Sistema de Entrenamiento en Realidad Virtual para La Prevención y Respuesta ante Catástrofes

Facultad De Ingeniería

Director
Ing. Edison Cañón

Universidad de Cundinamarca

Sergio Alejandro López Bernal
Bladimir Salinas Luque

2025

Sistema de Entrenamiento en Realidad Virtual par Primeros Respondientes Civiles

Trabajo presentado como requisito parcial para optar al título de
Ingeniero en Sistemas y Computación

Autores:
Sergio Alejandro López Bernal
Bladimir Salinas Luque

Director:
Ing. Edison Cañón

Facultad de Ingeniería
Universidad de Cundinamarca
Chía, Colombia
2025

AGRADECIMIENTOS

Quisiéramos expresar nuestro más sincero agradecimiento a todas las personas y así vez a la institución que hicieron posible la realización de este trabajo de grado. En primer lugar, agradezco profundamente a nuestro director de tesis, el Ing. Edison Cañón, por su guía, paciencia y valiosa orientación técnica a lo largo de todo el proyecto.

A la Universidad de Cundinamarca y a la Facultad de Ingeniería, por proporcionarnos los recursos no físicos para poder llevar a cabo este desarrollo. A nuestros profesores, por el conocimiento impartido durante nuestra formación.

Finalmente, un agradecimiento especial a nuestras familias y amigos, por su apoyo incondicional.

DEDICATORIA

El presente documento principalmente es dedicado a todas las personas que han estado presentes a lo largo de todo el tiempo transcurrido, pero también hacia el esfuerzo y dedicación no en este proyecto, si no en los casi 5 años de proceso que se han llevado a cabo para poder aprender y realizar todo lo que este proceso ha conllevado.

RESUMEN

La prevención y respuesta efectiva ante catástrofes naturales es un desafío global en la gestión de emergencias, este reto está latente en la actualidad debido a que no todos los países o comunidades tienen la capacidad e infraestructura para responder ante alguna situación de riesgo. los métodos de entrenamiento tradicionales, aunque valiosos, presentan limitaciones en realismo, escalabilidad y costos, resultando en una preparación inadecuada. este proyecto aborda dicha problemática mediante el desarrollo de un sistema de software basado en realidad virtual (rv) para la simulación de escenarios de emergencia. el objetivo principal es diseñar y construir un entorno virtual funcional que integre los mecanismos de interacción y los sistemas de seguimiento necesarios para la práctica de protocolos de primera respuesta civil. la metodología se estructura siguiendo las fases de un ciclo de vida de desarrollo de software: análisis, diseño, construcción y verificación. el sistema integra un motor de evaluación para registrar el desempeño del usuario según los protocolos establecidos. adicionalmente, el proyecto se enmarca conceptualmente en el objetivo de desarrollo sostenible 11 (ods 11) "ciudades y comunidades sostenibles", al proponer una herramienta tecnológica con potencial para promover la resiliencia y seguridad en entornos urbanos. el resultado de este trabajo es un artefacto de software verificado, sentando las bases para futuras investigaciones sobre su efectividad pedagógica.

Palabras clave: realidad virtual, simulación de emergencias, desarrollo de software, primer respondiente, rup ágil.

ABSTRACT

Effective prevention and response to natural disasters represent a global challenge in emergency management. This challenge is especially relevant today, as not all countries or communities have the capacity and infrastructure to respond to high-risk situations. Traditional training methods, while valuable, present significant limitations in realism, scalability, and cost, resulting in inadequate preparation for hazardous events. This project addresses this issue through the development of a software system based on virtual reality (vr) for the simulation of emergency scenarios. The main objective is to design and build a functional virtual environment that integrates the necessary interaction mechanisms and tracking systems for the practical training of civilian first responder protocols. The methodology follows the phases of the software development life cycle: analysis, design, construction, and verification, based on the agile unified process (agile rup) framework. The system incorporates an evaluation engine to record user performance according to established protocols. Conceptually, the project aligns with sustainable development goal 11 (sdg 11), “sustainable cities and communities,” by proposing a technological tool with the potential to promote resilience and safety in urban environments. The result of this work is a verified software artifact that establishes the foundation for future research on its pedagogical effectiveness.

Keywords: virtual reality, emergency simulation, software development, first responder, agile rup.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

INTRODUCCIÓN

La gestión del riesgo de desastres se ha consolidado en el siglo XXI como un desafío ineludible para el desarrollo sostenible y la seguridad humana a nivel global. En la última década, la población ha sido testigo de un alarmante incremento en la frecuencia e intensidad de catástrofes naturales tanto de situaciones que ponen en riesgo a las personas, en muchos casos por los efectos del cambio climático y la creciente urbanización en zonas de alta vulnerabilidad y así mismo el cambio de hábitos o des interés de las personas hacía el apoyo a la comunidad sin ánimo de lucro. Estos eventos no solo dejan pérdidas irreparables en términos de vidas humanas, sino que también generan pérdidas, y revierten años de progreso en comunidades enteras. Ante esta realidad, la preparación y la capacidad de respuesta de la sociedad se convierten en los pilares fundamentales para mitigar el impacto y construir una comunidad con un sentido de humanidad mucho mayor.

Trasladando esta realidad al contexto colombiano, la urgencia de una preparación efectiva se magnifica. Por su ubicación geográfica en el Cinturón de Fuego del Pacífico y la confluencia de tres placas tectónicas (Nazca, Caribe y Sudamericana), el país presenta una de las actividades sísmicas más altas del continente. El Sistema Nacional de Fallas Geológicas atraviesa el territorio, colocando a grandes centros urbanos y a millones de ciudadanos bajo una amenaza sísmica latente y constante. Esta condición geológica, sumada a la vulnerabilidad social e infraestructural, subraya la imperiosa necesidad de fortalecer las competencias de la población para actuar de manera adecuada antes, durante y después de un evento telúrico (Cardona, 2001).

En este marco, la prevención y respuesta efectiva ante catástrofes y situaciones que pueda presentar un peligro inminente representa un de los mayores desafíos en la gestión de emergencias. Los métodos tradicionales de entrenamiento, como los simulacros de evacuación y las charlas teóricas, aunque valiosos, presentan limitaciones significativas se limitan simplemente a dar pequeños cursos o capacitaciones que simplemente se quedan en algo lúdico o repetitivo. La principal deficiencia radica en la denominada "brecha psicológica", la abismal diferencia entre el conocimiento teórico de un protocolo y la capacidad de ejecutarlo bajo un ambiento lógicamente controlado pero que al mismo tiempo es mucho más inmersivo que simplemente estar obteniendo la teoría. Los simulacros convencionales, por su naturaleza predecible, rara vez logran replicar la confusión o una sensación un poco más inmersiva la cual

haga cometer errores, que es lo que normalmente pasa en una situación de estas, el miedo y la presión temporal que caracterizan a un desastre, donde el comportamiento real de los individuos a menudo no se corresponde con lo ensayado (Canto & Carpi Ballester, 2019).

Frente a este panorama, la realidad virtual (RV) emerge como un cambio de paradigma tecnológico. Más que una simple herramienta, la RV ofrece una nueva plataforma para el aprendizaje experiencial permitiendo la creación de entornos inmersivos, interactivos y, fundamentalmente, seguros (Bustos-Sánchez & Chacón-Medina, 2020). La capacidad de esta tecnología para generar una profunda sensación de "presencia" la percepción de estar realmente en el lugar simulado es clave para de cierta forma engañar a la mente y que este un poco mas alerta, permitiendo a los usuarios experimentar respuestas emocionales y cognitivas más auténticas. Las ventajas de la RV para este propósito son múltiples y han sido validadas, ofrece un entorno safe-to-fail (seguro para fallar), permite una repetibilidad ilimitada para la automatización de protocolos, y facilita la recolección de datos objetivos para una evaluación rigurosa del desempeño (Ingrassia et al., 2021).

En respuesta a este vacío, el presente trabajo de grado aborda el diseño, desarrollo y verificación de un "Sistema de Entrenamiento en Realidad Virtual para la Prevención y Respuesta ante Catástrofes", con un enfoque en la capacitación del primer respondiente civil. La contribución principal de este proyecto es la entrega de un artefacto de software funcional y verificado, que sirve como una solución tecnológica robusta a la problemática del entrenamiento. Se busca establecer una base de ingeniería sólida sobre la cual futuras investigaciones puedan medir el impacto pedagógico y el comportamiento humano en emergencias simuladas.

Adicionalmente, el desarrollo de este sistema se alinea conceptualmente con las metas del Objetivo de Desarrollo Sostenible (ODS) número 11: "Ciudades y Comunidades Sostenibles". Específicamente, se busca contribuir al proponer una herramienta tecnológica con el potencial de aumentar la resiliencia urbana, un pilar fundamental en la agenda de desarrollo para América Latina y el Caribe (CEPAL, 2018). El proyecto, por tanto, se consolida como un esfuerzo de ingeniería para la construcción de herramientas que promuevan comunidades más seguras.

1. PROBLEMA

1.1. Planteamiento del problema

En cualquier catástrofe o emergencia, como un sismo, un incendio estructural o un accidente con múltiples heridos, los primeros minutos son los más críticos para la supervivencia de las víctimas. Durante este lapso, antes de la llegada de los servicios de emergencia profesionales, la acción o inacción de los ciudadanos presentes se convierte en un factor determinante en el resultado final. Estos individuos, conocidos como primeros respondientes civiles, constituyen la primera línea de defensa de una comunidad. Sin embargo, a pesar de su rol crucial, estudios demuestran que la gran mayoría de la población carece de la preparación práctica y la autoeficacia necesarias para actuar de manera efectiva en estas situaciones de alta presión (Cruz Roja, 2020; Global First Aid, 2021).

Para comprender la relevancia de esta herramienta complementaria, es necesario profundizar en las dimensiones del problema que busca atender. El problema fundamental es la brecha entre la expectativa social de que un ciudadano ayude en una emergencia y la competencia real que posee para hacerlo de forma segura y eficaz. Los métodos de capacitación actuales, como cursos teóricos o simulacros tradicionales, a menudo resultan insuficientes para forjar las habilidades robustas que se requieren. El problema no es solo la falta de información, sino la incapacidad de los paradigmas de entrenamiento existentes para replicar la complejidad, el estrés y la imprevisibilidad de una emergencia real, un fenómeno conocido como “fracaso en la transferencia de habilidades (Medina Medina, 2013; Savickas et al., 2019).

El principal obstáculo para una primera respuesta ciudadana efectiva es el shock psicológico. Enfrentarse a una persona con una lesión grave o el caos de una evacuación genera una sobrecarga cognitiva y emocional que puede inhibir el pensamiento racional. El cerebro humano, bajo este estrés agudo, a menudo activa respuestas primarias de "lucha, huida o congelación", lo que dificulta el acceso a la memoria procedimental. Un curso teórico puede enseñar los pasos de la Reanimación Cardiopulmonar (RCP), pero no necesariamente entrena a la persona para ejecutar esa secuencia en un entorno caótico y con la presión de una vida en juego (Sørensen et al., 2021).

La capacitación para primeros respondientes enfrenta una contradicción fundamental: para ser efectiva, debe ser realista; pero para ser accesible, debe ser segura. Los métodos físicos

no pueden resolver este dilema. Es éticamente inaceptable simular de manera realista lesiones graves para que los ciudadanos practiquen (Lateef, 2010). Se recurre a maniquíes y simulaciones de baja fidelidad que no preparan para el impacto visual y emocional de una lesión real, limitando la adquisición de competencias no técnicas como el manejo del estrés y la toma de decisiones.

Además, los cursos prácticos de alta calidad son costosos y requieren instructores certificados, lo que limita drásticamente su escalabilidad. Gran parte de la población nunca accede a este tipo de formación, y quienes lo hacen enfrentan el problema de la degradación de habilidades (skill decay) con el tiempo si no hay un re-entrenamiento periódico, un desafío bien documentado en la literatura sobre resucitación (Perkins et al., 2018). El resultado es una sociedad con "islas" de individuos capacitados en un "océano" de personas sin preparación.

1.2. Pregunta problema

¿Cómo diseñar un sistema de entrenamiento en realidad virtual fundamentado en el curso de primer respondiente para fortalecer las competencias de actuación de primeros respondientes civiles ante eventos catastróficos?

2. OBJETIVOS

2.1.Objetivo general

Desarrollar un sistema de entrenamiento en realidad virtual fundamentado en el curso de primer respondiente para fortalecer las competencias de actuación de primeros respondientes civiles ante eventos catastróficos.

2.2. Objetivos específicos

1. Identificar y documentar los contenidos, protocolos y competencias del curso de primer respondiente a ser representados en el sistema de realidad virtual.

2. Definir los escenarios de emergencia, mecánicas de interacción y estructura del sistema de entrenamiento en realidad virtual.

3. Desarrollar e implementar el sistema integrando los componentes de realidad virtual, escenarios de emergencia e interfaz de usuario.

4. Verificar el funcionamiento técnico del sistema mediante pruebas de rendimiento y funcionalidad.

3. ALCANCES Y LIMITACIONES

3.1. Alcances

3.1.1. Curso de Primer Respondiente

El curso de primer respondiente es una formación orientada a capacitar a personas sin formación médica profesional para actuar de manera inmediata y adecuada ante una emergencia, mientras llegan los servicios especializados. Este tipo de curso contempla conocimientos básicos de atención prehospitalaria, incluyendo evaluación primaria de la víctima, reanimación cardiopulmonar (rcp), manejo de obstrucción de la vía aérea, control de hemorragias, inmovilización de fracturas y protocolos de actuación ante desastres naturales como sismos.

El presente proyecto toma como base conceptual estos contenidos, con el objetivo de facilitar y fortalecer el conocimiento práctico de la población mediante una experiencia inmersiva en realidad virtual (rv), permitiendo la simulación controlada de escenarios críticos sin riesgo real.

3.1.2. Alcance funcional del proyecto

El alcance del proyecto comprende el diseño, desarrollo e implementación de una aplicación de realidad virtual enfocada en la simulación interactiva de tres situaciones de emergencia contempladas en el curso de primer respondiente:

Lobby principal

Se desarrolla un entorno inicial que permite:

- Familiarizar al usuario con la realidad virtual.
- Implementar un sistema de locomoción por teletransporte.
- Incorporar mecánicas básicas de interacción (apuntar, seleccionar y manipular objetos).
- Permitir la selección de los módulos de entrenamiento.
- Este espacio funciona como entorno de introducción y adaptación a los controles del sistema.

Módulo de RCP

Simula una situación en la que el usuario debe:

- Realizar maniobras de reanimación cardiopulmonar (rcp).
- Ejecutar maniobras de desobstrucción de la vía aérea (maniobra de heimlich).
- El sistema detecta posicionamiento y movimiento de las manos para validar la ejecución de las acciones.

Módulo de heridas y fracturas

Simula un escenario de accidente donde el usuario debe:

- Utilizar elementos del entorno virtual.
- Inmovilizar correctamente una fractura de brazo mediante férulas u objetos del entorno.
- Se prioriza la interacción con objetos y la secuencia correcta de atención.

Módulo de sismos

Simula un evento sísmico en un entorno interior donde el usuario debe:

- Identificar riesgos inmediatos.
- Tomar elementos necesarios si es pertinente.
- Buscar un sitio seguro dentro del entorno.
- Aplicar protocolos básicos de autoprotección.

Alcance técnico

El desarrollo incluye:

- Uso del motor de videojuegos unity como plataforma principal.
- Programación en lenguaje c#.
- Creación de activos 3d mediante blender, autodesk maya, zbrush y adobe photoshop.
- Implementación de un sistema básico de telemetría para registrar acciones del usuario.
- Optimización para sistemas pc-vr utilizando como hardware de referencia el visor vr box.
- El entregable final corresponde a una aplicación funcional, empaquetada y operativa.

3.2. Limitaciones

3.2.1. Limitaciones de recurso humano

El equipo de desarrollo estuvo conformado únicamente por dos integrantes, lo que implicó:

- Priorización de funcionalidades.
- Limitación a tres módulos de entrenamiento.
- Exclusión de escenarios adicionales de emergencia.
- No implementación de inteligencia artificial avanzada para los avatares.
- No desarrollo de un sistema de perfiles de usuario persistente.
- La velocidad de desarrollo estuvo directamente condicionada por la capacidad operativa del equipo.

3.2.2. Limitaciones Tecnológicas

El desarrollo estuvo restringido al hardware disponible en la universidad, específicamente el visor vr box para pc-vr. Esto implicó:

- No realizar pruebas en visores autónomos.
- No validar compatibilidad multiplataforma.
- Ajustar la complejidad gráfica a la capacidad del equipo de cómputo disponible.
- Dependencia de licencias de software disponibles institucionalmente.

3.2.3. Limitaciones Investigativas

El proyecto se enfocó en el desarrollo del artefacto tecnológico, por lo que quedó fuera del alcance:

- La validación experimental de la efectividad pedagógica.
- Estudios estadísticos sobre retención de conocimiento.
- Evaluaciones comparativas frente a métodos tradicionales.
- La implementación de un estudio formal habría requerido recursos metodológicos y temporales adicionales propios de investigaciones en ciencias sociales o educación.

4. JUSTIFICACIÓN

El desarrollo de un sistema de entrenamiento en realidad virtual para la prevención y respuesta ante catástrofes se justifica como una respuesta directa y tecnológicamente superior a las profundas deficiencias de los métodos de capacitación tradicionales. La eficacia de estos métodos se ve comprometida por una brecha fundamental entre el conocimiento teórico y la capacidad de acción efectiva bajo estrés. La pertinencia de este proyecto no radica en la mera innovación tecnológica, sino en su capacidad para transformar a una ciudadanía informada en una población genuinamente preparada. Como lo señala Heldring et al., 2024), el entrenamiento en rv de alta fidelidad para primeros respondedores permite escenarios realistas y repetibles que aumentan la confianza y preparación de los participantes, lo que subraya la necesidad de fortalecer la cadena de supervivencia desde su primer eslabón: el ciudadano común.

Desde una perspectiva social y humana, el núcleo del problema no reside en la falta de información, sino en la parálisis cognitiva que impide la aplicación de dicho conocimiento. La capacitación tradicional, basada en simulacros de baja fidelidad emocional, no logra acondicionar al individuo para superar esta barrera psicológica. Aquí reside la justificación más crucial del proyecto: la realidad virtual, a través del principio de inoculación del estrés, permite sumergir al usuario en un entorno que replica de manera segura pero realista los estímulos de una emergencia. Investigaciones sobre el comportamiento en crisis confirman que el entrenamiento bajo condiciones simuladas de estrés mejora la toma de decisiones y reduce las respuestas de pánico (Moya et al., 2017). Al exponer sistemáticamente al individuo a la urgencia de una emergencia, se facilita la automatización de respuestas correctas, forjando una memoria procedimental robusta que puede activarse instintivamente. Así, la solución propuesta trasciende la simple enseñanza para convertirse en un verdadero entrenamiento conductual.

Pedagógicamente, el proyecto se justifica al proponer un cambio de paradigma, de un modelo centrado en la enseñanza a uno centrado en el aprendizaje. La rv transforma conceptos abstractos en experiencias vividas, donde el conocimiento se ancla a la acción. Más importante aún, el sistema aborda la crítica deficiencia de evaluación objetiva. Mientras un simulacro ofrece retroalimentación general, el entorno virtual funciona como un laboratorio de desempeño. Esto permite un ciclo de retroalimentación inmediato y personalizado, que, tal y como explican Ericsson, Krampe, & Tesch-Römer, 1993), es esencial para la adquisición de

competencias complejas a través de la práctica deliberada, un principio clave en la formación de habilidades expertas. Esta capacidad de análisis no solo empodera al individuo, sino que también ofrece una herramienta sin precedentes para medir la efectividad real de los programas de capacitación.

A nivel tecnológico y operativo, la rv ofrece una solución pragmática a las barreras de escalabilidad y seguridad. La organización de simulacros físicos es logísticamente compleja y costosa. El software de entrenamiento, en cambio, es infinitamente escalable, democratizando el acceso a una capacitación de alta calidad. Además, es éticamente imposible replicar los escenarios más críticos. La rv elimina este riesgo, permitiendo a los usuarios entrenar en un entorno 100% seguro. El potencial de las tecnologías inmersivas para transformar la educación en salud y emergencias es vasto, ofreciendo escenarios complejos y repetibles que son inviables en el mundo físico. En este sentido, Wu et al., (2024) muestran que la rv y la ar aplicadas al entrenamiento en reanimación cardiopulmonar pueden ser tan efectivas como los métodos tradicionales, validando la pertinencia de su uso en entornos críticos.

Este proyecto se justifica desde una perspectiva científica por su potencial para generar conocimiento empírico invaluable. El sistema es una plataforma de investigación para estudiar el comportamiento humano en crisis simuladas, ofreciendo insights para validar y refinar protocolos de emergencia. La brecha identificada no es solo tecnológica, sino también metodológica. Calisanie et al., 2025). evidencian que la simulación en rv mejora significativamente la preparación para desastres en comparación con métodos convencionales, incluso con efectos persistentes en el tiempo. Al crear un laboratorio virtual para el estudio de la toma de decisiones bajo presión, esta tesis no solo aplica conocimiento, sino que se posiciona para generar evidencia nueva y de alto impacto.

5. LÍNEAS DE INVESTIGACIÓN

El presente trabajo de grado se inscribe y contribuye de manera significativa a dos de las líneas de investigación institucionales, establecidas formalmente por el consejo académico de la universidad de cundinamarca (2021) en su acuerdo 009. La naturaleza interdisciplinaria del proyecto, que fusiona la ingeniería de software con la capacitación para la respuesta a emergencias, encuentra su fundamentación académica en los siguientes campos de estudio.

Aprendizaje, conocimiento, tecnologías, comunicación y digitalización

Esta línea constituye el eje central del proyecto. El desarrollo del sistema de entrenamiento en realidad virtual es una aplicación directa de tecnologías de vanguardia para abordar un desafío en la transferencia de conocimiento. La investigación se articula en torno a varios componentes clave de esta línea.

Tecnologías y digitalización:

El núcleo del trabajo es la creación de un artefacto tecnológico (software de rv) a través de la digitalización de escenarios de emergencia del mundo real. Se exploran las capacidades y limitaciones de las tecnologías inmersivas como plataforma para la simulación.

Aprendizaje y conocimiento:

El sistema está diseñado explícitamente como una herramienta para facilitar el aprendizaje procedimental. Su arquitectura y funcionalidades se basan en teorías del aprendizaje experiencial, buscando transformar el conocimiento teórico de los protocolos de primera respuesta en competencias prácticas y aplicables.

Comunicación:

El proyecto investiga una nueva forma de comunicación formativa, donde la interfaz de rv actúa como mediadora entre el conocimiento experto (los protocolos) y el usuario aprendiz, utilizando la interacción inmersiva y la retroalimentación multisensorial como canal principal.

6. MARCO TEÓRICO

6.1. Marco Referencial

El entrenamiento mediante entornos virtuales constituye una estrategia formativa que posibilita la práctica de procedimientos en escenarios simulados que reproducen condiciones reales de emergencia. A través de esta modalidad, los usuarios pueden enfrentarse a situaciones críticas relacionadas con primeros auxilios, tales como paro cardiorrespiratorio, lesiones traumáticas y eventos sísmicos, dentro de un espacio digital diseñado para el aprendizaje.

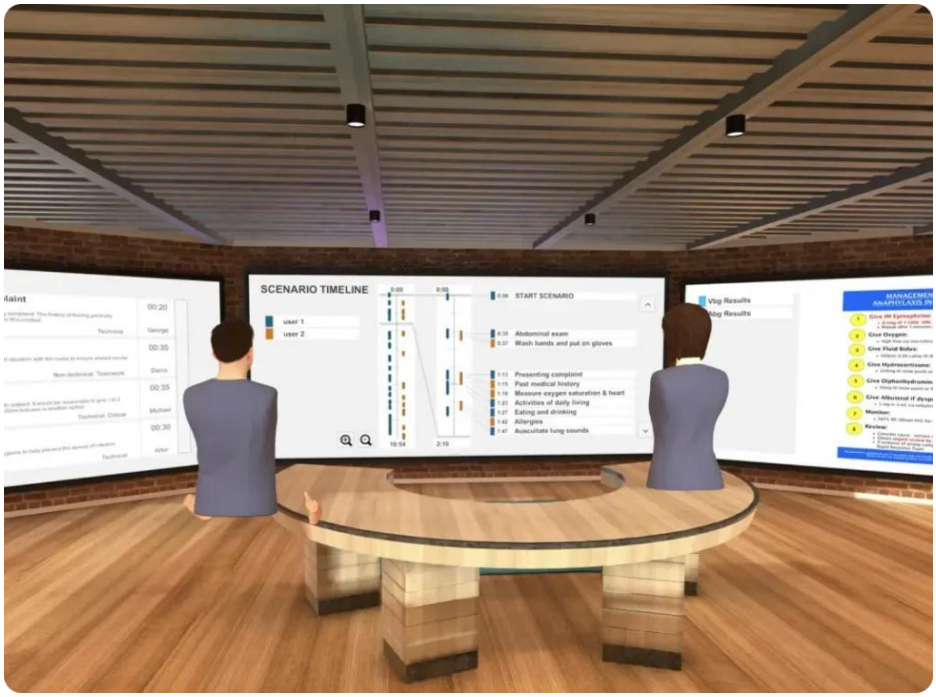
El sistema desarrollado recrea contextos de riesgo basados en los contenidos del curso de primer respondiente, permitiendo la interacción con objetos, herramientas y personajes virtuales. De esta manera, el participante no solo observa el procedimiento, sino que ejecuta acciones concretas, toma decisiones y sigue protocolos establecidos, fortaleciendo así la comprensión práctica de los conocimientos adquiridos previamente en teoría.

Oxford medical simulation (oms)

Para el desarrollo del presente proyecto se toma como referencia la plataforma **oxford medical simulation (oms)**, desarrollada por la empresa británica oxford medical simulation. Esta herramienta utiliza entornos de realidad virtual para entrenar a profesionales de la salud en la toma de decisiones clínicas en situaciones de emergencia, tales como paro cardiorrespiratorio, shock séptico y fallos respiratorios (Oxford Medical Simulation, 2020).

El sistema funciona mediante el uso de visores de realidad virtual que permiten al usuario interactuar con pacientes virtuales en escenarios hospitalarios simulados. A diferencia de los simuladores tradicionales basados en maniquíes físicos, oms presenta escenarios dinámicos donde el estado del paciente evoluciona según las decisiones tomadas por el participante.

En la práctica, la plataforma expone al usuario a un caso clínico donde debe evaluar signos vitales, solicitar exámenes, administrar tratamientos y responder ante cambios críticos. Durante la simulación, el software registra las acciones realizadas y genera un informe detallado que permite analizar la toma de decisiones, tiempos de respuesta y adherencia a protocolos clínicos (Oxford Medical Simulation, 2020).



Ilustracion: simulador oxford medical simulation (oms),

fuente: https://oxfordmedicalsimulation.com

Health scholars – vr cpr training

Otro referente relevante es el sistema de entrenamiento en reanimación cardiopulmonar desarrollado por **health scholars inc.**, el cual integra simulación en realidad virtual para capacitar en rcp básica y avanzada (Health Scholars, 2019).

Esta plataforma utiliza dispositivos vr para recrear escenarios de emergencia prehospitalaria, donde el usuario debe evaluar el estado de la víctima, activar el sistema de emergencias y aplicar compresiones torácicas siguiendo la frecuencia y profundidad recomendadas por guías internacionales.

El simulador proporciona retroalimentación inmediata respecto al ritmo de compresiones, posicionamiento y secuencia del procedimiento. Además, genera métricas de desempeño que permiten evaluar la precisión técnica del usuario y su capacidad de respuesta ante situaciones críticas (Health Scholars, 2019).

En la práctica, este sistema ha demostrado mejorar la retención de habilidades en comparación con métodos exclusivamente teóricos, al permitir la repetición constante del procedimiento en un entorno inmersivo.

ilustración: simulación ems health scholars – vr cpr training

fuente: https://healthscholars.com

Simx – vr medical simulation system

El sistema **simx** es una plataforma de simulación médica inmersiva utilizada por hospitales y fuerzas militares para entrenamiento clínico avanzado (SimX, 2021).

Simx emplea visores de realidad virtual para crear entornos tridimensionales interactivos donde el usuario puede atender pacientes virtuales en situaciones de trauma, hemorragias, fracturas y emergencias respiratorias. El sistema permite interacción con instrumentos médicos virtuales y toma de decisiones clínicas en tiempo real.

En la práctica, el simulador presenta escenarios complejos en los cuales el usuario debe identificar lesiones, priorizar intervenciones y aplicar protocolos establecidos. La plataforma registra datos de desempeño, permitiendo evaluar habilidades técnicas y cognitivas.

