ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

TESIS PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO SISTEMAS E INFORMÁTICA

AUTORES: ALMACHI TOAPANTA, DANIEL ALEJANDRO

BUSTAMANTE CRESPO, CHRISTIAN ANDRÉS

TEMA: ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN JUEGO LÚDICO EN 3D UTILIZANDO UN MOTOR GRÁFICO, TÉCNICAS DE IA Y TECNOLOGÍA CLOUD COMPUTING MULTIPLATAFORMA, PARA LA EMPRESA VLBS

DIRECTOR: ING. VILLACIS, CÉSAR

CODIRECTOR: ING. PRÓCEL, CARLOS

SANGOLQUÍ, FEBRERO 2014

CERTIFICACIÓN

Certificamos que el presente trabajo, fue realizado en su totalidad por los Sres. Christian Andrés Bustamante Crespo y Daniel Alejandro Almachi Toapanta como requerimiento parcial a la obtención del Título de **INGENIEROS DE SISTEMAS E INFORMÁTICA**, bajo nuestra supervisión.

Sangolquí, Febrero del 2014

_____	_____
Ing. César Villacís Silva	Ing. Carlos Prócel Silva
DIRECTOR	CODIRECTOR

DECLARACIÓN

Nosotros, Christian Andrés Bustamante Crespo y Daniel Alejandro Almachi Toapanta, declaramos que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

La Universidad de las Fuerzas Armadas (ESPE), puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normatividad institucional vigente.

_____ _____

Andrés Bustamante Daniel Almachi

AUTORIZACIÓN

Nosotros Christian Andrés Bustamante Crespo y Daniel Alejandro Almachi Toapanta autorizamos a la UNIVERSIDAD DE LAS FUERZAS ARMADAS (ESPE) la publicación en la Biblioteca Virtual de la Institución, del trabajo titulado “ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN JUEGO LÚDICO EN 3D UTILIZANDO UN MOTOR GRÁFICO, TÉCNICAS DE IA Y TECNOLOGÍA CLOUD COMPUTING MULTIPLATAFORMA, PARA LA EMPRESA VLBS”, que es de nuestra propia autoridad y responsabilidad.

Sangolquí, Febrero 2014

_____ _____

Andrés Bustamante Daniel Almachi

DEDICATORIA

Dedico este trabajo a mis Padres Jorge Luis Bustamante Espinoza y Flor Zenaida Crespo, quienes con su apoyo incondicional, enseñanzas y valores han forjado en mí un espíritu de perseverancia y obtención de mis objetivos y mis metas desde los inicios de mi infancia.

A mis hermanos Jorge Alejandro y Mario Fernando quienes con su entrega desmedida, consejos y apoyo constante estuvieron ahí siempre que los necesité.

A mis abuelitas Flor y Cecilia que jamás han perdido su confianza en mí y han dedicado horas de su tiempo para pedirle a mi Dios que me ayude siempre.

"Aquel que no ha fracasado nunca, es que no ha intentado nada"

Og Mandino

Andrés

DEDICATORIA

Dedico este trabajo a mis padres, Ximena Toapanta y Hugo Almachi, quienes me han enseñado valores importantes como la perseverancia, el esfuerzo y la humildad, a ellos que siempre me han apoyado desde el inicio, en los mejores y peores momentos.

A mi familia, amigos y compañeros de quienes he aprendido cada día algo nuevo y estuvieron durante todo el paso de la vida universitaria.

A mis profesores quienes supieron guiarme y compartieron no solo su conocimiento académico, también sus experiencias y consejos útiles para la vida fuera de la universidad.

“Basta un poco de espíritu aventurero para estar siempre satisfechos, pues en esta vida, gracias Dios, nada sucede como deseábamos, como suponíamos, ni como teníamos previsto.”

Noel Clarasó

Daniel

AGRADECIMIENTO

Le doy gracias a mi Dios por permitirme vivir cada día y ser la persona que soy y por ayudarme en todos los momentos de mi vida.

A mis padres por darme la oportunidad de recibir una Educación Superior, por su inmenso amor, entereza, cuidado, dedicación, amistad y consejo diario.

A mi compañero de tesis, que siempre ha estado ahí para lograr juntos el gran anhelo de convertirnos en los nuevos Ingenieros de la Patria.

Al Director y Codirector de tesis quienes con su experiencia y profesionalismo supieron encaminar de la mejor manera esta obra, y a todas las personas que de una u otra manera colaboraron en su culminación. Tomando en consideración que los hombres somos transeúntes y que las instituciones son permanentes, rindo homenaje a la Universidad de las Fuerzas Armadas (ESPE).

Andrés

AGRADECIMIENTO

Quiero Agradecer a mis padres y hermanos por apoyar de manera incondicional este sueño, respetando mis decisiones, inculcándome valores éticos y morales; los cuales se han plasmado en mi personalidad, y en la práctica de mis hechos.

A mis Familiares y Amigos que a lo largo de este tiempo han sido un apoyo emocional, encontrado siempre una mano en la cual apoyarme en los momentos difíciles.

Mis más sinceros agradecimientos a nuestro Director de Tesis Ing. Cesar Villacís y a nuestro Codirector Ing. Carlos Prócel quienes nos brindaron todo el apoyo para la realización de nuestro tema de tesis.

A todo el personal de la Dirección de Sistemas de la Universidad de las Fuerzas Armadas (ESPE) por brindarnos las facilidades para desarrollar este proyecto y colaboraron hasta el momento de su culminación.

Daniel

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Los video juegos educativos son un tipo de video juego que se aplican como parte del proceso de enseñanza-aprendizaje en los niños, para el desarrollo del pensamiento lógico y el desarrollo psicomotriz de los mismos. Este trabajo presenta el diseño, desarrollo, implementación y evaluación de un software educativo basado en video juegos denominado Multiplayer Challenge, enfocado a niños de 7 a 10 años. Para llevarlo a cabo se ha empleado la Metodología de Diseño Hipermedia Orientada a Objetos (OOHDM) para crear un entorno lúdico e interactivo, junto con la Inteligencia Artificial que controla las mecánicas del video juego, la búsqueda de caminos para el movimiento del avatar de la computadora y la planificación para la animación del avatar de los jugadores que intervienen en el juego. Se utilizan algoritmos con técnicas heurísticas para controlar el inicio y el fin de la partida del juego, así como también Unity 3D Game Engine con el motor de red multiplataforma Photon Cloud para controlar el funcionamiento del mundo virtual. Para modelar y animar los elementos 3D del juego se utilizó Autodesk Maya y para manipular las imágenes y texturas se utilizó Adobe Photoshop. Los resultados muestran que este software educativo estimula el raciocinio lógico y espacial, y el desarrollo psicomotriz en el área socio-personal.

Palabras Clave: Video juego, OOHDM, Unity 3D Game Engine, Juego Lúdico, IA.

ABSTRACT

Educational video games are applied as part of the teaching-learning process in children to stimulate logical thinking and psychomotor development. This paper presents the design, development, implementation and evaluation of educational software based on video games named Multiplayer Challenge, aimed at children from 7 to 10 years. To accomplish this we have used the Object Oriented Hypermedia Design Method (OOHDM) to create a fun and interactive environment with Artificial Intelligence that controls the mechanics of the video game, pathfinding to move the avatar and planning for avatar's animation of the players involved in the game. Heuristics algorithms are used to control the start and end of the game, Unity 3D Game Engine with cross-platform network engine Photon Cloud to control the behavior of the virtual world. For modeling and animating the 3D elements of the game we used Autodesk Maya and Adobe Photoshop to handle images and textures. The results show this educational software stimulates logical and spatial reasoning, and psychomotor development in the socio- personal area.

Keywords: Video game, OOHDM, Unity 3D Game Engine, Playful Game, AI.

CAPÍTULO 1

GENERALIDADES

1.1. Planteamiento del Problema

El desarrollo de aplicaciones en entornos distribuidos y la manera de consumir estos servicios dependen de la plataforma, tecnología y el contenido que la aplicación demande, siendo los entornos Web las principales plataformas que usan este tipo de tecnologías.

El desarrollo de video juegos de tipo MMO (Massively Multiplayer Online) incrementa el entretenimiento de los usuarios al hacer que interactúen entre sí en tiempo real en un entorno virtual. Trabajos como los de (Padilla Zea, González Sánchez, Gutiérrez, Cabrera, & Paderewski, 2008), (Mendoza Barros & Galvis Panqueva, 1998)han demostrado que el software educativo desde un punto de vista colaborativo (varios jugadores a la vez) puede enseñar a los estudiantes las ventajas del trabajo en equipo, desarrollan destrezas de respuesta rápida, investigación, lógica de tomas de decisiones, y habilidades de planeación a largo plazo.

Con el uso de la tecnología Cloud Computing como servidor virtual en la Nube se pretende desarrollar un video juego lúdico que demuestre las posibilidades y alcances que se lograría al realizar una video juego lúdico MMO competitivo y en 3D que pueda ser implementada en el mayor número

de plataformas posibles como Windows, Mac, Web, y Android y que todos los usuarios indistintamente de la plataforma puedan interactuar directamente a través de servidores virtuales generados en la Nube y sin necesidad de ningún tipo de registración o subscripción.

1.2. Justificación e importancia

La tecnología en entornos 3D es aprovechada por varios medios y en varias áreas como la educación, un área que apenas está usando este tipo de tecnología y que ha demostrado tener éxito, como lo menciona (Van Eck, 2006), en su publicación “Digital game-based learning: It’s not just the digital natives who are restless”, para un alumno es más interesante que mirar un libro de texto, es más real, dinámico y entretenido lo que facilita comprender el concepto y aumentar la concentración y memoria.

Para la enseñanza de historia, ciencia, matemáticas, física, etc. La tecnología 3D aporta dinámicas virtuales, simulaciones tridimensionales y animaciones reales que enfocan el aprendizaje y la educación desde una perspectiva más directa.

Actualmente las aplicaciones lúdicas enfocadas en la educación han pasado de video juegos en 2D a aplicaciones de entretenimiento educativo en 3D, pero no se ha visto que motiven con competitividad al ser jugadas entre varios usuarios simultáneamente. Uno de los problemas es que se requiere no solo de un software, sino también de un servidor dedicado para este tipo aplicaciones, por este motivo se desarrolló un video juego que

demuestre el potencial del uso de Cloud Computing en video juegos lúdicos 3D multiplataforma.

La investigación y el desarrollo que se realizó, estuvo orientado a fomentar el crecimiento de este tipo de aplicaciones con perspectiva didáctica, en este caso se aprovechó el concepto de Cloud Computing para desarrollar el video juego de razonamiento lógico y espacial en 3D y se lo implementó en plataformas Windows, Mac, Web y Android, los mismos que interactúan conjuntamente en el entorno virtual compitiendo a la vez por ganar el juego.

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar un video juego lúdico en 3D mediante la utilización de un motor gráfico, y tecnología Cloud Computing multiplataforma, aplicando OOHDM e implementarla en plataformas Windows, Mac, Web y Android.

1.3.2. Objetivos Específicos

- Analizar los conceptos teóricos acerca de aplicativos de software 3D, video juegos didácticos y realidad virtual para dispositivos móviles.
- Utilizar OOHDM con UML para el análisis y diseño del video juego en 3D.

- Implementar las mecánicas del juego con técnicas de Inteligencia Artificial.
- Implementar las librerías del juego utilizando un motor gráfico multiplataforma.
- Implementar el escenario virtual a base del modelamiento 3D utilizando una herramienta de modelado tridimensional.
- Desarrollar el video juego utilizando la metodología OOHDM para las diferentes plataformas del juego.

1.4. Alcance

- El video juego lúdico desarrollado se orienta en el proceso de enseñanza – aprendizaje del razonamiento lógico y espacial del usuario, mediante la utilización de tecnología virtual 3D y un servidor virtual con tecnología Cloud Computing.
- La aplicación de software desarrollada, se enfocó en el diseño y construcción de un video juego lúdico utilizando elementos de multimedia como: texto, imágenes, sonido, y animaciones.
- El video juego se desarrolló en tercera persona y los enemigos son otros usuarios en el mismo. Además las mecánicas con Inteligencia Artificial que se desarrollarlo sirvieron para aumentar la dificultad del juego.
- El video juego es de carácter competitivo, ya que un usuario interactúa directamente con otros usuarios en la misma escena del juego.

- Cada escena virtual contiene hasta seis usuarios simultáneamente.
- El video juego se diseñó y construyó para funcionar en plataforma Windows, Mac, Web y Android.
- El video juego tiene un menú de acceso, una opción de ayuda y genera escenarios virtuales en los que pueden acceder cada grupo de jugadores.
- El video juego despliega a los usuarios resultados finales de la carrera, mas no almacena esta información, ya que los usuarios se conectan de manera aleatoria y llevar un registro en este caso no es relevante.
- El video juego permite el acceso simultáneo al entorno virtual de cualquier usuario independientemente de la plataforma, permitiendo de esta manera que compita un usuario desde un teléfono con sistema operativo Android contra otros jugadores desde cualquier otro dispositivo.
- La entrada (input) de jugabilidad es diferente dependiendo del dispositivo al que este destinado.
- La aplicación no almacena información de ningún tipo, no permite la comunicación entre usuarios y la aplicación no es una red social.

Con este tema se indujo en el desarrollo investigativo de aplicaciones multiplataforma orientadas a la educación, en ambientes virtuales y aprovechando los servicios que brinda un motor de red y un motor gráfico si se los sabe utilizar conjuntamente.

1.5. Metodología

Para el desarrollo de esta aplicación multimedia se adoptó la Metodología de Diseño de Hipermedia Orientado a Objetos (OOHDM – Object Oriented Hypermedia Design Method), desarrollada por (Schwabe & Rossi, 1998). Esta metodología consta de cuatro etapas: Diseño Conceptual, Diseño Navegacional, Diseño Abstracto de Interfaz e Implementación. Cada etapa define un esquema objeto específico en el que se introducen nuevos elementos o clases.

En la *primera etapa* se desarrolló un modelo conceptual representado por las clases y las relaciones entre las mismas.

En la *segunda etapa*, se definió la estructura de navegación a través del video juego mediante la realización de modelos navegacionales, los cuales son representados por "nodos", "enlaces", y el esquema de contexto navegacional, que permite la estructuración del espacio de navegación en subespacios en los que se indica la información que será mostrada al usuario.

En la *tercera etapa* se especificó la interfaz abstracta, en la cual se define la forma en la cual deben aparecer los contextos navegacionales. Se incluye aquí el modo en que dichos objetos de interfaz activarán la navegación y el resto de funcionalidades de la aplicación.

En la *cuarta etapa*, se desarrolló el video juego con la concreción de los modelos navegacionales y de interfaz en objetos particulares con sus correspondientes contenidos y sus posibilidades de navegación.

En el desarrollo de este proyecto en lo que respecta al marco teórico se utilizó una metodología de trabajo basada en la investigación bibliográfica de fuentes de Información, y consultas en Internet. En esta etapa del proyecto se obtuvo la información necesaria y válida que permitió establecer el marco teórico referencial, el cual proporciona el soporte teórico – técnico necesario para la consecución del proyecto. Se cumplieron las siguientes actividades como parte de una metodología de desarrollo estándar:

- Análisis
- Diseño
- Desarrollo
- Pruebas

El sistema desarrollado se compone en su totalidad de componentes hipermedia. OOHDM se aplicó en conjunto con la herramienta UML (Lenguaje de Moldeamiento Unificado) para la generación de modelos.

1.6. Herramientas

En la Tabla 1 se hace una descripción de las herramientas utilizadas en el presente trabajo.

Tabla 1 - Descripción herramientas

Nombre	Descripción
Windows 7 Professional	Sistema Operativo parte de la familia de sistemas operativos desarrollados por la empresa Microsoft para gestionar los recursos de hardware y proveer servicios a los programas de aplicación. Esta versión incluye las mismas características que Home Premium, como también Escritorio Remoto, Sistema de Archivos Encriptados (EFS), y el modo Windows XP. (Orchilles, 2010)
Mac OS X Lion	Es la octava versión de los sistemas operativos de Apple basado en Unix. Un cambio bastante notorio en la interfaz de usuario, asemejándose al dispositivo móvil iPad. (Pogue, 2011)
Android OS	Android es una plataforma de código abierto diseñado para dispositivos móviles. Defendido por Google y es propiedad Open Headset Alliance. (Gargenta, 2011)
Lenguaje C#	C # es un lenguaje diseñado específicamente para programar en el Framework .NET de Microsoft. El Framework. NET consiste en un entorno de tiempo de ejecución denominado Common Language Runtime (CLR), y un conjunto de bibliotecas de clases, que proporcionan una plataforma de desarrollo que pueden ser explotadas por una variedad de lenguajes y herramientas. (Albahari, Drayton, & Merrill, 2002)
Unity 3D Game Engine	Unity 3D Game Engine es un ecosistema de desarrollo de juegos: un potente motor de renderizado totalmente integrado con juego completo de herramientas intuitivas y flujos de trabajo rápidos para crear contenido 3D interactivo; publicación multiplataforma, miles de activos de calidad, listos para usar en la Tienda de Activos y una comunidad de intercambio de conocimientos. (Unity Technologies, 2013).
Autodesk® Maya®	Es un software de animación en 3D que proporciona un conjunto completo de funciones creativas para realizar animación 3D, modelado, simulación, renderización y composición dentro de una plataforma de producción sumamente ampliable. Ahora Autodesk Maya incluye tecnología de visualización de última generación, flujos de trabajo de modelado más rápidos y nuevas herramientas para gestionar datos complejos. (Autodesk, Inc., 2013).
Star UML	StarUML es un proyecto de código abierto rápido, flexible, extensible, con varias características y de libre disposición que se ejecuta en la plataforma Win32. El objetivo del proyecto StarUML es construir una herramienta de modelado de software y plataforma que es un reemplazo convincente de herramientas UML comerciales como Rational Rose, Together, etc. (Star UML, 2013).
Photon Cloud	Combina la facilidad de uso de la red de Unity 3D Game Engine con el rendimiento y la fiabilidad de Photon Cloud. Funciona a través de diferentes redes (móviles). (Exit Games, 2013).

