PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE MECANISMOS LÓGICOS DE SOPORTE DE ELEMENTOS Y GESTIÓN DE SECUENCIA DE UN VIDEOJUEGO DE ROL Y ESTRATEGIA EN TIEMPO REAL

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

Piero David Montano Vega

ASESORES: Claudia Zapata
Johan Baldeón

Lima, Diciembre del 2012

Resumen

El presente proyecto de fin de carrera tiene como fin la implementación de las reglas y la lógica de control de un videojuego que permite contar una historia a través de sus personajes. Se exploran características de juego de rol (RPG), centrándose en el control y la evolución de un personaje a través del tiempo en un contexto narrativo; y características de un juego de estrategia real (RTS), presentando un entorno donde se enfatizan desafíos estratégicos, tácticos y logísticos.

En el primer capítulo se presenta la problemática y el contexto que sugiere la implementación de la lógica de control de un videojuego de las características señaladas en el primer párrafo. Además se propone una solución luego de presentar los conceptos principales que faciliten la comprensión del presente documento y un análisis cronológico de la evolución de los elementos de juego de rol y estrategia en tiempo real a través de los juegos más representativos de ambos géneros según el autor.

En el segundo capítulo se realiza un análisis de viabilidad del proyecto y se identifican los requerimientos de la solución propuesta, según la metodología seleccionada, que sirven para presentar en el tercer capítulo el diseño de la solución. El diseño comprende la arquitectura de software y la exploración de cada una de las componentes que la integran y finalmente, el diseño del sistema de directorios y eventos que configuran el videojuego.

En el cuarto capítulo se presentan las herramientas, tecnología y estándares de programación usadas para la construcción de la solución. Además se proponen las pruebas automatizadas y las pruebas de regresión para la verificación de las exigencias del producto.

Finalmente en el quinto capítulo se presentan las observaciones del proyecto donde se evalúa la escalabilidad de la solución y las recomendaciones para la implementación de nuevas funcionalidades en trabajos futuros. Las conclusiones finales son expuestas en el contexto de los resultados esperados y los objetivos planteados al inicio del proyecto.

Dedicatoria

El presente trabajo está dedicado a mis padres, Ricardo Montano Torres y Carmen Vega Santos. Siempre estaré agradecido por su apoyo, esfuerzo e inspiración.

Agradecimientos

En primer lugar quiero agradecer a mis asesores de tesis, el profesor Johan Baldeón y la profesora Claudia Zapata, por su constante apoyo y disposición para con el presente proyecto.

En segundo lugar quiero agradecer a Henry Carbajal y Andrea Cáceres por su ayuda continua a lo largo del proyecto, logrando que el videojuego sea una realidad y un motivo para sentirnos orgullosos de nuestros conocimientos e imaginación.

Finalmente, quiero agradecer al Grupo Avatar PUCP que materializó la idea del videojuego original y que nos permitió desenvolvernos libremente en el desarrollo del mismo.

Tabla de Contenido

Índice de figuras

Índice de tablas

Introducción

La industria de los videojuegos se ha venido mostrando como una de las más rentables en el mundo de entretenimiento. El impacto del crecimiento de esta industria ha supuesto una aparición de nuevas formas de producir y desarrollar videojuegos, sin implicar meramente aspectos lúdicos y creativos, sino sociales, psicológicos, educativos y culturales.

El desarrollo de videojuegos no puede reducirse a un conjunto de prácticas específicas de implementación debido a que se pueden plantear diferentes objetivos y propósitos singulares. Esta premisa implica que no existe una sola forma de desarrollar un videojuego y es por ello que el presente proyecto de fin de carrera no propone una solución al establecimiento de prácticas específicas de desarrollo, sino presenta el proceso de implementación de un videojuego de rol y de estrategia en tiempo real, que pretende ser una guía en el planteamiento y adaptación para futuros proyectos de características similares.

El proyecto se centra en la implementación de las reglas y la lógica de un videojuego que permite contar una historia a través de sus personajes, es por ello que reúne características de juego de rol (RPG), cuya naturaleza se centra en el control y la evolución de un personaje a través del tiempo en un contexto narrativo. Además posee elementos de un juego de estrategia real (RTS) ya que se cuenta con un entorno donde se enfatizan desafíos estratégicos, tácticos y logísticos.

Se debe aclarar que el proyecto se centrará en el planteamiento de reglas y su implementación lógica, facilitando el desarrollo del manejo gráfico del videojuego y la inteligencia artificial del mismo, cuyos temas serán desarrollados por proyectos complementarios al presente, cubriendo la totalidad de la implementación del videojuego.

1. Capítulo 1: Generalidades

En el presente capítulo se expondrá el contexto y la problemática del proyecto. Además se presenta la revisión de los conceptos que permitirán entender el documento y un repaso de las soluciones que se encuentran en el mercado. Finalmente se detallará el esquema de trabajo del proyecto y la sustentación de la solución.

1.1. Definición de la problemática

Los videojuegos son un tipo especial de aplicación multimedia, un producto de entretenimiento que requiere la participación activa del usuario (David Callele; Eric Neufeld; Kevin Schneider 2005). Un videojuego que permita contar una historia a través de la interacción y las hazañas de sus personajes requiere un planteamiento de reglas, estás a su vez requieren de un análisis para su implementación lógica. Sin embargo, las reglas y la implementación del código no siempre representan lo mismo, debido a que las reglas son herramientas abstractas que sirven para pensar en la estructura formal de un videojuego y no necesariamente se manifiestan de manera literal en el código de implementación de un programa (Zimmerman; Salen 2003).

El desarrollo de un videojuego, de cualidades como las mencionadas en el párrafo anterior, representa un esfuerzo en el análisis de tres puntos de vista que se centran en la organización del juego, la experiencia del usuario final y el contexto social e histórico. Respectivamente se puede identificar esquemas de diseño de videojuegos: Reglas, Juego y Cultura (Zimmerman; Salen 2003). Las Reglas que se centran en la lógica esencial y las estructuras matemáticas del videojuego, el Juego que representa la participación e interacción del jugador y finalmente, la Cultura que se centra en el contexto cultural y social que define el propósito del videojuego.

Siguiendo estos esquemas se puede determinar que un videojuego que permita contar una historia debe poseer reglas que limiten las acciones del jugador y que le posibilite cumplir con los objetivos del juego. Además, de que divierta y capte la atención del usuario a través de la participación e interacción con los elementos del juego, que estarán envueltos en un contexto definido por el guión y la historia que se pretenderá contar, además de los objetivos del mismo. El presente proyecto se centrará en las reglas de juego, sin embargo los esquemas definidos nos son mutuamente exclusivos

ni pretenden establecer un taxonomía estricta como sus autores aclaran (Zimmerman; Salen 2003); tanto la interacción y participación del jugador como el contexto del videojuego servirán para diseñar las reglas.

Para el propósito de este proyecto se debe definir la naturaleza del videojuego, que se encuentra fuertemente ligado a las reglas, debido a que se deben establecer los elementos del videojuego y la interacción que se espera con el jugador. Para ello, se ha elegido una naturaleza híbrida de juego de rol (RPG) y estrategia de tiempo en tiempo real (RTS). La primera naturaleza de juego se centra en el control y la evolución de un personaje en el tiempo en un contexto narrativo (Zimmerman; Salen 2003). Y la segunda naturaleza se concentra en un modo principal de juego que tiene un entorno en tiempo real donde se enfatiza los desafíos estratégicos, tácticos y logísticos.

Implementar elementos de un juego de naturaleza RPG permitirá contar la historia desde la perspectiva de los protagonistas mediante el manejo de diálogos y decisiones que tome el jugador a lo largo del juego. Además será capaz de evolucionar a sus personajes en el tiempo según los objetivos que cumpla. Mientras que elementos de un juego de naturaleza RTS permitirán simular batallas dándole al jugador la posibilidad de plantear una estrategia. Además, tendrá la posibilidad de formar un ejército de unidades militares de diferentes características y habilidades.

Por lo expuesto, el videojuego en cuestión requiere de mecanismos que soporten los elementos mencionados en el párrafo anterior, de manera que gestionen la secuencia lógica del juego.

1.2. Marco conceptual

Para comprender de mejor manera el problema planteado y la solución que se presentará, se deben considerar los siguientes conceptos:

1.2.1. Juego

Según Erik Zimmerman (2003) un juego es “Es una actividad voluntaria e interactiva, en donde uno o más jugadores siguen las reglas que limitan su comportamiento, promulgando un conflicto artificial que desemboca en un resultado cuantificable”, dicho

concepto es usado por Nicolas Esposito (2005) para hacer una definición corta y simple de videojuego; como “un juego electrónico soportado por un dispositivo audiovisual y que puede estar basado en una historia”. Esta acepción está basada en la definición de jugar de Erik Zimmerman (2003), “Jugar es el espacio libre de movimiento dentro de una estructura más rígida. Jugar existe debido, y a pesar de, las estructuras más rígidas de un sistema” mencionado en su libro *Rules of Play*.

1.2.2. Juegos de rol (RPG) y de estrategia en tiempo real (RTS)

A lo largo de la evolución de los videojuegos, se han presentado una adaptación de los tipos de juegos existentes. Muchos juegos de tablero han tenido una adaptación en un videojuego y los juegos de rol (RPG) son un claro ejemplo (Figura 1.1).

Según Murray (1999) los “juegos de rol son teátricos de una manera no tradicional, pero emocionante. Los jugadores son a la vez actores y público, y los acontecimientos que retratan tienen la inmediatez de la experiencia personal.”**,** donde los jugadores asumen los papeles de los personajes en un escenario ficticio. Los jugadores asumen la responsabilidad de llevar a cabo objetivos y tareas dentro de una narrativa, ya sea por acción literal, o por medio de un proceso estructurado de toma de decisiones o desarrollo del personaje. Las acciones son adoptadas por los jugadores según un sistema formal de reglas y directrices pueden llevar al éxito o fracaso del juego. La adaptación de este tipo de juego se ha visto reflejada en los videojuegos del mismo nombre, donde los jugadores toman decisiones que afectan el resultado del mismo.

Figura 1.1. Juego de Rol - Baldur’s Gate 1999 (Baldur’s Gate, 2013)

Otro ejemplo claro de un juego que ha tenido una clara aceptación son los videojuegos de Estrategia en tiempo real (RTS), este género que según Dan Adams (2006) puede definirse como un tipo de "juego que incluye construir edificaciones y manejo de recursos", donde se hace hincapié en la habilidad de pensar y planificar una estrategia para lograr la victoria. Una definición simple de un RTS es el propuesto por Rob Pardo de Blizzard Enterteinment, responsable del desarrollo de Warcraft (Figura 1.2), juego de esta naturaleza, enfatiza que "un juego estratégico es el que tiene como modo principal de juego, un entorno en tiempo real" donde se enfatizan los desafíos estratégicos, tácticos y logísticos (Adams, 2006). Este género de videojuego también puede ofrecer retos económicos y de exploración de escenarios.

Los juegos de estrategia en tiempo real a diferencia de los juegos de estrategia basados en turnos, no progresa en forma incremental. Los jugadores de estrategia basados en turnos (Eldridge, 2012) "a menudo toman decisiones de pensamiento deliberado y extenso. Construir edificios y ejércitos son elementos muy comunes en este tipo de juegos". Muchos elementos de juegos de estrategia basados en turnos han sido adoptados por los juegos de estrategia en tiempo real (RTS). El nombre de Juegos de Estrategia de Tiempo Real puede acuñársele a Brett Sperry (Keh, 2002) que expone que "la historia explorará la manera en como Westwood Studios ha contribuido en la creación del género a través de su innovador juego de Dune II: La construcción de un Imperio", en sus propias palabras explica cómo se terminó llamando el género de este videojuego (Keh, 2002):

"Un hecho menos conocido fue el nombramiento oficial del género que iba a ser utilizado para explicar el juego a la prensa y los jugadores. No fue sino hasta algún tiempo después de que el juego estaba en desarrollo que decidí llamarlo 'Estrategia en tiempo real' - que parece obvio ahora, pero había un montón de ida y vuelta entre llamarlo un "Juego de guerra en tiempo real", "Guerra en tiempo real", Juego de guerra', ó "Juego de estrategia." […] Pero al final, era mejor llamarlo un 'RTS', porque eso es exactamente lo que era."

En este tipo de juego la posición y el manejo de los personajes en un escenario buscan asegurar una zona del mapa sobre la que tienen control y/o destruir los bienes y estructuras del oponente. En las palabras de David Keh (2002), "Una interfaz básica, por ejemplo, en un determinado juego RTS puede ser dividida a grandes rasgos en tres secciones: Un mapa detallado, un mapa de radar y una barra de comandos. Sin importar la línea argumental, cada batalla comienza cuando el jugador tiene recursos

básicos, una zona simple de construcción y de alguna manera un escenario más largo que no está descubierto. El jugador debe explorar el mapa, recolectar recursos, construir edificaciones y entrenar unidades de combate.” Más adelante menciona que se usan conceptos de desarrollo tecnológico que permita tener una ventaja sobre tus contrincantes, el control indirecto de tus unidades, e inteligencia artificial por parte del enemigo.

Figura 1.2. Juego RTS – Warcraft: Orcs & Humans (Blizzard, 2002)

Debido a la diversidad de géneros y con el fin de cubrir mayores elementos de juego se han desarrollado videojuegos que mezclan géneros, ocasionando géneros híbridos. Tal es el caso de los dos géneros revisados en este marco conceptual. Este género híbrido RPG/RTS incorpora elementos de un juego tradicional de rol y de juegos de estrategia. Generalmente se acentúa el manejo estratégico de los personajes y su desenvolvimiento táctico en el escenario, sin embargo les permite interactuar con los personajes, estructuras y objetos del escenario, además de poseer atributos, poderes, y acciones especiales dignas de un juego rol. En este género se ha visto disminuido la exploración y explotación del mapa y sea destacado el manejo estratégico de las unidades por parte del jugador.

1.2.3. Elementos de un videojuego RTS/RPG

Los elementos de un videojuego de género híbrido, como lo es el de estrategia en tiempo real y de rol (RTS/RPG), son adquiridos por ambas naturalezas de videojuegos. A continuación se pretende cubrir los conceptos pertinentes a lo que se refiere al entendimiento del contenido de este proyecto.

Al definir videojuego (Esposito, 2005) en el apartado 2.2.1 se hizo referencia a un dispositivo audiovisual, este dispositivo debe tener una interfaz, la cual le permita generar una comunicación con el usuario. Chris Crawford (1990) hace una aguda observación sobre estas interfaces:

“Si bien es cierto que muchas innovaciones de interfaz de usuario aparecieron por primera vez en los entornos académicos y de investigación, la primera comercialización de las muchas innovaciones aparecieron en los juegos. Ventanas de desplazamiento, joysticks, trackballs, interfaces Point-And-Click, haga doble clic, clic y arrastrar [...] aparecieron en los juegos [...] mucho antes que la interfaz de Macintosh se presentara.”

Esta interfaz debe proporcionarle un alto grado de interacción al usuario a tal punto que “debe haber una interactividad continua. El usuario siempre debe tener algo que hacer, y el programa nunca debe congelarse. El usuario nunca debe perder el control de esta interacción” (Rawlins, 2012) según las palabras de Gregory J. E. Rawlins, profesor asociado de la Universidad de Indiana.

Los objetivos de un videojuego son “los que proporcionan de un propósito a la experiencia y ayudan a situar incidentes en una línea de tiempo o en un camino, lo que también ayuda a recordar estos incidentes” (Rawlins, 2012) según Gregory J. E. Rawlins. Además aclara que en el camino hacia el logro de estos objetivos, “el jugador se enfrenta a una serie de desafíos. Cada reto superado le dice al jugador que debido a su progreso se avanza cada vez más hacía el punto de culminación de juego, es decir al objetivo.”

