Universidad de las Ciencias Informáticas

FACULTAD 4

DEPARTAMENTO DE INFORMÁTICA

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Título: Sistema de Simulación de Penales de Futbol

Autor: Sergio Daniel Ortiz Jova

Tutor: Ing. Maydalis Hernández Pérez

Ing. Luis Ángel Llull Cespedez

Ing. Kevin Jorge Montes Lorenzo

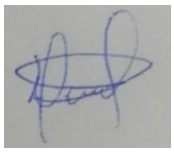
La Habana, junio 2022

DECLARACIÓN DE AUTORÍA

Declaro ser autor del presente trabajo de diploma y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los 16 días del mes de noviembre del año 2022.

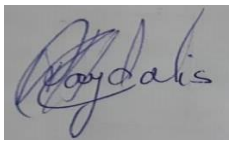
Sergio Daniel Ortiz Jova



_____

Firma del Autor

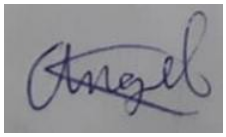
Ing. Maydalis Hernández Pérez



_____

Firma de la Tutora

Ing. Luis Ángel Llull Cespedez



_____

Firma del Tutor

Datos de Contacto

Tutor:

Ing. Maydalis Hernández Pérez

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo electrónico: mhernandezp@uci.cu

Tutor:

Ing. Luis Ángel Llull Cespedez

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo electrónico: lallull@uci.cu

Agradecimientos

Quisiera por sobre todas las cosas darle las gracias a mi familia que me ha apoyado tanto, a mi madre que estuvo todo el tiempo junto a mí y siempre me ofreció su apoyo, al igual que mi padre que siempre ha sido un ejemplo a seguir, al igual que mi pequeña hermana, que espero ayudarla en todo lo que necesite, a mis tíos, mis primos y a mis queridos abuelos que siempre los llevaré en mi corazón, ya que fueron una parte fundamental de mi niñez y mi educación.

También quisiera agradecer a mis profesores que me han dado la enseñanza necesaria para llegar donde estoy hoy en día, a mis tutores que me han ayudado realmente bastante en mi trabajo de diploma.

A mis compañeros de la universidad que, de una forma u otra me han ayudado.

Realmente se los agradezco mucho, siempre serán un tesoro para mí.

Dedicatoria

No hay cosa que me preocupe más que mi familia por eso a ella le dedico cada centímetro de mí. A mi padre Irving, mi madre Zahily, mi hermana Juliana y a mis abuelos Sergio, Julián, Teresa, Neida, Pititi y Pedro.

Resumen

En la Universidad de las Ciencias Informáticas existe el centro de desarrollo de software CTI que se encarga de desarrollar productos y servicios informáticos asociados a Entornos Interactivos 3D. En dicho centro se realizó este trabajo el cual tuvo como objetivo el desarrollo de un sistema de simulación de penales de futbol que permitió aumentar la productividad de los jugadores. El desarrollo del mismo fue guiado por la metodología de desarrollo de software XP. Para la implementación de este se utilizó el motor de videojuego Unity 3D y como lenguaje de desarrollo C#. Para validar que la solución cumpliera con los requisitos definidos por el cliente, se aplicaron pruebas unitarias y pruebas de aceptación. Lo cual demostró que se cumplió satisfactoriamente el objetivo del trabajo.

Palabras claves: videojuego, simulación, Unity 3D, futbol.

Abstract

At the University of the Information-Technology Sciences exists the center of development of software CTI that takes upon itself to develop products and information-technology correlated services Interactive Environment 3D. Which accomplished this work itself in the aforementioned center the development of a system of simulation that it enabled of prisons of soccer aimed at the players productivity increasing. The development of XP was the same guided for the methodology of development of software. Unity utilized the motor of video game himself for the implementation of this 3D and like language of development C #. In order to validate that the solution abide by the requirements defined by the customer, unitary proofs and proofs of approval were applicable. Which proved that the objective of work came true satisfactorily.

Keywords: Videogame, simulation, Unity 3D, soccer.

Índice

Índice de tablas

Introducción

En el mundo actual existen muchas industrias para el ocio como son los videojuegos, shows, documentales, animados. Para su desarrollo se utilizan un sinnúmero de tecnologías para llamar más al espectador o usuario y comerciar más sus productos. De estas tecnologías algunas son muy conocidas como son la inteligencia artificial, el streaming, la realidad aumentada y el almacenamiento en la nube (1) que son actualmente muy utilizadas en estos campos, sobre todo en la de los videojuegos.

Estos son aplicaciones interactivas orientadas al entretenimiento que, a través de ciertos mandos o controles, permiten simular experiencias en la pantalla de un televisor, una computadora u otro dispositivo electrónico. Estos se diferencian de otras formas de entretenimiento, como las películas, en que deben ser interactivos; en otras palabras, el usuario debe involucrarse activamente con el contenido (2).

En la actualidad existen diferentes tipos de videojuegos como: de acción, aventura, arcade, deportivo, estrategia, simulación, musicales, entre otros (3). Los cuales en muchas ocasiones según la concepción para el cual fue creado puede utilizarse para aprender medicina, educación, deportes, música. Por ejemplo, entre los más utilizados para este fin se encuentran: Project Hospital, Surgeon Simulator, La torre del conocimiento, Discover Babylon, FIFA, NBA, Tetris Effect, Guitar Hero (4).

Este tipo de entretenimiento es muy demandado en estos tiempos, cerca del 40% de la población juega videojuegos (5), ellos tributan a la felicidad, mejora el desarrollo físico, estimula la creatividad, mejora las relaciones sociales, disminuye el nivel de estrés, mejora el desempeño en diferentes actividades (6).

En Cuba donde uno de los pilares fundamentales es la informatización no solo de los sectores más importantes como la educación, medicina y comercio, sino que también se realizan proyectos relacionados con el entretenimiento y específicamente lo de los videojuegos.

En el país hay varias instituciones que se dedican al desarrollo de videojuegos entre los que se encuentra la Universidad de las Ciencias Informáticas (UCI), dentro ella existe un centro llamado centro de tecnologías interactivas (CTI) que es el encargado de desarrollar productos y servicios informáticos asociados a Entornos Interactivos 3D (7) y una de sus ramas son los videojuegos. Varios son los productos que se desarrollan en este centro, como son: La neurona 1 y 2, Villa Tesoro, Caos Numérico, Kuba Kart, Coliseum, Super Claria, Aventuras en la Manigua, entre otros (8).

INTRODUCCIÓN

En el mundo existen sistemas que se utilizan para simular penales de futbol, pero su desarrollo y utilización son muy altos, ya que, estos dependen de pantallas grandes, sensores y cámaras, equipos que por lo costos que son, no pueden ser adquiridos ni estar al alcance de todo aquel que lo necesite, por lo que este centro se propuso como meta la concepción de un juego de futbol que no solo permita el entretenimiento de los jugadores que lo utilicen, sino que sirva para los equipos profesionales del país a aumentar su productividad. Específicamente en el juego se necesita un mecanismo que ayude a perfeccionar la manera de pensar de los jugadores cuando se encuentran en los penales de futbol, de manera que se cumpla con el objetivo fundamental que es Meter un Gol. Pues actualmente, los equipos de futbol, para realizar sus prácticas, cuenta con grabaciones de partidos, donde estudian el cómo tirar un penal, provocando esto que sea un poco engorroso a la hora de estudiar el cómo se va a ejecutar dicho proceso y que se dificulte en ocasiones la toma de decisiones porque no se puede simular la jugada en tiempo real. Estos videos también traen como consecuencia que, tanto los entrenadores como los mismos jugadores se agoten más de lo necesario, ya que estos tienen que estar grabando cada video y, además, que ellos se enfoquen en lo que hacen en el video llevándolos hacia la memorización

Por lo anteriormente planteado se define como **problema de investigación** ¿Cómo crear un sistema que ayude a aumentar la productividad de los jugadores en los penales de futbol? El **objetivo** de esta investigación es desarrollar un sistema de simulación de penales de futbol que permita aumentar la productividad de los jugadores en los penales de fútbol. El **campo de acción** sería la simulación de penales de futbol en un entorno interactivo 3D.

El **objeto de estudio** está vinculado a las técnicas para la simulación de penales de futbol.

Para dar cumplimiento al objetivo planteado es necesario realizar las siguientes **tareas de investigación**:

- Elaboración del marco teórico de la investigación mediante el estudio de sistemas informáticos que permitan la simulación de penales
- Definición de la metodología de desarrollo de software, herramientas y tecnologías a utilizar para el desarrollo del sistema.
- Planificación del sistema teniendo en cuenta lo establecido en la metodología de desarrollo de software seleccionada.
- Estudio de todo lo relacionado con la simulación de futbol desde sus inicios hasta la actualidad.

- Diseño del sistema según la metodología de desarrollo de software seleccionada.
- Implementación de un sistema que permita la simulación de penales de futbol.
- Ejecución de pruebas al sistema para garantizar su correcto funcionamiento.
- Validación de la propuesta de solución.

Para la realización de las tareas expuestas anteriormente se emplearon los siguientes **métodos de investigación:**

Los **métodos teóricos** constituyen el enfoque general para abordar los problemas científicos, permitiendo profundizar en las regularidades esenciales de los fenómenos.

Permiten revelar las relaciones esenciales del objeto de investigación no observables directamente, cumpliendo así una función gnoseológica importante al posibilitar la interpretación conceptual de los datos empíricos encontrados, la construcción y desarrollo de teorías (9).

El método teórico puede tener (o puede ocurrir una unión entre ellos): Análisis, síntesis, inducción, deducción, abstracción, concreción, histórico, lógico y modelación. Para el desarrollo de la investigación se emplearon las siguientes:

- Histórico-Lógico: permitió realizar un estudio de los principales conceptos asociados a la simulación de futbol desde su origen hasta la actualidad, viendo la evolución del mismo (10).
- Analítico-Sintético: permitió durante el proceso investigativo el estudio, análisis e identificación de los conceptos relacionados con la temática abordada. Facilitó la realización de un análisis de las herramientas similares que responden al campo de acción definiéndose un conjunto de indicadores para realizar una comparación entre las herramientas, con el objetivo de identificar características que tributen al desarrollo de la propuesta.
- Modelado: permitió crear abstracciones con el objetivo de explicar la realidad. Es empleado como herramienta para la comprensión del problema a resolver y para la creación de los modelos que permitan el diseño de la solución.

Los **métodos empíricos** estudian los fenómenos, objetos y procesos observables, confirmados a través de la hipótesis y la teoría, en otras palabras, obtiene el conocimiento a partir de la observación de la realidad, está basado en la experiencia. Dependiendo del objetivo de la investigación, puede haber diferentes pasos para aplicar el método empírico.

El método empírico puede tener: Observación, experimento, consulta a expertos, historia de vida, estudios de caso, medición y encuesta. De ellos en el sistema se aplicarán los siguientes:

- Entrevista: se entrevistó a jugadores de futbol con el fin de encontrar la información necesaria relacionado con los problemas a la hora del disparo a puerta, también se entrevistó a especialistas del CTI para obtener información de las herramientas empleadas en la institución con el fin de utilizarlas en la simulación. El tipo de entrevista que se va a aplicar es semiestructurado.
- Observación: posibilitó estudiar y observar el funcionamiento de sistemas similares, identificándose funcionalidades y características que se tuvieron en cuenta para el desarrollo de la solución (11).
- Encuesta: se empleó para validar el resultado con el empleo de la técnica de Iadov, que constituye una vía indirecta para el estudio de la satisfacción, debido a que los criterios que se emplean tienen su fundamentación en las relaciones de tres preguntas a través del cuadro lógico de Iadov, intercaladas dentro de un cuestionario y que son desconocidas para el sujeto.
- Revisión documental: permitió la obtención de información relacionada con la temática abordada mediante el asesoramiento a través de diferentes artículos y libros.

El presente trabajo cuenta con la siguiente **estructura capitular**:

Capítulo 1: **Fundamentación Teórica**:

En este capítulo se plasmarán conceptos asociados al dominio del problema, además se harán estudios de los sistemas homólogos con el objetivo de encontrar características similares.

Se caracterizará el sistema de simulación sobre la base de los antecedentes, conjuntamente se definirá el marco teórico utilizado, de igual modo se expondrá las tendencias del sistema de simulación en el plano histórico y adicionalmente se diagnosticará la situación actual.

Se realizará una breve reseña y se valorará las soluciones existentes del sistema y su integración a los videojuegos y se explicarán las metodologías, herramientas y lenguajes que serán utilizadas en la construcción del sistema de simulación.

Capítulo 2: **Planificación y diseño de la solución propuesta al problema científico**: Se realiza una descripción de la propuesta de solución y se exponen los requisitos funcionales y no funcionales identificados. Teniendo en cuenta la metodología de

desarrollo de software seleccionada se realizará la planificación y el diseño de la propuesta de solución, y se presentan los artefactos ingenieriles que documentan el proceso de desarrollo del software.

Capítulo 3: **Implementación, prueba y validación de la propuesta**:

En este capítulo se presentan los elementos relacionados con la implementación del sistema. Se define la estrategia de pruebas a realizar para verificar el cumplimiento de los requisitos de software y se presenta el resultado del proceso de prueba y de la validación del sistema.

Capítulo 1: Fundamentación Teórica

La base teórica del presente trabajo se encuentra sustentada a la descripción de los principales conceptos relacionados con la problemática, al estudio realizado a sistemas similares para poder determinar las características esenciales de las mismas. Se describen las herramientas, tecnologías y metodología que se ajustan al desarrollo del sistema de simulación de penales.

1.1 Conceptos asociados al dominio del problema

Los conceptos asociados al dominio del problema son importantes cuando se quiere comprender como funciona un sistema de simulación, para ello se tuvo en cuenta varios conceptos como son: la simulación, videojuego, diseño y scripts.

Simulación

Es el artificio contextual que hace referencia a la investigación de una hipótesis o un conjunto de hipótesis de trabajo utilizando modelos para la enseñanza y el aprendizaje. Thomas T. Goldsmith Jr. y Estle Ray Mann la definen como: “Simulación es una técnica numérica para conducir experimentos en una computadora digital. Estos experimentos comprenden ciertos tipos de relaciones matemáticas y lógicas, las cuales son necesarias para describir el comportamiento y la estructura de sistemas complejos del mundo real a través de largos períodos (12)”.

A decir de Robert E. Shannon, es: "La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias dentro de los límites impuestos por un cierto criterio o un conjunto de ellos para el funcionamiento del sistema" (13).

La simulación 3D es un proceso muy utilizado para recrear elementos de la realidad, con la finalidad de establecer las correcciones necesarias, siendo este, uno de los objetivos a perseguir para poder desarrollar el videojuego lo más parecido a la realidad.

Simulación digital

Es una técnica que permite imitar o simular en un ordenador el comportamiento de un sistema real o hipotético según ciertas condiciones particulares de operación. (14)

Los modelos de simulación se clasifican en: estático/ dinámico, determinístico/ estocástico, discreto/ continuo y físico/ analógico/ simbólico. (15)

Los modelos deben cumplir con las siguientes características:

- Un modelo es un objeto que se utiliza para representar cualquier otra entidad compleja mediante un proceso de abstracción en un formato adecuado de acuerdo con las características de interés de un objeto real o hipotético.
- Un modelo es una representación simplificada de un sistema que se empleará para facilitar, comprender, cambiar, prever y posiblemente controlar el comportamiento mismo.
- Puede ser substituto de un sistema físico concreto.
- Debe representar el conocimiento que se tiene de un sistema de modo que facilite su interpretación.
- Debe ser tan sencillo como sea posible siempre y cuando represente los aspectos de interés. (14)

Casi siempre cuando se hace uso de esta técnica se emplean otros equipos como son pantallas gigantes, múltiples televisores, entre otras, que tienen un alto precio en el mercado; aunque también se pueden basar en programas informáticos como son aplicaciones y videojuegos.

Videojuegos

También conocido como juego de video, es un juego electrónico en el que uno o más personas interactúan por medio de un controlador, con un dispositivo que muestra imágenes de video (14). Este puede estar en una única o ser hecha para diferentes plataformas como son: ordenador, maquina arcade, videoconsola o un dispositivo móvil como teléfono celular o tabletas.

Para realizar la simulación de un videojuego se deben seguir los siguientes pasos:

1. Trasladar al papel todos los elementos del simulador.
2. Determinar los objetivos.
3. Determinar el público objetivo.
4. Pensar en qué consola/ dispositivo se va a hacer.
5. Pensar en el género del videojuego.
6. Diseñar el mundo del videojuego.
7. Diseñar las mecánicas.
8. Diseñar los niveles.
9. Diseñar a los personajes
10. Diseñar el contenido. (17)

Los videojuegos simulados deben poseer la mayoría de las características existentes en el mundo real, como son las físicas y demás particularidades que inciden sobre los objetos simulados.

Para esto los desarrolladores de videojuegos han buscado que los usuarios se comprometan más con estos, utilizando NPC (Non Playable Character o personaje no jugador en español). Luego de que se obtiene un entendimiento de lo que se va a hacer en el videojuego, este pasa por una etapa de diseño.