Fuente: Elaborado por autores de la investigación

CAPÍTULO 2

MARCO TEÓRICO

2.1. Video juegos

2.1.1. Breve historia de los video juegos

Un video juego es un programa informático interactivo destinado al entretenimiento que puede funcionar en varios dispositivos: computadoras, consolas, teléfonos móviles, etc.; integra audio y vídeo, permite disfrutar de experiencias que, en varios casos, sería complicado de vivir en la realidad. Su estructura narrativa es variada. Así, se encuentran argumentos basados en la parábola, la alegoría, la crónica, los relatos de viaje, los cuentos clásicos, los mitos, los relatos oníricos o los juegos de rol.

Aunque se ha dicho que su origen es lúdico, hoy en día se han ampliado y sobrepasado los límites del entretenimiento, porque se han abierto posibilidades de aplicación en el ámbito educativo. Se podría situar el origen de esta modalidad de entretenimiento digital en el primer simulador de vuelo diseñado en los Estados Unidos para el entrenamiento de pilotos. (García Fernandez, 2005)

El primer video juego fue “Space War”, parecido al posteriormente famoso “Asteroids”. Al parecer fue creado en 1961 por Steve Rusell en un computador del tamaño de un armario que contaba con una pequeña pantalla como la de un televisor.

En 1972 se desarrolla el primer video juego, llamado PONG, que consiste en una simplificada partida de ping-pong virtual. En cuanto a la preocupación por el efecto de estos artilugios en la conducta infantil, las primeras luces de alarma se encienden en 1977, fecha en la que la firma Atari colocó en el mercado el primer sistema de video juegos en cartucho para computador, que alcanzó un notable éxito comercial en EEUU. También empiezan a proliferar en los establecimientos de ocio, máquinas electrónicas que permitían jugar “Pacman” o “Space Invaders”.

Con la generalización del uso de las computadoras personales en las dos últimas décadas del siglo XX, comienzan a aparecer las primeras empresas dedicadas a la producción de video juegos. Los video juegos hoy en día son más que un producto informático; también son un negocio, un instrumento de información y formación, un objeto de investigación y un fenómeno social. (García Fernandez, 2005).

2.1.2. Clasificación, Tipo y Géneros

Existen factores a tomarse en cuenta para limitar y clasificar un video juego. Es importante conocer este tipo de clasificaciones para entender cómo afecta cada tipo de juego, por ejemplo, la diversión o a la interacción que el jugador experimentará.

i. Juegos Individuales o en Grupo

De acuerdo con (Zagal, Rick, & Hsi, 2006) cuando se juega en grupo se puede hacerlo competitivamente, colaborativamente y cooperativamente. Esta clasificación se basa en la forma en la que se asocian los jugadores para realizar sus objetivos o cómo consiguen sus metas, y no en cómo realizan las tareas para lograrlo.

- **Video juegos Individuales:** Es aquel en el que interactúa un solo jugador. Este se enfrenta a la Inteligencia Artificial del juego para conseguir una serie de objetivos dentro de la trama.
- **Video juegos Competitivos:** Un jugador consigue su propio éxito personal. Este objetivo final puede ser totalmente opuesto al de otros jugadores, por lo que generalmente al ganar un jugador, pierden el resto de jugadores.
- **Video juegos Colaborativos:** Son aquellos en donde el éxito individual se transforma en que el grupo consiga la meta deseada, es aquí donde aparece el concepto “equipo”. La meta es común, global y única para todos los individuos del juego.
- **Video juegos Cooperativos:** Cada uno de los participantes pueden tener sus propios objetivos individuales, aunque se agrupan en equipos para buscar un beneficio conjunto para la consecución de sus propias metas, que pueden obtener gracias a los demás. Es decir, el equipo no es primordial para conseguir la meta del jugador, sino algo circunstancial.

ii. Clasificación según Género

Partiendo de la propuesta de (Mendoza Barros & Galvis Panqueva, 1998) y (Rollings & Morris, 2003) se puede clasificar los juegos en los siguientes géneros, independientemente si son individuales o en grupo:

- **Acción:** La acción, el movimiento, combates y batallas son los factores predominantes. El jugador debe reaccionar con rapidez durante el juego. Juegos de este tipo pueden ser "*Fall Out", "God of War" o "Halo".*
- **Aventuras**: Se asemejan a una "película interactiva" donde el jugador es el protagonista del mismo. Juegos de este tipo son *"The Adventures of Tintin: Secret of the Unicorn The Game" o "Fable".*
- **RPG:** Siglas en ingles de Role Playing Game, es un tipo de video juego en que el usuario desempeña un papel, durante la trama el jugador debe mejorar sus habilidades interactuando con el entorno. El objetivo principal de estos juegos es que el jugador viva una historia. Ejemplos de este género son: *"Fable III", "Dark Souls" y la serie de "The Elder Scrolls".*
- **Estrategia:** En este tipo de juegos el usuario debe crear modelos, planear y dirigir operaciones, manejar recursos y desarrollar planificaciones a largo plazo para lograr objetivos concretos. Ejemplos de este género *"Civilization", "Halo Wars", "XCOM: Enemy Unknown".*
- **Simulación:** Son juegos en los que se reproduce las actividades del mundo virtual de una forma semejante a la real. Ejemplos: *"Microsoft*

Flight Simulator", "Battlestations: Pacific", "The Sims 3" (simulador social)" y "Football Manager 2006".

- **Educativos y Entrenamiento Mental:** Es un juego cuyo objetivo es transmitir algún conocimiento o estimular habilidades mentales. Un juego educativo puede abarcar cualquiera de los otros géneros. Generalmente están dirigidos a niños de menos de 10 años aunque pueden ser para usuarios de cualquier edad. Referentes en este campo son *"Aprende con Pipo"*, la saga de Nintendo *"Brain Training", "Musical Notes".*
- **Carreras:** Las carreras de vehículos son una de las características predominantes. El desafío es que el jugador manifieste su pericia al volante del vehículo. En este género destacan: *"F1 2013", "Forza Motorsport 2"* o *"Need For Speed Shift".*
- **Lucha:** La esencia primordial de estos juegos es la lucha, la cual puede ser de cualquier tipo conocido. El desafío de estos es que el jugador gane una serie de combates para lograr un campeonato. La complejidad de estos juegos es que para conseguir movimientos especiales (combos) el jugador debe presionar una combinación de teclas. Ejemplos de este género son: *"Street Fighter"* , *"Mortal Kombat", "WWE All Stars"*
- **Puzzles (Rompecabezas):** Se resuelven diferentes tipos de rompecabezas, espaciales o lógicos. Estos juegos prueban la habilidad y la destreza del jugador. El más representativo de este grupo es *"Tetris",* aunque también están *"Portal", "Peggle".*

- **Musicales:** La esencia del juego es la música, el jugador puede tocar algún instrumento, bailar o cantar tipo *karaoke*, Ejemplos: *"Just Dance", "Rock Band"*, *"Guitar Hero"*.
- **Deportes:** Son aquellos que simulan los deportes como el fútbol, soccer, baloncesto, golf, etc. Se puede destacar *"Pro Evolution Soccer"*, *"Wii Sport"* o *"FIFA 2014".*
- **Plataformas:** El protagonista avanza a través de un mapa con distintas plataformas u objetos situados a diferentes alturas para llegar a la meta. Algunos exponentes son: *"Sonic the Hedgehog"*, *"Super Mario Bros", "Mega Man"*.
- **Disparo (Shooters):** También conocidos como FPS (First Person Shooter). El protagonista se encuentra en la cámara de primera persona, y el objetivo de estos es abrirse paso a base de disparos. Juegos de este género son: *"Doom"*, *"Quake, "Halo", "Call Of Duty"*.
- **MMOG:** Siglas en inglés de *Massively Multiplayer Online Game*, conocidos como juegos multijugador en línea, se caracterizan porque los jugadores pueden interactuar con otros de forma competitiva o colaborativa. Los jugadores se pueden agrupar en *clanes* para mejorar habilidades. El ejemplo más destacado es el video juego *"World of WarCraft"*.
- **Mixtos:** estos juegos son una mezcla de diferentes géneros, poseen aspectos propios de alguno o todos los géneros anteriores y son un reto para los diseñadores, ya que obligan a dominar diferentes

conceptos del juego. Ejemplos de estos son: *"The Legend of Zelda"*, *"Castlevania"*, *"Uncharted"*, *"Metal Gear"*, o la saga *"GTA"*.

iii. Clasificación según su Contenido

Otra de las formas de clasificar los video juegos es identificando cómo se va a mostrar el contenido interactivo con el usuario. De acuerdo con (Yacci, Haake, & Rozanski, 2004) esta clasificación puede ser:

- **Juegos intrínsecos:** El contenido se ajusta con precisión a la acción del juego. Por ejemplo, un simulador de vuelo está enlazado con precisión a la habilidad de volar un avión. No se puede convertir fácilmente el simulador de vuelo en un juego que enseñe historia, pues no corresponde con la naturaleza del juego.
- **Juegos extrínsecos:** El contenido se "inserta" en la estructura del juego y procede del exterior. Por ejemplo, el juego llamado *"Smarty Party"* las preguntas no son "propias" del juego, sino que son cuestiones relativas a ciencia, historia, letras, etc., las cuales proceden del "exterior" (Prensky, 2001).
- **Juegos a ritmo propio:** En este tipo de juego el tiempo puede ser detenido, de esta manera el jugador puede planificar detalladamente, ejecutar y evaluar las acciones. El ajedrez es un juego a ritmo propio.
- **Juegos a ritmo externo:** El jugador reacciona a un conjunto de condiciones que cambian constantemente. Aunque se puede anticipar

el momento en que ocurrirán algunos de los eventos, se hace énfasis en la rapidez y agilidad al seleccionar las respuestas o las herramientas. Un ejemplo es responder al servicio en el tenis, es impredecible y dependiente del tiempo de reacción y la selección de la respuesta (Singer & Williams, 1998).

2.2. Video juegos, educación y desarrollo intelectual

El uso de los video juegos en la ayuda para determinados aprendizajes y entrenamientos es positivo, tal y como se demuestra en el tratamiento de los problemas de aprendizaje, la ayuda para resolverlos, para responder a cuestiones relacionadas con la escuela, las drogas, la familia, aspectos morales, etc. Los video juegos permiten aumentar la motivación para el aprendizaje de diversas materias como las matemáticas y las ciencias.

Además pueden ser utilizados como entrenamiento eficaz en programas de tipo viso-motor, desarrollo del pensamiento reflexivo, mejora de las habilidades de los pilotos de avión, reducción del número de errores de razonamiento, mayor control de los tiempos de reacción, y servir de enfrentamiento ante situaciones vitales que pueden ser simuladas, como es el caso de la resolución de problemas, tema en el que se muestran eficaces. (Balerdi, 2009)

De acuerdo con (García Fernández & Bringué Sala, 2010) se pueden mencionar algunos factores de los video juegos que fomentan la motivación:

- Carácter lúdico de los aprendizajes.

- Dificultad creciente y progresiva de las habilidades, pero adaptada al ritmo de cada uno, posibilidad de repetir y corregir los errores.
- Recompensa inmediata, que además responde a un plan predeterminado y conocido.
- Reconocimiento social de los logros adquiridos, inscripción personalizada de las puntuaciones alcanzadas o los niveles superados.
- Estimulación simultánea a múltiples niveles: visual, auditivo, etc.
- Identificación con héroes o personajes que fomentan la imitación.

A la vista está que, habitualmente, ni la escuela ni la familia, agentes educativos por excelencia, proponen aprendizajes que posean estas características, o al menos no de una manera tan perseverante e intensa, por esta razón (Greenfield, 1996) investigó el aprendizaje producido en niños y niñas de 12 a 16 años que utilizaron video juegos de aventuras. Las principales conclusiones obtenidas fueron que:

- Aumentaban las estrategias de lectura visual de imágenes y de lectura del espacio tridimensional.
- Ayudaban a trabajar el aprendizaje por observación y la verificación de hipótesis.
- Facilitaban la comprensión de simulaciones científicas.
- Incrementaban las estrategias para recibir y procesar información recibida de varias fuentes simultáneamente (procesamiento en paralelo).

(Mcfarlane, Sparrowhawk, & Heald, 2002) reconocen que los video juegos allanan la adquisición y el desarrollo de ciertas estrategias fundamentales para el aprendizaje: la resolución de problemas, el aprendizaje de secuencias, el razonamiento deductivo y la memorización. También, simplifican la realización de trabajos en grupo de tipo cooperativo o en colaboración y el aprendizaje basado en la resolución de tareas.

2.3. MMOG (Massively Multiplayer Online Game)

Un juego MMO es aquel que tiene la capacidad de reunir a varios jugadores a la vez conectados a través de Internet. El término MMOG se da por las siglas en inglés “Massively Multiplayer Online Game” que literalmente significa "Juego Masivo Multijugador en Línea". Los juegos más comunes de este tipo son los llamados juegos de navegador, que son diseñados para ser jugados directamente en cualquier navegador de Internet, como Mozilla Firefox, o Internet Explorer, por lo que normalmente no se necesita instalar ningún tipo de software para comenzar a jugar, aunque actualmente se encuentran diversos tipos de juegos multijugador para las diferentes consolas en el mercado como Xbox 360, Play Station 3, o Nintendo Wii. (JuegosMMO.net, 2013)

2.4. Ingeniería de Software

La ingeniería del software es el establecimiento y uso de principios sólidos de la ingeniería para obtener económicamente un software confiable y que funcione de modo eficiente en máquinas reales. (Pressman, 2006).

Como cualquier ingeniería esta debe estar sustentada en un compromiso con la calidad, como se puede ver en la Figura 1, referente a los estratos de la Ingeniería de Software.

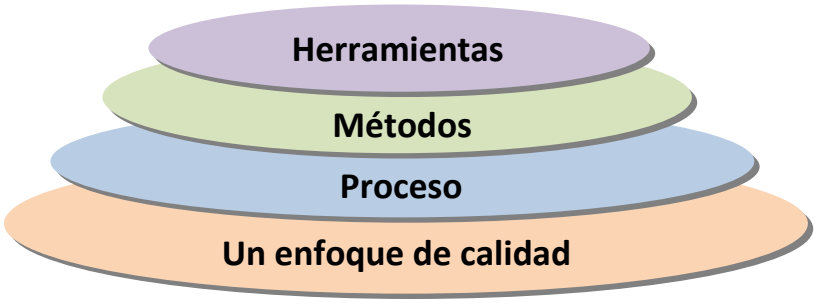

Figura 1 - Estratos de la Ingeniería de Software

Fuente: Pressman, 2006, pag. 24

- El ***enfoque de calidad*** ayuda a fomentar una cultura de mejora continua del proceso, produciendo enfoques efectivos en la Ingeniería de Software.
- El ***proceso*** es el elemento que mantiene juntos a los estratos de la tecnología y permite el desarrollo racional y a tiempo del software, es la base para el control de la gestión de los proyectos de software, establece el marco de trabajo para aplicar los métodos técnicos, se establecen los fundamentos y se asegura la calidad y el cambio se maneja de forma apropiada.

- Los ***métodos*** proporcionan la forma técnica para construir software, los métodos son un conjunto de tareas que incluyen la comunicación, análisis de requerimientos, modelado de diseño, construcción del programa, pruebas y soporte.
- Las ***herramientas*** proporcionan el soporte automatizado para el proceso y los métodos.

2.5. Aplicaciones 3D

Las aplicaciones 3D manipulan los mismos planos que una aplicación 2D, pero con la ventaja de poder situarlos en un espacio tridimensional. De esta forma diferentes elementos planos pueden estar contenidos en planos distintos, también se pueden manipular directamente formas geométricas tridimensionales (como prismas, cilindros, esferas, etc.) sin necesidad de descomponerlas en elementos planos que formen una figura que sea equivalente.

