Universidad de las Ciencias Informáticas

Facultad 4

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Módulo genérico para el desarrollo de videojuegos TRPG en Unity.

Autor:

Miguel Antonio Hernández Bormey

Tutora:

MSc. Liudmila Pupo Peña

Co-tutor:

Ing. Jorge Lázaro Sirés

La Habana, junio de 2018.

"Año 60 de la Revolución"

Nintendo®

En mi targeta de presentación, soy presidente coorporativo. En mi mente, soy desarrollador. Pero en mi corazón, soy un gamer.

Satoru Iwata

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año ____.

_____

Miguel Antonio Hernández Bormey

(Autor)

_____ _____

MSc. Liudmila Pupo Peña

(Tutora)

Ing. Jorge Lázaro Sirés

(Co-tutor)

Datos de contacto

Autor:

Miguel Antonio Hernández Bormey

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: mabormey@estudiantes.uci.cu

Tutora:

MSc. Liudmila Pupo Peña

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: lpupo@uci.cu

Co-tutor:

Ing. Jorge Lázaro Sirés

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: jlsires@uci.cu

A mi madre que siempre me ha apoyado, cuidado y querido sin importar lo difícil que haya sido. Por apoyar mis sueños y enseñarme que con esfuerzo se puede lograr todo.

A Carlos, por ser una figura paterna, un amigo y un maestro que me entiende y guía en todo momento. Por demostrarme que siempre existe más de un camino y de que uno debe de estar preparado para los improvistos de la vida.

A mi abuela por siempre brindarme su amor y apoyo incondicional por tantos años.

A mi tía Betty, que sin su ayuda y consejos no fuese posible salir adelante y lograrlo, a pesar de la distancia y el tiempo.

A Bárbaro, por ser un amigo sin igual e irremplazable, que comparte sus puntos de vistas e ideas de forma sincera en todo momento, sin importar que tan dura pueda llegar a ser la verdad.

Después de cinco años de esfuerzo y dedicación, el haber logrado una hazaña como el desarrollo de una tesis como esta, resulta gratificante su conclusión. Sin embargo, al recordar el trayecto recorrido para lograrlo, puedo observar todo lo logrado durante ese tiempo, las personas conocidas, las amistades realizadas, los disgustos encontrados y los conocimientos adquiridos. Aun así, el haber alcanzado tales resultados no hubiese sido posible sin la colaboración de varias personas a las cuales les deseo dedicar este pequeño espacio para agradecer su apoyo.

A la persona que me trajo al mundo, que sufrió y se esforzó en todo momento el criarme y educarme sola; mi madre. Muchas veces he discutido y estado en desacuerdo con ella, pero no importa lo que yo haya dicho, porque para mí ella es única. Desde que me volví un adulto, siempre estuvo en mi conciencia todo lo que ella ha hecho por mí y que mejor manera de agradecérselo, que dándole el placer de que su hijo se gradué y se vuelva ingeniero. Muchas gracias por tu amor y apoyo incondicional, por aguantar siempre mi manera de ser, por criarme y educarme tu sola, por todo lo que has hecho por mí.

A mi padrastro, que a pesar de que no estamos relacionados por sangre, para mí es lo más cercano a un padre. Gracias por enseñarme las dificultades de la vida y de como siempre debo de intentar de buscar una solución a los problemas. Por cuidarme y regañarme cuando hizo falta, a pesar de no ser tu hijo. Por compartir siempre tu opinión a mis preguntas e ideas. Gracias por estar presente y por todo.

Agradecer a mi tutora y cotutor, que a pesar de los imprevistos y el poco tiempo que compartimos, me han ayudado a llegar hasta el final.

A Bárbaro y Yusnelys, por ser unos amigos siempre tan amables y bondadosos a pesar de las molestias que yo pueda haberles provocado. El haber llegado hasta aquí no fuese posible sin su

ayuda y sinceridad. A los dos los considero como unos amigos irremplazables. Espero que podamos seguir trabajando juntos creando nuevas ideas y proyectos.

A mi tía Betty, que sin ella no fuese sido posible haber llegado tan lejos. Gracias por tu constante apoyo y preocupación por mí, a pesar de la distancia y de lo complicada que es tu vida.

Gracias a todas esas personas que de alguna han contribuido hacer como lo que soy y como soy actualmente.

Resumen

Los videojuegos forman parte de la cultura universal en la actualidad, siendo uno de los mayores mercados que se destaca principalmente en la industria del entretenimiento. En Cuba se han dado pequeños pasos de avance en el desarrollo de esta industria. La gama de géneros de videojuegos desarrollados hasta el momento es muy reducida, debido a la poca disponibilidad de recursos gratis para ello. En base a lo planteado, surge la necesidad de desarrollar medios que permitan la elaboración de videojuegos, para aumentar la disponibilidad y reducir la complejidad que implica trabajar en algún género en específico.

La presente investigación tiene como objetivo la creación de un módulo que facilite el desarrollo de videojuegos del género: Juegos de Rol Táctico (TRPG, por sus siglas en inglés). La propuesta de solución brinda la posibilidad de un medio para la elaboración de videojuegos de una dificultad de desarrollo elevada. Para reducir la complejidad, se aportan los principales sistemas que lo definen y un conjunto de componentes que facilitan la creación de diferentes recursos, donde la mayoría serán usados mediante el arrastrar y soltar sobre el objeto deseado. Como resultado, se obtuvo un prototipo de módulo genérico que permite la creación de terrenos y recursos configurables y los sistemas bases que definen a los TRPG, como el de combate y el desplazamiento. También se obtiene un medio que permite ampliar la gama de géneros posibles a desarrollar, dando un paso más para alcanzar la creación de una industria de videojuegos en el país.

Palabras clave: componentes, módulos, RPG, TRPG, Unity, videojuegos.

Índice

Índice de figuras

Índice de tablas

Introducción

Los videojuegos son actualmente parte de la cultura universal, la mayoría se utilizan para el entretenimiento visual o como herramienta educativa. La clasificación de los mismos se divide por géneros, siendo los Juegos de Rol (RPG, por sus siglas en inglés) uno de los más populares y vendidos en el mundo según las estadísticas que provienen de páginas como App Annie, Statista y Metacritic (1) y (2).

Existe hoy en día gran confusión al catalogar los videojuegos RPG; en consecuencia, se han realizado muchos debates donde las personas plantean sus opiniones desde distintos puntos de vista. Muchos opinan que existen tres condiciones que deberían cumplirse para que un videojuego pueda ser considerado de rol: la creación del personaje, la evolución del personaje y la libertad absoluta para moverse y realizar diversas acciones en un escenario (es decir, la ausencia de linealidad) (3). Tomando en cuenta estas condiciones, se plantea que, muchos videojuegos que son considerados de rol no cumplen esos parámetros, mientras que otros que no son considerados de rol, si los cumplen.

Los RPG presentan un conjunto de características principales, una de ellas es la narrativa, donde se hace énfasis en la historia. Otra es el desarrollo del personaje, donde se vuelve más fuerte y adquiere nuevas habilidades mediante la superación de obstáculos que se le presentan. La exploración y búsqueda es otra que es elemento importante del género, donde los protagonistas viajan alrededor del mundo completando misiones o encargos. Y por último el combate, que es la forma que tienen los jugadores de enfrentarse a los enemigos y la forma en que se desarrollan estos combates depende del subgénero al que pertenece el videojuego (4).

Este género presenta muchas vertientes, o también llamados subgéneros que, dependiendo de la región, las personas se decantan más por uno en específico. Entre los diferentes subgéneros que presenta, se destacan los famosos Juegos de Rol Multijugador Masivo Online (MMORPG, por sus siglas en inglés) como los más vendidos y los RPG japoneses (JRPG, por sus siglas en inglés), famosos por su belleza visual y fuerte concentración en la historia. Otros subgéneros destacados son los RPG Estratégicos (SRPG, por sus siglas en inglés) y los RPG Táctico (TRPG, por sus siglas en inglés), ambos basados en la estrategia (4).

El desarrollo de videojuegos del género RPG o de sus subgéneros, resulta bastante complejo, debido a los diversos sistemas que funcionan de forma intrínseca. Existen diversos módulos y programas para el desarrollo de videojuegos del género RPG, de forma genérica, que su gran mayoría se encuentran disponibles de forma gratuita. En cambio, hay una gran escasez de estas facilidades para el desarrollo de algunos de sus subgéneros, siendo el género TRPG uno de los más afectados ya que los mejores módulos y herramientas son privados, mientras que las libres, presentan grandes cantidades de errores y poca calidad. A la vez, existe el problema de que estos recursos se encargan de desarrollar un área del sistema

en específico, surgiendo la necesidad de usar en ocasiones más de uno. Esto trae como consecuencia una gran diversidad de errores, producto a que la mayoría de estas herramientas que se utilizan no son compatibles entre sí, evitando que exista un flujo de trabajo estable.

Muchas compañías de videojuegos, con el objetivo de facilitar el trabajo a los desarrolladores, crean softwares o conjuntos de componentes para el desarrollo de un género especifico, que pueden ser reutilizados cada vez que sea necesario para crear un nuevo videojuego. De esta manera se acelera y facilita la producción, a la vez que se mantiene un flujo estable de trabajo. Estas herramientas, por lo general, son solo para uso exclusivo de la compañía que la desarrolla.

Cuba se encuentra actualmente en un proceso de industria emergente, donde se desea lograr una industria de videojuegos. Producto a la colaboración de los Estudios de Animación ICAIC (Instituto Cubano del Arte e Industria Cinematográficos) y el Centro de Entornos Interactivos 3D[1] (Vertex), perteneciente a la Facultad 4 de la Universidad de las Ciencias Informáticas (UCI), se han logrado algunos avances, obteniéndose productos como La Superclaria, La Chivichana, Sagua La Aldea Embrujada, Especies Invasoras y La Neurona. Siguiendo este ejemplo, muchos interesados en el país han empezado a desarrollar por su cuenta sus propias ideas de videojuegos, aumentado vertiginosamente la cantidad de personas interesadas cada año. Este auge se puede evidenciar mediante la realización de eventos como el *Global Game Jam* y el *Pachamama Game Jam*, ambos eventos desarrollados en la UCI.

Existe todavía una gran gama de géneros de videojuegos que el centro Vertex no ha trabajado, uno de ellos son los TRPG, género en el que se desea incursionar. El centro no ha podido desarrollar ningún proyecto alrededor de la temática planteada debido a la escaza existencia de componentes o softwares de forma gratuita que faciliten el trabajo. También por la incompatibilidad y los errores que se producen al querer utilizar módulos o softwares de terceros. De igual manera sucede con la dificultad que conlleva el desarrollo de un videojuego de este tipo desde cero. Dichos problemas impiden desarrollar videojuegos TRPG, lo que dificulta ampliar el repertorio de proyectos a desarrollar.

En base a lo planteado anteriormente, se define como **problema a resolver:** ¿Cómo viabilizar el desarrollo de videojuegos de tipo TRPG para el centro Vertex de la UCI, sobre la plataforma Unity?

Se plantea como **objeto de estudio:** desarrollo de videojuegos. Se establece como **objetivo general** de la investigación: Implementar un módulo que permita viabilizar el desarrollo de videojuegos del tipo TRPG para el centro Vertex de la UCI, sobre la plataforma Unity. Se define como **campo de acción:** desarrollo de videojuegos de tipo TRPG.

[1] Tres dimensiones

Para dar solución al objetivo planteado, se definen las siguientes **tareas de la investigación:**

- Análisis del proceso de desarrollo de videojuegos; para sentar las bases de la investigación en curso.
- Análisis de la estructura de los RPG y TRPG; para comprender los sistemas que los integran y su funcionamiento.
- Investigación del proceso de desarrollo de módulos y componentes en Unity; para realizar un correcto desarrollo de la solución.
- Estudio de los elementos correspondientes a la metodología de desarrollo de software seleccionada con el objetivo de llevar a cabo un desarrollo organizado.
- Caracterización de las herramientas y tecnologías seleccionadas para permitir una correcta implementación del sistema.
- Implementación de las funcionalidades necesarias para dar solución a la problemática planteada.
- Aplicación de las diferentes pruebas propuestas por la metodología escogida para validar el correcto funcionamiento del módulo a desarrollar.

Para la investigación científica se utilizaron los métodos teóricos y empíricos:

Métodos teóricos:

- **Analítico-Sintético**: Permite realizar el estudio teórico de la investigación, facilitando el análisis de documentos y la extracción de información relevante para el desarrollo de videojuegos y módulos.
- **Histórico-Lógico**: Método que permite estudiar todo lo relacionado con el proceso de desarrollo de videojuegos, módulos, RPG y TRPG; para obtener un conocimiento histórico de su evolución y comportamiento a nivel nacional e internacional.
- **Modelación**: Utilizado para la modelación de diagramas, representación del proceso de desarrollo y permitir un mejor entendimiento de la solución a implementar.

Métodos empíricos:

- **Medición**: El sistema obtenido como resultado de la investigación será medido a través las pruebas unitarias y de aceptación garantizando su correcto funcionamiento.

En pos de estructurar el proceso de desarrollo del presente trabajo de una manera organizada y coherente, se dividió el mismo en tres capítulos:

Capítulo 1. Fundamentación Teórica: Se abordan los elementos teóricos más importantes que sirven de base para la realización de la investigación; a la vez, se proporciona una descripción de las herramientas, tecnologías y metodología de desarrollo de software a utilizar para dar solución al problema planteado.

Capítulo 2. Propuesta de solución. Exploración, planificación y diseño: Se describe la propuesta de solución creada. Se detallan las funcionalidades y características que tendrá la propuesta del sistema, lo que posibilitará un mayor entendimiento del mismo. Se especifican las historias de usuario definidas, que recogen todas las funcionalidades a implementar. Se muestran además otros artefactos propios de la metodología de desarrollo, que se generan en estas etapas.

Capítulo 3. Implementación y pruebas: Capítulo donde se describen los detalles correspondientes relacionados a la implementación del sistema. Se exponen las tareas de programación o ingenierías generadas por cada historia de usuario. Se observan las pruebas realizadas al sistema, así como los principales resultados obtenidos.

Capítulo 1: Fundamentación teórica

En el presente capítulo se realiza un análisis de diferentes herramientas y procesos para el desarrollo de videojuegos. Se describen los conceptos fundamentales relacionados con el dominio del problema y se realiza un análisis de las posibles soluciones. Se hace una investigación sobre el origen de los RPG, características que definen a los TRPG y el desarrollo de los videojuegos. Se analizan algunas de las tecnologías, metodologías, lenguajes y herramientas posibles a utilizar para el desarrollo de la solución.

1.1 Historia de los RPG

El género RPG nació para imitar en el mundo de los videojuegos a los populares juegos de rol de mesa. En dichos juegos de rol de mesa (o de tablero), el DJ o director del juego (más conocido popularmente como *master*) era el encargado de plantear la aventura al grupo de PJ (personajes jugadores) y de hacer cumplir las reglas, basándose en los correspondientes libros y tablas y también en su sentido común. Los jugadores asumían el rol de un personaje que iba mejorando sus atributos y habilidades mediante la experiencia. Los personajes que no manejaban los jugadores y enemigos eran controlados por el director del juego. Se considera que los videojuegos de rol jamás podrán acercarse a la libertad y a las inmensas posibilidades que ofrecen los de tablero, cuyos únicos límites están en la imaginación de los jugadores. Los jugadores pueden hacer lo que quieran (siempre y cuando las tiradas de dados les fueran favorables) e ir a donde lo considerasen oportuno para resolver la aventura (el director del juego debía ser un gran improvisador además de justo árbitro) (3).

Los primeros videojuegos de género RPG nacieron en la década de los 80's con la salida de *Dungeons & Dragons*, llevado para la computadora TRS-80. El juego tuvo características únicas para esas fechas, las cuales hoy día son fundamentales en el género, tales como un analizador sintáctico por línea de comandos, generación de personajes, una tienda para comprar equipamiento, combate, trampas y calabozos por explorar (5).

Para 1982 fue desarrollado el primer videojuego RPG en la Atari llamado *Dragonstomper*, luego en 1984 *Bokosuka Wars* salió a la venta con el que nacería un nuevo sub-género que se conoce como SRPG, donde los jugadores deben de avanzar en un mapa por casillas y atacar a los enemigos del bando contrario. Durante el intervalo de los años 80's y 90's los videojuegos RPG se especializaban en ser juegos para un solo jugador, no fue hasta mitad y finales de los 90's con la masificación del Internet que empezaron a nacer los MMORPG, títulos como *Runescape*, *Lineage* y *World of Warcraft* (5).

Lo que define a los RPG es fórmula clásica de la creación/optimización de un personaje, un sistema de inventario de objetos y/o habilidades de personajes, toma de decisiones, elección en los diálogos y compra-

venta de objetos (5). Tomando como base la fórmula antes mencionada, se han desarrollado muchos subgéneros de los RPG, lográndose una gran diversidad y que con el tiempo han ido evolucionando. Entre los subgéneros más destacados encontramos los RPG occidental o RPG, como comúnmente se le conoce; los RPG japonés o más conocidos como JRPG. Otros subgéneros que se destacan son: los RPG de Acción (ARPG, por sus siglas en inglés), los *roguelike*, los RPG dinámicos, los RPG de tiros (RPS, por sus siglas en inglés), los famosos MMORPG ya antes mencionado, que es un género relativamente moderno. Por último, se destacan los RPG basados en la estrategia, que se dividen en dos subcategorías, los llamados SRPG, basado más en la estrategia en tiempo real; y los TRPG, enfocado en la estrategia por turnos, basado en juegos de tableros como el ajedrez, donde el sistema de batalla combina elementos y mecánicas típicas de los RPGs con la estrategia (5) y (6).

1.2 TRPGs

Son videojuegos que se basan en el combate, en escenarios con cuadrículas, donde el valor táctico es fundamental, forzando al jugador planear una estrategia previa para poder ganar cada combate desarrollado (5). Son, a la vez, un subgénero de los RPG, principalmente por poseer elementos de los videojuegos de estrategia, incorporado de forma alternativa al tradicional sistema de los RPG. Al igual que un RPG común, el jugador controla una cantidad finita de personajes y batalla contra una cantidad similar de enemigos. Como muchos RPG, la muerte es usualmente temporal, al terminar cada batalla, los personajes ganan puntos de experiencia y aumentan sus estadísticas. También se gana una segunda experiencia que representa el tiempo para aprender una nueva habilidad, o se gastan puntos o dinero para adquirirla. Las batallas están sujetas a condiciones de victoria y de derrota, las cuales el jugador tendrá que cumplir para desbloquear o hacer disponible nuevos mapas o para avanzar en la historia (7).

