**Zaletros**

**Software Design Document**

**Version <1.0>**

*Courtney Higgins*

*Amanda Truong*

*Anna Khachaturova*

2019-05-31

**Table of Contents**

[**1 Introduction**](#_4897ig1ttbpg) **3**

[1.1 Purpose](#_g856x853xooa) 3

[1.2 Scope](#_jd6zoseicim1) 3

[1.3 Definitions, Acronyms, and Abbreviations](#_bxzc8dtw9qga) 3

[1.4 References](#_2xktbcnn9x9h) 4

[1.5 Overview](#_ynasvtxwo2c1) 4

[**2 Use Cases**](#_t2dx4cbmyvhq) **4**

[2.1 Actors](#_xqh3j0eixsvx) 4

[2.1.1 Player](#_l887fotakv06) 4

[2.1.2 CPU](#_7nzd87v21oat) 5

[2.2 List of Use Cases](#_dh58rnh911f1) 5

[2.3 Use Case Diagrams](#_dt2blxnsfcci) 5

[**3 Design Overview**](#_64ets9qh5bws) **7**

[3.1 Introduction](#_x7glu4ndfz7n) 7

[3.2 System Architecture](#_fv4svus6o846) 7

[3.3 System Interfaces](#_2te03im7i6yq) 10

[3.3.1 User Interfaces](#_bfhljsoynr43) 10

[3.3.2 Interface Design](#_hu1bx4wdcz22) 10

[3.4 Constraints and Assumptions](#_4q9a24fgxaoj) 11

[3.5 System Object Model and Collaboration Diagrams](#_qtbc8ryrzfnt) 11

[**4 Data Design**](#_55xmvgl98d1y) **11**

[4.1 File Storage](#_k03wj6mg54v8) 11

[4.2 File Storage Structure](#_1j31qm4c2abg) 11

[**5 Game Design**](#_y9ydfdl9dhk4) **13**

[5.1 Overview of Game Design](#_6ck9hp3itoku) 13

[5.2 Skill Trees](#_ms3zdh8b9asz) 14

[5.3 Gear Design](#_tyknd1ayviga) 15

[5.4 Boss Design](#_vjjlo6ba1s49) 15

[5.5 Map Design](#_ne0o8ncjtaut) 15

[5.6 Character Design](#_6cfwz0z1fn6b) 16

### **1 Introduction**

The Software Design Document is a document to provide aid in the software development by providing details for how the software should be built. Within this document are narrative and graphical documentation to assist in the software design for this project.

#### **1.1 Purpose**

The purpose of the Software Design Document is to provide information on the design of a system fully enough to allow for software development to proceed with and understanding of what is to be built and how it is expected to be built.

#### **1.2 Scope**

This Software Design Document is for the development of a basic game which will work as proof of concept for use of building a game that has basic features with the ability to scale into a larger game. This Software Design Document is focused on the base level system and critical parts of the system and game design. The focus of this Software Design Document is placed on the basic game construction as well as base level features of the game.

#### **1.3 Definitions,** Acronyms, and **Abbreviations**

**Aseprite -**  A program that is used to make sprites with custom animations.

**Attack -** A stat each party member will have that increase the amount of damage that abilities and basic attacks will do.

**Boss -** A certain monster harder than others in the same area, who has more moves and advanced mechanics.

**CD -** Cool down. The amount of turns until an ability can be used again.

**Collision Object** - any object which needs to have physics applied to it. Used to create walls, boundaries or other non-traversable characters in game.

**CPU** - refers to Central Processing Unit, in gaming terminology a CPU is a computer player against which the users plays against.

**Equip -** The action of a user associating an item with a certain character.

**Experience Points -** Points that are award to each character upon winning a battle.

**GDScript** - a high level programming language that is similar to Python. It is used to program functionalities in Godot as well as the controls, memory management and user inputs.

**Gear -** Items that characters will equip that will give them certain stats.

**Godot** - a free and open source engine for create 2D or 3D games. Allows the user to easily animate sprites, create levels and write scripts for the game.

**HP -** Health Points. A stat each party member will have that shows health. When it hits zero, the character will no longer be able to take actions.

**Kinematic Body** - or a character, is a controller used to render a playable sprite as well as apply physics to one.

**Node** - a building block within a scene upon which the user builds a level or a menu.

**NPC -** Non-playable character.

**Potency -** The amount of damage an ability will do before attack is considered

**RPG -** Role-Playing Game

**Scene** - can refer to a level or a menu of a game. A scene is the current layer at which the user is looking at. A game or a project, can be composed of multiple scenes for each layer of the game. Typically, a game made in Godot is made out of a tree of scenes.

**SDD -** Software Design Document.

**Sprite -** A model in the game for characters, bosses, NPC’s, and monsters.

**Tilemap** - a set of elements which are used to build out a level on the scene. Each tile or a piece of a map can be added to the map as many times as needed. This makes the level design faster.

**Z-Index** - a value that is used to organize the order of elements on the scene.

#### **1.4 References**

**Godot Docs** - <https://docs.godotengine.org/en/3.1/index.html>

**Zaletros Proposal -** [https://docs.google.com/document/](https://docs.google.com/document/d/1TtA6PdXuOfqJEdItpVGXUJthwhGg0hqmycuJRh69hdE/edit?usp=sharing)

#### 1.5 Overview

The design document for Zalteros game is broken down into 5 sections. The sections of the Game Design Document are:

1. Introduction
2. Use Cases
3. Design Overview
4. Data Design
5. Game Design

### 2 Use Cases

#### 2.1 Actors

##### 2.1.1 Player

As this is a single player game, there will be one actor per game session, and the actor will be playing against computer bosses. The actor or the player will be in control of the four characters whose activity they will control in game.

