

---

Desarrollo de una aplicación para recorridos virtuales mediante el uso de realidad aumentada para visitas guiadas en el Centro de Innovación y Tecnología de la Universidad del Valle de Guatemala

---

Roberto Javier Mombiela Herrera





Universidad del Valle de Guatemala  
Facultad de Ingeniería  
Licenciatura en Ingeniería en Ciencia de la Computación y Tecnologías de la  
Información

**Desarrollo de una aplicación para recorridos virtuales mediante el uso de realidad aumentada para visitas guiadas en el Centro de Innovación y Tecnología de la Universidad del Valle de Guatemala**

**Desarrollo de Modelado 3D y Algoritmo de *Pathfinding* en Aplicación de Recorridos Virtuales con *Unity***

PARA OPTAR AL GRADO DE LICENCIATURA EN:  
Ingeniería en Ciencia de la Computación y Tecnologías de la Información

EN MODALIDAD DE MEGAPROYECTO TECNOLÓGICO

Roberto Javier Mombiela Herrera 20067

Guatemala  
28 de noviembre de 2024

Vo.Bo.:

(f) \_\_\_\_\_  
Ing. Carlos Alonso

Tribunal Examinador:

(f) \_\_\_\_\_  
Ing. Carlos Alonso

(f) \_\_\_\_\_  
Ing. Dulce Chacon

(f) \_\_\_\_\_  
Ing. Moises Alonso

Fecha de aprobación: Guatemala, 28 de noviembre de 2024.

---

## Agradecimientos

---

Quiero aprovechar este espacio para agradecer profundamente a mi familia, cuyo apoyo incondicional ha sido una fuente constante de motivación en mi vida. Su comprensión y aliento me han permitido enfrentar cada desafío con determinación y fuerza.

Asimismo, quiero extender un agradecimiento especial a Paulina Schnoor, José Hernández, Pablo González y Javier Valle. Gracias por su apoyo y colaboración durante todo este proceso. Cada uno de ustedes ha contribuido de manera significativa a mi crecimiento personal y profesional, y estoy verdaderamente agradecido por tenerlos a mi lado en este viaje.

Quiero expresar mi más sincero agradecimiento a la Ingeniera Dulce Chacón y al Ingeniero Gabriel Barrientos por su dedicación y esfuerzo al impartir la clase de Megaproyecto. Su orientación constante, sus observaciones detalladas y sus valiosas correcciones a lo largo del curso fueron fundamentales para el desarrollo de este trabajo.

Agradezco de manera especial al Ingeniero Carlos Alonso, quien, como mi asesor, me brindó orientación estratégica y aportes significativos para la conceptualización y ejecución de este proyecto. Su disposición para compartir sus conocimientos y su apoyo constante en cada etapa, tanto del desarrollo del sistema como del trabajo escrito, fueron invalúables para el resultado final.

Asimismo, extiendo mi agradecimiento a todo el equipo del Megaproyecto del Tour de Realidad Aumentada de la UVG por su compromiso, esfuerzo y colaboración. Gracias por el arduo trabajo y la disposición para apoyarnos mutuamente a lo largo de este proceso. Su compañerismo y dedicación fueron esenciales para alcanzar los objetivos propuestos y superar los desafíos que enfrentamos juntos.

Este trabajo no habría sido posible sin el apoyo de todos ustedes.

---

## Índice

---

<b>Agradecimientos</b>	<b>III</b>
<b>Resumen</b>	<b>xI</b>
<b>1. Introducción</b>	<b>1</b>
<b>2. Justificación</b>	<b>2</b>
<b>3. Alcance</b>	<b>4</b>
<b>4. Objetivos</b>	<b>5</b>
4.1. Objetivo General . . . . .	5
4.2. Objetivos Específicos . . . . .	5
<b>5. Marco Teórico</b>	<b>6</b>
5.1. Conceptos Clave . . . . .	6
5.1.1. Realidad Aumentada . . . . .	6
5.1.2. Continuo de Virtualidad . . . . .	6
5.1.3. Problema de Registro . . . . .	7
5.1.4. Indoor Navigation . . . . .	7
5.1.5. Pathfinding . . . . .	8
5.1.6. Algoritmo A* . . . . .	8
5.1.7. Heurística . . . . .	8
5.1.8. Interfaz de Usuario (UI) . . . . .	8
5.1.9. Pruebas de Estres . . . . .	8
5.2. Fundamentos Teóricos . . . . .	9
5.2.1. Principios de AR . . . . .	9
5.2.2. Fundamentos del Pathfinding . . . . .	10
5.2.3. Funcionamiento del Algoritmo A* . . . . .	10
5.3. Tecnologías y Herramientas . . . . .	12
5.3.1. Blender . . . . .	12
5.3.2. Unity . . . . .	13
5.3.3. AI NavMesh . . . . .	13
5.3.4. AR Foundation . . . . .	13
5.3.5. AR Session . . . . .	13
5.3.6. XR Origin . . . . .	14
5.3.7. ARCore y ARKit . . . . .	14

5.3.8. C# . . . . .	14
<b>6. Metodología</b>	<b>15</b>
6.1. Modelado del CIT utilizando Blender . . . . .	15
6.2. Construcción de la escena dentro de Unity . . . . .	16
6.3. Desarrollo del algoritmo de <i>pathfinding</i> y estructura del código . . . . .	16
6.3.1. Implementación del algoritmo de <i>pathfinding</i> basado en A* . . . . .	16
6.3.2. Implementación del algoritmo de <i>pathfinding</i> basado en NavMesh . . . . .	17
6.3.3. Definición de rutas y objetivos . . . . .	17
6.3.4. Navegación entre niveles y escaneo de códigos QR . . . . .	17
6.3.5. Mejora de la experiencia de usuario: implementación de una flecha de navegación	18
6.4. Integración de la interfaz de usuario . . . . .	18
6.4.1. Integración de módulos adicionales . . . . .	19
6.5. Configuración de <i>plug-ins</i> necesarios . . . . .	19
6.6. Pruebas internas y pruebas de usuario . . . . .	19
6.6.1. Pruebas de los modelos 3D . . . . .	19
6.6.2. Pruebas del código y aplicación . . . . .	20
6.6.3. Pruebas de rendimiento . . . . .	20
6.6.4. Pruebas de usuario . . . . .	20
<b>7. Resultados</b>	<b>21</b>
7.1. Modelado del CIT utilizando Blender . . . . .	21
7.2. Construcción la escena dentro de Unity . . . . .	29
7.3. Desarrollo del algoritmo de <i>pathfinding</i> y estructura del código . . . . .	30
7.3.1. Algoritmo desarrollado . . . . .	31
7.3.2. Implementación exitosa del algoritmo basado en NavMesh . . . . .	31
7.3.3. Rutas y objetivos dentro del tour virtual . . . . .	32
7.3.4. Navegación entre niveles mediante códigos QR . . . . .	34
7.3.5. Flecha de navegación y experiencia del usuario . . . . .	34
7.4. Integración de la interfaz de usuario . . . . .	35
7.4.1. Integración de módulos adicionales . . . . .	37
7.5. Configuración de <i>plug-ins</i> necesarios . . . . .	38
7.6. Pruebas internas y pruebas de usuario . . . . .	38
7.6.1. Pruebas de los modelos 3D y el algoritmo <i>pathfinding</i> . . . . .	38
7.6.2. Pruebas del código y aplicación . . . . .	39
7.6.3. Pruebas de rendimiento . . . . .	40
7.6.4. Pruebas de usuario . . . . .	42
7.6.5. Pruebas de usuario segunda ronda . . . . .	46
<b>8. Discusión</b>	<b>50</b>
8.1. Modelado del CIT utilizando Blender . . . . .	50
8.2. Construcción la escena dentro de Unity . . . . .	51
8.3. Desarrollo del algoritmo de <i>pathfinding</i> y estructura del código . . . . .	52
8.4. Integración de la interfaz de usuario . . . . .	53
8.5. Configuración de plug-ins necesarios . . . . .	54
8.6. Pruebas internas y pruebas de usuario . . . . .	54
<b>9. Conclusiones</b>	<b>57</b>
<b>10. Recomendaciones</b>	<b>59</b>
<b>Bibliografía</b>	<b>63</b>
<b>Anexos</b>	<b>64</b>

<b>A. Repositorio de Github</b>	<b>65</b>
<b>B. Cuestionario de Evaluación del AR Tour</b>	<b>66</b>
<b>C. Figuras</b>	<b>68</b>
<b>D. Fragmentos de Código</b>	<b>73</b>

---

## Índice de figuras

---

5.1. Continuo de Virtualidad . . . . .	7
7.1. Clave pasillos nivel 1 . . . . .	22
7.2. Clave pasillos nivel 2 . . . . .	23
7.3. Clave pasillos nivel 3 . . . . .	24
7.4. Clave pasillos nivel 4 . . . . .	25
7.5. Clave pasillos nivel 6 . . . . .	26
7.6. Clave pasillos nivel 7 . . . . .	27
7.7. Plano original nivel 2 . . . . .	28
7.8. Contorno trazado en plano nivel 2 . . . . .	28
7.9. Modelo 3D nivel 2 . . . . .	28
7.10. Escena Completa en Unity . . . . .	29
7.11. Elementos esenciales para la AR . . . . .	30
7.12. Elementos del ambiente . . . . .	30
7.13. Comparación entre A* pathfinding (Imagen A) y Unity pathfinding (Imagen B). . . . .	32
7.14. Rutas definidas dentro de Unity . . . . .	33
7.15. Código QR elevadores principales nivel 2 . . . . .	34
7.16. Comparación entre la App utilizando línea (Imagen A) y utilizando flecha (Imagen B) como guía. . . . .	35
7.17. Flujo de la aplicación dentro del menú de inicio. Esta figura incluye las imágenes A, B, C y D, que representan diferentes etapas del proceso. . . . .	36
7.18. Flujo de la aplicación dentro del recorrido. Esta figura incluye las imágenes A, B, C y D, que representan diferentes etapas del proceso. . . . .	36
7.19. Flujo inicial dentro de la aplicación actualizada. Esta figura incluye las imágenes A, B, C y D, que representan diferentes etapas del proceso. . . . .	37
7.20. Flujo de minijuegos dentro de la aplicación actualizada. Esta figura incluye las imágenes A, B, C y D, que representan diferentes etapas del proceso. . . . .	38
7.21. Comparación del consumo entre la función original (Imagen A) y la función modificada (Imagen B). . . . .	41
7.22. Opinión de usuarios sobre la intuitividad de la aplicación . . . . .	43
7.23. Opinión de usuarios sobre la experiencia general de la aplicación . . . . .	43
7.24. Opinión de usuarios sobre la precisión de la aplicación . . . . .	44
7.25. Opinión de usuarios sobre la herramienta de navegación . . . . .	44
7.26. Opinión de usuarios sobre el uso de códigos QR . . . . .	45
7.27. Experiencia de usuarios con retrasos en la aplicación . . . . .	45
7.28. Opinión de usuarios sobre la intuitividad de la aplicación (ronda 2) . . . . .	47

7.29. Opinión de usuarios sobre la experiencia general de la aplicación (ronda 2) . . . . .	47
7.30. Opinión de usuarios sobre la precisión de la aplicación (ronda 2) . . . . .	48
7.31. Opinión de usuarios sobre la herramienta de navegación (ronda 2) . . . . .	48
7.32. Opinión de usuarios sobre el uso de códigos QR (ronda 2) . . . . .	49
7.33. Experiencia de usuarios con retrasos en la aplicación (ronda 2) . . . . .	49
 C.1. Plano original nivel 1 . . . . .	69
C.2. Contorno trazado en plano nivel 1 . . . . .	69
C.3. Modelo 3D nivel 1 . . . . .	69
C.4. Plano original nivel 2 . . . . .	69
C.5. Contorno trazado en plano nivel 2 . . . . .	69
C.6. Modelo 3D nivel 2 . . . . .	69
C.7. Plano original nivel 3 . . . . .	70
C.8. Contorno trazado en plano nivel 3 . . . . .	70
C.9. Modelo 3D nivel 3 . . . . .	70
C.10. Plano original nivel 4 . . . . .	70
C.11. Contorno trazado en plano nivel 4 . . . . .	70
C.12. Modelo 3D nivel 4 . . . . .	70
C.13. Plano original nivel 6 . . . . .	71
C.14. Contorno trazado en plano nivel 6 . . . . .	71
C.15. Modelo 3D nivel 6 . . . . .	71
C.16. Plano original nivel 7 . . . . .	71
C.17. Contorno trazado en plano nivel 7 . . . . .	71
C.18. Modelo 3D nivel 7 . . . . .	71
C.19. Código QR elevadores principales nivel 1 . . . . .	72
C.20. Código QR elevadores secundarios nivel 2 . . . . .	72
C.21. Código QR elevadores principales nivel 3 . . . . .	72
C.22. Código QR elevadores secundarios nivel 3 . . . . .	72
C.23. Código QR elevadores principales nivel 4 . . . . .	72
C.24. Código QR elevadores secundarios nivel 4 . . . . .	72
C.25. Código QR elevadores principales nivel 6 . . . . .	72
C.26. Código QR elevadores secundarios nivel 6 . . . . .	72
C.27. Código QR elevadores principales nivel 7 . . . . .	72

---

## Índice de tablas

---

7.1.	Porcentajes de error promedio por nivel . . . . .	21
7.2.	Comparación de medidas teóricas y reales nivel 1 . . . . .	22
7.3.	Comparación de medidas teóricas y reales nivel 2 . . . . .	23
7.4.	Comparación de medidas teóricas y reales nivel 3 . . . . .	24
7.5.	Comparación de medidas teóricas y reales nivel 4 . . . . .	25
7.6.	Comparación de medidas teóricas y reales nivel 6 . . . . .	26
7.7.	Comparación de medidas teóricas y reales nivel 7 . . . . .	27
7.8.	Resultados del error de medición promedio en metros para cada nivel del edificio. . . . .	39

---

## Índice de Códigos

---

7.1. Estructura del objeto Node . . . . .	31
7.2. Código que invoca a la función de A* desarrollado manualmente . . . . .	31
7.3. Código que invoca a la función de <i>pathfinding</i> integrada en Unity . . . . .	31
7.4. Estructura del objeto Route . . . . .	32
7.5. Estructura del objeto Level . . . . .	33
7.6. Estructura del objeto Target . . . . .	33
7.7. Fragmento de código que crea una nueva textura en cada momento . . . . .	41
7.8. Fragmento de código que crea una nueva textura después de eliminar la anterior . . . . .	41
D.1. Función principal del algoritmo A* . . . . .	74
D.2. Función para crear la cuadricula . . . . .	75
D.3. Función para detectar códigos QR . . . . .	76
D.4. Función para reorientar al usuario . . . . .	77
D.5. Función optimizada para detectar códigos QR . . . . .	78
D.6. Función para obtener los siguientes cruces en la ruta . . . . .	79
D.7. Función para manejar correctamente el siguiente punto . . . . .	79

---

## Resumen

---

Este trabajo presenta el desarrollo del modelado 3D y algoritmo de *pathfinding* en la aplicación de recorridos virtuales del CIT de la Universidad del Valle de Guatemala utilizando Unity. El proyecto combina los modelos tridimensionales de los niveles del CIT con la implementación del algoritmo de *pathfinding*. El desarrollo de este proyecto explica la metodología que se siguió para poder crear una aplicación, desde cero, completamente funcional y capaz de guiar a usuarios dentro del CIT a través de un tour virtual.

Para poder verificar y documentar la precisión y confiabilidad de la aplicación, se hicieron pruebas tanto de los modelos, como de la aplicación. Las pruebas destacan un problema importante, el problema del registro, el cual indica que los objetos del mundo virtual deben de estar alineados con los objetos del mundo real para evitar cualquier error dentro de la aplicación, y garantizar una experiencia de usuario satisfactoria. A pesar de los problemas que presenta la aplicación, la mayoría de los usuarios indicó tener una experiencia agradable a la hora de utilizarla, ya que es intuitiva y una gran innovación para la institución.

Este trabajo representa una versión preliminar del proyecto, que ha logrado integrar una experiencia de recorrido virtual en el CIT de la Universidad del Valle de Guatemala. Aunque la aplicación está funcional en gran medida, presenta algunas fallas, como el problema del registro, que afectan la alineación precisa entre el mundo virtual y el real. Aun así, los resultados obtenidos indican que la aplicación es innovadora e intuitiva, ofreciendo una base sólida para su desarrollo futuro y para mejorar la experiencia de los usuarios en la institución.

# CAPÍTULO 1

---

## Introducción

---

En el contexto actual de la educación y la tecnología, la implementación de herramientas innovadoras se ha convertido en un aspecto fundamental para mejorar la experiencia de aprendizaje y promover la interacción de los usuarios con su entorno [1]. En este sentido, el desarrollo de aplicaciones de recorridos virtuales mediante el uso de realidad aumentada ha emergido como una solución prometedora para enriquecer las visitas guiadas en entornos educativos [2].

El presente trabajo se enmarca en el desarrollo de una aplicación de recorridos virtuales utilizando Unity, centrada en la integración de modelado 3D y algoritmos de *pathfinding*. Este proyecto tiene como objetivo principal mejorar la experiencia de los usuarios durante las visitas guiadas en el Centro de Innovación y Tecnología de la Universidad del Valle de Guatemala.

En particular, este trabajo se enfoca en el desarrollo del modelado 3D de la universidad y la implementación de un algoritmo de *pathfinding*. La integración de estas tecnologías permite a los usuarios interactuar de manera inmersiva con el entorno del Centro de Innovación y Tecnología, brindándoles una experiencia educativa, enriquecedora y personalizada.

Para lograr este objetivo, se llevó a cabo un proceso de investigación para seleccionar las tecnologías de modelado 3D y *pathfinding* más adecuadas, considerando la compatibilidad con la plataforma Unity y las necesidades específicas del proyecto. Además, se desarrolló un algoritmo de *pathfinding* utilizando Unity AR Foundation, y se integrará con los *plugins* ARKit y ARCore para asegurar su funcionalidad en dispositivos iOS y Android.

Finalmente, se realizaron pruebas exhaustivas del sistema desarrollado para garantizar su fiabilidad y eficacia en entornos reales, y se documentará el proceso de diseño, implementación y pruebas del sistema para su mantenimiento y futuras mejoras.

## CAPÍTULO 2

---

### Justificación

---

El desarrollo de una aplicación de recorridos virtuales mediante el uso de realidad aumentada para dispositivos móviles surge como una respuesta a la necesidad de mejorar la experiencia de los visitantes en el Centro de Innovación y Tecnología (CIT) de la Universidad del Valle de Guatemala (UVG). Esta iniciativa encuentra su motivación en diversas razones que hacen imperativa su realización.

Uno de los principales impulsos que motivaron este proyecto surge de la experiencia vivida por un estudiante al presenciar las dificultades que enfrentan los visitantes al navegar por el CIT. Esta situación se evidenció cuando su padre se perdió durante las distinciones académicas del año pasado. Esta anécdota pone de manifiesto la necesidad de contar con herramientas que faciliten la orientación y la navegación dentro de este complejo educativo.

Además de brindar una solución práctica para este problema, la aplicación de recorridos virtuales con realidad aumentada tiene el potencial de transformar por completo la experiencia de los visitantes. Al proporcionar una guía interactiva y personalizada, esta tecnología no solo facilitará la navegación por el Centro de Innovación y Tecnología, sino que también permitirá a los usuarios explorar de manera más profunda los recursos y servicios que ofrece este espacio [3].

Asimismo, esta iniciativa no solo está dirigida a visitantes externos, sino también a la comunidad estudiantil de la UVG. La aplicación de recorridos virtuales con realidad aumentada puede convertirse en una herramienta educativa innovadora, permitiendo a los estudiantes acceder a contenido multimedia y actividades de aprendizaje contextualizadas dentro del CIT.

Por otro lado, se destaca la importancia de utilizar Unity como una solución para este problema. El uso de Unity y la implementación de AR Foundation, con soporte para ARKit y ARCore, permite abarcar tanto a usuarios de iOS como de Android [4]. Esto asegura que un mayor número de personas pueda acceder y beneficiarse de la guía virtual ofrecida por la aplicación, ampliando así su alcance y utilidad para todos los visitantes.

En particular, la implementación de la funcionalidad de realidad aumentada ofrece una oportunidad única para enriquecer la experiencia de los usuarios. La realidad aumentada permite superponer información digital sobre el entorno físico, lo que facilita la navegación y la comprensión del espacio del Centro de Innovación y Tecnología. Esta función no solo proporciona una guía visual interactiva y personalizada, sino que también añade un elemento innovador y atractivo a la experiencia de los visitantes, lo que puede aumentar su interés y participación en el recorrido [5].

De esta manera, la inclusión de la realidad aumentada en la aplicación no solo garantiza un mayor alcance y accesibilidad para los usuarios, sino que también mejora significativamente la calidad y la profundidad de la experiencia de visita en el Centro de Innovación y Tecnología.

Por lo tanto, el desarrollo de esta aplicación responde a la necesidad de mejorar la experiencia de los visitantes en el CIT, así como de proporcionar una herramienta educativa innovadora para la comunidad estudiantil. Mediante el uso de la realidad aumentada, se busca no solo facilitar la navegación, sino también enriquecer el conocimiento y la interacción con este importante espacio educativo.

## CAPÍTULO 3

---

### Alcance

---

Este proyecto se enfoca en poder desarrollar una aplicación de realidad aumentada que sea capaz de proporcionar un tour virtual dentro del edificio CIT. Debido a que esta es la primera versión del proyecto, se decidió utilizar tecnologías accesibles y económicamente viables. Por lo tanto, para esta versión se optó por utilizar modelos tridimensionales del CIT en lugar de utilizar tecnologías como sensores de localización. Los seis modelos 3D de cada uno de los niveles que formar parte del tour están basados en los planos que fueron proporcionados por la universidad.

La elección de trabajar con modelos 3D, en lugar de utilizar sensores de localización, responde a la necesidad de mantener los costos bajos sin comprometer la calidad del proyecto. Este enfoque permite una experiencia inmersiva para los usuarios, cumpliendo con los objetivos planteados. Aunque este enfoque es propenso a errores y limita la interacción en tiempo real con el entorno físico, cumple con el objetivo de poder demostrar que un tour de realidad aumentada es posible dentro del CIT. Esto abre las puertas a la posibilidad de poder desarrollar futuras versiones, aunque más costosas, más precisas, permitiendo una aplicación completamente en tiempo real y sin errores, resultando en una experiencia satisfactoria para el usuario.

# CAPÍTULO 4

---

## Objetivos

---

### 4.1. Objetivo General

Garantizar la congruencia y alineación precisa entre el mundo virtual y el real en la Aplicación de Recorridos Virtuales del Centro de Innovación y Tecnología (CIT) de la Universidad del Valle de Guatemala mediante modelado 3D y herramientas de navegación en Unity.

### 4.2. Objetivos Específicos

- Mapear y modelar en 3D el CIT, asegurando una representación precisa y detallada del entorno.
- Utilizar el modelo en Unity para agregar puntos de inicio y establecer rutas completas y detalladas para los recorridos virtuales.
- Desarrollar un algoritmo de *pathfinding* basado en el modelo y las rutas establecidas, que permita guiar a los usuarios de manera eficiente y precisa a través del CIT.
- Integrar la interfaz de usuario para poder desplegar información relevante durante los recorridos.
- Configurar los plug-ins necesarios, en Unity, para que la aplicación sea compatible tanto en Android como en iOS.
- Realizar pruebas del modelo 3D, el algoritmo de *pathfinding* y la integración de realidad aumentada en entornos reales para asegurar la fiabilidad y eficacia de la aplicación.

# CAPÍTULO 5

---

Marco Teórico

---

## 5.1. Conceptos Clave

En esta sección se presentan los conceptos fundamentales relacionados con el proyecto, incluyendo la realidad aumentada, la navegación interior, el *pathfinding*, el algoritmo A\*, heurísticas y el problema de registro. Estos conceptos proporcionan una base teórica para comprender el contexto en el que se desarrolla la aplicación.

### 5.1.1. Realidad Aumentada

La realidad aumentada (AR por sus siglas en inglés) es una variante de la realidad virtual, la cual sumerge al usuario por completo en una realidad sintética. A diferencia de la realidad virtual, la cual no permite que el usuario pueda ver el mundo real a su alrededor, la AR sí lo permite. Lo especial de la AR es que superpone objetos e información virtual encima de la realidad. Por lo tanto, la AR combina el mundo real con el virtual, a diferencia de la realidad virtual, la cual reemplaza el mundo real por completo con uno virtual. [6]

En el campo del turismo y las visitas guiadas, la AR puede ser una herramienta valiosa, debido a que utilizando un dispositivo móvil, los visitantes pueden obtener información adicional a la que se presenta en vivo y de una manera más interactiva e interesante. Las características únicas de los dispositivos móviles, como flexibilidad, facilidad de uso y personalización, los hacen útiles para tanto proveedores como consumidores turísticos [7]. Por lo tanto, la AR junto con la *indoor navigation* pueden combinarse para poder crear experiencias interactivas en el mundo de los tours guiados.

### 5.1.2. Continuo de Virtualidad

El Continuo de Virtualidad describe el rango de posibilidades entre el mundo físico o real y el mundo completamente digital o virtual. Este concepto, propuesto por Milgram y Kishino en 1994, se enfoca en cómo se combinan elementos reales y digitales en distintos entornos. La realidad mixta cubre la mayor parte del continuo, donde lo real y lo virtual se fusionan en una sola visualización [8].

Como se puede observar en la Figura 5.1, el continuo se divide en cuatro categorías:

1. **Entorno real:** Solo objetos físicos.
2. **Realidad aumentada:** El entorno real se complementa con elementos digitales.
3. **Virtualidad aumentada:** El entorno virtual incluye objetos físicos.
4. **Entorno virtual:** Solo objetos digitales.

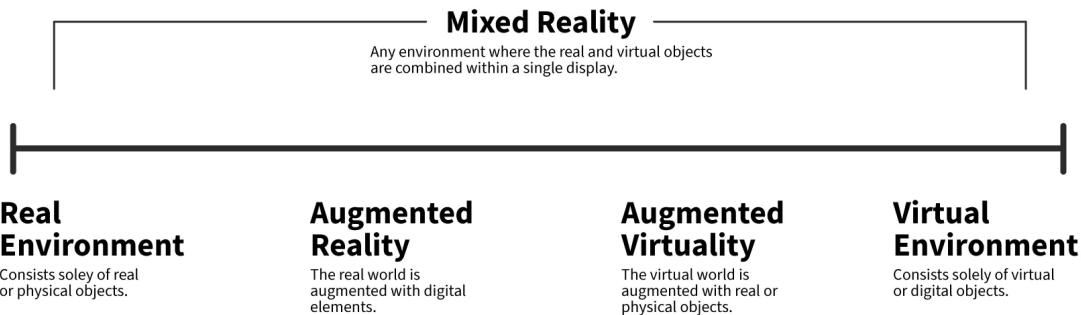


Figura 5.1: Continuo de Virtualidad [8]

### 5.1.3. Problema de Registro

El problema de registro es uno de los problemas fundamentales que limita las aplicaciones de AR. En la AR, los objetos virtuales y los del mundo real deben de estar precisamente alienados para poder mantener la ilusión de que ambos mundos coexisten [9]. La precisión en el registro es crucial para aplicaciones interactivas donde los usuarios interactúan con ambos mundos simultáneamente.