Diseño

Es el proceso previo de configuración mental o concepción de la idea del videojuego, en la búsqueda de una solución en cualquier campo. Esta involucra a varias dimensiones que van más allá del aspecto, la forma y el color, abarcando también la función de un objeto y su interacción con el usuario.

Durante el proceso se debe tener en cuenta además de la funcionalidad, la operatividad, la eficiencia y la vida útil del objeto de diseño (15). En este caso es necesario tener en cuenta un diseño que ayude a simular como serán los personajes, campos, escenarios, etcétera. Luego de la etapa de diseño, el videojuego pasa por el estudio de las físicas de videojuegos a aplicar.

Sistemas homólogos

A continuación, se hará un estudio de los sistemas homólogos a nivel internacional relacionadas con el futbol, desde videojuegos hasta aplicaciones relacionadas con el diseño de ejercicios y aplicaciones para porteros, para ello se definirán criterios de comparación y se arribarán a las conclusiones.

eFootball

Anteriormente conocido como Pro Evolution Soccer abreviado como PES y en un momento denominado eFootball Pro Evolution Soccer, abreviado como eFootball PES y en Japón conocido como Winning Eleven es una saga videojuego de futbol desarrollado en Japón por la empresa Konami y cuenta con plataforma para Windows, Xbox, PlayStation, entre otras. En ella se encuentran distintos modos de juego como son: amistoso, entrenamiento, penales, liga, copa, liga máster; para jugarlo puede existir un jugador, múltiples jugadores o en línea (17).

FIFA

Es una saga de videojuegos de futbol publicados anualmente por Electronics Arts, bajo el sello EA Sports creado en Japón con una amplia gama de plataformas como son Windows, Xbox, PlayStation, Nintendo, SEGA, Game Boy, Android, entre otras; puede ser jugado por un jugador o múltiples jugadores, como modo de juego se tiene: amistoso, entrenamiento, penales, liga, supercopa, Bundesliga, etcétera. Su primer lanzamiento fue en 1993 y ha cambiado de disimiles manera, comenzando con su

perspectiva isométrica hasta la actualidad con la implementación de inteligencia artificial a los jugadores. Uno de los aspectos importantes era el sonido del público del estadio, este estaba basado en grabaciones en vivo y le aportaban un mayor realismo al consumidor (18).

Mini Football

Desarrollado por Miniclip.com es un videojuego de futbol con un total de 5 jugadores en el campo donde no existe el fuera de juego, el partido tiene una duración de 45 minutos (dos partes de 20 minutos más un descanso de 5 minutos, si llegan a quedar empatados se irán a penales) con plataforma Android y IOS (19). Cuenta con un sistema de control intuitivo compuesto por tres botones básicos, uno para correr, otro para pasar el balón y el último para ejecutar disparos certeros, además de contar con un joystick para mover a los jugadores. Como modos de juego se encuentran: amistoso, liga y penales. Puede jugarse single player o multiplayer (20).

Penalti virtual

Desarrollado por GTA es un simulador de futbol que utiliza un sensor de movimiento con el cual se recogen todos los datos referentes al movimiento del pie, para el montaje técnico se necesita de una pantalla gigante LED, un equipo de sonido, un espacio nivelado de 8x7 metros para la instalación y sensor de movimiento. El porcentaje de acierto de los lanzamientos a portería se pueden configurar. (24)

Luego del estudio de los sistemas homólogos se tiene la siguiente tabla donde se comparan los sistemas homólogos según el mercado, la plataforma, si cuentan con sistema de penales, su código fuente, documentación de la tecnología y las estadísticas (conteo de goles por la derecha, conteo de goles por la izquierda, conteo de goles por el medio y los fallos):

Tabla 1 Comparación de sistemas comparativos. Elaboración propia.

Video-juego	Mercado	Plataforma	Sistema de penales	Código Fuente	Documentación de la tecnología	Estadísticas
eFootball	Privativo	Windows	Si	Cerrado	Página oficial	No
FIFA	Privativo	Windows	Si	Cerrado	Página oficial	No
Mini Football	Privativo	Android, IOS	No	Cerrado	No presenta página oficial	No

Penalti virtual	Privativo	Windows	Si	Cerrado	Página oficial	No

Después del estudio de estas herramientas se determinó que ninguna de estas soluciones responde al objetivo planteado de la investigación, ya que; son de código cerrado y de licencia privativa. Sin embargo, tienen características y requisitos funcionales que se pueden ajustar al desarrollo de la solución. Por ejemplo, los sistemas eFootball, FIFA y Penalti virtual, tienen implementado sistemas de penales, en donde el usuario puede lanzar la pelota a la dirección que desee ejerciendo una fuerza en el balón. Además, entre las características que prevalecen están las relacionadas con el diseño del sistema informático del simulador. Del mismo, se tendrán presente las animaciones y el diseño de los objetos (pelota, portería, jugador y el estadio).

Por otro lado, se estudiaron otros simuladores de penales de futbol que para su utilización es necesario de una infraestructura (hardware) que es muy costosa como es el caso de Penalti virtual. Por lo que; no constituye una solución viable para el cumplimento del objetivo. Su estudio estuvo vinculado al sistema informático para el diseño de sistema que se desarrollará. También, hay que enfatizar que ninguno de estos sistemas, tienen implementado estadísticas que permitan a un equipo de futbol o a una persona mejorar su rendimiento en el campo.

Por último, en una búsqueda realizada a nivel Nacional no se encontró ningún simulador de penales o videojuegos de futbol desarrollados, lo que evidencia la novedad de esta investigación.

1.2 Metodología de desarrollo de software, lenguajes y herramientas para el modelado

En la presente tarea se realizó un estudio para determinar la metodología de desarrollo de software a utilizar, sus roles, características de la metodología a emplear, lenguajes de programación y modelado y IDE’s de desarrollo.

A decir de Pressman, “Una metodología es el conjunto ordenado de pasos a seguir para cumplir un objetivo. Dicho objetivo, en la ingeniería de software, es el desarrollo de software de alta calidad que cumple con las necesidades del cliente dentro de un plan y un presupuesto predecible. Para esto es necesario proveer un enfoque disciplinario para asignar tareas y responsabilidades dentro del desarrollo del sistema, determinar un camino metódico y sistemático para desarrollar, diseñar y validar una arquitectura y

reducir en gran medida los riesgos que representa la construcción de sistemas de software".

La metodología de desarrollo de software se divide en 2 grupos bien definido: metodología ágil y tradicional. A continuación, se darán a conocer las diferencias entre estas metodologías.

Tabla 2 Comparación de metodología

Aspectos	Metodología ágil	Metodología tradicional
Enfoque	Adaptación	Predictivo
Éxito de medición	Valor del negocio	Conformación de planificar
Tamaño del proyecto	Pequeño	Grande
Estilo de gestión	Descentralizada	Autocrático
Perspectiva para el cambio	Cambio y Adaptabilidad	Cambio y Sostenibilidad
Documentación	Bajo	Pesado
Ciclos	Muchos	Limitados
Tamaño del equipo	Pequeño	Grande

Por las características vistas anteriormente y con el objetivo del trabajo se puede valorar que se escogerá una metodología ágil, entre las existentes se encuentran: Scrum, Programación extrema o eXtreme Programming (XP), Kanban, Lean. De ellas el CTI trabaja con la metodología XP

La metodología XP potencia las relaciones interpersonales como clave para el éxito en el desarrollo de software, esto promueve el trabajo en equipo preocupándose por la experiencia de los desarrolladores y propiciando un buen ambiente creados por los mismos.

Esta metodología va a ser la utilizada en la aplicación y se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, por lo que esto logra un rápido desenvolvimiento de los requisitos para el cumplimento del trabajo, se logra una comunicación fluida entre todos los roles presentes en el mismo, logra la simplicidad en las soluciones implementadas y coraje para enfrentar los cambios (25).

XP es adecuada para proyectos los cuales tienen requisitos imprecisos y con muchos cambios, donde pueden existir altos riesgos técnicos.

Cuenta con varias características como son:

- Metodología basada en prueba y error para obtener un software que funcione realmente.
- Fundamentada en principios.
- Está orientada hacia quien produce y usa software (el cliente participa muy activamente).
- Reduce el coste del cambio en todas las etapas del ciclo de vida del sistema.
- Combina las que han demostrado ser las mejores prácticas para desarrollar software, y las lleva al extremo.
- Cliente bien definido.
- Los requisitos pueden cambiar.
- Grupo pequeño y muy integrado (2-12 personas).
- Equipo con formación elevada y capacidad de aprender.

El ciclo de vida ideal de XP consiste cuatro etapas: Planeación, Diseño, Codificación y Pruebas.

PLANEACIÓN

La Metodología XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores. El proyecto comienza recopilando las historias de usuarios, las que constituyen a los tradicionales casos de uso. Una vez obtenidas estas historias de usuarios, los programadores evalúan rápidamente el tiempo de desarrollo de cada una.

Los conceptos básicos de la planificación son:

- **Las Historias de Usuarios**, las cuales son descritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar.
- **El Plan de Entregas (Release Plan)**, establece que las historias de usuarios serán agrupadas para conformar una entrega y el orden de las mismas. Este cronograma será el resultado de una reunión entre todos los actores del proyecto (cliente, desarrolladores, gerentes, etc.). Típicamente el cliente ordenará y agrupará según sus prioridades las historias de usuario. El cronograma de entregas se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores.

- **Plan de Iteraciones (Iteration Plan)**, las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación. Asimismo, para cada historia de usuario se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores. Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir
- **Reuniones Diarias de Seguimiento (Stand – Up Meeting)**, el objetivo es mantener la comunicación entre el equipo y compartir problemas y soluciones

DISEÑO

La Metodología XP hace especial énfasis en los diseños simples y claros. Como puntos clave se encuentran:

- **Simplicidad**, Un diseño simple se implementa más rápidamente que uno complejo. Por ello XP propone implementar el diseño más simple posible que funcione.
- **Soluciones "Spike"**, Cuando aparecen problemas técnicos, o cuando es difícil de estimar el tiempo para implementar una historia de usuario, pueden utilizarse pequeños programas de prueba (llamados "Spike"), para explorar diferentes soluciones. "Architectural spikes" son iteraciones utilizadas para probar aproximaciones tecnológicas, y "spikes" son periodos de trabajo utilizados para reducir riesgos. Los "spikes" se introducen en los procesos de planificación de entrega.
- **Recodificación ("Refactoring")**, Consiste en escribir nuevamente parte del código de un programa, sin cambiar su funcionalidad, a los efectos de crearlo más simple, conciso y entendible. Las metodologías de XP sugieren re codificar cada vez que sea necesario.
- **Metáforas**, XP sugiere utilizar este concepto como una manera sencilla de explicar el propósito del proyecto, así como guiar la estructura del mismo. Una buena metáfora debe ser fácil de comprender para el cliente y a su vez debe tener suficiente contenido como para que sirva de guía a la arquitectura del proyecto.

CODIFICACIÓN

Sin la codificación la aplicación carecería de sentido alguno, son el funcionamiento del sistema, sin ella la aplicación solo sería un cuerpo sin músculos y esqueleto. En ella se encuentran varios puntos de importancia, los cuales son enunciados adelantes:

- **Disponibilidad del Cliente**, Uno de los requerimientos de XP es tener al cliente disponible durante todo el proyecto. No solamente como apoyo a los desarrolladores, sino formando parte del grupo. El Involucramiento del cliente es fundamental para que pueda desarrollarse un proyecto con la metodología XP. Al comienzo del proyecto, el este debe proporcionar las historias de usuarios. Pero, dado que estas historias son expresamente cortas y de "alto nivel", no contienen los detalles necesarios para realizar el desarrollo del código. Estos detalles deben ser proporcionados por el cliente, y discutidos con los desarrolladores, durante la etapa de desarrollo.
- **Uso de Estándares**, XP promueve la programación basada en estándares, de manera que sea fácilmente entendible por todo el equipo, y que facilite la recodificación.
- **Programación Dirigida por las Pruebas ("Test-Driven Programming")**, En las metodologías tradicionales, la fase de pruebas, incluyendo la definición de los test, es usualmente realizada sobre el final del proyecto, o el final del desarrollo de cada módulo. La metodología XP propone un modelo inverso, primero se escribe los test que el sistema debe pasar. Luego, el desarrollo debe ser el mínimo necesario para pasar las pruebas previamente definidas. Las pruebas a los que se refiere esta práctica, son las pruebas unitarias, realizados por los desarrolladores. La definición de estos test al comienzo, condiciona o "dirige" el desarrollo.
- **Programación en Pares**, XP propone que se desarrolle en pares de programadores, ambos trabajando juntos en un mismo ordenador. Si bien parece que ésta práctica duplica el tiempo asignado al proyecto (por ende, los costos en recursos humanos), al trabajar en pares se minimizan los errores y se logran mejores diseños, compensando la inversión en horas. El producto obtenido es por lo general de mejor calidad que cuando el desarrollo se realiza por programadores individuales.
- **Integraciones Permanentes**, Todos los desarrolladores necesitan trabajar siempre con la "última versión". Realizar cambios o mejoras sobre versiones antiguas causan graves problemas, y retrasan al proyecto. Es por eso que XP promueve publicar lo antes posible las nuevas versiones, aunque no sean las

últimas, siempre que estén libres de errores. Idealmente, todos los días deben existir nuevas versiones publicadas. Para evitar errores, solo una pareja de desarrolladores puede integrar su código a la vez.

- **Propiedad Colectiva del Código**, En un proyecto XP, todo el equipo puede contribuir con nuevas ideas que apliquen a cualquier parte del proyecto. Asimismo, una pareja de programadores puede cambiar el código que sea necesario para corregir problemas, agregar funciones o re codificar.
- **Ritmo Sostenido**, La Metodología XP indica que debe llevarse un ritmo sostenido de trabajo. El concepto que se desea establecer con esta práctica es planificar el trabajo de forma a mantener un ritmo constante y razonable, sin sobrecargar al equipo.

PRUEBAS

Las pruebas es el medio que nos lleva a saber si un hecho es real o es falso, son importantes ya que consiguen detectar un porcentaje de errores, influido también por el poco tiempo que se dispone. En este caso se encuentran varios puntos que son imprescindible a la hora de aplicar la metodología XP, los cuales son:

- **Pruebas Unitarias**, Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Por otra parte, como se mencionó anteriormente, las pruebas deben ser definidas antes de realizar el código (“Test-Driven Programmming”). Que todo código liberado pase correctamente las pruebas unitarias, es lo que habilita que funcione la propiedad colectiva del código.
- **Detección y Corrección de Errores**, Cuando se encuentra un error (“Bug”), éste debe ser corregido inmediatamente, y se deben tener precauciones para que errores similares no vuelvan a ocurrir. Asimismo, se generan nuevas pruebas para verificar que el error haya sido resuelto.
- **Pruebas de Aceptación**, Son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El Cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Asimismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta que pase correctamente todas las pruebas de aceptación. (26)

Sistema de diseño 3D

Consiste en utilizar software para crear una representación matemática de un objeto o una forma tridimensionales. El objeto creado se denomina modelo 3D; estos modelos tridimensionales se usan para el diseño generado por computadora. Este se usa en una variedad de industrias para ayudar a los artistas a desarrollar, comunicar, documentar, analizar y compartir sus ideas (27).

Es un conjunto de técnicas que permiten proyectar en tres dimensiones. El primer paso consiste en idear los objetos, construcciones y piezas tridimensionales antes de modelarlas o construirlas. En el mundo existe una amplia gama de sistema de diseño 3D, entre ellos se encuentran: Blender, Unreal Engine, Cinema 4D, Zbrush, Autodesk Maya, Autodesk 3ds Max y Rhino, a continuación, se realizará un estudio de las tecnologías homólogas seleccionadas, se hará un estudio comparativo, se escogerá una de estas tecnologías con su versión y se arribará a una conclusión.

Blender

Desarrollado por Blender Foundation y como presidente de esta organización y software Ton Roosendaal es un software informático multiplataforma, dedicado especialmente al modelado, iluminación, renderizado, la animación, rigging, edición de video, creación de gráficos tridimensionales, composición, texturizado y utilizado para muchos tipos de simulaciones. También de composición digital utilizando la técnica procesal de nodos, edición de vídeo, escultura incluye topología dinámica y pintura digital. El programa fue inicialmente distribuido de forma gratuita, pero sin el código fuente, con un manual disponible para la venta, aunque posteriormente pasó a ser software libre. Actualmente es compatible con todas las versiones de Windows, macOS, GNU/Linux, Android, Solaris, FreeBSD e IRIX (28). Posee una basta documentación y una gran gamma de tutoriales en Internet, aparte de su documentación tecnológica.

Unreal Engine

Es un motor de juego, de software libre, creado por la compañía Epic Games, mostrado inicialmente en el shooter en primera persona Unreal en 1998. Aunque se desarrolló principalmente para los shooters en primera persona, se ha utilizado con éxito en una variedad de otros géneros, incluyendo videojuegos de sigilo, lucha, MMORPG y otros RPG. Con su código escrito en C++, el Unreal Engine presenta un alto grado de portabilidad y es una herramienta utilizada actualmente por muchos desarrolladores de juegos. Aunque es un software libre Epic Games impuso como único requisito pagar un 5% de los primeros 3 000 dólares que se ganen si se desarrolla y se comercializa (29).

