UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 5

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

Título: “Prototipo de videojuego RPG de base por turnos”

Autor:

Dashiel Lázaro Portel Batista

Tutores:

Ing. Roberto E. Pérez Ozete

Ing. Victor Manuel Armas Pis

La Habana, junio de 2016

“Año 58 de la Revolución”

"Si supiese que es lo que estoy haciendo, no lo llamaría investigación, ¿verdad?"

Albert Einstein

DECLARACIÓN DE AUTORÍA

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

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

Dashiel Lázaro Portel Batista

_____

Firma del Autor

Ing. Roberto E. Pérez Ozete

_____

Firma del Tutor

Ing. Victor Manuel Armas Pis

_____

Firma del Tutor

DEDICATORIA

A mi familia que tanto me han dado y especialmente a mi abuela.

Resumen

Los videojuegos son productos cada vez más complejos, lo que requiere un gasto considerable de recursos humanos y presupuesto. Compañías y empresas de toda índole han optado por el desarrollo de prototipos, que sirvan de modelo para desarrollar futuros videojuegos, ahorrando tiempo en la decisión de que mecánicas se implementarán y cuáles no, y de qué forma deberían implementarse, así como su implementación en sí. También, los prototipos de videojuegos permiten de manera rápida, probar nuevas ideas en el público, obteniendo información concisa de los puntos negativos y positivos de esta. El presente trabajo refleja el proceso de desarrollo de un prototipo de videojuego RPG (*Role Playing Game*) de base por turnos, que permita: la selección de clases, aumentar los atributos dependiendo de la selección del usuario, aprender habilidades de combate, reconocimiento de terreno y un sistema de batalla por turnos. Además, el empleo de un modelo matemático basado en funciones logarítmicas, exponenciales y lineales para el aumento de los atributos no controlados por el usuario, que influyen en la evolución del personaje.

Palabras claves: Base por turnos, Prototipo, RPG, Videojuegos.

Índice de contenido

ÍNDICE

Índice de figuras

Índice de diagramas

Introducción

Los videojuegos como medio de entretenimiento han evolucionado en los últimos años, convirtiéndose en una industria multimillonaria. Dicha industria cuenta a nivel mundial con una comunidad de millones de jugadores que se incrementa cada día.

Desarrollar un videojuego es un proceso que va mucho más allá de un típico desarrollo de software y necesita de diversas habilidades como son: programar, modelar y diseñar (1); dándole una importancia elevada a la creatividad. Además, los videojuegos unifican el arte, la ciencia y la tecnología. Hoy en día debido a la gran cantidad de funciones que brindan los motores de juegos, se les han dado otros usos como son la realización de sistemas informáticos más enfocado en las empresas, el marketing, la salud, la educación y las ciencias militares.

En cuanto a los procesos y técnicas de implementación, el desarrollo de videojuegos, toma elementos de las distintas disciplinas de la computación, como son: la computación gráfica, simulaciones físicas, *gameplay* (lógica del juego y sus reglas), la programación de red y programación del motor (2). Los videojuegos hacen un amplio uso de la Inteligencia Artificial para producir la ilusión de inteligencia en el comportamiento de los personajes que no pueden ser manipulados por el jugador.

En Cuba varios han sido los esfuerzos desarrollados en pos de la creación de videojuegos, aunque con limitaciones tecnológicas pronunciadas y lo complejo que se vuelve realizar videojuegos con calidad y que sean atractivos al público. Una de las instituciones pioneras es la Universidad de las Ciencias Informáticas (UCI), específicamente el Centro de Entornos Interactivos 3D (VERTEX); el cual tiene como meta desarrollar videojuegos de diferentes géneros, aplicables al entretenimiento, la educación, salud, y otras áreas. Entre los videojuegos ya realizados se encuentran: Súper Claria, Meteorix y Chivichana.

La principal estrategia del centro VERTEX es fomentar el mercado de los videojuegos a nivel nacional e internacional. Por esto se trabaja en la creación de un repositorio de prototipos de videojuegos para su futura implementación. Uno de los prototipos a desarrollar es un videojuego

RPG de base por turnos, debido a que el centro VERTEX actualmente carece de un prototipo de este tipo.

Con lo anterior expuesto, se define como **problema** a resolver**:** ¿Cómo desarrollar un prototipo de videojuegos RPG de base por turnos?

El **objeto de estudio** se definió como el proceso de desarrollo de prototipos de videojuegos.

El **campo de acción** son los videojuegos RPG de base por turnos.

El **objetivo general** es desarrollar un prototipo de videojuego RPG de base por turnos, para el repositorio de videojuegos del centro VERTEX.

Para lograr el objetivo trazado se acometieron las siguientes **tareas**:

- Análisis del estado del arte para la realización de un videojuego RPG de base por turnos.
- Captura de requisitos funcionales y no funcionales que debe cumplir el prototipo de videojuego.
- Desarrollo de los artefactos ingenieriles necesarios para el flujo de trabajo de Análisis y Diseño.
- Implementación de la solución.
- Pruebas al prototipo del videojuego.

Entre los **métodos científicos** utilizados se destacan los siguientes:

Métodos teóricos:

El **método histórico-lógico** se utilizó para analizar la evolución a través del tiempo de los videojuegos RPG desde su surgimiento hasta los tiempos actuales.

El **método sistémico** se utilizó para determinar los componentes que forman parte del proceso de desarrollo de prototipos videojuegos RPG de base por turnos y analizar la interacción que existen entre estos.

Métodos empíricos:

La **observación** se empleó para observar distintos videojuegos RPG de base por turnos. Estos sirvieron como objeto de análisis y comparación para establecer las características y elementos fundamentales que deben estar presente en un prototipo de videojuego del mismo género.

Estructura capitular

El presente trabajo de diploma está estructurado en tres capítulos:

Capítulo I: Fundamentación teórica.

En este capítulo se hará referencia al marco teórico y metodológico de la investigación donde se realiza un estudio del estado del arte. Se establecen las pautas a seguir en el desarrollo de un videojuego, haciendo énfasis en las tecnologías, herramientas y metodología a utilizar en el proceso de desarrollo del prototipo.

Capítulo II: Propuesta de solución.

En este capítulo se describe la propuesta de solución para la situación problemática anteriormente planteada y se muestran los artefactos generados al usar la metodología de software seleccionada.

Capítulo III: Implementación y pruebas.

En este capítulo se expone el modelo de dominio del prototipo de videojuego. Además de explicar el funcionamiento y descripción de los *scripts* creados y los componentes reutilizables con los que cuenta el prototipo. Se aplican y muestran los resultados de las distintas pruebas de videojuegos para la búsqueda de errores.

Capítulo I: Fundamentación teórica

1.1 Introducción

En el presente capítulo, se dará una descripción de los conceptos principales relacionados con los prototipos y videojuegos con el objetivo de establecer un lenguaje común, que permita avanzar con el resto del trabajo. Se profundiza en el estudio de los videojuegos RPG de base por turnos. Se establecen las pautas a seguir en el desarrollo y elaboración de un videojuego, haciendo énfasis en las tecnologías, herramientas y metodología a utilizar en el proceso de desarrollo del prototipo.

1.2 ¿Qué es un prototipo?

Según la Real Academia de la Lengua Española, prototipo es: “Ejemplar original o primer molde en que se fabrica una figura u otra cosa” (3).

Fabrice Kordon dice: “Un prototipo es un modelo ejecutable de un sistema que refleja con precisión un subconjunto elegido de sus propiedades, tales como formatos de presentación, resultados calculados, o los tiempos de respuesta. Los prototipos son útiles para la formulación y validación requisitos, la resolución de problemas de diseño técnico y de apoyo de diseño asistido por ordenador de software y componentes de hardware de los sistemas propuestos…” (4).

Se puede agregar que prototipo es una representación de un sistema, aunque no es un sistema completo. Este posee las características del sistema final o parte de ellas.

Características:

Según Walter Maner (5), las características de un buen prototipo son:

- Ejecución: trabaja lo suficientemente bien con la entrada activa que le procura el usuario para permitir un test de usabilidad.
- Maduración: puede evolucionar con el suficiente refinamiento hasta el producto final.
- Representación: tiene el aspecto y las características de actuación del sistema en

planificación.

- Perspectiva: como mínimo simula un 20% de las funciones que los usuarios utilizarán el 80% del tiempo.

Los prototipos suelen catalogarse en dos categorías básicas: prototipos de baja fidelidad y prototipos de alta fidelidad. Los prototipos de baja fidelidad se caracterizan por utilizar materiales distintos al del producto final, son baratos, simples, fáciles de producir y particularmente útiles en las fases iniciales del desarrollo durante el diseño conceptual. Los prototipos de alta fidelidad son aquellos que se parecen al producto final y utiliza sus mismos materiales. Marc Rettig (6) desaconseja el uso de prototipos de alta fidelidad pues:

- Necesitan mucho tiempo para crearse.
- Las pruebas tienden a centrarse en aspectos superficiales.
- Los desarrolladores se resisten a cambiar algo que les ha llevado horas crear.
- Crea excesiva expectación.

Ventajas:

Marion Buchenau y Jane Fulton Suri (7), identificaron tres tipos diferentes de actividades dentro del proceso de diseño y desarrollo de los prototipos que son muy valiosos:

- Entender y comprender las experiencias de los usuarios existentes y el contexto.
- Exploración y evaluación de ideas de diseño.
- Comunicar ideas para una audiencia.

El uso de prototipos trae ventajas sustanciales como son las expuestas por Lucho Salazar (8):

- Permiten el desarrollo de un sistema a partir de requisitos poco claros o cambiantes. Esto ocurre con cierta frecuencia en muchos proyectos de software.
- Como información complementaria a los requisitos constituyen un gran apoyo a las estimaciones de esfuerzo de todas las áreas, incluyendo proveedores.

- Permite a todos los involucrados entender bien y mejor el problema antes de la implementación final.

1.3 ¿Qué es un videojuego?

Los videojuegos nacen de los juegos que, antes de la aparición de los computadores, eran utilizados para fines lúdicos. Por ello, antes de definir a los videojuegos es necesario tener claro el concepto de juego: “Un juego es una actividad interactiva voluntaria, donde uno o más jugadores siguen determinadas reglas que definen su comportamiento, promulgando un conflicto artificial cuyo desenlace es cuantificable” (9).

Wolfgang Kramer propone que los juegos se componen de dos elementos: componentes y reglas. Define a los componentes como el hardware y a las reglas como el software (…), y plantea una serie de criterios para diferenciar a los juegos con reglas de aquellos que no los tienen. Sus criterios son: todo juego debe ofrecer experiencia de juego en grupo o en solitario, igualdad de posibilidades de ganar, libertad de tomar decisiones, participación activa y finalmente, la capacidad de sumergirse en el mundo del juego. Mientras que los juegos con reglas tienen algunas características adicionales: reglas de juego, objetivo, curso del juego variable (sujeto al azar) y la competencia entre jugadores (10).

Mike Zyda abordó en el año 2005 un concepto en el cual también tiene en cuenta las reglas como elemento esencial de los juegos y fue más allá al definir los videojuegos como: “Una prueba mental, llevada a cabo frente a una computadora de acuerdo con ciertas reglas, cuyo fin es la diversión o esparcimiento, o ganar una apuesta” (11).

Entonces una definición sencilla para videojuego fue la dada por Nicolas Esposito: “Un videojuego es un juego que con el que interactuamos, gracias a un aparato audiovisual y que podría estar basado en una historia” (9).

Finalmente el autor considera que la definición dada por Tavinor sintetiza estas definiciones para construir, lo que él llama, la definición disyuntiva de los videojuegos: “X es un videojuego si y sólo

si (a) se trata de un artefacto en un medio visual digital, (b) está concebido principalmente como un objeto de entretenimiento, y (c) está destinado a proporcionar dicho entretenimiento a través del empleo de una o ambas de las siguientes modalidades de participación: sujeta a reglas de juego o ficción interactiva" (12).

1.4 Clasificación de los videojuegos

Lograr una clasificación es útil para los diferentes aspectos de los videojuegos: en estudios teóricos para facilitar la investigación, la búsqueda de ejemplos y referencias. En materia de *game design* (diseño y planificación de las mecánicas y el contenido del videojuego) para encontrar técnicas comunes de diseño entre juegos de un mismo tipo; en el desarrollo de herramientas para encontrar los problemas comunes de implementación y desarrollar herramientas más precisas que faciliten el desarrollo de los juegos y reduzcan los errores y tiempos de producción. Desde un punto de vista comercial, para ayudar a los consumidores a encontrar los juegos que coinciden con sus gustos personales; en cambio a las empresas distribuidoras y desarrolladoras para aprender de los gustos de sus consumidores.

Sin embargo, no existe una taxonomía universal aceptada, y se hace muy difícil agrupar los juegos atendiendo a una clasificación específica. Esto se debe a que existen muchos puntos de encuentros entre ellos y además, influye la constante y rápida evolución de la industria y de los videojuegos en sí.

A continuación, se exponen cómo se clasifican los videojuegos según Marqués Pere (13):

- Juegos de lucha.
- Juegos de combate.
- Juegos de disparo.
- Plataforma.
- Simuladores.
- Juegos de deporte.

- Estrategia.
- Aventura.
- Los juegos de rol (RPG).
- Juegos de guerra.
- Simuladores de sistema.
- Juegos de sociedad.
- Ludo-educativos.

1.5 Videojuegos RPG

Las siglas RPG hacen referencia al término "*Role Playing Game",* que describe los títulos donde el jugador asume el papel de un personaje virtual para embarcarse en una aventura épica. Para entender que son los videojuegos RPG, primero se debe definir y caracterizar qué es un juego de rol tradicional, ya que son la fuente de inspiración y base de los RPG.

Marc Brell en su trabajo titulado "Juegos de rol" expone que un juego de rol en un principio es un juego, con todo lo que esto conlleva: libertad, diversión, entretenimiento, no realidad, reglas propias. Por otro lado, es de rol: modelo de comportamiento social. Es por esto que los juegos de rol se definen como aquellas actividades de ocio en que los participantes interpretan modelos de comportamiento que no son los suyos (14).

Los juegos de rol en la práctica son juegos de mesa que en la mayoría de los casos involucra un tablero, unos dados y un libro de reglas para que el personaje comience un viaje imaginario hasta lograr un fin dado.

Por otra parte, John H. Kim dice que los RPG es un género de videojuegos que usa elementos del juego de rol tradicional llevando al usuario a un mundo virtual donde a través de un personaje protagonista interpreta un papel con el cual ha de mejorar sus habilidades, ganar experiencia, ítems y otras características a medida que interactúa con el entorno y otros personajes a lo largo de una historia. Esta historia sigue un curso de acuerdo a sus decisiones y acciones (15).

En dicho género los usuarios asumen el “rol” de un personaje, como su nombre lo indica. Estos personajes son imaginarios al igual que sus historias o tramas en las cuales el usuario interpreta y narra sus diálogos y acciones. La inmersión e interacción en este tipo de experiencias depende en su mayoría de la imaginación de los jugadores y de su compromiso con el juego, a través del arte, historia y contexto en el que se juega. Es común encontrar escenarios donde los jugadores se disfrazan y decoran escenarios para aumentar la sensación de inmersión, así mismo algunos adoptan posturas, frases y acciones que reflejan los personajes que están controlando.

Características

Matías Daniel Espinoza trató algunos de los elementos fuertemente asociados a los juegos de RPG, que son: el desarrollo estadístico de personajes, sofisticado inventario de poderes humanos y sobrenaturales, ambiente fantástico, recursos monetarios y objetos diversos. A continuación, se introducen algunos conceptos que identifican los elementos gráficos más relevantes que podemos encontrar en un videojuego RPG (16):

- Protagonista: Corresponde al personaje controlado por el jugador (este elemento es llamado también como Personaje Jugador).
- Personaje: Corresponde a los personajes que no están bajo el control directo del jugador en un juego de rol; su comportamiento (que se compone generalmente por acciones específicas y parlamentos) se encuentra por lo general prescrito y es automático, y actúa a partir de las acciones del Protagonista (este elemento es llamado también como Personaje no Jugador).
- Ítem: Corresponden a elementos que pueden ser coleccionados y/o ser utilizados para lograr algún fin durante el juego.
- Decorativo: Corresponde a elementos gráficos que tienen la función de “decorar” el mundo virtual mejorando la percepción del entorno de la historia. También sirven para restringir los movimientos del jugador principal.
- Escenario: Los distintos escenarios conforman el mundo virtual completo donde el protagonista puede desplazarse. Sobre la superficie de cada escenario encontramos

personajes, ítems, decorativos y al protagonista (solo si este último se está desplazando sobre el escenario).

Un videojuego llega a su fin en 10 horas promedio, sin embargo, los RPG cuentan historias que fácilmente alcanzan las 40 horas de duración, sin considerar las aventuras opcionales, que pueden llegar a triplicar la estadía en el mundo virtual. Por lo que es característico que sean juegos muy extensos (17) (18).

1.5.1 RPG de base por turnos (*Turned Based RPG*)

Limpiar el honor de alguien que se consideraba ofendido por algún motivo fue el trasfondo que originó el duelo como práctica en la nobleza entre los siglos XV y XX. Una práctica que podría definirse como un combate reglado entre dos contendientes, con armas mortales, y con el objetivo de que el ofendido pudiese restaurar su honor. Las normas que regían las condiciones de cada duelo podían variar en función de lo que propusiese la parte provocada, como por ejemplo el tipo de arma o si el duelo sólo podía finalizar con la muerte de uno de los dos contendientes. En cualquier caso, una vez aceptadas las reglas del "juego", se aceptarían y respetarían bajo el honor de ambos caballeros. Esto es muy parecido a lo que ocurre en los videojuegos RPG de base por turnos.

En el sistema de batalla de los RPG de base por turnos, el escenario es cerrado; generado como paréntesis a todo lo acontecido hasta el momento. Efectivamente habrá en algunos casos maniobras para la huida, pero sería el equivalente a recoger las figuras del tablero de ajedrez. Es decir, el desarrollo de un combate por turnos bebe de los juegos de tablero, siendo el ajedrez la piedra filosofal de la estrategia**.** En algunos casos de videojuegos RPG de base por turnos se puede encontrar un escenario con una rejilla para la gestión del movimiento de los personajes. Si bien es cierto, por otra parte, que los primeros RPG por turnos no ofrecían la posibilidad de desplazarse dentro del escenario de la contienda. Por tanto, la acción pasa a ser puramente estratégica.

Analizar la información de pantalla como el orden en los turnos de acción, los puntos fuertes de tus personajes, así como los puntos débiles de tus oponentes, se torna imprescindible para interpretar

y actuar en consecuencia. Se trata por tanto de un proceso más próximo a la capacidad de análisis y conocimiento del propio sistema de turnos, que de la habilidad intuitiva en la pulsación de botones.

En los RPG de base por turnos el usuario ordena la acción básica y el personaje en cuestión la ejecutará desarrollando una serie de acciones de forma automática, como desenvainar su espada, desplazarse hasta donde está el enemigo, golpear y volver al punto de partida. Toda una simbología mientras se realizan los cálculos que determinarán el resultado de la acción. Siendo así, la cronología del combate se entenderá como una línea discontinua donde cada segmento representa un turno realizado. Una fórmula que ofrecerá al jugador cierto margen de maniobra dando paso a una acción más meditada y estratégica (19).