En proyectos como la creación de tours en AR, por ejemplo, cualquier error de registro puede romper la inmersión y afectar negativamente la experiencia del usuario. A diferencia de los Entornos Virtuales (VE por sus siglas en inglés), donde los errores son menos evidentes y pueden causar conflictos visuales o sensoriales menos notables, en AR, incluso pequeños desajustes son fácilmente detectables debido a la alta resolución del sistema visual humano, requiriendo una precisión angular extremadamente alta para asegurar una experiencia de usuario satisfactoria [9].

### 5.1.4. Indoor Navigation

La navegación interior o *indoor navigation* se refiere a la tecnología y métodos utilizados para determinar la posición de personas y objetos dentro de edificios, donde el uso de GPS no es viable. Esta tecnología puede ser utilizada en cualquier entorno cerrado, en este caso, universidades. Entre las tecnologías posibles para implementar la navegación interior se incluyen sensores de proximidad, WiFi y Bluetooth [10]. Adicionalmente, herramientas de modelado 3D, pueden modelar los interiores de los edificios para crear representaciones precisas que faciliten la navegación [11].

La navegación interior permite que los usuarios puedan navegar dentro de los edificios, ya que al saber la ubicación del usuario y del objetivo, el sistema utiliza dichos parámetros para poder crear una ruta, utilizando un algoritmo de *pathfinding*.

### 5.1.5. Pathfinding

Los problemas de búsqueda o el *pathfinding* se refieren al proceso de encontrar el camino más eficiente entre dos puntos en un espacio determinado [12]. El objetivo es poder encontrar una ruta, evitando obstáculos y minimizando costos. Los costos que se buscan minimizar son: tiempo, distancia, precio y en algunos casos, gasolina[13]. En este contexto, el *pathfinding*, tiene como objetivo, poder guiar a los usuarios del CIT y poder crear las rutas que estos navegaron.

### 5.1.6. Algoritmo A\*

El algoritmo A\* es un algoritmo que se puede utilizar para resolver problemas de *pathfinding*. El algoritmo A\* fue desarrollado en 1986 y gracias a su eficiencia, es comúnmente utilizado en muchos campos de la ciencia de la computación, como: videojuegos, mapas y *pathfinding* [14].

Este algoritmo es una variante avanzada de la búsqueda *best-first search* que utiliza heurísticas para evaluar los caminos disponibles. Al combinar la eficiencia de la búsqueda *greedy* y la precisión del algoritmo de Dijkstra, A\* es tanto completo como óptimo, asegurando siempre encontrar la mejor solución posible [14]. En el contexto de este proyecto, A\* ha sido seleccionado como el algoritmo de *pathfinding* para garantizar que la navegación sea lo más eficiente posible.

### 5.1.7. Heurística

Una heurística es un atajo o una estrategia cognitiva que permite solucionar un problema, tomar una decisión o emitir un juicio de manera eficiente[15]. En términos de ciencias de computación, una heurística es una función que permite aproximar la solución de un problema, optimizando el tiempo de cómputo y los recursos. En el contexto del *pathfinding*, las heurísticas guían el algoritmo hacia la solución al estimar el costo restante para llegar al objetivo, lo que permite reducir la cantidad de nodos explorados y mejorar la velocidad de la búsqueda [16].

### 5.1.8. Interfaz de Usuario (UI)

La interfaz de usuario (UI por sus siglas en inglés) se refiere a los elementos con los que los usuarios interactúan directamente en productos o servicios digitales, como páginas web o aplicaciones. Este diseño abarca aspectos visuales como colores, tipografías, iconos, botones y animaciones. Aunque la UI se enfoca en lo que es visible, su objetivo principal es ser funcional, además de estético [17]. En el contexto de este proyecto, la UI fue clave para integrar la interacción entre el usuario y la aplicación, y para lograr un diseño visualmente atractivo que facilitara una experiencia más intuitiva y agradable para los usuarios.

### 5.1.9. Pruebas de Estres

Las pruebas de estrés, o de resistencia, son una técnica utilizada en el proceso de testing de software para evaluar los límites de un sistema bajo condiciones extremas de carga. Durante estas pruebas, se simulan múltiples peticiones que exceden los niveles normales de uso, con el objetivo de identificar posibles fallos, como errores en el código o bloqueos de información. Un aspecto crucial de estas pruebas es verificar que el sistema pueda recuperarse y volver a funcionar adecuadamente tras una falla [18].

## 5.2. Fundamentos Teóricos

En esta sección se explican las teorías y principios que sustentan el proyecto, abordando aspectos como los fundamentos del *pathfinding*, los principios detrás de la realidad aumentada y el funcionamiento del algoritmo A\*. Estos fundamentos teóricos son esenciales para comprender cómo se implementan y aplican las tecnologías en el contexto del proyecto.

### 5.2.1. Principios de AR

Aplicaciones de AR aumentan el mundo real con información digital. Como se puede observar en la figura 5.1, la AR se encuentra cerca del lado real del continuo de virtualidad, en donde los entornos reales se encuentran en un extremo y los entornos virtuales en el otro [19]. Por lo tanto, la AR está dentro de un contexto más general denominado, Realidad Mixta, el cual se refiere a un espectro multi-eje de áreas que cubren la AR, realidad virtual, telepresencia, entre otras [9]. El resultado, mejorar la realidad que normalmente vemos, en lugar de reemplazarla. Esta tecnología toma el entorno real del mundo presente y proyecta o muestra imágenes y sonidos digitales en él [20].

#### Componentes

- **Generador de Escenas:** Es el dispositivo que se encarga de renderizar la escena para el usuario. Actualmente, este no es de los problemas principales en la AR, debido a que la cantidad de objetos y elementos que se tienen que renderizar es baja [9].
- **Sistema de Seguimiento:** Para asegurar un funcionamiento óptimo de las aplicaciones de AR, es crucial que los objetos reales y virtuales estén alineados correctamente. El sistema de seguimiento, por lo tanto, es uno de los problemas más importantes en la AR, más que nada por el *registration problem*. Por ende, el sistema de seguimiento debe garantizar que la superposición de información digital sobre el entorno real sea precisa para mantener la ilusión de coexistencia de ambos mundos [9].
- **Unidad de Visualización:** La unidad de visualización es cualquier dispositivo que sea capaz de renderizar los objetos virtuales a través de su cámara [21]. Algunos dispositivos populares y más avanzados son los lentes especiales. Pero la AR no se limita a solamente utilizar estos dispositivos, hoy en día, los celulares inteligentes más avanzados son capaces de soportar y correr aplicaciones de AR.

#### Tipos

1. **Basada en marcadores:** Este tipo de aplicaciones utilizan un marcador, como una imagen o un objeto 3D, que activa la experiencia aumentada cuando se escanea con una cámara o un visor AR. Esta capa digital puede incluir texto, imágenes, videos, sonido, modelos 3D y animaciones [20].
2. **Sin marcadores:** Aplicaciones de AR sin marcadores, como su nombre lo indica, no necesitan marcadores para activar las aumentaciones. En su lugar, el contenido digital se superpone en superficies físicas seleccionadas por el usuario bajo demanda, como suelos o mesas [19].
3. **Basada en ubicación:** Las aplicaciones de AR basadas en ubicación, las cuales son el foco de este proyecto, presentan contenido digital a los usuarios cuando llegan a ubicaciones específicas en el mundo físico. Este tipo de AR rastrea la posición del usuario y proporciona información relevante en función de su ubicación [19].

### 5.2.2. Fundamentos del Pathfinding

El *pathfinding* se basa en principios matemáticos y computacionales para encontrar la ruta más eficiente en un entorno. Un problema de búsqueda se puede definir formalmente por varios componentes clave:

- **Espacio de Estados:** Conjunto de posibles estados del entorno.
- **Estado Inicial:** Punto de inicio del agente o usuario.
- **Estados Objetivos:** Destinos a los que el agente desea llegar.
- **Acciones Disponibles:** Movimientos posibles del agente desde un estado dado.
- **Modelo de Transición:** Resultado de cada acción.
- **Función de Costo de Acción:** Costo numérico de aplicar una acción para llegar a otro estado.

Una secuencia de acciones forma un camino, y una solución es un camino desde el estado inicial hasta un estado objetivo. Los costos de acción son aditivos, y una solución óptima tiene el costo de camino más bajo [12].

#### Algoritmos *Pathfinding*

- **Algoritmos de Búsqueda Ciega:** Los algoritmos de búsqueda ciega exploran el espacio de estados sin utilizar información adicional sobre el objetivo, basándose solo en la estructura del problema [12].
  - **Búsqueda en Anchura (Breadth-First Search, BFS):** Explora sistemáticamente todos los estados en un nivel antes de pasar al siguiente nivel. Es completa y óptima para problemas con costos de acción uniformes.
  - **Búsqueda en Profundidad (Depth-First Search, DFS):** Explora lo más profundo posible a lo largo de cada rama antes de retroceder. No es completa ni óptima, pero puede ser más eficiente en términos de memoria.
- **Algoritmos de Búsqueda Informada:** Los algoritmos de búsqueda informada utilizan información adicional, generalmente heurística, para dirigir la búsqueda de manera más eficiente hacia el objetivo [12].
  - **Búsqueda Greedy Best-First:** Utiliza solo la función heurística para guiar la búsqueda, sin considerar el costo acumulado. No es completa ni óptima, pero puede ser más rápida en encontrar soluciones en ciertos casos.
  - **A\* Search:** Es una mejora del *Best-First Search* que utiliza una función heurística para guiar la búsqueda hacia el objetivo, combinando el costo acumulado y una estimación del costo restante. Es completa y óptima si la heurística es admisible.

### 5.2.3. Funcionamiento del Algoritmo A\*

El algoritmo A\* es un algoritmo de búsqueda informada que combina los enfoques del algoritmo greedy best-first search y el algoritmo Dijkstra, utilizando heurísticas para mejorar la eficiencia del proceso. El objetivo de este algoritmo es poder encontrar el camino más corto entre un punto inicial y un objetivo en un grafo [14]. A\* logra esto mediante la evaluación de nodos con una función de coste estimada que combina dos componentes principales: el costo real desde el nodo inicial hasta el nodo actual y una estimación del costo restante hasta el nodo objetivo [22].

## Fórmula

La función de evaluación que utiliza el algoritmo A\* se representa como [12]:

$$f(n) = g(n) + h(n) \quad (5.1)$$

Donde:

- $n$ : Es el nodo actual.
- $g(n)$ : Es el costo real del camino, desde el nodo inicial hasta el nodo actual  $n$ .
- $h(n)$ : Es el costo estimado del camino más corto desde  $n$  hasta el objetivo. Este costo es estimado por una función heurística y dicha función afecta directamente la eficiencia y la precisión del algoritmo. La heurística debe ser admisible, es decir, nunca debe de sobreestimar el costo real restante.
- $f(n)$ : Es el costo del camino más eficiente que continúa desde  $n$  hasta el objetivo.

La función  $f(n)$  determina el orden en el cual se exploran los nodos. El nodo con el valor de  $f(n)$  más bajo es el siguiente en ser explorado, ya que se espera que este ofrezca el camino de menor costo hacia el objetivo [14].

## Uso de Heurísticas

Una heurística es una estimación del costo restante  $h(n)$  para llegar al objetivo y su función es guiar al algoritmo hacia la meta de manera más eficiente, minimizando la exploración de nodos innecesarios. Existen diferentes heurísticas que se pueden utilizar según el tipo de entorno [23]:

- **Distancia Manhattan:** En cuadrículas donde solo se permiten movimientos horizontales y verticales.

$$h(n) = \text{abs}(n.x - \text{objetivo}.x) + \text{abs}(n.y - \text{objetivo}.y) \quad (5.2)$$

- **Distancia Diagonal:** En cuadrículas que permiten movimientos en ocho direcciones.

$$\begin{aligned} dx &= \text{abs}(n.x - \text{objetivo}.x) \\ dy &= \text{abs}(n.y - \text{objetivo}.y) \\ h(n) &= D \cdot (dx + dy) + (D2 - 2D) \cdot \min(dx, dy) \end{aligned} \quad (5.3)$$

Donde:

- $D$ : Es el tamaño de cada nodo
- $D2$ : Es la distancia diagonal entre nodos, usualmente  $\sqrt{D}$

- **Distancia Euclídea:** Cuando se permiten movimientos en cualquier dirección en un espacio continuo. Básicamente, la distancia entre el nodo actual y el objetivo, la cual se obtiene con la fórmula de distancia.

$$h(n) = \sqrt{(n.x - \text{objetivo}.x)^2 + (n.y - \text{objetivo}.y)^2} \quad (5.4)$$

Si la heurística no sobreestima el costo, es decir, que es admisible, A\* garantiza el camino más corto. En el contexto de este proyecto, el código desarrollado utiliza la distancia Manhattan, por otro lado, el algoritmo proporcionado de Unity, no menciona públicamente qué especificaciones utiliza, pero debido al funcionamiento y a que permite movimientos en cualquier dirección, se supone que utiliza el Algoritmo A\* con distancia Euclídea.

### Pseudocódigo

El algoritmo A\* expande los nodos en función de su costo total estimado, combinando el costo desde el inicio con una heurística que estima el costo restante hasta el objetivo [14].

---

**Algorithm 1** Pseudocódigo Algoritmo A\* [22].

---

```

1: procedure A*( $G, start, goal$ )
2:   Entrada: Grafo  $G = (V, E)$ , nodo inicial  $start$ , nodo objetivo  $goal$ .
3:   Salida: El camino más corto desde  $start$  hasta  $goal$ , si existe.
4:    $openSet \leftarrow \text{PriorityQueue}()$ 
5:    $cameFrom \leftarrow \text{Map}()$ 
6:    $gScore \leftarrow \text{Map}()$  con valor predeterminado  $\infty$ 
7:    $fScore \leftarrow \text{Map}()$  con valor predeterminado  $\infty$ 
8:    $gScore[start] \leftarrow 0$ 
9:    $fScore[start] \leftarrow h(start)$                                  $\triangleright$  Estimación del costo total
10:   $openSet.insert(start, fScore[start])$ 
11:  while  $openSet$  no esté vacío do
12:     $current \leftarrow openSet.removeMin()$ 
13:    if  $current = goal$  then                                      $\triangleright$  Si alcanzamos el objetivo
14:      return RECONSTRUIRCAMINO( $cameFrom, current$ )
15:    end if
16:    for each  $neighbor$  de  $current$  do
17:       $tentative\_gScore \leftarrow gScore[current] + w(current, neighbor)$ 
18:      if  $tentative\_gScore < gScore[neighbor]$  then     $\triangleright$  Si se encuentra un mejor camino
19:         $cameFrom[neighbor] \leftarrow current$ 
20:         $gScore[neighbor] \leftarrow tentative\_gScore$ 
21:         $fScore[neighbor] \leftarrow gScore[neighbor] + h(neighbor)$            $\triangleright f = g + h$ 
22:        if  $neighbor$  no está en  $openSet$  then
23:           $openSet.insert(neighbor, fScore[neighbor])$ 
24:        end if
25:      end if
26:    end for
27:  end while
28:  return "No se encontró un camino"
29: end procedure

```

---

## 5.3. Tecnologías y Herramientas

Esta sección detalla las tecnologías y herramientas que se utilizan en el proyecto, proporcionando una descripción más profunda de software de modelado 3D, motores de juegos y *frameworks*. La comprensión de estas herramientas es crucial para el desarrollo y la implementación exitosa del proyecto.

### 5.3.1. Blender

Blender es un programa gratuito dedicado a la creación 3D. Este programa soporta todos los componentes del proceso 3D: modelados, montajes, simulaciones, animaciones, renderizados y hasta edición y creación de videojuegos [24]. Su tecnología avanzada e interfaz permiten a los desarrolladores crear modelos 3D detallados y precisos, lo cual es crucial para proyectos que requieren

representaciones visuales y realistas, ya que se tiene el control total del modelo, desde los colores y materiales hasta las dimensiones.

### 5.3.2. Unity

Unity es un motor de juegos multiplataforma. Hoy en día es la plataforma líder para crear y operar contenido 3D en tiempo real. Desde su lanzamiento, más de 1.3 millones de usuarios han utilizado la versión más reciente, hasta la fecha, de la plataforma (Unity 2022 LTS) [25].

La plataforma permite que los desarrolladores puedan crear contenido de una manera fácil y eficiente, ya que esta proporciona las funciones integrales más importantes que hacen a un juego funcionar. Algunas de estas funciones incluyen: renderizado 3D, detección de colisiones y física [26]. Por lo tanto, no es necesario que desarrolladores tengan que crear todo desde cero por sí mismos, ya que pueden utilizar dichas funciones como apoyo y de este modo ahorrarse tiempo y esfuerzo.

Adicionalmente, Unity también cuenta con un administrador de paquetes o *package manager*. Un paquete es un contenedor que almacena diversas características o activos, como herramientas y bibliotecas del editor, herramientas de tiempo de ejecución, colecciones de activos y plantillas de proyectos [27]. Esto facilita el proceso de desarrollo para los creadores, ya que pueden aprovechar estas herramientas y recursos según sus necesidades. Específicamente, en este proyecto se utilizan dos paquetes de dicho administrador, estos son AR Foundation y AI Navmesh, los que proporcionan herramientas de RA y de navegación interior.

### 5.3.3. AI NavMesh

El AI Navmesh es un paquete de Unity que define áreas navegables y no navegables dentro de una escena [28]. En este contexto, se utilizó el NavMesh para poder desarrollar un algoritmo de *pathfinding* que guiara a los usuarios a través del modelado 3D del CIT, permitiendo que los usuarios reciban indicaciones precisas mientras se desplazan.

### 5.3.4. AR Foundation

AR Foundation es un marco de trabajo de Unity creado específicamente para el desarrollo de AR. Este marco de trabajo permite que los desarrolladores puedan agregar componentes correspondientes a su proyecto, dependiendo de las necesidades del proyecto y de las características de AR que se deseen implementar. La ventaja de utilizar AR Foundation es que los proyectos creados, se pueden implementar en diferentes dispositivos AR móviles, ya que este marco de trabajo incluye funciones centrales de ARKit y ARCore. Al mezclar dichas funciones con las funciones únicas de Unity, esto permite crear aplicaciones robustas y publicarlas en cualquier tienda de aplicaciones [29].

### 5.3.5. AR Session

La sesión de AR o el AR Session es un GameObject fundamental en Unity para habilitar la realidad aumentada en las plataformas de destino. Este objeto se encarga de iniciar y gestionar la sesión de AR, permitiendo que las funcionalidades de AR estén activas en la aplicación [30]. En el contexto de este proyecto, el AR Session se utiliza para garantizar que el sistema de realidad aumentada funcione correctamente en la plataforma móvil y permita la interacción de los usuarios con los elementos de AR dentro del tour virtual.

### 5.3.6. XR Origin

El XR Origin es otro GameObject clave en los proyectos de realidad aumentada, ya que permite el seguimiento del dispositivo y transforma los objetos detectados (trackables) en el sistema de coordenadas de Unity. Este objeto es esencial para que el dispositivo rastree con precisión los movimientos y posiciones en el espacio real [30]. En el contexto de este proyecto, el XR Origin se utiliza para asegurar que los usuarios puedan moverse de manera fluida y precisa a través del tour virtual, con un seguimiento exacto de su posición dentro del entorno de realidad aumentada.

### 5.3.7. ARCore y ARKit

ARCore y ARKit son marcos de trabajo desarrollados por Google y Apple respectivamente, que permiten a crear experiencias de realidad aumentada [31]. Ambas tecnologías pueden ser integradas en Unity a través de AR Foundation, lo que facilita la creación de aplicaciones AR tanto en dispositivos Android como IOS.

### 5.3.8. C#

C# (pronunciado C-Sharp) es un lenguaje de programación orientado a objetos desarrollado por Microsoft para su plataforma .NET. Con raíces en la familia de lenguajes C, tiene muchas similitudes con otros lenguajes ampliamente utilizados como C++ y Java. Desde su lanzamiento en 2002, C# ha evolucionado y se ha convertido en una opción preferida para el desarrollo de aplicaciones móviles, de escritorio, sitios web, servicios web, videojuegos, realidad virtual y muchas otras áreas [32].

En este proyecto, C# se utiliza dentro del entorno de Unity para implementar las funcionalidades clave del tour virtual. Unity ha adoptado C# debido a su facilidad de uso y porque, a través de un compilador JIT (compilador en tiempo de ejecución), puede traducir el código a un formato nativo eficiente [33]. Esto hace que C# sea fundamental para manejar las interacciones de los usuarios y los elementos de realidad aumentada en el tour del CIT.

# CAPÍTULO 6

---

## Metodología

---

Para el desarrollo de este proyecto, se empleó una metodología sistemática y estructurada, abordando cada uno de los objetivos específicos definidos previamente, asegurando así la consecución de los resultados esperados.

### 6.1. Modelado del CIT utilizando Blender

El primer paso en el desarrollo del proyecto del tour de realidad aumentada (AR por sus siglas en inglés) fue el mapeo y el modelado 3D del Centro de Innovación y Tecnología (CIT). Este paso fue fundamental para el proyecto, ya que los modelos resultantes fueron la base sobre la cual se trabajó el resto del proyecto. Esta fase implicó el uso de Blender para crear un modelo tridimensional detallado y preciso de cada uno de los niveles que formaron parte del tour.

Para poder crear dichos modelos, se trabajó con los planos proporcionados por la universidad, los cuales contenían toda la información relevante del CIT: las dimensiones de los pasillos, el contorno del nivel, caminos y otros elementos importantes. Para poder garantizar que el modelo fuera preciso, se llevó un proceso adicional, el cual consistió en tomar medidas, manualmente, de todos los pasillos del CIT, lo que permitía verificar y corregir (de ser necesario) las medidas de cada modelo.

Para completar el modelado inicial en Blender, se siguió un flujo de trabajo bien definido. En este proceso, algunos recursos en línea sirvieron de referencia para optimizar el flujo de trabajo y asegurar precisión en el ajuste de las dimensiones y el trazado de las paredes [34] [35]. Este flujo de trabajo consistió en los siguientes pasos:

1. Importación y Redimensionamiento de Imágenes: Se importaron los planos arquitectónicos del CIT en formato PNG a Blender. A continuación, se ajustaron las dimensiones de las imágenes para que coincidieran con las medidas reales de cada nivel en el edificio, asegurando precisión en el modelado tridimensional.
2. Trazado de Estructuras en 2D: Una vez ajustadas las dimensiones, se trazaron las paredes siguiendo los contornos de cada nivel en dos dimensiones. Este proceso permitió construir las estructuras base necesarias para levantar el modelo en 3D.

Al tener todas las mediciones manuales y todos los contornos listos, se corrigió cualquier desvia-

ción de las dimensiones originales del plano. Esto permitió asegurar que los contornos fueran lo más precisos posible. Finalmente, para poder obtener los modelos en 3D, se elevaron los contornos en la dimensión Z, lo cual resultó en los modelos finales para utilizar dentro de la aplicación.

## 6.2. Construcción de la escena dentro de Unity

Al finalizar el modelo 3D de los seis niveles, se importaron los modelos en Unity para poder crear la escena completa del tour. Crear la escena dentro de Unity implicó la instalación del marco de trabajo de Unity llamado AR Foundation y la integración de la AR Session y el XR Origin. Este paso también consistió en definir el ambiente del tour: definir las rutas disponibles, con sus niveles correspondientes, los cuales a su vez consistían en sus objetivos y códigos QR correspondientes. Finalmente, la implementación del NavMesh, permitió definir las áreas navegables dentro del tour.

Para completar este proceso, se creó un proyecto de realidad aumentada en Unity, configurando el marco de trabajo de AR Foundation, que permite la implementación de capacidades de realidad aumentada. Como parte de esta configuración, se reemplazó la cámara principal por el XR Origin y se integró el AR Session, elementos necesarios para habilitar la interacción del usuario con el entorno de realidad aumentada. Luego, se añadió el NavMesh, lo que permitió definir las áreas navegables dentro de los modelos tridimensionales. Este flujo de trabajo se diseñó tomando como referencia material de apoyo en línea que sirvió como guía básica para la implementación del proyecto de realidad aumentada [36].

Al tener la base del proyecto lista, se realizaron una serie de pasos adicionales, los cuales garantizaban que la aplicación se acoplara a las necesidades del proyecto. Primero se creó el ambiente dentro de la escena, el cual contiene todos los elementos que forman parte de la aplicación. El ambiente se ordenó de una manera para que cada nivel contenga su modelo, la posición de sus objetivos y la posición de sus códigos QR. Finalmente, se definieron las rutas disponibles en el tour; asignándole sus respectivos niveles a cada una y seleccionando los objetivos correspondientes a cada nivel.

## 6.3. Desarrollo del algoritmo de *pathfinding* y estructura del código

El siguiente paso consistió en desarrollar el algoritmo de *pathfinding* y el resto del código que se utilizó para tener una aplicación completamente funcional. El objetivo era que la aplicación guiara a los usuarios por el tour de manera eficiente y precisa.

### 6.3.1. Implementación del algoritmo de *pathfinding* basado en A\*

Para poder completar este paso se tuvo que implementar un algoritmo de *pathfinding*, el cual pudiera guiar a los usuarios por todo el CIT. Para esto se desarrolló un algoritmo A\* en C#, el cual fuera capaz de guiar a un usuario de un punto de origen a un punto objetivo. Específicamente, este paso consistió en crear dos clases:

- **Clase Node:** Representaba cada uno de los nodos que el usuario podía recorrer.
- **Clase AStarPathFinding:** Contenía la lógica central del algoritmo A\*, que incluía la generación de una cuadrícula para definir las áreas navegables, el cálculo de la heurística utilizando la distancia Manhattan, la identificación de los nodos vecinos y el cálculo del camino más corto.

Desafortunadamente, este algoritmo no se pudo utilizar, ya que la cuadrícula que creaba el código, no coincidía con el NavMesh que estaba definido dentro de los modelos. Esto resultaba en una ruta creada, pero en otras coordenadas dentro del entorno de Unity, lo que significa que la ruta no conectaba directamente al usuario y al objetivo.

### 6.3.2. Implementación del algoritmo de *pathfinding* basado en NavMesh

Por lo tanto, se siguieron nuevamente los pasos del tutorial de **FireDragonGamingStudio**, el cual explicaba el *pathfinding* se podía lograr utilizando una función propia del NavMesh, llamada `calculatePath()`. Por lo cual, se implementó esta función como la base del código, la cual permitía guiar al usuario, por medio de una línea, de manera precisa a través del NavMesh.

### 6.3.3. Definición de rutas y objetivos

Una vez implementada la solución del *pathfinding*, se procedió a completar el resto del código. Esto implicó cambiar toda la estructura de la aplicación, ya que para este punto solo se permitía guiar a un usuario de un punto inicial a un solo objetivo, lo cual no funcionaba para el proyecto, ya que se necesitaba que la aplicación fuera capaz de guiar a un usuario a través de un tour completo. Para esto se implementaron tres clases nuevas:

- **Clase Target:** Representaba un objetivo o punto dentro del recorrido.
- **Clase Level:** Contenía un conjunto de objetivos pertenecientes a un nivel específico.
- **Clase Route:** Definía una ruta completa, compuesta por varios niveles y sus respectivos objetivos.

Con esta nueva estructura establecida, la aplicación permite almacenar un listado de rutas, donde cada ruta consiste en niveles y cada nivel, a su vez, en una lista de objetivos. Con esta estructura, se implementó la funcionalidad para que el usuario pueda elegir la ruta preferida, definiendo a su vez los niveles y objetivos correspondientes de esa ruta. Asimismo, el usuario ahora puede ser guiado a través de varios puntos dentro de esa ruta, en lugar de detenerse al alcanzar un solo objetivo.

