CONFLICTO ARMADO, EL VIDEOJUEGO

BIBIANA ALVARADO LEÓN
ALEJANDRO GONZÁLEZ GIRALDO

CORPORACIÓN UNIVERSITARIA MINUTO DE DIOS
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INFORMÁTICA Y ELECTRÓNICA
PROGRAMA DE TECNOLOGÍA EN INFORMÁTICA
BOGOTÁ
I-2013

CONFLICTO ARMADO, EL VIDEOJUEGO

BIBIANA ALVARADO LEÓN
ALEJANDRO GONZÁLEZ GIRALDO

Trabajo de grado Para optar al título de
Tecnólogo en informática

Asesor:
Ing. Gerardo Cañas Morales

CORPORACIÓN UNIVERSITARIA MINUTO DE DIOS
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INFORMÁTICA Y ELECTRÓNICA
PROGRAMA DE TECNOLOGÍA EN INFORMÁTICA
BOGOTÁ
I-2013

Nota de aceptación

Firma del presidente del jurado

Firma del jurado N°1

Firma del jurado N°2

Bogotá D.C. Junio de 2013

DEDICATORIA

A:

Dios, por habernos permitido llegar hasta este punto y habernos dado salud para lograr nuestros objetivos.

Nuestros maestros, Ing. Sócrates Rojas por su constante apoyo para la culminación del software de a tesis, al Ing., Gerardo Cañas por su orientación en el proceso de presentación de la misma, al Ing. Luis Pérez por su guía en el desarrollo y culminación del trabajo escrito para la sustentación de la tesis, y a todos los maestros que con su dedicación y esfuerzo aportaron a nuestra formación tecnológica y marcaron cada etapa de nuestro camino universitario brindándonos asesorías y resolviendo dudas presentadas en la elaboración de la tesis.

AGRADECIMIENTOS
BIBIANA ALVARADO

Le agradezco a:

Dios, por haberme acompañado a lo largo de mi carrera, por brindarme una vida llena de aprendizajes y experiencias.
Mi familia, por brindarme la posibilidad de estudio y por guiarme durante toda la vida.
Mis compañeros de universidad, por haber hecho de mi etapa universitaria un trayecto de vivencias que nunca olvidaré

AGRADECIMIENTOS ALEJANDRO GONZÁLEZ

Le agradezco a:
Dios, por ser mi guía espiritual en todo momento.
Mi familia, por acompañarme en el paso a paso.
Mis amigos, por encontrar en ellos la motivación para seguir adelante.

AGRADECIMIENTOS GENERALES

Agradecemos al profesor Gerardo Cañas Morales, por la orientación prestada en la Tesis. Al profesor Sócrates por su constante apoyo y sugerencias durante el desarrollo de la tesis. A la universidad Minuto de Dios que nos brindó la posibilidad de estudio contando con una excelente formación

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

RESUMEN

Conflicto Armado, EL VIDEOJUEGO

Colombia se puede identificar desde varios puntos de vista, el videojuego el conflicto armado presenta diferentes situaciones ficticias sobre los roles que desempeñan los principales actores de Colombia mediante este software se podrá interactuar y vivir nuevas experiencias desde diferentes ámbitos sociales, económicos y políticos.

ABSTRACT

Conflicto Armado, EL VIDEOJUEGO

Colombia can be identified from several viewpoints, the game Conflicto Armado, presents different fictional situations about the roles that major actors Colombia by this software can interact and enjoy new experiences from different social, economic and political.

INTRODUCCIÓN

El presente proyecto busca desarrollar un videojuego sobre Colombia donde concientice a los usuarios que lo jueguen sobre la violencia que existe en este país, donde misión a misión descubran estadísticas reales de la violencia, de los inocentes que cayeron, de los integrantes de los grupos armados que murieron, y de cómo con una problemática ficticia creada por los desarrolladores puede terminar Colombia.

Asimismo busca mostrar la implementación de la metodología de desarrollo ágil Scrum en el desarrollo de un videojuego, la cual iteración a iteración pauta el análisis, esquema, proceso, ejecución y pruebas donde se visualizará el ciclo de vida habitual del software a desarrollar.

A lo largo de este proyecto se definirá con más detalle los roles, flujos y artefactos de Scrum como metodología de desarrollo, además de definirán los escenarios, personajes, recursos y demás objetos que serán involucrados en el desarrollo del videojuego.

A lo largo de este trabajo se va a definir con mayor detalle cuales son las partes fundamentales de Scrum y qué papel juegan en cada paso las personas que integran el equipo, e implementar esta metodología para poder sacar un mayor provecho de una manera óptima, las actividades a realizar en el proceso y de qué manera tienen que ser distribuidas las mismas

- TITULO DEL PROYECTO

Conflicto Armado, El Videojuego

El nombre del proyecto se debe a que trata de un juego de estrategia para computador, el juego al estar en un estado digital se convierte en un vídeo juego, el cual es una aplicación informática para que el usuario interactúe con la máquina y pueda cumplir con el objetivo principal: Entretenerse.

Pero por qué ¿Conflicto Armado?, el juego aunque tiene como objetivo principal el entretenimiento no deja de lado la toma de conciencia en un país como Colombia que vive bajo un conflicto armado desde la década de los 60, intentando de esta forma que el jugador tome una postura y sienta el conflicto del país como un tema más real.

- PLANTEAMIENTO DEL PROBLEMA

Aunque un vídeo juego no siempre necesita tener una problemática real, y en nuestro caso muchas de las situaciones que se van a jugar tienen una problemática ficticia o de simple entretenimiento, dicho software también tratará una visión sobre el conflicto armado en Colombia.

La toma de conciencia de un conflicto de hace más de 50 años es importante para nosotros como sociedad y la mejor manera de llegarle a una sociedad joven y tecnológica es creando interés en un tema que nos concierne en este momento, esta problemática que ha venido destruyendo a nuestro país desde hace ya muchos años, llevándose con ello muchas vidas inocentes, muchos secuestrados tanto civiles como servidores públicos, si no tomamos conciencia de este conflicto no podremos tener fe de un mejor mañana.

En la actualidad no existen videojuegos que traten sobre el conflicto armado en Colombia, es por esto que incursionando en una forma tecnología de gran acogida entre los jóvenes pretendemos hacer llegar la historia, el presente y posibles futuros de una realidad que muchos han olvidado o aun peor, una realidad de violencia que la aceptan y viven con ella.

- ALCANCE Y JUSTIFICACIÓN

Diseñar un videojuego de estrategia en 2d, este se planteara por medio de RPGMaker herramienta especializada en creación de 2d; con 3 escenarios uno de aprendizaje donde el jugador aprenderá a moverse obtener recursos y crear sus tropas, los grupos armados contarán con 1 escenario en el que tendrán que poner una bomba, por otra parte el ejército contará con 1 escenario distinto La liberación de unos secuestrados. Además se obtendrán recursos de una sola manera dependiendo de la campaña que se escoja; el ejército por su parte obtendrá recursos de las granjas y los grupos armados obtendrán del cultivo de la cocaína. Asimismo cada campaña contará con tropas para fortalecer la misma y atacar a su enemigo (campesinos ingenieros y arquitectos respectivamente), igualmente estas tropas tendrán que ser creadas en su respectivo edificio; todos en el centro urbano.

EL videojuego no podrá tener multijugador, aunque se establece para futuras actualizaciones la implementación de este por medio de la tecnología Cliente–Servidor.

Ante todo el entretenimiento, pasar un par de horas frente al computador creando las mejores estrategias para ganar territorio, conseguir recursos y vencer al equipo enemigo para conseguir supremacía sobre un terreno.

En la parte social se pretende llegar a un grupo de población tecnológica que es ajeno a la problemática del conflicto armado para crear conciencia y de esta forma sentir más la bandera Colombiana en nuestros corazones.

Como ¿Crear conciencia? El videojuego se desarrolla en 2 campañas

distintas, el bien y el mal, cada campaña cuenta con 3 escenarios, pasado, presente y un posible futuro extremista (desde nuestro punto de vista), al finalizar cada escenario y al comienzo del mismo se brindara información de hechos reales (excepto en los escenarios del futuro) sobre estadísticas, terrenos, violencia, retroalimentación, tratados, avances y demás informes para la siembra de conciencia en el gamer.

- JUSTIFICACIÓN

Gracias a la tecnología podemos llegar a cualquier parte sin ni siquiera levantarnos de la silla, y es por esta razón una idea puede transmitirse a gran velocidad, masivamente y aunque se pueda transmitir la idea en muchas ocasiones no llega porque el tema no es de interés para el receptor, es por esta razón que la creación de un videojuego llega a quien realmente necesita hacerlo, llegarle a la población joven.

La justificación del conflicto armado en Colombia durante la historia ha sido un problema de los peores sucesos que han sucedido en nuestra nación, tenemos que saber y tomar conciencia de que todos estamos involucrados y tenemos que saber en qué nos está perjudicando este conflicto. Es este el momento en el cual debemos llegar a las personas y mostrar y/o enseñarles que problema nos traerá en el futuro no muy lejano si no hacemos nada, teniendo muy claro los antecedentes históricos o por lo contrario nos motivamos a crear ideas nuevas para cambiar la historia de Colombia.

Pero la mejor justificación sobre este tema tal vez sea crear un vídeo juego para entretenerse. (Y tal vez -tristemente- muchas personas solo lo usen para este fin).

- OBJETIVOS

- Objetivo General

Diseñar un videojuego de estrategia para la problemática del territorio Colombiano, por medio de recursos, ejércitos y tácticas, donde el usuario se entretenga hasta la finalización del mismo.

- Objetivos Específicos

    - Diseñar una interfaz agradable para la entretención del usuario.

    - Generar estadísticas en cada escenario donde el usuario sentará conciencia del conflicto Armado que existe en Colombia.

    - Crear IA (Inteligencia Artificial) en el enemigo para que este realice las acciones pertinentes y dificulte al usuario completar las misiones.

- RPG MAKER VX

RPGMaker es un programa que da la facilidad de crear un propio juego estilo RPG sin la necesidad de saber programar demasiado, fue creado por la empresa japonesa ASCII parte de la corporación ENTERBRAIN.

En el RPGMaker básicamente te da la opción fácil e intuitiva de crear un propio juego 2D orientado al estilo RPG, se puede seleccionar la música, poner mensajes entre otros comandos que dan una amplia gama de posibilidades, cuando se abre el RPGMaker también se encuentra un editor de mapas, una base de datos y un editor de eventos, además de un editor de scripts ofreciendo capacidades avanzadas de programación. En todas las versiones del RPGMaker también es necesario descargar el RTP (Run time Package) este incluye todos los materiales básicos para crear nuestro juego como música, sonidos, personajes, escenarios y gráficos de mapas.

Este programa es un editor de juegos RPG, juegos de rol. Con este

programa se puede diseñar todo desde cero a través de sencillos menús:

- Protagonistas: Permite crear los personajes del juego con sus propias imágenes, historia, estadísticas y demás.
- Base de datos: Permite manejar toda la base de datos de equipo, inventario, hechizos, habilidades entre otras.
- Mapas: Permite crearlos desde cero, a través de un sencillo sistema de casillas. Además puede crear sus propias plantillas o descargar otras de la red para usar las que más gusten.
- Variables: Permite controlar todas las variables de los combates, enemigos, comportamiento, aparición entre otras.
- Historia: Permite crear todo lo referente a la historia, es decir diálogos, videos, música, sonidos, secuencias manuales y automáticas, Además de esto permite crear la personalización de la introducción y de los diferentes menús que se deseen implementar en el videojuego.

El RPGMaker Vx maneja en su editor de scripts el lenguaje de programación Ruby el cual es un lenguaje de programación dinámico y de código abierto enfocado en la simplicidad y productividad. Su elegante sintaxis se siente natural al leerla y fácil al escribirla. Ruby es orientado a objetos: todos los tipos de datos son un objeto, incluidas las clases y tipos que otros lenguajes definen como primitivas, (enteros, booleanos, y "nil"). Toda función es un método, las variables siempre son referencias a objetos, no los objetos mismos. Ruby soporta herencia con enlace dinámico, mixins y métodos singleton (pertenecientes y definidos por una sola instancia más que definidos por la clase). A pesar de que Ruby no soporta herencia múltiple, las clases pueden importar módulos como mixins. La sintaxis procedural está soportada, pero todos los métodos definidos fuera del ámbito de un objeto son realmente métodos de la clase Object. Como esta clase es padre de todas las demás, los cambios son visibles para todas las clases y objetos.

- **INGENIERÍA DEL PROYECTO**

La ingeniería del proyecto pretende definir cada una de las fases en las que los desarrolladores pueden dividir el proyecto, el conjunto de las fases definidas se conoce como ciclo de vida del software o proyecto a desarrollar. A continuación se realiza una introducción de la metodología escogida para desarrollar el proyecto.

- MARCO TEÓRICO DEL MODELO DE DESARROLLO

La metodología de desarrollo que se utilizará en el desarrollo de este proyecto será SCRUM esta se basa en construir primero la funcionalidad de mayor valor para el cliente y en los principios de inspección continua, adaptación, auto-gestión e innovación

Con la metodología Scrum el cliente se entusiasma y se compromete con el proyecto dado que lo ve crecer iteración a iteración. Así mismo le permite en cualquier momento realinear el software con los objetivos de negocio de su empresa, ya que puede introducir cambios funcionales o de prioridad en el inicio de cada nueva iteración sin ningún problema.

Tanto Scrum como Programación Extrema (XP) requieren que los equipos completen algún tipo de producto potencialmente liberable al final de cada iteración. Estas iteraciones están diseñadas para ser cortas y de duración

fija.

Este enfoque en entregar código funcional cada poco tiempo significa que Scrum no tiene tiempo para teorías. No persigue dibujar el modelo UML perfecto en una herramienta CASE, escribir el documento de requisitos perfecto o escribir código que se adapte a todos los cambios futuros imaginables. En vez de eso, Scrum se enfoca en que las cosas se hagan. Esta metodología puede que se equivoque por el camino, pero también es muy consciente de que la mejor manera de encontrar dichos errores es dejar de pensar en el software a un nivel teórico de análisis y diseño y sumergirse en él, ensuciarse las manos y comenzar a construir el producto.

- ¿Por qué una metodología ágil y no una tradicional?

Porque las metodologías ágiles dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque muestra la efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

A continuación se muestra una tabla de diferencias entre las metodologías ágiles y las metodologías tradicionales

Metodologías Tradicionales	**Metodologías Ágiles**
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo	Basadas en heurísticas provenientes de prácticas de producción de código
Cierta resistencia a los cambios	Especialmente preparados para cambios durante el proyecto
Impuestas externamente	Impuestas internamente (por el equipo)
Proceso mucho más controlado, con numerosas políticas/normas	Proceso menos controlado, con pocos principios.
El cliente interactúa con el equipo de desarrollo mediante reuniones	El cliente es parte del equipo de desarrollo

Más artefactos	Pocos artefactos
Más roles	Pocos roles
Grupos grandes y posiblemente distribuidos	Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio
La arquitectura del software es esencial y se expresa mediante modelos	Menos énfasis en la arquitectura del software
Existe un contrato prefijado	No existe contrato tradicional o al menos es bastante flexible

Tabla N°1: Diferencias metodología ágil y tradicion al

A continuación se muestra la tabla de las diferencias por etapas y enfoque metodológico de los modelos de desarrollo:

MODELOS TRADICIONALES	**ETAPA**	**MODELOS AGILES**
Planificación predictiva y “aislada”	Análisis de requerimientos	Planificación adaptativa: Entregas frecuentes + colaboración del cliente
	Planificación	
Diseño flexible y Extensible + modelos + Documentación exhaustiva	Diseño	Diseño Simple: Documentación Mínima + Focalizado en la comunicación
Desarrollo individual con Roles y responsabilidades estrictas	Codificación	Transferencia de conocimiento: Programación en pares + conocimiento colectivo
Actividades de control: Orientado a los hitos + Gestión mini proyectos	Pruebas	Liderazgo-Colaboración: empoderamiento +auto-organización
	Puesta en Producción	

Tabla N°2: Diferencias por etapas y enfoque metodol ógico

Asimismo se muestra la tabla de diferencias de algunas metodologías ágiles de acuerdo a las características del proyecto:

Modelo de Proceso	Tamaño del Proceso	Tamaño del Equipo	Complejidad del Problema
RUP	Medio / Extenso	Medio / Extenso	Medio / Alto
XP	Pequeño / Medio	Pequeño	Medio / Alto
SCRUM	Pequeño / Medio	Pequeño	Medio / Alto

Tabla N°3: Diferencias por características del proyecto

Cabe resaltar que con un equipo pequeño de desarrollo se pueden realizar grandes proyectos, de alta complejidad para el caso de XP y Scrum.

Esta metodología, tiene ventajas como:

- Entrega de un producto funcional al finalizar cada Sprint.
- Visualización del proyecto día a día.
- Equipos integrados y comprometidos con el proyecto,
- Resultados anticipados.
- Reducción sistemática de los riesgos del proyecto.
- Productividad y calidad.
- Es apto para proyectos pequeños y de rápido movimiento

Igualmente tiene algunas desventajas, sin embargo para este proyecto no son muy relevantes.

- Si los integrantes del equipo no están centrados, el proyecto nunca se completará o incluso fallará.
- Si alguno de los integrantes del equipo se marcha durante el desarrollo puede tener un efecto negativo enorme en el desarrollo del proyecto.

- El control de la calidad del proyecto es difícil de implementar y cuantificar a menos que el equipo de test puedan llevar a cabo testeo de regresión después de cada sprint.

- Roles

En el modelo de desarrollo Scrum solo hay 3 roles *Product Owner* (dueño del producto), *the Team (*el equipo*)*, y el *ScrumMaster (*Maestro Scrum). Todas las responsabilidades de manejo de un proyecto se dividen entre estos tres papeles que se definen a continuación:

Product Owner: Es el encargado de preocuparse por los intereses de cada integrante del grupo, así como estima la parte financiera durante todo el proyecto, este es responsable de ver que los Product Backlog se estén cumpliendo a cabalidad.