Como la mayoría de los RPG, los personajes poseen un equipamiento y una clase que define su estilo de combate, que, entre batallas, se podrán cambiar accediendo al personaje. También se incluye un mundo que explorar, siendo por lo general un mundo abierto con múltiples mazmorras y varios pueblos. Los *non-player characters* (NPC) son personajes gestionados por la Inteligencia Artificial[2] (IA) o por comandos pre configurados del sistema. Los NPC pueden presentar diálogos relevantes para avanzar en la historia o desatar alguna misión después de cumplir ciertas condiciones, o para realizar la función de tiendas (7).

Entre las características que definen a los RPGs, se encuentran que existe un conjunto de estadísticas que determinan los atributos del personaje, que aumentan para volverlo más fuerte y un sistema basados en menú de batalla que utiliza el videojuego, que hace uso de estas estadísticas como base. Los TRPG

[2] En inglés sería *Artificial Intelligence System* (IA).

generalmente incluyen estos aspectos, pero también incorporan la estrategia como modo de juego, así como el movimiento táctico en una cuadrícula. Otro elemento que presentan es la premiación de la experiencia de los personajes de manera individual; esto provoca que, si se mantiene a un personaje débil alejado de la lucha, en una zona segura, el desarrollo de sus estadísticas se verá por debajo en comparación a otros personajes (8).

Antiguamente, la mayoría de los juegos de este género no incluían un modo de juego multijugador o juego en red, así como la exploración, siendo esta una característica principal común de los RPG (7). Los mismos son descendientes de los juegos de estrategia tradicionales, como el ajedrez y de los juegos de mesa de juegos de rol y de guerra, como *Dungeons & Dragons*.

En la mayoría de los juegos de este tipo la historia resulta ser la esencia del mismo, ambientada por lo general, en un mundo de fantasía, donde ocurren diferentes eventos que determinan el curso de la historia y el crecimiento del personaje principal. Muchos de los juegos de rol de hoy día contienen más de un arco argumental en la historia, donde existen más de un final, que solo se puede alcanzar dependiendo de las decisiones tomadas en momentos decisivos de la trama. Esto es un factor clave para mantener al jugador motivado a seguir jugando, permitiéndole una mayor libertad en sus decisiones, así como la necesidad de re-jugar el juego para poder ver o desbloquear otros posibles arcos en la historia (5).

1.3 Posicionamiento de la cámara en los videojuegos

El mundo visual que es observable por las personas es tridimensional, pero la mayoría de los juegos TRPG, antiguamente, eran juegos en 2D[3]. “La proyección” es, de manera sencilla, la forma “aplanada” de una vista 3D en 2D (9).

Existen varias proyecciones populares usadas en juegos 2D. La más popular es la de tener la cámara exactamente en un eje mayor, lo que significa en uno de los ejes principales que definen las dimensiones (x, y, z). Esta es muy común en los juegos de acertijos y de desplazamiento línea a línea de manera lateral, donde cada baldosa[4] es un simple cuadrado y la tercera dimensión no es visible en absoluto (9). Ver Fig. 1 Tres proyecciones comunes de un mismo objeto. Esa vista es a menudo de forma aérea, o directamente desde un lado. Si se mira un cubo en esta vista, solo un lado sería visible (el lado opuesto), como se observa en la Fig. 2 Nikki y los robots: vista de lado.

[3] Dos dimensiones

[4] Más conocido por *tile* en inglés y usado principal mente para referirse al elemento que conforma la cuadrícula.

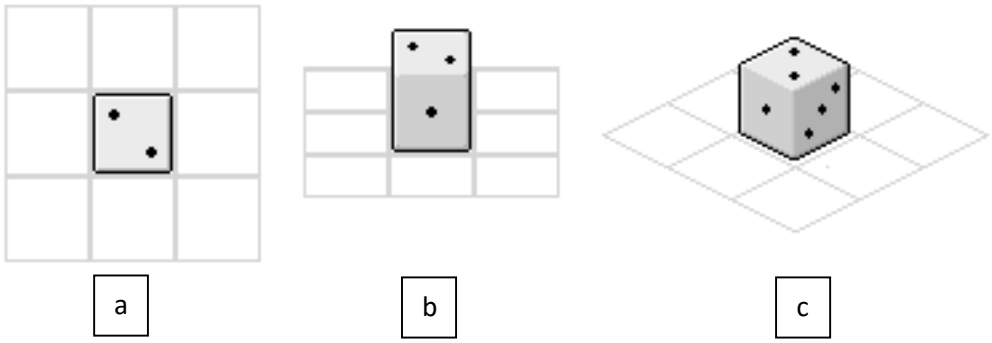


Fig. 1 Tres proyecciones comunes de un mismo objeto (tomado de (9))

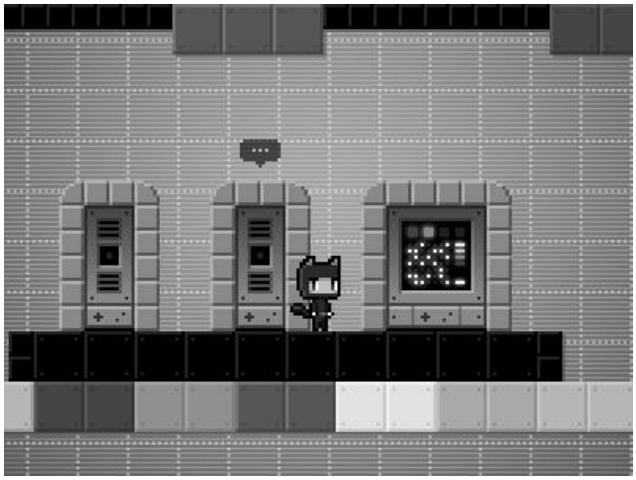


Fig. 2 Nikki y los robots: vista de lado (tomado de (9))

La siguiente proyección más común usada también usa baldosas de forma cuadrada, pero el ángulo de la cámara cambia para poder ver la tercera dimensión; ver Fig. 1 Tres proyecciones comunes de un mismo objeto. Los juegos que usan esta proyección agregan movimiento en esa tercera dimensión. Esta proyección es comúnmente vista en versiones anteriores de RPG de consola. La cámara virtual es inclinada para obtener esta vista (9). Si se mira un cubo en esta vista, dos caras son visibles (la cara superior y una cara de lado). La Fig. 3 Liberated Pixel Cup: vista superior y ladomuestra un ejemplo de este tipo de vista utilizada en los videojuegos.

Fig. 3 Liberated Pixel Cup: vista superior y lado (tomado de (9))

La última proyección es la proyección isométrica. Para la proyección isométrica se inclina la cámara a lo largo de dos ejes (se gira la cámara 45 grados hacia un lado, luego 30 grados hacia abajo.) Esto crea un diamante o rombo con forma de cuadrícula[5], donde los espacios cuadriculados están al doble de ancho y de alto; ver Fig. 1 Tres proyecciones comunes de un mismo objeto. Este estilo fue popularizado por los juegos de estrategia y los RPG de acción, como se puede observar en las figuras Fig. 4 Age of Empire II, Fig. 5 Diablo II, Fig. 6 Starcraft y Fig. 7 Starcraft II; estilo que es todavía muy utilizado por muchos videojuegos hasta el día de hoy (9). Si se viera un cubo en esta vista, tres lados son visibles (la cara superior y dos caras de los lados).

[5] Conocido como *grid* en inglés

Fig. 4 Age of Empire II (tomado de (10))

Fig. 5 Diablo II (tomado de (11))

Fig. 6 Starcraft (tomado de (12))

Fig. 7 Starcraft II (tomado de (13))

1.3.1 ¿Qué es una proyección isométrica?

Las proyecciones isométricas pertenecen a la familia de las perspectivas axonométricas; sus propiedades más importantes son, que la escala del objeto representado no depende de su distancia al observador, ver Fig. 9 Perspectiva isométrica 2, y que dos líneas paralelas en realidad son también paralelas en su representación axonométrica, ver Fig. 8 Perspectiva isométrica 1 (14). Quiere decir que los objetos no se verán afectados por la profundidad y trabajarán siempre sobre una base de 30 grados, o sea, que simplemente no existe punto de fuga.

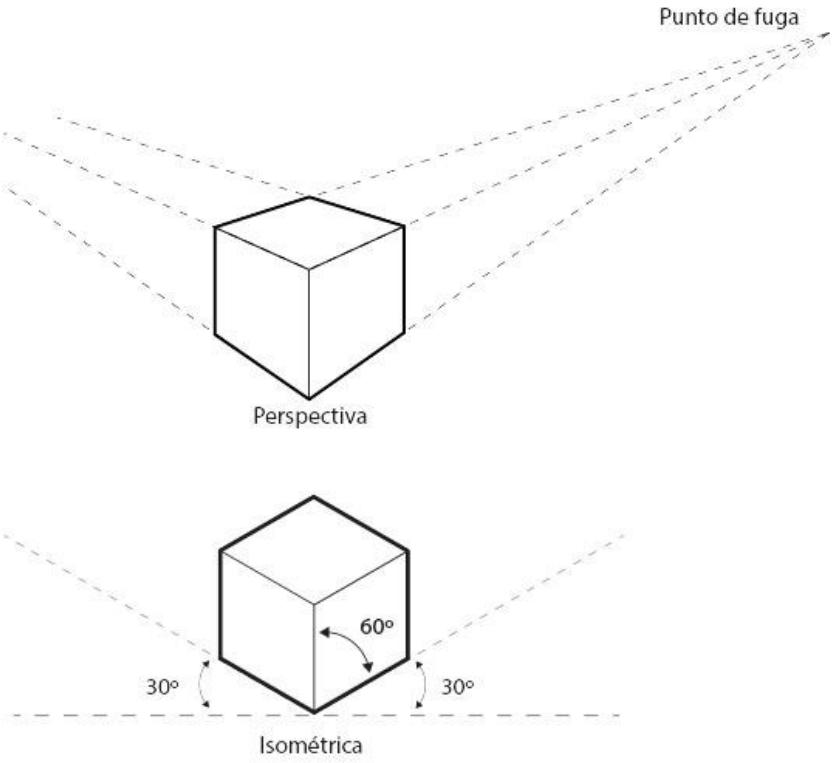


Fig. 8 Perspectiva isométrica 1 (tomado de (14))

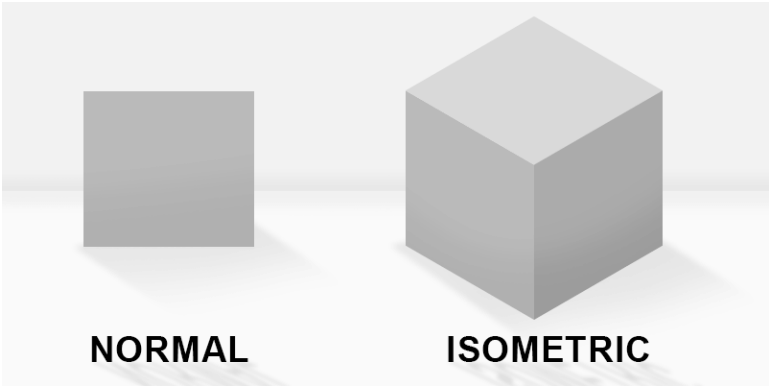


Fig. 9 Perspectiva isométrica 2 (tomado de (15))

Una de las principales ventajas de este sistema a la hora de crear videojuegos es que los objetos, al carecer de perspectiva, son fácilmente duplicables y apilables en un entorno 2D. Como se aprecia en la siguiente Fig. 10 Duplicación de objetos usando la perspectiva isométrica (tomado de (14)) (14).

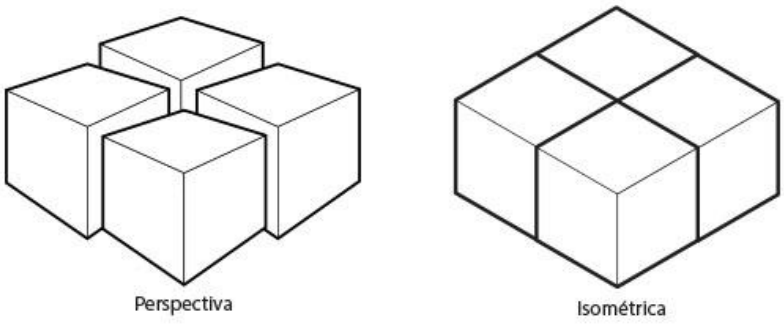


Fig. 10 Duplicación de objetos usando la perspectiva isométrica (tomado de (14))

La vista o perspectiva isométrica, es generalmente utilizada para crear una ilusión en 3D para juegos creados en 2D. Denominado con frecuencia como pseudo 3D o 2.5D[6].

1.4 Proceso de desarrollo de un videojuego

La proliferación en los últimos años de los videojuegos *indies*[7] puede haber creado la impresión de que crear un videojuego es algo sencillo y a alcance de todo el mundo. Esta impresión, sin ser del todo falsa, está bastante lejos de la realidad; porque, aunque es cierto que con suficiente perseverancia casi cualquier persona puede crear un videojuego (pero la calidad del mismo ya es un tema aparte), el desarrollo de éste está muy lejos de ser un proceso sencillo ya que existe todo un proceso científico y tecnológico detrás que hace posible la creación de estos programas interactivos (16).

El proceso en si es similar a la creación de un software en general, pero se diferencia en la cantidad de aportes creativos necesarios, entre los que se destacan: la música, historia, diseño de personajes y niveles (16). Se considera la plataforma que se tiene como objetivo para la cual se va a desarrollar, el género en el que va estar enfocado y la forma de visualización que presentará el juego ya sea 2D, 2.5D o 3D. Se destaca que al igual que el desarrollo de un software, el de un videojuego es considerado también un proceso iterativo, donde los diseñadores pasan por cada uno de los pasos de desarrollo repetidas veces con el objetivo de cambiar y mejorar cualquier aspecto, hasta que se considere que se ha obtenido un mejor resultado (16).

Entre los procesos que se realizan para el desarrollo de un videojuego se encuentran:

[6] Perspectiva isométrica
[7] Videojuego independiente

- **Concepción de la idea del videojuego:** cuando se tiene una idea inicial en mente comienza esta etapa, en la que se plantean los aspectos fundamentales que conformarán el juego a desarrollar. Se debe a que rara vez se tiene todo el concepto de lo que será el juego en mente cuando se pone a diseñar, entre los que se encuentran:
    - Género: en base a que géneros se va a desarrollar el juego. De no corresponder a un género muy conocido, se deben especificar sus características.
    - Jugabilidad: lo que provisionará la diversión del juego a la hora de jugarlo. Lo que determina la calidad del juego en términos de sus reglas de funcionamiento y de su diseño como juego.
    - Conceptos: se toman algunas ideas sueltas o que surjan en el momento acerca de cómo debe lucir el juego en cuanto a personajes, ambientación y música.

  Para facilitar el desarrollo de este proceso y sus conceptos, se utilizan los métodos de lluvia de ideas y pensamiento activo ya que es la etapa inicial (17).
- **Evaluación del mercado:** fase opcional que permite ahorrar tiempo, esfuerzo y dinero. Si se piensa ganar dinero con el juego, se realiza una investigación previa del mercado para averiguar que está teniendo éxito en el momento: que géneros, mecánicas, argumentos y apartados gráficos están de moda. Esto permitiría evitar riesgos innecesarios en un mercado muy saturado donde la competencia es más fuerte y para saber que evitar se debe de conocer el mercado en el que se va a trabajar (16).
- **Viabilidad técnica:** fase opcional y previa al diseño del videojuego propiamente dicho, que permite ahorrar tiempo, esfuerzo y dinero. Después de haberse obtenido el diseño del juego, se crea un prototipo para saber a ciencia cierta cuanto tiempo y dinero va costar el proyecto, donde se considera que dependiendo de la idea exista la posibilidad de que no se cuente con los recursos necesarios. Además, antes de empezar con el proceso del diseño, se esclarecen las plataformas para las que se lanzará el juego y la plataforma de desarrollo que se va a utilizar ya que es muy diferente diseñar juegos para móviles, consolas y PC[8] (16).
- **Diseño:** fase donde se crea el Documento de Diseño, segunda piedra angular del proyecto, después de la idea original. En él se define detalladamente el género, las mecánicas del juego, el número de jugadores, los escenarios, el sonido, el arte conceptual y todos los demás factores que definen al juego a desarrollar. El objetivo de este documento es servir de referencia para todos los trabajadores involucrados en el proyecto y para que cuando se incorpore un nuevo integrante al equipo de trabajo, sea suficiente con darle este documento para se haga una idea clara del videojuego a desarrollar. En la redacción del mismo, deben participar integrantes de todas las partes que conformarán el desarrollo del videojuego, con el objetivo de que una vez finalizado sea definitivo, que quede una

[8] *Personal computer*

imagen sencilla, fácil de entender e inalterable de cómo deberá de ser el videojuego una vez terminado. Aunque la realidad es que, en lo que avanza el proyecto se harán cambios sobre la marcha, una vez que la producción esté a medias y los plazos de entrega cerca.

