Zombie Defense

Juan Sebastián Gamboa Botero, David Suárez Acosta

Instituto Tecnológico de Costa Rica, Ingeniería en Computación, Programación Orientada a Objetos.

***Abstract*—This is a game of strategy where three characters, controlled by a player, have to protect a castle and survive to the different monster hordes that will increase in difficulty with the game progression. Four types of enemies (Skeletons, Zombies, Slimes and Ghosts) will attack the castle or attack the first player character in their sight, each of them will also have different abilities as well as damage and health variations.**

The three characters the player can control will have their own special qualities such as more damage and more movement, and in each turn a character can move, attack and equip or use an item, be careful because some weapons can make sounds that will be heard by the enemies. Finally, the map will have different obstacles the player and enemies will have to avoid in order to move into a place, and you can get items from chests in it or by killing an enemy.

I. INTRODUCCIÓN

Este proyecto consiste en crear un juego de estrategia en donde un jugador controla a tres personajes y con estos deberá defender un castillo y sobrevivir a las diferentes hordas de zombies que intentaran llegar a la base. Tanto los personajes del jugador como los enemigos tienen habilidades diferentes que modificará sus cualidades y el jugador podrá moverse, atacar o usar items. Los enemigos tendrán un diferente orden de prioridad y es atacar a un jugador, acercarse a un jugador si está en su rango de visión, ir hacia un sonido o ir hacia la base.

El juego se realiza en un mapa de casillas donde las casillas pueden tener paredes, ser inaccesibles o tener otra cualidad como ser una plataforma elevada, y tiene que haber tanto una base para los personajes del jugador como lugares donde los enemigos van a aparecer. Por último, para este proyecto se necesita utilizar una interfaz gráfica que muestre el tablero, los personajes e información importante y se necesita usar GitHub para la transferencia de código y archivos.

Los objetivos del proyecto son: aplicar conceptos de ingeniería de software, utilizar herencia para el diseño de clases. Además, se ocupa tener una documentación tanto interna, que explique la función de los algoritmos, como externa. A continuación, se mostrará la solución que se realizó para cumplir con los objetivos del proyecto, así como las dificultades que aparecieron y el progreso que se tuvo a lo largo de su desarrollo.

II. DISEÑO

A. Diagrama de clases

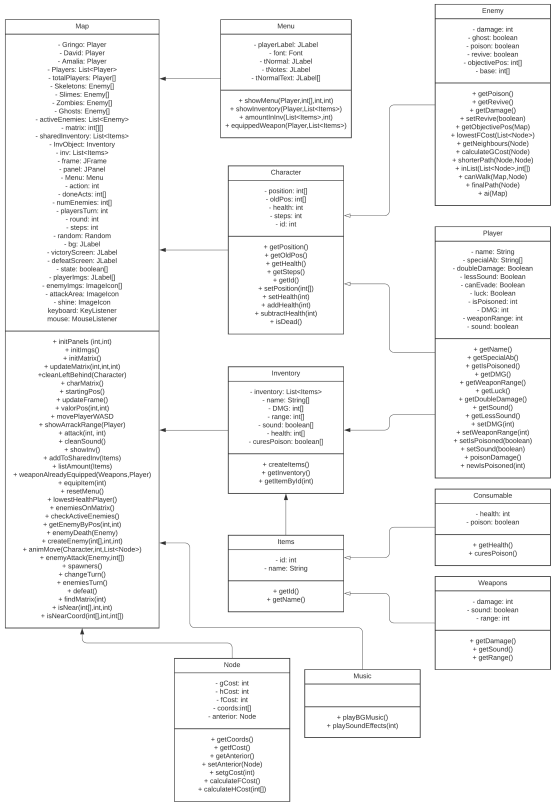

Fig. 1. Diagrama de clases UML para la estructura de las clases del proyecto.

B. Diagrama de casos de uso

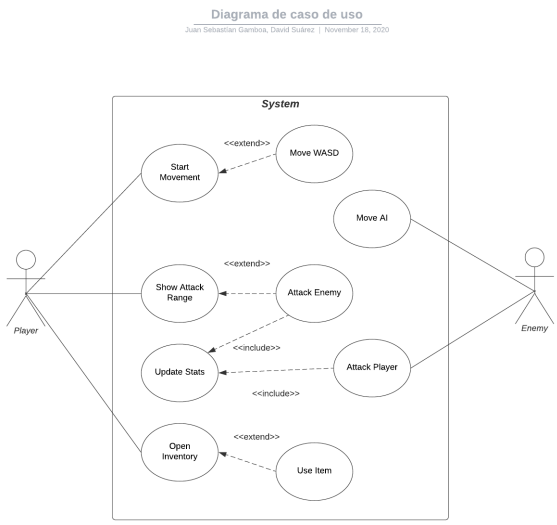

Fig. 2. Diagrama de casos de uso para la interacción Jugador-CPU con el sistema.