The Team: Son los responsables de hacer que los requerimientos sean un hecho y cumplirlos en su totalidad.

Scrum Master: Es el que debe enseñar la metodología a los integrantes del equipo, y velar porque se cumpla a cabalidad.

- Product Backlog

También llamado lista de requerimientos es un documento que contiene descripciones detalladas de todos los requerimientos y funcionalidades. El Product Owner es el responsable de usar este documento para asegurar que todas las funcionalidades sean implementadas y producidas de manera correcta.

- Reuniones

Son de carácter vital ya que sin estas reuniones no se puede desarrollar el software a cabalidad hay 5 tipos de reuniones en Scrum y se especifican a continuación:

Primera Reunión: Se realiza una reunión de planificación de la iteración y esta se divide en 2 partes.

- Selección de Requisitos: El cliente presenta al equipo la lista de requisitos priorizada del proyecto, esta tiene una duración máxima de 4 horas.

- Planificación del Sprint: El equipo elabora la lista de tareas de la iteración necesarias para desarrollar los requisitos a los que se ha comprometido, esta tiene una duración máxima de 4 horas.

El Sprint es el período en el cual se lleva a cabo el trabajo. Es recomendable que la duración de los Sprint sea de una manera constante. Al final de cada uno de estos, el equipo deberá presentar un producto servible y probado para que el cliente de su punto de vista y posibles sugerencias para el mejoramiento del mismo. Durante el sprint, nadie puede cambiar el Sprint Backlog, lo que significa que los requisitos están congelados durante el sprint.

Reunión diaria Scrum: Se reúnen el equipo Scrum diariamente durante 15 minutos como máximo, se dialoga sobre el trabajo que debe realizar cada integrante y se informa de los problemas al Scrum Master.

Planificación del Sprint: El Product Owner, y the Team, se reúnen antes de cada Sprint, para ver la funcionalidad del producto que se trabajará, el Product Owner, presenta el Product Backlog, para que el equipo pueda verlo, y decidir que avance se puede hacer durante el Sprint, esta tiene una duración máxima de 4 horas.

Durante esta reunión, el Product Owner identifica los elementos del Product Backlog que quiere ver finalizados y los informa al equipo. En aquel

momento, el equipo determina la cantidad de ese trabajo que puede comprometerse a finalizar durante el siguiente sprint.

Retrospectiva del Sprint: Se reúne todo el equipo y discuten acerca del Sprint que se terminó, además planean que hacer para que el siguiente Sprint sea más productivo, lo que se tiene en cuenta en esta reunión es la facilidad de comunicación entre todos, y la oportunidad que se les da de opinar acerca del tema.

Revisión del Sprint: Se realiza una revisión del proyecto al finalizar el Sprint. Luego de la fecha de finalización del Sprint, el equipo presenta el incremento del producto que pudo construir, de acuerdo a esta revisión se decide que hacer posteriormente.

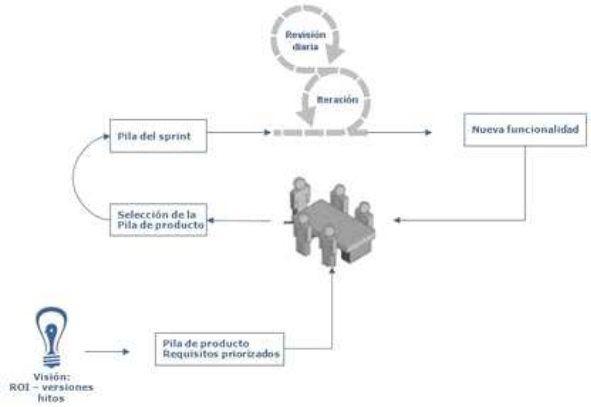


Figura N°1: Diagrama de flujo del modelo Scrum

- Planeación de Sprint

En el desarrollo de la metodología se indica que deben ser creados los Sprint donde se muestra el tiempo y las tareas que se realizarán en cada uno de estos.

Para el presente proyecto se implementaron los siguientes Sprint:

Sprint 1	
Título: Maker	**Fecha de Realización:**
Objetivo: Escoger el maker que se implementará en el desarrollo del videojuego	
Criterios de Aceptación: El maker debe permitir adjuntar una galería completa (imágenes, videos, música, sonidos, scripts) para la personificación del videojuego.	
Tareas a Realizar: • Seleccionar el mejor maker para desarrollar el videojuego • Verificar que permita adjuntar imágenes • Verificar que permita adjuntar personajes • Comprobar que permita crear escenarios grandes	

Tabla N°4: Sprint 1

Sprint 2	
Título: Personajes	**Fecha de Realización:**
Objetivo: Escoger los personajes que van a representar cada una de las campañas del videojuego.	
Criterios de Aceptación: Los personajes deben hacer parte del conflicto armado colombiano real.	
Tareas a Realizar: • Seleccionar los personajes principales de cada campaña • Seleccionar los personajes secundarios de los escenarios	

Tabla N°5: Sprint 2

Sprint 3	
Título: Imágenes	**Fecha de Realización:**
Objetivo: Crear las imágenes que se pondrán para cada personaje en los diálogos	
Criterios de Aceptación: Las imágenes tienen que tener un tamaño de 96x96 px con fondo transparente para que estás se vean en los diálogos correctamente.	
Tareas a Realizar: • Definición de la posición de las imágenes que se pondrán en los diálogos. • Búsqueda de las imágenes. (Cada una con su bibliografía correspondiente). • Probar en el escenario los diálogos con las imágenes implementadas.	

Tabla N°6: Sprint 3

Sprint 4	
Título: Charas	**Fecha de Realización:**
Objetivo: Realizar la personificación de los personajes de cada escenario	
Criterios de Aceptación: Los charas tienen que tener un tamaño de 96x128px con fondo transparente. Cada chara debe tener 12 imágenes ubicadas de la manera 3x4	
Tareas a Realizar: • Buscar un programa que permita la personificación de los charas. • Averiguar un programa que permita la unión de todos los charas en uno solo para que el maker los reconozca. • Probar en el escenario todos los charas implementados	

Tabla N°7: Sprint 4

Sprint 5	
Título: Diseño Campaña Aprendizaje	**Fecha de Realización:**
Objetivo: Realizar el prototipo de la campaña de aprendizaje del videojuego	
Criterios de Aceptación: Debe contener todas las acciones que se implementarán en las otras campañas.	
Tareas a Realizar: • Realizar el diseño de la física del escenario • Definir los movimientos del jugador. • Definir la forma de manejar dos personajes • Definir la forma de mostrar el mapa en la pantalla de juego • Definir la forma de pelear del jugador. • Definir la forma de obtención de los recursos. • Realizar decisiones de toma de conciencia. • Realizar el diagrama de flujo del escenario	

Tabla N°8: Sprint 5

Sprint 6	
Título: Diseño Campaña del Bien	**Fecha de Realización:**
Objetivo: Realizar el prototipo de la campaña del bien del videojuego	
Criterios de Aceptación: Debe contener una misión y los personajes adecuados para esa.	
Tareas a Realizar: • Realizar el diseño de la física del escenario • Definir la misión a realizar en esta campaña • Realizar el diagrama de flujo del escenario	

Tabla N°9: Sprint 6

Sprint 7	
Título: Diseño Campaña del Mal	**Fecha de Realización:**
Objetivo: Realizar el prototipo de la campaña del mal del videojuego	
Criterios de Aceptación: Debe contener una misión y los personajes adecuados para esa.	
Tareas a Realizar: • Realizar el diseño de la física del escenario • Definir la misión a realizar en esta campaña • Realizar el diagrama de flujo del escenario	

Tabla N°10: Sprint 7

Sprint 8	
Título: Diseño de Diagramas de Casos de Uso	**Fecha de Realización:**

Objetivo: Diseñar los diagramas de casos de uso del videojuego
Criterios de Aceptación: Los Diagramas deben contener las especificaciones correspondientes de cada uno.
Tareas a Realizar: • Diseñar los diagramas de casos de uso del videojuego • Realizar las especificaciones de cada diagrama de caso de uso

Tabla N°11: Sprint 8

Sprint 9	
Título: Diseño de Diagramas de Estructura Estática	**Fecha de Realización:**
Objetivo: Diseñar los diagramas correspondientes al modelado de la estructura estática del videojuego	
Criterios de Aceptación: Los Diagramas deben contener las especificaciones correspondientes de cada uno.	
Tareas a Realizar: • Diseñar los diagramas de clase del videojuego • Realizar las especificaciones de cada diagrama de clases	

Tabla N°12: Sprint 9

Sprint 10	
Título: Diseño de Diagramas de Interacción del sistema	**Fecha de Realización:**
Objetivo: Diseñar los diagramas correspondientes al modelado de la interacción del sistema del videojuego	
Criterios de Aceptación: Los Diagramas deben contener las especificaciones correspondientes de cada uno.	
Tareas a Realizar: • Diseñar los diagramas colaboración del videojuego • Realizar las especificaciones de cada diagrama de colaboración • Diseñar los diagramas secuencia del videojuego • Realizar las especificaciones de cada diagrama de secuencia	

Tabla N°13: Sprint 10

Sprint 11	
Título: Diseño de Diagramas Dinámicos	**Fecha de Realización:**
Objetivo: Diseñar los diagramas correspondientes al modelado dinámico (Objetos) del videojuego	
Criterios de Aceptación: Los Diagramas deben contener las especificaciones correspondientes de cada uno.	
Tareas a Realizar: • Diseñar los diagramas estados del videojuego • Realizar las especificaciones de cada diagrama de estados • Diseñar los diagramas actividades del videojuego • Realizar las especificaciones de cada diagrama de actividades	

Tabla N°14: Sprint 11

Sprint 12	
Título: Diseño de Diagramas de Implementación	**Fecha de Realización:**
Objetivo: Diseñar los diagramas correspondientes al modelado de implementación del videojuego	
Criterios de Aceptación: Los Diagramas deben contener las especificaciones correspondientes de cada uno.	
Tareas a Realizar: • Diseñar los diagramas componentes del videojuego • Realizar las especificaciones de cada diagrama de componentes • Diseñar los diagramas distribución del videojuego • Realizar las especificaciones de cada diagrama de distribución	

Tabla N°15: Sprint 12

Sprint 13	
Título: Comprobación Final	**Fecha de Realización:**
Objetivo: Comprobar todo el documento y todo el funcionamiento del juego.	
Criterios de Aceptación: El videojuego en general debe funcionar correctamente, así como el documento en general debe estar aprobado por el tutor.	
Tareas a Realizar: • Comprobación y erradicación de los posibles errores. • Hacer pruebas para visualizar el correcto funcionamiento del videojuego	

Tabla N°16: Sprint 13

- MODELO DE DESARROLLO

El modelo de desarrollo a utilizar en el presente proyecto es el método ágil SCRUM, ya que sus particularidades permiten una implementación de procesos de desarrollo que permiten entregables en periodos cortos de tiempo, esto es permisible ya que SCRUM permite la adaptabilidad de cambios para así aumentar las posibilidades de éxito en el proyecto.

- CICLO DE VIDA

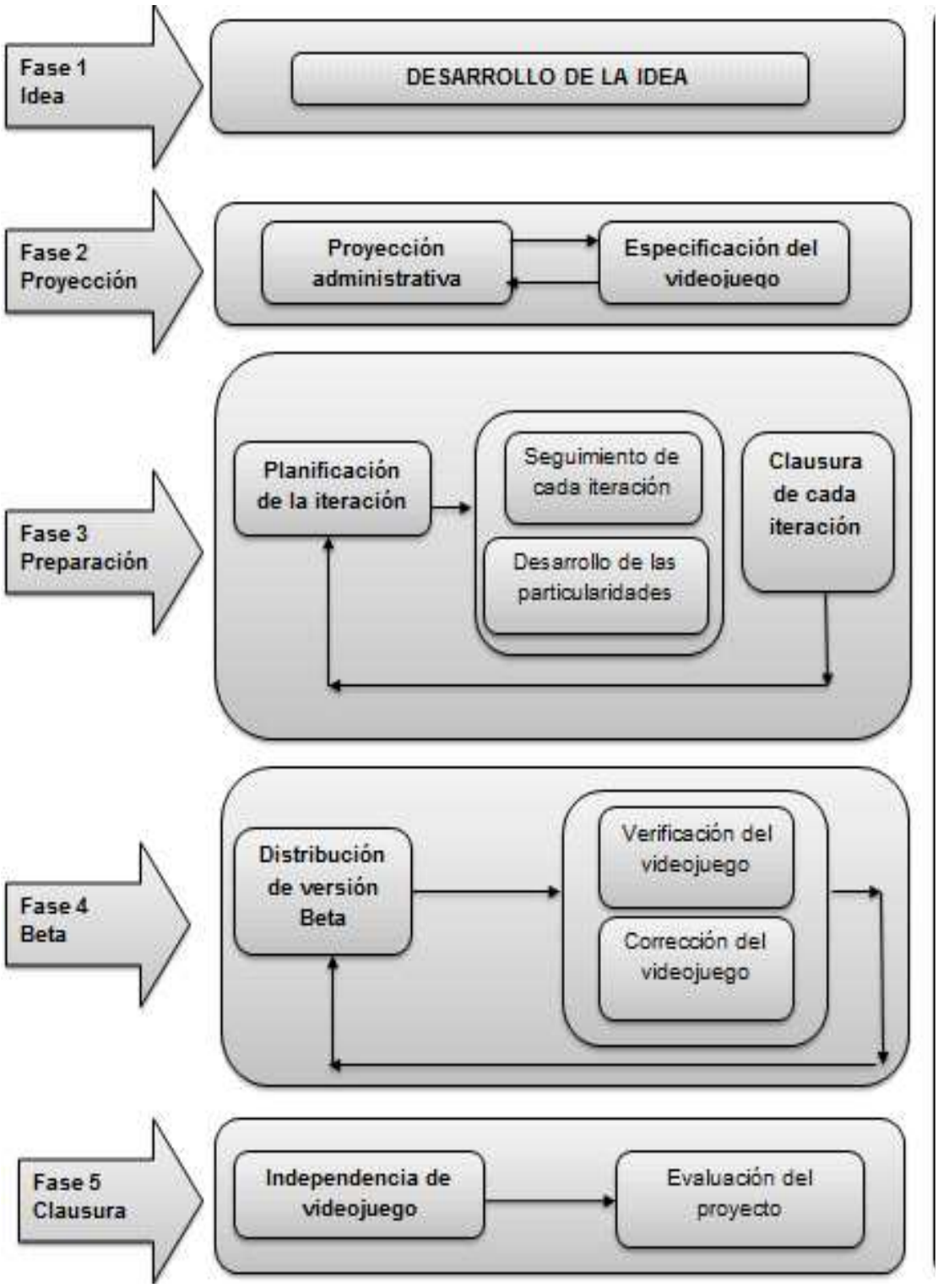

Fase 1
Idea
DESARROLLO DE LA IDEA
Fase 2
Proyección
Proyección administrativa
Especificación del videojuego
Fase 3
Preparación
Planificación de la iteración
Seguimiento de cada iteración
Desarrollo de las particularidades
Clausura de cada iteración
Fase 4
Beta
Distribución de versión Beta
Verificación del videojuego
Corrección del videojuego
Fase 5
Clausura
Independencia de videojuego
Evaluación del proyecto

Figura N°2: Diagrama de Ciclo de Vida SCRUM

Durante el ciclo de vida del modelo de desarrollo se tramitan los riesgos para minimizar el impacto de dificultades. Ya que en cualquiera de las fases se pueden presentar riesgos por uno o varios motivos de programación, diseño o desarrollo. El ciclo de vida del modelo de desarrollo utilizado para diseñar y desarrollar el videojuego se describirá a continuación:

En la fase 1 (Idea) se tiene como objetivo principal definir la idea del videojuego, los elementos correspondientes al mismo: definir los personajes involucrados en este, la caracterización de cada personaje, la historia que se manejara en el transcurso del mismo, las campañas que se utilizarán y los niveles de cada una, además de esto se definirá los elementos técnicos a utilizar en el mismo (Lenguaje de programación, game maker entre otras herramientas para el diseño del mismo).

En la fase (2) Proyección se tiene como objetivo planificar el cronograma para realizar las siguientes fases del proyecto (características técnicas y administrativas), dividir las tareas que el equipo de trabajo debe cumplir esto consiste en describir y priorizar los particularidades funcionales y no funcionales del proyecto. Asi mismo se debe hacer uso de UML para establecer la interacción del usuario con el sistema propuesto, esta interacción se puede visualizar por medio de los diagramas de estructura estática (Diagrama de clases), de interacción del sistema (Colaboración, secuencia, componentes, distribución) y los diagramas de modelado dinámico (Casos de uso, Estado, Actividad).

En la fase (3) Preparación se tiene como objetivo la implementación del videojuego, lograr una versión ejecutable de este al analizar cada una de las interacciones cumpliendo con los requerimientos funcionales a cabalidad ya que de estos derivan las particularidades a implementar en cada una de las tareas programadas en el equipo de trabajo.

En la fase (4) Beta se tiene como objetivo evaluar y ajustar disímiles

aspectos del videojuego, además se verifica que el gameplay (Forma como los jugadores interactúan con el videojuego) tenga la conexión que se quiere con cada jugador, ya que en esta fase se maneja una versión beta del videojuego permite detectar y eliminar los errores que se producen a medida que el videojuego avanza. Al da por terminada esta fase se tendrá el producto final del videojuego que se entregará en la última fase del ciclo de vida del modelo de desarrollo

En la fase (5) Clausura se evalúa el proyecto de acuerdo a los requerimientos funcionales y no funcionales que se plantearon desde un comienzo, además de esto se evalúan los errores, problemas, éxitos, soluciones presentados en el desarrollo del mismo; así como se verifica el cumplimiento total de los objetivos del desarrollo de este proyecto.

- **ANÁLISIS Y DISEÑO**

Es importante tener en cuenta que la usabilidad de un sistema no sólo se encuentra ligada a la apariencia del software para el usuario sino principalmente al modo en el que el usuario puede utilizar este, es decir la interacción con el mismo, por tanto está relacionada con la estructura general del sistema y con la lógica del negocio por lo tanto a continuación se definirán los requerimientos funcionales y no funcionales del sistema y posteriormente el diseño del mismo.