1.5.2 Evolución de los videojuegos RPG

El género RPG comenzó a mediados de los años 70, inspirado por juegos de rol de mesa. Gary Gyga y Dave Arneson (20) crearon el primer juego de rol del mundo: *Dungeons & Dragons* (1974), revolucionando los juegos de mesas y fascinando a millones de personas.

Figura 1: Juego de mesa Dungeons & Dragons.

En *Dungeons & Dragons* los participantes escogían a su personaje, y lanzando un dado y siguiendo las reglas de un libro, comenzaban un viaje imaginario a través de unas mazmorras habitada por un dragón. Posteriormente, un estudiante universitario creó el juego *Dungeon* para las primeras computadoras.

En la época de los 80s, los RPG ya estaban instaurados como forma de entretenimiento y se extendieron por medio mundo. El descubrimiento de los RPG en Japón fue una auténtica revolución, tanto fue así que comenzaron a crear sus propios videojuegos de rol. Este es el caso de *Ultima III: Exodus (1983)* (21), el que ha influido de manera notoria en el desarrollo de los actuales videojuegos de RPG, ya que es el primer juego que incluyó personajes animados, la posibilidad de controlar a un grupo de personajes, con una pantalla aparte de combates, y creando un sistema de combate más elaborado, con armas y magias. Esta identidad adquirida, es la que continuamente es llamada *JRPG (Japanese Role-Playing Game)* (20).

En 1985 irrumpe el videojuego *Phantasia*. Este introduce el concepto de *automapping*, y provee al jugador de pistas e información de trasfondo.

En 1986 aparece *Dragon Quest*, considerado la base de los juegos de rol en consola. Otro de los juegos más importantes es *Final Fantasy*, que introdujo la vista lateral y fue considerado norma en los juegos.

Figura 2: Final Fantasy I.

Uno de los juegos RPG más importantes de consola es *Zelda II: The Adventure of Link* (1987) (22) el cual es uno de los primeros juegos ARPG de la historia, a diferencia del original *The Legend of Zelda* que a pesar de ser un juego de RPG es más cercano a un juego de aventuras-acción.

En 1988 apareció el primero juego de la saga *Gold Box*, llamado *Pool of Radiance*, el cual está basado en las reglas de *Advance Dungeons & Dragons*, ofreciendo una vista en primera persona,

y combates tácticos en tercera persona. Con la saga *Gold Box* se popularizaron los combates por turnos (23).

Figura 3: Pool of Radiance.

A principio de los 90 los RPG eran más cercanos a la novela de fantasía. Hacia el final de la década se fueron convirtiendo en videojuegos mucho más cinemático, al tener largas escenas dramatizadas entre los personajes, sobre todo los JRPG. Comenzando así la edad dorada de este género con juegos clásicos como: *The Bards Tale, Wastedland*, La saga *Might and Magic* y la continuación de la saga *Ultima* (23).

El incesante desarrollo de títulos RPG y la lógica evolución tecnológica produjo la llegada de un nuevo subgénero que hasta ahora se ha mantenido como uno de los más jugados y rentables para las compañías: el MMORPG. El primer título destacado fue *Neverwinter Nights*, un juego que salió a la venta en 1991 (24).

Figura 4: Neverwinter Nights.

El nuevo milenio trajo consigo títulos que supieron mantener el listón de antaño, con secuelas de *Baldur's Gate, Final Fantasy* o Diablo. La revolución, que ya forma parte de la cultura popular como en los años ochenta lo fue *Dungeons & Dragons*, llegó con *World of Warcraft*. Un éxito en parte gracias a una evolución no solo ligada al apartado gráfico, sino a una libertad dentro de un mismo marco argumental.

1.5.3 *Dragon Quest*

Dragon Quest VI: Los Reinos Oníricos fue lanzado el 9 de diciembre de 1997 y es un videojuego de RPG de base por turnos (25), que se caracteriza por extensos mapas para explorar, donde existen mundos llenos de pueblos, mazmorras y castillos. Aunque los Reinos Oníricos no es el *Dragon Quest* más complicado de todos, es un juego exigente con el usuario por el nivel de dificultad. Los combates son aleatorios, las batallas aparecen muy frecuentemente mientras se avanza por mazmorras u otros lugares del mapa. El sistema de batalla usado es de base por turnos puro, en el cual se puede determinar en la mayoría de los casos cómo actúan los miembros del grupo de combate.

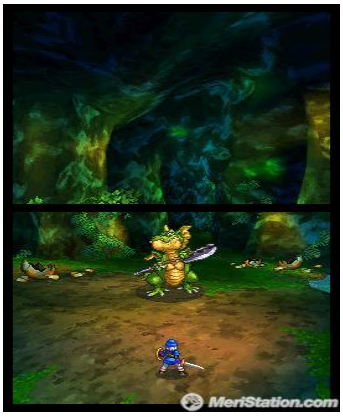


Figura 5: Dragon Quest VI.

Uno de los elementos más destacados de *Dragon Quest VI* se encuentra en las vocaciones de los protagonistas (26). Este está compuesto por un sistema de clases en el que el Héroe y los miembros de su equipo adquieren nuevas habilidades y conjuros. En algunos momentos del juego, se pueden cambiar las clases de los personajes, potenciando así todo tipo de atributos y dejando

a un lado otros. Esto permite crear personajes muy completos e impulsa la dinámica del juego, permitiendo al usuario adaptar a los protagonistas a los cambios que va sufriendo la historia, y a las necesidades y retos que van surgiendo.

1.5.4 *Final Fantasy*

Final Fantasy XIII es uno de los videojuegos de la franquicia del mismo nombre. Fue lanzado el 17 de diciembre de 2009 para la consola PS3 de Sony (27), y posteriormente para otras plataformas. *Final Fantasy XIII* se diferencia de sus antecesores e intenta ir un poco más allá en el sistema de batalla basado en turnos. El sistema está regido por una barra formada por segmentos, llamada Barra ATB (*Active Time Battle* - Batalla en Tiempo Activo). Esta nos permite ejecutar acciones. Una vez hayan sido realizadas las acciones, se debe esperar a que los segmentos se vuelvan a llenar para que el personaje dé comienzo a su siguiente cadena de acciones. Cada habilidad que se realice puede consumir uno o varios segmentos. Por ejemplo: "Atacar" consume un segmento, mientras que la habilidad "Ataque circular", consume dos. Así que, si se tienen cuatro segmentos, se puede realizar "Atacar" dos veces y "Ataque circular" una. Es un sistema peculiar que ha gustado mucho entre los fans de la franquicia.

En resumen, el sistema consiste en una serie de acciones encadenadas que dependen de una cuenta atrás que, tras terminarse, hace que se desencadenen. El usuario elige las acciones, esperando que se llene la barra ATB, dichas acciones se realizan y la barra vuelve a llenarse. El sistema es completamente activo, así que mientras se eligen las acciones, la barra se está llenando, no existe pausa alguna (28).

Figura 6: Final Fantasy XIII.

El menú de combate es la base de operaciones y con el cual el usuario interactúa con el sistema. Este permite acceder a la lista de habilidades aprendidas. Dicho menú está formado por cuatro secciones:

- Comando automático: Esta opción selecciona automáticamente el mejor conjunto de ataques para ese momento en concreto.
- Habilidades: Permite acceder a la lista de habilidades aprendidas.
- Habilidades PT: Este tipo de habilidades son especiales, su uso está limitado al líder (en *Final Fantasy XIII* solo se puede manipular al jugador principal, a diferencia de otras versiones de la saga donde se manejan todos los integrantes del grupo de combate), así que el resto de miembros no podrán ejecutarla las habilidades PT. Junto a estas aparece una cifra, que indica el número de puntos tácticos que son necesarios para utilizarlas. Los puntos tácticos se consiguen en los combates. Cuantas más estrellas se consigan en un combate, más probable será obtener un punto táctico, el cual se almacenará hasta que decidamos emplearlo en el uso de una habilidad PT. Se pueden almacenar hasta un máximo de cinco puntos tácticos al mismo tiempo, así que una vez que se llega a esta cifra, no se ganarán más.
- Objetos: Se refiere a los objetos consumibles que se obtienen a través del juego y que se almacenan. Estos no precisan de la barra ATB, así que no necesitarán ningún tiempo de carga para ser utilizados.

1.5.5 ¿Qué es ineludible en un buen RPG de base por turnos?

James Liu hizo una excelente recopilación de qué características y elementos debería tener un buen juego de RPG de base por turnos (29):

- Drama entre los personajes.
- Historias originales que envuelvan a los personajes.
- Objetos consumibles (Ítems).
- Personajes amigos que se unan en la travesía.
- Armas secretas y complejas de adquirir.
- Jefes difíciles de derrotar que no sean obligatorios.
- Modificación de las estadísticas del personaje.
- La sensación de que el personaje se hace cada vez más fuerte (*Nivelación-up*).
- Historias entrelazadas. Siempre es impresionante cuando un RPG puede contar una historia desde dos perspectivas.
- Múltiples finales. Historias no lineales.
- Las decisiones que toma el usuario afectan la historia del juego.
- Misiones secundarias.
- Desenlace lento de los acontecimientos. El descubrimiento de conspiraciones.
- ¡Salvar el mundo! Dar un sentido al juego.
- Clases de trabajos configurables por el usuario.

También James Liu propuso algo interesante a la hora de crear un juego RPG de base por turnos: eliminar los encuentros aleatorios con los enemigos. “Los encuentros al azar son tedioso, incluso en las consolas” (29). Su variante es que los jugadores puedan elegir los enemigos a los que se enfrentan mediante un botón "Aceptar" o "Cancelar", pero siempre manteniendo algunos enemigos ineludibles que no permitirían que el usuario cancele.

Es importante tener claro para qué plataforma se va a desarrollar el videojuego RPG de base por turnos. Los que sean creados para móviles deben concretarse más en pequeñas animaciones visuales, y hacer menos énfasis en el audio y efectos de sonido (29). Esto se debe a que muchos usuarios que juegan en móviles no tienen el sonido activado cuando están en un viaje público o al jugar en las noches antes de dormir. En cambio, cuando se juega en consolas o computadoras, el sonido es de suma relevancia.

1.5.6 Características del prototipo de RPG de base por turnos

Al realizar el estudio del arte sobre los juegos RPG de base por turnos y los casos especiales de *Dragon Quest* VI y *Final Fantasy XIII* que son videojuegos excelencia de dicho género, se definieron las características esenciales que deben estar presentes en el prototipo a desarrollar:

- Modificación manual de las estadísticas y clases del personaje. En dependencia de qué atributos decida aumentar el usuario con los puntos de experiencia obtenidos en los combates, el personaje se especializará en una de las siguientes clases: mago, guerrero o pícaro.
- La sensación de que el personaje se hace cada vez más fuerte (*Nivelación-up*).
- Sistema de batalla por turnos activo. La barra de tiempo se incrementa mientras se elige la acción a realizar.
- El enemigo puede atacar más lento o más rápido que los personajes en dependencia de sus atributos.
- Enemigos visibles, dando libertad al usuario de elegir a cuál enfrentarse.
- Enemigos activos, que al entrar en su área de vigilancia se vuelven peligrosos (aumenta la posibilidad de combate).
- Menú de combate.
- Menú de habilidades. Habilidades que el jugador principal puede aprender.

1.5.7 Funciones matemáticas en los videojuegos RPG

La mayoría de los valores estadísticos de un personaje en un videojuego RPG pueden utilizar distintas funciones que definen su comportamiento durante el videojuego. El uso de funciones incide en el inicio del videojuego para que este sea atrayente y dinámico para el jugador, con grado de complejidad aceptable (30). Igualmente, las funciones logran que el progreso del jugador sea cada vez un reto mayor al aprovechar las curvas y crecimientos que presentan estas. A continuación, se detallan algunas funciones que pueden ser usadas (31):

- **Funciones logarítmicas:** Tienen la forma **f(x) =** $\log_a x$ donde la base "**a**" es un número real y positivo pero distinto de 1. Si el valor de "**x**" es igual a 1 el resultado de la función es 0. En la función logarítmica (cuando **a** > 1) cuanto mayor es la base del logaritmo, más cerca del eje **X** está (números más pequeños). El dominio de la función son los reales positivos puesto que no existe el logaritmo de un número negativo. Estas funciones son utilizadas para obtener valores que crezcan rápidamente al principio con cada incremento de "**x**".

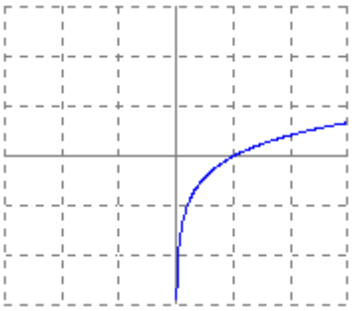


Figura 7: Función logarítmica.

- **Funciones exponenciales:** Tienen la forma **y=a^b**, siendo "**a**" un número real positivo y "**b**" un número real. La ecuación **f(x) = e^b** define a la función exponencial de base "**e**". Donde "**e**" es un número irracional igual a **2.71828...** Estas funciones crecen lentamente al principio y después más rápido. Se utilizan para obtener números que en un principio aumentan lentamente con cada nuevo valor que adquiere la "**b**".

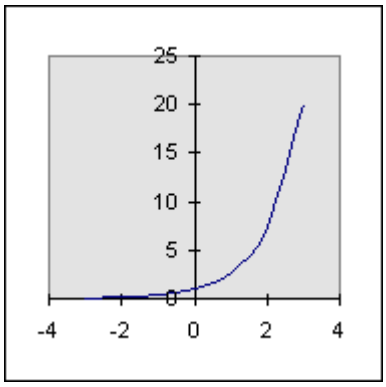


Figura 8: Función exponencial.

- **Funciones lineales:** Tienen la forma **f(x) = mx +b**, donde "**m"** y "**b"** son constantes reales y "**x"** es una variable real. La constante **m** es la pendiente de la recta, y "**b"** es el punto de corte de la recta con el eje **Y**. La pendiente "**m**" es la inclinación de la recta con respecto al eje de abscisas. Si se modifica "**b"**, entonces la recta se desplazará hacia arriba o hacia abajo. La representación de la función en el plano cartesiano es una línea recta. Son utilizadas para generar números que crecen a un ritmo constante.

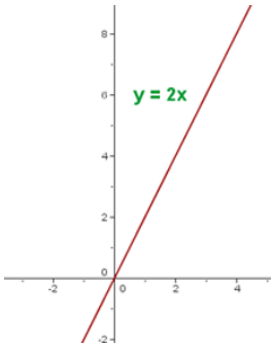


Figura 9: Función lineal.

1.6 Metodología de desarrollo de software

Clinton Keith en su libro "*Agile game development with Scrum*" (32) analiza como en los comienzos de los videojuegos el uso de metodologías de desarrollo para proyectos de una sola persona (o de un equipo muy chico) no era ni aplicable ni necesario. Sin embargo, con el crecimiento de las capacidades del hardware, el crecimiento de los equipos de desarrollo y el aumento de la

complejidad de los proyectos, la falta de metodologías se tradujo en retrasos, pérdidas, y productos de calidad inferior a las planificadas. La necesidad de administrar el desarrollo de los proyectos de maneras no arbitrarias se hizo evidente y obligó a que la industria comience a utilizar metodologías de desarrollo más formales (33).

Lo que se conoce como "El triángulo del proyecto" (*ver figura 10*) establece que para todo proyecto de software solamente dos de los tres objetivos presentes en el triángulo pueden ser cumplidos al mismo tiempo en un proyecto. Los objetivos comprendidos en el triángulo son: respetar el presupuesto asignado, terminar a tiempo, y conseguir un producto de alta calidad (34).

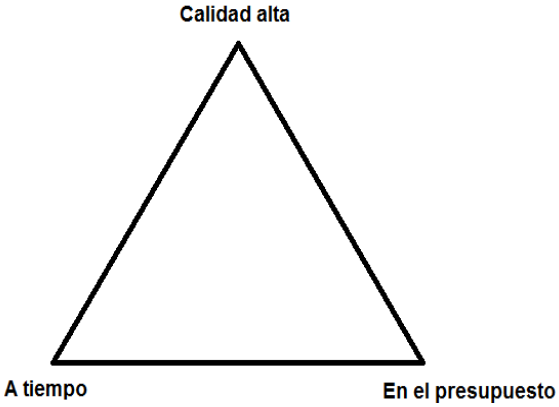


Figura 10: "El triángulo del proyecto".

Juegos como *The Sims* (Maxis) y *Ultima Online*, si bien no cumplieron con los objetivos de llegar a tiempo y cumplir con el presupuesto asignado, lograron ser juegos de una buena calidad y lograron alcanzar un nivel de críticas y de ventas exitosas. Otros como *Duke Nukem Forever* con catorce años de desarrollo (acompañado de muchos problemas, traspasos de tecnologías y diversas empresas desarrolladoras) y *StartCraft II* (Blizzard) con ocho años de desarrollo se alejaron completamente del objetivo de terminar a tiempo. Ambos juegos cuentan con una calidad totalmente dispar y la expectativa generada por los años de espera de los jugadores subió el nivel de calidad esperado. *Duke Nuken Forever* con una calidad cuestionable según la crítica logró una aceptación muy pobre y un nivel de ventas bajo. Por el contrario, el juego de *Blizzard* fue un éxito tanto en ventas como en críticas convirtiéndose rápidamente en un nuevo clásico (33) (35) (36).

Erik Bethke en su libro "*Game Development and Production"* dice que las metodologías de desarrollo son una de las herramientas con las cuales cuentan los desarrolladores de videojuegos para combatir este problema (33).

A partir de estudios realizados a diferentes videojuegos se pudo verificar que en la mayoría de los casos utilizaban metodologías de desarrollo basadas en la metodología ágil Scrum, por lo que se decide utilizarla para guiar el desarrollo del prototipo de videojuego RPG de base por turnos.

1.6.1 Metodología Ágil Scrum

Los procesos y metodologías agiles se basan en el manifiesto ágil (37) que plantea:

- Individuos y sus interacciones frente a procesos y herramientas.
- Software en funcionamiento frente a documentación exhaustiva.
- Colaboración del cliente frente a una negociación de contrato.
- Respuesta al cambio frente a seguir un plan.

El núcleo de las metodologías ágiles es: "La satisfacción del cliente a través de la temprana y continua entrega de componentes de software útiles" (38). El desarrollo ágil es útil en proyectos donde los requerimientos no están bien comprendidos desde el comienzo y es necesaria la creación de nuevas tecnologías para ayudar a su descubrimiento. Esto requiere, por parte del cliente que se encuentre involucrado profundamente en el proceso de desarrollo. Estas características definen perfectamente al desarrollo de la mayoría de los juegos de la actualidad.

Scrum es una metodología ágil para administrar y controlar el desarrollo de software de un producto en forma iterativa e incremental. Una de sus características es que no indica prácticas específicas a seguir durante el desarrollo (39). Esto brinda flexibilidad y permite ajustar el proceso a la realidad y forma de trabajo de cada proyecto, así como a los diferentes requerimientos de los clientes.