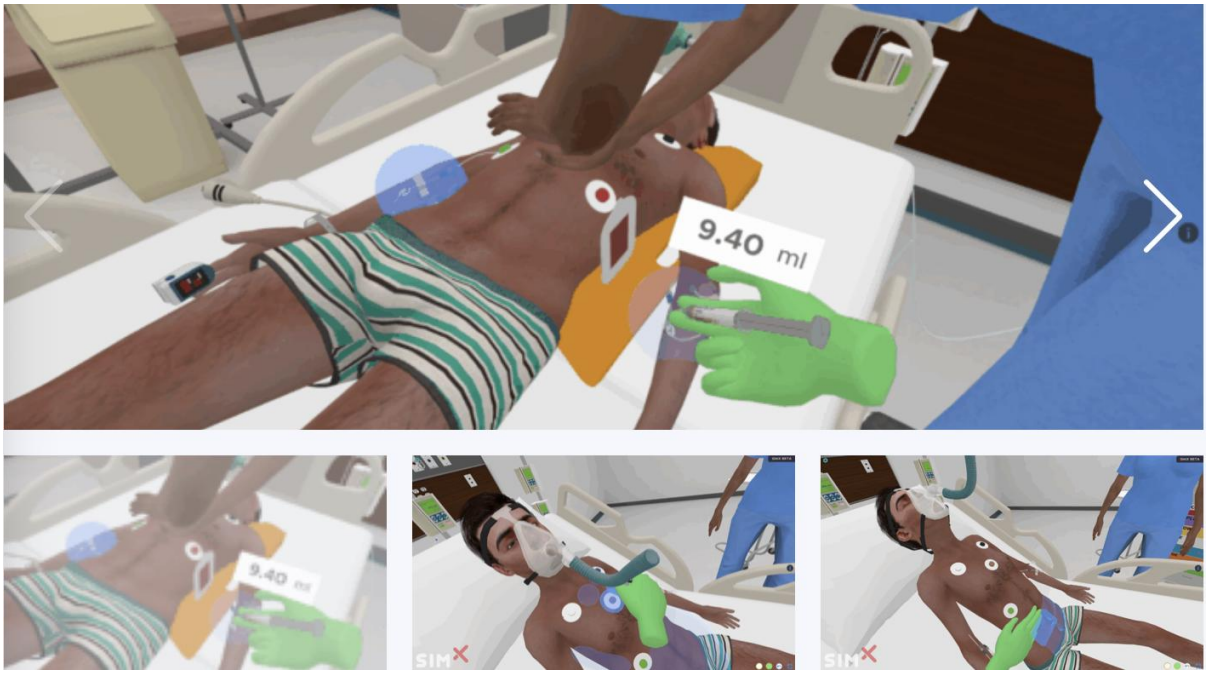


ilustración: simx – vr medical simulation system

Fuente: https://www.simxvr.com

Teorías del aprendizaje

La premisa central de este proyecto es que el entrenamiento para emergencias no puede limitarse a la transmisión pasiva de información. Debe ser un proceso activo que construya competencias robustas y aplicables. Para ello, nos apoyamos en teorías del aprendizaje que priorizan la experiencia como motor del conocimiento.

Constructivismo y aprendizaje experiencial

En contraposición a los modelos conductistas que ven al aprendiz como un receptor pasivo, el constructivismo postula que el conocimiento se "construye" activamente a través de la interacción del individuo con su entorno esto complementa la idea principal del proyecto como tal. Una de sus vertientes más influyentes es el modelo de aprendizaje experiencial de david kolb. Esta teoría genera una propuesta que el aprendizaje significativo es un ciclo continuo de cuatro etapas: experiencia concreta, donde el individuo se involucra en una nueva experiencia; observación reflexiva, donde reflexiona sobre dicha experiencia desde múltiples perspectivas; conceptualización abstracta, donde integra sus observaciones en conceptos lógicos y teorías; y experimentación activa, donde utiliza esas teorías para tomar decisiones y resolver problemas, lo que a su vez genera nuevas experiencias.

La relevancia de este modelo para el proyecto es directa y profunda. Un sistema de entrenamiento en realidad virtual (rv) es la encarnación tecnológica del ciclo de kolb. La simulación proporciona la experiencia concreta de una emergencia de manera segura. El sistema de evaluación y la posibilidad de revisar la propia actuación facilitan la observación reflexiva. Los debriefings o resúmenes al final de cada escenario ayudan a la conceptualización abstracta, conectando las acciones del usuario con los protocolos correctos. Finalmente, la capacidad de repetir el escenario permite la experimentación activa, probando diferentes estrategias hasta alcanzar la maestría. Como lo expone García-Bullé, (2020) en su análisis sobre tecnologías educativas, el aprendizaje inmersivo facilita un paso fundamental del "saber qué" (conocimiento declarativo) al "saber cómo" (conocimiento procedimental), que es precisamente el objetivo de la capacitación para primeros respondientes.

Figura 1

ciclo de kolb

Nota. El diagrama ilustra el proceso de aprendizaje continuo donde la experiencia se transforma en conocimiento. Adaptado de Kolb (1984).

Teoría del aprendizaje situado

Propuesta por Lave & Wenger, (1991), esta teoría argumenta que el aprendizaje no es una actividad puramente cognitiva y aislada, sino un proceso social y cultural que está ligado al contexto en el que ocurre. Sostiene que el conocimiento es más duradero y transferible cuando se adquiere en una "comunidad de práctica" y en un contexto auténtico, similar a aquel donde se aplicará. El aula tradicional o un manual de primeros auxilios representan contextos desvinculados de la realidad de una emergencia. En contraste, un entorno de rv bien diseñado constituye un contexto situado de alta fidelidad. Al simular no solo las tareas, sino también las presiones ambientales, sociales y emocionales de una emergencia, el sistema propuesto crea las condiciones para que el aprendizaje sea más significativo y la transferencia de habilidades al mundo real sea mucho más probable.

Gamificación como estrategia de aprendizaje

Deterding et al., (2011) definen la gamificación como el uso de elementos y mecánicas de diseño de videojuegos en contextos no lúdicos con el fin de aumentar la motivación, el engagement y la participación activa del usuario. A diferencia de los juegos puramente recreativos, la gamificación aplicada al aprendizaje busca aprovechar los mecanismos psicológicos que hacen que los juegos sean intrínsecamente motivadores, como la progresión, el reto, la retroalimentación inmediata y la sensación de logro, para potenciar experiencias formativas más efectivas (Kapp, 2012).

En el contexto del presente proyecto, la gamificación no es un elemento decorativo sino una decisión de diseño fundamentada pedagógicamente. El sistema incorpora mecánicas de juego que cumplen funciones específicas en el proceso de aprendizaje: la retroalimentación inmediata permite al usuario comprender las consecuencias de sus decisiones en tiempo real, la progresión de dificultad mantiene un nivel de desafío apropiado que evita tanto la frustración como el aburrimiento, y la repetibilidad sin consecuencias reales crea un entorno psicológicamente seguro donde el error se convierte en oportunidad de aprendizaje. Juntos, estos elementos favorecen la práctica deliberada y sostenida que, según la literatura sobre adquisición de habilidades, es fundamental para el desarrollo de competencias robustas y aplicables (Ericsson et al., 1993).

Síntesis del marco pedagógico

Las teorías expuestas no operan de manera aislada sino que se integran de forma coherente en el diseño del sistema de entrenamiento. El constructivismo y el aprendizaje experiencial de kolb fundamentan la decisión de que el usuario aprenda mediante la acción y la reflexión, no mediante la recepción pasiva de información. El aprendizaje situado justifica la creación de escenarios de alta fidelidad que replican las condiciones reales de una emergencia, favoreciendo la transferencia de habilidades. La gamificación aporta las mecánicas que sostienen la motivación y hacen del error una herramienta de aprendizaje.

Es importante señalar que el proyecto adopta un enfoque centrado en el aprendizaje, donde el usuario es el protagonista activo de su proceso formativo. El sistema no pretende "enseñar" en el sentido instruccional tradicional, sino facilitar un entorno donde el usuario pueda construir, practicar y refinar sus competencias de actuación ante emergencias. La validación experimental de la efectividad pedagógica del sistema constituye un estudio posterior que excede el alcance del presente proyecto.

Teorías psicológicas de la respuesta a emergencias

Comprender cómo funciona la mente humana bajo estrés es indispensable para diseñar un entrenamiento que realmente funcione cuando más se necesita. Este pilar se enfoca en las bases psicológicas que justifican el uso de la simulación para mejorar la resiliencia y la toma de decisiones.

Modelo de inoculación de estrés (mie)

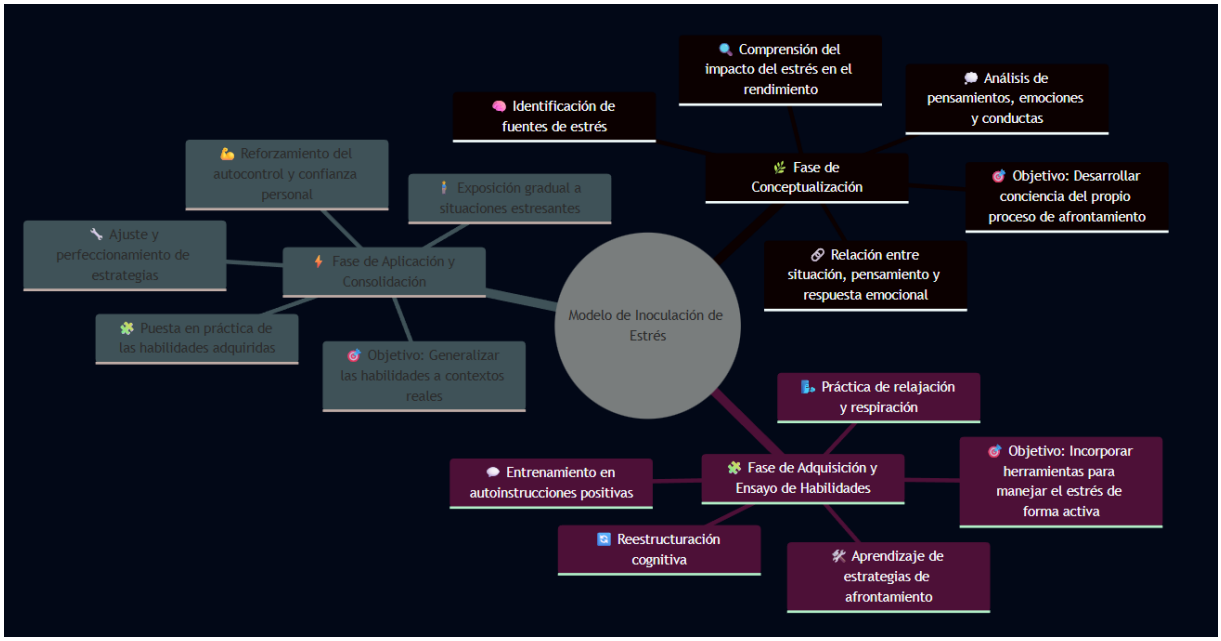
Desarrollado por donald meichenbaum, este modelo cognitivo-conductual es uno de los pilares del presente documento. El proceso consta de tres fases, conceptualización, donde el individuo entiende la naturaleza de su respuesta al estrés; adquisición de habilidades, donde aprende y practica técnicas específicas para manejarlo; y aplicación, donde pone a prueba esas habilidades en situaciones de estrés progresivamente más intensas.

El sistema de entrenamiento en rv es una herramienta ideal para implementar el mie. La simulación expone al usuario a los estresores de una emergencia (presión de tiempo, estímulos visuales y auditivos caóticos, la "responsabilidad" por la vida de un avatar) en un entorno seguro. Los escenarios pueden ser diseñados con niveles de dificultad crecientes,

permitiendo una exposición gradual. Al mismo tiempo que enfrenta el estrés simulado, el usuario practica las habilidades de afrontamiento (los protocolos de primera respuesta), fortaleciendo su autoeficacia y automatizando su comportamiento. Como explica bados (2020) en su tratado sobre técnicas cognitivo-conductuales, el ensayo de conducta en escenarios simulados es un componente esencial para reducir la ansiedad y mejorar el desempeño en situaciones temidas, validando el enfoque de este proyecto para mitigar la parálisis por pánico.

Figura 2

fases del modelo de inoculación de estrés



Nota. Adaptación basada en el modelo de inoculación de estrés de donald Meichenbaum. Elaboración propia

Toma de decisiones naturalista (tdn) y el modelo de reconocimiento primario

En una emergencia, no hay tiempo para un análisis deliberado y exhaustivo de todas las opciones. Las teorías clásicas de toma de decisiones racionales no aplican en ninguno de los módulos de este caso. El campo de la tdn estudia cómo las personas, especialmente los expertos, toman decisiones en entornos complejos, inciertos y de alto riesgo. El modelo de reconocimiento primario de decisiones (rpd) de gary klein es particularmente relevante. Sostiene que los respondientes experimentados no comparan opciones, sino que utilizan su vasta experiencia para reconocer patrones en la situación. Este reconocimiento activa un

"guion" de acción plausible, que es evaluado mentalmente y ejecutado. Si funciona, se continúa, si no, se ajusta.

El objetivo de este sistema de entrenamiento es acelerar la adquisición de esta "experiencia" a través de la repetición de múltiples escenarios de los distintos modelos que se tienen. Al enfrentarse a diversas emergencias simuladas (desmayos, quemaduras, sismos), el usuario comienza a construir un repertorio de patrones y sus correspondientes "guiones" de acción. La simulación permite condensar años de posible experiencia en horas de entrenamiento, desarrollando una intuición entrenada que es la base para una toma de decisiones rápida y efectiva en el mundo real.

Teorías de la interacción humano-computador (hci) y la realidad virtual

La efectividad de la simulación no depende solo de la fidelidad gráfica, sino de la calidad de la interacción y la capacidad del sistema para hacer que el usuario se sienta "allí". Este pilar aborda las bases teóricas que guían el diseño de la experiencia virtual.

Teoría de la presencia y la inmersión

Es crucial distinguir estos dos conceptos. La inmersión es una cualidad tecnológica del sistema, definida por su capacidad para aislar al usuario de los estímulos del mundo real y presentarle un entorno virtual consistente y amplio (ej. Un campo de visión de 360°, audio especializado).

La meta de cualquier sistema de entrenamiento en rv es maximizar la inmersión tecnológica para inducir un alto grado de presencia. Si el usuario no se siente "presente", la simulación se convierte en un simple videojuego y su valor formativo se diluye. Por ello, el diseño de la interfaz, la latencia del sistema, la coherencia de las interacciones y la credibilidad del entorno son aspectos guiados por esta teoría. Como resume (Vera, 2022), la capacidad de la rv para generar presencia es su principal ventaja diferencial en el ámbito educativo, ya que transforma al estudiante de un observador a un participante activo en el fenómeno estudiado. El diseño de nuestro sistema se centrará en optimizar los factores tecnológicos que, según la literatura de hci, son los principales predictores de la sensación de presencia.

6.2. Marco conceptual

6.2.1. Conceptos de atención prehospitalaria y gestión del riesgo

Primer respondiente

Se denomina primer respondiente a la persona que actúa inicialmente ante una situación de emergencia, brindando atención básica mientras se activa el sistema de emergencias. Su función principal es estabilizar la escena, evaluar el estado de la víctima y aplicar procedimientos básicos de primeros auxilios hasta la llegada de personal especializado (IFRC, 2018).

Reanimación cardiopulmonar (rcp)

Es un procedimiento de emergencia que combina compresiones torácicas y ventilaciones con el objetivo de mantener la circulación sanguínea y la oxigenación en una persona que ha sufrido un paro cardiorrespiratorio. Su aplicación inmediata aumenta significativamente las probabilidades de supervivencia (American Heart Association, 2020).

Fractura

Se define como la pérdida de continuidad de un hueso causada generalmente por un trauma o impacto. En la atención inicial, el manejo adecuado consiste en evitar el movimiento de la zona afectada y realizar una inmovilización provisional para prevenir complicaciones adicionales (OMS, 2019).

Herida

Es una lesión que compromete la integridad de los tejidos blandos del cuerpo. Puede clasificarse según su profundidad y mecanismo de producción. El tratamiento inicial se basa en la limpieza, control del sangrado y protección de la zona afectada (Cruz Roja Española, 2017).

Protocolo p.a.s. (proteger, avisar, socorrer)

Es una secuencia de actuación utilizada en primeros auxilios. Primero se debe garantizar la seguridad del entorno, posteriormente activar el sistema de emergencias y finalmente brindar asistencia a la persona afectada (IFRC, 2018)

Gestión del riesgo

Conjunto de acciones orientadas a identificar, evaluar y reducir los riesgos ante posibles eventos adversos. En el contexto de desastres naturales como los sismos, implica la preparación, respuesta y recuperación frente a emergencias (UNDRR, 2015).

Sismo

Movimiento brusco de la corteza terrestre producido por la liberación de energía acumulada en el interior del planeta. Puede generar daños estructurales y situaciones de riesgo que requieren protocolos específicos de autoprotección y evacuación (USGS, 2022).

6.2.2. Conceptos tecnológicos

Realidad virtual

Entorno digital tridimensional generado por sistemas informáticos que permite al usuario interactuar con escenarios simulados en tiempo real, generando una sensación de inmersión (Burdea & Coiffet, 2003)..

Simulación

Proceso mediante el cual se recrea una situación real a través de un modelo controlado, con el propósito de analizar comportamientos o entrenar habilidades sin exponer al usuario a riesgos reales (Banks et al., 2010).

Raycast

Técnica utilizada en motores gráficos que consiste en proyectar un rayo virtual desde un punto determinado para detectar colisiones con objetos dentro del entorno digital (Unity Technologies, 2021).

Collider

Es un componente que permite detectar colisiones entre objetos dentro del entorno virtual. Su función es garantizar la interacción física coherente entre el usuario y los elementos del escenario (Unity Technologies, 2021).

6.2.3. Conceptos de ingeniería.

Asset

Elemento digital que puede integrarse dentro de un proyecto de desarrollo interactivo. Puede incluir modelos tridimensionales, texturas, sonidos, animaciones o scripts que forman parte del entorno virtual (Unity Technologies, 2021).

Blender

Software de creación tridimensional utilizado para el diseño y edición de modelos 3d. Permite desarrollar objetos, escenarios y animaciones que posteriormente pueden integrarse en motores de desarrollo para su uso en aplicaciones interactivas (Blender Foundation, 2023).

Casos de uso

Herramienta de análisis empleada en ingeniería de software para describir la interacción entre un usuario y el sistema, especificando las acciones que conducen a un resultado funcional dentro de la aplicación (Jacobson, 1992).

Interfaz de usuario

Conjunto de elementos visuales y funcionales que permiten la comunicación entre el usuario y el sistema. Incluye menús, botones, indicadores y demás componentes que facilitan la navegación e interacción dentro del entorno virtual (Shneiderman et al., 2018).

Modelado 3d

Proceso de construcción digital de objetos en tres dimensiones mediante software especializado, con el propósito de representar elementos reales dentro de un entorno virtual interactivo (Vince, 2011).

Google cardboard

Es un visor de realidad virtual de bajo costo diseñado para funcionar con dispositivos móviles. Permite visualizar entornos tridimensionales mediante el uso de lentes que generan sensación de profundidad e inmersión. En el desarrollo del proyecto, este dispositivo se emplea como medio de acceso al sistema, garantizando portabilidad, accesibilidad económica y facilidad de uso (Google VR, 2019).

Realidad virtual

Tecnología que genera entornos digitales tridimensionales con los que el usuario puede interactuar en tiempo real, creando una sensación de presencia dentro del escenario simulado (Sherman & Craig, 2019).

6.3. Marco ingenieril

Unity

Es un motor de desarrollo multiplataforma orientado a la creación de aplicaciones interactivas en dos y tres dimensiones. Su arquitectura se basa en un sistema de componentes, donde cada objeto dentro de la escena puede incorporar diferentes módulos como físicas, renderizado y programación.

"unity proporciona un entorno flexible y extensible para la creación de experiencias interactivas en tiempo real" (Unity Technologies, 2021). Esta característica lo convierte en una herramienta ampliamente utilizada en el desarrollo de simuladores y aplicaciones de realidad virtual.

En el desarrollo del sistema de entrenamiento en atención de emergencias, unity permitió integrar escenarios tridimensionales, configurar colisiones, programar eventos y gestionar la interacción del usuario mediante scripts en c#. Además, facilitó la exportación del proyecto a dispositivos android, requisito fundamental para su ejecución en visores de realidad virtual móvil.

El entorno de trabajo de unity ofrece múltiples herramientas que fortalecen el desarrollo del simulador:

- Editor integrado para diseño de escenas y configuración de iluminación.
- Sistema de componentes para añadir funcionalidades específicas a cada objeto.
- Motor de físicas para simulación de gravedad y colisiones (Millington, 2010).
- Sistema de interfaz gráfica (ui) para retroalimentación visual.
- Exportación multiplataforma compatible con dispositivos móviles (Unity Technologies, 2021).

Google cardboard

Es un visor de realidad virtual diseñado para funcionar con teléfonos inteligentes, permitiendo la visualización de entornos tridimensionales mediante lentes que generan efecto estereoscópico.

"la realidad virtual móvil permite experiencias inmersivas utilizando el hardware disponible en dispositivos inteligentes" (Sherman & Craig, 2019). En este contexto, google cardboard representa una alternativa accesible para implementar entornos de simulación sin requerir equipos especializados de alto costo.

El sistema utiliza sensores del teléfono móvil como el acelerómetro y el giroscopio para detectar el movimiento de la cabeza del usuario y reflejarlo en el entorno virtual en tiempo real, generando sensación de presencia.

Las características técnicas que influyeron en el desarrollo del proyecto fueron:

- Dependencia del hardware del dispositivo móvil.
- Limitaciones en capacidad gráfica y procesamiento.
- Interacción basada en la mirada (gaze interaction).
- Necesidad de optimización para mantener estabilidad en la tasa de fotogramas (lavalle, 2017).

Estas condiciones determinaron decisiones de diseño enfocadas en la optimización del rendimiento y la simplicidad de interacción.

Blender

Es una herramienta de modelado tridimensional utilizada para la creación y edición de objetos 3d, ampliamente empleada en proyectos de animación, simulación y desarrollo interactivo.

En el proyecto, blender fue utilizado para diseñar objetos personalizados necesarios para representar escenarios de emergencia, incluyendo estructuras arquitectónicas y elementos interactivos.

Entre sus principales funcionalidades se encuentran:

- Modelado poligonal.
- Aplicación de materiales y texturas.
- Exportación en formatos compatibles con motores gráficos.
- Optimización básica de geometría para dispositivos móviles.

Lenguaje de programación c#

Es un lenguaje de programación orientado a objetos utilizado en el entorno .net y ampliamente integrado en el motor unity.

En el simulador, c# permitió programar la lógica de interacción, validación de procedimientos y control de eventos dentro de cada nivel. La implementación incluyó el uso de estructuras condicionales, control de temporizadores y gestión de colisiones.

Entre sus principales características aplicadas en el proyecto se destacan:

- Programación orientada a objetos.
- Manejo estructurado de excepciones.
- Recolección automática de memoria.
- Integración directa con componentes del motor unity (Unity Technologies, 2021).

Diseño e implementación de los niveles del simulador

El sistema fue estructurado en tres niveles independientes, cada uno orientado a representar una situación específica basada en el curso de primer respondiente. Cada escenario fue desarrollado como una escena modular dentro del motor de desarrollo.

Desde el enfoque ingenieril, cada nivel integra componentes físicos, lógicos y de interfaz que permiten validar las acciones realizadas por el usuario y ofrecer retroalimentación inmediata.

Simulación de sismo

Este nivel recrea un entorno afectado por un evento sísmico, donde el usuario debe recolectar objetos prioritarios y dirigirse a una zona segura.

Desde el punto de vista técnico se implementaron:

- Objetos interactivos con colliders tipo trigger.
- Sistema de selección mediante raycast por mirada.
- Validación de objetos recolectados mediante scripts en c#.
- Zona segura configurada como área condicional de finalización.

La lógica del sistema verifica que se cumplan los objetivos antes de permitir completar el escenario.

Simulación de heridas y fracturas

Este escenario simula la inmovilización de una fractura en miembro superior, validando que el usuario seleccione el elemento correcto y lo ubique adecuadamente.

La implementación incluyó:

- Detección de selección correcta mediante raycast.
- Colliders específicos en puntos del modelo 3d.
- Validación de secuencia lógica del procedimiento.
- Retroalimentación visual en caso de error.

Simulación de rcp

Este nivel está orientado a la práctica de compresiones torácicas dentro de un entorno virtual controlado.

La implementación técnica incluye:

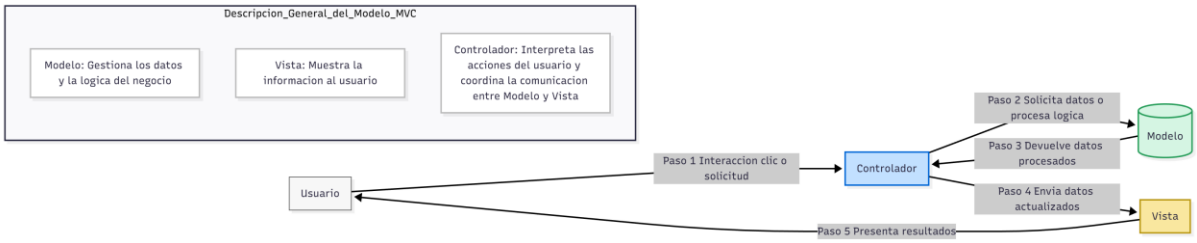
- Sistema de temporización para controlar ritmo de compresiones.
- Registro de repeticiones mediante eventos programados.
- Interfaz gráfica para mostrar progreso.
- Validación condicional basada en parámetros predefinidos.

Arquitectura de software

Para garantizar la modularidad, escalabilidad y mantenibilidad del sistema, se diseñó una arquitectura de software basada en una adaptación del patrón arquitectónico modelo-vista-controlador (mvc), muy común en el desarrollo de aplicaciones interactivas.

Figura 3

diagrama de responsabilidades entre la presentación.



Nota. *El diagrama ilustra la separación de responsabilidades entre la presentación (vista), la lógica de negocio (modelo) y la gestión de la interacción (controlador).*

El modelo

Representa el cerebro y la memoria de la simulación. Esta capa es completamente independiente de cómo se ven las cosas. Incluye:

Estado de la simulación: clases de c# que contienen los datos puros del escenario (ej. Signos vitales de la víctima, tiempo restante, puntuación del usuario).

Lógica de protocolos: un conjunto de scriptable objects y servicios que definen las reglas de cada protocolo de emergencia. Por ejemplo, la lógica que determina si una compresión de rcp es "correcta" reside aquí, sin saber nada sobre animaciones o modelos 3d.

Sistema de telemetría: el módulo responsable de registrar los eventos y el estado de la simulación en un formato de datos estructurado (json).

La vista

Es todo lo que el usuario ve y oye. Su única función es representar visualmente el estado del modelo.

Entornos y activos 3d: los modelos 3d, texturas, shaders y sistemas de partículas.

Interfaz de usuario (ui): los paneles virtuales, textos, temporizadores y barras de progreso.

Sistema de audio: el motor de audio espacializado que reproduce los sonidos de la simulación.

El controlador

Actúa como el intermediario. Su trabajo es capturar la entrada del usuario y traducirla en comandos para el modelo, y luego notificar a la vista que debe actualizarse.

Gestor de entrada: scripts que leen los datos de los controladores de vr (posición, rotación, estado de los botones).

Mecánicas de interacción: los scripts que definen qué sucede cuando el usuario "agarra" un objeto o "presiona" un botón. Por ejemplo, cuando el usuario realiza el gesto de compresión, el controlador lo detecta, envía el dato de "fuerza" y "posición" al modelo para que sea evaluado, y luego le dice a la vista que reproduzca la animación y el sonido correspondiente.

Flujos de proceso

Flujo de proceso: interacción y evaluación de rcp

1. **Inicio:** el usuario selecciona el módulo de rcp. El *gestor de escenarios* carga la escena y los activos correspondientes.
2. **Detección de interacción:** el *input manager* (controlador) detecta constantemente la posición de los controladores de vr.
3. **Validación de posición:** cuando el usuario coloca sus manos sobre el pecho del avatar, unos *colliders* invisibles en la vista notifican al *controlador de rcp*.
4. **Ejecución de acción:** el usuario realiza el movimiento de compresión. El controlador mide el desplazamiento vertical (deltay) y el tiempo transcurrido desde la última compresión (deltatime).
5. **Envío a modelo:** el controlador envía estos datos (deltay, deltatime) al *modelo de protocolo rcp*.

6. **Lógica de evaluación:** el modelo compara los datos recibidos con los umbrales definidos (ej. Profundidadcorrecta = 5cm, frecuenciaideal = 110 cpm). Determina si la compresión fue "correcta", "muy superficial" o "muy profunda".

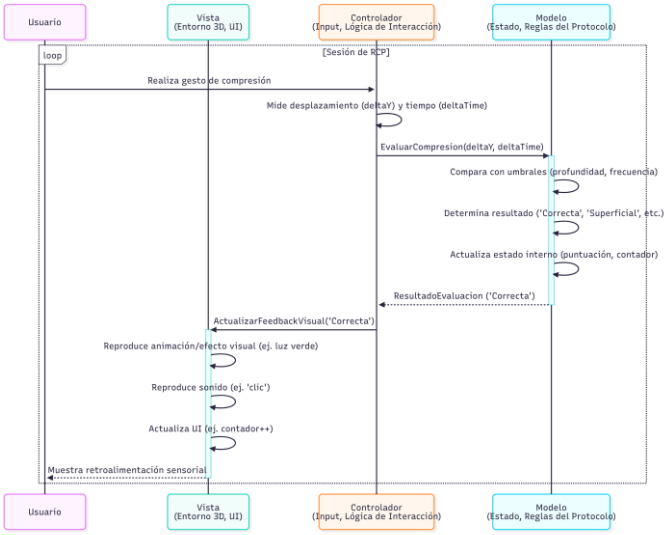
7. **Actualización de estado:** el modelo actualiza su estado interno (aumenta el contador de compresiones correctas, actualiza la puntuación).

8. **Notificación a vista:** el controlador notifica a la vista que el estado ha cambiado.

9. **Retroalimentación al usuario:** la vista responde proporcionando retroalimentación inmediata: reproduce un sonido de "clic" si fue correcta, muestra un indicador visual verde y actualiza la ui.

Figura 4

diagrama de secuencia del flujo mvc



Nota. *El diagrama representa la interacción secuencial entre los componentes de la arquitectura modelo-vista-controlador (mvc), elaboración propia.*

Flujo de proceso

1. **Modelado base (low-poly):** se crea la malla 3d base con una cantidad de polígonos optimizada en **blender** o **autodesk maya**.