Cinema 4D

Es un software privativo programado en C++ para la creación de gráficos y animación 3D desarrollado originariamente para Commodore Amiga por la compañía alemana Maxon, y portado posteriormente a plataformas Windows, Macintosh, OS 9 y OS X. Permite la modelación de primitivas, splines y polígonos, también permite la texturización y la animación. Sus principales virtudes son una muy alta velocidad de renderización, una interfaz altamente personalizable y flexible, y una curva de aprendizaje muy vertical; en poco tiempo se aprende mucho (30).

Zbrush

Software privativo desarrollado por Pixologic para el modelado 3d, escultura y pintura digital que constituye un nuevo paradigma dentro del ámbito de la creación de imágenes de síntesis gracias al original planteamiento de su proceso creativo (31).

Autodesk Maya

Desarrollado por Autodesk, Autodesk Maya también conocido como Maya es un programa informático dedicado al desarrollo de gráficos 3D por ordenador, efectos especiales, animación y de dibujo. Maya se caracteriza por su potencia y las posibilidades de expansión y personalización de su interfaz y herramientas. MEL (Maya Embedded Language) es el código que forma el núcleo de Maya y gracias al cual se pueden crear scripts y personalizar el paquete. Es una herramienta de efectos visuales, con un uso muy extendido debido a su gran capacidad de ampliación y personalización (32).

Autodesk 3ds Max

Anteriormente 3D Studio Max es un software privativo programado en C++ para la creación de gráficos y animación 3D desarrollado por Autodesk, en concreto la división Autodesk Media & Entertainment anteriormente Discreet. Creado inicialmente por el Grupo Yost para Autodesk. Con su arquitectura basada en plugins, es uno de los programas de animación 3D más utilizado, especialmente para la creación de videojuegos, anuncios de televisión, en arquitectura o en películas (33).

Rhino

También conocido como Rhinoceros 3D es una herramienta de software para modelado en tres dimensiones basado en NURBS. Es un software privativo de diseño asistido por computadora creado por Robert McNeel & Associates, originalmente como un agregado para AutoCAD de Autodesk. El programa es comúnmente usado para el diseño industrial, la arquitectura, el diseño naval, el diseño de joyas, el diseño automovilístico,

CAD/CAM, prototipado rápidos, ingeniería inversa, así como en la industria del diseño gráfico y multimedia (34).

Como resumen tenemos la siguiente tabla:

Tabla 3 Tabla comparativa. Elaboración propia

Programa	Sistema Operativo	Mercado	Documentación de la tecnología	Empleada para simular
Blender	Windows, macOS, GNU/Linux, Android, Solaris, FreeBSD e IRIX	Software libre	Basta documentación, gran comunidad y página oficial	Si
Unreal Engine	Windows	Software libre	Página oficial	No
Cinema 4D	Windows y Macintosh (OS 9 y OS X).	Software privativo	Página oficial	No
Zbrush	Windows	Software privativo	Página oficial	No
Autodesk Maya	Windows, macOS y Linux	Software privativo	Página oficial	No
Autodesk 3ds Max	Windows	Software privativo	No presenta página oficial	No
Rhino	Windows y macOS	Software privativo	Página oficial	No

Blender se puede utilizar para crear visualizaciones en 3D, como imágenes fijas, animaciones en 3D, tomas VFX, edición de video y simulaciones. Blender es adecuado para individuos y pequeños estudios que se benefician de su flujo de trabajo unificado y su activo proceso de desarrollo. Es una herramienta de diseño 3D muy cómoda, versátil y de fácil uso al usuario, por eso la elección, independientemente de que también es utilizada en el CTI. Se escogió Blender en su versión v2.83 como sistema de diseño 3D.

Software para la creación de videojuegos

Los softwares para la creación de videojuegos, motor de videojuegos o game engine son programas especializados de computadora que se utilizan para construir un videojuego.

El motor de videojuego se considera el núcleo que conjunta todos elementos que conforman un videojuego: las reglas, objetivos, arte, animaciones, diálogos, textos, sonidos, motor de renderizado, motor de físicas, gestión de memoria, rutinas de programación, cinemáticas, interfaces de usuario, etcétera.

Toda esta información es tomada por el motor y reinterpretada para crear un ambiente en donde el desarrollador y posteriormente el usuario podrá interactuar con esos elementos.

Hoy en día existen varios softwares para la creación de videojuegos, de ellos se encuentran: Unity 3D, Unreal Engine, Scratch, M.U.G.E.N, Roblox, Stencyl y Godot. De ellos se realizará un estudio de las tecnologías homólogas seleccionadas, se hará un estudio comparativo, se escogerá una de estas tecnologías con su versión y se arribará a una conclusión.

Unity 3D

Unity 3D es un motor de videojuegos que funciona con el lenguaje C#. Igualmente, puede ser usado por aquellas personas que tengan conocimientos en programación y que deseen en un futuro dedicarse a crear videojuegos.

Cabe destacar que este software cuenta con una versión gratuita en la que podrás desarrollar juegos para Mac, Web y PC. Este programa posee un editor visual que te permitirá importar sonidos, texturas y modelos 3D (35).

Unreal Engine

Unreal Engine es una de las herramientas 3D más utilizadas para programar videojuegos, pero requiere de conocimientos en C++ para utilizarla. Lo bueno es que a través de su página oficial encontrarás tutoriales en línea totalmente gratuitos y capacitación por parte de un instructor especializado.

La presente aplicación está recomendada para aquellas personas que sean apasionadas por la creación de videojuegos y deseen formar parte de la industria como profesionales (29).

Scratch

Scratch es un programa gratuito que ha sido diseñado para enseñar a los niños cómo crear un juego de forma fácil. De este modo, sirve para estimular en ellos la imaginación. Este programa fue desarrollado por un grupo de profesionales del Instituto Tecnológico de Massachusetts. También, mencionamos que se caracteriza por tener un lenguaje de programación visual donde tanto los niños como jóvenes utilizarán animaciones predefinidas (36).

M.U.G.E.N

M.U.G.E.N fue lanzado en 1999 y es un programa gratuito para el desarrollo de videojuegos de lucha. Gracias a su versatilidad y sencillez se ha convertido en uno de los programas más utilizados.

Es un motor de videojuegos en dos dimensiones desarrollado usando el lenguaje de programación C que originalmente utilizó la biblioteca de programación Allegro. La versión actual utiliza la biblioteca SDL (37).

Roblox

Roblox tiene una trayectoria de 13 años en el mercado. Con él, los desarrolladores o jugadores podrán crear mundos nuevos, además de comunicarse con otros usuarios. Este programa está disponible para PC, Xbox One, dispositivos móviles IPhone y Android.

Cabe destacar que con esta plataforma podrás crear videojuegos como Minecraft y Lego. Los usuarios pueden crear sus propios mundos virtuales con el sistema de creación de juegos llamado Roblox Studio, desarrollado por Roblox Corporation. Es una versión sandbox y el lenguaje de programación que se utiliza es Luau (38).

Stencyl

Stencyl es un programa para la creación de videojuegos en 2D y ofrece un equilibrio entre los buenos resultados y la facilidad de uso. Los juegos que puedes crear en este programa estarán disponibles para ordenadores, dispositivos móviles o páginas Web.

De igual forma, en su sitio oficial ofrece un kit educativo que permitirá enseñar a los niños cómo crear un videojuego. El software está disponible gratuitamente, con la selección de opciones de publicación disponibles si se compra el programa. Se programo en Java, ActionScript, Objetive-C, C++ y Haxe (39).

Godot

En este top tenemos el programa Godot, el cual es de código abierto y gratuito. Con él podrás crear videojuegos en 2D y 3D, según tus preferencias. Integrado a ello, te

permite programar en Linux, OS X y Windows, utilizando GDScript, C++ o C# como lenguajes de programación. Finalmente, mencionamos que este software posee un sistema de animación flexible para crear juegos innovadores y con buenos gráficos 3D. Incluso, sirve para hacer juegos de realidad virtual (40).

En la siguiente tabla se compararán estos softwares para la creación de juegos y se dará una conclusión:

Tabla 4 Tabla comparativa. Elaboración propia

Software	Gráficos	Mercado	Lenguaje	Plataforma
Unity 3D	2D, 3D	Libre	C#	Multiplataforma
Unreal Engine	2D (con ayuda de herramientas externas), 3D	Libre	C++	Multiplataforma
Scratch	2D	Libre	Scratch	PC
M.U.G.E.N.	2D	Libre	C	PC
Roblox	3D	Libre	Lua	Multiplataforma
Stencyl	2D	Libre	Haxe, ActionScript	PC
Godot.	2D, 3D	Libre	GDScript, C++, C, C#, VBScript	Multiplataforma

Este software para la creación de videojuegos es practico, cómodo e intuitivo, aparte de esto, tiene una buena documentación y tutoriales accesible a cualquiera que quiera emplearlo, que, al igual que Blender, es utilizada en el CTI con este propósito. Se escogió Unity 3D en su versión 2018.3.5f1 como motor de juego.

Lenguajes

Lenguaje de programación

Un lenguaje de programación es un lenguaje formal o artificial, es decir, un lenguaje con reglas gramaticales bien definidas que le proporciona a una persona, en este caso

el programador, la capacidad de escribir o programar una serie de instrucciones o secuencias de órdenes en forma de algoritmos con el fin de controlar el comportamiento físico o lógico de un sistema informático, de manera que se puedan obtener diversas clases de datos o ejecutar determinadas tareas.

Es una forma de comunicarnos con una computadora, Tablet o celular e indicarle qué queremos hacer. A todo este conjunto de órdenes escritas mediante un lenguaje de programación se le denomina programa informático (41). Como se va a utilizar el motor de juego Unity 3D y esta emplea C# como lenguaje de programación, a continuación, se caracterizará dicho lenguaje.

C# v1.21.5

C# (o C Sharp) es un lenguaje de programación muy popular multiparadigma desarrollado y estandarizado por la empresa Microsoft como parte de su plataforma .NET. Es un poco menos flexible y compatible que C++, pero algunos motores como Unity permiten programar con él y no está limitado a un determinado sistema operativo o plataforma; se pueden crear juegos para iOS, Android, Windows PlayStation y Xbox.

En su versión tiene como características se tiene:

- Sintaxis sencilla que facilita al desarrollador la escritura del código
- Sistema de tipo unificado, permitiendo realizar operaciones comunes y que los valores de todos los tipos se puedan almacenar, transportar y utilizar de manera coherente
- Orientaciones a componentes, aunque es un lenguaje orientado a objeto, también lo es a componentes porque permite definir propiedades sin necesidad de crear métodos o usar eventos sin tratar con punteros a funciones
- Espacio de nombres que se puede aislar o agrupar código mediante bibliotecas
- Multihilo donde se pueden dividir el código en múltiples hilos de ejecución, trabajar en paralelos y sincronizarlos al final.
- Es flexible

Es el lenguaje a escoger por ser multiplataforma, por lo que permite exportar los videojuegos a ordenadores, dispositivos móviles con Android o iOS, y las diferentes videoconsolas de mercado, además, es compatible con Unity 3D, muy cómodo, cuenta con mucha documentación y tutoriales que ayudan a los programadores cunado tienen alguna inquietud, igualmente, es utilizado por el CTI para la programación de videojuegos en Unity.

Lenguaje de modelado

El lenguaje de modelado a utilizar será UML, ya que es un lenguaje diseñado para visualizar, especificar y documentar un sistema, este ofrece un estándar para describir un plano del sistema, incluyendo aspectos conceptuales tales como, procesos, funciones del sistema, y aspectos en concreto como expresiones de lenguaje de programación, esquemas de base de datos y compuestos reciclados.

Fue creado para forjar un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento. Es una técnica para la especificación de sistemas en todas sus fases (42).

Además, cuentan con las siguientes características:

- Permiten una visión descendente del sistema.
- Poseen componentes gráficos con algo de apoyo textual.
- El modelo resultante debe ser transparente (fácil de comprender).
- Poseen mínima redundancia (el aumento de redundancia, disminuye la transparencia del modelo y aumenta las tareas de mantenimiento).

La herramienta de modelado utilizada para este lenguaje fue Visual Paradigm v16.3 ya que a pesar de que utiliza este lenguaje de modelado, es la herramienta ideal para ingenieros de software, analistas de sistema y arquitectura de sistema que están interesados en la construcción de sistemas a grandes escalas, entre las características de Visual Paradigm tenemos:

- Diseño centrado en casos de uso.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Editor de detalles de casos de uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Licencia: gratuita y comercial.

- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Generación de código - Modelo a código, diagrama a código.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Disponibilidad en Windows y Linux.
- Capacidades de ingeniería directa e inversa.

La utilización de esta herramienta permite aumentar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del software, así como aumenta el conocimiento informático de una empresa ayudando así a la búsqueda de soluciones para los requisitos.

IDE de desarrollo

Un entorno de desarrollo integrado IDE siglas del acrónimo del término inglés Integrated Development Environment, es una aplicación de software, que proporciona servicios integrales para facilitarle al programador de computadora el desarrollo de software.

Hace que la tarea del programador sea más sencilla, gracias a las herramientas que tiene incorporadas, como compiladores, depuradores o bibliotecas, y esto se traduce en un aumento de la productividad (43). Para el lenguaje C# existen varios IDE's de desarrollo, entre ellos se encuentra: Visual Studio Code, Eclipse aCute, Rider, DevExpress y Codeanywhere. De ellos se realizará un estudio de las tecnologías homólogas seleccionadas, se hará un estudio comparativo, se escogerá una de estas tecnologías con su versión y se arribará a una conclusión.

Visual Studio Code

Es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. Es gratuito y de código abierto, aunque la descarga oficial está bajo software privativo e incluye características personalizadas por Microsoft. Fácil de trabajar con Git y otros proveedores de gestión de configuración de software. Ofrece refactorización y depuración de código. Este IDE de C# es fácilmente extensible y personalizable (44).

Eclipse aCute

Se basa en OmniSharp-Roslyn y Language Server Protocol para la edición, y en netcoredbg y Debug Adapter Protocol para la depuración. La integración con esas herramientas está impulsada por Eclipse LSP4E. Eclipse aCute se basa en TM4E y una

gramática de compañero de texto para proporcionar resaltado de sintaxis en el IDE. Proporciona plantillas de código listas para usar. Esta herramienta valida automáticamente la sintaxis. Eclipse le permite administrar el proyecto de forma remota (45).

Rider

Es un IDE .NET multiplataforma basado en la plataforma IntelliJ y ReSharper. Entre sus características se encuentran: Análisis del código, Edición de código, Refactorización, Ejecutor de pruebas de unidad, Depurador y más herramientas, Permite trabajar con SQL y bases de datos, Navegación y búsqueda y Extensibilidad. Permite dividir el editor ya sea horizontal o verticalmente y también permite ejecutar y depurar pruebas unitarias basadas en NUnit, xUnit.net o MSTest. Ayuda a visualizar las dependencias del proyecto en su solución (46).

DevExpress

DevExpress es uno de los mejores IDE de C#, que lo ayuda a crear experiencias de usuario elegantes y de alto impacto con el emulador. También proporciona sistemas de soporte de decisiones de alto rendimiento y paneles de análisis. Tiene componentes de la interfaz de usuario de escritorio, ofrece archivo de oficina y API de PDF, ofrece desarrollo de aplicaciones .Net multiplataforma (47).

Codeanywhere

Codeanywhere es uno de los mejores IDE de C# que lo ayuda a ahorrar tiempo al implementar un entorno de desarrollo en segundos. También le permite codificar, aprender, construir y colaborar en sus proyectos.

Proporciona un entorno de desarrollo completo con memoria y espacio en disco dedicados. Puede seguir con sus herramientas de desarrollo y flujos de trabajo favoritos y ayuda a ver e insertar posibles terminaciones para acelerar su codificación (48).

En la siguiente tabla se compararán estos IDE's y se arribará una conclusión:

Tabla 5 Tabla comparativa. Elaboración propia

IDE de desarrollo	**Mercado**	**Plataforma**	**Documentación de la tecnología**
Visual Studio Code	Libre	Windows, Linux	Presenta gran cantidad de documentación,

			gran variedad de tutoriales y página oficial
Eclipse aCute	Libre	Máquina virtual java	Página oficial
Rider	Libre	Windows	Página oficial
DevExpress	Libre	Windows	Página oficial
Codeanywhere	Libre	Windows	Página oficial

Visual Studio Code es muy agradable a la vista, es una herramienta que tiene soporte nativo para gran variedad de lenguajes como C#, es multiplataforma y compatible con Unity, utiliza gráficos de vanguardia, asimismo, es utilizado en el CTI por su comodidad y versatilidad. Se escogerá Visual Studio Code v1.43.1 como IDE de desarrollo.

Conclusiones del capítulo

Con el estudio de las herramientas similares se identifica que todas tienen licencia propietaria por lo que; no se pueden utilizar para el cumplimiento del objetivo propuesto. Sin embargo, su estudio posibilitó la identificación de características que se tendrán en cuenta en el desarrollo de la solución.

