**MANUAL TÉCNICO**

SOFTWARE RECREATIVO CON APLICACIÓN DE CONCEPTOS ENSEÑADOS EN PROGRAMACIÓN ORIENTADA A OBJETOS



SANTIAGO GRISALES ZAMORA

YONY ALEJANDRO PEÑA

CARLOS FELIPE PALACIO

2021

**ÍNDICE**

**Portada…………………………...………………………………………………………..1**

**Índice……..………………………………………………………………………………..2**

**Introducción…………..…………………………………………………………………..3**

**Objetivo General y específico del sistema..……………………………………..….4**

**Procesos………………………………………………………………………………......5**

**Herramientas para el desarrollo………...……………………………………………..6**

**Requerimientos…………...…………………………………………….………………..7**

**Modelo UML……………………………………………….……………………………....8**

**INTRODUCCIÓN**

El siguiente software recreativo es un juego creado en Unity que se basa en la historia del caballero Herz, el cual trata de buscar la cura a una enfermedad que se originó en su pueblo natal, por lo cual decide ir a buscarla dentro de una mazmorra de tipo laberinto en el norte y en la que se encontrará monstruos y otras sorpresas en el camino.

El juego se llama The last hope’, y trata de probar la habilidad motriz que tienen las personas para dirigir a un personaje en distintos lugares de la mazmorra en los que se encontrará enemigos y con los que tendrá que luchar hasta conseguir encontrar al último de los monstruos y el más fuerte, el cual tendrá el acceso al único antídoto que podría curar la vida de la gente de su pueblo.

El software no necesita de mucha inteligencia y habilidad para jugarlo, pero en cambio nos ofrece una cantidad variada de escenarios, entidades, ítems y demás cosas que nos permiten tener una excelente experiencia de juego.

**OBJETIVO GENERAL**

Divertir al usuario mediante el uso de interfaz que sea agradable a su vista y que permita que la historia del personaje principal pueda llevarse a cabo, además se espera que la habilidad del usuario pueda ser puesta en práctica al encontrar el camino hacia el antídoto mediante el uso del laberinto.

**OBJETIVO ESPECÍFICO**

Llamar la atención de los usuarios para que se entretengan extraordinariamente con nuestro juego gracias a la cantidad de eventos que pueden ocurrir antes de encontrar al jefe final y salvar a su pueblo.

**PROCESOS**

**De entrada**

1. Ingresa e inicia el programa desde la portada.
2. Selecciona el nivel que quiere jugar.
3. Utiliza las flechas y la tecla ‘Intro’ del teclado para moverse y escoger diálogos.
4. Usa la tecla ‘e’ del teclado para abrir el inventario y los uso y se mueve con el clic izquierdo del ratón.

**De salida**

1. Movimiento del jugador.
2. Selección de diálogos, poderes y pociones.
3. Visualización de inicio al juego, de movimiento a través del laberinto, del sistema de combate y el ganar o perder de la partida.

**HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO.**

**C++**

C++ es un lenguaje diseñado para extender el lenguaje C, este fue elegido para cumplir con los requisitos del sistema, se usó con el paradigma de la programación orientada a objetos.

**VISUAL STUDIO**

Visual Studio es un entorno de desarrollo integrado con el cual el desarrollador podrá **crear y desarrollar softwares**como aplicaciones web y móviles, sitios o servicios web en entornos compatibles con la plataforma .NET.

**VISUAL STUDIO CODE**

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

**GITHUB**

GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador.

**UNITY**

**Unity es lo que se conoce como un motor de desarrollo o motor de juegos**. El término **motor de videojuego, game engine, hace referencia a un software**el cual tiene una serie de rutinas de programación que permiten el diseño, la creación y el funcionamiento de un entorno interactivo; es decir, de un videojuego.

**REQUERIMIENTOS**

**No técnicos**

1. Polimorfismo, herencia, sobreescritura y sobrecarga.
2. Al menos una clase abstracta.
3. Separación de responsabilidades.
4. Manejo de excepciones.

**Opcionales**

1. Usar alguna librería para que el videojuego se vea en ventana gráfica.
2. Implementar más mecánicas y sistemas creativos para darle un mejor sentido al juego rpg.

**Mínimos**

1. Seguimiento con GitHub.
2. Estándar de codificación lowerCamelCase.
3. Identación y organización del código decente.
4. Código documentado y entendible para cualquier lector.
5. Buenas prácticas de programación.

