PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DESARROLLO DE LA MECÁNICA Y DINÁMICA DE UN VIDEOJUEGO SERIO 3D EN TERCERA PERSONA

Tesis para optar por el Título de Ingeniero Informático, que presenta el bachiller:

Luis Christian Fernández Martínez

ASESOR: Johan Baldeón

Lima, Diciembre del 2012

Resumen

Actualmente en nuestro país se manifiesta un resurgimiento de viejas ideologías violentistas que en un pasado dañaron gravemente a nuestra sociedad y dejaron terribles secuelas. Estas corrientes de pensamiento son impulsadas por movimientos pseudo-políticos que, obteniendo ventaja de la desinformación y de la carencia vivencial de la era del terror por parte de los grupos más jóvenes, preparan un relanzamiento de sus viejas propuestas. Por ello, se considera que, el uso de un videojuego serio en Internet para satisfacer la necesidad de informar a la población joven-adolescente sobre las consecuencias de estas ideologías, tendría un impacto positivo en este grupo social.

El producto propuesto es un videojuego en tres dimensiones que permite al jugador ejecutarlo desde cualquiera de los navegadores web más populares. La inteligencia artificial de los personajes no controlados por el jugador incluye el diseño de comportamiento y el trazado de ruta. Por otro lado, la mecánica permite al jugador interactuar con el entorno, los objetos y los personajes. La inmersión en el videojuego, el impacto emocional del mismo y el compromiso del jugador es facilitado por el entorno 3D. La difusión de la información es realizada a través de la historia de fondo del juego, la cual puede ser revisada como texto escrito. Sin embargo, la riqueza del juego proviene de la interacción con un escenario plagado de enemigos controlados por el computador, cuyas acciones violentas son observadas por el jugador el cual asume el rol de observador imponente ante los maltratos perpetrados por los mismos; y que, además, posee recursos de acción muy limitados, a veces casi nulos —entre ellos completar pequeñas misiones antes de completar la misión principal. Esta mecánica recrea en cierta manera los sentimientos de impotencia generados en una situación real. Cumpliendo así el objetivo principal de informar a través de un juego de dichas experiencias de violencia.

Gracias a Dios, a Kelly, mi amada esposa, a mis padres, a mi hermana y a todos aquellos amigos y hermanos que me apoyaron durante todo este tiempo.

Tabla de Contenido

Índice de Tablas

Índice de Figuras

Introducción

Actualmente en nuestro país se manifiesta un resurgimiento de viejas ideologías violentistas que en un pasado dañaron gravemente a nuestra sociedad y dejaron terribles secuelas. Estas corrientes de pensamiento son impulsadas por movimientos pseudo-políticos que, obteniendo ventaja de la carencia vivencial de la era del terror por parte de los grupos más jóvenes, preparan un relanzamiento de sus viejas propuestas.

A pesar de la existencia de amplia información bibliográfica —libros, artículos, documentales, etc. — y de la profunda memoria colectiva del periodo de violencia en las generaciones más viejas, los sectores más jóvenes de nuestra sociedad carecen de la información necesaria para identificar dichas ideologías como elementos repudiables para nuestro país. Esto se debe a múltiples factores que no forman parte del presente estudio, sin embargo la solución es tan compleja como simple al mismo tiempo: crear memoria colectiva en las generaciones recientes y futuras.

El presente trabajo no pretende dar una solución completa a esta problemática, pero sí aportar a la generación de memoria colectiva mediante la difusión de experiencias que permitan recrear lo negativo de estos sucesos, a través de una herramienta tecnológica como los videojuegos que es atractiva para el público al cual se desea informar.

A lo largo de los siguientes capítulos se hace énfasis en el proceso de producción de un videojuego, desde la selección del género, el número de dimensiones y el diseño de la jugabilidad, hasta la selección del motor, implementación de la mecánica, algoritmos de trazado de rutas e inteligencia artificial, integración con los gráficos y pruebas.

Para ello se delimita el problema al que se pretende dar solución, el cual escapa de la problemática social propiamente dicha y se concentra en el requerimiento nacido de la misma: la difusión de información a través de un medio atractivo como lo es un videojuego. Además, con una evaluación adecuada del estado del arte de otras soluciones se esboza una solución inicial con un alcance realista y concreto para el problema planteado.

Como todo proyecto formal, único y acotado en términos de tiempo, alcance y costos, es gestionado por una metodología que se detalla más adelante y que es basada en dos metodologías de gestión de proyectos reconocidas mundialmente.

Por otro lado, también se adapta una metodología de desarrollo de producto apropiada para el proceso de construcción de un videojuego. Dentro de este proceso, en la primera fase, se define la lista de requisitos de usuario que permiten dar una solución adecuada al problema que enfrenta este proyecto, entre otros, el género del videojuego, la cantidad de dimensiones (2D ó 3D), los personajes y la jugabilidad.

Se demuestra que la solución es técnicamente viable, económicamente factible y temporalmente realizable. Dicha solución planteada dentro del análisis es la línea base del producto el cual evoluciona con el desarrollo del proyecto debido a que, como la mayoría de documentos en la gestión de producto y proyectos, es un documento vivo.

Y para finalizar esta fase inicial, análoga al análisis y diseño, se define la tecnología que se usará —un motor de propósito general en este caso— sobre la cual se diseña el motor para este videojuego en particular. Además, se definen los algoritmos de ruta óptima e inteligencia artificial, física y diseño de acciones que el personaje realiza, entre otros.

En la segunda fase, equivalente a la implementación, se realiza la construcción del producto. Es aquí donde se elaboran los elementos estético-gráficos (los cuales quedan fuera del alcance de este proyecto) y la mecánica del juego que incluye el motor, los algoritmos, la inteligencia artificial, etc.; para finalmente integrarlos en cada versión ejecutable del juego liberado. También se presenta la estrategia de pruebas dentro de esta fase y junto a ella, los resultados obtenidos por cada iteración.

Finalmente, el último capítulo, incluye las observaciones y conclusiones desprendidas de este proyecto sobre el cual se realizan recomendaciones y el trabajo futuro realizable para ampliar el alcance del proyecto.

1 Generalidades

Cuando un trabajo de ingeniería pretende atacar un problema moral-social corre el riesgo de caer en la subjetividad. Para evitar aquello, en el presente capítulo, se da una definición objetiva de la problemática, conceptos, estado del arte y solución planteada junto a los objetivos del presente proyecto.

1.1 Definición del Problema

El Perú se ha convertido en una economía emergente (El Comercio 2012a; Gestión 2012), con un aumento constante del uso de tecnologías y donde el porcentaje de la población que hace uso de la Internet crece con cada año que pasa (véase la Tabla 1.1). Dentro de este escenario, más de la mitad del total de la población joven —de 12 a 18 y de 19 a 24 años— explota este recurso, en otras palabras, los mayores usuarios de la Internet en el Perú son los jóvenes y adolescentes.

Además de ello, otros datos estadísticos ofrecidos por el INEI en el informe técnico *Las Tecnologías de Información y Comunicación en los Hogares* (véase la Tabla 1.2) muestran que, del total de la población que hizo uso de Internet en 2011, el 91.6% lo utilizó para obtener información. Dicho de otro modo, la actividad principal

realizada por los internautas es la búsqueda de información; este fenómeno también se observa en todos los años anteriores analizados.

Año / Trimestre	Total	Sexo		Grupos de edad					
		Hombre	Mujer	6 a 11 años	12 a 18 años	19 a 24 años	25 a 40 años	41 a 59 años	60 y más
Indicadores Anuales									
2007	31,1	34,9	27,2	22,6	53,2	54,6	30,5	16,0	4,0
2008	31,7	35,5	27,7	23,6	54,3	55,1	31,5	16,3	4,9
2009	33,9	37,9	29,8	26,2	56,1	57,8	34,5	18,8	5,1
2010	34,8	38,9	30,5	26,4	59,0	58,7	35,2	19,3	6,0
2011	36,0	39,7	32,3	28,9	59,3	60,6	36,6	21,1	5,8

Tabla 1.1. Perú: Población de 6 años y más que hace uso de Internet por sexo y grupos de edad (INEI 2012: 22).

Año: 2007 – 2011, porcentaje del total de población de 6 y más años de edad de cada sexo y grupo de edad. Fuente: Instituto Nacional de Estadística e Informática (INEI), Las Tecnologías de Información y Comunicación en los Hogares, Trimestre: Abril-Mayo-Junio, 2012. (INEI 2012: 22)

Dentro de este contexto; donde el acceso a información es irrestricto y la cantidad de documentos electrónicos, artículos, documentales, informes —entre ellos el informe final de la Comisión de la Verdad y Reconciliación—, entre otros, es muy grande; se esperaría tener una población joven muy bien informada. Sin embargo, aparentemente esto no es así, sino que algunos jóvenes tienen poca o ninguna noción del pasado de terror que se vivió en el Perú durante los años 80 y principios de los 90. Dos encuestas periodísticas realizadas a dos grupos de jóvenes en Lima parecen evidenciarlo (El Comercio 2012b; 2012c).

Por otro lado, el uso de Internet para actividades de entretenimiento, entre ellas el consumo de videojuegos, se ha incrementado en los últimos años. Como se aprecia en la Tabla 1.2, en 2007 el 35.8% del total de usuarios de Internet hacía uso del mismo para realizar actividades de entretenimiento como jugar videojuegos, entre otras; sin embargo, en 2011 este valor ascendió hasta 66.2%. Este incremento es considerable, pues de 2007 a 2011 casi se ha duplicado y muestra una tendencia a seguir creciendo. Estas cifras muestran que el uso de medios considerados para el entretenimiento, como canales de difusión de información, es una excelente alternativa a los medios tradicionales.

Año / Trimestre	Comunicarse (e-mail, chat, etc)	Obtener información	Actividades de Entretenimiento (juego de video, obtener películas, música, etc).	Educación formal y actividades de capacitación	Operaciones en banca electrónica y otros servicios financieros	Transacciones (Interactuar) con organizaciones estatales, autoridades públicas.
			Indicadores anuales			
2007	74,4	79,1	35,8	7,4	3,8	2,4
2008	74,8	84,6	45,4	9,0	4,5	3,5
2009	75,7	88,5	60,2	13,7	6,7	5,8
2010	75,4	91,3	63,9	10,5	5,7	6,5
2011	75,9	91,6	66,2	8,8	5,6	6,6

Tabla 1.2. Perú: Población de 6 años y más que usa Internet por tipo de actividad que realiza.

Año: 2007 - 2011 (porcentaje sobre el total de usuarios de Internet). Fuente: Instituto Nacional de Estadística e Informática (INEI), Las Tecnologías de Información y Comunicación en los Hogares, Trimestre: Abril-Mayo-Junio, 2012. (INEI 2012: 32)

Además de ello, un estudio sobre el uso de servicios a través de Internet realizado por Xavier Bringué y Charo Sádaba muestra que, de los medios considerados para el entretenimiento en Perú, los juegos en red son utilizados por el 56% de los internautas (2008: 48). Asimismo, en dicho estudio se encuentra que el consumo de contenidos relacionados con el ocio por parte de los jóvenes se contrapone con fuerza a la posibilidad de buscar en Internet fuentes de información y conocimientos educativos o culturales –sólo dos de cada diez menores de edad visitan habitualmente recursos educativos o culturales los cuales normalmente están asociados al desarrollo de tareas escolares– (2008: 52). Finalmente, de los contenidos más visitados en Perú, los juegos ocupan el segundo lugar con un 59% frente al 37% de los contenidos educativos y al 31% de los culturales (2008: 52).

Por ello, se considera que, el uso de un videojuego serio en Internet como medio de difusión de información, tendría un gran impacto en el estrato social joven-adolescente.

Este producto debe alcanzar un impacto visual con cierto nivel de realismo de modo que, con un buen trabajo realizado en este tópico, el videojuego logre sentirse sólido, real y atractivo para el jugador quien tomará el juego con más seriedad por considerarlo con mayor valor intrínseco (Schell 2008: 347). La experiencia vivencial debe ser la suficiente para que el usuario pueda adentrarse en la historia y seguirla de principio a fin. Un escenario 3D lo suficientemente interactivo e inmersivo

permitirá el incremento del componente emocional y del compromiso del jugador en el juego de video (Jeong 2008: 191).

Por todo lo anterior se propone el desarrollo de un videojuego en 3D, que funcione sobre una plataforma web, como herramienta en la difusión de experiencias que muestren el accionar de algunos grupos extremistas.

En este punto, la problemática se enfoca en el desarrollo de un videojuego, el cual compromete múltiples áreas del conocimiento y requiere de un equipo multidisciplinar para obtener un producto completo. Entre estos diferentes enfoques están: diseño de videojuegos —*game design*—, programación, creación artística y gestión de proyectos (Rabin 2010: xiv).

El diseño del videojuego es el punto de partida del desarrollo, es donde se define la experiencia que se ofrece a través de los diferentes componentes que forman parte del mismo (Rabin 2010: 62). Dichos componentes son, en esencia, el contenido —*gameplay*, misiones, niveles, historia y guion— y las reglas —mecánica— del videojuego, elementos que deben mantener al jugador motivado para lograr los objetivos del juego (Brathwaite 2009: 2; Bethke 2003: 39).

A pesar de que el diseño es tan crítico, y debido a que el tema es multidisciplinar, queda fuera del alcance del presente trabajo monográfico el análisis del caso sobre el cual se basa la historia, el desarrollo del guion que se aplica a los diálogos y el modelado del escenario y el de los personajes —elementos diversos que exponen la dinámica que se vivía por aquellos días. Sin bien es cierto, dichos elementos son utilizados y mencionados en los documentos pertinentes, en este proyecto el diseño se centra en el desarrollo de la mecánica —código y reglas del juego— y dinámica —aplicación de las reglas al contexto— del videojuego; según el marco de trabajo *MDA*: mecánica, dinámica y estética (Hunicke 2004: 2). Donde el último elemento —la estética—, según dicho marco, define las emociones que se desea evocar en el jugador a través de la experiencia que ofrece el juego; por tanto, su definición tampoco está dentro del alcance.

Por otro lado, el arte en un juego es parte fundamental del mismo. Un buen componente visual puede volver a captar la atención del jugador a un videojuego que podría haber pasado por alto (Schell 2008: 347). Las diferentes áreas creativas que componen el arte de un juego de video son: arte conceptual, diseño de

interfaces, modelado 3D, modelado de personajes, texturizado, animación y elaboración de guiones gráficos o *storyboards* (Bethke 2003: 45-49). Pese a ello, el enfoque de este trabajo no contempla el desarrollo del mismo y el arte está integrado como elemento externo al producto final.

Dentro de la creación artística se encuentran también los componentes de audio, de los cuales los más importantes son: voz en *off*, efectos de sonido y música (Bethke 2003: 49-50). Es cierto que, para la generación de los efectos de sonido, interviene la ingeniería de sonido; sin embargo este trabajo tampoco da cobertura a la producción de estos componentes los cuales están integrados al producto final como elementos externos.

Finalmente, la programación del videojuego conforma la problemática central de este proyecto. Sin embargo, este componente representa el trabajo más complejo del desarrollo de un videojuego —dicha complejidad puede llegar a tal nivel que, en algunas ocasiones, el producto puede retrasarse, estar por encima del presupuesto o tener muchos defectos—, por ello debe desagregarse con el fin de definir cuáles subcomponentes se elaboran y cuáles no (Bethke 2003: 41).

Por ello, se exponen los diferentes subcomponentes que conforman la programación del videojuego con el objetivo de acotar el problema adecuadamente (Bethke 2003: 43-45; Rabin 2010:167- 169,421-639):

i. La mecánica del videojuego, consiste en la codificación de la reglas definidas en el diseño, es aquí donde se elabora la física del juego, el funcionamiento de las armas y las acciones de los personajes, entre otras reglas particulares al juego;
ii. el motor de gráficos 3D, con un alto nivel de complejidad matemática, tiene como objetivo renderizar un escenario tridimensional en los píxeles de una pantalla plana;
iii. sistema de animación de personajes, que en su sentido más plano se refiere a la implementación de un árbol de huesos que, con transformaciones espaciales, permite dar movimiento a diferentes partes de un personaje de forma independiente;
iv. inteligencia artificial, la cual tiene dos grandes subcomponentes:
    a. la emulación de la inteligencia en los oponentes con la finalidad de ofrecer un adecuado nivel de desafío al jugador y al mismo tiempo consumir pocos recursos;

b. trazado de rutas o *pathfinding* cuyo algoritmo más ampliamente usado en la industria de los videojuegos es el A* —pronunciado A estrella.

v. interfaz de usuario, elabora cada una de las pantallas que posee el videojuego, menús de navegación, *HUD* —del inglés *"head-up display"* que muestra información del jugador como salud, inventario, progreso, etc. —;

vi. programación de audio, codifica los efectos de sonido 3D, voz en *off* y música de fondo, además ofrece herramientas para el control del mismo como paneo, volumen, control de tono por cada canal individual de sonido;

vii. y finalmente, redes y arquitectura multijugador, implementa el código a bajo nivel que permite temporizar eventos —basado en turnos o tiempo real—, uso de dispositivos de E/S compartidos y conectividad de red.

Como se puede apreciar en el párrafo anterior, la programación de un videojuego involucra muchas especialidades dentro de la misma; por tanto no es posible, hoy en día, que un solo programador elabore un producto como este sin apoyarse de alguna herramienta, librería o motor del algún tipo (Rabin 2010: 167). Por ello, la problemática del producto se centra en la implementación de la mecánica, la inteligencia artificial y la elaboración de la interfaz de usuario; para los demás componentes se hace uso de alguna librería o motor de videojuego.

Dentro de esta problemática, se tiene la mecánica del videojuego, esta depende del diseño del mismo y, según el contexto, posee líneas generales que se ajustan al propósito del juego. Una de ellas es no permitir al jugador asumir una postura agresiva sino que sea testigo del terror que este grupo subversivo infundía a sus víctimas. Otra, es que el sector más golpeado por este grupo fue el rural-campesino (CVR 2003a: 316). Finalmente, que el grupo de enemigos actuaba brutalmente y hacía uso de distintos tipos de armas: de guerra —FAL, HK, etc. —, menores —pistolas, carabinas, "quesos rusos", etc. — y blancas —lanzas, machetes, etc. — (CVR 2003b: 131). Por tanto se puede esbozar como reglas básicas del juego que deben ser implementadas:

i. mecánica de acciones del tipo sigilo para el personaje principal;

ii. un escenario rural con una física realista;

iii. enemigos que hagan uso de al menos tres tipos de armas y que cuenten con inteligencia artificial.

Para poder conseguir un nivel adecuado de desafío por parte de los enemigos, se tiene que elaborar una cuidadosa inteligencia artificial para estos, esta debe contar de dos componentes principales:

i. la IA que permite que estos reaccionen ante le presencia del jugador principal ofreciendo la posibilidad de ser evadidos y que, al mismo tiempo, no resulte muy fácil;
ii. y el algoritmo de trazado de ruta que permite a los enemigos desplazarse por el terreno 3D del escenario.

Finalmente la interfaz de usuario que debe seguir algunos estándares definidos por la industria en videojuegos similares con la finalidad que el jugador se encuentre familiarizado con los mismos, si es que tiene experiencia con videojuegos de este tipo; o que pueda adaptarse con cierta facilidad a la *HUD*.

El resto del documento analiza cómo elaborar la mecánica, la *IA* y el *HUD* del videojuego. Además de ello identifica qué librería o motor de videojuego se usa.

1.2 Marco Conceptual

El presente documento utiliza una serie de conceptos clave cuya comprensión debe darse por anticipado. Para tal fin se sintetizan dichos conceptos y se detallan a continuación:

1.2.1 Videojuego

Uno de los primeros estudios formales respecto a los videojuegos fue realizado en 1984 por Patricia Greenfield, lamentablemente fue uno de los pocos que se realizaron por esa década y la siguiente. Y es por eso que, aún hoy, muchas de las hipótesis trazadas en su investigación siguen sin ser estudiadas (Greenfield 2010: 1-2). Por ello, vemos que la mayoría de investigaciones científicas en el campo de los videojuegos tiene poco más de diez años de antigüedad (Aarseth 2001).

Sin embargo, los videojuegos nacen de los juegos que, antes de la aparición de los computadores, eran utilizados para fines lúdicos. Por ello, antes de definir a los videojuegos es necesario tener claro el concepto de juego: “Un juego es una actividad interactiva voluntaria, donde uno o más jugadores siguen determinadas reglas que definen su comportamiento, promulgando un conflicto artificial cuyo desenlace es cuantificable” (Esposito 2005: 2). Sin embargo, este concepto deja de

lado a aquellos que no poseen reglas de juego, esto es, a los juegos que dan al jugador total libertad para realizar acciones; por ejemplo, los juegos con juguetes o los rompecabezas.

Por esa razón, Wolfgang Kramer propone que los juegos se componen de dos elementos: componentes y reglas. Define a los componentes como el *hardware* y a las reglas como el software —sin dejar de considerar que hablamos de juegos como concepto general—, y plantea una serie de criterios para diferenciar a los juegos con reglas de aquellos que no los tienen. Sus criterios son: (i) todo juego debe ofrecer experiencia de juego en grupo o en solitario, (ii) igualdad de posibilidades de ganar, (iii) libertad de tomar decisiones, (iv) participación activa y, finalmente, (v) la capacidad de sumergirse en el mundo del juego. Mientras que, los juegos con reglas tienen algunas características adicionales: (i) reglas de juego, (ii) objetivo, (iii) curso del juego variable (sujeto al azar) y (iv) la competencia entre jugadores (2000).

Una vez definido el concepto de juego se puede dar una posible definición sencilla para videojuego: **"**Un videojuego es un juego que con el que interactuamos, gracias a un aparato audiovisual y que podría estar basada en una historia**"** (Esposito 2005: 2).

En otras palabras, el estudio de los videojuegos se ha convertido en una nueva rama dentro del estudio de los juegos. En este último, existen diferentes modelos teóricos para definir a los videojuegos, pero son tres de ellos los que lo describen mejor: el enfoque narratológico —que los define como narraciones o historias interactivas—, ludológico —que enfatiza la naturaleza lúdica de los mismos—, y de ficción interactiva —que, como su nombre lo dice, define a los videojuegos como ficción con la que se puede interactuar— (Tavinor 2008).

Finamente, Tavinor sintetiza estas definiciones para construir, lo que él llama, la definición disyuntiva de los videojuegos: "X es un videojuego si y sólo si (a) se trata de un artefacto en un medio visual digital, (b) está concebido principalmente como un objeto de entretenimiento, y (c) está destinado a proporcionar dicho entretenimiento a través del empleo de una o ambas de las siguientes modalidades de participación: sujeta a reglas de juego o ficción interactiva" (2008).

1.2.2 Videojuego serio

A principios de los 90's el término *Edutaiment* —educación a través del entretenimiento— se hizo popular con la expansión del mercado de las computadoras a pesar de no estar restringido a ellas. Los primeros videojuegos educativos emergieron por esta época cuyo público objetivo era principalmente niños pequeños y de prescolar, con enfoque en matemáticas, letras y ciencias. Lamentablemente estos videojuegos fracasaron por no ofrecer, además del conocimiento, entretenimiento para los infantes. (Susi 2007: 2)

La industria de los videojuegos para el entretenimiento fue creciendo, mientras que los videojuegos educativos no tuvieron acogida; además, la tecnología evolucionó hasta que, a fines de los 90's el concepto de juego serio fue revaluado. A partir de *America's Army*, uno de los más exitosos videojuegos educativos, se marca el punto de inicio del movimiento de los videojuegos serios. (Susi 2007: 2).

Dicho término no es nuevo, fue acuñando por primera vez por Clark Abt en la década de los 60, haciendo referencia a los juegos de mesa que se usaban en las aulas de clase para recrear las estrategias de guerra usadas durante la Primera Guerra Mundial (Marcano 2008: 97).

Contrariamente a lo que el término pudiera suponer, los juegos serios generan las mismas emociones que sus pares comerciales aunque este objetivo a veces se considera secundario—. Sin embargo, se utiliza el término "serio" con la finalidad de complementar al concepto de juego los conceptos de responsabilidad, realidad y reflexión sobre las consecuencias de las acciones que se toman en el mismo (Marcano 2008: 97). Esta dualidad, formada por la diversión y reflexión, distingue a los videojuegos serios de los comerciales.

Además de ello, los videojuegos serios buscan desarrollar en los jugadores nuevos conocimientos y capacidades; con aplicaciones distintas como entrenamiento, publicidad, simulación, educación e información (Susi 2007: 3). Los videojuegos como mecanismos de reclutamiento y entrenamiento han logrado sus objetivos y aún se siguen utilizando —*America's Army* es el mejor ejemplo de esto—.

Sin embargo, a pesar de los aspectos positivos de los videojuegos serios, luego de ser utilizados en las aulas de estudio, se desató una controversia. Esto último sucedió debido a que, existe cierto escepticismo respecto al uso de videojuegos en

la educación por parte de los investigadores más conservadores, quienes creen que estos no son compatibles con el aprendizaje institucionalizado pues, según argumentan, los juegos no enseñan temas específicos. Por el contrario, investigadores como Kurt Squire, aseguran que el aprendizaje basado en videojuegos revolucionará las instituciones educativas; otros como Steven Johnson, han argumentado que los juegos permiten que los niños puedan resolver problemas de manera más rápida y creativa (Brown 2008: 118-119). En síntesis, es difícil negar que los videojuegos atraen la atención de los jóvenes y son una buena herramienta pedagógica. Finalmente, los videojuegos de tipo multijugador, promueven el aprendizaje a través de la competencia, recompensa, cooperación y el trabajo en equipo; estos permiten que el alumno adquiera un papel protagónico en su propio desarrollo intelectual y convierten al instructor en un facilitador para el aprendizaje (Brown 2008: 120).