En la metodología ágil Scrum, cada equipo recibe tareas del denominado "Scrum Master". Este tiene la tarea de gestionar varios de los equipos pequeños, asignando proyectos y supervisando

los progresos. A cada miembro del equipo se le asigna un aspecto diferente del proyecto y debe guiarlo hasta su terminación. Una vez que se establecen los deberes, el equipo comienza un *sprint*, el cual consiste en un proceso donde todo el equipo trabaja al unísono sobre un aspecto específico del proyecto coordinando entre sí sus tareas. Al final, cuando el ítem de desarrollo está listo, se integra dentro del juego y es presentado para el hito establecido (40).

Según la descripción que realiza Ken Schwaber (41), Scrum se estructura en tres fases denominadas *pre-game*, *game* y *post-game* como se muestra en la Figura 11. Durante el *pre-game* se define el producto basado en las características conocidas, estimando su tiempo y costo. También se analiza el sistema a construir, se define la arquitectura y se realiza un diseño de alto nivel de la solución. La fase de *game* consta de iteraciones, llamadas *sprints* que duran de dos a seis semanas y donde se desarrollan las características del producto. Al comienzo de cada una se realiza su planificación, donde se describen, priorizan y estiman las características que se van a desarrollar, y al concluir se evalúa su resultado. El *post-game* es el cierre del proyecto, donde se prepara la liberación del producto, se verifican las versiones a entregar y se realiza la documentación final.

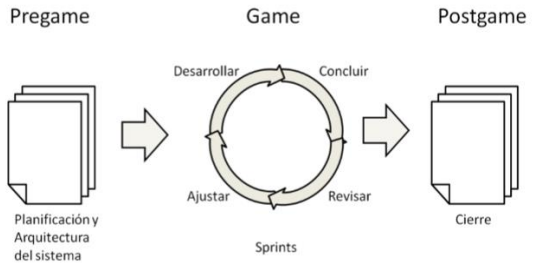


Figura 11: Las fases de Scrum.

1.7 Herramientas utilizadas

A continuación, se dará una breve explicación de cada uno de las herramientas utilizadas durante la realización del prototipo RPG de base por turnos.

Adobe Photoshop CS6:

El software Adobe Photoshop CS6 es una herramienta de edición de imágenes con nuevas opciones creativas respecto a sus versiones anteriores. Posee el motor *Adobe Mercury Graphics Engine* para lograr un rendimiento increíblemente rápido. Retoca con las nuevas funciones basadas en el contenido y crea diseños y películas fascinantes mediante los nuevos flujos de trabajo y herramientas rediseñadas.

Diseño 3D. Autodesk 3ds Max:

Autodesk 3ds Max proporcionan potentes herramientas integradas de modelado, animación y dibujado en 3D que permiten a los artistas y los diseñadores dedicar más esfuerzo a la creatividad en lugar de a las dificultades técnicas. Ofrece herramientas especializadas a los desarrolladores de juegos, creadores de efectos visuales, diseñadores de gráficos de movimiento y otros profesionales de la creatividad que trabajan en el diseño de medios.

Motor de videojuegos:

Existe actualmente una amplia gama de motores de videojuegos, con diferentes tipos de licencias y orientados a cumplir distintos tipos de propósitos. Se puede encontrar motores comerciales y gratuitos.

Unity 3D es un motor perteneciente a *Unity Technologies*. Es un software que pretende facilitar el desarrollo de los videojuegos, aportando una tecnología para la creación de características y funcionalidades que a la vez son reutilizables. Unity 3D incluye muchos elementos ya creados que son de gran ayuda, por ejemplo, funciones pre establecidas como *OnGUI* para crear elementos gráficos. También el sistema de animación de Unity 3D permite tratar a los objetos tanto genéricos como humanoides, lo que implica poder reutilizar las animaciones y los "*animator controller*" en diferentes modelos 3D.

Unity 3D es la herramienta seleccionada por el centro VERTEX para el desarrollo de videojuegos. Unity 3D soporta lenguajes como Java, C# y Boo. Actualmente es uno de los motores más utilizados en estudios pequeños debido a la facilidad de aprenderlo, la cantidad de elementos

disponibles para descargar en *Asset Store* que agilizan el desarrollo, así como reutilizar los de otros proyectos.

Conclusiones parciales

El estudio del estado del arte permitió establecer una línea evolutiva del género RPG en los videojuegos. Este análisis arrojó características fundamentales que deben estar presentes en el prototipo a desarrollar. De igual manera se precisa que las funciones logarítmicas, exponenciales y lineales permiten modelar el comportamiento de los distintos valores estadísticos en un videojuego RPG. Además, la metodología Scrum es ágil y proporciona ventajas a la hora del desarrollo. También esta metodología no desmotiva, y hace factible y visible la evolución de un producto, ya que está orientada a las personas más que a los procesos, y propone dividir el desarrollo en pequeñas etapas a superar.

Capítulo II: Propuesta de solución

2.1 Introducción

El presente capítulo contiene las especificaciones necesarias para comenzar el proyecto, estas van desde los elementos formales del prototipo de videojuego a desarrollar hasta el documento de diseño del mismo. También se muestran los artefactos generados al usar la metodología Scrum.

2.2 Fase *pre-game*

Durante la fase *pre-game* se define el producto, basado en las características conocidas del prototipo de videojuego del presente trabajo.

En *pre-game* el videojuego debe ser descubierto, es decir, lo que realmente son los objetivos del videojuego. Se escoge el lenguaje de programación, plataforma y otras herramientas necesarias para el desarrollo. *Pre-game* permite al equipo centrarse en la ejecución del videojuego, en lugar de experimentar con él. Define el sentido que tendrá la fase g*ame*, sin ahogar la creatividad que pueda surgir allí. Durante esta fase el trabajo será encontrar el concepto del juego ideal y diseño (42).

También el desarrollo de un prototipo ayudará a visualizar el factor diversión que el videojuego o una parte de él puede ofrecer. El prototipo debe proporcionar una navegación básica y simplificada para el usuario y las funciones necesarias para poner a prueba.

Los pasos de la planificación de esta fase son:

- Definir los **elementos formales** del videojuego.
- **Documento de diseño** del videojuego.
- Desarrollo de un ***backlog*** completo.

2.2.1 Elementos Formales

“Formal” es en el sentido matemático y científico: algo que puede ser explícitamente definido. Los elementos formales de un videojuego son "átomos", en el sentido que son los elementos más pequeños de un juego que pueden ser aislados y estudiados individualmente (43). A continuación, se analizarán y expondrán los elementos formales del prototipo de videojuego del presente trabajo.

Jugadores:

Los **jugadores** es el elemento formal que implica la cantidad de personas que pueden intervenir y afectar el desarrollo del videojuego.

El prototipo de videojuego del presente trabajo está diseñado para un único jugador.

Objetivos:

El **objetivo** de un videojuego siempre debe ser realizable. Este responde a la pregunta clásica: ¿Qué intentan hacer los jugadores, y para qué lo hacen? Una vez que se conoce el **objetivo**, muchos de los otros elementos formales se revelan de manera sencilla (44), pues sin este se complejiza desarrollar cualquier idea.

El **objetivo** del videojuego a desarrollar es fortalecer al jugador principal y que este aprenda habilidades de combate.

Procedimiento del juego:

El **procedimiento** de un videojuego es lo que el jugador debe realizar para lograr el objetivo antes descrito. Es fundamental a la hora de establecer las características básicas del videojuego, así como su jugabilidad.

En el prototipo del presente trabajo, el jugador debe enfrentarse a los enemigos; los cuales son visibles en todo momento, siempre y cuando el ángulo de visión del jugador lo permita. Si el jugador entra en el radio de acción de algún enemigo, comienza el combate por turnos. Derrotar enemigos

proporciona experiencia que se acumulará hasta que el jugador alcance el siguiente nivel. Alcanzar un nuevo nivel será cada vez más complejo, ya que la experiencia necesaria para lograrlo se incrementa. Además, subir de nivel proporciona puntos especiales con los cuales se aumentan los atributos: inteligencia, destreza, fuerza y constitución.

Recursos:

Recursos es una categoría amplia, y se usa para denotar todo lo que esté bajo control del jugador. Son los elementos usados por el jugador con el fin de cumplir los objetivos. Su utilidad y escasez debe ser balanceada (45).

Los atributos son **recursos** presentes en el prototipo de videojuego de base por turnos, estos son utilizados para medir y calcular los distintos valores estadísticos de los actores (entiéndase por actores al personaje principal y los enemigos) como: el poder de ataque, defensa, velocidad, entre otros.

A continuación, se detallan dichos atributos y a cuáles valores estadísticos de los actores afectan:

- **Inteligencia**: Permite al actor correspondiente poseer una mayor cantidad de puntos de magia o “mana” (a continuación, se les llamará “mana” a los puntos de magia), así como aumentar la defensa contra los ataques mágicos. En el caso del jugador principal, el “mana” es usado para utilizar las distintas habilidades que ha obtenido durante el combate. Si el valor del “mana” llega a cero, no podrán ser ejecutadas más habilidades. La inteligencia está asociada con la tendencia de Mago o Hechicero, esto permite aumentar el daño de las habilidades mágicas.
- **Constitución**: Este atributo está asociado con la constitución física del personaje y su vitalidad. Permite aumentar la vida del actor correspondiente. Mientras más alto se tenga este atributo, más vida poseerá el actor al que se le aumente. El actor pierde vida cuando recibe daño de los ataques que realiza su oponente durante el combate.
- **Destreza**: Permite aumentar la velocidad de ataque del actor correspondiente y la posibilidad de que el ataque sea más efectivo (mayor daño). El aumento de la velocidad es

constante. Este atributo está asociado con la tendencia de Pícaro o Cazador, ambos hacen uso de la destreza para obtener ventaja en el combate.

- **Fuerza**: Este atributo está asociado con la fuerza física del personaje y con la tendencia de Guerrero. Al obtener mayor fuerza el personaje puede ocasionar golpes que incrementen el daño. También, permite aumentar la defensa del actor correspondiente contra los ataques físicos (ataques que no consumen “mana”).

El prototipo presenta una modalidad para puntuar libremente los atributos antes expuestos de la manera que estime conveniente. La puntuación se hará basándose en diferentes estrategias o tácticas que el jugador elija para enfrentar sus enemigos. Estas tácticas pueden ser: mayor cantidad de vida para poder recibir más daño, mayor cantidad de “mana” para lanzar habilidades, más defensa para reducir el daño del oponente, o fortalecer el ataque para vencer al enemigo de manera rápida.

Estos atributos son directamente proporcionales a las habilidades correspondientes. Si la habilidad que lanza el jugador principal es mágica, y en el jugador predomina la inteligencia, entonces el daño de la habilidad va aumentar. Así mismo ocurre cuando lance alguna habilidad de fuerza o destreza, el daño de estas aumenta respectivamente.

Como se había mencionado, cada atributo afecta valores estadísticos como son: “mana”, vida, velocidad o defensa. Esto ocurre mediante el uso de funciones matemáticas para cada caso, las cuales se detallan a continuación:

Función utilizada para incrementar los valores estadísticos asociados a los atributos constitución e inteligencia.

La función **y= a*Log(x)+c** es utilizada para calcular de forma acumulativa el “mana” del actor cuando el atributo inteligencia aumente un valor; de igual forma también se usa para calcular de forma acumulativa la vida del actor cuando el atributo constitución aumente un valor. En dicha función “**y**” es el “mana” o vida a aumentar según sea el caso, “**x**” el valor del atributo

correspondiente, "**c**" una constante que traslada la función en el eje **Y**, si se desea que la función comience a generar resultados mayores que el valor definido en "**c**", y "**a**" es otra constante que aumenta la diferencia entre cada resultado del logaritmo al aumentar el valor de "**x**", a mayor valor de "**a**" la curva de la función se vuelve más pronunciada. Resaltar que si el valor de "**a**" es muy pequeño la diferencia entre los resultados de la función son decimales, algo que se debe evitar para el cálculo de la vida y el "mana", ya que el jugador apreciaría un incremento de estos valores casi lineal, haciendo el juego predecible.

Para los desarrolladores que usen el prototipo se recomienda que el valor de "**a**" en la función que utilice el atributo constitución, sea mayo que el valor de "**a**" en la función logarítmica que usa el atributo inteligencia, ya que esto permitirá que la vida del jugador principal se incremente más que el "mana", siendo una de las tendencias que presentan la mayoría de los juegos RPG de base por turnos. Además, las bases del logaritmo en ambas funciones deben ser iguales, para evitar una desproporción en el crecimiento entre el "mana" y la vida.

En la figura 12 se ejemplifica con algunos valores asignados a las variables, como se comporta la función **y=a*Log(x)+c**. En este caso la función se usa para el cálculo del "mana" y en esta se aprecia como el "mana" del actor será el acumulado del "mana" calculado para cada "**x**" mayor que 1 y menor o igual que 10, cuando el atributo inteligencia tiene valor 10.

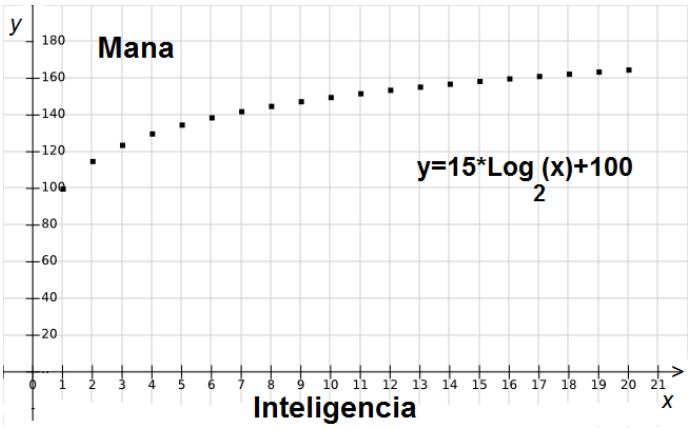


Figura 12: Comportamiento de la función logarítmica $f(x) = 15\ log_2 x$.

En la figura 13 se muestra el comportamiento de la función **y= a*Log(x)+c** con un valor diferente asignado a la variable "**a**". En este caso la función se usa para el cálculo de la vida. Además, en la figura se aprecia como la vida del actor será el acumulado de la vida calculado para cada "**x**" mayor que 1 y menor o igual que 6, cuando el atributo inteligencia tiene valor 6.

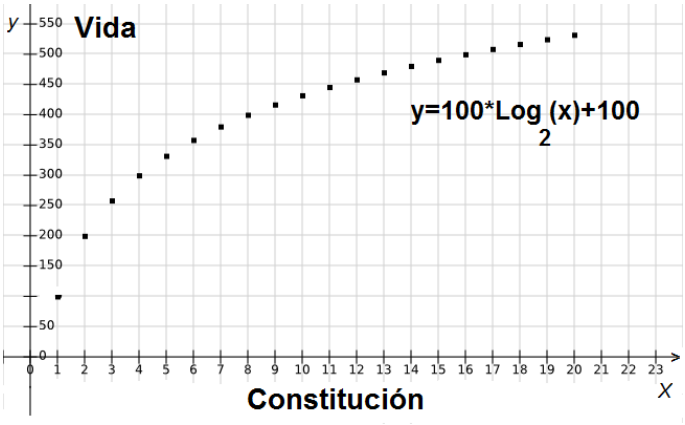


Figura 13: Comportamiento de la función logarítmica ***f(x) =100*** $\log_2 x$.

x	y= 15*Log(x)+100	y= 100*Log(x)+100
1	100	100
2	115	200
3	123	258
4	130	300
5	134	332
6	138	358
7	142	380
8	145	400
9	147	416
10	149	432
11	151	445
12	153	458
13	155	470
14	157	480
15	158	490

Figura 14: Resultados de dos funciones logarítmicas con distintos valores de "a".

Función utilizada para incrementar el valor estadístico "velocidad", asociado al atributo destreza.

El aumento de la velocidad en los actores es constante; cada vez que aumente un valor el atributo destreza, la velocidad de ataque del actor se calcula usando la función lineal **y = m(x-1) + c** para

x>1 (cuando "**x**" tiene valor 1 la velocidad de ataque es la definida como la inicial en el actor). En dicha función "**y"** es la velocidad de ataque del actor, "**x**" el valor del atributo destreza correspondiente y "**c**" es una constante. El valor de "**m**" es la pendiente, esta implica la porción de la destreza que aporta a la velocidad. La destreza crece por el impacto de la pendiente. El valor definido para "**c**" es correspondiente a la velocidad inicial de ataque definida para el actor. Se recomienda a los desarrolladores que la velocidad inicial de ataque de los enemigos sea menor que la del jugador principal, para así propiciar un inicio del videojuego con relativa facilidad para el jugador. Además, los valores para "**m**" y "**c**" deben ser decimales, para que el progreso de la barra de tiempo durante el combate sea visible (no ocurra de manera muy rápida).

Función utilizada para incrementar el valor estadístico "defensa", asociado al atributo fuerza.

Para calcular la defensa física de los diferentes actores en el videojuego se usa una función logarítmica. Cada vez que aumente un valor del atributo fuerza, la defensa física del actor aumenta en forma acumulativa bajo la función **y = a*Log(x)**, donde "**y**" es la defensa a aumentar, "**x**" el valor del atributo fuerza y "**a**" es una constante que amplía la diferencia entre cada resultado de la función al incrementar el valor de "**x**". El valor que tenga "**a**" va a ser correspondiente a la diferencia máxima que exista entre los valores obtenido al usar la función. Se recomienda a los desarrolladores que usen el prototipo, que la base del logaritmo sea menor o igual que 1.8, ya que si la base es mayor que 1.8 el crecimiento en un principio será más lento, dificultando los combates iniciales en el videojuego para el jugador. Este valor propuesto para la base del logaritmo es el resultado empírico obtenido a través de prueba y error en la búsqueda de resultados aceptables.

En la figura 15 se ejemplifica con algunos valores asignados a las variables, como se comporta la función **y=a*Log(x)**. Además, en la figura se aprecia que si atributo fuerza tiene valor 4, la defensa del actor será el acumulado de la defensa calculada para cada x mayor que 1 y menor que 5.

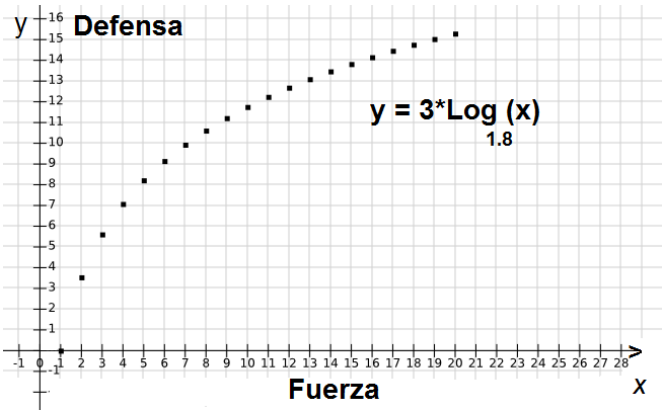


Figura 15: Comportamiento de la función logarítmica $f(x) = 3\ log_{1.8}\ x$.