Según Federico Peinado (2012), de la Universidad Complutense de Madrid, el nivel de un videojuego “es una porción del juego con uno o más objetivos claros pensada para jugarse como una unidad, en una sola sesión de juego”, donde aclara que “es inevitable otorgar una estructura a los componentes del juego, formando niveles,

fases, misiones, mundos, etc.” al definir una porción de un juego. Donde los objetivos claros definen el éxito del nivel, para pasar al siguiente.

Los escenarios son una parte importante en este tipo de videojuegos ya que le permiten al usuario interactuar con los objetos que se desenvuelven en el. Los comúnmente llamados mapas, definidos como “espacios activos que en esencia son un tipo de control" del nivel según Gregory J. E. Rawlins. Estos escenarios suelen ser a menudo de dos dimensiones en este tipo de juego, sin embargo existen escenarios isométricos que son usados en videojuegos de estrategia en tiempo real (Figura 1.3). Un escenario isométrico (Davison, 2005) es “la base de muchos juegos de estrategia de tiempo real (RTS), juegos de guerra y simulaciones. Esta distribución isométrica es la superficie del juego y usualmente está escondida” a los ojos del usuario. La isometría del escenario (Davison, 2005) “genera una sensación artificial de profundidad, como si el punto de vista del jugador es en algún lugar del cielo, mirando hacia abajo sobre el área de juego. Es artificial debido a que no se aplican efectos de perspectiva.”

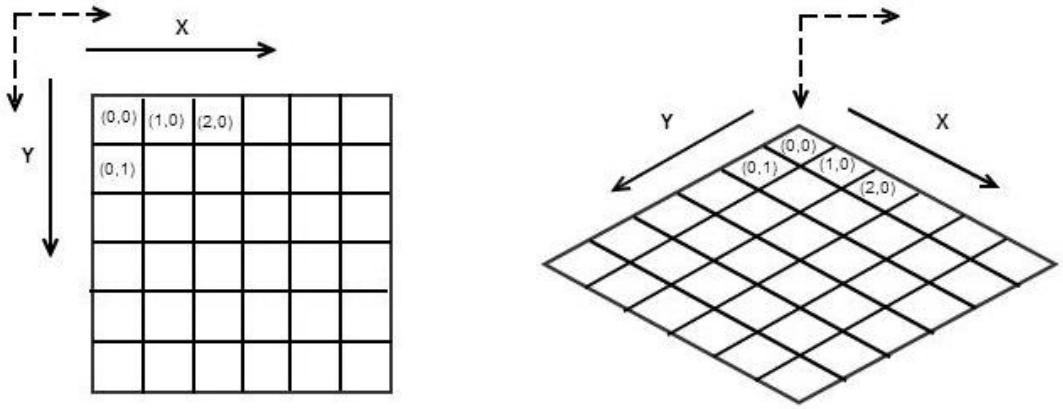

Figura 1.3. Distribución isométrica

Gregory J. E. Rawlins hace hincapié en cuatro puntos de la navegación (Rawlins, 2012) de un videojuego. Habla sobre la reversibilidad de la navegación. Es decir que “entrar a una habitación por una puerta significa que se puede salir de la sala al usarla. Esto permite que el jugador sepa que la exploración es (normalmente) inofensiva y por lo tanto, se recomienda, ya que cualquier acto de navegación se puede deshacer con sencillez y naturalidad.” Habla de la inmediatez y la coherencia de la navegación donde “se quiebra la suspensión de incredulidad al llegar fuera de la pantalla, es decir

fuera del mundo, al navegar". El tercer punto habla sobre el misterio, donde no todo el espacio del escenario se revela a la ver. Los jugadores "tienen que navegar para ver cosas nuevas." Finalmente, el realismo de la navegación debe generar una "sensación de distancia cubierta o de movimiento por parte de los objetos del escenario."

En el presente documento se empleará el término "archivos de configuración" refiriéndose a los archivos que permiten inicializar valores, actualizar estados de juego, presentar guiones, entre otras funciones. Y cuando se mencionen "mecanismos lógicos" se refieren a los artilugios que permitan implementar la lógica interna y funcional de un videojuego.

1.3. Objetivo general

Análisis, diseño e implementación de los mecanismos lógicos que gestionan los elementos y la secuencia lógica de un videojuego de rol y de estrategia en tiempo real, que permita contar una historia a través de la interacción de sus personajes y el cumplimiento de objetivos.

1.4. Objetivos específicos

1. Adaptar la arquitectura de software para el manejo de elementos (niveles, personajes y escenarios isométricos de dos dimensiones) del videojuego.

2. Analizar, diseñar e implementar los algoritmos que gestionen los elementos del videojuego.

3. Adaptar la arquitectura de software necesaria para gestionar la secuencia lógica (objetivos, misiones y eventos de interacción) del videojuego.

4. Analizar y diseñar e implementar los algoritmos que soporten la secuencia lógica del videojuego.

5. Verificar la idoneidad de los algoritmos implementados, mediante la integración al videojuego base.

1.5. Resultados esperados

1. **Resultado esperado del Objetivo 1:** Adaptación de la arquitectura de software del videojuego para la gestión de niveles, subniveles, personajes y escenarios isométricos de dos dimensiones según archivos de configuración.

2. **Resultado esperado del Objetivo 2:** Algoritmos de gestión de elementos del videojuego que soportan configuración inicial y en tiempo de ejecución.

3. **Resultado esperado del Objetivo 3:** Adaptación de la arquitectura de software del videojuego para la gestión de objetivos, misiones y eventos de interacción según archivos de configuración.

4. **Resultado esperado del Objetivo 4:** Algoritmos de gestión de elementos del videojuego que soportan configuración inicial y en tiempo de ejecución.

5. **Resultado esperado del Objetivo 5:** Prototipo que integra las funcionalidades desarrolladas al videojuego base.

1.6. Alcance del proyecto

El presente proyecto se centrará en la implementación de la lógica de un videojuego que le permita contar con elementos tanto de rol como de estrategia en tiempo real. Estos elementos sugieren un videojuego que cuente una historia a través de sus personajes principales, que tendrán a su mando un conjunto de unidades militares que busquen derrotar a las fuerzas enemigas. Además se podrá interactuar con el resto de personajes a través de conversaciones que resultarán en toma de decisiones que afectarán el futuro de la historia. El videojuego está dividido en niveles, que cuentan con objetivos que el jugador deberá cumplir para superar cada uno de ellos.

El videojuego estará alimentado por las entradas que se produzcan de la interacción con el usuario, activando eventos que le permitan cumplir con los objetivos de cada nivel. Para ello se requiere un desarrollo concurrente que permita reconocer los eventos ante la interacción del usuario en tiempo real. Sobre los escenarios del videojuego, se puede indicar que se manejarán solo dos dimensiones (ancho y largo) y

serán de naturaleza isométrica generando la sensación de profundidad en la perspectiva visual del usuario

Por lo expuesto, el presente proyecto de fin de carrera pretende implementar los mecanismos lógicos que gestionen la secuencia del videojuego según las reglas mencionadas y el soporte lógico del manejo de sus escenarios isométricos. No se cubrirá el manejo algorítmico de mecanismos de interacción y movimiento de elementos del videojuego, ni la implementación del manejo gráfico y de interfaz. Sin embargo, cabe resaltar que estos temas serán tratados en proyectos complementarios similares al presente que buscarán cubrir la totalidad del desarrollo del videojuego.

1.7. Limitaciones del proyecto

La principal limitación del proyecto ha sido la falta de material teórico disponible acerca del diseño de un videojuego de características similares al propuesto en el presente documento, sin embargo se debe aclarar que el material técnico es abundante y se pueden encontrar diversas fuentes sobre el área de interés como la bibliografía que este documente sugiere. Esta limitación ha invitado a que se realice revisiones de experiencias en videojuegos pasados que han servido a la consolidación del género RTS/RPG, y se pueden encontrar en la revisión del estado del arte en el apartado 1.7.

El planteamiento de la arquitectura de software del videojuego sugiere otra limitación en el proyecto debido a la poca teoría disponible, debido a que la industria de los juegos es una emergente, y su evolución se ha venido manifestando en nuevos géneros que sugieren una nueva distribución de componentes de software según las funcionalidades que plantea y las limitaciones técnicas que sugiere el uso de una plataforma (Hardware). Es por ello que se ha decidido adaptar una arquitectura propuesta por Michael Doherty, que se expone a detalle en el apartado 3.1.

Debido a que el presente proyecto solo abarca una parte del desarrollo del videojuego, se presenta el riesgo de incompatibilidad de la solución con las partes faltantes (manejo algorítmico de mecanismos de interacción y movimiento de elementos del videojuego, como la implementación del manejo gráfico y de interfaz). Sin embargo se ha visto adecuado informar de esta limitación a las personas encargadas de la implementación de las partes faltantes, llegándose a acuerdos que conciernen el desarrollo conjunto del videojuego.

1.8. Justificación del proyecto

A nivel de desarrollo, el producto que resulta del presente proyecto provee de mecanismos lógicos de gestión de elementos (nivel, personajes, escenarios) y secuencia lógica (objetivos, misiones, eventos de interacción) que facilitarán la implementación del videojuego, sirviendo de base para el desarrollo faltante del mismo (manejo algorítmico de mecanismos de interacción y movimiento de elementos del videojuego, como la implementación del manejo gráfico y de interfaz).

A nivel de usuario, el producto puede ser configurado de manera que pueda ajustarse a las especificaciones de un diseñador de videojuegos y a la historia que se pretende contar, debido a que cuentan con archivos de configuración que pueden ser alterados resultando en la creación de personajes a medida, distribución de escenarios, especificación de objetivos, misiones, diálogos entre personajes, entre otros.

Tanto el producto como el proyecto pretenden aportar a la comunidad de desarrolladores; el proceso de planteamiento de un videojuego de características de rol y estrategia en tiempo real, la adaptación que se hizo de la arquitectura propuesta por Michael Doherty, la gestión de los elementos y la secuencia lógica del videojuego y por último el planteamiento de configuración mediante recursos a nivel inicial y en tiempo de ejecución.

Cabe resaltar que hasta la fecha de publicación del presente documento, el producto ya ha sido parte del desarrollo del videojuego 1814: Rebelión del Cuzco, desarrollado por el grupo Avatar PUCP, que tiene como propósito contar lo sucedido en el levantamiento del Cuzco en 1814 contra la opresión de la fuerza realista.

1.9. Revisión del estado del arte

El objetivo de la siguiente revisión del estado del arte es un reconocimiento de los principales videojuegos en su género y como estos han adoptado las reglas generales tanto de RTS (estrategia en tiempo real) y RPG (de rol). Además se describirá la forma en cómo utilizan sus escenarios y niveles para el desarrollo de su trama. Al final de la revisión se presentará una conclusión.

1.9.1. Dune II (1992)

Dune 2 (Keh, 2002) está basado vagamente en las series de libros escritos por Frank Herbert. Y la trama se lleva a cabo en un planeta llamado Arrakis, también conocido como Dune (Figura 1.4). Existen tres razas diferentes que luchan por el control del planeta: Atreides, Ordos y Harkonnen. El jugador, en este juego, puede elegir entre cualquiera de las tres razas y cada una de estas tiene características únicas y diferentes fortalezas, como debilidades. En una partida típica, el jugador comienza con un lugar limitado de construcción, algunas unidades de infantería y algunos vehículos. Además solo puede ver una proporción de mapa, característica clásica de un juego RTS. Para poder revelar el mapa, el jugador debe explorar el mapa.

El jugador tiene como tareas construir edificios, lo que le permite construir otros más avanzados. Además debe desarrollar unidades de combate para vencer al enemigo y ganar más terreno. El enemigo de este videojuego cuenta con inteligencia artificial y el jugador puede ser atacado al aproximarse a la zona enemiga, de manera explorativa o con motivos de confrontación.

Figura 1.4. Dune II (Dune, 2013)

1.9.2. Age of Empires II (1999)

Este videojuego es de naturaleza RTS, es uno de los juegos más importantes del género (Microsoft Games, 1999) y ha sido inspiración para los nuevos juegos que adoptado muchas de sus características. Este juego implementa nuevas opciones de combate táctico, como formaciones de unidades, de patrullaje y de seguimiento, estas últimas implementadas con inteligencia artificial.

Cuenta con alternativas de combate, donde los jugadores pueden incrementar su economía y su civilización mediante intercambios y opciones de diplomacia. Se pueden trazar rutas de intercambios de bienes como también la generación de un mercado con nuevas opciones de comprar y vender recursos.

Nos presenta trece tipos de civilizaciones que poseen atributos únicos, tecnologías y edificios, además de unidades especiales basadas en su contexto histórico (Figura 1.5).

Figura 1.5. Age of empires II (Microsoft Games, 1999)

1.9.3. Warcraft III (2002)

Este videojuego RTS es la tercera entrega de la saga de Warcraft (Blizzard, 2002) creada por Blizzard Entertainment y presenta un mundo épico medieval donde se desarrollan cuatro razas importantes; humanos, orcos, elfos nocturnos y muertos vivientes (Figura 1.6).

Esta nueva entrega incluye el uso de héroes, lo que le permite al jugador experimentar la interacción de un personaje capaz de tener poderes que influyan sobre los otros personajes aliados y enemigos, además la capacidad de interactuar con el escenario y con los ítems distribuidos en él, característica que no suele ser de un tipo de juego de naturaleza RTS, sino de uno RPG. Cuenta con las características comunes de un juego de estrategia de tiempo real, como la recolección de recursos de los escenarios, desarrollar unidades militares, y la libertad de plantear una estrategia de batalla.

Figura 1.6. Warcraft III (Blizzard, 2002)

1.9.4. Starcraft II (2010)

Starcraft II es un videojuego RTS de ciencia ficción y de corte militar (StarcraftII 2010) creado por Blizzard Entertainment donde se desarrollan tres razas: Los Terrans, que son los humanos que han sido exiliados de la tierra, los Zerg cuya principal característica es que sus unidades y edificios son seres vivos, y por último los Protos, una especie que posee gran poder mental y tecnología avanzada (Figura 1.7).

Este videojuego posee un editor que permite crear campañas, escenarios y modificaciones al videojuego. Y permite publicar y compartir las creaciones mediante conexión a internet. Además cuenta con la capacidad de usar características RPG en las campañas, crear unidades tipo héroes y edificaciones a medida. No sólo permite crear elementos de videojuegos sino que permite modificar y crear una interfaz de usuario propia con características a medida.

Figura 1.7. Starcraft II (StarcraftII 2010)

1.9.5. Diablo III (2012)

Diablo III es la tercera entrega de Blizzard Entertainment (DiabloIII 2012) de la franquicia Diablo (Figura 1.8). Es un videojuego de acción RPG. El jugador puede elegir entre cinco tipos de personajes y cada uno de ellos cuenta con una característica especial. Bárbaro, cuyo recurso es la furia y esta aumenta cuando sufre o causa daño, si la furia llega a su punto máximo se convierte en una bestia de destrucción. Médico brujo, cuyo recurso es el mana, utiliza magia e invocaciones para eliminar a sus enemigos. Arcanista, cuyo recurso es el poder arcano, utilizan sus cuerpos como catalizadores de energía para manipular sus poderes. Monje, cuyo recurso es el espíritu, puede sanar heridas y debilitar a sus enemigos. Y por último el Cazador de demonios, cuyos recursos son el odio y la disciplina, es experto en el uso de arcos y ballestas.