2. **Mapeo uv:** se despliega la malla 3d en un espacio 2d (mapeo uv) para prepararla para el texturizado.

3. **Esculpido de detalles (high-poly):** el modelo base se importa en **zbrush**, donde se esculpen los detalles finos (arrugas, texturas de superficie, etc.), generando una versión con millones de polígonos.

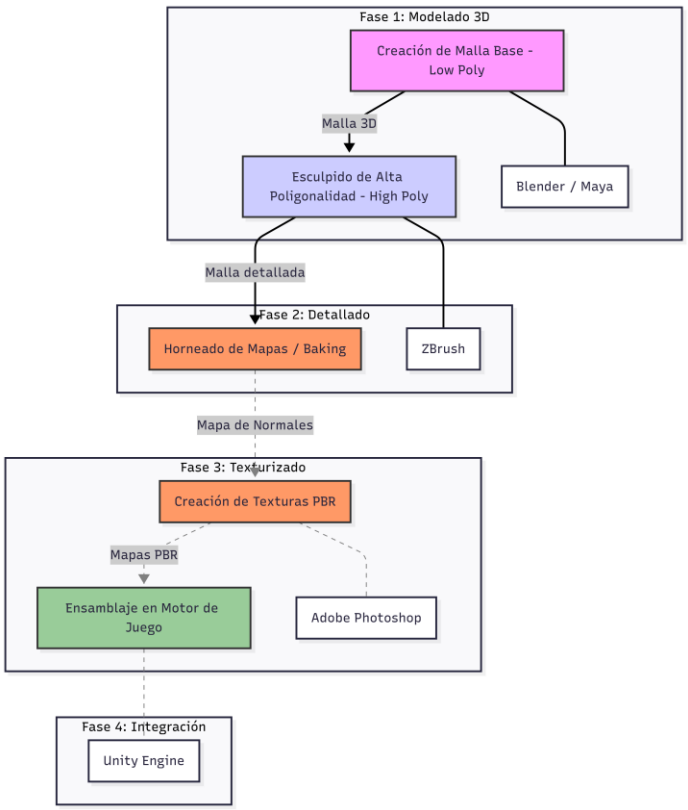
4. **Proceso de horneado (baking):** se utiliza el modelo de alta poligonalidad para "hornear" o transferir sus detalles a mapas de textura. El más importante es el **mapa de normales**, una textura que simula los detalles de la superficie sin añadir geometría real.

5. **Texturizado pbr:** en **adobe photoshop** (o un software como substance painter), se pintan los diferentes mapas de textura (color base, metalizado, rugosidad, etc.) Sobre el mapa uv, siguiendo el flujo de trabajo de renderizado basado en físicas (pbr).

6. **Integración en unity:** el modelo de baja poligonalidad (low-poly) y todos sus mapas de textura pbr se importan en unity. Se crea un material que utiliza estos mapas, logrando que un modelo ligero en términos de rendimiento aparente tener un nivel de detalle extremadamente alto.

Figura **5**

flujo de trabajo para el modelado y texturizado 3d



Nota. El diagrama muestra las fases del proceso de creación de un modelo tridimensional para entornos interactivos, elaboración propia.selección de herramientas y justificación, elaboración propia.

Tabla 1*: pila tecnológica utilizada en el desarrollo del sistema*

Fase	**Descripción**	**Herramientas comunes**
1. Concept art y referencias	Se desarrollan las ideas visuales iniciales, bocetos y recopilación de referencias que definen la estética y el estilo del proyecto.	Photoshop, procreate, pureref
2. Modelado 3d	Creación de la geometría base de los objetos o personajes en tres	Blender, maya, 3ds max

	dimensiones, enfocándose en proporciones y topología.	
3. Esculpido y detallado	Refinamiento del modelo con detalles de alta resolución, formas orgánicas y superficies complejas.	Zbrush, mudbox
4. Texturizado y materiales	Aplicación de texturas, creación de materiales pbr (albedo, normal, roughness, metallic) y mapas de superficie.	Substance painter, quixel mixer
5. Optimización y retopología	Reducción de polígonos, mapeo uv, baking de detalles y preparación del modelo para tiempo real.	Blender, marmoset toolbag
6. Integración en motor de juego / render en tiempo real	Importación de los activos al motor gráfico, configuración de shaders, iluminación y pruebas de rendimiento.	Unreal engine, unity

Nota. *La tabla resume el conjunto de herramientas de software y hardware que conformaron el entorno de desarrollo del proyecto.*

Motor gráfico - unity engine

Se selecciona unity como el motor de desarrollo principal. Aunque unreal engine ofrece una fidelidad gráfica superior "de caja", unity presenta varias ventajas estratégicas para este proyecto está basado en c#, un lenguaje de programación más accesible y con una curva de aprendizaje más suave que el c++ de unreal, posee un vasto asset store que acelera el desarrollo al ofrecer modelos 3d y herramientas pre-construidas; y cuenta con una comunidad de desarrolladores de rv extremadamente activa y un soporte robusto para una amplia gama de dispositivos, especialmente los de tipo standalone. Como señalan Mäkitie et al., (2018), la elección del motor debe balancear el rendimiento con la productividad del equipo de desarrollo,

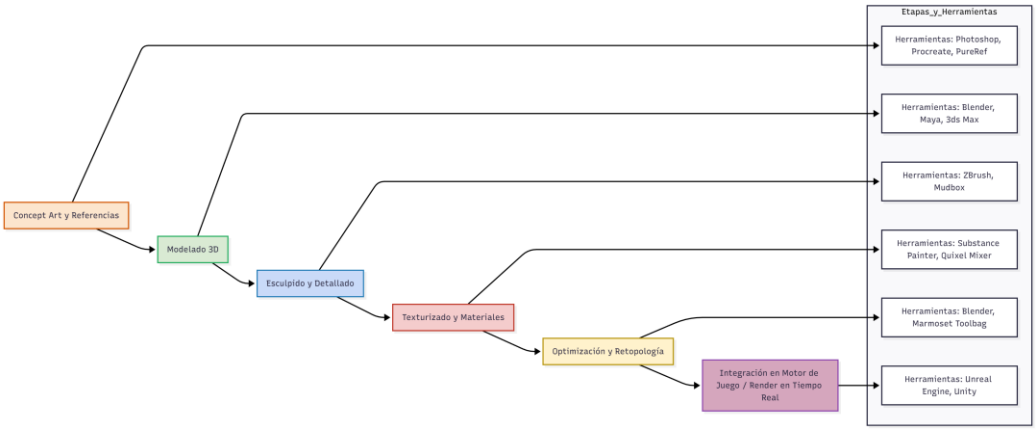
siendo unity una opción óptima para prototipado rápido y desarrollo iterativo en proyectos académicos y de pequeña escala.

Pipeline de creación de activos 2d y 3d

Para alcanzar el nivel de realismo visual requerido, se implementará un pipeline de producción de arte digital estándar en la industria, utilizando un conjunto de herramientas especializadas para cada etapa del proceso.

Figura 6

flujo de proceso del pipeline de producción



Nota. *El diagrama muestra la secuencia de etapas y herramientas empleadas en el pipeline de producción de arte 3d, desde la conceptualización hasta la integración en entornos de renderizado en tiempo real, elaboración propia.*

Modelado base (blender y autodesk maya): se utilizarán ambos programas para el modelado poligonal de los assets del entorno, personajes y objetos interactivos. Crearán las mallas base de baja poligonalidad (low-poly) que son eficientes para el renderizado en tiempo real en el motor de juego.

Esculpido de alto detalle (zbrush): para los objetos y personajes que requieran un alto nivel de detalle orgánico o superficial (ej. Texturas de rocas, detalles de una herida, pliegues de la ropa), se utilizará zbrush. En este programa se esculpirán versiones de muy alta poligonalidad (high-poly) de los modelos.

Texturizado (adobe photoshop): photoshop será la herramienta principal para la creación y edición de las texturas que darán color, materialidad y detalle a los modelos 3d. Se seguirá un flujo de trabajo de renderizado basado en físicas (pbr - physically based rendering), creando mapas de textura específicos como albedo (color), normal, oclusión ambiental, metálico y rugosidad. Este proceso es fundamental para que los objetos reaccionen a la luz de manera realista dentro de unity.

El flujo de trabajo consistirá en crear el modelo base en blender/maya, esculpir los detalles finos en zbrush, y luego "hornear" (bake) estos detalles en un mapa de normales. Este mapa de normales es una textura especial que, aplicada sobre el modelo de baja poligonalidad en unity, simula la apariencia del detalle de alta poligonalidad sin el costo de rendimiento computacional, logrando así un equilibrio óptimo entre calidad visual y performance.

Sistema de evaluación y recolección de datos (telemetría)

El análisis de datos es un componente central de esta tesis. Por ello, el sistema de telemetría se diseñará desde el principio para ser robusto y granular. El software registrará cada evento significativo en un archivo de registro con un formato estructurado, como json.

Un evento típico registrará, como mínimo: un timestamp (marca de tiempo), el tipo de evento (ej. Accion_usuario, evento_simulacion), el detalle de la acción (ej. Inicio_compresiones_rcp) y un conjunto de parámetros relevantes (ej. Profundidad, frecuencia, posicion_manos). Este enfoque, como describen Orji et al., 2020, permite una recolección de datos no intrusiva que puede ser utilizada para el análisis del comportamiento y la personalización de la experiencia de aprendizaje. Los datos cuantitativos objetivos recogidos por el sistema serán la base para la sección de "análisis de datos", permitiendo una evaluación rigurosa de la efectividad del entrenamiento. La implementación de un sistema de telemetría detallado es una de las contribuciones ingenieriles clave de este proyecto, permitiendo ir más allá de la simple creación de una simulación para convertirla en una herramienta de investigación científica, un concepto bien explorado por Schoenau-Fog, (2018) en el contexto de los "juegos serios".

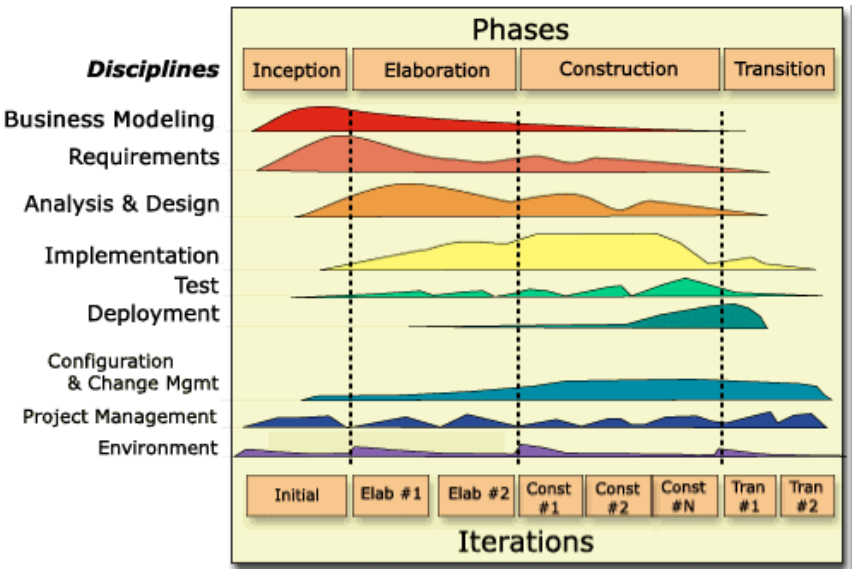
7. DISEÑO METODOLÓGICO

Para gestionar la complejidad del proyecto de manera estructurada, flexible y orientada a la calidad del producto final, como marco metodológico una adaptación del proceso unificado de rational (rup), aplicando sus principios de manera ágil. Esta metodología proporciona una guía que dirige el proyecto a través de fases, asegurando un progreso constante desde el análisis inicial de requisitos hasta la entrega del software.

El ciclo de vida del proyecto se estructuró con respecto a las fases rup, dentro de las cuales se ejecutaron las actividades necesarias para cumplir secuencialmente con los objetivos específicos.

Figura **7**

fases de la metodología ruo ágil aplicada al proyecto



Nota. *Tomada de rational unified process (rup) overview, por IBM, 2020, ibm*

Análisis y diseño del sistema

La fase de elaboración, la primera etapa práctica de la metodología tuvo como propósito fundamental mitigar los riesgos del proyecto, definir una línea base de la arquitectura y establecer un plan de construcción detallado. Durante esta fase se abordaron los dos primeros objetivos específicos, centrados en el análisis de requisitos y el diseño de la solución.

Análisis y especificación de requisitos

Para cumplir con el primer objetivo "identificar y especificar los requisitos funcionales y no funcionales del sistema de software, a partir del análisis de los protocolos estandarizados de primera respuesta para emergencias", se llevó a cabo un proceso de levantamiento de requisitos. Se realizó una investigación documental de los protocolos de actuación para primeros respondientes civiles publicados por organizaciones de referencia tal como la que fundamenta el presente documento, que es el primer respondiente de la alcaldía de bogotá.

Requisitos funcionales: las capacidades concretas que el software debía poseer, desglosadas por cada módulo de entrenamiento.

Requisitos no funcionales: los atributos de calidad del sistema, como la necesidad de una alta calidad visual para la inmersión, y la usabilidad para un público no técnico.

Diseño de la arquitectura y la experiencia de usuario

Una vez definidos los requisitos, se procedió a cumplir el segundo objetivo: "diseñar la arquitectura del software, la estructura de los entornos virtuales y las mecánicas de interacción del usuario, basándose en los requisitos definidos en la fase de análisis". Se diseñó una arquitectura de software modular de tres capas (presentación, lógica y datos) para asegurar la escalabilidad y mantenibilidad del código. Se conceptualizaron los escenarios virtuales y se planificó el pipeline de producción de arte 3d. Se diseñaron las mecánicas de interacción en vr aplicando principios de diseño centrado en el usuario. Se crearon prototipos de baja fidelidad (*greyboxing*) para refinar internamente los flujos de interacción, asegurando que el diseño fuera intuitivo antes de iniciar la construcción.

Desarrollo iterativo del software.

Esta fase fue el núcleo del desarrollo, donde se construyó el sistema de manera incremental para cumplir con el tercer objetivo: "construir e integrar los módulos funcionales del sistema de software, incluyendo los entornos 3d, las interacciones en realidad virtual y el sistema de seguimiento, mediante un proceso de desarrollo iterativo". Para gestionar esta fase, se utilizó el marco de trabajo scrum.

Desarrollo en sprints: la construcción se organizó en sprints de una semana de duración. Cada sprint fue momento que comenzaba con una reunión de planificación (*product backlog*) y terminaba con una revisión. La herramienta jira se utilizó para gestionar el flujo de trabajo.

Construcción e integración de módulos: durante los sprints de esta fase, se llevaron a cabo las siguientes actividades de desarrollo en unity engine con c#:

Modelado y creación de activos: se ejecutó el pipeline de arte 3d (blender, maya, zbrush, photoshop) para crear todos los entornos y objetos interactivos, cumpliendo la primera parte del objetivo específico 1.

Desarrollo de interacciones: se programaron las mecánicas de interacción en vr para permitir al usuario ejecutar los protocolos, cumpliendo con el objetivo específico 2.

Implementación del sistema de seguimiento: se programó la lógica de evaluación y el sistema de telemetría, cumpliendo con el objetivo específico

Fase de transición: consolidación y empaquetado del software

La fase final de la metodología se centró en la finalización y entrega del artefacto de software, cumpliendo así con el cuarto y último objetivo: "consolidar los módulos de software y los activos artísticos en una aplicación de realidad virtual final, empaquetada para su ejecución y operación".

Pruebas de integración:

Se realizaron pruebas internas para asegurar que todos los módulos desarrollados (entornos, interacciones, evaluación) funcionaban correctamente en conjunto como un sistema unificado y coherente. Se corrigieron los errores y conflictos que surgieron durante esta etapa.

Optimización de rendimiento:

Se llevaron a cabo tareas de optimización para asegurar que la aplicación mantuviera una tasa de fotogramas (fps) estable en el hardware de referencia (vr box), un requisito crítico para una experiencia de rv cómoda y libre de cinetosis.

- **Empaquetado final (build):** una vez que el sistema se consideró estable y completo según los requisitos definidos, se realizó el proceso de "build" o compilación final desde unity engine. El resultado de esta fase es el entregable principal de la tesis una aplicación ejecutable (.exe).

8. DESARROLLO DEL PROYECTO

8.1. Desarrollo de la metodología

8.1.1. Requisitos del sistema

Requisitos funcionales (rf):

- El sistema debe permitir la práctica de protocolos de reanimación cardiopulmonar (rcp) con medición de frecuencia y profundidad.
- El sistema debe integrar un módulo para la atención de traumas (golpes, heridas y fracturas).
- El sistema debe registrar el desempeño del usuario mediante un motor de evaluación y telemetría.
- El sistema debe permitir la navegación en el entorno virtual mediante locomoción por teletransporte.

Requisitos no funcionales (rnf):

- Rendimiento en tiempo real: el sistema debe ejecutarse de forma fluida en el entorno de realidad virtual.
- Tiempo de carga: los escenarios deben cargar rápidamente al iniciar cada simulación.
- Usabilidad: la interfaz debe ser clara e intuitiva para el usuario.
- Retroalimentación inmediata: el sistema debe responder en tiempo real a las acciones realizadas.
- Compatibilidad vr: debe funcionar correctamente con el hardware de realidad virtual definido.
- Integridad de datos: la telemetría generada debe almacenarse sin errores.
- Estabilidad: la aplicación no debe presentar fallos durante la simulación.
- Modularidad: debe permitir agregar nuevos escenarios sin afectar el sistema existente.
- Seguridad: debe proteger la información generada en el sistema.
- Mantenibilidad: el código debe facilitar futuras actualizaciones y mejoras.

Reglas de negocio

- Validación de rcp: el sistema registra como exitosa la compresión solo si la profundidad está entre 5 cm y 6 cm.

- Frecuencia de maniobra: se debe mantener un ritmo de 100 a 120 compresiones por minuto; desviaciones de +5 segundos penalizan el puntaje.
- Secuencialidad: es obligatorio completar el "aseguramiento del área" antes de habilitar el uso de insumos médicos.
- Tiempo de reacción (sismo): el usuario dispone de 10 segundos para ubicarse en zona segura tras la activación de la alerta háptica.
- Uso de suministros: el empleo de un insumo incorrecto para el tipo de herida detectada genera un fallo automático en la métrica de precisión.
- Integridad de datos: la visualización de resultados finales está condicionada al guardado exitoso del archivo log en formato json.

Tabla 2

requisitos funcionales.

Id	Requisito funcional	Descripción
Rf01	Entrenamiento de rcp	El sistema debe permitir la práctica de reanimación cardiopulmonar con medición de frecuencia y profundidad en tiempo real.
Rf02	Módulo de trauma	El sistema debe integrar escenarios para la atención de lesiones físicas (golpes, heridas y fracturas).
Rf03	Motor de evaluación	El sistema debe registrar el desempeño del usuario mediante telemetría para su análisis posterior.
Rf04	Navegación inmersiva	El sistema debe permitir el desplazamiento en el entorno virtual mediante el método de teletransporte.

Tabla 3

requisitos no funcionales.

Id	Requisito no funcional	Descripción
Rnf01	Rendimiento en tiempo real	El sistema debe ejecutarse de forma fluida en vr.
Rnf02	Tiempo de carga	Los escenarios deben cargar rápidamente.
Rnf03	Usabilidad	La interfaz debe ser clara e intuitiva.
Rnf04	Retroalimentación inmediata	El sistema debe responder en tiempo real a las acciones.
Rnf05	Compatibilidad vr	Debe funcionar correctamente con el hardware vr definido.
Rnf06	Integridad de datos	La telemetría debe almacenarse sin errores.
Rnf07	Estabilidad	La aplicación no debe presentar fallos durante la simulación.
Rnf08	Modularidad	Debe permitir agregar nuevos escenarios fácilmente.
Rnf09	Seguridad	Debe proteger los datos del sistema.
Rnf10	Mantenibilidad	El código debe facilitar futuras mejoras.

Tabla 4

reglas de negocio.

Id	Regla de negocio	Descripción / criterio de aceptación
Rn01	Validación de rcp	Se registra como exitosa la compresión solo si la profundidad oscila entre 5 cm y 6 cm.
Rn02	Frecuencia de maniobra	Ritmo obligatorio de 100 a 120 compresiones por minuto. Desviaciones mayores a 5 seg. Restan puntaje.

Rn03	Secuencialidad de atención	El uso de insumos médicos se habilita únicamente tras completar el "aseguramiento del área".
Rn04	Tiempo de reacción	Límite de 10 segundos para ubicarse en zona segura tras la activación de la alerta háptica por sismo.
Rn05	Precisión de suministros	El uso de un insumo incompatible con el tipo de herida genera un fallo automático en la evaluación.
Rn06	Integridad de telemetría	La visualización de resultados depende de la escritura exitosa del archivo log en el almacenamiento.
Rn07	Umbral de seguridad	El sistema pausa la simulación automáticamente si el rendimiento cae por debajo de 30 fps.

8.1.2. Cronograma

Tabla 5

sprint

Sprint	Objetivo del sprint	Funcionalidades clave entregadas
1	Establecer el núcleo del sistema y la interacción base.	Creación del proyecto en unity, sistema de locomoción por teletransporte, menú principal interactivo.
2	Desarrollar la mecánica central del módulo de rcp.	Interacción de compresiones, medición en tiempo real de frecuencia y profundidad, retroalimentación visual básica.
3	Completar el módulo de rcp y el sistema de evaluación.	Lógica de evaluación del protocolo completo de rcp, pantalla de resultados, integración del avatar de la víctima.

4	Implementar las mecánicas principales del módulo de trauma.	Sistema de agarre de objetos del botiquín, mecánica de aplicación de presión directa sobre heridas.
5	Completar el módulo de trauma.	Mecánica de inmovilización de fracturas con férulas y vendas, lógica de evaluación del protocolo p.a.s.
6	Desarrollar el entorno y los efectos del módulo de sismo.	Creación del escenario interior, implementación del sistema de físicas para caída de objetos y efectos de cámara.
7	Finalizar el módulo de sismo e implementar la telemetría.	Lógica de evaluación de la postura de protección, implementación del sistema de registro de datos en formato json.
8	Integración final, optimización y corrección de errores.	Pruebas de integración de todos los módulos, optimización de rendimiento y depuración general.

8.1.3. Arquitectura escenario

Se diseñó una arquitectura basada en el patrón modelo-vista-controlador (mvc), adaptada a entornos de realidad virtual. Esta estructura separa la lógica de los protocolos de emergencia (modelo), la representación visual de los escenarios en unity (vista) y el procesamiento de las interacciones del usuario a través de los sensores del casco y mandos (controlador).

Figura ***8***

diseño de escenario del lobby

Nota. *Módulo lobby, elaboración propia*

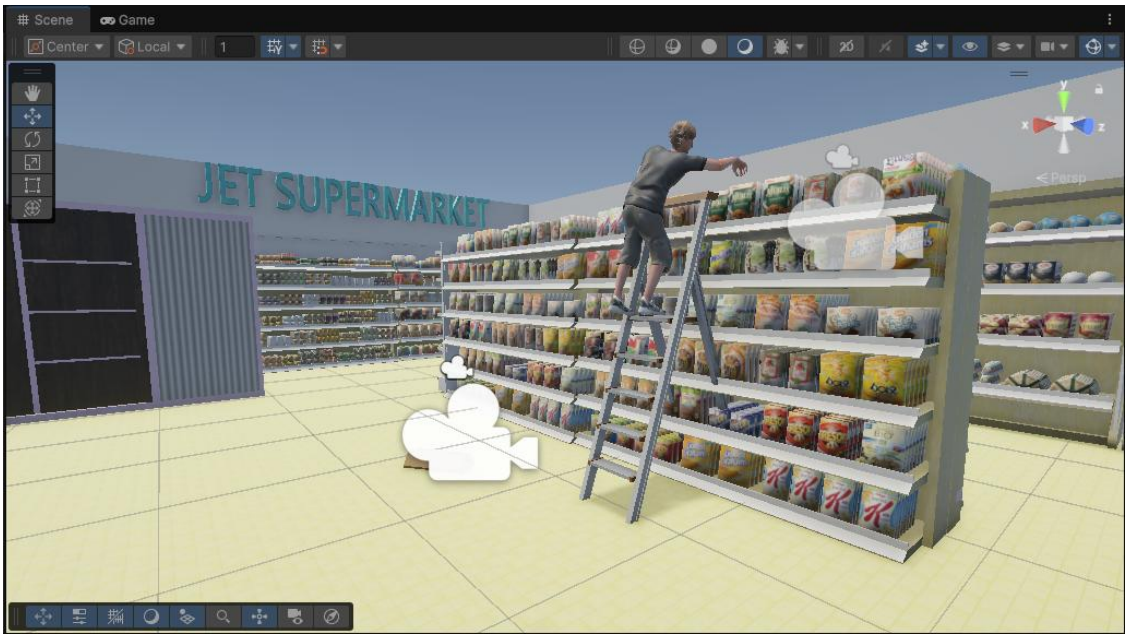
Figura ***9***

diseño del escenario de sismos

Nota. *Módulo de sismos, elaboración propia*

Figura ***10***

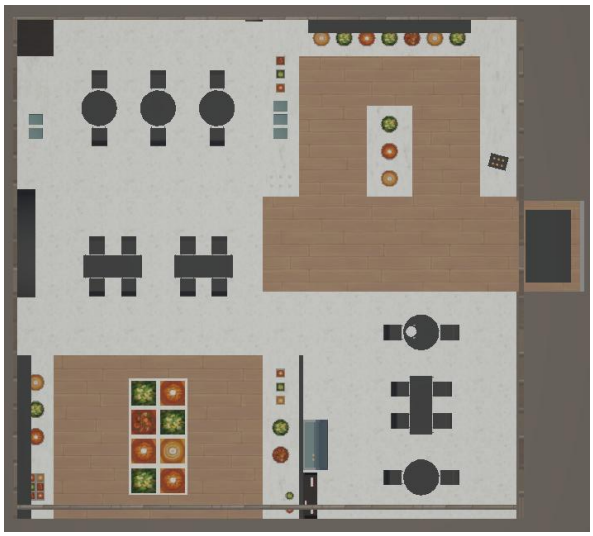
diseño del escenario de heridas y fracturas



Nota. *Módulo herida y fracturas, elaboración propia.*

Figura ***11***

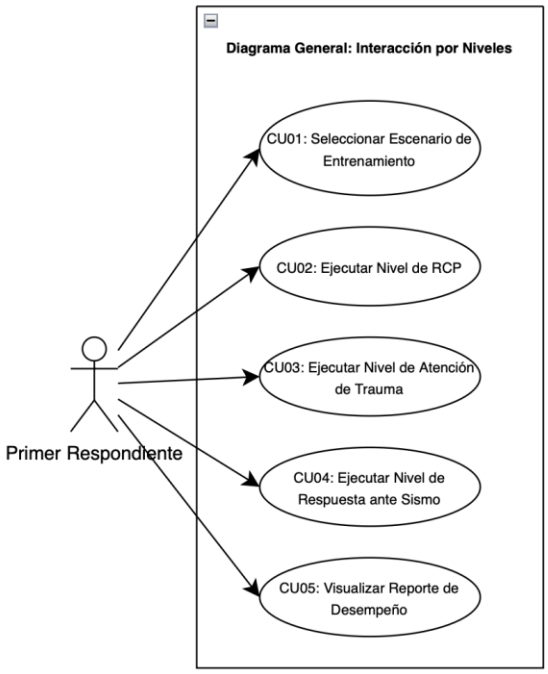
diseño del escenario de rcp



Nota. *Módulo RCP, elaboración propia.*

Figura ***12***

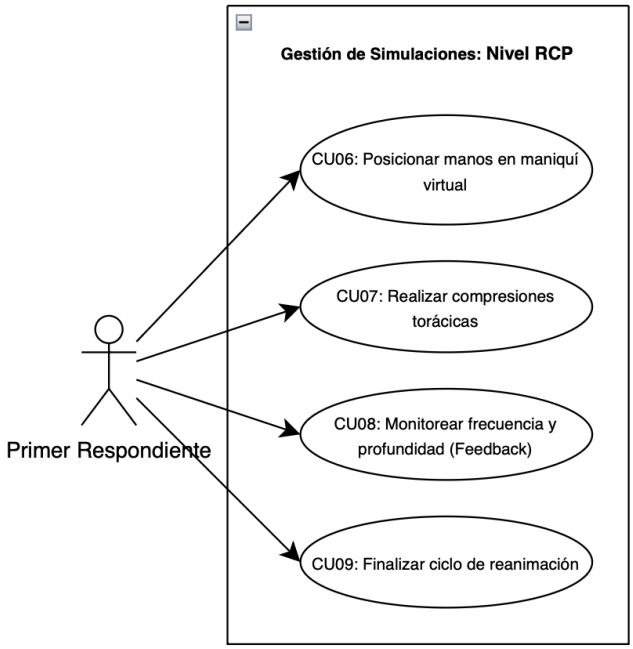
diagrama de casos de uso interacción por niveles.