- DEFINICIÓN DE REQUERIMIENTOS

Teniendo en cuenta los alcances de este proyecto, se plantean los siguientes requerimientos, los cuales son necesarios para lograr el funcionamiento ideal de la aplicación.

- Requerimientos Funcionales

  - El videojuego debe permitir la selección del bando a manejar.
  - El videojuego deberá permitir obtener recursos de diferentes formas.
  - El videojuego permitirá la construcción de diferentes edificaciones y esta se basara en las tecnologías adquiridas por campaña.
  - El videojuego debe otorgar un ganador y un perdedor.
  - El videojuego debe tener reglas claras para su aplicación.
  - El videojuego deberá tener distintas misiones.
  - El videojuego debe brindar varios tipos de escenarios para su desarrollo.
  - El videojuego debe poder almacenar y cargar partidas.

- El videojuego debe permitir al usuario crear sus propias estrategias sociales y militares.
- El videojuego debe permitir escoger el sexo del jugador y así mismo cambiar la apariencia en el juego.
- El videojuego debe permitir seleccionar una dificultad.
- El videojuego debe manejarse con el teclado o con un control de juegos.
- El videojuego debe contar con un mapa del sitio de la misión.
- El videojuego debe permitir cargar varios elementos para atacar.
- El videojuego debe permitir ingresar el nombre del jugador.
- El videojuego debe permitir cerrarse mediante una opción de salir.

- Requerimientos No Funcionales

Teniendo en cuenta los requerimientos funcionales del proyecto se necesitan los siguientes requerimientos no funcionales para lograr que el proyecto funcione:

- El videojuego debe accederse desde un computador Windows.
- El videojuego estará basado en hechos históricos documentados.
- El videojuego debe mostrar el estado del jugador en cada partida.
- El videojuego debe guardar automáticamente para evitar pérdida de información.
- El videojuego debe manejar una interfaz agradable para el usuario.
- El videojuego debe manejar un juego intuitivo.
- El videojuego debe manejar charas e imágenes de acorde a las

misiones expuestas.

- El computador debe tener 2Gb de RAM para correr a una velocidad óptima el videojuego.
- El computador debe contar con 1Gb de Disco Duro para correr el videojuego.
- El videojuego debe tener sonido de acorde a los escenarios.
- El videojuego debe mostrar el contenido del texto de manera legible y acorde a los escenarios.
- El computador debe contar con una tarjeta de video de 512Mb para correr el videojuego óptimamente.
- El formato de visualización debe ser robusto y consistente para todas las funciones que se implementen, se debe realizar un diseño del videojuego fácil de administrar y mantener
- El videojuego debe permitir visualizar al usuario los recursos que obtiene en pantalla.

- Matriz de Requerimientos

MATRIZ DE REQUERIMIENTOS						
Aclaración: Los requerimientos que están en color rojo, son requerimientos que no se alcanzaran a implementar						
Identificador	**Descripción**	**Fuente**	**Prioridad**	**Tipo**	**Estado**	**Usuarios Involucrados**
REQUERIMIENTOS FUNCIONALES						
Identificador 1	El videojuego debe permitir la selección del bando a manejar.	Programador	Alto	Funcional	Desarrollado	Programador
Identificador 2	El videojuego deberá permitir obtener recursos de diferentes formas.	Programador	Medio	Funcional	Desarrollado	Programador
Identificador 3	El videojuego permitirá la construcción de diferentes edificaciones y esta se basara en las tecnologías adquiridas por campaña	Programador	Medio	Funcional	Desarrollado	Diseñador

Identificador 4	El videojuego debe otorgar un ganador y un perdedor.	Programador	Alto	Funcional	Desarrollado	Programador
Identificador 5	El videojuego debe tener reglas claras para su aplicación.	Programador	Alto	Funcional	Desarrollado	Programador
Identificador 6	El videojuego deberá tener distintas misiones.	Programador	Alto	Funcional	Desarrollado	Programador
Identificador 7	El videojuego debe brindar varios tipos de escenarios para su desarrollo.	Programador	Alto	Funcional	Desarrollado	Diseñador
Identificador 8	El videojuego debe poder almacenar y cargar partidas.	Programador	Alto	Funcional	Desarrollado	Programador
Identificador 9	El videojuego debe permitir al usuario crear sus propias estrategias sociales y militares.	Programador	Medio	Funcional	Desarrollado	Usuario
Identificador 10	El videojuego debe permitir escoger el sexo del jugador y así mismo cambiar la apariencia en el juego.	Programador	Bajo	Funcional	Desarrollado	Programador
Identificador 11	El videojuego debe permitir seleccionar una dificultad.	Programador	Medio	Funcional	Desarrollado	Programador
Identificador 12	El videojuego debe manejarse con el teclado o con un control de juegos.	Programador	Alto	Funcional	Desarrollado	Programador
Identificador 13	El videojuego debe contar con un mapa del sitio de la misión.	Programador	Medio	Funcional	Desarrollado	Programador
Identificador 14	El videojuego debe permitir cargar varios elementos para atacar.	Programador	Medio	Funcional	Desarrollado	Programador
Identificador 15	El videojuego debe permitir ingresar el nombre del jugador.	Programador	Bajo	Funcional	Desarrollado	Programador
Identificador 16	El videojuego debe permitir cerrarse mediante una opción de salir.	Programador	Alto	Funcional	Desarrollado	Programador

Tabla N°17: Matriz de Requerimientos Funcionales

MATRIZ DE REQUERIMIENTOS						
Aclaración: Los requerimientos que están en color rojo, son requerimientos que no se alcanzaran a implementar						
Identificador	**Descripción**	**Fuente**	**Prioridad**	**Tipo**	**Estado**	**Usuarios Involucrados**
REQUERIMIENTOS NO FUNCIONALES						
Identificador 1	El videojuego debe accederse desde un computador Windows.	Programador	Alto	No Funcional	Desarrollado	Programador
Identificador 2	El videojuego estará basado en hechos históricos documentados	Programador	Alto	No Funcional	Desarrollado	Programador
Identificador 3	El videojuego debe mostrar el estado del jugador en cada partida.	Programador	Alto	No Funcional	Desarrollado	Programador

Identificador 4	El videojuego debe guardar automáticamente para evitar pérdida de información.	Programador	Alto	No Funcional	Desarrollado	Programador
Identificador 5	El videojuego debe manejar una interfaz agradable para el usuario.	Programador	Alto	No Funcional	Desarrollado	Diseñador
Identificador 6	El videojuego debe manejar un juego intuitivo.	Programador	Alto	No Funcional	Desarrollado	Programador
Identificador 7	El videojuego debe manejar charas e imágenes de acorde a las misiones expuestas.	Programador	Alto	No Funcional	Desarrollado	Diseñador
Identificador 8	El computador debe tener 2Gb de RAM para correr a una velocidad óptima el videojuego.	Programador	Alto	No Funcional	Desarrollado	Programador
Identificador 9	El computador debe contar con 1Gb de Disco Duro para correr el videojuego.	Programador	Alto	No Funcional	Desarrollado	Programador
Identificador 10	El videojuego debe tener sonido de acorde a los escenarios.	Programador	Alto	No Funcional	Desarrollado	Programador
Identificador 11	El videojuego debe mostrar el contenido del texto de manera legible y acorde a los escenarios.	Programador	Alto	No Funcional	Desarrollado	Diseñador
Identificador 12	El computador debe contar con una tarjeta de video de 512Mb para correr el videojuego óptimamente.	Programador	Alto	No Funcional	Desarrollado	Programador
Identificador 13	El formato de visualización debe ser robusto y consistente para todas las funciones que se implementen, se debe realizar un diseño del videojuego fácil de administrar y mantener	Programador	Alto	No Funcional	Desarrollado	Programador
Identificador 14	El videojuego debe permitir visualizar al usuario los recursos que obtiene en pantalla.	Programador	Alto	No Funcional	Desarrollado	Diseñador

Tabla N°18: Matriz de Requerimientos No Funcionales

- DESCRIPCIÓN DEL SISTEMA

El sistema se compone del realizamiento de un videojuego del conflicto armado de Colombia, es una aplicación que le permite al usuario conocer estadísticas del conflicto armado en el país así como la interacción con un sistema común (tecnología).

La toma de conciencia de un conflicto de hace más de 50 años es importante para nosotros como sociedad y la mejor manera de llegarle a una sociedad joven y tecnológica es creando interés en un tema que nos concierne en este momento, esta problemática que ha venido destruyendo a nuestro país

desde hace ya muchos años, llevándose con ello muchas vidas inocentes, muchos secuestrados tanto civiles como servidores públicos, si no tomamos conciencia de este conflicto no podremos tener fe de un mejor mañana.

El sistema está compuesto por 2 campañas, con 1 nivel cada una donde hará parte del ejército o del grupo armado completando al transcurso del mismo las misiones y obteniendo estadísticas en cada una.

El videojuego contará con interfaces de consultas, salidas y captura de datos para permitir la interacción correcta entre el usuario y el sistema.

3.2.1 Interfaz de Usuario

El videojuego Conflicto Armado, El Videojuego se llevo a cabo mediante los lineamientos del programa RPG Maker y en este se muestran las diferentes interfaces para elaborar los tipos de consultas, salidas y captura de datos.

- Interfaz de Captura de Datos

Esta interfaz muestra la captura de datos del nombre del jugador.

Figura N°3: Interfaz Captura de Datos, Ingreso de Nombre

- Interfaz de Consultas y de Salidas

Esta es la primera interfaz de consulta del videojuego y en esta le pedirá escoger el nivel de dificultad que desea en el videojuego.

Figura N°4: Interfaz de consulta, Dificultad

En la siguiente interfaz de consulta se pide al usuario escoger la campaña que desea jugar.

Figura N°5: Interfaz de Consulta, Campañas

En la siguiente interfaz de consulta se pide al usuario modificar algunas de las opciones del juego (Dificultad o personalización del jugador)

Figura N°6: Interfaz de Consulta, Opciones

En la siguiente interfaz de consulta se pide al usuarios iniciar la partida, ver una descripción de la misma o cancelar la acción.

Figura N°7: Interfaz de Consulta, Inicio Campaña

En la siguiente interfaz de consulta se pide al usuario escoger si es hombre o mujer para definir el personaje principal del videojuego.

Figura N°8: Interfaz de Consulta, Sexo

En la siguiente interfaz de consulta se visualizan las opciones que le permite al jugador realizar el Centro Urbano del Juego; (Dejar Recursos, Reclutar Unidades).

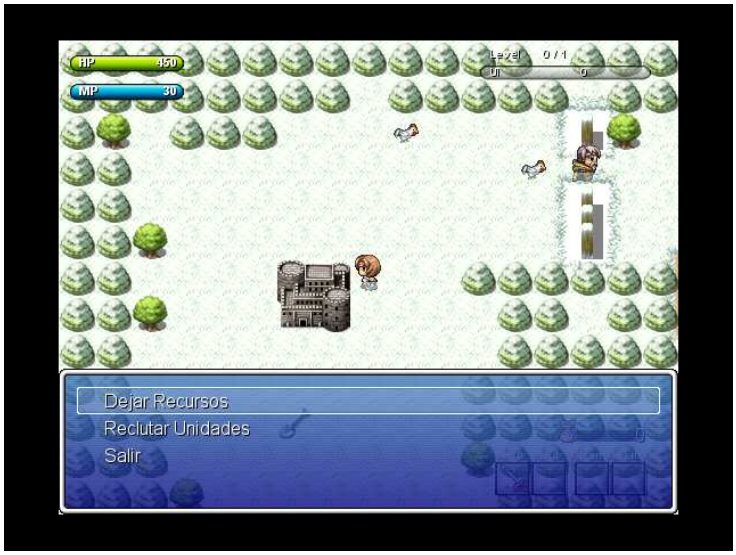


Figura N°9: Interfaz de Consulta, Centro Urbano

En la siguiente interfaz de consulta se visualizan las unidades que se pueden reclutar en el Centro Urbano del juego; (Constructor, Ingeniero y Salir).

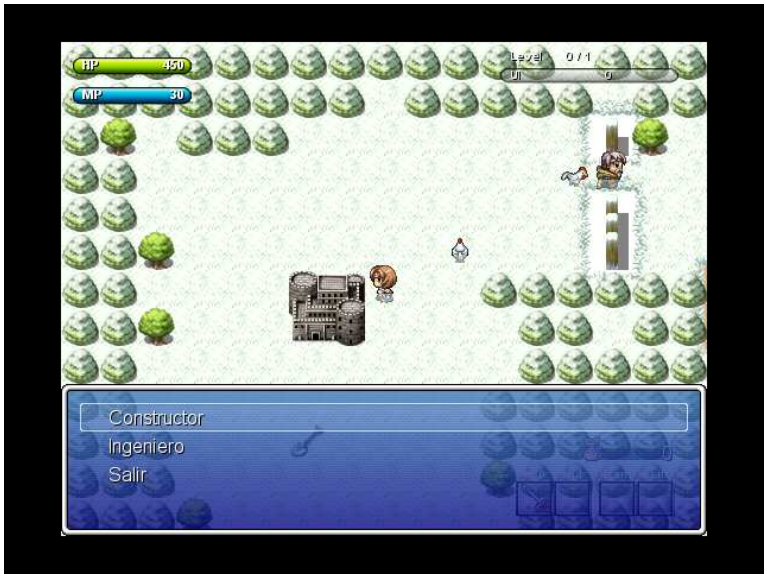


Figura N°10: Interfaz de Consulta, Reclutar Unidade s

La interfaz de consulta principal consta de 6 items: Objetos, Técnicas, Equipamiento, Estado, Guardar y Fin del Juego y en cada ítem podrá visualizar una interfaz de salida que se mostrarán a continuación.

Figura N°11: Interfaz de Consulta, Menú Principal

La interfaz de salida al seleccionar Objetos mostrará los objetos que tiene en su poder.

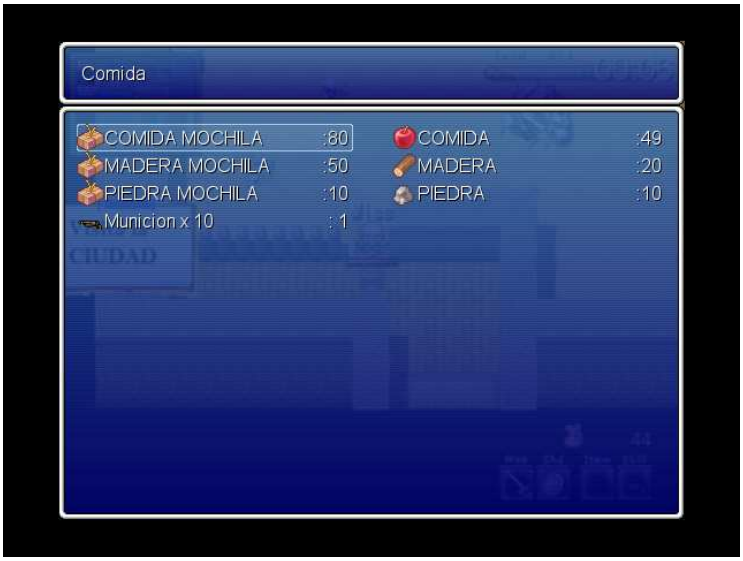


Figura N°12: Interfaz de Salida, Objetos

La interfaz de salida al seleccionar técnicas mostrará las técnicas aprendidas durante el transcurso del videojuego.

Figura N°13: Interfaz de Salida, Técnicas

La interfaz de salida al seleccionar equipamiento mostrará cada una de las armas que posee el jugador durante el transcurso del videojuego.

Figura N°14: Interfaz de Salida, Equipamiento

La interfaz de salida al seleccionar estado mostrará todo el estado del jugador, el nivel en el que se encuentra, el nombre del jugador principal, La experiencia que ha adquirido, cuanta experiencia falta para pasar al siguiente nivel, con que equipamiento cuenta en ese momento, y los niveles de ataque, defensa, inteligencia y agilidad.

Figura N°15: Interfaz de Salida, Estado

La interfaz de salida al seleccionar guardar mostrará los ficheros que tiene guardados o que desea guardar.

Figura N°16: Interfaz de Salida, Guardar

La interfaz de salida al seleccionar fin del juego mostrará una nueva interfaz de consulta donde el usuario debe escoger si desea volver al título, salir a Windows y volver.

Figura N°17: Interfaz de Salida, Fin de Juego

La interfaz de consulta de inicio permite al usuario escoger un nuevo juego, permite continuar si se ha guardado alguna partida o permite salir del videjuego.

Figura N°18: Interfaz de Consulta, Inicio

La siguiente interfaz de salida muestra las misiones que el jugador principal debe completar para rescatar el videojuego.

Figura N°19: Interfaz de Salida, Misiones

En la siguiente interfaz de salida se muestra al jugador el mensaje de que la personalización ha sido correcta.

Figura N°20: Interfaz de Salida, Personalización

La interfaz de diálogos es de salida y en esta se visualizan las conversaciones del narrador y de los diferentes jugadores secundarios del videojuego.

Figura N°21: Interfaz de Salida, Diálogos

La interfaz de salida de los recursos permite visualizar al usuario la cantidad

obtenida en cada uno de los recursos (comida, madera, piedra y oro).

Figura N°22: Interfaz de Salida, Recursos

La interfaz de Construir es de Consulta ya que permite al usuario escoger que edificio desa construir (Cuartel, Establo y Casas). Al seleccionar cualquiera de estas opciones mostrará una interfaz de salida que el usuario visualizara por medio de un diálogo después de la selección.

Figura N°23: Interfaz de Consulta, Construir

La interfaz de ingresar a los lugares es de consulta ya que le permite al usuario escoger si desea entrar o no, si selecciona entrar aparecerá en el campo correspondiente y si selecciona no entrar saldra al campo en el que se encontraba.