Cabe resaltar que este juego permite que los personajes interactúen mediante diálogos y permite al jugador recoger elementos del escenario al tocarlos. Además los escenarios son presentados con cierta isometría, dando la sensación de profundidad al jugador. Existen elementos en el escenario que brindan cierta ventaja al jugador, como los Santuarios, que al purificarlo te provee de una mejor a las habilidades a los personajes aliados, los Pozos curativos que te restauran la vida y los clásicos Arcones, que contienen armas, oro y tesoros que puedes recoger. Este juego además permite que tengas seguidores que se unen a tu personaje, ofreciéndote su poder ofensivo y curativo.

Figura 1.8. Diablo III (Diablo III 2012)

1.9.6. Conclusión

Esta revisión de los juegos más importantes en su género nos permite observar como se han utilizado los elementos de videojuego en contextos diferentes y con el propósito de contar una historia o cumplir objetivos.

Cabe resaltar, que estos juegos han sido expuestos de manera cronológica, Dune II salió al mercado en 1992 (Keh, 2002), Age of empires en 1999 (Microsoft Games, 1999) y Warcraft 3 hizo lo propio en el 2002 (Blizzard, 2002), lo que permite al lector darle una idea de cómo estos juegos íconos en su género han brindado al anterior características que han funcionado y funcionalidades que se han repetido a lo largo de varias generaciones de videojuegos.

Starcraft II presenta un sofisticado editor que permite crear escenarios y campañas, compartiendo uno de los objetivos de este proyecto que pretende gestionar los escenarios y misiones del videojuego, sin embargo cuenta con limitaciones de control de historia y no cumple con los elementos de rol que se pretenden contar para el proyecto, que Diablo III sí tiene.

Diablo III es un ejemplo claro de los elementos de rol que este proyecto pretende desarrollar. La interacción del personaje principal con el resto de personajes mediante diálogos e intercambio de recursos, el desarrollo del personaje en el tiempo, la interacción con los elementos del escenario y el desarrollo de una historia según las acciones del personaje son los elementos RPG que este proyecto pretender desarrollar, sin embargo no cuenta con elementos RTS que también están en el alcance del mismo.

El fin de esta revisión es adoptar esos elementos que han sido probados exitosos e implementarlos en este proyecto, aprovechando la herencia que han dejado los videojuegos mencionados anteriormente.

1.9.7. Discusión sobre los resultados de la revisión del estado del arte

Tanto la revisión de los conceptos que apoyan este proyecto de fin de carrera como la revisión del estado del arte, han servido para posicionar de manera realista este proyecto en un estado de conocimiento que ha servido para definir su alcance y limitaciones, brindando un enfoque actual y de soporte conceptual necesario para definir los objetivos específicos del presente proyecto.

Uno de los objetivos de la revisión del estado del arte y el marco conceptual ha sido proporcionar a los lectores una perspectiva fresca y actual de la industria de desarrollo de los videojuegos y de los conceptos que se manejan, logrando un mejor entendimiento de este proyecto de fin de carrera.

Como se ha podido observar en el estado del arte, muchos de los elementos del tipo de juego híbrido RPG/RTS (Juego de estrategia de tiempo real y de rol), ya han sido probados como exitosos. Y muchos de estos elementos se pretenden implementar en este proyecto, sin embargo se les proporcionará un enfoque único que permita contar una historia de manera innovadora y que se espera sea un aporte en la comunidad de desarrollo de videojuegos.

1.10. Plan de proyecto

Se ha optado utilizar el marco de trabajo SCRUM para la gestión del proyecto, debido a que promueve un modelo incremental, lo que se ajusta al desarrollo de un videojuego. Aprovechando la agilidad y flexibilidad que promueve debido a la limitación del tiempo con el que se cuenta para la culminación del proyecto en cuestión.

Los incrementos del desarrollo permiten que se esté evaluando constantemente el producto, generando entradas en el siguiente incremento. Estos incrementos no serían permitidos en el uso de un modelo de proyecto rígido como el de cascada (Royce 1970) que puede restar la creatividad, limitar las habilidades y anular la retroalimentación; valores que promueve SCRUM (Deemer; Benefield; Larman; Voode). La implementación de los mecanismos lógicos, que soportan las reglas de un videojuego, debe estar ligada a una continua revisión de los mismos, lo que SCRUM permite mediante la entrega en forma de sprints.

1.10.1. Sprints

SCRUM es un marco de trabajo iterativo e incremental que presenta una estructura de desarrollo basado en ciclos de trabajo que tienen una duración que varía entre una y cuatro semanas, llamados sprints. Estos ciclos tienen una fecha de culminación fija y no pueden extenderse (Deemer; Benefield; Larman; Voode).

Los sprints permiten que se pueda mejorar el trabajo conforme van culminando los ciclos, debido a la retroalimentación que genera la finalización de cada uno. Estos sprints serán útiles en el desarrollo del presente proyecto debido a que permitirá evaluar el avance y promoverá la revisión. Además se podrán realizar ajustes en base a la revisión del anterior sprint. Los sprints acordados tendrán la duración de dos semanas y serán siete en total (Tabla 1.1).

Los elementos (items) que se desarrollarán en cada uno de los sprints no podrán ser modificados en el transcurso del mismo, y se mantendrá una comunicación informativa y constante del proceso de un sprint para facilitar el conocimiento de demoras o bugs. Los elementos que se asignarán en cada uno de los sprints se obtendrán de una lista priorizada de funcionalidades que se anexa a este documento que serán determinadas por la metodología de desarrollo.

1.10.2. Roles y control del proyecto

SCRUM recomienda usar roles como el Product Owner quien representa al cliente y vela por que se cumpla lo que se requiere según la lista priorizada de funcionalidades. En el caso de este proyecto de fin de carrera el asesor asumirá el rol de Product Owner. Mientras que las labores del Equipo Scrum, que incluye a las personas involucradas en el desarrollo del proyecto, y del Scrum Master, encargado de dirigir el proyecto, serán llevadas a cabo por el autor de este documento.

El control del proyecto se podrá monitorear con un diagrama Burndown que representa el proceso del proyecto a largo del tiempo, contabilizándose las horas estimadas y restantes para finalizar el proyecto. La importancia del diagrama es que muestra el proceso para alcanzar los objetivos del proyecto en términos de cuanto trabajo queda para finalizarlo y no en cuantas horas se ha logrado dicho proceso (Deemer; Benefield;

Larman; Voode). Como anexo a este documento, se presenta el diagrama Burndown del presente proyecto.

1.10.3. Parámetros de SCRUM en el proyecto.

El presente proyecto tendrá las siguientes características:

- El proyecto contiene siente Sprints. El primer Sprint (0) estará dedicado a la presentación del proyecto y el análisis del mismo. El último Sprint (6) estará dedicado a pruebas y cierre del proyecto. Los Sprints intermedios serán dedicados para el desarrollo en general.

- Cada uno de los Sprints cuentan con catorce días, excepto el primero que demanda más tiempo debido a la investigación y análisis previo.

- Las reuniones con el asesor son semanales y cada Sprint culmina con una reunión y presentación de los resultados del mismo.

- Cada Sprint con excepción del primero cuenta con una etapa de reflexión donde se revisan los resultados y se ajustan las exigencias del proyecto, aprovechando la flexibilidad del marco de trabajo.

A continuación se presenta el calendario de Sprints.

	Fecha inicio	Fecha fin
Sprint 0	20/08/2012	19/09/2012
Desarrollo de capítulo 1 y 2	20/08/2012	17/09/2012
Desarrollo inicial del mapa del proyecto	18/09/2012	19/09/2012
Identificación inicial de riesgos	20/09/2012	20/09/2012
Sprint 1	21/09/2012	04/09/2012
Revisión del plan	22/09/2012	22/09/2012
Construcción	23/09/2012	02/10/2012
Presentación	03/10/2012	03/10/2012
Reflexión	04/10/2012	04/10/2012
Sprint 2	05/10/2012	18/10/2012
Revisión del plan	05/10/2012	05/10/2012

Construcción	06/10/2012	16/10/2012
Presentación	17/10/2012	17/10/2012
Reflexión	18/10/2012	18/10/2012
Sprint 3	19/10/2012	01/11/2012
Revisión del plan	19/10/2012	19/10/2012
Construcción	20/10/2012	30/10/2012
Presentación	31/10/2012	31/10/2012
Reflexión	01/11/2012	01/11/2012
Sprint 4	02/11/2012	15/11/2012
Revisión del plan	02/11/2012	02/11/2012
Construcción	03/11/2012	03/11/2012
Presentación	04/11/2012	14/11/2012
Reflexión	15/11/2012	15/11/2012
Sprint 5	16/11/2012	29/11/2012
Revisión del plan	16/11/2012	16/11/2012
Compilación del documento final	16/11/2012	16/11/2012
Construcción	17/11/2012	27/11/2012
Presentación	28/11/2012	28/11/2012
Reflexión	29/11/2012	29/11/2012
Sprint 6	30/11/2012	14/12/2012
Revisión del plan	30/11/2012	30/11/2012
Pruebas finales	01/12/2012	10/12/2012
Reflexión final	11/12/2012	12/12/2012
Cierre	13/12/2012	14/12/2012

Tabla 1.1. Calendario del proyecto.

1.11. Descripción y sustentación de la solución

Para poder implementar las reglas de un videojuego de naturalezas RPG/RTS se requiere de mecanismos que gestionen la secuencia lógica del videojuego. Estos mecanismos de gestión se alimentarán de archivos de configuración que soporten inicialización de variables, generen estructuras y especifiquen eventos (Figura 1.9). La elección del uso de estos archivos de configuración se basa en el incremento de productividad y el ahorro del tiempo, además promueve la creatividad mediante la flexibilidad de actualización de estos archivos (Buckland 2005).

1.11.1. Gestión de escenarios

Por medio archivos de configuración se podrán gestionar los escenarios del videojuego. Se contarán con escenarios externos e internos, siendo escenarios externos los asociados a un nivel de juego mientras que los internos pertenecen a un escenario externo. Un escenario externo puede ser un espacio geográfico, mientras que un escenario interno pueden ser las edificaciones que se encuentran en dicho espacio. El gestor de escenario controlará la distribución lógica, las dimensiones y elementos internos asociados al escenario. Cabe señalar que los escenarios son de dos dimensiones e isométricos, lo que resulta en controles especiales de gestión para este tipo de distribuciones.

1.11.2. Gestor de personajes

Este gestor se encargará de controlar a los personajes del videojuego. Se alimentará archivos de configuración para la creación inicial de los personajes. Cada personaje debe estar asociado a un nivel de juego y debe pertenecer a un bando. Por bando se refiere a uno aliado, neutral o enemigo, esta especificación influye en las acciones que podrá realizar el personaje, debido a que un personaje aliado será controlado por el usuario, mientras que el bando enemigo y neutral será controlado por la inteligencia artificial del videojuego. Los archivos de configuración inicial o de inicialización especificarán los atributos del personaje, que incluye los puntos de ataque, defensa, vida, entre otras características. El gestor de personajes controla diferentes tipos de personajes que cuentan con características que heredan de un personaje tradicional, sin embargo pueden presentar características particulares que requieren de controles especiales. Un tipo de personaje puede ser un héroe, que cuenta con atributos especiales como el de respeto, inteligencia y dinero, estas características requieren ser gestionadas de manera específica, además éste puede contar con un inventario de objetos.

1.11.3. Gestor de misiones

Este gestor se encargará de controlar los objetivos del videojuego y será alimentado con archivos de inicialización que especifiquen los eventos asociados a un nivel de juego específico. Para que una misión se desarrolle en el tiempo debe tener eventos asociados, de esta manera el jugador deberá cumplir metas activando eventos de

control para poder cumplir la misión. Este gestor manejará un historial de misiones que servirá como una memoria del avance del jugador en el videojuego hasta cierto punto. Se consideran tres tipos de misiones; principal, secundaria y de apoyo. Una misión principal representa el objetivo del nivel, es decir que para poder continuar con el siguiente nivel, el usuario ha debido de concluir el nivel en donde se encuentra. Una misión secundaria representa objetivos extras que no influyen en la continuidad del jugador en los niveles de juego, sin embargo le permiten desarrollar a sus personajes, percibir recursos y ganar experiencia. Estas misiones principales suelen tener una historia que no impacta en el arco argumental del videojuego, sin embargo soporta la historia y aporta a la diversión del jugador.

1.11.4. Gestor de eventos

Este gestor se encargará de controlar los eventos de una misión. Estos eventos son alimentados por archivos de configuración en el transcurso del videojuego, y son activados mediante una acción incurrida por el jugador. Se controlarán diversos eventos en los que se incluyen los eventos de conversación, que posibilitarán el dialogo entre personajes. Los eventos de decisión permitirán al jugador tomar una decisión entre dos o más opciones resultando en una alteración de los atributos del personaje o en una activación de otro evento. Los eventos de compra-venta permitirán al jugador la posibilidad de adquirir nuevo objetos y vender los objetos de su inventario. Más adelante se detallarán todos los tipos de eventos que permiten ser controlados por el gestor de eventos.

1.11.5. Gestor de niveles

Este gestor se encargará de manejar los niveles del videojuego, según una configuración mediante archivos que establecen valores iniciales. Este gestor deberá controlar los gestores de escenarios y los gestores de personajes, asociándolos a un nivel. Además debe contener estructuras que le permita manejar los gestores de misiones, que a su vez manejan los gestores de eventos que permiten al usuario cumplir con los objetivos del videojuego.

1.11.6. Gestor de secuencia principal

Es el gestor principal del juego y estará encargado de controlar el gestor de niveles según el número de niveles configurados. Cabe resaltar que este gestor también controla mecanismos que permiten la interacción entre personajes y manejan el movimiento de los mismos. A continuación se presenta un gráfico que permite visualizar la relación entre los gestores.

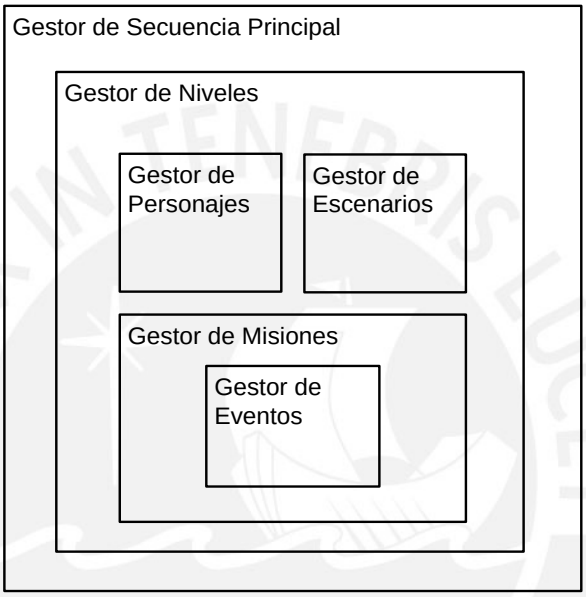

Figura 1.9. Relación de mecanismos de gestión.

2. Capítulo 2: Análisis

En el presente capítulo se detallará la metodología a utilizar en el desarrollo del software, y la adaptación al proyecto. Se especificará el análisis de la solución y las exigencias identificadas agrupadas en módulos de requerimientos.

2.1. Metodología aplicada para el desarrollo

A continuación se detallará la metodología aplicada para el desarrollo del software y las razones de su elección. Además se explicarán las adaptaciones de la metodología para su utilización en el proyecto y se detallarán los artefactos que se obtendrán como resultado de la aplicación de la metodología.