Daño

El recurso **daño**, es usado a la hora de los combates cada vez que el actor realiza alguna acción de ataque. El **daño** infligido por un actor a otro, se calcula usando la ecuación:

daño = ataque – defensaContrario

ataque = *50 *(fuerza + 0.4*destreza + 0.2*inteligencia)*

Siendo la fuerza, destreza e inteligencia el valor de los atributos de quien realiza el ataque. La "defensaContrario" es el valor de la defensa física del actor (ver atributo fuerza). En caso que la defensa del contrario sea mayor que el ataque realizado, entonces el valor que arroja la ecuación es negativo. Si esto ocurre, el valor del daño calculado se toma como 0.

Experiencia

El recurso **experiencia** es esencial, siendo la columna vertebral del prototipo de videojuego del presente trabajo. La experiencia necesaria para alcanzar cada nivel se calcula usando una función exponencial. Dicha función es: *"**e**^{b}"* donde "***e***" es el número de Euler y "**b**" un número real positivo que aumenta con cada nivel a calcular. Se recomienda a los desarrolladores que utilicen el prototipo, que el valor inicial de "**b**" sea mayor que 4 y el incremento de esta variable en cada nivel a calcular sea menor o igual que 0.4; esto permite que subir de nivel sea relativamente asequible

en el inicio del videojuego, ya que la función para los primeros valores que toma "**b**" arroja resultados con muy poca diferencia entre uno y otro, lo que se traduce en que el jugador alcance los primeros niveles de manera rápida.

En la figura 16 se muestra el comportamiento de la función para los primeros 13 niveles del jugador principal, siendo 5 el valor inicial de "**b**" y un incremento de esta variable en 0.3 con cada nivel a calcular.

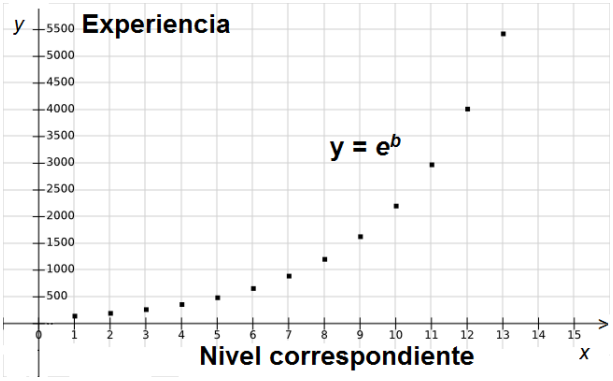


Figura 16: Comportamiento de la función exponencial para el cálculo de la experiencia necesaria por nivel.

Información:

La **información** es todo lo que es visible al usuario. Responde a la pregunta: ¿Qué tanto del estado del juego es visible para el jugador? (44).

En el presente videojuego la información que se ofrece al usuario de los enemigos es prácticamente nula, solo se podrá conocer el daño de los ataques que lancen y el daño que estos reciban del jugador principal. La vida, "mana", experiencia, nivel, y atributos del enemigo, son ocultos en todo momento. En cambio, del jugador principal se muestra información de todos los atributos, así como su "mana", vida, experiencia ganada y experiencia necesaria para alcanzar el siguiente nivel.

Reglas:

Sin las **reglas**, el videojuego no existiría; son su trasfondo y lo que da sentido a las acciones del jugador, que ha de respetarlas para poder seguir jugando. Imponen límites y fuerzan a tomar

determinados caminos para alcanzar una meta. Entre sus rasgos más importantes se destacan los siguientes (46):

- Limitan la acción del jugador. Son conjuntos de instrucciones.
- Son explícitas y no ambiguas.
- Están compartidas por todos los jugadores.
- Son fijas.
- Están entrelazadas y limitan el espacio del juego.
- Pueden moverse de un juego a otro.

Existen tres categorías de **reglas**: disposición (lo que se realiza al inicio del juego), progresión del juego (lo que pasa durante el juego), y resolución (qué condiciones hacen que el juego termine, y cómo es determinado el resultado en base al estado del juego) (44).

Las **reglas** establecidas para el prototipo de videojuego del presente trabajo son:

Reglas de disposición:

- El jugador no podrá usar ninguna habilidad, hasta que esta sea aprendida.
- Para moverse de un lado a otro lo hará caminando o corriendo.

Reglas de progresión:

- Huir del combate no es posible, la única forma de salir de este es obteniendo la victoria, es decir bajar la vida del enemigo a cero. Para lanzar una “Habilidad” durante el combate, la “mana” actual del jugador debe ser igual o mayor al coste de “mana” que tenga dicha “Habilidad”.
- Sistema de batalla por turnos activo. Mientras se elige que acción realizar por parte del usuario en el combate, el enemigo puede atacar sin respetar la espera.

Reglas de disposición:

- Si la vida del jugador principal es cero, el juego acaba.

Controladores

Según el diccionario de informática y tecnología (47), en un videojuego los **controladores** son los dispositivos empleados para controlar el personaje principal, objetos u otras entradas provistas por el juego de computadora. **Controladores** de videojuegos pueden ser: teclado, ratón, palanca de mando (*joystick*), u otro dispositivo diseñado para jugar que pueda recibir entradas.

En el prototipo de videojuego de base por turnos, el usuario ejecutará todas las acciones del jugador principal mediante el uso del teclado y mouse.

Conflicto:

El **conflicto** se genera a través de reglas, procedimientos y situaciones que obstaculizan cumplir los objetivos del videojuego (45).

El **conflicto** en este videojuego es básicamente lograr reaccionar adecuadamente a cada uno de los enemigos para vencerlos. Los enemigos varían en su forma de ataque y composición (pueden ser más proclives a habilidades que a ataques físicos o viceversa), por lo que se requiere de inteligencia y agilidad de parte del usuario para lograr derrotarlos eligiendo la acción que más daño le provoquen.

2.2.2 Componentes a desarrollar

El prototipo a desarrollar va a estar compuesto por varios componentes que pueden ser reutilizados, en su totalidad o en parte, en otros proyectos con características similares, con la idea de evitar soluciones redundantes a problemas que ya se han sido tratados con anterioridad.

- **Componente jugador principal:** Permitirá interactuar con el terreno incluyendo un sistema de locomoción que permite al personaje desplazarse de un lugar a otro. También incluirá un grupo de atributos modificables y estadísticas que se generan a partir de estos, usando las

funciones matemáticas antes descritas. El componente jugador principal a la vez poseerá un grupo de habilidades que podrá aprender a lo largo del videojuego.

- **Componente enemigo:** Incluirá un sistema de locomoción que permita al enemigo dado desplazarse de un lugar a otro. También contendrá un grupo de atributos modificables y estadísticas que se generan a partir de estos, usando las funciones matemáticas antes descritas. Se le dará la capacidad de reaccionar a la presencia cercana del jugador principal.
- **Componente terreno:** Posibilitará identificar el campo visual del jugador principal para que este esté en consonancia con la zona de combate correspondiente. Ya que los combates por turnos no ocurrirán en el terreno principal, sino en un terreno aparte con las mismas características visuales de donde se encontraba el jugador.
- **Componente simulador de combate:** Permitirá establecer el marco donde se desarrolla el combate por turnos, estableciendo el momento en que cada actor realiza la acción de ataque, así como los cálculos necesarios para el daño que estos ocasionen. Establecerá las reglas a seguir durante el combate en sí y proporcionará información acerca del estado del personaje principal.

2.2.3 Documento de Diseño

El Documento de diseño es una parte fundamental para el proceso de desarrollo de un videojuego. Es el punto de referencia, todos los componentes del equipo de trabajo lo usarán como guía para su trabajo (48). Es una labor de síntesis y de creación de concepto (49). Por lo tanto, es primordial que sea claro y que su formato sea accesible.

Concepto	
Título	Prototipo de videojuego RPG de base por turnos
Estudio/Diseñadores	Dashiel Lázaro Portel Batista
Género	RPG de base por turnos
Plataforma	Computadora personal
Versión	1

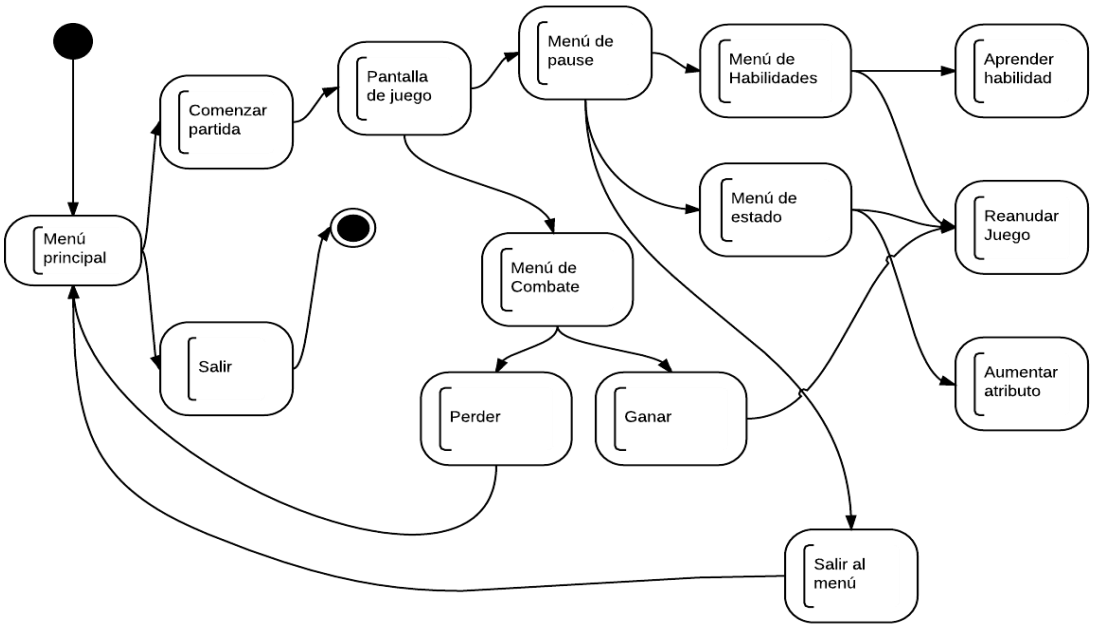
Sinopsis de Jugabilidad y Contenido	El jugador principal se encuentra en un mundo plagado de enemigos. Tendrá que enfrentarse a cada uno de ellos, para volverse más fuerte.
Mecánica	El jugador puede caminar libremente por el ambiente o correr si así lo desea, y puede elegir que atributos aumentar del jugador principal o que habilidades aprender a medida que vaya venciendo los enemigos.
Tecnología	Hardware (Requerimientos mínimos) • Tarjeta Gráfica con 512 MB de video. • Memoria RAM de 1 GB. • Microprocesador Dual-Core 1.2 GHz. Software: Gráficos 2D: • Photoshop CS6 Gráficos 3D: • Autodesk 3DMax Motor de Videojuego • Unity 3D Lenguaje de Programación: • JavaScript • C Sharp
Público	Está dirigido a todas las edades.

Historial de Versiones
Versión 1 (15-3-2016)
Visión General del Juego

El prototipo de videojuego RPG de base por turnos surge como una herramienta de apoyo a los futuros videojuegos que se desarrollen del mismo género. Con el mismo se logra tener implementadas funcionalidades básicas de los videojuegos RPG de base por turnos, como son: el simulador de combate, los controles básicos del jugador principal, los cálculos básicos de ataques y subir nivel. Así como mejorar habilidades y atributos del jugador principal, y los menús asociados a estos.

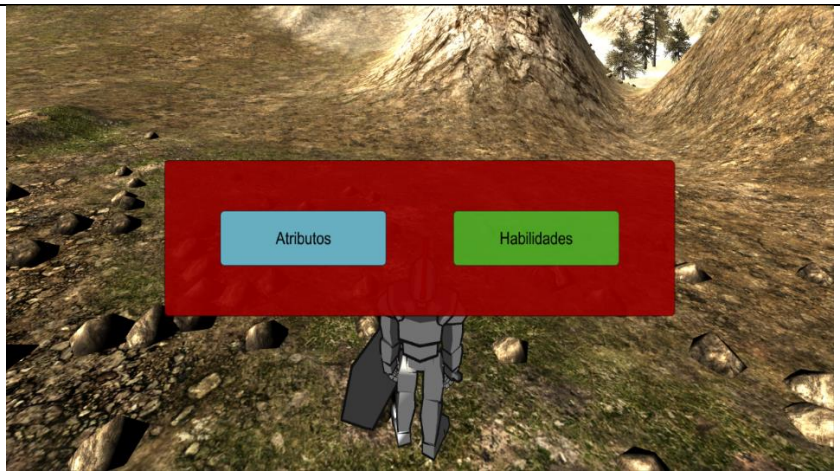
Mecánica del Juego	
Cámaras	Cámara 3D en 3era Persona.
Periféricos	Teclado y mouse.
Controles	Los controles que se utilizan durante el combate son: • Click izquierdo del mouse para interactuar con los comandos: “Atacar” y “Habilidad”. • X-Axis y Y-Axis del mouse para mover la cámara. Los controles que se utilizan fuera del combate son: • Tecla “W” para avanzar (caminar). • Teclas “D” y “A” para rotar a la derecha e izquierda respectivamente. • Tecla “Espacio” para saltar. • Teclas “Shift” + “W” para correr. • Tecla “Escape” para mostrar el menú de pause. • Click izquierdo del mouse para interactuar con los botones del menú de estado.
Puntajes	Cada enemigo derrotado proporciona experiencia que permite al jugador subir de nivel. Al alcanzar un nuevo nivel, obtendrá puntos especiales para mejorar los atributos y aprender habilidades.

Estado del juego



Interfaces

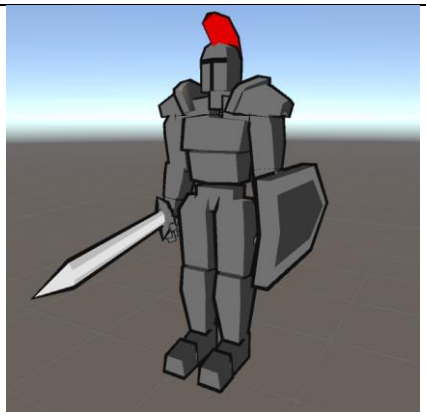
Menú de Pause

Descripción de la Pantalla	Interfaz cuando el usuario presiona la tecla “Escape”. En esta el usuario podrá acceder al Menú de Estado o al Menú de Habilidades al presionar el botón “Atributos” o el botón “Habilidades” respectivamente.
Imagen	

Menú de Estado	
Descripción de la Pantalla	Interfaz cuando el usuario presiona con click izquierdo del mouse el botón “Atributos” del Menú de Pause. En esta el usuario podrá: • Aumentar los diferentes atributos, siempre y cuando se tengan puntos especiales disponibles. • Ver la información referente al nivel actual del jugador, el ataque básico, la defensa mágica y física, la experiencia ganada hasta el momento, y la necesaria para alcanzar el siguiente nivel. • Visualizar dos barras; la barra roja muestra la vida actual del usuario en base a la vida máxima y la barra azul muestra el “mana” actual del usuario en base al “mana” máximo.
Imagen	
Menú de Habilidades	
Descripción de la Pantalla	Interfaz cuando el usuario presiona con click izquierdo del mouse el botón “Habilidades” del Menú de Pause. En esta el usuario podrá: • Ver las habilidades disponibles para aprender, así como el “mana”, el daño que ocasiona, y el progreso de aprendizaje de cada una. • Ver la especialidad de cada habilidad: mago, guerrero o pícaro. Al coincidir la especialidad de la habilidad con la del jugador, el daño se multiplica.

	• Seleccionar la habilidad a aprender, al dar *click* izquierdo encima de esta.
Imagen	
Menú de Combate	
Descripción de la Pantalla	Interfaz que se muestra al comenzar el combate. Está conformada por: • Una fina barra azul que indica el "mana" actual del jugador. Cada habilidad que se utilice consume "mana" lo que disminuye esta barra en cierta cantidad. • La barra roja indica la vida actual del jugador, esta disminuye a medida que el jugador recibe daño y al llegar a cero implica la muerte del jugador. • Una barra gris que se llenará progresivamente y al estar llena indica el momento a partir del cual el usuario puede realizar una acción; una vez realizada la acción la barra volverá a llenarse de nuevo. • En blanco y dentro de un círculo, está el número que indica el nivel actual de jugador. • Los comandos "Atacar" y "Habilidad". "Atacar" implica un ataque simple sin gasto de "mana". "Habilidad" implica un ataque con un gasto de "mana" correspondiente a la habilidad elegida.

Imagen	

Personajes	
Personaje Principal	
Nombre del personaje	Guerrero
Descripción	Guerrero medieval que posee un gran escudo y una espada de doble filo.
Imagen	
Habilidades	Las fortalezas del personaje son su ataque físico y habilidades de combate cuerpo a cuerpo.
Enemigos	
Minotauro	
Nombre del personaje	Minotauro

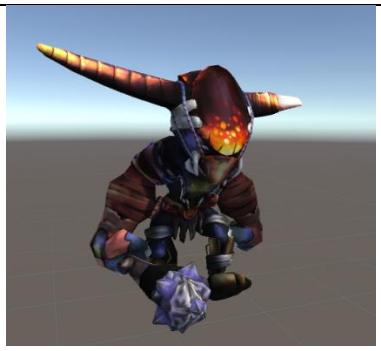
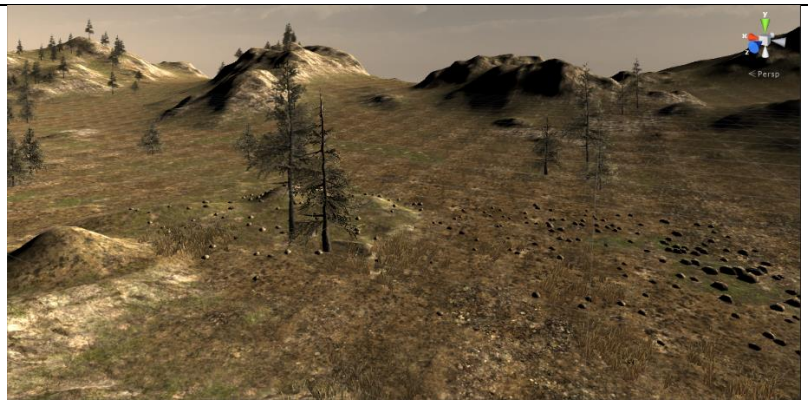
Descripción	Pequeño minotauro que habita en las cuevas de las montañas. Salen a cazar y suelen ser muy peligrosos.
Imagen	
Habilidades	Las fortalezas del personaje radican en el combate cuerpo a cuerpo. Realiza un salto mortífero.
Cazador de las montañas	
Nombre del personaje	Cazador
Descripción	Hombre de gran corpulencia armado con una enorme espada y un escudo.
Imagen	
Habilidades	Realiza una arremetida y ataque con espada que ocasiona mucho daño.
Calavera	
Nombre del personaje	Calavera
Descripción	Cuerpo consumido de un antiguo guerrero, que ha sido invocado para traer la muerte. Se mueve lentamente, esperando la oportunidad para atacar a cualquier ser vivo.