C. Cronograma

Activity	Dates																							
	26 Oct	27 Oct	28 Oct	29 Oct	30 Oct	31 Oct	1 Nov	2 Nov	3 Nov	4 Nov	5 Nov	6 Nov	7 Nov	8 Nov	9 Nov	10 Nov	11 Nov	12 Nov	13 Nov	14 Nov	15 Nov	16 Nov	17 Nov	18 Nov
Concept (Art)																								
Concept (Players)																								
Concept (Enemies)																								
Concept (Map)																								
UML Diagram of Classes																								
Classes Structure																								
Map Matrix																								
Menu																								
Mouse & Action Listener																								
Turn																								
AI																								
Player's Movement																								
Player's Attack																								
Player's Abilities																								
Player's Death/Revive																								
Inventory/Items																								
Spawners/Rounds																								
Sound																								
Victory/Defeat																								
Music																								

Fig. 3. Cronograma de actividades de Kanban, hecho en Github.

D. Diagrama de secuencia

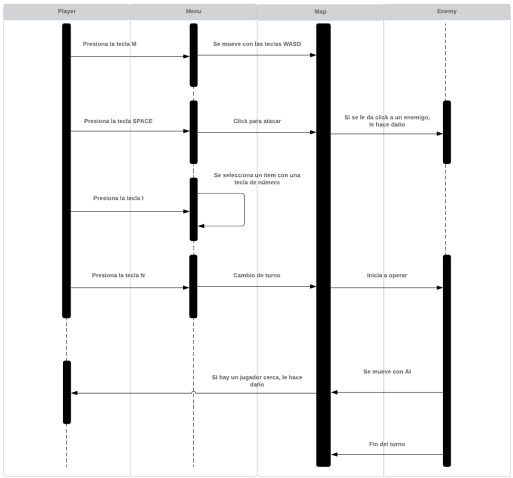

Fig. 4. Diagrama de secuencia para la visualización de un turno y las diferentes interacciones.

E. División de actividades

GUI swing: Gamboa
Clase Map: Gamboa, Suárez
Clases Character y Player: Gamboa
Clases Items, Weapons y Consumables: Gamboa
Clases Inventory y Menu: Gamboa
Diseño gráfico (Menu, victoria/derrota): Gamboa
Clase Node: Suárez
Creación del Mapa: Suárez
Edición de personajes: Gamboa, Suárez
AI y clase Enemy: Suárez
UML, Diagrama de casos de uso y Diagrama de secuencias: Suárez
Documentación: Suárez

III. MÉTODOS

Para este proyecto se hicieron ciertos cambios o adaptaciones para lograr un desarrollo de juego entretenido y balanceado. El juego consiste en sobrevivir a una cierta cantidad de hordas de enemigos, estos enemigos, además de incluir zombies, tienen esqueletos, slimes y fantasmas, y cada uno de estos tipos tiene diferentes cualidades. Los esqueletos son los enemigos básicos, tiene un daño y vida promedio; los slimes tienen poco daño y vida, pero tienen la habilidad de revivir, es decir que no importa cuanto daño se le inflija con un ataque, este se muere después de mínimo dos ataques; los zombies tienen más vida y daño que un esqueleto, tienen la habilidad de envenenar a los jugadores, este veneno les hará daño cada cambio de ronda y se quita después de un tiempo o con una opción especial; los fantasmas son los enemigos más fuertes de vida y daño, además son los únicos personajes que pueden atravesar objetos y por ende llegar más rápido a un objetivo.

A su vez, hay tres tipos de personajes de jugador, Gringo que hace doble daño y no crea sonido, David que puede moverse una mayor distancia y tiene más probabilidad de obtener ítems de los enemigos muertos y Amalia que tiene más vida y tiene una probabilidad para esquivar un ataque. Hay diferentes tipos de armas con su propio daño y rango, y que podría o no hacer ruido, igualmente hay pociones de curación, para quitar veneno o para revivir un personaje muerto. El juego se realiza en un mapa hecho con assets de internet [1] y se utilizó la aplicación Tiles [2] para dibujarlo y exportarlo como imagen a Java, esto se hizo con base a un video tutorial de YouTube [3]. Este mapa se manipuló en el código con una matriz en donde cada espacio es un espacio de coordenadas del mapa.

Para la realización de este proyecto, se dividió el proceso en las clases Map, Menu, Inventory, Node, Music, Character, Player, Enemy, Items, Weapons y Consumable, cada una de las cuales tiene su propio objetivo. Map es la clase central que funciona como una clase que controla el juego y las demás clases, esta tiene las instancias principales, funciones más importantes y el main. Esta clase contiene la matriz del mapa y en esta se manejó todo lo relacionado con la interfaz gráfica del mapa. En la clase de Menu se tiene lo relacionado con la información y opciones de acciones que puede realizar un jugador con un personaje, en esta también se manejó la interfaz y se visualiza en un espacio del mapa listo para lo mismo. Character, Player y Enemy son clases que utilizan herencia ya que Character es la superclase y la cual le transmite atributos y funciones que Player y Enemy comparten, como vida, posición o getters y setters. Player contiene las cualidades que cada personaje de jugador va a tener, como las variaciones en sus cualidades dependiendo de las habilidades. Enemy, al igual que Player, maneja las habilidades y cualidades de los diferentes enemigos, pero además tiene todas las funcionas relacionadas con el AI. Items, Weapons y Consumables también utiliza herencia con Items como la superclase y en general se manejan los ítems y las diferentes cualidades que cada uno va a tener, también dependiendo del tipo de ítem. Las clases de Inventory, Node y Music son clases adicionales que son usadas como apoyo para realizar estructuras o funciones, Inventory permite realizar el sistema de guardado de ítems, Node permite un manejo de listas dinámicas, proceso principalmente necesario para el AI y Music maneja el sonido y musica del juego.