2.1.1. Crystal Clear

Crystal Clear es una metodología ágil de desarrollo de software propuesta por Alistair Cockburn. Está orientada a proyectos que cuenten con equipos de menos de diez personas y promueve el desarrollo de software en incrementos y la comunicación fluida con el equipo y el usuario. Crystal Clear será usada en el presente proyecto de fin de carrera.

El uso de esta metodología complementa el uso de SCRUM, metodología definida para la gestión del proyecto, y considera que incorporar elementos de otras metodologías es una parte natural de Crystal Clear (Cockburn 2004). Como se definió anteriormente, se usarán Sprints de duraciones cortas para el desarrollo del software, lo que sugiere Crystal Clear para las entregas del mismo.

La naturaleza ágil de esta metodología sugiere que la gestión del proyecto y la documentación que resulte no sean un obstáculo para el equipo de desarrollo, sino un soporte que no debe ser eliminado sino reducido a lo esencial y necesario (Cockburn 2004).

A continuación se expondrán las propiedades de Crystal Clear y como se aplicarán al presente proyecto, además se identificará la relación que tienen con el marco de trabajo SCRUM.

2.1.2. Propiedades

Crystal Clear se enfoca más en las propiedades que puede tener un proyecto que en los procedimientos del mismo (Cockburn 2004). Es por ello que plantea propiedades que debe tener un proyecto que pretenda utilizar Crystal Clear como metodología de desarrollo. A continuación se presentan las propiedades que son aplicables al presente proyecto.

- Entregas Frecuentes

  Consiste en presentar parte del software que ya ha sido probado y funcione correctamente según las exigencias planteadas. En el presente proyecto esta propiedad es aplicable debido a que funciona con las entregas mediante incrementos de SCRUM que son calendarizadas por los Sprints definidos.

- Mejoras reflexivas

  Consiste en realizar una reflexión, al final de cada iteración, sobre lo que está funcionando en el proyecto y lo que no. Discutir lo que se podría mejorar y los cambios que se pueden hacer en la siguiente iteración. Reflexionar y mejorar.

- Enfoque

  Consiste en conocer los objetivos del proyecto y las tareas inmediatas para conseguirlos, además tener la paciencia de llevar a cabo dichas tareas. En el presente proyecto se usarán herramientas que permitan facilitar este cometido.

- Fácil acceso a usuarios expertos

  Consiste en mantener una retroalimentación continua con un usuario experto que facilite la implementación y las pruebas de las entregas continuas, además que promueva la calidad continua y facilite decisiones de diseño del producto. En el presente proyecto, el autor y el asesor del mismo harán de usuarios de la herramienta que se pretende implementar.

2.1.3. Proceso de la metodología

Alistair Cockburn define tres etapas del proyecto, cada una con actividades respectivas.

a) Trazado de Actividades

1. Definición del equipo

   En esta actividad se define al equipo del proyecto, sin embargo como este un proyecto de fin de carrera y tiene asociado a un solo alumno se distribuirá los roles entre el autor y el asesor del proyecto.

2. Análisis exploratorio 360°

   En esta actividad se debe establecer si el proyecto es significativo y puede ser entregado usando la tecnología establecida. A través de una evaluación de la viabilidad del proyecto estableciendo las exigencias del mismo y el alcance que se planteará. El presente proyecto cuenta con un análisis exploratorio para la implementación de la herramienta en cuestión.

3. Moldeamiento de la metodología

   En esta actividad el equipo puede ponerse de acuerdo en las reglas y convenciones que definirán el desarrollo del proyecto. En el presente proyecto se han definido reglas con el desarrollo de este proyecto y con el asesor del mismo. Además se ha definido un plan de trabajo que se seguirá estrictamente, como las reuniones semanales con los asesores de proyecto. Estas convenciones y reglas pueden ser cambiadas al final de cada iteración, en el caso de este proyecto al final de cada Sprint en la etapa de reflexión, además se pueden añadir otras según el rendimiento observado.

4. Construcción del plan inicial de proyecto

En esta actividad se define un plan base de proyecto que podrá estar sujeto a cambios según el proyecto se desarrolle (una revisión en cada iteración). En el presente proyecto se ha definido un plan de proyecto usando el marco de trabajo SCRUM, además de la calendarización de los Sprints asociados.

b) Series de dos o más ciclos de entrega

1. Recalibración del plan de entregas

   Después de la primera entrega se contará con nueva información y experiencia, lo que permitirá hacer ajustes en el plan de proyecto, además se podrá conocer el rendimiento de la ejecución del proyecto lo que permitirá manejar los tiempos. Además se pueden revisar los requerimientos, actualizándolos si fuera el caso.

2. Iteraciones

   Cada iteración de desarrollo, definido por un Sprint en el presente proyecto, cuenta con dos semanas de duración. Se adaptará el ciclo de iteración de Crystal Clear, donde en el primer día se definen las actividades a realizar, y se acuerdan los objetivos de la iteración. Se procederá a construir lo planeado, luego se integrará el incremento al proyecto base y se realizarán las pruebas pertinentes.

3. Demostración y reflexión

   A través de la demostración se podrá presentar lo implementado al usuario. En el caso del presente proyecto esta demostración se llevará a cabo en una reunión con los asesores del proyecto y al final de ésta se realizará una reflexión del la iteración, evaluando los cambios que se pueden realizar para la siguiente.

c) Ritual de conclusión: Cierre del proyecto

1. Preparación y entrega

En esta actividad se prepara el producto para ser entregado a los verdaderos usuarios, sin embargo como el presente proyecto tiene como fin implementar una herramienta que permita el manejo de reglas de un videojuego, su entrega final será la integración al videojuego en sí. Para ello se tendrá en cuenta las pruebas de integración pertinentes y los ajustes del caso.

2. Retrospectiva y celebración

Crystal Clear recomienda que al final de una entrega se reflexione sobre el trabajo concluido y se evalue lo que se puede mejorar para las siguientes entregas o los siguientes proyectos. Además recomienda que se deba celebrar la entrega en forma de premiación por el trabajo logrado.

2.1.4. Artefactos

A continuación se presentan los artefactos que se utilizarán en el proyecto. Cabe aclarar que muchos de ellos son sugeridos por Crystal Clear, mientras que otros son artefactos adoptados de otras metodologías. Cabe aclarar que Cockburn promueve el uso artefactos de otras metodologías si el proyecto lo requiere.

- Catálogo historias de usuario. Especificación de las exigencias de las funcionalidades de la solución.

- Mapa del proyecto. Diagrama que muestra la distribución del trabajo y la dependencia entre entregables, de manera que se pueda determinar el orden en que se deben realizar.

- Plan de lanzamientos. Cuadro que contiene las fechas de los entregables asociados a los Sprints del proyecto. Se pueden observar los entregables a entregar en cada uno de los hitos.

- Catálogo de riesgos. Lista de riesgos que pueden perjudicar el proyecto. A cada riesgo se le ha estimado una probabilidad, un retraso estimado en unidad

de semanas, y un índice producto de los anteriores, de tal manera que se puede obtener la prioridad de mitigación o prevención del mismo.

- Diagrama de clases base. Este diagrama permitirá conocer las principales clases de la solución y la relación entre ellas.

- Arquitectura del sistema. Se presentarán diversos diagramas que representan la arquitectura propuesta en el presente proyecto, se presentarán las componentes de la solución y su relación.

- Burn Down Chart. Diagrama que muestra el progreso del proyecto a lo largo del tiempo, de tal manera que se puede visualizar tanto el progreso planificado como el real.

- Pruebas. Se usarán pruebas automatizadas bajo el framework JUNIT, estás serán explicadas a detalle en el capítulo 4.

2.2. Identificación de requerimientos

Esta sección listará los requerimientos funcionales y no funcionales con los que la herramienta deberá contar.

2.2.1. Requerimientos funcionales

Se identificarán los requerimientos, que permitirán desarrollar la solución al planteamiento de reglas de un videojuego RPG/RTS de dos dimensiones, presentados en cuatro módulos que mantienen una relación con los gestores mencionados en el capitulo anterior (Figura 1.9). La solución:

Módulo de escenarios	
Código	**Requerimiento**
ESCE-01	Permitirá crear un escenario de dos dimensiones mediante un archivo de configuración.
ESCE-02	Permitirá configurar las dimensiones y la distribución lógica de un escenario.

ESCE-03	Permitirá configurar los elementos internos de un escenario y los accesos a otros escenarios mediante un archivo de configuración.
ESCE-04	Permitirá leer los archivos de configuración e inicialización de los gestores de escenarios.
ESCE-05	Permitirá gestionar la isometría de los escenarios en tiempo de ejecución.

Tabla 2.1 Requerimientos del módulo de escenarios.

Módulo de personajes	
Código	**Requerimiento**
PER-01	Permitirá crear personajes mediante archivos de configuración
PER-02	Permitirá inicializar los atributos de un personaje, configurar sus acciones y asociarlo a un bando específico (Aliado, neutral o enemigo).
PER-03	Permitirá gestionar los batallones de un personaje. (Conjunto de personajes de un mismo bando) mediante archivos de configuración
PER-04	Permitirá controlar las acciones de un personaje (Curar, atacar, patrullar, seguir, entre otras.)
PER-05	Permitirá gestionar los bandos de los personajes y su comportamiento.
PER-06	Permitirá controlar a más de un personaje a la vez.
PER-07	Permitirá controlar los atributos del personaje en tiempo de ejecución.
PER-08	Permitirá controlar el inventario de objetos de un personaje.
PER-09	Permitirá que un personaje pueda acceder a un escenario y salir del mismo.
PER-10	Permitirá controlar el nivel de experiencia de los personajes.

Tabla 2.2 Requerimientos del módulo de personajes.

Módulo de Niveles	
Código	**Requerimiento**
NIV-01	Permitirá gestionar los niveles del videojuego asociándolos a un escenario y a un conjunto de personajes mediante el cumplimiento de objetivos definidos en las misiones.
NIV-02	Permitirá gestionar las misiones asociadas al nivel, manteniendo un historial de misiones realizadas por el usuario mediante la activación de diferentes eventos.
NIV-03	Permitirá crear archivos de configuración con las misiones del nivel y sus parámetros de activación y visibilidad.
NIV-04	Permitirá controlar eventos de diferentes misiones concurrentemente.

Tabla 2.3. Requerimientos del módulo de niveles.

Módulo de eventos	
Código	**Requerimiento**
EVE-01	Permitirá crear eventos mediante archivos de configuración.
EVE-02	Permitirá configurar y controlar eventos de conversación entre dos personajes.
EVE-03	Permitirá configurar y controlar eventos de toma de decisiones por parte del jugador.
EVE-04	Permitirá configurar y controlar eventos de intercambio de objetos entre dos personajes.
EVE-05	Permitirá configurar y controlar eventos de cambios de bandos de personajes.
EVE-06	Permitirá configurar y controlar eventos que midan atributos para activar ciertos eventos.
EVE-07	Permitirá configurar y controlar eventos para que se finalice un nivel.
EVE-08	Permitirá configurar y controlar eventos que se activen

	al obtener cierto objeto.
EVE-09	Permitirá configurar y controlar eventos que verifiquen que se ha eliminado uno o más personajes de un batallón.
EVE-10	Permitirá configurar y controlar eventos que midan si un personaje está al borde de la muerte.
EVE-11	Permitirá configurar y controlar eventos que finalicen una misión específica.
EVE-12	Permitirá configurar y controlar eventos que activen un acceso a un escenario interno.
EVE-13	Será posible configurar y controlar eventos que actualicen una misión con información de culminación.
EVE-14	Permitirá configurar y controlar eventos que actualicen información de un evento que pertenece a la misión en desarrollo.
EVE-15	Permitirá configurar y controlar eventos que activen una nueva misión.
EVE-16	Permitirá configurar y controlar eventos que actualicen información de un evento que no pertenece a la misión en desarrollo.

Tabla 2.4 Requerimientos del módulo de eventos.

2.2.2. Requerimientos no funcionales

A continuación se muestra la lista de requerimientos no funcionales de la herramienta.

No funcionales	
Código	**Requerimiento**
NOF-01	Ser desarrollado en lenguaje Java, usando la plataforma OpenJDK.
NOF-02	Debe tener compatibilidad con la implementación de mecanismos de interacción y movimiento de

	personajes.
NOF-03	Debe facilitar el dibujado de la interfaz gráfica del videojuego y de los elementos gráficos.
NOF-04	Debe facilitar su actualización futura.
NOF-05	Los tiempos de ejecución deben ser evaluados con el Log4Java y mantenidos al mínimo posible de performance.
NOF-06	Soportar acciones concurrentes sobre la herramienta.
NOF-07	El software final deberá ejecutarse en ordenadores que posean desde 1GB de memoria RAM.

Tabla 2.5. Requerimientos no funcionales

2.3. Análisis de la solución

Como parte del análisis exploratorio 360° que propone Crystal Clear se procederá a analizar la viabilidad de la solución propuesta (Cockburn 2004). Este análisis será realizado desde un enfoque costo-beneficio, técnico y económico. Además se propondrá una definición base de la solución propuesta.

2.3.1. Análisis costo-beneficio

En la siguiente tabla se muestran los costos y los beneficios de la solución propuesta.

Costo	Beneficio
Costos de hardware	Facilidad de configuración de reglas de un videojuego RPG/RTS
Costos de mantenimiento	Desarrollo basado en elementos de videojuegos exitosos
Costos de desarrollo continuo	Posibilidad de mantener la información de configuración organizada

Costos de integración al videojuego base.	Facilidad de integración con el dibujado de la interfaz gráfica y los mecanismos que la soportan.
Costos de integración al videojuego base.	Facilidad de integración con los mecanismos que soportan la interacción y el movimiento de personajes.

Tabla 2.6 Tabla costo-beneficio

Debido a que la solución forma parte del desarrollo de un videojuego RTS/RPG, esta no cuenta con beneficios económicos propios, sin embargo el videojuego final podrá ser financiado por medio de publicidad o por fondos concursales.

Cabe resaltar que esta herramienta también podrá ser usada independientemente como una herramienta que permita plantear las reglas y la gestión principal de un videojuego que pretenda contar una historia.

2.3.2. Análisis técnico

Para poder facilitar algunas de las tareas de gestión del proyecto y el desarrollo del mismo se hará uso de herramientas de control de versiones, gestión de proyecto, tecnologías de programación y entornos de programación. Para ellos se evaluarán las algunas de las opciones existentes mediante cuadros comparativos que facilitarán la selección según sea el caso. A continuación se presentan las herramientas.

a) Herramienta de control de versiones

   Para el control de versiones del presente proyecto se tiene como opciones a GIT y Subversion, ambos de naturaleza diferente - centralizado y distribuido respectivamente - y se procederán a comparar en el siguiente cuadro.