Imagen	
Habilidades	Realiza un ataque con espada que ocasiona mucho daño.

Niveles	
Terreno Montañoso	
Título del nivel	Terreno Montañoso.
Encuentro	Primer escenario en el que comienza el juego.
Descripción	Es un terreno rocoso con varias elevaciones. La tierra tiene un aspecto oscuro, con poca vegetación.
Enemigos	Minotauros y Cazadores.
Imagen	
Terreno Ártico	
Título del nivel	Terreno con nieve.
Encuentro	El jugador llega a esta escena una vez pase por un desfiladero entre dos elevaciones que conecta al Terreno Montañoso con este terreno.

Descripción	Es un terreno rodeado de montañas que hacen un pequeño valle. La tierra está cubierta de nieve y con grandes árboles.
Enemigos	Calaveras.
Imagen	

Equipo de Desarrollo	
Dashiel Lázaro Portel Batista	
Detalles de Producción	
Fecha de inicio	15/11/2015
Fecha de fin	15/4/2016

2.2.4 Producto *Backlog*

El *Backlog* es un “entregable” propuesto por Scrum que permite organizar y predecir las tareas que el equipo de desarrollo debe efectuar. Contiene descripciones genéricas de todos los requerimientos, funcionalidades deseables, entre otros.

Sinopsis	
	Este es un prototipo de videojuego RPG de base por turnos, con las mecánicas del jugador principal, los menús de estados y habilidades y el simulador de combate ya realizados.
	Product Backlog

Orden de prioridad por valor al juego	Ítem
1	Arte y funcionalidad básica del personaje principal (moverse en el mapa, correr, saltar y girar).
2	Diseño, interfaz y funcionalidad del menú de estado (aumentar los atributos).
3	Arte y funcionalidad de los enemigos (moverse en su radio de acción)
4	Funcionalidad para cambiar de lugar al entrar en combate (lugar de batalla con mismas característica visuales del mundo donde se encontraba el jugador principal).
5	Interfaz y funcionalidad del personaje principal cuando entra en combate (esperar, atacar, regresar, lanzar habilidad, celebrar cuando gane el combate y morir cuando su vida sea cero).
6	Funcionalidad de los enemigos durante el combate (atacar, esperar, regresar y morir cuando su vida sea cero).
7	Funcionalidad para regresar al lugar donde se encontraba el jugador principal antes de entrar en combate, actualizando la vida, “mana” y experiencia.
8	Diseño, interfaz y funcionalidad del menú de habilidades (aprender habilidad).

2.3 Fase *game*

En esta etapa, se tiene ya el alcance del proyecto bien definido, y por lo tanto una buena idea de lo que realmente es el prototipo de videojuego de base por turnos a desarrollar y lo que se debe hacer para lograrlo.

La fase de *Game* que suele llamarse *sprint* o fase de desarrollo, es donde se desarrollan las funcionalidades del juego. Su duración debe ser constante en cada ciclo. Sin embargo, este requisito no es estricto debido a que a veces el tiempo de cada *sprint* es diferente (42).

Todo el desarrollo del prototipo de videojuego del presente trabajo se ha dividido en cuatro *sprints.*

2.3.1 *Sprints Backlog*

El *Sprints Backlog* es un documento detallado donde se describe el cómo el equipo va a implementar los requisitos durante cada *sprint*. Un *sprint* en Scrum es el término que denomina a una iteración que está acotada en el tiempo, usualmente entre dos y cuatro semanas, durante la cual el equipo trabaja para convertir los ítem del producto *Backlog* en un incremento del producto potencialmente productivo (50). La duración del *sprint* debería ser lo suficientemente larga para crear algo de valor y con la suficiente calidad.

Sprint # 1					
Objetivo del *Sprint*:	Desarrollar las funcionalidades imprescindibles del jugador principal.			Fecha de inicio del sprint:	15/11/2015
				Fecha final del *sprint*:	10/12/2015
				Días laborables durante el *sprint*:	10
				Horas de trabajo por día:	4
		Prioridad en el producto *backlog*	**Horas de trabajo estimadas**	**Responsable**	**Estado**
Ítem del producto *backlog*	Personaje Principal	1			comprometido
tarea	Arte y movimiento de cámara.		1	Dashiel(Programador)	Terminado
tarea	*Animator* con los estados esperar, correr, saltar y girar.		3	Dashiel(Programador)	Terminado
tarea	Implementación para movimiento con controles de teclado y mouse.		3	Dashiel(Programador)	Terminado
tarea	*Animator* con los estados de combate (atacar, regresar, celebrar y morir).		3	Dashiel(Programador)	Terminado
	Menú de estado	2			comprometido
tarea	Arte y diseño de la interfaz del menú de estado.		3	Dashiel(Programador)	Terminado
tarea	Implementación para controles con el teclado del menú de estado.		1	Dashiel(Programador)	Terminado

tarea	Implementación para algoritmos de aumentar atributos, y calcular experiencia.		5	Dashiel(Programador)	Terminado

Sprint # 2					
Objetivo del *Sprint*:	Desarrollar las funcionalidades imprescindibles de los enemigos, y los algoritmos y métodos usados durante el combate.			Fecha de inicio del *sprint*:	15/1/2016
				Fecha final del *sprint*:	15/2/2016
				Días laborables durante el *sprint*:	15
				Horas de trabajo por día:	4
		Prioridad en el producto *backlog*	**Horas de trabajo estimadas**	**Responsable**	**Estado**
Ítem del producto *backlog*	Enemigos	3			comprometido
tarea	Arte y *Animator* de los enemigos con los estados atacar, regresar, esperar y morir.		4	Dashiel(Programador)	Terminado
tarea	Implementación de IA de los enemigos (moverse en su radio de acción).		4	Dashiel(Programador)	Terminado
	Simulador de Combate	4			Comprometido
tarea	Implementación de funcionalidad para cambiar de lugar al entrar en combate.		1	Dashiel(Programador)	Terminado
tarea	Implementación para calcular el turno de ataque del enemigo y jugador principal.		5	Dashiel(Programador)	Terminado
tarea	Implementación para calcular el daño que ocasiona el que ataca.		5	Dashiel(Programador)	Terminado
tarea	Implementación para actualizar el estado de los contrincantes durante el combate y después de finalizado este.		5	Dashiel(Programador)	Terminado
tarea	Implementación de funcionalidad para regresar al lugar anterior del jugador principal si salió victorioso del combate		1	Dashiel(Programador)	Terminado

Sprint # 3					
Objetivo del *Sprint*:	Desarrollar el ambiente visual del simulador de combate, así como las funcionalidades del menú de combate.			Fecha de inicio del *sprint*:	1/3/2016
				Fecha final del *sprint*:	25/3/2016
				Días laborables durante el *sprint*:	10
				Horas de trabajo por día:	4
		Prioridad en el	**Horas de trabajo estimadas**	**Responsable**	**Estado**

		producto *backlog*			
Ítem del producto *backlog*	Menú de combate	5			Comprometido
tarea	Arte y diseño de la interfaz del menú de combate.		3	Dashiel(Programador)	Terminado
tarea	Animación y posicionamiento de las cámaras usadas en el combate.		1	Dashiel(Programador)	Terminado
tarea	Implementación para controles con el teclado del Menú de Combate.		1	Dashiel(Programador)	Terminado
tarea	Implementación para comando “Atacar”.		5	Dashiel(Programador)	Terminado
tarea	Implementación para comando “Habilidad”.		5	Dashiel(Programador)	Terminado

***Sprint* # 4**					
Objetivo del *Sprint*:	Desarrollar las funcionalidades imprescindibles del Menú de Habilidades.			Fecha de inicio del *sprint*:	25/3/2016
				Fecha final del *sprint*:	15/4/2016
				Días laborables durante el *sprint*:	8
				Horas de trabajo por día:	4
		Prioridad en el producto *backlog*	**Horas de trabajo estimadas**	**Responsable**	**Estado**
Ítem del producto *backlog*	Menú de Habilidades.	6			comprometido
tarea	Arte y diseño de la interfaz del Menú de Habilidades.		5	Dashiel(Programador)	Terminado
tarea	Implementación para controles con el teclado del Menú de Habilidades.		3	Dashiel(Programador)	Terminado
tarea	Implementación para calcular experiencia necesaria para aprender la habilidad elegida.		5	Dashiel(Programador)	Terminado

Conclusiones parciales

Se realizó el documento de diseño el cual permite guiar al equipo de trabajo durante el desarrollo. Los artefactos generados que establece la metodología de desarrollo de software Scrum, propician todos los elementos necesarios para comenzar la fase de implementación de manera objetiva y organizada. Además, los *sprints* definidos en la fase game arrojan pequeñas partes del producto, en cortos periodos de tiempo, que pueden ser probados. Las funciones matemáticas utilizadas para el crecimiento de los valores estadísticos proporcionan un inicio de videojuego fácil y atrayente para el jugador.

Capítulo III: Implementación y pruebas

3.1 Introducción

Como una herramienta necesaria de ayuda y guía durante el desarrollo del videojuego, en el presente capítulo se expone el modelo de dominio del prototipo de videojuego. Además de explicar el funcionamiento y descripción de los *scripts* creados durante la fase de desarrollo, los componentes reutilizables con los que cuenta el prototipo y las pruebas realizadas a este una vez terminada dicha fase.

3.2 Modelo de dominio

Un modelo del dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés (51).

GameObjects son los objetos fundamentales en Unity 3D. En las escenas estos objetos pueden instanciarse en múltiples ocasiones, lo que se traduciría en múltiples objetos del mismo tipo en dicha escena. Un *GameObject* puede estar conformado por otros *GameObjects*, estableciendo una relación jerarquía entre ellos.

No obstante, los *GameObjects* no logran mucho por sí mismos. Dependiendo de qué tipo de objeto se desea crear, ya sea un personaje, un ambiente o un efecto especial, entre otros, a los *GameObjects* se les pueden adjuntar un grupo de componentes que brinda Unity 3D para darle determinadas propiedades según sea necesario, y los scripts que dictarán el comportamiento de cada uno en los diferentes casos.

El modelo de dominio que se presenta a continuación está orientado a la relación de los distintos *GameObjects* creados en el prototipo de videojuego (*ver figura 17*).

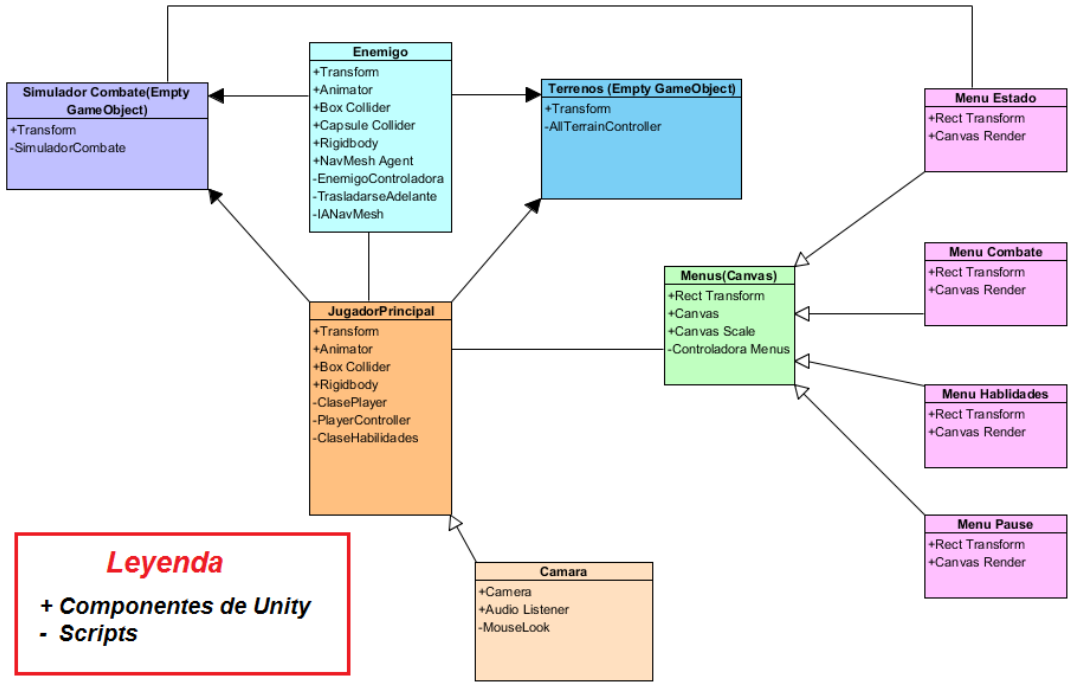


Figura 17: Modelo de dominio.

GameObjects que conforman el modelo de dominio:

- Enemigo: Contiene los componentes y *scripts* necesarios para el correcto comportamiento de los enemigos en el videojuego, así como sus características.
- Jugador Principal: Contiene los componentes y *scripts* necesarios que definen el comportamiento del jugador principal durante las diferentes acciones que realice el usuario. También contiene las características del jugador principal.
- Scripts: Contiene los *scripts* encargados de controlar y guiar el combate por turnos, así como la selección de la zona de combate correspondiente.
- Menús (Canvas): Contiene las interfaces 2D de los diferentes menús usados en el videojuego: Menú de Pause, Menú de Habilidades, Menú de Combate y Menú de Estado.
- Camera: Permite la visualización en tercera persona del personaje principal, y contiene el *script* necesario para ser controlada por el usuario durante el videojuego.

3.3 Componentes de Unity 3D

Por defecto, todos los *GameObjects* tienen automáticamente un componente *Transform*. Esto es porque el *Transform* dicta donde está ubicado el objeto, así como la escala y rotación que presenta. Sin un componente *Transform*, el *GameObject* no tendría un lugar en el mundo. Además, existen componentes para el trabajo con las mallas; componentes para la funcionalidad de audio y cámara; componentes relacionados con la física (*Colliders* y *Rigidbodys*), sistemas de partículas, sistemas de búsqueda de ruta (*NavMesh*), componentes personalizados de terceros, entre otros.

3.3.1 Componentes de visión y audio

- ***Camera*:** Las cámaras son los dispositivos que capturan y muestran el mundo al jugador. El número de cámaras a utilizar en una escena es ilimitado y la configuración puede hacerse en cualquier orden y en cualquier región de la pantalla.
- **Audio *Source*:** Este componente reproduce un clip de audio en la escena. El clip se puede escuchar mediante un *Audio Listener* o por medio de un mezclador de sonido. *Audio Source* puede reproducir cualquier tipo de secuencias de audio y puede ser configurado para reproducir estas en 2D, 3D, o como una mezcla (*SpatialBlend*).
- **Audio *Listener*:** Este componente actúa como un dispositivo de entrada de micrófono. Se recibe información desde cualquier fuente de audio que figura en la escena y reproduce sonidos a través de los altavoces del ordenador.

3.3.2 Componentes de colisiones

- ***Box Collider*:** Este componente tiene forma de un cubo y detecta colisiones de objetos que penetren en este. Es muy usado en puertas, paredes, plataformas, torso humano, cajas, entre otros.
- ***Capsule Collider*:** Este componente está compuesto por dos medias esferas unidas por un cilindro. Se usa para detectar colisiones.

- ***Sphere Collider*:** Este componente tiene forma de esfera y detecta colisiones de objetos que penetren en esta. El *Sphere Collider* se puede redimensionar a escala uniforme. Es muy utilizado en piedras, pelotas y otros cuerpos con forma esférica.
- ***Rigidbody*:** Permite a los *GameObjects* actuar bajo el control de la física. Este puede recibir fuerzas para que los objetos se mueven de una manera realista. Cualquier *GameObject* que contenga un *Rigidbody* va a estar influenciado por la gravedad, este actúa bajo fuerzas a través de secuencias de comandos, o interactúa con otros objetos.

3.3.3 Componentes de animación y navegación

- ***Animator*:** Permite organizar y mantener un conjunto de animaciones en un personaje u objeto. En la mayoría de situaciones, es normal tener varias animaciones y cambiar entre estas cuando se producen determinadas condiciones del juego.
- ***NavMesh Agent*:** Este componente ayuda a crear personajes que se evitan entre sí mientras se mueven a un objetivo. Los personajes o agentes razonan sobre el mundo del videojuego utilizando el *NavMesh* y saben cómo evitarse entre ellos, así como a otros obstáculos presentes. Para lograr este fin buscan el camino más corto libre de obstáculos entre dos puntos dados.

3.3.4 Componentes de interfaz (Canvas)

- **Canvas:** Representa un espacio abstracto donde se presenta la interfaz de usuario. Todos los elementos de la interfaz deben ser hijos de un *GameObject* que tiene un componente de *Canvas* adjunto.
- **Canvas *Scaler*:** Se utiliza para controlar la densidad total de píxeles y los elementos de la interfaz al escalar la pantalla. Esta escala afecta a todo los *GameObject* hijos del *Canvas*, incluyendo los tamaños de fuente y bordes de imagen.
- ***Text*:** Muestra una pieza no interactiva de texto para el usuario. Esto se puede utilizar para proporcionar subtítulos o etiquetas para otros controles de la interfaz o para visualizar instrucciones u otro texto.

- ***Rect Transform*:** Este componente representa un rectángulo en el que se pueden colocar elementos de la interfaz de usuario. Si el objeto padre de un *Rect Transform* también es un *Rect Transform*, entonces este puede ser colocado y dimensionado en relación con el padre.

3.4 Scripts

Los *scripts* son un componente esencial en todos los videojuegos realizados en Unity 3D. Incluso el videojuego más simple necesita de los *scripts* para responder a las acciones del usuario y organizar eventos que sucedan cuando deban. Más allá de esto, los *scripts* se pueden utilizar para crear efectos gráficos, controlar el comportamiento físico de los objetos o incluso poner en práctica un sistema de inteligencia artificial a medida para los personajes del videojuego (52).

A continuación, se detallan algunos *scripts* implementados para este prototipo de videojuego, que son independientes a los componentes reutilizables del prototipo:

SonidosController:

Reproduce, pausa o detiene el sonido correspondiente a una posición dada en el arreglo de sonidos.

AtributosScript:

Transforma un valor entero en un *string*, y lo muestra en el componente *Text* correspondiente.

ControladoraMenus:

Permite mostrar y desactivar los menús de habilidades, estado y pause. Así como actualiza los diferentes valores que contienen dichos menús. También controla los mensajes que se muestran en pantalla.

MenuInicioScript:

Permite cargar la escena principal del videojuego o salir de este.

Para consultar la descripción gráfica de todos los *scripts* implementados ver **Anexo 1**.

3.5 Componentes reutilizables del prototipo

Los componentes reutilizables con que cuenta el prototipo poseen un grupo de características que lo personalizan como: valores iniciales, desenvolvimiento, su posición en el mapa, entre otras. Estos pueden ser reutilizados, en su totalidad o en parte en otros proyectos de videojuego RPG de base por turnos.

Durante la fase *pre-game* se definieron las características generales de estos componentes incluyendo su comportamiento básico, en la fase *game* se materializaron como componentes en los *sprints* realizados, y en la implementación se han desarrollado como "componentes reutilizables del prototipo". En el Manual de Usuario se detalla como modificar cada uno de estos componentes por separado.