### 6.3.4. Navegación entre niveles y escaneo de códigos QR

Al tener la estructura del código completa, en teoría, ya se podía completar una ruta, pero surgía otro problema, como cambiar de niveles dentro del tour. Este problema surge, ya que los modelos 3D son elementos individuales y no están conectados de ningún modo dentro de Unity. Lo que significa que el NavMesh no conecta a los diferentes niveles y por ende, no se logra establecer una ruta de un nivel al otro.

Para resolver esto, se utilizaron nuevamente los tutoriales de **FireDragonGamingStudio**, en donde explica que se pueden utilizar códigos QR para resetear la posición del usuario a un punto específico. Utilizando esta lógica, se pensó en utilizar el código que fuera capaz de escanear un código QR, los cuales se posicionaron estratégicamente afuera de los elevadores, tanto principales como secundarios de cada nivel. Esto permite que a la hora de completar un nivel, el usuario pueda escanear el código QR del siguiente nivel, lo que a su vez reposiciona al usuario en ese mismo punto, colocándolo en el nuevo nivel y guiándolo hacia el siguiente objetivo.

Asimismo, esta misma lógica también se utilizó al inicio de la aplicación. Esto se hizo con el objetivo de que a la hora de iniciar el recorrido, el algoritmo *pathfinding* no se active automáticamente,

sino que tenga que esperar a que el usuario escanee el código QR posicionado en el punto exacto donde inicia la ruta, para poder activarse.

### 6.3.5. Mejora de la experiencia de usuario: implementación de una flecha de navegación

Con la estructura del código finalizada, se decidió mejorar la experiencia del usuario sustituyendo la línea que guiaba el recorrido por una flecha. Inicialmente, la flecha debía seguir el camino de la línea, pero surgieron problemas cuando la flecha intentaba guiar al usuario por puntos demasiado cercanos entre sí, lo que provocaba errores y retrasos en la actualización de la dirección.

Para solucionar esto, se implementó un enfoque en el cual la flecha apuntaba solo hacia el siguiente cruce importante en la ruta (con un ángulo mayor a 20 grados), en lugar de apuntar al siguiente punto inmediato. Esto mejoró el rendimiento, pero todavía había casos en los que la flecha no actualizaba correctamente si el usuario no pasaba exactamente por los puntos de la ruta.

La solución final consistió en dibujar una línea perpendicular desde el punto de la ruta, lo que permitía a la flecha actualizarse incluso si el usuario pasaba ligeramente fuera del punto exacto. Además, se implementó una lógica de detección de proximidad que activaba la actualización de la flecha un metro antes de alcanzar el punto objetivo, asegurando una navegación fluida y sin confusiones para el usuario.

Finalmente, para poder mejorar la experiencia de usuario, se decidió utilizar una flecha en lugar de utilizar la línea, como herramienta de guía. El razonamiento fue que la flecha era menos invasiva y a su vez mejoraba la experiencia de usuario. La lógica que se utilizó es que la flecha debía de seguir la ruta que la línea dibujada, por lo tanto, la línea no se eliminó por completo, sino que solo se escondió, lo que permitió que la flecha se pudiera guiar de la línea.

Pero esto trajo problemas, ya que la línea básicamente es un conjunto de varios puntos. La flecha, por lo tanto, trataba de guiar al usuario por cada uno de esos puntos, haciendo que la flecha se trabara y no se actualizara a tiempo para permitir un comportamiento fluido. La solución para este problema fue que la flecha solo apuntara hacia el siguiente cruce en la ruta, y no a cada uno de los puntos que formaban parte de la ruta. Esta mejora sí ayudó, ya que a este punto la flecha solo se actualizaba cuando el usuario pasaba un cruce dentro de la ruta con un ángulo mayor a 20 grado. Pero había otro problema, y este era que, ya que la flecha se guiaba por la línea, si el usuario pasaba a un lado del punto objetivo y no exactamente encima de él, la flecha no detectaba este objetivo como completo y seguía apuntando hacia atrás, resultando en una experiencia confusa.

La solución final consistió en tomar la línea perpendicular desde el siguiente punto objetivo. De este modo, si un usuario pasaba al lado del punto objetivo, siempre tendría contacto con la línea perpendicular, actualizando el nuevo objetivo. Adicionalmente, se implementó una lógica de detección de proximidad, la cual actualizaba el nuevo objetivo un metro antes de alcanzar el objetivo actual, asegurando la navegación fluida que se buscaba con esta herramienta.

## 6.4. Integración de la interfaz de usuario

Una vez finalizado el desarrollo del algoritmo y la estructura del código, se procedió a implementar la interfaz de usuario (UI) para permitir la interacción del usuario con la aplicación y mostrar información relevante sobre el tour.

Para crear una interfaz completa, se añadieron varios elementos de UI proporcionados por Unity, como paneles, botones, y textos dinámicos. Estos elementos permitieron que la aplicación no solo

fuerza funcional, sino también intuitiva y fácil de usar.

Además de los elementos básicos de UI, se implementaron cálculos e indicadores para mostrar información relevante durante el tour, tales como:

- Distancia entre el usuario y el siguiente objetivo.
- Nombre o descripción del siguiente objetivo.
- Nivel actual en el que se encontraba el usuario.
- Número total de objetivos en el nivel.
- Porcentaje del tour completado.

Cada uno de estos elementos fue vinculado con su respectiva variable en el código, asegurando que los datos mostrados se actualizaran en tiempo real. A su vez, se añadió lógica a los botones para que se activaran en el momento adecuado y ejecutaran las funciones correspondientes cuando el usuario los presionara.

#### 6.4.1. Integración de módulos adicionales

Además, en este paso también se realizaron las integraciones con los otros módulos del proyecto. Primero se hizo la integración con el módulo *Desarrollo de UX/UI en Aplicación de Recorridos Virtuales con Unity*. Esta integración reemplazó la interfaz de usuario que se trabajó en este módulo, por una interfaz más atractiva visualmente y más intuitiva. Luego se hizo la integración con el módulo *Desarrollo de minijuegos para el recorrido virtual de la Universidad del Valle de Guatemala*. En esta integración, se añadió la funcionalidad de tener minijuegos en ciertos puntos de interés, lo que permitió añadirle interactividad adicional al tour.

### 6.5. Configuración de *plug-ins* necesarios

Para garantizar la compatibilidad de la aplicación con dispositivos Android e iOS, se configuraron los *plug-ins* ARKit y ARCore, respectivamente. Unity, a través de su marco de trabajo AR Foundation, incluye estos *plug-ins* por defecto, lo que facilitó su instalación y configuración dentro del proyecto. De este modo, se buscó asegurar que la aplicación funcionara de manera consistente en ambas plataformas al ejecutarse.

### 6.6. Pruebas internas y pruebas de usuario

A lo largo del desarrollo del proyecto, se realizaron tanto pruebas internas como pruebas de usuario para asegurar que la aplicación funcionara de manera óptima.

#### 6.6.1. Pruebas de los modelos 3D

Las pruebas internas se llevaron a cabo durante todo el proceso de desarrollo. Estas pruebas fueron cruciales, ya que permitieron evaluar cada funcionalidad a medida que se completaba, identificando y corrigiendo errores en tiempo real. Un ejemplo claro de esto fue el modelado de los niveles del

CIT. Con cada nivel terminado, se realizaron pruebas para verificar si el modelo era congruente con la estructura real del CIT. Estas pruebas revelaron discrepancias en las medidas de varios pasillos, lo que llevó a la decisión de tomar medidas manuales adicionales para corregir los modelos.

### 6.6.2. Pruebas del código y aplicación

De manera similar, se realizaron pruebas continuas al implementar el algoritmo de *pathfinding*. Estas pruebas se llevaron a cabo en la universidad, comprobando que tanto el *pathfinding* como los modelos 3D estuvieran alineados y funcionaran correctamente. A medida que se añadían nuevas funcionalidades, como cálculos adicionales o características del código, se verificaba su correcto funcionamiento. Unity permitió ejecutar muchas de estas pruebas dentro de la aplicación, lo que facilitó el proceso de depuración. Sin embargo, también se realizaron pruebas con el dispositivo en mano, desplazándose físicamente por el tour para asegurar una experiencia fluida en condiciones reales.

### 6.6.3. Pruebas de rendimiento

Al concluir el desarrollo del código, se hicieron pruebas adicionales de rendimiento para medir el consumo de memoria de la aplicación. Estas pruebas resultaron esenciales, ya que revelaron un problema con la funcionalidad de escaneo de códigos QR, la cual consumía demasiados recursos, provocando que la aplicación se cerrara inesperadamente tras unos segundos. Este descubrimiento permitió optimizar el código y solucionar el problema. Estas pruebas de rendimiento se llevaron a cabo en dispositivos Android e iOS para garantizar que la aplicación funcionara de manera uniforme en ambos sistemas operativos.

### 6.6.4. Pruebas de usuario

Finalmente, se realizaron pruebas de usuario con la aplicación completa para obtener retroalimentación sobre su funcionamiento y facilidad de uso. El grupo objetivo que se seleccionó para estas pruebas fue el de los estudiantes de la UVG, ya que ellos forman parte de las personas que utilizarán la versión final y a su vez, fue un grupo fácil de acceder. Estas pruebas fueron clave para ajustar detalles finales y asegurar que la experiencia del usuario fuera satisfactoria. Como parte de este proceso, se creó un formulario de Google que se distribuyó a varios estudiantes de la Universidad del Valle de Guatemala. El formulario contenía preguntas clave sobre la aplicación, evaluando aspectos como la claridad de la interfaz, la facilidad de uso, si la aplicación era intuitiva o no, y la experiencia general del usuario. Las respuestas obtenidas proporcionaron información valiosa que ayudó a realizar ajustes finales, optimizando la usabilidad y mejorando la experiencia general del tour virtual.

# CAPÍTULO 7

---

## Resultados

---

En este capítulo se presentan los resultados obtenidos a lo largo del desarrollo del proyecto. Se han organizado de manera que corresponden a los diferentes aspectos clave abordados, permitiendo una comprensión clara y detallada de los avances alcanzados. Cada sección ofrece una visión de los hallazgos en relación con las medidas, comparaciones y análisis realizados, brindando una base sólida para las conclusiones y futuras recomendaciones.

### 7.1. Modelado del CIT utilizando Blender

Como resultado del proceso de modelado tridimensional, se compararon las medidas obtenidas de los planos con las medidas reales. En las Tablas 7.2 a 7.7, se presentan los resultados para los niveles 1, 2, 3, 4, 6 y 7 del CIT. Cada tabla muestra los datos de ancho y largo de cada pasillo, junto con el porcentaje de error correspondiente para cada medición. Para facilitar la identificación de los pasillos evaluados en cada nivel, las Figuras 7.1 a 7.6 actúan como clave visual, indicando a cuál pasillo se refiere cada tabla.

En la Tabla 7.1, se presenta un resumen del porcentaje de error promedio obtenido para cada nivel, el cual permite observar la precisión general de los modelos tridimensionales. El porcentaje de error promedio general de todos los niveles es de **4.87 %**.

Tabla 7.1: Porcentajes de error promedio por nivel

Nivel	Porcentaje de Error Promedio (%)
Nivel 1	8.37
Nivel 2	3.80
Nivel 3	6.30
Nivel 4	2.59
Nivel 5	3.53
Nivel 6	6.72
<b>Promedio General</b>	<b>4.87</b>

Tabla 7.2: Comparación de medidas teóricas y reales nivel 1

Pasillo	Medida en Plano (m)	Medida Real (m)	Porcentaje Error (%)
Ancho Passillo 1	3.2	2.9	10.34
Largo Passillo 1	13.9	13	6.92
Ancho Passillo 2	3.2	4.9	34.69
Largo Passillo 2	13.9	13	6.92
Ancho Passillo 3	1.4	1.5	6.67
Largo Passillo 3	13.9	13	6.92
Ancho Passillo 4	6.4	6.5	1.54
Largo Passillo 4	29.6	28.5	3.86
Ancho Passillo 5	3.2	3.6	11.11
Largo Passillo 5	58.5	57.5	1.74
Ancho Passillo 6	4.1	4.2	2.38
Largo Passillo 6	16.6	17	2.35
Ancho Passillo 7	4	4.2	4.76
Largo Passillo 7	62.9	62.8	0.16
Ancho Passillo 8	2.5	1.9	31.58
Largo Passillo 8	74	72.2	2.49

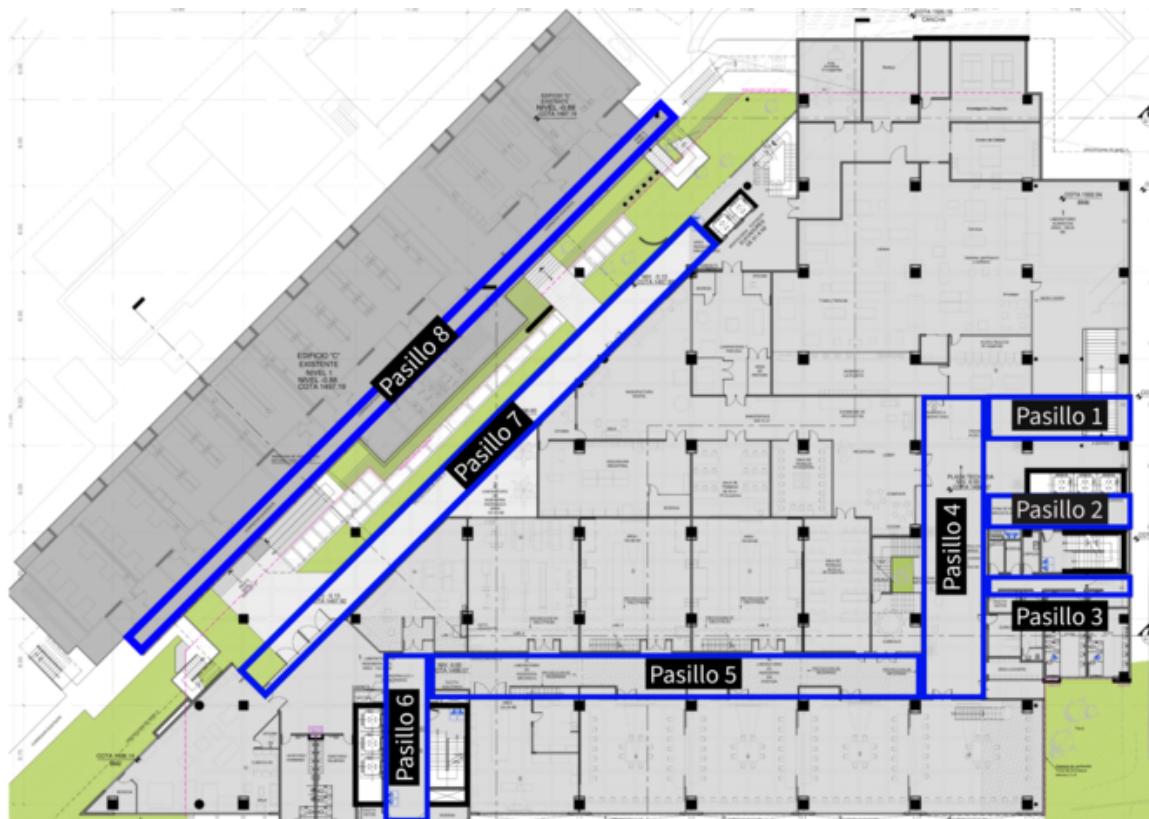


Figura 7.1: Clave pasillos nivel 1

Tabla 7.3: Comparación de medidas teóricas y reales nivel 2

Pasillo	Medida en Plano (m)	Medida Real (m)	Porcentaje Error (%)
Ancho Passillo 1	3.2	4.9	34.69
Largo Passillo 1	13.9	13	6.92
Ancho Passillo 2	1.4	1.5	6.67
Largo Passillo 2	13.9	13	6.92
Ancho Passillo 3	7	7	0
Largo Passillo 3	13	13	0
Ancho Passillo 4	5.9	5.8	1.72
Largo Passillo 4	18	18	0
Ancho Passillo 5	3.9	3.9	0
Largo Passillo 5	19.5	19.5	0
Ancho Passillo 6	4.2	4.2	0
Largo Passillo 6	13.2	13.4	1.49
Ancho Passillo 7	1.95	2	2.5
Largo Passillo 7	18.5	18.5	0
Ancho Passillo 8	2.8	3	6.67
Largo Passillo 8	13.5	13.5	0

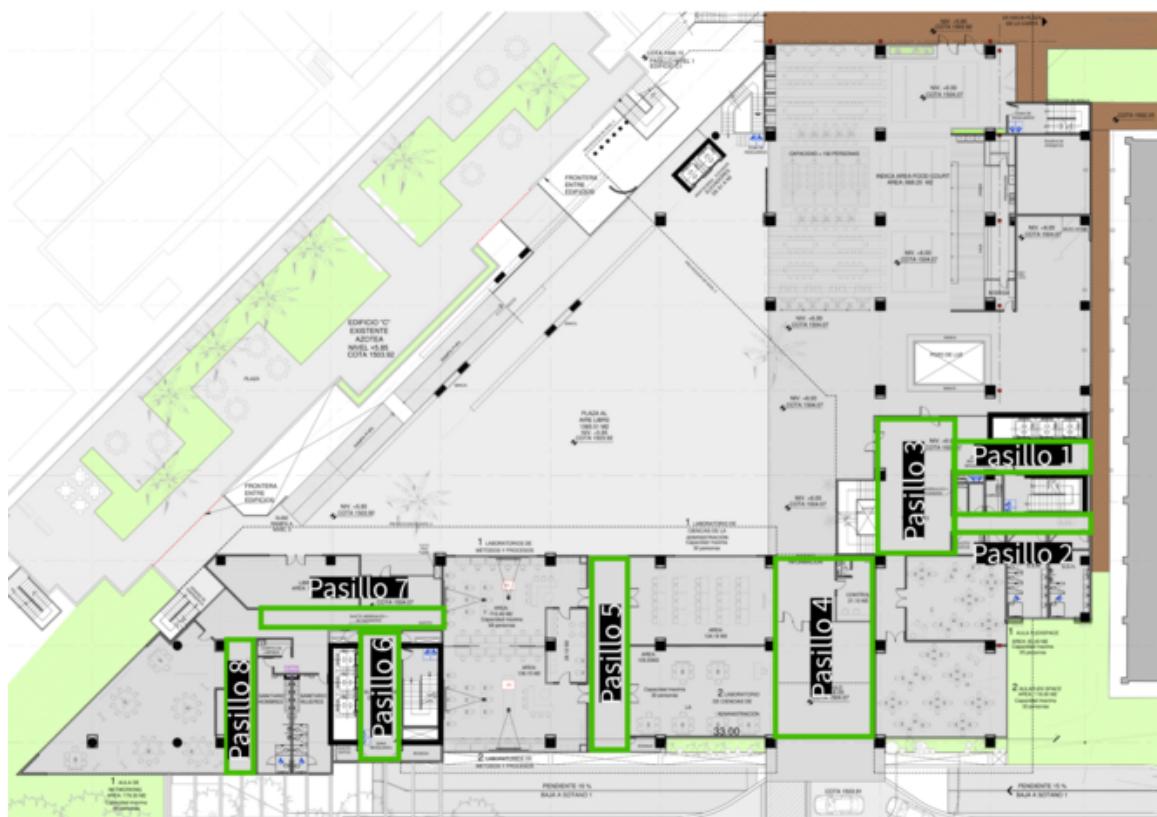


Figura 7.2: Clave pasillos nivel 2

Tabla 7.4: Comparación de medidas teóricas y reales nivel 3

Pasillo	Medida en Plano (m)	Medida Real (m)	Porcentaje Error (%)
Ancho Passillo 1	1.7	1.7	0
Largo Passillo 1	11.4	11.4	0
Ancho Passillo 2	1.4	1.4	0
Largo Passillo 2	20.8	20.8	0
Ancho Passillo 3	1.5	1.4	7.14
Largo Passillo 3	23.9	23.9	0
Ancho Passillo 4	6.7	8.9	24.72
Largo Passillo 4	11.4	13.5	15.56
Ancho Passillo 5	3.2	4.9	34.69
Largo Passillo 5	11.4	13.5	15.56
Ancho Passillo 6	1.4	1.5	6.67
Largo Passillo 6	13.9	13	6.92
Ancho Passillo 7	2.9	2.4	20.83
Largo Passillo 7	25.2	25.4	0.79
Ancho Passillo 8	2.9	3	3.33
Largo Passillo 8	41.3	42.3	2.36
Ancho Passillo 9	7.2	7.3	1.37
Largo Passillo 9	12.3	12.4	0.81
Ancho Passillo 10	1.9	2.1	9.52
Largo Passillo 10	21.7	21.8	0.46
Ancho Passillo 11	4.1	4.2	2.38
Largo Passillo 11	15	15.6	3.85

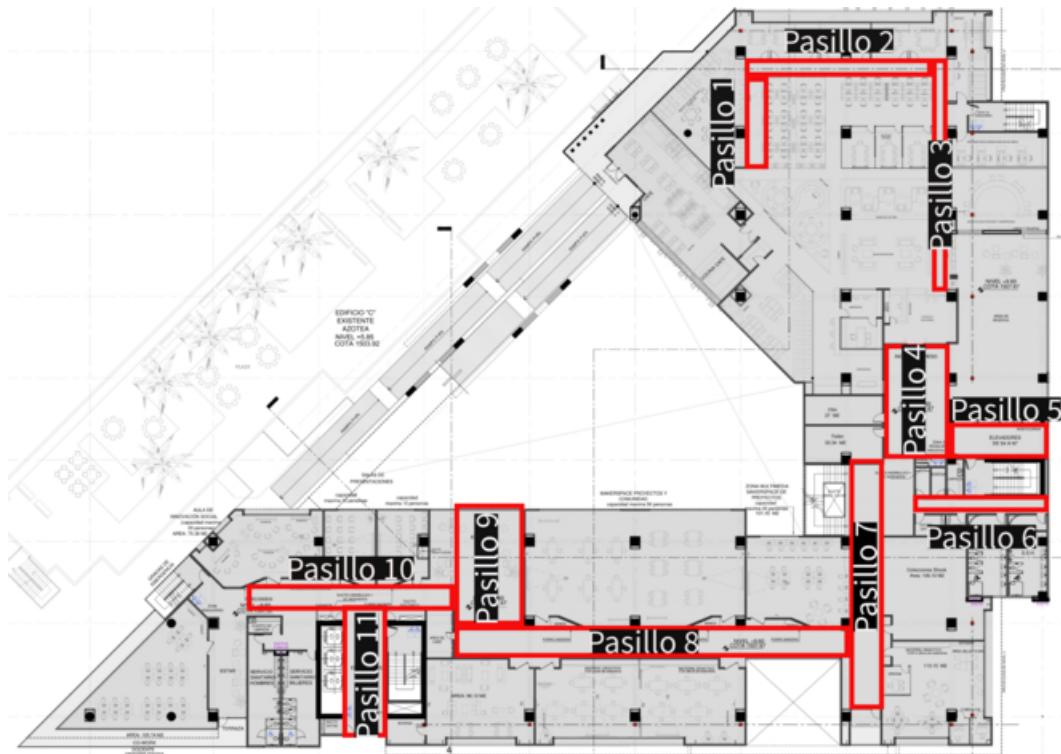


Figura 7.3: Clave pasillos nivel 3

Tabla 7.5: Comparación de medidas teóricas y reales nivel 4

Pasillo	Medida en Plano (m)	Medida Real (m)	Porcentaje Error (%)
Ancho Passillo 1	4.9	5	2
Largo Passillo 1	22.9	23.1	0.87
Ancho Passillo 2	7.8	8.1	3.7
Largo Passillo 2	19.5	19.7	1.02
Ancho Passillo 3	3.8	3.8	0
Largo Passillo 3	17.2	17.6	2.27
Ancho Passillo 4	6.5	6.5	0
Largo Passillo 4	20.8	20.5	1.46
Ancho Passillo 5	3.2	4.9	34.69
Largo Passillo 5	13.8	13.8	0
Ancho Passillo 6	1.4	1.5	6.67
Largo Passillo 6	13.9	13	6.92
Ancho Passillo 7	4.7	4.8	2.08
Largo Passillo 7	12.5	12.4	0.81
Ancho Passillo 8	2.9	3	3.33
Largo Passillo 8	33.3	33.1	0.6
Ancho Passillo 9	4.3	4.2	2.38
Largo Passillo 9	9.7	9.8	1.02
Ancho Passillo 10	1.9	2	5
Largo Passillo 10	12.5	12.5	0
Ancho Passillo 11	4.2	4.2	0
Largo Passillo 11	15	15.5	3.23

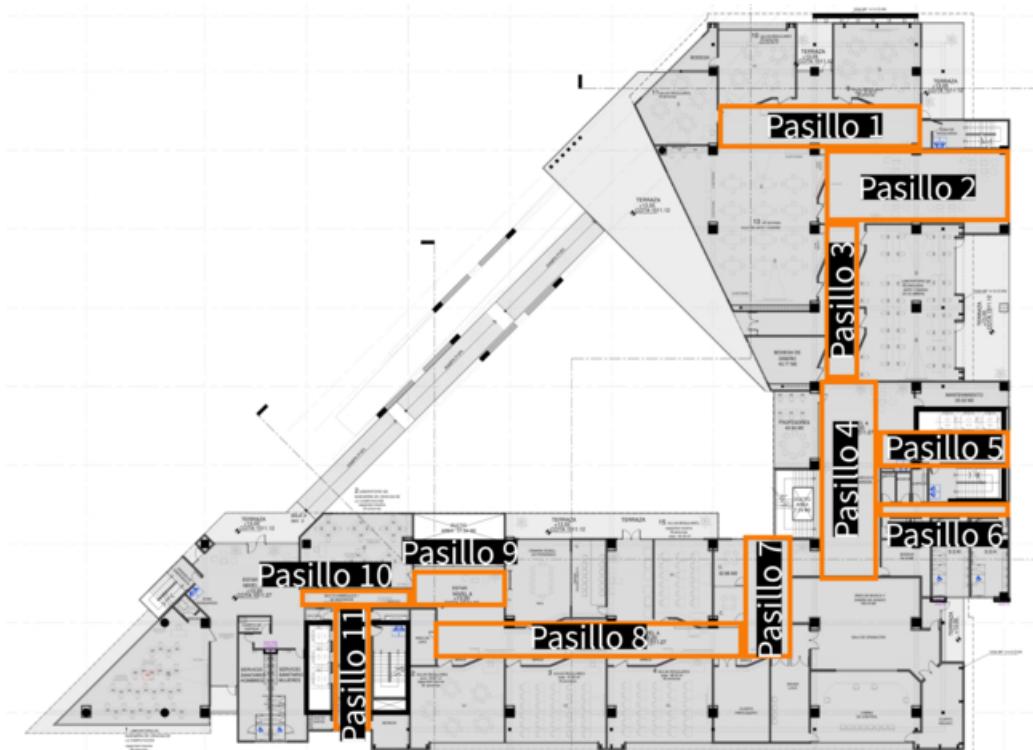


Figura 7.4: Clave pasillos nivel 4

Tabla 7.6: Comparación de medidas teóricas y reales nivel 6

Pasillo	Medida en Plano (m)	Medida Real (m)	Porcentaje Error (%)
Ancho Passillo 1	2.9	3	3.33
Largo Passillo 1	34.5	34.5	0
Ancho Passillo 2	3.2	4.9	34.69
Largo Passillo 2	25.4	25.7	1.17
Ancho Passillo 3	1.4	1.5	6.67
Largo Passillo 3	13.9	13	6.92
Ancho Passillo 4	6.3	6.4	1.56
Largo Passillo 4	17	17	0
Ancho Passillo 5	2.9	3	3.33
Largo Passillo 5	49.1	49.2	0.2
Ancho Passillo 6	6.8	6.9	1.45
Largo Passillo 6	9.7	9.8	1.02
Ancho Passillo 7	1.9	2	5
Largo Passillo 7	18.9	18.7	1.07
Ancho Passillo 8	4.2	4.2	0
Largo Passillo 8	15.1	15.4	1.95