Característica	Subversion	GIT
Velocidad		X
Multiplataforma	X	X
Repositorio único	X	
Control de acceso	X	X
Facilidad de rastreo de cambios	X	
Check out parcial	X	
Facilidad de fusión y ramificación		X
Madurez de Interfaz de usuario	X	

Tabla 2.7 Tabla de comparación de herramientas de control de versiones

Según lo analizado, se ha optado por usar Subversion debido a que a pesar de que no cuente con facilidades de fusión y manejo de ramificaciones de proyecto, cuenta con la facilidad de rastreo de cambios y además podemos hacer operaciones parciales (Checkout). Un sistema de versiones GIT sería mejor aprovechado si se tratará de un proyecto que cuente con un equipo debido a su facilidad de trabajar localmente y la rapidez con la cuenta, sin embargo como el proyecto de fin de carrera será desarrollado por una persona se podrá trabajar con una versión única del repositorio.

b) Herramienta de gestión de proyecto

Se ha decidido usar una herramienta de gestión de proyectos ágiles que sigan las directrices de SCRUM, para ello se ha optado por comparar dos de ellas, Agilefant y IceScrum. A continuación se presenta el cuadro comparativo:

Característica	Agilefant	IceScrum
Uso de puntos de historia		X
Gestión de horas por tarea	X	X
Visualización de pizarra de tareas		X
Gestión de Burn Down Chart	X	X
Gestión de pruebas de aceptación		X
Reportes	X	
Rastreo de cambios		X
Gestión de entregas	X	X

Tabla 2.8 Tabla de comparación de herramientas de gestión de proyecto

Como resultado del análisis se ha optado por utilizar IceScrum como herramienta de gestión de proyecto debido a que cuenta con características que permiten planear las tareas mediante un calendario y una pizarra. Además es adecuado para proyectos cortos, y permite contabilizar las horas que mantiene el diagrama de Burn Down al día.

c) Herramientas de programación

A continuación se presentan algunas observaciones sobre el uso de herramientas de programación para el presente proyecto:

a. Se ha optado por usar el lenguaje de programación Java.
b. Se usará la plataforma de Java de código libre OpenJDK.
c. Se usará el entorno de programación NetBeans.
d. Se usará el sistema operativo Windows 7.

Más adelante en el capítulo 4 se detallará el uso de estas herramientas.

2.3.3. Análisis económico

En el siguiente cuadro se expone el resumen de los costos del presente proyecto. Se debe tener en consideración la mano de obra del autor, la computadora utilizada para desarrollar el producto y además los servicios que se han requerido para su propósito.

Concepto	**Horas**	**Costo (S/.)**	**Subtotal (S/.)**
Mano de obra			
Generalidades	150	20 x hora	3000
Análisis	160	20 x hora	3200
Diseño y construcción	200	40 x hora	8000
Conclusiones	30	20 x hora	600
			15800
Otros			

Computadora			800
Luz, internet			1200
			2000
		Total	**17800**

Tabla 2.9 Tabla de resumen de costos

2.4. Definición de clases base

En la tabla siguiente se presenta a las principales clases que permitirán la implementación de la solución.

Clase	Definición
Juego	Cuenta con la información relacionada a la gestión de niveles
Nivel	Representa a una porción del videojuego, tiene asociado un escenario y objetivos.
Mapa	Escenario de un nivel. Puede ser externo o interno (una edificación)
Misión	Objetivo de un nivel, puede ser principal, secundaria o de apoyo a la historia.
Personaje	Carácter del videojuego que cuenta con atributos y acciones que le permiten al usuario desenvolverse en el juego
Evento	Son activados por acciones del usuario y soportan el cumplimiento de objetivos del nivel
Celda	Es la mínima estructura que compone un escenario.
Acceso	Son las entradas o salidas de un nivel (Interno o externo)
Bando	Conjunto de personajes que tiene un fin en el videojuego (Aliado, enemigo y neutro).
Batallón	Conjunto de personajes que pertenecen a un bando y que poseen un líder y características especiales.

Tabla 2.10 Tabla de definiciones de clase

A continuación se presentan algunas de las dependencias de las clases mencionadas anteriormente mediante un diagrama de clases.

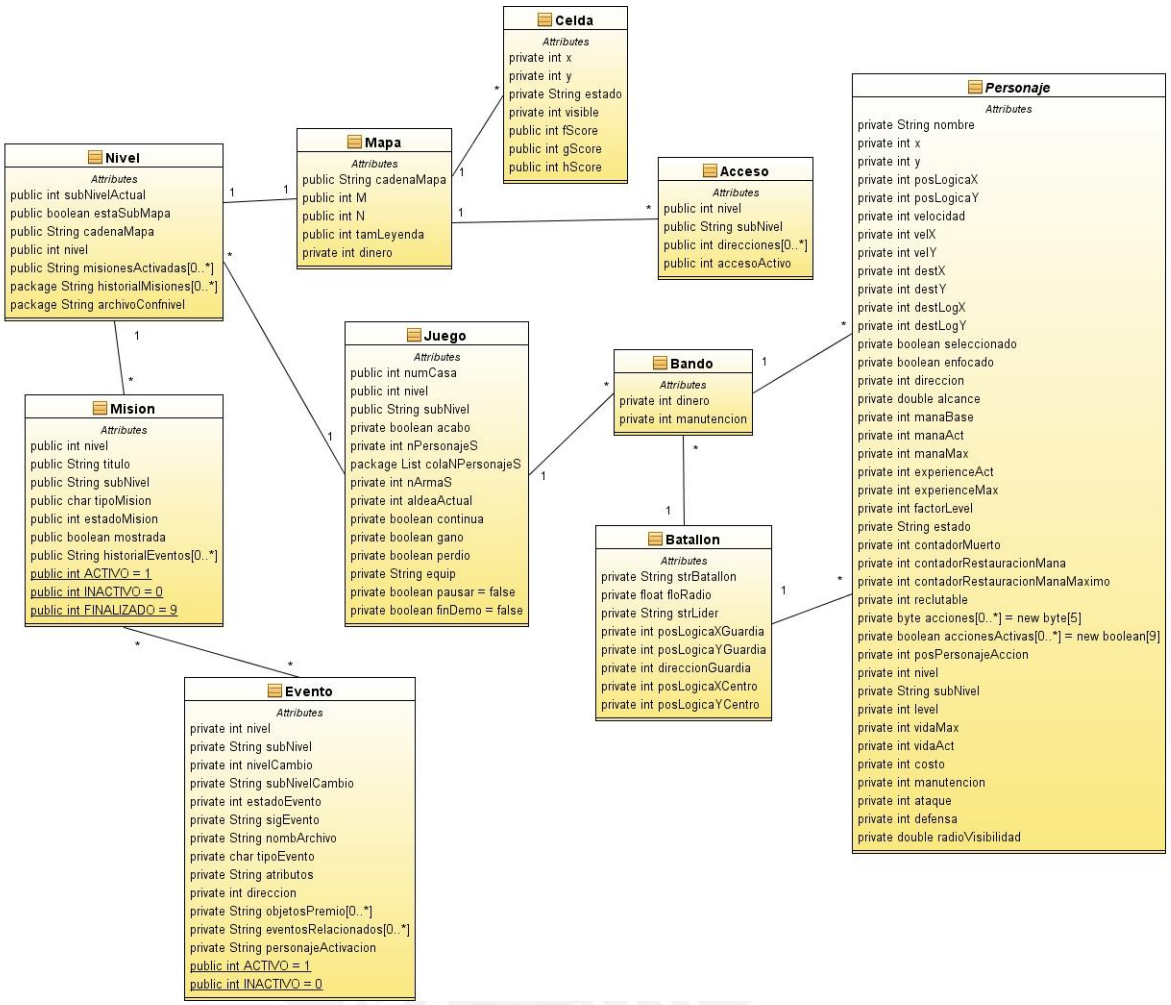

Figura 2.1. Diagrama de clases

3. Capítulo 3: Diseño

En el presente capítulo se definirán aspectos del diseño de la solución, tales como la arquitectura que soportará el cumplimiento de los requerimientos y la configuración de los archivos que alimentarán las estructuras que faciliten su funcionamiento. Es preciso acotar que el alcance de este proyecto no implementa una interfaz gráfica, sin embargo el funcionamiento lógico que plantea requiere de entradas, como la configuración del sistema de directorios que se explicará en este capítulo.

3.1. Arquitectura de la solución

Debido a la naturaleza del proyecto se ha optado por usar como base una arquitectura propuesta por Michael Doherty, orientada a los juegos de tiempo real. Esta arquitectura está diseñada para maximizar la reutilización de recursos (Doherty 2003) y presenta una estructura centralizada que tiene como núcleo un sistema de objetos.

Se dice que esta estructura está centralizada porque los componentes funcionales de alto nivel; el motor del juego, la simulación y el gestor de datos interactúan directamente con el sistema de objetos, como se muestra en la siguiente ilustración.

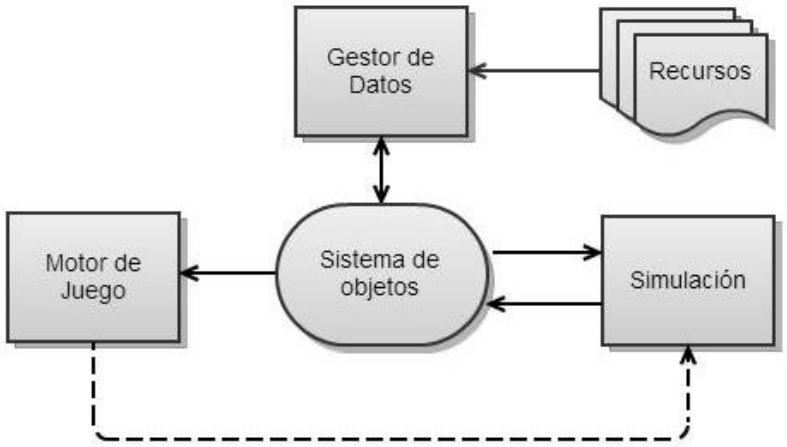

Figura 3.1. Componentes de alto nivel

El motor de juego es responsable de presentar el juego al usuario y recibir las acciones que este realiza. Entre sus funciones se incluye la generación de la interfaz gráfica y audio. La simulación es responsable de actualizar el estado del mundo virtual

del videojuego como respuesta a las acciones del usuario y a las reglas del videojuego. El gestor de datos es responsable de la recuperación de datos del juego, desde un sistema de archivos u otro almacenamiento persistente, con el fin de mantener actualizado el estado del mismo. El sistema de objetos, que interactúa con los demás componentes de alto nivel, es responsable de mantener vigente la información que describe todos los objetos del videojuego. Este sistema reside en memoria para que la información que contiene pueda ser accedida en tiempo real (Doherty 2003).

Se puede notar que el motor de juego solo tiene una relación de lectura de datos con el sistema de objetos, lo que le permitirá realizar la interacción con el usuario, sin embargo mantiene una relación de entrada con la simulación, esto quiere decir que la información obtenida de la interacción con el usuario debe ser manejada y/o transformada en la simulación según las reglas del videojuego para poder ser actualizada en el sistema de objetos. La relación del gestor de datos con el sistema de objetos es de entrada y salida en el tiempo, es decir se pueden obtener los datos de la configuración inicial y en tiempo de ejecución. En el apartado 3.2 se explicará mejor como se llevará a cabo esta carga y la configuración de los archivos que se utilizarán. A continuación se describirá en mayor detalle cada componente.

3.1.1. Motor de juego

Este componente es responsable de interactuar directamente con el jugador, a través del hardware de la computadora y el sistema operativo. A continuación se presentan algunos de los módulos que pertenecen a este componente de alto nivel.

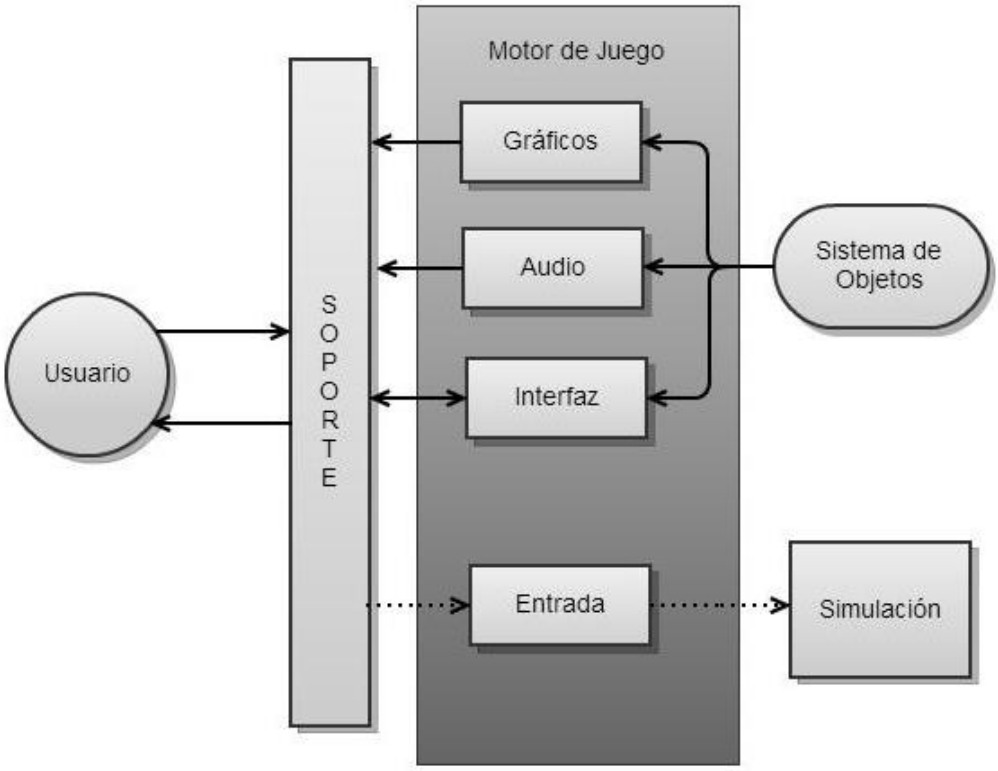

Figura 3.2. Motor de juego

En el gráfico mostrado, la barra vertical que interactúa con el usuario, representa la computadora del usuario. Sin embargo la barra no solo representa el hardware, sino también incluye el sistema operativo, los drivers de los dispositivos, y software de soporte como la maquina virtual de Java (JavaVM) en el caso de este proyecto de fin de carrera. Una característica de los módulos del motor de juego es que su descomposición funcional es natural y el acoplamiento entre ellos es limitado. Además el flujo de datos en este tipo de módulos es unidireccional (Doherty 2003). Se puede apreciar en el gráfico que la información ingresada por el usuario, ya sea por el uso del teclado o el mouse, esta será transferida a la simulación para ser procesada, y no directamente al sistema de objetos. La información con la que cuentan estos módulos del motor de juegos es brindada por el sistema de objetos en un determinado momento de la secuencia del juego. Por ejemplo, si un personaje se encuentra librando una batalla con un sable o una escopeta, el sonido del blandir del sable o el disparo del arma, será ejecutada según la información del estado de acción del personaje en un determinado momento, estos datos serán proporcionados por el sistema de objetos.

Se debe aclarar que la solución propuesta en este proyecto de fin de carrera no implementará el motor de juego, sin embargo si interactuará con él con la finalidad de obtener las entradas del usuario y actualizará la información del sistema de objetos para que puedan ser mostrados por el motor de juego.

3.1.2. Simulación

Según Doherty, la simulación es el corazón del juego. La simulación comprende el mundo virtual del videojuego y mantiene las reglas de mismo. Los módulos que comprenden este componente de alto nivel se presentan en la siguiente imagen.