Nota. *Caso de uso por niveles, elaboración propia.*

Figura ***13***

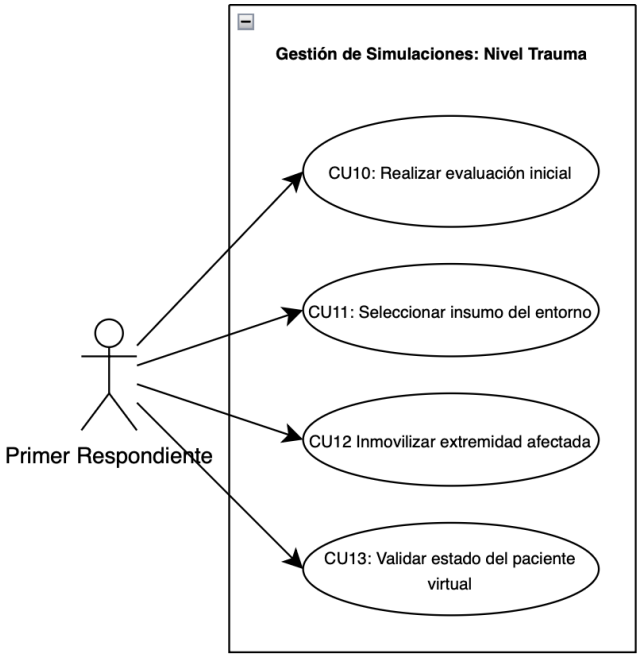
diagrama de casos de uso simulación módulo de rcp.



Nota. *Caso de uso módulo RCP, elaboración propia.*

Figura ***14***

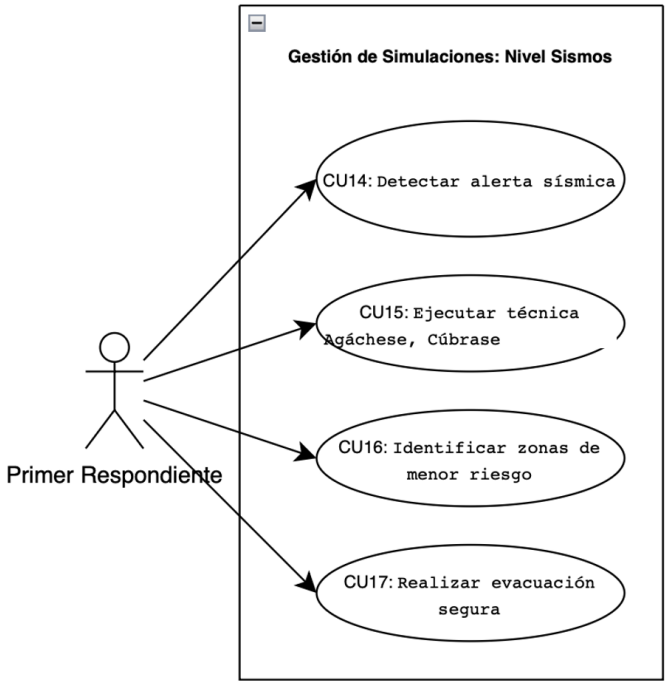
diagrama de casos de uso simulación módulo fracturas y heridas.



Nota. *Caso de uso módulo fracturas y herida, elaboración propia.*

Figura ***15***

diagrama de casos de uso simulación módulo de sismos.



Nota. *Caso de uso módulo de sismos, elaboración propia.*

Especificaciones de casos de uso

Tabla **6**

caso de uso cu01.

Elemento	**Descripción**
Código	Cu01
Nombre	Seleccionar escenario de entrenamiento
Actores	Primer respondiente
Descripción	Navegar por la interfaz del entorno principal (lobby) para elegir e iniciar uno de los módulos de simulación disponibles.
Flujo principal	1. El sistema carga el lobby 2. El usuario utiliza el controlador vr para apuntar al panel de selección.3. El usuario presiona el botón de selección sobre el módulo deseado (rcp, trauma o sismo).4. El sistema valida la selección y carga el entorno 3d correspondiente.

Excepciones	1. Intento de seleccionar opción bloqueada: el sistema omite la acción.2. Falla en la carga del escenario: el sistema retorna al lobby.
Precondiciones	Haber iniciado la aplicación y encontrarse en el lobby principal.
Postcondiciones	Visualización del escenario virtual seleccionado.

Tabla 7

caso de uso cu02.

Elemento	**Descripción**
Código	Cu02
Nombre	Ejecutar nivel de rcp
Actores	Primer respondiente
Descripción	Iniciar y completar la simulación correspondiente a la atención de un paro cardíaco súbito.
Flujo principal	1. El usuario ingresa al nivel.2. El sistema presenta el avatar de la víctima.3. El usuario aplica los pasos de la cadena de supervivencia.4. El sistema evalúa las acciones y finaliza la simulación.
Excepciones	Abandono mediante menú de pausa: el sistema descarta el progreso y retorna al lobby.
Precondiciones	Haber seleccionado el escenario rcp.
Postcondiciones	Habilita la visualización del reporte de desempeño (cu05).

Tabla 8

caso de uso cu03.

Caso de uso	**Cu03 ejecutar nivel de atención de trauma.**
Actores	Primer respondiente.
Descripcion	Iniciar y transitar por toda la simulación correspondiente a la atención de heridas, hemorragias y fracturas siguiendo el protocolo p.a.s. (proteger, avisar, socorrer).

Flujo principal	1. El usuario ingresa al nivel de trauma. 2. El sistema presenta el escenario de un accidente con un avatar lesionado y un botiquín virtual. 3. El usuario interactúa evaluando las heridas, seleccionando insumos y aplicando los primeros auxilios (presión directa, inmovilización). 4. El sistema evalúa internamente las acciones y finaliza la simulación al estabilizar al paciente virtual.
Excepciones	1. El usuario abandona la simulación mediante el menú de pausa, el sistema detiene la ejecución, descarta el progreso no finalizado y vuelve al lobby.
Precondiciones	Haber iniciado la aplicación y seleccionado el escenario de trauma desde el lobby interactivo.
Postcondiciones	Generación de datos de telemetría y habilitación de la visualización del reporte de desempeño (cu05).

Tabla 9

caso de uso cu04.

Caso de uso	**Cu04 ejecutar nivel de respuesta ante sismo.**
Actores	Primer respondiente.
Descripcion	Iniciar y transitar por toda la simulación correspondiente a la supervivencia, autoprotección y evacuación durante y después de un evento telúrico.
Flujo principal	1. El usuario ingresa al nivel de sismo. 2. El sistema inicia la simulación en un entorno cerrado que experimenta un sismo mediante físicas de caída de objetos, vibración de cámara y sonidos ambientales. 3. El usuario aplica técnicas de autoprotección ("agáchese, cúbrase ") y se desplaza hacia zonas seguras. 4. Una vez cesa el movimiento sísmico, el usuario identifica la ruta de evacuación y abandona el lugar. 5. El sistema evalúa el tiempo de reacción y las decisiones tomadas, finalizando el nivel.

Excepciones	1. El usuario abandona la simulación mediante el menú de pausa, el sistema detiene la ejecución, descarta el progreso no finalizado y vuelve al lobby.
Precondiciones	Haber iniciado la aplicación y seleccionado el escenario de sismos desde el lobby interactivo.
Postcondiciones	Generación de datos de telemetría y habilitación de la visualización del reporte de desempeño (cu05).

Tabla 10

caso de uso cu05.

Elemento	**Descripción**
Código	Cu05
Nombre	Visualizar reporte de desempeño
Actores	Primer respondiente
Descripción	Mostrar un informe detallado basado en la telemetría recolectada durante la simulación.
Flujo principal	1. El sistema detecta la finalización del escenario.2. Procesa los datos json guardados.3. Muestra panel con puntuación, aciertos y errores.4. El usuario retorna al lobby.
Excepciones	Error en lectura de telemetría: mensaje de error y retorno al menú principal.
Precondiciones	Haber finalizado un nivel de entrenamiento.
Postcondiciones	Retorno al lobby principal.

Tabla 11

caso de uso cu06.

Elemento	**Descripción**
Código	Cu06
Nombre	Posicionar manos en maniquí virtual
Actores	Primer respondiente
Descripción	Ubicar los controladores vr en la posición correcta sobre el esternón.

Flujo principal	1. Acercamiento mediante locomoción.2. Ubicación de manos virtuales.3. Detección mediante colliders.4. Validación de posición.
Excepciones	Posición incorrecta: no se habilita la mecánica y se registra error.
Precondiciones	Estar dentro del nivel rcp.
Postcondiciones	Sistema listo para registrar compresiones.

Tabla 12

caso de uso cu07.

Elemento	**Descripción**
Código	Cu07
Nombre	Realizar compresiones torácicas
Actores	Primer respondiente
Descripción	Ejecutar movimiento vertical rítmico para simular rcp.
Flujo principal	1. Movimiento vertical.2. Medición delta y y deltatime.3. Envío de datos al modelo de protocolo.
Excepciones	Inactividad: pausa y registro de evento.
Precondiciones	Cu06 completado.
Postcondiciones	Datos enviados al motor de evaluación.

Tabla 13

caso de uso cu08.

Elemento	**Descripción**
Código	Cu08
Nombre	Monitorear frecuencia y profundidad
Actores	Primer respondiente
Descripción	Brindar retroalimentación sensorial inmediata.
Flujo principal	1. Comparación con umbrales ideales.2. Clasificación de ejecución.3. Retroalimentación visual y sonora.
Excepciones	Compresión errónea: indicador visual negativo.
Precondiciones	Cu07 en ejecución.

Postcondiciones	Actualización de contadores de desempeño.

Tabla 14

caso de uso cu09.

Elemento	**Descripción**
Código	Cu09
Nombre	Finalizar ciclo de reanimación
Actores	Primer respondiente
Descripción	Culminar protocolo rcp tras objetivo alcanzado.
Flujo principal	1. Detección de ciclo cumplido.2. Detención de captura.3. Guardado de telemetría.4. Pantalla final.
Excepciones	N/a
Precondiciones	Escenario en curso.
Postcondiciones	Generación de datos y transición a reporte.

Tabla 15

caso de uso cu10.

Caso de uso	**Cu10 realizar evaluación inicial.**
Actores	Primer respondiente.
Descripcion	Inspeccionar visualmente al paciente para identificar lesiones críticas siguiendo el protocolo p.a.s.
Flujo principal	1. El usuario se aproxima al avatar accidentado. 2. Observa las lesiones generadas visualmente. 3. El sistema detecta mediante raycasting que el usuario enfoca la herida correcta.
Excepciones	1. El usuario intenta aplicar un tratamiento sin evaluar previamente la herida; el sistema registra error de secuencia.
Precondiciones	Estar en el escenario del nivel de trauma.
Postcondiciones	Lesión identificada, permitiendo continuar con el tratamiento.

Tabla 16

caso de uso cu11.

Caso de uso	**Cu11 seleccionar insumo del entorno.**
Actores	Primer respondiente.
Descripcion	Agarrar y manipular objetos médicos virtuales del botiquín.
Flujo principal	1. El usuario acerca su mano virtual a un objeto interactivo. 2. Presiona el botón de agarre. 3. El sistema valida la física y vincula el objeto a la mano del usuario.
Excepciones	1. Si el usuario suelta el botón antes de posicionar el objeto, este cae por efecto de las físicas del motor.
Precondiciones	Botiquín virtual abierto y visible.
Postcondiciones	Objeto sostenido listo para aplicarse.

Tabla **17**

caso de uso cu12.

Caso de uso	**Cu12 inmovilizar extremidad afectada.**
Actores	Primer respondiente.
Descripcion	Posicionar férulas y vendas sobre la fractura para restringir el movimiento.
Flujo principal	1. El usuario traslada la férula hacia la extremidad lesionada. 2. El sistema detecta la colisión con la zona afectada. 3. El usuario suelta el objeto. 4. El sistema acopla visualmente la férula y valida la acción.
Excepciones	1. Si se posiciona en una extremidad sana, el sistema registra error y no acopla el objeto.
Precondiciones	Haber seleccionado el insumo correcto (cu11).
Postcondiciones	Extremidad inmovilizada correctamente.

Tabla **18**

caso de uso cu13.

Caso de uso	**Cu13 validar estado del paciente virtual.**
Actores	Primer respondiente.

Descripcion	Comprobar que la hemorragia se detuvo o la extremidad quedó segura tras la intervención.
Flujo principal	1. El motor de evaluación verifica las mecánicas aplicadas. 2. La vista actualiza el estado visual. 3. El sistema determina el éxito del protocolo.
Excepciones	1. Si la intervención fue insuficiente, el sangrado se reanuda y se registra la falla.
Precondiciones	Haber aplicado un tratamiento.
Postcondiciones	Finalización del escenario de trauma.

Tabla 19

caso de uso cu14.

Caso de uso	**Cu14 detectar alerta sísmica.**
Actores	Primer respondiente.
Descripcion	Reconocer las señales ambientales que simulan el inicio del temblor.
Flujo principal	1. El gestor de escenarios activa el evento sísmico. 2. Se reproduce audio espacializado. 3. Se activan efectos de vibración y físicas de caída de objetos.
Excepciones	1. No aplica. Evento controlado por el sistema.
Precondiciones	Iniciar el nivel de sismos.
Postcondiciones	Inicio del conteo de tiempo de reacción.

Tabla 20

caso de uso cu15.

Caso de uso	**Cu15 ejecutar técnica agáchese, cúbrase y sujétese.**
Actores	Primer respondiente.
Descripcion	Adoptar la postura de protección reduciendo la altura física.
Flujo principal	1. El usuario desciende físicamente su cuerpo. 2. El visor vr registra la disminución del eje y. 3. El sistema interpreta la postura como acción correcta. 4. Se evalúa el tiempo de reacción.

Excepciones	1. Permanecer de pie genera penalización por impacto.
Precondiciones	Evento sísmico en curso (cu14).
Postcondiciones	Acción registrada en la telemetría.

Tabla 21

caso de uso cu16.

Caso de uso	**Cu16 identificar zonas de menor riesgo.**
Actores	Primer respondiente.
Descripcion	Desplazarse hacia áreas seguras evitando zonas de peligro.
Flujo principal	1. El usuario analiza el entorno. 2. Se mueve hacia estructura resistente. 3. Ingresa a área segura validada por triggers. 4. El sistema registra la acción correcta.
Excepciones	1. Ubicarse en zona peligrosa genera penalización.
Precondiciones	Evento sísmico activo.
Postcondiciones	Usuario protegido de colisiones virtuales.

Tabla 22

caso de uso cu17.

Caso de uso	**Cu17 realizar evacuación segura.**
Actores	Primer respondiente.
Descripcion	Abandonar el entorno de forma segura tras finalizar el movimiento telúrico.
Flujo principal	1. El sistema indica fin del sismo. 2. El usuario identifica ruta de evacuación. 3. Se desplaza evitando escombros. 4. Llega al punto seguro.
Excepciones	1. Pisar zonas peligrosas genera registro de error.
Precondiciones	Evento sísmico finalizado.
Postcondiciones	Finalización del módulo y generación de resultados.

8.1.4. Diagrama de actividades

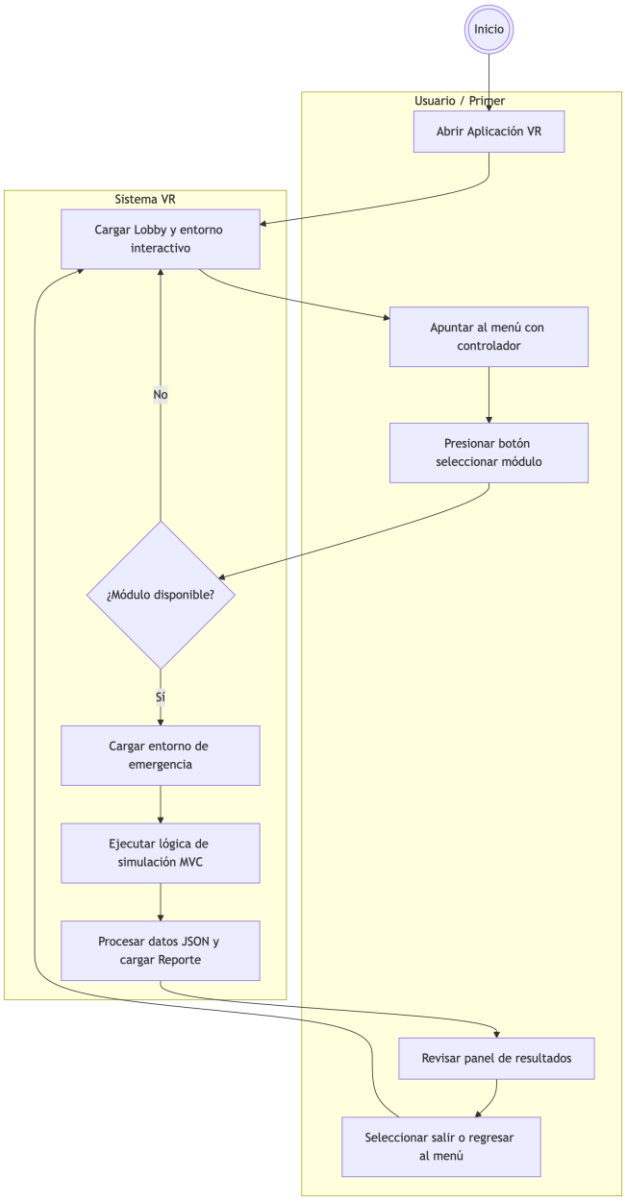
Se modelaron los flujos lógicos que el usuario debe seguir para completar con éxito cada módulo de entrenamiento. Estos diagramas representan la toma de decisiones en tiempo real bajo situaciones de estrés simulado.

Actividad de evaluación: el sistema monitorea cada acción del usuario; si la profundidad de la compresión es insuficiente, se genera una retroalimentación visual inmediata y se registra el error en el archivo de telemetría.

Flujo principal y lobby

Figura **16**

ingreso del lobby da01.



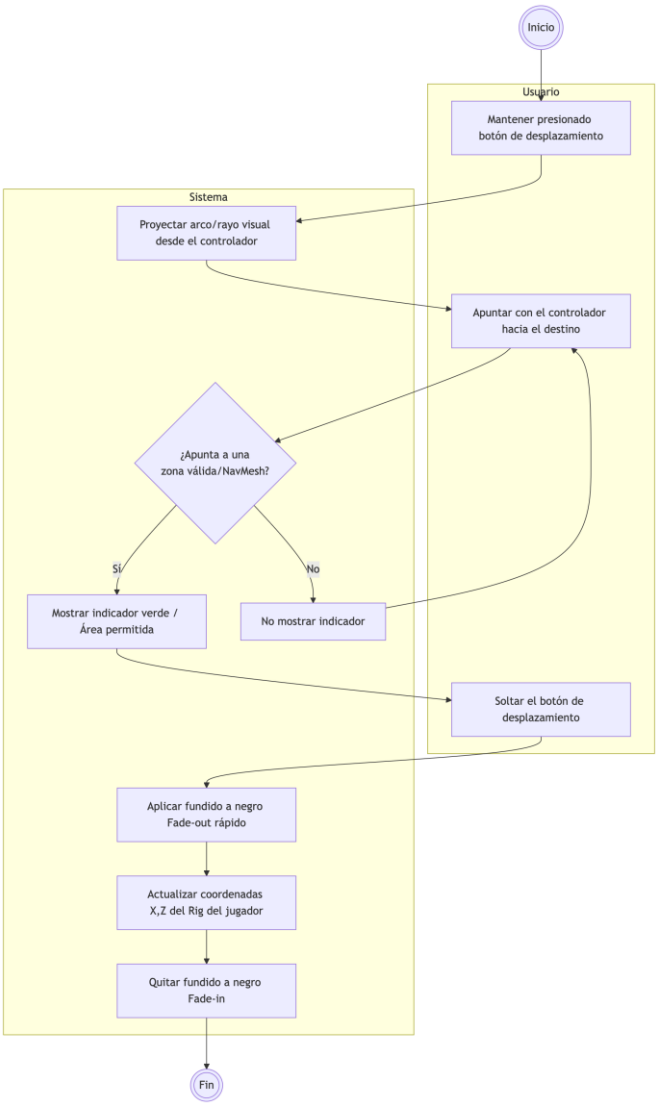
Nota. *Ingreso del lobby, elaboración propia.*

Este diagrama muestra cómo el usuario ingresa a la aplicación, elige un módulo, interactúa y luego visualiza su informe.

Locomoción

Figura **17**

flujo del movimiento da02.



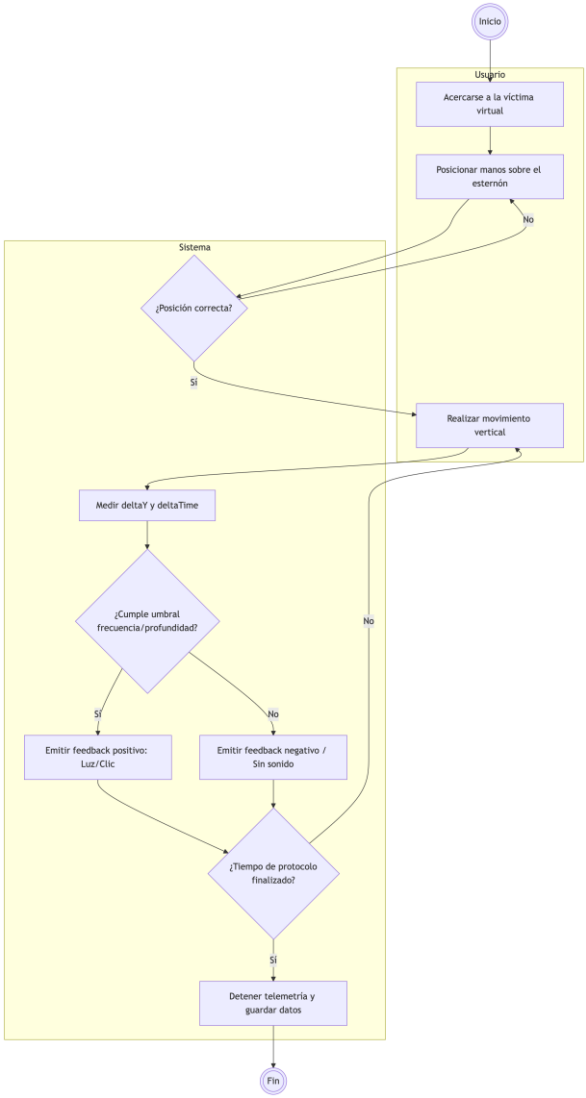
Nota. *Flujo de movimiento, elaboración propia.*

Este diagrama explica cómo interactúa el usuario para moverse por el entorno virtual de forma segura (evitando mareos/cinetosis).

Módulo de RCP

Figura **18**

módulo de rcp da03



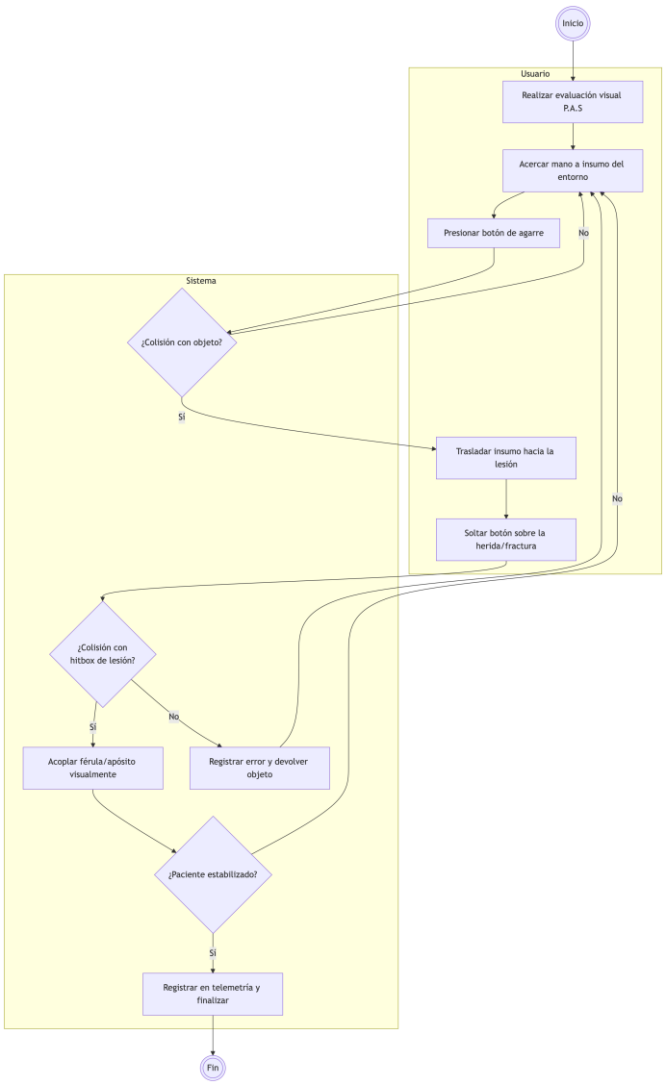
Nota. *Ingreso módulo RCP, elaboración propia.*

Este diagrama representa el flujo interactivo de la reanimación cardiopulmonar, donde la telemetría valida en tiempo real.

Módulo de heridas y fracturas

Figura ***19***

modulo heridas y fracturas da04

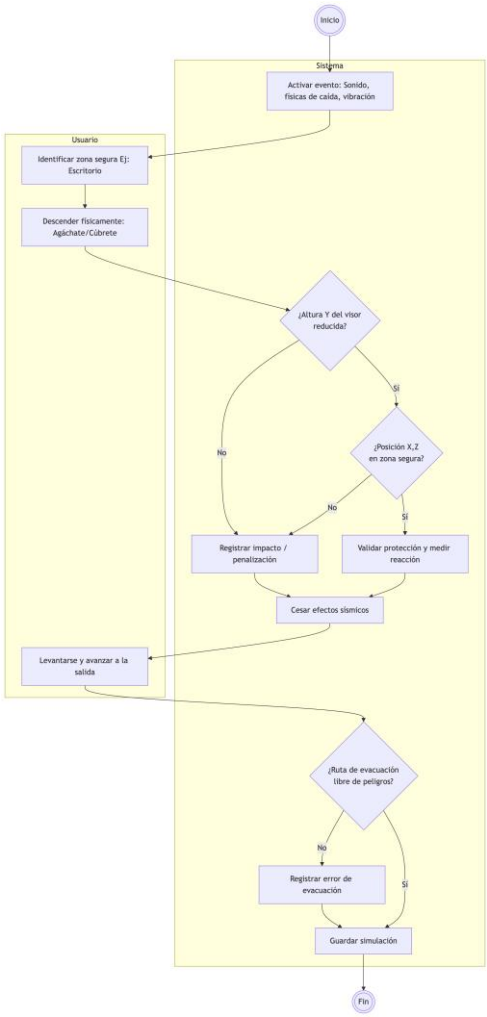


Nota: Este diagrama representa el protocolo p.a.s. para atender hemorragias o inmovilizar fracturas manipulando objetos, elaboración propia.

Módulo de Sismos

Figura 20

módulo de sismos da05.

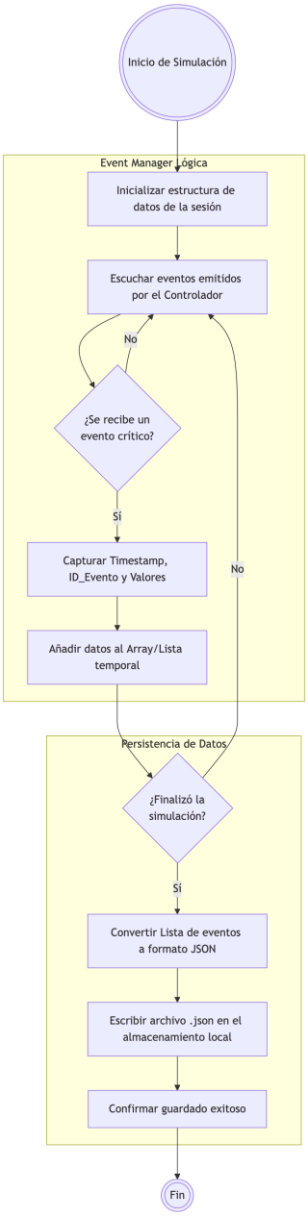


Nota: Este diagrama mapea la respuesta del usuario ante una alerta del entorno usando físicas, donde la posición de la cámara (visor) valida la postura de protección, elaboración propia.

Sistema de telemetría

Figura **21**

telemetría sistema da06.



Nota:Este diagrama no es de interacción directa, sino del flujo interno del software, elaboración propia.