Dentro del concepto de videojuego serio, existe un conjunto de características propias de que los diferencia de sus pares comerciales. Estos son: la resolución de problemas, el aprendizaje, la simulación y la comunicación los cuales se pueden apreciar con mayor claridad en la Tabla 1.3. (Susi 2007: 6).

	Juegos serios	Juegos de entretenimiento
Ejecución de tareas vs. Rica experiencia	Énfasis en la resolución de problemas	Inclinación por una buena experiencia de juego
Enfoque	Elementos importantes para el aprendizaje	Diversión
Simulaciones	Supuestos necesarios para simulaciones realistas	Proceso simplificado de simulación
Comunicación	Debe emular la comunicación natural (no perfecta)	Ofrece a menudo una comunicación perfecta

Tabla 1.3. Diferencias entre los juegos serios y los juegos para el entretenimiento (Susi 2007: 6).

Finalmente, la categorización de los juegos serios más ampliamente aceptada, según su área de aplicación, es la siguiente (Susi 2007: 10):

- Juegos militares; son utilizados para el entrenamiento militar y los videojuegos de este tipo incrementan considerablemente habilidades como la coordinación ojo-mano, trabajo en equipo o la capacidad de realizar múltiples tareas a la vez.

- Juegos de gobierno; simulan diferentes situaciones ante las cuales el jugador debe responder utilizando políticas de gobierno que tienen repercusiones en el mundo virtual que gobierna.
- Juegos educativos; son una subcategoría de los videojuegos serios cuyo fin fundamental es el aprendizaje (Becker 2010: 23).
- Juegos corporativos; utilizados para el entrenamiento asistido dentro de grandes empresas.
- Juegos para la salud; son aquellos cuya aplicación está relacionada con el cuidado de la salud: ejercicio físico, entrenamiento médico, terapias de distracción, rehabilitación, etc.
- Adicionalmente existen otras categorías no contempladas en esta lista como los *newsgames* o los social *change games*.

1.2.3 Marco de trabajo MDA

El marco de trabajo *MDA* —por sus siglas en inglés— es uno de los modelos para videojuegos más conocidos (Rabin 2010: 67) que los divide en tres componentes fundamentales: mecánica, dinámica y estética. Estos componentes se pueden observar desde dos diferentes perspectivas (véase Figura 1.1): la del diseñador y la del jugador (Hunicke 2004: 2).

Estos componentes son:

- **Mecánica:** describe los componentes particulares del juego en cuestión tales como acciones, comportamientos y mecanismos de control en el contexto del juego, pero en el nivel de representación de datos y algoritmos (Hunicke 2004: 2, 4-5).
- **Dinámica:** describe el comportamiento en tiempo de ejecución de la mecánica actuando sobre los comandos y acciones del jugador —entradas— y las respuestas del sistema —salidas— a través del tiempo con la finalidad de crear la experiencia estética del mismo (Hunicke 2004: 2, 4).
- **Estética:** describe las respuestas emocionales deseables evocadas en el jugador, cuando interactúa con el sistema de juego (Hunicke 2004: 2, 4).

Desde la perspectiva del diseñador, la mecánica da lugar al comportamiento dinámico del sistema, que a su vez conduce a determinadas experiencias estéticas.

Desde la perspectiva del jugador, la estética establece el tono del juego, que nace en la dinámica observable y, finaliza en la mecánica operable (Hunicke 2004: 2).

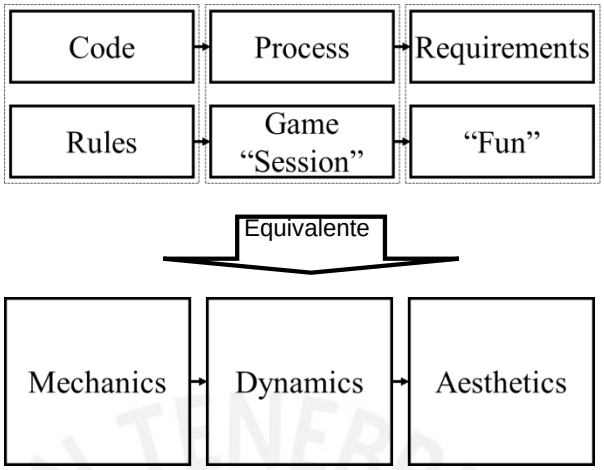


Figura 1.1. Los videojuegos vistos como software (Leblanc 2004)

1.2.4 Diseño de videojuegos (*game design*)

Como se mencionó anteriormente, el diseño del videojuego es la fase inicial del proceso de pre-producción donde se desarrollan los elementos esbozados en el concepto del videojuego los cuales crean la experiencia para el jugador (Rabin 2010: 62; Keith 2012: 131). Dichos elementos son: el contenido —*gameplay*, misiones, niveles, historia y guion— y las reglas —mecánica— del videojuego (Brathwaite 2009: 2, 5; Bethke 2003: 39).

Dentro del diseño de un videojuego, un elemento fundamental es el *gameplay*, el cual es el *núcleo* o *centro dinámico* de un juego. En síntesis, el diseñador desea transmitir la experiencia del juego. Este núcleo está basado en una *mecánica central* como puede ser el atacar a tus enemigos soplando sobre ellos, girar las fichas por turno, vender unidades a otro jugador, entre otros. Esta *mecánica central* conduce a una *dinámica central* que es un patrón particular de juego —conocida también como la visión del juego— la cual debe ser lo suficientemente sintética como para poder describirse en una sola frase y a lo más contener dos oraciones (Brathwaite 2009: 6).

La mayoría de dinámicas centrales o *gameplay* ya fueron desarrolladas y aparecen en los videojuegos constantemente en sus formas puras o fusionadas, entre ellas tenemos: control territorial *Risk* o *Carcassonne*—, predicción —La Ruleta o Piedra-Papel-Tijeras—, razonamiento espacial —Tres en raya, *Pente* y *Connect Four*—,

destrucción —del tipo jugador contra jugador o *FPS*—, construcción —*SimCity* o Series *Caesar*—, colección —que hace uso de patrones para asociar objetos—, persecución o evasión —*Pac-Man*—, intercambio o negociación —*Pokemon* o *Animal Crossing*—, carreras —gana el primero en llegar a la meta o conseguir un logro particular— y supervivencia —con una constante lucha de vida o muerte como principal actividad en el videojuego— (Brathwaite 2009: 6-8).

Luego de ello, se establece el género del videojuego, el cual, según Tom Apperley, parte de los tipos de interacción disponibles en el juego, dicho en otras palabras, de la dinámica central o *gameplay* (2006: 9-10). Algunos de los géneros más conocidos son: aventura, acción, plataforma, luchas, disparos en primera persona —*FPS*—, estrategia en tiempo real —*RTS*—, estrategia basada en turnos, juegos de rol —*RPG*—, juegos de rol multijugador masivos en línea —*MMORPG*—, simulación, carreras, deportes, rítmicos, educativos, serios, de sigilo –similares a los *FPS* o de disparos, pero metódicos, con enfoque en subterfugios y escapes planificados— y de supervivencia al horror —subgénero de los *FPS*, hace uso de parajes desolados y monstruos acechando en la oscuridad— (Rabin 2010: 36-40).

Finalmente se elabora el estilo visual del videojuego —2D, 3D, isométrico—, acciones que realiza el jugador, diagrama de controles, diseño de la interfaz de usuario, diagrama de flujo del menú, historia de fondo, personalidad de los protagonistas, diseño del nivel y de la misión, descripción de la cinemáticas, efectos de sonido, y música (Bethke 2003: 101-127).

1.2.5 Sistema de componentes de objetos de juego

Un objeto de juego o *game object* es la entidad básica que existe en el mundo virtual del juego, está compuesto de un elemento de transformación que define su ubicación y orientación, además de información de su estado o valor de sus propiedades, y métodos para obtener y/o cambiar su estado (Stoy 2006: 393). Por otro lado, también pueden ser piezas lógicas de contenido interactivo que realicen tareas como renderizado, *path finding*, animación, persistencia, etc., por consecuencia no todas son visibles para el jugador (Bilas 2002: 5). Como algunos ejemplos se tiene: árboles, rocas, monstruos, enemigos, puertas, artículos de inventario, personajes principales, etc. De igual manera se puede tener: *triggers*, secuencias de cámaras, motores de ascensores, etc., los cuales no tienen una representación visual para el jugador.

El concepto de *game object* es el mismo que se usa en la programación orientada a objetos, por tanto la forma natural de esquematizar este sistema sería usando el modelo de herencia jerárquico tradicional (véase Figura 1.2), sin embargo esta no es una buena práctica. Esto se debe a que a medida que el sistema crece, la cantidad de información de estados y métodos para acceder a ellos se incrementa hasta llegar a ser inmanejable produciendo principalmente dos tipos de problemas: (i) objetos con funcionalidad duplicada si esta se implementa al nivel de las hojas del árbol jerárquico o (ii) objetos con sobrecarga de funcionalidades innecesarias si esta se implementa a un nivel cercano a la raíz del árbol —el anti-patrón *the blob*— (West 2007). Además, si se considera que, durante el desarrollo, los videojuegos están sometidos a cambios constantes los cuales obligan a una refactorización continua, esto agrava el problema, por tanto ningún tipo de esquema de herencia de objetos tradicional va a ser adecuado para videojuegos de mediano o gran tamaño (Bilas 2002: 8).

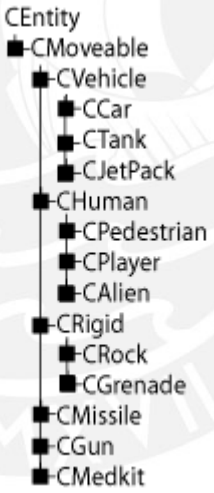


Figura 1.2. Estructura jerárquica de entidades de juego o game entity (West 2007).

El problema del modelo de herencia tradicional en videojuegos no es nuevo, por ello hace varios años atrás se propuso usar el *Sistema de componentes de objetos de juego —GOCS* por sus siglas en inglés— (Stoy 2006: 394). En este sistema cada objeto de juego es formado a partir de la integración de un conjunto de componentes especializados e independientes, como se puede apreciar en la Figura 1.3. En este sistema cada objeto posee únicamente la funcionalidad que

necesita y estas pueden ser removidas o se pueden añadir nuevas si se necesita (West 2007).

La implementación de este sistema se realiza a través de interfaces, cada componente hereda de un componente base y se generan familias de componentes que heredan de dicha base, el componente base publica un método virtual de actualización: *update*, que se ejecuta por cada iteración o *frame* del juego y cada familia de componentes publica métodos adicionales de acuerdo a su especialización. Por ejemplo, se puede definir una familia de componentes para el renderizado de objetos los cuales, además del método *update* tienen un método *render* (Stoy 2006: 394).

La gestión de los componentes se realiza a través de gestor global y varios contenedores que se utilizan cuando un *game object* posee más de un componente de la misma familia (Stoy 2006: 397). Finalmente, cada componente le pertenece a un único *game object* propietario el cual se convierte en una agregación de múltiples componentes de diferentes familias.

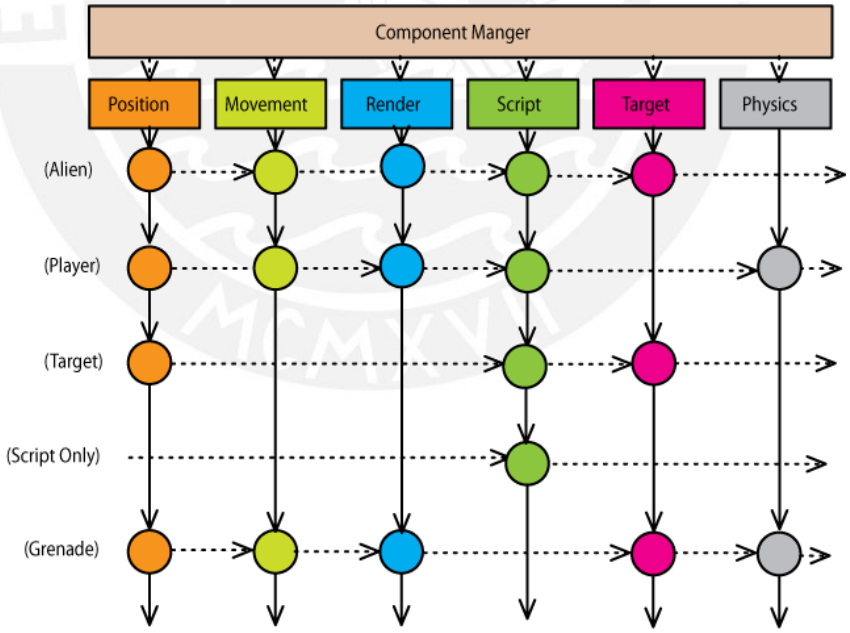


Figura 1.3. Composición de objetos de juego usando componentes (West 2007).

1.2.6 Algoritmo de trazado de ruta óptima

El trazado de la ruta óptima en una red de nodos es uno de los problemas de la Inteligencia Artificial más estudiados y más frustrantes en la industria de los videojuegos (Cui 2011:125). El algoritmo de Edsger Wybe Dijkstra, diseñado en 1959, fue uno de los primeros en dar una solución a este problema que, además, tuvo una gran repercusión en el mundo científico y aplicación en diversos campos debido a su eficiencia —de complejidad $O(n^2)$ donde n es el número de nodos utilizados para encontrar el camino óptimo— (Sánchez 2006: 2). Donde el principio de dicho algoritmo, el cual se aprecia en la Figura 1.4, se basa en que, si se tienen tres nodos: u, v y w; y el camino más corto entre u y v pasa por w, entonces el camino elegido entre v y w es el más corto entre ambos. (Sánchez 2006: 2).

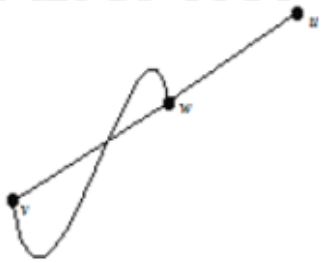


Figura 1.4. Algoritmo de Dijkstra (Sánchez 2006: 2)

Sobre este principio, el algoritmo de Dijkstra se puede implementar para hallar la ruta más óptima entre dos nodos de un arreglo bidimensional como representación computacional de un mapa. Cada nodo se puede representar como una celda de la matriz y la distancia o coste entre dos nodos adyacentes es siempre la misma. En este escenario el algoritmo trabaja visitando, a partir del nodo inicial, cada uno de los vértices no examinados alrededor de este y añadiendo un costo a la ruta desde el inicio hasta dicho nodo, y se expande hasta examinar el nodo de destino. Esto garantiza que, siempre que exista, el camino con menos coste será hallado (Patel 2009).

Sin embargo, el algoritmo de Dijkstra analiza múltiples nodos innecesariamente y genera una sobrecarga de procesamiento y consumo de memoria. Una alternativa a dicho algoritmo que hace un mejor uso de los recursos es el algoritmo Greedy Best-First-Search, el cual utiliza una heurística para estimar la ruta más corta examinando solamente los nodos con el valor de la heurística más baja hasta llegar al objetivo; con la salvedad de que no garantiza que esta ruta vaya a ser hallada (Coles 2007: 124-125).

Por otro lado, los mapas en los videojuegos no son tan simples como una matriz bidimensional totalmente transitable; los terrenos poseen obstáculos y el algoritmo de búsqueda debe poder evitarlos eficientemente. En este escenario el algoritmo más apropiado es el Dijkstra, sin embargo el trabajo que realiza es muy grande y por ende la carga para el computador (Patel 2009). Por ello se propuso el algoritmo A*, el cual utiliza lo mejor de ambos algoritmos vistos, realizando un análisis de todos los nodos cuyo costo total —representado por el costo exacto desde el punto inicial hacia cualquier punto n, más una heurística: $f(n) = g(n) + h(n)$— sea menor (Cui 2011: 126).

Además de esto, el algoritmo A* es probablemente el más usado en la IA para videojuegos. Este algoritmo también garantiza que, siempre que exista, el camino con coste más bajo será hallado; además es óptimo siempre y cuando la heurística *h(n)* sea admisible, en otras palabras, menor o igual al costo real. Por otro lado, no existe otro método de búsqueda de la ruta óptima, que utilice la misma heurística, que examine menos nodos que A* (Cui 2011: 126). La Figura 1.5 muestra el comportamiento de los algoritmos Dijkstra, Greedy BFS y A*; primera, segunda y tercera columna respectivamente; cuando no existen obstáculos y cuando los hay. Los nodos pintados indican que fueron evaluados por el algoritmo (Patel 2009).

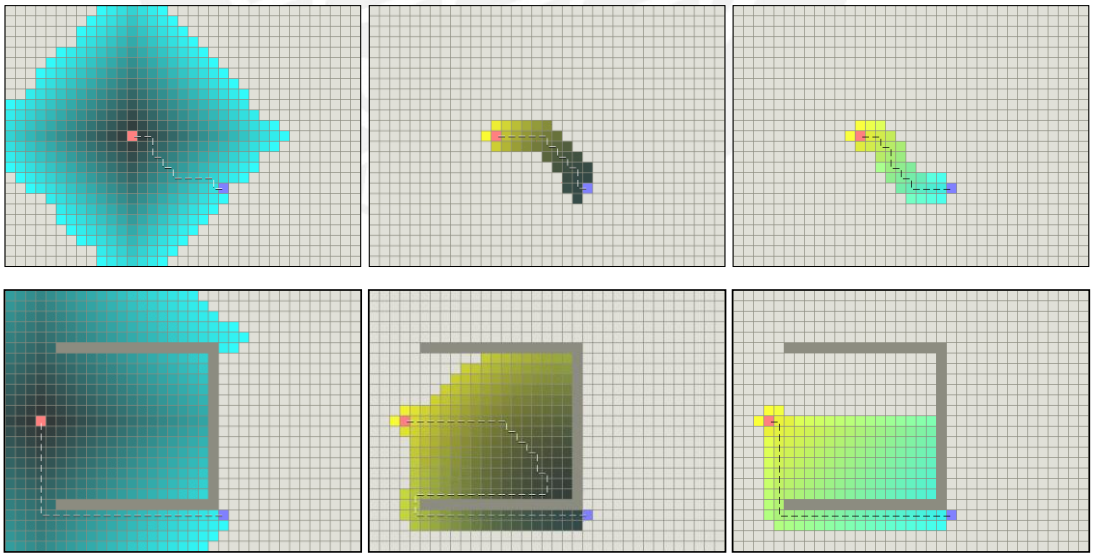


Figura 1.5. Comportamiento de los algoritmos Dijkstra, Greedy BFS y A* (Patel 2009).

El algoritmo A* ha sido optimizado de múltiples formas para su uso en videojuegos, entre las optimizaciones más populares se tiene (Cui 2011: 126):

- Espacio de búsqueda; consiste en reducir la carga de procesamiento para el algoritmo optimizando la representación del mapa.
    - Búsqueda de ruta jerárquica A* (HPA*); utilizada para mapas muy extensos, agrupa grandes áreas del mapa en vecindarios y luego se realiza una búsqueda de alto nivel, al llegar al vecindario objetivo se traza una ruta al punto al que se desea llegar.
    - Mallas de navegación (NavMesh); ampliamente utilizada para mapas en tres dimensiones donde se agrupa un conjunto de polígonos convexos que indican las superficies que no contienen obstáculos. El algoritmo traza una ruta a nivel de polígonos hasta que el personaje se encuentra en el mismo polígono que el objetivo, en este paso sólo sigue una línea recta hasta alcanzarlo.
- Función heurística; es la que la determina la eficiencia de A*, si el valor de esta heurística fuera 0 se tornaría en un simple algoritmo Dijkstra. Si la heurística sobrestima ligeramente el costo de la ruta la velocidad del algoritmo se incrementa y el resultado es una ruta razonable.
- Uso de memoria; una de las implementaciones más usadas es A* de profundidad iterativa o IDA*, el cual consiste en definir un costo máximo para la ruta si se alcanza este umbral y no se ha encontrado una solución, otra búsqueda es iniciada con el costo máximo incrementado en uno.
- Estructura de datos; consiste en utilizar tablas hash para evaluar si un nodo se ubica en la lista de nodos abiertos o la lista de nodos cerrados inmediatamente. Para la lista de nodos abiertos conviene utilizar una cola de prioridades.

1.2.7 Inteligencia artificial para personajes controlados por computador

Los personajes controlados por computador o NPC —por sus siglas en inglés: *non player character*— le dan a los videojuegos de un solo jugador mayor profundidad y jugabilidad. Además de esto, existe un gran número de jugadores que no entra a rondas en línea con otros jugadores sino que juegan de modo unipersonal. Por ello, con la finalidad de ofrecer una mejor experiencia de juego la IA para videojuegos debe construir competidores cuyo comportamiento sea similar al humano (Scott 2002: 16).

Dado que los *NPCs* deben mostrar rasgos de comportamiento humano, se debe definir cuáles son esos rasgos. Bob Scott, en su artículo *The Illusion of Intelligence*, ofrece un conjunto de patrones propios de un jugador humano (2002: 16-18):

- Predecibilidad e impredecibilidad; el jugador humano suele mostrar comportamientos impredecibles con la finalidad de distraer a su oponente como, por ejemplo, atacar al oponente más fuerte en un *FPS* o dar un pase largo desde media cancha en un juego soccer. Contrariamente y muchas veces en la misma ronda de juego muestra también comportamientos predecibles como utilizar la misma ruta en su mapa preferido en un *FPS* o tener una secuencia de comandos casi inalienable en un *RTS*. Esta dualidad es propia del ser humano y es muy difícil de imitar dentro de la IA para videojuegos. Se debe lograr una aleatoriedad suficiente como para que el jugador no pierda interés al repetir una partida y una predictibilidad suficiente como para que el jugador pueda descifrar la estrategia usada y pueda contrarrestarla.
- Cooperación; es otro de los rasgos característicos del jugador humano, este siempre tiende a formar alianzas con otros jugadores con la finalidad de protegerse mutuamente, cubriéndose del enemigo en caso de un *FPS* o protegiendo la ciudad del otro como si fuera su ciudad en caso de un RTS.
- Factor sorpresa; es en primera instancia el elemento clave que va a conseguir que el trabajo realizado en la IA sea admirable. En un *RTS* se puede utilizar ataques de doble envolvimiento o pinza, hostigamientos, emboscadas, etc.; en un *FPS* se puede utilizar recursos como las emboscadas, fuego de supresión, maniobras de flaqueo, etc. Una clase de sorpresas que agregan rasgos humanos a modo de elemento cómico secundario es la “estupidez creíble”, la cual permite que los personajes tomen decisiones incongruentes que suelen ser contraproducentes pero que aportan de modo indirecto a la credibilidad de la AI; dicho comportamiento es muy difícil de lograr y debe ser mínimo.

Por otro lado, la IA debe estar diseñada en función al jugador la cual le debe permitir obtener victorias y derrotas de modo equilibrado. Desarrollar este tipo de IA no es tan sencillo como desarrollar lo opuesto que es dejar que el jugador siempre gane o pierda (Scott 2002: 18). La clave para lograrlo es parametrizar un conjunto de comportamientos y características básicas de modo que estos sean personalizables a través de la data y no del código (Orkin 2002: 33). De esta forma

la IA puede variar esos valores en función al número de victorias o derrotas que el jugador tenga (Scott 2002: 18).

Finalmente, hay una disyuntiva que enfrentar al desarrollar la IA: el hacer o no trampa (Scott 2002: 19). El hacer trampa o *cheating* consiste en aprovechar que se tiene acceso a información adicional que el jugador no percibe —como ubicación de los enemigos, tipos de armas que poseen, lugar hacia donde se dirigen, pre-análisis del mapa, etc.— para modificar el comportamiento de los *NPCs* y darles "cierta ventaja". Existe cierta controversia respecto al tema, pero en suma siempre será necesario utilizar este recurso para contrarrestar la ventaja que finalmente posee un jugador humano frente a la máquina (Scott 2002: 19-20).

1.2.8 Motor de videojuegos

El término *motor de videojuegos* hace referencia a todo el código de un videojuego que forma el conjunto de componentes centrales —sistema de renderizado de componentes 3D, sistema de detección de colisiones, sistema de audio, etc.—; el cual permite a los componentes del videojuego —la mecánica, elementos gráficos, mapas, elementos de audio, etc.— ejecutarse en una capa superior (Rabin 2010: 168; Gregory 2009: 11).

Uno de los objetivos de un motor de videojuegos es aislar el código del juego —mecánica, lógica, AI, etc.— del hardware sobre el cual se ejecuta. Esta abstracción obedece a una arquitectura general formada por diversas capas (véase Anexo J):

- *SDKs* de terceros: *DirectX* y *OpenGL* (gráficos a bajo nivel), STL (librería estándar de C++), *Havok* y *PhysX* (colisiones y física), etc. (Gregory 2009: 31-33).
- Capa de independencia de la plataforma: sistemas de archivos, TCP/UDP, librería de hilos, etc. (Gregory 2009: 34).
- Sistemas centrales: librería *Math*, *parsers XML*, generador de números alatorios, reproducción de video, etc. (Gregory 2009: 34-35).
- Gestión de recursos: permite administrar los activos o *assets* del videojuego tales como modelos 3D, texturas, materiales, fuentes, esqueletos, colisionadores, mapas o mundos, etc. (Gregory 2009: 35-36).
- Conjunto de motores especializados como: motor de renderizado de bajo nivel, motor de colisiones y física, motor de audio, motor de animaciones,

motor de efectos visuales, interfaces de usuario, sistema de scripting, etc. (Gregory 2009: 36-48).

- Finalmente, se tiene la capa de subsistemas específicos que contienen los elementos propios del videojuego en cuestión. La línea divisoria entre el motor y el videojuego en sí se torna difusa en esta capa. Dicha capa contiene la mecánica del juego, manejo de cámaras, IA, etc. (Gregory 2009: 48-49).