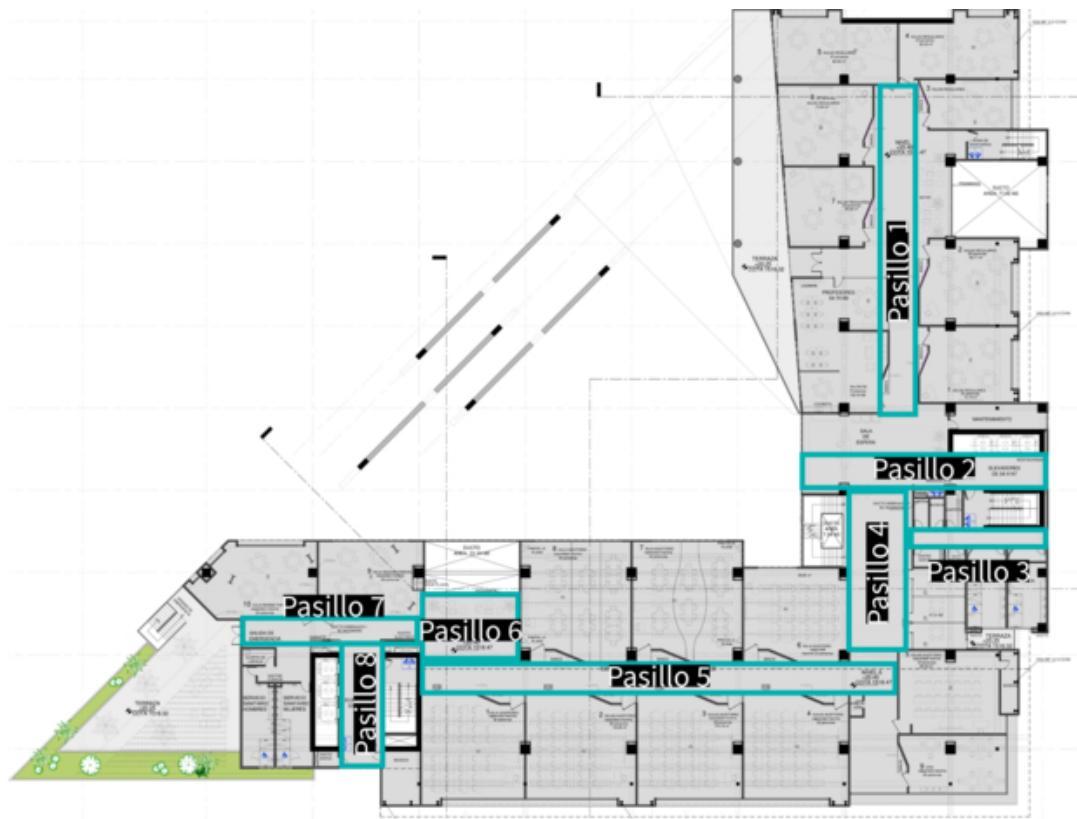


Figura 7.5: Clave pasillos nivel 6

Tabla 7.7: Comparación de medidas teóricas y reales nivel 7

Pasillo	Medida en Plano (m)	Medida Real (m)	Porcentaje Error (%)
Ancho Passillo 1	8.4	8.4	0
Largo Passillo 1	14	14.2	1.41
Ancho Passillo 2	3.9	4	2.5
Largo Passillo 2	20.2	20.3	0.49
Ancho Passillo 3	3.2	4.9	34.69
Largo Passillo 3	16.4	15.8	3.8
Ancho Passillo 4	1.4	1.5	6.67
Largo Passillo 4	13.9	13.3	4.51
Ancho Passillo 5	7.1	7.2	1.39
Largo Passillo 5	7.4	7.4	0
Ancho Passillo 6	3.4	3.4	0
Largo Passillo 6	10.7	10.5	1.9
Ancho Passillo 7	5.3	11	51.82
Largo Passillo 7	62.6	60.2	3.99
Ancho Passillo 8	4.2	4.2	0
Largo Passillo 8	11.9	12	0.83

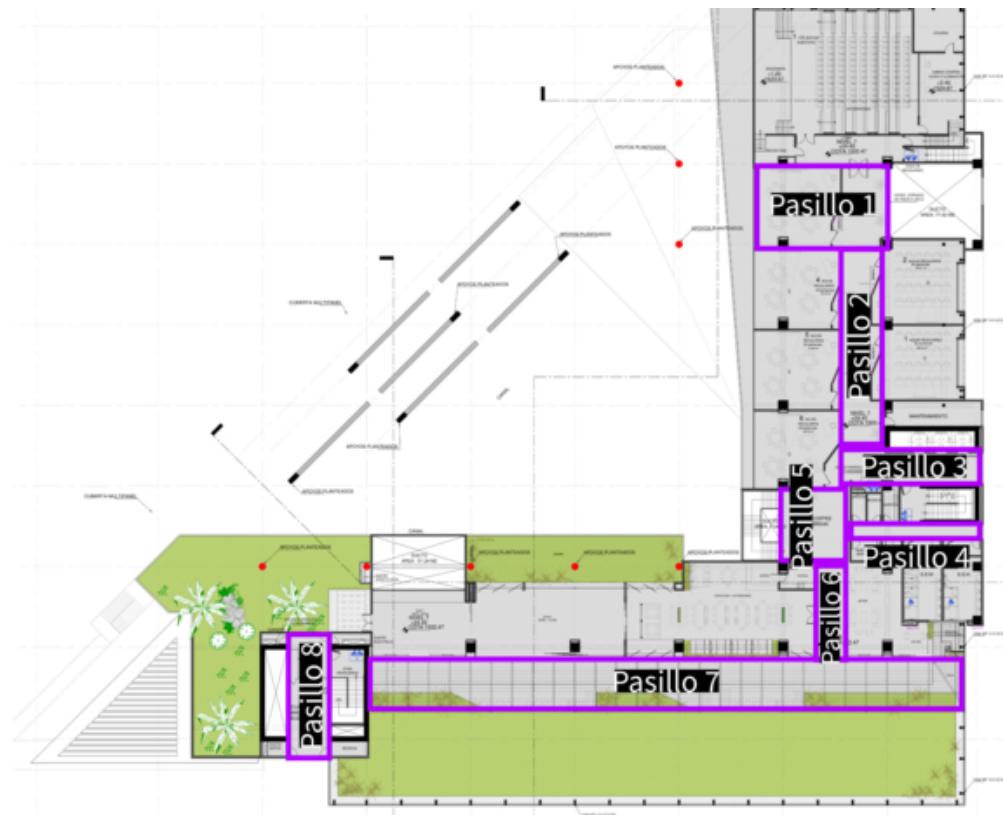


Figura 7.6: Clave pasillos nivel 7

Durante el proceso de modelado, se realizaron ajustes en los modelos 3D basados en las medidas manuales, mejorando así la precisión dimensional. En las Figuras 7.7 a 7.9, se presentan imágenes comparativas que muestran las etapas clave del proceso de modelado para el Nivel 2: el plano original, el plano en Blender con el contorno trazado y el modelo 3D final. A modo de ejemplo, en esta sección solo se muestran los resultados del proceso que se realizó con el Nivel 2, mientras que el resto de los modelos de los otros niveles se pueden encontrar en el Anexo C.



Figura 7.7: Plano original nivel 2

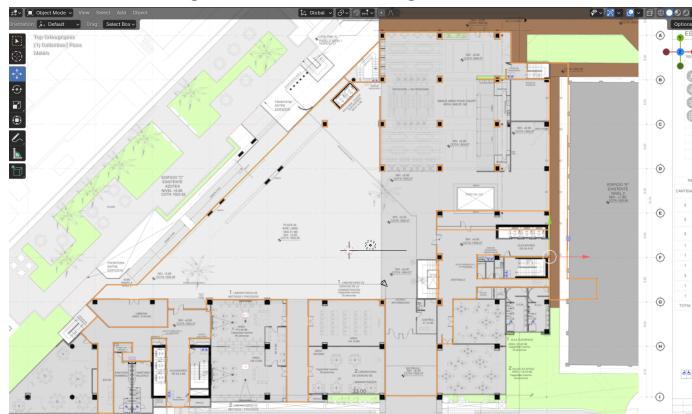


Figura 7.8: Contorno trazado en plano nivel 2

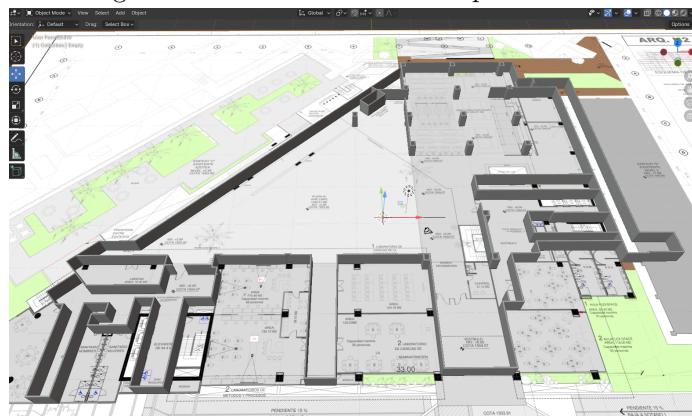


Figura 7.9: Modelo 3D nivel 2

## 7.2. Construcción la escena dentro de Unity

Los resultados de la construcción de la escena dentro de Unity incluyen una serie de imágenes, las cuales muestran el ambiente final. La Figura 7.10 muestra los seis modelos 3D y su área navegable correspondiente, definida por el NavMesh. Adicionalmente, se pueden observar otras dos imágenes, cuáles muestran el resto de los elementos que forman parte del ambiente.

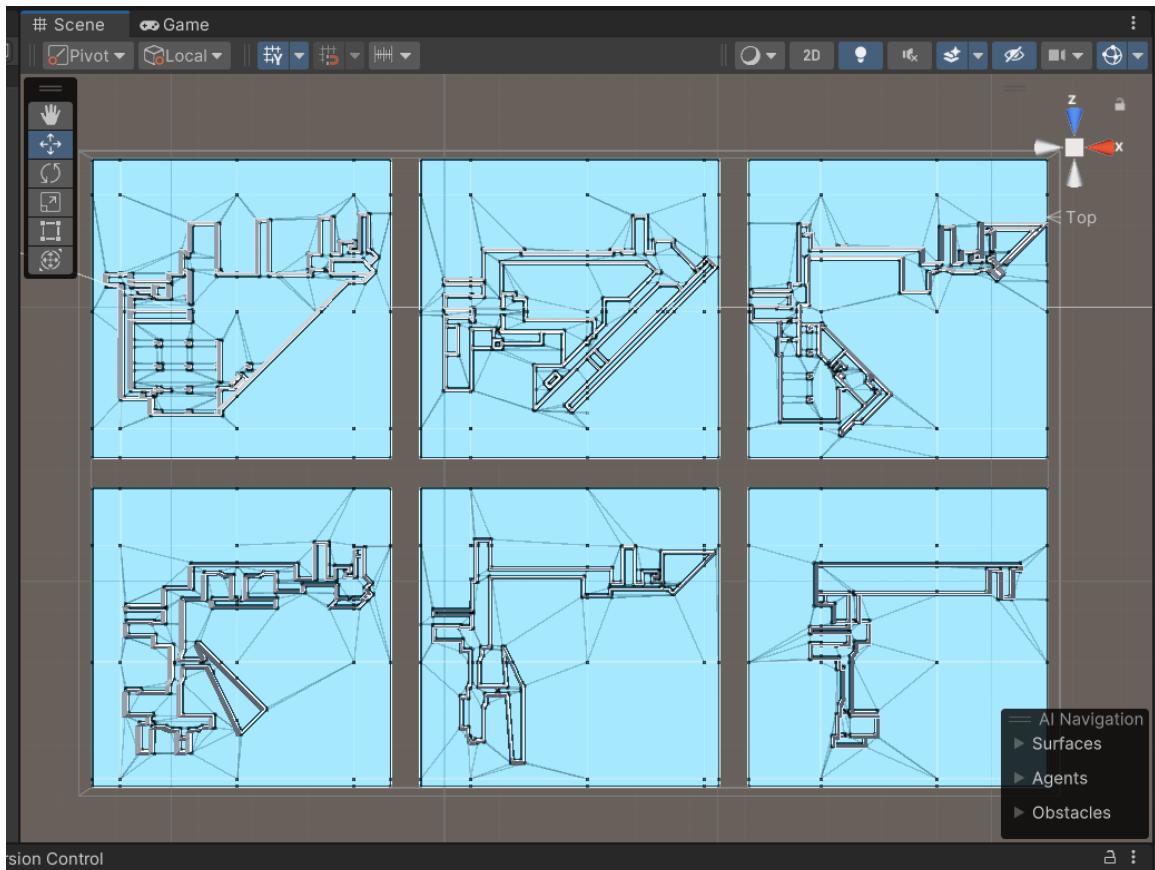


Figura 7.10: Escena Completa en Unity

Para ilustrar los elementos utilizados en la creación del entorno de realidad aumentada en Unity, la Figura 7.11 muestra la configuración de los componentes esenciales. Estos elementos gestionan la interacción con el entorno virtual, habilitan la cámara del dispositivo y permiten la movilidad dentro de la escena. Por otro lado, la Figura 7.12 presenta los elementos que componen el ambiente completo del tour virtual, organizados por nivel. Cada nivel contiene su modelo tridimensional, junto con los objetivos y códigos QR específicos. Los objetivos y códigos QR fueron posicionados para coincidir con ubicaciones precisas en el entorno físico.



Figura 7.11: Elementos esenciales para la AR



Figura 7.12: Elementos del ambiente

### 7.3. Desarrollo del algoritmo de *pathfinding* y estructura del código

Los resultados del desarrollo del algoritmo de *pathfinding* y la estructura del código, se presentan de manera ordenada en sus respectivas secciones. Cada sección presenta imágenes y, de ser necesario, fragmentos del código resultante, los cuales apoyan a tener una mejor comprensión sobre los problemas y soluciones que surgieron al seguir la metodología.

### 7.3.1. Algoritmo desarrollado

El enfoque principal para el desarrollo del *pathfinding* era poder desarrollar un algoritmo basado en el algoritmo A\*. A continuación se muestra el código de la clase Nodo, la cual funcionaba como base para el algoritmo. El resto del código correspondiente a la implementación completa del algoritmo se encuentra en la clase ASTarPathFinding, de la cual se presenta la función principal `findPath(startPos, targetPos)` en el Anexo D.1.

```

1  public class Node
2  {
3      public bool walkable;
4      public Vector3 position;
5      public int gridX;
6      public int gridY;
7      public Node parent;
8      public int gCost; // costo de movimiento desde el nodo inicial
9      public int hCost; // heuristica de costo de movimiento al nodo final
10     public int fCost { get { return gCost + hCost; } } // costo total
11
12    // Constructor de la clase
13    public Node(bool _walkable, Vector3 _position, int _gridX, int _gridY)
14    {
15        walkable = _walkable;
16        position = _position;
17        gridX = _gridX;
18        gridY = _gridY;
19    }
20 }
```

Código 7.1: Estructura del objeto Node

### 7.3.2. Implementación exitosa del algoritmo basado en NavMesh

Para implementar la función de pathfinding `NavMesh.CalculatePath()` de Unity, se realizaron modificaciones en la función `CalculateAndDrawPath()` del código principal del tour virtual. A continuación se presentan los fragmentos de código principales, que muestran las llamadas a las funciones correspondientes en ambas implementaciones de pathfinding.

El Código 7.2 ilustra la invocación de la función de pathfinding A\* desarrollada manualmente. Adicionalmente, el Código 7.3 muestra la implementación de la función integrada de Unity `CalculatePath()`, utilizada para calcular rutas en el entorno del NavMesh.

```

1  GameObject currentTarget = routes[currentRouteIndex].Levels[
2      currentLevelIndex].Targets[currentTargetIndex].PositionObject;
// Utilizar A* para encontrar el camino
3  List<Node> pathNodes = aStarPathfinding.FindPath(transform.position,
4      currentTarget.transform.position);
```

Código 7.2: Código que invoca a la función de A\* desarrollado manualmente

```

1  GameObject currentTarget = routes[currentRouteIndex].Levels[
2      currentLevelIndex].Targets[currentTargetIndex].PositionObject;
// Utilizar CalculatePath para encontrar el camino
3  if (NavMesh.CalculatePath(transform.position, currentTarget.transform.
4      position, NavMesh.AllAreas, path))
```

Código 7.3: Código que invoca a la función de *pathfinding* integrada en Unity

Además, se presenta la Figura 7.13 que ilustra el comportamiento del algoritmo desarrollado manualmente en contraste con el de Unity, permitiendo observar las diferencias en la forma en que ambos calculan rutas en el entorno virtual. La ruta dibujada por el algoritmo desarrollado se muestra en verde en la esquina superior derecha de la Imagen A, mientras que la ruta dibujada por el algoritmo integrado de Unity se muestra en rojo dentro del mapa en la Imagen B.

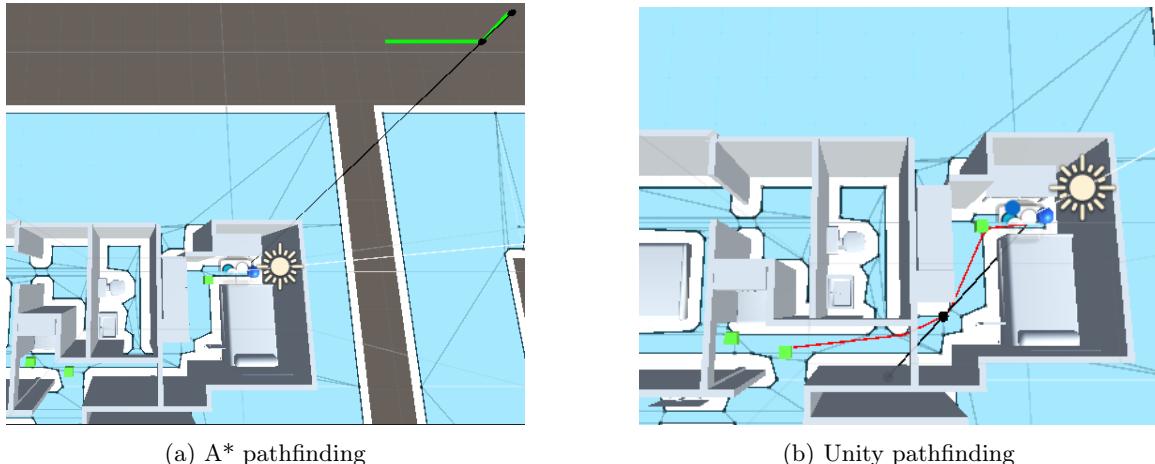


Figura 7.13: Comparación entre A\* pathfinding (Imagen A) y Unity pathfinding (Imagen B).

### 7.3.3. Rutas y objetivos dentro del tour virtual

Se establecieron y estructuraron las rutas dentro del tour virtual, utilizando una jerarquía de clases para organizar el recorrido en el entorno. La estructura está compuesta por tres objetos principales: *Route*, *Level* y *Target*, los cuales permiten definir y manejar las rutas eficientemente. Una instancia de *Route* es una lista de objetos de tipo *Level*, y cada objeto *Level* contiene una lista de *Target*, que representan los objetivos específicos en cada nivel.

La estructura de la clase *Route* se muestra en el Código 7.4, en donde cada instancia de *Route* contiene el nombre de la ruta y una lista de niveles. En el Código 7.5 se presenta la clase *Level*, que incluye un índice del nivel y una lista de objetivos. Finalmente, la clase *Target*, presentada en el Código 7.6, define cada objetivo mediante su nombre y el objeto de posición correspondiente en el entorno virtual.

```

1  using System;
2  using UnityEngine;
3
4  [Serializable]
5  public class Route
6  {
7      public string RouteName; // Nombre de la ruta
8      public List<Level> Levels; // Lista de niveles en la ruta
9 }
```

Código 7.4: Estructura del objeto Route

```

1  using System;
2  using UnityEngine;
3
4  [Serializable]
5  public class Level
6  {
7      public string LevelIndex; // Número del nivel
8      public List<Target> Targets; // Lista de objetivos en el nivel
9  }

```

Código 7.5: Estructura del objeto Level

```

1  using System;
2  using UnityEngine;
3
4  [Serializable]
5  public class Target
6  {
7      public string Name; // Nombre del objetivo
8      public GameObject PositionObject; // Posición del objetivo
9  }

```

Código 7.6: Estructura del objeto Target

Para el manejo de múltiples rutas, se lograron definir dos recorridos diferentes para el tour. A partir de esta información, se establecieron los niveles y los objetivos específicos para cada una de las rutas, los cuales se configuraron manualmente en Unity como variables. Estas variables son utilizadas por el código para gestionar y controlar el flujo del tour dentro de la aplicación. Seguidamente, se incluye la Figura 7.14 que ilustra cómo se definieron las rutas dentro de Unity.

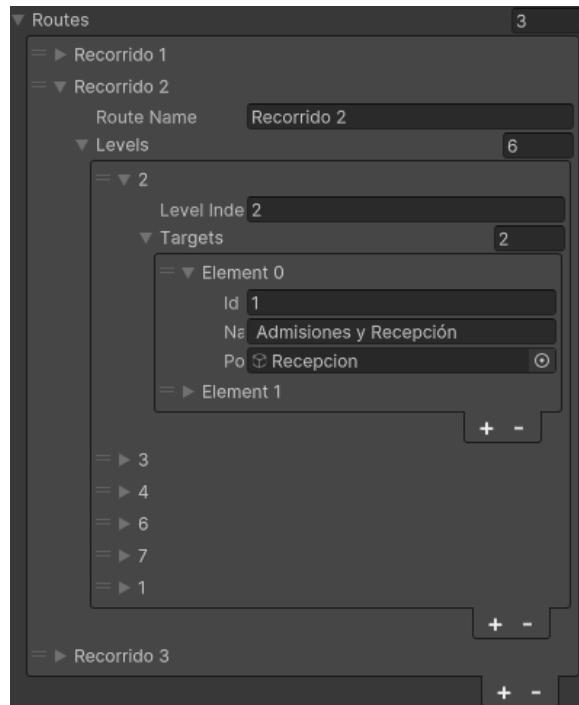


Figura 7.14: Rutas definidas dentro de Unity

### 7.3.4. Navegación entre niveles mediante códigos QR

La implementación del escaneo de los códigos QR para cambiar de nivel recalibró la posición del usuario, permitiendo resumir el tour desde el siguiente nivel. A continuación, se incluye una imagen que muestra el diseño de los códigos QR, los cuales fueron colocados específicamente en los elevadores principales y secundarios de cada nivel. Como ejemplo, se presenta el código QR de los elevadores principales del nivel 2, mientras que el resto de los códigos QR se pueden consultar en el Anexo C.

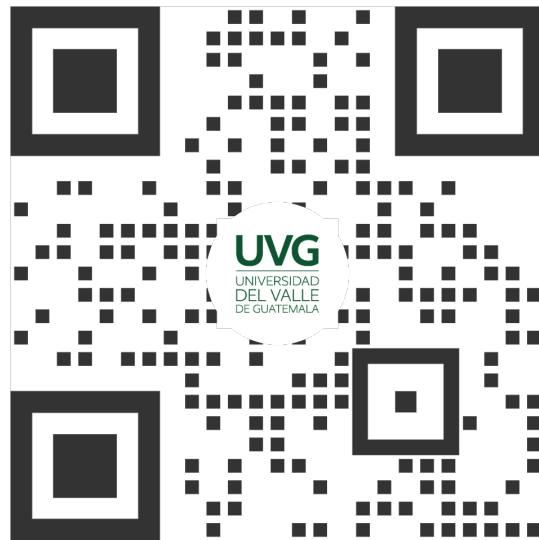
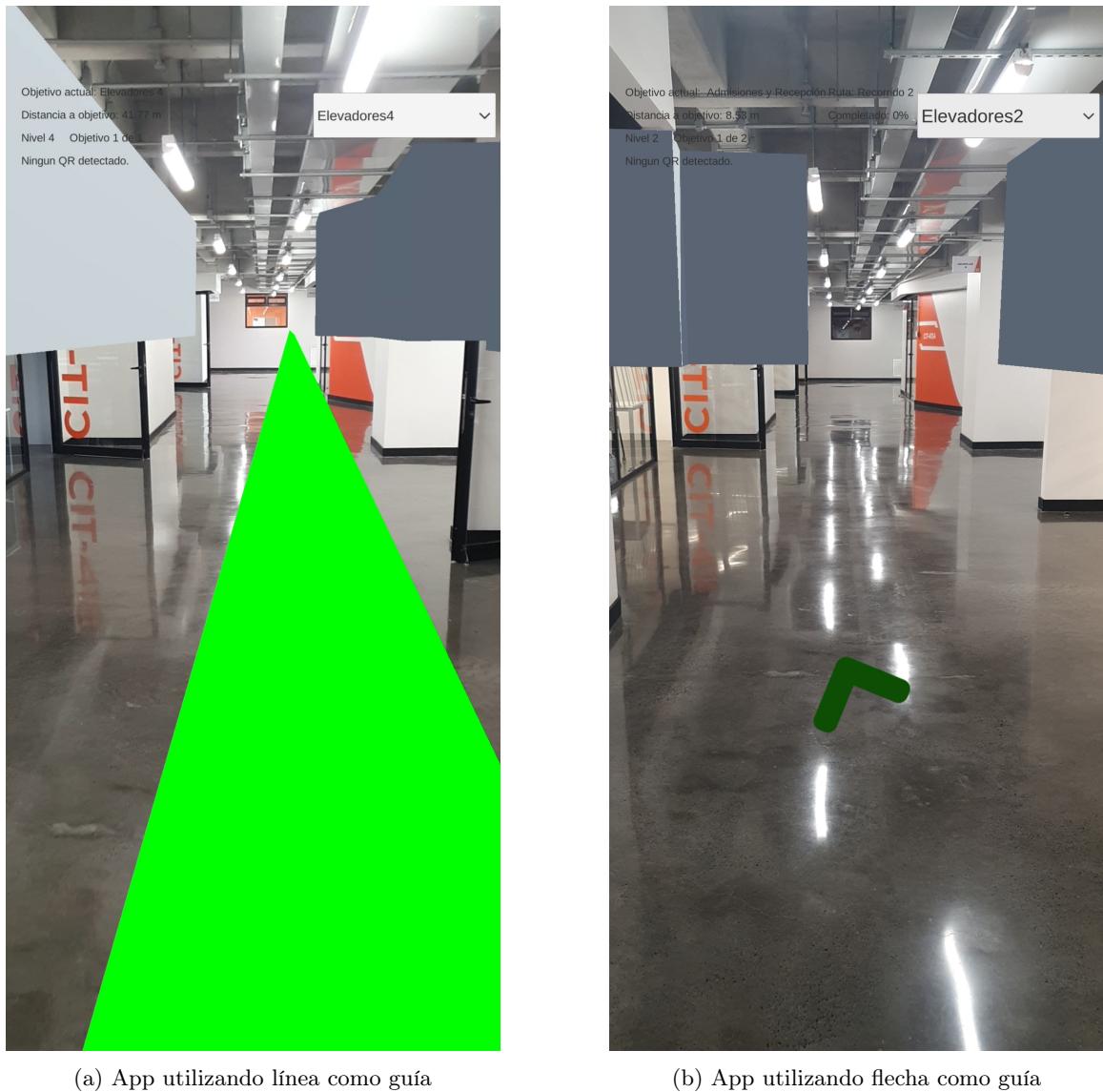


Figura 7.15: Código QR elevadores principales nivel 2

Adicionalmente, se presentan las dos funciones principales de la clase QRCodeRecenter, la que contiene toda la lógica para el funcionamiento de los códigos QR dentro del tour. La primera función, que se encuentra en el Anexo D.3, es la encargada de activar la cámara y estar en constante monitoreo para detectar un código QR. La manera en la que esto funciona es que los códigos QR solo almacenan una cadena de texto, nada más. Por lo tanto, si la cámara detecta un código QR, esta obtiene el texto plano y hace una llamada la siguiente función, con el texto como parámetro. Esta otra función, también disponible en el Anexo D.4, recibe el texto y lo compara con los nombres de los códigos QR que se definieron en el ambiente. Si la cadena escaneada coincide con alguno de estos elementos, esta función reorienta al usuario en la posición correspondiente de este elemento y reanuda el tour.