- **Planificación:** fase en la que se identifican las tareas necesarias para la realización del videojuego y se distribuye entre los distintos integrantes del equipo de desarrollo. También se fijan plazos para la ejecución de dichas tareas y reuniones claves. Esto permite que los diferentes equipos de desarrollo trabajen en paralelo, acortando drásticamente el tiempo de producción. Por lo general, las estimaciones realizadas se quedarán cortas la mayoría de las veces, por todo tipo de improvistos (16).
- **Preproducción:** etapa en la que se le asigna el proyecto a un pequeño equipo, con el objetivo de verificar la factibilidad de la idea (18).
- **Producción:** es el proceso de desarrollo propiamente dicho. En esta etapa se realizan todas las tareas especificadas en la fase de planificación, teniendo como guía el Documento de Diseño. Esta fase incluye los procesos de programación, donde se realiza la codificación del programa; el proceso de ilustración donde se crean los *sprites* y cuadrículas; el desarrollo de la interfaz gráfica de usuario y el HUD[9], que serán los elementos mediante los que el usuario interactuará con el juego; la animación y modelación 3D; la grabación de sonidos, voces y música; y la creación de herramientas para acelerar el proceso de desarrollo (16) y (17).
- **Pruebas:** al igual que un software, los videojuegos deben pasar en su desarrollo por una etapa donde se corrigen los errores pertenecientes al proceso de programación y asegurar su funcionalidad. Por lo general, esta etapa está constituida por tres fases (16) y (17):
    - **Pruebas físicas:** realizada por los diseñadores y programadores del juego. Se crean prototipos que simulan los eventos que pueden suceder en el juego. Un prototipo físico puede utilizar papel y lápiz, tarjetas de índice o incluso ser actuado fuera. Sobre la base de los resultados de estas pruebas se puede hacer una mejor aproximación al balance del videojuego, pueden prevenir problemas de programación. El objetivo es jugar y perfeccionar este simplista modelo antes de que un solo programador, productor o artista gráfico estén cada vez más introducidos en el proyecto. De esta manera, el diseñador del juego recibe retroalimentación instantánea en lo que piensan los jugadores del juego y pueden ver inmediatamente si están logrando sus objetivos.

[9] Siglas en inglés de visualización cabeza-arriba, también conocido como Barra de Estado.

- **Pruebas alpha:** llevada a cabo por un pequeño grupo de personas que estén vinculas a la producción, con el propósito de corregir los errores más graves y mejorar elementos de la jugabilidad no observados en el Documento de Diseño.
- **Pruebas beta:** pruebas realizadas por un equipo externo, generalmente jugadores, los que pueden ser contratados para ese momento o que pertenezcan a un grupo específico del proyecto. Con estas pruebas el videojuego debe de salir con la menor cantidad de errores posibles.

- **Mantenimiento:** después de que el juego está terminado y se encuentre en su versión final, se comercializa su lanzamiento al mercado, donde aparecerán nuevos errores o posibles mejoras. Al recopilar toda la información posible de los jugadores, en esta etapa, se aprovecha y se realizan los cambios necesarios para mejorar el juego en todos sus aspectos. Las correcciones o mejoras realizadas se lanzan en forma de parches o actualizaciones, que en ocasiones pueden incluir características nuevas y contenido adicional para el juego, que se pone a la venta a un precio inferior al juego original, siendo conocido como una expansión (16).

1.5 Módulos o sistemas que conforman a los TRPG

Mediante la investigación realizada, se pudo determinar un conjunto de características similares entre varios juegos del género TRPG. Estas características definen la esencia de los juegos de este género, que pueden subdividirse en diferentes módulos, en base a sus funciones. El subdividir las características que definen a los TRPG en módulos o sistemas individuales, brinda una vista panorámica de los diferentes sistemas que interactúan en los videojuegos de este género. También, esta vista panorámica ayuda a definir los requisitos necesarios que harán falta para la propuesta de solución.

Generador de terrenos (*BoardGenerator*): Entre las características más notables encontradas están el tipo de terreno y su comportamiento. En este módulo se encontrarían la mayoría de las funciones que gestionan el funcionamiento del terreno:

- Los terrenos están conformados por un conjunto de baldosas en una cuadrícula, donde los objetos de la escena ya sea una unidad o ambientación de la escena, son puestos en la parte superior de la baldosa y de forma centrada.
- Para simular una elevación en el terreno, se superponen baldosas unas arriba de la otra.
- Otra característica importante es que está bien definidos los límites del terreno, de manera que se puede apreciar visualmente estos límites.
- En todo momento se tiene constancia de las coordenadas de cada baldosa; estas coordenadas son fundamentales para el Sistema de búsqueda del camino más corto (más conocido como *pathfinding* en inglés), sistema encargado del desplazamiento de las unidades.

Navegación (*Navigation*): Los juegos de este género, para ayudar al jugador, presentan un sistema de navegación encargado de determinar los obstáculos y lugar de destino deseado; así como hallar el camino más corto para llegar a ese destino. También estaría encargado del control de las diferentes interfaces gráficas:

- Se resaltan las baldosas para delimitar la distancia y lugares a los que puede alcanzar una unidad.
- El sistema reconoce los obstáculos desplegados en el terreno, así como los objetos con que se puede interactuar.
- Dependiendo del tipo de desplazamiento que presente la unidad seleccionada, realiza el cálculo del camino más corto. En muchos juegos de este género, existen unidades que pueden pasar a través o estacionarse en baldosas que no son accesibles para otras unidades.
- La característica o función de mayor relevancia de este sistema es el cálculo del camino más corto para llegar al destino deseado, donde se tiene en cuenta un conjunto de factores.

Inteligencia Artificial (IA): Elemento clave en la mayoría de los juegos. La IA es usada por lo general, para el control de las unidades enemigas y varios subprocesos en los juegos; de igual manera sucede en los de este género:

- Encargada de la gestión de las unidades enemigas para los combates, así como procesos secundarios como la venta de objetos.
- Encargada del análisis y procesamiento de la información del terreno y de la ubicación de las unidades en cada turno para crear y decidir una estrategia en base a un conjunto de comportamientos definidos, dependiendo de las características de la unidad activa. Los comportamientos pueden ser patrones pre-establecidos o una IA de mayor complejidad que decida por sí misma, como un autómata.

Sistema de Batalla (*Battle System*): El sistema de batalla de los TRPG presenta un conjunto de características claves que definen a los juegos de este género, así como algunos de los RPG. Es el componente de mayor relevancia en el sistema ya que controla la mayoría de los procesos de importancia e interacciona con la mayoría de los demás sistemas.

- Una característica clave que se puede apreciar a simple vista en la mayoría de los juegos de este género es el combate por turnos.
- El sistema se encarga de gestionar el orden de los turnos de las diferentes unidades tomando en cuenta, por lo general, la cantidad de acciones que realizó la unidad en su turno anterior, así como algunas de sus estadísticas o los estados alterados activos en la unidad que puedan afectar la llegada del mismo.

- Es característico en los TRPG, así como en algunos RPG y subgéneros del mismo, que el jugador controle a más de una unidad para pelear. Las batallas tienen definido un límite de unidades posibles a usar, obligando al jugador a escoger un pequeño grupo del total de unidades que posea, denotándose de esta forma la presencia de la estrategia.
- Cada vez que una unidad actúa, se debe de confirmar todas las acciones que se realizan. Si una acción no es del agrado del jugador, como moverse a una posición por error, si no ha confirmado la acción aun, puede volver a la posición anterior.
- Es característico de los RPG que las unidades o personajes presenten habilidades especiales como el uso de la magia.
- Es muy común en los juegos TRPG, el cálculo de probabilidades de acierto de una acción, donde se toman en cuenta un conjunto de estadísticas para realizar dicha operación, así como la dirección de la que proviene y hacia donde mira el oponente.
- Muchos juegos de este género tienen presente darle la oportunidad al jugador de establecer la orientación en la que se quedara mirando la unidad activa antes de confirmar por completo la terminación del turno.
- Es común que exista la presencia de estados alterados en juegos del género RPG, los mismos pueden ser beneficiosos o negativos. La razón de su presencia es permitir al jugador aumentar su repertorio estratégico, brindándole la oportunidad de una mayor gama de acciones, así como para aumentar la dificultad cuando las usa el enemigo.
- Es natural en la mayoría de los juegos RPG, el encuentro de batallas aleatorias, donde el sistema crea un conjunto de enemigos al azar. El objetivo principal de esta funcionalidad es evitar la monotonía del juego y que el jugador se aburra. También, se hace con el objetivo de brindar la oportunidad de obtener dinero, objetos o experiencia para fortalecer a las unidades.
- Otra característica clave en los juegos del género TRPG, es la existencia de condiciones de victoria y derrota. Estas condiciones varían dependiendo de muchos factores, como la historia del juego o el tipo de batalla que se quiera simular.

Sistema de trabajos o clases (*Job System*): Al igual que el sistema de batalla, este sistema es fundamental. En él se encuentran características esenciales que definen a los RPG, como son el uso de las estadísticas de las unidades, así como la definición de las diferentes clases o trabajos que pueden hacer las unidades.

- Al igual que la mayoría de los RPG, las unidades presentan un conjunto de estadísticas que definen su fuerza.
- Presenta un sistema de subida de niveles a base de premiación por puntos o experiencia al terminar una batalla. Cuando una unidad sube de nivel, sus estadísticas aumentan.

- Es característico en los RPG las presencia de más de una clase o trabajo a usar, lo mismo sucede en los TRPG. Estos trabajos definen el crecimiento de las estadísticas al subir de nivel, así como las habilidades disponibles a utilizar por la unidad.
- Otra característica clave que es muy evidente en los juegos RPG es la presencia de objetos o artículos a utilizar. Estos pueden ser de una gran variedad, tales como objetos claves para el cumplimiento de una misión, consumibles o equipamientos como armaduras o armas. En el caso de los objetos consumibles y de los tipos de equipamiento, tienen una influencia directa en las estadísticas de la unidad que los usa. La razón principal es ayudar al jugador a mejorar las estadísticas de las unidades mediante el uso de estos objetos.

Se puede llegar a desglosar en más componentes las diversas funcionalidades presentes en los videojuegos de este género; pero en los módulos presentados con anterioridad, se definen las características principales, básicas y necesarias para el desarrollo de un videojuego de este género.

1.6 Metodología de desarrollo

Una metodología de desarrollo de software se refiere a un marco de trabajo que es usado para estructurar, planear y controlar el proceso de desarrollo en sistemas de información (19). A lo largo del tiempo, una gran cantidad de métodos han sido desarrollados diferenciándose por su fortaleza y debilidad. Actualmente existen diferentes metodologías de desarrollo de software, debido a que todos los proyectos de software no son iguales y por tanto necesitan un enfoque de desarrollo acorde a sus características particulares. Dos de los principales criterios que se usan para diferenciarlas son el tamaño del personal y la criticidad del sistema. Sobre la base de estas opciones, el equipo del proyecto selecciona la metodología ligera o pesada que desea para su proyecto (20).

Los proyectos difieren en cuanto a composición y prioridades. Las personas de un proyecto poseen además diferentes niveles de experiencia, principios y habilidades (20). Todos estos aspectos deben ser considerados para seleccionar correctamente una metodología de desarrollo de software.

1.6.1 Fundamentación de la metodología de desarrollo seleccionada

XP[10] fue la metodología seleccionada debido a que se adapta en gran medida al tipo de proyecto a desarrollar, las condiciones de trabajo y las prácticas utilizadas. Se enuncian varias de las razones que motivaron la selección de esta metodología.

[10] XP: eXtreme Programing.

- El proyecto posee poca envergadura. XP está concebida para ser utilizada dentro de proyectos pequeños (21).
- El cliente forma parte del equipo de desarrollo. Mediante la aplicación de XP se puede lograr una retroalimentación mayor y lograr un producto acorde a las necesidades definidas.
- El riesgo de desarrollo es elevado debido al corto tiempo de entrega planteado y a los continuos cambios de requerimientos. XP está diseñada para mitigar los riesgos en proyectos con estas características.
- Las funcionalidades del sistema pueden variar. El sistema debe cambiar y ampliar sus funcionalidades de forma que sea capaz de ajustarse a cada nuevo requerimiento. Uno de los principios básicos de XP es que el cambio frecuente de las funcionalidades es algo normal en el proceso de desarrollo.
- Algunas prácticas de XP como las entregas pequeñas, el diseño simple y el Desarrollo Guiado por Pruebas (TDD[11]) son parte de la filosofía de desarrollo del equipo (20).

1.6.2 Exploración, Planificación y Diseño

Exploración: Primera fase del proceso de desarrollo de software que define XP. En ella, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología (22).

Las HU[12] son la forma en que se especifican los requisitos del sistema con el uso de la metodología XP. Estas se escriben desde la perspectiva del cliente, aunque los desarrolladores pueden brindar también su ayuda en la identificación de las mismas (22). Son escritas en lenguaje natural, sin un formato predeterminado, no excediendo su tamaño de unas pocas líneas de texto. Además, guían la construcción de las pruebas de aceptación y son utilizadas para estimar tiempos de desarrollo.

Se define la clasificación de las HU atendiendo a varios criterios, para facilitar su organización y correcta planificación. Los parámetros definidos en cada una de las HU se detallan a continuación:

- Número: Para mantener el orden de las historias.
- Nombre: Para identificar la HU.

[11] TDD: del inglés Test Driven Development.
[12] HU: Historia de Usuario

- Prioridad:
    1. Alta: Constituyen funcionalidades principales del sistema, o forman parte esencial de la arquitectura del mismo.
    2. Media: Son funcionalidades importantes y de gran valor para el usuario pero que no impiden poner el proyecto en marcha si no se tienen.
    3. Baja: Funcionalidades que serían deseables tener y que podrían incluirse.
- Riesgo en desarrollo:
    1. Alto: En caso de tener algún error de implementación, pueden afectar la disponibilidad e integralidad del sistema.
    2. Medio: En caso de presentar errores retrasan la entrega de la versión.
    3. Bajo: En caso de presentar errores, estos pueden ser tratados con facilidad.
- Iteración asignada: Número de la iteración a la que ha sido asignada.
- Puntos estimados: Tiempo estimado de desarrollo para completar la HU. Un punto es equivalente a una semana ideal de programación (40 horas).
- Descripción: Se define la funcionalidad que se quiere desarrollar.
- Observaciones: Detalles a tener en cuenta para desarrollar la HU correctamente.
- Prototipo de interfaz: Es un boceto representativo de la vista de usuario (22).

Planificación: En esta fase el cliente establece la prioridad de cada HU y en correspondencia los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente (22).

Diseño: Durante el diseño de la solución, la máxima simplicidad posible es la clave para el éxito de XP. Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. La complejidad innecesaria y el código extra deben ser evitados en todo momento (19) y (20).

1.7 Tecnologías y herramientas

Diversas son las herramientas actualmente utilizadas para el desarrollo de videojuegos que se encuentran a disposición de los usuarios ya sea para cualquier tipo de plataforma a la que se desee lanzar el juego, tanto si es para un desarrollo independiente o corporativo. A continuación, se realizará una breve descripción de las tecnologías y herramientas a utilizar.

1.7.1 Motor de videojuego. Concepto

Un motor de videojuego es un término que hace referencia a una serie de rutinas de programación que permiten el diseño, la creación y la representación de un videojuego. Su funcionalidad básica es proveer al

videojuego de un motor de renderizado para los gráficos 2D y 3D, motor físico o detector de colisiones, sonidos, scripting, animación, IA, redes, *streaming*, administración de memoria y un escenario gráfico. El proceso de desarrollo de un videojuego puede variar notablemente por reutilizar o adaptar un mismo motor de videojuego para crear diferentes juegos (23).

Otros conceptos muy ligados a los motores de videojuegos son las API[13] y SDK[14]. Las API son las interfaces de software ofrecidas por los sistemas operativos, bibliotecas y servicios para que el usuario pueda aprovecharlas en la creación del juego. Un SDK es un conjunto de bibliotecas, API y herramientas disponibles para la programación. La mayoría de los motores de videojuegos proporcionan API en sus SDK(23).

1.7.2 Unity 5.6.0f3

Unity es un motor de videojuego multiplataforma creado por Unity Technologies. Está disponible como plataforma de desarrollo para Microsoft Windows, OS X, Linux. Tiene soporte de compilación con diferentes tipos de plataformas. A partir de su versión 5.4.0 ya no soporta el desarrollo de contenido para navegador a través de su plugin web, en su lugar se utiliza WebGL. Unity tiene dos versiones: Unity Professional (pro) y Unity Personal (24).

Tiene soporte para mapeado de relieve, mapeado de reflejos, mapeado por paralaje, oclusión ambiental en espacio de pantalla, sombras dinámicas utilizando mapas de sombras, render de texturas y efectos de post-procesamiento de pantalla completa. Soporte integrado para Nvidia, (a partir de Unity 3.0) con soporte en tiempo real para mallas arbitrarias y para capas de colisión (25).

Unity presenta cuatro opciones de licencia: Personal, Plus, Pro, Enterprise. Cada una de las licencias presentan diferencias (26), como se puede apreciar en la siguiente tabla:

Tabla 1 Tipos de licencias de Unity (tomado de (26))

Nombre de Licencia	Todas las plataformas y características del motor	Cola de construcción en la Nube	Multijugador	Capacidad de ingresos	Reporte de desempeño	Fuente de código y soporte premium	Precio
Personal	Si	Normal	20 CCUs[15]	$100.000	No	No	Libre

[13] Interfaz de Programación de Aplicaciones o *Application Programming Interface*.
[14] Kit de Desarrollo de Software o *Software Development Kit.*
[15] *Concurrent user* o usuarios concurrentes.

Plus	Si	Prioridad	50 CCUs	$200.000	Si	No	$35 mensual
Pro	Si	Construcciones concurrentes	200 CCUs	Ilimitado	Si	Si	$125 mensual
Enterprise	Si	Usuarios dedicados	Personalizado	Ilimitado	Si	Si	Precio negociable

Unity es un motor de videojuegos basado en componentes y provee muchas capas de juego que soportan al programador de la mecánica del sistema. Como ejemplo, se puede diseñar diferentes estados de transición de animación en la interfaz gráfica del *Animator*. Otra característica de Unity, es que su API está orientada a objetos y se compone de más de un centenar de clases, donde una tercera parte aproximadamente están relacionadas entre sí por términos de herencia, siendo *Object* la clase base de la que heredan muchas de las funcionalidades utilizadas (24).