Debe notarse que, manipular elementos geométricos tridimensionales, implica tener capacidad para almacenar la información necesaria para describir la forma y posición de dichos elementos en un espacio tridimensional. También implica disponer de un gestor de visualización que permita generar de forma automática todas las proyecciones que el usuario pueda requerir (resolviendo todos los problemas de ocultación, iluminación, etc.), de forma que el conjunto de formas geométricas se comporte como una escena tridimensional que se ve a través de la pantalla plana del

computador. (Aleixos Borrás, Piquer Vicent, Galmes Gual, & Company Calleja, 2002).

2.6. Inteligencia Artificial

Existen varias definiciones de la Inteligencia Artificial, la utilizada en este trabajo es la propuesta por la *Encyclopedia Of Artificial Intelligence:*

“La IA es un campo de la ciencia y la ingeniería que se ocupa de la comprensión, desde el punto de vista informático, de lo que se denomina comúnmente comportamiento inteligente. También se ocupa de la creación de artefactos que exhiben este comportamiento”

2.6.1. Inteligencia Artificial en Video Juegos

La Inteligencia Artificial en la mayoría de los video juegos modernos constan de 3 características básicas: la habilidad para mover los personajes, la habilidad para tomar decisiones hacia donde moverse, y la habilidad de pensar táctica o estratégicamente. Incluso si se reemplaza la Inteligencia Artificial basada en estados a una amplia gama de técnicas, todas cumplen con los 3 requisitos básicos anteriores. (Millington & Funge, 2009)

2.6.2. Técnicas de Inteligencia Artificial

i. Heurísticas

Una heurística es una regla empírica, una aproximación a la solución que tal vez funcione en varias situaciones pero no en todas.

Las técnicas heurísticas se aplican en las búsquedas en la Inteligencia Artificial, denominadas búsquedas heurísticas, las cuales son un área central de la misma y sus algoritmos han sido ampliamente utilizados en la planificación, juego de roles y en agentes de control. (González-Calero & Gómez-Martín, 2011)

ii. Planificación

Es un proceso mediante el cual un agente busca un plan para lograr cierto objetivo. El plan es una serie de acciones a tomar, partiendo de un estado inicial hasta cumplir el objetivo.

En planificación, los estados se representan por un conjunto de hechos que son verdaderos en un estado del mundo. (Inteligencia Computacional, 2013) En la Figura 2 se puede apreciar una representación general de un proceso de planificación.

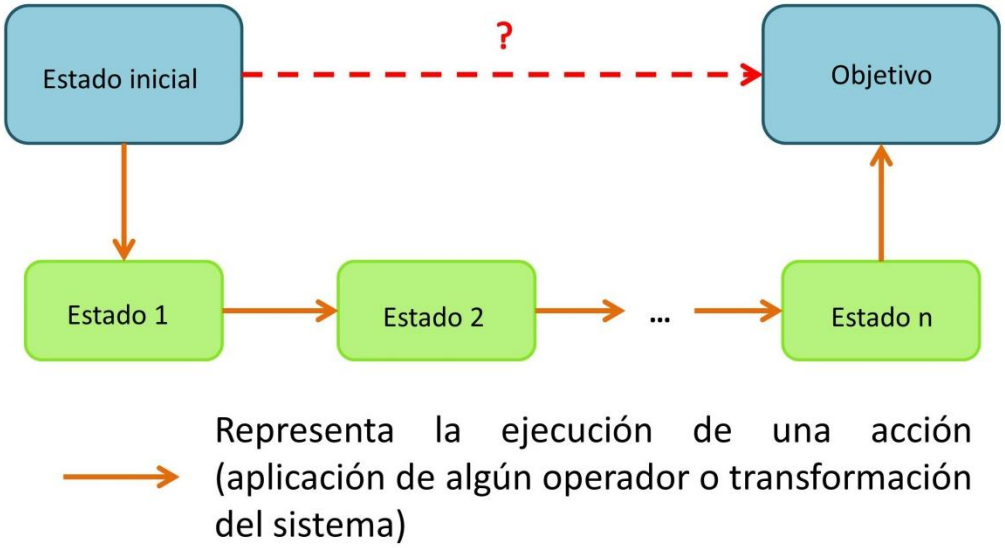

Figura 2 - Representación general planificación

Fuente: Fernando Berzal, diapositiva 6

iii. Búsqueda de caminos (Pathfinding)

La búsqueda o planificación de caminos, se ha convertido en un elemento esencial en la programación de juegos de computador. Un ejemplo es el clásico juego de *Pacman*. Un buen programa debería encontrar la trayectoria más corta desde cada fantasma a la posición del *Pacman* y planificar el movimiento consecuentemente en lugar de moverse aleatoriamente hasta dar con él.

Otro ejemplo es el juego de estrategia Age of Empires (véase Figura 3). El objetivo del juego es conquistar el mundo con una de las civilizaciones disponibles. Para ello se parte de la situación caótica resultante de la caída del imperio romano y se intenta construir un nuevo imperio, lo cual implica explorar y potencialmente enfrentarse con otras civilizaciones a través de diversas campañas. La planificación de caminos se utiliza para mover

unidades de combate por el terreno evitando obstáculos. La planificación de caminos es una forma exhaustiva de analizar el terreno de combate. (Alfonso Galipienso, Cazorla Quevedo, Colomina Pardo, Escolano Ruiz, & Lozano Ortega, 2003)

Figura 3 - Captura Juego Age of Empires III

Fuente: http://www.giantbomb.com/images/1300-11333

2.7. Computación en la Nube (Cloud Computing)

Es un nuevo concepto que se utiliza para hacer referencia a un conjunto de herramientas y servicios a los que se puede acceder únicamente a través de Internet. Estas plataformas permiten conectar diferentes dispositivos -como equipos de escritorio, tablets o teléfonos celulares- y aplicaciones informáticas, para acceder a información que se puede elaborar, compartir y almacenar en Internet. (Caccuri, 2013)

2.7.1. Beneficios

De acuerdo con (Inergis, 2012), algunos de los beneficios del Cloud Computing son:

- Integración probada de servicios Red. Por su naturaleza, la tecnología de *Cloud Computing* se puede integrar con el resto de las aplicaciones empresariales (tanto software tradicional como Cloud Computing basado en infraestructuras), ya sean desarrolladas de manera interna o externa.
- Prestación de servicios a nivel mundial. Las infraestructuras de *Cloud Computing* proporcionan mayor capacidad de adaptación, recuperación completa de pérdida de datos (con copias de seguridad) y reducción de los tiempos de inactividad.
- Una infraestructura 100% de *Cloud Computing* permite al proveedor de contenidos o servicios en la Nube prescindir de instalar cualquier tipo de software, ya que éste es provisto por el proveedor de la infraestructura o la plataforma en la Nube. Un beneficio del *Cloud Computing* es que requiere menor inversión para empezar a trabajar.
- Actualizaciones automáticas que no afectan negativamente a los recursos de TI. Al actualizar a la última versión de las aplicaciones, el usuario se ve obligado a dedicar tiempo y recursos para volver a personalizar e integrar la aplicación. Con el *Cloud Computing* no hay que decidir entre actualizar y conservar el trabajo, dado que esas personalizaciones e integraciones se conservan automáticamente durante la actualización.

- Contribuye al uso eficiente de la energía. En este caso, a la energía requerida para el funcionamiento de la infraestructura. En la Nube, la energía consumida es sólo la necesaria, reduciendo notablemente el desperdicio.

2.7.2. Desventajas

Algunas desventajas del Cloud Computing descritas por (Cruz Zeballos, 2012) son:

- La centralización de las aplicaciones y el almacenamiento de los datos originan una interdependencia de los proveedores de servicios.
- La disponibilidad de las aplicaciones están sujetas al acceso a Internet.
- Los datos "sensibles" del negocio no residen en las instalaciones de las empresas, lo que podría generar un contexto de alta vulnerabilidad para la sustracción o robo de información.
- Seguridad. La información de la empresa debe recorrer diferentes nodos para llegar a su destino, cada uno de ellos (y sus canales) son un foco de inseguridad. Si se utilizan protocolos seguros, HTTPS por ejemplo, la velocidad total disminuye debido a la sobrecarga que éstos requieren.
- Escalabilidad a largo plazo. A medida que más usuarios empiecen a compartir la infraestructura de la Nube, la sobrecarga en los servidores de los proveedores aumentará, si la empresa no posee un

esquema de crecimiento óptimo puede llevar a degradaciones en el servicio o altos niveles de jitter.

- Privacidad. La información queda expuesta a terceros que pueden copiarla o acceder a ella.

2.8. Metodología OOHDM para el desarrollo de aplicaciones 3D

2.8.1. Definición

OOHDM es una metodología de desarrollo de aplicaciones multimedia extensas basada en modelos, cuyo objetivo es el diseño de aplicaciones hipermedia. OOHDM está compuesta por cuatro fases llamadas: Diseño Conceptual, Diseño Navegacional, Diseño de Interfaz Abstracto e Implementación. Estas fases de OOHDM se llevan a cabo en una combinación de modelos de desarrollo incremental, iterativo y basado en prototipos, como se describe a continuación en la Tabla 2.

Tabla 2 - Etapas de la metodología OOHDM

Etapas	Productos	Formalismos	Mecanismos	Descripción
Obtención de Requerimientos	Casos de Uso (actores, escenarios).	Plantillas del formato del documento, Diagramas de Interacción de Usuario (UIDs).	Técnicas de Observación, Entrevistas.	Se crea un documento que describe actividades y requerimientos de los usuarios.
Diseño Conceptual	Clases, subsistemas, relaciones, atributos.	Modelos Orientados a Objetos	Clasificación, agregación, generalización y especialización	Se modela la semántica del dominio de la aplicación.
Diseño Navegacional	Nodos, enlaces, estructuras de acceso, contextos navegacionales, transformaciones de navegación.	Vistas Orientadas a Objetos, Cartas de navegación orientadas a objetos, Clases de Contexto.	Clasificación, agregación, generalización y especialización.	Se tiene en cuenta el perfil del usuario y las tareas. Se enfatiza en los aspectos cognitivos. Se crea la estructura de navegación de la aplicación.
Diseño de Interfaz Abstracta	Objetos de la interfaz abstracta, respuestas a eventos externos, transformaciones de la interfaz	Vistas Abstractas de Datos (ADV), Diagramas de Configuración, Cartas de navegación de los ADVs	Mapeado entre la navegación y los objetos visibles	Se modelizan los objetos visibles. Se describe la interfaz para los objetos de navegación. Se define el aspecto de los objetos de la interfaz.
Implementación	Aplicación en funcionamiento	Los soportados por el entorno	Los que provea el entorno	Se realiza la puesta en producción del sistema.

Fuente: Schwabe & Rossi, 1998, pág. 2

2.8.2. Obtención de requerimientos

Como en todo proyecto informático, la obtención de requerimientos es una etapa de importancia, ya que los errores más costosos son los que se comenten en esta etapa al no tener claro que es lo que se necesita. OOHDM divide esta etapa en cinco sub-etapas: identificación de roles y tareas, Especificación de escenarios, Especificación de casos de uso, Especificación de UIDs (Diagramas de Interacción de usuarios) y Validación de casos de uso y UIDs.

i. Identificación de roles y tareas

En esta etapa se debe identificar los diferentes roles que podrían cumplir cada uno de los potenciales usuarios de la aplicación. Un análisis rápido de la aplicación a desarrollarse, permite identificar el siguiente rol: Jugador.

Después de identificar el rol de cada usuario se debe identificar la tarea que deberá soportar la aplicación, como por ejemplo para el rol jugador: Buscar salas disponibles de juego.

ii. Especificación de escenarios

Los escenarios son descripciones narrativas de cómo la aplicación, será utilizada. En esta sub-etapa, cada usuario deberá especificar textual o

verbalmente los escenarios que describen su tarea. (Soto De Giorgis, Palma Muñoz, & Roncagliolo De La Horra, 2011, pág. 3).

iii. Especificación de casos de uso

Los casos de uso se utilizan para especificar el comportamiento entre los actores (usuarios) y el sistema, agrupando las tareas representadas en los escenarios existentes. Es importante identificar la información relevante en cada uno de ellos.

iv. Especificación de UIDs

De acuerdo a UML, los diagramas de secuencia, de colaboración y de estado son capaces de representar un caso de uso. Sin embargo, la especificación de los casos de usos usando estas técnicas es un amplio trabajo y puede anticiparse a tomar algunas decisiones de diseño. Para evitar esto OOHDM propone la utilización de una herramienta llamada UID, que permite representar en forma rápida y sencilla los casos de uso generados en la etapa anterior. Para obtener un UID desde un caso de uso, la secuencia de información intercambiada entre el usuario y el sistema debe ser identificada y organizada en las interacciones. Identificar la información de intercambio es crucial ya que es la base para la definición de UIDs. (Soto De Giorgis et al., 2011, pág. 3)

v. Validación de casos de uso y UIDs

En esta etapa, el desarrollador deberá interactuar con cada usuario para validar los casos de uso y UIDs obtenidos, mostrando y explicando cada uno de ellos para ver si los usuarios están de acuerdo. El usuario deberá interceder sólo en aquellos casos de uso y UIDs en los que participa. (Soto De Giorgis et al., 2011)

2.8.3. Diseño Conceptual

Durante el diseño conceptual se construye un modelo que represente al dominio de la aplicación usando las técnicas del modelado orientado a objetos. La mayor diferencia con UML es el uso de atributos con varios valores y el uso de direcciones en las relaciones. Las clases conceptuales deben ser realizadas usando agregación y jerarquías de generalización/especialización. El principal interés durante esta etapa es captar las semánticas del dominio tan neutras como sean posibles, con un poco de énfasis por los tipos de usuarios y tareas. OOHDM no recomienda ningún método particular para crear el esquema de las clases conceptuales, cualquiera de las metodologías conocidas pueden ser empleadas (OMT, UML), como se puede ver en la Figura 4.

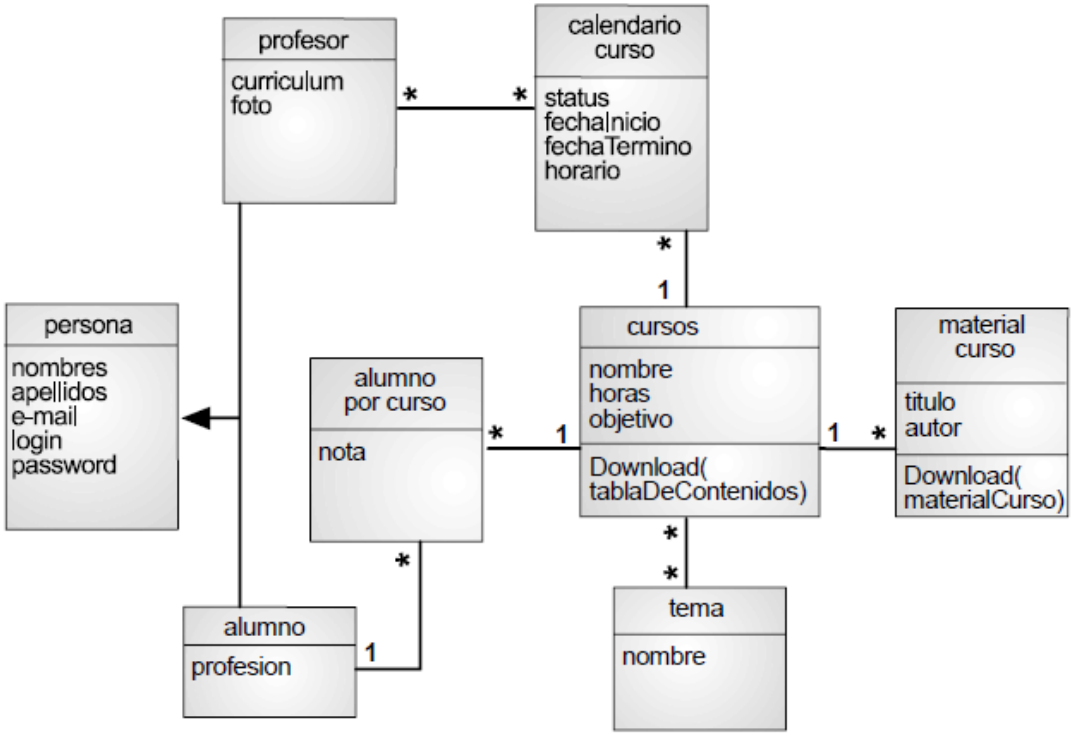

Figura 4 - Esquema Conceptual

Fuente: Soto De Giorgis et al., 2011, pág. 4

2.8.4. Diseño Navegacional

En esta etapa de la metodología se pretende desarrollar una topología navegacional que permita a la aplicación ejecutar todas las tareas requeridas por el usuario. (Soto De Giorgis et al., 2011)

i. Aplicación del diseño navegacional

Una vez que ya se han diseñado todos los diagramas de contexto, uno para cada caso de uso con sus respectivas tarjetas de especificación, es necesario realizar la unión de todos los diagramas para formar uno sólo. El

diagrama resultante corresponderá al diagrama de contexto de toda la aplicación, como se puede ver en la Figura 5.