1.3 Estado del Arte

La problemática central del presente trabajo es la solución al problema de ingeniería sobre la construcción de un videojuego: implementación de la mecánica, algoritmia empleada en la navegación sobre superficies 3D y estrategia en la implementación de la AI de los *NPCs*. Por ello, se revisará algunas soluciones para estos problemas puntuales.

1.3.1 Videojuegos serios cuya temática es la violencia

A nivel global, existen diversas iniciativas para informar y sensibilizar a través de videojuegos, sobre el modo de actuar de diferentes grupos violentistas, sus consecuencias y la respuesta por parte de la sociedad hacia los mismos. También, existen diversos videojuegos cuya temática se focaliza en los conflictos de diversas partes del mundo como Latinoamérica y el medio oriente y propone algunas soluciones al mismo. A continuación, se discuten algunos de estos videojuegos.

1.3.1.1 September 12th: A Toy World

Es un videojuego desarrollado por Gonzalo Fransca y liberado en 2003 y funciona bajo el reproductor *Adobe Shockwave* Player para navegadores. El videojuego es sencillo y sus características son más propias de un simulador, sin embargo es el primero de una serie de videojuegos a los que se le acuña el término *newsgame* (GFC 2003).

El objetivo en el juego es sencillo: intentar eliminar a los terroristas que se infiltran entre la gente, inevitablemente esto genera daños colaterales y víctimas inocentes. Cuando un civil muere, los civiles a su alrededor se convierten en terroristas. No hay forma de ganar o perder simplemente mostrar que la violencia genera más violencia (GFC 2003).

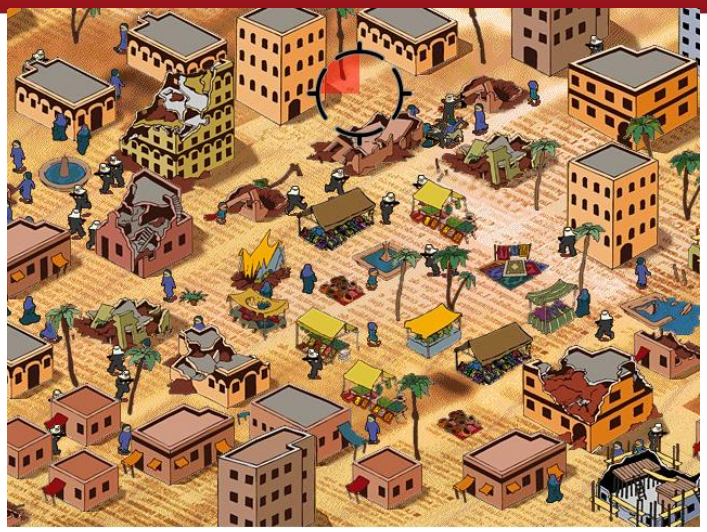


Figura 1.6. September 12th: A Toy World, captura de pantalla.

1.3.1.2 Peace Maker Game

Este videojuego está inspirado en el conflicto Palestino-Israelí y fue desarrollado por un grupo de estudiantes de la *Maestría en Tecnologías del Entretenimiento* (MET) de la universidad de Carnegie Mellon. Es un juego de simulación de gobierno donde el jugador puede asumir cualquiera de dos posibles roles: primer ministro israelí o presidente de palestina. El objetivo del juego es llegar a un acuerdo de paz entre ambas naciones dentro de un contexto político-social muy volátil (Carnegie Mellon News 2005).

Figura 1.7. Peace Maker, Interfaz de juego (Wikimedia Commons 2007).

El videojuego permite al jugador tomar decisiones estratégicas durante diferentes turnos las cuales tendrán repercusiones distintas —y no siempre las esperadas— según el nivel de dificultad que se haya elegido; los efectos de sus decisiones se podrán observar sobre un mapa de la región que muestra indicadores de convulsión social sobre diferentes regiones mostrando videos y audios reales del conflicto actual. La interfaz del juego incluye herramientas para la toma de decisiones dentro de cuatro grandes grupos: militares, políticas, de construcción o diplomáticas. Y refleja las diferentes perspectivas del conflicto para ambas naciones. Toda acción realizada por el jugador tiene un impacto social en diferentes líderes y grupos de interés; y un impacto gubernamental en diversos aspectos económicos, sociales y políticos; de los cuales, cada uno, es medido por diferentes termómetros dentro de dos grandes pestañas que los agrupan. Además, muestran indicadores de aprobación nacional y mundial.

1.3.1.3 Traces of Hope

Traces of Hope es un videojuego de realidad alternativa auspiciado por la *British Red Cross* y desarrollado por *Enable Interactive* en colaboración con diversas organizaciones como *Penguin Books, Reuters AlertNet*, etc. La historia del juego gira alrededor de Joseph, un adolescente de Uganda cuyo padre y hermana fueron asesinados y que, debido al caos y violencia perdió a su madre a quien busca ahora desesperadamente (ARGNet 2008).

El jugador asume un papel de detective que se comunica directamente con Joseph a través de videos, chat embebido en el *website* y diferentes sitios como *Yahoo Answers* con la finalidad de crear una realidad alternativa que busca ser lo más auténtica posible (ARGNet 2008).

Figura 1.8. Trace of Hope, Interfaz de juego (GFC 2008a).

1.3.1.4 Global Conflicts

Global Conflicts es una serie de videojuegos educativos usada como herramienta para la docencia y la enseñanza en las escuelas. Permite a los alumnos explorar, aprender y experimentar la problemática de diversos conflictos globales en sus diferentes dimensiones. Temas como democracia, derechos humanos, globalización, terrorismo, cambio climático y pobreza son tocados en los casos examinados (Global Conficts 2012).

La jugabilidad en cada videojuego es la misma, sin embargo el contexto es el que cambia. El jugador asume el papel de un periodista que deberá arribar al lugar donde se desarrolla el conflicto con el objetivo de construir un artículo a partir de una recopilación de fragmentos o citas de los diálogos con diferentes protagonistas del conflicto. Este puede obtener la información a partir de diálogos basados en un ambiente de confianza o ser más agresivo para obtener la información. El artículo final deberá contener mayor información de valor y ser lo más objetivo posible, en otras palabras, se debe recopilar información de todos los puntos de vista del conflicto sin favorecer a ninguno (Global Conficts 2012).

Algunos de los títulos que forman parte de la serie son (Global Conficts 2012):

- *Border Crossing* (México/USA); trata de una joven asesinada en la frontera de México con Estados Unidos. El jugador debe averiguar cómo fue asesinada, qué futuro le espera a su bebé, entre otros cuestionamientos. Además, podrá observar una sociedad donde se teme a la policía y a los

políticos, las personas invaden terrenos para poder vivir y la pobreza y la violencia son el pan del día. La temática se focaliza en: inmigración, intereses políticos, políticas fronterizas.

- El Patrón (Bolivia); una niña ha desaparecido de una chacra boliviana, en su búsqueda el jugador descubrirá que en dicho lugar sucede algo que está muy mal. La temática se focaliza en: esclavitud para pagar deudas, explotación de nativos y la corrupción.
- *Year One* (Bolivia); la convulsión social en Cochabamba y los disturbios han costado la vida de dos personas. El jugador deberá investigar el trasfondo de las tensiones entre los diferentes grupos étnicos y deberá investigar quién está detrás del conflicto. La temática se focaliza en: racismo, disturbios sociales, emancipación indígena.

Figura 1.9. Global Conflicts, de izquierda a derecha: Border Crossing, El Patrón y Year One (Global Conflicts 2012).

1.3.2 Mallas de navegación (*NavMesh*)

Como ya se mencionó anteriormente, las mallas de navegación son una variante del algoritmo A* que utiliza un forma diferente de representar el mapa. Ésta ha adquirido popularidad debido a que agrupa otras formas de representación de mapas: las grillas y los grafos de nodos o *waypoints.* Finalmente, esta forma de representación de mapas agrupan dos funcionalidades en una: trazado de rutas y detección de colisiones (Rabin 2010: 563-564).

Existen algunas variantes en la forma de representar la malla: una es usando el punto central de cada polígono como conjunto de nodos que demarcarán el camino que recorrerá el *NPC,* lamentablemente la calidad de la ruta trazada de esta manera no es muy buena pues el *NPC* debe dirigirse siempre al centro del polígono; otra posibilidad es utilizando el centro de los bordes que interconectan cada polígono como conjunto de nodos, esto incrementa la calidad de la ruta pero no llega a ser lo suficientemente buena, especialmente al momento de doblar en una esquina; una tercera posibilidad es usar los vértices como nodos del grafo, con

esto mejoramos la calidad de la ruta en las esquinas pero cualquier otro caso la ruta empeora; finalmente se puede crear un híbrido de los últimos dos casos, haciendo uso de los vértices y de los centro de los bordes para crear rutas de mayor calidad (Patel 2006). Todos estos casos se pueden apreciar visualmente en la Figura 1.10.

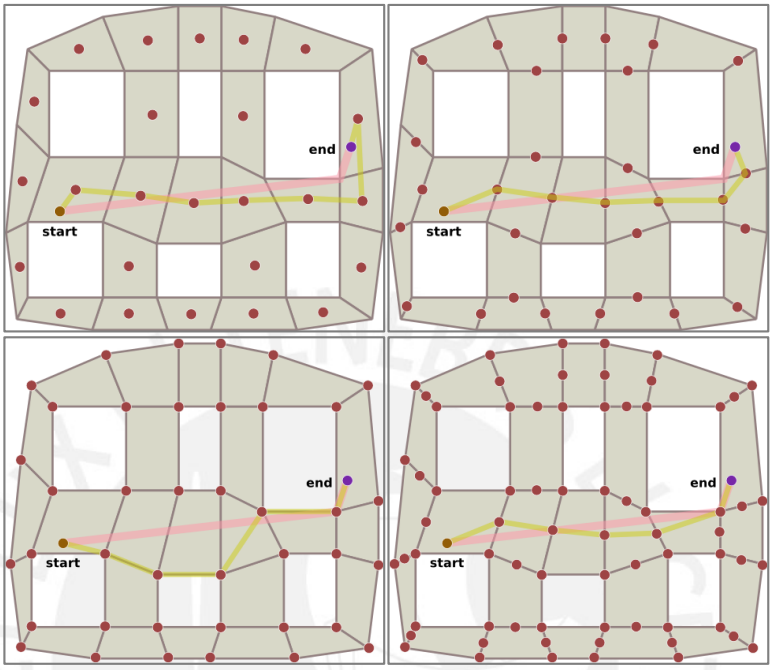


Figura 1.10. Representaciones de las mallas de navegación (Patel 2009).

Centro del polígono, borde el polígono, vértice del polígono e híbrido. La línea amarilla es la ruta obtenida, la línea rosa es la ruta ideal

1.3.3 Estrategias en el diseño de la IA para *NPCs*

La arquitectura de la IA de un videojuego 3D de tipo *FPS* o de tercera persona debe diseñarse siguiendo algunos lineamientos. Paul Tozour sugiere descomponer la IA en cuatro capas: movimiento, animación, combate y comportamiento. La capa de movimiento determina de qué manera el personaje va a moverse y no hacia dónde —esto lo determina la ruta obtenida del algoritmo A*. Por otro lado, la capa de animación se encarga de parametrizar y reproducir las secuencias de animaciones, especialmente aquellas que requieren un control más preciso del personaje en vez de simplemente invocar una reproducción; por ejemplo, agacharse y coger un libro de una mesa, requiere un gran nivel de control del personaje para asegurar una correcta animación. La siguiente capa es la de combate, la cual es responsable de seleccionar las tácticas de combate, acciones de ataque, recarga de municiones, etc. Finalmente, se tiene a la capa de comportamiento, la cual está situada sobre las otras capas y define el comportamiento general del personaje basándose en el

objetivo que se pretende lograr, usualmente se utiliza una máquina de estados finitos *FSM* para modelar todos los estados posibles del personaje (2002: 387-393).

Dentro de la capa de combate, una de las dificultades al momento de elaborar el algoritmo de inteligencia artificial para *NPCs* es diseñar el razonamiento táctico y estratégico de los mismos. Lars Lidén ofrece una solución, aplicada por la muy reconocida compañía *Valve Software*, para atacar a este problema. Parte sobre la base de usar una malla de navegación construida a partir de los nodos centrales de cada polígono; dichos nodos o puntos de paso deben contener información sobre su interconexión con los demás, dicho de otro modo, con cuáles de los otros existe línea de vista; esta información se obtiene realizando un pre-análisis del mapa antes de la ejecución del juego. Finalmente, si se tiene dos enemigos situados dentro de dos polígonos diferentes, los polígonos que se deberá evitar son todos aquellos que poseen línea de vista con los nodos pertenecientes a dichos polígonos; esto se aprecia más claramente en la Figura 1.11. Con este principio y algunas operaciones lógicas, se puede elaborar ataques inteligentes de un solo enemigo sin estar en la línea de vista de los otros, también se puede atacar desde un punto que esté junto a un polígono seguro para poder recargar u esconderse, e incluso realizar ataques de flanqueo —por la espalda— agregando la información de la dirección a la cual está mirando cada enemigo al grafo de nodos (2002: 212-215).

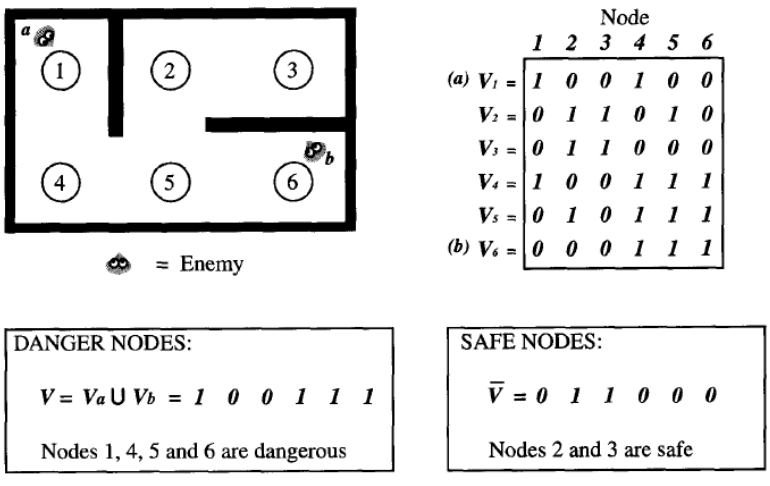


Figura 1.11. Nodos peligrosos y nodos seguros (Lidén 2002: 213).

1.4 Objetivo General

Implementar la mecánica e inteligencia artificial de un videojuego 3D, que funcione sobre una plataforma Web, para satisfacer la necesidad de informar a la población joven-adolescente sobre las consecuencias de las ideologías violentistas.

1.5 Objetivos Específicos

O1: Desarrollar la mecánica, definida en el diseño del videojuego, que permita controlar al personaje e interactuar con el escenario, los enemigos y los objetos.

O2: Adaptar el algoritmo para el trazado de rutas sobre superficies en 3 dimensiones, A*, haciendo uso de mallas de navegación y cuya ejecución no disminuya el rendimiento del videojuego, en otras palabras que la velocidad de renderizado se mantenga sobre un límite establecido.

O3: Desarrollar cada uno de los niveles de la arquitectura del componente de inteligencia artificial que se aplicará sobre los personajes no controlados por el jugador y cuya ejecución no disminuya el rendimiento del videojuego, en otras palabras que la velocidad de renderizado se mantenga sobre un límite establecido.

O4: Integrar, en el producto final, el arte, modelado por un tercero, con la mecánica e IA desarrollada para el videojuego.

1.6 Resultados Esperados

R1: Conjunto de componentes con la implementación de la mecánica del videojuego.

R2: Capa de la aplicación que contenga los componentes con la implementación del algoritmo A*, para el trazado de rutas sobre superficies en 3 dimensiones, usando mallas de navegación. Registro de variación de los *FPS* del videojuego haciendo uso del algoritmo seleccionado.

R3: Capa de la aplicación que contenga los componentes con la implementación de cada una de las capas del subsistema de inteligencia artificial que utilizarán los personajes no controlados por el jugador. Registro de variación de los *FPS* del videojuego haciendo uso del algoritmo seleccionado.

R4: Ejecutable que integre los componentes de la mecánica e inteligencia artificial en los objetos de juego de la aplicación, en otras palabras, cada uno de los personajes no controlados por el jugador, modelo del terreno, interfaz de usuario, y todos los elementos gráficos del videojuego.

1.7 Descripción y sustentación de la solución

Luego de observar la cantidad de trabajo implícita en la construcción de un videojuego 3D, los diversos conceptos involucrados en el mismo, todos los factores de los cuales depende su éxito o fracaso y la enorme cantidad de recursos humanos y tecnológicos necesarios; se decide enfocar el alcance de este proyecto en la capa más alta de la arquitectura del videojuego —la mecánica, IA y el HUD. Las capas más bajas, las cuales incluyen el renderizado de elementos gráficos 3D, sistema de gestión de recursos y componentes, sistema de cámaras, motor físico y otras tareas de bajo nivel se delegan al motor de videojuegos *Unity 3D*. Por otro lado, los elementos gráficos —modelos 3D del terreno y de los personajes, texturas y animaciones— y parte del diseño del *gameplay* —guion y caso base— fueron tercerizados y su elaboración no es cubierta en este trabajo. Por último, el juego debe permitir su ejecución en un navegador web, por esta razón adicional, el uso del motor *Unity 3D* se consideró adecuado para conseguir este objetivo.

1.7.1 Alcance

El presente trabajo, ubicado dentro del área de las *Ciencias de la Computación,* implica el desarrollo de los componentes mecánicos junto con los componentes de IA y la integración de los mismos con los componentes gráficos dentro de los objetos de juego del sistema de componentes del videojuego.

Dentro de estos límites, se tiene la mecánica del videojuego, la cual depende del diseño del mismo. Éste último define al videojuego como videojuego serio 3D situado dentro del género de acción y el subgénero de sigilo. El juego de video está en tercera persona principalmente y, como es propio del género, enfatiza el sigilo como uno de sus más importantes recursos. En este contexto el jugador se enfrenta

a diversos conflictos donde sus decisiones afectan la historia del juego, por ello, se le considera de narrativa emergente intencional.

Como parte del diseño de la mecánica es importante situar al videojuego dentro un género en particular para definir el *gameplay* —que es en esencia la mecánica central del juego—, esto último sirve para definir el conjunto de mecánicas que se implementan en la construcción del producto —el cual se realiza haciendo uso del motor de videojuego *Unity 3D*. Además, se aclara que se recreará únicamente un mundo virtual 3D —o nivel del juego— dentro del cual el jugador puede desplazarse, sujeto a las restricciones de la física y de la jugabilidad. Finalmente, dentro de este marco se definen algunas de las opciones de jugabilidad que serán implementadas, tales como caminar, correr, andar sigilosamente, agacharse, atacar, interactuar con objetos e interactuar con personajes.

Además de ello, se resalta que uno de los componentes más importantes del *gameplay* es que el personaje sea testigo de las injusticias que este grupo subversivo cometía a través de la infiltración y el avance furtivo —motivado por el rescate de un familiar— sin poder reaccionar de modo agresivo ante ello pero sí incrementando el nivel de ansiedad producto de la impotencia de cada injusticia observada. Este indicador de ansiedad, al llegar a cierto nivel, activará la posibilidad de que el jugador pueda atacar con diversas armas, sin embargo, al agredir a un subversivo, el nivel se incrementa mucho más y, al llegar al tope del nivel, el juego acaba y el jugador pierde la partida. El objetivo del juego es completar la misión sin llegar al tope del nivel de ansiedad.

Por tanto, parte de la problemática de ingeniería es el diseño e implementación del conjunto de reglas que se pueden esbozar como reglas o mecánica básica del juego:

i. mecánica de acciones del tipo sigilo para el personaje principal;
ii. interacción con objetos: recoger ítems, usar ítems, abrir puertas, etc.;
iii. capacidad de usar tres tipos de armas: blancas, menores y de guerra;
iv. e incremento y disminución del nivel de ansiedad a lo largo de las misiones en función a las decisiones del jugador.

Por otro lado, se tendrá tres grupos de personajes no controlados por el jugador o *NPC*. El primer grupo realizará una o más acciones predefinidas y tendrá diálogos prestablecidos. El segundo grupo poseerá un conjunto de reglas de IA para

interactuar con el personaje en modo cazador —este tendrá como objetivo ubicar y eliminar al jugador— y hará uso del algoritmo de trazado de ruta óptima *navmesh* y tácticas de combate. El último grupo utilizará un conjunto de reglas de IA que le permitirán desplazarse de un punto a otro del mapa sin ser descubierto o en modo de sigilo y también hará uso del algoritmo de trazado de ruta óptima *navmesh* y tácticas de evasión.

A partir de los requisitos de la IA para los *NPC*, se agrega a la problemática de ingeniería, el diseño e implementación del subsistema de IA, el cual que cuenta con dos componentes principales:

v. la IA, para cada tipo de *NPC*, que permite la interacción con el jugador principal;
vi. y el algoritmo de trazado de ruta que permite a los enemigos desplazarse por el terreno 3D del escenario.

Además, la integración de estos componentes mecánicos junto con los gráficos, dentro del sistema de componentes de objetos de juego, representa la última valla que la problemática de la implementación plantea.

Para la resolución de estos problemas se revisó mucha de la teoría generada desde principio de siglo sobre el desarrollo de videojuegos modernos. Por ello, en este proyecto se utilizan las mejores prácticas en el desarrollo de videojuegos empleadas por la industria, cuyo desarrollo ha sido fruto del ensayo y error, y que fueron registradas —en libros y artículos— por algunos pocos profesionales que no perdieron contacto con el mundo académico.

Finalmente, el videojuego tiene como público objetivo a la población joven-adolescente del Perú, los cuales tienen un fácil acceso al mismo debido a que pueden ejecutarlo desde cualquiera de los navegadores web más populares. La inmersión en el videojuego, el impacto emocional del mismo y el compromiso del jugador es facilitado por el entorno 3D. La difusión de la información es realizada a través de la historia de fondo del juego, la cual puede ser revisada como texto escrito. Sin embargo, la riqueza del juego proviene de la interacción con un escenario plagado de enemigos controlados por el computador, cuyas acciones violentas son observadas por el jugador el cual asume el rol de observador imponente ante los maltratos perpetrados por los subversivos; y que, además, posee recursos de acción muy limitados, a veces casi nulos —entre ellos completar

pequeñas misiones antes de completar la misión principal. Esta mecánica recrea en cierta manera los sentimientos de impotencia generados en una situación real. Cumpliendo así el objetivo principal de informar a través de un juego de dichas experiencias de violencia.

1.7.2 Limitaciones

El principal obstáculo encontrado al desarrollar el proyecto es la poca teoría existente acerca del diseño e implementación de sistemas de IA para videojuegos. Si bien es cierto, sí es posible encontrar información académica acerca de algoritmos de trazado de ruta, IA para robótica o similares; no es así cuando se refiere a la IA aplicada en videojuegos. Este campo requiere de conocimientos aplicados y experiencia en proyectos similares; sin embargo, este conocimiento lo acumulan las grandes empresas desarrolladoras de videojuegos. Lamentablemente, debido a la competencia que existe dentro del mercado, es muy difícil que las mismas publiquen sus soluciones o el código fuente empleado para resolver las diferentes problemáticas enfrentadas al implementar la IA de los videojuegos desarrollados. Son muy pocos los profesionales del mercado que también forman parte del mundo académico; este pequeño grupo sí tiene algunas publicaciones donde dan algunas luces sobre cómo resolver dichos problemas.

Finalmente, para el diseño de videojuegos no es relevante hacer un levantamiento de información para obtener los requisitos de los usuarios debido a que es una práctica muy común que el *game designer* defina los requisitos del videojuego en función al diseño del mismo (Bethke 2003: 213-214). Por ello, esto no representó una limitación.

1.8 Plan del proyecto

En el presente trabajo, como en la mayoría de proyectos, se identifica dos aspectos claves que deben ser gestionados por metodologías propias de la ingeniería. Estos aspectos son el proyecto en sí mismo y el videojuego —el producto—. Para ello se utilizó como base dos metodologías muy conocidas y ampliamente usadas: *PMI* —para la gestión del proyecto— y *Rational Unified Process* —para el desarrollo del producto—, de las cuales se identificaron algunas herramientas que se usaron en este proyecto y no la metodología completa.

1.8.1 Fundamentos de dirección de proyectos del *PMI*

La *Guía de los Fundamentos para la Dirección de Proyectos (Guía del PMBOK)* es una norma muy reconocida y registrada por la *American National Standards Institute (ANSI)* con el código *ANSI/PMI* 99-001-2008. Esta norma define métodos, proceso y prácticas que son producto de la evolución de las buenas prácticas ejercidas por profesionales en todo el mundo dedicados a la gestión de proyectos (PMI 2009: 10).

Esta norma define un grupo de 5 grandes procesos —que aplican a cualquier proyecto— que agrupan a otros 42 subprocesos —los cuales dependen del proyecto en particular. Dichos grandes grupos de procesos son: iniciación, planificación, ejecución, seguimiento y control, y cierre. Finalmente, cada uno de los 42 subprocesos están contenidos dentro de alguna de las nueve áreas de conocimiento que esta norma menciona (PMI 2009: 12):

De todos estos procesos se utilizó en este proyecto un grupo muy reducido de los mismos, principalmente aquellos que ofrecen herramientas que permiten gestionar el alcance y tiempo:

- La estructura de desglose de trabajo o EDT; la cual se encuentra dentro del área de conocimiento de la gestión del alcance, permite establecer de modo visual cuáles son los paquetes de trabajo que este proyecto produce (PMI 2009).
- La lista de actividades que conforman el cronograma; la cual se encuentra dentro del área de conocimiento de la gestión del tiempo, permite que, a partir del EDT, se pueda desglosar cada paquete de trabajo en actividades y, finalmente, con un tiempo estimado asignado a cada una de esas actividades, se establezca el tiempo total del proyecto —plasmado en el cronograma—, junto con los costos por uso de recursos, identificación de ruta crítica, etc. (PMI 2009).