8.1.5. Diagrama de formularios (interfaz de usuario)

Figura **22**

ingreso al lobby (inicio)

Fuente: elaboración propia

Figura ***23***

ingreso al módulo de rcp

Fuente: elaboración propia

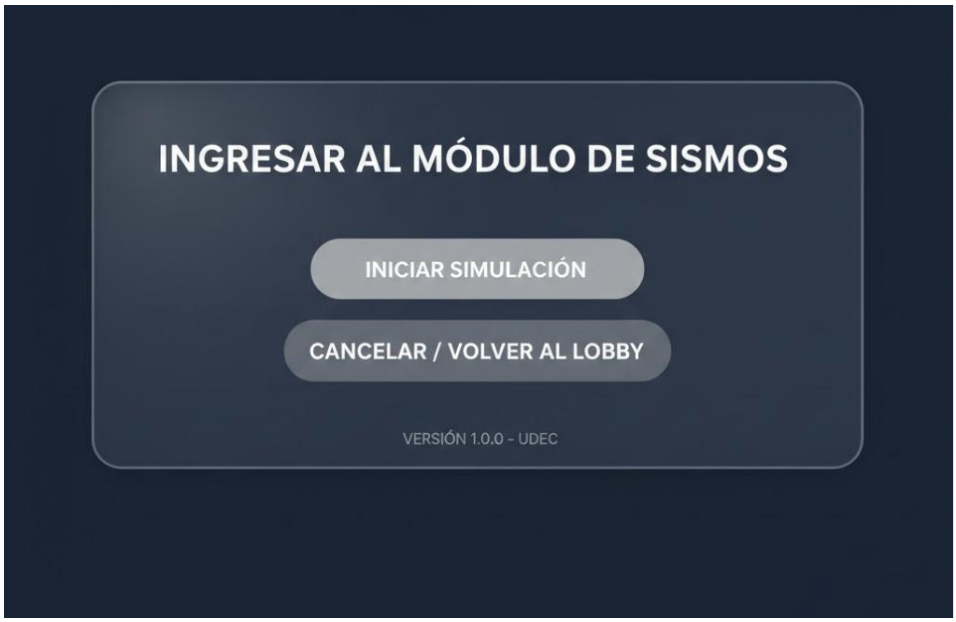
Figura **24**

ingreso al módulo de heridas y fracturas

Fuente: elaboración propia

Figura **25**

ingreso al módulo de sismos

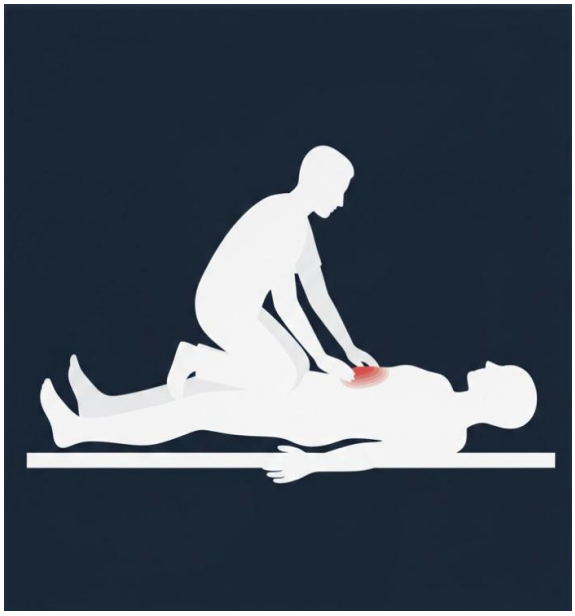


Fuente: elaboración propia

8.1.6. Wireframe

Figura ***26***

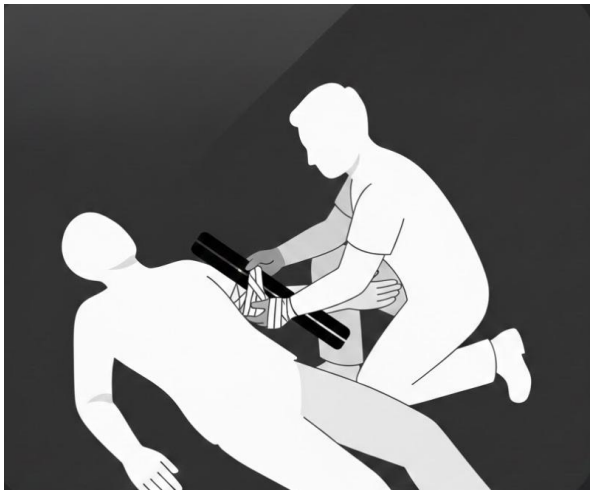
maniobra de rcp.



Nota. *Maniobra de RCP, elaborada mediante IA.*

Figura **27**

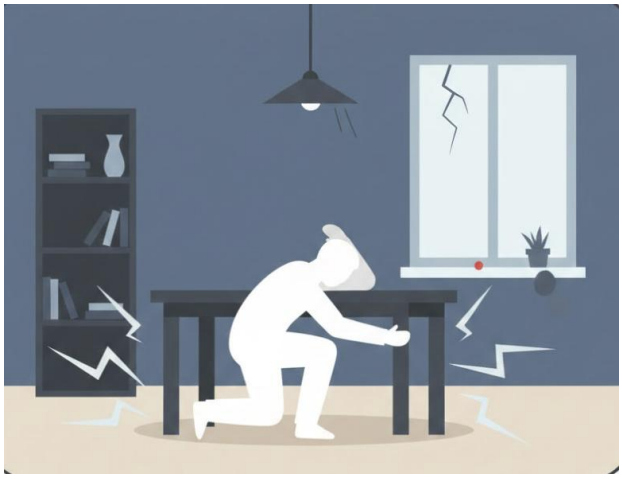
inmovilización de extremidad fracturada



Nota. *Inmovilización de extremidad, elaborada mediante IA*

Figura **28**

representación de sismos.



Nota. *Representación de sismos, elaborada mediante IA.*

8.1.7. Construcción del entorno virtual

Adquisición y selección de recursos tridimensionales (assets)

La fase inicial para la construcción de los entornos virtuales del simulador se realizó enfocada en la gestión de recursos gráficos. Para garantizar el equilibrio entre una alta fidelidad visual y el rendimiento técnico exigido por las plataformas de realidad virtual (manteniendo una tasa de refresco estable para evitar la cinetosis), se determinó implementar un flujo de trabajo híbrido para la obtención de los modelos tridimensionales (3d). Este enfoque combinó la creación manual de modelos específicos y la adquisición de recursos preexistentes a través de repositorios especializados.

Para la obtención de gran parte del mobiliario, utilería médica y elementos decorativos que componen las distintas escenas (como el escenario de fracturas, sismos y reanimación), se recurrió a plataformas de distribución de activos digitales de alto estándar en la industria. Principalmente, se hizo uso de sketchfab (https://sketchfab.com), un repositorio amplio en visualización y gestión de modelos 3d y realidad aumentada (ar).

Figura ***29***

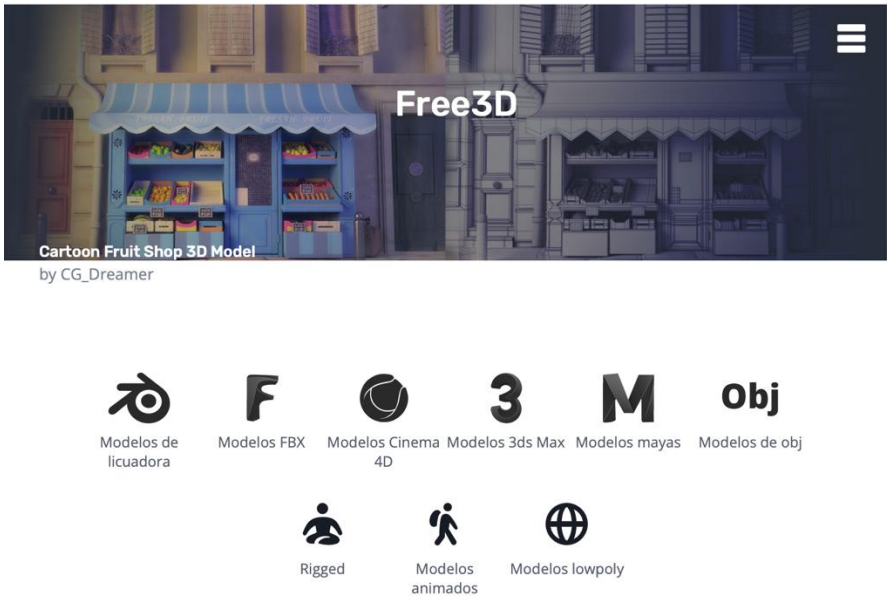
sketchfab.

Nota: Para complementar este catálogo de recursos tridimensionales, el equipo empleó de manera sinérgica la plataforma free3d, tomada de la página principal de Sjetckfab.

La inclusión de este segundo repositorio obedeció a la necesidad técnica de encontrar mobiliario específico y activos estructurales en formatos altamente compatibles con el motor gráfico (como .fbx y .obj). Adicionalmente, esta plataforma facilitó la obtención de mallas base que posteriormente fueron importadas, reescaladas y refinadas manualmente en autodesk maya. Durante la adquisición de estos elementos, se mantuvo una política estricta de selección mediante filtros de búsqueda enfocados en modelos de baja poligonalidad (low-poly).

Figura **30**

página de free3d.

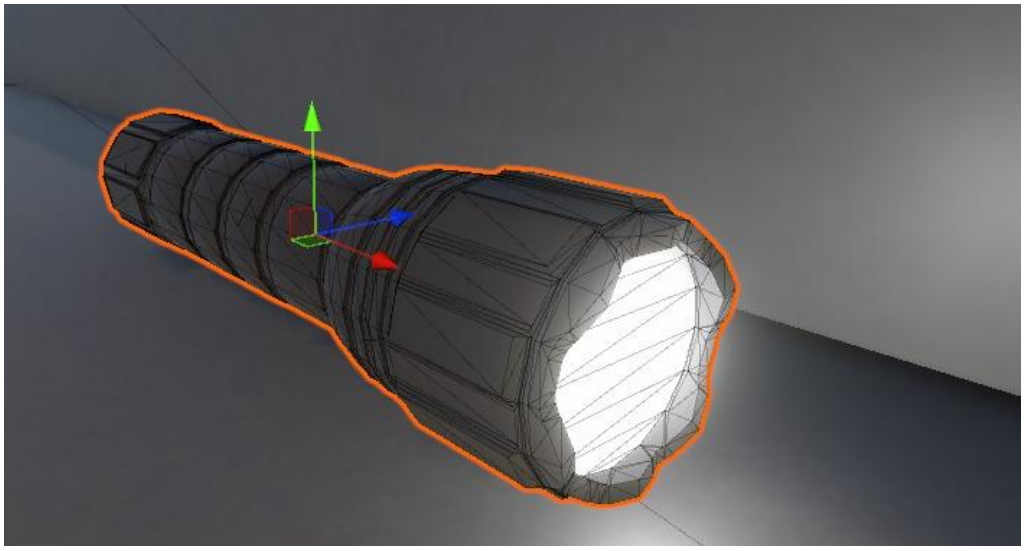


Nota. *Tomada de la página principal de Free3D.*

Entre los objetos obtenidos desde repositorios especializados de assets digitales, tenemos, una linterna, un botiquín de primeros auxilios y botellas de agua. Estos elementos fueron seleccionados en función de su pertinencia dentro de la escena y su potencial de interacción con el usuario. Posteriormente, se integraron y configuraron dentro del motor gráfico, ajustando propiedades como escala, materiales y colisiones para garantizar coherencia visual y funcional. Además, se añadieron otros componentes complementarios que enriquecen la ambientación general del escenario y fortalecen la experiencia inmersiva, permitiendo una interacción más dinámica dentro del entorno virtual.

Figura **31**

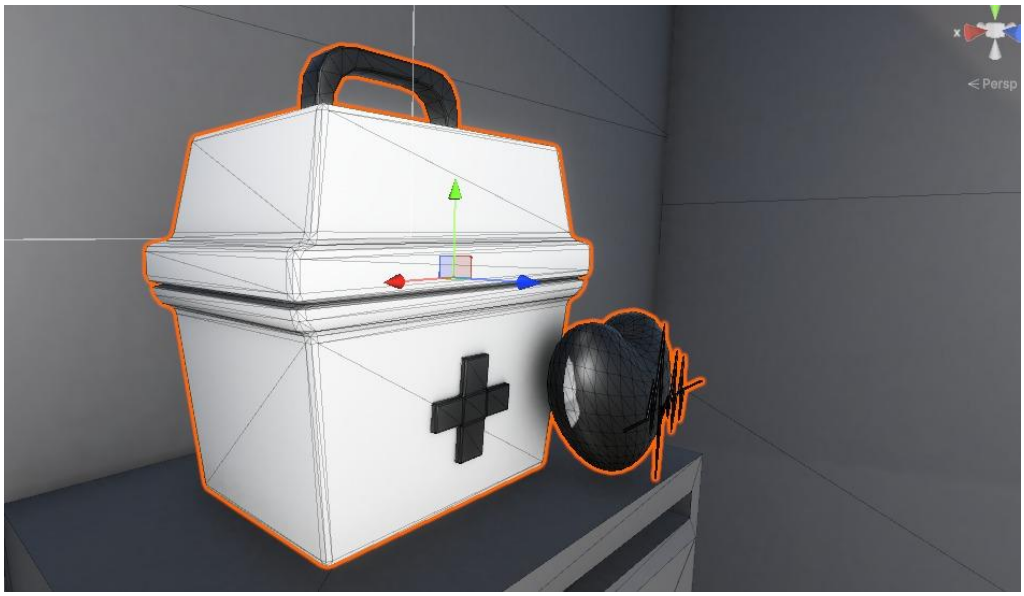
modelo 3d linterna



Nota. *Modelo 3D linterna, elaboración propia.*

Figura **32**

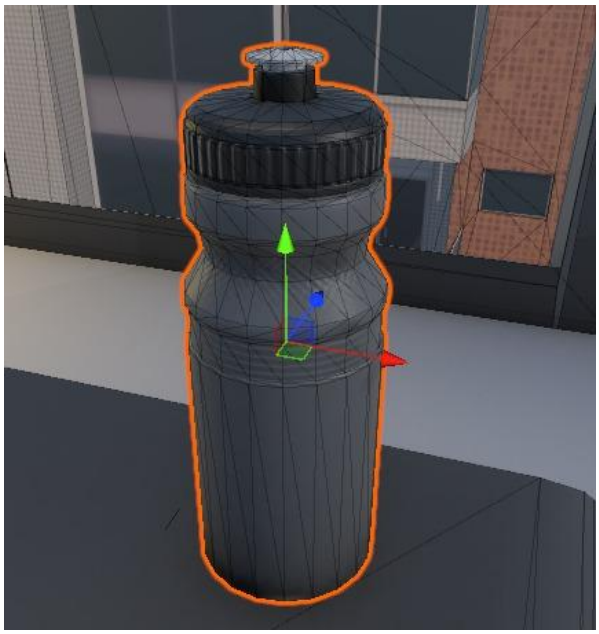
modelo gratuito de un botiquín primeros auxilios



Nota. *Modelo 3D botiquín, tomado gratuitamente de sketchfab.*

Figura **33**

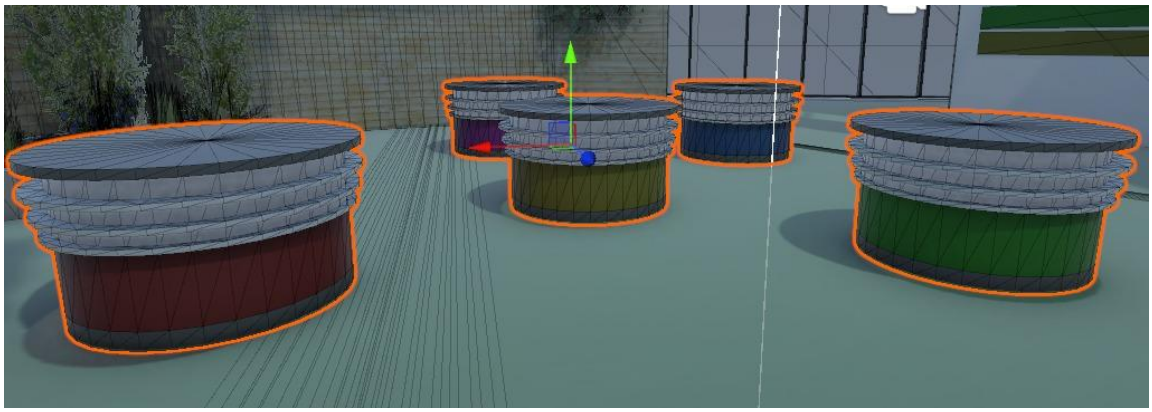
modelo 3d botella con agua



Nota. *Modelo 3D botella, elaboración propia.*

Figura **34**

modelo gratuito de objetos del escenario

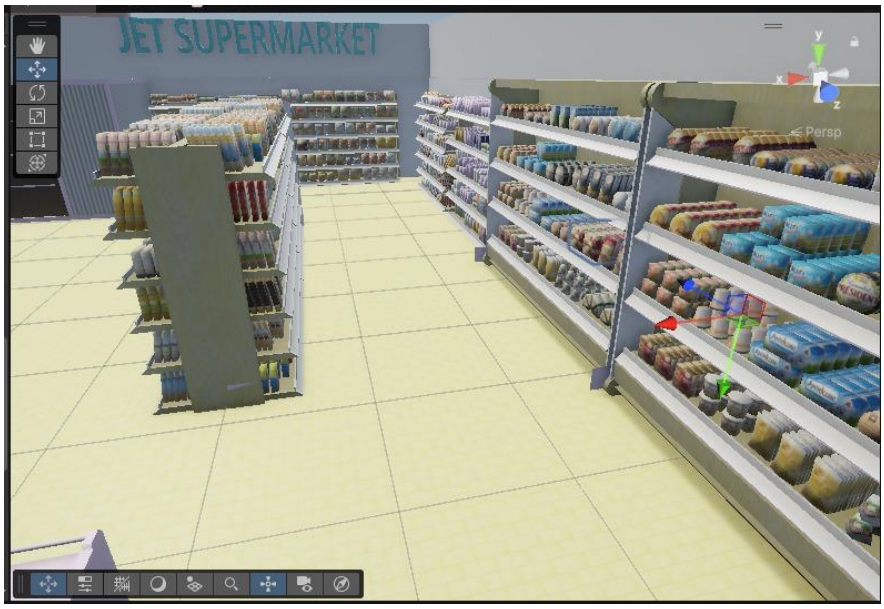


Nota. *Modelo 3D props del escenario, tomado gratuitamente de Free3D.*

Posteriormente, se realizó la adecuación de algunos elementos del entorno con el fin de optimizar el rendimiento de la aplicación. En este proceso, un escenario previamente importado fue modificado mediante la eliminación de objetos innecesarios y componentes decorativos que incrementaban la carga poligonal y el consumo de recursos. Esta depuración permitió mejorar la fluidez del entorno virtual, reduciendo tiempos de renderizado y favoreciendo una experiencia más estable para el usuario.

Figura **35**

modelo de supermercado modificado

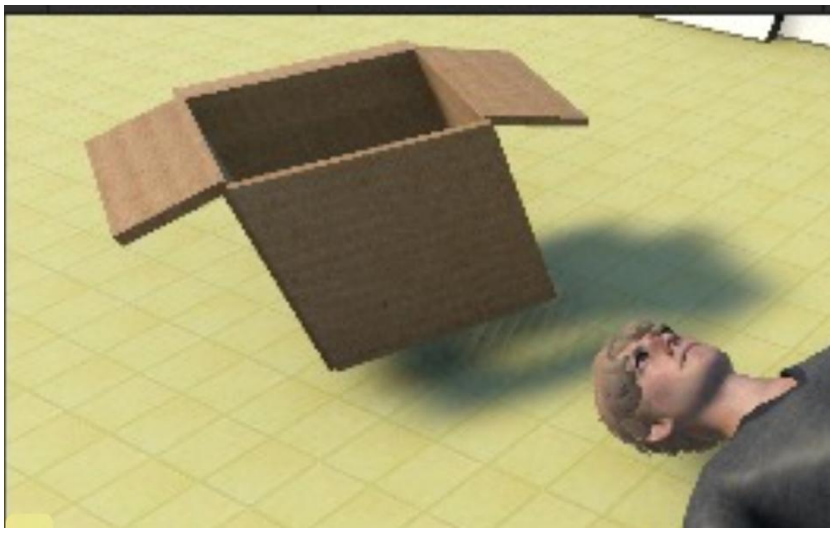


Nota. *Modelo 3D supermercado, Tomado de sketchfab y editado.*

Se diseñó y configuró un objeto interactivo correspondiente a una caja de cartón. Este elemento fue adaptado para permitir la manipulación directa dentro del entorno, incorporando una acción específica que posibilita retirar un fragmento del cartón como parte de la dinámica de interacción. Para ello, se ajustaron propiedades físicas, colisiones y eventos de activación, garantizando una respuesta coherente y realista dentro de la simulación.

Figura **36**

caja de cartón elaborada en blender



Nota. *Modelo 3D caja, elaboración propia.*

Posteriormente, se incorporó al entorno virtual un modelo tridimensional correspondiente a una persona, el cual representa al individuo lesionado dentro de la simulación. Este modelo fue obtenido desde una plataforma especializada de recursos digitales y posteriormente integrado al motor gráfico, donde se realizaron ajustes de escala, posición y orientación para garantizar su correcta adaptación al escenario. Asimismo, se configuraron materiales y componentes necesarios para asegurar coherencia visual con el entorno, permitiendo que el personaje cumpla adecuadamente su función dentro de la narrativa y dinámica de la escena.

Figura **37**

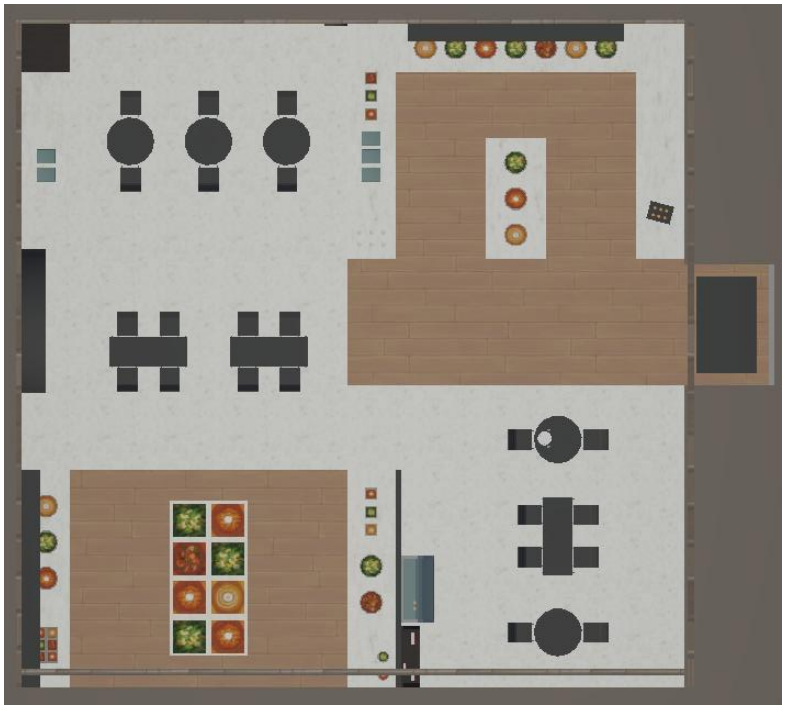
modelo gratuito de persona herida

Nota. *Modelo 3D persona herida, tomado gratuitamente de Free3D.*

Finalmente, se procedió a la integración y organización de los escenarios previamente configurados, consolidando todos los elementos dentro de una estructura funcional y coherente. En esta fase se realizó el ensamblaje definitivo del entorno, ubicando los objetos descargados y modelados en sus respectivas posiciones, verificando jerarquías en la escena y ajustando parámetros de iluminación, sombras y materiales para lograr uniformidad visual.

Figura **38**

modelo escenario del módulo de rcp



Nota. *Modelo escenario RCP, elaboración propia.*

Figura **39**

modelo escenario del módulo de sismos

Nota. *Modelo escenario sismos, elaboración propia.*

Figura **40**

modelo escenario del módulo de fracturas y heridas

Nota. *Modelo escenario fracturas y heridas, elaboración propia.*

Posteriormente, se llevó a cabo un proceso de mejora y refinamiento de las interfaces de usuario. En esta etapa se ajustaron aspectos relacionados con la disposición de los elementos gráficos, tipografías, colores y tamaños, buscando mayor claridad visual, coherencia estética y facilidad de uso. Asimismo, se revisó la funcionalidad de botones, indicadores y mensajes emergentes, garantizando una navegación intuitiva y una correcta respuesta ante las acciones del usuario.

Una vez realizadas las mejoras correspondientes, se consolidaron las versiones finales de las interfaces, verificando su integración adecuada con los distintos escenarios y módulos del sistema.

Figura **41**

imágenes de diseños finales de interfaces

Nota. *Interfaces de usuario, elaboración propia.*

En la fase de programación se inició con la implementación del módulo de movimiento del personaje, el cual tiene como función principal gestionar únicamente los desplazamientos dentro del entorno virtual. Este componente fue configurado para operar sobre el display 2, asegurando que la visualización correspondiente se proyectara correctamente en el dispositivo asignado durante las pruebas.

Programación de mecánicas de interacción

Figura 42

código de movimiento sobre el personaje

```
using UnityEngine;
using UnityEngine.InputSystem;

[RequireComponent(typeof(CharacterController))]
public class MotionObjectController : MonoBehaviour
{
    [Header("Movimiento")]
    public float moveSpeed = 5f;
    public float gravity = -9.81f;

    private CharacterController controller;
    private Vector3 velocity;

    [Header("Referencia Cámara")]
    public Transform cameraTransform;

    void Start()
    {
        controller = GetComponent<CharacterController>();

        if (cameraTransform == null)
        {
            cameraTransform = Camera.main.transform;
        }
    }

    void Update()
    {
        MovePlayer();
        ApplyGravity();
    }

    void MovePlayer()
    {
        Gamepad gamepad = Gamepad.current;

        if (gamepad == null)
            return;

        Vector2 input = gamepad.leftStick.ReadValue();

        // Dirección de la cámara (horizontal)
        Vector3 forward = cameraTransform.forward;
        Vector3 right = cameraTransform.right;

        forward.y = 0f;
        right.y = 0f;

        forward.Normalize();
        right.Normalize();

        // Movimiento relativo a la cámara
        Vector3 move = forward * input.y + right * input.x;

        controller.Move(move * moveSpeed * Time.deltaTime);
    }

    void ApplyGravity()
    {
        if (controller.isGrounded && velocity.y < 0)
        {
            velocity.y = -2f;
        }

        velocity.y += gravity * Time.deltaTime;

        controller.Move(velocity * Time.deltaTime);
    }
}
```

Nota. *Código de movimiento, elaboración propia.*

Se desarrolló un script orientado a la activación dinámica de interfaces gráficas (canvas) según la ubicación del personaje dentro del entorno virtual. El funcionamiento se basa en el uso de zonas delimitadas con colliders configurados como “trigger”, las cuales detectan la presencia del personaje mediante la verificación de su etiqueta (tag).

Cada zona cuenta con un collider en modo is trigger, permitiendo que, al ingresar el personaje —identificado con el tag correspondiente (por ejemplo, “pj”)—, se ejecute la activación del canvas asociado. De igual manera, al salir de la zona, el sistema puede desactivar la interfaz para evitar saturación visual o interferencias con otras áreas del entorno.

Figura **43**

código para la activación de canvas según la zona

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.InputSystem;

public class ControladorLetrero : MonoBehaviour
{
    public GameObject panelDelLetrero;
    public int indiceDeLaEscenaACargar;

    private static ControladorLetrero triggerActual;
    private bool jugadorDentro = false;

    void Start()
    {
        if (panelDelLetrero != null)
            panelDelLetrero.SetActive(false);
    }

    void Update()
    {
        if (!jugadorDentro) return;

        if (Gamepad.current != null && Gamepad.current.buttonSouth.wasPressedThisFrame)
        {
            Debug.Log("Cargando escena índice: " + indiceDeLaEscenaACargar);
            SceneManager.LoadScene(indiceDeLaEscenaACargar);
        }
    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            jugadorDentro = true;

            if (panelDelLetrero != null)
                panelDelLetrero.SetActive(true);
        }
    }

    private void OnTriggerExit(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            jugadorDentro = false;

            if (panelDelLetrero != null)
                panelDelLetrero.SetActive(false);
        }
    }
}
```

Nota. *Código activación de canvas, elaboración propia.*

En la etapa siguiente se implementó el sistema de cambio de escena, encargado de gestionar la navegación entre los distintos módulos del entorno virtual. Para ello, se desarrolló un script que centraliza la activación de paneles informativos y la carga de escenas específicas según la selección realizada por el usuario.

El componente programado permite, en primera instancia, mostrar paneles correspondientes a cada módulo, asegurando que el botón asociado quede seleccionado dentro del sistema de eventos. Esta configuración facilita la interacción mediante controles físicos, como el gamepad, garantizando que el enfoque de navegación se asigne correctamente al elemento activo.

Figura **44**

activación de canvas según la posición

Nota. *Activación de canvas, elaboración propia.*

Posteriormente, se integró la funcionalidad de carga de escenas utilizando el gestor de escenas del motor gráfico. A través de métodos específicos, el sistema redirige al usuario al módulo correspondiente —ya sea rcp, sismo o fracturas y heridas— ejecutando la transición de manera directa y controlada. Además, se incorporaron mensajes de depuración que permiten verificar en consola la correcta ejecución de cada acción, facilitando el proceso de pruebas y validación.

Figura **45**

código para el cambio de escenas

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.EventSystems;

public class LoadScenes : MonoBehaviour
{
    public GameObject panelRCP;
    public GameObject panelSismo;

    public GameObject botonRCP;
    public GameObject botonSismo;

    // ============================
    // ACTIVAR PANEL RCP
    // ============================
    public void MostrarDialogoRCP()
    {
        panelRCP.SetActive(true);

        EventSystem.current.SetSelectedGameObject(null);
        EventSystem.current.SetSelectedGameObject(botonRCP);

        Debug.Log("Botón RCP seleccionado");
    }

    // ============================
    // ACTIVAR PANEL SISMO
    // ============================
    public void MostrarDialogoSismo()
    {
        panelSismo.SetActive(true);

        EventSystem.current.SetSelectedGameObject(null);
        EventSystem.current.SetSelectedGameObject(botonSismo);

        Debug.Log("Botón SISMO seleccionado");
    }

    // ============================
    // CARGAR ESCENAS
    // ============================
    public void CargarModulo1()
    {
        Debug.Log("Cargando Modulo_1_RCP");
        SceneManager.LoadScene("Modulo_1_RCP");
    }

    public void CargarModulo3()
    {
        Debug.Log("Cargando Modulo_3_Sismo");
        SceneManager.LoadScene("Modulo_3_Sismo");
    }

    public void CargarModulo2()
    {
        Debug.Log("Cargando Modulo_2_Heridas");
        SceneManager.LoadScene("Fracturas Y heridas");
    }
}
```

Nota. *Cambio de escena, elaboración propia.*

Posteriormente, se desarrolló un script destinado a la lectura y validación de las entradas provenientes del joystick. En particular, se configuró el d-pad para la navegación dentro de las interfaces y menús, permitiendo el desplazamiento entre botones y opciones seleccionables.

Por otro lado, el stick izquierdo (left stick) fue asignado al control de movimiento y otras acciones principales dentro del entorno virtual.