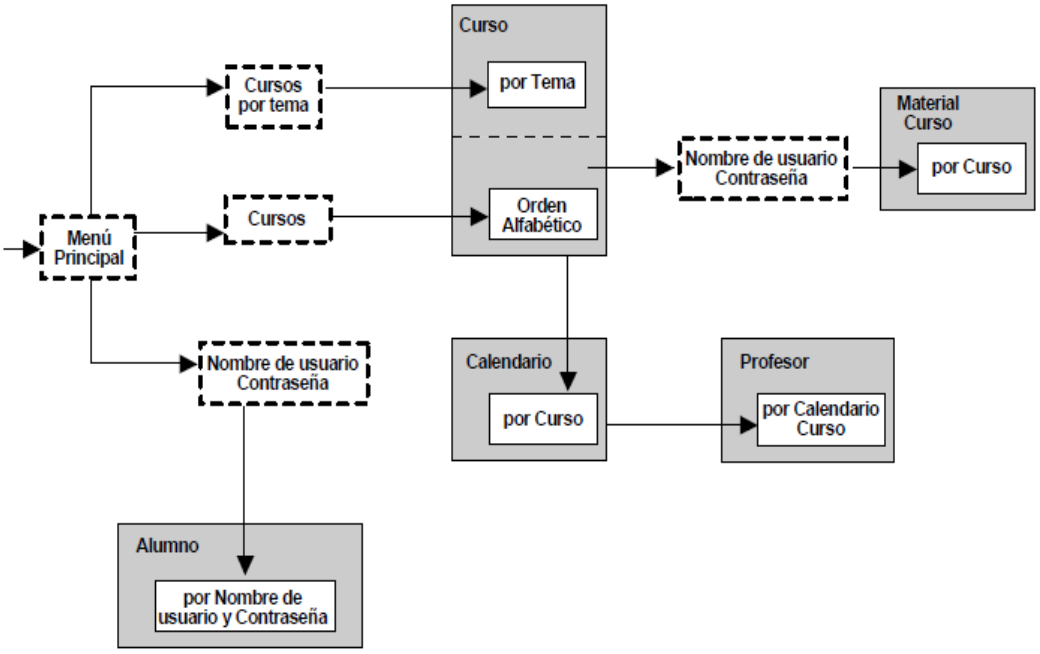

Figura 5 - Diagrama de contexto final

Fuente: Soto De Giorgis et al., 2011, pág. 7

ii. Esquema de clases navegacionales

Las posibles vistas del hiperdocumento se generan a través de las llamadas clases navegacionales que son tipos predefinidos de clases conformados por: nodos, enlaces, anclas y estructuras de acceso que son los índices o recorridos guiados, que representan los posibles caminos de acceso a los nodos, como se puede ver en la Figura 6.

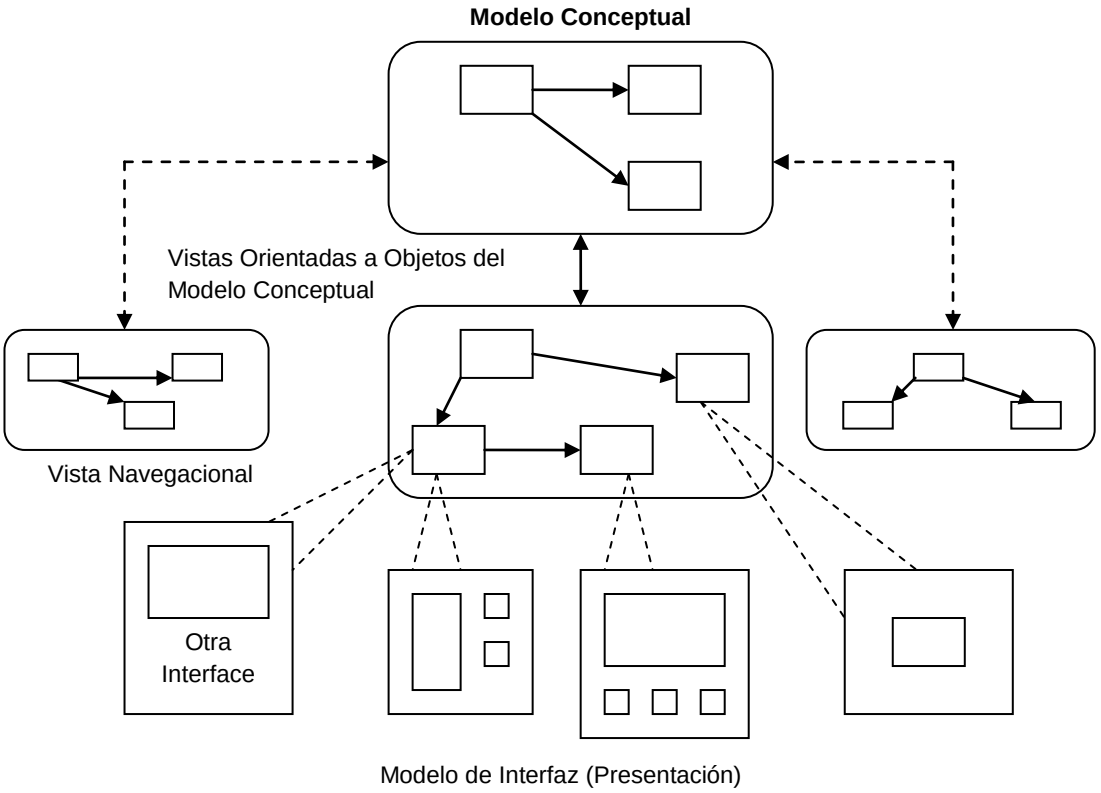

Figura 6 - Esquema del diseño navegacional

Fuente: An Object Oriented Approach to Web-Based Application Design, pág. 5

Nodos

- Contenedores de información básicos.
- Se definen como vistas orientadas a objetos de las clases conceptuales definidas en la fase de Diseño Conceptual.
- Contienen atributos simples y enlaces.

Enlaces

- Identifican relaciones.
- Implementan las relaciones definidas en el esquema conceptual.
- Las clases de los enlaces especifican sus atributos, comportamiento y los objetos fuente y destino.
- Representan las posibles formas de comenzar la navegación.

Estructuras de Acceso

- Actúan como índices o diccionarios.
- Útiles para ayudar al usuario final a encontrar la información deseada.
- Ejemplos: menús, índices y tours guiados.

iii. Esquema de contextos navegacionales

Un esquema de contexto navegacional es el que organiza el espacio navegacional en conjuntos convenientes o sub-espacios que pueden ser recorridos en un orden particular y que deberían ser definidos como caminos para ayudar al usuario a lograr la tarea deseada. (Soto De Giorgis et. al., 2011)

Contexto navegacional

- Es un conjunto de nodos, enlaces, clases de contextos y otros contextos navegacionales (contextos anidados).
- Inducido de clases navegacionales
- Se definen por extensión o enumerando sus miembros
- Un índice o un tour guiado definen contextos de navegación

Para cada UID se crearán diagramas de contexto y tarjetas de especificación que detallan la información contenida en el diagrama. En la Figura 7 se grafica el diagrama de contexto correspondiente al UID del caso de uso “Buscando un curso dado un tema”. (Soto De Giorgis et al.,2011)

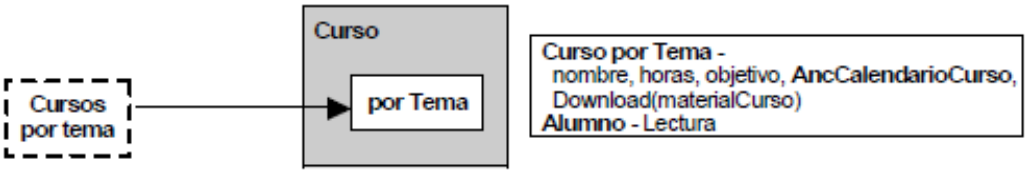

Figura 7 - Diagrama de contexto correspondiente al UID del caso de uso "Buscando un curso dado un tema"

Fuente: Soto De Giorgis et al., 2011, pág. 5

2.8.5. Diseño de Interfaz Abstracta

Una vez finalizado el diseño navegacional, será necesario especificar las diferentes interfaces de la aplicación. Esto significa definir de qué manera aparecerán los objetos navegacionales en la interfaz y cuales objetos activarán la navegación. (Soto De Giorgis et. al., 2011) Para esto se utilizaran los siguientes diagramas:

- ***Diagrama de vista de datos abstractos.*** Son modelos formales de los objetos de la interfaz y son específicos por mostrar:
  - La estructura, los aspectos estáticos de los objetos de la interfaz usando composición.
  - La manera en cual están relacionados estáticamente con los objetos de navegación.
  - Cuál es su comportamiento cuando reaccionan a eventos externos.

- ***Diagrama de configuración.*** En este diagrama se representan principalmente:
  - Los eventos externos provocados por el usuario, como *Clic o Doble Clic del Ratón* que maneja un ADV.
  - Los servicios que ofrece el ADV (como "visualización").
  - Las relaciones estáticas entre las ADVs.

2.8.6. Implementación

Una vez terminadas las etapas anteriores, el desarrollador posee un completo conocimiento del dominio del problema. Identificando la información que será mostrada, cómo estará organizada y cuales funciones permitirá ejecutar la aplicación. Además cuenta con una idea básica de cómo se verán las interfaces. Para comenzar con la implementación el desarrollador deberá elegir en donde almacenará los objetos y con qué lenguaje o herramienta desarrollará las interfaces. Es necesario aclarar que generalmente el desarrollador se encarga del lado técnico de la interfaz, en la parte gráfica y la apariencia final el encargado es el diseñador gráfico. (Soto De Giorgis et. al., 2011)

CAPÍTULO 3

ANÁLISIS Y DISEÑO DEL JUEGO LÚDICO

3.1. Especificación de requerimientos

La presente especificación de requerimientos pertenece al desarrollo del Caso de Estudio "Multiplayer Challenge (MC)", como Tesis para la obtención del título de Ingeniería en Sistemas e Informática y está desarrollada siguiendo las directrices de la metodología OOHDM y del Lenguaje Unificado de Modelado (UML) junto con la Ingeniería de Software.

3.1.1. Introducción

i. Propósito

El propósito del presente apartado es definir los requerimientos que debe tener el Juego "Multiplayer Challenge (MC)" a partir de ahora. Con la especificación de requerimientos se formalizarán las funcionalidades de la aplicación.

ii. Definiciones, acrónimos y abreviaturas

- **2D**.- Abreviatura utilizada para definir un ambiente de trabajo gráfico en dos dimensiones (two dimensions).

- **3D**.- Corresponde a la abreviatura utilizada para definir un ambiente de trabajo gráfico en tres dimensiones (three dimensions).
- **Game Engine**.- Término definido en español como Motor de Juegos que se define como un conjunto de herramientas de programación para manejar gráficos en 2D y 3D.
- **Gamer**.- En español Jugador o Video-jugador, que se define como el usuario de un video juego interactivo sea en 2D o en 3D.
- **MC**.- Corresponde a la abreviatura del video juego conocido como Multiplayer Challenge.
- **GUI**.- Siglas inglés de Graphics User Interface o Interface Gráfica de Usuario, la cual permite interactuar con dispositivos electrónicos con imágenes en vez de líneas de comando.
- **Assets.-** En Unity 3D Game Engine corresponde a toda la parte multimedia, como por ejemplo: modelos, texturas, materiales, sonidos, animaciones, etc.
- **Scripts.-** Es un programa simple, usado para realizar tareas como combinar componentes, interactuar con el sistema operativo o con el usuario.
- **Prefabs.-** un prefab es un tipo se asset reusable almacenado en la vista del proyecto.
- **Collider.-** es un componente de Unity 3D Game Engine que permite que el objeto al cual está enlazado, tenga un cuerpo rígido, y reaccionar a otros colliders en otros objetos. Es un detector de colisiones.

- **Trigger.-** un trigger se aplica a un collider para permitir que solo se sepa que un objeto pasó sobre otro sin que se produzca una colisión.
- **Avatar.-** representación gráfica que se asocia a un usuario para su identificación.
- **Skybox.-** es una envoltura alrededor de la escena que muestra el más allá del mundo virtual.

3.1.2. Identificación de Roles y Tareas

i. Roles

- **Jugador – Gamer**

Es el usuario que tiene acceso al juego para poder interactuar con la aplicación en 3D, como se puede ver en la Figura 8.

Figura 8 - Actores Multiplayer Challenge

Fuente: Elaborado por los autores de la investigación

ii. Tareas

- **Jugador – Gamer**

a) Crear una nuevo cuarto (room) de juego.

b) Entrar en un cuarto existente.

c) Competir en el video juego.

3.1.3. Especificación de Escenarios

i. Rol Jugador

- *Entrar en un cuarto de juego existente:* El jugador podrá unirse a un cuarto de juego creado por algún otro jugador.
- *Competir en el video juego:* El jugador una vez que ingrese a un cuarto juego podrá competir, ya sea con otros jugadores o contra la computadora.

3.1.4. Especificación de casos de uso por actor

i. Actor: Jugador

Como se puede observar los casos de uso del actor jugador en la Figura 9.

a) USR-JUG-MC-01: Ver ayuda del juego.
b) USR-JUG-MC-02: Entrar en un cuarto de juego.
c) USR-JUG-MC-03: Empezar partida.

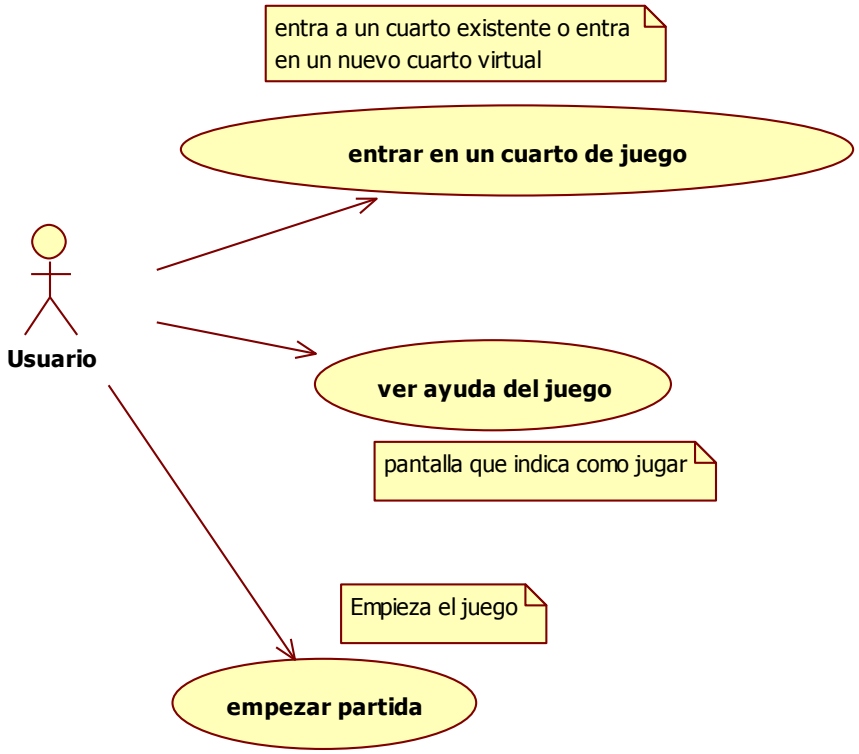

Figura 9 - Casos de uso del actor: jugador

Fuente: Elaborado por los autores de la investigación

A continuación se detallan las especificaciones de casos de uso del actor jugador.

- **USR-JUG- MC-01:Ver ayuda del juego**

  Descripción

  Despliega la información necesaria que indica cómo jugar.

  Actores

  Usuario.

Flujo de Eventos

- **Flujo Básico**

  El sistema muestra la información básica que indica cómo jugar, tanto el input como la mecánica lúdica del juego.

- **Flujos Alternativos**

  Ninguno.

- **Precondiciones**

  El actor debe ingresar a la opción "How to Play" desde el Menú principal del Juego.

- **Post condiciones**

  Ninguna.

- **USR-JUG- MC-02:Entrar en un cuarto de juego**

  Descripción

  El actor ingresa a una sala virtual para empezar la partida.

  Actores

  Usuario.

  Flujo de Eventos

  - **Flujo Básico**

    El actor crea un cuarto virtual para empezar la partida e ingresa automáticamente al mismo.

- **Flujos Alternativos**

  En caso de que ya existan cuartos creados el actor puede ingresar en cualquier cuarto ya creado.

- **Precondiciones**

  Debe entrar en un cuarto nuevo o en uno previamente creado donde el número de usuarios no llegue al máximo y mientras el juego no haya empezado.