Por otro lado, el desarrollo de videojuegos tiene sus propias etapas o fases que difieren ligeramente del modelo tradicional de *software*. Es por esta razón que, el plan de gestión del proyecto se basa en este esquema.

Se tienen las siguientes fases de desarrollo de videojuegos (Keith 2010: 131):

- **Concepto**: Es la etapa inicial del proceso, donde se formula la idea o temática principal del videojuego, el género, un primer bosquejo del *gameplay*, línea gráfica y los personajes principales. El desarrollo del concepto es casi puramente iterativo y es donde se genera la idea que sirve de base al proyecto. El producto generado es el documento o portafolio que contiene el concepto del juego y la utilidad principal es buscar la aceptación del sponsor y la aprobación del inicio del proyecto. Esta fase no tiene equivalente en el modelo tradicional de construcción de *software*.

- **Pre-producción**: Es la fase equivalente al análisis. Es aquí donde se define completamente el *gameplay* del videojuego, el concept art, la historia, los personajes y la tecnología a usar. Se define el procedimiento de construcción de los activos o componentes del videojuego, y se diseña la mecánica del videojuego —renderizado, física, IA, etc. También, se diseñan los niveles y otros componentes gráficos y mecánicos que definen la calidad de la producción. Esta etapa es totalmente iterativa e incremental.

- **Producción**: Equivale a la fase de implementación. Esta etapa se centra en la construcción de los componentes, integración de los mismos, eficiencia y mejoras incrementales. Se implementa la mecánica central diseñada durante la pre-producción y se construyen los componentes basados en ella.

- **Post-producción**: Es la fase equivalente a las pruebas en el desarrollo de *software*. El equipo se centra en pulir el producto eliminando los *bugs*. Esta etapa mejora el juego de forma incremental. Después de esto, el juego se somete a pruebas de *hardware*.

1.8.2 Estructura de desglose de trabajo (EDT)

En la Figura 1.12 se aprecia el EDT que aplica al proyecto presentado de la siguiente manera: un primer nivel por fases —concepto, preproducción, producción, postproducción— y un segundo nivel por entregables —versiones del concepto de videojuego, documentación de la preproducción y versiones del producto—. Finalmente, se resalta que, para la fase de producción, se maneja un tercer nivel por iteraciones dentro de las cuales se presentan los componentes principales que se desarrollarán por cada una, esto último a un cuarto nivel.

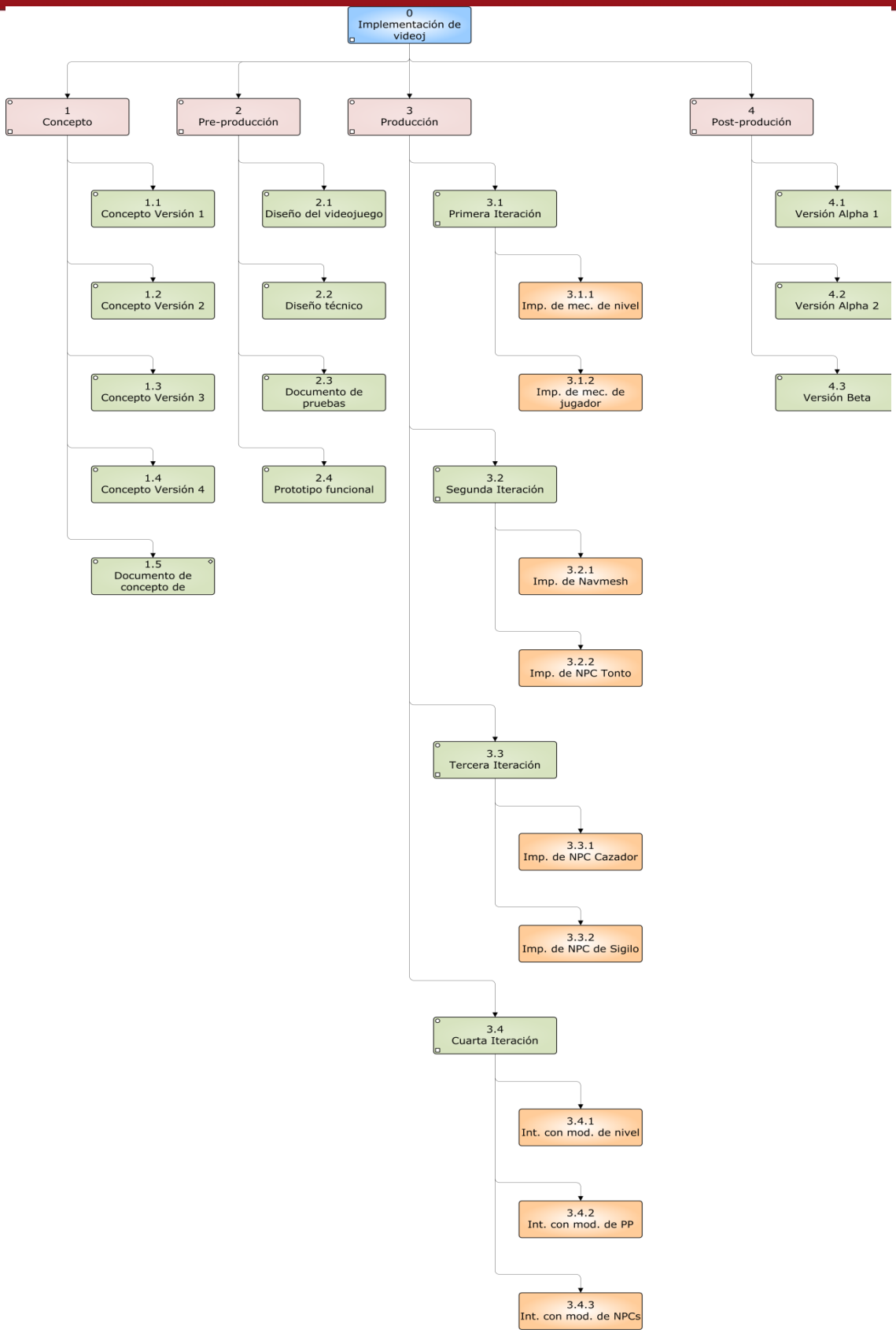


Figura 1.12. Estructura de desglose de trabajo (EDT) del proyecto

1.8.3 Cronograma (Diagrama de Gantt)

Para completar la planificación inicial se desarrolló el cronograma inicial (véase Figura 1.13), el cual se elaborara a partir de los entregables de trabajo del EDT (PMI 2009: 119). Dichos entregables están contenidos dentro de cada una de las fases y develan que, existe una gran cantidad de tiempo invertido en la elaboración del concepto del videojuego y la definición final de diseño o *game design* del videojuego. Ello se debe a que la construcción de un videojuego implica un fuerte componente creativo que debe ser diseñado a nivel técnico para implementarse en la fase de producción. Las pruebas que se realizan en la post-producción son principalmente las de integración y de aceptación finales. Cabe mencionar que, para la estimación de tiempos, se desglosó el diagrama hasta el nivel de las actividades.

Id.	Nombre de tarea	Comienzo	Fin	Duración
1	**Concepto**	**14/07/2012**	**10/08/2012**	**24d**
2	Concepto de videojuego	14/07/2012	10/08/2012	24d
3	**Pre-producción**	**11/08/2012**	**07/10/2012**	**50d**
4	Diseño del videojuego	11/08/2012	23/08/2012	11d
5	Diseño técnico	24/08/2012	29/09/2012	32d
6	Plan de pruebas	29/09/2012	29/09/2012	1d
7	Prototipo funcional	29/09/2012	07/10/2012	8d
8	**Producción**	**07/10/2012**	**02/12/2012**	**48d**
9	Primera Iteración	07/10/2012	20/10/2012	12d
10	Segunda Iteración	20/10/2012	30/10/2012	9d
11	Tercera Iteración	30/10/2012	18/11/2012	18d
12	Cuarta Iteración	18/11/2012	01/12/2012	12d
13	**Post-producción**	**01/12/2012**	**16/12/2012**	**12d 3.67h**
14	Versión Alpha 1	01/12/2012	04/12/2012	2d
15	Versión Alpha 2	06/12/2012	09/12/2012	4d
16	Versión Beta	09/12/2012	15/12/2012	6d

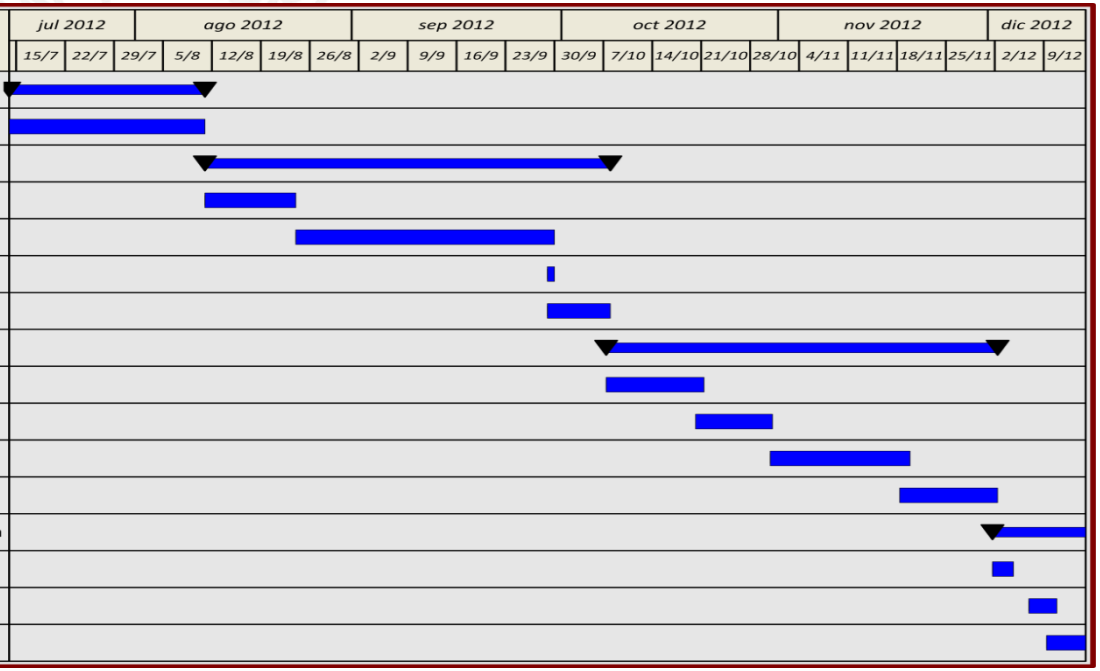


Figura 1.13. Diagrama de Gantt del proyecto

2 Análisis

Como se mencionó, la etapa de análisis está circunscrita a la fase de pre-producción en el desarrollo de videojuegos. En este capítulo se presenta la metodología que se usa para todo el resto del desarrollo de los componentes que corresponden al alcance de este proyecto. Además, se presentan los requisitos del sistema junto con el análisis de la solución propuesta.

2.1 Metodología aplicada para el desarrollo de la solución

Para poder desarrollar el producto se tuvo que seleccionar cuidadosamente una metodología que sea apropiada para un proyecto con un alcance corto como este y un equipo reducido a una persona. Se aclara estos dos factores debido a que, en la industria de los videojuegos, es común tener proyectos donde participan más de cien personas con presupuestos que pueden ser de varias decenas de millones de dólares (Keith 2012: xxi). Por ello, es importante notar el contexto dentro del cual los autores plantean sus metodologías, antes de tomar una decisión.

2.1.1 *Scrum* y *Extreme Programming (XP)*

Clinton Keith, en su libro *Agile Game Development with Scrum*, recomienda fuertemente el marco de gestión de proyecto *Scrum* junto con una metodología de desarrollo complementaria como *Extreme Programming* (2010: 210).

Scrum es un marco de trabajo para el desarrollo de productos de gran complejidad, no es una metodología, debido a que, sus prácticas no son lo suficientemente específicas para indicar, con debido detalle, qué debería hacer cada miembro de un equipo de desarrollo. El desarrollo de *software*, de modo más particular, de videojuegos, utilizando esta metodología se realiza a través de iteraciones, de dos a cuatro semanas de duración, llamadas *Sprints*. En cada iteración se desarrollan algunos de los requisitos, llamados *Product Backlog Item*, tomados de una lista mayor llamada *Product Backlog.* Al final de cada iteración se realizan revisiones junto con el cliente y su representante en el equipo el *Product Owner.* (Keith 2010: 36).

Por otro lado, *Extreme Programming* es una metodología ágil cuyo objetivo principal es potenciar las relaciones interpersonales como factor clave para el éxito en el desarrollo de *software*, promoviendo el trabajo en equipo, fomentando el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en: (i) la retroalimentación continua entre el cliente y el equipo, promoviendo así una comunicación fluida entre ambas partes; (ii) en la simplicidad de sus soluciones y; (iii) coraje para confrontar los cambios. Por todo lo anterior es bastante adecuada para proyectos con requisitos imprecisos y cambiantes, y donde existe un alto riesgo técnico. El ciclo de vida de XP consta de las siguientes fases: exploración, planificación de la entrega, iteraciones, producción, mantenimiento, muerte del proyecto (Letelier 2006).

De los dos párrafos anteriores podemos notar que *Scrum* y *XP* son ideales para productos de gran complejidad, con equipos grandes de desarrollo y muy susceptibles a cambios por poseer requisitos imprecisos. Estas características no aplican a este proyecto.

Por ello, luego de revisar otras alternativas se identificó a otro autor, Erik Bethke, que recomienda usar una adaptación de una metodología muy conocida y usada en proyectos de software de negocios: *Rational Unified Process* (2003: 81-82).

2.1.2 Rational Unified Process (RUP)

RUP es un proceso de ingeniería de *software* que, dentro de un enfoque disciplinado permite asignar tareas y responsabilidades dentro del equipo de desarrollo. Esto aumenta la capacidad predecir el cronograma y presupuesto. Además, las actividades de *RUP* están enfocadas en la producción de modelos de *software* visuales que proveen una rica representación semántica del sistema en desarrollo. Para ello se hace uso efectivo de *Unified Modeling Language —UML—* el cual provee los lineamientos para los modelos. Este es adecuado tanto para equipos pequeños como para equipos grandes de desarrollo de *software*. Finalmente, propone seis buenas prácticas de desarrollo de *software* (Rational Software 2001: 1-2):

1. Desarrollo iterativo de *software*
2. Gestión de requisitos
3. Uso de arquitecturas basadas en componentes
4. Modelado visual de *software*
5. Verificación de la calidad de *software*
6. Control de cambios del *software*

Del último párrafo se puede apreciar que *RUP* es aplicable para equipos pequeños, se enfoca en la asignación de tareas, permite desarrollar de forma iterativa el *software*, enfatiza la captura de los requisitos al inicio del proceso con posibilidad de gestionar los cambios durante el mismo, se enfoca en arquitecturas basadas en componentes como el sistema de componentes de objetos de juego que se mencionó en el capítulo 1 y provee muchas herramientas de modelado al apoyarse en *UML*.

2.1.3 Elección de la metodología

Dados estos fuertes argumentos y, considerando que el tesista cuenta con experiencia en el uso de *RUP*, la metodología de desarrollo a usar será un subconjunto de herramientas de *RUP*. Ellas son las siguientes:

Concepto

- **Documento de visión**

  Este artefacto sienta las bases que permiten tener una mejor comprensión de la aplicación a desarrollarse, pues brindan requisitos de muy alto nivel y características generales del producto. Dicho documento tiene un nombre

muy específico en el proceso de desarrollo de videojuego: Documento de Concepto de videojuego (Brathwaite 2009: 13).

Pre-producción

- **Catálogo de requisitos**

  Este artefacto lista el conjunto de requisitos funcionales y no funcionales obtenido del proceso de educción y, en este caso particular, de deducción de requisitos (Martínez 2002: 8). A esta lista se añaden dos campos importantes: paquete al que pertenecen —esto debido a que la arquitectura agrupa los componentes en paquetes y no módulos dado que esto no aplica—, y prioridad. El único usuario de este sistema es el jugador.

- **Diagrama de casos de uso**

  Es una abstracción de los requisitos que evidencian el nivel de compresión de los mismos. En este se modela las funcionalidades que están disponibles para el conjunto de actores (Rational Software 2001: 4). En este proyecto se tiene como único actor al jugador.

  La especificación de casos de uso es una descripción del flujo de eventos de interacción e intercambio de información entre el actor y el sistema (Rational Software 2001: 4). En este videojuego, la lógica detrás de un caso de uso es más compleja debido a que no es un sistema transaccional sino que es una aplicación mucho más algorítmica. Por ello, no se elaborarán especificaciones para los casos de uso, sino que, más adelante, se usará el diagrama de clases, diagramas de estados, diagramas de componentes, etc. para modelar la lógica detrás de los casos de uso.

- **Documento de diseño técnico**

  Mejor conocido como documento de diseño, que también es el nombre que se le da a al documento que contiene el *game design*; por ello, el uso del término *técnico* para evitar ambigüedades. Es un documento que especifica cómo el sistema cumple sus objetivos y de qué forma es implementado (Martínez 2002: 9). Además, en este tipo de proyectos, agrupa elementos propios del análisis —como el catálogo de requisitos y diagrama de casos de uso— y elementos propios del diseño —diagrama de clases, arquitectura de *software*, diagrama de paquetes, diagrama de estados, etc. (Bethke 2003: 129-158).

Producción

- **Videojuego desarrollado por iteraciones**

  Cualquier fase de *RUP* puede ser fragmentado en iteraciones. En la fase de desarrollo o producción cada iteración debe culminar con la entrega de un producto ejecutable. Al final de todas las iteraciones se tendrá el producto completo (Rational Software 2001: 7).

- **Plan de pruebas**

  Es un documento que contiene la estrategia de pruebas que se usará para probar el videojuego. Algunas de ellas son: pruebas beta, pruebas unitarias, pruebas de baja de caja negra, pruebas de caja blanca, etc. (Bethke 2003: 154). El plan va incluido en el documento de diseño técnico y los resultados de las pruebas se registran al final de cada ronda de pruebas ejecutadas en la fase de producción, para las unitarias, y post-producción, para las de integración, beta, etc.

Post-producción

- **Despliegue**

  Es el flujo de trabajo que produce entregas externas de la aplicación, en otros términos, empaqueta, instala y distribuye la aplicación (Rational Software 2001: 13). En este proyecto, la aplicación se instalará en un servidor web y se distribuirá a través del navegador.

  En esta fase también se conducen pruebas beta y pruebas de aceptación, las cuales serán ejecutadas para este proyecto (Rational Software 2001: 13).

2.2 Identificación de requisitos

Una de las grandes diferencias entre el desarrollo de software tradicional y el desarrollo de videojuegos es el proceso de toma de requisitos. En este tipo de proyectos el diseñador debe ir un paso adelante y evitar el proceso tradicional de educción de requisitos para dar lugar a un proceso creativo de definición de los mismos, el cual opcionalmente podría ser retroalimentado con alguna muestra de usuarios (Bethke 2003: 213-214).

Este proceso creativo tiene lugar principalmente en la fase de diseño de videojuego o *game design*, cuyo artefacto generado es el *documento de diseño del videojuego.*

A partir del *gameplay* y de la mecánica definida en dicho documento se elabora la lista de requisitos del videojuego (Bethke 2003: 82). En este punto, a modo de paréntesis, se hace énfasis que la elaboración de dicho documento contó con la participación de un artista, una sicóloga y otros *gamers* que colaboraron con dicho videojuego.

2.2.1 Catálogo de requisitos

Dentro de esta lista (véase la Tabla.2.1), cada requerimiento está identificado con un número, además pertenece a un paquete, posee una descripción que indicar que se debe poder realizar dentro del videojuego, además indica el tipo de requerimiento del cual se trata y finalmente la prioridad. Para este último campo, la escala es la siguiente:

- Alta
- Media
- Baja

Vale mencionar que el usuario, en este caso, siempre será el mismo: el jugador; por tanto no se relacionan los requisitos con el tipo de usuario.

Id	Paquete	Descripción (El videojuego debe permitir...)	Tipo	Prioridad
1	Control de personaje	Que el personaje pueda caminar sobre cualquier terreno transitable.	Funcional	ALTA
2	Control de personaje	Que el personaje pueda correr sobre cualquier terreno transitable.	Funcional	ALTA
3	Control de personaje	Que el personaje pueda caminar de puntillas para disminuir el nivel de ruido.	Funcional	ALTA
4	Control de personaje	Que el personaje pueda agacharse	Funcional	ALTA
5	Control de personaje	Que el personaje pueda avanzar mientras está agachado.	Funcional	ALTA
6	Control de personaje	Que el personaje pueda echarse sobre el suelo.	Funcional	MEDIA
7	Control de personaje	Que el personaje pueda avanzar mientras está echado en el suelo.	Funcional	MEDIA
8	Control de personaje	Que el personaje pueda atacar a su oponente.	Funcional	MEDIA
9	Interacción con objetos	Que el personaje pueda abrir una puerta.	Funcional	ALTA
10	Interacción con objetos	Que el personaje pueda usar un interruptor.	Funcional	ALTA
11	Interacción con objetos	Que el personaje pueda recoger un ítem.	Funcional	ALTA
12	Interacción con objetos	Que el personaje pueda desechar un ítem.	Funcional	ALTA
13	Interacción con objetos	Que el personaje pueda usar un ítem.	Funcional	ALTA
14	Interacción con objetos	Que el personaje pueda usar un arma	Funcional	MEDIA

Continúa.

Continuación.

Id	Paquete	Descripción (El videojuego debe permitir...)	Tipo	Prioridad
15	Inteligencia Artificial	Que el personaje pueda recibir información en forma de texto de parte de un NPC	Funcional	ALTA
16	Inteligencia Artificial	Que el personaje pueda ser perseguido por un NPC	Funcional	ALTA
17	Inteligencia Artificial	Que el personaje pueda ser perseguido y atacado por un NPC	Funcional	ALTA
18	Inteligencia Artificial	Que el personaje pueda ser evadido por un NPC	Funcional	ALTA
19	HUD	Ver nivel de ansiedad	Funcional	ALTA
20	HUD	Ver nivel de ruido	Funcional	ALTA
21	HUD	Ver posición en mini mapa	Funcional	MEDIA
22	Sistema de sonido	Escuchar música de fondo	Funcional	ALTA
23	Sistema de sonido	Escuchar efectos de sonido	Funcional	ALTA
24	Sistema de sonido	Regular el volumen de la música de fondo	Funcional	MEDIA
25	Sistema de sonido	Regular el volumen de los efectos de sonido	Funcional	MEDIA
26	Menú	Mostrar la pantalla de menú	Funcional	MEDIA
27	Menú	Navegar por las opciones de menú	Funcional	MEDIA
28	Menú	Mostrar la pantalla de carga de partida nueva	Funcional	MEDIA
30	Menú	Mostrar la pantalla de dificultad del videojuego	Funcional	BAJA
31	Menú	Mostrar la pantalla de configuración de teclado	Funcional	BAJA
32	Menú	Mostrar la pantalla de configuración de audio	Funcional	MEDIA
33	Menú	Mostrar una animación para la carga de videojuego	Funcional	BAJA
34	Menú	Mostrar una animación para la carga de partida	Funcional	BAJA
35	Sistema	Cargar nueva partida	Funcional	ALTA
38	Sistema	Abandonar una partida	Funcional	MEDIA
39	Sistema	Reproducir cinemática	Funcional	BAJA
40	Sistema	Configurar controles de entrada	Funcional	BAJA
41	Sistema	Configurar nivel de dificultad del videojuego	Funcional	BAJA
42	No aplica	Ejecutarse en los siguientes navegadores web: Internet Explorer 9, Mozilla Firefox 11, Google Chrome 21 y versiones compatibles	No funcional	ALTA
43	No aplica	Ejecutarse con una resolución mínima de 1024 x 576 píxeles.	No funcional	MEDIA
44	No aplica	Ejecutarse en los sistemas operativos Windows XP, Vista y 7.	No funcional	ALTA
45	No aplica	Ejecutarse terminales que posean tarjetas gráficas que permitan el renderizado de elementos 3D.	No funcional	ALTA

Tabla.2.1. Lista de requisitos

2.2.2 Casos de uso

Como ya se mencionó, los casos de uso son la abstracción de los requisitos, por ello en este diagrama se ha cubierto todos los requisitos funcionales de la lista con el diseño de casos de uso (véase Figura 2.1), de los cuales algunos agrupan varios requisitos en uno, como *mostrar menú,* y con algunos que cubren la necesidad directa de un único caso de uso —se puede apreciar esta relación más claramente en la matriz de trazabilidad—. También, se ha identificado a un único actor para todos ellos: el jugador; por ello, debido a que todos están relacionados con el mismo actor, no se han usado conectores entre el actor y cada caso de uso. La lista completa de casos de uso agrupados por paquetes se encuentra en el Anexo A.