**PARTES DEL JUEGO**

**Se acaba el juego**

**Interfaz de usuario gráfica

Descripción generada automáticamente**

**Sistema de combateForma, Rectángulo

Descripción generada automáticamente**

**Durante el juego.**

**Imagen que contiene Diagrama

Descripción generada automáticamente**

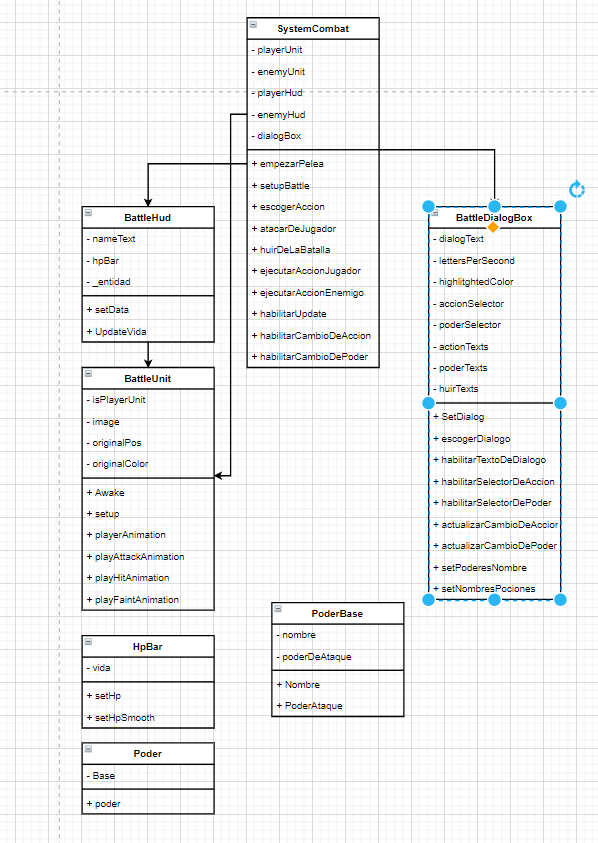
**Durante el juego.**

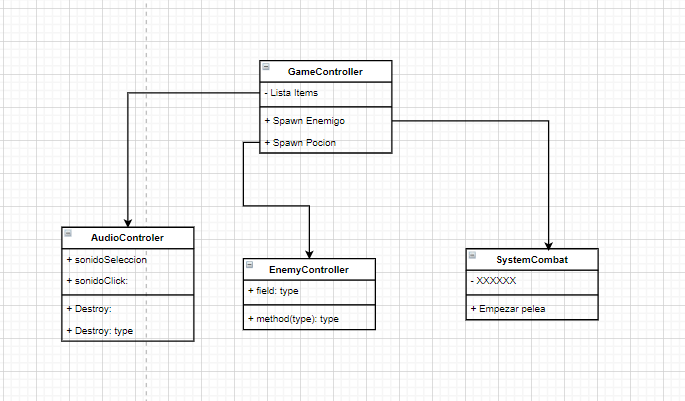
**Interfaz de usuario gráfica, Gráfico, Gráfico de cajas y bigotes

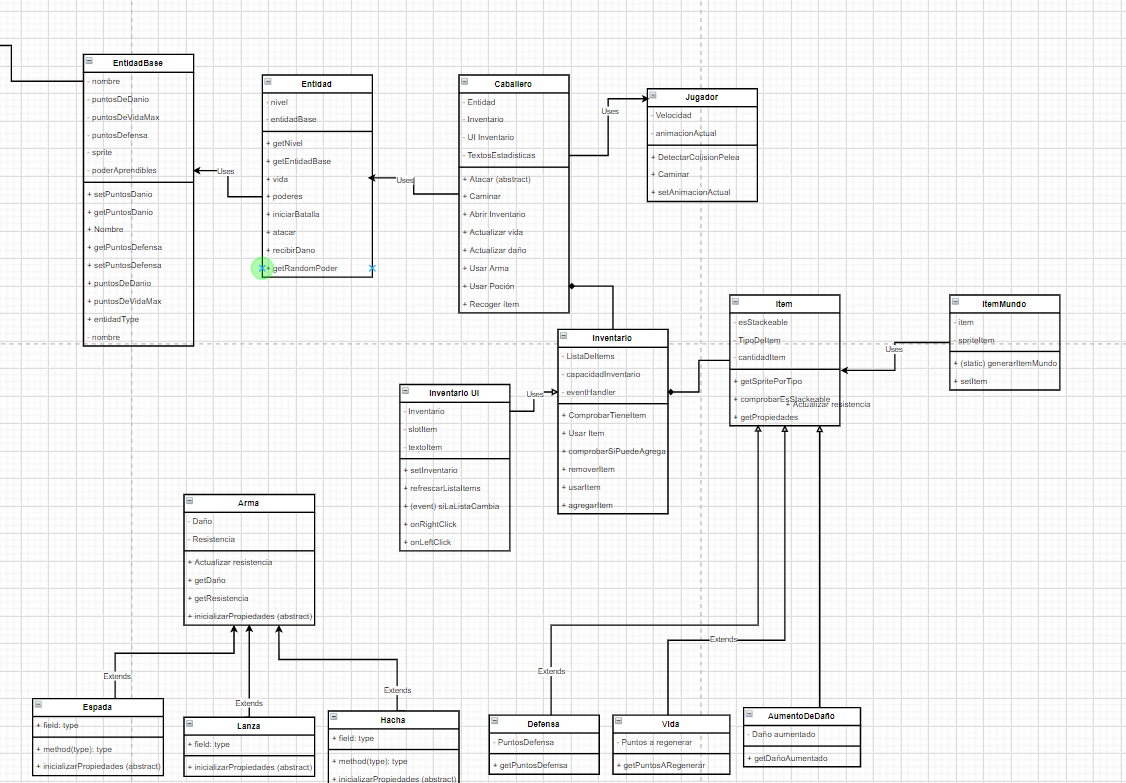
Descripción generada automáticamente**

**DISEÑO UML**

**Link:** https://app.diagrams.net/#G1lOifvho4x179TrDvg3RLwI4SqOs\_RppN

****

****

****