Las herramientas a utilizar para el desarrollo de la propuesta son: Blender versión 2.83 para el diseño en 3D, Unity 3D versión 2018.3.5f1 para la creación del videojuego y Visual Studio Code versión 1.43.1 como IDE de desarrollo.

Mientras que el lenguaje de programación a utilizar es C# versión 1.21.5 ya que; este es el lenguaje de programación con el que trabaja Unity 3D. Se definió que como metodología de desarrollo de software XP ya que; es la metodología utilizada en el CTI para el desarrollo de sus productos.

Capítulo 2: Planificación y diseño de la propuesta de solución

En este capítulo se aborda la propuesta de solución a la problemática planteada. Se describen los requisitos funcionales y no funcionales, para dar cumplimiento a los objetivos planteados.

Se realiza la captura de requisitos funcionales y no funcionales, la propuesta de solución, las historias de usuario, plan de iteraciones, tarjetas Clase-Responsabilidad-Colaborador (CRC), diagrama de clases de la solución, estándares de codificación y demás artefactos exigidos por la Metodología XP y sus fases.

Propuesta de solución

El simulador de penales permitirá renderizar un campo de futbol donde el usuario podrá visualizar diferentes objetos en el campo. Los objetos que se podrán visualizar son: Jugador, portero, pelota, portería y el mismo estadio. De estos objetos el portero y el jugador se encuentran animados. Estas animaciones consisten en: moverse por el campo y en el caso del portero detener o desviar el balón.

El usuario al visualizar el campo podrá seleccionar la posición en la que se trasladará el balón. Una vez que se seleccione la posición el jugador realizará una fuerza y se calculará de forma automática la trayectoria del balón hacia el punto seleccionado. Para poder realizar el cálculo automático se debe definir la interpolación que existe entre dos puntos, la cual será explicada en el capítulo 3.

Al realizar el tiro de forma automática el portero va a detener o desviar el balón. Además, al realizarse el primer tiro se estará llevando al unisonó la cantidad de goles teniendo en cuenta las siguientes direcciones: izquierda, derecha y medio y la cantidad de fallos, esto permite que, al realizarse otro tiro, el portero se mueva según el control de las estadísticas.

Fase 1: Planeación

Se cumple con lo establecido en la Fase I de la Metodología XP, donde el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.

Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de planeación toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

Determinación de requisitos

El análisis de requisitos es una de las tareas más importantes en el ciclo de vida del desarrollo de software, puesto que en ella se determinan los "planos" de la nueva aplicación.

El análisis de requisitos se puede definir como el proceso del estudio de las necesidades de los usuarios para llegar a una definición de los requisitos del sistema, hardware o software, así como el proceso de estudio y refinamiento de dichos requisitos (49).

Técnica de captura de requisitos

En el proceso de desarrollo de software los analistas emplean varias técnicas para obtener los requisitos del cliente. Históricamente, esto ha incluido técnicas tales como las entrevistas, o talleres con grupos para crear listas de requisitos.

Cuando sea necesario, el analista empleará una combinación de estos métodos para establecer los requisitos exactos de las personas implicadas, para producir un sistema que resuelva las necesidades del cliente (10). A continuación, se muestran algunas de estas técnicas:

Entrevista

Las entrevistas son un método común. Por lo general no se entrevista a toda la gente que se relacionará con el sistema, sino a una selección de personas que represente a todos los sectores críticos de la organización, con el énfasis puesto en los sectores más afectados o que harán un uso más frecuente del nuevo sistema (10).

Por otra parte, Hurtado (2008) opina que la técnica de entrevista es la información que se recoge solicitándola a otra persona. El investigador no puede tener la experiencia directa del evento; es otro quien la tiene, la información se obtiene dialogando.

Revisión documental

La revisión documental, como bien lo dice su palabra, consta de revisar los documentos parecidos al sistema y anotar apuntes o ideas que pueden servir para un futuro. Ninguna idea debería ser descartada ya que cuenta con una investigación científica y un respaldo teórico.

Según Hurtado (2008) afirma que una revisión documental es una técnica en donde se recolecta información escrita sobre un determinado tema, teniendo como fin

proporcionar variables que se relacionan indirectamente o directamente con el tema establecido, vinculando esta relaciones, posturas o etapas.

Lluvia de ideas

Todos los participantes pueden aportar distintas ideas en un ambiente libre de prejuicios. Ningún participante debe juzgar negativamente la propuesta de otros, sino que se anotan todas las ideas en una pizarra y serán evaluadas al final de la sesión.

El principio básico es no descartar de manera apresurada ningún planteo, de modo que existe la posibilidad de que surjan otras ideas derivadas de la idea original y se generan varios puntos de vista del problema. Puede existir el inconveniente de no llegar a una idea en común (10).

Objetivos medibles

Los requisitos formulados por los usuarios se toman como objetivos generales, a largo plazo, y en cambio se los debe analizar una y otra vez desde el punto de vista del sistema hasta determinar los objetivos críticos del funcionamiento interno que luego darán forma a los comportamientos apreciables por el usuario. Luego, se establecen formas de medir el progreso en la construcción, para evaluar en cualquier momento qué tan avanzado se encuentra el proyecto (50).

Selección de la técnica

Luego de analizar las posibles técnicas a utilizar se decidió aplicar la entrevista, la lluvia de ideas y el estudio de herramientas similares, de conjunto con la revisión documental debido a la dinámica de desarrollo del software.

Requisitos funcionales

Los requisitos funcionales describen la interacción entre el sistema y su ambiente independientemente de su implementación. El ambiente incluye al usuario y cualquier otro sistema externo que interactúa con el sistema (10).

Los requisitos funcionales definen funciones del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas.

La guía del Business Analysis Body of Knowledge (BABOK) en su versión 3, plantea la siguiente definición: “Los requerimientos funcionales son las descripciones explicitas del

comportamiento que debe tener una solución de software y que información debe manejar."

Estos expresan la capacidad o cualidades que debe tener la solución para satisfacer los requerimientos de los interesados de proyecto. Deben proporcionar una descripción lo suficientemente detallada para permitir el desarrollo e implementación de la solución y son los que más influyen en si la solución será aceptada o no por los usuarios (51).

Como requisitos funcionales del sistema se encuentran:

- RF1. Renderizar la escena de simulación de futbol.
- RF2. Seleccionar la posición en la que se trasladará el balón hacia la portería.
- RF3. Calcular la trayectoria del balón según la posición y la fuerza.
- RF4. Animar los avatares de la simulación del juego.
- RF5. Detener o desviar el balón lanzado por un jugador.
- RF6. Seleccionar efecto del balón de futbol.
- RF7. Mostrar las estadísticas del cálculo de la simulación.

Requisitos no funcionales

Los requisitos no funcionales describen aspectos del sistema que son visibles por el usuario que no incluyen una relación directa con el comportamiento funcional del sistema. Los requerimientos no funcionales incluyen restricciones como el tiempo de respuesta (desempeño), la precisión, recursos consumidos, seguridad, etcétera (52).

Se trata de requisitos que no se refieren directamente a las funciones específicas suministradas por el sistema (características de usuario). Alternativamente, definen restricciones del sistema tales como la capacidad de los dispositivos de entrada/salida y la representación de los datos utilizados en la interfaz del sistema (53).

Los requerimientos no funcionales son los que especifican criterios para evaluar la operación de un servicio de tecnología de información y especifican los criterios que debe cumplir para que sea adecuado para su uso (54).

Requisitos no funcionales:

Usabilidad

- RNF1. El sistema debe mostrar un mensaje de confirmación cuando el jugador desee salir del juego.

Eficiencia

- RNF2. La aplicación debe tener una respuesta lo más corta posible al ejecutar una acción para que su funcionamiento sea óptimo. Se toma que el tiempo mínimo que debe demorar al ejecutar la acción debe ser de 2 segundos como máximo.

Interfaz

- RNF4. Se debe mantener la misma distribución y diseño entre los elementos del sistema en la pantalla. Por lo que; el diseño debe ser Responsive.
- RNF5. La aplicación debe tener una interfaz clara y sencilla. Para dar cumplimiento a este requisito no funcional se tomaron en cuenta buenas prácticas las cuales se encuentran definidas en MeriStation (54):
  - ✓ Sensación de profundidad en los videojuegos
  - ✓ La influencia de la animación tradicional
  - ✓ Visión real a partir de la proyección cónica
  - ✓ Exploración de mundos tridimensionales

Software

- RNF6. Interprete de C# versión 1.21.5
- RNF7. Utilizar como lenguaje C# versión 1.21.5
- RNF8. Utilizar como sistema de diseño 3D a Blender versión 2.83.
- RNF9. Utilizar como software para la creación de videojuegos a Unity versión 2018.3.5f1
- RNF10. Utilizar como IDE de desarrollo a Visual Studio Code versión 1.43.1

Hardware

- RNF11. PC Intel i5 4ta generación o superior
- RNF12. 4 GB de RAM o superior
- RNF13. 500 GB de disco duro o superior
- RNF14. Tarjeta gráfica compatible con OpenGL y 256 MB de memoria o superior

Historias de usuario

Las historias de usuario son la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales.

El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (10).

Se realiza la selección del RF por el nivel de prioridad para ser desarrollado a continuación, las demás historias de usuario se encuentran en la sección Anexo 1 Historias de Usuario.

Tabla 6 Historia de usuario número 1

Historia de usuario	
Número: 1	**Nombre del requisito:** Seleccionar la posición en la que se trasladará el balón hacia la portería
Programador: Sergio Daniel Ortiz Jova	**Iteración asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 13 días
Riesgo en desarrollo: N/A	**Tiempo real:** 11 días
Descripción: Deberá permitir seleccionar la posición en la que se debe trasladar el balón hacia la portería.	
Observaciones: N/A	

Estimación de tiempo por historia de usuario

La programación bajo la metodología XP (programación extrema) basa sus procesos de planificación en estimaciones temporales de las historias de usuario, las cuáles deben ser realizadas por los desarrolladores durante las diversas reuniones de planificación.

Una historia de usuario es lo suficientemente pequeña como para que el equipo la desarrolle durante una entrega de una a tres semanas; más de tres semanas implica

que se debe señalar al cliente que debe dividir una historia de usuario y menos de una semana implica que la historia es demasiado sencilla y por lo que se deben unir dos o más de ellas para su mejor interpretación.

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto de estimación. Un punto de estimación, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos de estimación.

Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración (10).

En la siguiente tabla de estimación del tiempo de las historias de usuario, se usó la denotación de punto de estimación por semana y la denotación de puntos flotantes para indicar días.

Ejemplo:

Estimación de tiempo:

Tabla 7 Estimación de tiempo

Denotación de punto de estimación	**Interpretación**
1 punto	1 día (8 horas laborales)
1.5 puntos	1 día y 4 horas

Estimación de tiempo de HU

Tabla 8 Estimación de tiempo de Historia de Usuario

Historia de Usuario	**Punto estimado**
Renderizar la escena de simulación de futbol.	20
Seleccionar la posición en la que se trasladará el balón hacia la portería.	13

Calcular la trayectoria del balón según la posición y la fuerza.	18
Animar los avatares de la simulación del juego.	12
Detener o desviar el balón lanzado por un jugador.	8
Seleccionar efecto del balón de futbol.	2
Mostrar las estadísticas del cálculo de la simulación.	18

Las estimaciones realizadas permitieron confeccionar una evaluación puntual del tiempo de implementación de cada historia de usuario para la posterior elaboración del plan de iteración. Una vez realizadas las estimaciones fue preciso construir un plan de iteración donde se pueden agrupar estas historias y dar su cumplimiento de manera paulatina.

Plan de iteraciones

Un plan de iteración está constituido por un conjunto secuencial de actividades y tareas, cada una tiene recursos asignados y pueden depender a su vez de otras tareas. El plan de iteración se realiza una vez por cada iteración.

El contenido de la iteración puede variar, dependiendo de la posición dentro del ciclo de vida y de la naturaleza del proyecto. El plan de iteración incluye porciones relevantes de todas las disciplinas para cada iteración en particular, la prioridad de implementación se evalúa en base al cliente y el equipo de desarrollo, y el impacto del riesgo en base al juicio de experto.

Los planes de iteración por lo general muestran un planeamiento detallado de quién va a realizar una tarea/actividad de acuerdo en conformidad a que criterios de evaluación (10).

La siguiente tabla muestra por cada iteración, la(s) historia(s) de usuario que se realizan y de ella, todas las tareas ingenieriles que se ejecutaron en la misma.

Tabla 9 Número de iteración, historia de usuario y tarea ingenieril. Elaboración propia

No. de iteración	**Historia de usuario**	**Tarea ingenieril**
1	Renderizar la escena de simulación de futbol.	Realizar la creación de objetos 3D
		Ponerles los huesos a los objetos 3D realizado
		Exportar objetos al motor de videojuego
		Realizar las pruebas unitarias a la primera iteración
2	Seleccionar la posición en la que se trasladará el balón hacia la portería.	Implementación de un procedimiento para colocar el mouse en el medio de la pantalla cuando empiece el juego.
		Estudio de las fórmulas para calcular la posición del mouse en la pantalla.
		Implementación de un procedimiento guardar la posición del mouse al presionar el clic izquierdo.
	Calcular la trayectoria del balón según la posición y la fuerza.	Colocar los objetos en los lugares predefinidos.
		Buscar una fórmula para darle una fuerza al balón.
		Implementación de un método para aplicar una fuerza en la dirección del mouse.
		Implementación de un método para dirigir el balón hacia la posición guardada por el clic del mouse con la fuerza dada.
		Realizar las pruebas unitarias de la iteración número uno.
		Realizar las pruebas unitarias de la iteración número dos.

3	Animar los avatares de la simulación del juego.	Animar los objetos que lo requieran con los huesos.
	Detener o desviar el balón lanzado por un jugador.	Calcular la distancia de la posición del portero a los laterales de la portería.
		Seleccionar el lugar hacia donde cubrirá el portero.
	Seleccionar efecto del balón de futbol.	Asignar un peso al balón de futbol.
		Asignar el efecto de la gravedad al balón.
	Mostrar las estadísticas del cálculo de la simulación.	Crear la interfaz gráfica necesaria para llevar a cabo las estadísticas.
		Realizar cálculos matemáticos y estadísticos para contar la cantidad de goles.
		Realizar pruebas unitarias a la primera iteración.
		Realizar pruebas unitarias a la segunda iteración.
		Realizar pruebas unitarias a la tercera iteración.
		Realizar pruebas de aceptación al sistema.

A continuación, se puede observar tres tablas, una por cada iteración, de las tareas ingenieriles realizadas: realizar la creación de objetos 3D, implementación de un procedimiento para colocar el mouse en el medio de la pantalla cuando empiece el juego y animar los objetos que lo requieran con los huesos. Las demás tareas de ingeniería se encuentran en la sección Anexo 6 Tareas ingenieriles.

Tabla 10 Tarea ingenieril realizar la creación de objetos 3D.

Tarea de ingeniería	
Número de tarea: 1	**Número de historia de usuario**: 1

Nombre de tarea: Realizar la creación de objetos 3D.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 4
Fecha inicio: 22/3/2022	**Fecha fin**: 25/3/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Crear los objetos 3D en Blender.	

Tabla 11 Tarea ingenieril Implementación de un procedimiento para colocar el mouse en el medio de la pantalla cuando empiece el juego

Tarea de ingeniería	
Número de tarea: 5	**Número de historia de usuario**: 2
Nombre de tarea: Implementación de un procedimiento para colocar el mouse en el medio de la pantalla cuando empiece el juego.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 3
Fecha inicio: 14/4/2022	**Fecha fin**: 16/4/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Cuando comience el juego el mouse deberá aparecer hacia el centro de la pantalla.	

Tabla 12 Tarea ingenieril animar los objetos que lo requieran con los huesos

Tarea de ingeniería	
Número de tarea: 14	**Número de historia de usuario**: 4
Nombre de tarea: Animar los objetos que lo requieran con los huesos.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 12
Fecha inicio: 23/5/2022	**Fecha fin**: 4/6/2022

Programador responsable: Sergio Daniel Ortiz Jova
Descripción: Se deberá animar los objetos con huesos que así lo requieran.

Tabla 13 Plan de iteraciones

No. de iteración	Historia de usuario	Prioridad	Esfuerzo estimado	
1	Renderizar la escena de simulación de futbol.	Alta	20	20
2	Seleccionar la posición en la que se trasladará el balón hacia la portería.	Alta	13	31
	Calcular la trayectoria del balón según la posición y la fuerza.	Media	18	
3	Animar los avatares de la simulación del juego.	Alta	12	40
	Detener o desviar el balón lanzado por un jugador.	Alta	8	

	Seleccionar efecto del balón de futbol.	Media	2	
	Mostrar las estadísticas del cálculo de la simulación.	Media	18	

Una vez realizado el plan de iteración se pudo agrupar las diferentes historias de usuario en 3 iteraciones teniendo en cuenta las características que rigen la metodología XP. Con este plan de iteraciones ya es posible realizar un plan de entrega que será entregado al cliente y que el grupo de desarrollo está obligado a hacer cumplir.

Plan de entrega

Esta clase de documento (convenido con el cliente) es un contrato marco con el cliente que contiene las cantidades y fechas de entrega. El objeto del documento de plan de entregas es recoger la relación de entregas del proyecto, especificando las actividades e hitos que la componen (55).