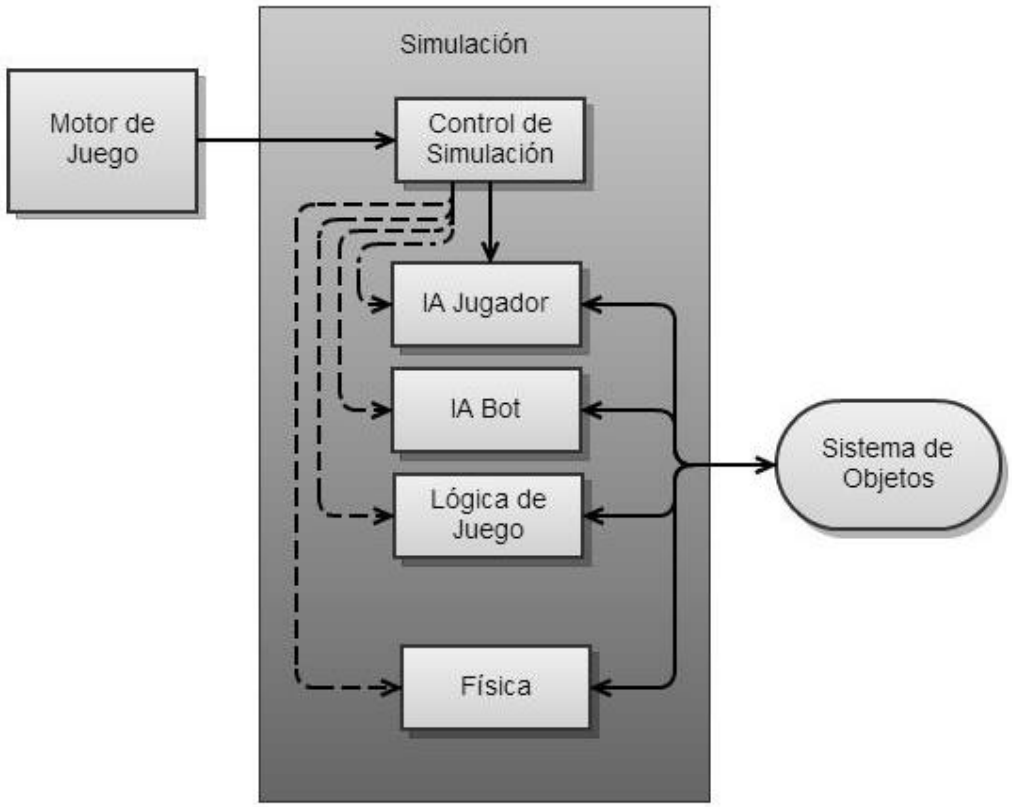

Figura 3.3. Simulación

El módulo de control de simulación recibe las entradas del jugador a través del motor de juego, esta información es compartida por el módulo de inteligencia artificial del jugador. Además comparte información de control tales como tiempo del sistema, tiempo transcurrido, entre otros datos; a los demás módulos de simulación, este flujo está representado por las líneas punteadas. El módulo de la física contiene la lógica de

interacción con el escenario o mundo virtual del juego, de acuerdo a las reglas especificadas. En el caso de este proyecto, se conoce que los escenarios son isométricos, es por ello que el desplazamiento de los personajes debe cumplir con la isometría del escenario. Además se respetan los obstáculos y los edificios como parte de la geografía del escenario. Los módulos de inteligencia artificial contienen el pensamiento interno y el proceso de decisión de los objetos animados del videojuego. En el presente proyecto se considera la inteligencia artificial de los personajes ante eventos específicos, como la interacción de un personaje enemigo y un personaje aliado en una batalla, u eventos que no dependen de eventos, como el patrullaje de personajes enemigos en una zona del escenario, donde el desplazamiento de los mismos se basa en decisiones de inteligencia artificial. Por último, Doherty define al módulo de lógica del juego como el encargado de los cálculos y restricciones inferidas a partir de las reglas del videojuego.

Estos módulos no interactúan directamente, sino por la información brindada por el sistema de objetos en un determinado momento del juego. Para explicar el flujo mencionado se detallará un ejemplo donde intervienen el módulo de inteligencia artificial y el módulo de física. Un personaje que requiere moverse de un lugar a otro es impulsado por el módulo de inteligencia artificial, sin embargo deberá respetar las reglas físicas del escenario, como la imposibilidad de caminar en el agua o sobre un obstáculo. Ésta información deberá ser obtenida del sistema de objetos, para un determinado momento en la secuencia del videojuego.

Cabe aclarar que en la solución que plantea este proyecto de fin de carrera se implementará parcialmente el componente Simulación de la arquitectura presentada. Los módulos de control de simulación, lógica del juego y física serán implementados en la solución propuesta.

3.1.3. Sistema de objetos

Doherty señala la importancia del sistema de objetos como interfaz central de los componentes de alto nivel y su función fundamental de guardián de los datos, acotando que requiere de un diseño e implementación pertinente debido al impacto que tendrá en el resto de las componentes, además debe poseer una flexibilidad que soporte los cambios en el diseño del videojuego. Para ello recomienda que este

sistema sea centralizado a objetos, basándose en el concepto encapsulamiento de información de la orientación a objetos, donde todos los datos relevantes para un objeto en particular son almacenados juntos como una unidad. Para acceder a la data de un objeto, primero se deberá acceder al mismo. En el presente proyecto este sistema estará compuesto por unidades que permitan la jerarquía de objetos, respetando el encapsulamiento mencionando por Doherty.

Cabe resaltar la interacción del sistema de objetos con el resto de componentes, el gestor de datos alimentará inicialmente las estructuras de información del sistema de datos y proveerá de información en tiempo de ejecución cuando el sistema de datos así lo requiera. El sistema de objetos proporcionará de información a la simulación y recibirá información procesada según las entradas del usuario. La información con la que cuenta el motor de juego también será proporcionada por el sistema de objetos. A continuación se muestra esta interacción con un ejemplo.

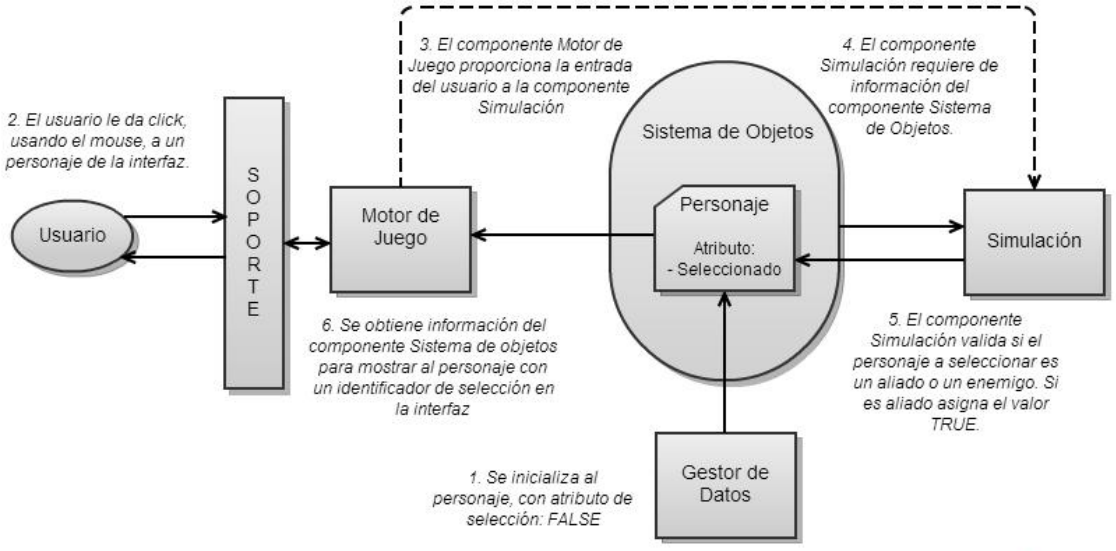

Figura 3.4. Interacción del Sistema de objetos con la arquitectura

En el gráfico se puede apreciar como el sistema de objetos interactúa con el resto de componentes. Según el ejemplo, el gestor de datos inicializa las estructuras del sistema de objetos, que incluye el objeto Personaje. El usuario podrá seleccionar un personaje mostrado en la interfaz del videojuego usando el cursor del mouse. Esta entrada del usuario es recibida por el motor de juego que a su vez proporciona dicha información a la simulación. Para determinar si el usuario puede seleccionar a ese personaje, la simulación requiere de información sobre posición de personaje, posición

del cursor, distribución del escenario e información sobre si el personaje es aliado o enemigo, para ello requiere del sistema de objetos los datos pertinentes. Cuando la simulación ha determinado si el personaje puede ser seleccionado actualiza la información del personaje en el sistema de objetos. El motor de juego obtiene esta información del sistema de objetos y la usa para pintar un indicador de selección asociado al personaje en la interfaz.

3.1.4. Gestor de datos

El gestor de datos es responsable de la recuperación de datos del juego, desde un sistema de archivos u otro almacenamiento persistente, con el fin de mantener actualizado el estado del sistema de objetos. El gestor de datos inicialmente puede inicializar las estructuras del sistema de objetos, sin embargo puede estar en constante alimentación de datos al sistema si éste lo requiere. En el presente proyecto de fin de carrera se contará con un conjunto de directorios donde se almacenará tanto la información inicial, como la información a la que se accederá en tiempo de ejecución. En el apartado 3.2 se explicará la distribución de este sistema de archivos que permitirá flexibilidad en la configuración del videojuego.

El gestor de datos solo interactúa con el sistema de objetos, más no con el resto de componentes para evitar la redundancia de data y un eventual conflicto. Esta interacción puede ser observada en el gráfico 3.4, donde se observa como el gestor de datos inicializa el objeto Personaje del sistema de objetos. Un ejemplo de alimentación de información que no sea de configuración inicial se podrá observar en la toma de decisiones por parte del usuario. Cuando un usuario debe tomar una decisión respecto a una pregunta planteada, tendrá la posibilidad de escoger entre varias opciones, cada una de ellas genera nuevas preguntas y a su vez nuevas decisiones. Esta información es tan vasta que no se encontrará almacenada en el sistema de objetos de manera permanente, sino se accederá mediante el gestor de datos al sistema de archivos y se recuperará la data requerida en tiempo de ejecución.

3.2. Sistema de directorios

Como parte del diseño de la solución se ha propuesto un sistema de directorios que permitirá alimentar a la estructuras de datos del videojuego. Ésta función estará a cargo del Gestor de datos, componente de la arquitectura seleccionada descrita en el

apartado 3.1.4, que configurará el videojuego según los archivos pertenecientes a este sistema de directorios.

Cabe resaltar que se usarán archivos planos de texto y archivos de formato XML para la configuración del videojuego. Los archivos de texto serán útiles para la carga de datos que contengan texto en cantidades importantes, como los guiones del videojuego, facilitando su lectura. Por otra parte, el uso de los archivos de formato XML permitirán la carga masiva de datos, debido a su fácil procesamiento y almacenamiento en la estructuras de datos del videojuego.

Como se mencionó en el apartado 3.1.4, los archivos que se encuentren en este sistema de directorios podrán ser de configuración inicial y otros de configuración en tiempo de ejecución. Es decir, los archivos de configuración inicial serán accedidos por el Gestor de Datos en el lanzamiento de la aplicación, mientras que los archivos de configuración en tiempo de ejecución serán accedidos según las entradas del usuario y lo que determine la simulación del videojuego al procesar la información recibida por el jugador. Cada uno de estos archivos tiene un formato específico que será adjuntado a este documento como anexo. A continuación se presenta el sistema de directorios mencionado.

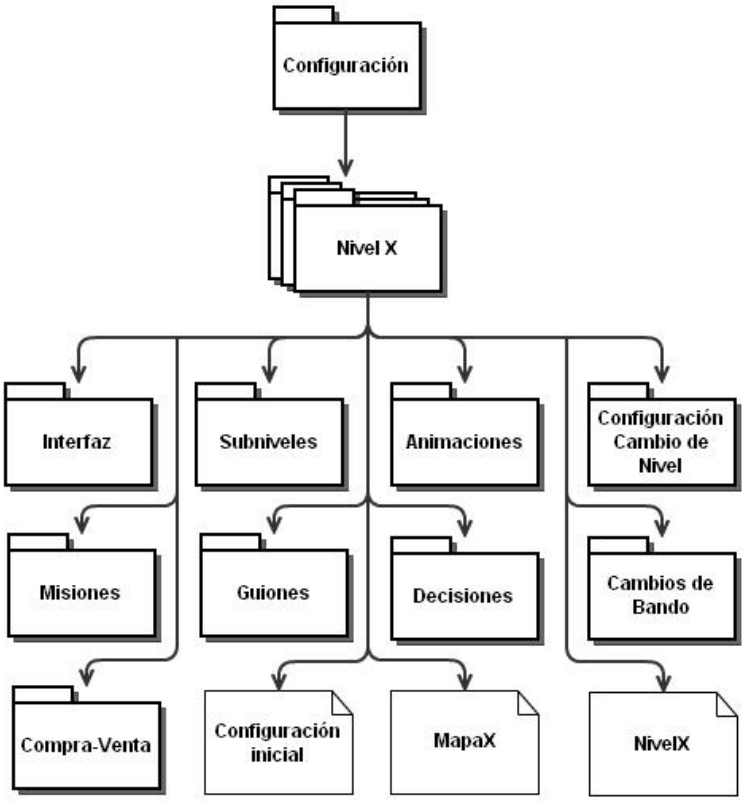

Figura 3.5 Sistema de directorios

Como se puede observar en la figura 3.5, el directorio raíz Configuración contiene la información de todos los niveles del videojuego. Los datos de cada nivel está representado por un directorio con el mismo nombre, en el gráfico se puede apreciar la configuración del Nivel X, que contiene a su vez tres archivos de configuración y nueve directorios. A continuación se definen los archivos y luego cada uno de los directorios del sistema.

El archivo de *Configuración inicial* contiene información sobre la posición inicial del personaje principal o héroe del nivel en el escenario. Se considera además la posibilidad de asignar un desplazamiento inicial a este personaje, generando una sensación de entrada al escenario o edificación. Además el enfoque de la interfaz sobre el escenario estará orientado a la posición inicial de este personaje.

El archivo *Mapa X* contiene información relacionada al escenario del nivel que lo contiene, como las dimensiones del mapa, el número de edificaciones que contiene y la distribución lógica del escenario. Como se menciono en los aparatados anteriores, la distribución del escenario estará compuesta por celdas lógicas, es por ello que cada una de estas celdas que lo comprenden tiene una representación alfabética registrada en este archivo. En la figura 3.6 se puede apreciar la distribución lógica de una fracción de un escenario del videojuego. Donde las letras *T* corresponden a terreno transitable, la letra *O* a terreno que no puede ser transitado y las letras restantes corresponden a edificaciones y obstáculos en el escenario.

Figura 3.6 Archivo de configuración Mapa X

El archivo *Nivel X* contiene información relacionada a los subniveles o edificaciones del escenario. Un subnivel contiene su propia distribución lógica y cuenta con elementos internos, esto se podrá apreciar más adelante. Entre la información que se tiene en este archivo se encuentra el número de subniveles asociados, el estado de acceso, puede ser cerrado o abierto, la zona de activación de entrada al subnivel, la posición inicial de aparición del personaje principal en el subnivel a acceder, las direcciones del personaje en las que puede acceder al subnivel, pueden ser ocho direcciones, que incluyen las cartesianas y las diagonales, y por último el efecto de desplazamiento en el subnivel asociado.

El directorio *Interfaz* posee información gráfica del videojuego relacionada al nivel que lo contiene. Está compuesta por dos directorios como se puede apreciar en la figura 3.7. El directorio *Escenarios* que contiene a los mapas del nivel y sus subniveles, la contraparte gráfica de los archivos de distribución lógica. El directorio *Slides* contiene información sobre el contexto e historia del nivel basado en el guión del videojuego en forma de diapositivas e imágenes. Este directorio no será usado en esta solución debido a que no se encuentra en el alcance de este proyecto.