3.5.1 Componente jugador principal

De entre todos los componentes, jugador principal destaca por ser aquel que es controlado por el usuario, que le representa y responde a las interacciones del teclado y mouse. Los valores de entradas que se le pueden modificar son:

- "Ptos Atributos": Define cuántos puntos especiales posee el desarrollador para asignar a los diferentes atributos en el Menú de Estado. Inicialmente tiene valor 0, es decir, deberá alcanzar un nuevo nivel para obtener puntos.
- "Ptos por Nivel": Define cuántos puntos especiales obtendrá con cada nivel alcanzado. Por defecto el valor es 3.
- "Veloc Barra Ataque": Define la velocidad inicial del jugador principal en los combates. Por defecto el valor es 0.006. Si se modifica este valor, afectará directamente el combate por turnos, permitiendo que sea más rápido o más lento que el enemigo a enfrentar.
- "Vida Inicial": Define el valor de la vida inicial del jugador principal en el videojuego. Por defecto el valor es 100.

- “Mana Inicial”: Define el valor del “mana” inicial del jugador principal en el videojuego. Por defecto el valor es 20.

A través del Menú de Estado, el jugador podrá interactuar con los distintos atributos del personaje principal como son: la fuerza, inteligencia, destreza y constitución. Estos atributos a medidas que son modificados durante el videojuego generan una serie de estadísticas: poder de ataque básico, defensa física, defensa mágica, vida, “mana” y experiencia. El valor por defecto de estos atributos en el jugador principal es 1. Si el jugador los incrementa haciendo uso de los puntos especiales disponibles, las estadísticas del personaje aumentan como se definieron en el apartado “Recursos” en el epígrafe “Elementos formales” del Capítulo 2.

En caso que se cree un personaje nuevo para el mismo proyecto o para cualquier otro se deben utilizar los mismos componentes que tiene el jugador principal en el modelo de dominio antes expuesto, entre los cuales están presentes los siguientes *scripts*:

ClasePlayer:

Controla y manipula todos los atributos relacionados con el personaje principal, como son fuerza, constitución, vida, “mana”, destreza, velocidad de ataque, poder de ataque, defensa mágica, defensa física, nivel, experiencia y así como la posición del jugador principal.

PlayerController:

Controla el movimiento del personaje. También controla todas las animaciones que el personaje debe realizar en cada acción que el usuario decida hacer, y se encarga de mezclar dichas animaciones.

ClaseHabilidades:

Controla y manipula todas las habilidades que el jugador aprenderá durante el videojuego.

HabilidadScript:

Indica qué habilidad eligió el usuario para aprender del Menú de Habilidades.

3.5.2 Componente enemigo

Durante la ejecución del videojuego el jugador no tiene ningún control sobre este componente. Todas las acciones que realiza están definidas por la inteligencia artificial de los *scripts* que lo componen, lo que le permite reaccionar a la presencia del jugador principal en un radio que puede ser modificado en el *Sphere Collider* adherido a los enemigos.

Los siguientes parámetros y atributos, pueden ser modificados según la necesidad del desarrollador:

- "Vida Inicial": Define el valor de la vida del enemigo en caso que el atributo "Constitución" tenga el valor por defecto.
- "Mana Inicial": Define el valor del "mana" del enemigo en caso que el atributo "Inteligencia" tenga el valor por defecto.
- "Experiencia Premio": Define el monto de experiencia que ganará el jugador principal si vence en el combate contra el enemigo correspondiente.
- "Fuerza Enemigo": A mayor valor, mayor es la defensa física del enemigo dado. Dirigirse al apartado "Recursos" en "Elementos Formales" del Capítulo II, para observar como el atributo afecta la defensa física del enemigo. El valor por defecto es 1, y se recomienda que sea mayor que 0.
- "Inteligencia Enemigo": A mayor valor, mayor es el "mana" del enemigo dado. Dirigirse apartado "Recursos" en "Elementos Formales" del Capítulo II, para observar como el atributo afecta el "mana" y la defensa mágica del enemigo. El valor por defecto es 1, y se recomienda que sea mayor que 0.
- "Destreza Enemigo": A mayor valor, mayor es la velocidad de ataque del enemigo dado. Dirigirse apartado "Recursos" en "Elementos Formales" del Capítulo II, para observar como el atributo afecta la velocidad de ataque del enemigo. El valor por defecto es 1, y se recomienda que sea mayor que 0.

- "Constitución Enemigo": A mayor valor, mayor es la vida del enemigo dado. Dirigirse apartado "Recursos" en "Elementos Formales" del Capítulo II, para observar como el atributo afecta la vida del enemigo. El valor por defecto es 1, y se recomienda que sea mayor que 0.

En caso que se desee crear un enemigo nuevo para el mismo proyecto o para cualquier otro se deben utilizar los mismos componentes que posee el enemigo en el modelo de dominio antes expuesto, entre los cuales están presentes los siguientes *scripts*:

ClaseEnemigo:

Controla y calcula todos los atributos relacionados con el enemigo, como son fuerza, constitución, vida, "mana", destreza, velocidad de ataque, poder de ataque, defensa mágica, defensa física, y posición.

EnemigoControladora:

Detecta cuando el jugador principal colisiona con el *Box Collider* del enemigo. Al colisionar, el *script* se encarga de notificar a los demás *scripts* relacionados para dar inicio al combate. Además, es el encargado de crear el objeto enemigo con los valores y atributos correspondientes.

TrasladarseAdelante:

Traslada hacia adelante el objeto al cual esté adherido, cuando se cumpla una condición dada.

IANavMesh:

Permite al enemigo desplazarse aleatoriamente entre varias posiciones definidas por el usuario, y hace que el enemigo reaccione a la presencia del jugador principal caminando hacia este cuando irrumpa en su radio de acción.

3.5.3 Componente terreno

Como se definió anteriormente, cuando el jugador principal entra en combate con algún enemigo, el lugar donde se desarrolle este (llamado zona de combate), tiene que estar en consonancia con el campo visual que tenía el jugador. Esto se debe a que las zonas de combates y el terreno principal están relacionados entre sí, pero no ocupan el mismo espacio ni *GameObject* en la escena de Unity 3D. Por lo tanto, el videojuego tiene que saber qué zona de combate corresponde a cada ambiente visual.

Para lograr el funcionamiento descrito, este componente está compuesto por el terreno principal del videojuego, por los *BoxColliders* que controlan el cambio entre un aspecto visual del terreno y otro una vez el jugador haga contacto con estos, y por las zonas de combates. Además, el componente utiliza los scripts que se detallan a continuación:

AlTerrainController:

Permite crear, modificar o destruir la zona de combate correspondiente al momento y lugar del jugador principal.

CambiarZonaCombate:

Indica cuando cambiar de zona de combate a la que esté relacionada con el campo visual del jugador. Está adherido a los *BoxColliders* en el mapa que regulan cuando el jugador principal cambia de ambiente visual.

ZonaCombatePosiciones:

Proporciona información de cuál es la posición a tomar en la zona correspondiente por el jugador y el enemigo durante el combate. Esta adherido a cada zona de combate que tenga el prototipo.

3.5.4 Componente simulador de combate

El componente no permite ninguna modificación y es totalmente dependiente de los demás componentes descritos. En este se utilizan las ecuaciones definidas en el Capítulo 2 para el cálculo del daño, y utiliza los valores correspondientes a la velocidad de ataque de ambos actores para establecer el orden de ejecución de las acciones por parte de cada uno. Además, que integra en sí la interfaz que se muestra al jugador (Menú de Combate). También indica a los componentes jugador principal y enemigo que deben hacer los actores en cada momento del combate, ya sea esperar, atacar, lanzar habilidad, morir, entre otros. El componente constantemente verifica la vida de los involucrados en la batalla, y si esta llega a cero finaliza el combate por turnos. Todo esto lo logra utilizando los siguientes *scripts*:

SimuladorCombate:

Establece y controla todas las reglas presentes durante el combate como son: el turno de ataque del jugador principal y el enemigo, el daño que ocasiona el ataque de cada uno y las condiciones para finalizar el combate. Así como indica las animaciones necesarias a realizar y los sonidos a reproducir. También controla todos los elementos de la interfaz presentes durante el combate.

ComandoScript:

Indica qué acción elige el usuario durante el combate.

LanzarHabilidad:

Indica qué habilidad elige el usuario para ejecutar durante el combate.

DetenerDesplazamientoEnemigo:

Detiene el desplazamiento del enemigo una vez que comience la animación “Atacar” correspondiente.

FinalizoAccionAtacar:

Notifica cuando finalizan las animaciones de ataques, y antes de retornar a su posición inicial.

FinalizoAccionEnemigo:

Notifica cuando el enemigo regresa a su posición inicial en el combate.

FinalizoAccionPlayerP:

Notifica cuando el jugador principal regresa a su posición inicial en el combate.

FinalizoGolpeoEnemigo:

Notifica cuando el enemigo termina cada animación de golpeo.

FinalizoGolpeoPlayer:

Notifica cuando el jugador principal termina cada animación de golpeo.

FinalizoSimulacionCombate:

Notifica cuando el jugador principal termina la animación “Celebrar” y por ende el fin del combate.

3.6 Pruebas

Las pruebas de videojuegos o *testing* es una parte vital del proceso de desarrollo. Es el componente que analiza si el videojuego está listo para ser publicado o no. Provee al proceso de desarrollo de un ojo crítico en la constante búsqueda de errores, inconsistencias, incoherencias, y perfección. El *testing* de videojuegos es la aplicación de atención al detalle metódicamente para encontrar oportunidades de mejoras en este, así como el descubrimiento y documentación de los errores e incoherencias (53).

No existe un método estándar para las pruebas, y la mayoría de las metodologías son desarrolladas por desarrolladores de videojuegos individuales. Las metodologías se están continuamente refinando y los *testing* se llevan a cabo según el tipo de juegos (54).

A pesar de que cada videojuego es distinto, darle un poco de estructura al *testing* maximiza los resultados independientemente del videojuego que se esté desarrollando, con este propósito, el *testing* se debe orientar hacia las siguientes categorías (53):

- Jugabilidad y Mecánicas: Las reglas del juego.
- Arte: Todos los componentes visuales del juego.
- Interfaz: Todas las formas de comunicación en primer plano con el jugador.
- Música y Efectos de Sonido: Todos los componentes relacionados con el audio.

3.6.1 Prueba de Humo

Prueba de humo consiste en verificar que el videojuego presente una velocidad adecuada sin interrupciones y que el flujo de las pantallas sea correcto, y es por esto que es la prueba que debe hacerse primero, aunque sea por corto tiempo y de manera atropellada. Aún si el videojuego se ha probado muchas veces y está casi listo para ser entregado, siempre se deben hacer pruebas de humo para asegurarse de que esté funcionando correctamente (55).

- ✓ El prototipo de videojuego durante la sucesión de las diferentes acciones del jugador principal, el recorrido de este por el terreno, y el cambio hacia el lugar de combate; presenta una velocidad adecuada sin interrupciones, manteniendo una velocidad estable entre 64 y 72 fotogramas por segundo (FPS). Además, la interacción con los diferentes menús es fluida.

3.6.2 Prueba de Remojo

Consiste en dejar el videojuego corriendo en un dispositivo específico por largos períodos de tiempo en una pantalla crítica, por ejemplo, dejarlo en la pantalla de pausa por unas 8 horas, y regresar para observar el comportamiento del juego. Se debe observar si el juego se ha ralentizado o presenta algún problema en el comportamiento de los diferentes elementos que lo componen. Si sucede algo así, entonces es una señal de que existe una gotera de memoria (datos basura que

no se destruyen correctamente y se almacenan constantemente en el dispositivo). Mientras más tiempo se deje el juego corriendo, más cantidad de datos basura se guardan en el dispositivo (55).

- ✓ El prototipo de videojuego de base por turnos se dejó ejecutando en el terreno principal durante ocho horas (*ver figura 18*). Al comenzar la prueba, la velocidad en la que se muestran los fotogramas en el videojuego oscilaban entre 64 y 72 fotogramas por segundo (FPS), con un uso de memoria RAM de alrededor de 300 Mb. Finalizada la prueba el videojuego aun corría en el mismo intervalo de fotogramas por segundo, y mantenía idéntico uso de memoria RAM. También los enemigos continuaron su movimiento de manera usual, sin daño alguno a su mecánica. Por lo tanto, el videojuego no presentó gotera de memoria, ni cambios en su comportamiento habitual.

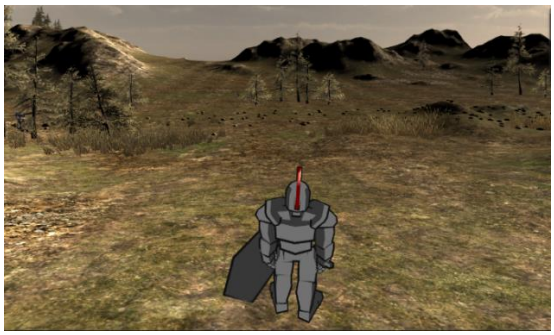


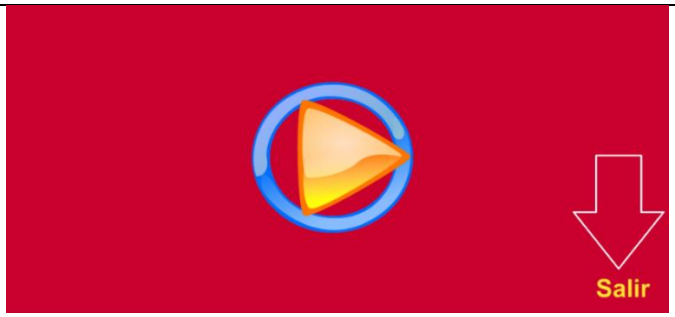
Figura 18: Pantalla del videojuego durante la Prueba de Remojo.

3.6.3 Prueba de Funcionalidad

La prueba de funcionalidad busca problemas generales en el juego en sí o de su interfaz de usuario, tales como problemas de estabilidad, problemas de mecánica de videojuego, y la integridad de los elementos del videojuego (54).

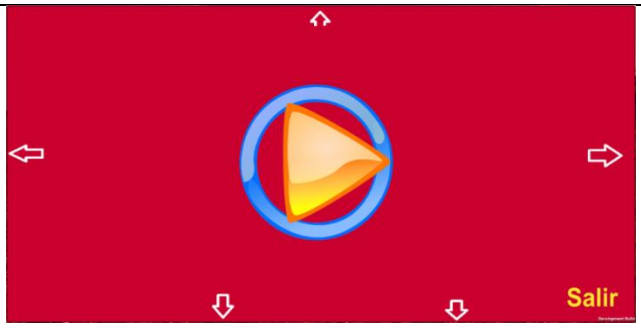
Durante las pruebas de funcionalidad fueron detectados varios errores en la interfaz gráfica y la jugabilidad. A continuación, se detallan.

Error funcional #1	
Id:	1

Resumen:	Error en la interfaz del Menú Inicio.
Descripción:	Al intentar presionar el botón salir del Menú Inicio, no se obtuvo respuesta alguna.
Pasos a Reproducir:	Paso 1: Abrir el juego normalmente. Paso 2: Presionar el botón salir.
Información Adicional:	Ver referencias (Imagen adjunta de la pantalla donde se detectó el error).
Comportamiento Esperado:	El videojuego debe cerrarse por completo.
Plataforma:	Dispositivo: Laptop Asus. Microprocesador: i7 4ta generación. RAM: 8 Gb. Sistema Operativo: Windows 10 x64.
Asignado:	Dashiel Portel Batista
Archivos Adjuntos:	
Frecuencia:	Ocurre el 100% de las veces.
Severidad:	Media

Error funcional #2	
Id:	2
Resumen:	Error en la interfaz del Menú de Estado.

Descripción:	Al agotarse los puntos disponibles para aumentar los atributos, el mensaje de alerta “No hay puntos”, es poco visible ya que aparece por detrás del Menú de Estado.
Pasos a Reproducir:	Paso 1: Abrir el juego normalmente. Paso 2: Presionar el botón “Jugar” en el Menú de Inicio. Paso 3: Presionar la tecla “Escape” una vez comenzado el videojuego normalmente. Paso 4: Presionar el botón “Atributos”. Paso 5: Aumentar los atributos hasta quedarse sin puntos.
Información Adicional:	Ver referencias (imagen adjunta de la pantalla donde se detectó el error).
Comportamiento Esperado:	El mensaje de alerta “No hay puntos” debe ser completamente visible al usuario.
Plataforma:	Dispositivo: Laptop Asus. Microprocesador: i7 4ta generación. RAM: 8 Gb. Sistema Operativo: Windows 10 x64.
Asignado:	Dashiel Portel Batista
Archivos Adjuntos:	
Frecuencia:	Ocurre el 100% de las veces.
Severidad:	Media

Error funcional #3	
Id:	3
Resumen:	Error en la interfaz del Menú de Inicio.
Descripción:	Al intentar volver al Menú Inicio una vez comenzado el videojuego, esta no ocupa todo el espacio de la pantalla, dejando margen en los bordes.
Pasos a Reproducir:	Paso 1: Abrir el juego normalmente. Paso 2: Presionar el botón “Jugar” en el Menú de Inicio. Paso 3: Presionar la tecla “Escape” una vez comenzado el videojuego normalmente. Paso 4: Presionar “Salir del juego.”
Información Adicional:	Ver referencias (imágenes adjuntas de la pantalla donde se detectó el error).
Comportamiento Esperado:	El Menú Inicio debe ocupar todo el tamaño de la pantalla.
Plataforma:	Dispositivo: Laptop Asus. Microprocesador: i7 4ta generación. RAM: 8 Gb. Sistema Operativo: Windows 10 x64.
Asignado:	Dashiel Portel Batista
Archivos Adjuntos:	 
Frecuencia:	Ocurre el 100% de las veces.
Severidad:	Baja

Error funcional #4	
Id:	4
Resumen:	Error en el funcionamiento de la cámara.
Descripción:	Si el usuario movió la cámara antes de entrar en combate, una vez finalizado este, la cámara se muestra donde se quedó. Dificultando la visión del usuario, ya que puede perder el hilo de los acontecimientos si el combate se prolonga.
Pasos a Reproducir:	Paso 1: Abrir el juego normalmente. Paso 2: Presionar el botón “Jugar” en el Menú de Inicio. Paso 3: Entrar en combate con algún enemigo. Paso 4: Vencer en el combate.
Información Adicional:	Ver referencias (imágenes adjuntas: la primera es antes de entrar en combate y la segunda una vez finalizado este)
Comportamiento Esperado:	La cámara debe tomar la posición por defecto, permitiendo visualizar bien al jugador principal.
Plataforma:	Dispositivo: Laptop Asus. Microprocesador: i7 4ta generación. RAM: 8 Gb. Sistema Operativo: Windows 10 x64.
Asignado:	Dashiel Portel Batista
Archivos Adjuntos:	 

Frecuencia:	Ocurre el 100% de las veces.
Severidad:	Alta

3.6.4 Pruebas de Regresión

Después de solucionados los distintos errores que arrojaron las pruebas antes expuestas, se procedió a comprobar si los errores continuaban (regresión), y se ejecutaron pruebas similares para observar si las soluciones conllevaron a la aparición de nuevos errores. En cualquier caso, el resultado fue satisfactorio al no arrojar ningún nuevo error.