Determina la duración de cada iteración, se presenta el plan de entrega elaborado para la fase de implementación teniendo en cuenta que el desarrollo del proyecto inicia el 22/3/2022 y concluirá el 7/7/2022:

Tabla 14 Plan de entrega

	Iteración No. 1	Iteración No. 2	Iteración No. 3
Cantidad de HU	1	2	3
Fecha de inicio	22/3/2022	14/4/2022	23/5/2022
Fecha de entrega	13/4/2022	21/5/2022	7/7/2022

Una vez realizado el plan de entrega se puede confirmar con el cliente que el proyecto durará 16 semanas aproximadamente, lo que significa que tendrá un tiempo de desarrollo de 4 meses aproximadamente.

Reuniones diarias de seguimiento

El planeamiento es esencial para cualquier tipo de metodología, es por ello que XP requiere de una revisión continua del plan de trabajo. A pesar de ser una metodología que evita la documentación exagerada, es muy estricta en la organización del trabajo.

Para la misma, el equipo de desarrollo definió desde el inicio aquellos espacios en los que el equipo y el cliente realizarían encuentros semanales para la evaluación del cumplimiento de los resultados de las diferentes iteraciones, así como el cumplimiento parcial, total o nuevas redefiniciones a los conceptos establecidos.

Las pequeñas entregas facilitarán el trabajo de los desarrolladores, haciendo posible suplir las necesidades del cliente (10).

Tabla 15 Reuniones diarias de seguimiento

Reunión de / Fecha		**Descripción**
Plan de entregas	1/11/2021	Este plan se ejecuta entre el equipo de trabajo y los clientes y se define el marco temporal de la realización del sistema. El cliente expone las historias de usuario a los integrantes de grupo, quienes estimarán el grado de dificultad de la implementación de cada historia. Luego de las historias de usuario, el cliente plantea las pruebas de aceptación con las cuales se comprueba que cada una de estas ha sido correctamente implementada.
Inicial de iteración	22/3/2022	Esta reunión es realizada previo a iniciar una iteración donde se organizan las actividades de programación a realizar. Las historias de usuario son traducidas a tareas y asignadas a los desarrolladores.
Diarias	Al inicio de cada día de trabajo	Estas reuniones se realizan al comienzo de la jornada laboral, donde todo el equipo de desarrollo se reúne para exponer los problemas e ideas que se estén presentando.

		Es de suma importancia evitar las discusiones largas, ya que se está utilizando tiempo laboral que puede ser destinado a la construcción del sistema.
Fin de iteración	13/4/2022 21/5/2022 7/7/2022	Estas reuniones se ejecutan al terminar cada iteración en conjunto con los clientes para presentar los avances en cada iteración y demostrar la aceptación por los mismos como muestra de una correcta implementación.

Con el plan de reuniones queda determinado la secuencia de las distintas actividades a realizar dentro del proceso de desarrollo de la solución, contribuyendo a la calidad del producto deseado. Una vez determinado los artefactos organizativos, se procede a la fase de diseño de la solución propuesta.

Se cumple con la Fase 1 de la Metodología XP donde el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de un mes.

Fase 2: Diseño

Teniendo en cuenta lo que plantea la metodología XP, en esta fase se confeccionan las tarjetas CRC para la descripción de las principales clases del sistema. Se define la arquitectura del sistema y los prototipos de interfaz de usuario que se utilizaran en esta simulación.

Tarjetas CRC

Las tarjetas CRC se elaboran durante la fase de diseño de la metodología XP para describir las entidades existentes en la aplicación. El uso de este tipo de tarjetas es una técnica de modelado que permite identificar las clases, responsabilidades y colaboraciones. El objetivo es obtener un diseño simple, elegante y fácil de comprender por parte de los programadores (10).

Según Pressman, la tarjeta CRC “es un conjunto de tarjetas índice estándar que representan clases”

Son una herramienta de brainstorming usada como metodología para el diseño de software orientado a objetos. Es una técnica para la representación de sistemas orientado a objetos, para pensar en objetos.

Son un puente de comunicación entre diferentes participantes. Permite ver las clases como algo más que depositario de datos, sino conocer el comportamiento de cada una en un alto nivel. Mas adelante se podrá observar un ejemplo de una tarjeta CRC del sistema de simulación del juego de futbol, las demás tarjetas CRC se encuentra en la sección Anexo 2 Tarjeta CRC.

Tabla 16 Tarjeta CRC número 1

Clase: Pelota	
Responsabilidad: Update: Se encarga de moverse hacia la portería pudiendo colisionar con la portería y el portero.	Clase relacionada: Player Portería Portero Estadio

Con la realización de las tarjetas CRC se evidencia la interrelación existente entre cada clase de la solución y sus diferentes funcionalidades. Teniendo en cuenta la interrelación de las entidades se hace necesario realizar prototipos que evidencien al cliente y al equipo de desarrollo el resultado que se obtendría con la solución.

Arquitectura

De acuerdo con Pressman, la arquitectura de software no es otra cosa que "…una descripción de los subsistemas y los componentes de un sistema informático y las relaciones entre ellos".

La arquitectura que define el CTI es la que define Unity para sus proyectos de videojuegos, el cual emplea una arquitectura híbrida entre la arquitectura en capas y la arquitectura basada en componentes.

Arquitectura en capas

La arquitectura basada en capas se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada (56).

Los principios comunes que se aplican cuando se diseña para usar este estilo de arquitectura incluyen:

- **Abstracción**. La arquitectura basada en capas abstrae la vista del modelo como un todo mientras que provee suficiente detalle para entender las relaciones entre capas.
- **Encapsulamiento**. El diseño no hace asunciones acerca de tipos de datos, métodos, propiedades o implementación.
- **Funcionalidad claramente definida**. El diseño claramente define la separación entre la funcionalidad de cada capa. Capas superiores como la capa de presentación envía comandos a las capas inferiores como la capa de negocios y la capa de datos y los datos fluyen hacia y desde las capas en cualquier sentido.
- **Alta cohesión**. Cada capa contiene funcionalidad directamente relacionas con la tarea de dicha capa.
- **Reutilizable**. Las capas inferiores no tienen ninguna dependencia con las capas superiores, permitiéndoles ser reutilizables en otros escenarios.
- **Desacople**. La comunicación entre las capas está basada en la abstracción lo que provee un desacople entre las capas.

Arquitectura basada en componentes

Una arquitectura basada en componentes describe una aproximación de ingeniería de software al diseño y desarrollo de un sistema. Esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas.

Esto provee un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma como se comparte el estado (57).

Los principios fundamentales cuando se diseña un componente es que estos deben ser:

- **Reusable**. Los componentes son usualmente diseñados para ser utilizados en escenarios diferentes por diferentes aplicaciones, sin embargo, algunos componentes pueden ser diseñados para tareas específicas.
- **Sin contexto especifico**. Los componentes son diseñados para operar en diferentes ambientes y contextos. Información específica como el estado de los

datos deben ser pasadas al componente en vez de incluirlos o permitir al componente acceder a ellos.

- **Extensible**. Un componente puede ser extendido desde un componente existente para crear un nuevo comportamiento.
- **Encapsulado**. Los componentes exponen interfaces que permiten al programa usar su funcionalidad. Sin revelar detalles internos, detalles del proceso o estado.
- **Independiente**. Los Componentes están diseñados para tener una dependencia mínima de otros componentes. Por lo tanto, los componentes pueden ser instalados en el ambiente adecuado sin afectar otros componentes o sistemas.

Patrones de diseño

Si bien la elaboración de un buen diseño del software contribuye directamente a la calidad del producto final, gran parte de esta yace en la utilización adecuada de los patrones de diseño y arquitectónicos existentes en la actualidad.

Los patrones son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan (58).

Seguidamente, se describen los principales patrones empleados en el diseño e implementación de la solución.

Patrón Flyweight

El motor de juego utiliza dicho patrón para reducir el uso de memoria y mejorar el rendimiento al reducir la creación de objetos. Este patrón busca objetos similares que ya existen para reutilizarlo en lugar de crear otros nuevos que sean similares. Esto se observa en la repetición de la misma escena del juego.

Patrón Observer

Este patrón permitió definir un mecanismo de suscripción para notificar al portero, jugador y portería sobre cualquier evento que le suceda al objeto balón que está siendo observado.

Patrón GRASP

- Alta Cohesión: Nos dice que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase.

La clase Portería.cs está altamente cohesionada con la pelota al ser responsables del envío en tiempo real de los datos obtenidos de los goles.

- Bajo Acoplamiento: Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases.

  La clase Player.cs es un ejemplo de bajo acoplamiento pues esta depende solamente del estadio para su funcionamiento.

- Experto: Es el principio básico de asignación de responsabilidades. Se indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo.

  La clase Pelota.cs es un ejemplo en el que se utiliza el patrón experto, pues esta clase contiene información de la mayoría de las clases.

Conclusiones parciales

En este capítulo se identificó un total de 7 requisitos funcionales y 15 requisitos no funcionales. Para el desarrollo de los requisitos funcionales se ejecutarán 3 iteraciones, en la primera iteración se realizará el RF1, en la segunda iteración RF2 y RF3 y en una tercera iteración los RF4, RF5 y RF6.

Por cada uno de los requisitos funcionales se describieron las historias de usuario por el cliente, las cuales permiten una mejor comprensión de lo que se quiere lograr con el requisito.

Por otro lado, se realizaron las tarjetas CRC las cuales describen las clases y responsabilidades del producto realizado.

Capítulo 3: Implementación, prueba y validación de la propuesta

En este capítulo se aborda la implementación de una función para el cálculo de la interpolación, los estándares de codificación del lenguaje seleccionado y las pruebas.

Implementación

En este epígrafe se aborda temas tales como: la implementación de una función para calcular la interpolación de dos puntos y los estándares de codificación.

Implementación de una función para calcular la interpolación de dos puntos

En la propuesta de solución descrita se definió que para poder calcular la trayectoria del balón era necesario calcular la interpolación entre dos puntos. Se define como interpolación a los valores medios evaluados dentro de una función que satisface a una serie de puntos representados de manera discreta (58).

Según la fórmula escogida de la librería Mathf, Lerp se basa en un número de coma flotante x, la cual devuelve la interpolación entre a y b. Aplicándolo a la solución de problema sería x= Mathf.Lerp(a, b, t) donde a es el valor mínimo de fuerza aplicada sobre la pelota, b es el valor máximo aplicado sobre la misma y t es el incremento desde a hasta b, dando como resultado x que sería la fuerza con la que se le da a la pelota que luego actuaría con el peso de la misma dándole un efecto.

```
ballForce = Mathf.Lerp(ballForceMin, ballForceMax, currentTime / totalTimer);
```

Ilustración 1:Utilización de la fórmula en el código

Estándares de codificación

En el proceso de desarrollo de un software siguiendo la metodología XP, es necesario que exista una adecuada comunicación entre los programadores del equipo de desarrollo. Para lograr estos estándares se establecen un conjunto de los mismos bien definidos por las convenciones de escritura de código C# (59).

Entre los estándares de codificación de C# se encuentran:

- Los nombres de clases y métodos siempre deben estar en Pascal Case.

```
public class Employee
{
```

```
    public Employee GetDetails ()
    {
        //...
    }
    public double GetBonus ()
    {
        //...
    }
}
```

- El argumento del método y las variables locales siempre deben estar en Camel Case.

```
public class Employee
{
    public void PrintDetails (int employeeId, String firstName)
    {
        int totalSalary = 2000;
        // ...
    }
}
```

- Evitar el uso de guiones bajos al nombrar identificadores.

```
// Correct
public DateTime fromDate;
public String firstName;


// Avoid
public DateTime from_Date;
public String first_Name;
```

- Evitar el uso de tipos de datos del sistema y prefiera usar los tipos de datos predefinidos.

```
// Correct
int employeeId;
string employeeName;
bool isActive;


// Avoid
Int32 employeeId;
String employeeName;
Boolean isActive;
```

- Siempre anteponer una interfaz con la letra I.

```
// Correct
public interface IEmployee
{
}
public interface IShape
{
}
public interface IAnimal
{
}

// Avoid
```

```
public interface Employee
{
}
public interface Shape
{
}
public interface Animal
{
}
```

- Para una mejor sangría y legibilidad del código, siempre alinear las llaves verticales.

```
// Correct
class Employee
{
    static void PrintDetails ()
    {
    }
}


// Avoid
class Employee
    {
    static void PrintDetails ()
    {
      }
}
```

- Utilizar siempre la palabra clave using cuando trabaje con tipos desechables. Eliminar automáticamente el objeto cuando el flujo del programa abandone el ámbito.

```
Using (var conn = new SqlConnection(connectionString))
{
    // use the connection and the stream
    using (var dr = cmd.ExecuteReader())
    {
     //
    }
}
```

- Declarar siempre el uso de las variables lo más cerca posible de su uso.

```
// Correct
String firstName = "Shubham";
Console.WriteLine(firstName);
//--------------------------


// Avoid
String firstName = "Shubham";
//--------------------------
//--------------------------
//--------------------------
Console.WriteLine(firstName);
```

- Declarar siempre las propiedades como privadas para lograr la encapsulación y garantizar la ocultación de datos.

```
// Correct
private int employeeId {get; set;}

// Avoid
public int employeeId {get; set;}
```

- Separar siempre los métodos, las diferentes secciones del programa en un espacio.

```
// Correct
class Employee
{
private int employeeId {get; set;}

public void PrintDetails ()
{
//------------
}
}

// Avoid
class Employee
{
private int employeeId {get; set;}


public void PrintDetails ()
{
//------------
}
}
```

- Las constantes siempre deben declararse en mayúsculas.

```
// Correct
public const int MIN_AGE = 18;
public const int MAX_AGE = 60;

// Avoid
public const int Min_Age = 18;
public const int Max_Age = 60;
```

Fase 4: Pruebas

Las pruebas de software (en inglés software testing) forman parte de la última fase que propone XP, con el objetivo de lograr una herramienta que cumpla con los requisitos previamente identificados. En el presente subíndice se describe la etapa de prueba de la propuesta de solución.

Las pruebas de software son las investigaciones empíricas y técnicas cuya meta es proporcionar información objetiva e independiente sobre la calidad del producto a la

parte interesada o Stakeholders. Es una actividad más en el proceso de control de calidad.

Consisten en la dinámica de la verificación del comportamiento de una aplicación en un conjunto finito de casos de prueba, debidamente seleccionados de por lo general infinitas ejecuciones de dominio, contra la del comportamiento esperado.

Son un conjunto de actividades que se ejecutan con el propósito de encontrar los potenciales fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo.

Por lo general, las pruebas involucran operaciones del sistema evaluando los resultados bajo condiciones controladas, lo que hace que la realización de pruebas al software sea un factor de vital importancia (10). En busca de comprobar el estado de la calidad del software, se siguen las siguientes etapas:

- Verificación de la interacción de componentes.
- Verificación de la integración adecuada de los componentes.
- Verificación de que todos los requisitos se han implementado correctamente.
- Identificación de los defectos encontrados y aseguramiento de su corrección antes de entregar el software.
- Diseño de pruebas que sistemáticamente identifiquen diferentes clases de errores, con la menor cantidad de tiempo y esfuerzo.

Estrategia de pruebas

La Metodología XP propone que las pruebas de software sean realizadas al término de cada iteración, garantizando el funcionamiento deseado y la aceptabilidad por el cliente para realizar una entrega funcional y acorde a las exigencias de un producto con calidad.

Las dos pruebas exigidas por la metodología, por su importancia y agilidad en el proceso; son las pruebas unitarias y de aceptación. Se hicieron las pruebas unitarias al código al finalizar cada iteración e igualmente se realizaron las pruebas de aceptación.

Pruebas unitarias

Las pruebas unitarias (o unit testing) son pruebas de caja blanca donde los componentes individuales del software se someten a pruebas. El propósito de estas es asegurar que cada unidad de trabajo funciona de forma correcta individualmente,

responde como se espera que deba responder, o falle como y cuando se supone que debe fallar. (61)

Se estima que las pruebas unitarias deben probar la unidad mínima de trabajo de un programa que es aquella que devuelve un valor o produce un cambio en el estado del programa, generalmente hablaríamos de un solo método o una función.

A diferencia de las pruebas de caja negra, las pruebas de caja blanca se ven como un cuadro blanco, o de vidrio, que permite ver la estructura y el flujo del software bajo prueba. Los planes de pruebas se hacen de acuerdo a los detalles de la implementación del software, tales como lenguaje de programación, la lógica y estilos. Los casos de prueba se derivan de la estructura del programa (10).

Se puede definir el unit testing como el aislamiento de una parte del código para comprobar que funciona sin problemas, se trata de pequeños test que tiene el propósito de validar el comportamiento de la lógica y de un objeto (60).

Estas pruebas se basan en el minucioso examen de los detalles procedimentales, donde se examinan los caminos lógicos del software proponiendo casos de prueba que examinen que todas las condiciones y/o bucles estén correctas para escoger si el estado real coincide con el esperado o afirmado (61).

Los principales puntos de estas pruebas son:

- Garantizar que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- Ejercitar todas las decisiones lógicas en las vertientes verdadera y falsa.
- Ejecutar todos los bucles en sus límites operacionales.
- Ejercitar las estructuras internas de datos para asegurar su validez.

