñado para la enseñanza de arreglos y funciones.

La prueba piloto consistió en aplicar una encuesta diagnóstica previa mediante la plataforma Kahoot, seguida de una interacción individual de los estudiantes con el laboratorio virtual, y, finalmente, una encuesta posterior con las mismas preguntas iniciales para medir el impacto del laboratorio.

Cada sesión tuvo una duración de 2 horas y se llevó a cabo en una sola sesión por grupo, con una participación promedio de entre 10 y 15 estudiantes por grupo. Las pruebas se realizaron en el laboratorio de sistemas del campus universitario, utilizando los equipos de la sala y el ejecutable del laboratorio desarrollado.

La metodología incluyó una evaluación inicial mediante Kahoot, la exploración individual del laboratorio virtual, y una evaluación final también individual. Se observó un aumento significativo en la atención y participación de los estudiantes durante la interacción con el laboratorio, así como una mejora en el porcentaje de respuestas correctas en la encuesta posterior.

Entre los hallazgos principales, se destacó que los estudiantes lograron comprender los conceptos de manera efectiva, describiendo la experiencia como entretenida y motivadora. Además, se identificaron mejoras para futuras implementaciones, como modificar las preguntas de las pruebas pre y post aplicación del laboratorio y espaciar las

evaluaciones en días diferentes. También se corrigieron algunos bugs gracias al feedback proporcionado por los estudiantes, lo que permitió optimizar la experiencia.

Este anexo incluye las fotografías tomadas durante las pruebas piloto, que documentan el desarrollo de las actividades y la interacción de los estudiantes con el laboratorio virtual.

Figura 21. Prueba piloto explicación del laboratorio

Figura 22. Prueba piloto Aplicación cuestionario

Figura 23. Prueba piloto prueba de laboratorio

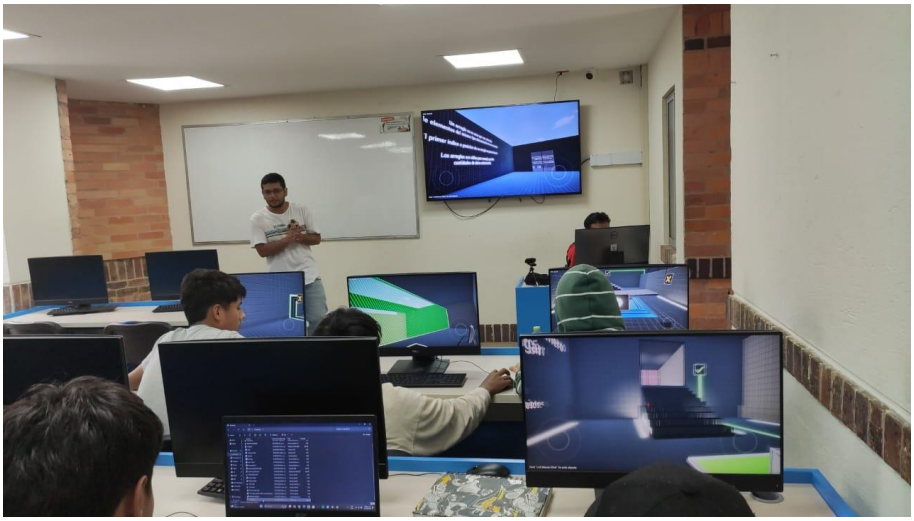
Figura 24. Prueba piloto resultados de prueba

Figura 25. Prueba piloto

Figura 26. Prueba piloto

Figura 26. Prueba piloto

Figura 26. Prueba piloto



Anexo 8. Cuestionarios aplicados a estudiantes

Preguntas aplicadas a estudiantes antes y después de la aplicación del laboratorio:

All (10) **Difficult questions (2)**

Question		Type
1	¿Qué es un arreglo?	Quiz
2	¿Cómo se accede a un elemento en un arreglo?	Quiz
3	¿Cuál es una característica importante de los arreglos?	Quiz
4	Si tienes un arreglo con cinco números: [1, 2, 3, 4, 5], ¿cuál es el valor en la tercera posición?	Quiz
5	¿Qué es una función?	Quiz
6	¿Para qué sirven los datos que recibe una función (input)?	Quiz
7	¿Qué significa el resultado (output) de una función?	Quiz
8	¿Qué sucede cuando una función no necesita devolver un resultado visible?	Quiz
9	¿Qué ventaja tiene usar un arreglo dentro de una función?	Quiz
10	Fue una buena experiencia?	True or false

Resultados Pre-Aplicación del juego :

Blocky bot experiencia de usuario

Played on	24 Oct 2024
Hosted by	AquiChagottv
Played with	15 players
Played	10 of 10

Overall Performance

Total correct answers (%)	54.67%
Total incorrect answers (%)	45.33%
Average score (points)	4444.07 points

Feedback

Number of responses	0				
How fun was it? (out of 5)	0.00 out of 5				
Did you learn something?	0.00% Yes		0.00% No		
Do you recommend it?	0.00% Yes		0.00% No		
How do you feel?	0.00% Positive		0.00% Neutral		0.00% Negative