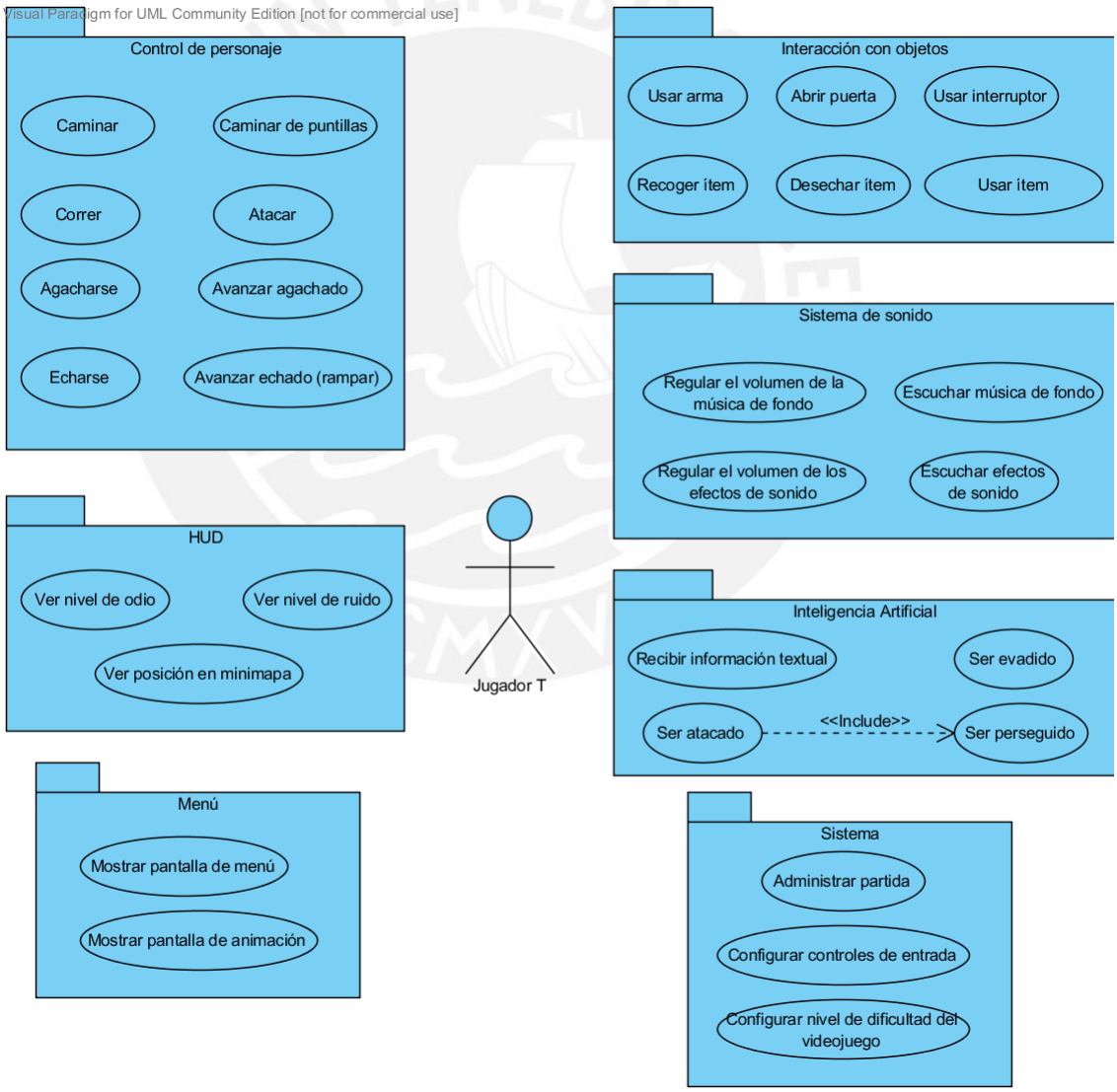


Figura 2.1. Diagrama de casos de uso por paquetes

2.2.3 Matriz de trazabilidad

La matriz de trazabilidad muestra la relación que existe entre los requisitos y cada caso de uso. Además los agrupa por paquetes según aquellos a los que corresponden.

Paquete	Caso de uso	Id	Descripción de requerimiento (El videojuego debe permitir...)
Control de personaje	Caminar	1	Que el personaje pueda caminar sobre cualquier terreno transitable.
Control de personaje	Correr	2	Que el personaje pueda correr sobre cualquier terreno transitable.
Control de personaje	Caminar de puntillas	3	Que el personaje pueda caminar de puntillas para disminuir el nivel de ruido.
Control de personaje	Agacharse	4	Que el personaje pueda agacharse
Control de personaje	Avanzar agachado	5	Que el personaje pueda avanzar mientras está agachado.
Interacción con objetos	Recoger ítem	11	Que el personaje pueda recoger un ítem.
Interacción con objetos	Desechar ítem	12	Que el personaje pueda desechar un ítem.
Interacción con objetos	Usar ítem	13	Que el personaje pueda usar un ítem.
Inteligencia Artificial	Recibir información textual	15	Que el personaje pueda recibir información en forma de texto de parte de un *NPC*
Inteligencia Artificial	Ser perseguido	16	Que el personaje pueda ser perseguido por un *NPC*
Inteligencia Artificial	Ser atacado	17	Que el personaje pueda ser perseguido y atacado por un *NPC*
Inteligencia Artificial	Ser evadido	18	Que el personaje pueda ser evadido por un *NPC*

Tabla 2.2. Matriz de trazabilidad de requisitos con casos de uso

2.3 Análisis de la solución

Este análisis incluye la justificación del sistema, el cual incluye el análisis técnico-económico y de viabilidad. También, se presentan algunas de las restricciones de tiempo y costos existentes y, finalmente, la línea base de la solución.

2.3.1 Justificación del proyecto

Antes de iniciar un proyecto se debe evaluar el beneficio que generará el mismo, para ello se evalúa los aspectos técnicos y económicos para finalmente dar un veredicto respecto a la viabilidad del mismo.

Como se discutió ampliamente en la definición de la problemática, el producto sirve para difundir una experiencia que debe mantenerse en la memoria colectiva. También, se mencionó que este producto no obedece a una necesidad funcional como un sistema o incluso un motor de videojuego, sino a una necesidad social. El objetivo, entre líneas, es dar un aporte para evitar que las ideologías violentistas resurjan; sin embargo, este enfoque social no se analiza por no encontrarse dentro de los objetivos presentados en este documento de ingeniería.

El impacto social que este producto lograría sería alto. Se ha revisado con cierto detalle, qué grupos sociales son los usuarios recurrentes de este tipo de tecnologías: los jóvenes adolescentes de entre 14 a 25 años de edad del Perú, sin embargo, debido a que el producto estará en un sitio web su alcance puede ampliarse. Este conjunto de usuarios podrá interactuar con las experiencias que el videojuego ofrezca. Dentro de estas experiencias se podrá apreciar el impacto que las ideologías extremistas ocasionaban a las familias y a las comunidades, y se deberá tomar decisiones para evitar que el daño de extienda a través de la semilla del odio y el rencor. La distribución se realizará a través de Internet en forma gratuita, mediante un complemento para navegadores web y los jugadores podrán jugar libremente el videojuego.

Los aspectos teóricos necesarios para el desarrollo de este proyecto han sido extraídos de diferentes fuentes bibliográficas y no se ha generado conocimiento o nuevos vacíos en el cuerpo teórico que deban ser cubiertos. En cambio se ha utilizado diferentes fuentes de distintos enfoques —sicológico, educacional, científico y de aplicación— para sentar las base teórica del producto.

2.3.2 Viabilidad del proyecto

Este proyecto tiene como objetivo la elaboración de un videojuego serio que se distribuirá a través de un sitio web y podrá jugarse en línea.

Como se mostró en el estado del arte, existen soluciones que buscan atacar la problemática planteada de distintas formas, sin embargo ninguna se adapta totalmente a los requisitos planteados. Esta solución posee características únicas que argumentan a favor de su desarrollo.

Por otro lado, se utiliza un motor de videojuegos gratuito, que facilita el desarrollo de la solución volviéndola viable para ser elaborada por un sola persona. Los activos (modelos 3D, audios, etc.) necesarios serán obtenidos de fuentes gratuitas.

2.3.2.1 Viabilidad técnica

La complejidad técnica de este proyecto en aspectos como, el renderizado 3D, motor de física, motor de audio, animación de modelos, etc., es resuelta con el uso de un motor de videojuegos. Dicho motor abstrae las capas más bajas de la arquitectura del videojuego.

La mecánica y la lógica de la IA son implementadas mediante el uso de *scripts* valiéndose del sistema de gestión de componentes de objetos de juego, que permite añadir funcionalidad a los objetos mediante el uso de *scripts* de código que definen el comportamiento y las reglas de juego que son implementadas.

Finalmente, la integración de los componentes generados es posible gracias al motor y al sistema de gestión de objetos de juego que permite añadir o remplazar modelos 3D y aplicarles luego los scripts de comportamiento. Esto brinda las capacidades técnicas para la integración de los componentes lógicos a los componentes artísticos o gráficos.

2.3.2.2 Viabilidad temporal

Se ha estimado el desarrollo de este proyecto en el trascurso de 5 meses, los cuales incluyen un aproximado de un mes para el desarrollo del concepto —el cual se realizó en colaboración con artistas, psicólogos y jugadores— y un mes más para el diseño del videojuego. Dicho proceso fue un trabajo creativo e iterativo y se toca brevemente en algunas partes de este documento.

Finalmente, para el desarrollo o producción del producto, el cual incluye el diseño técnico, desarrollo y pruebas, se estimó un total de tres meses y medio en ciclos iterativos dentro de cada fase. El tiempo estimado por semana para este proyecto fue de 24 horas, los cuales se redistribuyeron y amortiguaron en el aporte de algunos colaboradores en las fases de concepto y diseño de videojuego, las cuales requerían un mayor nivel de creatividad y menos de ingeniería propiamente dicha.

2.3.2.3 Viabilidad económica

Las herramientas utilizadas para el desarrollo de este producto no requieren ninguna inversión económica. El motor *Unity 3D* es gratuito, la herramienta *CASE Visual Paradigm UML* y el *IDE MonoDevelop* también lo son. Para el despliegue se usará un servidor web que aloje el producto de forma gratuita.

El equipo necesario para el desarrollo será el equipo personal del tesista con el *software* instalado, además de las instalaciones de la universidad en lo que refiere al acceso a Internet.

Solamente, a modo de estimación, se esboza un costo aproximado del producto. Para ello, se define una tarifa por hora que depende de la fase de desarrollo y alguna referencia del mercado. Las tarifas definidas para el proyecto son:

Grupo	Recurso	Horas por semana	Capacidad máxima	Tasa estándar
Concepto	Tesista	24	100%	S/. 30.00/hora
Pre-producción	Tesista	24	100%	S/. 40.00/hora
Producción	Tesista	24	100%	S/. 45.00/hora
Post-producción	Tesista	24	100%	S/. 30.00/hora

Tabla 2.3. Recursos del proyecto

Con esa información se define el costo de desarrollo, dichos costos son:

Nombre de tarea	Comienzo	Fin	Horas	Costo por hora	Total
Concepto	14/07/12	10/08/12	80h	S/. 40.00	S/. 3,200.00
Pre-producción	11/08/12	07/10/12	169h	S/. 42.00	S/. 7,098.00
Producción	07/10/12	01/12/12	156h	S/. 48.00	S/. 7,488.00
Post-producción	01/12/12	15/12/12	36h	S/. 40.00	S/. 1,440.00
					S/. 19,226.00

Tabla 2.4. Costos del proyecto

2.3.3 Definición base de la aplicación

El sistema cuenta con seis paquetes a nivel general —estos se detallan en la siguiente sección, los cuales integran y simplifican el esquema de paquetes planteado en el diagrama de casos de uso. Estos son:

2.3.3.1 Sistema de videojuego

Este paquete incluye al componente *AdministradorJuego* el cual almacena las variables más importantes del videojuego como nivel, dificultad, etc. Además, este tiene referencias a otros componentes críticos como el sistema de menús, el sistema de *HUD* y el control del personaje. También, contiene las cinemáticas y contiene las pantallas de animación y de carga.

2.3.3.2 Control de personaje

El control del personaje incluye componentes tales como el control propiamente dicho; en otras palabras, recibe los comandos de entrada del jugador y ejecuta las acciones del personaje, actualiza los estados y lanza las animaciones adecuadas. También, tiene un administrador de ítems que, a pesar de no funcionar como inventario, sí permite al jugador utilizar ciertos ítems en momentos específicos del juego. Finalmente, tiene una referencia directa al sistema de interacción con objetos y también controla el nivel de ruido el cual alerta a los *NPCs* que se encuentran dentro del rango de ataque a pesar de no estar dentro de su línea de vista.

2.3.3.3 Interacción con objetos

La interacción con objetos contiene métodos que son llamados al momento de que los colisionadores o *colliders* lanzan un *triger* cuando el jugador posiciona el puntero sobre ellos. También, se tienen gestores que poseen referencias a todos los elementos que agrupan.

2.3.3.4 Inteligencia artificial

El paquete de IA es uno de los más importantes y que, en el diagrama de clases de la Figura 2.2, se muestra de forma simplificada. Este agrupa controles para cada tipo de *NPC*, estados, animaciones y acciones. Sobre cada *NPC* existe un componente del tipo *AiNpc,* el cual contiene la lógica del comportamiento del mismo el cual tiene ciertos factores configurables como la agresividad.

Finalmente, cada *NPC* tiene una referencia al componente *AStar*, el cual construye las rutas que estos recorrerán cuando necesiten llegar a un objetivo específico. La malla de navegación es una colección de polígonos triangulares, los cuales están compuestos de 3 nodos que representan sus vértices.

2.3.3.5 Head Ups Display o Interfaz dentro del juego

El *HUD in game* contiene indicadores que están ubicados dentro de la vista del videojuego y brindan información importante al jugador, dado que estos sirven de apoyo para el objetivo del juego, según el tipo de *gameplay*. Los indicadores son: ubicación en el mini-mapa, nivel de ansiedad que determina la continuidad o término del juego —si el jugador permite que el nivel de ansiedad del personaje llegue al límite este perderá la partida— y el nivel de ruido el cual evidencia la ubicación del personaje para los *NPCs*.

2.3.3.6 Menú

Finalmente, el sistema de menús que contiene la lógica para el dibujado de la ventana —en *Unity* se debe dibujar la ventana basándose en texturas y gráficos 2D— y también se relaciona con el componente *Administrador-Juego* para poder alterar las variables que las opciones de configuración permiten como nivel de volumen o dificultad del juego.

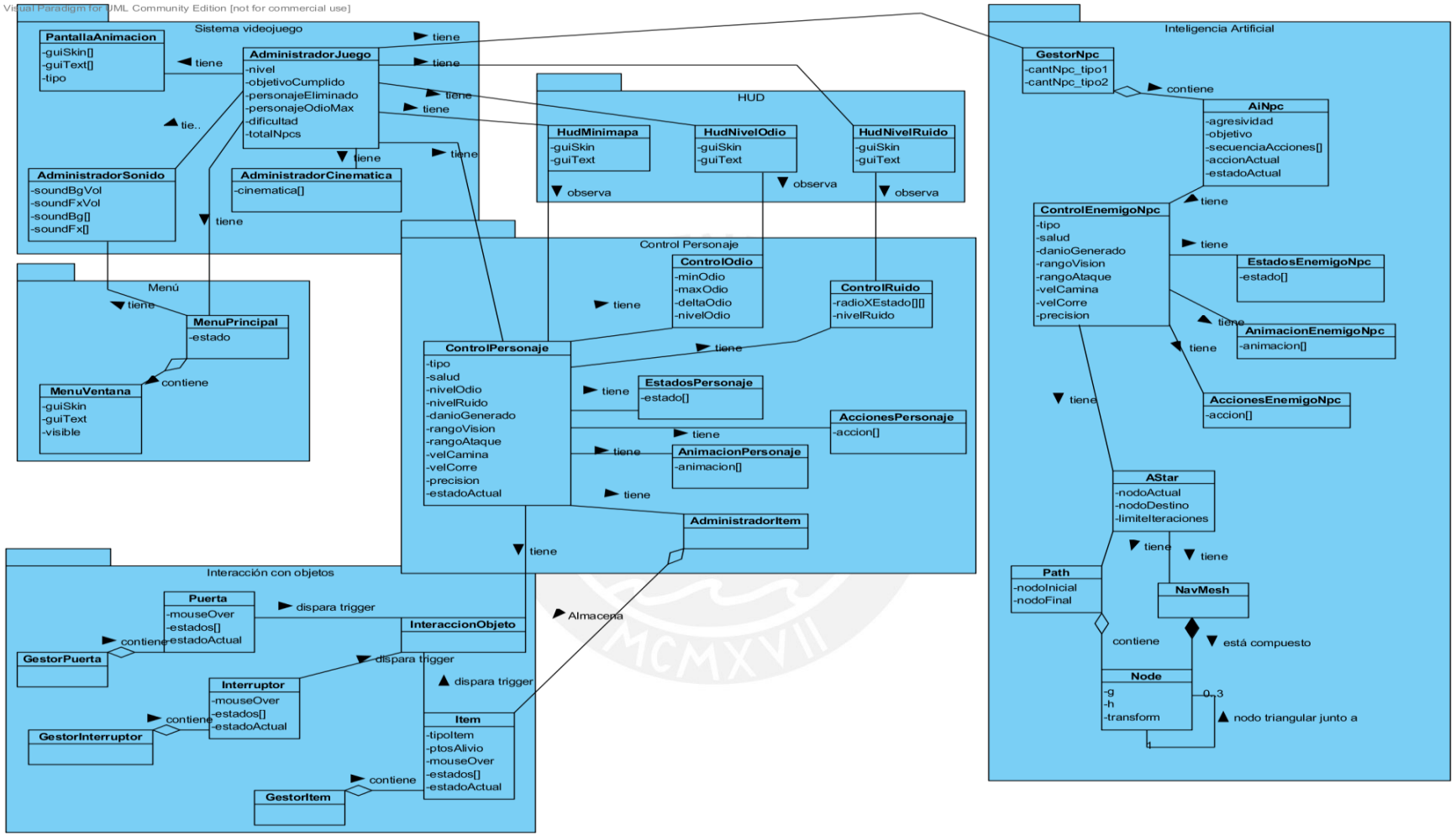


Figura 2.2. Diagrama de clases de diseño

3 Diseño

Dentro del proceso de producción de un videojuego, el diseño técnico forma parte de la fase de pre-producción del mismo y, como se mencionó en capítulos anteriores, se acuña el término *técnico* para diferenciarlo del diseño de la *jugabilidad* y contenido del videojuego, mejor conocido como *gamedesign.*

Por tanto, en este capítulo se expone la arquitectura de la aplicación, a través de una representación gráfica de los componentes que forman parte del producto desde diferentes ángulos del mismo. Por otro lado, se expone el diseño de la inteligencia artificial justificando la elección de la estrategia usada. Finalmente, se presenta el diseño del *HUD in game* junto con la navegabilidad de los menús de la aplicación.

3.1 Arquitectura de la solución

Este videojuego, como la mayoría, se construye en la capa más alta de una arquitectura más compleja: la capa de subsistemas específicos del juego (Gregory 2009: 29). Las capas subyacentes pertenecen propiamente al motor de videojuego —sistema de renderizado de componentes 3D, sistema de detección de colisiones

y física, sistema de audio 3D, gestión de activos de juego, etc.—. Esta sección se concentra en la capa de subsistemas específicos del juego.

3.1.1 Representación de la arquitectura

La arquitectura de este videojuego se ubica en la capa de subsistemas específicos del juego, dichos subsistemas son propios de este videojuego y deben respetar cierto conjunto de patrones arquitectónicos, los cuales son:

- **Orientado a objetos:** Esta aplicación considera el diseño orientado a objetos, teniendo en cuenta el uso de clases, y objetos instanciados a partir de las clases; además, debe permitir la manipulación de datos respetando el principio de encapsulamiento de la información a través de métodos *get* y *set,* siempre que sea posible —debido a que, para poder establecer algunas relaciones dentro del sistema de componentes de objetos de juego, es necesario declarar algunos parámetros como públicos—; permitir el uso de interfaces —las cuales son ampliamente utilizadas debido al sistema de componentes ya mencionado, con métodos como *update, start*, entre otros—; polimorfismo —dado el uso de componentes que heredan de distintas familias, en este caso en particular, la familia de *scripts* que define el comportamiento—; y, finalmente, la herencia considerando que, dentro del esquema de componentes que forman objetos de juego o *game objects*, la herencia sólo es factible dentro de una misma familia de componentes —*scripts* de comportamiento, *renders*, física, entre otros— y no para entidades —por ejemplo, esto es incorrecto: rifle que hereda de arma que hereda de objeto.

- **Orientado a eventos:** El videojuego responde casi exclusivamente a eventos, no es un sistema transaccional que responde al ingreso de datos, sino a las acciones del jugador dentro del videojuego. Únicamente, en el menú, el usuario configura ciertos parámetros que podrían traducirse como datos de entrada, por ejemplo: nivel de dificultad, nivel de volumen, etc. Además, dentro de la arquitectura de eventos existe un productor de eventos y un consumidor de eventos los cuales cuentan con un intermediario que es el administrador de eventos (HANSON 2005: 2). En este caso, el productor de eventos se encuentra en las capas más bajas, dentro del motor de videojuegos, este genera eventos que responden a los dispositivos de entrada como el teclado o mouse —los cuales son

consumidos por los componentes que se encuentran dentro de la capa de control del personaje—, también produce eventos que responden a las colisiones detectadas por el sistema de física y colisiones del motor —los cuales son consumidos por diferentes componentes del juego como el de interacción con objetos.

3.1.2 Diseño de la arquitectura

Uno de los aspectos claves del diseño de la arquitectura es el *sistema de componentes de objetos de juego*, como ya se ha mencionado, dicho sistema permite la construcción de objetos de juego a través de la integración de varios componentes de distintas familias. El conjunto de componentes en el cual este proyecto se enfoca son los *scripts* de comportamiento —los cuales definen la lógica del juego—, los de construcción de *HUD*, entre otros, los cuales se ubican en la capa de subsistemas particulares de este videojuego.

Como se mencionó en el marco teórico, este *sistema de componentes de objetos de juego* se contrapone al modelo jerárquico tradicional, el cual posee varias deficiencias (véase la Tabla 3.1) cuando se utiliza para el desarrollo de videojuegos, entre ellas la poca facilidad de reutilizar el código, un rendimiento menor, dificultad para comprender el código y, por ende, para mantenerlo entre otros (Wilson 2002; West 2007). Debido a ello, se tomó la decisión de utilizar el *sistema de componentes de objetos de juego* y tomarlo como factor crítico al momento de identificar el motor de videojuego que se utilizó —este proceso se detalla en el siguiente capítulo.

	Sistema de componentes de objetos de juego	**Modelo de herencia jerárquico tradicional**
Escalabilidad	Alta	Baja
Reusabilidad	Alta	Baja
Rendimiento	Alto	Medio
Facilidad de comprensión	Alta	Baja
Facilidad para implementar cambios	Alta	Baja
Facilidad de mantenimiento de código	Alta	Baja
Probabilidad de necesitar refactorización	Baja	Alta
Probabilidad de obtener consecuencias inesperadas	Baja	Alta

Continúa.

Continuación.

	Sistema de componentes de objetos de juego	Modelo de herencia jerárquico tradicional
Probabilidad de enfrentar el problema de herencia múltiple de diamante de la muerte	Nula	Media
Probabilidad de tener funcionalidades duplicadas	Baja	Alta (En caso de implementar la funcionalidad a nivel de las hojas del árbol de herencia)
Probabilidad de tener objetos con sobrecarga de funcionalidades innecesarias	Baja	Alta (En caso de implementar la funcionalidad a nivel de la raíz del árbol de herencia)

Tabla 3.1. Comparación entre el Sistema de componentes de objetos de juego y el Modelo de herencia jerárquico tradicional

En la Figura 3.1 se aprecia el diagrama de componentes de la aplicación, donde se muestra únicamente la familia de componentes del tipo script —de modo general, debido a que agrupa los scripts de comportamiento junto con los de manejo de interfaz—. En ese diagrama, los objetos de la clase *Game Object* son contenedores de componentes, dichos componentes típicamente son: modelos 3D, texturas, física, *colliders*, *transforms* (ubicación y tamaño), *scripts*, etc. De los cuales los *scripts* de comportamiento, heredan de la clase *MonoBehaviour* y poseen funciones que son llamadas en ciertos eventos: *Awake*, *Update*, *Start* y *Reset*. —Por ejemplo, *Update* es llamado una vez en cada *frame*—. Los *GO* u objetos de juego, pueden agrupar otros *GO* a modo de colección. Esto se expresa con una agregación entre *GOs* —por ejemplo, *gestorNPC* posee una relación de agregación con el *GO* npc. Dicha relación también se refleja a nivel de componentes, donde, el componente *GestorNpc*, posee una referencia a la colección de componentes del tipo *ControlNpc*. Además, la relación entre un componente contenido en un *GO*, se expresa con una línea que tiene un círculo del lado del *GO*. Además, las asociaciones entre componentes de comportamiento se expresan con líneas continuas.

Finalmente, el videojuego hace uso de un *plugin* para navegadores web el cual permite su ejecución en un navegador. Dicho *plugin* se ejecuta del lado del cliente y al ser activado inicia la descarga del videojuego hasta completarla sin requerir los servicios de ningún otro componente. Este es el único requerimiento no funcional que la aplicación presenta y que, además, es satisfecho por el diseño de la arquitectura de la solución.