Existen un gran conjunto de módulos para Unity que facilitan la realización de videojuegos. Según una investigación realizada con el objetivo de encontrar módulos que posibiliten la solución de la problemática presente o que faciliten la creación de una solución, se encontraron un conjunto de módulo de utilidad. Los módulos encontrados no pueden ser utilizados producto a que son privados y presentan un alto costo. A continuación, se mencionarán algunos de módulos:

- *RPG All in One (RPGAIO)* (27).
- *ORK Framework – RPG Engine* (27).
- *RPG Core Combat Creator* (27).
- *Dialogue System for Unity* (27).
- *THE Dialogue Engine* (27).
- *Turn Based Strategy Framework* (27).
- *RPG Battle System* (27).
- *Ultimate RPG Engine* (27).
- *Turn-Based ToolKit (TBTK)* (27).
- *RPG Map Editor* (27).

1.7.3 C#

Lenguaje de programación orientado a objetos, que fue desarrollado y estandarizado por Microsoft alrededor del año 2000, volviéndolo parte de su plataforma .NET[16], siendo uno de los lenguajes de programación

[16] Red

diseñados para la infraestructura de lenguaje común. Posteriormente fue aprobado como un estándar por la ECMA [17](ECMA-334) e ISO[18] (28).

Presenta una sintaxis básica derivada de C y C++y utiliza el modelo de objetos de la plataforma .NET, basado de forma similar al de Java, pero aplicando mejoras derivadas de otros lenguajes.

A pesar de que el lenguaje forma parte de la plataforma .NET, que es una API, C# es un lenguaje de programación independiente diseñado para desarrollar programas sobre dicha plataforma. Entre las diferentes características que definen su paradigma de programación se encuentran que es: estructurado, imperativo, orientado a objeto, dirigido por eventos, funcional, genérico y reflexivo; llegando a la conclusión de que es multiparadigma. Presenta un sistema de tipos estático, dinámico, fuerte, seguro, nominal. También es un lenguaje de programación que permite generar programas para varias plataformas como Windows, Android, iOS, Mac OS, Linux; siendo un lenguaje para el desarrollo multiplataforma (29).

1.7.4 Adobe Photoshop

Programa Utilizado generalmente para el retoque de fotografías y gráficos, desarrollado por *Adobe Systems Incorporated.* Líder mundial del mercado de las aplicaciones de edición de imágenes (30).

Trabaja en un espacio formado por múltiples capas donde se puede aplicar una gran variedad de efectos, textos, marcas y tratamientos. Usado extensivamente en multitud de disciplinas del campo del diseño y fotografía, como diseño web, composición de imágenes en mapa de bits, fotocomposición, edición; prácticamente en cualquier trabajo en el que se requiera el tratamiento de imágenes digitales. El mundo del desarrollo de los videojuegos es también un campo en el que es fuertemente utilizado, principalmente para la creación de los modelos base de personajes, ambientación e interfaz (30) y (31).

Photoshop está disponible para una gran variedad de idiomas y para los sistemas operativos Windows y Mac (30).

1.7.5 Microsoft Visual Estudio 2017

Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) de Microsoft. Utilizado para el desarrollo de programas de computadora, al igual que para páginas web, aplicaciones web, servicios web y aplicaciones móviles. Incluye un editor de código soportado por *IntelliSense* (componente de terminación de código) así como la re-descomposición de código. El depurador integrado que presenta trabaja tanto como un depurador de bajo nivel y a nivel de máquina (32).

[17] *Ecma International*
[18] La Organización Internacional de Normalización

Soporta 36 lenguajes de programación entre los que se encuentran C, C++, Visual Basic .NET, C#, JavaScript, XML, HTML y CSS como entre los utilizados para el trabajo en Visual Studio. También incluye diseñadores visuales para auxiliar el desarrollo de las aplicaciones (33). Estas herramientas incluyen:

- *Windows Form Designer:* utilizada para el desarrollo de GUI[19] de las aplicaciones usando *Windows Forms* (33).
- Diseñador de Clases es usado para publicar y editar las clases (incluyendo a sus miembros y sus accesos) usando modelado UML[20]. El Diseñador de Clases puede generar bosquejos de código C# y VB.NET, para las clases y los métodos. También puede generar diagramas de clases de clases escritas a mano (32).

Presenta a la vez un conjunto de otras herramientas variadas para diferentes fines, con el objetivo de facilitar el desarrollo ya sea para el trabajo individual o en equipo.

1.8 Conclusiones parciales

Mediante el desarrollo de este capítulo se logró una mejor compresión de las características que definen a los videojuegos del género TRPG, así como algunas de los RPG. Con el conocimiento del proceso necesario para la realización de un videojuego, se tiene una preconcepción de cómo debería estructurarse la propuesta de solución para ayudar a este proceso. Se analizaron un conjunto de softwares posibles a utilizar para la solución del problema donde se decidió la utilización de Unity como herramienta principal debido a que es la herramienta utilizada por el centro y por sus características.

[19] Interfaz Gráfica de Usuario
[20] *Unified Modeling Language* o Lenguaje Unificado de Modelado

Capítulo 2: Propuesta de solución. Exploración, planificación y diseño

En el siguiente capítulo se describe el prototipo de solución a desarrollar, se especifican las características principales que presentará la solución en base a la investigación realizada con anterioridad. Se hace referencia a las fases de Exploración, Planificación y Diseño que plantea la metodología de desarrollo seleccionada, XP. Se muestran la arquitectura y los patrones de diseño utilizados para la implementación de la solución.

2.1 Características de la propuesta de solución.

Como se pudo observar en la investigación realizada en el capítulo anterior, los videojuegos TRPG presentan una gran complejidad producto a la cantidad de sistemas que interactúan entre sí. En pos de facilitar el desarrollo de videojuegos de este género se decidió desarrollar un módulo que abarque las características de mayor relevancia. Debido a la decisión tomada de utilizar a Unity como herramienta principal de desarrollo, se decidió la utilización de una arquitectura basada en componentes, como arquitectura principal, para aprovechar sus características.

Con la realización de este módulo se obtiene un recurso que facilita el desarrollo de los videojuegos TRPG. También posibilita que el centro Vertex tenga a su disposición un nuevo género de videojuego en el que trabajar, aumentando de esta manera la gama de productos a desarrollar y su variedad.

A continuación, en la siguiente Fig. 11 Estructura del módulo propuesto como solución, se ilustra la composición del módulo a desarrollar como propuesta de solución. Se decidió representar la estructura del funcionamiento del módulo, para una mejor comprensión, similar a la descomposición de sistemas que conforman a los TRPG que se realizó con anterioridad en el capítulo anterior.

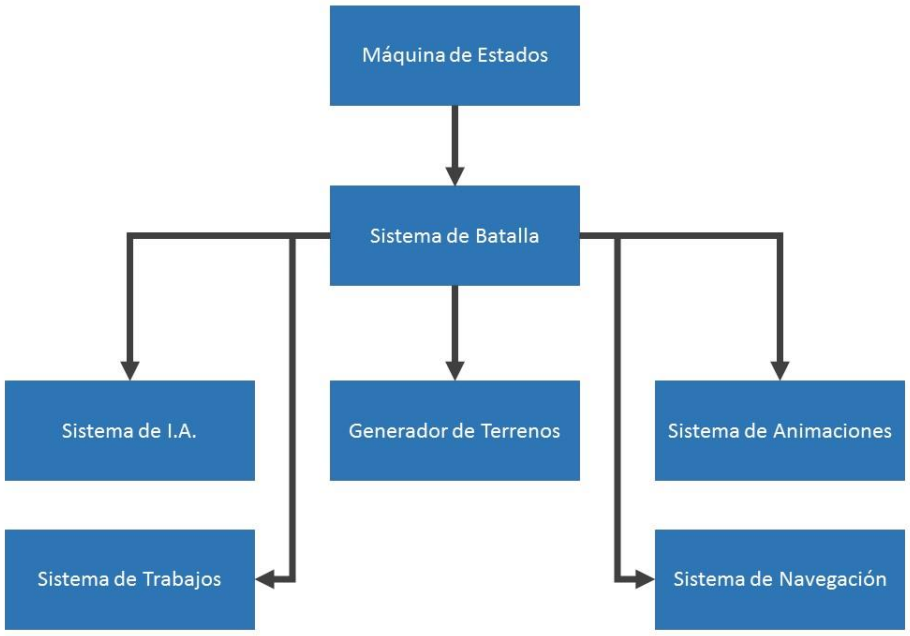


Fig. 11 Estructura del módulo propuesto como solución

2.2 Procesos de los sub-módulos para la generación de videojuegos TRPG

En la investigación realizada, se definieron los procesos clave a tener en cuenta para el desarrollo de la propuesta de solución. Estos procesos se definieron teniendo en cuenta las características antes presentadas en la sección 1.5. Para una mejor organización y compresión, los procesos se separaron en base a los módulos o sistemas a los que están más relacionados.

- Generador de Terrenos:
  - RF 1. Generar terreno de forma aleatoria.
  - RF 2. Reducir terreno de forma aleatoria.
  - RF 3. Agregar *tiles* de forma manual.
  - RF 4. Reducir *tiles* de forma manual.
  - RF 5. Agregar un *tile* en la parte superior de otro *tile* si ya existe un *tile* en la posición mediante la generación aleatoria.
  - RF 6. Agregar un *tile* en la parte superior de otro *tile* si ya existe un *tile* en la posición mediante la generación manual.
  - RF 7. Destruir de forma inmediata el *tile* si alcanza la altura cero al reducir mediante la forma aleatoria.
  - RF 8. Destruir de forma inmediata el *tile* si alcanza la altura cero al reducir mediante la forma manual.

RF 9. Poner de forma centrada el *tile* cuando se agregue arriba de otro.

RF 10. Modificar el terreno de forma manual.

RF 11. Guardar los terrenos generados.

RF 12. Cargar los terrenos guardados.

RF 13. Limpiar el terreno cuando se cargue un terreno guardado.

RF 14. Mostrar coordenadas **x** & **y** del *tile* en que se está posicionado.

RF 15. Especificar dimensiones del terreno a crear (cantidad de *tiles* de ancho, largo y alto).

RF 16. Limpiar terreno.

- Sistema de Batalla:

RF 17. Definir rango, área de efecto y potencia de cada habilidad de cada clase.

RF 18. Definir los efectos de las habilidades (alteración de estados, dañar salud, restaurar salud).

RF 19. Comprobar confirmación de acción.

RF 20. Confirmar orientación de la unidad después de terminar su turno.

RF 21. Gestionar orden de los turnos de las unidades en base a su estadística de velocidad.

RF 22. Gestionar orden de los turnos de las unidades en base a las acciones realizadas, que presentarán un costo de uso que afecta la velocidad en que llega el turno.

RF 23. Gestión de los estados alterados, su duración y de cómo influye en la unidad afectada después de ser aplicado.

RF 24. Gestión de las probabilidades de acierto de una acción en bases a las estadísticas de la unidad, dirección en la que se encuentra apuntando la unidad atacada y estados alterados que puedan afectar esta probabilidad.

RF 25. Gestionar la fábrica de personajes, donde se creará y configurará nuevas unidades.

RF 26. Gestionar condiciones de victoria y derrota de cada batalla.

- Sistema de Trabajos:

RF 27. Gestionar el sistema de estadísticas de las unidades.

RF 28. Gestionar subida de niveles.

RF 29. Gestionar la ganancia de experiencia dependiendo del nivel y clase de la unidad.

RF 30. Gestiona el sistema de equipamiento y de objetos, así como el cambio de las estadísticas en base a los mismos.

- Sistema de IA:

RF 31. Inteligencia avanzada, encargada de procesar la información del terreno en cada turno, analizar la información y aplicar patrón de ataque definido.

RF 32. Inteligencia básica, encargada de la gestión de las unidades enemigas en los combates y de procesos segundarios.

- Navegación:
  RF 33. Mostrar todos los *tiles* a los que una unidad puede alcanzar.
  RF 34. Reconocer los obstáculos desplegados.
  RF 35. Reconocer el tipo de desplazamiento que presenta la unidad.
  RF 36. Después de haberse seleccionado un *tile*, la unidad debe de recorrer el camino más corto para llegar aplicando la animación correspondiente al tipo de desplazamiento.
- Sistema de Animaciones:
  RF 37. Animación de los paneles.
  RF 38. Animación de desplazamiento de las unidades.

2.3 Características no funcionales del sistema.

Las características no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable (20).

Usabilidad

- Para una mejor orientación a la hora de trabajar con el sub-módulo de generación de terreno, los *tiles* seleccionados deben de mostrar un indicador que logre que el usuario perciba la acción de cuales están seleccionados.
- Para una mejor orientación a la hora de trabajar con el sub-módulo de generación de terreno, los *tiles* seleccionados deben de mostrar las coordenadas de su posición.
- Dentro de la carpeta de *Resources*, deben de existir *prefabs* o instrucciones que sirvan de guía y referencia para la utilización del módulo.

Software

El cliente o usuario final de este producto solo necesita tener instalado Unity, en su versión 5.6.0f3 como recomendación, para evitar posibles errores, siendo esta la versión en la que se desarrolló el módulo.

Disponibilidad

Para una mejor disponibilidad, el módulo genérico se convertirá en un *asset* de Unity, que se pondrá en un repositorio para una mayor y fácil distribución.

2.4 Elementos a tener en cuenta para utilizar la propuesta de solución.

Gracias al sistema de componentes de Unity y su API orientado a objetos, la mayoría de los componentes que conforman al módulo propuesto como solución, se podrán utilizar mediante el uso de arrastrar y soltar

sobre el objeto deseado. En la carpeta de *Resources*, existirán un conjunto de *prefabs* para utilizar, que servirán a la vez como muestras de ejemplo de las combinaciones correctas de los componentes. También existirá una escena llamada *Battle*, que servirá como ejemplo y guía del uso correcto de los componentes.

Existirán un conjunto de componentes, en la carpeta *Editor*, que modificarán parte de la interfaz de Unity, para facilitar la creación de objetos específicos para ser usados por el sistema del módulo. Uno de ellos es el *JobParser*; este traducirá la información contenida en la carpeta *Settings* para la creación de los trabajos y sus estadísticas por defecto.

A la vez, existirán un conjunto de componentes que no serán usados de forma directa, formarán parte del funcionamiento del sistema. Algunos de estos componentes son los que se encuentran en *Notification Center* y *State Machine*, todos dentro de la carpeta *Common*.

2.5 Exploración

2.5.1 Historias de usuario

Durante la fase de exploración, después de haber conocido los objetivos que se quieren lograr con el sistema, así como los requisitos no funcionales del mismo, se puede proseguir con el análisis de las funcionalidades con las que debe de contar la propuesta de solución para darle respuesta a esos objetivos. En relación con lo antes planteado identificaron las siguientes historias de usuario, donde el software debe de ser capaz de:

- Creación de terreno.
- Sistema de movimiento por turno.
- Sistema de estadísticas de unidades.
- Modificación de estadísticas por estados alterados.
- Variedad de patrones de acción por la IA.

Siguiendo las pautas planteadas por XP se definen las historias de usuarios correspondientes a la aplicación.

Tabla 2 Historia de Usuario - Sistema de movimiento por turno

Número: 1	**Nombre:** Sistema de movimiento por turno
Prioridad: Alta	**Iteración Asignada:**1
Riesgo en desarrollo: Alto	**Puntos estimados :**2
Descripción:	

Cada unidad se moverá en un turno determinado, que llegará en base a sus estadísticas. Cada acción que realice una unidad tendrá un costo que afectara en la velocidad en que llegara su próximo turno.
Observaciones: Los estados alterados pueden afectar la velocidad en que llegue el próximo turno de una unidad ya sea de manera positiva o negativa.
Prototipo:

Tabla 3 Historia de Usuario – Sistema de estadísticas de unidades

Número: 2	**Nombre:** Sistema de estadísticas de unidades
Prioridad: Alta	**Iteración Asignada:**1
Riesgo en desarrollo: Alto	**Puntos estimados :** 1
Descripción: Las unidades tendrán un conjunto de estadísticas base iniciales, que dependerán del trabajo o clase de la unidad. También tendrán un conjunto de estadísticas que determinarán el crecimiento de la unidad cuando suba de nivel. En pleno juego se mostrarán algunas de las estadísticas más relevantes para información del jugador.	
Observaciones: Estas estadísticas se podrán definir en las dos tablas de Excel en formato .cvs, que se encuentran en la dirección *Asset*s\Settings, para después convertir la información, mediante la herramienta implementada de PreProducción, Parse Jobs, en *prefabs* que pueda usar Unity.	
Prototipo:	

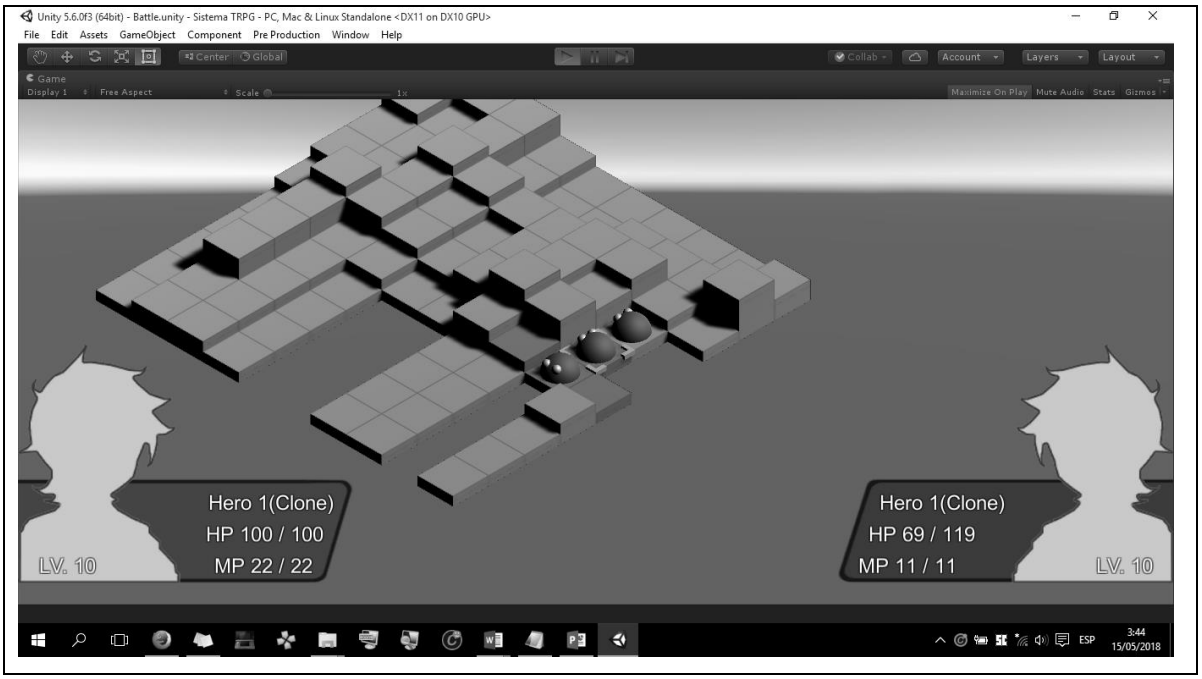


Tabla 4 Historia de Usuario - Modificación de estadísticas por estados alterados

Número: 3	**Nombre:** Modificación de estadísticas por estados alterados
Prioridad: Medio	**Iteración Asignada:**2
Riesgo en desarrollo: Medio	**Puntos estimados :** 1.5
Descripción: Los estados alterados pueden ser beneficiosos o negativos, dependiendo de sus efectos. La mayoría afectarán alguna estadística de la unidad implicada. Los estados alterados se aplicaran mediante alguna habilidad, objeto, equipo o parte del terreno.	
Observaciones: No necesariamente todos los estados alterados modificaran las estadísticas. Se pueden implementar efectos de estados alterados para modificar otras características como, por ejemplo, impedir que la unidad pueda atacar en cierta cantidad de turnos o que no se pueda mover.	
Prototipo:	