Switch tabs/pages to view other result breakdown

Resultados Post-Aplicación del juego :

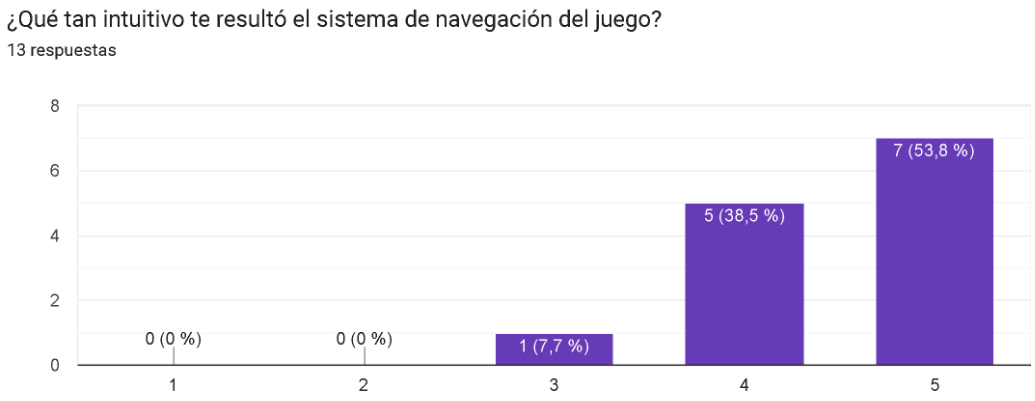
Blocky bot experiencia de usuario	
Played on	24 Oct 2024
Hosted by	AquiChagottv
Played with	16 players
Played	10 of 10

Overall Performance	
Total correct answers (%)	66.25%
Total incorrect answers (%)	33.75%
Average score (points)	5874.00 points

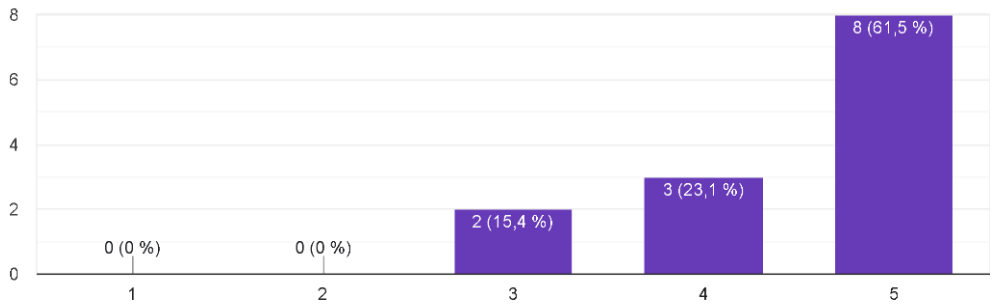
Feedback			
Number of responses	0		
How fun was it? (out of 5)	0.00 out of 5		
Did you learn something?	0.00% Yes	0.00% No	
Do you recommend it?	0.00% Yes	0.00% No	
How do you feel?	0.00% Positive	0.00% Neutral	0.00% Negative

Switch tabs/pages to view other result breakdown

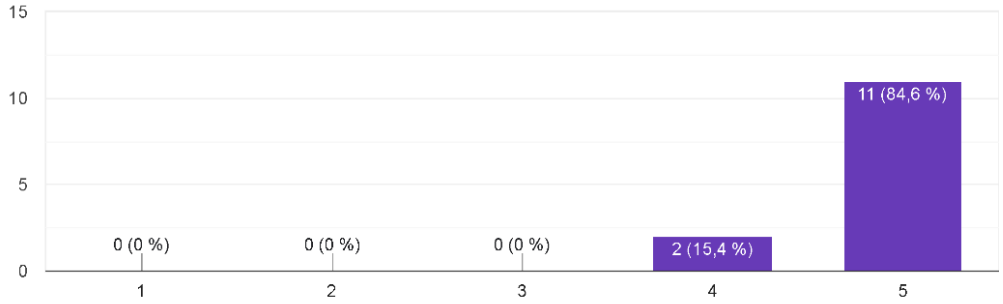
Preguntas aplicadas para medir la experiencia de usuario



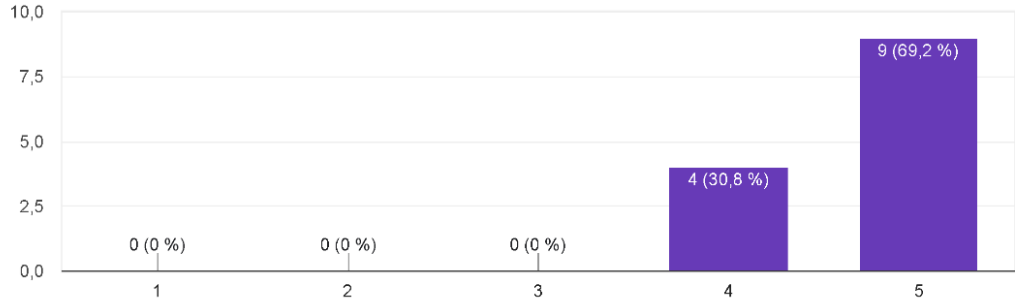
¿Qué tan clara fue la explicación de los conceptos de arreglos y funciones dentro del juego?
13 respuestas