### 7.3.5. Flecha de navegación y experiencia del usuario

Para la implementación de la flecha, que reemplazó a la línea con el objetivo de mejorar la experiencia del usuario, se utilizó una imagen 2D para representarla visualmente. Se creó una clase específica para manejar la flecha y un algoritmo adicional en la clase controladora del tour. La primera función de este algoritmo se utiliza para encontrar los siguientes cruces en la ruta, evitando que la flecha apunte a cada punto intermedio y enfocándose únicamente en los cruces. La segunda función permite que el siguiente punto de referencia se actualice antes de llegar al punto actual. Ambas funciones están disponibles en el Anexo D. A continuación, se presenta la Figura 7.16, la cual muestra la aplicación con la línea original (Imagen A) y con la flecha implementada (Imagen B).



(a) App utilizando línea como guía

(b) App utilizando flecha como guía

Figura 7.16: Comparación entre la App utilizando línea (Imagen A) y utilizando flecha (Imagen B) como guía.

## 7.4. Integración de la interfaz de usuario

La integración de la interfaz de usuario y la lógica del flujo resultaron en una experiencia interactiva y fluida para los usuarios del tour. El flujo general de la aplicación, resultó de la siguiente manera:

1. Al iniciar la aplicación, se despliega una lista de rutas disponibles. El usuario selecciona una ruta y presiona el botón para comenzar el tour.
2. En este punto, las cámaras y el *pathfinding* están deshabilitados. El usuario debe escanear el código QR correspondiente al punto de inicio del tour.
3. Una vez que se escanea el código QR, se habilita el *pathfinding*, y se muestra una flecha que guía al usuario hacia el siguiente objetivo.

4. Al acercarse a 1.50 metros de un objetivo, se despliega un botón que el usuario puede presionar para recalcular el *pathfinding* hacia el siguiente objetivo.
5. Al alcanzar el último objetivo de un nivel, la aplicación desactiva la flecha de navegación y muestra un mensaje indicando hacia qué nivel debe dirigirse el usuario, solicitando que escaneé el código QR correspondiente al nuevo nivel.
6. Este proceso se repite hasta llegar al último objetivo del último nivel, donde la aplicación desactiva nuevamente la flecha y muestra un mensaje indicando que el tour ha finalizado. También aparece un botón que permite al usuario reiniciar la aplicación y volver al inicio.

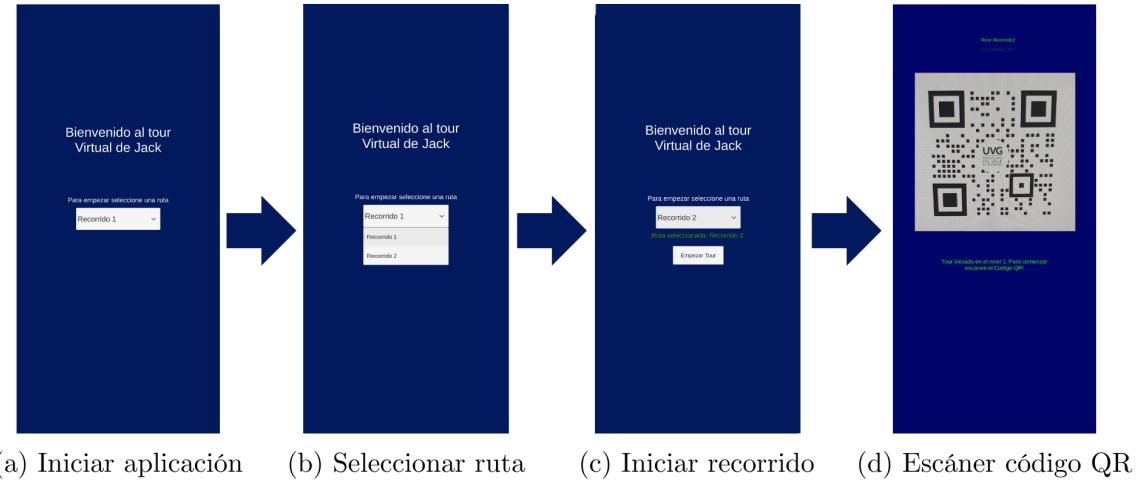


Figura 7.17: Flujo de la aplicación dentro del menú de inicio. Esta figura incluye las imágenes A, B, C y D, que representan diferentes etapas del proceso.

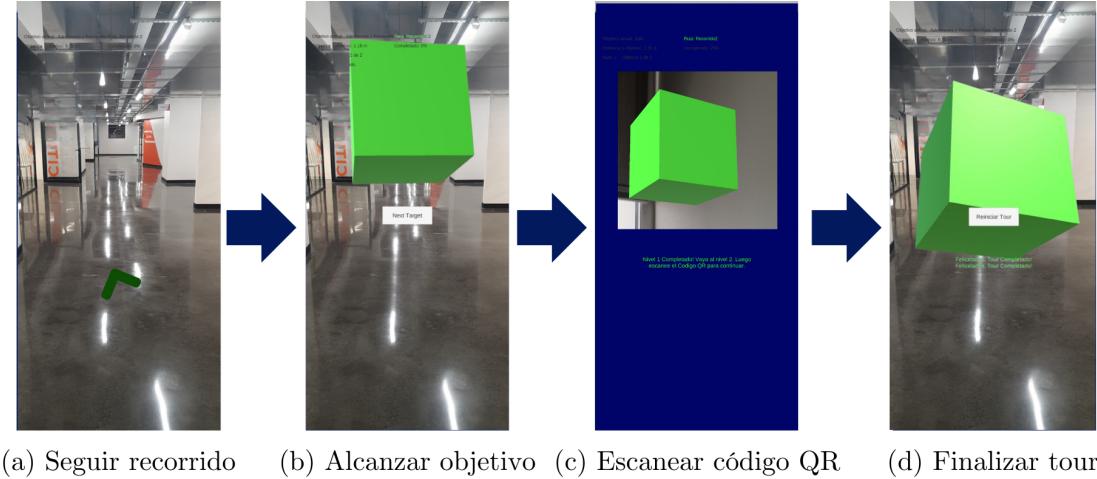


Figura 7.18: Flujo de la aplicación dentro del recorrido. Esta figura incluye las imágenes A, B, C y D, que representan diferentes etapas del proceso.

Los elementos de la interfaz que muestran información en tiempo real, como la distancia al próximo objetivo, el nombre del objetivo, el nivel actual y el porcentaje completado del tour, funcionaron adecuadamente. La vinculación de estos elementos con sus respectivas variables en el código permitió que los datos mostrados se actualizaran sin retrasos. Se validó que el porcentaje de tour completado se calculase correctamente y reflejase el progreso del usuario a lo largo del recorrido.

### 7.4.1. Integración de módulos adicionales

Seguidamente, se trabajó en la integración de los otros dos módulos del megaproyecto, los cuales permitieron obtener una aplicación completamente funcional, con una interfaz visualmente atractiva y con la funcionalidad adicional de agregar minijuegos al tour.

Para poder integrar el módulo de *Desarrollo de UX/UI en Aplicación de Recorridos Virtuales con Unity*, se tomó que base el flujo original, pero se hicieron mejoras. Para esto se utilizó una tecnología que ofrece Unity, llamada UI Toolkit, la cual permite crear elementos de UI con más facilidad y visualmente más atractivas. Habiendo completado estos elementos, se implementaron con este módulo, utilizando la lógica establecida para evitar errores.

Luego se hizo la integración con el módulo *Desarrollo de minijuegos para el recorrido virtual de la Universidad del Valle de Guatemala*. Esta integración consistió en utilizar la lógica de acercarse a un objetivo, ya que los tres minijuegos solo serían visibles para el usuario al acercarse a ciertos puntos de interés. Luego se tuvo que implementar una lógica para poder cambiar de escenas (entre la escena del tour y la escena de los juegos), sin afectar la funcionalidad del tour. Esto consistió en poder cargar las diferentes escenas en lugar de reemplazarlas por completo, lo que permite mantener el estado del tour y evita tener que empezar desde nuevo cada vez que se ingrese a un minijuego.

A continuación se muestran unas imágenes que muestran el resultado de la aplicación con los tres módulos combinados.

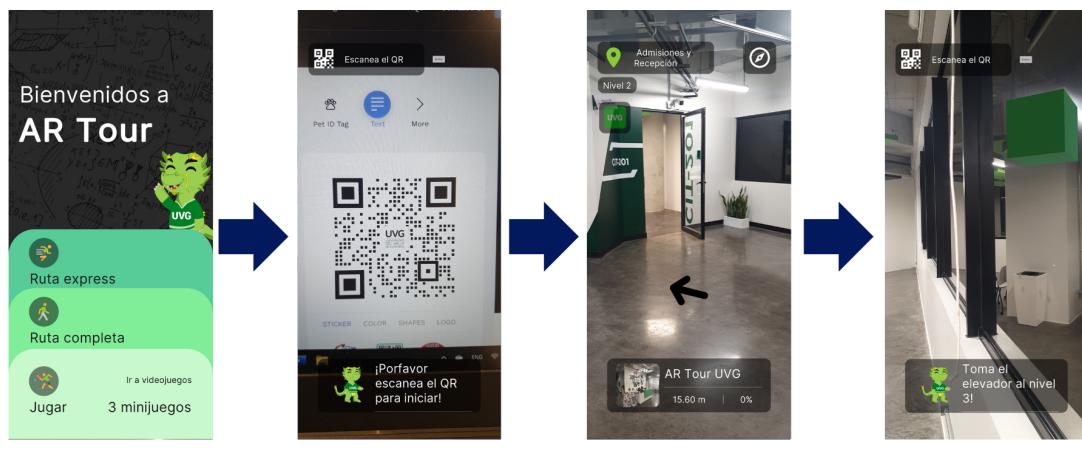
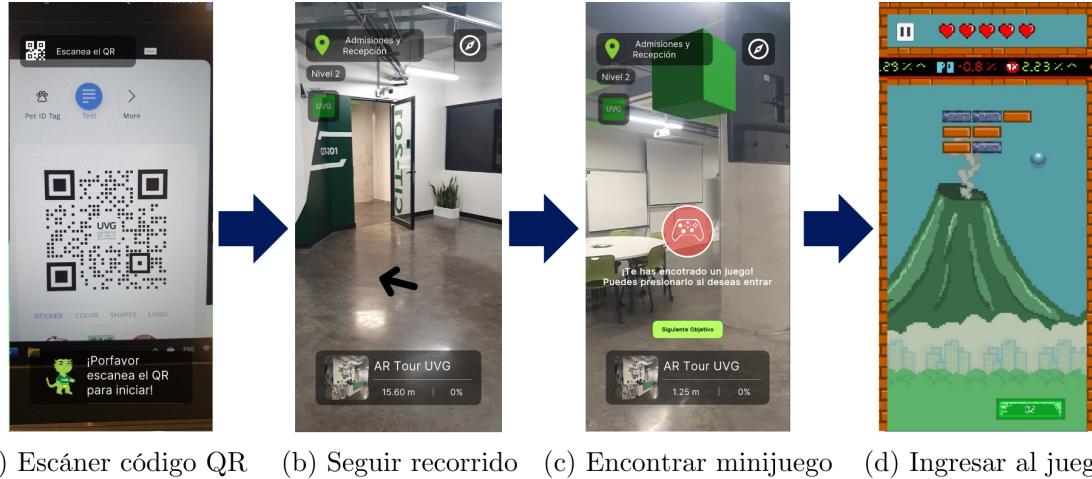


Figura 7.19: Flujo inicial dentro de la aplicación actualizada. Esta figura incluye las imágenes A, B, C y D, que representan diferentes etapas del proceso.



(a) Escáner código QR (b) Seguir recorrido (c) Encontrar minijuego (d) Ingresar al juego

Figura 7.20: Flujo de minijuegos dentro de la aplicación actualizada. Esta figura incluye las imágenes A, B, C y D, que representan diferentes etapas del proceso.

## 7.5. Configuración de *plug-ins* necesarios

Para poder permitir descargar la aplicación tanto en iOS como en Android, se configuraron los plug-ins para cada uno de los sistemas operativos; ARCore y ARKit, respectivamente. La configuración, aunque exitosa, al principio dio un par de problemas. Durante todo el proceso de desarrollo, se utilizó un dispositivo Android, por lo cual se utilizaron las configuraciones necesarias para este sistema operativo. Cuando, finalmente, se quiso intentar utilizar un dispositivo iOS, la descarga no dio ningún problema, pero a la hora de inicializar la aplicación, esta no detectaba ninguna cámara y no se podía habilitar desde los ajustes del teléfono. Esto significa, que la aplicación simplemente no funcionaba en iOS. Pero al darle otra revisada a los ajustes dentro de Unity y configurarlos para las necesidades de un teléfono de este sistema operativo; el problema se logró solucionar. Ahora la aplicación puede ser descargada y utilizada tanto en iOS como en Android.

## 7.6. Pruebas internas y pruebas de usuario

Para poder asegurar que la aplicación funcionara de manera óptima y con la menor cantidad de errores posibles, se realizaron pruebas, tanto internas como de usuario. Las pruebas internas se realizaron a lo largo de todo el proceso de desarrollo, lo cual permitió detectar fallas y errores y de ese modo implementar las mejoras inmediatamente. Las pruebas de usuario se realizaron al finalizar la aplicación, lo que permitió que los usuarios interactuaran con la aplicación completa, y de esa manera poder proporcionar realimentación útil para poder hacer cambios de ser necesario.

### 7.6.1. Pruebas de los modelos 3D y el algoritmo *pathfinding*

Después de completar los modelos y realizar las correcciones necesarias en las mediciones manuales, se llevaron a cabo pruebas para evaluar el error de medición de los modelos dentro de la aplicación. Estas pruebas consistieron en recorrer la distancia entre los elevadores principales y los elevadores secundarios en cada nivel del edificio. Se utilizó el algoritmo de Unity, para ver si era capaz de guiar al usuario a través de estos recorridos de prueba. La Tabla 7.8 muestra los resultados de estas mediciones, que reflejan el error de medición promedio en metros para cada nivel. El promedio

general del error de medición es de **1.64 m.**

Tabla 7.8: Resultados del error de medición promedio en metros para cada nivel del edificio.

Nivel	Error de Medición (m)
Nivel 2	1.42
Nivel 1	1.33
Nivel 3	1.60
Nivel 4	1.60
Nivel 6	1.40
Nivel 7	1.50

Es importante mencionar que el promedio de cada nivel se tomó después de realizar tres pruebas exitosas. Una prueba exitosa se define como una prueba en la que todo dentro de la aplicación funcionó a la perfección. Esto incluye: renderizar los modelos dentro de la aplicación sin problemas, escanear el código QR en la posición inicial exacta para que se alineara bien el modelo con la realidad, que el teléfono no se bloqueara y que la aplicación no fallara bajo ninguna circunstancia. Por ende, la tabla anterior muestra los resultados de 18 pruebas; sin tomar en cuenta las 9 pruebas que fallaron. Un punto interesante es que en algunos niveles, la aplicación fallaba más que en otros. Por ejemplo, en los niveles 4 y 6, la aplicación fallaba más a menudo que en los demás niveles y las fallas siempre ocurrían a la hora de cruzar el pasillo 8 (Figura 7.4) y el pasillo 5 (Figura 7.5) de cada nivel, respectivamente. Por otro lado, es importante mencionar que el algoritmo de *pathfinding* sí funcionó a la perfección, ya que esté dibujada correctamente la ruta desde el punto inicial al punto final y se actualizaba en tiempo real a medida que el usuario avanzaba dentro de la ruta.

### 7.6.2. Pruebas del código y aplicación

Durante todo el proceso de desarrollo de la aplicación, se realizaban pruebas de estrés para asegurarse que la aplicación y sus funcionalidades funcionaran según lo previsto. Para esto se realizó un proyecto aparte, con sus modelos correspondientes, para tener un ambiente de prueba y poder trabajar y probar el proyecto sin tener que estar en el CIT.

Con el ambiente de prueba creado, se hicieron todos los cambios y mejoras de la aplicación. Antes de implementar la funcionalidad en el repositorio de desarrollo, se implementaba y se probaba en el ambiente de prueba para ver si la funcionalidad funcionaba como debía. Si la funcionalidad no funcionaba como se esperaba, se realizaban cambios y se repetía el proceso de prueba hasta que funcionara de manera óptima. Media vez la funcionalidad funcionaba bien, esta se implementaba al repositorio de desarrollo. Este fue el proceso que se siguió desde el principio del desarrollo hasta el final, lo cual resultó exitoso, ya que nunca se tuvo que hacer pruebas en el repositorio de desarrollo, evitando errores y complicaciones en este.

En base a este proceso, de prueba y mejora, se lograron implementar de manera exitosa las siguientes funcionalidades:

- **Pruebas de cambio de objetivos:** Se realizaron varias pruebas con el botón de siguiente objetivo. Las pruebas que se realizaron, aseguraron que el botón se desplegara solamente cuando el usuario estuviera a 1.5 metros del objetivo actual y que a la hora de darle clic al botón, la aplicación recalculara la ruta, desde la posición actual, hacia el nuevo objetivo.
- **Pruebas de escaneo de códigos QR:** Asimismo, se realizaron varias pruebas con el escaneo del QR. Primero, se realizaron las pruebas para ver si a la hora de escanear el código QR, este sí reposicionaba al usuario en el siguiente nivel. Las pruebas eran exitosas, pero para que se acoplaran con el proyecto, se tuvo que hacer más pruebas para que el código QR también

hiciera que se recalculara la ruta hacia el nuevo objetivo del nuevo nivel. Finalmente, se hicieron pruebas para que el algoritmo de *pathfinding* se activara, al escanear el código QR inicial.

- **Pruebas de selección de rutas:** Además, se realizaron varias pruebas con el botón de selección de rutas. Se realizaron varias pruebas para asegurarse que este botón seleccionara el recorrido elegido por el usuario, y que configurara toda la aplicación para que esta funcionara en base a los niveles y objetivos dentro de dicho recorrido.
- **Pruebas de botón de reinicio:** Se realizaron varias pruebas con el botón de reinicio. Estas pruebas aseguraron que el botón reiniciaría la aplicación de manera correcta, reseteando todos los cálculos y textos del tour pasado y llevando al usuario de nuevo a la página principal para que se pudiera iniciar un nuevo recorrido.
- **Pruebas de textos y otros elementos UI:** Finalmente, se realizaron varias pruebas para asegurarse que los elementos UI estuvieran desplegando de manera correcta la información del tour y que los cálculos se estuvieran haciendo de manera correcta.

Al tener todos los cambios en el ambiente de desarrollo, se realizaron pruebas de simulación, para verificar que la aplicación funcionara como se esperaba. Esto se realizó dentro de Unity antes de descargar la versión final. Estas pruebas de simulación fueron bastante útiles, ya que funcionaban como unas pruebas teóricas, lo que significa que así como se comportaba la aplicación en estas pruebas, en teoría así debería de funcionar en la vida real.

Al asegurarse que la aplicación funcionara bien dentro de la simulación, era hora de hacer pruebas con la aplicación, dentro del CIT. Al hacer las pruebas, algunas fueron positivas y otras no tanto. Los aspectos positivos fueron que el flujo de la aplicación funcionaba a la perfección: se seleccionaba la ruta correcta para el tour, el escaneo de códigos QR fue exitoso, los botones de siguiente nivel y reiniciar aplicación funcionaron como se esperaba y los textos informativos desplegaban siempre la información correcta. Por otro lado, similar a lo que sucedió con las pruebas de los modelos 3D; la aplicación, resulta, era bastante propensa a errores, ya que había varios factores que la hacían fallar. Algunos de estos factores incluyen, el rendimiento de la aplicación, el lugar donde se escaneó el código QR y el comportamiento de los modelos, que por alguna razón, a veces parecían moverse, afectando la ruta del tour (este comportamiento de nuevo se vio más a menudo en los niveles 4 y 6).

### 7.6.3. Pruebas de rendimiento

Uno de los factores que afectaba la aplicación, como se menciona anteriormente, fue el rendimiento de la aplicación. Resulta que este era tan bajo, debido a al consumo excesivo de RAM, el cual llegaba a consumir hasta 3 GB de la memoria, lo que provocaba que la aplicación se cerrara solo unos segundos después de iniciarse. Esto se descubrió utilizando una herramienta que medía el rendimiento de la aplicación y resulta que lo que causaba este consumo excesivo era la función que escaneaba los códigos QR (disponible en el anexo D.3). Resulta que la función incluía el Código 7.7, el cual creaba, en todo momento, una nueva textura basada en una imagen capturada con la cámara, y no la destruía jamás; haciendo que el consumo de la RAM aumentara con cada momento que pasaba. Para poder solucionar esto, se tuvo que modificar la función, como se muestra en el Código 7.8 para que esta ya no creara una nueva textura en cada momento, sino que verificara si ya existía una y en caso de ser cierto, la destruía antes de crear una nueva (función completa disponible en el anexo D.5). Esto permitió que el rendimiento de la aplicación se mantuviera estable en todo momento, y el consumo de la RAM bajo de 3 GB a menos de 1 GB. Además, también se agregó una validación, para evitar que la cámara estuviera siempre escaneando por un código QR y que solo se habilitara cuando era necesaria.

```

1 cameraImageTexture = new Texture2D(
2     conversionParams.outputDimensions.x,
3     conversionParams.outputDimensions.y,
4     conversionParams.outputFormat,
5     false);
6
7 cameraImageTexture.LoadRawTextureData(buffer);
8 cameraImageTexture.Apply();
9 buffer.Dispose();

```

Código 7.7: Fragmento de código que crea una nueva textura en cada momento

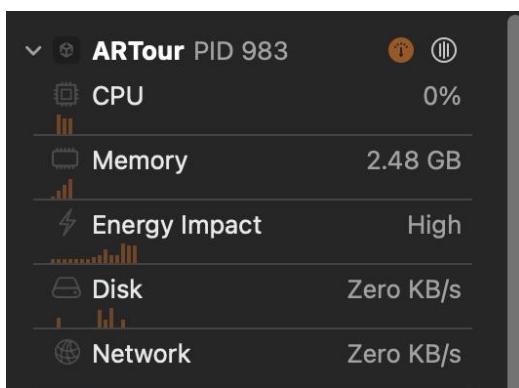
```

1 // Verificar si la textura existe y es del tamaño correcto
2 if (cameraImageTexture == null ||
3     cameraImageTexture.width != conversionParams.outputDimensions.x ||
4     cameraImageTexture.height != conversionParams.outputDimensions.y ||
5     cameraImageTexture.format != conversionParams.outputFormat)
6 {
7     // Destruir la textura anterior si existe
8     if (cameraImageTexture != null)
9     {
10         Destroy(cameraImageTexture);
11     }
12
13     // Crear una nueva instancia de la textura con los parámetros correctos
14     cameraImageTexture = new Texture2D(
15         conversionParams.outputDimensions.x,
16         conversionParams.outputDimensions.y,
17         conversionParams.outputFormat,
18         false);
19 }

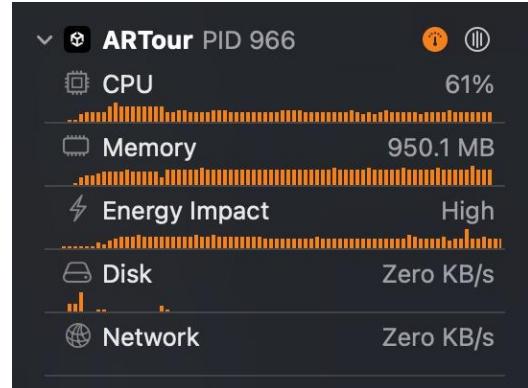
```

Código 7.8: Fragmento de código que crea una nueva textura después de eliminar la anterior

A continuación se presenta la Figura 7.21, la cual muestra la diferencia en el consumo de memoria RAM al utilizar dos fragmentos de código distintos. La Imagen A representa el consumo de memoria cuando se creaba una textura nueva en cada iteración, mientras que la Imagen B representa el consumo al implementar la lógica de eliminar la textura anterior antes de crear una nueva.



(a) Consumo utilizando la función original



(b) Consumo utilizando la función modificada

Figura 7.21: Comparación del consumo entre la función original (Imagen A) y la función modificada (Imagen B).

#### 7.6.4. Pruebas de usuario

Para obtener resultados exitosos de las pruebas de usuario, se creó un formulario de Google con 10 preguntas sobre la aplicación, enfocadas en la interfaz de usuario, la intuición de la aplicación y la precisión del algoritmo de pathfinding. Este formulario se presentó a 11 estudiantes de distintas carreras, como ingeniería industrial, ingeniería en alimentos, ingeniería en ciencias de la computación y producción musical, después de realizar un recorrido de prueba. Este recorrido incluía solo los dos primeros niveles del tour, ya que contenía todas las funcionalidades de la aplicación sin requerir que el usuario recorriera los seis niveles completos.

Para poder obtener resultados honestos, el proceso que se siguió fue el siguiente: se seleccionaron varios amigos y estudiantes al azar de la UVG y se les dio el teléfono. No se les explicó nada sobre como funcionaba la aplicación, solo se les mencionó que era un tour virtual. Dado esto, el estudiante, por sí solo, tuvo que ver de que se trataba la aplicación y como era que funcionaba. Al finalizar las pruebas, el estudiante llenaba el formulario en base a su experiencia con la aplicación. Estos fueron los comentarios que dejaron los usuarios después de utilizar la aplicación:

- “Mejorar los puntos de partida usados en el QR.”
- “Indicar que al salir del elevador se debe volver a escanear el código QR.”
- “Establecer puntos específicos para escanear el QR.”
- “Fue bastante fácil seguir la guía y los puntos, pero sí es importante definir los puntos de los QR para que no se desfase.”
- “La aplicación es bastante intuitiva y además considero que es un tour práctico que cualquiera puede ser capaz de realizarlo con solo un celular móvil.”
- “Excelente aplicación para navegar y conocer la Universidad del Valle de Guatemala.”
- “Establecer puntos específicos en los que se pueda leer el QR.”
- “Muy buena aplicación!”
- “En general buena aplicación. Pero a veces falla la precisión y al principio del objetivo es difícil saber para dónde ir.”
- “Habilitar ambas opciones de flecha y línea guía.”