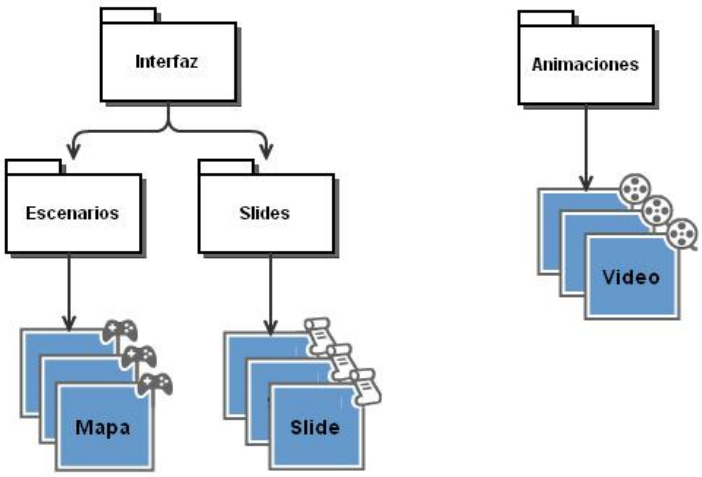

Figura 3.7 Directorios *Interfaz* y *Animaciones*

El directorio *Animaciones* contiene las secuencias de video relacionadas al nivel que las contiene. Estás secuencias soportarán la historia y el contexto del videojuego, brindando al jugador una experiencia completa. El manejo de este directorio no se encuentra en el alcance del presente proyecto.

El directorio *Subniveles* contiene la información sobre los niveles internos del nivel principal. Es decir, contiene información de edificaciones o espacios internos del escenario. Este directorio contiene dos directorios asociados como se puede apreciar en la figura 3.8, el primer directorio Datos contiene a los archivos que tiene información sobre la zona lógica de salida al escenario principal, es decir las celdas lógicas de la distribución del subnivel que activan el cambio de mapa. Las direcciones cartesianas y diagonales en que el personaje puede salir del subnivel y los datos que permiten el desplazamiento automático del personaje hacia la salida del escenario interno. El segundo directorio Mapas contiene información sobre la distribución lógica del escenario interno, como el visto en la figura 3.6. Cabe resaltar que los archivos en cada uno de estos dos directorios tendrán el mismo nombre haciendo alusión a un mismo subnivel.

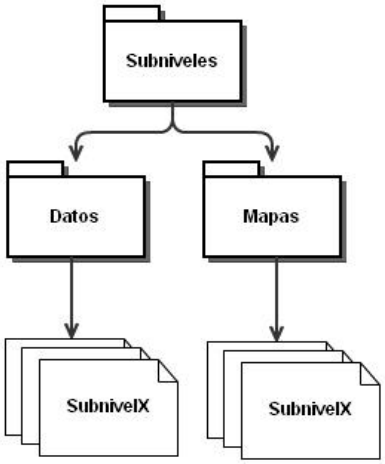

Figura 3.8 Directorio *Subniveles*

El directorio *Configuración Cambio de Nivel* contiene información sobre la inicialización de personajes en el escenario, configuración inicial de los elementos gráficos y los elementos que deben ser eliminados de la memoria al acceder al siguiente nivel. Como se puede apreciar en la figura 3.9, este directorio cuenta con un directorio y dos archivos. El que compete más a este proyecto de fin de carrera será el directorio de Personajes, ya que cuenta con archivos de configuración inicial de los personajes que son parte del nivel que los contiene. Se tienen tres archivos de formato .XML que hacen referencia a la inicialización de los bandos del videojuego, es decir al bando enemigo, aliado y neutral. Estos tres archivos son similares en contenido, ya que especifican a detalle los datos que tendrá cada uno de los personajes de cada bando, entre los que se encuentra, la posición inicial en el mapa, el nombre del personaje, el tipo de personaje, la velocidad de traslado, características especiales, atributos de

personaje, entre otros datos que se verán con más detalle más adelante. Además del directorio Personajes, se cuenta con dos archivos de configuración de cambio de nivel, el archivo *ConfCambioNivel* que tiene información sobre la inicialización de los elementos gráficos del nivel que incluyen slides y secuencias de video, y el segundo archivo, *LimpiarElementos* que contiene información sobre los elementos gráficos que ya no se usarán en niveles posteriores y deben ser eliminados de la memoria para un mejor rendimiento del videojuego.

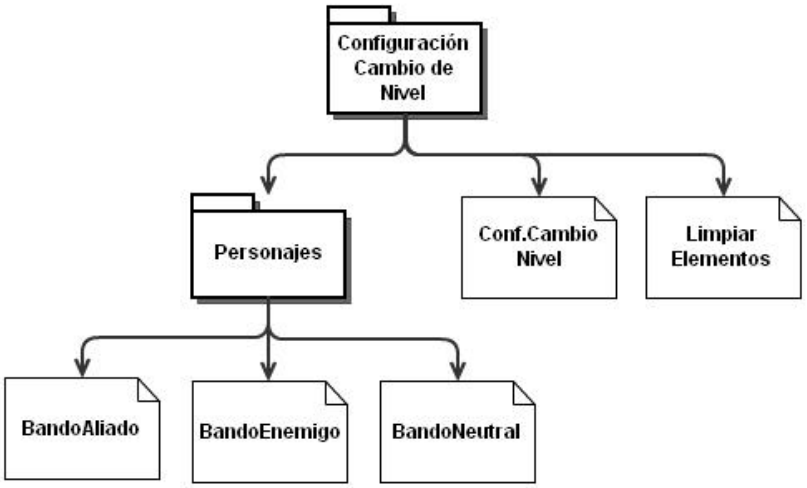

Figura 3.9 Directorio *Configuración Cambio de Nivel*

El directorio *Guiones* contiene a los archivos que representan una conversación en el videojuego en forma de guión. El nombre de los archivos es único y representan un evento en el nivel del videojuego, es decir que son pequeñas conversaciones que complementan la historia del videojuego y permiten al jugador interactuar con los personajes del escenario. Si se juntarán estos archivos en un orden que respete la secuencia de eventos de la historia se tendría un solo guión, sin embargo para propósitos del manejo de eventos y misiones en tiempo de ejecución del videojuego se requiere tenerlos separados.

El directorio Decisiones contiene a los archivos que representan la toma de una decisión por parte del jugador ante el planteamiento de una proposición de alguno de los personajes del videojuego o de algún evento particular. Estos archivos cuentan con la proposición, el número de opciones disponibles, las opciones disponibles y el efecto que causa, en los atributos del personaje, la elección de una opción.

El directorio *Cambios de Bando* contiene información sobre la transición de un personaje de un bando a otro, es decir un personaje puede cambiar de bando debido a un evento en el videojuego. Por ejemplo, un personaje que es neutral puede ser reclutado por el bando aliado para unirse a sus filas, ese evento necesitaría de este archivo para la configuración del cambio de bando.

El directorio *Compra-Venta* contiene información sobre el intercambio de objetos entre los personajes. El personaje principal cuenta con un inventario de objetos y puede conseguir nuevos objetos al intercambiarlos por dinero u obtener dinero al vender sus objetos en establecimientos de compra-venta. Esta configuración de intercambio e inventario esta definida en los archivos contenidos en este directorio.

El directorio de *Misiones* puede considerarse el directorio más importante ya que es el que contiene la información necesaria para el correcto flujo del videojuego. Los archivos contenidos en este directorio determinan los objetivos del nivel y los eventos que permiten su cumplimiento. En la siguiente sección se explicará en mayor detalle el contenido de este directorio, la interacción que tiene con los otros directorios, el diseño del manejo de eventos mediante estos archivos y los tipos de eventos que soporta la solución.

3.3. Manejo de eventos

Se puede apreciar en la figura 3.10 que el directorio *Misiones* cuenta con un número de directorios internos variable debido a que además de contar con una misión principal por nivel, esta solución permite la gestión de múltiples misiones concurrentes. Además cuenta con un archivo de configuración llamado *Misiones* que contiene información inicial de las misiones del nivel, los nombres de las misiones que coinciden con los nombres de los directorios asociados, y un indicador si una misión se considera activa desde el inicio del nivel.

Tanto las misiones secundarias como la misión principal tienen la misma estructura de directorio. Es por ello que el gráfico 3.10 se considera *Misión X* como el directorio ejemplo, este directorio contiene un directorio interno y dos archivos asociados. El archivo *Resumen* no se considera un archivo de configuración debido a que no tiene interacción con el Gestor de datos, sin embargo sirve como guía para el encargado de diseñar las misiones del juego, ya que contiene una descripción de la misión y sus

eventos. El segundo archivo de configuración *Misión X*, llamado de la misma forma que el directorio, cuenta con información de los eventos que definen los objetivos de la misión, es decir contiene el flujo de la misión en forma de eventos, que pueden ser guiones, decisiones, búsquedas de objeto, confrontaciones con el enemigo, etc.

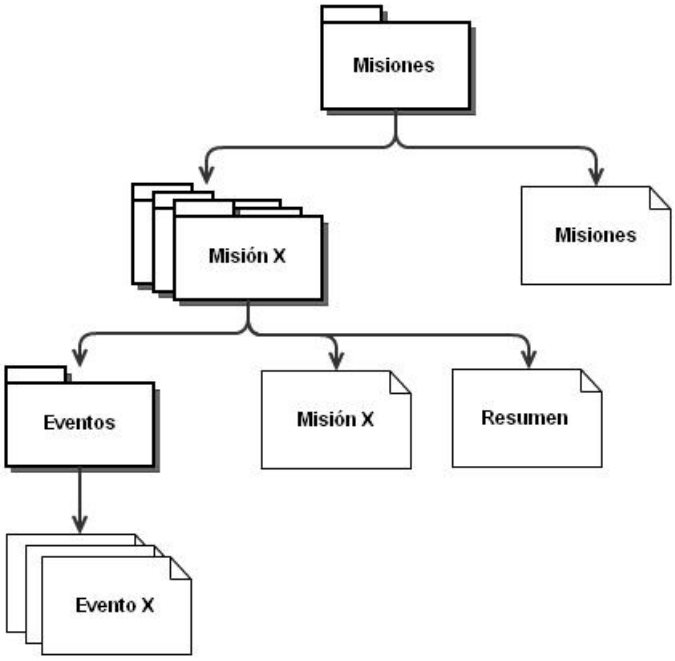

Figura 3.10. Directorio Misiones

El directorio *Eventos* incluye a los eventos que se han definido en el archivo de configuración *Misión X,* y tienen un archivo de configuración asociado con el mismo nombre definido en el archivo *Misión X.* Como ejemplo llamaremos a este archivo *Evento X* que cuenta con información asociada al evento, como tipo de evento, archivo de configuración asociado, impacto en los atributos del personaje que activa el evento, zona de activación del evento y eventos asociados. Este archivo de configuración es más complejo a los ya vistos, es por ello que se explicará en mayor detalle el contenido de *Evento X.*

Si se tratara de un evento de guión, el Evento X sería del tipo guión, tendría asociado un archivo de guión que se encuentra en el directorio Guiones, tendría una zona de activación especifica en el escenario que activa este evento, dado que este tipo de evento es de guión la zona de activación estaría cerca a un personaje del escenario

con el que se podría interactuar, por último tendría un evento asociado. Este evento asociado sería el próximo en activarse si el jugador continua en la línea de la misión.

4. Capítulo 4: Construcción

En el presente capítulo se definirán aspectos relacionados a la codificación de la solución y las herramientas utilizadas para su propósito. Además se expondrá el catálogo de pruebas del software.

4.1. Construcción de la solución

En este apartado se definirán las tecnologías utilizadas (lenguaje de programación, entorno de programación y librerías usadas), los estándares de programación y el patrón de programación usado.

4.1.1. Tecnologías de programación

Se ha optado por usar el lenguaje de programación Java por las principales razones:

- Es independiente de la plataforma donde se ejecute. (Oracle 2012)
- Debido a la extensa información que se puede encontrar sobre el uso de este lenguaje, códigos reutilizables, librerías, etc. Además existen comunidades que se enfocan en el desarrollo usando este lenguaje.
- Es un lenguaje orientado a objetos, lo que facilitará la modulación de la solución y generación de código reutilizable.
- Es un lenguaje libre, es decir no tiene costo de licencia de uso lo que permitirá disminuir costos de implementación.
- Provee del soporte de Garbage Collection, es decir el reciclaje y manejo de memoria será automatizado.
- Utiliza el concepto de especificación de excepciones, lo que permitirá al desarrollador identificar irregularidades de ejecución.

Se ha seleccionado NetBeans IDE como entorno de programación debido a las funcionalidades que ofrece, entre las razones que se consideraron se encuentran:

- Puede ser instalado en plataformas como Windows, Linux, Mac OS, entre otras.
- Permite al usuario la instalación de plugins y librerías de manera sencilla.
- Ofrece documentación completa sobre sus funcionalidades, además posee gran cantidad de usuarios, lo que permite que se tenga información en diversas comunidades de desarrollo en NetBeans.
- Es libre, lo que permite acceder a él sin mayores dificultades.
- Contiene un gestor de versiones integrado, refactorización de código, navegación de código, auto-completar, revisión sintáctica, exportación de diagramas de clases, entre otras funcionalidades.

Entre las herramientas que se han usado se encuentran:

- **JDK (Java Development Kit)**, esta herramienta incluye el entorno en tiempo de ejecución de Java, el compilador Java y los API de Java. (Oracle 2012)
- La librería **XStream**, que permite convertir objetos en archivos planos .XML y viceversa.
- **JMF (Java Media Framework)**, esta librería te permite ejecutar sonido y video en aplicaciones Java en diferentes formatos.

4.1.2. Estándares de programación

Para la codificación de la solución se ha propuesto un estándar de programación, que se define a continuación:

a) Consideraciones generales

La indentación del código será la propuesta por Netbeans, debido a que este entorno de programación ofrece una funcionalidad de ordenamiento de código que respeta el estilo Java. Las llaves de apertura de código se encontrarán en la misma línea de la instrucción inicial, y la llave de cierre estará a la altura del inicio de la misma instrucción. A continuación se presenta un ejemplo:

```
public static void Ejemplo(int x) {
    //Implementación.
}
```

Figura 4.1. Indentación de código

Se utilizarán los métodos internos de la clase ante una edición o consulta de una variable desde el exterior del ámbito de la misma, con el fin de mantener el código ante un eventual cambio interno. Es decir, se podrán acceder a los valores de la instancia de una variable mediante métodos get/set, por citar un ejemplo, para evitar la manipulación externa no controlada.