¿Qué tan fácil te resultó interactuar con los elementos del juego?
13 respuestas

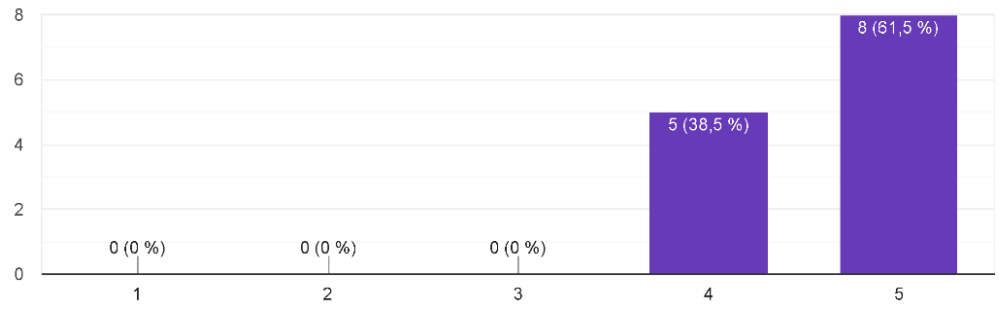


¿Qué tan bien se ajustaron las actividades del juego a tu nivel de conocimiento previo?
13 respuestas



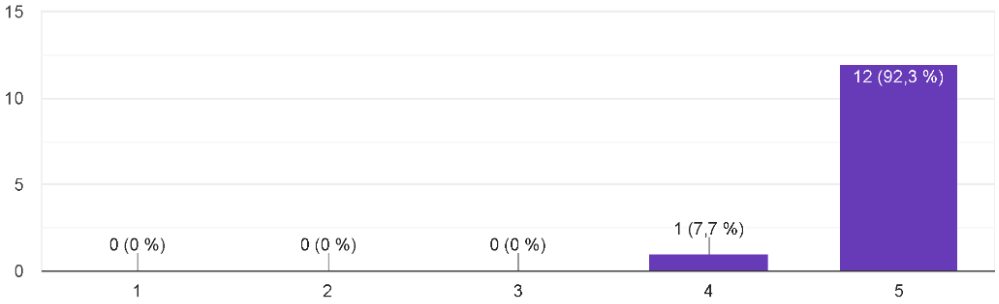
¿Qué tan motivado te sentiste para continuar jugando y aprendiendo con este juego?

13 respuestas



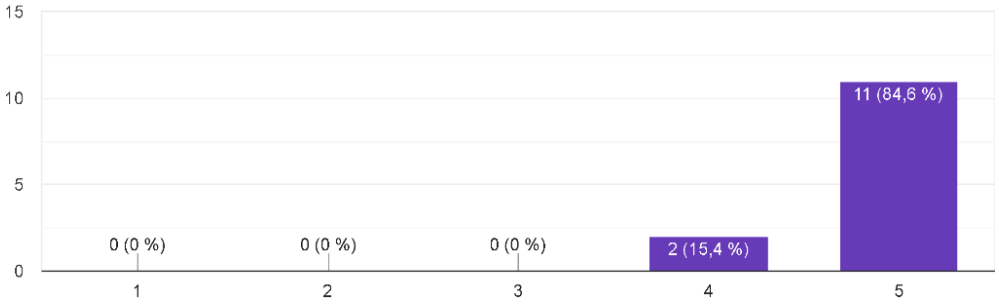
¿Qué tan rápido te acostumbraste a los controles del juego?

13 respuestas

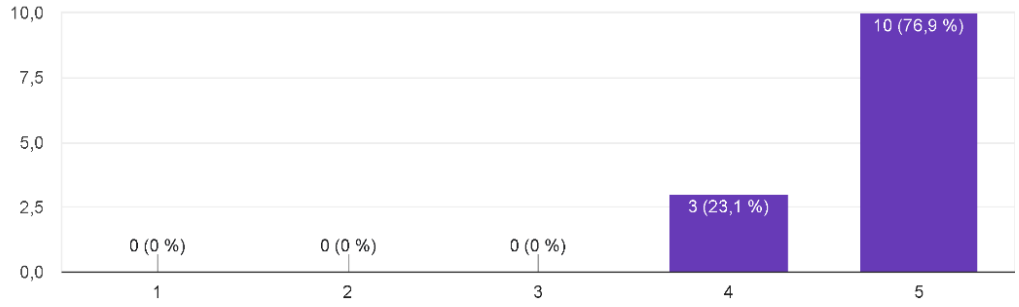


¿Qué tan claro fue el propósito de cada nivel o misión dentro del juego?

13 respuestas



¿Qué tan comprensible fue la interfaz gráfica del juego (menús, botones, gráficos)?
13 respuestas



¿Qué tan bien crees que este juego te ayudó a entender los conceptos de arreglos y funciones?
13 respuestas