A continuación se muestra todas las gráficas de la encuesta realizada para poder validar la satisfacción de los usuarios con la aplicación presentada.

¿Considera que la aplicación es suficientemente intuitiva?

11 respuestas

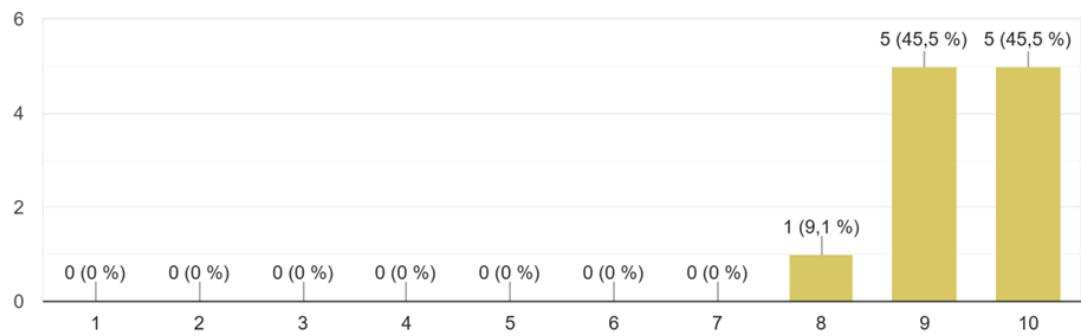


Figura 7.22: Opinión de usuarios sobre la intuitividad de la aplicación

¿Cómo describiría su experiencia general durante el recorrido virtual con el AR Tour?

11 respuestas

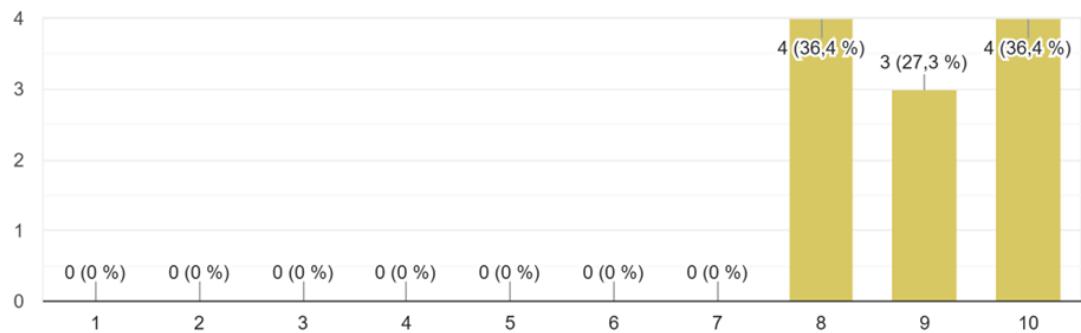


Figura 7.23: Opinión de usuarios sobre la experiencia general de la aplicación

¿Qué tan precisa le pareció la indicación proporcionada por el sistema de guía del recorrido?

11 respuestas

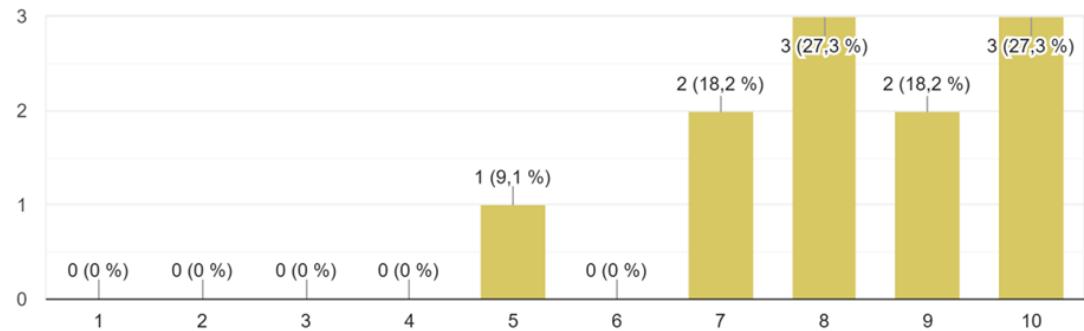


Figura 7.24: Opinión de usuarios sobre la precisión de la aplicación

En cuanto a las herramientas de navegación proporcionadas (línea de recorrido o flecha), ¿cuál considera que fue más efectiva para guiar su experiencia?

11 respuestas

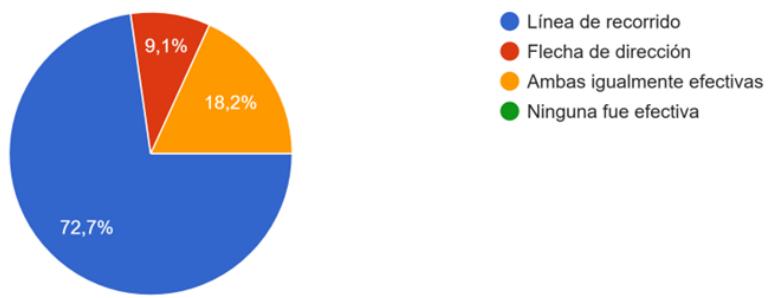


Figura 7.25: Opinión de usuarios sobre la herramienta de navegación

¿Le resultó sencillo ajustar su posición AR con el código QR?

11 respuestas

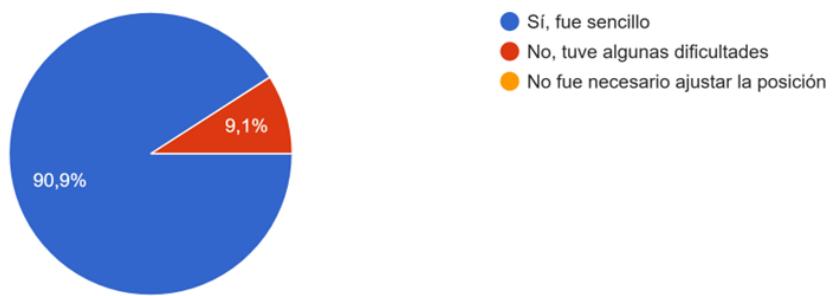


Figura 7.26: Opinión de usuarios sobre el uso de códigos QR

¿Hubo algún retraso en la detección de los puntos de interés del entorno físico?

11 respuestas

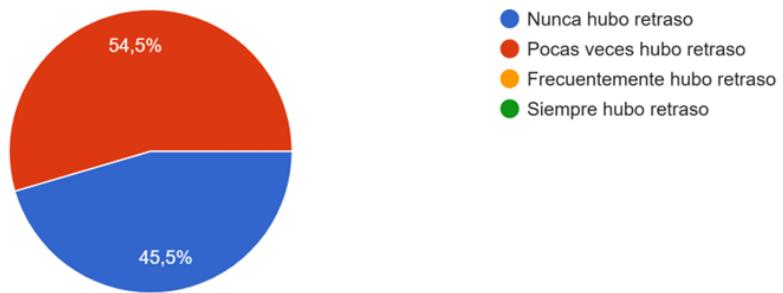


Figura 7.27: Experiencia de usuarios con retrasos en la aplicación

### 7.6.5. Pruebas de usuario segunda ronda

Para evaluar las mejoras realizadas a partir de la primera ronda y obtener una perspectiva más variada, se realizó una segunda ronda de pruebas de usuario. En esta ocasión, participaron 12 personas con características distintas a las de la primera ronda: 7 adultos (todos padres de familia), 4 jóvenes y 1 niño, ninguno de los cuales pertenecía a la Universidad del Valle de Guatemala. Este grupo permitió recopilar opiniones de personas de diferentes edades y contextos, ampliando la diversidad de las evaluaciones.

La metodología utilizada fue la misma que en la primera ronda. A cada participante se le proporcionó un teléfono con la aplicación instalada, sin dar instrucciones sobre su funcionamiento, permitiendo que descubrieran por sí mismos de qué se trataba y cómo usarla. Una vez completado el recorrido de prueba, se les pidió que respondieran un formulario de Google con las mismas 10 preguntas empleadas en la primera ronda. Para mantener los resultados organizados, las respuestas de esta segunda ronda se recopilaron en un formulario separado, facilitando así la comparación entre ambas etapas. Estos fueron los comentarios que dejaron los usuarios de la segunda ronda después de utilizar la aplicación:

- “Excelente aplicación pero necesita mejoras en el sistema de guía para ser mas preciso.”
- “Muy excelente idea y aplicación. Pero si necesita algunas mejoras. Se trabaja a veces.”
- “necesita ser mas precisa la interfaz para saber donde esta el nuevo QR al cambiar el piso.”
- “Buena aplicación! Solo le faltan pocos detalles para refinar.”
- “Los códigos QR en impresos y colocados sirvieron bastante.”
- “preferiría que se pudiera pausar el recorrido porque en medio del recorrido entro una llamada al dispositivo y la aplicación tuvo un poco de dificultades.”
- “Me encanto la aplicación.”
- “necesita mejorar como dar una introducción para saber como se usa y en realidad porque esta hecha la aplicación.”
- “Tuve algunos problemas con las fallas y la precisión de la guía.”
- “quisiera poder ver todos los puntos que voy a visitar antes de comenzar el recorrido virtual con esta aplicación.”
- “buena app”
- “La idea está excelente.”

A continuación se muestra todas las gráficas de la encuesta realizada para poder validar la satisfacción de los usuarios con la aplicación presentada.

¿Considera que la aplicación es suficientemente intuitiva?

12 respuestas

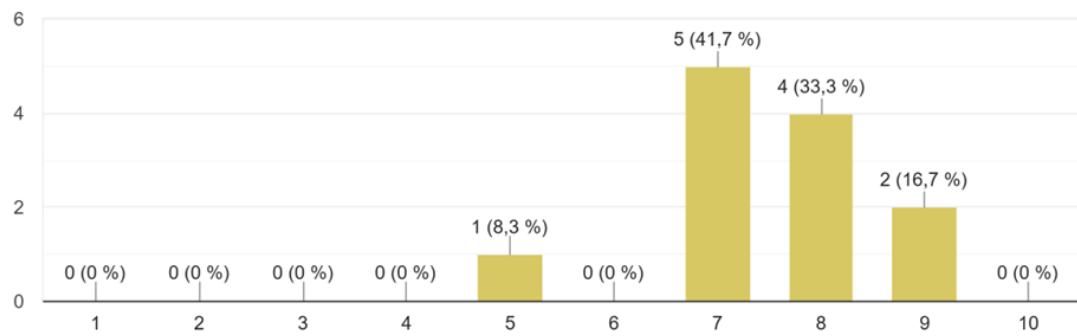


Figura 7.28: Opinión de usuarios sobre la intuitividad de la aplicación (ronda 2)

¿Cómo describiría su experiencia general durante el recorrido virtual con el AR Tour?

12 respuestas

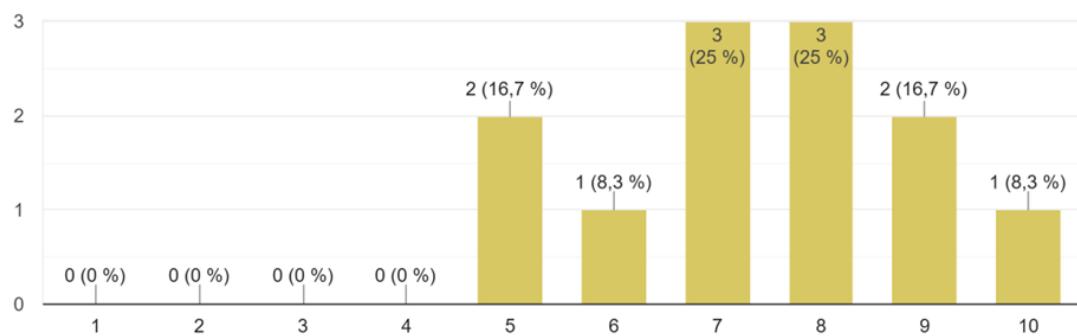


Figura 7.29: Opinión de usuarios sobre la experiencia general de la aplicación (ronda 2)

¿Qué tan precisa le pareció la indicación proporcionada por el sistema de guía del recorrido?

12 respuestas

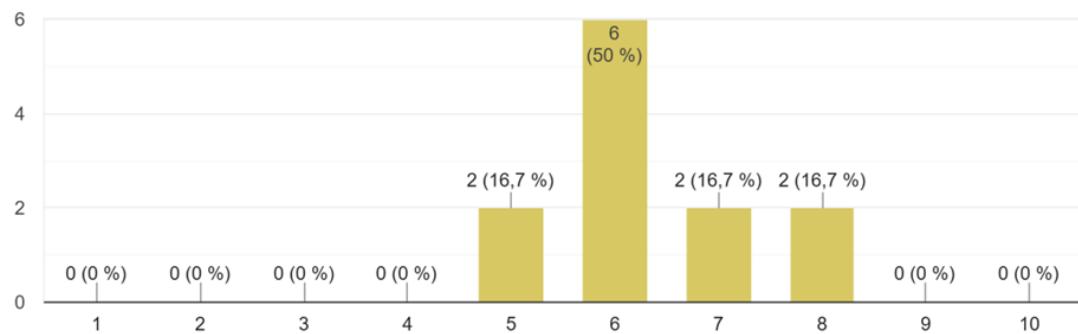


Figura 7.30: Opinión de usuarios sobre la precisión de la aplicación (ronda 2)

En cuanto a las herramientas de navegación proporcionadas (línea de recorrido o flecha), ¿cuál considera que fue más efectiva para guiar su experiencia?

12 respuestas

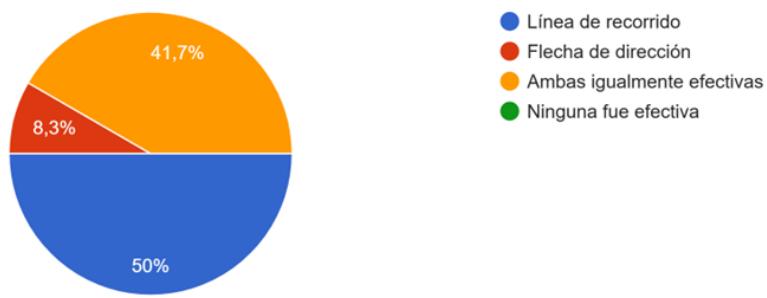


Figura 7.31: Opinión de usuarios sobre la herramienta de navegación (ronda 2)

¿Le resultó sencillo ajustar su posición AR con el código QR?

12 respuestas

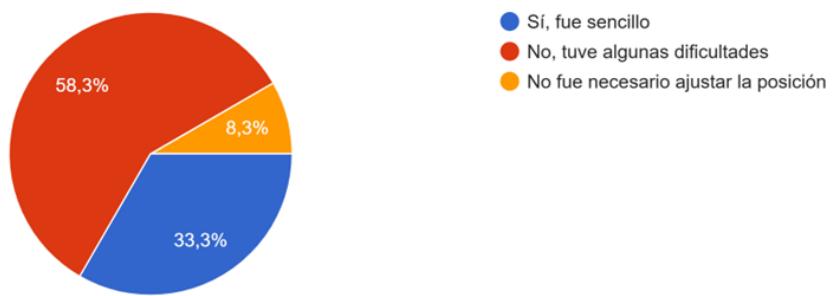


Figura 7.32: Opinión de usuarios sobre el uso de códigos QR (ronda 2)

¿Hubo algún retraso en la detección de los puntos de interés del entorno físico?

12 respuestas

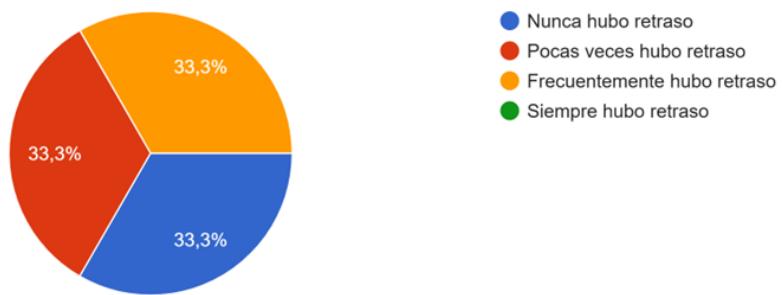


Figura 7.33: Experiencia de usuarios con retrasos en la aplicación (ronda 2)

# CAPÍTULO 8

---

## Discusión

---

Para poder entender mejor los resultados que se obtuvieron a lo largo del desarrollo del proyecto, es clave interpretarlos y analizarlos, ya que esto permite saber qué cosas salieron bien, que cosas no salieron como se esperaban y por qué.

El objetivo principal del proyecto fue Garantizar la congruencia y alineación precisa entre el mundo virtual y el real en la Aplicación de Recorridos Virtuales del (CIT) de la Universidad del Valle de Guatemala mediante modelado 3D y herramientas de navegación en Unity. Dicho objetivo se cumplió, ya que se lograron modelar los seis niveles que se utilizaron para el tour y también se logró implementar un algoritmo de *pathfinding* que fuera capaz de guiar a los usuarios a través del CIT, en la vida real. A pesar de que el objetivo se cumplió y varios resultados fueron positivos y fueron los esperados, es importante mencionar que no todos los resultados fueron los esperados. Esto significa que las expectativas no se cumplieron del todo, ya que a pesar de que el flujo de la aplicación funcionaba de forma correcta, había varios factores que resultaban en el problema del registro, el cual indica que los objetos en el mundo real y los objetos virtuales tienen que estar precisamente alineados [9]. Si los objetos en ambos mundos no están alineados, ambos mundos no pueden coexistir, y esto afectaba negativamente a los modelos dentro de la aplicación, lo que impactaba la experiencia del usuario.

### 8.1. Modelado del CIT utilizando Blender

Para empezar a entender por qué la aplicación se comporta del modo en que lo hace, se empezarán analizando los resultados del modelado 3D de los niveles del CIT. Primero, se puede observar que el nivel 5 del CIT no forma parte del tour y esto es porque en este nivel no se encuentra ningún punto de interés para mostrar, por lo que se decidió obviar este nivel. Luego, observando la Tabla 7.1, se puede ver que cada uno de los planos que fueron proporcionados por la universidad, tienen un cierto porcentaje de error, que van desde 2.59 % hasta 8.37 % y promedian un 4.87 % de porcentaje de error general. Esto significa que los planos, no eran congruentes con el edificio en la vida real, lo cual pudo haber afectado la aplicación de una manera muy grave.

Se puede pensar que un porcentaje de error promedio de 4.87 % no es lo suficientemente alto para poder afectar la aplicación, pero en realidad sí lo es, ya que este promedio resulta de los porcentajes de error de cada uno de los pasillos, y estos van desde un 0 % hasta un 51.82 %. Es cierto que el 51.82 % es un punto atípico, pero se puede observar que en cada una de las seis tablas que contienen

las medidas de los pasillos (desde la Tabla 7.2 hasta la Tabla 7.7), se encuentra un porcentaje de error que es equivalente al 34.69 %.

Este porcentaje de error se encuentra en el pasillo de los elevadores principales de cada uno de los niveles, y si se mira en términos de medidas, este error equivale a una diferencia de 1.7 metros, lo que es más que suficiente para afectar a la aplicación de una manera negativa. La razón por la cual esta diferencia podría afectar la aplicación, es debido al problema del registro, ya que en este caso, una diferencia de 1.7 metros, puede desalinear todo el algoritmo de *pathfinding* y, por lo tanto, puede guiar a un usuario directamente hacia una pared, haciendo que se rompa la inmersión y arruinando la experiencia del usuario.

Por lo tanto, el proceso de corrección de los modelos, basado en la comparación con medidas reales (Tablas 7.2 a 7.7), fue crucial para ajustar la alineación y reducir los errores en los planos. Esto permitió minimizar el problema de desajuste en el registro, mejorando la precisión del algoritmo de *pathfinding* y reduciendo el riesgo de que los usuarios fueran guiados hacia una pared, lo cual habría comprometido la inmersión y la experiencia general de navegación en la aplicación.

## 8.2. Construcción la escena dentro de Unity

La construcción de la escena dentro de Unity fue el objetivo específico más simple de cumplir, ya que consistía solo de importar los modelos 3D a Unity, configurar los elementos necesarios para obtener una experiencia de realidad aumentada y definir todo el ambiente, con el cual se interactúa dentro de la aplicación.

En la Figura 7.10 se puede observar como se ordenaron los seis modelos dentro de Unity, la razón por la que están lado a lado y no uno arriba del otro, es simplemente por comodidad. Si los modelos se hubieran colocado, uno encima del otro, para simular un edificio, esto no hubiera funcionado de esa manera, ya que no hay conexión entre los modelos. Por lo tanto, esa manera de ordenar los modelos solo hubiera complicado la manera de definir el resto del ambiente. Al estar todos lado a lado, se puede editar cada uno de los ambientes, independientemente y de una manera más sencilla.

En la Figura 7.11 se pueden observar varios elementos, los cuales se encargan de la interacción con el entorno, habilitar la cámara del dispositivo y la movilidad dentro de la escena. En resumen, esta imagen muestra la configuración que se agregó al proyecto de Unity para habilitar el funcionamiento de realidad aumentada dentro de la aplicación.

El posicionamiento de los elementos virtuales y su correspondencia con ubicaciones en la vida real fue una consideración fundamental para optimizar la experiencia del usuario. Aunque en la Figura 7.10 no se pueden apreciar estos detalles, la Figura 7.12 detalla cómo cada objetivo y código QR fue cuidadosamente posicionado en sus niveles correspondientes. La ubicación de los objetivos refleja los puntos de interés reales dentro del CIT, y se ajustaron para que su posición virtual coincidiera lo más posible con la física, minimizando así errores de registro. Los códigos QR, por su parte, se situaron estratégicamente cerca de los elevadores, facilitando a los usuarios la reanudación del tour al cambiar de nivel. Finalmente, para mejorar la precisión de la aplicación, estos códigos QR se instalaron físicamente en ubicaciones equivalentes en el edificio, reduciendo la posibilidad de desalineación entre el entorno virtual y el real.

### 8.3. Desarrollo del algoritmo de *pathfinding* y estructura del código

Para el desarrollo del algoritmo de *pathfinding*, aunque hubo un problema, es seguro decir que el objetivo se pudo cumplir con éxito. El objetivo inicial de esta fase era poder desarrollar un algoritmo de *pathfinding* basado en el algoritmo A\*, desde cero, para poder utilizar adentro de la aplicación. Como se puede ver en los fragmentos de código 7.1 y D.1, los cuales muestran el código de la clase nodo y la función principal del algoritmo A\*, respectivamente, si se logró desarrollar este algoritmo, aunque no con resultados positivos. La Figura 7.13, muestra el ambiente de desarrollo en el cual se codificó todo el proyecto antes de pasarlo al ambiente principal y en la Imagen A se puede ver que en la esquina superior derecha se encuentra una línea verde. Esta línea verde es el resultado del algoritmo desarrollado, y si se observa bien, se puede notar que en realidad la ruta no está del todo mal, ya que esta si dibuja la línea correcta para poder llegar de la esfera azul (en la esquina superior derecha del modelo de prueba) hasta uno de los cubos verdes que se encuentran en el pasillo. El problema es que la línea está desplazada, y no conecta el punto de inicio con el objetivo. Este comportamiento se debe a que el algoritmo desarrollado desde, no tiene conexión ninguna con el NavMesh que se utiliza para definir las áreas navegables dentro del tour. Por lo tanto, se tuvo que crear escribir el Código D.2, el cual describe una función que crea la cuadrícula sobre la cual se calcula la ruta. El problema fue que nunca se logró alinear la cuadrícula con el NavMesh, resultando en que la ruta siempre estuviera desplazada.

Por lo tanto, se optó por utilizar un método más sencillo, el cual consistió en implementar el algoritmo *pathfinding* que ya trae el NavMesh por defecto. Esta implementación simplificó todo el proceso, ya que por ser una función propia del NavMesh, este toma todo el NavMesh como la cuadrícula sobre la cual calcula la ruta. Esto resulta en una ruta precisa que conectara el punto de inicio con el objetivo final. Como se puede observar en la Imagen B de la Figura 7.13, se muestra una línea roja y no una línea verde. Se puede observar que en este caso, la línea sí conecta directamente la esfera azul con el cubo verde. Adicionalmente, se puede observar que aunque las dos líneas siguen el mismo patrón, la línea roja parece hacer un par de cruces extra. La línea verde nunca sigue esta ruta exacta, y esto es por lo mismo que la línea verde nunca logró alinearse con el NavMesh, resultando en una ruta imprecisa. Por lo tanto, se puede concluir que al utilizar este segundo método, se logró resolver por completo el problema que presentaba el método anterior, y a su vez se obtuvo un algoritmo de *pathfinding* completamente funcional.

La definición de las rutas y los objetivos dentro del tour parece como un paso simple, a comparación del desarrollo del algoritmo de *pathfinding*, pero la verdad es que su implementación fue más compleja de lo que parece. Para empezar se empezó estructurando el código en base a tres clases distintas: objetivos, niveles y rutas. Para asegurar que se implementaran con éxito, se implementó una por una, empezando con la más sencilla, los objetivos y terminando con la más complicada, las rutas. Como se puede observar en la Figura 7.14, esta implementación tuvo un resultado exitoso, ya que se pudieron definir todas las rutas disponibles, con sus niveles correspondientes y a su vez, sus objetivos correspondientes. La razón por la cual se decidió estructurar la aplicación de este modo, es porque la universidad cuenta con múltiples puntos de interés, y las diferentes rutas se diferencian por la cantidad de puntos de interés que contienen. Por lo tanto, la aplicación se estructuró de un modo en el cual pudiera conservar esta misma lógica y permitir que los usuarios pudieran escoger entre diferentes rutas.

Seguidamente, se puede observar la Figura 7.15, la cual muestra el ejemplo de como es que se ven todos los códigos QR que se colocaron dentro del CIT. Como se mencionó anteriormente, a la hora de hacer los modelos 3D, no se hizo ninguna conexión entre los niveles, lo que significa que cada modelo es un elemento independiente. En la escena dentro de Unity, por lo tanto, los niveles nunca se conectan entre sí, ya que conectarlos por medio de las escaleras o elevadores dentro de Unity significa un nivel de complicación más alto. Por lo tanto, se implementó la lógica para poder utilizar códigos QR, los cuales permiten reorientar al usuario en una nueva posición y desde esa posición

poder continuar con el tour. Esta lógica funciona por algunas razones, primero, simplifica el proceso y evita errores. La alternativa era permitir que el usuario seleccionara, a través del código, en cuál de los cuatro elevadores había subido. Con esta información, la aplicación recalcularía la posición del usuario en el nuevo nivel, comenzando desde el elevador elegido. Sin embargo, esta opción requería enumerar los elevadores y asegurarse de que los usuarios ingresaran correctamente su elección, lo que añadía un paso adicional en el proceso de navegación dentro de la aplicación.