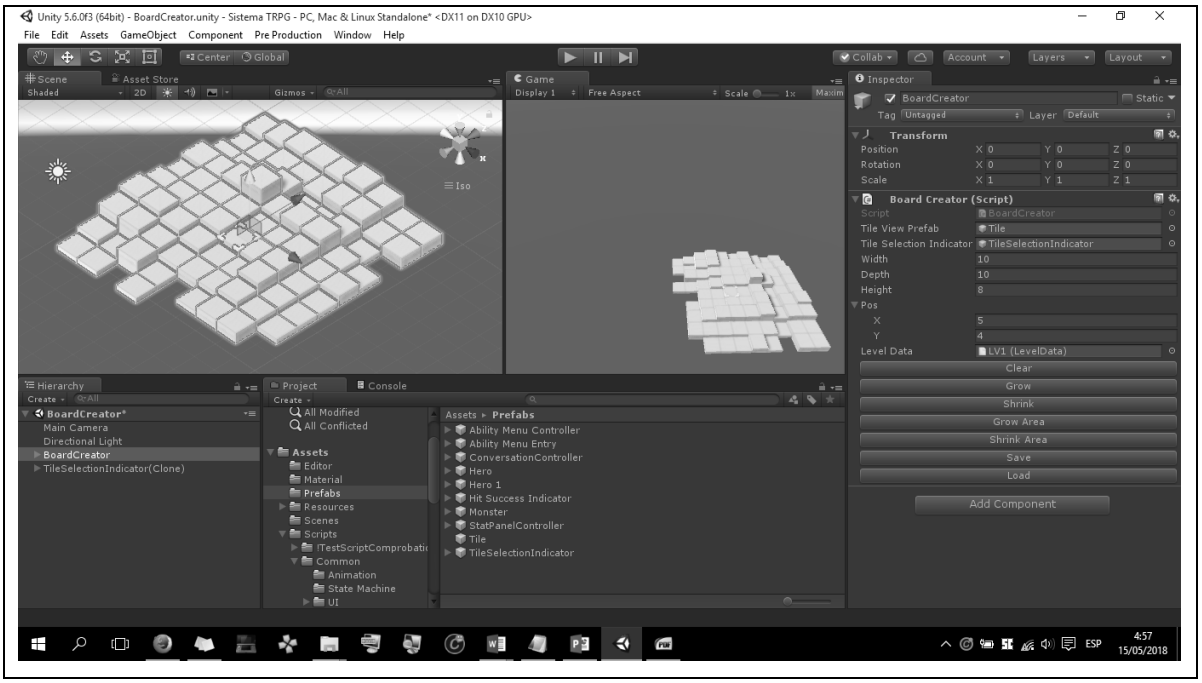
Tabla 5 Historia de Usuario - Variedad de patrones de acción por la IA

Número: 4	**Nombre:** Variedad de patrones de acción por la IA
Prioridad: Media	**Iteración Asignada:** 2

Riesgo en desarrollo: Medio	**Puntos estimados :** 1.5
Descripción: La IA tendrá un conjunto de patrones de ataque que definirán su comportamiento, dependiendo de las condiciones del juego o de la unidad que está controlando. En base a esto genera su propia estrategia de juego y toma una decisión donde realiza alguna acción como moverse, atacar, lanzar una habilidad o acercarse.	
Observaciones: Estos patrones pueden ser tanto ejecutados por una IA enemiga como aliada o una neutral.	
Prototipo:	

Tabla 6 Historia de Usuario - Creación de terreno

Número: 5	**Nombre:** Creación de terreno
Prioridad: Baja	**Iteración Asignada:**3
Riesgo en desarrollo: Bajo	**Puntos estimados :** 2
Descripción: Permite al usuario determinar las dimensiones máximas que tendrá el terreno. Da la posibilidad de crear los terrenos de forma aleatoria como manual, mediante la eliminación, creación y estiramiento de los *tiles.* Posibilita guardar y cargar la información de los terrenos creados. Permite limpiar el tablero para empezar la creación de un nuevo terreno.	
Observaciones: Permitirá la creación de un terreno básico y sencillo. En un futuro se puede mejorar para que sea un editor de mapas más completo que permita, además, agregar decoraciones, trampas u otro elemento.	
Prototipo:	



2.6 Planificación

2.6.1 Estimación del esfuerzo por historias de usuario

Para el desarrollo del sistema propuesto se realizó una estimación aproximada del esfuerzo que costaría para cada una de las HU identificadas, llegándose a los resultados que se muestran en la siguiente Tabla 7 Estimación del esfuerzo por HUD.

Tabla 7 Estimación del esfuerzo por HUD

HUD	Puntos Estimados
Sistema de movimiento por turno	2
Sistema de estadísticas de unidades	1
Modificación de estadísticas por estados alterados	1.5
Variedad de patrones de acción por la I.A	1.5
Creación de terreno	2

2.6.2 Plan de iteraciones

Tomando como referencia los aspectos antes explicados de la metodología en el capítulo 1, después de haberse definido las HU y estimado el esfuerzo para la realización de cada una de ellas, la aplicación a elaborar se desarrollará en 3 iteraciones, explicada más detalladamente a continuación:

Iteración 1

El objetivo de esta iteración es la implementación de las siguientes HU, que son las de mayor prioridad: Sistema de movimiento por turno y Sistema de estadísticas de unidades. Durante el transcurso de la iteración se crearán las bases de la arquitectura del sistema con una funcionalidad mínima. Al final de esta se deberá contar con una primera versión de prueba, que será mostrada al cliente con el objetivo de obtener una retroalimentación para el equipo de desarrollo.

Iteración 2

Esta iteración tiene como objetivo la implementación de las HU de prioridad media, Modificación de estadísticas por estados alterados y Variedad de patrones de acción por la IA. Al finalizar se contará con una versión más funcional del sistema, donde se podrá probar la mayoría de las funcionalidades de relevancia. Esta versión igualmente se mostrará al cliente con el objetivo de realizar cambios necesarios en base a la opinión del mismo.

Iteración 3

Durante esta iteración se implementarán las HU de menor prioridad, lográndose con su finalización una versión 1.0 del sistema bastante completa y cercana del producto final deseado. Con esta primera versión terminada, el sistema se encontrará listo para su utilización.

Duración de las iteraciones

Como parte del ciclo de vida de un proyecto que utiliza XP se crea el plan de duración de cada una de las iteraciones. Este plan se encarga de mostrar las historias de usuario que serán abordadas en cada una de las iteraciones, así como la duración estimada de estas últimas y el orden en que se implementarán las HU.

Tabla 8 Plan de duración de las iteraciones

Iteración	HU a implementar	Duración total de iteración
Iteración 1	Sistema de movimiento por turno	3
	Sistema de estadísticas de unidades	
Iteración 2	Variedad de patrones de acción por la IA	3
	Modificación de estadísticas por estados alterados	

Iteración 3	Creación de terreno	2

2.6.3 Plan de entregas

Se presenta el plan de entregas elaborado para la fase de implementación, que comienza el 18/04/2018. Como producto del mismo se harán entregas incrementales del sistema al finalizar cada iteración, en las fechas aproximadas que se indican a continuación.

Tabla 9 Plan de entrega de las iteraciones

Iteración	Fecha de Entrega
Iteración 1	9/05/2018
Iteración 2	30/05/2018
Iteración 3	13/06/2018

2.7 Diseño

2.7.1 Patrón arquitectónico

Como se había explicado con anterioridad, Unity es un motor de videojuegos basado en componentes y con una API orientada a objeto. La propuesta de solución a desarrollar es un conjunto de componentes que se relacionan entre sí, que pueden ser utilizados arrastrándolos hacia el objeto deseado, siendo de esta manera reusables. Muchos de los componentes se les pasa la información deseada, en vez de estas estar incluida, permitiendo que no tengan un contexto específico.

Se evidencia de igual manera, el uso de varios componentes para encapsular la información, permitiendo el uso de las funciones necesarias, pero sin revelar detalles internos. Por lo que se propone a utilizar una arquitectura basada en componentes, que es la que más aprovecha la estructura y ventajas de Unity.

2.7.2 Patrones de diseño

El término de patrón fue dado por primera vez en el año 1977 por el arquitecto Christopher Alexander, quien dio en su libro *A Pattern Language* la siguiente definición: “Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo ni siquiera dos veces de

la misma forma". En la ingeniería del software, un patrón constituye el apoyo para la solución a los problemas más comunes que se presentan durante las diferentes etapas del ciclo de vida del software (34).

Para el diseño de la aplicación se hizo uso de los Patrones Generales de Software para Asignar Responsabilidades (GRASP), así como de los patrones *Gang of Four* (GoF).

Patrones GRASP utilizados:

- **Experto:** Patrón que se evidencia fuertemente en la mayoría de las clases. Un ejemplo claro es en los scripts de *Turn* y *TurnOrderController*, encargados de la gestión y control de los turnos y de mantener constancia de las acciones realizada por la unidad.
- **Creador:** Las clases que tienen la responsabilidad de instanciación de objetos o componentes, donde se destacan el *BoardCreator*, encargado en la creación del terreno mediante la instanciación de baldosas; y *UnitFactory*, encargado de la creación de las unidades cada vez que inicia una batalla.
- **Controlador:** Patrón fuertemente evidenciado en las clases de terminación *Controller*, que se encuentran en la carpeta de igual nombre; se encargan de ser intermediarias entre una determinada interfaz y el algoritmo que la implementa. Siendo estas las que reciben la información entrada por el usuario y la que los envía a las distintas clases según el método llamado.
- **Fabricación Pura:** Patrón que se muestra con claridad, por el conjunto de clases bases para la reutilización y sobrecarga de diferentes métodos. Utilizado principalmente en las áreas de los estados alterados, el rango de acción y tipo de movimiento.
- **Alta cohesión:** Se aplica en la mayoría las clases, donde en cada una solo se implementan las funcionalidades que le correspondientes.

Patrones GoF utilizados:

***State* (Estado):** Patrón de comportamiento que permite que un objeto modifique su comportamiento cada vez que cambie su estado interno. El uso de este patrón se evidencia claramente mediante el uso de un Modelo de Maquina de Estado y los diferentes estados creados. Donde en cada estado se tendrá un comportamiento (35).

***Observer* (Observador):** Patrón de comportamiento al igual que el anterior; define una dependencia de uno a muchos entre objetos. Se puede observar el uso de este patrón mediante la utilización de un centro de notificaciones que permite el posteo de las notificaciones cada vez que se realice un cambio de estado.

***Adapter* o *Wrapper* (Adaptador o Envoltorio):** Patrón estructural que adapta una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla. Usado principalmente para los componentes encargados del movimiento y la subida de niveles.

***Singleton* (Patrón Solitario):** Patrón creacional que garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Utilizado en las clases claves del sistema como el centro de notificaciones, encargado de postear las notificaciones para informar a las clases correspondientes.

***Factory Method* (Método de Fabricación):** Patrón creacional, que centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la diversidad de casos particulares que se pueden prever, para elegir el subtipo a crear. Un ejemplo notable de la utilización de este patrón se evidencia en la implementación de las habilidades, que presentan una diversidad de componentes de diferentes funcionalidades (35).

***Object Pool* (Piscina de Objetos):** Utilizado cuando el costo de crear una clase es mayor que el de clonarla. Especialmente con objetos muy complejos. De esta manera se obtienen objetos nuevos a través de la clonación. Se evidencia su uso con la utilización de *PoolData*, *GameObjectPoolController* y *Poolable*, este último define si un objeto se puede clonar o no. Usado principalmente para el trabajo con las interfaces (35).

2.7.3 Tarjetas CRC

En la fase de Diseño, la metodología XP no requiere la representación del sistema mediante diagramas de clases utilizando notación Lenguaje Unificado de Modelado (UML). En su lugar se usan otras técnicas como las tarjetas Clase Responsabilidad Colaboración (CRC) como una extensión informal a UML. La técnica de las tarjetas CRC se puede usar para guiar el sistema a través de análisis guiados por la responsabilidad. Las clases se examinan, se filtran y se refinan en base a sus responsabilidades con respecto al sistema y las clases con las que necesitan colaborar para completar sus responsabilidades.

A continuación, se muestran seis tarjetas CRC obtenidas durante la etapa de diseño de la presente investigación. Las descripciones de las tarjetas restantes se encuentran en el Anexo 1.

Tabla 10 Tarjeta CRC Turn

Tarjeta CRC	
Clase: Turn	
Responsabilidad	**Colaboración**

• Mantiene constancia de las unidades. • Mantiene constancia de las acciones realizadas por las unidades. • Mantiene constancia de las coordenadas en las que se encuentra la unidad.	• BattleController • Unit • Ability • Tile • PlanOfAttack

Tabla 11 Tarjeta CRC TurnOrderController

Tarjeta CRC	
Clase: TurnOrderController	
Responsabilidad	**Colaboración**
• Gestiona el orden de los turnos de las unidades en base a sus estadísticas y de las acciones realizadas por los mismas, que tendrán un costo o peso. • Monitorea y postea notificaciones dependiendo del estado en que se encuentra.	• BattleController • Stats • Unit • BaseException

Tabla 12 Tarjeta CRC BattleState

Tarjeta CRC	
Clase: BattleState	
Responsabilidad	**Colaboración**
• Mantener referencia al BattleController, que es una	• BattleController

subclase de StateMachine, que determina cuando un estado se encontrara 'activo'. • Mantiene referencia de los objetos en escena, pertenecientes a BattleController, necesario para que los estados desarrollen su propia lógica. • Permite que si las referencias del BattleController cambian o se actualizan, todos los estados sigan apuntando a la entidad correcta.	• State • Driver • CameraRig • Board • LevelData • Transform • Point • Tile • AbilityMenuPanelController • StatPanelController • HitSuccessIndicator • Turn • Unit • FacingIndicator • BaseVictoryCondition

Tabla 13 Tarjeta CRC Stats

Tarjeta CRC	
Clase: Stats	
Responsabilidad	**Colaboración**
• Enviar notificaciones que permitirán la modificación de las estadísticas.	• StatTypes • ValueChangeException

• Comprobar si las estadísticas fueron cambiadas o si se anuló el cambio.	

Tabla 14 Tarjeta CRC Rank

Tarjeta CRC	
Clase: Rank	
Responsabilidad	**Colaboración**
• Determina la curva de crecimiento de nivel de las unidades. • Delimita los límites de nivel alcanzables (mínimo y máximo), así como las máxima cantidad de experiencia posible.	• Stats • StatTypes • ValueChangeException • EasingEquations

Tabla 15 Tarjeta CRC ExperienceManager

Tarjeta CRC	
Clase: ExperienceManager	
Responsabilidad	**Colaboración**
• Mantiene referencia de todos los componentes Rank en una lista de cada uno de las unidades héroes que se encuentran en batalla. • Distribuye experiencia ganada en base al nivel de las unidades.	• Rank

2.8 Conclusiones parciales

Con la culminación de este capítulo, se logró obtener una propuesta de solución acorde a las necesidades del cliente. Se logró definir la estructura funcional del módulo, así como sus requisitos funcionales y no

funcionales. Se definieron las HU según la metodología utilizada, lográndose definir las funcionalidades prioritarias que debe presentar la solución para satisfacer al cliente. Se obtuvo una planificación para guiar al proceso de desarrollo e implementación de la presente investigación basado en la estimación propuesta por XP para las HU. Se definió una arquitectura a utilizar, que aprovechara las características de Unity y que permitiera una fácil readaptación, configuración y organización de la solución.

Capítulo 3: Implementación y pruebas

La metodología XP plantea que el proceso de desarrollo del software debe realizarse de forma iterativa, obteniendo al culminar cada iteración un producto funcional que debe ser probado y mostrado al cliente para retroalimentar a los desarrolladores con la opinión de este. En el presente capítulo se describen los estándares de codificación utilizados para obtener un código limpio y legible. Así como se hace referencia a las tareas ingenieriles que rigen el desarrollo del módulo y las pruebas realizadas al mismo. Con los resultados obtenidos se demuestra la eficiencia y el correcto funcionamiento de la solución que se desarrolló para dar solución al problema planteado.

3.1 Estándares de codificación

Según Guido van Rossum (científico de la computación, conocido por ser el autor del lenguaje de programación Python), el código es leído muchas más veces de lo que es escrito. XP enfatiza la comunicación de los programadores a través del código, por lo que es indispensable que se sigan ciertos estándares de programación (del equipo, de la organización u otros estándares reconocidos para los lenguajes de programación utilizados). Los estándares de programación mantienen el código legible para los miembros del equipo, facilitando los cambios (36).