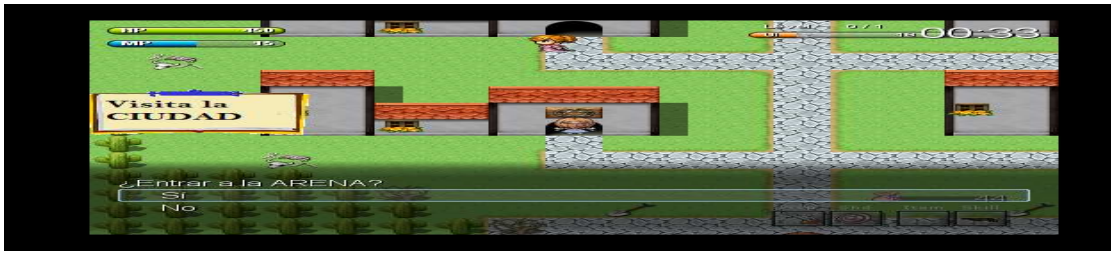


Figura N°24: Interfaz de Salida, Lugares

- DISEÑO DEL SISTEMA

El lenguaje de Modelamiento Unificado (UML- Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables.

El diseño del sistema se divide en 3 diferentes modelados, el primero es el modelado de estructura estática el cual va estar representado por los diagramas de clases del RPG Maker Vx, el segundo modelado es el de interacción del sistema que se divide en 2 el básico y el complejo. El modelado de interacción del sistema básico está compuesto por los diagramas de colaboración y los diagramas de secuencia y el modelado de interacción del sistema complejo está compuesto por los diagramas de componentes y los diagramas de distribución. Por último se encuentra el modelado dinámico que consta de los diagramas de casos de uso, diagramas de estado y diagramas de actividad.

Cada uno de los modelados anteriormente nombrados se describirá al transcurso de este capítulo, comenzando por el modelado de estructura

estática.

3.3.1 Modelado de Estructura Estática

Un diagrama de estructura estática muestra el conjunto de clases y objetos importantes que hacen parte de un sistema, junto con las relaciones existentes entre estas clases y objetos. Muestra de una manera estática la estructura de información del sistema y la visibilidad que tiene cada una de las clases, dada por sus relaciones con las demás en el modelo. El modelado de estructura estática se realizará con los diagramas de clases.

- Diagrama de Clases del RPG Maker Vx

Para realizar los diferentes diagramas de clases utilizados por el RPG Maker Vx directamente se tiene en cuenta un aspecto imprescindible; el lenguaje que maneja el maker en sus scripts es Ruby este es un lenguaje tipeado pero ni en sus variables ni métodos se necesita declarar el tipo de esta es decir no se declara si es una cadena, un entero, o un flotante simplemente se da el nombre a la variables o métodos, por tal motivo los diagramas de clases del RPG Maker Vx a continuación no tienen definido en su interior el tipo de variables o métodos.

Los diagramas que se visualizan a continuación son bastante complejos ya que son manejados directamente por el RPG Maker Vx; por lo tanto para una visualización adecuada revisar el documento digital del proyecto.

Diagrama de clases: Objetos del Juego

Game: Las clases de Game son las responsables de realizar todas las acciones del juego

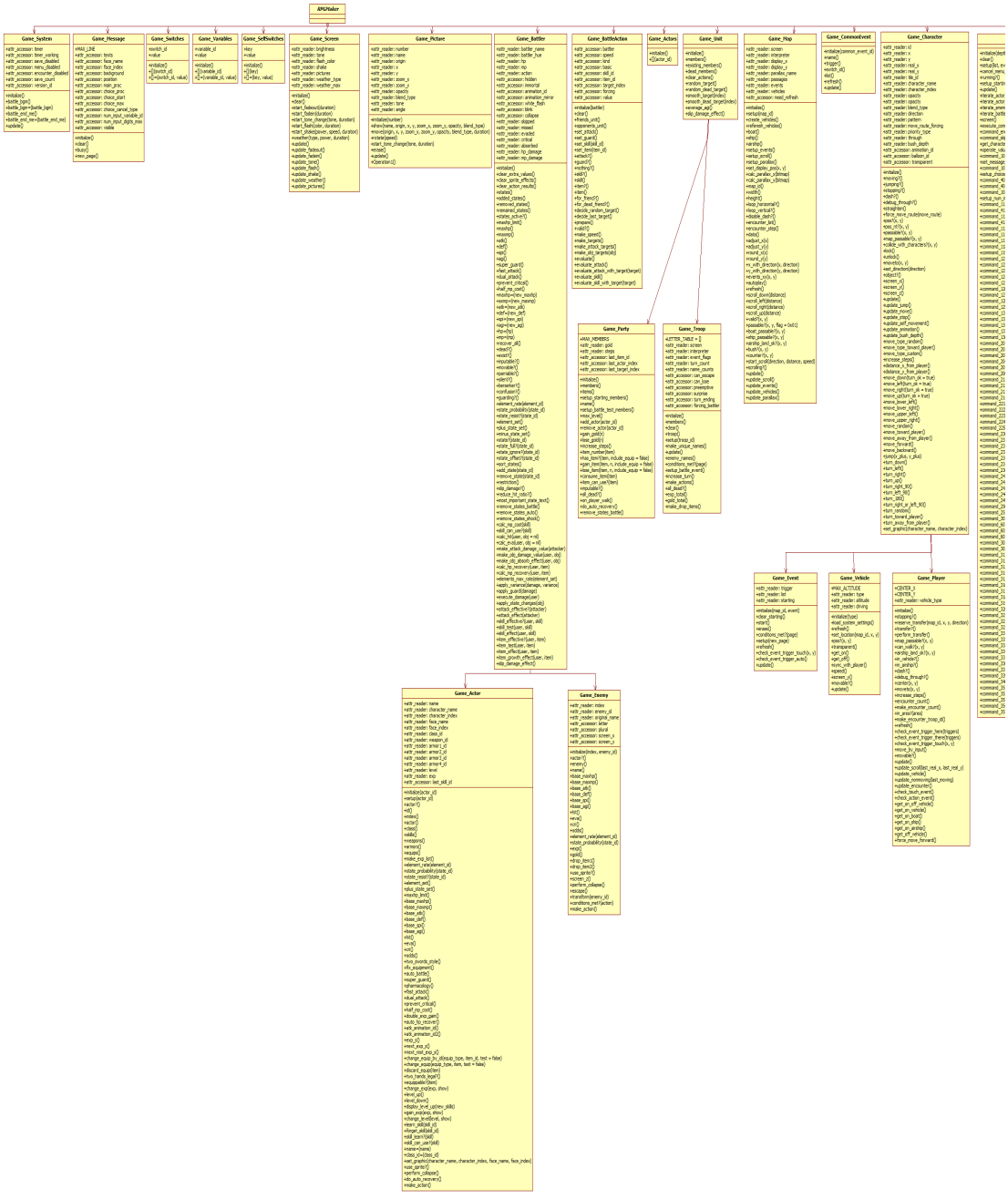

RPGMaker
Game_System
Game_Message
Game_Switches
Game_Variables
Game_SelfSwitches
Game_Screen
Game_Picture
Game_Battler
Game_BattleAction
Game_Actors
Game_Unit
Game_Map
Game_CommonEvent
Game_Character
Game_Party
Game_Troop
Game_Event
Game_Vehicle
Game_Player
Game_Actor
Game_Enemy

Figura N°25: Diagrama de clases del RPG Maker VX: O bjetos del Juego

Diagrama de clases: Sprites

Sprites: Las clases de Sprite son las responsables de todo lo relacionado con los gráficos y dibujos del juego.

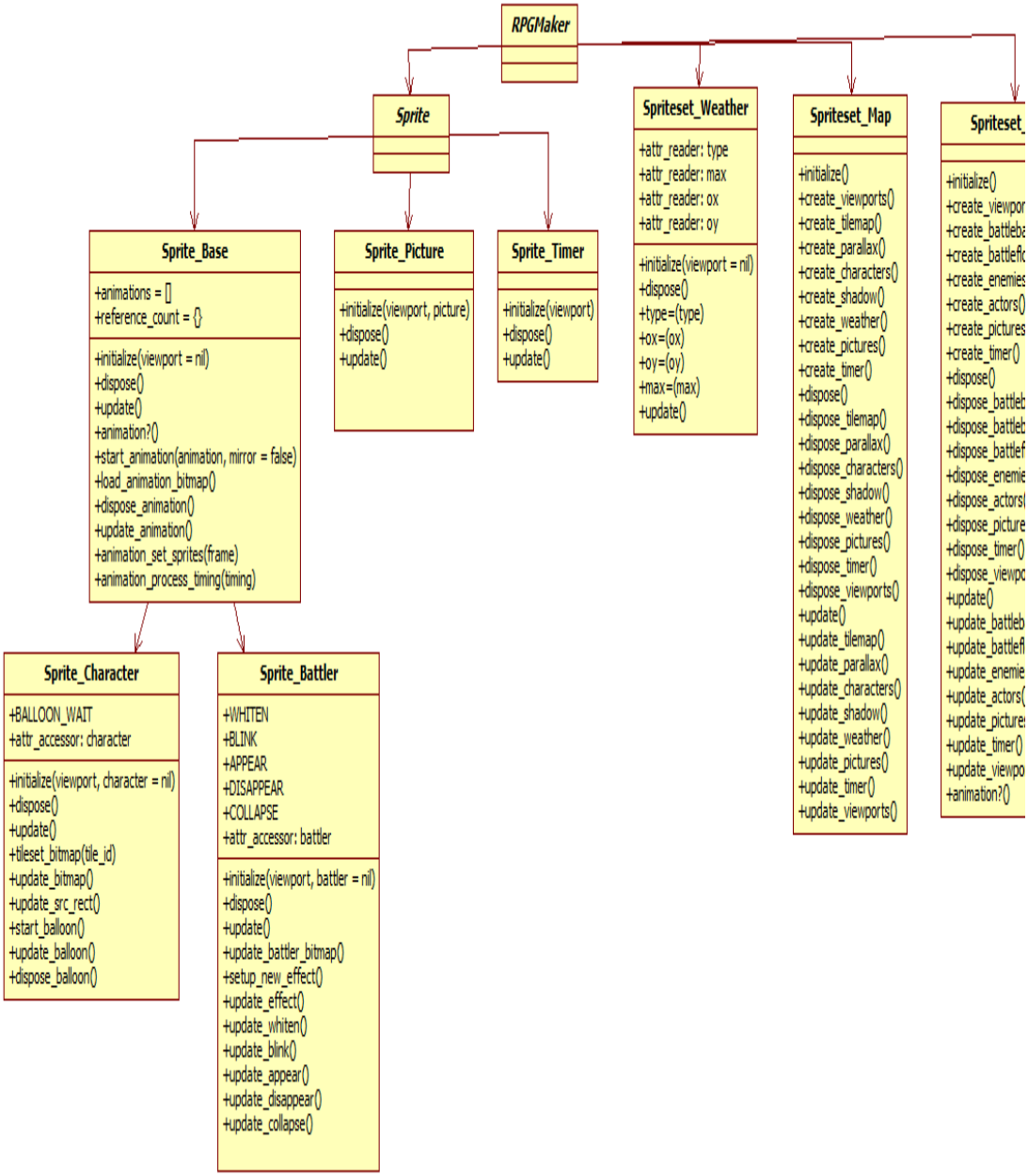

RPGMaker
Sprite
Spriteset_Weather
+attr_reader: type
+attr_reader: max
+attr_reader: ox
+attr_reader: oy
+initialize(viewport = nil)
+dispose()
+type=(type)
+ox=(ox)
+oy=(oy)
+max=(max)
+update()
Spriteset_Map
+initialize()
+create_viewports()
+create_tilemap()
+create_parallax()
+create_characters()
+create_shadow()
+create_weather()
+create_pictures()
+create_timer()
+dispose()
+dispose_tilemap()
+dispose_parallax()
+dispose_characters()
+dispose_shadow()
+dispose_weather()
+dispose_pictures()
+dispose_timer()
+dispose_viewports()
+update()
+update_tilemap()
+update_parallax()
+update_characters()
+update_shadow()
+update_weather()
+update_pictures()
+update_timer()
+update_viewports()
Sprite_Base
+animations = []
+reference_count = {}
+initialize(viewport = nil)
+dispose()
+update()
+animation?()
+start_animation(animation, mirror = false)
+load_animation_bitmap()
+dispose_animation()
+update_animation()
+animation_set_sprites(frame)
+animation_process_timing(timing)
Sprite_Picture
+initialize(viewport, picture)
+dispose()
+update()
Sprite_Timer
+initialize(viewport)
+dispose()
+update()
Sprite_Character
+BALLOON_WAIT
+attr_accessor: character
+initialize(viewport, character = nil)
+dispose()
+update()
+tileset_bitmap(tile_id)
+update_bitmap()
+update_src_rect()
+start_balloon()
+update_balloon()
+dispose_balloon()
Sprite_Battler
+WHITEN
+BLINK
+APPEAR
+DISAPPEAR
+COLLAPSE
+attr_accessor: battler
+initialize(viewport, battler = nil)
+dispose()
+update()
+update_battler_bitmap()
+setup_new_effect()
+update_effect()
+update_whiten()
+update_blink()
+update_appear()
+update_disappear()
+update_collapse()

Figura N°26: Diagrama de clases del RPG Maker VX: Sprites

Diagrama de clases: Ventanas

Windows: Las clases existentes en Windows, son todas las ventanas que pertenecen a una escena y que aparecen en el videojuego. Las ventanas son bloques que pueden ser fácilmente manipulados. El bloque de Titulo donde eliges las opciones para jugar es una ventana las secciones del menú default son ventanas.

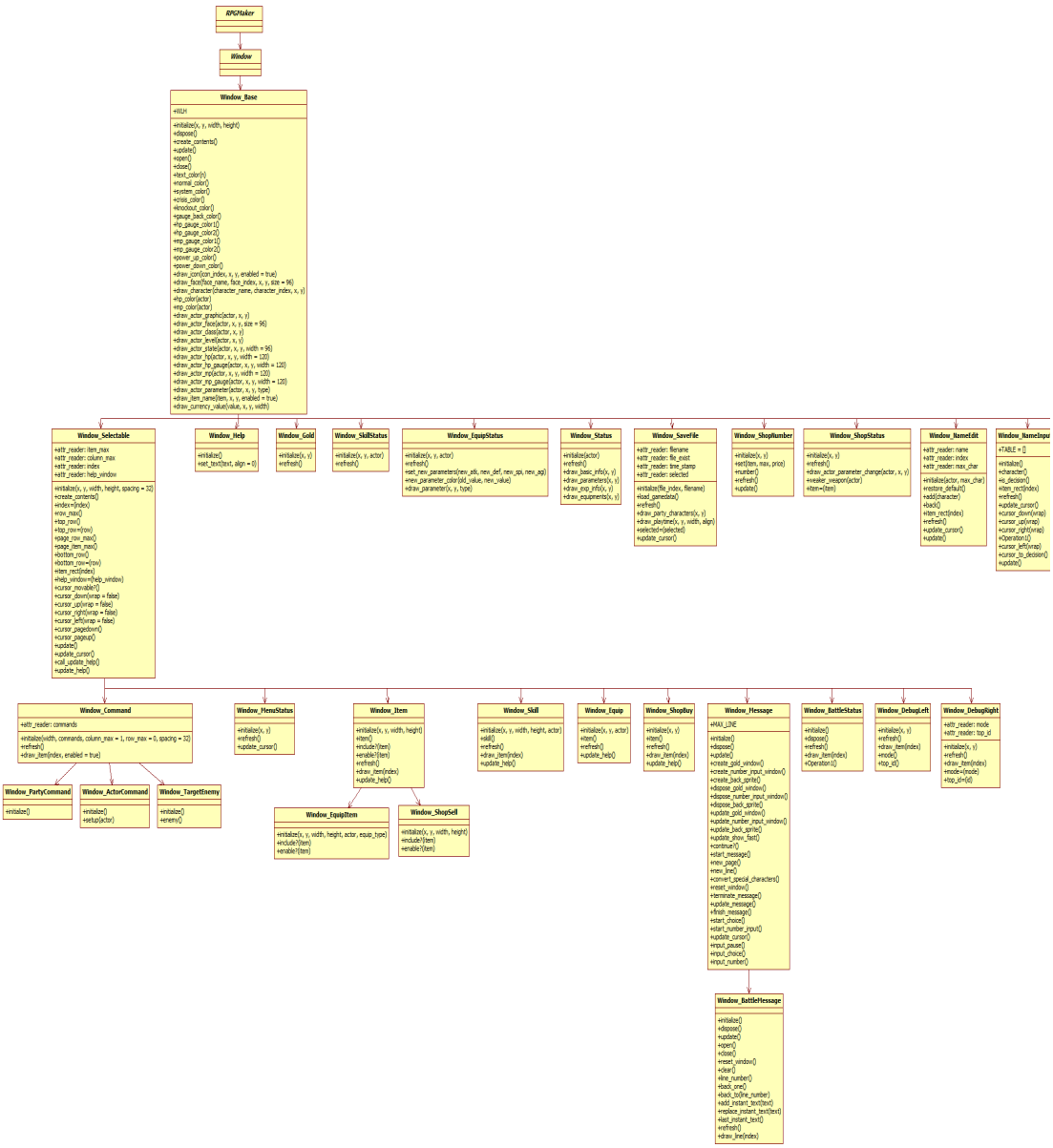

RPGMaker
Window
Window_Base
Window_Selectable
Window_Help
Window_Gold
Window_SkillStatus
Window_EquipStatus
Window_Status
Window_SaveFile
Window_ShopNumber
Window_ShopStatus
Window_NameEdit
Window_NameInput
Window_Command
Window_MenuStatus
Window_Item
Window_Skill
Window_Equip
Window_ShopBuy
Window_Message
Window_BattleStatus
Window_DebugLeft
Window_DebugRight
Window_PartyCommand
Window_ActorCommand
Window_TargetEnemy
Window_EquipItem
Window_ShopSell
Window_BattleMessage

Figura N°27: Diagrama de clases del RPG Maker VX: Ventanas

Diagrama de clases: Escenas

Scenes: El paquete de Scenes contiene las escenas en sí. La escena lo es todo, el mapa, el título es una escena, pero solo puede haber una escena actual. La escena es también un objeto y dentro de ella puede o no contener ventanas, en esta se actualiza y verifican datos para ejecutar instrucciones en un futuro.