Por lo tanto, una opción más viable fue la de implementar códigos QR y colocarlos de una manera estratégica. Con esta opción, al finalizar un nivel, la aplicación le indica al usuario a qué nivel debe ir, y el usuario se puede subir a cualquier elevador sin afectar nada de la ruta. Al salir del elevador, el usuario puede ver en la aplicación que debe de escanear el código QR del respectivo nivel, y gracias a la posición estratégica, el código QR estará afuera de los elevadores y será fácil de identificar. Una vez escaneado el código QR, la aplicación recalculará la ruta, desde ese mismo punto. Pero esta implementación también es propensa a errores, ya que de no estar los códigos QR en el lugar correcto, o no escanearse de la manera correcta, el problema del registro ocurrirá. Esto sucede porque la reorientación no logra alinear el modelo virtual con el mundo real. Otro error que puede suceder, es que dependiendo del dispositivo, la aplicación tarda cierto tiempo en reorientar al usuario, y en este pequeño momento, si el usuario se mueve, la reorientación no será exitosa, presentando nuevamente el problema del registro. Por lo tanto, aunque la implementación resultó ser una implementación exitosa y eficiente, se debe de asegurar que los usuarios realicen este paso con cuidado, para asegurar que la recalibración de la posición del usuario sea exitosa, y por ende, obtener una buena experiencia de usuario.

La implementación de la flecha de navegación, fue otro de los cambios que complicó el proceso del desarrollo. Como se mencionó en la sección de resultados, este cambio implicó un cambio en la estructura del código en términos de la herramienta guía. Ya que la línea se dibujaba a lo largo de toda la ruta, no necesitaba más lógica para funcionar de manera correcta. Por otro lado, la flecha, la cual solo apunta hacia un punto, sí implicaba más lógica para que esta pudiera funcionar de manera correcta y no afectar la experiencia del usuario. En el Anexo D, se pueden encontrar las funciones que hicieron posible la implementación de la flecha. Estas dos funciones garantizan que la flecha funcione de manera correcta y con una transición más fluida.

Como se puede observar en la Figura 7.16, la implementación de ambas herramientas fue exitosa, ya que logran guiar correctamente al usuario a través del CIT. La diferencia y la razón por la cual se decidió utilizar la flecha, es simplemente debido a que esta es menos invasiva y por ende, mejora la interfaz de usuario, y la experiencia de usuario, ya que esta herramienta no desconcentra tanto al usuario.

## 8.4. Integración de la interfaz de usuario

A pesar de que este módulo dentro del megaproyecto no requería una interfaz de usuario, se integró una interfaz de prueba para permitir que el usuario fuera capaz de interactuar con la aplicación y de este modo poder probar todo el flujo y el funcionamiento completo. Esto fue útil, ya que permitió hacer pruebas con la aplicación durante el proceso del desarrollo. Además, la inclusión de todos los textos informativos sirven para darle más vida a la aplicación y resultan en una experiencia más interactiva e inmersiva para el usuario. Como se puede observar en las Figuras 7.17 y 7.18, esta integración, fue exitosa, ya que es una interfaz bastante amigable y un flujo bastante sencillo e intuitivo. A su vez, integrar una interfaz de prueba sobre el flujo de la aplicación, sirvió como base para el módulo de la interfaz de usuario, facilitando el proceso de dicho módulo. Asimismo, este paso facilitó la integración de ambos módulos, ya que la lógica era bastante similar.

Además, la implementación de los otros dos módulos fue exitosa, logrando una aplicación completa, funcional y con un diseño fácil e intuitivo de usar. En la Figura 7.19 se puede observar el nuevo

frontend, resultado del módulo *Desarrollo de UX/UI en Aplicación de Recorridos Virtuales con Unity*. Este figura muestra un frontend más minimalista y limpio, lo cual resultó en una aplicación más visualmente atractiva e intuitiva.

En la figura 7.20, se puede observar el resultado del módulo *Desarrollo de minijuegos para el recorrido virtual de la Universidad del Valle de Guatemala*, el cual logró diseñar los minijuegos para la aplicación. Esta figura muestra cómo se implementaron dichos minijuegos en el flujo y cómo no afecta de ninguna manera la aplicación. Asimismo, es un flujo bastante fácil de seguir y el diseño de los minijuegos es congruente con el diseño minimalista y llamativo del frontend.

## 8.5. Configuración de plug-ins necesarios

La aplicación fue originalmente planeada para que funcionara tanto en iOS como Android, y esta fue una de las razones por las cuales se eligió utilizar Unity para el desarrollo de este proyecto. Unity, con su marco de trabajo ARFoundation permite crear aplicaciones de realidad aumentada, y además ARFoundation permite instalar los plug-ins de ARCore y ARKit para permitir utilizar una aplicación de realidad aumentada en ambos sistemas operativos. Al ver los resultados, se puede concluir que se cumplió este objetivo con éxito, ya que se logró descargar y utilizar la aplicación en ambos sistemas operativos.

## 8.6. Pruebas internas y pruebas de usuario

En el desarrollo de cualquier proyecto, las pruebas, específicamente las pruebas de estrés y las pruebas de usuario, son fundamentales para asegurar un producto final funcional, y este proyecto no fue la excepción. Las pruebas, en este caso, permitieron identificar áreas de mejora, optimizar el rendimiento de la aplicación y garantizar una experiencia de usuario satisfactoria.

Las pruebas de estrés a lo largo del desarrollo de la aplicación, como se menciona en los resultados, fueron exitosas. Este proceso fue fundamental para poder probar cada una de las funcionalidades. Permitiendo encontrar fallas e identificar mejoras, las pruebas permitieron desarrollar una aplicación completamente funcional. Como se mencionó anteriormente, se utilizó un ambiente de desarrollo; paso importante para que se probaran todos los nuevos cambios en este ambiente, sin tener que arruinar nada de lo que ya estaba aprobado en el ambiente principal.

Las pruebas de usuario también fueron exitosas para detectar y corregir ciertos errores. Estas pruebas se realizaron con 11 usuarios de diferentes carreras, incluyendo ingeniería industrial, ingeniería en alimentos, ingeniería en ciencias de la computación y producción musical. La variedad en los perfiles de los usuarios permitió obtener resultados desde distintos puntos de vista, enriqueciendo así el análisis de la experiencia de usuario. Como se mencionó en los resultados, se presentó a los usuarios un recorrido que solo incluía los primeros objetivos y niveles del tour. Esto permitió evaluar si los usuarios comprendían la aplicación y podían utilizarla sin ayuda adicional, mientras interactuaban con todas sus funcionalidades. Al limitar la prueba a los dos primeros niveles, los usuarios pudieron dar un feedback completo sin necesidad de realizar el recorrido completo, optimizando su tiempo.

Con los resultados del formulario, se puede observar en las Figuras 7.22 y 7.23 que en general las personas tuvieron una buena experiencia de usuario, y creen que la aplicación es intuitiva y fácil de utilizar. Por otro lado, la Figura 7.24, indica que aunque la orientación parece ser, en su mayoría, precisa, algunos usuarios sí identificaron problemas con la precisión del recorrido. Esto vuelve a resaltar que la aplicación es muy propensa a errores, debido a que hay muchos factores que pueden desencadenar el problema del registro.

Uno de estos factores se puede ver en la Figura 7.27, que muestra que el 54 % de los usuarios experimentaron algún tipo de retraso al usar la aplicación, afectando la alineación entre el mundo virtual y el real. Esto sugiere que la aplicación se trababa ocasionalmente, causando fallas e imprecisiones en el algoritmo de *pathfinding*.

Otro punto importante de mencionar es el escaneo de los códigos QR y como lo percibieron los usuarios. La Figura 7.26 muestra que un 90 % de los usuarios indicó que ajustar su posición les resultó como algo sencillo de realizar, pero en los comentarios, todos indican que se necesitan definir los puntos específicos en los que se pueda leer el código QR. Esto significa que, aunque los usuarios piensan que si es intuitivo y sencillo escanear un código QR, ellos no sabían donde pararse exactamente, y algunos lo hacían sin cautela, causando más de algún problema con la alineación del mundo real y el virtual.

Por otro lado, las pruebas de usuario también sirvieron para identificar un error en uno de los pasos del desarrollo: la implementación de la flecha en vez de la línea. Originalmente, se pensó que la flecha resultaría en una experiencia de usuario más satisfactoria, ya que es menos invasiva, pero resulta que este razonamiento estaba mal. Como se puede observar en la Figura 7.25, el 72 % de los usuarios prefirieron la línea sobre la flecha. Cuando se les preguntó por qué eligieron esta opción, ellos dijeron que era porque esta traza la ruta completa de inicio a fin, ayudándoles a anticipar giros y evitando confusiones. En contraste, la flecha generó problemas al atravesar pasillos rectos, ya que solo apuntaba hacia adelante, sin indicar si el usuario debía girar a la izquierda o derecha después del pasillo. Esto provocó que los usuarios se desorientaran y, en algunos casos, tomaran la dirección equivocada. Además, algunos usuarios sugirieron combinar ambas herramientas, aprovechando la claridad de la línea y la discreción de la flecha para distintos momentos del recorrido. Así, estas pruebas ayudaron a reconsiderar el uso de la línea y a mejorar la interfaz, haciéndola más precisa y fácil de seguir sin perder sutileza.

En la segunda ronda de pruebas, los resultados variaron significativamente debido a la diversidad en el perfil de los participantes. Este grupo incluyó 7 adultos (padres de familia), 4 jóvenes y 1 niño, ninguno de los cuales pertenecía a la Universidad del Valle de Guatemala. Al analizar los resultados, se puede observar que esta diversidad de usuarios introdujo nuevos desafíos y perspectivas.

En cuanto a la intuitividad y la experiencia general de usuario, como se puede observar en las Figuras 7.28 y 7.29, aunque los resultados siguen siendo positivos, ya no son completamente excelentes como en la primera ronda. Esto se debe principalmente a que algunos adultos y el niño encontraron más difícil adaptarse a la aplicación, lo que subraya la importancia de considerar diferentes niveles de familiaridad tecnológica al diseñar la interfaz.

En términos de precisión, la Figura 7.30 muestra que varios usuarios experimentaron problemas significativos. Estos se deben en gran parte a dificultades que tuvieron los usuarios con el escaneo de los códigos QR (Figura 7.32, y con los retrasos que experimentaron estos al utilizar la aplicación (Figura 7.33). Esto es más evidente en usuarios mayores y en el niño, quienes no estaban familiarizados con los detalles técnicos de la aplicación y tuvieron más dificultades para ajustarse a las instrucciones implícitas.

A pesar de estos desafíos, una mejora notable fue la herramienta guía utilizada. Como se observa en las Figuras 7.25 y 7.31, la línea de recorrido siguió siendo la opción favorita entre los participantes. Este cambio, implementado a partir de los resultados de la primera ronda, demostró ser una decisión acertada, ya que ayudó a minimizar la confusión y permitió a los usuarios anticipar giros con mayor facilidad.

Por lo tanto, los resultados de la segunda ronda reflejan la importancia de realizar pruebas con grupos más diversos, ya que permiten identificar problemas que no se presentaron en la primera fase. Factores como la edad, la familiaridad con la tecnología y la falta de relación con la institución donde se desarrolló el proyecto influyeron directamente en la percepción de la aplicación y en los resultados obtenidos. Esto resalta la necesidad de diseñar soluciones más inclusivas y robustas para

un público más amplio.

Tras las pruebas de usuario, se realizaron pruebas finales en los modelos 3D y en el algoritmo de *pathfinding* para evaluar su precisión en condiciones óptimas. Cuando la aplicación funcionaba correctamente, los resultados fueron favorables. La Tabla 7.8 muestra que el error de precisión en todos los modelos fue menor a 2 metros, y el promedio de error en las 18 pruebas realizadas fue de 1.64 metros, lo que indica una alta precisión de la aplicación. Sin embargo, como se mencionó anteriormente, no todas las pruebas fueron exitosas. Una prueba exitosa es aquella en la que no surge el problema de registro.

Los resultados revelan que, de las 27 pruebas realizadas, 9 presentaron problemas, lo que equivale a un 33.3 % de probabilidad de error. Esto significa que, de cada tres usuarios, al menos uno podría encontrarse con el problema de registro. A veces, este problema es provocado por factores simples, como un escaneo incorrecto del código QR o un movimiento excesivo del teléfono, aunque en ocasiones también se manifiesta a mitad del recorrido, incluso cuando todo parece estar funcionando bien.

Los resultados también indican que los niveles con más probabilidad de error son los niveles 4 y 6 y que el error ocurre, específicamente, cuando se están atravesando los pasillos 8 (Figura 7.4) y 5 (Figura 7.5) respectivamente. De manera similar, en otros niveles la aplicación tiende a fallar al cruzar los pasillos más largos. Esto parece deberse a que la longitud de estos pasillos ocasiona problemas en el renderizado de esas secciones del modelo, lo que desencadena el problema de registro.

Para investigar esta situación, se realizó una prueba en el nivel 6. Tras un escaneo exitoso que alineó con precisión el modelo 3D con las paredes reales, se inició la ruta. Al principio, todo funcionó correctamente, pero al llegar al pasillo 5, el modelo se desalineó ligeramente, rotándose unos grados. Esto provocó que, si el usuario caminaba en línea recta, eventualmente colisionara con una pared.

Este comportamiento está vinculado al rendimiento general de la aplicación, especialmente debido al alto consumo de recursos. Como se menciona en los resultados, al simular el comportamiento en Unity, el modelo 3D se alinea correctamente y la aplicación funciona de manera óptima. Sin embargo, en la implementación real, el alto consumo de RAM afecta el rendimiento. Tal como se observa en la Imagen B de la Figura 7.21, aunque se ha logrado disminuir el consumo de RAM, este aún alcanza casi 1 GB, lo cual impacta significativamente el desempeño de la aplicación. La Figura 7.27 confirma esta limitación, mostrando que varios usuarios experimentaron retrasos en la aplicación a causa del alto consumo de memoria. Este problema, sumado a la carga que representan los seis modelos y otros elementos adicionales, afecta el renderizado en cada nivel y provoca desalineaciones en tiempo real entre el entorno virtual y el mundo real.

A pesar de los esfuerzos por optimizar el rendimiento, como la implementación de un algoritmo de complejidad O(1) que calcula la distancia en línea recta hacia el objetivo en lugar de toda la ruta, el consumo de memoria sigue siendo excesivo. En conclusión, la aplicación aún no es viable para recorridos guiados, debido tanto a los problemas de registro que surgen por factores externos como al alto consumo de memoria. Por lo tanto, es esencial llevar a cabo revisiones adicionales para reducir el uso de recursos y mitigar los factores que afectan el registro, con el fin de mejorar la estabilidad y funcionalidad de la aplicación en futuras versiones.

# CAPÍTULO 9

---

## Conclusiones

---

- La aplicación permite a los usuarios realizar recorridos virtuales en el CIT mediante un modelo 3D integrado con herramientas de navegación en Unity. Sin embargo, las pruebas evidenciaron que la congruencia entre el modelo virtual y el entorno real aún presenta desafíos significativos, particularmente en el registro y la alineación precisa. Por lo tanto, es necesario implementar mejoras en la precisión del modelado y en la integración de tecnologías complementarias para garantizar recorridos efectivos y confiables.
- A través de Blender, se mapearon con precisión los seis niveles del CIT, estableciendo una base sólida para el resto del proyecto y asegurando una representación fiel del espacio.
- En Unity, se creó el ambiente completo, configurando la aplicación de realidad aumentada e integrando los modelos 3D junto con todos los puntos de interés y códigos QR de cada nivel.
- Aunque el algoritmo inicial de *pathfinding* presentó algunos inconvenientes, se implementó un enfoque alternativo que resultó eficiente para la navegación, demostrando su utilidad como guía confiable para los usuarios.
- La interfaz de usuario resultó ser fundamental para la aplicación, ya que no solo mejora la experiencia del recorrido, sino que también proporciona información valiosa al usuario, facilitando la interacción con la herramienta.
- La compatibilidad multiplataforma en Android e iOS fue un objetivo logrado gracias a Unity, que permitió gestionar configuraciones y plugins sin complicaciones, logrando así una descarga y uso exitoso en ambos tipos de dispositivos.
- Las pruebas, tanto internas como de usuario, fueron esenciales en el desarrollo del proyecto, ya que permitieron identificar áreas de mejora y realizar ajustes necesarios, optimizando el funcionamiento de la aplicación.
- En términos de impacto social, la herramienta permite a estudiantes y visitantes explorar entornos interactivos y aprender sobre tecnología de realidad aumentada. En el ámbito universitario, facilita la comprensión del entorno y, a futuro, podría usarse como plataforma educativa para diversos temas.
- En términos de impacto ambiental, la aplicación reduce la necesidad de recursos físicos, como guías o panfletos impresos, al centralizar toda la información del recorrido en un solo dispositivo. Esta digitalización contribuye positivamente al medio ambiente al disminuir el consumo de papel y materiales de impresión.

- En términos de escalabilidad, más allá del ámbito universitario, esta solución puede implementarse en una variedad de sectores como sitios turísticos, museos, hospitales y parques, mejorando la experiencia de los visitantes y ofreciendo una guía eficiente y accesible en diversas industrias.

# CAPÍTULO 10

## Recomendaciones

- **Utilizar planos actualizados:** Contar con planos precisos y actualizados del edificio o estructura es un paso crucial para garantizar el éxito de cualquier proyecto que implique navegación interior. Estos planos no solo permiten evitar problemas de registro al trazar el contorno, sino que también simplifican la creación de modelos tridimensionales y ahoran tiempo durante las etapas iniciales del desarrollo. Al trabajar con planos precisos, se pueden enfocar los esfuerzos del desarrollo en optimizar otras áreas del proyecto en lugar de corregir errores originados por inconsistencias en los planos.

En el proyecto, la falta de precisión en los planos generó retrasos significativos, lo que afectó el cronograma general. Este problema obligó a realizar ajustes manuales y verificaciones constantes, reduciendo el tiempo disponible para el desarrollo y las pruebas. Por ello, se recomienda que, antes de iniciar cualquier desarrollo, se validen los planos con mediciones físicas en el sitio, asegurándose de que reflejen con exactitud las dimensiones y características reales del entorno. Esta práctica no solo ahorra tiempo y recursos, sino que también reduce el margen de error en etapas posteriores.

- **Fijar códigos QR en posiciones estables:** La correcta ubicación y estabilidad de los códigos QR es fundamental para garantizar una experiencia de usuario fluida en sistemas de navegación interior. Los errores de escaneo suelen depender de factores como la posición del usuario, la iluminación y el movimiento del código QR, lo que puede dificultar su lectura. Al fijar los códigos en lugares estables y claramente visibles, se minimizan estos problemas, asegurando que los usuarios puedan utilizarlos sin contratiempos.

Además, se sugiere emplear identificadores visuales como stickers o marcas en el suelo para indicar al usuario dónde posicionarse al escanear el código. Esto no solo mejora la precisión del escaneo, sino que también facilita la interacción, especialmente para aquellos usuarios que no están familiarizados con la tecnología. Implementar estas medidas incrementa la tasa de escaneos exitosos y contribuye a una navegación interior más eficiente, reduciendo la frustración del usuario y fortaleciendo su confianza en el sistema.

- **Verificar el consumo de memoria del dispositivo:** Asegurar que la aplicación sea eficiente en el uso de recursos es vital para garantizar su funcionalidad en una amplia gama de dispositivos móviles. Durante el desarrollo, se observó que el consumo excesivo de memoria puede provocar fallas, como cierres inesperados o un rendimiento lento, lo que afecta negativamente la experiencia del usuario. Por ello, es esencial realizar pruebas exhaustivas en dispositivos con diferentes especificaciones técnicas para identificar posibles cuellos de botella.

Se recomienda también optimizar el código y los recursos utilizados en la aplicación, como modelos 3D, texturas y animaciones, para minimizar el uso de memoria. Herramientas de monitoreo como los perfiles de rendimiento de Unity pueden ser útiles para detectar áreas problemáticas y aplicar ajustes necesarios. Además, se sugiere implementar un sistema de limpieza de memoria en tiempo de ejecución, liberando recursos que ya no se utilizan, para evitar acumulaciones innecesarias y mejorar la estabilidad de la aplicación.

- **Considerar un enfoque de navegación con sensores UWB:** Para abordar los problemas de registro y mejorar la precisión de la navegación, se recomienda implementar sensores Ultra Wideband (UWB) en puntos estratégicos del edificio. Estos sensores ofrecen múltiples ventajas al permitir la detección precisa de la ubicación del dispositivo móvil del usuario mediante técnicas de triangulación, eliminando la dependencia de modelos tridimensionales manuales y reduciendo significativamente los errores de alineación.

Una característica destacada de los sensores UWB es su capacidad para manejar la verticalidad dentro de entornos multilevel, como el CIT. Esto se logra mediante el uso del identificador único (ID) de cada sensor, lo que permite determinar en qué nivel del edificio se encuentra el usuario. Esta funcionalidad no solo facilita guiar al usuario de manera precisa entre diferentes niveles, sino que también optimiza el cálculo de rutas al evitar la necesidad de realizar ajustes manuales o utilizar tecnologías adicionales, como los códigos QR.

Además, los sensores UWB emiten señales de alta frecuencia que son menos susceptibles a interferencias, lo que mejora la estabilidad y la fiabilidad de la navegación interior. Este enfoque también puede integrar datos en tiempo real para proporcionar una experiencia de usuario más fluida y robusta. Sin embargo, cabe destacar que esta solución puede aumentar los costos iniciales de instalación y configuración, así como requerir un mantenimiento periódico para asegurar un funcionamiento continuo y preciso.

---

## Bibliografía

---

- [1] The Daily Duck, “Innovative tools for enhancing learning outcomes,” 1 de marzo 2024, recuperado de <https://blogs.uoregon.edu/articles/2024/03/01/innovative-tools-for-enhancing-learning-outcomes/>.
- [2] F. Khan, “The benefits of using virtual tours in higher education marketing,” 9 de marzo 2023, recuperado de <https://www.linkedin.com/pulse/benefits-using-virtual-tours-higher-education-marketing-firdosh-khan/>.
- [3] MoldStud, “The potential of augmented reality in enhancing campus visits for admissions,” 24 de enero 2024, recuperado de <https://moldstud.com/articles/p-the-potential-of-augmented-reality-in-enhancing-campus-visits-for-admissions#:~:text=AR%20provides%20immersive%20virtual%20campus,helps%20students%20make%20informed%20decisions>.
- [4] ARCore, “Getting started with ar foundation,” 14 de mayo 2024, recuperado de <https://developers.google.com/ar/develop/unity-arf/getting-started-ar-foundation#:~:text=Unity's%20AR%20Foundation%20is%20a,via%20Unity's%20AR%20Foundation%20package>.
- [5] Rock Paper Reality, “Augmented reality in tourism and travel,” 18 de octubre 2023, recuperado de <https://rockpaperreality.com/insights/ar-use-cases/augmented-reality-in-tourism-and-travel/>.
- [6] R. Azuma, “A survey of augmented reality,” *Presence: Hughes Research Laboratories*, vol. 6, no. 4, pp. 355–385, 1997, recuperado de <https://www.cs.unc.edu/~azuma/ARpresence.pdf>.
- [7] H. Tuli and D. Arora, “Augmented and virtual reality in tourism,” *International Journal of Innovative Research in Engineering & Management (IJIREM)*, vol. 10, no. Special Issue-1, pp. 1–4, 2023, recuperado de <https://ijirem.org/DOC/Augmented%20and%20Virtual%20Reality%20in%20Tourism-474725.pdf>.
- [8] Interaction Design Foundation - IxDF, “What is virtuality continuum?” 21 de enero 2022, recuperado de <https://www.interaction-design.org/literature/topics/virtuality-continuum>.
- [9] R. Silva, J. C. Oliveira, and G. A. Giraldi, “Introduction to augmented reality,” National Laboratory for Scientific Computation, Av. Getulio Vargas, 333- Quitandinha-Petropolis-RJ, Brazil, Tech. Rep., 2024, recuperado de <https://www.lncc.br/~jauvane/papers/RelatorioTecnicoLNCC-2503.pdf>.
- [10] T. Kyuder, “Indoor navigation: A comprehensive guide,” 10 de julio 2023, recuperado de <https://navigine.com/blog/indoor-navigation-a-comprehensive-guide/>.

- [11] S. Slater, “Indoor navigation with 3d renderings,” 4 de diciembre 2018, recuperado de <https://concept3d.com/blog/indoor-mapping/indoor-navigation-with-3d-renderings/>.
- [12] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2020, recuperado de <https://dl.ebooksworld.ir/books/Artificial.Intelligence.A.Modern.Approach.4th.Edition.Peter.Norvig.%20Stuart.Russell.Pearson.9780134610993.EBooksWorld.ir.pdf>.
- [13] U. Hybesis, “Pathfinding algorithms: The four pillars,” septiembre 2023. [Online]. Available: <https://medium.com/@urna.hybesis/pathfinding-algorithms-the-four-pillars-1ebad85d4c6b>
- [14] S. M. Uddin, T. Hossain, Suhaila, T. T. Oishee, and A. F. Hasan, “Comprehensive review of the a\* algorithm: Theory, implementation, and applications,” Independent University, Bangladesh, Tech. Rep., verano 2021, recuperado de [https://www.researchgate.net/publication/378521690\\_Comprehensive\\_Review\\_of\\_the\\_A\\_Algorithm\\_Theory\\_Implementation\\_and\\_Applications](https://www.researchgate.net/publication/378521690_Comprehensive_Review_of_the_A_Algorithm_Theory_Implementation_and_Applications).
- [15] M. Suárez, “Lógica y argumentación: Heurística,” 3 de octubre 2024, recuperado de <https://quillbot.com/es/blog/logica-y-argumentacion/heuristica/>.
- [16] G. E. Mathew, “Direction based heuristic for pathfinding in video games,” in *Procedia Computer Science*, vol. 47. Elsevier, 2015, pp. 262–271, recuperado de [https://www.sciencedirect.com/science/article/pii/S1877050915004743?ref=pdf\\_download&fr=RR-2&rr=8cd7c4b1fcf38757](https://www.sciencedirect.com/science/article/pii/S1877050915004743?ref=pdf_download&fr=RR-2&rr=8cd7c4b1fcf38757).
- [17] M. de Gregorio, “Qué es ui o user interface,” 21 de julio 2021, recuperado de <https://openwebinars.net/blog/que-es-ui-o-user-interface/#:~:text=El%20Dise%C3%B3n%20de%20Interfaz%20o,aplicaci%C3%B3n%20un%20dispositivo%20cualquiera>.
- [18] Tamushi, “Pruebas de estrés de software: ¿qué son y para qué sirven?” 8 de julio 2022, recuperado de <https://www.testingit.com.mx/blog/pruebas-de-estres-de-software#:~:text=%C2%BFQu%C3%A9%20es%20una%20prueba%20de,de%20riesgo%20ante%20cargas%20extremas>.
- [19] A. Kleftodimos, M. Moustaka, and A. Evangelou, “Location-based augmented reality for cultural heritage education: Creating educational, gamified location-based ar applications for the prehistoric lake settlement of dispilio,” *Digital*, vol. 3, no. 1, pp. 18–45, enero 2023, enviado: 14 de noviembre de 2022 / Revisado: 9 de enero de 2023 / Aceptado: 11 de enero de 2023 / Publicado: 15 de enero de 2023. Recuperado de <https://www.mdpi.com/2673-6470/3/1/2>.
- [20] V. Agrawal and J. Patel, “A review: Augmented reality and its working,” *International Research Journal of Engineering and Technology (IRJET)*, vol. 4, no. 5, pp. 1–8, mayo 2017, e-ISSN: 2395-0056 / p-ISSN: 2395-0072. Recuperado de <https://www.irjet.net/archives/V4/i5/IRJET-V4I5118.pdf>.
- [21] I. Cibilić, V. Poslončec-Petrić, and K. Tominić, “Implementing augmented reality in tourism,” *Proceedings of the International Cartographic Association*, vol. 4, no. 1, pp. 1–5, 2021, recuperado de <https://ica-proc.copernicus.org/articles/4/21/2021/ica-proc-4-21-2021.pdf>.
- [22] D. M. Laparra, “Pathfinding algorithms in graphs and applications,” Master’s thesis, Universitat de Barcelona, 15 de enero 2019, recuperado de <https://deposit.ub.edu/dspace/bitstream/2445/140466/1/memoria.pdf>.
- [23] R. Belwariar, 30 de julio 2024, recuperado de <https://www.geeksforgeeks.org/a-search-algorithm/>.
- [24] Blender Foundation, “Blender about,” recuperado de <https://www.blender.org/about/>.
- [25] U. Technologies, “Welcome to unity,” 2023, recuperado de <https://unity.com/our-company>.
- [26] A. Sinicki, “What is unity? everything you need to know,” 20 de marzo 2021, recuperado de <https://www.androidauthority.com/what-is-unity-1131558/>.