- **Post condiciones**

  El usuario se registra en el servidor Photon Cloud de forma automática y se le asigna un avatar de juego.

- **USR-JUG- MC-02:Empezar partida**

  Descripción

  El actor indica que está listo para empezar la partida.

  Actores

  Usuario.

  Flujo de Eventos

  - **Flujo Básico**

    El actor informa que está listo para empezar la partida.

  - **Flujos Alternativos**

    En caso de que algún usuario de la red no indique que está listo para empezar, la partida no comienza.

  - **Precondiciones**

    Ninguna.

- **Post condiciones**

  Se despliega en la pantalla un mensaje que indica que el usuario está listo para empezar la partida.

ii. Actor: Game – GUI

En la Figura 10 se pueden observar los casos de uso del actor Game – GUI.

a) GUI-MC-01: Desplegar lista de cuartos virtuales activos.

b) GUI-MC-02: Desplegar regleta de figuras.

c) GUI-MC-03: Desplegar número de vueltas.

d) GUI-MC-04: Desplegar posiciones de todos los jugadores.

e) GUI-MC-05: Desplegar la posición final de la carrera.

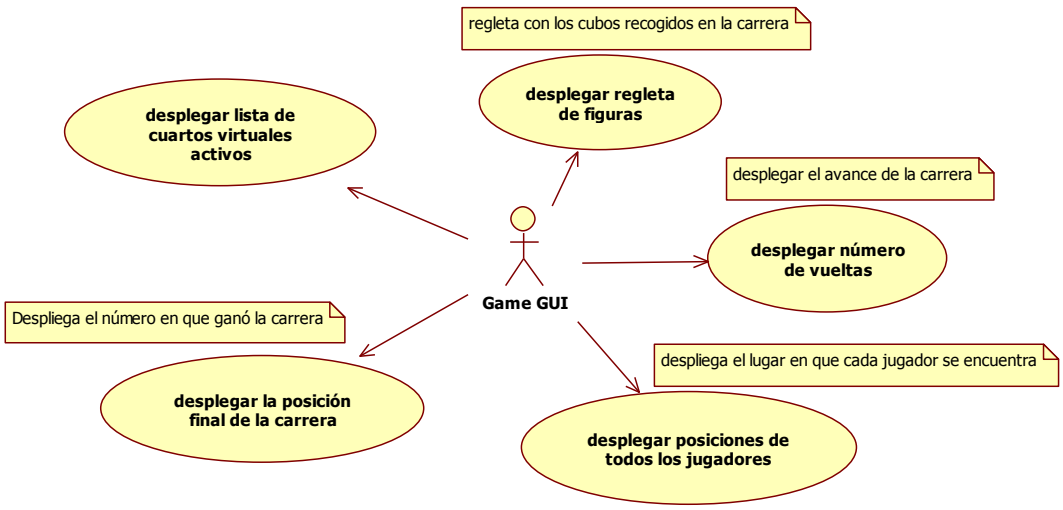

Figura 10 - Casos de uso del Game – GUI

Fuente: Elaborado por los autores de la investigación

- **GUI- MC-01:Desplegar lista de cuartos virtuales activos**

  Descripción

  El sistema muestra la lista de los cuartos virtuales en uso.

  Actores

  Game GUI.

  Flujo de Eventos

  - **Flujo Básico**

    El actor crea y despliega una lista de botones que corresponden a los cuartos virtuales en uso, el nombre del cuarto y el número de usuarios que se encuentran dentro.

  - **Flujos Alternativos**

    Ninguna.

  - **Precondiciones**

    Debe existir más de un cuarto virtual activo.

  - **Post condiciones**

    Ninguna.

- **GUI- MC-02:Desplegar regleta de figuras**

  Descripción

  El sistema muestra la regleta de figuras con las figuras recogidas por el avatar durante la carrera.

  Actores

  Game - GUI.

Flujo de Eventos

- **Flujo Básico**

  El actor despliega una regleta con las figuras recogidas por el avatar durante la carrera de forma automática.

- **Flujos Alternativos**

  Ninguna.

- **Precondiciones**

  Ninguna.

- **Post condiciones**

  Ninguna.

- **GUI- MC-03:Desplegar número de vueltas**

  Descripción

  El sistema muestra la cantidad de vueltas que el avatar a recorrido en la carrera.

  Actores

  Game - GUI.

  Flujo de Eventos

  - **Flujo Básico**

    El actor muestra la cantidad de vueltas que el avatar a recorrido en la carrera.

  - **Flujos Alternativos**

    Ninguna

- **Precondiciones**

  La carrera debe estar en curso.

- **Post condiciones**

  Ninguna

- **GUI- MC-04:Desplegar posiciones de todos los jugadores**

  Descripción

  El sistema muestra la lista de los jugadores y de sus posiciones actuales en la carrera.

  Actores

  Game - GUI.

  Flujo de Eventos

  - **Flujo Básico**

    El actor muestra la lista de los jugadores y de sus posiciones actuales en la carrera.

  - **Flujos Alternativos**

    Ninguna

  - **Precondiciones**

    La carrera debe estar en curso.

  - **Post condiciones**

    Ninguna

- **GUI- MC-05:Desplegar la posición final de la carrera**

  Descripción

  El sistema muestra la posición final con la que el avatar culmino la carrera.

  Actores

  Game - GUI.

  Flujo de Eventos

  - **Flujo Básico**

    El actor muestra la posición final con la que el avatar culmino la carrera.

  - **Flujos Alternativos**

    Ninguna.

  - **Precondiciones**

    El avatar debe culminar la carrera.

  - **Post condiciones**

    Ninguna.

iii. Actor: Avatar

En la Figura 11 se puede observar los casos de uso del actor avatar.

a) AV-MC-01: Generar movimiento vertical.

b) AV-MC-02: Generar rotación horizontal.

c) AV-MC-03: Saltar.

d) AV-MC-04: Enviar información por red.

e) AV-MC-05: Colisionar con objetos estáticos.

f) AV-MC-06: Colisionar con objetos dinámicos.

g) AV-MC-07: Obtener turbo.

h) AV-MC-08: Desactivar componentes.

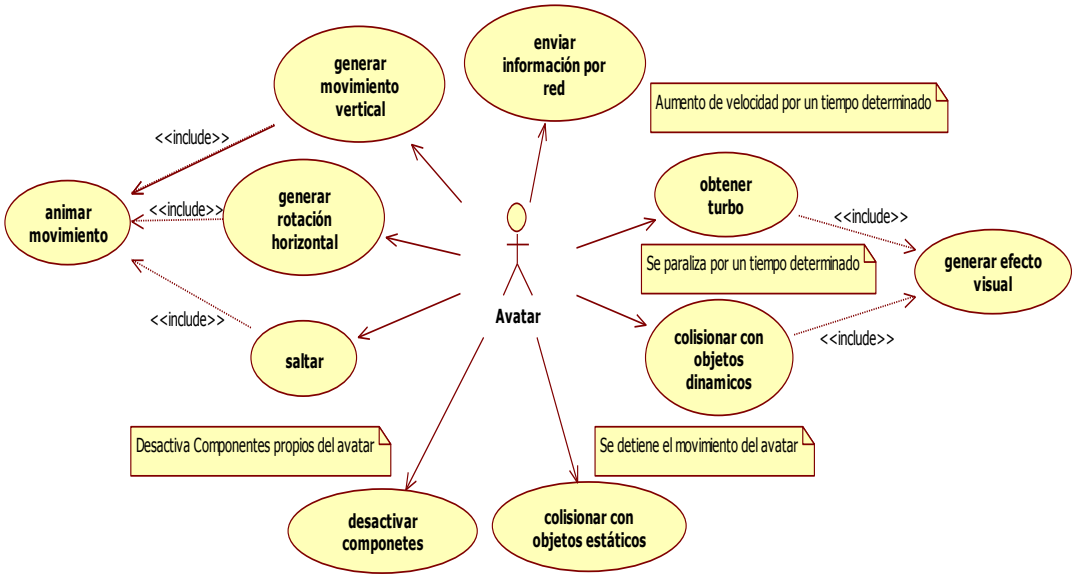

Figura 11 - Casos de uso avatar

Fuente: Ilustración propia

- **AV-MC-01: Generar movimiento vertical**

  Descripción

  Acción por la que el avatar se mueve verticalmente.

  Actores

  Avatar.

  Flujo de Eventos

  - **Flujo Básico**

    A través del input el avatar se mueve verticalmente en el eje de las Z, avanza o retrocede en el mundo virtual.

- **Flujos Alternativos**

  Si el avatar colisiona con un objeto estático o dinámico, se interrumpe el movimiento.

- **Precondiciones**

  El avatar no debe estar paralizado o colisionando con algún objeto estático del juego.

- **Post condiciones**

  Se genera la animación del movimiento.

- **AV-MC-02: Generar rotación horizontal**

  Descripción

  Acción por la que el avatar rota horizontalmente.

  Actores

  Avatar.

  Flujo de Eventos

  - **Flujo Básico**

    A través del input el avatar rota sobre el eje Y.

  - **Flujos Alternativos**

    Ninguna.

  - **Precondiciones**

    El avatar no debe estar paralizado o colisionando con algún objeto estático del juego.

  - **Post condiciones**

    Se genera la animación de la rotación.

- **AV-MC-03: Saltar**

  Descripción

  Acción por la que el avatar salta.

  Actores

  Avatar.

  Flujo de Eventos

  - **Flujo Básico**

    A través del input el avatar se mueve verticalmente sobre el eje Y, y por acción de una fuerza gravitatoria regresa al suelo.

  - **Flujos Alternativos**

    Ninguna.

  - **Precondiciones**

    El avatar no debe estar paralizado o colisionando con algún objeto estático del juego.

  - **Post condiciones**

    Se genera la animación del movimiento.

- **AV-MC-04:Enviar información por red**

  Descripción

  Acción por la que el avatar envía su información al resto de jugadores conectados.

  Actores

  Avatar.

Flujo de Eventos

- **Flujo Básico**

  El actor envía automáticamente la información de su posición y rotación de forma serializada al resto de usuarios en el cuarto virtual.

- **Flujos Alternativos**

  Ninguna.

- **Precondiciones**

  La carrera debe haber empezado.

- **Post condiciones**

  Ninguna.

- **AV-MC-05: Colisionar con objetos estáticos**

Descripción

Acción por la que el avatar colisiona con objetos estáticos del juego.

Actores

Avatar.

Flujo de Eventos

- **Flujo Básico**

  La colisión se produce cuando el componente Collider del avatar choca con el componente Collider de un objeto estático en el juego como los cubos de metal y las paredes de púas regados por la pista.

- **Flujos Alternativos**

  Ninguna.

- **Precondiciones**

  Colisión entre el avatar y un Collider.

- **Post condiciones**

  El avatar se detiene inmediatamente.

- **AV-MC-06: Colisionar con objetos dinámicos**

  Descripción

  Acción por la que el avatar colisiona con objetos dinámicos del juego.

  Actores

  Avatar.

  Flujo de Eventos

  - **Flujo Básico**

    La colisión se produce cuando el componente Collider del avatar entra en un componente Trigger de un objeto dinámico en el juego como las rocas, los cubos lúdicos y los meteoritos en la pista.

  - **Flujos Alternativos**

    Ninguna.

  - **Precondiciones**

    Colisión entre un el avatar y un Trigger.

- **Post condiciones**

  Si el objeto con el que colisiona es el cubo lúdico, el avatar recoge su información y se procesa en la regleta de figuras, adicionalmente el cubo se destruye.

  Si el objeto con el que colisiona es una roca o un meteorito, el avatar se paraliza por un tiempo determinado y se detiene automáticamente por el mismo tiempo.

  Si el objeto con el que colisiona es el área fuera de la pista, el avatar sufre una reducción de velocidad.

  En los tres casos se genera un efecto visual de colisión.

- **AV-MC-07: Obtener Turbo**

  Descripción

  Acción por la que el avatar cambia a velocidad modo turbo durante la carrera.

  Actores

  Avatar.

  Flujo de Eventos

  - **Flujo Básico**

    El avatar obtiene de forma automática un incremento de velocidad por un determinado período de tiempo.

- **Flujos Alternativos**

  Si el avatar ya está en velocidad modo turbo, se reinicia el tiempo de duración del mismo.

- **Precondiciones**

  El avatar debe coger la figura que complete un trío del mismo tipo de las figuras que se muestran en la regleta de figuras.

- **Post condiciones**

  Se genera un efecto visual de la velocidad modo turbo.

- **AV- MC-08: Desactivar Componentes**

  Descripción

  Acción por la que el actor desactiva los componentes que le otorgan funcionalidad controlada por el usuario.

  Actores

  Network Player.

  Flujo de Eventos

  - **Flujo Básico**

    El actor al ser instanciado en el juego desactiva los componentes que le otorgan funcionalidad controlada por el usuario de forma automática, de esta manera este actor es controlado únicamente con la información que recibe de los otros avatars que son controlados por los usuarios de la red.

- **Flujos Alternativos**

  Ninguno

- **Precondiciones**

  Otros usuarios deben conectarse e ingresar al cuarto de juego.

- **Post condiciones**

  Ninguna

iv. Actor: Jugador controlado con IA (PC Player)

En la Figura 12 se puede observar los casos de uso del actor PC Player.

1. PP-MC-01: Generar movimiento inteligente.

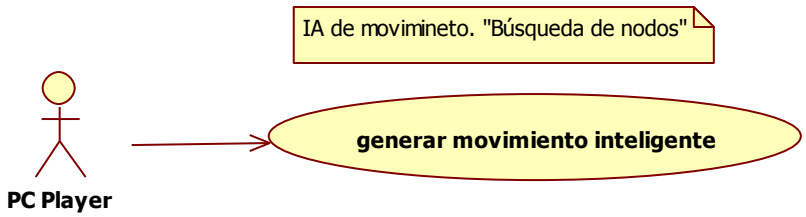

Figura 12 - Casos de uso del actor PC Player

Fuente: Elaborado por autores de la investigación

- **PP- MC-01: Generar movimiento inteligente**

  Descripción

  Acción por la que el actor sigue un algoritmo y se mueve automáticamente a través de la pista sin ser controlado por ningún usuario en la red.

Actores

PC Player.

Flujo de Eventos

- **Flujo Básico**

  El actor procesa un algoritmo de búsqueda y se dirige al nodo más próximo en la pista.

  Su posición y rotación se modifican automáticamente dentro del algoritmo.

- **Flujos Alternativos**

  Ninguno

- **Precondiciones**

  La carrera debe empezar.

- **Post condiciones**

  Ninguna.

3.1.5. Requerimientos no funcionales

- El sistema debe funcionar en las siguientes plataformas: iOS, Android, Windows, Web, y MAC OS X.
- El número mínimo de jugadores para empezar con la partida debe ser de uno sin incluir al PC-Player.

3.2. Diagramas de secuencia

Se encuentran en el anexo A del presente trabajo y representan el diseño de navegación de la aplicación.

3.3. Diseño Conceptual

3.3.1. Diagrama de clases

Se diseñaron las clases que intervienen en el video juego, adicionalmente en el diagrama se incluyeron como clases el Framework de Unity 3D Game Engine y del Framework de Photon Cloud, las cuales permiten el manejo de objetos 3D y la conexión con la Nube.