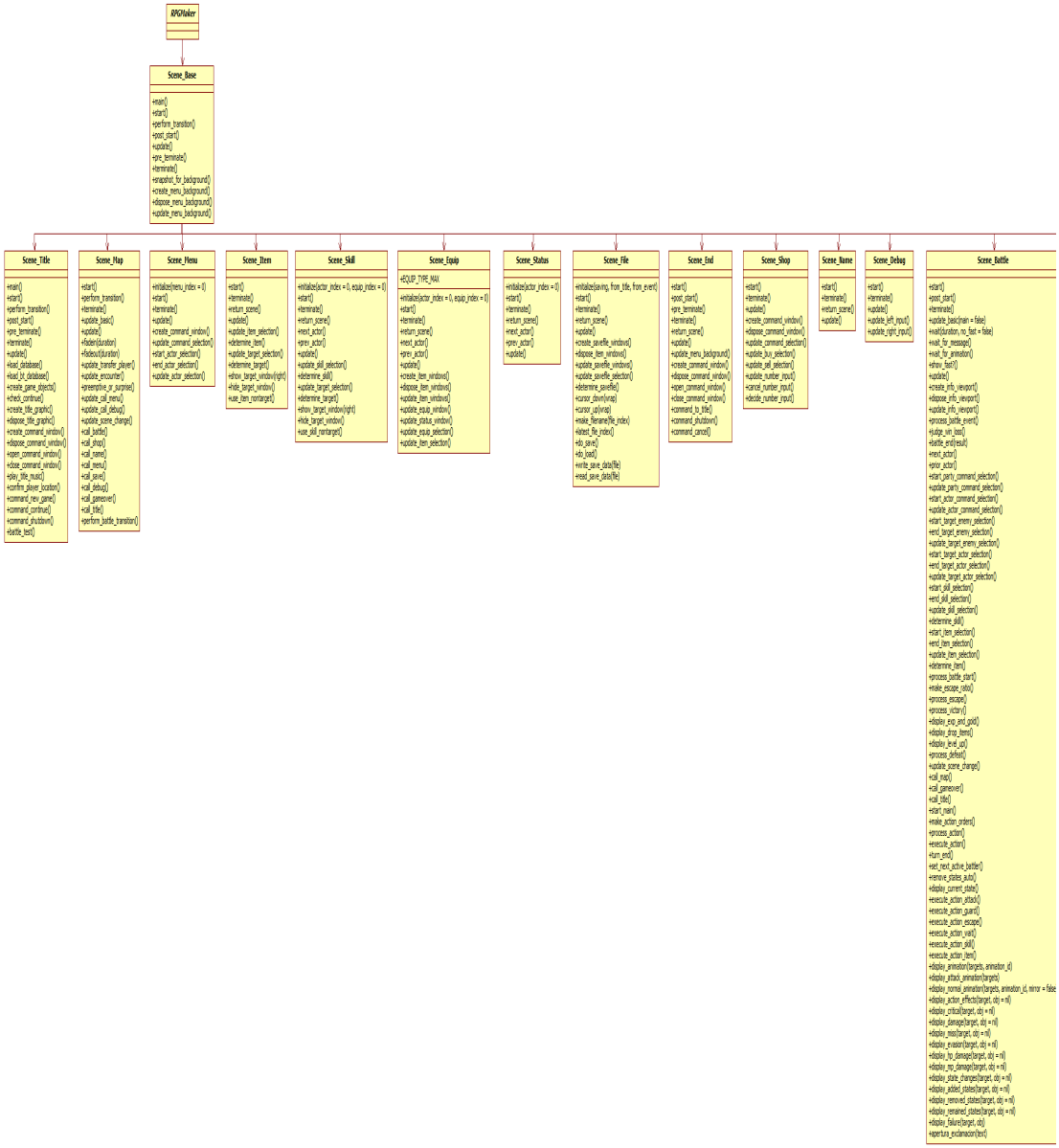

RPGMaker
Scene_Base
+main()
+start()
+perform_transition()
+post_start()
+update()
+pre_terminate()
+terminate()
+snapshot_for_background()
+create_menu_background()
+dispose_menu_background()
+update_menu_background()
Scene_Title
+main()
+start()
+perform_transition()
+post_start()
+pre_terminate()
+terminate()
+update()
+load_database()
+load_bt_database()
+create_game_objects()
+check_continue()
+create_title_graphic()
+dispose_title_graphic()
+create_command_window()
+dispose_command_window()
+open_command_window()
+close_command_window()
+play_title_music()
+confirm_player_location()
+command_new_game()
+command_continue()
+command_shutdown()
+battle_test()
Scene_Map
+start()
+perform_transition()
+terminate()
+update_basic()
+update()
+fadein(duration)
+fadeout(duration)
+update_transfer_player()
+update_encounter()
+preemptive_or_surprise()
+update_call_menu()
+update_call_debug()
+update_scene_change()
+call_battle()
+call_shop()
+call_name()
+call_menu()
+call_save()
+call_debug()
+call_gameover()
+call_title()
+perform_battle_transition()
Scene_Menu
+initialize(menu_index = 0)
+start()
+terminate()
+update()
+create_command_window()
+update_command_selection()
+start_actor_selection()
+end_actor_selection()
+update_actor_selection()
Scene_Item
+start()
+terminate()
+return_scene()
+update()
+update_item_selection()
+determine_item()
+update_target_selection()
+determine_target()
+show_target_window(right)
+hide_target_window()
+use_item_nontarget()
Scene_Skill
+initialize(actor_index = 0, equip_index = 0)
+start()
+terminate()
+return_scene()
+next_actor()
+prev_actor()
+update()
+update_skill_selection()
+determine_skill()
+update_target_selection()
+determine_target()
+show_target_window(right)
+hide_target_window()
+use_skill_nontarget()
Scene_Equip
#EQUIP_TYPE_MAX
+initialize(actor_index = 0, equip_index = 0)
+start()
+terminate()
+return_scene()
+next_actor()
+prev_actor()
+update()
+create_item_windows()
+dispose_item_windows()
+update_item_windows()
+update_equip_window()
+update_status_window()
+update_equip_selection()
+update_item_selection()
Scene_Status
+initialize(actor_index = 0)
+start()
+terminate()
+return_scene()
+next_actor()
+prev_actor()
+update()
Scene_File
+initialize(saving, from_title, from_event)
+start()
+terminate()
+return_scene()
+update()
+create_savefile_windows()
+dispose_item_windows()
+update_savefile_windows()
+update_savefile_selection()
+determine_savefile()
+cursor_down(wrap)
+cursor_up(wrap)
+make_filename(file_index)
+latest_file_index()
+do_save()
+do_load()
+write_save_data(file)
+read_save_data(file)
Scene_End
+start()
+post_start()
+pre_terminate()
+terminate()
+return_scene()
+update()
+update_menu_background()
+create_command_window()
+dispose_command_window()
+open_command_window()
+close_command_window()
+command_to_title()
+command_shutdown()
+command_cancel()
Scene_Shop
+start()
+terminate()
+update()
+create_command_window()
+dispose_command_window()
+update_command_selection()
+update_buy_selection()
+update_sell_selection()
+update_number_input()
+cancel_number_input()
+decide_number_input()
Scene_Name
+start()
+terminate()
+return_scene()
+update()
Scene_Debug
+start()
+terminate()
+update()
+update_left_input()
+update_right_input()
Scene_Battle
+start()
+post_start()
+terminate()
+update_basic(main = false)
+wait(duration, no_fast = false)
+wait_for_message()
+wait_for_animation()
+show_fast()
+update()
+create_info_viewport()
+dispose_info_viewport()
+update_info_viewport()
+process_battle_event()
+judge_win_loss()
+battle_end(result)
+next_actor()
+prior_actor()
+start_party_command_selection()
+update_party_command_selection()
+start_actor_command_selection()
+update_actor_command_selection()
+start_target_enemy_selection()
+end_target_enemy_selection()
+update_target_enemy_selection()
+start_target_actor_selection()
+end_target_actor_selection()
+update_target_actor_selection()
+start_skill_selection()
+end_skill_selection()
+update_skill_selection()
+determine_skill()
+start_item_selection()
+end_item_selection()
+update_item_selection()
+determine_item()
+process_battle_start()
+make_escape_ratio()
+process_escape()
+process_victory()
+display_exp_and_gold()
+display_drop_items()
+display_level_up()
+process_defeat()
+update_scene_change()
+call_map()
+call_gameover()
+call_title()
+start_main()
+make_action_orders()
+process_action()
+execute_action()
+turn_end()
+set_next_active_battler()
+remove_states_auto()
+display_current_state()
+execute_action_attack()
+execute_action_guard()
+execute_action_escape()
+execute_action_wait()
+execute_action_skill()
+execute_action_item()
+display_animation(targets, animation_id)
+display_attack_animation(targets)
+display_normal_animation(targets, animation_id, mirror = false
+display_action_effects(target, obj = nil)
+display_critical(target, obj = nil)
+display_damage(target, obj = nil)
+display_miss(target, obj = nil)
+display_evasion(target, obj = nil)
+display_hp_damage(target, obj = nil)
+display_mp_damage(target, obj = nil)
+display_state_changes(target, obj = nil)
+display_added_states(target, obj = nil)
+display_removed_states(target, obj = nil)
+display_remained_states(target, obj = nil)
+display_failure(target, obj)
+apertura_exclamacion(bool)

Figura N°28: Diagrama de clases del RPG Maker VX: E scenas

3.3.2 Modelado de Interacción del Sistema (Básico)

Un modelo de interacción define el principio mediante el cual se diseña la interacción en la interfaz. Este modelo es un enfoque holístico de las acciones y funcionalidades de la herramienta, por lo que indica de forma general como se diseñan los comportamientos del sistema frente a las distintas acciones.

El modelado de interacción del sistema, representa la forma en como un Cliente (Actor) u Objetos (Clases) se comunican entre sí en petición a un evento. Esto implica recorrer toda la secuencia de llamadas, de donde se obtienen las responsabilidades claramente. El modelado de interacción del sistema (Básico) se realizara con los diagramas de colaboración y los diagramas de secuencia

- Diagrama de Colaboración

Es un diagrama de interacción que resalta la organización estructural de los objetos, que envían y reciben mensajes de las iteraciones que están indicadas por un número.

A diferencia de los diagramas de secuencia, pueden mostrar el contexto de la operación (cuáles objetos son atributos, cuáles temporales) y ciclos en la ejecución.

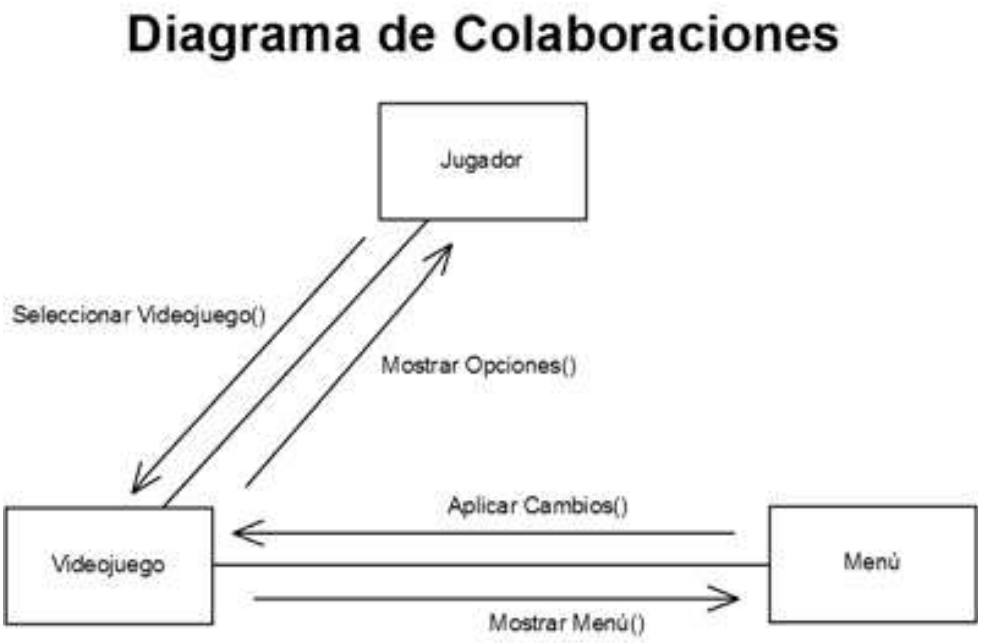


Figura N°29: Diagrama de colaboraciones

- Diagrama de Secuencia

El diagrama de secuencias es un diagrama de interacciones que resalta la ordenación temporal de los mensajes.

Diagrama de Secuencia de Jugar

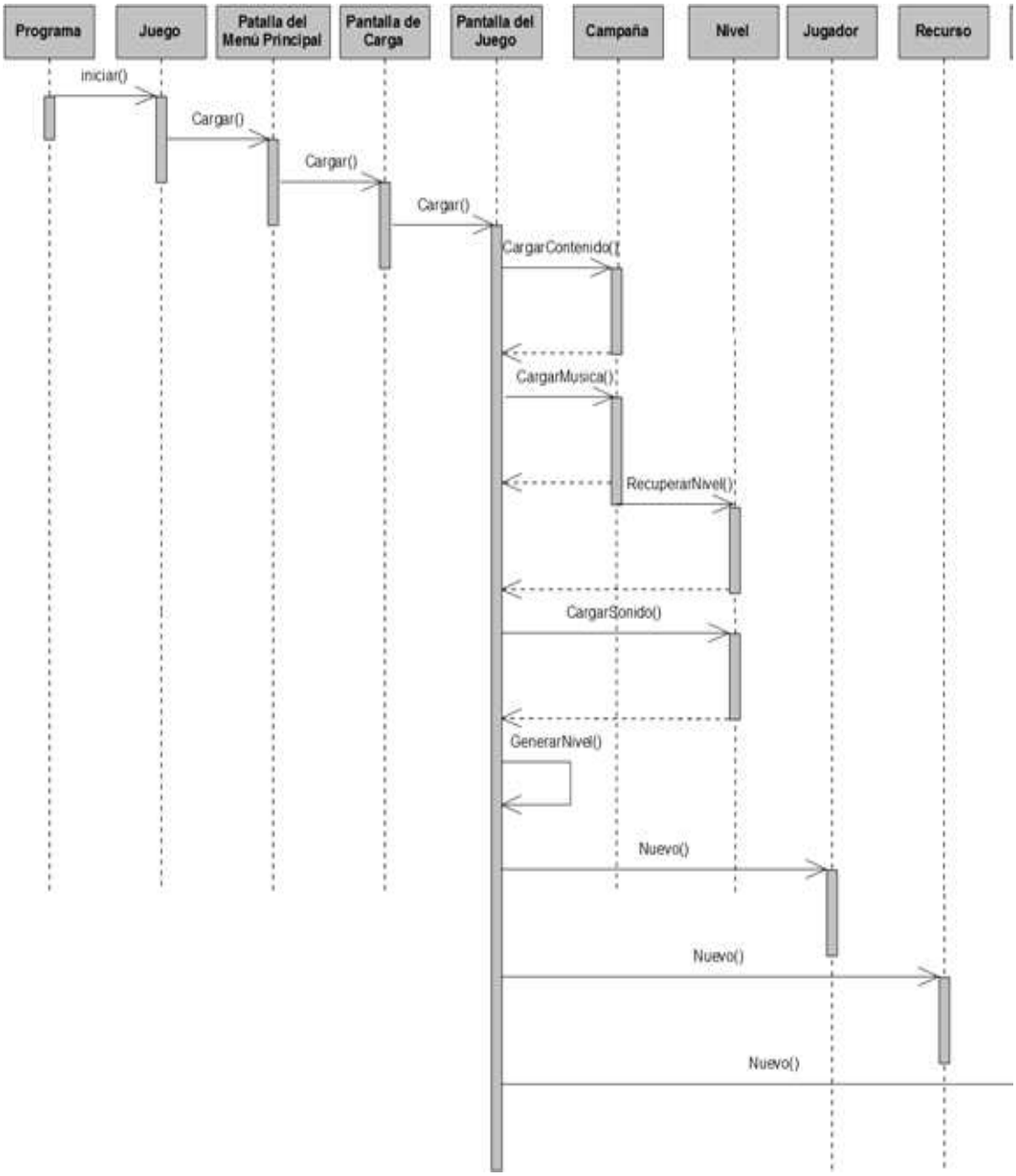

Programa
Juego
Pataila del Menú Principal
Pantalla de Carga
Pantalla del Juego
Campaña
Nivel
Jugador
Recurso
iniciar()
Cargar()
Cargar()
Cargar()
CargarContenido()
CargarMusica()
RecuperarNivel()
CargarSonido()
GenerarNivel()
Nuevo()
Nuevo()
Nuevo()