Conclusiones parciales

En el presente capítulo se abordan los aspectos relacionados a la implementación del prototipo de videojuego de base por turnos. Se definió el modelo de dominio que permitió conceptualizar la problemática planteada para la creación del videojuego. Se explican los componentes de Unity utilizados y los *scripts* que se implementaron para definir el comportamiento de los diferentes elementos del prototipo. Por último, se muestran los resultados de las pruebas a las que fue sometido.

Conclusiones generales

Se obtuvo un prototipo de videojuego RPG de base por turnos el cual brinda funcionalidades características de este género como son los combates por turnos, movimiento en terreno, inteligencia artificial de personajes no jugables, aumento de atributos y aprendizaje de habilidades.

La utilización de las funciones logarítmicas permite modelar el comportamiento de variables que aumentan su valor rápidamente al principio.

La utilización de las funciones lineales permite modelar el comportamiento de variables que aumentan su valor a un ritmo constante.

La utilización de las funciones exponenciales permite modelar el comportamiento de variables que un principio aumentan su valor lentamente.

La utilización de la metodología de desarrollo de software Scrum es una buena alternativa para el desarrollo de prototipos de videojuegos, debido a que describe un proceso iterativo e incremental, ideal para equipos de desarrollo pequeños, y facilita realizar ajustes durante el proceso de desarrollo.

El uso del documento de diseño en el proceso de desarrollo permitió una mejor organización para el comienzo de la implementación.

El uso de la plataforma de desarrollo de videojuegos Unity 3D es adecuado para prototipos de videojuegos debido a que ofrece facilidades para el trabajo con objetos 3D, a los cuales se les puede modificar su comportamiento mediante el uso de *scripts* y componentes nativos de dicho motor.

Recomendaciones

A continuación de reflejan algunas recomendaciones que se consideran necesarias para futuras versiones del producto:

- Creación de inventario.
- Interacción con objetos.
- Agregar la funcionalidad de guardar y cargar partidas.
- Agregar diálogos entre los distintos personajes.

Referencias bibliográficas

1. Cómo crear un videojuego desde cero. *wikiHow.* [En línea] http://es.wikihow.com/crear-un-videojuego-desde-cero.

2. Adrian. Cómo empezar a programar videojuegos. *Genbeta:dev.* [En línea] 2013. http://www.genbetadev.com/programacion-de-videojuegos/como-empezar-a-programar-videojuegos.

3. Diccionario de la lengua española. [aut. libro] Real Academia Española. *Diccionario de la lengua española,.* 23.ª ed., Edición del Tricentenario.

4. Kordon, Fabrice. *An Introduction to Rapid System Prototyping.* 2002. ISSN: 0098-5589.

5. Maner, Walter. Sidar. *Sidar.* [En línea] 1997. http://www.sidar.org/recur/desdi/traduc/es/visitable/maner/Prototipado.htm.

6. *Prototyping for Tiny Fingers.* Rettig, Marc. 1994.

7. Suri, Marion Buchenau y Jane Fulton. *Experience Prototyping.* New York : s.n., 2000.

8. Salazar, Lucho. Gazafatonario IT. [En línea] 2012. http://www.gazafatonarioit.com/2012/07/ventajas-y-desventajas-del-uso-de.html.

9. Nicolas, Esposito. *A Short and Simple Definition of What a Videogame Is.* 2005. ISSN: 2342-9666.

10. Kramer, Wolfgang. *The Games Journal.* 2000.

11. Zyda, Mike. *From Visual Simulation to Virtual Reality to Games.* 2005.

12. Tavinor, Grant. The Definition of Videogame. *Contemporary Aesthetics.Volumen 6. .* 2008.

13. Marqués, Pere. *Los videojuegos.* 2000. Vol. vol. 8.

14. Brell, Marc. *Juegos de rol.* 2006.

15. Kim, John H. [En línea] 2003. http://www.darkshire.net/~jhkim/rpg/whatis/computer.html.

16. VIivanco, Matías Daniel Espinoza. *Desarrrllo de juego educativo RPG en tetéfonos móviles.* Santiago de Chile : s.n., 2009.

17. ¿Cuánto debe durar un videojuego? *3DJuegos.* [En línea] 2013. http://www.3djuegos.com/foros/tema/20234293/0/cuanto-debe-durar-un-videojuego/.

18. Albert García. ¿Cuánto debe durar un RPG? *Eurogamer.es.* [En línea] 2010. http://www.eurogamer.es/articles/cuanto-debe-durar-un-rpg.

19. Factory, Raul. La balanza del RPG: combate en tiempo real o por turnos. *Zehngames.* [En línea] 2014. http://www.zehngames.com/articulos/la-balanza-del-rpg-combate-en-tiempo-real-o-por-turnos/.

20. Anime Delta. *Anime Delta.* [En línea] 2010. http://anime-delta.forolatin.com/t372-evolucion-de-los-videojuegos-rpg.

21. Ultima III: Exodus . *Games Nostalgia.* [En línea] http://gamesnostalgia.com/en/game/exodus-ultima-iii.

22. Zelda II: The Adventure of Link. *Ocremix.* [En línea] http://ocremix.org/game/65/zelda-ii-the-adventure-of-link-nes.

23. Vallejo, Enrique Javier Lopez. *Conceptualización y desarrollo de un videojuego RPG de base por turnos.* 2014.

24. The First MMORPG - The Early History of the Genre. [En línea] http://mmohuts.com/news/the-first-mmorpg-the-early-history-of-the-genre/.

25. Dragon Quest VI: Los Reinos Oníricos. *MeriStation.* [En línea] http://www.meristation.com/nintendo-ds/dragon-quest-vi-los-reinos-oniricos/juego/1524444.

26. MeriStation. Dragon Quest VI. *MeriStation.* [En línea] http://www.meristation.com/nintendo-ds/dragon-quest-vi-los-reinos-oniricos/analisis-juego/1524444?p=1.

27. Final Fantasy. *Steam.* [En línea] 2013. http://store.steampowered.com/app/292120/.

28. EliteGuias. Guía Final Fantasy XIII. *EliteGuias.* [En línea] http://www.eliteguias.com/guias/f/ff13/final-fantasy-xiii-p2.php.

29. Liu, James. BoxCat. *BoxCat.* [En línea] http://box-cat.com/site/2013/07/11/re-imagining-jrpg-designing-for-mobile-audiences-part-2/.

30. Algorithm for dynamically calculating a level based on experience points? *Game Development.* [En línea] 2011. http://gamedev.stackexchange.com/questions/13638/algorithm-for-dynamically-calculating-a-level-based-on-experience-points.

31. How to develop RPG Damage Formulas? *Game Development.* [En línea] 2012. http://gamedev.stackexchange.com/questions/14309/how-to-develop-rpg-damage-formulas.

32. Keith, Clinton. *Agile game development with Scrum.* s.l. : Addison-Wisley. ISBN-10: 0321618521 ISBN-13: 978-0321618528 .

33. Bethke, Erik. *Game Development and Production.* s.l. : Wordware Publishing, Inc. ISBN: 1-55622-951-8.

34. Iglesias, Alejandro Adrián. *Desarrollo de Videojuegos.* Buenos Aires : s.n., 2011.

35. Onyett, Charles. Duke Nukem Forever Review. *IGN.* [En línea] 2011. http://pc.ign.com/articles/117/1176231p1.html.

36. Clayman, David. Starcraft II breaks 3 million sold. *IGN.* [En línea] 9 de 2010. http://pc.ign.com/articles/111/1117338p1.html.

37. Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert Cecil Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland y Dave Thomas. *Manifiesto ágil.* 2001.

38. Bates, Bob. *"Game Design".* 2004. ISBN-10: 1592004938.

39. Pekka Abrahamsson, Outi Salo, and Jussi Ronkainen. *Agile Software Development Methods.* s.l. : VTT Publications, 2002.

40. D.S. Cohen, Sergio A. Bustamante II. *"Producing Games: From Business and Budgets to Creativity and Design.* 2009. ISBN: 978-0-240-81070-6.

41. Beedle, Ken Schwaber and Mike. *Agile Software Development with Scrum.* s.l. : Prentice Hall PTR, 2001.

42. Gabriel Alexander Armendáriz Barreno, Milton Gonzalo Saltos Guaraca. *Adaptación de las metodologías ágiles Scrum y Extreme Game Develpomente en una metodoloíga para desarrollo de videojuegos Android. Caso Práctico: Desarrollo de un videojuego.* Riobamba, Ecuador : s.n., 2013.

43. Level 3: Formal Elements of Games. *Game Desing Concepts.* [En línea] 2009. https://gamedesignconcepts.wordpress.com/2009/07/06/level-3-formal-elements-of-games/.

44. Game Design Concepts. *Level 3: Formal Elements of Games.* [En línea] 2009. https://gamedesignconcepts.wordpress.com/2009/07/06/level-3-formal-elements-of-games/.

45. Dr.Omar Correa Madrigal, Ing. Jaime González Campistruz. Introducción al Desarrollo de Videojuegos. *rvirtual.* [En línea] https://rvirtual.uci.cu/files/globalgamejamconf2016/Conferencia_GamePlay_DrC_Omar_Correa.pdf.

46. Lacasa, Pilar. La reglas del juego. *Aprende y juega con EA.* [En línea] 2009. http://www.aprendeyjuegaconea.com/index.php?n3=14.

47. Definición de controlador de juego (control). *Alegsa.* [En línea] 2010. http://www.alegsa.com.ar/Dic/controlador%20de%20juego.php.

48. Sarea, Ikaskidetza. Documento de diseño de juego (DDJ). *SlideShare.* [En línea] 2013. http://es.slideshare.net/ikaskidetza/plantilla-ddj.

49. González, Daniel. El documento de diseño. *Pcactual.* [En línea] http://www.pcactual.com/articulo/zona_practica/paso_a_paso/4573/documento_diseno.html.

50. Demachy, Thomas. Extreme Game Development: Right on Time, Every Time. *Gamasutra.* [En línea] http://www.gamasutra.com/view/feature/2827/extreme_ga%20me_development_right_on_.php.

51. Larman, Craig. *UML y Patrones. 2ª Edición.* s.l. : Prentice Hall, 2003 .

52. Scripting. *Manual de Unity.* [En línea] Unity, 2016. http://docs.unity3d.com/es/current/Manual/ScriptingSection.html.

53. Tere. Game Testing 101: Lo básico. *El Tiempo. Blogs.* [En línea] 2015. http://blogs.eltiempo.com/the-9-bit-game/2015/02/09/game-testing-101-lo-basico/.

54. Testing - Metodología. *JUEGALIBRE.* [En línea] 2015. http://juegalibre.virtual.uniandes.edu.co/index.php/2015/07/27/testing-metodologia/.

55. Tere. Game Testing 101: Métodos y Reportes. *El Tiempo.* [En línea] 2015. http://blogs.eltiempo.com/the-9-bit-game/2015/02/23/game-testing-101-metodos-y-reportes/.

Glosario de términos

***Inteligencia artificial**:* Programa de computación diseñado para realizar determinadas operaciones que se consideran propias de la inteligencia humana, como el autoaprendizaje.

Asset Store: Es el hogar de una creciente biblioteca de *assets* comerciales y gratuitos creados por *Unity* Technologies y miembros de la comunidad.

RAM: Son las siglas de *random access memory*, un tipo de memoria de ordenador. En la RAM se cargan todas las instrucciones que ejecuta la unidad central de procesamiento (procesador) y otras unidades del computador.

FPS: Las imágenes por segundo (fotogramas por segundo o cuadros por segundo) (en inglés *frames per second*, FPS) es la medida de la frecuencia a la cual un reproductor de imágenes genera distintos fotogramas (*frames*).

Frame: Se denomina *frame* en inglés, a un fotograma o cuadro, una imagen particular dentro de una sucesión de imágenes que componen una animación.

Microprocesador: Es el circuito integrado central más complejo de un sistema informático; a modo de ilustración, se le suele llamar por analogía el “cerebro” de un computador.

Anexos

Anexo 1. Diagramas de los *scripts* usados en el prototipo de videojuego:

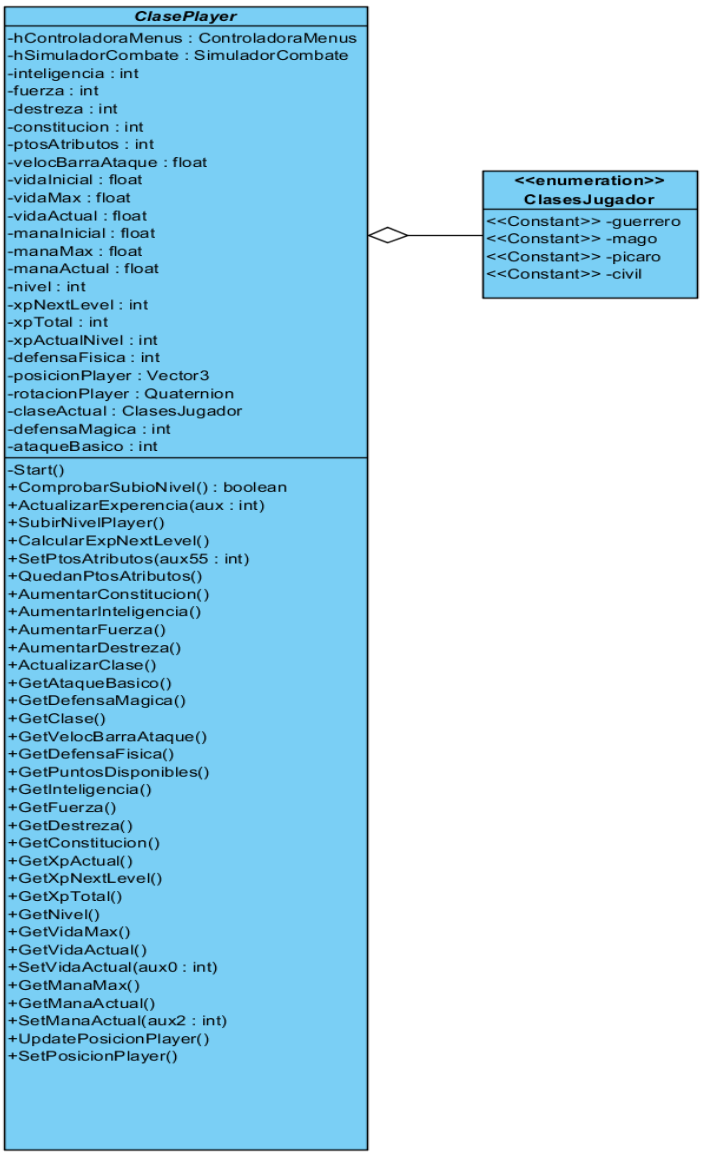


Diagrama 1: Script "ClasePlayer".

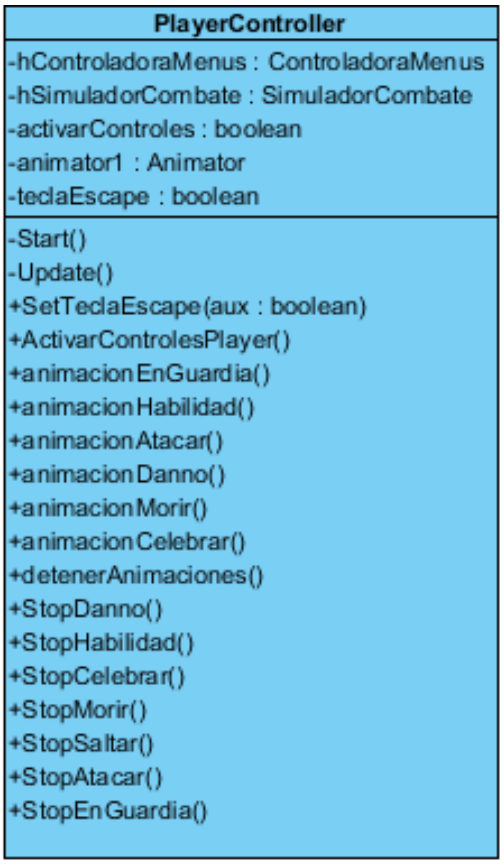


Diagrama 2: Script "PlayerController".

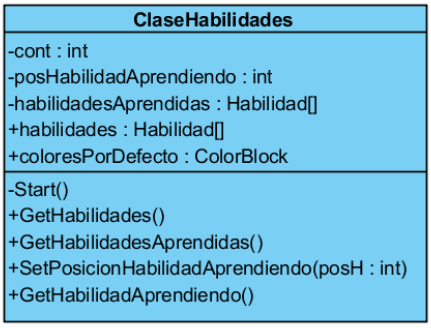


Diagrama 3: Script "ClaseHabilidades".

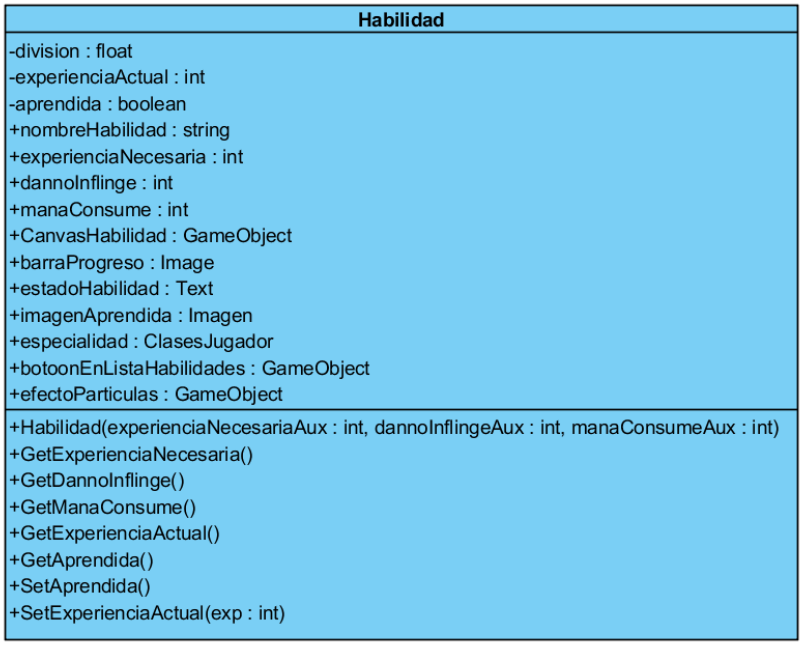


Diagrama 4: Script "Habilidad".

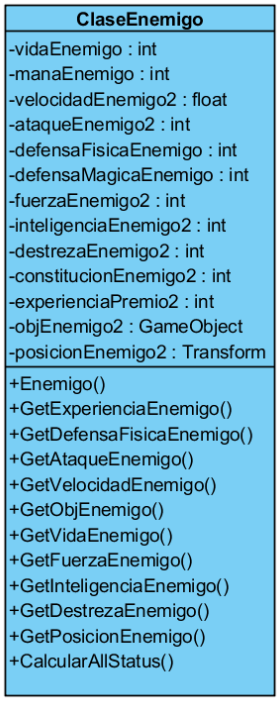


Diagrama 5: Script "ClaseEnemigo".