Figura 46

debug gamepad

```
using UnityEngine;
using UnityEngine.InputSystem;

public class GamepadDebug : MonoBehaviour
{
    void Update()
    {
        if (Gamepad.current == null) return;

        Vector2 dpad = Gamepad.current.dpad.ReadValue();
        Vector2 left = Gamepad.current.leftStick.ReadValue();

        if (dpad != Vector2.zero)
            Debug.Log("DPAD: " + dpad);

        if (left != Vector2.zero)
            Debug.Log("LEFT STICK: " + left);
    }
}
```

Nota. *Código debug, elaboración propia.*

En el módulo correspondiente a heridas y fracturas, se implementó un sistema de interacción basado en la detección de objetos mediante raycast, permitiendo una manipulación controlada y contextual dentro del entorno virtual. Para ello, se definieron dos tipos principales de objetos: el objeto agarrable, correspondiente al elemento manipulable (en este caso, el cartón), y el objeto receptor o agarrador, representado por la mano del personaje.

Figura **47**

código de objeto agarrable

```
using UnityEngine;

[RequireComponent(typeof(Rigidbody))]
public class Agarrable : MonoBehaviour
{
    private Material miMaterial;
    private Color colorOriginal;

    [Header("Feedback Visual")]
    [Tooltip("Color que tomará el objeto cuando el jugador apunte hacia él.")]
    public Color colorResaltado = Color.white;

    void Start()
    {
        // Esta parte no cambia: guardamos el material y su color inicial
        Renderer renderer = GetComponent<Renderer>();
        if (renderer != null)
        {
            miMaterial = renderer.material;
            colorOriginal = miMaterial.color;
        }
    }

    // --- HEMOS BORRADO OnMouseEnter y OnMouseExit ---

    // NUEVA FUNCIÓN PÚBLICA para que otro script nos ordene resaltar
    public void Resaltar()
    {
        if (miMaterial != null)
        {
            miMaterial.color = colorResaltado;
        }
    }

    // NUEVA FUNCIÓN PÚBLICA para que otro script nos ordene quitar el resaltado
    public void QuitarResaltado()
    {
        if (miMaterial != null)
        {
            miMaterial.color = colorOriginal;
        }
    }
}
```

Nota. *Objeto interactuable, elaboración propia.*

Figura **48**

código de objeto agarrador

```
using UnityEngine;
using UnityEngine.InputSystem;

public class Agarrador : MonoBehaviour
{
    // Variables para el objeto agarrado
    private GameObject objetoAgarrado;
    private float distanciaObjeto;

    // --- NUEVA VARIABLE para controlar el resaltado ---
    private Agarrable ultimoObjetoResaltado;

    [Header("Configuración de Agarre")]
    public float distanciaDeAgarre = 5f;
    public LayerMask capaDeAgarrables;

    [Header("Referencias")]
    public Transform puntoDeOrigenDelRayo;
    private Camera camaraJugador;

    void Start()
    {
        camaraJugador = Camera.main;
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;

        if (puntoDeOrigenDelRayo == null)
        {
            puntoDeOrigenDelRayo = camaraJugador.transform;
        }
    }

    void Update()
    {
        // --- LÓGICA DE RESALTADO ---
        Ray ray = new Ray(puntoDeOrigenDelRayo.position, camaraJugador.transform.forward);
        RaycastHit hit;

        if (Physics.Raycast(ray, out hit, distanciaDeAgarre, capaDeAgarrables))
        {
            Debug.Log("El rayo ha golpeado a: " + hit.transform.name);
            // Si el rayo golpea un objeto agarrable...
            Agarrable agarrable = hit.transform.GetComponent<Agarrable>();
            if (agarrable != null)
            {
                Debug.Log("¡Objeto agarrable detectado! Intentando resaltar.");
                // Si es un objeto diferente al que estábamos resaltando antes
                if (ultimoObjetoResaltado != agarrable)
                {
                    // Apagamos el resaltado del anterior (si lo había)
                    if (ultimoObjetoResaltado != null) ultimoObjetoResaltado.QuitarResaltado();

                    // Resaltamos el nuevo y lo guardamos como el último
                    agarrable.Resaltar();
                    ultimoObjetoResaltado = agarrable;
                }
            }
        }
        else // Si el rayo no golpea nada
        {
            // Si estábamos resaltando algo, lo apagamos
            if (ultimoObjetoResaltado != null)
            {
                ultimoObjetoResaltado.QuitarResaltado();
                ultimoObjetoResaltado = null; // Olvidamos la referencia
            }
        }

        // --- LÓGICA DE AGARRE (no cambia) ---
        if (Mouse.current.leftButton.wasPressedThisFrame)
        {
            if (objetoAgarrado == null)
            {
                if (ultimoObjetoResaltado != null) // Si estamos apuntando a algo resaltado
                {
                    objetoAgarrado = ultimoObjetoResaltado.gameObject;
                    objetoAgarrado.GetComponent<Rigidbody>().isKinematic = true;
                    distanciaObjeto = hit.distance;
                }
            }
            else
            {
                objetoAgarrado.GetComponent<Rigidbody>().isKinematic = false;
                objetoAgarrado = null;
            }
        }

        // --- LÓGICA DE MOVIMIENTO (no cambia) ---
        if (objetoAgarrado != null)
        {
            Vector3 nuevaPosicion = ray.GetPoint(distanciaObjeto);
            objetoAgarrado.transform.position = nuevaPosicion;
        }
    }
}
```

Nota. *Código objeto agarrador, elaboración propia.*

El funcionamiento del sistema se basa en la emisión constante de un rayo desde el punto de vista del usuario. Cuando el raycast detecta que el jugador está observando el objeto agarrable, se activa un efecto visual de resaltado o borde, indicando que el elemento puede ser interactuado. Esta retroalimentación visual facilita la identificación de objetos disponibles y mejora la experiencia de usuario.

Figura **49**

borde activado por raycast

Nota. *Borde activador, elaboración propia.*

Una vez confirmado el comando de interacción, el objeto se desplaza automáticamente hacia la posición predefinida de la mano, simulando el proceso de agarre. Posteriormente, al dirigir la vista hacia el objeto de destino, el sistema vuelve a utilizar la detección por raycast para validar la acción de colocación. Si se cumplen las condiciones establecidas, el objeto es trasladado a la posición del destino, completando así la secuencia de interacción.

Figura **50**

zona de destino del objeto

```
using UnityEngine;
using UnityEngine.InputSystem; // Añadimos la referencia al nuevo sistema

public class ZonaDeEntrega : MonoBehaviour
{
    [Header("Configuración de la Zona")]
    [Tooltip("El Tag del objeto que esta zona debe aceptar.")]
    public string tagObjetoCorrecto;

    [Tooltip("Punto exacto donde el objeto se 'ajustará'. Si se deja vacío, usará la posición de esta zona.")]
    public Transform puntoDeAjuste;

    private Color colorOriginal;
    private Renderer miRenderer;

    void Start()
    {
        GetComponent<Collider>().isTrigger = true;
        miRenderer = GetComponent<Renderer>();
        if (miRenderer != null)
        {
            colorOriginal = miRenderer.material.color;
        }
    }

    void OnTriggerEnter(Collider other)
    {
        if (miRenderer != null && other.CompareTag(tagObjetoCorrecto))
        {
            miRenderer.material.color = Color.green;
        }
    }

    // Se llama continuamente mientras un objeto permanece dentro del Trigger.
    void OnTriggerStay(Collider other)
    {
        // AQUÍ ESTÁ EL CAMBIO IMPORTANTE:
        // Comprobamos si el objeto es el correcto Y si el jugador ha HECHO CLIC (para soltarlo).
        if (other.CompareTag(tagObjetoCorrecto) && Mouse.current.leftButton.wasPressedThisFrame)
        {
            // Pequeña comprobación para asegurarnos de que el jugador no está intentando coger el objeto de la zona.
            // Si el Agarrador.cs suelta el objeto, lo pondrá en null ANTES de que este código se ejecute.
            // Esto es un poco más avanzado, pero funciona bien en este caso.
            // Básicamente, si se hace clic mientras el objeto está en la zona, lo consideramos como una entrega.

            GameObject objeto = other.gameObject;

            // Ajustamos la posición y rotación
            if (puntoDeAjuste != null)
            {
                objeto.transform.position = puntoDeAjuste.position;
                objeto.transform.rotation = puntoDeAjuste.rotation;
            }
            else
            {
                objeto.transform.position = transform.position;
            }

            objeto.GetComponent<Rigidbody>().isKinematic = true;
            Debug.Log("¡Objeto colocado correctamente!");
        }
    }

    void OnTriggerExit(Collider other)
    {
        if (miRenderer != null && other.CompareTag(tagObjetoCorrecto))
        {
            miRenderer.material.color = colorOriginal;
        }
    }
}
```

Nota. *Código zona de destino, elaboración propia.*

Figura **51**

acciones activar borde de destino y colocar objeto

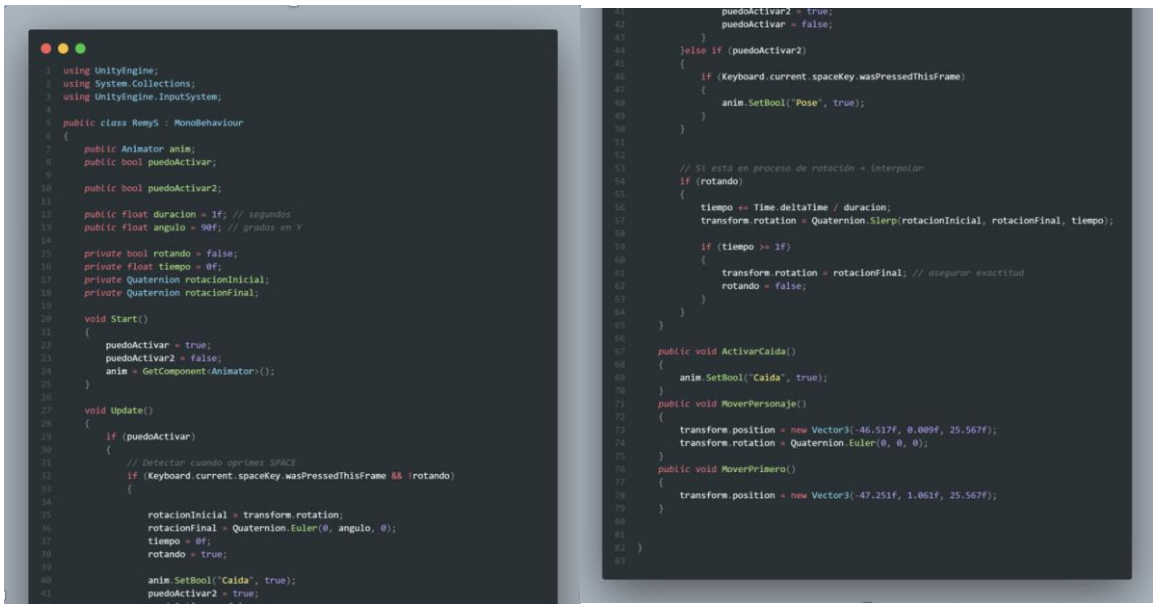
Nota. *Borde destino, elaboración propia.*

En esta fase del desarrollo se implementó la animación del personaje con el propósito de representar de manera visual el evento que da origen al escenario clínico del módulo. Para ello, se diseñó una secuencia animada en la que el personaje desciende por unas escaleras, pierde el equilibrio y sufre una caída que culmina en una fractura en uno de sus brazos.

La animación fue estructurada como una secuencia progresiva que incluye desplazamiento inicial, pérdida de estabilidad, caída y postura final en el suelo. Se ajustaron parámetros de transición y tiempos dentro del sistema de animación para garantizar fluidez y coherencia en los movimientos, evitando cortes bruscos entre estados. Asimismo, se cuidó la postura final del personaje para que reflejara visualmente la lesión en la extremidad superior, reforzando la narrativa del escenario.

Figura **52**

código de las animaciones del personaje (heridas y fracturas)



```
using UnityEngine;
using System.Collections;
using UnityEngine.InputSystem;

public class RemyS : MonoBehaviour
{
    public Animator anim;
    public bool puedoActivar;

    public bool puedoActivar2;

    public float duracion = 1f; // segundos
    public float angulo = 90f; // grados en Y

    private bool rotando = false;
    private float tiempo = 0f;
    private Quaternion rotacionInicial;
    private Quaternion rotacionFinal;

    void Start()
    {
        puedoActivar = true;
        puedoActivar2 = false;
        anim = GetComponent<Animator>();
    }

    void Update()
    {
        if (puedoActivar)
        {
            // Detectar cuando oprimes SPACE
            if (Keyboard.current.spaceKey.wasPressedThisFrame && !rotando)
            {

                rotacionInicial = transform.rotation;
                rotacionFinal = Quaternion.Euler(0, angulo, 0);
                tiempo = 0f;
                rotando = true;

                anim.SetBool("Caida", true);
                puedoActivar2 = true;
                puedoActivar = false;
            }
        }else if (puedoActivar2)
        {
            if (Keyboard.current.spaceKey.wasPressedThisFrame)
            {
                anim.SetBool("Pose", true);
            }
        }


        // Si está en proceso de rotación → interpolar
        if (rotando)
        {
            tiempo += Time.deltaTime / duracion;
            transform.rotation = Quaternion.Slerp(rotacionInicial, rotacionFinal, tiempo);

            if (tiempo >= 1f)
            {
                transform.rotation = rotacionFinal; // asegurar exactitud
                rotando = false;
            }
        }
    }

    public void ActivarCaida()
    {
        anim.SetBool("Caida", true);
    }
    public void MoverPersonaje()
    {
        transform.position = new Vector3(-46.517f, 0.009f, 25.567f);
        transform.rotation = Quaternion.Euler(0, 0, 0);
    }
    public void MoverPrimero()
    {
        transform.position = new Vector3(-47.251f, 1.061f, 25.567f);
    }


}
```

Nota. *Código animaciones personaje, elaboración propia.*

Figura **53**

Caída de personaje

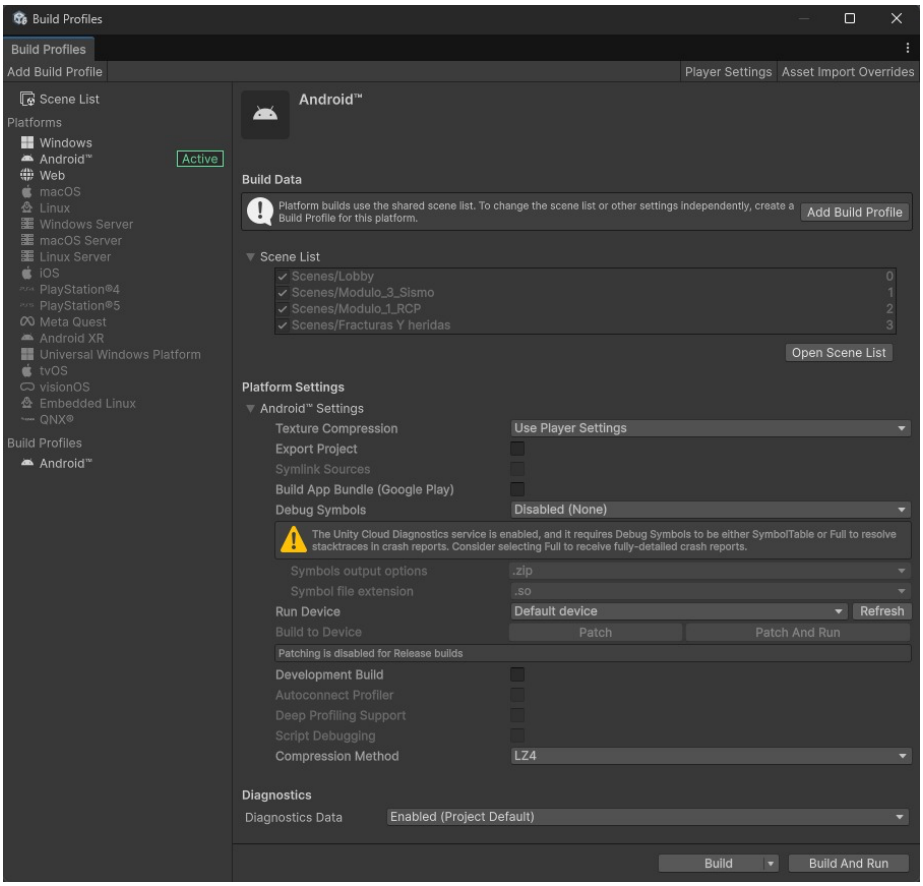
Fuente: elaboración propia.

Para garantizar la correcta ejecución del sistema de entrenamiento en dispositivos móviles mediante gafas de realidad virtual tipo cardboard, fue necesario realizar la configuración del plugin google cardboard xr dentro del motor unity. Esta configuración permitió habilitar el renderizado estereoscópico, el seguimiento de movimiento mediante sensores del dispositivo y la compatibilidad con el sistema operativo android.

A continuación, se describen los pasos realizados durante la configuración del entorno.

Figura ***54***

configuración de build profiles.



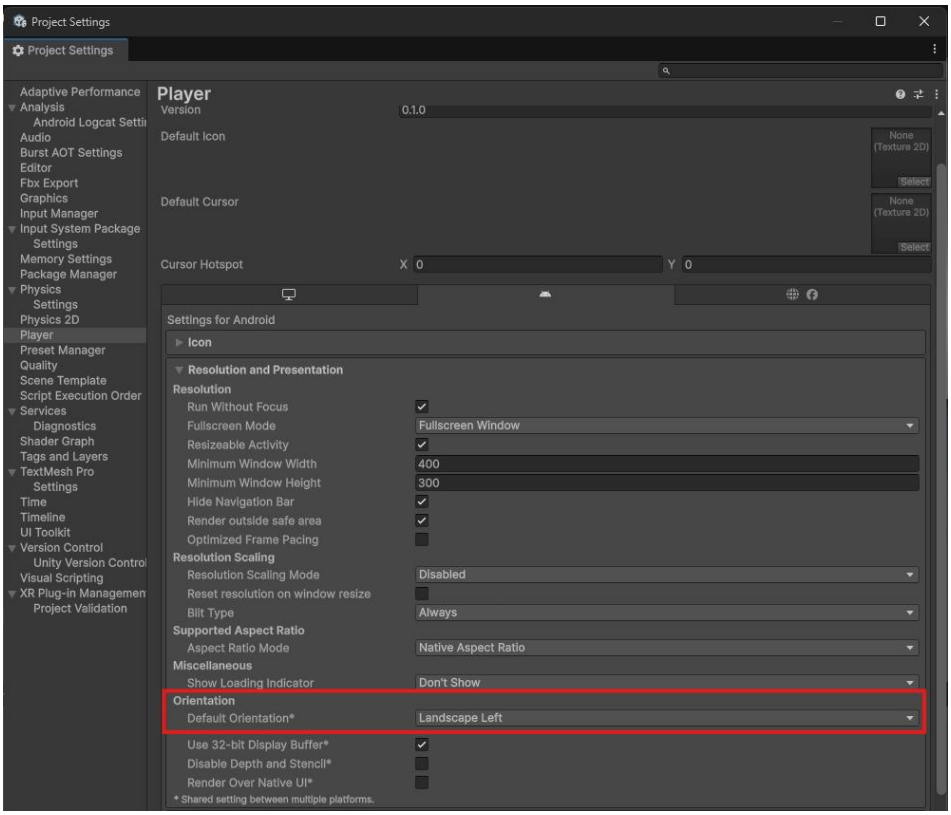
Fuente: elaboración propia.

Se estableció la ejecución directa en el dispositivo móvil mediante conexión usb, activando el modo de depuración (usb debugging) y utilizando la opción *run device* desde unity. Esta estrategia permitió reducir significativamente los tiempos de instalación, facilitando pruebas iterativas constantes durante el desarrollo.

Asimismo, se verificó la correcta inclusión de las escenas dentro del apartado *scenes in build*, asegurando que cada nivel contara con su respectivo indicador de carga.

Figura **55**

configuración de orientación del dispositivo.



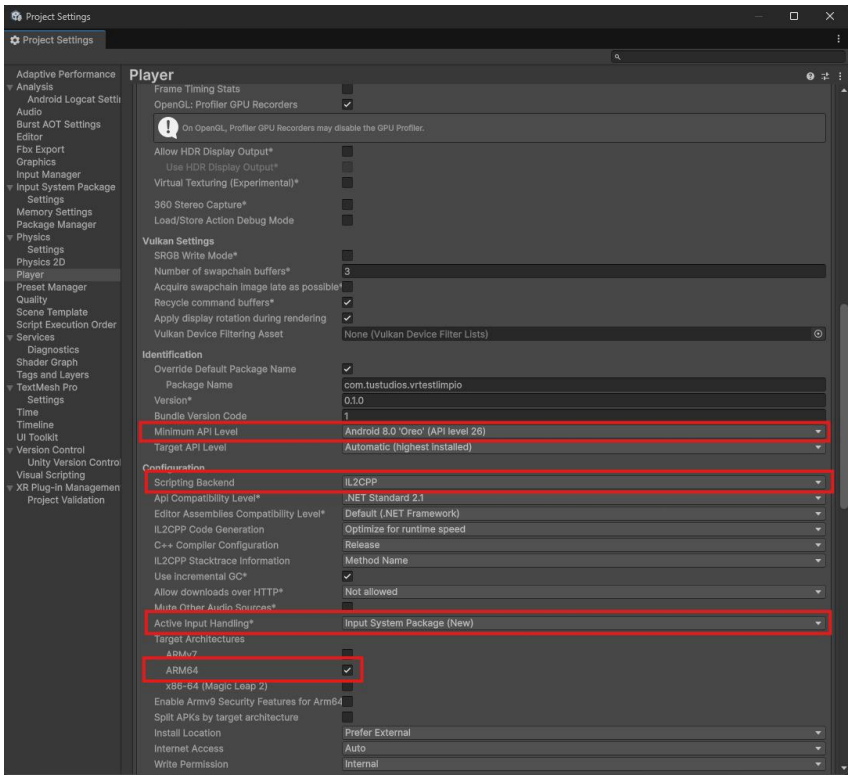
Fuente: elaboración propia.

Posteriormente, dentro del apartado project settings, se revisó la configuración de orientación del dispositivo.

Se estableció la orientación fija en modo horizontal (landscape), garantizando la correcta visualización estereoscópica en las gafas cardboard. Este ajuste se realizó considerando las dimensiones del dispositivo móvil, el tipo de visor utilizado y los requerimientos de renderizado vr.

Figura **56**

configuración de compatibilidad del proyecto.



Fuente: elaboración propia.

En el mismo apartado de project settings, se revisaron parámetros esenciales para la compatibilidad del proyecto con dispositivos android.

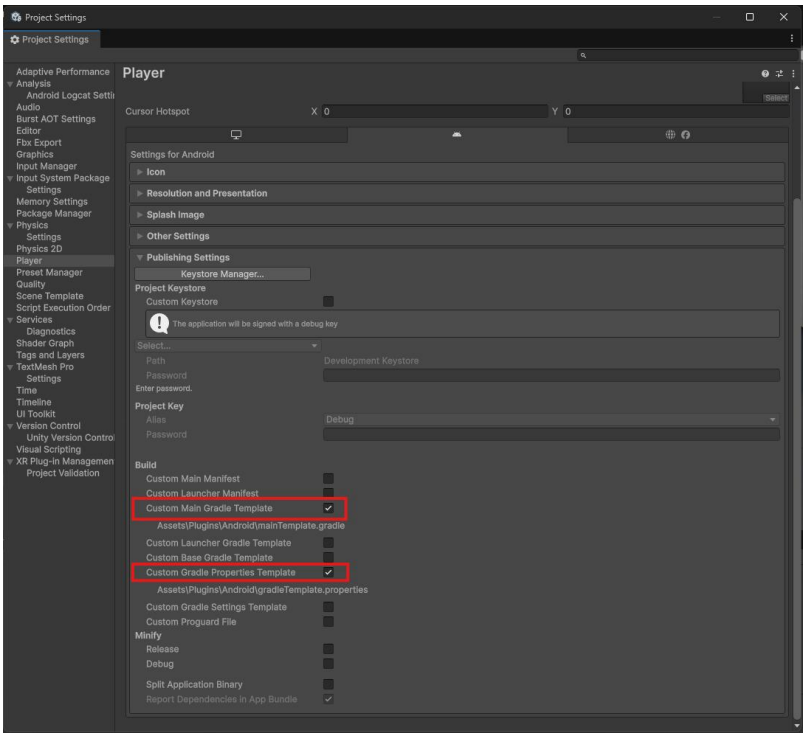
Se habilitaron las opciones:

- Custom main gradle template
- Custom gradle properties template

La activación de estas plantillas permitió integrar adecuadamente las dependencias requeridas por el plugin cardboard, evitando conflictos en la generación del archivo apk y garantizando la correcta compilación del proyecto.

Figura **57**

activación de custom gradle templates.



Fuente: elaboración propia.

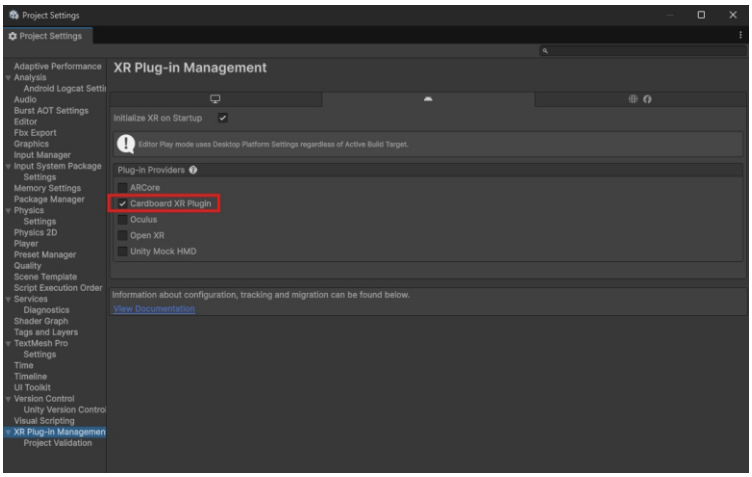
Dentro del sistema de administración de xr en unity, se verificó la activación del cardboard xr plugin.

Esta configuración permitió:

- Renderizado en modo estereoscópico.
- División de pantalla automática para visor vr.
- Seguimiento de movimiento mediante sensores del dispositivo (giroscopio y acelerómetro).

Figura **58**

activación del cardboard xr plugin.



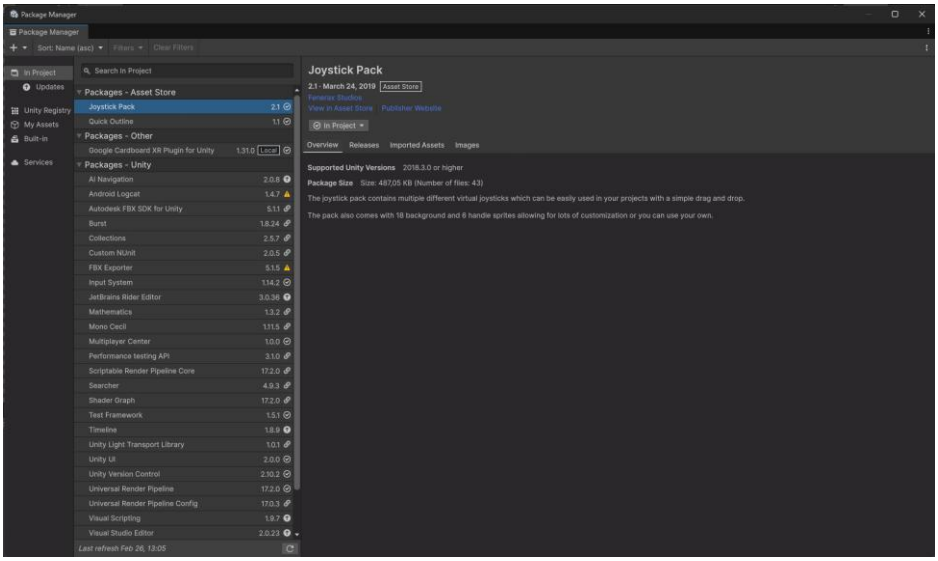
Fuente: elaboración propia.

Finalmente, se utilizó el package manager de unity para confirmar la correcta instalación del paquete google cardboard xr y verificar que no existieran conflictos de versiones.

Se validó que todas las dependencias necesarias estuvieran correctamente integradas en el proyecto y que el plugin se encontrara actualizado.

Figura **59**

verificación mediante package manager.



Fuente: elaboración propia

9. TESTER

9.1. Metodología de Evaluación

Con el fin de garantizar una evaluación sistemática y reproducible del sistema de entrenamiento desarrollado, se definió una metodología de evaluación estructurada que rige la ejecución de todas las pruebas descritas en el presente capítulo. Esta metodología contempla el perfil de los evaluadores, los criterios de éxito, las métricas de desempeño registradas y la forma en que los resultados se relacionan con la evidencia estadística del Anexo 5.

9.1.1. Alcance y perfil de los evaluadores

Las pruebas fueron realizadas por los dos integrantes del equipo de desarrollo, quienes asumieron roles diferenciados durante cada sesión de evaluación: uno en calidad de evaluador técnico (verificando el comportamiento interno del código) y el otro como usuario de aceptación (interactuando con el sistema sin intervención del evaluador). Dado que el proyecto corresponde a un artefacto de software verificado como requisito de la tesis, y no a un estudio de efectividad pedagógica, la muestra de usuarios no requirió criterios estadísticos de representatividad poblacional. La validación pedagógica con muestras amplias se proyecta como trabajo futuro (ver sección 11).

Para las pruebas de aceptación de usuario (sección 9.2.5), uno de los integrantes utilizó el dispositivo Google Cardboard en condiciones reales de uso, siguiendo únicamente las instrucciones provistas por el propio simulador, sin orientación externa del evaluador.