Por lo que es necesario definir un conjunto de pautas que permitan una legibilidad concisa del código que lo haga consistente. Por lo que, para el desarrollo de este proyecto, se utilizó como base los estándares de codificación utilizados por C#, mediante el uso de “Reglas para codificación en C#”(37). Entre las pautas más destacadas de la guía citada, se destacan:

- Mantener las clases y ficheros cortos, con no más de 2 000 líneas de código y que estén claramente divididas en estructuras.
- Los comentarios de línea se utilizan para explicar línea a línea el código fuente. También se utilizan para comentar líneas de código temporalmente.
- Se recomienda realizar solo una declaración por línea, para permitir añadir un comentario explicativo a dicha declaración.
- Iniciar variables locales lo antes posible; si es posible, durante la declaración.
- Cada línea debe contener sólo una sentencia.
- Las líneas en blanco mejoran la legibilidad del código. Separan los bloques de código que están relacionados lógicamente.
- Para el convenio de nombres usar los estilos PasCal y caMel, en sus respectivos momentos.

3.2 Fase de implementación

Fase donde se realiza la implementación de las historias de usuario seleccionadas y se llevan a cabo las pruebas unitarias y de aceptación respectivas de cada una. En la inicialización de cada iteración, se realiza una revisión del plan de iteraciones, donde se realizan modificaciones de ser necesario. Las HU se descomponen en tareas de ingeniería; dichas tareas son utilizadas por los programadores. Las mismas pueden escribirse mediante la utilización del lenguaje técnico y no necesariamente tienen que ser entendibles para el cliente (22).

Siguiendo la planificación realizada, se realizan tres iteraciones de desarrollo, obteniéndose como resultado un producto listo para su utilización y despliegue. A continuación, se detallan las tareas ingenieriles por iteraciones.

3.2.1 Iteración 1

Iteración donde se abordan las HU de mayor prioridad y se construyen las bases arquitectónicas del sistema, con el objetivo de obtener prototipo con las funcionalidades críticas para ser mostrado al cliente y obtener a la vez, una retroalimentación con el mismo. Las restantes tareas ingenieriles de esta iteración y de iteraciones continuas, se podrán observar en el Anexo 2**: Tareas de ingeniería**.

HU - Sistema de movimiento por turno

Tabla 16 Tarea de ingeniería - StateMachine

Tarea de Ingeniería	
Nombre: StateMachine	
HU: Sistema de movimiento por turno	**Puntos Estimados:** 0.2
Fecha Inicio: 18/04/2018	**Fecha Fin:** 18/04/2018
Descripción:	Elemento de suma importancia que mantiene la referencia del estado actual activo y gestiona los cambios de estados.

Tabla 17 Tarea de ingeniería – Turn

Tarea de Ingeniería	
Nombre: Turn	
HU: Sistema de movimiento por turno	**Puntos Estimados:** 0.2
Fecha Inicio: 19/04/2018	**Fecha Fin:** 19/04/2018
Descripción:	Hace constancia de la información de las acciones que realizan las unidades en cada turno. Define lo que sería la constancia de un turno.

Tabla 18 Tarea de ingeniería – TurnOrderController

Tarea de Ingeniería	
Nombre: TurnOrderController	
HU: Sistema de movimiento por turno	**Puntos Estimados:** 0.2
Fecha Inicio: 20/04/2018	**Fecha Fin:** 20/04/2018
Descripción:	Gestiona la llegada de los turnos de cada unidad en base a sus estadísticas y la cantidad de acciones realizadas en el turno anterior, en base a un costo de acción.

Tabla 19 Tarea de ingeniería – BattleController

Tarea de Ingeniería	
Nombre: BattleController	
HU: Sistema de movimiento por turno	**Puntos Estimados:** 0.2
Fecha Inicio: 23/04/2018	**Fecha Fin:** 23/04/2018
Descripción:	Hace referencia a todos los componentes de relevancia que serán utilizados por el sistema. Da paso al InitBattleState, iniciador del estado de batalla.

Tabla 20 Tarea de ingeniería – InitBattleState

Tarea de Ingeniería	
Nombre: InitBattleState	
HU: Sistema de movimiento por turno	**Puntos Estimados:** 0.2
Fecha Inicio: 24/04/2018	**Fecha Fin:** 24/04/2018
Descripción:	Primer estado que instancia todos los elementos necesarios y que da paso al próximo estado. Entre los elementos instanciados de relevancia se encuentran los del terreno, las condiciones de victoria y las unidades.

Tabla 21 Tarea de ingeniería – BattleState

Tarea de Ingeniería	
Nombre: BattleState	
HU: Sistema de movimiento por turno	**Puntos Estimados:** 0.2

Fecha Inicio: 25/04/2018		**Fecha Fin:** 25/04/2018
Descripción:	Hace referencia al BattleController, que es una subclase de StateMachine, que determina cuando un estado se encontrará 'activo'. Mantendrá referencia a los objetos en escena, necesarios para que los estados desarrollen su propia lógica y para asegura que todos los estados sigan apuntando a la entidad correcta en caso de que se realicen cambios en el BattleController.	

Tabla 22 Tarea de ingeniería – Movement

Tarea de Ingeniería		
Nombre: Movement		
HU: Sistema de movimiento por turno		**Puntos Estimados:** 0.2
Fecha Inicio: 26/04/2018		**Fecha Fin:** 26/04/2018
Descripción:	Clase base que define los diferentes tipos de movimientos. Controlará la orientación de la unidad. Tendrá las funciones que realizarán la búsqueda de las baldosas que se encuentren en el rango de la unidad y obtendrá las mismas en base al rango de movimiento que presente la unidad.	

Tabla 23 Tarea de ingeniería – WalkMovement

Tarea de Ingeniería		
Nombre: WalkMovement		
HU: Sistema de movimiento por turno		**Puntos Estimados:** 0.2
Fecha Inicio: 27/04/2018		**Fecha Fin:** 27/04/2018
Descripción:	Modifica la función de búsqueda de las baldosas cercanas, así como la animación a realizar, en base a las características definidas por el tipo de movimiento que se le dé. En este caso se definió un tipo de movimiento normal a través de las baldosas, donde se mantiene en cuenta la altura posible a alcanzar por la unidad.	

Tabla 24 Tarea de ingeniería – MoveTargetState

Tarea de Ingeniería
Nombre: MoveTargetState

HU: Sistema de movimiento por turno	**Puntos Estimados:** 0.2
Fecha Inicio: 30/04/2018	**Fecha Fin:** 30/04/2018
Descripción:	Estado que permite interactuar con el terreno mediante el controlador de entradas (InputController). En el mismo se selecciona la ubicación a donde se desee desplazar a la unidad.

Tabla 25 Tarea de ingeniería – MoveSequenceState

Tarea de Ingeniería	
Nombre: MoveSequenceState	
HU: Sistema de movimiento por turno	**Puntos Estimados:** 0.2
Fecha Inicio: 01/05/2018	**Fecha Fin:** 01/05/2018
Descripción:	Estado que provoca la animación de recorrido del camino hacia la ubicación seleccionada y espera que se termine antes de regresar a el estado de selección de la unidad.

3.3 Pruebas

Las pruebas son otra parte fundamental de la metodología XP, que se dividen en Unitarias y de Aceptación. Las pruebas aseguran el correcto funcionamiento de los códigos implementados, permitiendo asegurar una calidad del producto y reducir la posibilidad de errores no descubiertos. A la vez, posibilita evitar el surgimiento de errores no deseados cuando se realizan modificaciones (20).

Las pruebas de aceptación son creadas de las historias de usuario, con el objetivo de verificar que se hayan cumplido las funcionalidades asignadas y satisfacer a su vez, las necesidades del cliente. Estas pruebas son pruebas de sistema de caja negra. Cada prueba de aceptación representa algún resultado esperado del sistema en una determina situación (19) y (22).

Para la realización de estas pruebas correctamente, se elaboraron casos de prueba para comprobar el funcionamiento de la solución en base a las HU planteadas. Seguidamente se describen los casos de prueba correspondientes a la HU Sistema de movimiento por turno.

Tabla 26 Caso de Prueba de Aceptación - Cambio de unidad

Caso de Prueba de Aceptación	
Nombre:	Cambio de unidad
Historia de Usuario:	Sistema de movimiento por turno

Descripción:	Se prueba que al finalizar un turno, se cambie de unidad correctamente.
Precondiciones:	Deben de existir al menos dos unidades en el escenario ya sean enemigas o heroicas.
Pasos de ejecución:	• Confirmar en el estado de EndFacingState la dirección en la que la unidad apuntará.
Resultados esperados	El TileSelectionIndicator se posiciona en la unidad correspondiente al cambiar de turno.

Tabla 27 Caso de Prueba de Aceptación - Movimiento – Selección de baldosa

Caso de Prueba de Aceptación	
Nombre:	Movimiento – Selección de baldosa
Historia de Usuario:	Sistema de movimiento por turno
Descripción:	Se comprueba la llegada del estado MoveTargetState. Se confirma su presencia mediante la posibilidad de seleccionar una baldosa dentro del rango posible de la unidad activa como objetivo de desplazamiento. Se comprueba la correcta selección de la baldosa objetivo al terminar el estado de MoveSequenceState.
Precondiciones:	Deben de existir al menos una unidad y un terreno en el escenario.
Pasos de ejecución:	• Seleccionar en el menú la acción de mover. • Desplazar el indicar y seleccionar una baldosa.
Resultados esperados	Las baldosas posibles a seleccionar cambiaran de color.

Tabla 28 Caso de Prueba de Aceptación - Movimiento - Animación

Caso de Prueba de Aceptación	
Nombre:	Movimiento - Animación
Historia de Usuario:	Sistema de movimiento por turno
Descripción:	Se comprueba la llegada del estado MoveSequenceState.

	Se confirma que se desplace la unidad activa hacia la baldosa objetivo seleccionada con anterioridad mediante la animación correspondiente al tipo de movimiento que tenga incorporado.
Precondiciones:	Deben de existir al menos una unidad y un terreno en el escenario.
Pasos de ejecución:	• Confirmar la baldosa objetivo en el estado MoveTargetState
Resultados esperados	La unidad se mueve con la animación correspondiente.

De las tres iteraciones realizadas, los resultados arrojados por las pruebas aplicadas demostraron que en la primera iteración se encontraron un total de tres no conformidades pertenecientes a la HU Sistema de movimiento por turno. En la segunda iteración se encontraron de igual manera tres no conformidades, dos pertenecientes a la HU Variedad de patrones de acción por la IA y una a Modificación de estadísticas por estados alterados. En la última iteración realizada no se encontraron inconformidades.

3.4 Conclusiones parciales

Con la realización de este capítulo se logró definir un estándar de codificación, que permitiera una mejor comprensión del código creado manteniéndolo legible y consistente. Permite a la vez, la posibilidad de realizar cambios con facilidad. También se elaboraron las tareas de ingeniería correspondientes a las funcionalidades descritas por las HU, obteniendo así un modelo de implementación viable para la presente investigación. Mediante las pruebas de aceptación, se pudieron corregir los errores detectados, permitiendo mejorar la calidad de la solución.

Conclusiones

En la presente investigación se llega a las siguientes conclusiones:

- La definición de las características distintivas de los videojuegos TRPG permitió identificar y separar por sub-módulos los elementos principales que definen este género.
- El módulo desarrollado viabiliza el proceso de desarrollo de los videojuegos de este género.

Recomendaciones

En esta investigación se recomienda lo siguiente:

- Seguir incorporando nuevas funcionalidades a cada uno de los sub-módulos que faciliten y mejoren el desarrollo de videojuegos de tipo TRPG.
- Utilizar el módulo brindado para la producción de videojuegos de tipo TRPG en el centro Vertex.

Referencias bibliográficas

1. Video Game Reviews, Articles, Trailers and more - Metacritic. *Metacritic* [online]. [Accessed 17 January 2018]. Available from: http://www.metacritic.com/game

2. U.S. most popular video game genres 2016 - Statista. *Statista* [online]. [Accessed 17 January 2018]. Available from: https://www.statista.com/statistics/189592/breakdown-of-us-video-game-sales-2009-by-genre/

3. ACAL, Daniel. ¿Qué es un juego de rol? *HobbyConsolas* [online]. 2 September 2013. [Accessed 10 June 2018]. Available from: https://www.hobbyconsolas.com/opinion/que-es-juego-rol-4169

4. MANUEL, Víctor. *Desarrollo de entorno para videojuegos en HTML5, basado en tiles*. Universidad Politécnica de Cataluña, 2014.

5. OU, Abenamar. [RETRO GAMING] Un poco de la historia de los RPG | OPINIÓN. *Yume* [online]. 24 July 2017. Available from: http://www.revistayumecr.com/retro-gaming-poco-la-historia-los-rpg-opinion

6. RPG - ElOtroLado. *ElOtroLado* [online]. [Accessed 22 June 2018]. Available from: https://www.elotrolado.net/wiki/RPG

7. TUOMINEN, Marko. *Multi-Player Tactical Role Playing Game with a Java server and an Android client*. Tampere University of Technology, 2012.

8. CILVA, David, DOYLE, Paul, HILL, Stephen and JUDISH, Rachel. *Web Based Tactical Role Playing Game in JavaScript*. June 2014.

9. BELLANGER, Clint. *Isometric Tiles Introduction*. 16 October 2016.

10. Recomendadme algun anime similar a Age of empires II - Off Topic. *Foro Meristation* [online]. [Accessed 25 June 2018]. Available from: http://meristation.as.com/zonaforo/topic/2379112/Página 1 de 2 - Recomendadme algun anime similar a Age of empires II - escribió en Off Topic: Recuerdo que cuando era mas joven probe IMPERIVM III pero no estaba a la altura

11. Diablo 2 - PC Review and Download | Old PC Gaming. [online]. [Accessed 25 June 2018]. Available from: http://www.oldpcgaming.net/diablo-2-review/Developer: Blizzard Genre: Action Role-Playing Release: June 29, 2000

12. PARRISH, Kevin. Facebook's CherryPi suffered defeat in "StarCraft," but a win for its AI team. *Digital Trends* [online]. 9 October 2017. [Accessed 25 June 2018]. Available from: https://www.digitaltrends.com/computing/facebook-cherrypi-ai-bit-loses-starcraft-brood-war/Facebook competed with independent developers ,institutions, and companies in StarCraft: Brood War using its CherryPi artificial intelligence.

13. Starcraft II | tuexpertojuegos.com. [online]. [Accessed 25 June 2018]. Available from: https://www.tuexpertojuegos.com/tag/starcraft-ii/

14. RIVAS, Jonathan. Creación de tiles isométricas usando el sistema SSR. *{}GameDev* [online]. 21 October 2013. Available from: https://www.gamedev.net/creacion-de-tiles-isometricas-usando-el-sistema-ssr

15. Quickly Learn to Create Isometric Games Like Clash of Clans or AOE. *"TheAppGuruz"* [online]. [Accessed 25 June 2018]. Available from: http://www.theappguruz.com/blog/create-isometric-games-like-clash-of-clans-crossy-roads-age-of-empire-etcQuickly Learn how to Create Isometric Games Like Clash of Clans or AOE. Quickly Create Isometric Games Like Clash of Clans or AOE Tutorial.

16. MUÑOZ, Mario. ¿Cómo se crea un videojuego? *FSGamer El Correo* [online]. 7 May 2015. Available from: http://www.fsgamer.com/como-se-crea-un-videojuego

17. ¿Cómo se crea un videojuego? *PerúEduca - Sistema Digital Para el Aprendizaje* [online]. 10 September 2016. Available from: http://www.perueduca.pe/como-se-crea-un-videojuego

18. ¿Sabes cómo se hace un video juego? *PlanetaCurioso* [online]. 14 July 2008. Available from: https://www.planetacurioso.com/sabes-como-se-hace-un-videojuego

19. ROGER S., Pressman. *Software Engineering: A Practitioner's Approach*. 7. McGraw-Hill, 2010. ISBN 978-0-07-301933-8. 2008048802

20. COCKBURN, A. *Selecting a project's methodology*. IEEE Software, 2000.

21. LETELIER TORRES, Patricio and SÁNCHEZ LÓPEZ, Emilio A. (eds.). *Metodologías Ágiles en el Desarrollo de Software*. Alicante – España, 2003.

22. KENT BECK. Extreme Programming Explained. *Embrace Change*. 2000.

23. ZERBST, Stefan and DUVEL, Oliver. *3D Game Engine Programming*. United States of America : Thomson Course Technology PTR, 2004. ISBN 1-59200-351-6.

24. Unity3D.College. *Unity3D.College* [online]. 22 November 2016. [Accessed 16 January 2018]. Available from: https://unity3d.college/2016/11/22/unity-extension-methods/Top Unity Tutorials, Tips, & Tricks

25. CHAVARRO RICO, Cristian Steven and ARBELAEZ, Mauricio Valencia. *Comparación de motores de videojuegos para la creación de juegos serios*. Universidad Tecnológica de Pereira, 2015.

26. Unity - Manual. *Unity* [online]. [Accessed 16 January 2018]. Available from: https://unity3d.com/Documentation/en/Manual

27. Asset Store. [online]. [Accessed 25 June 2018]. Available from: https://assetstore.unity.com/

28. ARCHER, Tom. *A Fondo C#*. España : McGraw-Hill, [no date]. ISBN 84-481-3246-7.

29. FERGUSON, Jeff, PATTERSON, Brian, BERES, Jason, BOUTQUIN, Pierre and GUPTA, Meeta. *La Biblia de C#*. Madrid - España : Anaya Multimedia, [no date]. ISBN 84-415-1484-4.

30. PESIS, Hernán. *Photoshop profesional*. 1. Argentina : Fox Andina, 2013. ISBN 978-987-1857-80-7.

31. MARTÍNEZ, Véronique and FUSTÉ, Joan (eds.). *Photoshop Newsletter*. Playa San Juan - Alicante (España) : Vaima, [no date]. ISBN 1139-3408.

32. SHARP, John. *Microsoft® Visual C# ® 2010 Step by Step*. United States of America : Microsoft Press, 2010. ISBN X16-81630.

33. SHEPHERD, George. *Microsoft® ASP.NET 4 Step by Step*. United States of America : Microsoft Press, 2010. ISBN X16-81630.

34. GUERRERO, Carlos A., SUÁREZ, Johanna M. and GUTIÉRREZ, Luz E. Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. *Información tecnológica*. 2013. Vol. 24, no. 3, p. 103–114. DOI 10.4067/S0718-07642013000300012.