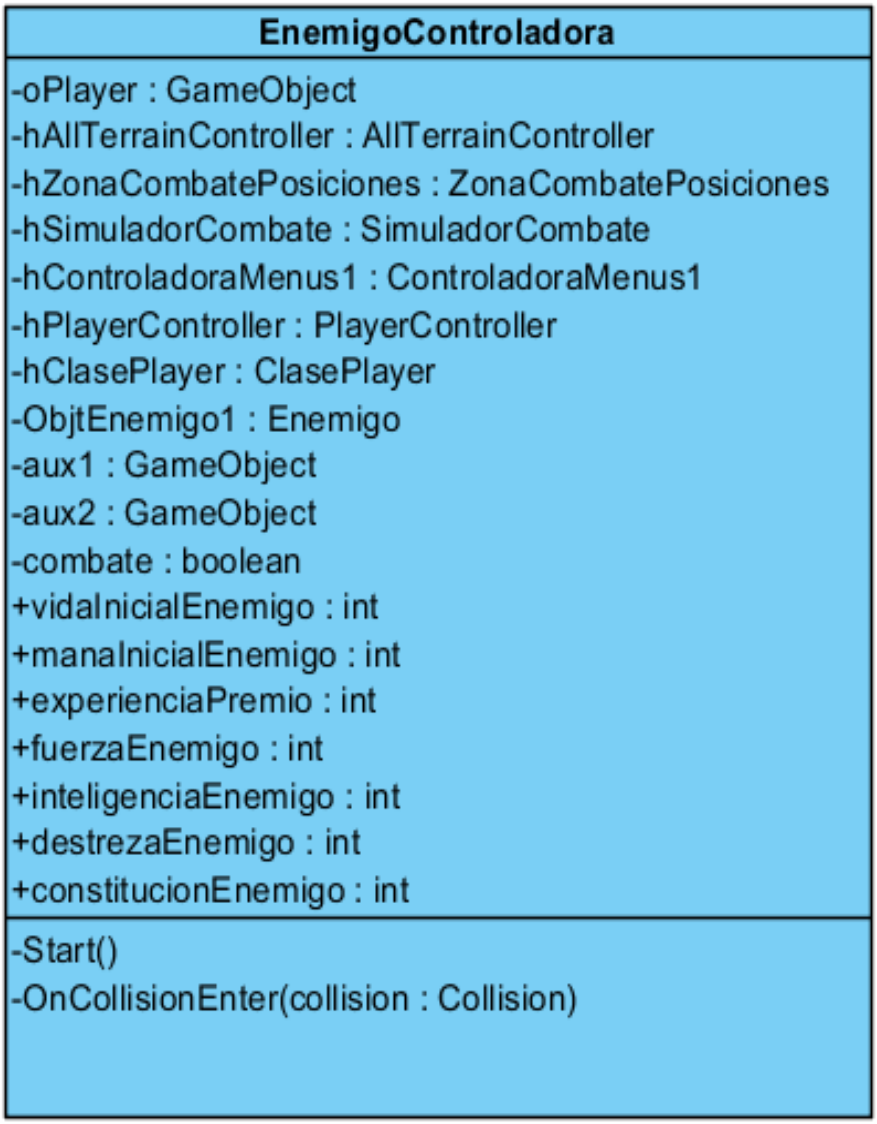


Diagrama 6: Script "EnemigoControladora".

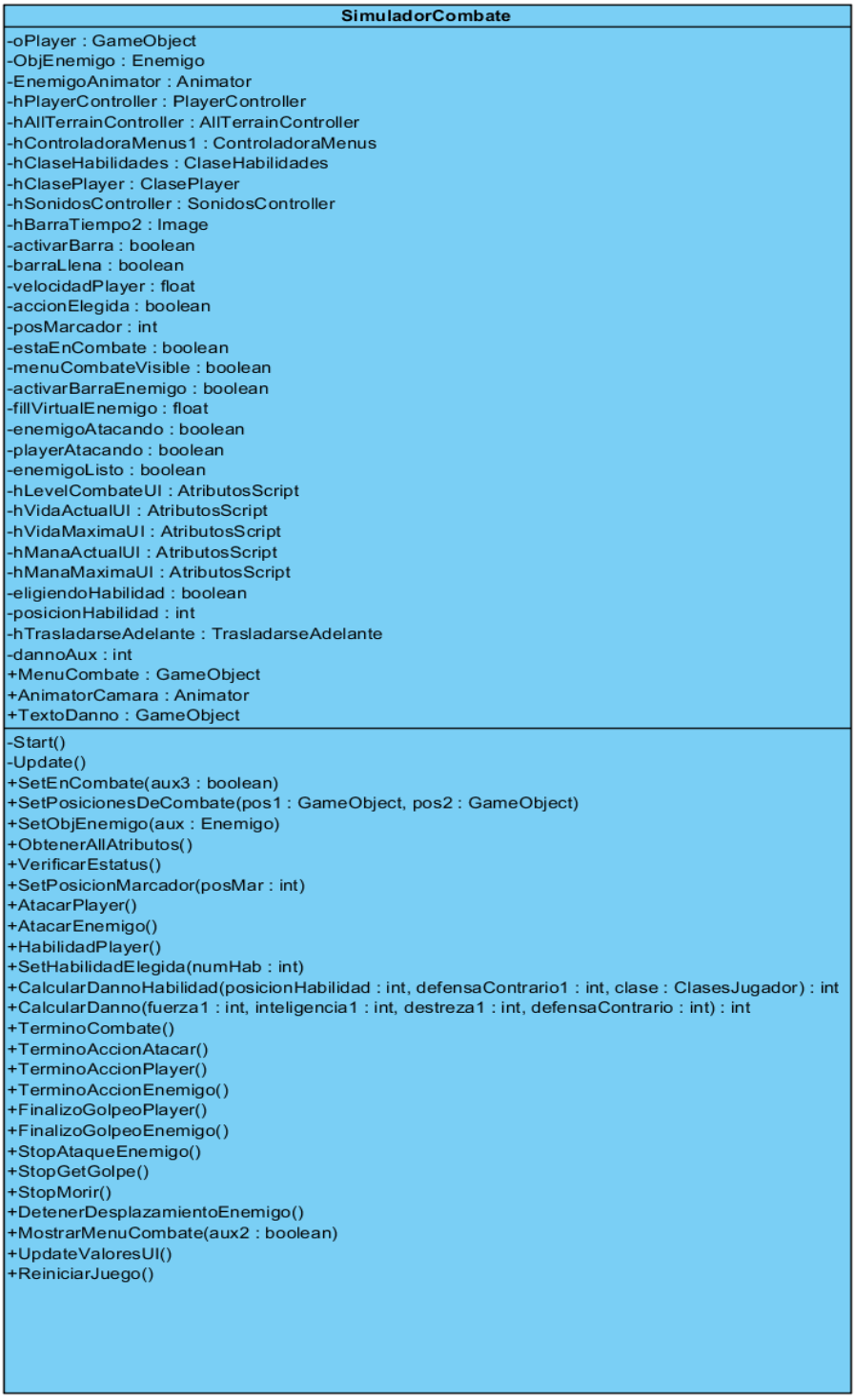


Diagrama 7: Script "SimuladorCombate".

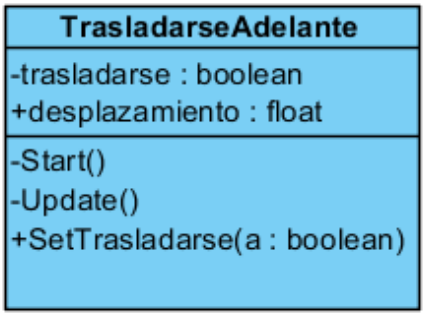


Diagrama 8: Script "TrasladarseAdelante".

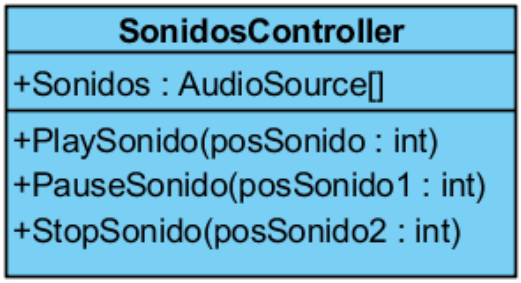


Diagrama 9: Script "SonidosController".

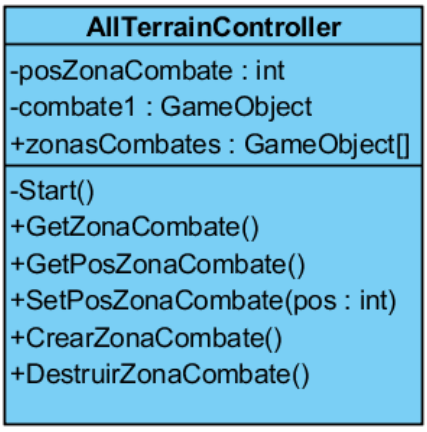


Diagrama 10: Script "AlTerrainController".

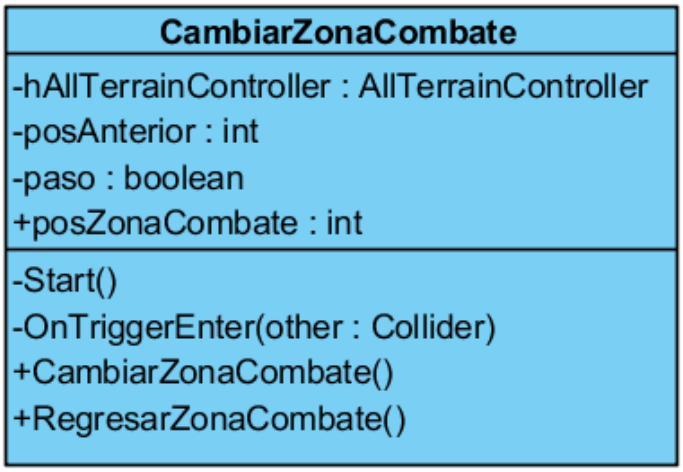


Diagrama 11: Script "CambiarZonaCombate".

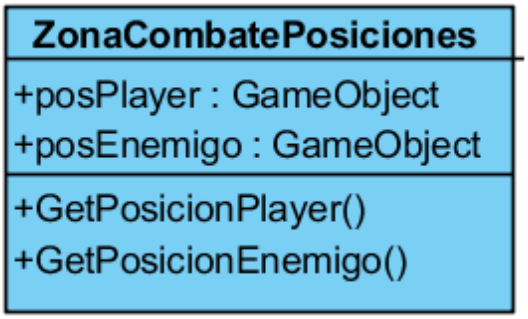


Diagrama 12: Script "ZonaCombatePosiciones".

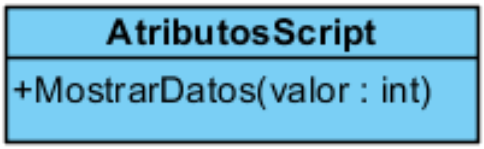


Diagrama 13: Script "AtributosScript".

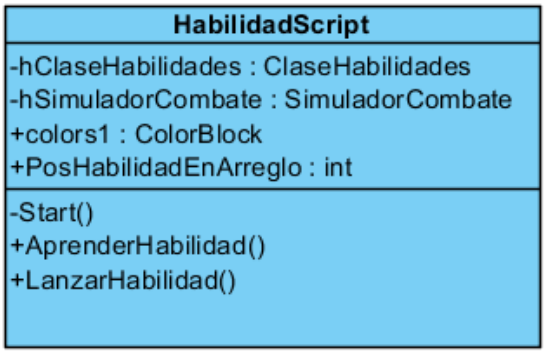


Diagrama 14: Script "HabilidadScript".

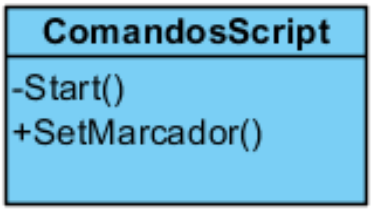


Diagrama 15: Script "ComandosScript".

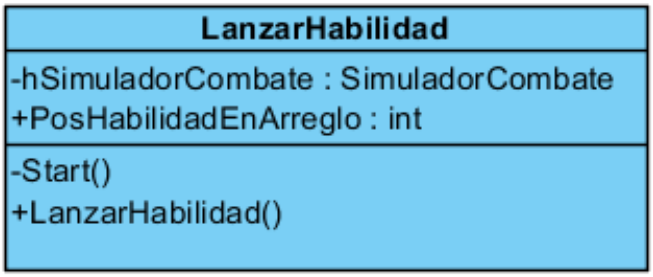


Diagrama 16: Script "LanzarHabilidad".

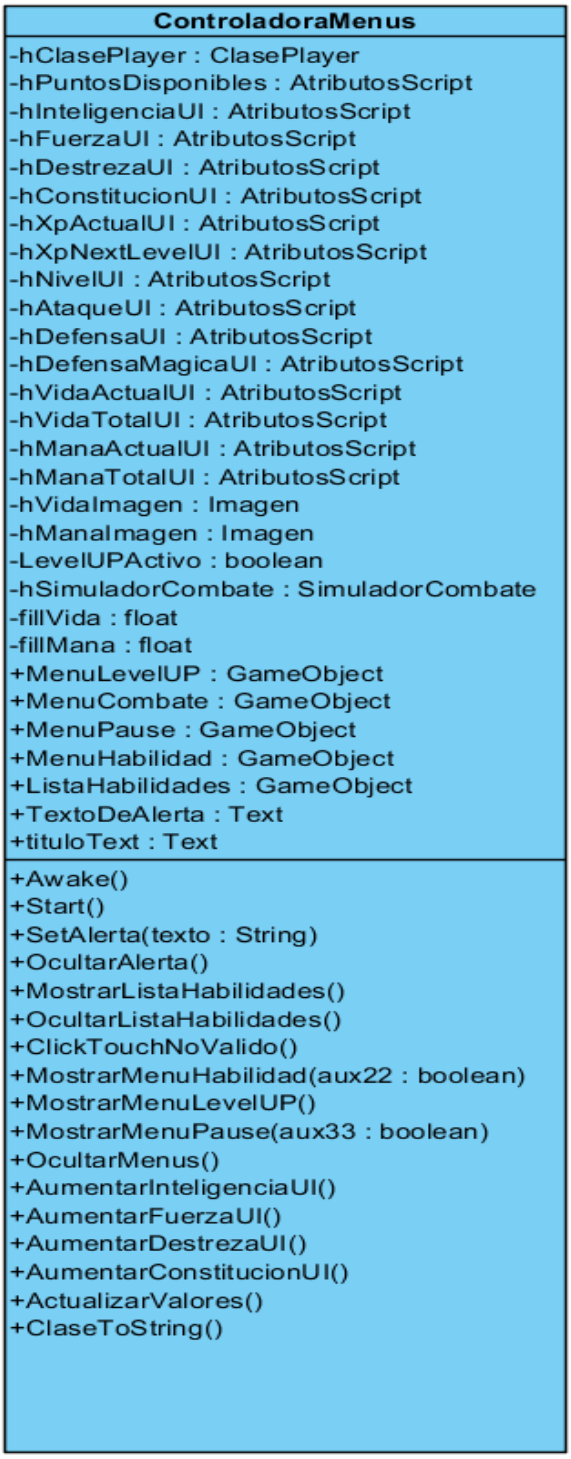


Diagrama 17: Script "ControladoraMenus".

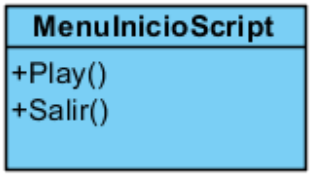


Diagrama 18: Script "MenuInicioScript".

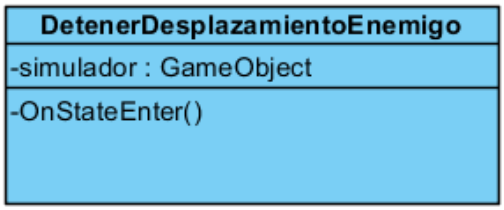


Diagrama 19: Script "DetenerDesplazamientoEnemigo".

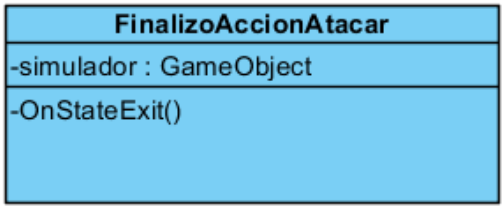


Diagrama 20: Script "FinalizoAccionAtacar".

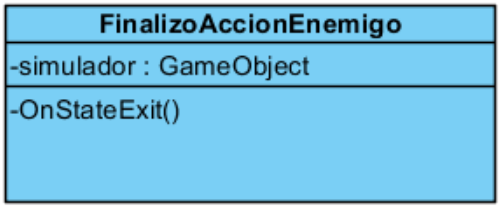


Diagrama 21: Script "FinalizoAccionEnemigo".

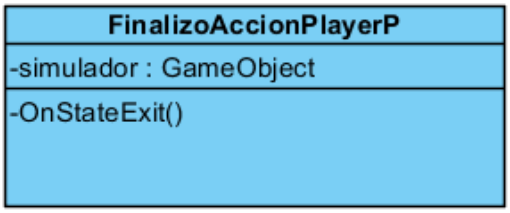


Diagrama 22: Script "FinalizoAccionPlayerP".

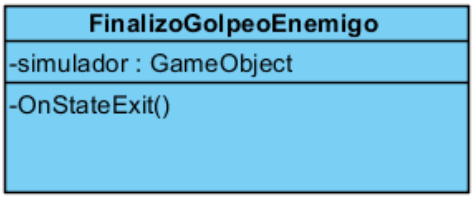


Diagrama 23: Script "FinalizoGolpeoEnemigo".

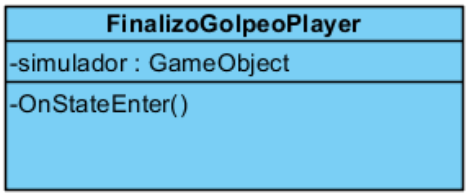


Diagrama 24: Script "FinalizoGolpeoPlayer".

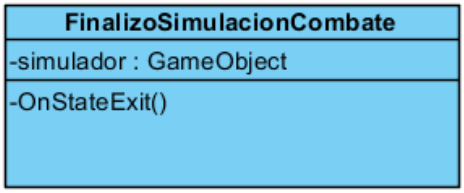


Diagrama 25: Script "FinalizoSimulacionCombate".

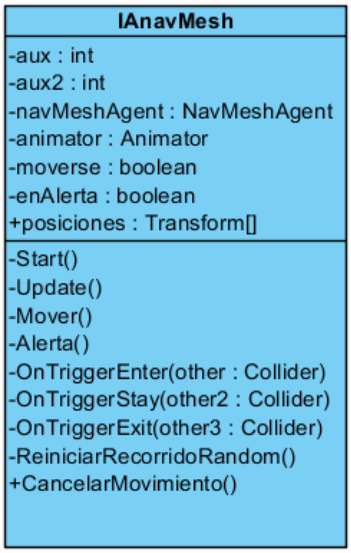


Diagrama 26: Script "lAnavMesh".