El AI se basó en un método conocido de Inteligencia Artificial llamado A* PathFinding que funciona buscando el camino más corto en un tablero, para esto se usó como base un video tutorial que explica la teoría del método [4] y un video que muestra un ejemplo de código de A* PathFinding para Unity [5]. El método también incluía el uso de listas dinámicas, por lo que se usó como base un video tutorial [6].

Otros aspectos importantes incluyen el uso de referencias para el trabajo. Texto multilíneas en una JLabel [7], música con swing [8], bajar el volumen de swing [9], video explicando la música con swing [10], el uso de Random en Java [11] y el keylistener y mouselistener [12]. Para el manejo de archivos y organización del trabajo, se utilizó GitHub y GitHub Desktop, el link para ingresar al repositorio y al código es el siguiente: https://github.com/JGamboaB/POO_Proyecto2.git

IV. CONCLUSIÓN

Para este proyecto se programó en Java aplicando conceptos de la ingeniería de software, se puso en práctica el paradigma orientado a objetos ya que se realizaron diferentes clases, cada una de estas enfocada en objetos diferentes, se hizo una interfaz gráfica por medio de swing y se utilizó la herencia en dos casos: la superclase Character que le hereda atributos básicos como posición y vida a las clases Player y Enemy, y la superclase Items que le hereda a las clases Weapons y Consumables.

Durante el desarrollo del trabajo se hicieron adaptaciones para que el juego y sus mecánicas tuvieran relación con la temática que se escogió y para el manejo del mapa, se transformó en un una matriz para ser manejado como un arreglo.

REFERENCIAS

[1] ”Zelda-like tilesets and sprites”, OpenGameArt.org, 2020. [Online]. Available: https://opengameart.org/content/zelda-like-tilesets-and-sprites. [Accessed: 18- Nov- 2020].

[2] ”Tiled Map Editor”, Tiled Map Editor, 2020. [Online]. Available: https://www.mapeditor.org/. [Accessed: 19- Nov- 2020].

[3] Youtube.com, 2020. [Online]. Available: https://www.youtube.com/watch?v=ZwaomOYGuYo. [Accessed: 19- Nov- 2020].

[4] Youtube.com, 2020. [Online]. Available: https://www.youtube.com/watch?v=-L-WgKMFuhE. [Accessed: 19- Nov- 2020].

[5] Youtube.com, 2020. [Online]. Available: https://www.youtube.com/watch?v=alU04hvz6L4. [Accessed: 19- Nov- 2020].

[6] Youtube.com, 2020. [Online]. Available: https://www.youtube.com/watch?v=5k6xDsSyVA8t=133s. [Accessed: 19- Nov- 2020].
[7] M. JLabel, S. Lybon and W. Think, "Multiline text in JLabel", Stack Overflow, 2020. [Online]. Available: https://stackoverflow.com/questions/685521/multiline-text-in-jlabel. [Accessed: 19- Nov- 2020].
[8] Using javax.sound.sampled.Clip to play, T. Thomas, T. Thomas and A. Peturis, "Using javax.sound.sampled.Clip to play, loop, and stop multiple sounds in a game. Unexpected Errors", Stack Overflow, 2020. [Online]. Available: https://stackoverflow.com/questions/11919009/using-javax-sound-sampled-clip-to-play-loop-and-stop-multiple-sounds-in-a-game. [Accessed: 19- Nov- 2020].
[9] Java and J. Cajthaml, "Audio volume control (increase or decrease) in Java", Stack Overflow, 2020. [Online]. Available: https://stackoverflow.com/questions/953598/audio-volume-control-increase-or-decrease-in-java. [Accessed: 19- Nov- 2020].
[10] Youtube.com, 2020. [Online]. Available: https://www.youtube.com/watch?v=TErboGLHZGA. [Accessed: 19- Nov- 2020].
[11] G. [duplicate] and S. Wombat, "Generating a Random Number between 1 and 10 Java", Stack Overflow, 2020. [Online]. Available: https://stackoverflow.com/questions/20389890/generating-a-random-number-between-1-and-10-java. [Accessed: 19- Nov- 2020].
[12] Youtube.com, 2020. [Online]. Available: https://www.youtube.com/watch?v=2ZXiuh0rg3Mlist=PLWtYZ2ejMVJkjOuTCzIk61j7XKfpIR74K. [Accessed: 19- Nov- 2020].