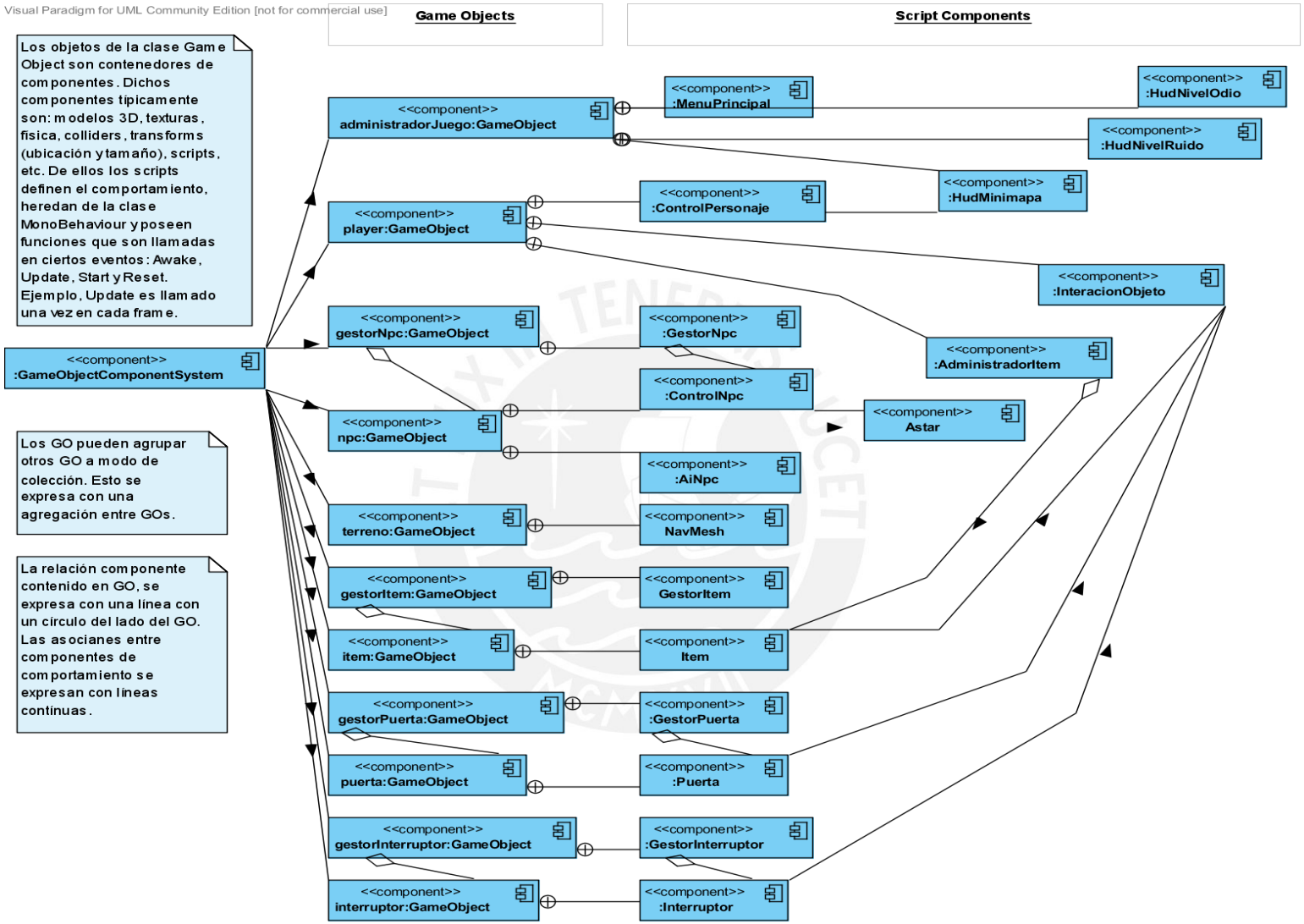


Figura 3.1. Diagrama de componentes de la aplicación utilizando el sistema de componentes de objetos de juego

3.1.3 Vista lógica

La vista lógica de la aplicación, la cual se aprecia en la Figura 3.2, muestra al conjunto de paquetes que contienen los componentes de la mecánica, IA y *HUD* del videojuego agrupados por capas según su funcionalidad. Dichas capas son:

- Sistema general: el cual contiene los componentes del sistema de videojuego que llevan el registro de los parámetros generales que afectan al videojuego en conjunto como: nivel, dificultad, cantidad de enemigos, entre otros. Por otro lado, el sistema del *Head-Up Display* o *HUD* —que muestra una interfaz que brinda información al jugador dentro del videojuego— y el sistema de menú —el cual dibuja todas las ventanas correspondientes del sistema de menú y establece las relaciones entre los controles de volumen, dificultad, entre otros, con él mismo—, también pertenecen a esta capa. Finalmente, el sistema de sonido, el cual se encarga de separar los efectos de sonido de la música de fondo y de regular los niveles de audio, también se encuentran en esta capa.
- Personaje: Esta capa agrupa los paquetes que corresponden al control del personaje en sí mismo junto con el sistema de interacción con objetos, el cual contiene los consumidores de eventos de colisión que forman parte de la lógica del videojuego, en otras palabras, de la mecánica del mismo.
- *Non Player Character*: Esta capa contiene, al igual que el personaje, el paquete de control del *NPC*, el cual, a diferencia de este último, es controlado por el paquete de inteligencia artificial de esta capa. El paquete de IA, también hace uso del paquete de trazado de ruta —donde se encuentran los componentes que permiten la representación del mapa haciendo uso de una malla de navegación y los componentes que contienen el algoritmo A* de trazado de rutas— para obtener el camino que recorrerá el *NPC* desde su ubicación actual hasta el destino que la IA elija.
- Finalmente, se tiene la capa que contiene todos los recursos del motor de videojuegos, entre ellos los sistemas de colisión y de física, el sistema de renderizado, el sistema de componentes, el sistema administrador de eventos, etc.

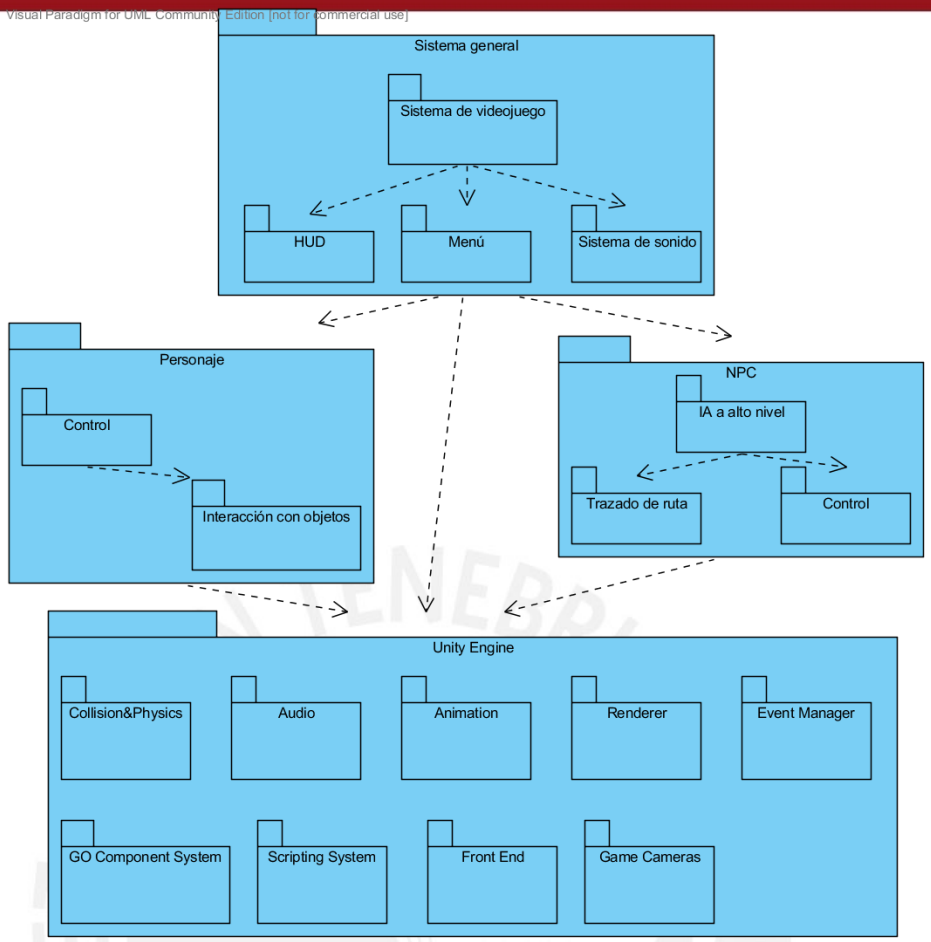


Figura 3.2. Vista lógica de la aplicación

3.1.4 Vista de despliegue

La vista de despliegue mostrada en Figura 3.3 diagrama cuál será la distribución de los componentes de la aplicación en el ambiente de producción. Dicho esquema es en realidad bastante sencillo, debido a que, en resumen, el navegador web descarga el compilado del videojuego del servidor web y lo muestra en la página web donde se encuentra embebido.

- **Navegador web:** El navegador web se ejecuta en el equipo del cliente, el cual hace uso del *plugin* del motor de videojuegos para mostrar, embebido dentro de la página, el videojuego.
- **Servidor web:** El servidor web almacena dos artefactos, uno de ellos es la página web, la cual puede ser un simple archivo plano html, y el compilado del videojuego, el cual será descargado a partir de la petición de algún cliente.

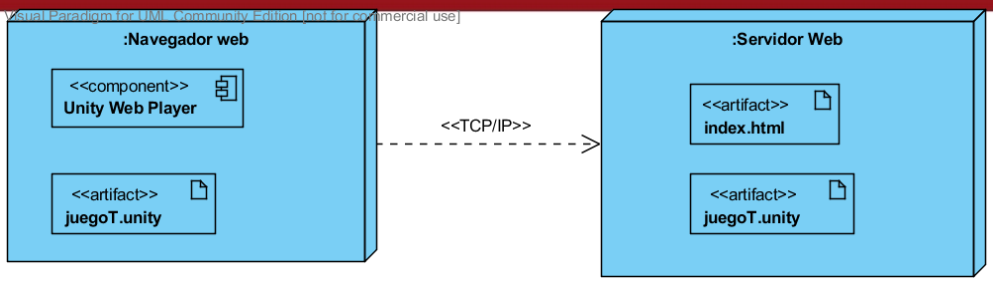


Figura 3.3. Vista de despliegue de la aplicación

3.2 Diseño de Inteligencia Artificial

Una parte importante de este proyecto consiste en el diseño de la IA, parte de la misma consiste en desarrollar el comportamiento de los *NPCs* y la otra en implementar el algoritmo de trazado de rutas A*.

3.2.1 Estrategias de diseño de comportamiento de *NPCs*

El comportamiento de los *NPC* se diseñó utilizando los siguientes criterios:

- Que sea fácil de entender.
- Que sea fácil de implementar.
- Que permita modelar un conjunto reducido de comportamientos, debido a que solo se tienen dos grupos grandes de *NPCs*, de los cuales, el grupo de los enemigos se subdivide en dos cuyo comportamiento varía ligeramente.
- Que el nivel del comportamiento modelado sea medio o bajo, dado que el nivel de interacción con el personaje es limitado, dicho de otro modo, no requiere de gran complejidad.

3.2.1.1 Máquinas de estado finito (FSM)

Esta es una de las primeras técnicas para modelar el comportamiento de los *NPCs*, la cual se basa en el concepto tradicional de máquinas de estados, por tanto reconoce dicho lenguaje formal basado en grafos. En esencia, el *NPC* permanece en un estado hasta que un evento o acción condicionada lo impulsa a cambiar de estado y de tipo de comportamiento. Por otro lado, tiene como puntos a favor que es fácil de entender y de implementar —es posible usar directamente sentencias del tipo *if-then-else* para su implementación. Además de ello, existe una clara separación entre el diseño de la jugabilidad y la implementación de la misma, dicho de otro modo, el programador puede implementar fácilmente el comportamiento que el diseñador del videojuego modela. Sin embargo, el comportamiento que se puede modelar con esta estrategia es relativamente simple (Vassos 2012).

3.2.1.2 Árboles de comportamiento (BTs)

Los árboles de comportamiento se basan en un nivel más refinado de condiciones y estrategias. Estos utilizan el conjunto de tareas como estructura básica común, las cuales son ejecutadas de acuerdo a ciertas condiciones, y se diferencian de las *FSM,* debido a que agrupan secuencias de tareas dentro de una estructura de árbol y, si no se cumple alguna condición, estos pasan a la siguiente rama inmediata superior para continuar la ejecución (véase Anexo K) (Vassos 2012).

3.2.1.3 Planeamiento de acciones orientado a objetivos (GOAP)

El planeamiento de acciones orientado a objetivos utiliza una serie de acciones disponibles asociadas a un conjunto de objetivos que el *NPC* debe perseguir y que poseen precondiciones para ejecutarse. Esto permite administrar fácilmente un conjunto de comportamientos autogenerados muy grande, además de obtener diferentes comportamientos que satisfacen distinta condiciones bajo los mismos objetivos, sin embargo, esta técnica de modelamiento del comportamiento es difícil de entender y de implementar (véase Anexo L) (Vassos 2012).

3.2.1.4 Cuadro comparativo de estrategias de diseño de comportamiento

Finalmente, basado en los criterios mencionados, los cuales se aprecian en la Tabla 3.2, y teniendo en cuenta los objetivos planteados, se decidió que la estrategia a usar será las *FSM*, debido a que satisfacen los criterios mencionados al inicio de esta sección.

Criterio	Máquinas de estado finito (*FSM*)	Árboles de comportamiento (*BTs*)	Planeamiento de acciones orientado a objetivos (*GOAP*)
Utilización en otros videojuegos	Ampliamente usado	Desde 2004	Desde 2005
Rendimiento	Alto	Alto	Medio
Nivel de dificultad para entender	Bajo	Bajo	Alto
Nivel de dificultad para implementar	Bajo	Medio	Alto
Nivel de separación entre diseño y programación	Medio	Medio	Bajo
Nivel de complejidad del comportamiento modelado	Medio	Medio	Alto
Permite extender el	Sí	Sí	No

Continúa.

Continuación.

Criterio	Máquinas de estado finito (*FSM*)	Árboles de comportamiento (*BTs*)	Planeamiento de acciones orientado a objetivos (*GOAP*)
comportamiento			
Puede recibir información extra del escenario	Sí	Sí	No
Permite manejar grandes cantidades de comportamientos	No	No	Sí
Tipo de técnica	Reactivo	Reactivo	Clásico
Videojuegos que lo implementan	*Half Life 2*	*Halo 2, Grand Theft Auto IV*	*FEAR*

Tabla 3.2. Cuadro comparativo entre las estrategias para el modelado del comportamiento de los NPCs

3.2.2 Diagrama de máquina de estados finitos para *NPCs*

En la Figura 3.4 se puede apreciar el diagrama de estados de uno de los *NPCs* que se implementó en la aplicación. En él se aprecia el modelado del comportamiento definido en el *game design*. El estado inicial de este *NPC* es *Patrulla,* mientras el *NPC* no encuentre un objetivo dentro de la línea de vista seguirá en este estado, en caso que este encuentre un objetivo dentro de su línea de vista, deberá validar si es de tipo civil —el personaje principal es de tipo civil— o de tipo militar, según el tipo del objetivo cambiará su estado a *Ataca,* en el primer caso, o a *Escapa*, en el segundo. Esto se debe a que este enemigo es del tipo 1, el cual posee armas blancas, y no puede enfrentarse a los militares con armas de guerra. Luego, si el objetivo es de tipo civil y se encuentra dentro del rango de ataque, el *NPC* cambia su estado a *Ataca* y esto activa la acción de atacar. Por último, mientras ataca, puede recibir daño de parte de su objetivo, por ello continuamente verificará si su nivel de salud es mayor al mínimo permitido, en caso contrario, deberá escapar; sin embargo, si el nivel llegar a ser menor que cero, el *NPC* cambia su estado a *Muere* y finalmente es eliminado del sistema de objetos de juego. Véase los Anexos H e I para ver las demás máquinas de estados.

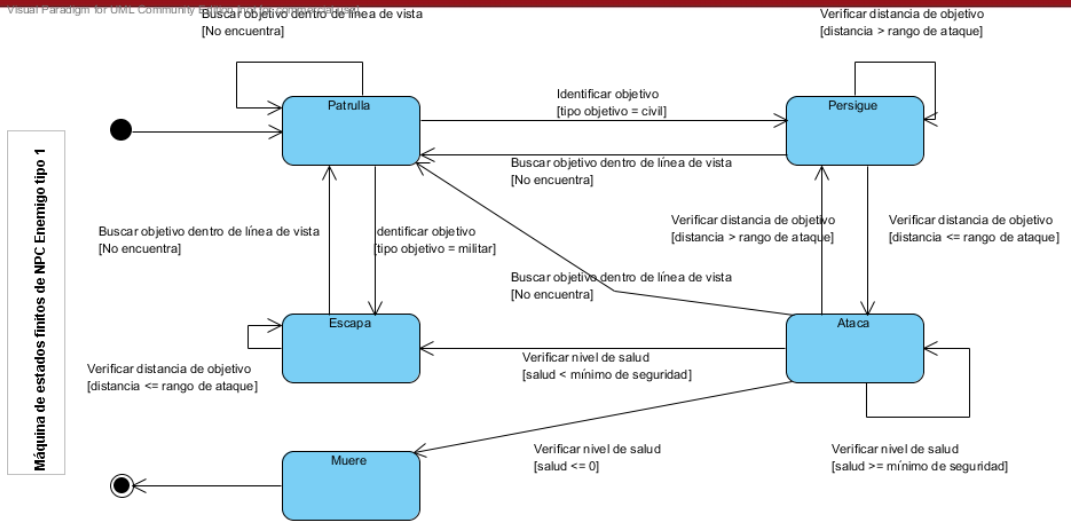


Figura 3.4. Máquina de estados finitos del *NPC* Enemigo tipo 1

3.2.3 Forma de representación del área navegable

El algoritmo seleccionado para el trazado de rutas es la elección por excelencia para los videojuegos: el algoritmo A* (Cui 2011: 126).Por ello, no se realiza ningún tipo de experimentación numérica para contrastarlo con algún otro. Sin embargo, la forma de representación del mapa sí es un factor crítico al momento de implementar el algoritmo, debido a que afecta al rendimiento del algoritmo y a la calidad de la ruta encontrada.

3.2.3.1 Grilla

La grilla es una subdivisión uniforme del mundo del juego en formas regulares más pequeñas llamadas celdas. La forma de las celdas puede ser variada, cuadrados, triángulos y hexágonos, estas tienen la ventaja de que son fáciles de entender, pero la gran desventaja de que involucran tiempo de procesamiento para cada pequeño movimiento que se quiera dar (Patel 2006).

3.2.3.2 Grafo de Nodos

Los grafos de nodos son una alternativa común a las grillas, los cuales utilizan una representación poligonal. Dentro de un grafo de nodos es necesario que cada uno almacene información de los nodos con los cuales están conectados, el problema de esto es que agrega una gran complejidad cuando se tienen múltiples nodos conectados en simultáneo. Donde la cantidad de conexiones entre nodos puede llegar hasta n^2, donde n es la cantidad de nodos (Patel 2006).

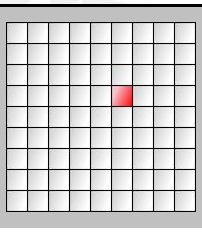
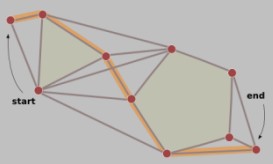
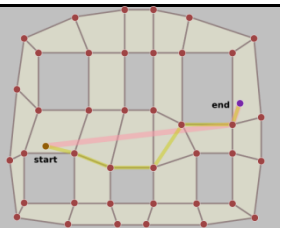
3.2.3.3 Malla de navegación

En lugar de representar los obstáculos con polígonos, es posible representar las áreas transitables con polígonos que no se superpongan. Dichas áreas pueden contener información adicional, por ejemplo, “requiere nadar” o “costo de movimiento 2”. Una de las ventajas de esta forma de representación es que los obstáculos no requieren ser almacenados, además la malla en sí misma puede servir para limitar el avance hacia abismos (Patel 2006).

3.2.3.4 Cuadro comparativo de formas de representar el área navegable

Luego de evaluar estas tres formas de representación del mapa, considerando que la superficie navegable del juego es muy amplia y que la cantidad de obstáculos es grande, debido a que el terreno posee gran cantidad de árboles y otros objetos, se decidió utilizar las mallas de navegación para representar la superficie navegable del mapa.

La forma de representar la malla es a través de polígonos triangulares de perímetros variables, ya que estos se adaptan al terreno navegable. Además, estos polígonos triangulares son representados por objetos que están compuestos por tres vértices. El movimiento, a través de estos polígonos triangulares, es suavizado al utilizar los puntos medios de sus aristas.

	Grilla	Nodos	Malla
Representación gráfica			
Estructura de datos	Arreglo bidimensional de nodos	Colección de nodos	Colección de polígonos compuestos de nodos
Identificación y conexión con nodos vecinos	Por ubicación en el arreglo	Referencia a los nodos con los que tiene línea de vista	Referencia a polígonos adyacentes
Dificultad de implementación	Fácil	Medio	Medio
Posibilidad de suavizar movimiento	No	No	Sí
Rendimiento en superficies	Bajo	Bajo	Medio

Continúa.

Continuación.

	Grilla	Nodos	Malla
grandes			
Desventaja	Ideal para áreas pequeñas	La cantidad de conexiones entre nodos adyacentes es de n^2	La cantidad de conexiones entre polígonos crece con la cantidad de lados del polígono

Tabla 3.3. Cuadro comparativo entre formas de representación del mapa de navegación

3.3 Diseño de interfaz gráfica

La interfaz gráfica del videojuego se compone de dos secciones importantes, una de ellas es el menú de opciones y la otra es el *HUD in game* o interfaz de información al usuario en la vista del videojuego.

3.3.1 Estándares de interfaz

Los estándares de interfaz en los videojuegos están definidos por la industria, por ello es una buena práctica utilizar características específicas de los videojuegos del mercado para diseñar las interfaces de un videojuego nuevo (Bethke 2003:222).

- Uno de los criterios aplicados es utilizar los indicadores del nivel de atributos relevantes al personaje —por ejemplo, salud, maná, etc.— en la parte baja de la pantalla. Algunos videojuegos de referencia son *Counter Strike, Half Life, Mu, Diablo,* etc
- Otro de los criterios aplicados es el uso de un mapa que muestra la ubicación del personaje en tiempo real. Dicho mapa suele ser circular, con una versión cartográfica del mapa en vez de una vista detallada, en varios juegos el mapa suele ser la vista de un radar, en este caso, debido al *gamedesign*, el mapa es una versión cartográfica del mismo con fondo transparente. Algunos videojuegos de referencia son *Metal Gear Solid, Call of Duty, Grand Theft Auto, World of Warcraft.*

3.3.2 Pantalla *HUD in game*

Sobre esos estándares se diseña la interfaz dentro del juego, la cual es minimalista (véase Figura 3.5). Esta contiene un indicador del nivel de ansiedad, el cual define la continuidad del jugador en el videojuego, además, se muestra un indicador del nivel de ruido y la posición en el mapa de estilo cartográfico. Por último, algunos indicadores extras se ubican en la base de la pantalla, como indicadores de estados de alerta y objetos recogidos.

Figura 3.5. Patrón de diseño del *HUD in game* del videojuego.

3.3.3 Menú de opciones

El menú de opciones es mostrado en dos oportunidades: al inicio del juego (ver Figura 3.6 Menú principal del videojuego.) y en el momento en que el usuario llama al menú pausando el videojuego. Al inicio del videojuego muestra una pantalla de inicio o carga antes de mostrar el menú principal.

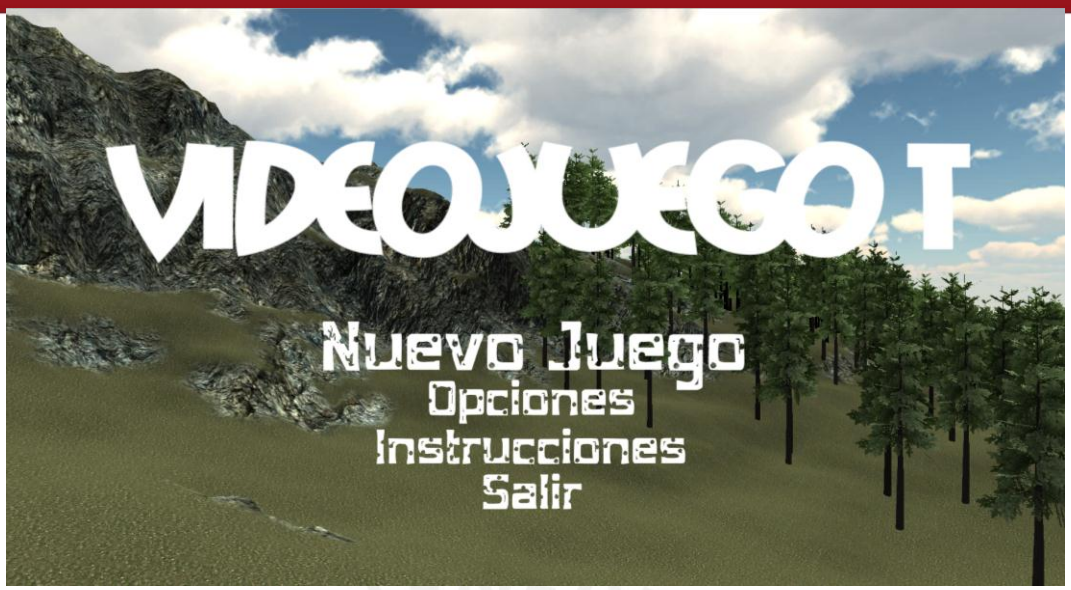


Figura 3.6 Menú principal del videojuego.

El menú principal muestra un conjunto de opciones que el usuario puede seleccionar, entre ellas puede cargar una nueva partida, ver las opciones de configuración, ver las instrucciones, etc. El esquema de navegación de este menú es el que se muestra en la Figura 3.7 Diagrama de flujo de navegación del menú de opciones.