- [27] U. Technologies, “Unity’s package manager,” 8 de febrero 2024, recuperado de <https://docs.unity3d.com/Manual/Packages.html#:~:text=The%20Package%20Manager%20also%20supports,examples%2C%20tutorials%20and%20Editor%20extensions>.
- [28] ——, “Working with navmesh agents,” 2024, recuperado de <https://learn.unity.com/tutorial/working-with-navmesh-agents#>.
- [29] ——, “Arfoundation,” 2024, recuperado de <https://unity.com/es/unity/features/arfoundation>.
- [30] ——, “Scene setup,” 2024, recuperado de <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.0/manual/project-setup/scene-setup.html#:~:text=The%20AR%20Session%20GameObject%20enables,AR%20will%20not%20function%20properly>.
- [31] P. Sáez, “Arkit vs arcore,” 29 de marzo 2023, recuperado de <https://www.onirix.com/es/arkit-vs-arcore/>.
- [32] W. Schools, “Introduction to c#,” 2024, recuperado de [https://www.w3schools.com/cs/cs\\_intro.php](https://www.w3schools.com/cs/cs_intro.php).
- [33] A. Mutel, “Unity and .net, what’s next?” 2022, recuperado de <https://unity.com/blog/engine-platform/unity-and-net-whats-next>.
- [34] T. C. Essentials, “Importing reference images to scale in blender,” 19 de junio 2020, [Video de YouTube]. Recuperado de [https://www.youtube.com/watch?v=N\\_u6Hk78EA&t=376s](https://www.youtube.com/watch?v=N_u6Hk78EA&t=376s).
- [35] A. Topics, “How to make 3d floor plan in blender,” 24 de mayo 2024, [Video de YouTube]. Recuperado de <https://www.youtube.com/watch?v=Q4rbqUbhYXY&t=744s>.
- [36] FireDragonGameStudio, “Indoor navigation - videos 2 al 6 de la playlist,” 25 de enero 2022, [Video de YouTube]. Recuperado de [https://www.youtube.com/watch?v=IuNs0AeATg0&list=PLOIYTFRd0Ho7iOI\\_cnUZxXK6KiCFGhU1s](https://www.youtube.com/watch?v=IuNs0AeATg0&list=PLOIYTFRd0Ho7iOI_cnUZxXK6KiCFGhU1s).

## Anexos

## ANEXO A

---

### Repository de Github

---

#### Repository de Github

**Enlace:** <https://github.com/ARTourUVG/ARTour>

Inicialmente, el desarrollo del proyecto comenzó utilizando Unity Version Control, una herramienta de versionamiento integrada en Unity que facilitaba el trabajo colaborativo del equipo. Sin embargo, esta herramienta no permitía compartir el enlace del repositorio con otros interesados ni ofrecía una capacidad de colaboradores adecuada para los objetivos del proyecto. Por esta razón, al final del proceso se decidió migrar el código a un repositorio público en GitHub, con el propósito de facilitar el acceso al código fuente y la documentación para la entrega final.

## ANEXO B

---

### Cuestionario de Evaluación del AR Tour

---

#### Introducción

Este cuestionario tiene como objetivo evaluar la experiencia de usuario con la aplicación AR Tour, una herramienta de realidad aumentada diseñada para proporcionar recorridos virtuales interactivos. Las preguntas están orientadas a medir diversos aspectos de la aplicación, incluyendo su usabilidad, efectividad de las herramientas de navegación, rendimiento técnico y adaptabilidad para diferentes tipos de usuarios.

#### Formulario de Evaluación

1. ¿Considera que la aplicación es suficientemente intuitiva?

1-10. Escala de valoración donde:

- 1 = Nada interactiva
- 10 = Muy interactiva

2. ¿Cómo describiría su experiencia general durante el recorrido virtual con el AR Tour?

1-10. Escala de valoración donde:

- 1 = Muy insatisfactoria
- 10 = Muy satisfactoria

3. ¿Qué tan precisa le pareció la indicación proporcionada por el sistema de guía del recorrido?

1-10. Escala de valoración donde:

- 1 = Muy imprecisa
- 10 = Muy precisa

4. En cuanto a las herramientas de navegación proporcionadas (línea de recorrido o flecha), ¿cuál considera que fue más efectiva para guiar su experiencia?
- Línea de recorrido
  - Flecha de dirección
  - Ambas igualmente efectivas
  - Ninguna fue efectiva
5. ¿Le resultó sencillo ajustar su posición AR con el código QR?
- Sí, fue sencillo
  - No, tuve algunas dificultades
  - No fue necesario ajustar la posición
6. ¿Hubo algún retraso en la detección de los puntos de interés del entorno físico?
- Nunca hubo retraso
  - Pocas veces hubo retraso
  - Frecuentemente hubo retraso
  - Siempre hubo retraso
7. ¿Considera que la interfaz gráfica de la aplicación es adecuada tanto para usuarios con experiencia previa en realidad aumentada como para usuarios sin experiencia?
- Sí, es adecuada para ambos tipos de usuarios
  - Sí, pero se podría mejorar para usuarios sin experiencia
  - Sí, pero se podría mejorar para usuarios con experiencia
  - No, no es adecuada para ninguno de los dos
8. En términos generales, ¿qué recomendaciones u observaciones podrías brindar sobre la aplicación AR Tour?

*[Espacio para respuesta abierta]*

## ANEXO C

---

### Figuras

---

#### Introducción

En este anexo se presentan las imágenes que documentan los aspectos más relevantes del proceso de desarrollo de la aplicación de realidad aumentada. A través de estas imágenes, se busca brindar al lector una visión detallada de cómo se fue estructurando y avanzando en el proyecto, desde el modelado 3D hasta la integración de los elementos que permiten la interacción con el usuario. Primero, se incluyen imágenes del proceso de modelado, donde se puede observar cómo se trazaron los planos en Blender, adaptándose a las dimensiones reales y levantando las estructuras en 3D. Estas imágenes permiten comprender la precisión y el detalle necesarios en cada etapa del modelado, que son fundamentales para lograr una alineación efectiva en el entorno de realidad aumentada.

Posteriormente, se presentan imágenes de los códigos QR utilizados en el proyecto. Estos códigos cumplen una función vital al permitir el reposicionamiento del usuario dentro del entorno virtual, asegurando que la ubicación sea precisa y evitando desajustes en el registro. Las imágenes de los códigos QR muestran tanto el diseño final como su ubicación en puntos estratégicos, lo que resulta clave para optimizar la navegación y mejorar la experiencia del usuario durante el recorrido virtual.

## Figuras del proceso de modelado



Figura C.1: Plano original nivel 1



Figura C.4: Plano original nivel 2

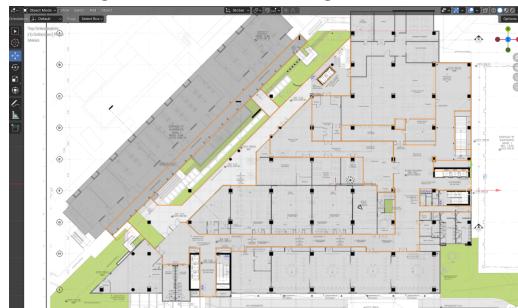


Figura C.2: Contorno trazado en plano nivel 1

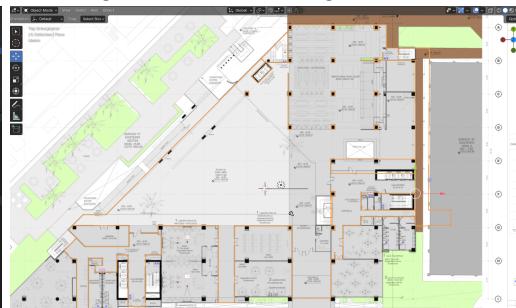


Figura C.5: Contorno trazado en plano nivel 2

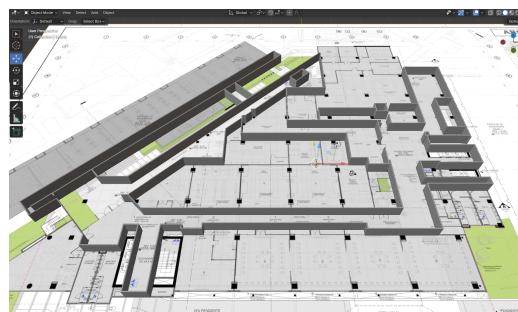


Figura C.3: Modelo 3D nivel 1



Figura C.6: Modelo 3D nivel 2

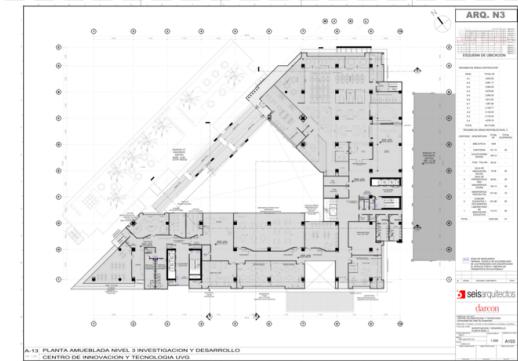


Figura C.7: Plano original nivel 3



Figura C.10: Plano original nivel 4

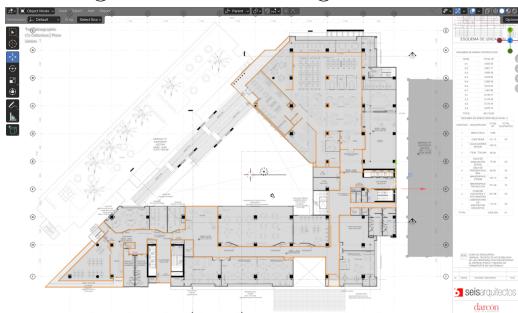


Figura C.8: Contorno trazado en plano nivel 3

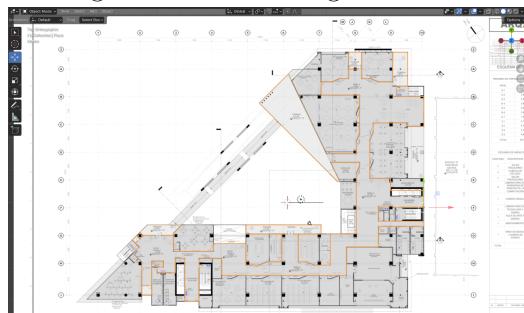


Figura C.11: Contorno trazado en plano nivel 4

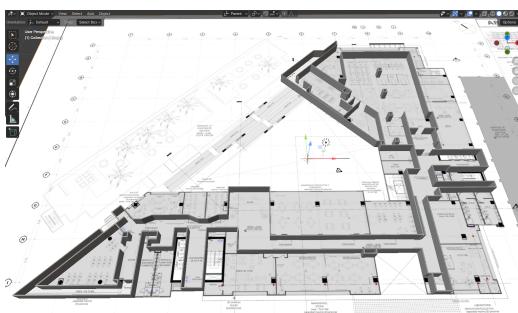


Figura C.9: Modelo 3D nivel 3



Figura C.12: Modelo 3D nivel 4



Figura C.13: Plano original nivel 6

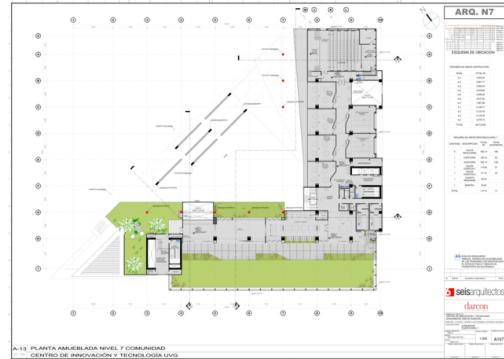


Figura C.16: Plano original nivel 7

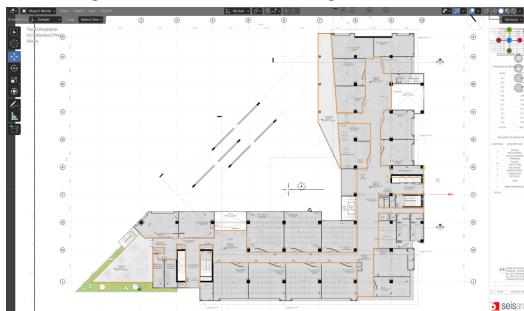


Figura C.14: Contorno trazado en plano nivel 6



Figura C.17: Contorno trazado en plano nivel 7

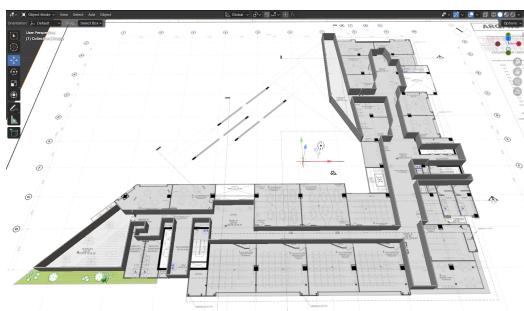


Figura C.15: Modelo 3D nivel 6

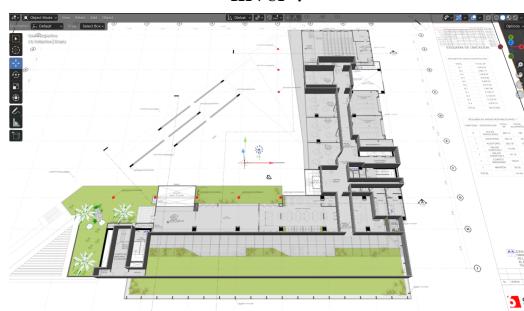


Figura C.18: Modelo 3D nivel 7

## Figuras de los códigos QR

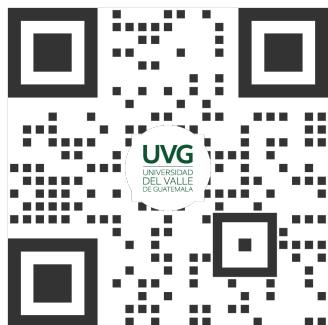


Figura C.19: Código QR elevadores principales nivel 1



Figura C.20: Código QR elevadores secundarios nivel 2

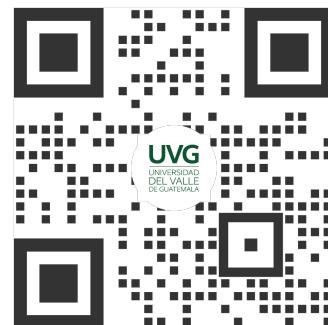


Figura C.21: Código QR elevadores principales nivel 3



Figura C.22: Código QR elevadores secundarios nivel 3



Figura C.23: Código QR elevadores principales nivel 4



Figura C.24: Código QR elevadores secundarios nivel 4

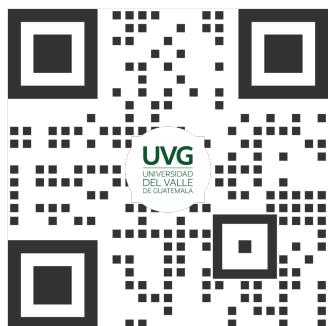


Figura C.25: Código QR elevadores principales nivel 6



Figura C.26: Código QR elevadores secundarios nivel 6

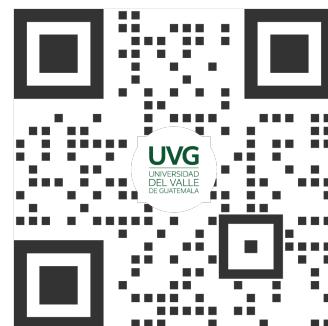


Figura C.27: Código QR elevadores principales nivel 7

## ANEXO D

---

### Fragmentos de Código

---

#### Introducción

En este anexo se muestran los fragmentos de código más importantes desarrollados para el funcionamiento de la aplicación de realidad aumentada, los cuales ilustran las soluciones implementadas para enfrentar los principales retos técnicos del proyecto. Uno de los elementos centrales es el algoritmo de pathfinding basado en A\*, el cual se adaptó específicamente para la navegación en el modelo tridimensional del CIT. Este algoritmo fue optimizado para maximizar la eficiencia y reducir el consumo de memoria, de modo que la navegación guiada sea más fluida y responda adecuadamente a los cambios de posición del usuario.

Además, se incluye el código destinado a la detección de los códigos QR y el reposicionamiento del usuario. Esta función es esencial para que el sistema mantenga una ubicación precisa del usuario en el entorno virtual, utilizando los códigos QR como puntos de referencia para corregir posibles desajustes. Finalmente, se presenta el código que controla el comportamiento de la flecha de navegación, la cual orienta al usuario en el recorrido virtual. Este fragmento asegura que la flecha ajuste su dirección conforme el usuario se desplaza y que la orientación sea intuitiva y precisa, mejorando así la experiencia de navegación. Los códigos incluidos aquí representan los aspectos más técnicos y desafiantes del proyecto y ofrecen una visión de cómo se abordaron dichos desafíos en el desarrollo de la aplicación.

## Fragmentos de código del algoritmo desarrollado basado en A\*

```

1 // Funcion para encontrar el camino entre dos puntos
2 public List<Node> FindPath(Vector3 startPos, Vector3 targetPos)
3 {
4     Node startNode = NodeFromWorldPoint(startPos);
5     Node targetNode = NodeFromWorldPoint(targetPos);
6
7     if (startNode == null || targetNode == null)
8     {
9         return null; // salida temprana si alguno de los nodos es nulo
10    }
11    List<Node> openSet = new List<Node>();
12    HashSet<Node> closedSet = new HashSet<Node>();
13    openSet.Add(startNode);
14
15    while (openSet.Count > 0)
16    {
17        Node currentNode = openSet[0];
18        for (int i = 1; i < openSet.Count; i++)
19        {
20            if (openSet[i].fCost < currentNode.fCost || openSet[i].fCost
21                == currentNode.fCost && openSet[i].hCost < currentNode.
22                hCost)
23            {
24                currentNode = openSet[i];
25            }
26        }
27        openSet.Remove(currentNode);
28        closedSet.Add(currentNode);
29
30        if (currentNode == targetNode)
31        {
32            return RetracePath(startNode, targetNode);
33        }
34
35        foreach (Node neighbor in GetNeighbors(currentNode))
36        {
37            if (!neighbor.walkable || closedSet.Contains(neighbor))
38            {
39                continue;
40
41                int newMovementCostToNeighbor = currentNode.gCost +
42                GetDistance(currentNode, neighbor);
43                if (newMovementCostToNeighbor < neighbor.gCost || !openSet.
44                Contains(neighbor))
45                {
46                    neighbor.gCost = newMovementCostToNeighbor;
47                    neighbor.hCost = GetDistance(neighbor, targetNode);
48                    neighbor.parent = currentNode;
49                    if (!openSet.Contains(neighbor))
50                        openSet.Add(neighbor);
51                }
52            }
53        }
54
55        return null; // no se encontro un camino
56    }
57 }
```

Código D.1: Función principal del algoritmo A\*

```
1 // Funcion para crear el la cuadricula
2 public void CreateGrid()
3 {
4     gridSizeX = Mathf.RoundToInt(gridWorldSize.x / nodeDiameter);
5     gridSizeY = Mathf.RoundToInt(gridWorldSize.y / nodeDiameter);
6     grid = new Node[gridSizeX, gridSizeY];
7
8     for (int x = 0; x < gridSizeX; x++)
9     {
10         for (int y = 0; y < gridSizeY; y++)
11         {
12             // ajustar segun el tamano de la cuadricula
13             Vector3 worldPoint = new Vector3(x * nodeDiameter, 0, y *
14                 nodeDiameter);
15             bool walkable = true;
16             grid[x, y] = new Node(walkable, worldPoint, x, y);
17         }
18     }
}
```

Código D.2: Función para crear la cuadricula

## Fragmentos de código para detectar QR y reposicionar al usuario

```
1 // Funcion para detectar el QR y reorientar la posicion
2 private void OnCameraFrameReceived(ARCameraFrameEventArgs eventArgs)
3 {
4     if (!cameraManager.TryAcquireLatestCpuImage(out XRCpuImage image))
5     {
6         return;
7     }
8
9     var conversionParams = new XRCpuImage.ConversionParams
10    {
11        inputRect = new RectInt(0, 0, image.width, image.height),
12        outputDimensions = new Vector2Int(image.width / 2, image.height
13                                         / 2),
14        outputFormat = TextureFormat.RGBA32,
15        transformation = XRCpuImage.Transformation.MirrorY
16    };
17
18    int size = image.GetConvertedDataSize(conversionParams);
19    var buffer = new NativeArray<byte>(size, Allocator.Temp);
20    image.Convert(conversionParams, buffer);
21    image.Dispose();
22
23    cameraImageTexture = new Texture2D(
24        conversionParams.outputDimensions.x,
25        conversionParams.outputDimensions.y,
26        conversionParams.outputFormat,
27        false);
28
29    cameraImageTexture.LoadRawTextureData(buffer);
30    cameraImageTexture.Apply();
31    buffer.Dispose();
32
33    var result = reader.Decode(cameraImageTexture.GetPixels32(),
34                               cameraImageTexture.width, cameraImageTexture.height);
35    if (result != null)
36    {
37        SetQrCodeRecenterTarget(result.Text);
38    }
39    else
40    {
41        UpdateLog("Ningun QR detectado.");
42    }
43}
```

Código D.3: Función para detectar códigos QR

```
1 // Funcion para reorientar la posicion del usuario
2 private void SetQrCodeRecenterTarget(string targetText)
3 {
4     if (positionSet) return; // Evitar reorientar la posicion si ya ha
5         sido reorientada
6
7     Target currentTarget = navigationTargetObjects.Find(x => x.Name.
8         ToLower().Equals(targetText.ToLower()));
9     if (currentTarget != null)
10    {
11        session.Reset();
12        sessionOrigin.transform.position = currentTarget.PositionObject.
13            transform.position;
14        sessionOrigin.transform.rotation = currentTarget.PositionObject.
15            transform.rotation;
16
17        UpdateLog($"Exitosamente reorientado a: {currentTarget.Name}");
18
19        positionSet = true; //Marcar la posicion como reorientada
20        // Llamar al metodo OnNextTargetButtonClicked en el script
21        SetNavigationTarget
22        SetNavigationTarget.Instance.OnNextTargetButtonClicked();
23        SetLineToggleTrue(); // Set lineToggle to true
24        StartCoroutine(ResetPositionSet()); // Empezar la corutina para
25            reiniciar la posicion
26    }
27    else
28    {
29        UpdateLog($"No se encontro un objetivo para: {targetText}");
30    }
31}
```

Código D.4: Función para reorientar al usuario

```

1 // Funcion para detectar el QR y reorientar la posicion
2 private void OnCameraFrameReceived(ARCameraFrameEventArgs eventArgs)
3 {
4     if (!SetNavigationTarget.Instance.IsSearchingForQRCode) return;
5     if (!cameraManager.TryAcquireLatestCpuImage(out XRCpuImage image))
6     {
7         return;
8     }
9
10    var conversionParams = new XRCpuImage.ConversionParams
11    {
12        inputRect = new RectInt(0, 0, image.width, image.height),
13        outputDimensions = new Vector2Int(image.width / 2, image.height
14            / 2),
15        outputFormat = TextureFormat.RGBA32,
16        transformation = XRCpuImage.Transformation.MirrorY
17    };
18
19    int size = image.GetConvertedDataSize(conversionParams);
20    var buffer = new NativeArray<byte>(size, Allocator.Temp);
21    image.Convert(conversionParams, buffer);
22    image.Dispose();
23
24    // Verificar si la textura existe y es del tama o correcto
25    if (cameraImageTexture == null ||
26        cameraImageTexture.width != conversionParams.outputDimensions.x
27            ||
28        cameraImageTexture.height != conversionParams.outputDimensions.y
29            ||
30        cameraImageTexture.format != conversionParams.outputFormat)
31    {
32        // Destruir la textura anterior si existe
33        if (cameraImageTexture != null)
34        {
35            Destroy(cameraImageTexture);
36        }
37
38        // Crear nueva instancia de la textura con par metros correcto
39        cameraImageTexture = new Texture2D(
40            conversionParams.outputDimensions.x,
41            conversionParams.outputDimensions.y,
42            conversionParams.outputFormat,
43            false);
44    }
45    // Actualizar los datos de la textura
46    cameraImageTexture.LoadRawTextureData(buffer);
47    cameraImageTexture.Apply();
48    buffer.Dispose();
49
50    var result = reader.Decode(cameraImageTexture.GetPixels32(),
51        cameraImageTexture.width, cameraImageTexture.height);
52    if (result != null)
53    {
54        SetQrCodeRecenterTarget(result.Text);
55    }
56}

```

Código D.5: Función optimizada para detectar códigos QR

## Fragmentos de código para asegurar el comportamiento correcto de la flecha

```

1 // Funcion para detectar puntos de giro en el camino de navegacion
2 List<Vector3> GetTurningPoints(Vector3[] corners)
3 {
4     List<Vector3> turningPoints = new List<Vector3>();
5
6     for (int i = 1; i < corners.Length - 1; i++)
7     {
8         Vector3 previousSegment = (corners[i] - corners[i - 1]).normalized;
9         Vector3 nextSegment = (corners[i + 1] - corners[i]).normalized;
10
11         // Calcular el angulo entre los segmentos
12         float angle = Vector3.Angle(previousSegment, nextSegment);
13
14         // Si el angulo es mayor a 20 grados, considerar el punto como
15         // un punto de giro
16         if (angle > 20f)
17         {
18             turningPoints.Add(corners[i]);
19         }
20
21         // Asegurarse de agregar el ultimo punto como un punto de giro
22         if (corners.Length > 0)
23         {
24             turningPoints.Add(corners[corners.Length - 1]);
25         }
26
27     return turningPoints;
28 }
```

Código D.6: Función para obtener los siguientes cruces en la ruta

```

1 // Funcion para detectar si un punto esta cerca de un segmento
2 private bool IsNearSegment(Vector3 pointA, Vector3 pointB, Vector3
3     position, float threshold)
4 {
5     Vector3 closestPoint = Vector3.Project(position - pointA, (pointB -
6         pointA).normalized) + pointA;
7     float distanceToSegment = Vector3.Distance(closestPoint, position);
8
9     // Revisar si el punto mas cercano esta dentro del segmento
10    if (Vector3.Dot(pointB - pointA, closestPoint - pointA) < 0 ||
11        Vector3.Dot(pointA - pointB, closestPoint - pointB) < 0)
12    {
13        return distanceToSegment < threshold;
14    }
15
16    return distanceToSegment < threshold;
17 }
```

Código D.7: Función para manejar correctamente el siguiente punto