9.1.2. Criterios de éxito

Se definieron los siguientes criterios de éxito para declarar el sistema como verificado:

Criterio técnico-funcional: todos los módulos deben ejecutarse sin errores críticos (crashes) durante una sesión completa de cada nivel.

Criterio de rendimiento: la aplicación debe mantener una tasa mínima de 30 fps en el hardware de referencia (Google Cardboard con dispositivo móvil), conforme al requisito no funcional RNF01.

Criterio de usabilidad: el usuario de aceptación debe ser capaz de completar al menos un módulo de entrenamiento sin asistencia externa, siguiendo únicamente las instrucciones del sistema.

Criterio de integridad de datos: el sistema de telemetría debe generar y almacenar correctamente el archivo de registro en formato JSON al finalizar cada sesión, conforme a la regla de negocio RN06.

Criterio de corrección lógica: los validadores de protocolo (profundidad de RCP, secuencialidad en trauma, tiempo de reacción en sismos) deben activarse y desactivarse conforme a las reglas de negocio RN01 a RN05.

9.1.3. Métricas de desempeño registradas

Durante las sesiones de prueba se registraron las siguientes métricas de manera directa o a través del sistema de telemetría implementado:

Tabla 23

Métricas de desempeño registradas durante las sesiones de prueba.

Categoría	Métrica	Instrumento de medición
Rendimiento técnico	Frames por segundo (FPS) promedio	GameTurbo / Unity Profiler
Rendimiento técnico	Uso de CPU (%)	GameTurbo
Corrección funcional	Activación/desactivación de triggers y colliders	Revisión de consola Unity
Corrección funcional	Validación de protocolo RCP (profundidad y ritmo)	Script de evaluación en tiempo real
Corrección funcional	Secuencialidad en módulo de trauma	Log de telemetría JSON
Corrección funcional	Tiempo de reacción en módulo de sismos	Contador interno de la escena
Usabilidad	Capacidad de completar un módulo sin asistencia	Observación directa (prueba de aceptación)
Integridad de datos	Generación correcta del archivo JSON de telemetría	Verificación manual del archivo generado

9.1.4. Protocolo de ejecución de pruebas

Cada sesión de prueba siguió el siguiente protocolo estandarizado: (1) instalación de la versión compilada (.apk) en el dispositivo móvil de referencia; (2) verificación del entorno físico para la prueba con cardboard (espacio libre, iluminación adecuada); (3) ejecución del módulo seleccionado desde el inicio sin intervenciones; (4) registro manual y automático de las métricas definidas; (5) revisión de la consola de Unity y del archivo JSON generado al finalizar; y (6) documentación de errores encontrados y correcciones aplicadas en la siguiente iteración.

9.2. Pruebas de caja blanca

Las pruebas de caja blanca se realizaron con el propósito de evaluar el comportamiento interno del sistema, verificando la correcta ejecución de la lógica programada, el flujo de estados y la validación de condiciones dentro del motor de simulación en realidad virtual.

Este tipo de pruebas permitió analizar directamente el código fuente, las estructuras condicionales, los algoritmos de validación y la interacción entre los distintos módulos del sistema.

9.2.1. pruebas de funcionamiento de colliders

Durante las pruebas de caja blanca se realizó la validación interna del comportamiento de los **colliders** y **triggers**, elementos fundamentales para la detección de colisiones físicas y activación de interfaces dentro del sistema de entrenamiento en realidad virtual.

Durante las primeras pruebas se presentó un inconveniente frecuente relacionado con la interacción entre colliders y triggers. En el nivel de atención de heridas y fracturas, al acercarse al paciente virtual, el canvas de instrucciones no se activaba correctamente o se activaba múltiples veces de manera intermitente. Asimismo, en algunos casos el sistema registraba múltiples colisiones simultáneas con un mismo objeto médico.

Tras revisar el código y la configuración en el inspector de unity, se determinó que:

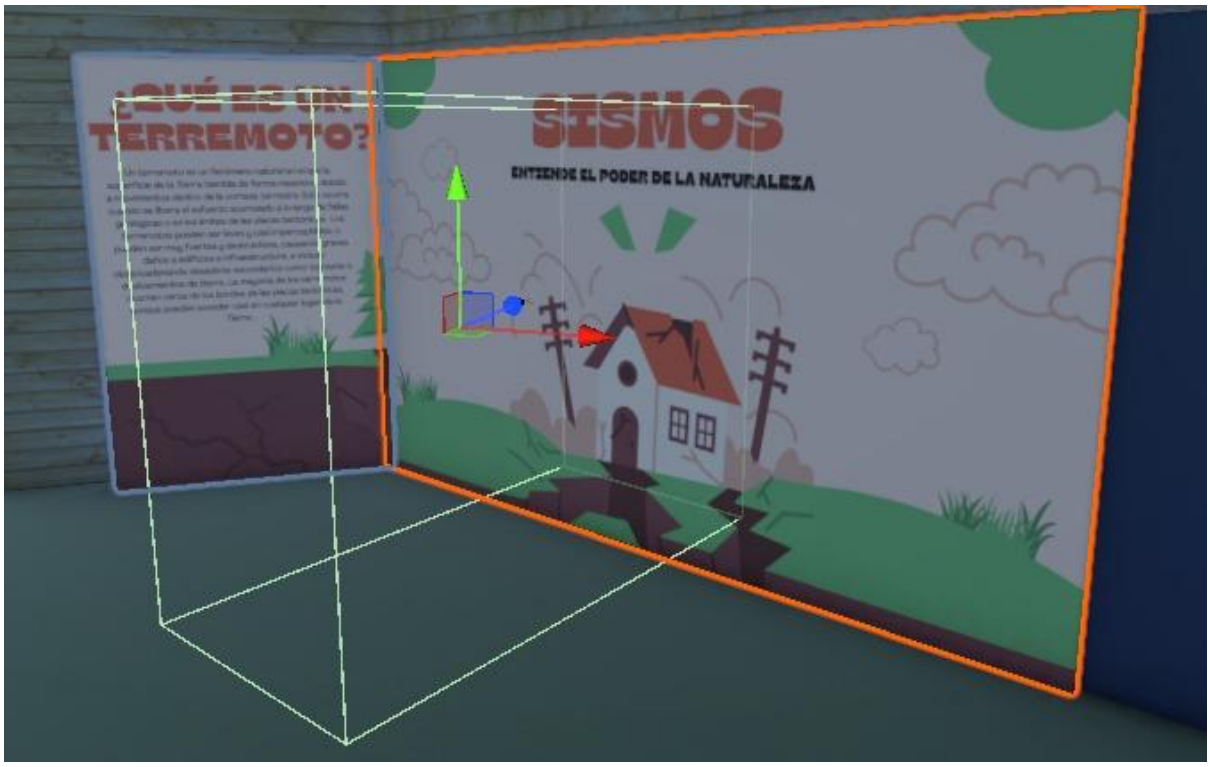
- Algunos objetos tenían activada simultáneamente la opción **is trigger** y requerían comportamiento físico.

- Existían colliders superpuestos en el mismo objeto.
- El jugador no tenía correctamente configurado el componente **rigidbody**, requisito necesario para la detección adecuada de eventos ontriggerenter().

Adicionalmente, en ciertos escenarios los canvas se activaban repetidamente debido a que no existía una condición booleana que controlara la ejecución única del evento.

Figura 60

colliders de activación de canvas de niveles



Fuente: elaboración propia

9.2.2. Pruebas de condiciones booleanas y estructuras if

Durante la fase de pruebas internas del sistema, se realizó la validación de las condiciones booleanas encargadas de controlar el flujo de ejecución de las animaciones y transiciones del personaje dentro del entorno virtual. Estas variables (puedoactivar, puedoactivar2 y rotando) eran responsables de permitir o bloquear determinadas acciones del usuario durante la simulación.

En las primeras ejecuciones del sistema, se observó un comportamiento inconsistente en la secuencia de animación del personaje. Al presionar la tecla correspondiente, la animación

de caída se activaba correctamente; sin embargo, en algunas ocasiones el personaje volvía a ejecutar la misma animación de forma repetida o quedaba bloqueado sin permitir avanzar a la siguiente fase (pose final).

Adicionalmente, durante el proceso de rotación interpolada, el sistema permitía múltiples activaciones simultáneas si el usuario presionaba repetidamente la tecla, generando interrupciones en la interpolación y resultados inesperados en la orientación del personaje.

Figura 61

código de lógica animaciones

```
    void Update()
    {
        if (puedoActivar)
        {
            // Detectar cuando oprimes SPACE
            if (Keyboard.current.spaceKey.wasPressedThisFrame && !rotando)
            {

                rotacionInicial = transform.rotation;
                rotacionFinal = Quaternion.Euler(0, angulo, 0);
                tiempo = 0f;
                rotando = true;

                anim.SetBool("Caida", true);
                puedoActivar2 = true;
                puedoActivar = false;
            }
        }else if (puedoActivar2)
        {
            if (Keyboard.current.spaceKey.wasPressedThisFrame)
            {
                anim.SetBool("Pose", true);
            }
        }


        // Si está en proceso de rotación → interpolar
        if (rotando)
        {
            tiempo += Time.deltaTime / duracion;
            transform.rotation = Quaternion.Slerp(rotacionInicial, rotacionFinal, tiempo);

            if (tiempo >= 1f)
            {
                transform.rotation = rotacionFinal; // asegurar exactitud
                rotando = false;
            }
        }
    }
```

Fuente: elaboración propia

9.2.3. Pruebas de límites de velocidad de movimiento

Durante la fase de pruebas internas del sistema se realizó la validación del controlador de movimiento implementado mediante el componente charactercontroller. Esta validación fue especialmente importante debido a que el desplazamiento del usuario dentro del entorno vr debía ser estable, progresivo y controlado, evitando movimientos bruscos que afectaran la experiencia inmersiva.

En las primeras pruebas funcionales del nivel de simulación (especialmente en el escenario de sismo y desplazamiento hacia zonas seguras), se detectó que el personaje presentaba movimientos excesivamente rápidos cuando el valor de movespeed era incrementado para mejorar la fluidez.

Tras el análisis del código, se determinó que el parámetro movespeed actuaba como multiplicador directo del vector de movimiento. Al no existir inicialmente una validación de límites máximos permitidos, cualquier incremento en este valor provocaba un desplazamiento proporcionalmente mayor

Para corregir el problema se realizaron pruebas controladas ajustando el valor de movespeed hasta determinar un rango óptimo (entre 3f y 5f) que equilibrara fluidez y estabilidad, evitando valores superiores que comprometieran la precisión en la interacción, manteniendo el uso de time.deltatime para asegurar independencia de la tasa de frames, y validando que el personaje no atravesara triggers sin activar eventos; adicionalmente, se verificó que la condición if (controller.isgrounded && velocity.y < 0) restableciera correctamente la velocidad vertical al tocar el suelo, evitando acumulaciones innecesarias que generaran rebotes o micro saltos.

Figura **62**

código de movimientos

```cs
using UnityEngine;
using UnityEngine.InputSystem;

[RequireComponent(typeof(CharacterController))]
public class MotionObjectController : MonoBehaviour
{
    [Header("Movimiento")]
    public float moveSpeed = 5f;
    public float gravity = -9.81f;

    private CharacterController controller;
    private Vector3 velocity;

    [Header("Referencia Cámara")]
    public Transform cameraTransform;

    void Start()
    {
        controller = GetComponent<CharacterController>();

        if (cameraTransform == null)
        {
            cameraTransform = Camera.main.transform;
        }
    }

    void Update()
    {
        MovePlayer();
        ApplyGravity();
    }

    void MovePlayer()
    {
        Gamepad gamepad = Gamepad.current;

        if (gamepad == null)
            return;

        Vector2 input = gamepad.leftStick.ReadValue();

        // Dirección de la cámara (horizontal)
        Vector3 forward = cameraTransform.forward;
        Vector3 right = cameraTransform.right;

        forward.y = 0f;
        right.y = 0f;

        forward.Normalize();
        right.Normalize();

        // Movimiento relativo a la cámara
        Vector3 move = forward * input.y + right * input.x;

        controller.Move(move * moveSpeed * Time.deltaTime);
    }

    void ApplyGravity()
    {
        if (controller.isGrounded && velocity.y < 0)
        {
            velocity.y = -2f;
        }

        velocity.y += gravity * Time.deltaTime;

        controller.Move(velocity * Time.deltaTime);
    }
}
```

Fuente: elaboración propia

9.2.4. Pruebas de distancia del raycast

Durante las pruebas, se detectaron inconsistencias al seleccionar y manipular objetos en el entorno virtual. Los usuarios podían recoger elementos a distancias irreales o a través de otros modelos, el sistema a veces ignoraba objetos visibles, y ocurrían comportamientos físicos erráticos al momento de soltarlos.

Estas fallas se originaron porque el parámetro de distancia del *raycast* no estaba calibrado correctamente para la escala del entorno. Además, las capas (*layermask*) de algunos objetos estaban mal configuradas, lo que confundía el impacto del rayo, y el componente *rigidbody* de varios elementos se encontraba ausente o con parámetros incorrectos.

Para resolverlo, se ajustó la variable de distancia a un rango de interacción lógico y realista para el usuario. Asimismo, se corrigió la asignación de las *layermask* en todos los objetos interactuables y se configuró debidamente el *rigidbody* en cada elemento, logrando finalmente un sistema de agarre preciso, estable y natural.

Figura 63

raycast

```
using System.Collections;
using UnityEngine.InputSystem;
using UnityEngine;

public class RaycastPickup : MonoBehaviour
{
    public float distancia = 10f; // rango del rayo
    public LayerMask capaInteractuable; // capa para los objetos que se pueden agarrar
    public Transform mano; // donde se "pegará" el objeto

    private GameObject objetoTomado;

    void Update()
    {
        if (objetoTomado == null)
        {
            // Lanzar rayo desde el centro de la cámara hacia adelante
            Ray rayo = new Ray(Camera.main.transform.position, Camera.main.transform.forward);
            RaycastHit hit;

            if (Physics.Raycast(rayo, out hit, distancia, capaInteractuable))
            {
                Debug.Log("Miras a: " + hit.collider.name);

                // Al presionar click izquierdo, agarrar objeto
                if (Input.GetMouseButtonDown(0))
                {
                    objetoTomado = hit.collider.gameObject;
                    objetoTomado.transform.SetParent(mano);
                    objetoTomado.transform.localPosition = Vector3.zero;
                    objetoTomado.GetComponent<Rigidbody>().isKinematic = true;
                }
            }
        }
        else
        {
            // Soltar objeto con click derecho
            if (Input.GetMouseButtonDown(1))
            {
                objetoTomado.GetComponent<Rigidbody>().isKinematic = false;
                objetoTomado.transform.SetParent(null);
                objetoTomado = null;
            }
        }
    }
}
```

Nota. *Código raycast, elaboración propia.*

9.3. Pruebas de caja negra

9.3.1. Pruebas de funcionalidad del lobby

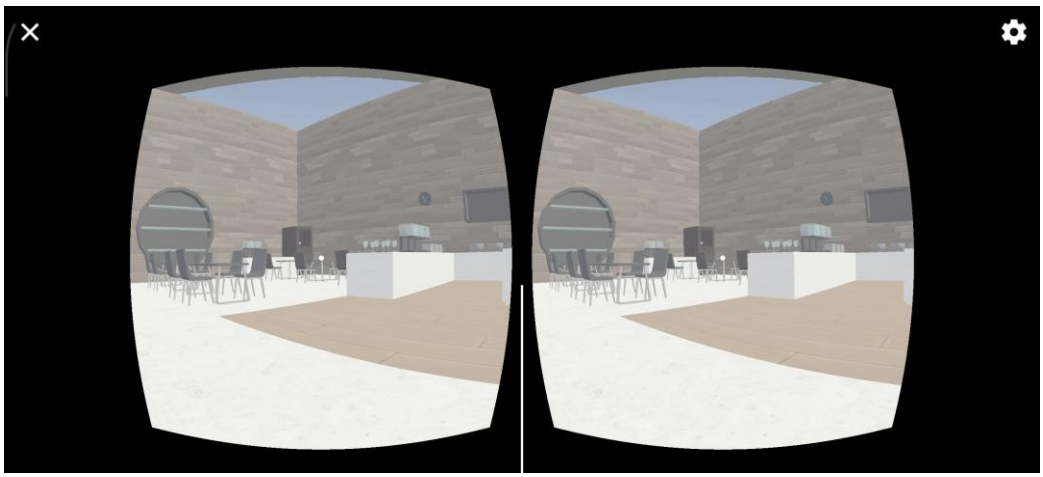
Durante las pruebas de navegación libre en el lobby 3d, el desplazamiento del usuario era errático y avanzaba dando saltos. Además, al entrar y salir rápidamente de las zonas de interacción, el canvas de selección se quedaba atascado en la pantalla. Sumado a esto, si el jugador presionaba el botón de "seleccionar módulo" varias veces seguidas, el juego se congelaba y colapsaba.

Los saltos al caminar se debían a que el colisionador de la zona de interacción no estaba configurado como "trigger", provocando que el avatar chocara físicamente con una barrera invisible. El problema del canvas ocurría porque el evento de salida del área no estaba bien vinculado para ocultar la interfaz. El colapso del juego sucedía porque el botón no se bloqueaba tras el primer clic, intentando cargar el pesado modelo 3d varias veces simultáneamente.

Se activó la casilla de "trigger" en la zona de interacción, restaurando un movimiento fluido y sin colisiones indebidas. Se programó la ocultación inmediata del canvas al abandonar el área y se configuró el botón para que se deshabilite tras el primer toque, garantizando una transición única y estable hacia los módulos de entrenamiento.

Figura 64

Prueba del Lobby



Nota. *Prueba lobby, elaboración propia.*

9.3.2. Pruebas funcionales – nivel rcp

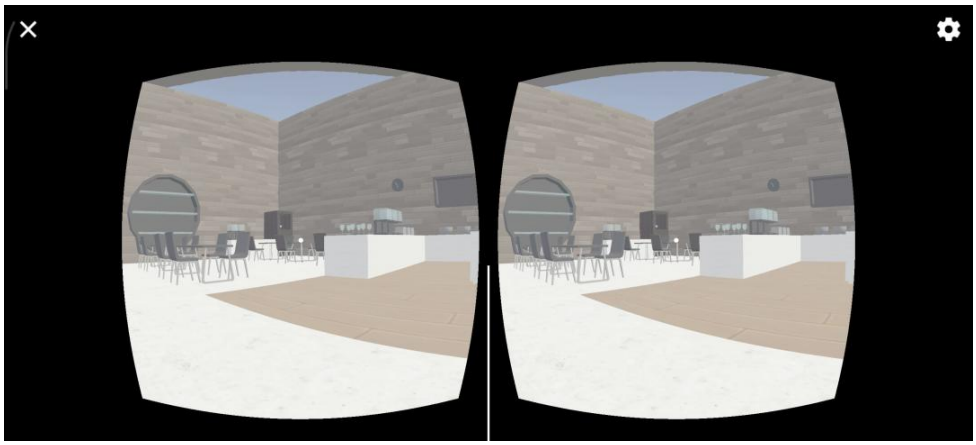
Durante las pruebas del módulo de rcp, se detectó que al presionar el botón siguiendo el ritmo exacto indicado por el sistema, muchas pulsaciones se registraban como fallidas. Por el contrario, si el usuario ignoraba el ritmo y presionaba el botón desesperadamente lo más rápido posible, el simulador lo registraba como un éxito total y otorgaba una puntuación perfecta, arruinando el propósito del entrenamiento médico.

Este comportamiento errático ocurría porque el sistema no tenía un tiempo de bloqueo (cooldown) entre pulsaciones, permitiendo acumular múltiples entradas "válidas" en un solo segundo al presionar los controles al azar. Además, el temporizador que evaluaba el ritmo dependía directamente de los fotogramas por segundo (fps) del juego en lugar de un reloj en tiempo real, lo que desfasaba la ventana de acierto si el juego sufría una leve caída de rendimiento.

Se implementó una ventana de tiempo estricta y un tiempo de enfriamiento en la programación del botón, ignorando por completo cualquier pulsación extra que se haga fuera del compás esperado (evitando la trampa de presionar rápido). También se independizó la lógica de evaluación gráfica del rendimiento del procesador usando variables de tiempo real, logrando que el sistema ahora exija y premie únicamente la precisión y el ritmo constante del jugador.

Figura 65

Prueba del nivel de RCP



Nota. *Prueba módulo RCP, elaboración propia.*

9.3.3. Pruebas funcionales – nivel heridas y fracturas

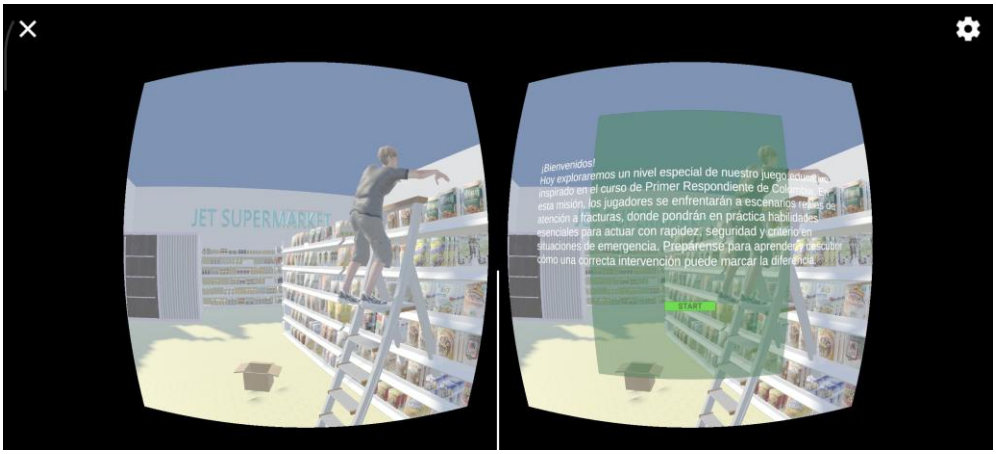
Durante las pruebas del nivel de heridas y fracturas, se observó que el usuario podía agarrar el trozo de cartón con el raycast mientras el trabajador aún estaba cayendo de las escaleras en la animación inicial, rompiendo toda la secuencia lógica del escenario. Además, tras escuchar los audios y leer los canvas con las instrucciones, al intentar colocar el cartón para inmovilizar el brazo lesionado, el objeto rara vez se adhería, obligando al jugador a apuntar con extrema precisión a un punto invisible para lograr que encajara.

El error de secuencia ocurría porque el componente agarrable del cartón estaba habilitado desde el primer segundo de la escena, sin esperar a que la animación de la caída y los audios terminaran. Por otro lado, la dificultad para colocar el cartón se debía a que la zona de destino en el brazo del trabajador era un colisionador demasiado pequeño y no estaba correctamente emparentado al hueso del modelo 3d, quedando desfasado de la posición visual del brazo tras la animación.

Se programó un bloqueo inicial en el cartón, logrando que el raycast solo pueda interactuar con él una vez que la animación de la caída finaliza y las instrucciones del canvas y audio se han reproducido por completo. Asimismo, se amplió el colisionador en el brazo del paciente y se vinculó directamente a su esqueleto virtual, creando una zona de "imán" mucho más permisiva y natural que asegura el acoplamiento del cartón de forma fluida.

Figura **66**

zona de destino del brazo.



Nota. *Prueba módulo heridas y fracturas, elaboración propia.*

9.3.4. Pruebas funcionales – nivel sismos

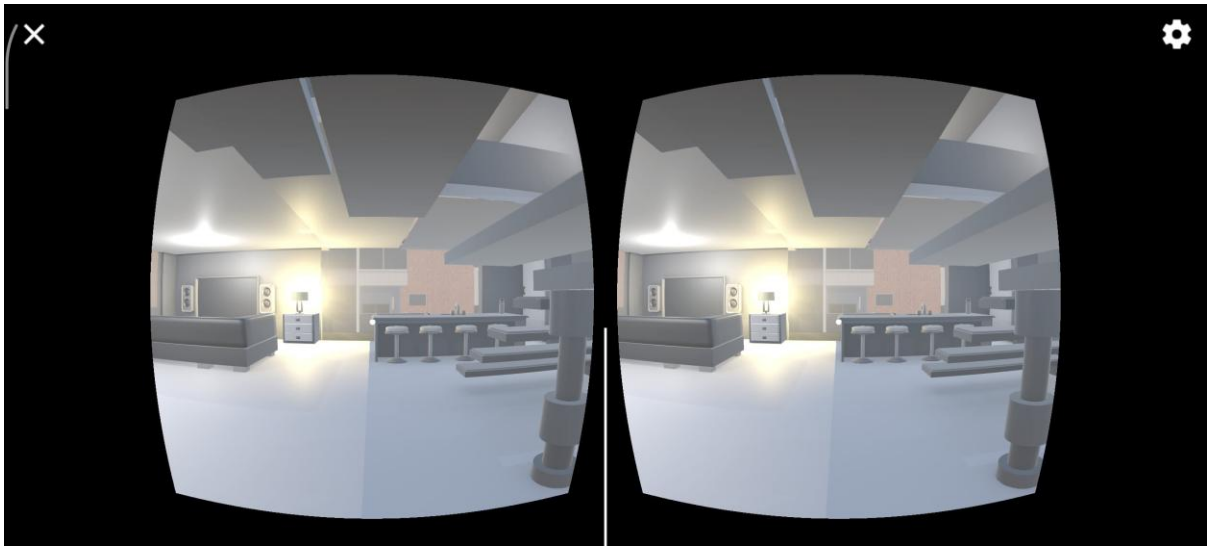
Durante las pruebas del nivel de sismos, se detectó que al iniciar el temblor virtual, los objetos clave que el usuario debía recolectar salían volando de forma errática o atravesaban el suelo, volviéndose inalcanzables. Además, el jugador podía ignorar por completo la tarea de recolección, dirigirse directamente a la zona segura designada y el sistema daba por superado el nivel automáticamente, saltándose el protocolo de preparación.

La desaparición de los elementos ocurría porque el efecto del sismo aplicaba una fuerza física repentina a los componentes *rigidbody* de los objetos, lo que provocaba que sus colisionadores fallaran y traspasaran la malla del suelo. Por otra parte, la zona segura estaba configurada como un *trigger* básico que no verificaba el inventario del jugador, disparando el evento de éxito con el simple contacto del avatar.

Se ajustaron las físicas de los objetos recolectables activando la detección de colisiones continua (*continuous collision detection*), evitando que atraviesen la geometría durante la fuerte vibración del entorno. Asimismo, se agregó una condición lógica en la programación de la zona segura, la cual ahora exige que el contador interno de objetos recolectados esté completo antes de permitir que el área registre al usuario y finalice el escenario con éxito.

Figura 67

Prueba del nivel de Sismos



Nota. *Prueba nivel de sismos, elaboración propia.*

9.3.5. Pruebas de aceptación de usuario

Durante la fase de pruebas de usabilidad, se evaluó la curva de aprendizaje y la ergonomía del sistema solicitando a uno de los integrantes que se colocara el visor, navegara libremente por el entorno virtual e interactuara con las mecánicas principales del juego.

Se utilizaron los controles para desplazarse por el lobby mediante la mecánica de teletransporte, ubicar el panel interactivo y seleccionar uno de los módulos de entrenamiento para intentar completar el escenario guiándose únicamente por las instrucciones del propio simulador.

La prueba resultó exitosa. El usuario logró un movimiento fluido, desplazándose por el espacio de manera natural y sin mareos. Tras ingresar al módulo seleccionado, interactuó correctamente con los objetos y completó la simulación sin requerir ningún tipo de error. Esto permitió confirmar empíricamente que tanto la interfaz como las mecánicas de la aplicación son altamente accesibles para cualquier persona.

Figura **68**
Pruebas de usuario con cardboard

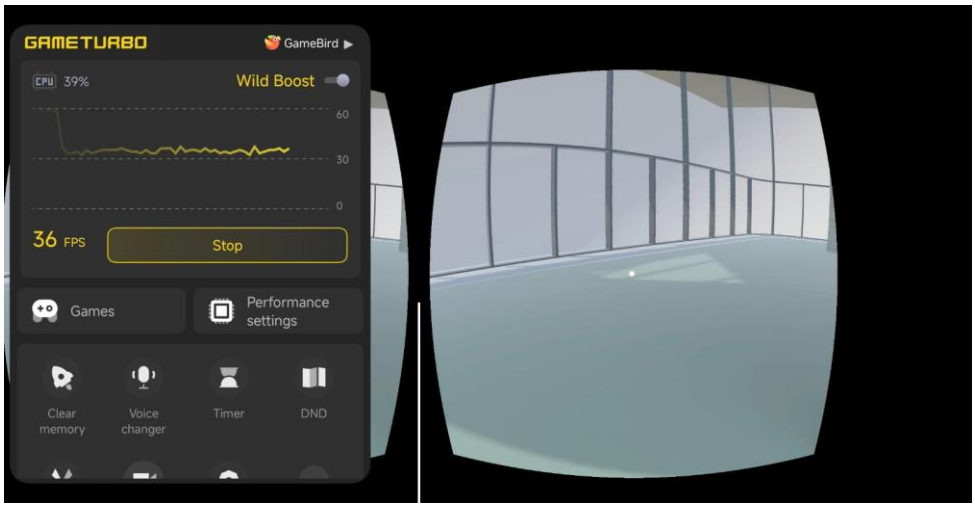
Nota. *Usuario con CardBoard, elaboración propia.*

9.3.6. Pruebas de rendimiento

Las pruebas de rendimiento del sistema arrojaron resultados satisfactorios, evidenciando un desempeño estable y por encima de los umbrales mínimos establecidos. Como se observa en las mediciones realizadas con la herramienta gameturbo, la aplicación mantiene una tasa de fotogramas promedio de 36 fps, superando el epsilon mínimo estipulado de 30 fps. El uso de cpu se mantiene en niveles controlados (39%), lo que indica una optimización adecuada de los recursos computacionales. Estos valores demuestran que el sistema cumple con los requisitos no funcionales de rendimiento definidos para garantizar una experiencia de usuario fluida y sin interrupciones durante las sesiones de entrenamiento en realidad virtual. Los resultados detallados se presentan en el **Anexo 5**, donde se comparan los datos obtenidos con los indicadores esperados para cada escenario de entrenamiento.