Debido a la disminución en un alto por ciento del número de errores existentes en los sistemas gracias a la aplicación de las pruebas de caja blanca, estas son consideradas como uno de los tipos de pruebas más importantes, por mejorar la calidad y confiabilidad del software analizado. Para ello existen diferentes técnicas entre las que se encuentra la del camino básico, a continuación, se muestra un ejemplo de la aplicación de esta técnica.

Para aplicar la técnica de camino básico se debe introducir una sencilla notación para la representación del flujo de control, el cual puede representar por un grafo de flujo, donde cada nodo del grafo corresponde a una o más sentencias de código. (63)

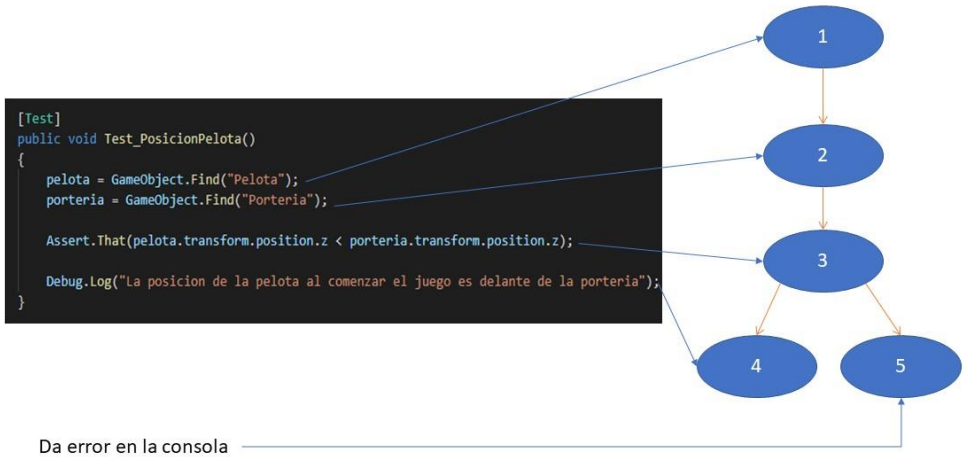
Luego de realizar la prueba de caja blanca mediante la prueba del camino básico a la funcionalidad detener o desviar el balón lanzado por el jugador se determinó que la complejidad ciclomática era 3 por lo que el método es sencillo y de poco riesgo para el sistema (61).

v(G) = e – n + 2, donde e representa el número de aristas y n el número de nodos.

e = 4 n = 5

v(G) = número de regiones cerradas en el grafo + 1

Tabla 17 Camino básico



Posibles caminos:

1,2,3,4

1,2,3,5

Para el desarrollo de las pruebas unitarias se utilizó la misma herramienta que Unity, esta brinda la posibilidad de crear pruebas unitarias, en la cual se utiliza las aserciones para comprobar que se cumpla una condición.

En la siguiente figura se muestra un ejemplo de caso de prueba unitaria realizada al procedimiento PosicionPelota de la clase Pelota, el cual en un primer momento error, ya que la pelota no estaba en la posición inicial. Dicho error fue solucionado en un segundo momento. En la sección Anexo 4 Pruebas unitarias se pueden encontrar otros ejemplos de estas pruebas, realizadas a algunos de los principales procedimientos de la solución.

Tabla 18 Caso de prueba unitaria Test_PosicionPelota.

Caso de prueba unitaria	Resultado

`[Test]`
`public void Test_PosicionPelota()`
`{`
`  pelota = GameObject.Find("Pelota");`
`  porteria = GameObject.Find("Porteria");`
`  Assert.That(pelota.transform.position.z < porteria.transform.position.z);`
`  Debug.Log("La posicion de la pelota al comenzar el juego es delante de la porteria");`
`}`
	

Pruebas de aceptación

Las pruebas de aceptación son pruebas de caja negra definidas por el cliente para cada historia de usuario, y tienen como objetivo asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas. En efecto, las pruebas de aceptación corresponden a una especie de documento de requerimientos en XP, ya que marcan el camino a seguir en cada iteración, indicándole al equipo de desarrollo hacia donde tiene que ir y en qué puntos o funcionalidades debe poner el mayor esfuerzo y atención (10).

Las pruebas de caja negra son un método de ensayo en el que los datos de prueba se derivan de los requisitos funcionales especificados sin tener en cuenta la estructura final del programa. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos (61).

Las pruebas de aceptación son más importantes que las pruebas unitarias dado que significan la satisfacción del cliente con el producto desarrollada y estas son creadas a partir de las Historias de Usuario.

Mientras más amplio sea el espectro de elementos de entrada para realizar la prueba, las probabilidades de encontrar problemas en el software aumentan y, por lo tanto, será más confiable la calidad del software.

Este tipo de pruebas permitió encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las bases de datos externas.
- Errores de rendimiento.

- Errores de inicialización y terminación.

Para preparar los casos de prueba es necesario un número de datos que ayuden a la ejecución de estos casos y que permitan que el sistema se ejecute en todas sus variantes. Estos datos pueden ser válidos o no para el software y son seleccionados atendiendo a las especificidades de la funcionalidad a probar.

Enfoque de prueba seleccionado

En la realización de las pruebas realizadas al nuevo videojuego de simulación de penales de futbol se decide utilizar el enfoque de caja negra debido a que permite obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales del sistema. En las pruebas a realizar se pretende comprobar el funcionamiento del componente modificando los factores externos con los que trabaja y validar que los resultados obtenidos son los esperados.

Las pruebas de caja negra no son una alternativa a las técnicas de prueba de caja blanca. Son un enfoque complementario. Se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación.

Los casos de prueba de la caja negra pretenden demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

- Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- Técnica del Análisis de Valores Límites: esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Se selecciona la Técnica de Partición de Equivalencia para realizar las pruebas al videojuego de simulación de penales de futbol porque es la que más se ajusta a la forma en que se pretende ejercitar las funcionalidades añadidas mediante diferentes entradas, y comprobar el comportamiento del sistema con las mismas.

La partición equivalente es un método de prueba de la caja negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de

prueba. Un caso de prueba ideal descubre de forma inmediata una clase de error que de otro modo requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (65).

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Típicamente una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana (si o no). Las clases de equivalencia se pueden definir de acuerdo a las siguientes directrices:

- Si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos inválidas.
- Si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos inválidas.
- Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una inválida.
- Si una condición de entrada es booleana, se define una clase válida y una inválida.

Casos de prueba basados en Historias de Usuario

La confirmación de un sistema de software es un proceso continuo en cada etapa del ciclo de vida del software. Los casos de prueba no son más que un conjunto de condiciones o variables bajo las cuáles el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio. Debido a que en el la solución se emplea una gran cantidad de procedimientos encargados de mostrar algún contenido visual o contribuir a esto y también a que son pocas las funcionalidades que se utilizan, se decidió realizar las pruebas unitarias programadas de manera manual para adaptarlas a dicha situación y verificar si el contenido visual de este se muestra correctamente. Dichas pruebas se realizaron en cada una de las iteraciones definidas, para evitar el arrastre de errores lógicos a iteraciones posteriores.

Eliminar los errores que representa el código no es suficiente para garantizar que la solución cumple con las especificaciones del cliente, esto trajo como consecuencia realizar la ejecución de las pruebas de aceptación. A continuación, aparece un ejemplo de las pruebas de aceptación realizadas a la solución, las restantes se encuentran en la sección Anexo 5 Pruebas de aceptación.

Tabla 19 Caso de prueba de aceptación P1HU1

Caso de Prueba de Aceptación	
Código: P1HU1	**Número de Historia de Usuario**: 1
Historia de usuario: Renderizar la escena de simulación de futbol.	
Condiciones de ejecución: Se debe crear los objetos anteriormente en Blender y luego exportarlo hacia Unity.	
Entrada/Pasos para la ejecución: El usuario al entrar en el juego debe ver los objetos con los que pueda interactuar.	
Resultado esperado: En la interfaz del juego deben aparecer todos los objetos con los que se pueda, de forma directa o indirecta, interactuar.	
Evaluación de la prueba: Resultado satisfactorio.	

A continuación, se muestran algunos conjuntos de datos que evidencian el comportamiento del videojuego de simulación de penales de futbol en los escenarios de prueba a los que se sometió el sistema. En la siguiente tabla se muestran los datos obtenidos por la funcionalidad que captura los cinco procesos que más CPU están consumiendo.

Tabla 20 Procesos que consumen más CPU

No.	Memoria	Proceso	Identificador	CPU
1	390 MB	Unity	9684	0.3%
2	6 MB	Unity Sharder	2936	0.1%
3	5 MB	C Sharp	11457	0.1%
4	118 MB	Host servicio: Servicio de usuario de difusión	9692	1.2%

5	300 MB	Visual Studio Code	3156	0.4%

Resultado de las pruebas

Con el objetivo de verificar que todos los requisitos funcionales fueron desarrollados correctamente se aplicaron pruebas específicas por cada una de las Historias de Usuario. La siguiente figura muestra el número de Casos de Prueba (CP) realizados, las No Conformidades (NC) detectadas y las No Conformidades Resueltas (NCR) durante las tres iteraciones realizadas. En la primera iteración se detectó una No Conformidad que fueron solucionadas completamente, durante la segunda iteración fue detectada una No Conformidad que fue debidamente solucionada y en la última iteración se detectó una No Conformidad que fue solucionada completamente dando un total de tres No Conformidades.

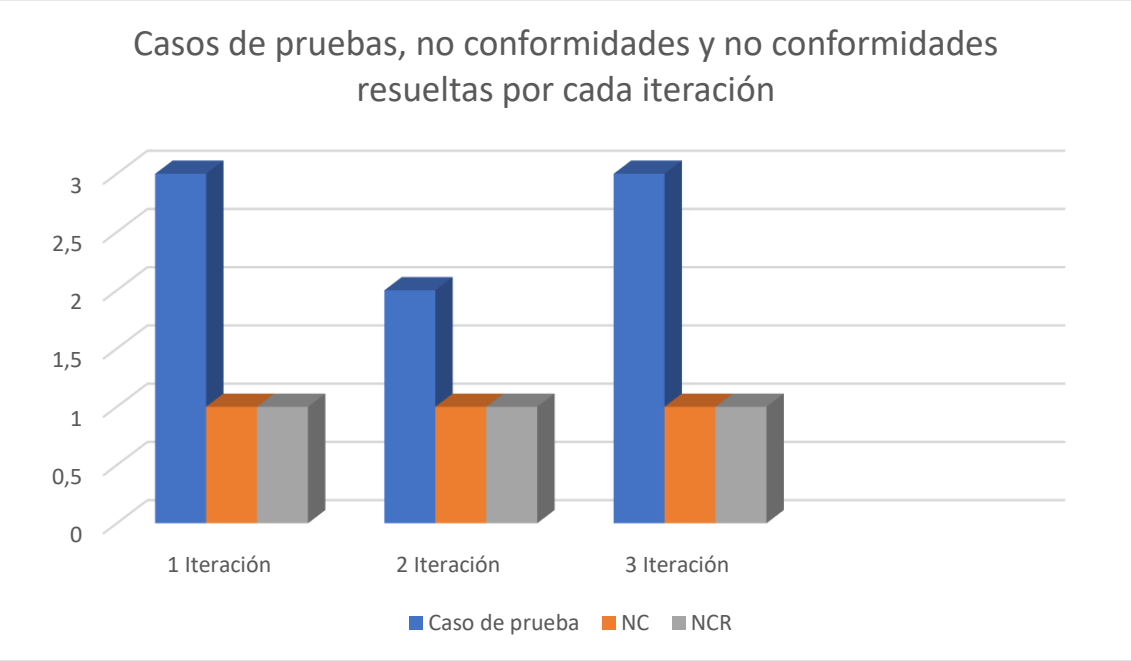


Ilustración 2 CS, NC y NCR

La siguiente tabla muestra la No Conformidad detectada en la primera iteración y su respectiva respuesta por parte del desarrollador. Las demás No Conformidades detectadas se encuentran en la sección Anexo 3 No Conformidades.

Tabla 21 No Conformidades detectadas en la primera iteración

Elemento	No.	No Conformidad	Aspecto Correspondiente	Etapa de detección	Clasificación	Estado NC	Respuesta del equipo de desarrollo
Aplicación	1	Se detectó un error ortográfico en la escena principal	Error ortográfico	Prueba	N	PD:01/05/2022 RA:02/05/2022	Se arreglo el error ortográfico encontrado en la escena principal.

Conclusiones del capítulo

Los resultados de estas pruebas a la plataforma demostraron su efectividad para ser desplegada.

Los diferentes métodos de pruebas aplicados a la solución durante el ciclo de vida del software permitieron comprobar los errores existentes y mejorar la calidad de los resultados.

Con el objetivo de verificar el correcto comportamiento y la estabilidad de las nuevas funcionalidades del videojuego de simulación de penales de futbol fueron aplicadas una serie de pruebas funcionales capaces de determinar si la nueva versión del componente ofrece los resultados que se esperaban.

Se aplicaron un total de ocho casos de pruebas durante las que se detectaron tres No Conformidades: Una en la primera iteración, una en la segunda iteración y una en la tercera iteración.

La realización de las pruebas permitió confirmar el correcto funcionamiento del nuevo componente, así como su estabilidad en diferentes escenarios, ofreciendo los resultados esperados.

Conclusiones generales

Considerando los resultados descritos en este informe, la necesidad y el objetivo planteado por la investigación se arriban a las siguientes conclusiones:

- El análisis de las diferentes fuentes, herramientas y soluciones tecnológicas para la simulación de videojuego de penales de futbol permitió identificar las características claves que distinguen la plataforma propuesta y su adaptación al contexto nacional.
- El empleo de las herramientas y tecnologías seleccionadas para la implementación de la solución propició la correspondencia entre los resultados obtenidos y los esperados, lo cual pudo asegurar el nivel de precisión en el análisis y diseño de la aplicación.
- La realización de pruebas unitarias y pruebas de aceptación permitió comprobar el correcto funcionamiento del componente ante todos los escenarios posibles. Estas pruebas arrojaron resultados satisfactorios en relación al código y el conjunto de interfaces implementadas.
- Los resultados de la investigación que describe este informe demostraron la viabilidad de la solución propuesta para la simulación de videojuego de penales de futbol, cuyo desarrollo contribuye a la visibilidad y posicionamiento de la producción de aplicaciones hechos en cuba.
- El empleo de la metodología XP permitió que se trazara una buena planificación de todo el proyecto, ahorrando tiempo y recursos.
- Se desarrolló un sistema de simulación de penales de futbol que permite aumentar la eficiencia de los equipos de balompié que lo utilicen mediante el uso de las estadísticas según las áreas de la portería.

Recomendaciones

Con el objetivo de mejorar el funcionamiento y utilidad de la simulación de videojuego de penales de futbol se recomienda:

- Integrar la aplicación con algún videojuego de futbol que pueda surgir más adelante.
- Mejorar aspectos gráficos.
- Desarrollar mejores métricas para el estudio de los resultados obtenidos.

Glosario de Términos

Biblioteca: en ciencias de la computación, una biblioteca es un conjunto de implementaciones funcionales y codificación en un lenguaje de programación que se emplean para desarrollar software.

Lenguaje de modelado: se entiende por lenguaje de modelado cualquier lenguaje artificial que puede ser utilizado para expresar la información, el conocimiento o sistemas en una estructura que está definida por un conjunto coherente de reglas.

Lenguaje de programación: un lenguaje de programación es un conjunto de reglas, notaciones, símbolos y/o caracteres que permiten a un programador poder expresar el procesamiento de datos y sus estructuras en la computadora. Cada lenguaje posee sus propias sintaxis. También se puede decir que un programa es un conjunto de órdenes o instrucciones que resuelven un problema específico basado en un lenguaje de programación

Métricas: una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.

Plugin: es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica.

Plataforma: en informática, una plataforma es un sistema que sirve como base para hacer funcionar determinados módulos de hardware o de software con los que es compatible. Dicho sistema está definido por un estándar alrededor del cual se determina una arquitectura de hardware y una plataforma de software (incluyendo entornos de aplicaciones). Al definir plataformas se establecen los tipos de arquitectura, sistema operativo, lenguaje de programación o interfaz de usuario compatibles.

Renderizar: es un término usado para referirse al proceso de generar una imagen desde un modelo 3D.

Referencias Bibliográficas

1. **Westreicher, Guillermo.** Industria del entretenimiento. [En línea] 10 de Mars de 2022. https://economipedia.com/definiciones/industria-del-entretenimiento.html.

2. **Pérez, Julián y Gardey, Ana.** *Definicion.de.* [En línea] 2010. https://definicion.de/videojuego/.

3. **Claro.** Tipos de videojuegos: cómo se dividen. [En línea] 17 de November de 2021. https://www.claro.com.co/institucional/tipos-de-videojuegos/.

4. **Vandal.** Ránking de los mejores videojuegos. [En línea] https://vandal.elespanol.com/rankings/videojuegos.