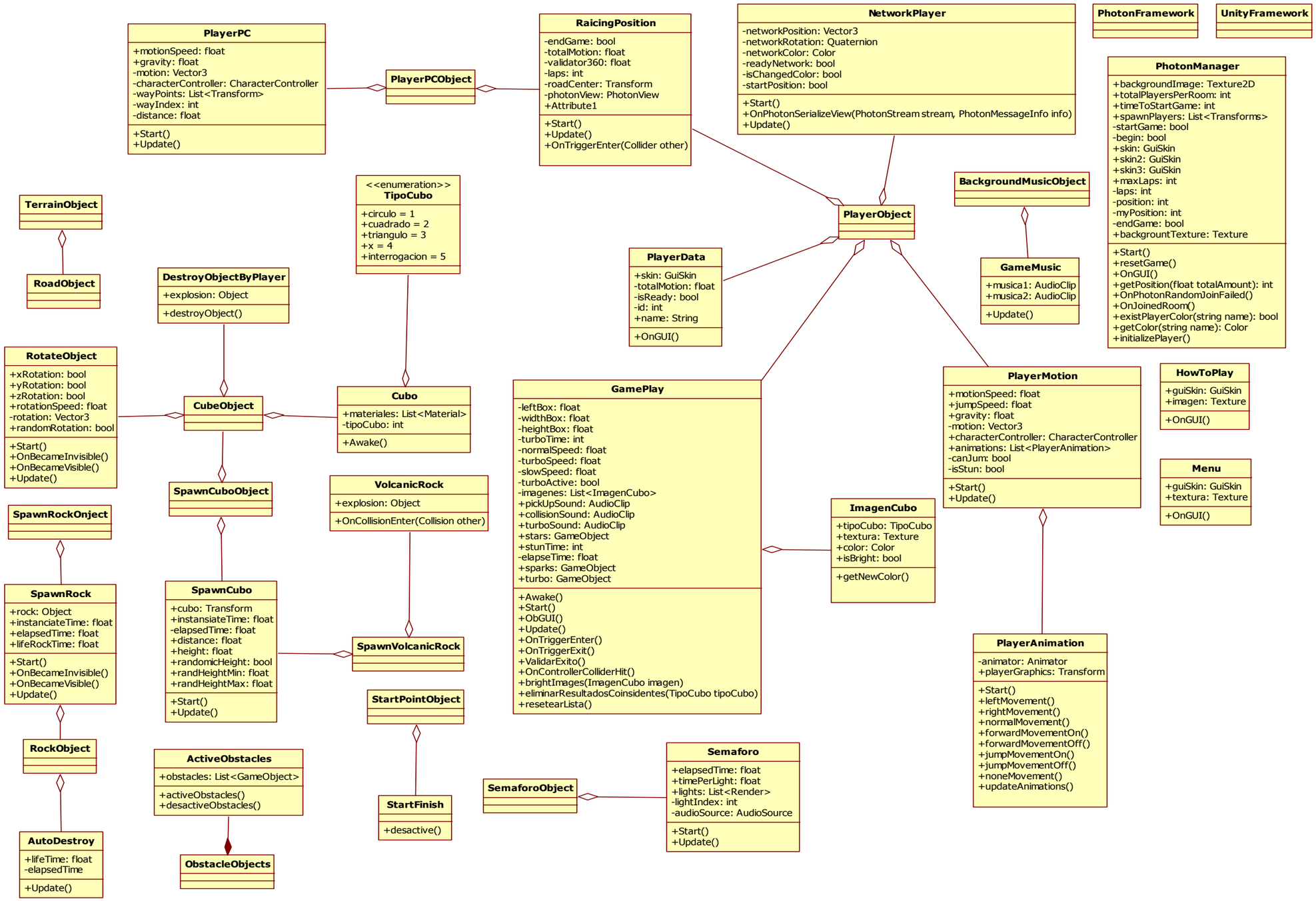

Figura 13 - Diagrama de clases MC

Fuente: Elaborado por los autores de la investigación

3.4. Diseño Navegacional

El sistema está basado en escenas, por lo tanto contiene una mezcla adecuada de estética, contenido y tecnología. Posee los siguientes objetos y contextos navegacionales:

i. Objetos Navegacionales

- Escena del Menú Principal.
- Escena de ayuda del juego.
- Escena de la sala principal en la red (lobby).
- Escena de la competencia.

ii. Contextos Navegacionales

- Unirse a un cuarto de juego.
- Competir.

3.4.1. Esquema de Contextos Navegacionales

El sistema posee una estructura compuesta basada en el concepto de diseño arquitectónico de la Ingeniería de Software.

i. Menú Principal

En la Figura 14 se muestra el contexto navegacional del menú principal de la aplicación:

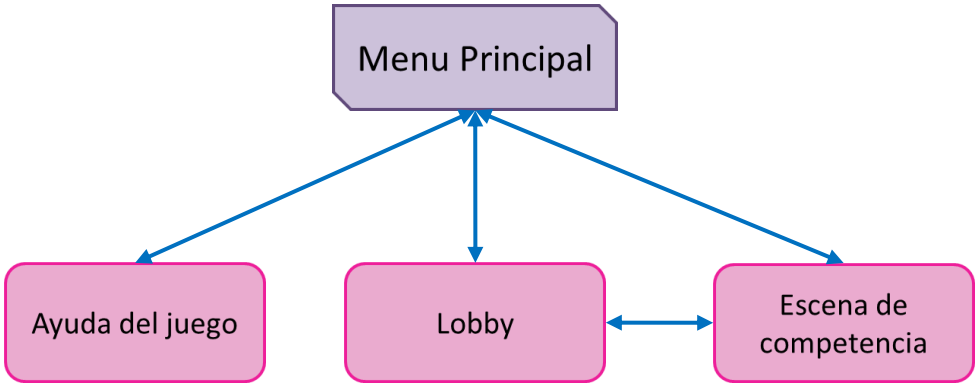

Figura 14 - Esquema de contexto Menú Principal

Fuente: Elaborado por los autores de la investigación

3.4.2. Arquitectura del Sistema

En la Figura 15 se puede observar la arquitectura del video juego de tipo cliente - servidor, en este caso se dispone de dos servidores que se encuentran en la Nube. Photon Server es un servicio proporcionado por Photon Cloud y permite la conexión simultánea de varios jugadores. También se encuentra Unity Support Server en el cual se alojan los plugins que permiten que juegos desarrollados con Unity 3D Game Engine puedan ejecutarse en los exploradores Web existentes.

3.5. Diseño de Interfaz Abstracta

Las interfaces están basadas tanto en una arquitectura de contenido (forma en la que los objetos se estructuran para su presentación y navegación) como en una arquitectura de Aplicación Desktop, Móvil y Web

(forma en la que la aplicación se estructura para gestionar la interacción del usuario).

Se detallan a continuación las interfaces del usuario Jugador, ya que posee una interacción directa con el sistema.

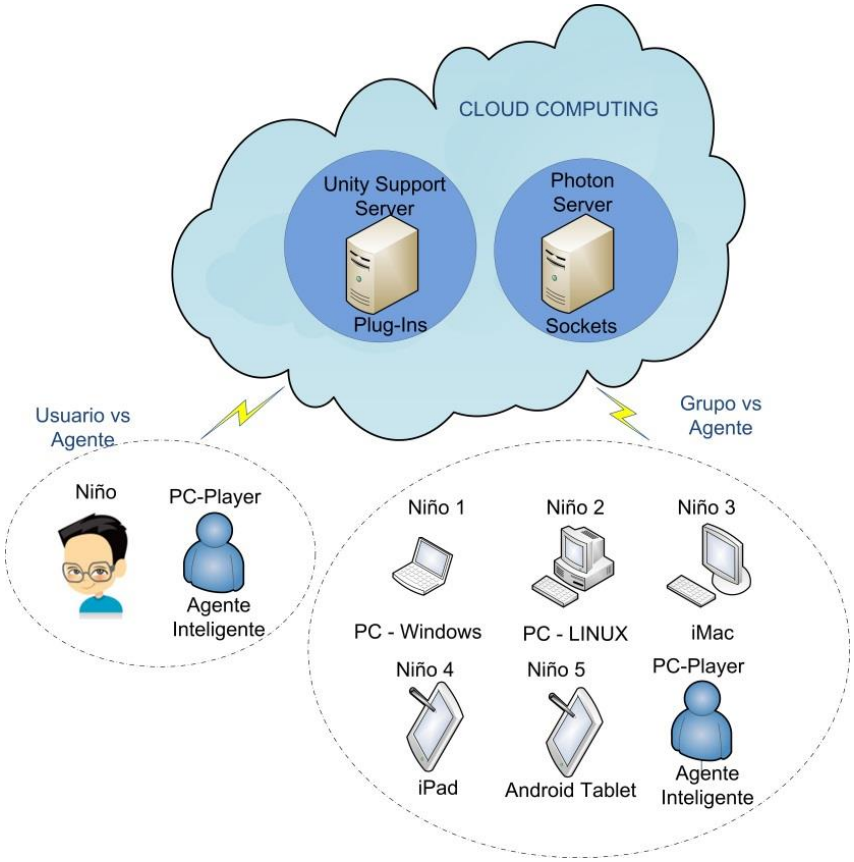

Figura 15 - Arquitectura del Multiplayer Challenge

Fuente: Elaborado por los autores de la investigación

3.5.1. Vista de Datos Abstracta

- Nodo de la escena del Menú Principal.
- Nodo de la escena de Como Jugar.
- Nodo de la escena del Lobby.

CAPÍTULO 4

IMPLEMENTACIÓN Y PRUEBAS DEL VIDEO JUEGO CON UNITY 3D GAME ENGINE

4.1. Unity 3D Game Engine

4.1.1. Definición

Es un motor gráfico 3D para Windows y Mac que viene empaquetado como una herramienta para crear juegos, aplicaciones interactivas, visualizaciones y animaciones en 3D en tiempo real. Unity 3D Game Engine puede publicar contenido para múltiples plataformas como Windows, Mac, Nintendo Wii y iPhone. El motor también puede publicar juegos basados en la web usando el plugin Unity webplayer, que es un componente que permite interpretar código y los mapas de archivos 3D. (Creighton, 2010)

4.1.2. Características

i. Instalación

Puede ser instalado bajo ambientes Windows y Mac, para lo cual se puede descargar gratuitamente la licencia de prueba en la dirección Web: http://unity3d.com/unity/download/ Una vez descargada e instalada la aplicación aparecerá el siguiente ícono de activación del paquete. (Ver Figura 16)

Figura 16 - El ícono de carga del paquete Unity

Fuente: http://stats.unity3d.com/

ii. Interfaz de Usuario

La Interfaz de Usuario de Unity 3D Game Engine tiene 5 áreas principales de trabajo, numeradas en la Figura 17.

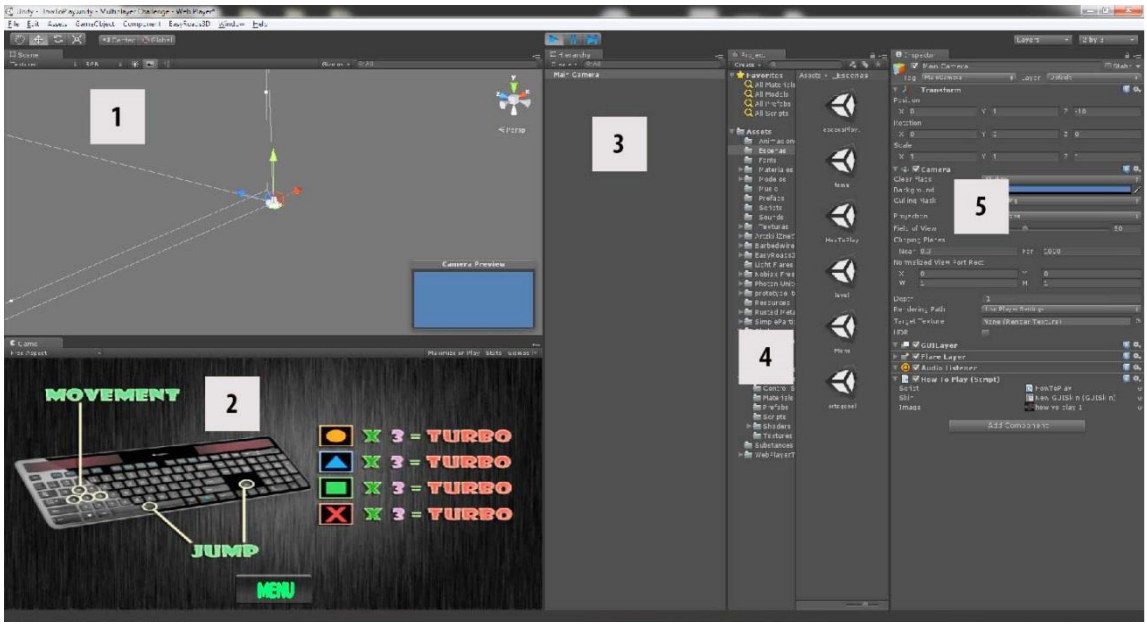

Figura 17 - Interfaz de Usuario de Unity 3D Game Engine

Fuente: Elaborado por los autores de la investigacion

1 ***Vista de Escena:*** Es el área de construcción de cada escena de manera visual en Unity 3D Game Engine.

2 ***Vista de Juego:*** Se obtiene una previsualización del video juego. En cualquier momento se lo puede reproducir y probarlo en esta vista.

3 ***Vista de Proyecto:*** Es la librería de assets para un video juego en Unity 3D Game Engine. Se puede importar objetos 3D a la librería de distintas aplicaciones, texturas y crear otros objetos como Scripts.

4 ***Vista de Jerarquía:*** La vista de jerarquía contiene todos los objetos en la escena actual.

5 ***Vista de Inspector:*** Al seleccionar un objeto la vista de inspector mostrará las propiedades de este, donde se pueden personalizar las características. También contiene la configuración para ciertas herramientas como la del generador de terrenos si se tiene uno seleccionado.

iii. Modos de Visualización

Por defecto la vista de escena tiene una perspectiva 3D. Se puede cambiar esto por un número de vistas Ortográficas: top-down, side, front con el “Gizmo de Perspectiva” que se encuentra en la parte superior derecha de la vista de la escena. (Ver Figura 18)

Figura 18 - Gizmo de Perspectiva

Fuente: Elaborado por autores de la investigación

- ***Perspectiva 3D top-down (arriba-abajo):*** La escena y los objetos son vistos desde arriba y desde abajo, para lo cual se hace clic en el cono verde del “Gizmo de Perspectiva.”
- ***Perspectiva 3D side (lado):*** La escena y los objetos se visualizan desde un lado, para lo cual se hace clic en el cono rojo del “Gizmo de Perspectiva”.
- ***Perspectiva 3D front (frontal):*** Se visualiza la escena y los objetos vistos desde la parte frontal, para lo cual se hace clic en el cono azul del “Gizmo de Perspectiva”.

iv. Configuración de la Visualización

En la esquina superior izquierda de la vista de escena se encuentra un conjunto de botones para cambiar la configuración de la visualización. De izquierda a derecha se tienen tres botones: Render Mode, Color Modes, Light On/Off, Skybox, Fogs and Lens Flare Effects. (Ver Figura 19)

Figura 19 - Configuración de la Visualización

Fuente: Elaborado por los autores de la investigación

- ***Render Mode:*** Por defecto se encuentra en “Textured”. Al hacer clic en este botón aparece una lista desplegable con diferentes opciones de renderizado:
  - ***Textured:*** Las superficies se renderizan en la vista en tiempo real.

- ***Wireframe:*** Las superficies no se renderizan, solo se ve la malla.
- ***Textured Wireframe:*** Las superficies se renderizan, pero también se ve la malla.

- ***Color Modes: P***or defecto aparece como “RGB”. Al hacer clic sobre este botón aparece una lista desplegable que muestra los modos de color disponibles: *RGB*, *Alpha*, *Overdraw, Minimaps.*
- ***Light On/Off:*** Enciende o apaga la iluminación del escenario. Sin iluminación se tiene un mejor rendimiento, pero la escena aparecerá oscura. La iluminación encendida afecta a toda la escena generando brillos y sombras.
- ***Skybox, Fogs and Lens Flare Effects:*** El último botón activa y desactiva estos tres efectos. Es útil por razones de rendimiento o visibilidad al trabajar sobre una escena.

v. Botones de Control

A lado de las opciones de visualización se tiene una fila con 4 botones, como se muestra en la Figura 20. Se puede usar Q, W, E, R para alternar entre cada uno de los controles, que se detalla a continuación:

Figura 20 - Botones de Control

Fuente: Elaborado por los autores de la investigación

- ***Hand Tool (Q):*** Permite moverse alrededor en la vista de escena. Presionar *Alt* permitirá rotar, *Command/Ctrl* permitirá hacer zoom y *Shift* incrementará la velocidad de movimiento mientras se usa la herramienta.
- ***Translate Tool (W):*** Permite mover cualquier objeto seleccionado en la escena en los ejes (x, y, z).
- ***Rotate Tool (E):*** Rota cualquier objeto seleccionado en la escena.
- ***Scale Tool (R):*** Escala cualquier objeto seleccionado en la escena.

4.1.3. Librerías de Unity 3D Game Engine

Las principales librerías que maneja el Unity 3D Game Engine son cuatro de las cuales se han utilizado tres, a continuación se explican cada una:

i. Creación de Escenas (Scenes Creator)

Unity 3D Game Engine tiene un DLL que maneja la creación de escenas 3D donde se ubican todos los elementos u objetos del juego como planos, terrenos, cielo, personajes, etc., como se puede ver en la Figura 21.

Figura 21 - Creación de escenas

Fuente: Elaborado por los autores de la investigación

ii. Generador de Terrenos (Terrain Generator)

Unity 3D Game Engine tiene un DLL para generar terrenos, los mismos que son generados como una malla plana que se puede texturizar y esculpir sin salir del editor. Tienen algunas propiedades como longitud del terreno y algunas propiedades que controlan el nivel de detalle del terreno, como se puede ver en la Figura 22 y 23. A continuación se explicará cada una de las propiedades más importantes de los terrenos (terrain):

- ***Width:*** El ancho en metros del terreno.
- ***Lenght:*** La longitud en metros del terreno.
- ***Height:*** La máxima altura en metros del terreno.
- ***Height map resolution:*** La resolución de la altura del mapa (height map).

- ***Detail Resolution:*** La resolución de los detalles del mapa, más resolución, más precisión a la hora de dibujar los detalles sobre el terreno y colocar objetos.
- ***Control Texture Resolution:*** La resolución de las texturas pintadas sobre el terreno, más resolución produce más detalle, menor solución produce más rendimiento.
- ***Base Texture Resolution:*** Esta es la resolución base de la textura que se renderiza.