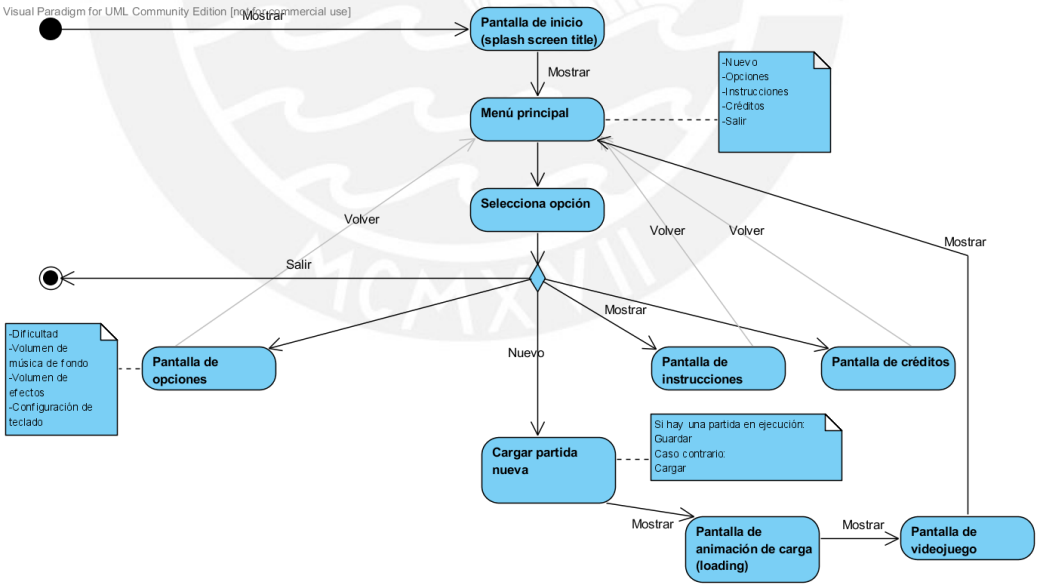


Figura 3.7 Diagrama de flujo de navegación del menú de opciones.

4 Construcción

Esta sección describe el proceso de construcción y pruebas los cuales pertenecen a las fases de producción y post-producción. En ella de describe el proceso de selección de las tecnologías usadas y las ventajas de brindan. Además se describe la estrategia de pruebas y los tipos de pruebas usadas en esta fase.

4.1 Construcción

En el desarrollo de videojuego, la construcción o fase de producción, involucra un conjunto de procesos de elaboración de los diferentes componentes que formarán parte de cada objeto de juego. En un proyecto como este, dichos componentes son los scripts de la capa de subsistemas específicos y los modelos 3D. Este documento da cobertura a la elaboración de los scripts de la capa de subsistemas específicos, a las tecnologías usadas para dar soporte a dicha capa y la tecnología usada para elaborar los componentes de dicha capa.

4.1.1 Motor de videojuego

Como se vio anteriormente, la arquitectura de un videojuego sitúa al motor en las capas más bajas y a la mecánica en la capa más alta (véase Anexo J). En otras palabras, para poder implementar la mecánica, se requiere toda la complejidad de un motor de propósito general de videojuegos.

Por ello, se realizó una comparación de algunos de los muchos motores existentes en el mercado, pre-seleccionados principalmente por su popularidad o su uso en algunos videojuegos con cierto reconocimiento. Se ha considerado las versiones libres de pago para todos ellos, con las limitaciones que ello implica, dicho en otros términos, no se considera las características de las versiones de pago sino de las gratuitas al momento de hacer la comparación. Dichos motores son:

4.1.1.1 HPL Engine 2

Es un motor de videojuegos 3D con licencia *GNU GPL,* el cual permite la creación e importación de modelos 3D, mapas 3D y materiales. Además, utiliza un sistema de componentes de objetos de juego para construir entidades las cuales se definen usando archivos XML (Página oficial Frictional Games 2012).

Además de ello, provee un editor de partículas con diseño personalizable y atributos que le otorgan movimiento, materiales y detectan colisiones. Por otro lado, posee su propio sistema de *scripts* con una sintaxis casi idéntica a C++ (Página oficial *Frictional Games* 2012).

Finalmente, permite realizar ajustes en el juego haciendo uso de archivos de configuración para registrar recursos e incluso materiales que son utilizados en los modelos 3D (Página oficial Frictional Games 2012).

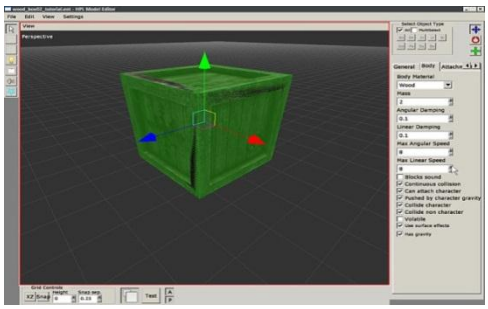


Figura 4.1. Editor gráfico de HPL Engine 2

4.1.1.2 id Tech 4

Es el motor del muy popular videojuego *Doom 3*, de licencia *GNU GPL*, utiliza *Open GL* como API gráfica, *Open AL* como API para el sistema de audio y posee un motor de física propio. Su editor gráfico de niveles es del tipo *WYSIWYP* —por las siglas en inglés *What You See Is What You Print*— el cual se puede apreciar en la Figura 4.2 (Página oficial de idTech 4 2012).

Además de ello, su sistema de *scripting* es de sintaxis similar a C++ y, aunque no posee un editor gráfico de *HUDs*, posee un sistema de *scripting* adicional específico para este fin el cual es lo suficiente potente como para embeber mini-juegos dentro del juego principal (Página oficial de idTech 4 2012).

Finalmente, tiene como desventajas que la tecnología empleada en este motor data del año 2004 y su interfaz gráfica no ofrece muchas herramientas visuales para el tratamiento de los modelos.

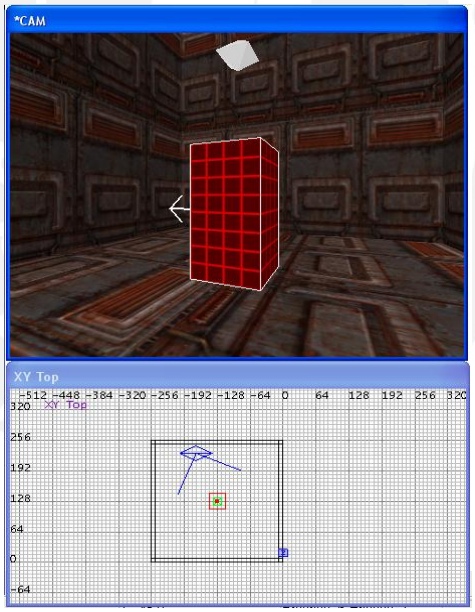


Figura 4.2. Editor gráfico de id Tech 4

4.1.1.3 Unreal Engine 3

Unreal Engine 3 es un motor de videojuego de clase AAA el cual posee muchos títulos de videojuegos muy reconocidos desarrollados con dicho motor. La calidad de los gráficos renderizados por dicho videojuego es de muy alto nivel, utiliza la API gráfica *DirectX* 11 y permite publicar videojuegos para los sistemas *Windows* e *iOS* en proyectos sin fines de lucro y, en su versión de pago, para *Mac OS X*, *Android*,

Adobe Flash Player y las consolas *PS3*, *Xbox 360* y *Wii U* (Página oficial de *Unreal Engine* 2012).

Utiliza el lenguaje de *scripting UnrealScript* con sintaxis similar a C++ y *Java*, el motor de física *PhysX de Nvidia*, además de su propio sistema de librerías de IA, *pathfinding*, control de cámaras, sistemas de partículas, entre otros. Además de ello, posee características avanzadas propias de un motor de su categoría como física para vestimenta que anima las prendas de vestir y telas que se incluyen en el escenario 3D, integración con código nativo —por ejemplo algunas librerías *DLL* en *Windows*—, etc. Finalmente, ofrece múltiples herramientas gráficas para la edición de niveles (véase Figura 4.3), modelado de personajes y objetos 3D haciendo uso de mayas, esqueletos, materiales, entre otros y un editor de *HUDs* (Página oficial de *Unreal Engine* 2012).

Sin embargo, no es posible embeber el videojuego en un navegador web, la forma de distribución es mediante descargas. Además de ello, por ser un motor de videojuegos de clase AAA utilizado para desarrollar enormes proyectos que requieren de equipos muy grandes de desarrollo, la cantidad de conocimiento requerida para realizar un proyecto es muy grande; por ello, recomiendan en su web la lectura de un manual de dos tomos de casi mil páginas cada uno para adquirir pericia en el uso de motor (Página oficial de *Unreal Engine* 2012).

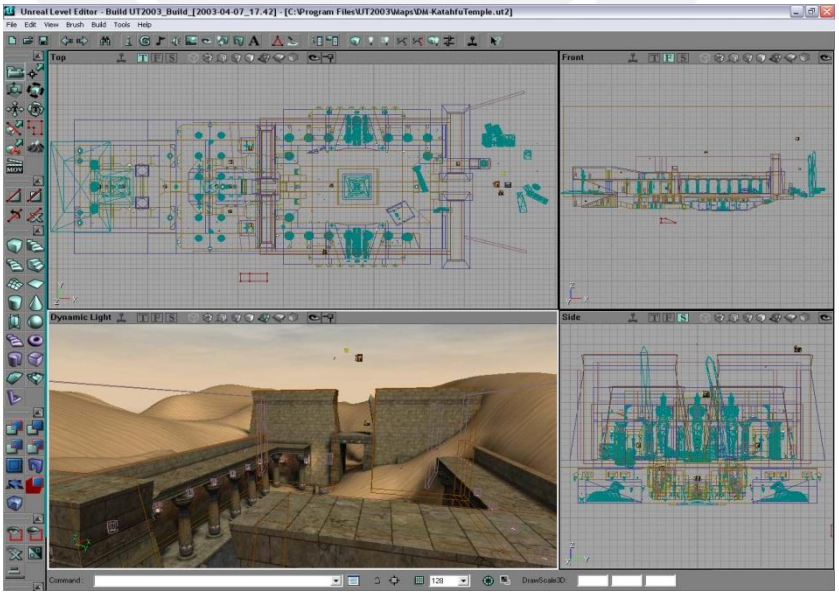


Figura 4.3. Editor gráfico de Unreal Engine 3

4.1.1.4 CryEngine 3

CryEngine 3 es otro motor de videojuego de clase AAA; es el motor del muy conocido videojuego *Crysis 2,* el cual sorprendió por su calidad gráfica la cual marcó un hito en los juegos para PC que se mantiene hasta hoy en día. Además, utiliza la *API* gráfica de *DirectX* 9,10 y 11, también posee un sistema de física propio y sus plataformas de publicación son *Windows* en su versión gratuita para desarrollos no comerciales y *Xbox 360, PlayStation 3, Wii U, iOS* y *Android* en su versión de pago (Página oficial de *Crytek* 2012).

La programación se puede realizar usando C++ para crear nuevas clases que heredan de las existentes usando la interface *IGame*; sin embargo se implementa la mecánica del juego, la IA, la definición de entidades y objetos de juego, usando el lenguaje Lua, este último es de tipo *scripting* y, además, se puede integrar con C++. Además, ofrece librerías de IA, *pathfinding*, control de cámara, sistemas de partículas, audio 3D, física para animación de vestimenta, entre otras características. Por último, también ofrece un conjunto de herramientas de apoyo visuales como un editor de niveles, herramientas de modelado 3D y un editor de terreno (Página oficial de Crytek 2012).

Sin embargo, al igual que *Unreal Engine 3*, no es posible embeber el videojuego en un navegador web. Además, este motor presenta una curva de aprendizaje muy alta. (Página oficial de Crytek 2012).

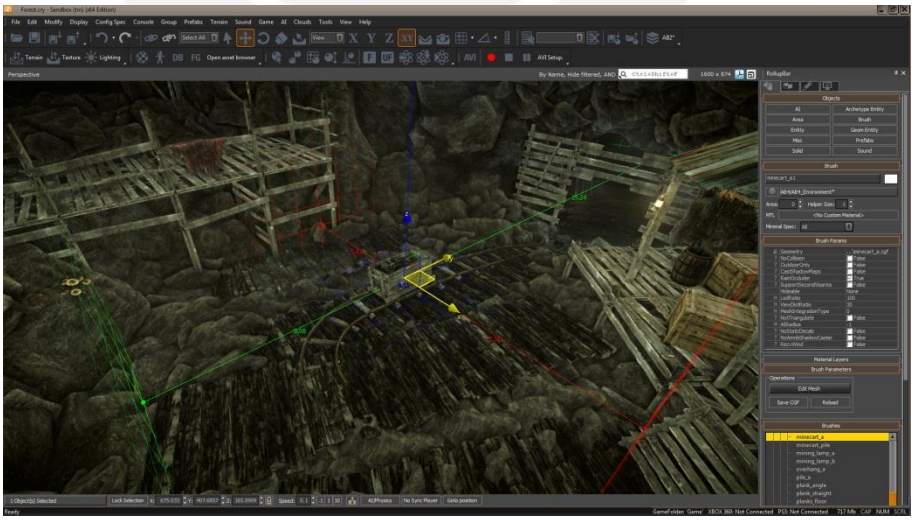


Figura 4.4. Editor gráfico de CryEngine 3

4.1.1.5 Unity 3D

Unity es un motor de videojuegos que lanzó al mercado su versión 1.0 beta el 6 de junio de 2005 para la plataforma *Mac OS X* e inmediatamente después para *iOS*, características que lo posicionaban como una alternativa para el desarrollo de videojuegos orientados al muy popular *iPhone* que estaba en pleno auge. Además de ello, construyeron el producto basándose en una filosofía de democratización dando la oportunidad de desarrollar videojuegos a un conjunto menos exclusivo de desarrolladores y artistas con un producto robusto y simple de usar (Página oficial de Unity 2012).

La versión actual de motor es el 4.0.1, el cual ofrece las siguientes características: gráficas de calidad AAA haciendo uso de *DirectX 11* y *Open GL*, con un renderizador muy potente que no tiene mucho que envidiar a los motores anteriormente mencionados. Capacidad de publicar en múltiples plataformas: *Windows, Mac OS X* y *Ubuntu* en su versión gratuita —con varias limitaciones respecto a la versión de pago que no restan funcionalidad al producto— también, en esta misma versión ofrece la libertad de usarse en desarrollos comerciales y, en su versión de pago se puede publicar para *Xbox 360, PlayStation 3, Wii U, Adobe Flash Player, iOS, Android* y navegadores web —haciendo uso de un *webplayer*, de descarga gratuita, compatible con *Internet Explorer, Safari, Chrome* y *Firefox*— con varias opciones de licenciamiento (Página oficial de Unity 2012).

Unity está totalmente orientado al concepto de *GOCS* —sistema de componentes de objetos de juego—, por tanto los objetos de juego están conformados de diferentes componentes incluyendo archivos de *script* que agregan la mecánica y lógica al juego. Esta arquitectura permite que se pueda elaborar y ejecutar prototipos muy rápidamente y, considerando que los *scripts* pueden ser escritos en lenguaje *JavaScript, C#* y *Boo* —cuya sintaxis es similar a *Python*—, la curva de aprendizaje se puede recorrer con cierta rapidez (Página oficial de Unity 2012).

Finalmente, ofrece un conjunto librerías que pueden ser utilizadas para elaborar productos con mucha rapidez como el motor de física *PhysX*, controles de cámara en primera y tercera persona, física *ragdoll*, física para vestimenta, sistemas de partículas, arquitectura cliente-servidor de carga distribuible, etc., también, otorga herramientas de apoyo gráfico muy poderosas, como un editor de niveles del tipo

WYSYWYP (véase Figura 4.5*)*, importación de modelos, editor de terrenos, herramientas de modelado y texturizado, etc. (Página oficial de Unity 2012).

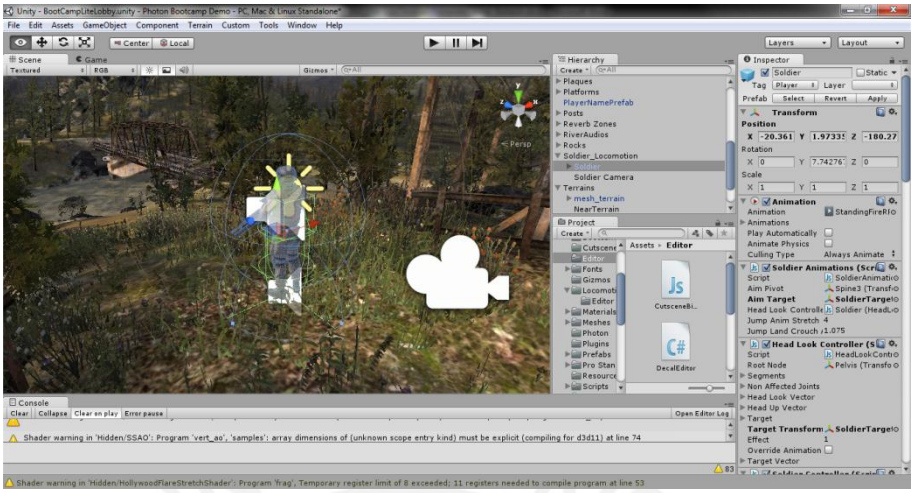


Figura 4.5. Editor gráfico de Unity 3D

4.1.1.6 Cuadro comparativo de motores de videojuego 3D

En el capítulo 1 se evaluaron las características cualitativas de cada de cada uno de los motores de videojuego presentados. Finalmente, en esta sección se elabora un cuadro comparativo donde se evalúa las características cuantitativas para asignarle un puntaje a cada motor. Esta comparación se realizó teniendo en cuenta los objetivos de este proyecto en particular y asignándole un valor cuantitativo a los atributos de acuerdo a su relevancia para el proyecto. Se le dio principal atención a las características que facilitaban la integración de los modelos 3D, el diseño del nivel, la ubicación de fuentes de audio y principalmente la publicación para navegadores web.

En la Tabla 4.1 se muestra el cuadro comparativo teniendo en cuenta el valor o peso en función a los objetivos del proyecto según los siguientes criterios:

- Indiferente: 1
- Valor agregado: 2
- Necesario: 3
- Muy necesario: 4

Motores en versión libre de cargo	***HPL Engine 2***	***idTech 4***	***Unreal Engine 3 (UDK)***	***CryEngine 3 SDK***	***Unity 3D***	*Valor cuantitativo*
Lenguaje de scripting	C++	C++	*UnrealScript*	Lua language, C++	C#, JavaScript, Boo	--

Continúa.

Continuación.

Motores en versión libre de cargo	*HPL Engine 2*	*idTech 4*	*Unreal Engine 3 (UDK)*	*CryEngine 3 SDK*	*Unity 3D*	*Valor cuantitativo*
Plataformas de publicación	*Windows, Mac OS X, Linux*	*Windows, Mac OS X, Linux*	*Windows, iOS*	*Windows*	*Windows, Mac OS X, Linux, Unity Player Web*	--
Licencia de software	Propietaria/ GNU GPL	GNU GPL	Propietaria	Propietaria	Propietaria	--
API Gráfica	*OpenGL*	*OpenGL*	*DX 9, 11*	*DX 9, 10, 11*	*DX 9, DX 11, OpenGL*	--
Física	*Newton's physics code*	Sistema Propio	*PhysX*	Sistema Propio	*PhysX*	--
Pathfinding	Sí	Sí	Sí	Sí	No	*1*
Importación multiformato de modelos 3D	No	No	Sí	No	Sí	*3*
Control de cámara	Sí	Sí	Sí	Sí	Sí	*3*
Audio 3D	Sí	Sí	Sí	Sí	Sí	*2*
Sistema de partículas	Sí	Sí	Sí	Sí	Sí	*2*
Integración con navegadores web	No	No	No	No	Sí	*4*
Física para vestimenta (Physical Clothing)	No	No	Sí	Sí	Sí	*2*
Física de muñeca de trapo (Rag dolls)	Sí	Sí	Sí	Sí	Sí	*2*
Editor gráfico integrado	Sí	Sí	Sí	Sí	Sí	*3*
Editor gráfico WYSIWYP	Sí	Sí	Sí	Sí	Sí	*3*
Editor de HUD	No	No	Sí	No	No	*2*
Editor de terreno	No	No	Sí	Sí	Sí	*2*
Videojuegos representativos	Penumbra, Amnesia: *The Dark Descent*	*Doom 3, Quake 4, Wolfenstein*	*Gears of War, Infinity Blade, America's Army 3*	*Crysis 2*	*Bad Piggies, Global Conflicts: Palestine, UberStrike, Temple Run: Brave*	--
Calificación cuantitativa	**16**	**16**	**25**	**20**	**26**	

Tabla 4.1. Comparación de motores 3D de videojuegos

Se puede apreciar de este cuadro que, el motor *Unity* 3D obtiene un puntaje ligeramente superior al motor *Unreal Engine 3*. Esta estrecha diferencia se puede expandir si consideramos la evaluación cualitativa y la curva de aprendizaje más plana que ofrece *Unity.* Por ello, el motor que se usa en proyecto es *Unity* 3D.

4.1.2 Lenguaje de *scripting*

El sistema de componentes de objetos de juego provee al motor la capacidad de realizar cambios en la funcionalidad en tiempo de ejecución. Esto es posible gracias al sistema de *scripting*.

Unity ofrece tres tipos de lenguajes de script, ellos son *C#*, *Java Script* y *Boo*, cada uno de ellos tiene el mismo objetivo, sin embargo presentan sutiles diferencias las cuales son analizadas a continuación en la Tabla 4.2.

	C#	***JavaScript (UnityScript)***	***Boo (Python)***
Paradigma	Imperativo, orientado a objetos, funcional, genérico, reflexivo	Imperativo, orientado a objetos, funcional, reflexivo	Imperativo, orientado a objetos, funcional, genérico, reflexivo
Carga en la sintaxis	Alta	Baja	Media
Tipificación	Fuerte	Débil	Fuerte
Verificación de tipos	Estático con verificación dinámica opcional	Estático con verificación dinámica opcional	Estático con verificación dinámica opcional
Seguridad de tipos	Alta	Baja (Directiva "#pragma strict" cambia a Alta)	Alta
Ejemplos en la documentación oficial de *Unity*	Media	Alta	Baja
***IDEs* que *Unity* permite vincular**	*MonoDevelop-Unity, Visual Studio, Visual Studio Express C#*	*MonoDevelop-Unity*	*MonoDevelop-Unity*
Experiencia del tesista	Media	Baja	Nula

Tabla 4.2. Cuadro comparativo entre los lenguajes de scripting.

Dado estos resultado, se selecciona a C# como lenguaje de scripting para todos los componentes de comportamiento.

4.1.3 Entorno de desarrollo

Dentro de Unity es posible utilizar dos *IDEs* de desarrollo, *MonoDevelop* y *Visual Studio*. A continuación se realiza una comparación entre ambas opciones (véase la Tabla 4.3), mostrando como el más adecuado a *MonoDevelop* debido a su integración y capacidad de depuración.

	MonoDevelop-Unity	***Visual Studio* Express C#**	***Visual Studio***
Nivel de integración con *Unity*	Alta	Baja	Baja
Permite depuración	Sí	No	No
Autocompletado (*Intellisense*)	No	Sí	Sí
Capacidad de identificación de errores de sintaxis	Media	Alta	Alta
Herramientas de	Nula	Baja	Alta

Continúa.

Continuación.

	MonoDevelop-Unity	***Visual Studio* Express C#**	***Visual Studio***
refactorización			
Cantidad de características extra del editor	Baja	Media	Alta
Disponibilidad de *plugins*	Baja	Baja	Alta
Licencia	*OpenSource*	*Freeware*	*Comercial*

Tabla 4.3. Cuadro comparativo entre entornos de desarrollo

4.2 Pruebas

En esta sección se describe la estrategia de pruebas a utilizada para asegurar el correcto funcionamiento de la aplicación. También se justifica su elección y se incluye parte del catálogo de pruebas para mostrar la forma en que fue utilizada.

4.2.1 Estrategia de pruebas

La estrategia de pruebas es utilizada para demostrar que la aplicación presenta un funcionamiento completo en términos de eficiencia y eficacia. Dicha estrategia es ejecutada desde adentro hacia afuera, en otras palabras, primero se verifica el funcionamiento de los componentes atómicos o unitarios de la aplicación para luego llegar a efectuar pruebas de caja negra (*véase Tabla 4.4*) con la finalidad de buscar fallos al momento de ejecutar la aplicación y definir los requisitos mínimos y recomendados de la misma. Finalmente, en la fase de post-producción se realizan pruebas beta con jugadores voluntarios en busca de registrar fallos que no pudieran haberse percibido en la diversidad de equipos y configuraciones que los usuarios tienen.

4.2.2 Tipos de pruebas

Los tipos de prueba a ejecutar son los siguientes:

4.2.2.1 Pruebas unitarias y pruebas de caja blanca

Las pruebas unitarias son el tipo de procedimiento de pruebas más sencillo de utilizar. Consiste en escribir un conjunto de pruebas para llevar a la pieza de software a lo largo de todos los rangos de valores de entrada válidos y los no válidos (Bethke 2003: 154-155). Con este enfoque, cada componente dentro de los paquetes es evaluado con un conjunto de eventos de entrada posible.

4.2.2.2 Pruebas de caja negra

Otro tipo de pruebas son las de caja negra. Este es el tipo de prueba que comúnmente se llevan a cabo en los videojuegos. Para esto se elabora una lista de verificación de referencia y se pide a un grupo de voluntarios ajenos al código, que prueben diversos escenarios siguiendo la lista de verificación y con la libertad de obviarla para realizar verificaciones menos estructuradas (Bethke 2003: 154).