5. **Dayan, Mariano.** La población mundial juega videojuegos. [En línea] 6 de May de 2022. https://www.ole.com.ar/esports/poblacion-mundial-videojuegos_0_Cr91uwT9-.html.

6. **Ramírez, Piedad.** Importancia de los videojuegos para formar personalidad e inteligencia. [En línea] 11 de January de 2017. https://itsoftware.com.co/content/importancia-de-los-videojuegos/.

7. **Lombera Rodríguez, Hassán.** Centro de Tecnologías Interactivas. [En línea] https://www.uci.cu/investigacion-y-desarrollo/centros-de-desarrollo/centro-de-tecnologias-interactivas.

8. **VERTEX.** COSMOX. [En línea] 2022. https://cosmox.uci.cu/.

9. **Sol, Del y otros.** Los métodos teóricos: una necesidad de conocimiento en la investigación científico-pedagógica. [En línea] 3 de February de 2017. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2077-28742017000400021.

10. **Escalona Mayo, Ramón Emanuel y Lima Mateo, Luis Ruben.** Plataforma SOFIA para la distribución electrónica de libros digitales en Cub. La Habana, La Habana, Cuba : s.n., June de 2019.

11. **QuestionPro.** Métodos de investigación: Qué son y cómo elegirlos. [En línea] https://www.questionpro.com/blog/es/metodos-de-investigacion/.

12. *Introducción a la simulación y a la generación de números pseudoaleatorios.* **Dpto. Computación.** 2011, Introducción a la simulación y a la generación de números pseudoaleatorios, pág. 8.

13. **Shannon, Robert E. y otros.** *Systems Simulation: The Art and Science.* s.l. : Prentice Hall, 1976.

14. *Simulacion Digital.* **López Octaviano, Marco Antonio.** México : Jocotitlan Edo de México, 2008.

15. **Benjamin Ruiz, Tadep.** Clasificación de los modelos de simulación. *Clasificación de los modelos de simulación.* [En línea] 04 de febrero de 2012. [Citado el: 24 de agosto de 2022.] https://sites.google.com/site/benjaminruiztadeo/home.

16. **Baer, Ralph H., Rusch, William T. y Harrison, William L.** TELEVISION GAMING APPARATUS AND METHOD. [En línea] 25 de april de 1972. https://www.freepatentsonline.com/3659285.html.

17. **Tokio School.** ¿Cómo diseñar un videojuego? Descubre los pasos. *¿Cómo diseñar un videojuego? Descubre los pasos.* [En línea] 26 de Mayo de 2021. [Citado el: 24 de Agosto de 2022.] https://www.tokioschool.com/noticias/como-disenar-videojuego/.

18. **Significados.** Diseño. [En línea] 31 de May de 2022. https://www.significados.com/diseno/.

19. **On, Mahm.** ¡Y acción! Estudio de física en videojuegos. *¡Y acción! Estudio de física en videojuegos.* [En línea] Noticias Técnicas. [Citado el: 9 de Septiembre de 2022.] https://es.comsumersforinternetcompetition.com/article/1960-video-game-physics/.

20. **Konami.** eFootball. *eFootball.* [En línea] 2022. https://www.konami.com/efootball/es/page/gameoverview.

21. **EA Sports.** EA Sports FIFA. *EA Sports FIFA.* [En línea] 2022. https://www.ea.com/es-es/games/fifa.

22. **Fernández, Yúbal.** Mini Football: qué es, qué ofrece y cómo se juega . [En línea] 29 de September de 2020. https://www.xataka.com/basics/mini-football-que-que-ofrece-como-se-juega.

23. **Martínez, Carlos.** Mini Football. *Mini Football.* [En línea] 4 de Juny de 2022. https://miniclip-minifootball.uptodown.com.

24. **Difadi.com.** Simulador de penales. *Simulador de penales.* [En línea] 2018. [Citado el: 30 de Agosto de 2022.] https://www.simuladoresvirtuales.com/simulacion-vitual/simulador-de-penaltis/.

25. **Letelier, Patricio y Penadés, María Carmen.** Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [En línea] 15 de Decembre de 2005. http://www.cyta.com.ar/ta0502/v5n2a1.htm.

26. **Alarcón Barbán, Enier.** Una Guía para conocer XP. Boyeros, La Habana, Cuba : s.n.

27. **AUTODESK.** PROGRAMA DE DISEÑO 3D. [En línea] https://www.autodesk.mx/solutions/3d-design-software.

28. **Blender.** Blender. *Blender.* [En línea] 2022. https://www.blender.org.

29. **Unreal Engine.** Unreal Engine. *Unreal Engine.* [En línea] 2022. https://www.unrealengine.com.

30. **maxon.** Cinema 4D. *Cinema 4D.* [En línea] 2022. https://www.maxon.net/cinema-4d.

31. **pixologic.** Zbrush. *Zbrush.* [En línea] May de 2022. https://pixologic.com.

32. **Autodesk.** Maya. *Maya.* [En línea] 2022. https://www.autodesk.com/overview.

33. **autodesk.** 3ds Max. *3ds Max.* [En línea] 2022. https://autodesk.es/overview.

34. **McNeel, Robert.** Rhino. *Rhino.* [En línea] 2022. https://www.rhino3d.com.

35. **Unity.** Unity 3D. *Unity 3D.* [En línea] 2022. https://unity.com/es.

36. **Scratch.** Scratch. *Scratch.* [En línea] 2022. https://scratch.mit.edu.

37. **M.U.G.E.N.** mugen. *mugen.* [En línea] 2022. https://mugen.softonic.com.

38. **roblox.** roblox. *roblox.* [En línea] 2022. https://www.roblox.com.

39. **Stencyl.** Stencyl. *Stencyl.* [En línea] 2021. https://www.stencyl.com.

40. **Linietsky, Juan y Otros.** Godot. *Godot.* [En línea] 2022. https://godotengine.org.

41. **CILSA.** ¿Qué es un lenguaje de programación? [En línea] 2017. https://desarrollarinclusion.cilsa.org/tecnologia-inclusiva/que-es-un-lenguaje-de-programacion/.

42. **Lucidchart.** ¿Para qué necesitas crear un diagrama UML? [En línea] https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml.

43. **Felipe.** Concepto de IDE y cuáles son sus características. [En línea] 3 de September de 2021. https://www.hostingplus.pe/blog/concepto-de-ide-y-cuales-son-sus-caracteristicas/.

44. **Microsoft.** Visual Studio Code. *Visual Studio Code.* [En línea] 2022. https://code.visualstudio.com.

45. **aCute, Eclipse.** Eclipse aCute. *Eclipse aCute.* [En línea] 2022. https://projects.eclipse.org.

46. **JetBrains.** Rider. *Rider.* [En línea] 2022. https://www.jetbrains.com.

47. **Developer Express.** DevExpress. *DevExpress.* [En línea] 2022. https://www.devexpress.com.

48. **Codeanywhere.** Codeanywhere es un IDE en la nube. *Codeanywherees un IDE en la nube.* [En línea] 2022. https://codeanywhere.com.

49. **García Muñoz, Carlos.** *Física General.* Santiago Burbano de Ercilla : Editorial Tébar, 2003. ISBN 8495447827.

50. **Agusto, Raúl Monferre.** Especificación de Requisitos Software según el estándar de IEEE 830. Castelló de la Plana, Castelló, España : s.n., 2001.

51. **Mesias, Fernando Torres.** INGENIERÍA DE REQUERIMIENTOS: MÉTODOS Y TÉCNICAS. Aguascalientes, Aguascalientes, México : s.n., 2014.

52. **PMOinformatica.** ¿Qué es un requerimiento funcional? . [En línea] 30 de May de 2018. http://www.pmoinformatica.com/2018/05/que-es-requerimiento-funcional.html.

53. **Quiroga, Juan Pablo.** Requerimientos No Funcionales. [En línea] 2015. https://slideplayer.es/slide/2617509/.

54. **Blog, Requeridos.** Requerimientos Funcionales y No Funcionales, ejemplos y tips. [En línea] 20 de April de 2018. https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a.

55. **PMOinformatica.** Requerimientos no funcionales: Una clasificación. [En línea] 13 de April de 2015. http://www.pmoinformatica.com/2015/04/requerimientos-no-funcionales-una.html.

56. **Andalucía.** Plantilla Plan de Entregas. [En línea] http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/451.

57. **Peláez, Juan.** Arquitectura basada en capas. [En línea] 30 de May de 2009. https://geeks.ms/jkpelaez/2009/05/30/arquitectura-basada-en-capas/.

58. —. Arquitectura basada en Componentes. [En línea] 18 de April de 2009. https://geeks.ms/jkpelaez/2009/04/18/arquitectura-basada-en-componentes/.

59. **Blancarte, Oscar.** Introducción a los patrones de diseño: Un enfoque práctico(Spanish Edition). 18 de October de 2016.

60. **Acervo Lima.** Estándares de codificación C#. [En línea] https://es.acervolima.com/estandares-de-codificacion-c/.

61. **Alvarado, Illya.** LA IMPORTANCIA DE LAS PRUEBAS UNITARIAS PARA COMPROBAR FRAGMENTOS DE CÓDIGO. [En línea] 31 de Mars de 2020. https://ceroideas.es/la-importancia-de-las-pruebas-unitarias-para-comprobar-fragmentos-de-codigo/.

62. **Pressman, Roger S.** *Ingeniería del software - Un enfoque práctico. Séptima edición.* Mexico : McGraw-Hil, 2010.

63. **Braude, Eric J.** *Ingeniería de software, una perspectiva orientada a objetos.* s.l. : Alfaomega Grupo Editor, 2003.

64. **G. Piattini, Mario, y otros.** *Análisis y diseño de aplicaciones informáticas de gestión, una perspectiva de ingeniería del software.* s.l. : Alfaomega, 2004.

65. **Patton, Ron.** *Software Testing.* s.l. : Sams Publishing, 2005.

66. **Pressman, Roger S.** *Ingeniería de Software Un Enfoque Práctico.* s.l. : McGraw-Hill, 2005.

67. **Significado.** Método empírico. [En línea] https://www.significados.com/metodo-empirico/.

68. **H. Bravo, Andres.** [En línea] https://gamedevtraum.com/es/programacion-informatica/teoria-de-programacion/que-es-script-programacion/.

69. **Ertheo.** Las mejores aplicaciones informáticas para entrenadores de fútbol y para gestionar plantillas. [En línea] 12 de Decembre de 2018. https://www.ertheo.com/blog/aplicaciones-informaticas-entrenadores-futbol/.

70. **Microsoft.** Code editing. Redefined. [En línea] 2022. https://code.visualstudio.com/.

71. **Adrián, Yirda.** Marco Teórico. [En línea] 22 de July de 2021. https://conceptodefinicion.de/marco-teorico/.

72. **Etecé.** ¿Qué es un marco teórico? [En línea] 8 de February de 2022. https://concepto.de/marco-teorico/.

73. **Martin, Sara.** Las ventajas de crear prototipos para tu proyecto digital. [En línea] 10 de August de 2020. https://www.hiberus.com/crecemos-contigo/las-ventajas-de-crear-prototipos-para-tu-proyecto-digital/.

74. **López, Andrés.** Score! Hero. *Score! Hero.* [En línea] 7 de April de 2021. https://score-hero.uptodown.com/android.

75. **Rosso, Raúl.** Captain Tsubasa: Dream Team. *Captain Tsubasa: Dream Team.* [En línea] 3 de Juny de 2022. https://captain-tsubasa-fight-dream-team.uptodown.com.

76. **López, Andres.** Top Eleven 2020. *Top Eleven 2020.* [En línea] 7 de Juny de 2022. https://eleven-manager-de-futbol.uptodown.com.

77. *Simulacion Digital.* **López Octaviano, Marco Antonio.** México : Jocotitlan Edo de México, 2008.

Anexos

Anexo 1 Historias de usuario

Tabla 22 Historia de usuario Renderizar la escena de simulación de futbol

<table>
<tr><th colspan="2">Historia de usuario</th></tr>
<tr><td>Número: 2</td><td>Nombre del requisito: Renderizar la escena de simulación de futbol.</td></tr>
<tr><td>Programador: Sergio Daniel Ortiz Jova</td><td>Iteración asignada: 1</td></tr>
<tr><td>Prioridad: Alta</td><td>Tiempo estimado: 30 días</td></tr>
<tr><td>Riesgo en desarrollo: N/A</td><td>Tiempo real: 29 días</td></tr>
<tr><td colspan="2">Descripción:
Deberá permitir renderizar todas las escenas de simulación de futbol, así como, los objetos que aparecerán en ella.</td></tr>
<tr><td colspan="2">Observaciones: A continuación, se podrá observar dos ejemplos de la renderización de la simulación:
 


</td></tr>
</table>

Tabla 23 Historia de usuario Calcular la trayectoria del balón según la posición y la fuerza.

Historia de usuario	
Número: 3	**Nombre del requisito:** Calcular la trayectoria del balón según la posición y la fuerza.
Programador: Sergio Daniel Ortiz Jova	**Iteración asignada:** 2
Prioridad: Media	**Tiempo estimado:** 18 días
Riesgo en desarrollo: N/A	**Tiempo real:** 19 días
Descripción: Deberá permitir mediante las fórmulas propuestas en la reunión, calcular la trayectoria del balón según la posición entrada por el mouse y la fuerza que se seleccionó.	
Observaciones: N/A	

Tabla 24 Historia de usuario Animar los avatares de la simulación del juego.

Historia de usuario	
Número: 4	**Nombre del requisito:** Animar los avatares de la simulación del juego.
Programador: Sergio Daniel Ortiz Jova	**Iteración asignada:** 3
Prioridad: Alta	**Tiempo estimado:** 12 días
Riesgo en desarrollo: N/A	**Tiempo real:** 14 días
Descripción: El videojuego deberá mostrar todas las animaciones realizadas en Blender y exportadas a Unity, cuando se llamen a cada una de ellas.	

Observaciones: N/A

Tabla 25 Historia de usuario Detener o desviar el balón lanzado por un jugador.

Historia de usuario	
Número: 5	**Nombre del requisito:** Detener o desviar el balón lanzado por un jugador.
Programador: Sergio Daniel Ortiz Jova	**Iteración asignada:** 3
Prioridad: Alta	**Tiempo estimado:** 8 días
Riesgo en desarrollo: N/A	**Tiempo real:** 8 días
Descripción: Deberá permitir al portero detener o desviar el balón lanzado por el jugador mediante el uso de métricas que lo ayudaran a decir para que lugar tirarse.	
Observaciones: N/A	

Tabla 26 Historia de usuario Seleccionar efecto del balón de futbol.

Historia de usuario	
Número: 6	**Nombre del requisito:** Seleccionar efecto del balón de futbol.
Programador: Sergio Daniel Ortiz Jova	**Iteración asignada:** 3
Prioridad: Media	**Tiempo estimado:** 10 días
Riesgo en desarrollo: N/A	**Tiempo real:** 11 días
Descripción: Deberá permitir seleccionar el efecto del balón de futbol con que el jugador disparará hacia la portería.	

Observaciones: N/A

Tabla 27 Historia de usuario Mostrar las estadísticas del cálculo de la simulación.

Historia de usuario	
Número: 7	**Nombre del requisito:** Seleccionar efecto del balón de futbol.
Programador: Sergio Daniel Ortiz Jova	**Iteración asignada:** 3
Prioridad: Media	**Tiempo estimado:** 10 días
Riesgo en desarrollo: N/A	**Tiempo real:** 11 días
Descripción: Deberá permitir seleccionar el efecto del balón de futbol con que el jugador disparará hacia la portería.	
Observaciones: N/A	

Anexo 2 Tarjeta CRC

Tabla 28 Tarjeta CRC Player

Clase: Player	
Responsabilidad: Update: Espera a que el jugador de click en el escenario, luego este va hacia la pelota y le da con la fuerza que dejo presionada el usuario, ejecutando las animaciones correspondientes según cada caso.	Clase relacionada: Pelota Estadio

Tabla 29 Tarjeta CRC Portería

Clase: Portería	
Responsabilidad: Update: Se encarga de marcar el gol y llevar las estadísticas de la simulación.	Clase relacionada: Pelota Estadio

Tabla 30 Tarjeta CRC Portero

Clase: Portero	
Responsabilidad: Update: Trata de detener o desviar el balón que se dirige hacia la portería escogiendo uno de los lugares predefinidos de forma aleatoria (hacia arriba, abajo, derecha o izquierda)	Clase relacionada: Portero Estadio Pelota

Tabla 31 Tarjeta CRC Estadio

Clase: Estadio	
Responsabilidad: Update: Clase sobre la cual se mueven todas las demás clases.	Clase relacionada: Player Portería Portero Pelota

Anexo 3 No Conformidades

Tabla 32 No Conformidad 2

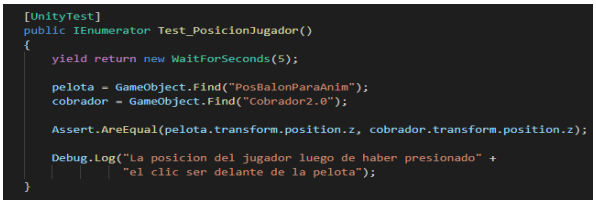
Elemento	No.	No Conformidad	Aspecto Correspondiente	Etapa de detección	Clasificación	Estado NC	Respuesta del equipo de desarrollo
Aplicación	2	Se detectó una animación faltante en el portero	Portero	Prueba	S	PD:25/05/2022 RA:28/05/2022	Se agregó la animación faltante al portero para cubrir la parte de la portería faltante.