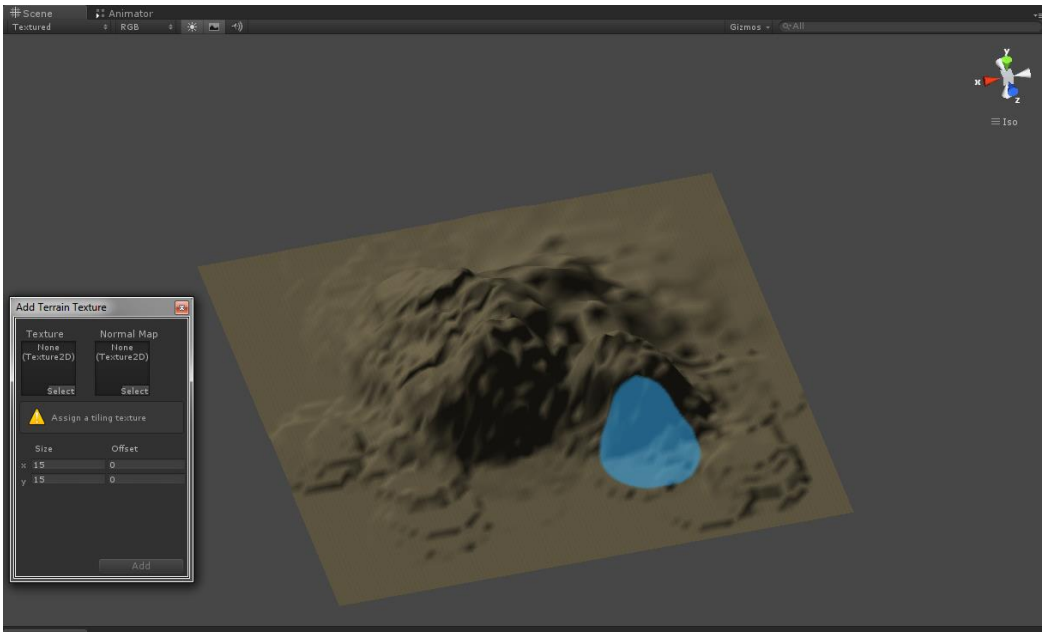

Figura 22 - Generador de Terrenos

Fuente: Elaborado por autores de la investigación

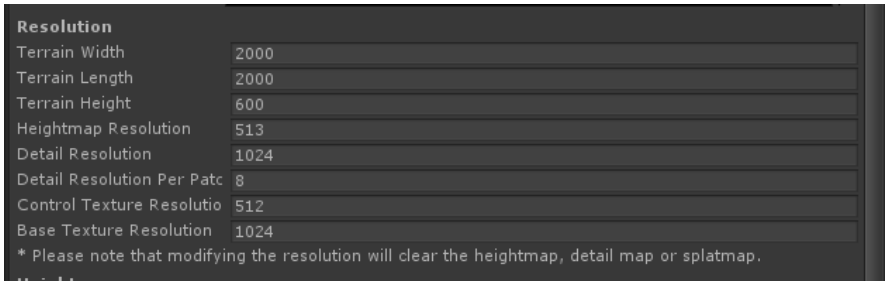

Figura 23 - Propiedades del terreno generado

Fuente: Elaborado por autores de la investigación

iii. Configuración del Renderizado (Render Settings)

Unity 3D Game Engine tiene una DLL para configurar el renderizado de los objetos 3D de un juego y añadir efectos especiales como: Niebla (Fog), Color de la Niebla (Fog Color), Luz de Ambiente (Ambient Light), Material de la Caja del Cielo (Skybox Material), Fuerza de la Luz (Halo Strenght), Fuerza del Fuego (Flare Strenght), Textura de la Luz (Halo Texture), Mancha de Galleta (Spot Cookie), como se puede ver en la Figura 24.

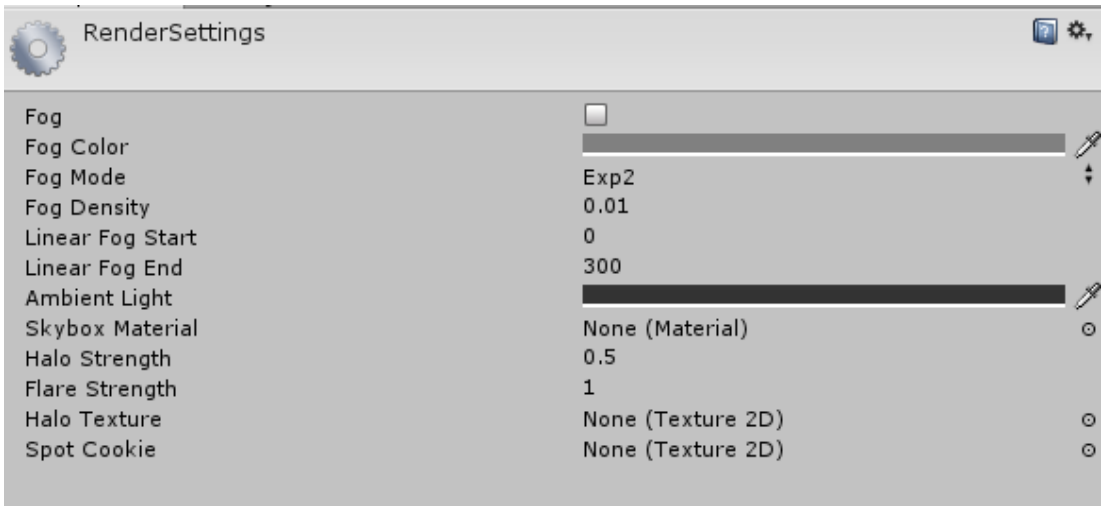

Figura 24 - Ajuste del Renderizado

Fuente: Elaborado por los autores de la investigación

4.1.4. Estructura de la interfaz visual hecho con Unity 3D Game Engine

En la Figura 25 se puede apreciar la interfaz de bienvenida del juego.

Figura 25 - Estructura de la interfaz de bienvenida del video juego hecho con Unity 3D Game Engine

Fuente: Elaborado por autores de la investigación

i. Componentes

Son programas independientes entre sí que poseen sus propias clases dentro del núcleo de Unity 3D Game Engine. La mayor parte vienen instalados con la aplicación y otros pueden adicionarse.

La instalación básica de Unity 3D Game Engine incluye los siguientes componentes:

- El componente **ScenesCreator**.
- El componente **TerrainGenerator**.
- El componente **RenderSettings**.
- El componente **FirstPerson Controller**.
- El componente de programación de scripts de Unity 3D Game Engine conocido como **MonoBehaviour**.

4.2. Construcción del video juego

4.2.1. Elaboración de los modelos 3D en Autodesk Maya

Se usó la herramienta Autodesk Maya para crear todos los modelos que se encuentran en el video juego. En la Figura 26 se puede observar el modelamiento del avatar, en la Figura 27 se observa el modelo de la meta del juego, en la Figura 28, se observa el modelo de la machacadora.

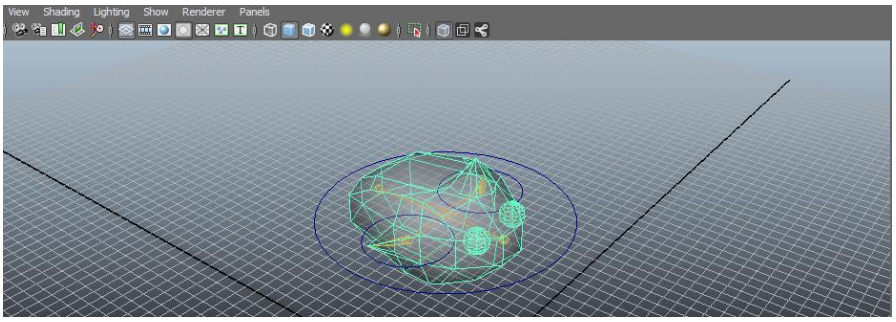

Figura 26 - Modelamiento del avatar en Autodesk Maya

Fuente: Elaborado por los autores de la investigación

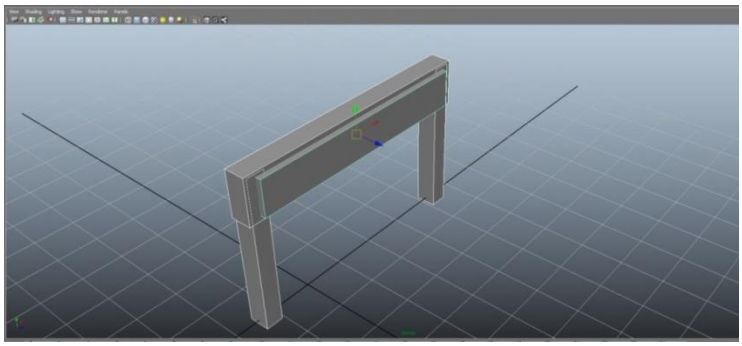

Figura 27 - Modelo de la meta en Autodesk Maya

Fuente: Elaborado por los autores de la investigación

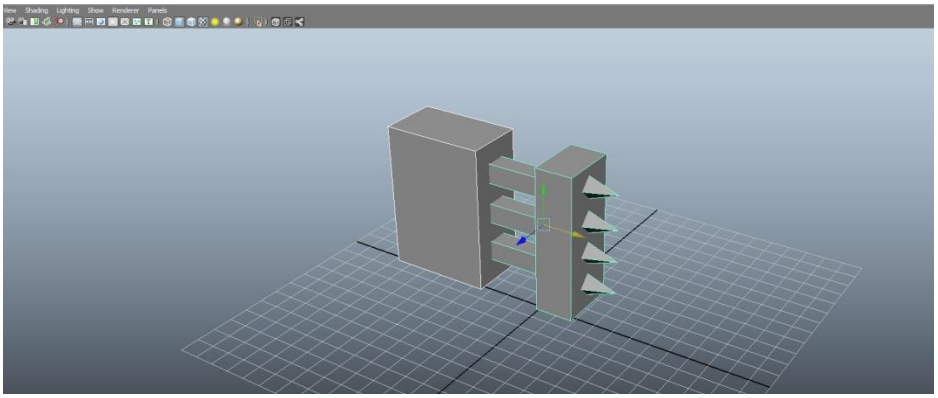

Figura 28 - Modelo de la machacadora

Fuente: Elaborado por los autores de la investigación

4.2.2. Desarrollo del juego en Unity 3D Game Engine

El video juego fue desarrollado en su totalidad con Unity 3D Game Engine y con la herramienta de programación Mono C# para la creación de las clases del juego del componente GameApp creado, las mismas que son:

- **PlayerPC:** Controla todo el movimiento del avatar de la PC durante la carrera.
- **RaicingPosition:** Controla la posición de cada avatar en la carrera y el número de vueltas que tiene.
- **NetworkPlayer:** Envía y recibe información a través de la red.
- **PhotonManager:** Controla toda la información de la red, actualiza y muestra en pantalla la información de todos los jugadores.
- **DestroyObjectByPlayer:** Esta clase es el componente de los cubos lúdicos y permite que se autodestruyan después de ser colisionados únicamente por los avatars.

- **PlayerData:** Contiene la información de cada jugador, la misma que es actualizada por el componente NetworkPlayer cada vez que un avatar entra al juego.
- **GameMusic:** Reproduce música de fondo en el juego.
- **RotateObject:** Es el componente de los objetos 3D que están diseñados para rotar de forma aleatoria o predeterminada.
- **Cubo:** Es el componente de cada uno de los cubos lúdicos, permite la instanciación automática de los mismos, asignación de color y figura.
- **GamePlay:** Maneja cada avatar a lo largo del juego.
- **PlayerMotion:** Controla el input de entrada para el movimiento del avatar:
- **PlayerAnimation:** Controla la animación del avatar dependiendo del movimiento que se realice.
- **ImagenCubo:** Representa la información 2D de cada figura en al regleta lúdica.
- **VolcanicRock:** Controla el movimiento gravitacional de las rocas volcánicas.
- **SpawnCubo:** Controla el tiempo de instanciación e instancia al cubo después de un tiempo determinado.
- **ActiveObstacles:** Activa o desactiva los obstáculos en el juego.
- **SpawnRock:** Controla el tiempo de instanciación e instancia a las rocas después de un tiempo determinado.
- **AutoDestroy:** Destruye un objeto después de un tiempo determinado.

4.3. Desarrollo e implementación de mecánicas del video juego con Inteligencia Artificial

El presente juego maneja “x” mecánicas que funcionan de forma automática, las cuales se desarrollaron usando tres técnicas de Inteligencia Artificial que son:

4.3.1. Heurísticas

Esta técnica se usó para la generación de algoritmos de validación de estados de los obstáculos, ya que estos deben de activarse siempre y cuando el avatar se encuentre en la respectiva área de obstáculos. Tanto las rocas como los meteoritos tienen un ciclo de vida y se crean automáticamente y destruyen de forma aleatoria en puntos estratégicos 3D distribuidos en el entorno. Existen paredes invisibles que indican a los algoritmos encargados de generar los respectivos obstáculos, que el avatar ha entrado en el área y que empiece el ciclo de vida de la mecánica de juego determinada.

Esta técnica también es la encargada de validar la regleta de figuras geométricas recogidas por el avatar. Estas figuras tienen un identificador propio y el algoritmo heurístico se encarga de evaluar cuáles de todas las figuras ya se encuentran 2 y 3 veces repetidas en la regleta, donde ejecuta automáticamente un evento respectivo por cada estado.

4.3.2. Búsqueda de Caminos

Este tipo de técnica usada principalmente en laberintos, se utilizó para dar al avatar una inteligencia propia y sea capaz de recorrer por si solo toda la carrera, evitando en lo posible los obstáculos y representando un enemigo difícil de vencer en la carrera. La técnica comprende un conjunto de puntos 3D esparcidos estratégicamente en la pista y a través de un algoritmo de búsqueda, el avatar es capaz de ubicar el punto más cercano en la pista y de esta forma avanzar progresivamente como si lo hiciera un jugador más del juego. Añadiendo fórmulas trigonométricas se logró validar el estado actual y el número de vueltas que el avatar PC-Player ha recorrido en la carrera. La ecuación utilizada para calcular la distancia entre dos puntos es la siguiente:

$$\mathrm{d} = \sqrt{({x_2}^2 - {x_1}^2) + ({y_2}^2 - {y_1}^2) + ({z_2}^2 - {z_1}^2)}$$

Dónde:

$\boldsymbol{x_1}$: Valor de la coordenada en el eje x del punto donde se encuentra el avatar.

$\boldsymbol{x_2}$: Valor de la coordenada en el eje x del punto donde se quiere llegar.

$\boldsymbol{y_1}$: Valor de la coordenada en el eje y del punto donde se encuentra el avatar.

$\boldsymbol{y_2}$: Valor de la coordenada en el eje y del punto donde se quiere llegar.

z_1: Valor de la coordenada en el eje z del punto donde se encuentra el avatar.

z_2: Valor de la coordenada en el eje z del punto donde se quiere llegar.

4.3.3. Planificación

Esta técnica se usó para la animación del avatar, esta planificación automática determina la animación que debe ejecutar el modelo 3D correspondiente al avatar dependiendo del estado actual en el que se encuentra y de la variable de entrada que se le asigne. Esta red de estados, se formó usando 3 variables de entrada: *RUN*, que representa la movilidad del avatar, *JUMP*, que representa si ha saltado el avatar y *DIRECTION*, que representa la dirección de movimiento del avatar (Ver Figura 61 en el Anexo C).

4.4. Pruebas de la aplicación

Se ha establecido el siguiente proceso de pruebas que inicia con la verificación de una adecuada navegabilidad, estándar de colores, tamaño y tipo de letra; para finalizar con la verificación de infraestructura y seguridad de la aplicación. (Ver Figura 29)

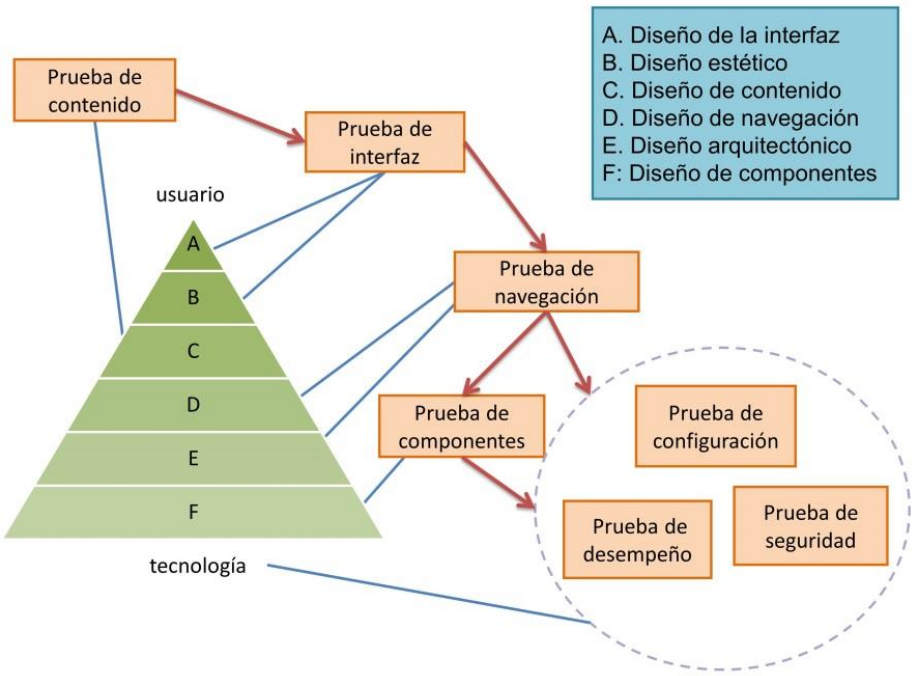

Figura 29 - Proceso de prueba

Fuente: Pressman, 2006, pág. 609

4.4.1. Prueba de contenido

Se ha realizado con varios usuarios una revisión minuciosa de los siguientes tipos de contenido:

- Estático**:** Referente a la información estática que se muestra en la aplicación.