Figura N°30: Diagrama de Secuencia: Jugar

Diagrama de Secuencia de Mover Persona

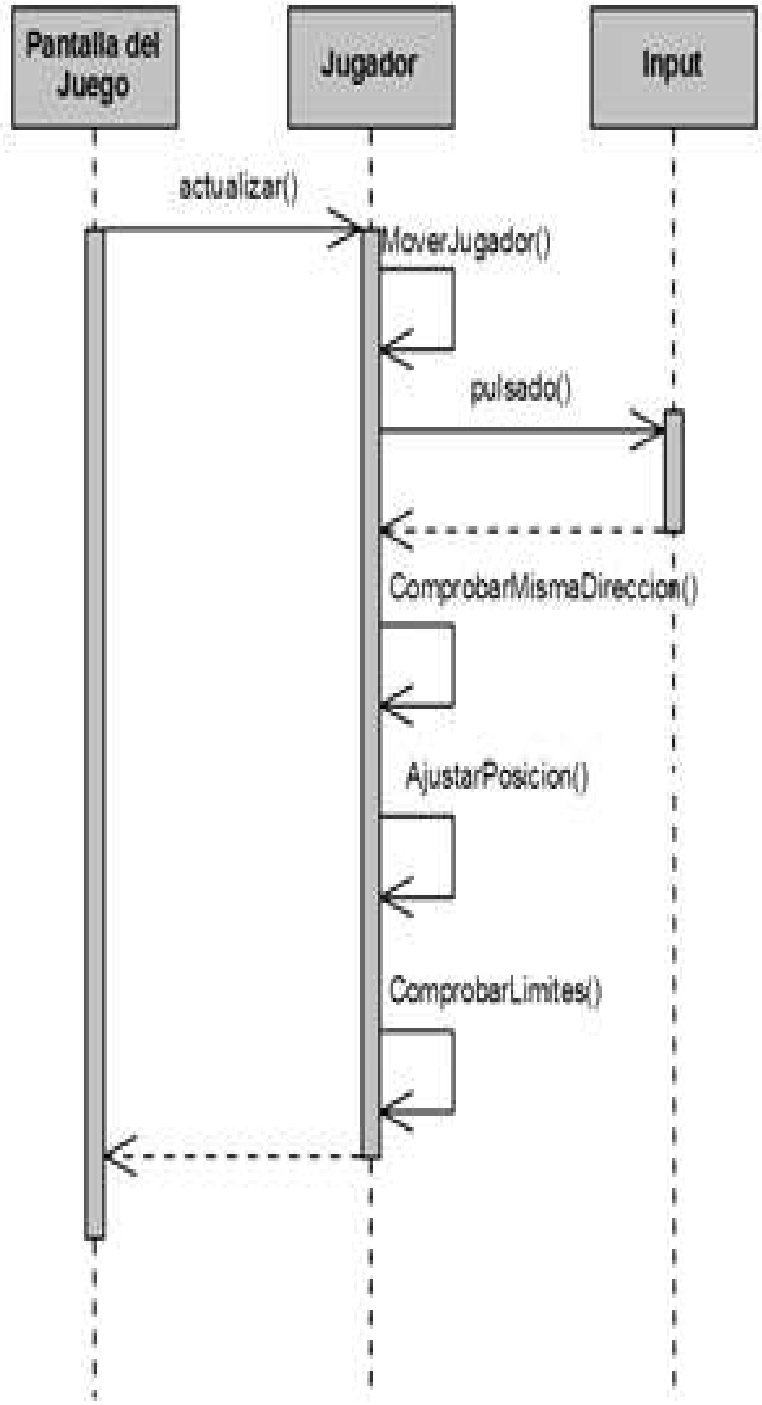


Figura N°31: Diagrama de Secuencia: Mover Personaje

Diagrama de Secuencia de Disparar

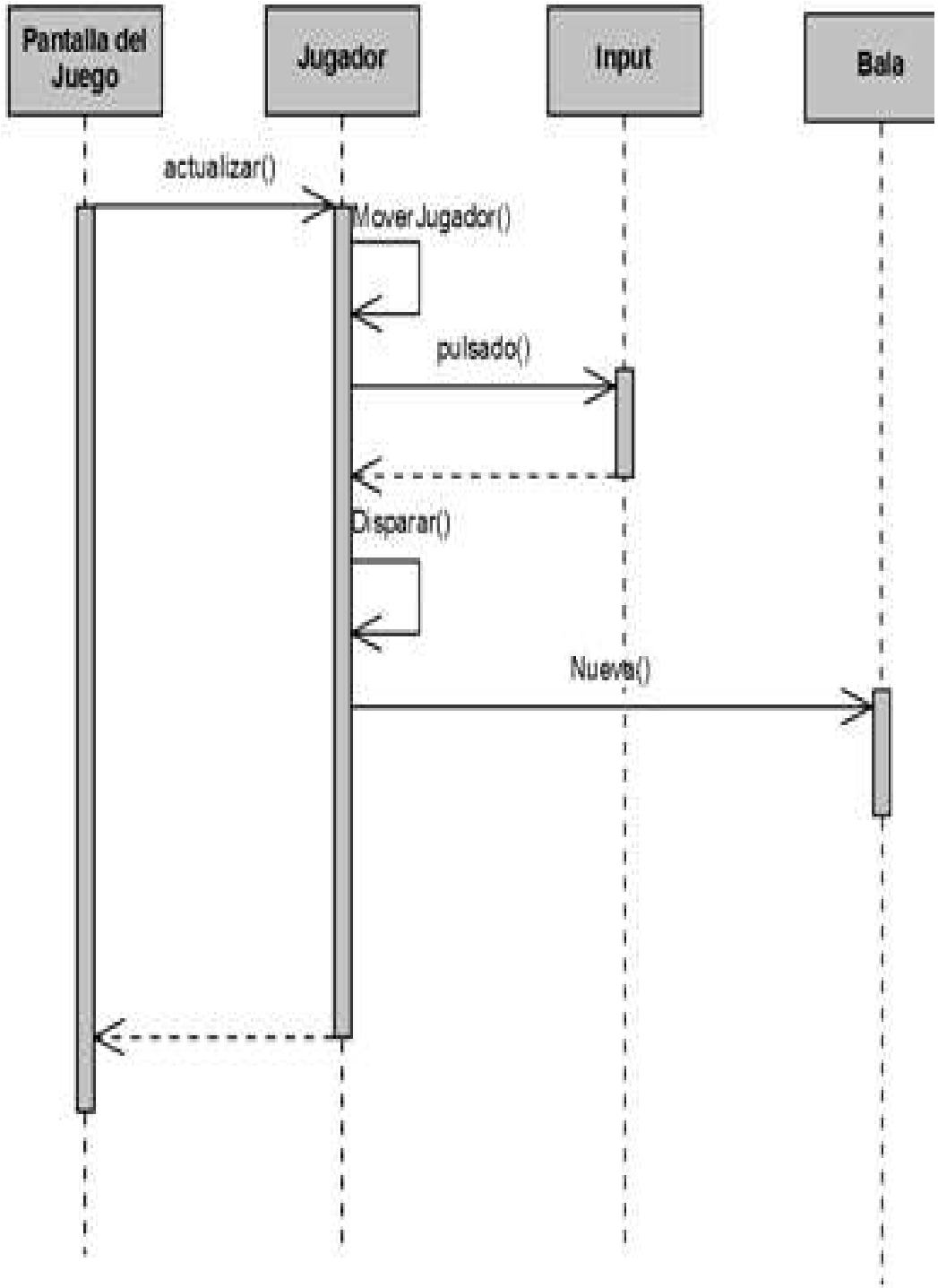


Figura N°32: Diagrama de Secuencia: Disparar

3.3.3 Modelado Dinámico (Objetos)

El modelado dinámico está constituido por los aspectos de un sistema relacionados con el tiempo y con los cambios en los objetos y sus relaciones a lo largo del tiempo. Además Con el modelado dinámico se describe el control en el sistema, es decir, las secuencias de operaciones que ocurren como respuesta a estímulos externos, sin tener en cuenta lo que hacen las operaciones, sobre qué operan o cómo se implementan.

El modelado dinámico se realizará con los diagrama de casos de uso, diagrama de estado y diagrama de actividad

- Diagrama de Casos de Uso

Los diagramas de casos de uso documentan el comportamiento de un sistema desde el punto de vista del usuario. Por lo tanto los casos de uso determinan los requisitos funcionales del sistema, es decir, representan las funciones que un sistema puede ejecutar.

Su ventaja principal es la facilidad para interpretarlos, lo que hace que sean especialmente útiles en la comunicación con el cliente.

Los diagramas de casos de uso son indispensables en el desarrollo de un videojuego ya que muestran cada una de las acciones a realizar en el mismo, de manera general y específica de acuerdo a la amplitud del videojuego.

En el caso del Conflicto Armado, El videojuego se manejara 22 diagramas de casos de uso que describirán de manera explícita la acción más básica de visualización en la interfaz del videojuego hasta la acción más compleja que realiza el jugador en el momento de interactuar con este.

Diagrama de Casos de Uso

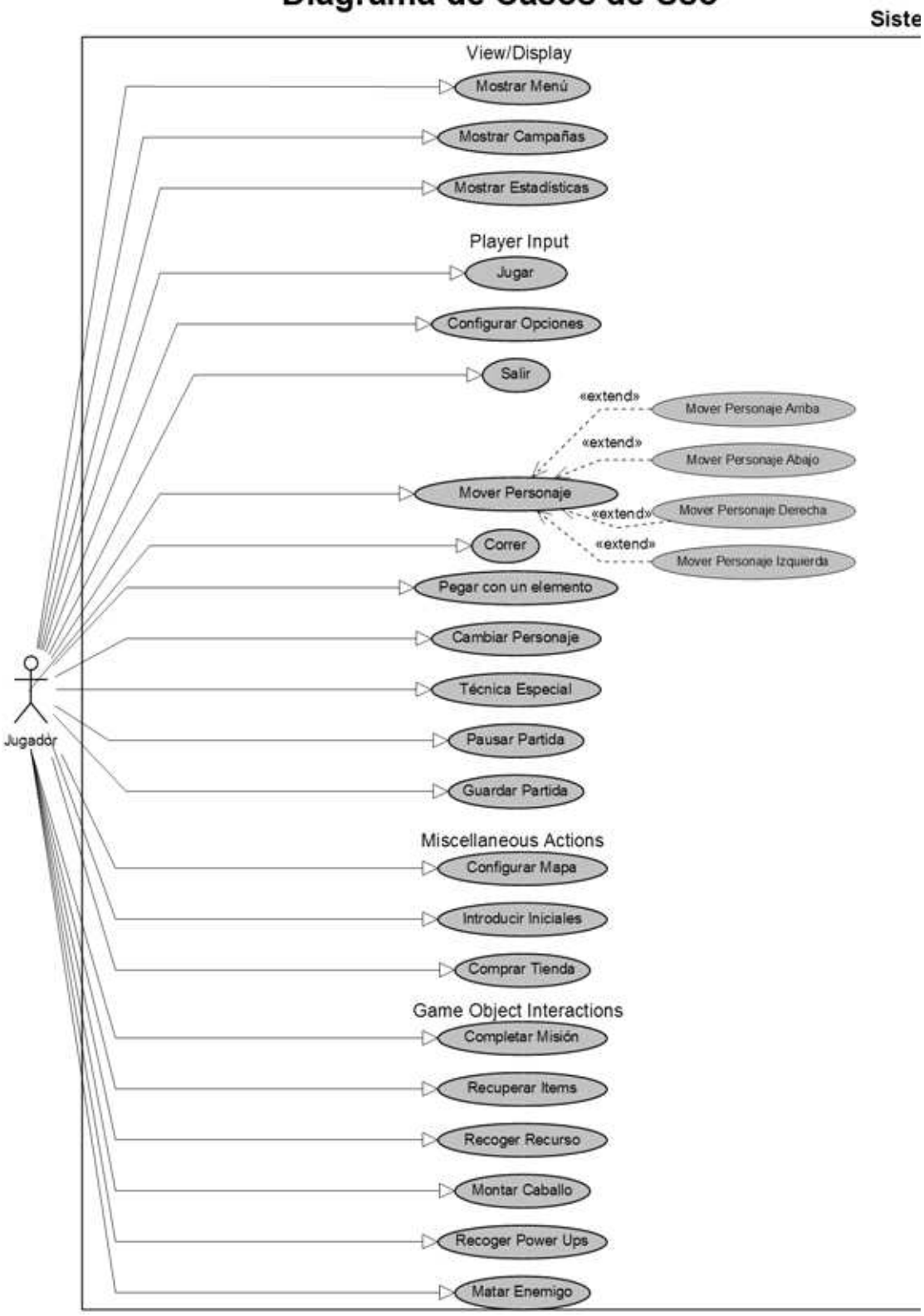


Figura N°33: Diagrama de Casos de Uso

- Especificaciones de Caso de Uso

Como se ve en el diagrama de casos de uso, los casos de uso quedarían agrupados de la siguiente manera:

View/Display: Mostrar menú, mostrar campañas y mostrar estadísticas. En player input: Jugar, configurar opciones, salir, mover personaje (mover personaje arriba, mover personaje abajo, mover personaje derecha, mover personaje izquierda), correr, pegar con un elemento, cambiar personaje, técnica especial, pausar partida, guardar partida.

Miscellaneous Actions: Configurar mapa, introducir iniciales, comprar tienda.

En Game object interactions: completar misión, recuperar ítems, recoger recurso, montar caballo, recoger power ups y matar enemigo.

A continuación, se muestran las tablas con la especificación textual de cada caso de uso, para mayor comprensión. De cada caso de uso se especificarán los siguientes términos:

- **Identificador:** Representa de forma unívoca cada caso de uso. La sintaxis de nombrado será la siguiente: CU-XX, donde XX serán 2 dígitos que numeran ordenadamente los casos de uso.
- **Nombre:** Es el título corto que se le da a cada caso de uso, de acuerdo al diagrama anterior.
- **Actores:** Son los usuarios que intervienen directamente en la realización del caso de uso.
- **Objetivo:** Se refiere al objetivo concreto del caso de uso.
- **Precondiciones:** Condiciones que deben darse previamente en el

sistema para que el caso de uso pueda efectuarse.

- **Postcondiciones:** Indican el estado del sistema después de la ejecución del caso de uso.

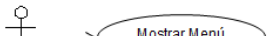
IDENTIFICADOR: CU-01		
	**Nombre**	Mostrar Menú
	Actor(es)	Jugador
	Objetivo	Mostrar el menú de inicio del juego
	Precondiciones	• Haber arrancado el juego
	Postcondiciones	• Se muestra el menú correspondiente a la entrada del juego

Tabla N°19: Especificación Caso de Uso Mostrar Menú

IDENTIFICADOR: CU-02		
	**Nombre**	Mostrar Campañas
	Actor(es)	Jugador
	Objetivo	Mostrar las campañas que tiene el juego
	Precondiciones	• Haber arrancado el juego
	Postcondiciones	• Se muestra el menú correspondiente a la entrada de las campañas

Tabla N°20: Especificación Caso de Uso Mostrar Camp añas

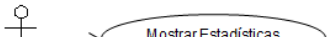
IDENTIFICADOR: CU-03		
	**Nombre**	Mostrar Estadísticas
	Actor(es)	Jugador
	Objetivo	Mostrar el menú de estadísticas del jugador
	Precondiciones	• Haber comenzado una partida
	Postcondiciones	• Se muestra el menú correspondiente a la entrada de las estadísticas del jugador

Tabla N°21: Especificación Caso de Uso Mostrar Estadísticas

IDENTIFICADOR: CU-04		
	**Nombre**	Jugar
	Actor(es)	Jugador
	Objetivo	Comenzar una partida desde el primer nivel
	Precondiciones	• Haber arrancado el juego
	Postcondiciones	• El sistema arranca el primer nivel • El jugador comienza la partida

Tabla N°22: Especificación Caso de Uso Jugar

IDENTIFICADOR: CU-05		
	**Nombre**	Configurar Opciones
	Actor(es)	Jugador
	Objetivo	Acceder al menú de opciones para configurarlas de acorde al jugador
	Precondiciones	• Haber arrancado el juego
	Postcondiciones	• El sistema carga el menú de opciones

Tabla N°23: Especificación Caso de Uso Configurar Opciones

IDENTIFICADOR: CU-06		
	**Nombre**	Salir
	Actor(es)	Jugador
	Objetivo	Cerrar la ventana del juego
	Precondiciones	• Haber arrancado el juego
	Postcondiciones	• Salir de la aplicación

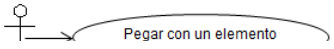
Tabla N°24: Especificación Caso de Uso Salir

IDENTIFICADOR: CU-07		
	**Nombre**	Mover Personaje
	Actor(es)	Jugador
	Objetivo	Desplazar al personaje en el eje de coordenadas X o Y en una unidad positiva o negativa
	Precondiciones	• Haber comenzado una partida • Que la posición de destino no esté ocupada por un objeto no pasable
	Postcondiciones	• El personaje se desplaza en el eje de coordenadas X o Y de forma positiva o negativa

Tabla N°25: Especificación Caso de Uso Mover person aje

IDENTIFICADOR: CU-08		
	**Nombre**	Correr
	Actor(es)	Jugador
	Objetivo	Desplazar al personaje en el eje correspondiente de una manera más rápida
	Precondiciones	• Haber comenzado una partida • Que la posición de destino no esté ocupada por un objeto no pasable
	Postcondiciones	• El personaje se desplaza en el eje de coordenadas correspondiente de manera más rápida

Tabla N°26: Especificación Caso de Uso Correr

IDENTIFICADOR: CU-09		
	**Nombre**	Pegar con un elemento
	Actor(es)	Jugador

	Objetivo	Pegar a los enemigos del juego con un elemer
	Precondiciones	• Haber comenzado una partida • Haber obtenido el elemento
	Postcondiciones	• Un elemento que le permita pegar

Tabla N°27: Especificación Caso de Uso Pegar con un elemento

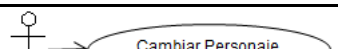
IDENTIFICADOR: CU-10		
	**Nombre**	Cambiar Personaje
	Actor(es)	Jugador
	Objetivo	Cambiar el personaje según corresponda la ac
	Precondiciones	• Haber comenzado una partida • Haber comprado el personaje
	Postcondiciones	• Un nuevo personaje

Tabla N°28: Especificación Caso de Uso Cambiar Personaje

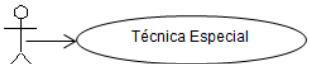
IDENTIFICADOR: CU-11		
	**Nombre**	Técnica Especial
	Actor(es)	Jugador
	Objetivo	Lanzar una técnica especial a los enemigos del
	Precondiciones	• Haber comenzado una partida • Haber obtenido la técnica especial
	Postcondiciones	• Efecto de acuerdo a la técnica especial u

Tabla N°29: Especificación Caso de Uso Técnica Especial

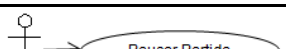
IDENTIFICADOR: CU-12		
	**Nombre**	Pausar Partida
	Actor(es)	Jugador
	Objetivo	Pausar temporalmente una partida activa
	Precondiciones	• Haber comenzado una partida
	Postcondiciones	• Se muestra el menú de Pausa • El jugador podrá volver a la partida o salir