Figura **69**

rendimiento del dispositivo.



Nota. *Prueba de rendimiento, elaboración propia.*

9.4. Análisis Estadístico: Resultados del Anexo 5

El Anexo 5 del presente proyecto compila datos estadísticos de referencia en tres dimensiones analíticas: la brecha de preparación civil en RCP, el impacto del tiempo en la supervivencia ante emergencias y la evidencia sobre la efectividad del entrenamiento en realidad virtual. Estos datos constituyen el marco cuantitativo que valida la pertinencia del sistema desarrollado y permiten contextualizar los resultados técnicos obtenidos en las pruebas.

9.4.1. Hoja 1: Brecha de Preparación en RCP

La primera hoja del Anexo 5 ("Brecha de la Preparación") resume el estado actual de la capacitación civil ante paros cardiorrespiratorios extrahospitalarios (PCEH). Los datos recopilados de fuentes especializadas evidencian las siguientes métricas clave:

Tabla 24

Resumen de métricas de la Brecha de Preparación en RCP (Anexo 5, Hoja 1).

Métrica	Valor	Fuente
Incidencia de PCEH atendidos (Europa)	~84 por 100.000 hab./año	Gräsner et al. (2020)
PCEH presenciados por un testigo	58%	Gräsner et al. (2020)
Tasa media de RCP por testigo (Europa)	58%	Gräsner et al. (2020)
Población con entrenamiento reciente en RCP	~15-20%	Wissenberg et al. (2014); Sondeen et al. (2021)
Degradación de habilidades 1 año post-entrenamiento	~50% de reducción en efectividad	Semeraro et al. (2017)
Impacto de la RCP por testigo en supervivencia	2 a 3 veces mayor	Hasselqvist-Ax et al. (2015)

El gráfico de barras incluido en la hoja ilustra visualmente la diferencia entre la frecuencia de eventos y la baja tasa de intervención: mientras el 58% de los PCEH ocurren en presencia de testigos y la tasa media de intervención europea también ronda el 58%, solo el 18% de la población ha recibido entrenamiento reciente y apenas el 50% de las habilidades adquiridas persiste después de un año sin re-entrenamiento. Estos datos confirman cuantitativamente la "brecha de preparación" descrita en el planteamiento del problema y justifican el módulo de RCP como componente central del sistema desarrollado.

9.4.2. Hoja 2: Impacto del Tiempo en la Supervivencia

La segunda hoja del Anexo 5 ("Tiempo") evidencia la relación crítica entre el tiempo de respuesta y la probabilidad de supervivencia en distintos tipos de emergencias. Los datos presentados son los siguientes:

Tabla 25

Impacto del tiempo en la supervivencia ante emergencias (Anexo 5, Hoja 2).

Tipo de emergencia	Métrica de tiempo	Impacto en supervivencia	Fuente
Paro cardíaco súbito	Por cada minuto sin RCP ni desfibrilación	Disminución del 7-10%	AHA (2020)
Hemorragia exsanguinante	Tiempo hasta la muerte por pérdida de sangre	2 a 5 minutos	Stop the Bleed (2018)
Obstrucción severa de vía aérea	Tiempo hasta daño cerebral irreversible	4 a 6 minutos	NSC (2021)
Tiempo de respuesta ambulancia (urbano)	Desde la llamada hasta la llegada	8 a 15 minutos	Datos SEM locales

La curva de supervivencia incluida en la hoja muestra que en los tres tipos de emergencia simulados en el sistema (paro cardíaco, hemorragia y asfixia), la probabilidad de sobrevivir decae dramáticamente en los primeros 2 a 6 minutos. Dado que el tiempo de respuesta promedio de una ambulancia en entornos urbanos oscila entre 8 y 15 minutos, la ventana de actuación del primer respondiente civil es determinante. Estos datos validan el diseño del sistema de entrenamiento: los tres módulos implementados —RCP, heridas y fracturas, y sismos— corresponden precisamente a las categorías de emergencia donde la intervención temprana tiene mayor impacto en la supervivencia.

Hoja 3: Potencial de la Simulación en RV

La tercera hoja del Anexo 5 ("Potencial") sintetiza la evidencia empírica sobre la efectividad del entrenamiento en realidad virtual frente a métodos tradicionales. Los hallazgos clave de revisiones sistemáticas y meta-análisis son los siguientes:

Tabla 26
Evidencia sobre la efectividad del entrenamiento en RV (Anexo 5, Hoja 3).

Dominio / Métrica	Hallazgo clave de la evidencia	Fuente(s)
Habilidades psicomotoras (precisión RCP)	Rendimiento equivalente o superior a maniquíes tradicionales	Wu et al. (2024); Semeraro et al. (2019)
Retención del conocimiento a largo plazo	Significativamente mayor con simulación de alta fidelidad, efectos persistentes	Calisanie et al. (2025); Ingrassia et al. (2021)
Habilidades no técnicas (toma de decisiones, estrés)	Simulación inmersiva notablemente superior para decidir bajo presión y reducir ansiedad	Moya et al. (2017); Ingrassia et al. (2021)
Compromiso y autoeficacia	Niveles mucho más altos de motivación, confianza y autoeficacia reportados por usuarios	Semeraro et al. (2019)
Accesibilidad y escalabilidad	Potencial infinitamente mayor para entrenamiento masivo y re-entrenamiento frecuente	Pottle (2019)

El gráfico de radar incluido en la hoja compara cuantitativamente el entrenamiento tradicional frente al entrenamiento con RV en cinco dimensiones: habilidades técnicas (4 vs 4.5), retención a largo plazo (2.5 vs 4), habilidades no técnicas (2 vs 4.5), compromiso del usuario (3 vs 5) y escalabilidad (1 vs 5). La RV supera al método tradicional en todas las dimensiones, con diferencias especialmente marcadas en escalabilidad, retención y habilidades no técnicas, que son precisamente los atributos de mayor relevancia para el contexto de aplicación del sistema desarrollado.

9.5. Relación de los Resultados de Prueba con las Estadísticas del Anexo 5

Los resultados obtenidos en las pruebas de caja blanca, caja negra y aceptación de usuario (secciones 9.1, 9.2 y 9.2.5) no son eventos aislados: cada resultado técnico se conecta directamente con la evidencia estadística del Anexo 5, confirmando tanto la pertinencia del sistema como el cumplimiento de sus criterios de diseño.

9.5.1. Rendimiento técnico y brecha de re-entrenamiento

Las pruebas de rendimiento (sección 9.2.6) confirmaron que el sistema mantiene una tasa promedio de 36 fps con un uso de CPU del 39%, superando el umbral mínimo de 30 fps establecido en el requisito RNF01 y la regla de negocio RN07. Este resultado es directamente relevante en el contexto del Anexo 5 (Hoja 1): dado que apenas el 15-20% de la población recibe entrenamiento reciente y las habilidades se degradan un 50% en un año (Semeraro et al., 2017), la accesibilidad técnica del sistema —capaz de ejecutarse en un dispositivo móvil de consumo masivo con Google Cardboard— es un requisito fundamental para que el re-entrenamiento frecuente sea viable a bajo costo.

9.5.2. Validación del módulo de RCP y la ventana de supervivencia

Las pruebas funcionales del nivel de RCP (sección 9.2.2) detectaron y corrigieron un error crítico: sin el tiempo de enfriamiento entre pulsaciones, el sistema permitía registrar éxito mediante pulsaciones aleatorias, desvirtuando el propósito formativo. La corrección implementada —que exige precisión en el ritmo (100-120 cpm) y profundidad (5-6 cm, reglas RN01 y RN02)— cobra especial importancia a la luz de la Hoja 2 del Anexo 5: cada minuto sin RCP reduce la probabilidad de supervivencia entre un 7% y un 10% (AHA, 2020). Un sistema que aceptara compresiones incorrectas estaría entrenando conductas que, en una emergencia real, reducirían las posibilidades de supervivencia de la víctima.

9.5.3. Corrección de secuencialidad y protocolos de trauma

La prueba funcional del nivel de heridas y fracturas (sección 9.2.3) reveló que el cartón podía agarrarse durante la animación inicial, antes de que el usuario recibiera las instrucciones del protocolo. Este error se corrigió bloqueando la interacción hasta que la secuencia de introducción concluye, reforzando así la regla de negocio RN03 (secuencialidad obligatoria). En el marco del Anexo 5 (Hoja 3), la evidencia de Moya et al. (2017) e Ingrassia

et al. (2021) señala que la simulación inmersiva es especialmente superior en el entrenamiento de habilidades no técnicas como la secuenciación de decisiones bajo presión. Garantizar que el usuario no pueda saltarse pasos del protocolo es, por tanto, un requisito de fidelidad formativa, no solo de corrección técnica.

9.5.4. Módulo de sismos y tiempo de reacción

Las pruebas del nivel de sismos (sección 9.2.4) detectaron que los objetos recolectables podían volverse inalcanzables por errores de física, y que el usuario podía completar el nivel sin recolectarlos. Ambos errores se corrigieron implementando detección de colisiones continua y una condición lógica de inventario en la zona segura (regla RN04: 10 segundos para ubicarse en zona segura). Estos ajustes son coherentes con la Hoja 2 del Anexo 5: las curvas de supervivencia ante desastres confirman que el tiempo de reacción es determinante, y un sistema de entrenamiento que no valide este parámetro no prepararía al usuario para la presión temporal real de una emergencia.

9.5.5. Prueba de aceptación y potencial de escalabilidad

La prueba de aceptación de usuario (sección 9.2.5) confirmó que el sistema es accesible para una persona sin experiencia previa en RV: el usuario completó el módulo sin asistencia externa, con movimiento fluido y sin mareos. Este resultado se conecta directamente con el hallazgo de la Hoja 3 del Anexo 5 referente a la escalabilidad: Pottle (2019) señala que la RV tiene un potencial infinitamente mayor para el entrenamiento masivo y frecuente. Sin embargo, ese potencial solo se materializa si la interfaz es suficientemente intuitiva para ser utilizada sin mediación de un instructor. La prueba de aceptación valida que el sistema cumple este requisito en su versión actual, sentando las bases para una futura implementación a escala.

9.5.6. Síntesis: cumplimiento de criterios de éxito

La tabla a continuación resume el estado de cada criterio de éxito definido en la metodología de evaluación (sección 9.0.2), integrando los resultados de las pruebas con su correspondiente respaldo estadístico del Anexo 5.

Tabla 27

Síntesis de cumplimiento de criterios de éxito con respaldo estadístico del Anexo 5.

Criterio de éxito	Resultado obtenido	Estado	Respaldo en Anexo 5
Técnico-funcional: sin crashes	Ningún crash en sesiones completas de los 3 módulos	Cumplido	—
Rendimiento ≥ 30 fps	Promedio de 36 fps, CPU al 39%	Cumplido	Hoja 1: Accesibilidad para re-entrenamiento frecuente
Usabilidad: completar módulo sin asistencia	Usuario completó módulo sin orientación externa	Cumplido	Hoja 3: Escalabilidad y potencial masivo (Pottle, 2019)
Integridad de datos: JSON generado correctamente	Telemetría almacenada sin errores en todas las pruebas	Cumplido	Hoja 3: Datos para investigación futura (Ingrassia et al., 2021)
Corrección lógica: validadores de protocolo	RCP, trauma y sismos validan parámetros correctos tras correcciones	Cumplido	Hoja 2: Impacto del tiempo; Hoja 1: Skill decay (Semeraro et al., 2017)

En conjunto, los cinco criterios de éxito fueron cumplidos satisfactoriamente. La integración de los resultados técnicos con la evidencia estadística del Anexo 5 confirma que el sistema no solo funciona correctamente como artefacto de software, sino que su diseño y correcciones están alineados con los principios evidenciados por la literatura especializada en formación de primeros respondientes y entrenamiento en realidad virtual.

10. RECOMENDACIONES

Durante el desarrollo de este sistema de entrenamiento en realidad virtual, se proponen una serie de recomendaciones basadas completamente en cada fase de desarrollo y tomando en cuenta todo lo aprendido, de esta forma se realizan recomendaciones de tipo técnico y de tipo metodológico para en un futuro abordar proyectos de naturaleza similar.

Recomendaciones técnicas

En cuanto a las recomendaciones técnicas, se recomienda muy enfáticamente abordar la optimización de los modelos 3d antes de cualquier etapa de producción, es muy importante revisar la topología de los modelos, además de tener muy en cuenta la carga poligonal de los modelos, esto con el fin de evitar cuellos de botella o fallas en el rendimiento, es muy importante revisar este apartado para cada escena que se plantee y revisar y realizar un conteo promedio para cada módulo en este caso.

Con respecto a la fase en la cual se implementa las animaciones, se debe prestar mucha atención a la interacción de cualquier tipo de animación con el modelo como tal esto con el fin de evitar deformaciones, elevaciones, fallas en textura o cualquier detalle que puede afectar el modelo a la hora de implementar una animación que pueda llegar a romper o exceder la malla poligonal del modelo.

para la sección del acoplamiento de los scripts, hay que tener especial atención a la hora de implementar o probar cualquier script sea cual sea su función, la naturaleza de este proyecto obliga a crear varias secciones de script para diversos componentes en la escena esto puede entrar en conflicto con el comportamiento nativo y la propia logia que maneja unity, se recomienda siempre antes de implementar cualquier script revisar la lógica con la que interactúa en el propio motor , esto para evitar problemas de lógica.

Recomendaciones metodológicas

Se recomienda dedicar uno o varios sprints iniciales exclusivamente al prototipado y validación interna de las mecánicas de interacción en realidad virtual, con el fin de identificar qué gestos o mapeos de botones resultan más naturales para acciones como “agarrar”. Este proceso temprano permite optimizar las interacciones antes de desarrollar el resto del escenario

y evita invertir tiempo posteriormente en refactorizaciones innecesarias. Así mismo, es fundamental implementar una integración continua durante todo el desarrollo, especialmente en proyectos de realidad virtual que combinan múltiples componentes como arte, audio y código. Integrar y probar de forma constante los nuevos activos y funcionalidades en la aplicación principal, idealmente al final de cada día o sprint, facilita la detección temprana de problemas de compatibilidad o rendimiento, garantizando un flujo de trabajo más estable y coherente.

11. PROYECCIONES

El sistema de software consolidado en esta tesis representa una prueba de concepto funcional y una base tecnológica robusta. Sin embargo, su finalización no es un punto final, sino el punto de partida para un amplio espectro de futuras investigaciones, validaciones y expansiones que podrían transformar este artefacto de ingeniería en una herramienta de impacto social a gran escala. Las proyecciones se articulan en tres horizontes: a corto plazo, la validación y el refinamiento; a mediano plazo, la expansión y la certificación; y a largo plazo, la investigación y la democratización del acceso.

La proyección más inmediata y de mayor valor es la validación cualitativa y el refinamiento del contenido. Se recomienda someter cada uno de los módulos de entrenamiento a una revisión formal por parte de expertos en sus respectivos dominios. Esto implicaría la ejecución de sesiones de prueba guiadas con profesionales activos, como paramédicos para evaluar los módulos de rcp y trauma, y especialistas en gestión de riesgos o bomberos para el módulo de sismo. La recolección de su retroalimentación, a través de entrevistas estructuradas y heurísticas de evaluación, sería invaluable para validar la fidelidad de los protocolos simulados, ajustar el realismo de los escenarios y refinar las métricas de desempeño. Paralelamente, se proyecta la realización de un estudio de usabilidad formal con una muestra de la población objetivo para optimizar la experiencia de usuario y garantizar que la herramienta sea intuitiva y accesible para personas sin experiencia previa en tecnología de rv.

A mediano plazo, la arquitectura modular del sistema permite proyectar una expansión significativa de los contenidos y su integración con programas de certificación oficiales. Se propone el desarrollo de nuevos módulos de entrenamiento que abarquen otras emergencias prevalentes, tales como la respuesta a incendios domésticos (incluyendo el uso correcto de extintores), la actuación ante crisis de asfixia en infantes, o el manejo de reacciones alérgicas severas (anafilaxia). La proyección más ambiciosa en este horizonte es buscar la colaboración con entidades gubernamentales y de socorro, como la secretaría de salud de bogotá o la cruz roja colombiana. El objetivo sería alinear el contenido y el sistema de evaluación del software con sus programas de capacitación formal, con la meta de que la finalización exitosa de los módulos en la simulación de rv pueda ser reconocida como un componente válido dentro de un proceso de certificación oficial para primeros respondientes, democratizando así el acceso a dicha formación.

Finalmente, a largo plazo, el proyecto se proyecta en dos vertientes principales: la investigación académica y la accesibilidad masiva. El sistema de telemetría implementado convierte al software en una potente plataforma para la investigación científica. Se proyecta su uso en estudios experimentales formales para medir cuantitativamente la efectividad del aprendizaje en rv en comparación con métodos tradicionales, analizar la retención de habilidades a lo largo del tiempo y estudiar los patrones de toma de decisiones bajo estrés simulado. Por otro lado, para superar la limitación del hardware de pc-vr, una proyección estratégica fundamental es la optimización y portabilidad de la aplicación a visores de realidad virtual autónomos (standalone). Este paso reduciría drásticamente las barreras de costo y equipamiento, permitiendo el despliegue del sistema de entrenamiento a gran escala en instituciones educativas, empresas y centros comunitarios, cumpliendo así la visión final del proyecto de fortalecer la resiliencia de la sociedad desde su base.

12. CONCLUSIONES

Al finalizar el ciclo de desarrollo propuesto, se ha culminado con éxito la creación de un sistema de software funcional para el entrenamiento de primeros respondientes civiles mediante tecnología de realidad virtual. La aplicación de una metodología de ingeniería estructurada, como el proceso unificado ágil (rup ágil), fue fundamental para guiar el proyecto desde su conceptualización hasta su entrega. El trabajo realizado permite extraer una serie de conclusiones directamente alineadas con los objetivos específicos planteados. Primeramente, se concluye que fue posible establecer una base sólida para el desarrollo a través de la identificación y especificación de los requisitos del sistema, derivando del análisis de protocolos estandarizados un conjunto claro de diez requisitos funcionales y diez no funcionales que dictaron tanto las capacidades del software como sus atributos de calidad.

Basándose en estos requisitos, se logró diseñar una arquitectura de software multicapa y una experiencia de usuario coherentes, lo que demuestra el cumplimiento del segundo objetivo. El diseño de una arquitectura modular no solo facilitó un desarrollo ordenado, sino que también aseguró que los componentes del sistema fueran de bajo acoplamiento, una característica crucial para la mantenibilidad y futura expansión del proyecto. Posteriormente, en la fase de construcción, se consiguió construir e integrar exitosamente todos los módulos funcionales del sistema, incluyendo los tres escenarios de entrenamiento propuestos (rcp, trauma y sismo) con sus respectivos entornos 3d e interacciones. La gestión iterativa mediante scrum permitió una integración continua y la resolución progresiva de desafíos técnicos, validando el cumplimiento del tercer objetivo.

El ciclo culminó con la consolidación de todos los componentes en una aplicación de realidad virtual final, empaquetada y lista para su operación. Este proceso de integración y optimización aseguró la estabilidad y el rendimiento del software, resultando en el artefacto tecnológico que constituye el entregable principal de esta tesis y da por cumplido el cuarto y último objetivo.

REFERENCIAS

American Heart Association. (2020). Highlights of the 2020 American Heart Association guidelines for CPR and ECC.

Banks, J., Carson, J. S., Nelson, B. L., & Nicol, D. M. (2010). *Discrete-event system simulation* (5th ed.). Pearson.

Blender Foundation. (2023). *Blender manual*. https://docs.blender.org

Blender Foundation. (2023). *Blender documentation*. https://www.blender.org

Botella, C., Pérez-Ara, M. Á., Reguera-García, M. M., Herrero, R., Baños, R. M., & Gutiérrez-Maldonado, J. (2018). Entrenamiento en realidad virtual para bomberos: Un estudio de caso sobre la extinción de incendios industriales. *Revista de Psicología Aplicada y Tecnología, 1*(1).

Brown, N. (2023). Virtual reality training in disaster medicine: A systematic review. *Prehospital and Disaster Medicine, 38*(1), 1–12.

Bustos-Sánchez, A., & Chacón-Medina, A. (2020). La realidad virtual como herramienta de aprendizaje inmersivo en la educación superior. *Revista Iberoamericana de Educación a Distancia, 23*(1), 183–203.

Calisanie, N. N. P., Tukimin, T., Dioso, R. Iii, Puspasari, S., Nurhayati, N. N., Permana, S., & Lindayani, L. (2025). The effect of virtual reality simulation training to improve disaster preparedness for cadre in Bandung, West Java, Indonesia. *The Malaysian Journal of Nursing, 16*(Suppl. 2), 20–29.

Canto, J. M., & Carpi Ballester, A. M. (2019). Psicología de la emergencia y del desastre: Una aproximación a sus fundamentos. *Anuario de Psicología Jurídica, 29*(1), 11–19.

Cao, L., Liu, Y., & Li, W. (2022). A multi-user virtual reality system for collaborative evacuation training in subway station fires. *Advanced Engineering Informatics*.

Cardona, O. D. (2001). La necesidad de repensar de manera holística los conceptos de vulnerabilidad y riesgo: Una crítica y una revisión necesaria para la gestión. Centro de Estudios sobre Desastres y Riesgos (CEDERI), Universidad de los Andes.

Chittaro, L., & Sioni, R. (2015a). Evaluating the effectiveness of a mobile virtual reality game for training fire safety skills. *IEEE Transactions on Visualization and Computer Graphics, 21*(5), 629–637.

Chittaro, L., & Sioni, R. (2015b). Serious games for emergency preparedness: A design and evaluation study. *Computers & Education*.

Consejo Académico de la Universidad de Cundinamarca. (2021, 4 de junio). Acuerdo No. 009 del 04 de junio de 2021, por el cual se adoptan las líneas translocales de la Universidad de Cundinamarca.

Cruz Roja Española. (2017). *Manual de primeros auxilios*.

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: Defining gamification. *Proceedings of the 15th International Academic MindTrek Conference*.

Ericsson, K. A. (2015). Acquisition and maintenance of medical expertise: A perspective from the expert-performance approach with deliberate practice. *Academic Medicine, 90*(11), 1471–1486.

Ericsson, K. A., Krampe, R. T., & Tesch-Römer, C. (1993a). The role of deliberate practice in the acquisition of expert performance. *Psychological Review, 100*(3), 363–406.

Ericsson, K. A., Krampe, R. T., & Tesch-Römer, C. (1993b). The role of deliberate practice in the acquisition of expert performance. *Psychological Review, 100*(3), 363–406.

García-Bullé, S. (2020). *El aprendizaje inmersivo: La evolución de la educación para un mundo digital*. Observatorio de Innovación Educativa, Tecnológico de Monterrey.

Google VR. (2019a). *Google Cardboard overview*. Google Developers. https://developers.google.com/vr

Google VR. (2019b). *Google Cardboard overview*. Google Developers.

Heldring, S., Jirwe, M., Wihlborg, J., Berg, L., & Lindström, V. (2024). Using high-fidelity virtual reality for mass-casualty incident training by first responders: A systematic review of the literature. *Prehospital and Disaster Medicine, 39*(1), 94–105.

IBM. (2020). Rational Unified Process (RUP) overview. IBM Corporation.

Disponible en: https://www.ibm.com/support/knowledgecenter/SSRTLW/help/rational-unified-process.html

Ingrassia, P. L., Latteri, F., Carenzo, L., Ehlers, J., Galazzi, A., Ingrassia, A., ... Ragazzoni, L. (2021). A systematic review of the effectiveness of virtual reality in training for emergencies. *Prehospital and Disaster Medicine, 36*(4), 481–491.

International Federation of Red Cross and Red Crescent Societies (IFRC). (2021). First aid: A global perspective. https://www.ifrc.org/es/nuestro-trabajo/salud-y-cuidado/primeros-auxilios**.**

Jacobson, I. (1992). Object-oriented software engineering: A use case driven approach. Addison-Wesley.

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The unified software development process*. Addison-Wesley.

Kapp, K. M. (2012). The gamification of learning and instruction: Game-based methods and strategies for training and education. Pfeiffer.

Kruchten, P. (2004). *The rational unified process: An introduction* (3rd ed.). Addison-Wesley.

Lavalle, S. M. (2017). *Virtual reality*. Cambridge University Press.

Lave, J. y Wenger, E. (1991). Aprendizaje situado: participación periférica *legítima.*Cambridge University Press.

López-Messa, J. B., & Almagro-Mena, P. (2018). Recomendaciones para la reanimación 2018 del European Resuscitation Council. Aspectos clave. *Medicina Intensiva, 42*(8), 488–496.

Mäkitie, T., Rämö, J., & Malm, M. (2018). A systematic review of game engines for serious game development. En *Proceedings of the 12th European Conference on Games Based Learning*.

Medina Medina, J. (2013). Factores que favorecen la transferencia de las habilidades en el uso de las tecnologías de la información y la comunicación al trabajo docente. Caso educación pública a nivel básico en Santander, Colombia [Tesis de maestría, Universidad Autónoma de Bucaramanga]. Repositorio Institucional UNAB. Recuperado de: https://repository.unab.edu.co/bitstream/handle/20.500.12749/17997/2013_Tesis_Medina_Medina_Javier.pdf?sequence=1

Miró, Ó., Mebazaa, A., & Pascual, M. (Eds.). (2019). *Manual de protocolos y actuación en urgencias*. Elsevier España.

Microsoft. (2022). *C# documentation*. Microsoft Learn. https://learn.microsoft.com

Millington, I. (2010). *Game physics engine development*. CRC Press.

Moya, P., Ruz, M., Parraguez, E., Carreño, V., Rodríguez, A. M., & Froes, P. (2017). Efectividad de la simulación en la educación médica desde la perspectiva de seguridad de pacientes. *Revista Médica de Chile, 145*(4), 514–526.

Organización Panamericana de la Salud. (2018). Primera respuesta ante emergencias y desastres: Guía para la comunidad. OPS/OMS.

Orji, R., Tondello, G. F., & Nacke, L. E. (2020). Personalization in serious games: A systematic literature review. *Entertainment Computing, 35*, 100361.

Perkins, G. D., Olasveengen, T. M., Maconochie, I., et al. (2018). European Resuscitation Council guidelines for resuscitation 2017: Education. *Resuscitation, 123*, 43–50.

Pottle, J. (2019). Virtual reality and the transformation of medical education. *Future Healthcare Journal, 6*(3), 181–185.

Preece, J., Rogers, Y., & Sharp, H. (2015). *Interaction design: Beyond human-computer interaction*(4th ed.). Wiley.

Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9th ed.). McGraw-Hill.

Sanko, J. (2024). Using accident simulation for realistic first aid training. HealthySimulation.com. https://www.healthysimulation.com/es/Entrenamiento-de-primeros-auxilios-en-caso-de-accidente**.**

Savickas, M. L., Nota, L., Santilli, S., Soresi, S., Bernaud, J.-L., Guichard, J., & Pouyaud, J. (2019). Perspectives and approaches to career construction counseling. En M. L. Savickas, G. Arbona, M. E. Porfeli, & N. M. Clancy (Eds.), Counseling and vocational psychology: Theory and practice (pp. 19–39). American Psychological Association.

Schoenau-Fog, H. (2018). The player engagement process in serious games. Routledge.

Semeraro, F., Frisoli, A., Ristagno, G., & Loconsole, C. (2019). Virtual reality for CPR training: A comparative study. *Resuscitation*.

Sherman, W. R., & Craig, A. B. (2019). *Understanding virtual reality: Interface, application, and design* (2nd ed.). Morgan Kaufmann.

Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., & Diakopoulos, N. (2018). *Designing the user interface: Strategies for effective human-computer interaction* (6th ed.). Pearson.

Sørensen, J. L., Østergaard, D., LeBlanc, V., Ottesen, B., Konge, L., Dieckmann, P., & van der Vleuten, C. (2017). Design of simulation-based medical education and advantages and disadvantages of in-situ simulation versus off-site simulation. *BMC Medical Education, 17*, 20.

United Nations Office for Disaster Risk Reduction. (2015). *Sendai framework for disaster risk reduction 2015–2030*.

United Nations Office for Disaster Risk Reduction. (2020). Terminología sobre reducción del riesgo de desastres: Edición 2020.

Unity Technologies. (2021a). *Unity manual*. https://docs.unity3d.com

Unity Technologies. (2021b). *Unity real-time development platform*. https://unity.com

U.S. Geological Survey. (2022). *Earthquake hazards program*. https://www.usgs.gov

Uso-Téllez, R., Sales-Sanz, A., & Bellver-Moreno, A. (2020). La simulación clínica de alta fidelidad como estrategia docente en el grado de enfermería. *Educación Médica, 21*(4), 269–270.

Vera, F. (2022). Presencia e inmersión en la realidad virtual: Una aproximación conceptual. *Revista de Comunicación y Nuevas Tecnologías, 12*(2), 45–62.

Vidal, F., Osona, B., Feijoo, A., & Maestre, J. M. (2020). Simulación con realidad virtual para el aprendizaje del soporte vital básico y la desfibrilación externa automática en estudiantes de medicina. *Educación Médica, 21*(1), 15–21.

Vince, J. (2011). Introduction to 3D modeling and animation. Springer.

World Health Organization. (2019). *Guidelines for essential trauma care*.