35. FREEMAN, Eric, FREEMAN, Elisabeth, SIERRA, Kathy and BATES, Bert. *Head First Design Patterns*. 1. United States of America : O'Reilly Media, 2004. ISBN 0-596-00712-4.

36. LETELIER, Patricio. *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*. Universidad Politécnica de Valencia, [no date].

37. DIEZ, Leonardo. *Reglas para codificación en C#*. September 2003.

Bibliografía

1. ALGUNAS PECULIARIDADES DE UNITY. [en línea], 2017. [Consulta: 14 enero 2018]. Disponible en: http://unityscripts.blogspot.com/2011/10/algunas-peculiaridades-de-unity.html.

117 Games Like Valkyria Chronicles in 2018 – Games Like. [en línea], [sin fecha]. [Consulta: 18 enero 2018]. Disponible en: https://www.moregameslike.com/valkyria-chronicles/.

ACAL, D., 2013. ¿Qué es un juego de rol? HobbyConsolas [en línea]. [Consulta: 10 junio 2018]. Disponible en: https://www.hobbyconsolas.com/opinion/que-es-juego-rol-4169.

Adding to Unity's Built-In Classes Using Extension Methods. [en línea], [sin fecha]. [Consulta: 16 enero 2018]. Disponible en: https://www.gamasutra.com/blogs/JoshSutphin/20131007/201829/Adding_to_Unitys_BuiltIn_Classes_Using_Extension_Methods.php.

Algorithm for determining turn order in a turn-based rpg. - For Beginners - GameDev.net. [en línea], [sin fecha]. [Consulta: 18 enero 2018]. Disponible en: https://www.gamedev.net/forums/topic/574548-algorithm-for-determining-turn-order-in-a-turn-based-rpg/.

ANDRADES, G., [sin fecha]. Unity and the editor tools. [en línea]. [Consulta: 16 enero 2018]. Disponible en: https://www.gamasutra.com/blogs/GuillermoAndrades/20160302/267041/Unity_and_the_editor_tools.php.

ARCHER, T., [sin fecha]. A Fondo C#. España: McGraw-Hill. ISBN 84-481-3246-7.

Arquitectura basada en Componentes – Blog de Juan Peláez en Geeks.ms. [en línea], [sin fecha]. [Consulta: 16 mayo 2018]. Disponible en: https://geeks.ms/jkpelaez/2009/04/18/arquitectura-basada-en-componentes/.

BELLANGER, C., 2016. Isometric Tiles Introduction. 16 octubre 2016. S.l.: s.n.

c# - Extending Unity UI components with custom Inspector - Stack Overflow. [en línea], [sin fecha]. [Consulta: 16 enero 2018]. Disponible en: https://stackoverflow.com/questions/29052183/extending-unity-ui-components-with-custom-inspector.

C# - Patrones de Diseño – Estrategia. [en línea], [sin fecha]. [Consulta: 14 enero 2018]. Disponible en: https://msdn.microsoft.com/es-es/communitydocs/net-dev/csharp/patrones-de-diseno.

Centro de Entornos Interactivos 3D (Vertex) | Universidad de las Ciencias Informáticas. [en línea], [sin fecha]. [Consulta: 22 enero 2018]. Disponible en: http://www.uci.cu/investigacion-y-desarrollo/centros-de-desarrollo/centro-de-entornos-interactivos-3d-vertex.

CHAVARRO RICO, C.S. y ARBELAEZ, M.V., 2015. Comparación de motores de videojuegos para la creación de juegos serios. S.l.: Universidad Tecnológica de Pereira.

CILVA, D., DOYLE, P., HILL, S. y JUDISH, R., 2014. Web Based Tactical Role Playing Game in JavaScript. junio 2014. S.l.: s.n.

COCKBURN, A., 2000. Selecting a project's methodology. S.l.: IEEE Software.

CONGER, D. y LITTLE, R., 2006. Creating Games in C++: A Step-by-Step Guide. S.l.: New Riders. ISBN 0-7357-1434-7.

¿Cuál es la diferencia entre un juego de rol y | rpg-theory. [en línea], [sin fecha]. [Consulta: 10 junio 2018]. Disponible en: http://www.jugonestop.com/pregunta/540/cual-es-la-diferencia-entre-un-juego-de-rol-y-un-juego-de-mesa.

[Debate] RPG vs Juego de Rol en 3DJuegos. 3DJuegos [en línea], [sin fecha]. [Consulta: 10 junio 2018]. Disponible en: https://www.3djuegos.com/foros/tema/46117322/0/debate-rpg-vs-juego-de-rol/.

DIEZ, L., 2003. Reglas para codificación en C#. septiembre 2003. S.l.: s.n.

Displaying a digital countdown timer - Unity 5.x Cookbook [en línea], 2015. S.l.: s.n. [Consulta: 18 enero 2018]. ISBN 978-1-78439-136-2. Disponible en: https://www.packtpub.com/mapt/book/game_development/9781784391362/1/ch01lvl1sec12/displaying-a-digital-countdown-timer.

Extend the Unity3d Editor. Ray Wenderlich [en línea], [sin fecha]. [Consulta: 16 enero 2018]. Disponible en: https://www.raywenderlich.com/130721/extend-unity3d-editor.

FERGUSON, J., PATTERSON, B., BERES, J., BOUTQUIN, P. y GUPTA, M., [sin fecha]. La Biblia de C#. Madrid - España: Anaya Multimedia. ISBN 84-415-1484-4.

FREEMAN, Eric, FREEMAN, Elisabeth, SIERRA, K. y BATES, B., 2004. Head First Design Patterns. 1. United States of America: O'Reilly Media. ISBN 0-596-00712-4.

FUENTES AGUSTÍ, A., [sin fecha]. Desarrollo de un videojuego de aventuras en C# sobre Unity. S.l.: Universidad Politécnica de Valencia.

GUERRERO, C.A., SUÁREZ, J.M. y GUTIÉRREZ, L.E., 2013. Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. Información tecnológica, vol. 24, no. 3, pp. 103-114. ISSN 0718-0764. DOI 10.4067/S0718-07642013000300012.

How to make a turn-based RPG fantastic. Tech in Asia [en línea], 2014. [Consulta: 18 enero 2018]. Disponible en: https://www.techinasia.com/how-to-make-a-turn-based-rpg-fantastic.

ISMOMO1, 2015. Diagrama de clases. Unity3Dtutorial [en línea]. [Consulta: 14 enero 2018]. Disponible en: https://unity3dtutorial.wordpress.com/2015/11/12/diagrama-de-clases/.

J. AYANGBEKUN, O. y O. AKINDE, I., 2014. Development of a Real-Time Strategy Game. Abeokuta, Nigeria: Crescent University.

KONOVALOV, S. y MISSLINGER, S., 2006. Extreme Programming. 23 mayo 2006. S.l.: s.n.

La Guía de Arquitectura Versión 2.0a. [en línea], [sin fecha]. [Consulta: 16 mayo 2018]. Disponible en: https://geeks.ms/jkpelaez/2009/04/18/arquitectura-basada-en-componentes/.

LANKENAU, D., 2014. Development of a Tactics RPG for Android Tablets - A Major Qualifying Project Report. S.l.: Worcester Polytechnic Institute.

LETELIER, P., [sin fecha]. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). S.l.: Universidad Politécnica de Valencia.

LETELIER TORRES, P. y SÁNCHEZ LÓPEZ, E.A., 2003. Metodologías Ágiles en el Desarrollo de Software. Alicante – España: s.n.

Level 5.5: Prototyping Real-Time Systems: Game Design Concepts. [en línea], [sin fecha]. [Consulta: 18 enero 2018]. Disponible en: https://learn.canvas.net/courses/3/pages/level-5-dot-5-prototyping-real-time-systems.

MANUEL, V., 2014. Desarrollo de entorno para videojuegos en HTML5, basado en tiles. S.l.: Universidad Politécnica de Cataluña.

MCLAUGHLIN, B.D., POLLICE, G. y WEST, D., 2006. Head First Object-Oriented Analysis and Design. 1. United States of America: O'Reilly Media. ISBN 0-596-00867-8.

Misión | Universidad de las Ciencias Informáticas. [en línea], [sin fecha]. [Consulta: 22 enero 2018]. Disponible en: http://www.uci.cu/universidad/mision.

MUÑOZ, M., 2015. ¿Cómo se crea un videojuego? FSGamer El Correo [en línea]. Disponible en: http://www.fsgamer.com/como-se-crea-un-videojuego.

NATHALE SOARES, A., FLÁVIA GAZZINELLI, M., DE SOUZA, V. y LOBATO ARAÚJO, L.H., 2014. The Role Playing Game (RPG) as a pedagogical strategy in the trainning of the nurse: an experience report on the creation of a game. , pp. 600-608.

Object.Equals Method (Object) (System). [en línea], [sin fecha]. [Consulta: 16 enero 2018]. Disponible en: https://msdn.microsoft.com/en-us/library/bsc2ak47(v=vs.110).aspx.

Object.GetHashCode Method (System). [en línea], [sin fecha]. [Consulta: 16 enero 2018]. Disponible en: https://msdn.microsoft.com/en-us/library/system.object.gethashcode(v=vs.110).aspx.

OU, A., 2017. [RETRO GAMING] Un poco de la historia de los RPG | OPINIÓN. Yume [en línea]. Disponible en: http://www.revistayumecr.com/retro-gaming-poco-la-historia-los-rpg-opinion.

Patrones GRASP. Patrones GoF. Diferencia entre GRASP y GoF - TuxNots. [en línea], [sin fecha]. [Consulta: 16 mayo 2018]. Disponible en: http://tuxnots.com.ar/materias-de-la-facu/metodologia-de-sistemas/patronesgrasppatronesgofdiferenciaentregraspygof.

(primero), [sin fecha]. 6 - Extending Unity. [en línea]. [Consulta: 16 enero 2018]. Disponible en: https://msdn.microsoft.com/en-us/library/dn178462(v=pandp.30).aspx.

RIVAS, J., 2013. Creación de tiles isométricas usando el sistema SSR. {}GameDev [en línea]. Disponible en: https://www.gamedev.net/creacion-de-tiles-isometricas-usando-el-sistema-ssr.

ROGER S., P., 2010. Software Engineering: A Practitioner's Approach. 7. S.l.: McGraw-Hill. ISBN 978-0-07-301933-8. 2008048802

RPG - ElOtroLado. ElOtroLado [en línea], [sin fecha]. [Consulta: 22 junio 2018]. Disponible en: https://www.elotrolado.net/wiki/RPG.

¿Sabes cómo se hace un video juego? PlanetaCurioso [en línea], 2008. Disponible en: https://www.planetacurioso.com/sabes-como-se-hace-un-videojuego.

TAL, L., 2015. Extending Unity's Default Inspectors. Random Bits [en línea]. [Consulta: 16 enero 2018]. Disponible en: http://www.tallior.com/extending-unity-inspectors/.

The Top Video Games Of 2017. [en línea], [sin fecha]. [Consulta: 17 enero 2018]. Disponible en: https://www.forbes.com/sites/erikkain/2017/01/23/the-top-video-games-of-2017/#736ea3f31475.

Top 10 Greatest Video-Game Genres - TheTopTens®. TheTopTens [en línea], [sin fecha]. [Consulta: 17 enero 2018]. Disponible en: https://www.thetoptens.com/video-game-genres/.

TUOMINEN, M., 2012. Multi-Player Tactical Role Playing Game with a Java server and an Android client. S.l.: Tampere University of Technology.

Turn Based RPGs for beginners • r/patientgamers. reddit [en línea], [sin fecha]. [Consulta: 18 enero 2018]. Disponible en: https://www.reddit.com/r/patientgamers/comments/2eogac/turn_based_rpgs_for_beginners/.

Unity - Building a Custom Inspector. Unity [en línea], [sin fecha]. [Consulta: 16 enero 2018]. Disponible en: https://unity3d.com/es/learn/tutorials/topics/interface-essentials/building-custom-inspector.

Unity - Extension Methods. Unity [en línea], [sin fecha]. [Consulta: 16 enero 2018]. Disponible en: https://unity3d.com/es/learn/tutorials/topics/scripting/extension-methods.

Unity - Manual. Unity [en línea], [sin fecha]. [Consulta: 16 enero 2018]. Disponible en: https://unity3d.com/Documentation/en/Manual.

Unity Tutorials | Tactics RPG Systems. [en línea], [sin fecha]. [Consulta: 18 enero 2018]. Disponible en: http://unitytutorials.ca/systems/full-games/tactics-rpg-systems.

[Unity] Unity3D overview of basic concepts to look for beginners (a): GameObject. [en línea], [sin fecha]. [Consulta: 14 enero 2018]. Disponible en: http://www.programering.com/a/MDO4UDNwATY.html.

Unity3D.College. Unity3D.College [en línea], 2016. [Consulta: 16 enero 2018]. Disponible en: https://unity3d.college/2016/11/22/unity-extension-methods/.

U.S. most popular video game genres 2016 - Statista. Statista [en línea], [sin fecha]. [Consulta: 17 enero 2018]. Disponible en: https://www.statista.com/statistics/189592/breakdown-of-us-video-game-sales-2009-by-genre/.

Video Game Reviews, Articles, Trailers and more - Metacritic. Metacritic [en línea], [sin fecha]. [Consulta: 17 enero 2018]. Disponible en: http://www.metacritic.com/game.

VLOJ, M., [sin fecha]. Hibernate - Un poco de Teoría y ejemplos · Mai Vloj. [en línea]. [Consulta: 16 enero 2018]. Disponible en: https://radw2020.github.io/2017/01/18/Hibernate-Teor%C3%ADa-y-Ejemplos/.

ZUCCONI, A., 2015. Extension methods in C#. Alan Zucconi [en línea]. [Consulta: 16 enero 2018]. Disponible en: https://www.alanzucconi.com/2015/08/05/extension-methods-in-c/.

MARTÍNEZ, V. y FUSTÉ, J., [sin fecha]. Photoshop Newsletter. Playa San Juan - Alicante (España): Vaima. ISBN 1139-3408.

PESIS, H., 2013. Photoshop profesional. 1. Argentina: Fox Andina. ISBN 978-987-1857-80-7.

SHARP, J., 2010a. Microsoft® Visual C# ® 2010 Step by Step. United States of America: Microsoft Press. ISBN X16-81630.

SHARP, J., 2010b. Windows® Communication Foundation 4 Step by Step. United States of America: O'Reilly Media. ISBN 978-0-7356-4556-1.

SHEPHERD, G., 2010. Microsoft® ASP.NET 4 Step by Step. United States of America: Microsoft Press. ISBN X16-81630.

EBERLY, D.H., [sin fecha]. 3D Game Engine Design. S.l.: Morgan Kaufmann Publishers.

GREGORY, J., 2009. Game Engine Architecture. United States of America: A K Peters, Ltd. ISBN 978-1-4398-6526-2.

ZERBST, S. y DUVEL, O., 2004. 3D Game Engine Programming. United States of America: Thomson Course Technology PTR. ISBN 1-59200-351-6.

BOEYKENS, S., 2013. Unity for Architectural Visualization. S.l.: Packt Publishing Ltd. ISBN 978-1-78355-906-0.

Anexo 1: Tarjetas CRC

Tabla 29 Tarjeta CRC BattleController

Tarjeta CRC	
Clase: BattleController	
Responsabilidad	**Colaboración**
• Mantiene referencia de todos los objetos en escena y componentes de importancia. • Da inicio a el combate o estado inicial de batalla.	• StateMachine • StatPanelController • AbilityMenuPanelController • BattleMessageController • HitSuccessIndicator • InitBattleState • ComputerPlayer • Turn • CameraRig • Board • LevelData • Transform • FacingIndicator • Tile • Point • Unit

Tabla 30 Tarjeta CRC InitBattleState

Tarjeta CRC	
Clase: InitBattleState	
Responsabilidad	**Colaboración**
• Crea los elementos objetos y componentes necesarios en la escena antes de dar paso al siguiente estado. • Cambia al estado de CutSceneState.	• BattleState • Tile • UnitFactory • Unit • DefeatTargetVictoryCondition • Health • TurnOrderController • CutSceneState

Tabla 31 Tarjeta CRC Unit

Tarjeta CRC	
Clase: Unit	
Responsabilidad	**Colaboración**
• Define que es una unidad en la lógica del sistema. • Permitirá hacer referencia a elementos que son características de una unidad. • Asegura de que se deje de hacer referencia a una ubicación antigua	• Tile • Directions • Transform

después de que la unidad se ha movido. • Vincula la referencia de la unidad y la baldosa donde se localiza. • Asegura de que la unidad sea ubicada en el centro de la baldosa.	

Tabla 32 Tarjeta CRC Tile

Tarjeta CRC	
Clase: Tile	
Responsabilidad	**Colaboración**
• Define que es un tile en la lógica del sistema. • Hace referencia a los objetos que se encuentren en la parte superior del tile. • Permite el crecimiento y reducción de los tiles. • Convierte y guarda la información del tile en vector3. • Convierte un vector3 en valores de posición y altura para ser usados por el sistema. • Actualiza la información del tile cada vez que es modificado.	• Point • GameObject • Vector3 • Transform

Tabla 33 Tarjeta CRC ComputerPlayer

Tarjeta CRC	
Clase: ComputerPlayer	
Responsabilidad	**Colaboración**
• Encargado de gestionar los componentes responsables de la A.I. • Crear y rellena un plan de ataque. • Comprueba si es posible realizar el plan de ataque. • Comprueba y calcula cual es la ubicación más provechosa para aplicar el plan de ataque. • Comprueba y calcula cual es la mejor acción a tomar. • Busca cual es el enemigo más cercano. • Se acerca al enemigo más cercano si no es posible aplicar el plan de ataque. • Determina la orientación de la unidad al terminar el turno.	• BattleController • Unit • Alliance • PlanOfAttack • AttackPattern • Ability • AbilityRange • Tile • AttackOption • Directions • Movement • Targets • Stats • StatTypes

Tabla 34 Tarjeta CRC AttackOption

Tarjeta CRC	
Clase: AttackOption	
Responsabilidad	**Colaboración**

• Impide que la IA se mueva a un tile que pueda tener un efecto negativo para la unidad. • Agrega una marca en los tiles que tengan una unidad del tipo deseado para la aplicación de la acción tomada, como constancia de su ubicación. • Puntea la acción a tomar en base a cuantos objetivos son del tipo deseado. • Devuelve la localización del tile que resulte más efectivo para la aplicación de la opción tomada. • Indica cual es el ángulo más efectivo para la aplicación de la acción, si esta lo requiere. • Puntea la acción a tomar en base a cuantos objetivos son del tipo deseado y se considera el ángulo de ataque de cada marca.	• Mark • Tile • Directions • Unit • Ability • HitRate • Facings

Tabla 35 Tarjeta CRC AttackPattern

Tarjeta CRC	
Clase: AttackPattern	
Responsabilidad	**Colaboración**
• Organiza los selectores de habilidad.	• BaseAbilityPicker • PlanOfAttack

• Crear los patrones de ataque que determinan el comportamiento de la IA.	