Tabla N°30: Especificación Caso de Uso Pausar Partida

IDENTIFICADOR: CU-13		
	**Nombre**	Guardar Partida
	Actor(es)	Jugador
	Objetivo	Guardar el estado de la partida activa
	Precondiciones	• Haber comenzado una partida
	Postcondiciones	• Mostrar el menú de Guardar • El jugador podrá volver a la partida o salir

Tabla N°31: Especificación Caso de Uso Guardar Partida

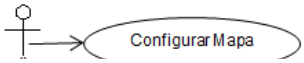
IDENTIFICADOR: CU-14		
	**Nombre**	Configurar Mapa
	Actor(es)	Jugador
	Objetivo	Mostrar un mapa en la parte de la pantalla que desee el jugador
	Precondiciones	• Haber comenzado una partida • Haber obtenido el mapa
	Postcondiciones	• Mostrar el mapa de acuerdo a configuración del jugador

Tabla N°32: Especificación Caso de Uso Configurar Mapa

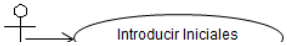
IDENTIFICADOR: CU-15		
	**Nombre**	Introducir Iniciales
	Actor(es)	Jugador
	Objetivo	Introducir un nombre que identifique al jugador en el juego
	Precondiciones	• Haber comenzado una partida
	Postcondiciones	• En los diálogos se muestra el nombre correspondiente

Tabla N°33: Especificación Caso de Uso Introducir Iniciales

IDENTIFICADOR: CU-16		
	**Nombre**	Comprar Tienda
	Actor(es)	Jugador
	Objetivo	Comprar elementos necesarios o valiosos en una tienda
	Precondiciones	• Haber comenzado una partida • Tener recursos suficientes para el elemento
	Postcondiciones	• Elemento adquirido, abrir menú para utilizarlo

Tabla N°34: Especificación Caso de Uso Comprar Tien da

IDENTIFICADOR: CU-17		
	**Nombre**	Completar Misión
	Actor(es)	Jugador
	Objetivo	Completar la misión de acuerdo a la campaña escogida
	Precondiciones	• Haber comenzado una partida
	Postcondiciones	• Mostrar estadísticas reales • Mostrar el menú de inicio

Tabla N°35: Especificación Caso de Uso Completar Mi sión

IDENTIFICADOR: CU-18		
	**Nombre**	Recuperar Ítems
	Actor(es)	Jugador
	Objetivo	Recuperar salud y energía para seguir combatiendo
	Precondiciones	• Haber comenzado una partida • Tener bajo la barra de salud y energía
	Postcondiciones	• Recuperar 100% la barra de salud y energía del jugador

Tabla N°36: Especificación Caso de Uso Recuperar Ít ems

IDENTIFICADOR: CU-19

	**Nombre**	Recoger Recurso
	Actor(es)	Jugador
	Objetivo	Recoger la comida de las granjas
	Precondiciones	• Haber comenzado una partida
	Postcondiciones	• Nuevos recursos para el jugador y se inicia el contador nuevamente

Tabla N°37: Especificación Caso de Uso Recuperar Re curso

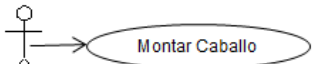
IDENTIFICADOR: CU-20		
	**Nombre**	Montar Caballo
	Actor(es)	Jugador
	Objetivo	Montar un caballo para andar más rápido y entrar a zonas especiales
	Precondiciones	• Haber comenzado una partida • Haber obtenido un caballo
	Postcondiciones	• Mostrar el jugador en el caballo

Tabla N°38: Especificación Caso de Uso Montar Cabal lo

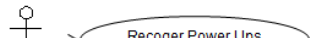
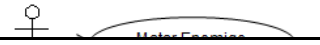
IDENTIFICADOR: CU-21		
	**Nombre**	Recoger Power Ups
	Actor(es)	Jugador
	Objetivo	El jugador recoge un Power Up del mapa y se le aplica su efecto
	Precondiciones	• Haber comenzado una partida • Haber destruido un enemigo que contenía un Power Up • Haber encontrado un cofre que contenía un Power Up
	Postcondiciones	• Se aplica el efecto correspondiente al Power Up al personaje • El Power Up desaparece del mapa

Tabla N°39: Especificación Caso de Uso Recoger Powe r Ups

IDENTIFICADOR: CU-23		
	**Nombre**	Matar Enemigo

	Actor(es)	Jugador
	Objetivo	El jugador golpea al enemigo hasta matarlo
	Precondiciones	• Haber comenzado una partida
	Postcondiciones	• El enemigo muere y desaparece del mapa

Tabla N°40 Especificación Caso de Uso Matar Enemigo

- Diagrama de Estado

El diagrama de estados es un diagrama utilizado para identificar cada una de las rutas o caminos que puede tomar un flujo de información luego de ejecutarse cada proceso. Además permite identificar bajo qué argumentos se ejecuta cada uno de los procesos y en qué momento podrían tener una variación y asimismo permite visualizar de una forma secuencial la ejecución de cada uno de los procesos.

Diagrama de Estados

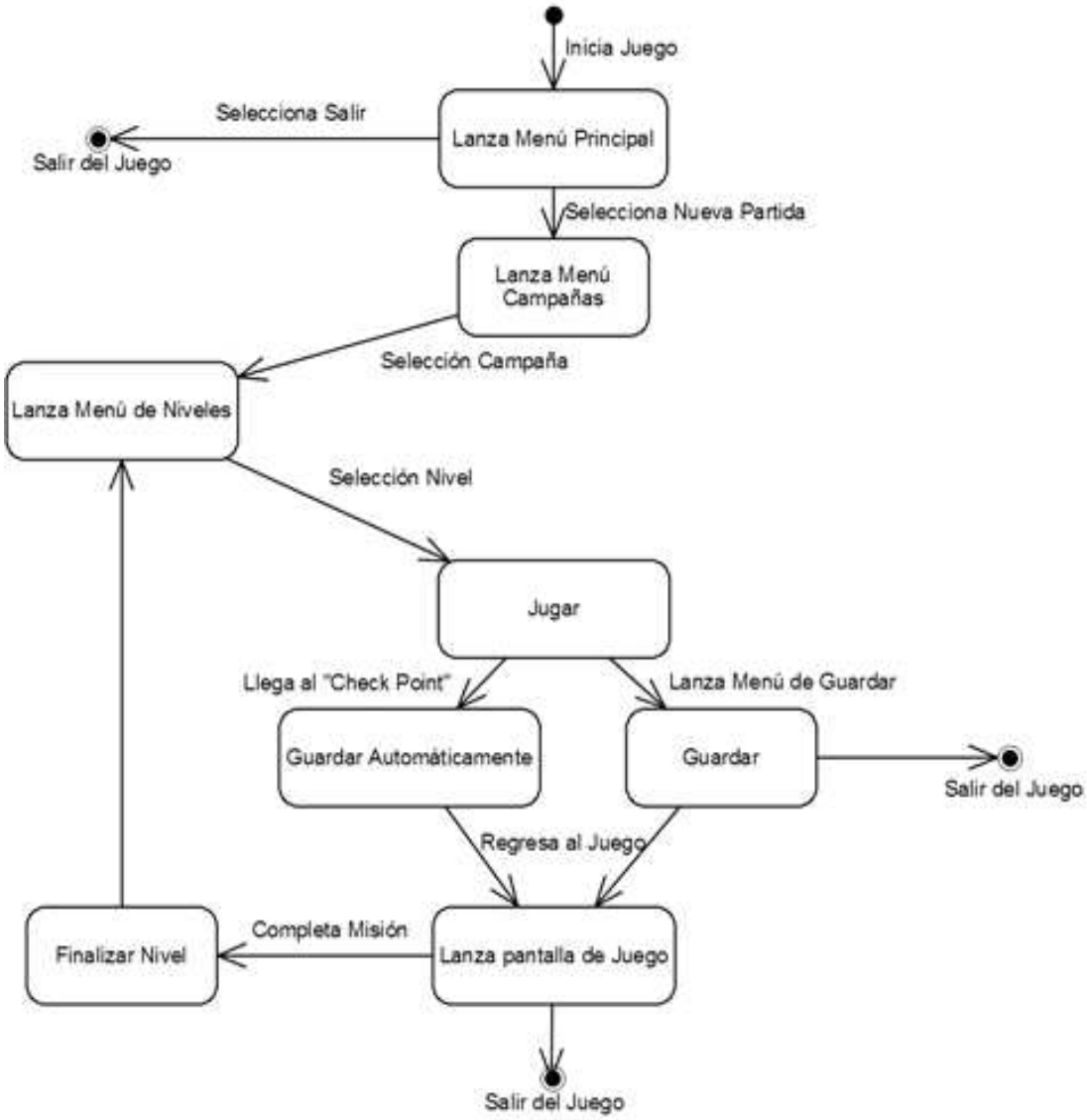


Figura N°34: Diagrama de Estado

- Diagrama de Actividad

El diagrama de actividad muestra el flujo de actividades dentro de un sistema. Además es un diagrama de flujo del proceso multi-propósito que se usa para modelar el comportamiento del sistema. Los diagramas de actividad se pueden usar para modelar un Caso de Uso, o una clase, o un método complicado.

Una vez especificados los casos de uso, los cuales representan la interacción entre usuario y sistema, quedan más claras las distintas funcionalidades que el usuario puede realizar. Para que quede más claro el comportamiento del sistema se mostrará a continuación su diagrama de actividad, que recoge las transiciones y eventos.

Los diagramas de actividad describen la secuencia de las actividades en un sistema. En ellos, las transiciones entre estados se producen como consecuencia de eventos y pueden tener un procesamiento asociado. Representan los sucesos que influyen en el comportamiento y evolución del sistema, describiendo tanto su comportamiento normal (ejecución típica), como excepcional (errores o excepciones). A continuación se muestra el diagrama actividad del sistema:

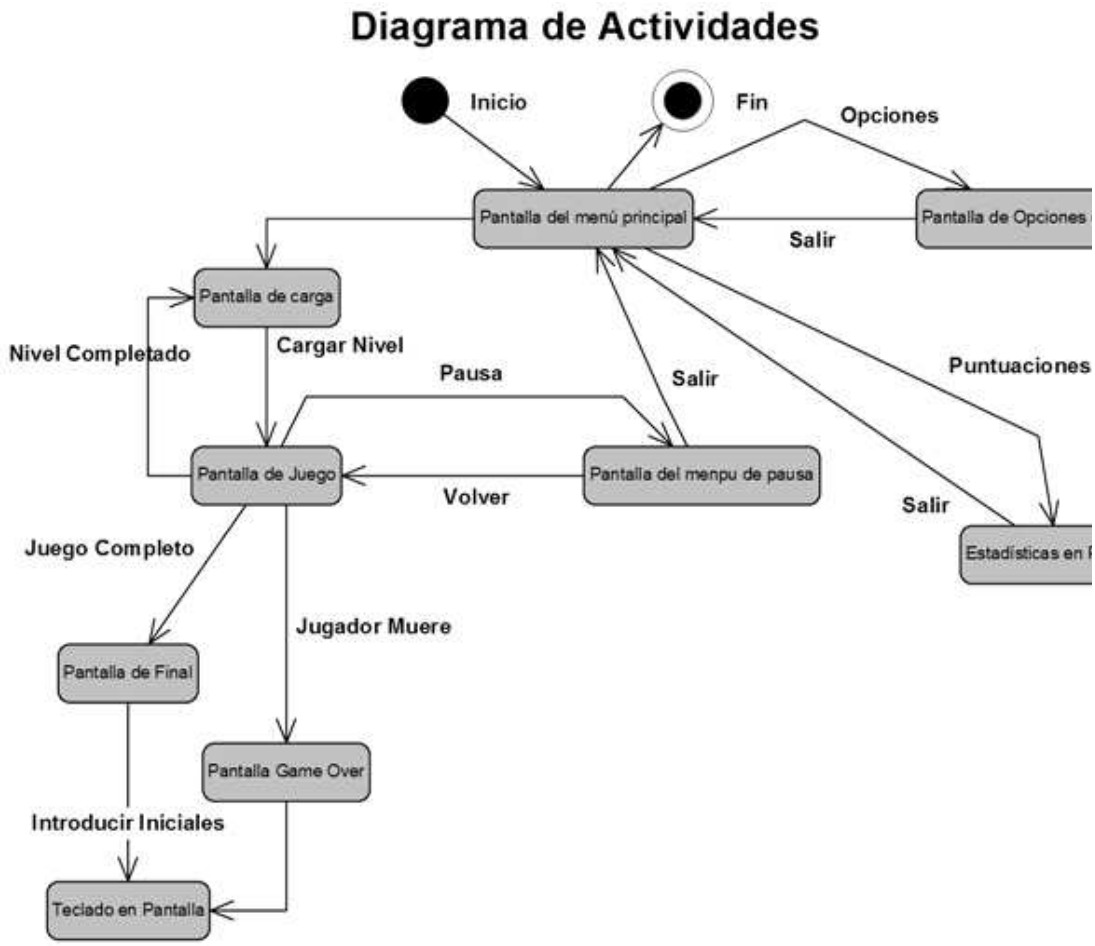


Figura N°35: Diagrama de Actividad

3.4.4 Modelado de Interacción del Sistema (Complejo)

El modelado de interacción del sistema describe como grupos de objetos colaboran para conseguir algún fin. Este modelado muestra objetos, así como los mensajes que se pasan entre ellos dentro del caso de uso. El modelado de interacción del sistema (Complejo) se realizará con los

diagramas de componentes y los diagramas de distribución.

- Diagrama de Componentes

En diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. Los diagramas de Componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema.

El diagrama de componentes del Conflicto Armado, El videojuego se realizó de acuerdo a la estructura de la carpeta de ejecución del mismo.

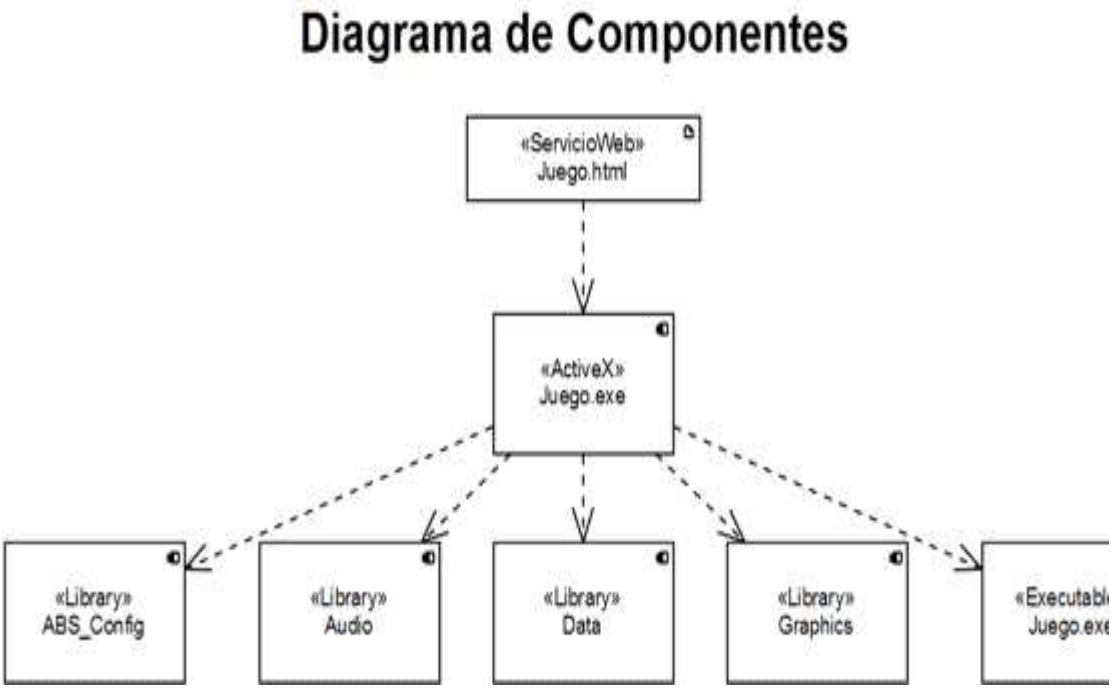


Figura N°36: Diagrama de Componentes

- Diagrama de Distribución

Los diagramas de distribución muestran la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Un nodo es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, que generalmente tiene algo de memoria y, a menudo, capacidad de procesamiento. Los nodos se utilizan para modelar la topología del hardware sobre el que se ejecuta el sistema. Representa típicamente un procesador o un dispositivo sobre el que se pueden desplegar los componentes.