Se usará comentarios para especificar las funcionalidades implementadas. Se evitará la redundancia de comentarios y se respetará la ortografía, además se evitará tener código comentado a menos que sea de importancia. Como estándar de especificación de código se usará por el propuesto por Javadoc. En este estándar se puede apreciar el uso de descripciones de método, el uso de identificador de autor, fecha, versión, especificación de parámetros, excepciones, entre otros (Javadoc 2012). En la figura 4.2 se muestra un ejemplo.

```
/**
* Retorna el cuadrado del número ingresado
* Se devuelve el cuadrado de numeroInt2
*
* @param   numeroInt2    un número base
* @return  numeroInt1    el cuadrado del número base
* @author  piero         el nombre del autor
*/
public Integer getInteget(Integer numeroInt2) {
    Integer numeroInt1 = 0;
    numeroInt1 = numeroInt2 * numeroInt2;
    return numeroInt2;
}
```

Figura 4.2. Comentarios de código

b) Estándares de nombramiento

En la siguiente tabla se muestran los casos generales de nombramiento:

Tipo de Variable	Estándar	Ejemplo
String	strVariable	strLinea
Integer	intVariable	intCantidad
Boolean	boolVariable	boolResponder
Lista	listVariable	listPersonajes
Arreglo	arrVariable	arrStrVestiduras
Float	floVariable	floDenominador
Double	douVariable	douNumerador

Tabla 4.1. Estándar de nombramiento

Para las variables globales se detallarán en letras mayúsculas como el siguiente caso:

String [ ] STRDIRECCIONES;

Para el nombramiento de instancias de clases definidas para la solución que contengan solo una palabra (clase simple), se tomará una vocal y dos consonantes como se muestra en los ejemplos a continuación:

Clase Evento:	evt
Clase Héroe:	her
Clase Personaje:	per

Para el nombramiento de instancias de clases definidas para la solución que contengan más de una palabra (clase compuesta), se tomará la primera letra de la primera palabra que lo compone y se usará el formato de nombramiento de la clase simple para la segunda palabra, como se muestra en los ejemplos a continuación:

Clase IndigenaHonda:	iHon
Clase SoldadoCaballeria:	sCab
Clase GestorNivel:	gNiv

Para asignar las vocales o consonantes del formato de clase simple se debe aplicar el criterio y no ceñirse a la asignación de la primera vocal y las dos primeras consonantes.

c) Patrón de abstracción

Como patrón de abstracción se usará una adaptación de la arquitectura propuesta por Doherty, donde se tendrán los componentes de la solución en forma de paquetes. Donde la simulación contendrá la implementación de la lógica de juego, control de los eventos, la inteligencia artificial y las herramientas que controlan la interacción y movimiento de los personajes. El gestor de datos contendrá el gestor de misiones, escenarios, niveles y configuración de cambios de nivel. El sistema de objetos contendrá las estructuras y clases del juego, como los personajes, los batallones y bandos. La encapsulación que facilita Java, nos permitirá heredar clases como por ejemplo la de personaje, donde las clases que representan a los distintos tipos de personaje podrán tenerla como padre. Por último el motor de juegos contendrá la implementación de interfaces del videojuego, el manejo de audio y video, el dibujador de interfaz y las ventanas del videojuego.

Cabe resaltar que el alcance de esta tesis no incluye el manejo gráfico de interfaces, ni la inteligencia artificial de interacción y movimiento de personajes.

4.2. Pruebas

En este apartado se definirá lo relacionado a las pruebas realizadas con la finalidad de garantizar la obtención de un producto que cumpla con las exigencias especificadas. Se definirá el tipo de prueba realizado y la estrategia de pruebas utilizada. Además se presentarán los casos de prueba más significativos.

4.2.1. Pruebas automatizadas

La metodología del presente proyecto, Crystal Clear, recomienda el uso de pruebas automatizadas para la verificación del producto, con la finalidad disminuir la probabilidad de fallos de la aplicación y avaluar los resultados arrojados, es decir es una herramienta que permite al desarrollador determinar cual podría ser el error de la aplicación en un determinado escenario.

Las pruebas automatizadas son implementadas en código y no como un caso de prueba, donde el usuario debe seguir pasos específicos para comprobar una funcionalidad de la aplicación. Los beneficios que trae el usar este tipo de pruebas se detallan a continuación:

- Facilitan la ejecución de pruebas de regresión, debido a que se puede determinar un error por cambio en el código al evaluar resultados de una prueba automatizada ejecutada previa al cambio y posterior al mismo, identificando con mayor rapidez el error.

- Las pruebas automatizas no requieren de un usuario para que puedan ser ejecutadas, además pueden realizarse numerosas veces y en un tiempo reducido, lo que disminuye el esfuerzo que se requiere para probar una funcionalidad de la aplicación.

- Las pruebas automatizadas ejecutan precisamente las mismas operaciones cada vez que son usadas, de tal manera que poseen mayor fiabilidad al eliminar el error humano.

4.2.2. Estrategia de pruebas

Cockburn no especifica en qué momento deben realizarse las pruebas automatizadas, sin embargo nos propone dos alternativas: al finalizar la codificación y en paralelo. La primera alternativa es la más tradicional ya que los programadores y testers suelen escribir sus pruebas después de que el código ya fue escrito, sin embargo no tienen suficiente energía como para escribirlas al final (Cockburn 2004). Es por ello que la segunda alternativa cobra sentido, el desarrollo guiado por pruebas de software (TDD, test-driven development sus siglas en inglés).

El presente proyecto inicialmente realizó sus pruebas al final de la codificación de una clase o un módulo, sin embargo con la refactorización de código y la implementación de nuevas funcionalidades se decidió adoptar la práctica del TDD, que permite escribir pruebas para refactorizar el código como práctica de desarrollo (Beck 2003).

Por lo expuesto anteriormente se ha decidido usar JUNIT, un marco de trabajo de pruebas para el lenguaje de programación Java, que nos permitirá realizar las pruebas automáticas a la aplicación desarrollada en el presente proyecto. JUNIT puede integrarse al IDE seleccionado, facilitando nuestra implementación de nuestros casos de prueba.

JUNIT cuenta con herramientas que nos permiten implementar nuestras pruebas con mayor facilidad entre las que se encuentran:

- Al definir un método de una clase de prueba se antepone la notación *@Test*, para identificar que se trata de un método que probará una funcionalidad.

- Proporciona la clase *Assert* que tiene métodos estáticos que nos permite establecer afirmaciones que deben cumplirse en la prueba para que esta pueda ser catalogada como exitosa.

- La notación *@Before y @After* nos permite definir métodos que pueden especificar rutinas que deben ejecutarse antes de una prueba o después de una prueba correspondientemente.

- La notación *@RunWith y @SuiteClasses,* la primera identifica un conjunto que de pruebas que serán ejecutadas una a una, y la segunda es la notación que contiene dichas pruebas.

El uso de esta conjunto de librerías podrá ser visualizado en el siguiente apartado donde se expone alguno de los casos de pruebas más importantes del proyecto.

4.2.3. Casos de prueba

A continuación se presentan algunos de los casos de prueba más significativos de la solución. El plan de pruebas completo se encuentra en los anexos de este documento, donde se exponen los resultados obtenidos como consecuencia de su ejecución.

Módulo	Nombre de Prueba	Descripción
Juego	testVerificaEvento()	Verifica la zona de activación del evento o el personaje de activación del mismo
Juego	testRestauraMana()	Se restauran los puntos de magia de los personajes que poseen dicha característica.
Juego	testRevisaAcceso()	Se revisan los accesos de cada nivel y subnivel en tiempo de ejecución.
Personaje	testCurar()	Se alteran los puntos de vida de la unidad curada y se restan los puntos de magia de la curadora.
Personaje	testAgregaObjeto()	Se agregan objetos al inventario de un personaje.
Evento	testGetObjetosPremio()	Se obtienen los objetos que se podrán ganar al pasar el evento
Evento	testGetEventosRelacionados()	Se obtienen los eventos relacionados al evento para su ejecución.
Escenarios	testGetCelda()	Se obtiene el contenido de una celda especifica del mapa
Escenarios	testAactualizaMapa()	Se actualiza el escenario con los nuevos estados de celda

Tabla 4.2. Extracto de Plan de Pruebas

5. Capítulo 5: Observaciones, conclusiones y recomendaciones

En el presente capítulo se presentan las observaciones más importantes del trabajo realizado, además se detallarán las conclusiones determinadas según los objetivos propuestos y los resultados esperados. Finalmente se evaluará la escalabilidad de la solución mediante propuestas y recomendaciones para nuevas funcionalidades y trabajos futuros.

5.1. Observaciones

Como se ha mencionado a lo largo del documento, la solución propuesta resuelve parcialmente el desarrollo de un videojuego de naturaleza híbrida RPG/RTS, centrándose en la implementación de la lógica del juego y al control de la secuencia principal del videojuego, según reglas determinadas y previamente analizadas.

Con la finalidad de probar la viabilidad de la solución y la satisfacción de las exigencias que compete el desarrollo de un videojuego que cuente con características RPG/RTS, se ha visto adecuado emplear lo desarrollado en la implementación de un videojuego propuesto por Avatar PUCP, grupo multidisciplinar perteneciente a la Universidad Católica del Perú (PUCP) que cuenta con experiencia en desarrollo, empleo e investigación referida a entornos virtuales 3D, videojuegos y nuevas tecnologías (Avatar 2012). El videojuego propuesto por el grupo Avatar es parte de una actividad conmemorativa por el bicentenario de la rebelión del Cuzco en 1814 y pretender ser una herramienta de apoyo y motivación al proceso de aprendizaje de la historia de la independencia del Perú.

El lenguaje de programación seleccionado ha permitido implementar lo definido en el diseño de la solución, facilitando la abstracción y encapsulamiento de funcionalidades. Además los estándares de programación del lenguaje ha facilitado la documentación del código desarrollado.

El uso de una metodología ágil como Crystal Clear ha permitido que el proyecto sea flexible en la elección del contenido de los entregables, ha promovido la reflexión de cada uno de ellos y ha permitido el monitoreo constante del proyecto.

5.2. Conclusiones

Al culminar el proyecto se puede concluir que se han adaptado de manera adecuada los elementos de géneros de rol y estrategia en tiempo real a la solución planteada, proponiendo mecanismos lógicos de gestión de elementos (nivel, personajes, escenarios) y secuencia lógica (objetivos, misiones, eventos de interacción) del videojuego.

La arquitectura propuesta por Michael Doherty ha sido de gran utilidad para definir la arquitectura del presente proyecto. Se podría decir que se ha probado como exitosa al finalizar la implementación y al comprobar su naturaleza escalable, que se adecua a la solución propuesta.

El manejo de eventos implementado en la solución propone el aditamento de nuevas reglas de juego. Su diseño fue planteado de manera que permita crear nuevos tipos de eventos con los que el usuario pueda interactuar con el jugador. La escalabilidad de la solución se detallará en el apartado 5.3, sin embargo se puede indicar la acertada disposición del software para ampliar su catálogo de funcionalidades.

La solución planteada permite configurar el videojuego mediante archivos planos, es decir que la complejidad y la duración de juego dependerá solo del usuario que configure los niveles y objetivos del mismo. El software es capaz de procesar un número de niveles de juego solo limitado por la configuración que se le plantee.

La solución se ha integrado al videojuego base de manera adecuada debido a la definición pertinente de las componentes de alto nivel. El nivel de abstracción con la que fue implementada permite que la solución sea escalable en el tiempo. Las recomendaciones y trabajos futuros se definen en mayor detalle en el siguiente apartado.

5.3. Recomendaciones y trabajos futuros

Se debe aclarar que la solución presentada en el presente documento será complementada por otros dos proyectos de la misma naturaleza (proyecto de fin de carrera). Uno se enfocará en la implementación del dibujado de la interfaz y el manejo gráfico del videojuego y el otro en la implementación de la interacción y movimiento del

personaje (inteligencia artificial). Se considera que el presente trabajo sirva como base para los proyectos mencionados.

Durante la implementación de la solución se ha observado que el archivo de configuración de la distribución lógica de los escenarios (Figura 3.6) presenta un llenado engorroso. Es por ello que se recomendaría como un trabajo futuro la implementación de un editor de escenarios, donde se facilite el diseño de la distribución lógica y los elementos internos de un escenario mediante una herramienta interactiva que tenga como salida un archivo de configuración similar al mostrado en la figura 3.6.

Un caso similar se presenta con los archivos de configuración de niveles y eventos, ya que mientras más complejo sea el diseño del nivel, más difícil será para el usuario mantener una gestión de sus archivos. Es por ello que se recomendaría como trabajo futuro, una herramienta que sirva como intermediario entre el gestor de datos y el sistema de directorios. Esta herramienta le permitiría crear un nivel a un usuario, y asociar los eventos, diseñar los guiones y decisiones, entre otras funcionalidades, de una manera más interactiva.

El código de la solución se ha escrito en el idioma español, sin embargo se propone como recomendación la traducción de código al idioma inglés, debido a que la solución está planteada para ser reutilizada en juegos de la misma naturaleza y orientada a su mejora continua. Para la descripción de métodos y comentarios de código se recomienda aplicar el mismo procedimiento.

6. Capítulo 6: Bibliografía

AVATAR PUCP

2012 “Grupo Avatar Pucp”. Lima, Perú. Consulta: 10 de Abril de 2012 <http://avatar.inf.pucp.edu.pe/index.php?seccion=quienes&id=0>

ZIMMERMAN, Eric (autor) y SALEN, Katia (autor)

2003 Rules of Play: Game design fundamentals. Edición 2003, Publicación: The MIT Press | ISBN: 0262240459

ESPOSITO, Nicolas

2005 A Short and Simple Definition of What a Videogame Is. Vancouver: University of Vancouver

MURRAY, Janet H.

1999 Hamlet on the Holodeck: The Future of Narrative in Cyberspace. Edición 1999. Publicación: Cambridge (MA): MIT Press.

ADAMS, Dan

2006 The state of the RTS. IGN articles. Consulta: 12 de Junio del 2012. <http://pc.ign.com/articles/700/700747p1.html>

ELDRIDGE, Justin

2012 The definition of turn-based strategy games whit game examples. Examiner articles. Consulta: 13 de Junio del 2012. <http://www.examiner.com/article/the-definition-of-turn-based-strategy-games-with-game-examples>

KEH, David

2002 Westwood: The Building of a Legacy. Stanford: University of Stanford. Marzo de 2002. SUID: 4832754.

PEINADO, Federico

2012 Diseño – Niveles [Diapositivas]. Madrid. Universidad Complutense de Madrid. Consulta: 13 de Junio de 2012. <http://www.fdi.ucm.es/profesor/fpeinado/courses/gamedesign/gamedesign_levels_es.pdf>

RAWLINS, Gregory J. E.

2012 Game interfaces. Indiana University. Bloomington. Consulta: 13 de Junio de 2012. <http://www.cs.indiana.edu/rawlins/website/interface/game.html>

CRAWFORD, Chris

1990 The art of human-computer interface design: Lessons from Computer Game Design. Editorial Addison – Wesley.

DAVISON, Andrew

2005 Killer Game Programming In Java. Editorial: O Reilly & Associates. ISBN: 9780596007300.

BLIZZARD ENTERTAINMENT

2002 Warcraft:Reign of Chaos. Consulta: 14 de Junio del 2012. <http://us.blizzard.com/en-us/games/war3/>

STARCRAFTII

2010 StarCraft2. Consulta: 14 de Junio del 2012. <http://us.battle.net/sc2/es/>

DIABLOIII

2010 Diablo3. Consulta: 14 de Junio del 2012. <http://eu.battle.net/d3/es/>

BALDUR'S GATE

2013 Baldur's Gate. Consulta: 10 de Mayo del 2013. <http://www.baldursgate.com/>

DUNE

2013 Dune. Consulta: 10 de Mayo del 2013. <http://dune2k.com>

MICROSOFT GAMES

1999 Age of empires: The age of Kings. Consulta: 14 de Junio del 2012. <http://www.microsoft.com/games/age2/>

ROYCE, Winston. W.

1970 Managing the Development of Large Software Systems: Concepts and Techniques. Proceedings, IEEE WESCON.

COCKBURN, Alistair

2004 Crystal Clear: A Human-Powered Methodology for Small Teams. Primera edición. Stoughton: Pearson Education.

PRENSKY, Mark

2001 Digital game-based learning. Original de University of California. McGraw-Hill. ISBN 0071363440, 9780071363440.

BUCKLAND, Mat

2005 Programming Game AI by Example. Wordware publishing. ISBN 1-55622-078-2

DOHERTY, Michael

2003 A software architecture for games. University of the Pacific Department of Computer Science Research and Project Journal (RAPJ), vol. 1, no. 1, 2003.

ORACLE
2012 Oracle. Consulta: 20 de Noviembre de 2012. <http://www.oracle.com/technetwork/topics/newtojava/overview/index.html>

JAVADOC
2012 How to Write Doc Comments for the Javadoc Tool. Consultado: 20 de Noviembre de 2012. <http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html>

BECK, Kent
2003 Test Driven Development: By Example. Editorial Addison-Wesley Professional. ISBN-13: 978-0321146533