Tabla 33 No Conformidad 3

Elemento	No.	No Conformidad	Aspecto Correspondiente	Etapa de detección	Clasificación	Estado NC	Respuesta del equipo de desarrollo
Aplicación	3	Se detectó un error en el cálculo de las métricas	Error de cálculo.	Prueba	N	PD:30/05/2022 RA:31/05/2022	Se le dio solución del cálculo de las métricas.

Anexo 4 Pruebas unitarias

Tabla 34 Caso de prueba unitaria Test_PosicionJugador

Caso de prueba unitaria	Resultado
	

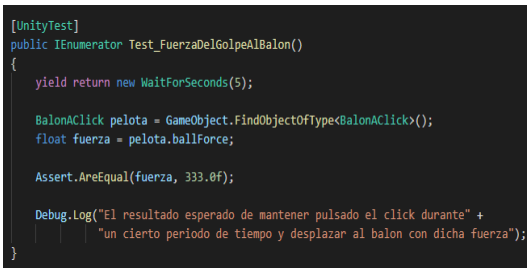
```
[UnityTest]
public IEnumerator Test_PosicionJugador()
{
    yield return new WaitForSeconds(5);

    pelota = GameObject.Find("PosBalonParaAnim");
    cobrador = GameObject.Find("Cobrador2.0");

    Assert.AreEqual(pelota.transform.position.z, cobrador.transform.position.z);

    Debug.Log("La posicion del jugador luego de haber presionado" +
              "el clic ser delante de la pelota");
}
```

Tabla 35 Caso de prueba unitaria Test_FuerzaDelGolpeAlBalon

Caso de prueba unitaria	Resultado
	

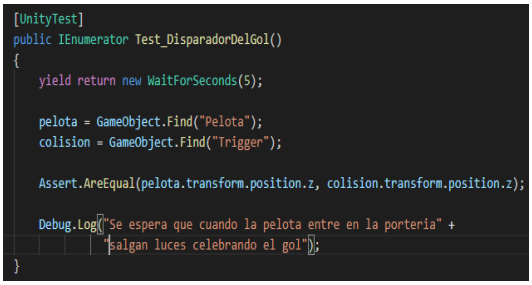
```
[UnityTest]
public IEnumerator Test_FuerzaDelGolpeAlBalon()
{
    yield return new WaitForSeconds(5);

    BalonAClick pelota = GameObject.FindObjectOfType<BalonAClick>();
    float fuerza = pelota.ballForce;

    Assert.AreEqual(fuerza, 333.0f);

    Debug.Log("El resultado esperado de mantener pulsado el click durante" +
              "un cierto periodo de tiempo y desplazar al balon con dicha fuerza");
}
```

Tabla 36 Caso de prueba unitaria Test_DisparadorDelGol

Caso de prueba unitaria	Resultado
	

```
[UnityTest]
public IEnumerator Test_DisparadorDelGol()
{
    yield return new WaitForSeconds(5);

    pelota = GameObject.Find("Pelota");
    colision = GameObject.Find("Trigger");

    Assert.AreEqual(pelota.transform.position.z, colision.transform.position.z);

    Debug.Log("Se espera que cuando la pelota entre en la porteria" +
              "salgan luces celebrando el gol");
}
```

Anexo 5: Pruebas de aceptación

Tabla 37 Caso de prueba de aceptación P2HU2

Caso de Prueba de Aceptación	
Código: P2HU2	**Número de Historia de Usuario**: 2
Historia de usuario: Seleccionar la posición en la que se trasladará el balón hacia la portería.	
Condiciones de ejecución: La pelota debe estar en la posición del campo predefinida.	
Entrada/Pasos para la ejecución: El usuario, mediante un click, debe seleccionar hacia donde se trasladará la pelota.	
Resultado esperado: La pelota deberá trasladarse a esa posición.	
Evaluación de la prueba: Resultado satisfactorio.	

Tabla 38 Caso de prueba de aceptación P3HU3

Caso de Prueba de Aceptación	
Código: P3HU3	**Número de Historia de Usuario**: 3
Historia de usuario: Calcular la trayectoria del balón según la posición y la fuerza.	
Condiciones de ejecución: La pelota y el cobrador debe estar en la posición del campo predefinida.	
Entrada/Pasos para la ejecución: El usuario, mediante un click, debe seleccionar hacia donde se trasladará la pelota y luego dejar el click presionado unos segundos para añadirle una fuerza al balón.	
Resultado esperado: La pelota deberá trasladarse a esa posición según la fuerza que le fue asignada.	
Evaluación de la prueba: Resultado satisfactorio.	

Tabla 39 Caso de prueba de aceptación P4HU4

Caso de Prueba de Aceptación	
Código: P4HU4	**Número de Historia de Usuario**: 4
Historia de usuario: Animar los avatares de la simulación del juego.	
Condiciones de ejecución: Los objetos animados deben tener la animación cargada en Unity.	
Entrada/Pasos para la ejecución: Al abrir la escena del juego los objetos animados deben tener la animación de inactivo, luego al usuario hacer click se debe de ejecutar la animación del caminar y después hacer la animación de la patada. El portero deberá de moverse según las estadísticas.	
Resultado esperado: Se ejecutan las animaciones según los diferentes casos que puedan existir en el escenario.	
Evaluación de la prueba: Resultado satisfactorio.	

Tabla 40 Caso de prueba de aceptación P5HU5

Caso de Prueba de Aceptación	
Código: P5HU5	**Número de Historia de Usuario**: 5
Historia de usuario: Detener o desviar el balón lanzado por un jugador.	
Condiciones de ejecución: El portero debe estar cargado y en la posición inicial predefinida.	
Entrada/Pasos para la ejecución: Al lanzar el balón, el portero debe ser capaz de detener o desviar el balón para que este no marque en la portería.	
Resultado esperado: El portero debe defender la portería tratando de que no le marquen gol.	
Evaluación de la prueba: Resultado satisfactorio.	

Tabla 41 Caso de prueba de aceptación P6HU5

Caso de Prueba de Aceptación	
Código: P6HU5	**Número de Historia de Usuario**: 5
Historia de usuario: Portero moviéndose según estadísticas.	
Condiciones de ejecución: El portero debe estar cargado y en la posición inicial predefinida.	
Entrada/Pasos para la ejecución: Al lanzar el balón, el portero debe ser capaz de detener el balón para que este no marque en la portería y este se deberá mover según las estadísticas defendiendo el área con mayor cantidad de goles marcados.	
Resultado esperado: El portero debe defender la portería tratando de que no le marquen gol en el área que mayor cantidad de goles tenga.	
Evaluación de la prueba: Resultado satisfactorio.	

Tabla 42 Caso de prueba de aceptación P7HU6

Caso de Prueba de Aceptación	
Código: P7HU6	**Número de Historia de Usuario**: 6
Historia de usuario: Seleccionar efecto del balón de futbol.	
Condiciones de ejecución: La pelota debe estar montada en Unity.	
Entrada/Pasos para la ejecución: Al lanzar el balón, debe actuar una fuerza de gravedad según el peso del balón.	
Resultado esperado: El balón debe hacer un efecto según su peso.	
Evaluación de la prueba: Resultado satisfactorio.	

Tabla 43 Caso de prueba de aceptación P8HU7

Caso de Prueba de Aceptación	
Código: P8HU7	**Número de Historia de Usuario**: 7
Historia de usuario: Mostrar las estadísticas del cálculo de la simulación.	
Condiciones de ejecución: La parte de la sección en la interfaz debe estar montada en el escenario.	
Entrada/Pasos para la ejecución: Al marcar gol, en dependencia de las áreas predefinidas, las estadísticas deben de cambiar.	
Resultado esperado: Las estadísticas cambian según el área que marque gol.	
Evaluación de la prueba: Resultado satisfactorio.	

Anexo 6 Tareas ingenieriles

Primera iteración

Tabla 44 Tarea ingenieril ponerles los huesos a los objetos 3D realizado.

Tarea de ingeniería	
Número de tarea: 2	**Número de historia de usuario**: 1
Nombre de tarea: Ponerles los huesos a los objetos 3D realizado.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 4
Fecha inicio: 26/3/2022	**Fecha fin**: 30/3/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Ponerles los huesos a los objetos 3D que llevan animaciones.	

Tabla 45 Tarea ingenieril exportar objetos al motor de videojuego.

Tarea de ingeniería

Número de tarea: 3	**Número de historia de usuario**: 1
Nombre de tarea: Exportar objetos al motor de videojuego.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 31/3/2022	**Fecha fin**: 1/4/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Luego de terminar de trabajar con Blender exportar los objetos hacia Unity 3D.	

Tabla 46 Tarea ingenieril realizar las pruebas unitarias a la primera iteración.

Tarea de ingeniería	
Número de tarea: 4	**Número de historia de usuario**: 1
Nombre de tarea: Realizar las pruebas unitarias a la primera iteración.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 10
Fecha inicio: 2/4/2022	**Fecha fin**: 13/4/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Se realizan pruebas unitarias a la primera iteración.	

Segunda iteración

Tabla 47 Tarea ingenieril estudio de las fórmulas para calcular la posición del mouse en la pantalla.

Tarea de ingeniería	
Número de tarea: 6	**Número de historia de usuario**: 2
Nombre de tarea: Estudio de las fórmulas para calcular la posición del mouse en la pantalla.	

Tipo de tarea: Desarrollo	**Puntos estimados**: 5
Fecha inicio: 18/4/2022	**Fecha fin**: 22/4/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Calcular la posición del mouse cuando se da click en la pantalla.	

Tabla 48 Tarea ingenieril implementación de un método para guardar la posición del mouse al presionar el clic izquierdo.

Tarea de ingeniería	
Número de tarea: 7	**Número de historia de usuario**: 2
Nombre de tarea: Implementación de un método para guardar la posición del mouse al presionar el clic izquierdo.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 5
Fecha inicio: 23/4/2022	**Fecha fin**: 28/4/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Se deberá guardar la posición del mouse en una variable para utilizarla en el cálculo de la trayectoria.	

Tabla 49 Tarea ingenieril colocar los objetos en los lugares predefinidos.

Tarea de ingeniería	
Número de tarea: 8	**Número de historia de usuario**: 3
Nombre de tarea: Colocar los objetos en los lugares predefinidos.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 5
Fecha inicio: 29/4/2022	**Fecha fin**: 5/5/2022
Programador responsable: Sergio Daniel Ortiz Jova	

Descripción: Se colocarán los objetos creados en posiciones específicas.

Tabla 50 Tarea ingenieril buscar una fórmula para darle una fuerza al balón.

Tarea de ingeniería	
Número de tarea: 9	**Número de historia de usuario**: 3
Nombre de tarea: Buscar una fórmula para darle una fuerza al balón.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 6/5/2022	**Fecha fin**: 9/5/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Se buscará una fórmula para añadirle una fuerza al balón.	

Tabla 51 Tarea ingenieril implementación de un método para aplicar una fuerza en la dirección del mouse.

Tarea de ingeniería	
Número de tarea: 10	**Número de historia de usuario**: 3
Nombre de tarea: Implementación de un método para aplicar una fuerza en la dirección del mouse.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 3
Fecha inicio: 10/5/2022	**Fecha fin**: 12/5/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Luego de buscar la fórmula para añadirle una fuerza al balón, este debe de ser aplicado hacia la posición del mouse anteriormente guardada.	

Tabla 52 Tarea ingenieril implementación de un método para dirigir el balón hacia la posición guardada por el clic del mouse con la fuerza dada.

Tarea de ingeniería	
Número de tarea: 11	**Número de historia de usuario**: 3
Nombre de tarea: Implementación de un método para dirigir el balón hacia la posición guardada por el clic del mouse con la fuerza dada.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 4
Fecha inicio: 13/5/2022	**Fecha fin**: 17/5/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Mover el balón con la fuerza dada hacia la posición del mouse.	

Tabla 53 Tarea ingenieril realizar las pruebas unitarias de la iteración número uno.

Tarea de ingeniería	
Número de tarea: 12	**Número de historia de usuario**: 3
Nombre de tarea: Realizar las pruebas unitarias de la iteración número uno.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 18/5/2022	**Fecha fin**: 19/5/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Realizar las pruebas unitarias a la primera iteración.	

Tabla 54 Tarea ingenieril realizar las pruebas unitarias de la iteración número dos.

Tarea de ingeniería	
Número de tarea: 13	**Número de historia de usuario**: 3

Nombre de tarea: Realizar las pruebas unitarias de la iteración número dos.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 20/5/2022	**Fecha fin**: 21/5/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Realizar las pruebas unitarias a la segunda iteración.	

Tercera iteración

Tabla 55 Tarea ingenieril calcular la distancia de la posición del portero a los laterales de la portería.

Tarea de ingeniería	
Número de tarea: 15	**Número de historia de usuario**: 5
Nombre de tarea: Calcular la distancia de la posición del portero a los laterales de la portería.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 3
Fecha inicio: 6/6/2022	**Fecha fin**: 8/6/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Se calculará dichas distancias para que el portero se ubique en el medio de la portería y este puede atrapar los balones en ambas direcciones.	

Tabla 56 Tarea ingenieril seleccionar el lugar hacia donde cubrirá el portero.

Tarea de ingeniería	
Número de tarea: 16	**Número de historia de usuario**: 5
Nombre de tarea: Seleccionar el lugar hacia donde cubrirá el portero.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 5

Fecha inicio: 9/6/2022	**Fecha fin**: 14/6/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Haciendo uso de las estadísticas el portero debe decir hacia donde es mejor lanzarse para detener o desviar el balón.	

Tabla 57 Tarea ingenieril asignar un peso al balón de futbol.

Tarea de ingeniería	
Número de tarea: 17	**Número de historia de usuario**: 6
Nombre de tarea: Asignar un peso al balón de futbol.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 1
Fecha inicio: 15/6/2022	**Fecha fin**: 15/6/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Se le debe asignar un peso al balón para que la gravedad actúe sobre él.	

Tabla 58 Tarea ingenieril asignar el efecto de la gravedad al balón.

Tarea de ingeniería	
Número de tarea: 18	**Número de historia de usuario**: 6
Nombre de tarea: Asignar el efecto de la gravedad al balón.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 1
Fecha inicio: 16/6/2022	**Fecha fin**: 16/6/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: El balón hará el efecto de la gravedad cuando se ejecute la trayectoria.	

Tabla 59 Tarea ingenieril crear la interfaz gráfica necesaria para llevar a cabo las estadísticas

Tarea de ingeniería	
Número de tarea: 19	**Número de historia de usuario**: 7
Nombre de tarea: Crear la interfaz gráfica necesaria para llevar a cabo las estadísticas	
Tipo de tarea: Desarrollo	**Puntos estimados**: 3
Fecha inicio: 17/6/2022	**Fecha fin**: 20/6/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Se crea la interfaz gráfica para mostrar los resultados	

Tabla 60 Tarea ingenieril realizar cálculos matemáticos y estadísticos para contar la cantidad de goles.

Tarea de ingeniería	
Número de tarea: 20	**Número de historia de usuario**: 7
Nombre de tarea: Realizar cálculos matemáticos y estadísticos para contar la cantidad de goles.	
Tipo de tarea: Desarrollo.	**Puntos estimados**: 7
Fecha inicio: 21/6/2022	**Fecha fin**: 28/6/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Se realizan los cálculos matemáticos y estadísticos necesarios para actualizar las variables en la interfaz.	

Tabla 61 Tarea ingenieril realizar pruebas unitarias a la primera iteración.

Tarea de ingeniería	
Número de tarea: 21	**Número de historia de usuario**: 7
Nombre de tarea: Realizar pruebas unitarias a la primera iteración.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 29/6/2022	**Fecha fin**: 30/6/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Se le realizaran las pruebas unitarias a la primera iteración.	

Tabla 62 Tarea ingenieril realizar pruebas unitarias a la segunda iteración.

Tarea de ingeniería	
Número de tarea: 22	**Número de historia de usuario**: 7
Nombre de tarea: Realizar pruebas unitarias a la segunda iteración.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 1/7/2022	**Fecha fin**: 2/7/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Se le realizaran las pruebas unitarias a la segunda iteración.	

Tabla 63 Tarea ingenieril realizar pruebas unitarias a la tercera iteración.

Tarea de ingeniería	
Número de tarea: 23	**Número de historia de usuario**: 7
Nombre de tarea: Realizar pruebas unitarias a la tercera iteración.	

Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 4/7/2022	**Fecha fin**: 5/7/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Se le realizaran las pruebas unitarias a la tercera iteración.	

Tabla 64 Tarea ingenieril realizar pruebas de aceptación al sistema.

Tarea de ingeniería	
Número de tarea: 24	**Número de historia de usuario**: 7
Nombre de tarea: Realizar pruebas de aceptación al sistema.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 6/7/2022	**Fecha fin**: 7/7/2022
Programador responsable: Sergio Daniel Ortiz Jova	
Descripción: Se le realizaran las pruebas de aceptación al sistema.	