El diagrama de distribución del Conflicto Armado, El videojuego se realizó de acuerdo a una posible versión en html5 que se ejecutaría desde la web,

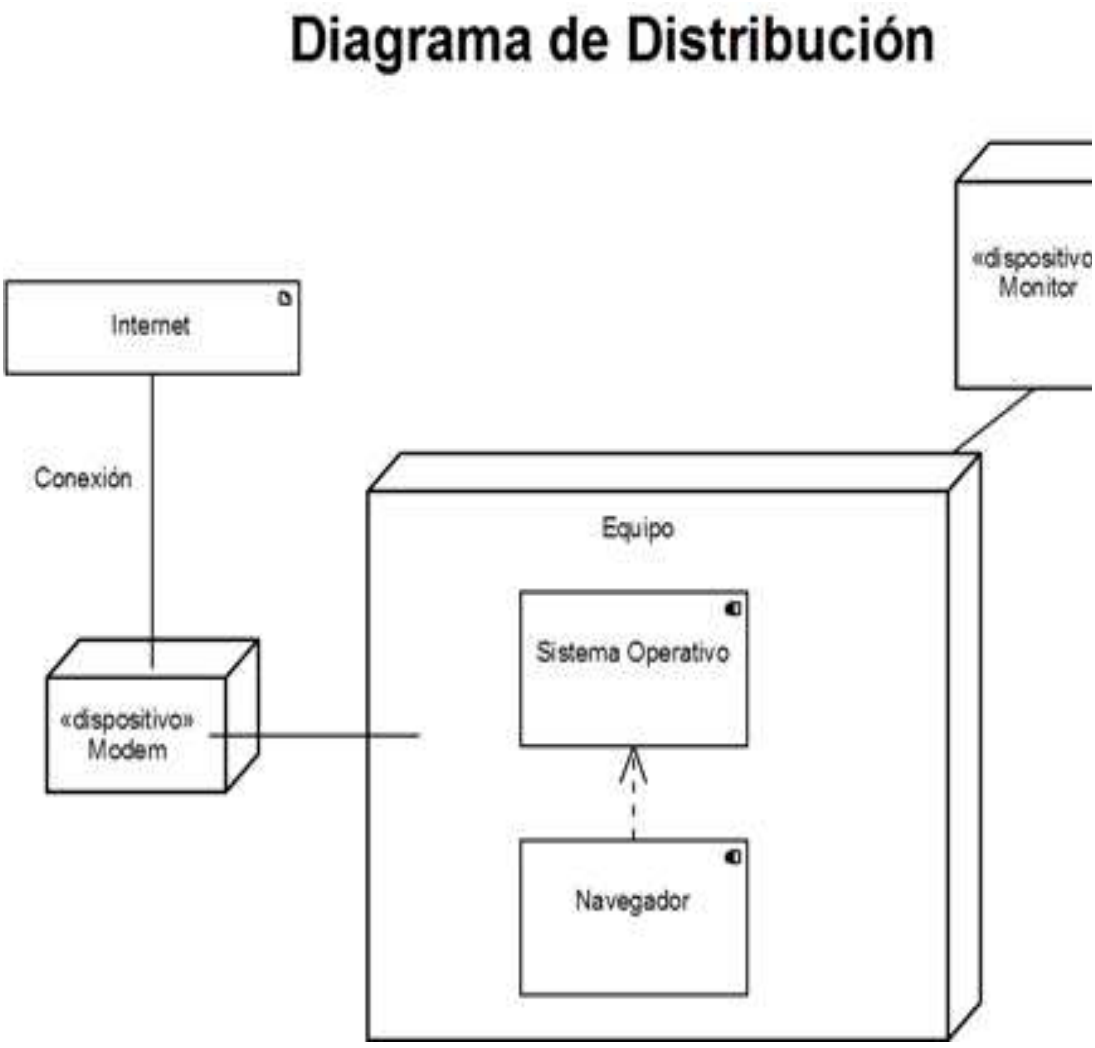


Figura N°37: Diagrama de Distribución

4. DESARROLLO

4.1 ESPECIFICACIONES TÉCNICAS

4.1.1 Software

Para el desarrollo del software se utilizó un RPG como base; el RPG se llama RPG Maker Vx este se encuentra desarrollado sobre el lenguaje de Ruby. Además se trabajó en el sistema operativo Windows.

Para el desarrollo de los charas (personajes) se utilizó el programa Chibi carácter Creador Programa que permite crear charas masculinos t femeninos este se encuentra en la página http://www.famitsu.com/freegame/tool/chibi/index1.html, además se utilizó Character Creator Studio V2 para convertirlos en charas aceptados por el RPG Maker Vx.

Para el desarrollo de las caras de los personajes se utilizó Face Maker, para hacer un dibujo animado del personaje, además se utilizaron los charas y con la ayuda del programa Photoshop se recortaron, rotaron y se hicieron al tamaño correspondiente del RPG Maker Vx.
Para el desarrollo de los diagramas se utilizaron varios programas ya que uno solo no nos permitió realizarlos de la mejor manera, los programas manejados fueron Pacestar UML Diagrammer y StarUML.

4.1.2 Hardware

El videojuego fue realizado en un equipo portátil con las siguientes especificaciones.

- Samsung Ultrabook NP530U4C
- Color: Plateado
- Memoria Ram: 6GB
- Memoria caché: 3MB

- Disco duro: 1TB
- Procesador Intel Core i5 3317U
- Velocidad procesador: 1,70 GHz
- Pantalla LCD con retroiluminación LED de 14"
- Tarjeta gráfica: Intel HD Graphics 4000
- Lector de tarjetas 3 en 1
- Conexión Wi-Fi y conectividad Bluetooth
- Cámara web y micrófono integrada
- Sistema operativo Windows 8 (64 bits)

5. GLOSARIO

- Videojuego: Es un dispositivo electrónico que, a través de ciertos mandos o controles, permite simular juegos en la pantalla de un televisor, una computadora u otro dispositivo electrónico.

- Ítem: Es un elemento típico en los videojuegos cuya función es otorgar una mejora temporal o definitiva al jugador en alguna de sus cualidades ya se de manera positiva o negativa.

- Power up: Término similar a ítem, hace referencia a los elementos que el jugador puede encontrar en el juego y que afectan de manera positiva i negativa a alguna de sus cualidades.

- RPG: Siglas de Role Playing Game, hace referencia al género de los videojuegos que usa elementos de los juegos de rol tradicionales.

- Scroll. En un videojuego, se denomina Scholl al desplazamiento en

2D de los gráficos que conforman el escenario. Se puede hablar de “scroll horizontal” cuando la acción se desarrolla horizontalmente, “scroll vertical” cuando se desarrolla verticalmente y “scroll parallax” cuando se mueven dos o más planos de scroll para dar una cierta sensación de profundidad al videojuego.

- Modder: Un mod en informática es cualquier tipo de cambio a algún programa, mejorándolo o modificándolo respecto a la forma original del mismo. Un modder es aquel que realiza dicho cambio. En los videojuegos, sobre todo en los de PC, son muy típicos los mods que modifican la funcionalidad del juego, por ejemplo añadiendo nuevas capacidades a los personajes.

- Videojuego comercial.: Se consideran videojuegos comerciales aquellos publicados, producidos o simplemente respaldados por grandes firmas dedicadas al ocio electrónico como Nintendo, SEGA, Sony, Microsoft, Konami etc y cuyo desarrollo es totalmente profesional.

- Videojuego casual: El videojuego casual u ocasional es aquel que está dirigido al grupo de jugadores no tradicionales, usuarios relativamente nuevos, que dedican pocas horas al juego, o que conciben los videojuegos como una forma adicional de ocio. En cuanto a temática, los juegos casuales suelen ser juegos deportivos, sociales o de lógica mental. Presentan reglas simples y jugabilidad fácil, siendo el objetivo principal brindar una experiencia del tipo "pick up and play" (poner y jugar), orientándose a usuarios de cualquier edad y nivel de habilidad.

- Videojuego bajo demanda: Sistema de publicación de videojuegos

basado en la descarga. Al usuario no se le proporciona un dispositivo físico, si no que tras adquirir el juego lo descarga y lo almacena en un dispositivo (generalmente disco duro) asociado a la consola. Suelen ser juegos con varios años, o de consolas de generaciones anteriores.

- Videojuego arcade: Arcade es el término genérico que se aplicaba a las antiguas máquinas recreativas de videojuegos disponibles en lugares públicos de ocio o salones recreativos. Actualmente, se ofrecen muchos de estos juegos en las plataformas online de descarga de las consolas de última generación, por lo que dichos juegos también son identificados con el término arcade. A su vez, y por compartir espacio de descarga, los juegos no comerciales que se ofrecen a través de estas plataformas online también son conocidos como juegos arcade. Muchos, son nuevas versiones de estos juegos antiguos, con mejoras técnicas, gráficas o de jugabilidad.

- Videojuego indie. Este término está asociado a la comunidad de desarrolladores de XBOX 360 y hace referencia a los juegos publicados por desarrolladores amateur a través de la plataforma online de dicha consola. Están sujetos a condiciones especiales de publicación y venta ya comentadas en este documento.

- Videojuego de disparos en primera persona: También conocido como FPS, de inglés “first person Shooter”, o simplemente Scooter; es un género de videojuegos y subgénero de los videojuegos de disparos que se desarrolla desde la perspectiva del personaje protagonista y en la que, típicamente, sólo se ven las manos del personaje.

- Mando Analógico: El término hace referencia a los controles añadidos

a los gamepad de las consolas similares a los joystick. Fueron introducidos en los mandos con el auge de las 3D, puesto que permiten controlar a los personajes en escenarios en tres dimensiones con mucha mayor precisión que las crucetas. La mayoría de los mandos actuales incluyen dos analógicos: uno que sustituye la funcionalidad de la cruceta y otro que sustituye parte de la funcionalidad de los botones de acción.

- Sandbox. Género de los videojuegos, también conocido como acción-aventura, que se caracterizan por permitir que el jugador pueda hacer lo que él quiera, viajar libremente por el mapa e interactuar con casi todo lo que este a su disposición sin tener que seguir una línea argumental prefijada. Estos juegos son una mezcla de géneros, entre ellos disparos, luchas y carreras.

- Motor de Videojuegos: una serie de rutinas de programación que permiten el diseño, la creación y la representación de un videojuego, La funcionalidad básica de un motor es proveer al videojuego de un motor de renderizado para los gráficos 2D y 3D, motor físico o detector de colisiones, sonidos, scripting, animación, inteligencia artificial, redes, streaming, administración de memoria y un escenario gráfico.

6. CONCLUSIONES

Con la realización de este proyecto, hemos aprendido a utilizar el RPG Maker VX, así como a manejar, ver y programar con el lenguaje Ruby, que hasta el momento era un lenguaje desconocido para nosotros. Se encontraron diferentes características notables que se pueden destacar del aplicativo

implementado en el proyecto, la facilidad para crear y manejar eventos que gracias a estos se permite la producción de las acciones básicas y complejas de cada personaje involucrado en el mismo, además la importación y exportación de imágenes y sonidos para componer y manejar en el videojuego.

RPG Maker Vx no posee un editor para programar decentemente, no permite realiza debugs ni mostrar los errores de programación por lo tanto la programación de los scripts a utilizar es complicada y dificultosa. Aun así se ha utilizado un nuevo script para visualizar el mapa, para el inicio del videojuego y la forma de pelear de los personajes.

En la culminación del software se observa como al final se diseñó un juego de estrategia donde el jugador (usuario) por medio de recursos ejército y estrategias lograr culminar la misión propuesta. Al visualizar como los usuarios interactúan con el videojuego se observa que la interfaz es adecuada y el usuario debe generar estrategias para la inteligencia artificial que tienen sus enemigos.

Este proyecto con un poco más de tiempo se puede llegar a realizar algo más espectacular, ya que existe una versión en 3d para RPG Maker u otros makers que se podría utilizar para un futuro proyecto similar. Los objetivos establecidos previamente citados se completaron correctamente.

7. BIBLIOGRAFÍA

- Booch Grady. Jacobson Ivar. Rumbaugh James. (2000). El lenguaje unificado de modelado. Manual de referencia. Edición en español.

España. Addison Wesley,

- Pressman, Roger S. (2002). Ingeniería del Software: Un enfoque práctico. Quinta edición. España. Mc Graw Hill.

- Unified Modeling Language. UML: disponible en URL: http://www.uml.org [Consulta en Mayo 15 del 2013]

- BenKo. (Diciembre 8, 2008). Guía “Beginner’s Guide To Rpg Maker VX”.

- Yuki Hayabusa, (Julio 17, 2008). RPG Maker VX. Disponible en URL: http://tkool.jp/products/rpgvx/spa/index.html. [Consulta en Mayo 15 del 2013]

- RPG Recolution. RGSS Script Reference > Class Hierarchy. Disponible en URL: http://www.rpgrevolution.com/rgss-script-ref/class-hierarchy.html [Consulta en Mayo 15 del 2013]

- Foros Comunidad RPG Maker. Disponible en URL: http://rpgmaker.es/ [Consulta en Mayo 15 del 2013]

- Foros Comunidad RPG Maker VX. Disponible en URL: http://www.rpgmakervx.net/ [Consulta en Mayo 15 del 2013]

- Ánemus. Guía de Ruby en base a RPG Maker XP (RGSS)

- Mundo RPG Maker Lo mejor en recursos y portes. Disponible en URL: http://mundo-rpgmaker.jimdo.com/ [Consulta en Mayo 15 del 2013]

Manual de Usuario

CONTENIDO

Pág.

- **INSTALACIÓN**

El Juego Conflicto Armado, EL VIDEOJUEGO no precisa de instalación, es posible jugar el juego en cualquier PC Windows la velocidad dependerá de la capacidad del PC.

Una vez deseo jugarlo, bastará con ejecutar el Archivo Game.exe para lanzar el juego.

- **PANTALLAS**

Una vez lanzado el Game aparecerá el Menú Principal. Para desplazarse por

los menús deben utilizar las flechas de desplazamiento. Para seleccionar una opción de un menú se utiliza la tecla Enter. Para salir de cualquier menú se utiliza la tecla Escape. Las opciones que se presentan en el Menú Principal son:

- Menú
- Cargar
- Salir

La opción Menú lanza el Inicio del Juego, Cargar abre el menú de partidas guardadas y salir cierra el juego.

Figura N°1: Menú Principal

Desde Menú es posible modificar los siguientes atributos del juego antes de comenzar una nueva partida:

- Campañas: Le permite al usuario escoger entre campaña de aprendizaje, Ejército y Grupo Armado.

Figura N°2: Opciones Menú Principal

- OPCIONES: le permite al usuario escoger la dificultad, personalizar al jugador (Hombre o Mujer) y cancelar la acción.

Figura N°3: Opciones Menú OPCIONES

Desde el menú de cada campaña permite realizar las siguientes acciones:

- Iniciar: Lanza una nueva partida del juego.
- Descripción de la campaña: Lanza una pequeña descripción de la campaña que ha decidido jugar.
- Cancelar: Cancela la acción ejecutada del momento.

Figura N°4: Opciones Menú de Cada Campaña

Al comenzar una nueva partida si el jugador oprime la tecla escape le permitirá modificar o visualizar los siguientes atributos:

Figura N°5: Opciones en el Juego

- Objetos: Le permitirá al jugador observar los objetos que tiene en su poder y si los desea equipárselos.

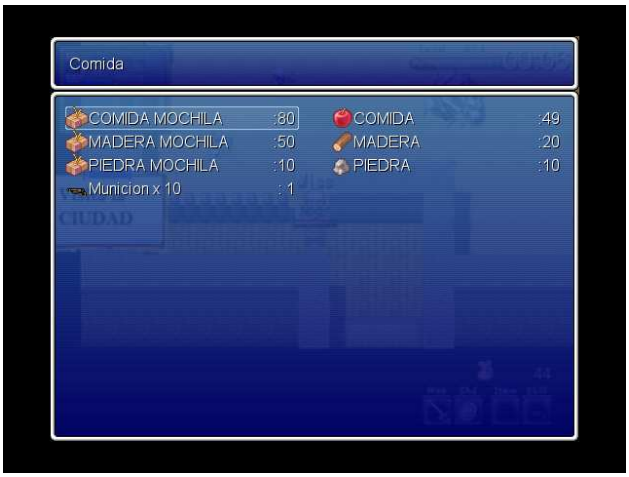


Figura N°6: Menú Objetos

- Técnicas: Le permitirá al jugador observar las técnicas aprendidas durante el juego y equiparlas para su beneficio.

Figura N°7: Menú Técnicas

- Equipamiento: Le permitirá al jugador equipar las armas, escudos, cascos, armaduras y accesorios encontrados en el juego o comprados en la tienda.

Figura N°8: Menú Equipamiento

- Estado: Le permitirá al jugador ver todo el estado del jugador, el nivel en el que se encuentra, el nombre del jugador principal, La experiencia que ha adquirido, cuanta experiencia falta para pasar al siguiente nivel, con que equipamiento cuenta en ese momento, y los niveles de ataque, defensa, inteligencia y agilidad que posee.

Figura N°9: Menú Estado

- Guardar: Le permitirá al jugador visualizar las partidas guardadas y así mismo seleccionar alguna para continuar.

Figura N°10: Menú Guardar

- Fin del Juego: Le permitirá al jugador salir al menú principal, salir del

juego y volver a la partida.

Figura N°11: Menú Salir

- **PANTALLA DE JUEGO Y NIVELES**

Como ya se ha indicado si se selecciona la opción iniciar en el Menú de Campañas, comienza una nueva partida, El juego contará con 3 niveles el primero será de entrenamiento, el segundo será una batalla siendo tu parte del ejército y el tercero será una batalla siendo tu parte del grupo armado. Si se dominan los controles del juego se recomienda no realizar la campaña de aprendizaje y pasar a alguna de las otras dos de una vez. A continuación se muestra la pantalla de juego, señalando con número cada uno de los valores mostrados.

Figura N°12: Pantalla de Juego Enumerada

- *HP: Muestra la vida del Jugador al transcurrir el juego.*
- *MP: Muestra la energía para realizar técnicas al transcurrir el juego.*
- *Misiones: Se habilitan misiones principales y secundarias, las principales deben realizarse en su totalidad para ganar el nivel, las secundarias se deben a decisiones morales del jugador y está en él si las cumple o no.*
- *Level: Muestra el nivel en el que se encuentra el jugador al transcurrir el juego.*
- *UI: Muestra la experiencia obtenida al transcurrir el juego.*

- *Oro: Muestra la cantidad de oro que posee en el transcurrir del juego.*
- *Ítems: Muestra los elementos equipados en cada momento por el jugador.*

- **CONTROLES**

Por último se muestran los controles en juego Game.exe. Si el jugador maneja al personaje con el teclado los controles serán los Siguientes:

- Flechas de dirección: Mueven al jugador arriba/abajo/izquierda/derecha.
- Shift: Permite al Jugador correr.
- Escape o X: Muestra las opciones en juego.
- A: Al tener equipado el escudo permite utilizarlo en Juego.
- D: Al tener equipada la técnica permite utilizarla en Juego,
- Enter, Espacio o Z: Al tener equipada el arma permite utilizarla en juego.
- S: Al tener equipados elementos (balas, pociones de vida, etc…) permite utilizarlos en juego.

Manual de Sistema

CONTENIDO

Pág.

- **INTRODUCCIÓN**

- **Objetivo**

Otorgar soporte a los usuarios del videojuego, teniendo un control e información oportuna del sistema.

- **Requerimientos**
  - Equipo Pentium II o superior
  - 2Gb de Memoria RAM
  - 1Gb de Disco Duro
  - 512Mb de Tarjeta Gráfica