A través de la técnica de observación y lectura de todos los elementos y enlaces de la aplicación; se ha logrado corregir los 4 aspectos relevantes de los contenidos que son:

- Errores tipográficos y/o equívocos gramaticales.
- Errores semánticos (información incompleta o ambigua)
- Errores en la organización de la información para ser mostrada al usuario final.

4.4.2. Prueba de interfaz de usuario

Se han tomado en cuenta las siguientes consideraciones.

i. Pruebas de mecanismos de la interfaz

- *Enlaces:* Cada uno de los enlaces de la aplicación, sean enlaces internos o externos a la aplicación, se enlacen al objeto deseado.
- *Ventanas Dinámicas:* La navegación en cada una de las ventanas de la aplicación maneja memoria dinámica para construir y destruir los objetos que tiene cada una de estas con los scripts de programación hechos en Mono C# para garantizar su correcto despliegue.

ii. Prueba de facilidad de uso

Para este tipo de pruebas se ha tomado en consideración aspectos como: grado de usabilidad (tiempo en encontrar lo que se busca), interacción con el usuario (menús desplegables, botones, estética [colores]), despliegue (resolución de la pantalla).

4.4.3. Prueba de navegación

Para la realización de esta prueba se ha establecido una verificación de todos los enlaces de la aplicación.

Se ha analizado junto a los usuarios que los enlaces creados lleven hacia el contenido adecuado y sobretodo que los enlaces sean comprensibles durante la navegación.

Se ha verificado que los nombres de los nodos sean significativos para los usuarios, como por ejemplo: a) Menu; b) How to Play; c) Start in New Room. Con esta prueba se ha logrado ejercitar ampliamente la navegación de la aplicación por parte de los desarrolladores y de los usuarios finales.

4.4.4. Prueba de configuración

Se ha analizado la arquitectura (Cliente / Servidor) que maneja Unity 3D Game Engine y Photon Cloud para crear aplicaciones de tipo MMO para las diferentes plataformas y se ha especificado lo siguiente:

- Se ha instalado el plugin de Unity para que funcione el video juego en un browser.
- Se ha generado un archivo ejecutable que corra bajo Windows, un applet para que corra en la Web, una aplicación para iOS, una aplicación para Android.

4.4.5. Prueba de seguridad

La aplicación no implementa ninguna seguridad de acceso, por lo que puede cualquier usuario acceder a la aplicación. Además la aplicación puede ser copiada e instalada en cualquier plataforma y no tendrá ningún problema de funcionamiento.

4.4.6. Prueba de desempeño

Con esta prueba se determinó cómo la aplicación y su entorno de trabajo bajo una arquitectura Cliente-Servidor respondió a varias condiciones de carga y soportó adecuadamente las transacciones en un ambiente en la Nube, con el número máximo de 6 usuarios a la vez. En este apartado no se aplica la fórmula de prueba de aplicaciones Web y distribuidas propuesta en el libro de Roger Pressman de Ingeniería de Software, debido a que no es una aplicación Web.

4.5. Despliegue de la aplicación

Para la puesta en producción de la aplicación es necesario tener lo siguiente:

- Compilar la aplicación con todos los componentes necesarios para que funcione adecuadamente.
- Generar un archivo ejecutable de la aplicación para instalar y correr en cualquier computadora con el sistema operativo Windows.
- Generar la aplicación con extensión “.apk” para el sistema operativo Android.
- Generar la aplicación para iOS, para instalarla en los dispositivos registrados en la cuenta de desarrollador.

4.6. Evaluación de resultados

Para probar la aplicación se seleccionó como proyecto piloto una escuela privada del cantón Quito, para lo cual se tomó una muestra de 162 alumnos de una población total de 216, que corresponden a los grupos de niños entre tercero y quinto grado de educación básica, donde cada grupo tiene un total de 72 estudiantes distribuidos en tres paralelos (A, B, C). Se analizó el número de estudiantes que vencieron al jugador PC-Player. De este análisis se pudo determinar los siguientes resultados:

- Para el grupo de tercero de básica, 4 estudiantes ganaron al jugador PC-Player;
- Para el grupo de cuarto de básica, 10 estudiantes ganaron al jugador PC-Player;
- Para el grupo de quinto de básica, 27 estudiantes ganaron al jugador PC-Player.

Estos resultados se muestran en la Tabla 3 con sus respectivos diagramas de barras en la Figura 30, de acuerdo al género y grado seleccionados:

Tabla 3 - Resultados de la carrera

Niños	Masculino		Femenino	
	Total	Porcentaje	Total	Porcentaje
Tercer Grado	4	13.33	0	0
Cuarto Grado	8	26.67	2	18.18
Quinto Grado	18	64.28	9	81.82
Total	**30**	**100**	**11**	**100**

Fuente: Elaborado por los autores de la investigación

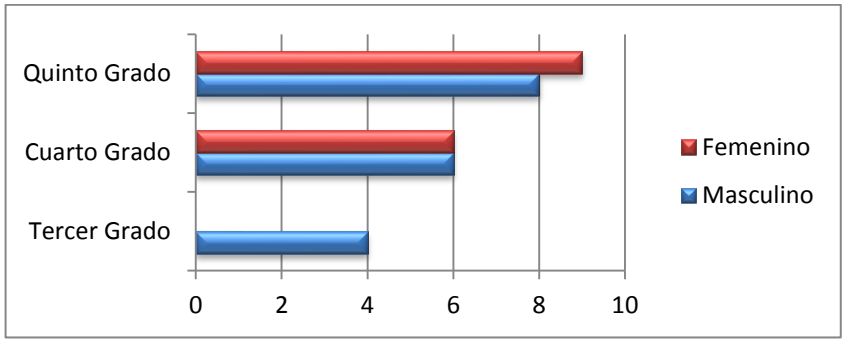

Figura 30 - Gráfico de barras de resultados de la carrera

Fuente: Elaborado por autores de la investigación

Los resultados muestran que este programa ayudó a estimular el raciocinio lógico y espacial, el desarrollo psicomotriz en el área socio-personal, de niños entre 7 y 10 años, mediante la identificación y recolección de figuras básicas en una carrera contra un agente inteligente. Se planteó a un grupo de estudiantes el reto de ganar a este agente inteligente (PC-Player). En la primera prueba algunos estudiantes lograron ganarle al agente inteligente. Luego de un entrenamiento de cerca de 2 semanas hubo un incremento de un 60% de estudiantes que lograron ganarle al agente inteligente. Desde el punto de vista de la Inteligencia Artificial, se pudo comprobar que la técnica de búsqueda de caminos permitió al avatar tener un desempeño competitivo durante la carrera.

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- El uso del Unity 3D Game Engine permitió crear una aplicación de software educativo para varias plataformas, reutilizando el código, ya que este motor tiene un grupo de librerías para renderizado, manejo de física básica y colisiones entre objetos en 3D que es lo básico para programar este tipo de aplicaciones.
- Autodesk Maya permitió crear y animar los modelos 3D del video juego, ya que el modelamiento se basa en la deformación de polígonos básicos y del uso de la extrusión como técnica básica de deformación de objetos en 3D.
- El motor de red Photon Cloud permitió utilizar los servicios de red para la conexión en tiempo real a la Nube, lo cual simula un ambiente de trabajo distribuido, donde varios usuarios (jugadores) acceden al mismo tiempo al juego.
- La integración de Unity 3D Game Engine con Autodesk Maya y Photon Cloud se logró mediante el uso de un lenguaje de programación de alto nivel como lo es C#, el cual permite crear librerías de clases para varias plataformas de una manera eficiente y portable.

- El uso de la técnica de Inteligencia Artificial, búsqueda de caminos, permitió que el avatar (PC-Player) pueda recorrer la pista de una manera eficiente, y ser competitivo con los otros usuarios (jugadores), ya que este avatar no puede adquirir la velocidad modo turbo durante toda la carrera.
- La técnica de planificación permitió animar al avatar de acuerdo al estado en que este se encuentra durante la carrera.
- La metodología OOHDM con UML aportó con diagramas útiles y prácticos que permitieron llevar un proceso de desarrollo organizado y eficiente.
- Con respecto a la parte investigativa, se tuvo que adquirir nuevos conocimientos referentes a Unity 3D Game Engine, y al motor de red Photon Cloud para la conexión distribuida de la aplicación, bajo una arquitectura cliente servidor con Cloud Computing.
- Los motores de juegos (game engines), las herramientas de diseño y modelado 3D, y las herramientas de programación orientadas a objetos se han ido fortaleciendo, tanto en sus arquitecturas como en sus funcionalidades en estos últimos años.

5.2. Recomendaciones

- Para la adecuada selección de componentes adicionales a ser instalados en Unity 3D Game Engine, se recomienda escoger los que tengan varios comentarios positivos en los foros y con mayores

votaciones, ya que serían los más calificados por su estabilidad y funcionalidad.

- Utilizar Unity 3D Game Engine cuando cubra hasta más de un 40% de los requerimientos solicitados por los usuarios, ya que con su variedad de extensiones y documentación actualizada se pueden adaptar y crear las funcionalidades adicionales.
- Formar grupos de trabajo de al menos dos personas o de pares para que uno de ellos se centre al diseño y modelado 3D, mientras que el otro se dedique al proceso de programación para acortar los tiempos de desarrollo e implementación de video juegos. Además se recomienda que tengan conocimientos básicos de programación orientada a objetos y de diseño gráfico, para que en el proceso de creación de video juegos adquieran los conocimientos extras necesarios para cumplir con este objetivo.
- Utilizar Unity 3D Game Engine para la creación y mantenimiento de video juegos, ya que ahorra tiempo y dinero, logra una verdadera concepción de las tareas de un desarrollador de aplicativos de simulación 3D.
- Se recomienda que se dé aplicaciones de la Inteligencia Artificial en video juegos, ya que en estos se puede entender de una manera más atractiva.

REFERENCIAS BIBLIOGRÁFICAS

Albahari, B., Drayton, P., & Merrill, B. (2002). *C# Essentials.* O'Reilly Media, Inc.

Aleixos Borrás, N., Piquer Vicent, A., Galmes Gual, V., & Company Calleja, P. (Junio de 2002). ESTUDIO COMPARATIVO DE APLICACIONES CAD DE MODELADO.

Alfonso Galipienso, M. I., Cazorla Quevedo, M. A., Colomina Pardo, O., Escolano Ruiz, F., & Lozano Ortega, M. A. (2003). *Inteligencia artificial: modelos, técnicas y áreas de aplicación.* Paraninfo.

Autodesk, Inc. (2013). *Autodesk, Inc.* Recuperado el 17 de Junio de 2013, de 2013 Autodesk, Inc.: http://www.autodesk.es/products/autodesk-maya/overview

Balerdi, F. E. (2009). *Gredos, Universidad de Salamanca.* Recuperado el 2 de Agosto de 2013, de http://gredos.usal.es/jspui/bitstream/10366/56438/1/TEE2001_V2_videojuegoseducacionpdf.pdf

Caccuri, V. (2013). *Educación con TICS.* USERSHOP.

Creighton, R. H. (2010). *Unity 3D Game Development by Example: A Seat-of-Your-Pants Manual for Building Fun, Groovy Little Games Quickly.* Packt Publishing Ltd.

Cruz Zeballos, M. A. (2012). Las APP en la Nube. *Revista de Información, Tecnología y Sociedad*(7), 53-56.

Exit Games. (2013). *Exit Games®*. Recuperado el 17 de Junio de 2013, de Exit Games®: http://cloud.exitgames.com/Unity

García Fernandez, F. (2005). Videojuegos: un análisis desde el punto de vista educativo.

García Fernández, F., & Bringué Sala, X. (2010). Educar hij@s Interactiv@s: Una reflexión práctica sobre las pantallas. *Foro Generaciones Interactivas*, 68.

Gargenta, M. (2011). *Learning Android.* O'Reilly Media, Inc.

González-Calero, P. A., & Gómez-Martín, M. A. (2011). *Artificial Intelligence for Computer Games.* Springer.

Greenfield, P. M. (1996). Video games as cultural artifacts. *P.M. Greenfield & R. R. Cocking (Eds)*, 35-46.

Inergis. (2012). *Inergis*. Recuperado el 18 de Febrero de 2014, de http://www.inergis.com/contenidos/general.action?idsupersection=2&idselectedsection=24&selectedsection=%BFPor%20qu%E9%20el%20Cloud%20Computing?&idparentmenu=146&idpage=44&idcomission=0&typetable=oferta_estudios

Inteligencia Computacional. (2013). *Inteligencia Computacional.* Recuperado el 15 de Agosto de 2013, de Inteligencia Computacional: http://www.smartcomputing.com.ar/capitulo6.aspx

JuegosMMO.net. (18 de Junio de 2013). *JuegosMMO.net*. Obtenido de JuegosMMO.net: http://juegosmmo.net/juegos-mmo/

Mcfarlane, A., Sparrowhawk, A., & Heald, Y. (2002). Report on the educational use of games.

Mendoza Barros, P., & Galvis Panqueva, A. (1998). JUEGOS MULTIPLAYER: JUEGOS COLABORATIVOS .

Mendoza Barros, P., & Galvis Panqueva, A. (1998). *JUEGOS MULTIPLAYER: JUEGOS COLABORATIVOS PARA LA EDUCACIÓN.*

Millington, I., & Funge, J. (2009). *Artificial Intelligence for Games.* CRC Press.

Orchilles, J. (2010). *Microsoft Windows 7 Administrator's Reference: Dive Deeper: Upgrading, Deploying, Managing, and Securing.* Syngress.

Padilla Zea, N., González Sánchez, J. L., Gutiérrez, F. L., Cabrera, M. J., & Paderewski, P. (2008). Diseño de Videojuegos Educativos Multijugador. Una Visión desde el Aprendizaje Colaborativo. España. Recuperado el 5 de Diciembre de 2013, de Diseño de videojuegos multijugador. Una visión desde el aprendizaje colaborativo: http://lsi.ugr.es/juegos/articulos/interaccion08-colaboracion.pdf

Pogue, D. (2011). *Mac OS X Lion: The Missing Manual.* O'Reilly Media, Inc.

Prensky, M. (2001). *Digital Natives, Digital Immigrants.*

Pressman, R. S. (2006). *Ingeniería del Software un enfoque práctico* (Sexta ed.). Madrid, España: McGraw Hill.

Rollings, A., & Morris, D. (2003). *Game Architecture and Design.*

Schwabe, D., & Rossi, G. (1998). Developing Hypermedia Applications using OOHDM.

Singer, R. N., & Williams, A. M. (1998). *An exploratory study in live tennis .*

Soto De Giorgis, R., Palma Muñoz, W., & Roncagliolo De La Horra, S. (2011). *Propuesta de un modelo navegacional para el desarrollo de aplicaciones basadas en OOHDM.* Recuperado el 30 de Junio de 2013, de http://tallerinf281.wikispaces.com/file/view/Aplicacion-OOHDM.pdf

Star UML. (2013). *Star UML*. Recuperado el 17 de Junio de 2013, de Star UML: http://staruml.sourceforge.net/en/about.php

Unity Technologies. (2013). *Unity Technologies*. Recuperado el 17 de Junio de 2013, de Sitio Web de Unity Technologies: http://spanish.unity3d.com/unity/

Van Eck, R. (2006). Digital Game-Based:It's Not Just the Digital Natives Who Are Restless. *Educause, 41*(2), 16-30.

Yacci, M., Haake, A., & Rozanski, E. (2004). *Operations and Strategy Learning in Edutainment Interfaces.*

Zagal, J. P., Rick, J., & Hsi, I. (Marzo de 2006). Collaborative games: Lessons learned from board games. *SIMULATION & GAMING, 37*(1), 24-40.