Tabla 36 Tarjeta CRC Ability

Tarjeta CRC	
Clase: Ability	
Responsabilidad	**Colaboración**
• Comprueba si se puede ejecutar una habilidad. • Aplica el daño y efectos que pueda tener la habilidad a ejecutar. • Comprueba si el objetivo seleccionado a aplicar la habilidad es válido.	• BaseException • Tile • Transform • BaseAbilityEffect • AbilityEffectTarget

Tabla 37 Tarjeta CRC AbilityArea

Tarjeta CRC	
Clase: AbilityArea	
Responsabilidad	**Colaboración**
• Determina el área de efecto o área efectiva que poseerá la habilidad.	• Tile • Board • Point

Tabla 38 Tarjeta CRC AbilityRange

Tarjeta CRC

Clase: AbilityRange	
Responsabilidad	**Colaboración**
• Determina el alcance que poseerá la habilidad. • Permite determinar la orientación a apuntar en caso de que la habilidad sea un patrón establecido. • Obtiene los tiles que se encuentran en el rango de la habilidad.	• Tile • Board • Unit

Tabla 39 Tarjeta CRC BaseAbilityPower

Tarjeta CRC	
Clase: BaseAbilityPower	
Responsabilidad	**Colaboración**
• Maneja conjunto de notificaciones relacionadas con el cálculo de daño. • Agrega modificadores de valores para la realización del cálculo de daño. • Determina la potencia que tendrá una habilidad dependiendo de su tipo.	• ValueModifier • BaseAbilityEffect • Unit

Tabla 40 Tarjeta CRC HitRate

Tarjeta CRC
Clase: HitRate

Responsabilidad	Colaboración
• Calcula las probabilidades de acierto de una acción.	• Tile • MatchException • Unit

Tabla 41 Tarjeta CRC BaseAbilityEffect

Tarjeta CRC	
Clase: BaseAbilityEffect	
Responsabilidad	**Colaboración**
• Maneja las notificaciones y los datos necesarios para realizar el cálculo de la probabilidad de acierto de una acción. • Maneja las notificaciones y los datos necesarios para realizar el cálculo de daño. • Aplica los efectos de las habilidades dependiendo de su tipo. • Ajusta el daño a aplicar dentro de un rango preestablecido.	• Tile • AbilityEffectTarget • HitRate • Unit • ValueModifier

Tabla 42 Tarjeta CRC AbilityEffectTarget

Tarjeta CRC	
Clase: AbilityEffectTarget	
Responsabilidad	**Colaboración**

• Determina el comportamiento de la habilidad dependiendo del tipo de unidad objetivo. • Determina cuando se aplica el efecto.	• Tile • Stats • Alliance • BaseAbilityEffect

Tabla 43 Tarjeta CRC Movement

Tarjeta CRC	
Clase: Movement	
Responsabilidad	**Colaboración**
• Determina el rango de movimiento de la unidad. • Aplica la animación correspondiente de desplazamiento en base al tipo de movimiento. • Reconoce los obstáculos y las baldosas ocupadas.	• Unit • Transform • Stats • Tile • TransformLocalEulerTweener • StatTypes • Tweener

Tabla 44 Tarjeta CRC Board

Tarjeta CRC	
Clase: Board	
Responsabilidad	**Colaboración**
• Cambia de color las baldosas para indicar el rango de una acción. • Representa los puntos cardinales.	• Tile • Point • LevelData

• Carga un terreno previamente hecho. • Realiza la función del camino más corto.	

Tabla 45 Tarjeta CRC BoardCreator

Tarjeta CRC	
Clase: BoardCreator	
Responsabilidad	**Colaboración**
• Crear terrenos. • Definir límites del terreno. • Guardar terrenos. • Cargar terrenos previamente realizados. • Limpiar el área de trabajo.	• Tile • Point • Transform • LevelData • Vector3 • GameObject

Tabla 46 Tarjeta CRC Status

Tarjeta CRC	
Clase: Status	
Responsabilidad	**Colaboración**
• Determina la duración en que permanece aplicado un estado alterado. • Elimina el estado alterado aplicado cuando termine su duración.	• StatusCondition • StatusEffect

• Define que es un estado alterado en la lógica del sistema.	

Tabla 47 Tarjeta CRC Job

Tarjeta CRC	
Clase: Job	
Responsabilidad	**Colaboración**
• Establecer las estadísticas base de la unidad cuando es creada en base al trabajo que lleve. • Modificar las estadísticas de la unidad después de subir de nivel. • Define que es un trabajo en la lógica del sistema.	• StatTypes • Stats • Feature

Tabla 48 Tarjeta CRC UnitFactory

Tarjeta CRC	
Clase: UnitFactory	
Responsabilidad	**Colaboración**
• Crear las unidades en la escena según los parámetros especificados en las recetas cargadas.	• GameObject • UnitRecipe

Tabla 49 Tarjeta CRC StateMachine

Tarjeta CRC
Clase: StateMachine

Responsabilidad	Colaboración
• Mantener referencia del estado actual. • Administrar el cambio de estados. • Impedir que suceda un cambio de estado durante una transición. • Marcar el inicio de las transiciones.	• State

Tabla 50 Tarjeta CRC NotificationCenter

Tarjeta CRC	
Clase: NotificationCenter	
Responsabilidad	**Colaboración**
• Permitir la comunicación entre las clases mediante el envío de notificaciones.	• Handler • SenderTable

Tabla 51 Tarjeta CRC EasingControl

Tarjeta CRC	
Clase: EasingControl	
Responsabilidad	**Colaboración**
• Gestiona las animaciones. • Define los estados en que se encuentra el juego.	• EventHandler • TimeType • PlayState • EndBehaviour • LoopType

Anexo 2: Tareas de ingeniería

HU - Sistema de estadísticas de unidades

Tabla 52 Tarea de ingeniería - StateMachine

Tarea de Ingeniería	
Nombre: StatTypes	
HU: Sistema de estadísticas de unidades	**Puntos Estimados:** 0.1
Fecha Inicio: 02/05/2018	**Fecha Fin:** 02/05/2018
Descripción:	Se definen los tipos de estadísticas que poseerán las unidades.

Tabla 53 Tarea de ingeniería - Stats

Tarea de Ingeniería	
Nombre: Stats	
HU: Sistema de estadísticas de unidades	**Puntos Estimados:** 0.2
Fecha Inicio: 02/05/2018	**Fecha Fin:** 02/05/2018
Descripción:	Componente que define las estadísticas que tendrá cada unidad. Controla las modificaciones a realizar sobre las estadísticas.

Tabla 54 Tarea de ingeniería - Rank

Tarea de Ingeniería	
Nombre: Rank	
HU: Sistema de estadísticas de unidades	**Puntos Estimados:** 0.2
Fecha Inicio: 03/05/2018	**Fecha Fin:** 03/05/2018
Descripción:	Determina la curva de crecimiento de las estadísticas con respecto al nivel que posea la unidad.

Tabla 55 Tarea de ingeniería - Job

Tarea de Ingeniería	
Nombre: Job	
HU: Sistema de estadísticas de unidades	**Puntos Estimados:** 0.2
Fecha Inicio: 04/05/2018	**Fecha Fin:** 04/05/2018

Descripción:	Define que es un trabajo en la lógica del sistema. Ajusta las estadísticas de la unidad con respecto al trabajo que posea.

Tabla 56 Tarea de ingeniería - Health

Tarea de Ingeniería	
Nombre: Health	
HU: Sistema de estadísticas de unidades	**Puntos Estimados:** 0.2
Fecha Inicio: 07/05/2018	**Fecha Fin:** 07/05/2018
Descripción:	Asegura de que la vida se mantenga dentro de un rango establecido; de que no vaya por debajo de cero y ni por encima de sus puntos máximos.

Tabla 57 Tarea de ingeniería - Mana

Tarea de Ingeniería	
Nombre: Mana	
HU: Sistema de estadísticas de unidades	**Puntos Estimados:** 0.1
Fecha Inicio: 08/05/2018	**Fecha Fin:** 08/05/2018
Descripción:	Asegura de que la energía mágica se mantenga dentro de un rango establecido; de que no vaya por debajo de cero y ni por encima de sus puntos máximos. En la llegada de cada turno aumenta en un uno por ciento del total el valor actual.

Iteración 2:

HU - Sistema de movimiento por turno

Tabla 58 Tarea de ingeniería - BaseException

Tarea de Ingeniería	
Nombre: BaseException	
HU: Modificación de estadísticas por estados alterados	**Puntos Estimados:** 0.1
Fecha Inicio: 09/05/2018	**Fecha Fin:** 09/05/2018
Descripción:	Define excepciones de la regla para cuando ocurre un cambio de estadística.

Tabla 59 Tarea de ingeniería - Modifier

Tarea de Ingeniería	
Nombre: Modifier	
HU: Modificación de estadísticas por estados alterados	**Puntos Estimados:** 0.1
Fecha Inicio: 09/05/2018	**Fecha Fin:** 09/05/2018
Descripción:	Clase base de los modificadores que realizan el cálculo de la variación de las estadísticas. Ordena los modificadores.

Tabla 60 Tarea de ingeniería - ValueModifier

Tarea de Ingeniería	
Nombre: ValueModifier	
HU: Modificación de estadísticas por estados alterados	**Puntos Estimados:** 0.1
Fecha Inicio: 09/05/2018	**Fecha Fin:** 09/05/2018
Descripción:	Clase base abstracta con objetivo arquitectónico que encapsula a los modificadores de valores.

Tabla 61 Tarea de ingeniería - AddValueModifier

Tarea de Ingeniería	
Nombre: AddValueModifier	
HU: Modificación de estadísticas por estados alterados	**Puntos Estimados:** 0.1
Fecha Inicio: 10/05/2018	**Fecha Fin:** 10/05/2018
Descripción:	Suma los valores.

Tabla 62 Tarea de ingeniería - ClampValueModifier

Tarea de Ingeniería	
Nombre: ClampValueModifier	
HU: Modificación de estadísticas por estados alterados	**Puntos Estimados:** 0.1
Fecha Inicio: 10/05/2018	**Fecha Fin:** 10/05/2018

Descripción:	Sujeta los valores máximo y mínimo.

Tabla 63 Tarea de ingeniería - MaxValueModifier

Tarea de Ingeniería	
Nombre: MaxValueModifier	
HU: Modificación de estadísticas por estados alterados	**Puntos Estimados:** 0.1
Fecha Inicio: 10/05/2018	**Fecha Fin:** 10/05/2018
Descripción:	Obtiene el valor máximo.

Tabla 64 Tarea de ingeniería - MinValueModifier

Tarea de Ingeniería	
Nombre: MinValueModifier	
HU: Modificación de estadísticas por estados alterados	**Puntos Estimados:** 0.1
Fecha Inicio: 11/05/2018	**Fecha Fin:** 11/05/2018
Descripción:	Obtiene el valor mínimo.

Tabla 65 Tarea de ingeniería - MultDeltaModifier

Tarea de Ingeniería	
Nombre: MultDeltaModifier	
HU: Modificación de estadísticas por estados alterados	**Puntos Estimados:** 0.1
Fecha Inicio: 11/05/2018	**Fecha Fin:** 11/05/2018
Descripción:	Modifica el valor en base a un delta.

Tabla 66 Tarea de ingeniería - MultValueModifier

Tarea de Ingeniería	
Nombre: MultValueModifier	
HU: Modificación de estadísticas por estados alterados	**Puntos Estimados:** 0.1
Fecha Inicio: 11/05/2018	**Fecha Fin:** 11/05/2018
Descripción:	Multiplica los valores.

Tabla 67 Tarea de ingeniería - ValueChangeException

Tarea de Ingeniería	
Nombre: ValueChangeException	
HU: Modificación de estadísticas por estados alterados	**Puntos Estimados:** 0.3
Fecha Inicio: 14/05/2018	**Fecha Fin:** 14/05/2018
Descripción:	Mantiene una lista de modificadores de valores para modificar el valor asignado en una excepción.

Tabla 68 Tarea de ingeniería - Feature

Tarea de Ingeniería	
Nombre: Feature	
HU: Modificación de estadísticas por estados alterados	**Puntos Estimados:** 0.1
Fecha Inicio: 15/05/2018	**Fecha Fin:** 15/05/2018
Descripción:	Permite la modificación de estadísticas mediante dos usos. El primero para casos de modificación temporal. El segundo para casos de modificación permanente.

Tabla 69 Tarea de ingeniería - StatModifierFeature

Tarea de Ingeniería	
Nombre: StatModifierFeature	
HU: Modificación de estadísticas por estados alterados	**Puntos Estimados:** 0.1
Fecha Inicio: 16/05/2018	**Fecha Fin:** 16/05/2018
Descripción:	Permite modificar cualquier estadística para cualquier estado alterado beneficioso o negativo.

Tabla 70 Tarea de ingeniería - Status

Tarea de Ingeniería
Nombre: Status

HU: Modificación de estadísticas por estados alterados	**Puntos Estimados:** 0.1
Fecha Inicio: 17/05/2018	**Fecha Fin:** 17/05/2018
Descripción:	Define que es un estado alterado en la lógica del sistema. Determina la duración de un estado alterado.

HU - Variedad de patrones de acción por la IA

Tabla 71 Tarea de ingeniería - ComputerPlayer

Tarea de Ingeniería	
Nombre: ComputerPlayer	
HU: Variedad de patrones de acción por la IA	**Puntos Estimados:** 0.9
Fecha Inicio: 18/05/2018	**Fecha Fin:** 24/05/2018
Descripción:	Gestiona los diferentes componentes que definen el comportamiento de la IA en combate.

Tabla 72 Tarea de ingeniería - PlanOfAttack

Tarea de Ingeniería	
Nombre: PlanOfAttack	
HU: Variedad de patrones de acción por la IA	**Puntos Estimados:** 0.1
Fecha Inicio: 24/05/2018	**Fecha Fin:** 24/05/2018
Descripción:	Guarda la referencia de un conjunto de datos que definen la mejor acción a tomar por la IA.

Tabla 73 Tarea de ingeniería - AttackPattern

Tarea de Ingeniería	
Nombre: AttackPattern	
HU: Variedad de patrones de acción por la IA.	**Puntos Estimados:** 0.1
Fecha Inicio: 25/05/2018	**Fecha Fin:** 25/05/2018

Descripción:	Organiza todos los seleccionadores de habilidad en una lista secuencial. Permite la creación de patrones de ataque que contendrán los comandos posibles a utilizar por la IA.

Tabla 74 Tarea de ingeniería - AttackOption

Tarea de Ingeniería	
Nombre: AttackOption	
HU: Variedad de patrones de acción por la IA.	**Puntos Estimados:** 0.2
Fecha Inicio: 25/05/2018	**Fecha Fin:** 28/05/2018
Descripción:	Calcula y marca los lugares beneficiosos a utilizar. Puntea y elige la mejor habilidad posibles.

Tabla 75 Tarea de ingeniería - BaseAbilityPicker

Tarea de Ingeniería	
Nombre: BaseAbilityPicker	
HU: Variedad de patrones de acción por la IA.	**Puntos Estimados:** 0.2
Fecha Inicio: 29/05/2018	**Fecha Fin:** 29/05/2018
Descripción:	Indica que se quiere usar una habilidad determinada.

Iteración 3:

HU - Creación de terreno

Tabla 76 Tarea de ingeniería - BoardCreator

Tarea de Ingeniería	
Nombre: BoardCreator	
HU: Creación de terreno	**Puntos Estimados:** 0.6
Fecha Inicio: 30/05/2018	**Fecha Fin:** 01/06/2018
Descripción:	Crea, guarda y carga los terrenos. Define los límites que tendrá el terreno.

Tabla 77 Tarea de ingeniería - Board

Tarea de Ingeniería	
Nombre: Board	
HU: Creación de terreno	**Puntos Estimados:** 0.6
Fecha Inicio: 04/06/2018	**Fecha Fin:** 06/06/2018
Descripción:	Cambia de color las baldosas para indicar el rango de una acción. Carga un terreno previamente hecho. Realiza la función del camino más corto. Define que es el terreno para la lógica del sistema.

Tabla 78 Tarea de ingeniería - Point

Tarea de Ingeniería	
Nombre: Point	
HU: Creación de terreno	**Puntos Estimados:** 0.4
Fecha Inicio: 07/06/2018	**Fecha Fin:** 08/06/2018
Descripción:	Mantiene las coordenadas del tile en el terreno.

Tabla 79 Tarea de ingeniería - LevelData

Tarea de Ingeniería	
Nombre: LevelData	
HU: Creación de terreno	**Puntos Estimados:** 0.1
Fecha Inicio: 11/06/2018	**Fecha Fin:** 11/06/2018
Descripción:	Guarda la información de las coordenadas de cada tile que conforma el terreno.

Tabla 80 Tarea de ingeniería - Tile

Tarea de Ingeniería	
Nombre: Tile	
HU: Creación de terreno	**Puntos Estimados:** 0.3
Fecha Inicio: 11/06/2018	**Fecha Fin:** 12/06/2018
Descripción:	Define que es un tile en la lógica del sistema. Hace referencia a los objetos que se encuentren en la parte superior del tile. Actualiza la información del tile cada vez que es modificado.

Anexo 2: Tareas de ingeniería