4.2.2.3 Pruebas beta

Finalmente, en la fase de post-producción se lleva a cabo un conjunto de pruebas sobre el producto totalmente desarrollado e integrado. El conjunto de voluntarios para esta prueba beta es un subconjunto del público objetivo, y dichas pruebas se realizan fuera del entorno de desarrollo, en otras palabras, en el ambiente donde finalmente se ejecutará el código (Bethke 2003: 154). Para este tipo de pruebas se elabora un formulario web que permite a los jugadores registrar toda falla detectada de forma detallada.

4.2.3 Lista de verificación de caja negra

A continuación se muestra la lista de verificación utilizada para las pruebas de caja negra y sus resultados.

Id	Paquete	Verificar si…	Resultado
1	Control de personaje	El personaje puede caminar sobre cualquier terreno transitable.	SÍ
2	Control de personaje	El personaje puede correr sobre cualquier terreno transitable.	SÍ
3	Control de personaje	El personaje puede caminar de puntillas para disminuir el nivel de ruido.	SÍ
4	Control de personaje	El personaje puede agacharse	SÍ
5	Control de personaje	El personaje puede avanzar mientras está agachado.	SÍ
6	Control de personaje	El personaje puede echarse sobre el suelo.	NO
7	Control de personaje	El personaje puede avanzar mientras está echado en el suelo.	NO
8	Control de personaje	El personaje puede atacar a su oponente.	SÍ
9	Interacción con objetos	El personaje puede abrir una puerta.	SÍ
10	Interacción con objetos	El personaje puede usar un interruptor.	SÍ
11	Interacción con objetos	El personaje puede recoger un ítem.	SÍ
12	Interacción con objetos	El personaje puede desechar un ítem.	SÍ
13	Interacción con objetos	El personaje puede usar un ítem.	SÍ
14	Interacción con objetos	El personaje puede usar un arma	SÍ

Continúa.

Continuación.

Id	Paquete	Verificar si…	Resultado
15	Inteligencia Artificial	El personaje puede recibir información en forma de texto de parte de un NPC	NO
16	Inteligencia Artificial	El personaje puede ser perseguido por un NPC	SÍ
17	Inteligencia Artificial	El personaje puede ser perseguido y atacado por un NPC	SÍ
18	Inteligencia Artificial	El personaje puede ser evadido por un NPC	SÍ
19	HUD	El incremento y decremento del nivel de ansiedad	SÍ
20	HUD	El incremento y decremento del nivel de ruido	SÍ
21	HUD	La actualización de la posición en mini mapa	SÍ
22	Sistema de sonido	Se puede escuchar música de fondo	SÍ
23	Sistema de sonido	Se puede escuchar efectos de sonido	SÍ
24	Sistema de sonido	Se puede regular el volumen de la música de fondo	SÍ
25	Sistema de sonido	Se puede regular el volumen de los efectos de sonido	SÍ
26	Menú	Se muestra la pantalla de menú	SÍ
27	Menú	Se puede navegar por las opciones de menú	SÍ
28	Menú	Se muestra la pantalla de carga de partida nueva	SÍ
30	Menú	Se muestra la pantalla de dificultad del videojuego	SÍ
31	Menú	Se muestra la pantalla de configuración de teclado	SÍ
32	Menú	Se muestra la pantalla de configuración de audio	SÍ
33	Menú	Se muestra una animación para la carga de videojuego	SÍ
34	Menú	Se muestra una animación para la carga de partida	SÍ
35	Sistema	Se puede cargar nueva partida	SÍ
38	Sistema	Se puede abandonar una partida	SÍ
39	Sistema	Se puede reproducir cinemática	SÍ
40	Sistema	Se puede configurar controles de entrada	SÍ
41	Sistema	Se puede configurar nivel de dificultad del videojuego	NO
42	No aplica	La aplicación se ejecuta en un navegador web	SÍ

Tabla 4.4. Lista de verificación de caja negra

5 Observaciones, conclusiones y recomendaciones

Este capítulo finaliza el presente documento con las observaciones realizadas al proyecto, las conclusiones extraídas del mismo y, finalmente, realiza un conjunto de recomendaciones a tomar en cuenta en la elaboración de proyectos similares.

5.1 Observaciones

Este proyecto fue concebido con la finalidad de elaborar una herramienta que pueda cubrir una necesidad social —dicha necesidad se discute en la descripción del problema; para ello se diseñó y construyó un videojuego que sirva como canal alternativo para la difusión de cierto contenido.

Uno de los aspectos importantes es la construcción de un videojuego es el *game design*, este proceso de diseño del juego es altamente creativo y es crítico para asegurar el éxito del proyecto. Este proceso tiene como punto de partida el desarrollo del concepto del videojuego y, una vez definido y aceptado, es diseñado en gran parte en la fase de pre-producción y su elaboración termina junto con el proyecto. Sin embargo, este proceso, por las razones previamente mencionadas, no es cubierto en el presente documento.

En este proyecto, la toma de requerimientos tiene lugar en el proceso creativo de diseño del diseñador del juego el que define finalmente los requisitos del mismo. Como todo proceso de esta naturaleza, es totalmente iterativo e incremental y susceptible a cambios y aportes de los diferentes miembros de un equipo de producción de juegos. Esta dinámica de elaboración del diseño permitió que el tesista pueda guiar este proceso valiéndose de otros estudiantes, profesionales y jugadores, para lograr elaborar el diseño final que se implementa, luego, en el producto.

El uso de una adaptación de *RUP*, valiéndose de las herramientas de *UML*, permitió un proceso de desarrollo bastante metódico y ordenado, además, brindó una visión bastante amplia, desde diferentes enfoques del producto, lo cual incrementó la calidad del mismo. Por otro lado, las herramientas de *PMI*, permitieron definir claramente cuáles eran los entregables en este proyecto y dentro de qué intervalos de tiempo se irían desarrollando.

El uso del motor de videojuegos *Unity 3D*, permitió que el proceso de desarrollo sea mucho más ágil, práctico, e intuitivo —una vez comprendido el sistema de componentes de objetos de juego. Además, el enfoque *WYSWYP* de *Unity,* facilitó considerablemente la integración con los modelos 3D para los personajes, objetos y terreno.

Esta aplicación es un videojuego serio, cuya finalidad, más allá de entretener al jugador, es llevarlo a la reflexión sobre un tema sensible para nuestra sociedad. Por ello, el diseño del mismo contiene elementos de juego que son atípicos, como el nivel de ansiedad, la restricción a usar armas y la posibilidad de que el personaje muera al primer disparo, etc.

5.2 Conclusiones

- Se cumple el objetivo de implementar la mecánica e inteligencia artificial de un videojuego 3D el cual se ejecuta sobre una plataforma web, haciendo uso de un *plugin* que permite embeber el videojuego en una página web.

- El esfuerzo y tiempo invertido en la investigación bibliográfica, tanto en la definición de conceptos, como en la elaboración del estado del arte

permitieron enriquecer los factores que se utilizaron en la toma de decisiones de las fases de pre-producción y producción.

- La elección de las máquinas de estado finito para el modelado del comportamiento de los *NPCs* facilitó notablemente la implementación del mismo y produjo un nivel de comportamiento bastante satisfactorio. Además de esto, el rendimiento del algoritmo fue bastante bueno.

- La implementación del algoritmo A* usando mallas de navegación fue relativamente difícil implementar, sin embargo, este nivel de dificultad fue compensado con el nivel de rendimiento del mismo el cual fue satisfactorio.

- El uso de la arquitectura de componentes de objetos de juego fue una excelente decisión, debido a que permitió el aislamiento entre las capas del motor y las del subsistema específico. Además de ello, fue posible modelar el comportamiento en componentes aislados del modelo, de la física, de las animaciones, etc. Finalmente, la integración con dichos componentes fue mucho más rápida al remplazar los modelos de prueba con los modelos finales provistos por el artista.

- El uso de algunos procesos de la metodología RUP junto con algunas herramientas de UML fue muy útil por la amplia visión que ofrece del sistema desde la fase de análisis y diseño. Sin embargo, dificulta la implementación de cambios, en fases avanzadas del diseño o implementación.

- Este proyecto es económica y técnicamente viable debido al uso de herramientas libres de pago, tanto en las fases de pre-producción haciendo uso de herramientas *case* de uso libre, como en las fases de producción y post-producción, haciendo uso de la versión libre del motor *Unity* y de un *IDE* de la misma naturaleza.

5.3 Recomendaciones y trabajo futuro

Se recomienda utilizar el presente trabajo para elaborar una cantidad más amplia de niveles —este proyecto cuenta con uno—, con la finalidad de extender el tiempo en el que los jugadores estarán expuestos al videojuego y, por ende, al mensaje

que este transmite. Además, junto con esta expansión de niveles, se recomienda incluir las funcionalidades de carga y guardado de partida, las cuales no fueron incorporadas debido a lo reducido del tiempo de juego.

Se recomienda fuertemente, la inclusión de un tutorial interactivo de juego, el cual guie al jugador, a lo largo de un nivel de prueba, a conocer todas las posibilidades que el *gameplay* ofrece.

Como trabajo futuro, se puede incluir una cantidad más amplia de *NPCs* y, junto con ello, una cantidad mucho más grande de comportamientos distintos. Para ello, sería bueno considerar el uso de *BTs* junto con *GOAP* para modelar el comportamiento de los mismos.

Además de ello, es posible ampliar la mecánica del mismo, permitiendo, entre otras funcionalidades, poder nadar, utilizar vehículos, entre otras mecánicas que se alineen con la principal.

Esta solución presenta un gran potencial a futuro, el alcance de este videojuego ha sido corto dadas las restricciones de recursos y tiempo que un proyecto de fin de carrera establecen. Sin embargo, el producto puede expandirse hasta alcanzar algunos estándares de juego del mercado, como por ejemplo, un tiempo de juego mínimo de 6 horas, la posibilidad de ejecutarse en sistemas móviles como *iOS* o *Android* —con los cambios en la mecánica que esto implica, como los controles en pantalla, entre otros— y finalmente el incrementar la calidad de los modelos 3D y el texturizado de los mismos, lo cual permita una calidad visual mucho mayor a la obtenida.

6 Bibliografía

AARSETH, Espen

2001 "Computer Game Studies, Year One". *Game Studies*. 2001, volumen 1. Consulta: 7 de octubre de 2012. <http://www.gamestudies.org/0101/editorial.html>

ALTERNATIVE REALITY GAMING NETWORK (ARGNET)

2008 "Traces of Hope: British Red Cross Launches ARG for Civilians and Conflict Month". *Alternative Reality Gaming Network.* 29 de septiembre. Consulta: 7 de octubre de 2012. <http://www.argn.com/2008/09/traces_of_hope_british_red_cross_launches_arg_for_civilians_and_conflict_month/>

APPERLEY, Thomas H.

2006 "Genre and game studies: Toward a critical approach to video game genres". *Simulation Gaming*. March 2006, vol. 37, núm. 1, pp.6-23

ASUNCION, Hazeline y otros

2011 "Serious Game Development as an Iterative User-Centered Agile Software Project". Consulta: 08 de octubre de 2011. <http://www.ceng.metu.edu.tr/~tcan/se542_s1011/Schedule/reading4.pdf>

BECKER, Katrin

2010 "Distinctions Between Games and Learning: A Review of Current Literature on Games in Education". En VAN ECK, Richard. *Gaming and cognition: theories and practice from the learning sciences.* Pennsylvania:IGI Global snippet, pp. 22-51.

BETHKE, Erik

2003 *Game Development and Production.* Plano: Wordware Publishing."MDA: A formal approach to game design and game research". *Proceedings of the Challenges in Game AI Workshop, 19th National Conference on Artificial Intelligence* . Consulta: 30 de septiembre de 2011. <http://www.aaai.org/Papers/Workshops/2004/WS-04-04/WS04-04-001.pdf>

BILAS, Scott

2002 *A Data-Driven Game Object System* [diapositivas]. GDC 2002. Consulta: 7 de octubre de 2012.

<http://scottbilas.com/files/2002/gdc_san_jose/game_objects_slides.pdf>

BRATHWAITE, Brenda y Ian SCHREIBER

2009 *Challenges for Game Designers.* Boston: Course Technology.

BRINGUÉ, Xavier y Charo SÁDABA

2008 *La generación interactiva en Iberoamérica. Niños y adolescentes ante las pantallas.* Barcelona: Colección Fundación Telefónica, Ariel.Consulta: 14 de octubre de 2012. <http://dspace.unav.es/dspace/bitstream/10171/7307/1/GeneracionInteractivaIberoamerica2008.pdf>

BROWN, Harry J.

2008 *Videogames and education.* Nueva York:M.E. Sharpe.

CARNEGIE MELLON NEWS

2005 "Carnegie Mellon's Entertainment Technology Center Develops PeaceMaker, a Videogame Simulation To Encourage Peace in the Middle East". *Carnegie Mellon News.* Pittsburgh, 27 de octubre. Consulta: 7 de octubre de 2012. <http://gestion.pe/noticia/1372582/peru-tercera-economia-emergente-mas-prometedora-segun-boomberg>

COLES, Andrew y Amanda SMITH

2007 "Marvin: A Heuristic Search Planner with Online Macro-Action Learning"*Journal of Artificial Intelligence Research.* 2007, vol. 28, pp. 119-156. Consulta: 7 de octubre de 2012. <http://www.aaai.org/Papers/JAIR/Vol28/JAIR-2804.pdf?q=marvin-online>

COMISIÓN DE LA VERDAD Y RECONCILIACIÓN (CVR)

2003 *Informe Final, Tomo VIII, Conclusiones Generales.* Lima.Consulta: 14 de agosto de 2012. <http://www.cverdad.org.pe/ifinal/pdf/TOMO%20VIII/CONCLUSIONES%20GENERALES.pdf>

COMISIÓN DE LA VERDAD Y RECONCILIACIÓN (CVR)

2003 *Informe Final, Tomo II, Apéndice 1 Ejército Guerrillero Popular.* Lima. Consulta: 14 de agosto de 2012. <*http://www.cverdad.org.pe/ifinal/pdf/TOMO%20II/CAPITULO%201%20-%20Los%20actores%20armados%20del%20conflicto/1.1.%20PCP-SL/APENDICE%201%20EL%20EJERCITO%20GUERRILLERO%20POPULAR.pdf*>

CRYTEK

2012 Página oficial de Crytek. Consulta: 7 de octubre de 2012. <http://www.crytek.com/cryengine>

CUI, Xiao y Hao SHI

2011 "A*-based Pathfinding in Modern Computer Games". *IJCSNS International Journal of Computer Science and Network Security*. 2011, vol. 11, núm.1, pp. 125-130. Consulta: 7 de octubre de 2012. <http://paper.ijcsns.org/07_book/201101/20110119.pdf>"Serious Game Development as an Iterative User-Centered Agile Software Project". Consulta: 08 de octubre de 2011. <http://www.ceng.metu.edu.tr/~tcan/se542_s1011/Schedule/reading4.pdf>

DICKHEISER, Mike

2006 *Game Programming Gems 6.*Michigan: Charles River Media.

EL COMERCIO

2012a "El Perú entre economías emergentes con capacidad fiscal y monetaria" *El Comercio.* Lima, 31 de enero. Consulta: 7 de octubre de 2012. <http://elcomercio.pe/economia/1368225/noticia-peru-entre-economias-emergentes-capacidad-fiscal-monetaria>

2012b "Jóvenes limeños no tienen idea de lo que fue la violencia terrorista" *El Comercio.* Lima, 23 de enero. Consulta: 1 de septiembre de 2012. <http://elcomercio.pe/lima/1364558/noticia-jovenes-limenos-no-tienen-idea-lo-que-fue-violencia-terrorista>

2012c *VIDEOENCUESTA: jóvenes responden quién fue Abimael Guzmán* [videograbación]. Lima: El Comercio. Consulta: 15 de septiembre de 2012. <http://elcomercio.pe/actualidad/1468704/noticia-videoencuesta-jovenes-responden-quien-fue-abimael-guzman>

ESPOSITO, Nicolas

2005 "A Short and Simple Definition of What a Videogame Is". *DiGRA 2005 Conference: Changing Views – Worlds in Play.*Consulta: 02 de octubre de 2011. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.1409&rep=rep1&type=pdf>

FRICTIONAL GAMES

2012 Página oficial Frictional Games. Consulta: 7 de octubre de 2012. <http://www.frictionalgames.com/site/>

GAMES FOR CHANGE (GFC)

2003 "September 12th: A Toy World". *Games for Change.* 2003. Consulta: 7 de octubre de 2012. <http://www.gamesforchange.org/play/september-12th-a-toy-world/>

GAMES FOR CHANGE (GFC)

2008a "Traces of Hope". *Games for Change.*2008. Consulta: 7 de octubre de 2012. <http://www.gamesforchange.org/play/traces-of-hope/>

GAMES FOR CHANGE (GFC)

2008b "Global Conflicts". *Games for Change.*2008. Consulta: 7 de octubre de 2012. <http://www.gamesforchange.org/play/global-conflicts/>

GESTIÓN

2012a "El Perú es la tercera economía emergente más prometedora, según Bloomberg" *Gestión.* Lima, 10 de febrero. Consulta: 7 de octubre de 2012. <http://gestion.pe/noticia/1372582/peru-tercera-economia-emergente-mas-prometedora-segun-boomberg>

GLOBAL CONFLICTS

2012 "Global Conflicts". *Global Conflicts*. Consulta: 7 de octubre de 2012. <http://www.globalconflicts.eu/>

GREENFIELD, Patricia

2010 "Video Games Revisited". En VAN ECK, Richard. *Gaming and cognition: theories and practice from the learning sciences.* Pennsylvania:IGI Global snippet, pp. 1-21.*The Art of Game Design: A book of lenses.* Massachusetts: Morgan Kaufmann.

GREGORY, Jason

2009 *Game Engine Architecture.* Wellesley: Taylor & Francis Group

HANSON, Jeff

2005 "Event-driven services in SOA",*Javaworld.* Consulta: 7 de octubre de 2012. <http://www.javaworld.com/javaworld/jw-01-2005/jw-0131-soa.html>

HUNICKE, Robin, Marc LEBLANC y Robert ZUBEK

2004 "MDA: A formal approach to game design and game research".

Proceedings of the Challenges in Game AI Workshop, 19th National Conference on Artificial Intelligence . Consulta: 30 de septiembre de 2011.
<http://www.aaai.org/Papers/Workshops/2004/WS-04-04/WS04-04-001.pdf>

IDTECH 4
2012 Página oficial de idTech 4. Consulta: 7 de octubre de 2012.
<http://idtech4.com/home/>

INSTITUTO NACIONAL DE ESTADÍSTICA E INFORMÁTICA (INEI)
2012 Las Tecnologías de Información y Comunicación en los Hogares. Trimestre: Abril-Mayo-Junio, 2012. Lima. Consulta: 1 de septiembre de 2012.
<http://www.inei.gob.pe/web/Biblioinei/BoletinFlotante.asp?file=15083.pdf>

JEONG, Eui Jun, Frank A. BIOCCA y Corey J. BOHIL
2008 "Effects of Representational Realism in 3D Violent Games". En SPAGNOLLI, Anna y Luciano GAMBERINI. *Presence 2008.* Padova: CLEUP Cooperativa Libraria Universitaria Padova, 191-200. Consulta: 08 de octubre de 2012.
<http://www.temple.edu/ispr/prev_conferences/proceedings/2008/jeong.pdf>

KEITH, Clinton
2010 *Agile Game Development with Scrum*. Boston: Addison-Wesley Professional.

KNIBERG, Henrik
2007 *Scrum and XP from the Trenches:How we do Scrum.*USA: C4Media. Consulta: 08 de octubre de 2011.
<http://infoq.com/minibooks/scrum-xpfrom-the-trenches>

KRAMER, Wolfgang
2000 "What Is a Game?". *The Game Journal*. 2000. Consulta: 7 de octubre de 2012.
<http://www.thegamesjournal.com/articles/WhatIsaGame.shtml>

LIDÉN Lars y VALVE SOFTWARE
2002 "Strategic and Tactical Reasoning with Waypoints". En RABIN 2002: 210-220.

LEBLANC, Marc

2004 *Mechanics, Dynamics, Aesthetics: A Formal Approach to Game Design* [diapositivas].Northwestern University. Consulta: 02 de octubre de 2011. <http://algorithmancy.8kindsoffun.com/MDAnwu.ppt>

LETELIER, Patricio

2006 "Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)". *Técnica Administrativa.* Vol. 05, Núm. 26. Consulta: 08 de octubre de 2011. <http://www.cyta.com.ar/ta0502/v5n2a1.htm>

MARCANO, Beatriz

2008 "Juegos serios y entrenamiento en la sociedad digital". *Teoría de la Educación. Educación y Cultura en la Sociedad de la Información*.Salamanca, 2008, vol. 9, núm. 3, pp. 93-107. Consulta: 7 de octubre de 2012. <http://dspace.unav.es/dspace/bitstream/10171/7307/1/GeneracionInteractivaIberoamerica2008.pdf><http://redalyc.uaemex.mx/redalyc/pdf/2010/201017343006.pdf>

MARTÍNEZ, Alejandro y Raúl MARTÍNEZ

2002 "Guía a Rational Unified Process". Escuela Politécnica Superior de Albacete. Universidad de Castilla la Mancha. Consulta: 08 de octubre de 2012. <http://sites.google.com/site/softqma/programa/unidad-iv-metodologias-utilizadas-para-el-desarrollo-del-software/Trabajo-GuiaRUP.pdf>

MUNICIPALIDAD DE MIRAFLORES

2012 *Tarata 20 años: Los jóvenes sí tenemos memoria* [videograbación]. Lima: Academia de Ciudadanos Lideres de la Municipalidad de Miraflores. Consulta: 7 de octubre de 2012. <http://www.youtube.com/watch?v=tegCKsQNCz0>

ORKIN, Jeff y MONOLITH PRODUCTIONS

2002 "12 Tips from the Trenches". En RABIN 2002: 29-35.

PATEL, Amit

2009 "Pathfinding: Introduction to A*". *Red Blob Games.*Consulta: 7 de octubre de 2012. <http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html>

PROJECT MANAGEMENT INSTITUTE(PMI)

2009 Guía de los Fundamentos para la Dirección de Proyectos (Guía del PMBOK®) Cuarta edición. Project Management Institute.

RABIN, Steve

2002 *Ai Game Programming Wisdom.* City: Chars River Media.

RABIN, Steve

2010 *Introduction to Game Development.*Segunda Edición. Boston: Cengage Learning.

RATIONAL SOFTWARE

2001 "Rational Unified Process: Best Practices for Software Development Teams". *Rational Software.*
Consulta: 08 de octubre de 2012.
<http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf>

REAL ACADEMIA ESPAÑOLA (RAE)

2001 *DICCIONARIO DE LA LENGUA ESPAÑOLA - Vigésima segunda edición*. Madrid: Espasa.

SÁNCHEZ, Gloria y Víctor M. LOZANO

2006 "Algoritmo de Dijkstra: Un Tutorial Interactivo". *Actas de las XII Jornadas de Enseñanza universitaria de la Informática (Jenui 2006).*pp. 573-580.Consulta: 7 de octubre de 2012.
Consulta: 14 de agosto de 2012.
<http://www.cverdad.org.pe/ifinal/pdf/TOMO%20VIII/CONCLUSIONES%20GENERALES.pdf><http://bioinfo.uib.es/~joemiro/aenui/procJenui/ProcWeb/actas2001/saalg223.pdf>

SCHELL, Jesse

2008 *The Art of Game Design: A book of lenses.* Massachusetts: Morgan Kaufmann.

SCOTT, Bob y STAINLESS STEEL STUDIOS

2002 "The Illusion of Intelligence". En RABIN 2002: 16-20."Computer Game Studies, Year One". *Game Studies*. 2001, volumen 1. Consulta: 7 de octubre de 2012.
<http://www.gamestudies.org/0101/editorial.html>

STOY, Chris y RED STORM ENTERTAINMENT

2006 "Game Object Component System". En Dickheiser 2006: 393-403.

SUSI, Tarja, Mikael JOHANNESSON y Per BACKLUND

2007 "Serious Games – An Overview", School of Humanities and Informatics University of Skövde, Sweden. Consulta: 7 de octubre de 2012. <http://www.autzones.com/din6000/textes/semaine12/SusiEtAl%282005%29.pdf>

TAVINOR, Grant

2008 "The Definition of Videogame". *Contemporary Aesthetics.*Volumen 6. Consulta: 02 de octubre de 2011. <http://www.contempaesthetics.org/newvolume/pages/article.php?articleID=492>

TOZOUR, Paul y ION STORM AUSTIN

2002 "First Person Shooter AI Architecture". En RABIN 2002: 386-396.

UNITY

2012 Página oficial de Unity.Consulta: 7 de octubre de 2012. <http://unity3d.com/>

UNREAL ENGINE

2012 Página oficial de Unreal Engine. Consulta: 7 de octubre de 2012. <http://www.unrealengine.com/>

VASSOS, Stavros

2012 *Introduction to AI strips planning* [diapositivas]. *University of Athens*. Consulta: 25 de octubre de 2012. <http://www.dis.uniroma1.it/~degiacom/didattica/dottorato-stavros-vassos/AI&Games-Lecture1-part2.pdf><http://www.gamesforchange.org/play/september-12th-a-toy-world/>

WIKIMEDIA COMMONS

2007 *File:PeaceMaker - Game interface.jpg.* Consulta: 7 de octubre de 2012. <http://en.wikipedia.org/wiki/File:PeaceMaker_-_Game_interface.jpg>

WILSON, Kyle

2002 "Game Object Structure: Inheritence vs Aggregation", *Game Architect*. Consulta: 7 de octubre de 2012. <http://gamearchitect.net/Articles/GameObjects1.html>

WEST, Mick

2007 "Evolve Your Hierarchy". *Cowboy Programming.*Consulta: 7 de octubre de 2012. <http://cowboyprogramming.com/2007/01/05/evolve-your-heirachy/>