##### 2.1.2 CPU

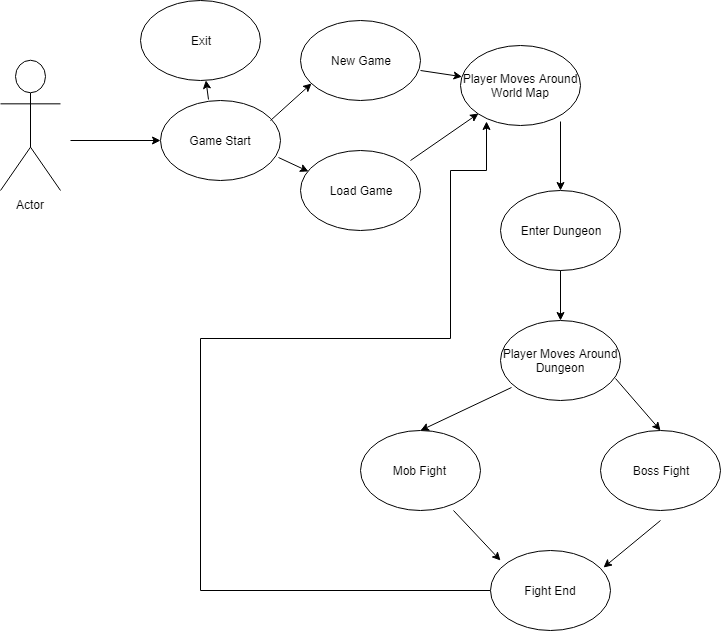
In the game, the cpu will be responsible for controlling boss and mob fights. For each cpu controlled character, there will be a predefined set of moves depending on players statistics. The cpu character will either be random through a random mob generated in a dungeon or world map, or through player actor entering boss area and continuing to fight the cpu boss character.

#### 2.2 List of Use Cases

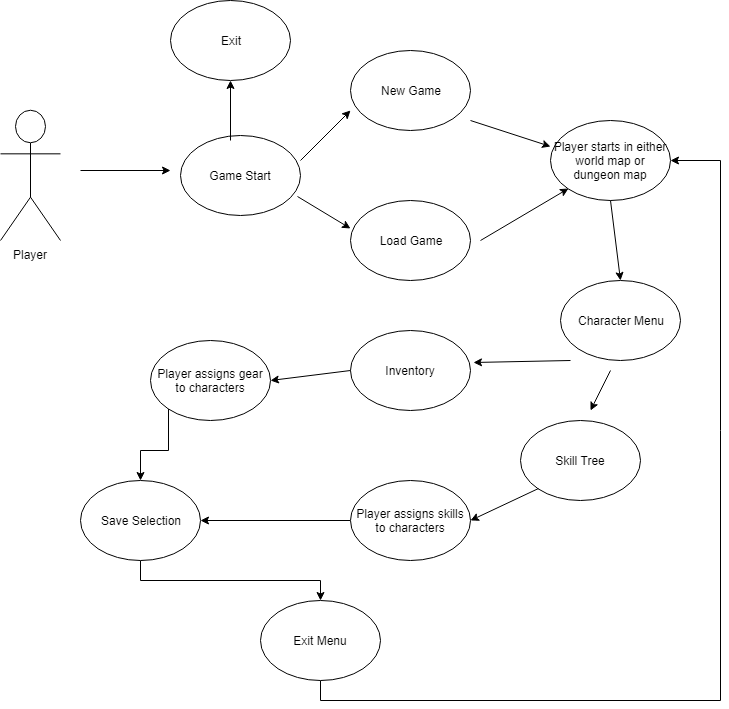
* User Entering Random Mob or Boss Battle Diagram (Overview)
* User Character Menu Interaction (Overview)
* User Fighting a Random Mob or Boss Diagram
* User Saving or Loading a Game File Diagram

#### 2.3 Use Case Diagrams

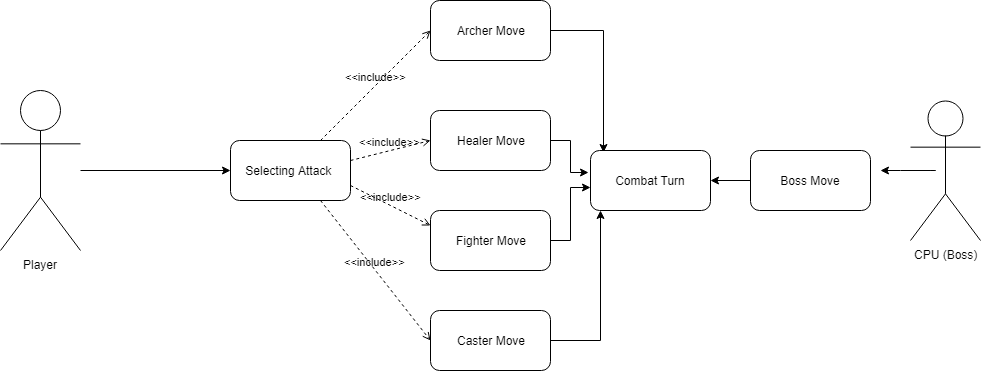
**User Entering Random Mob or Boss Battle Diagram** (Overview)



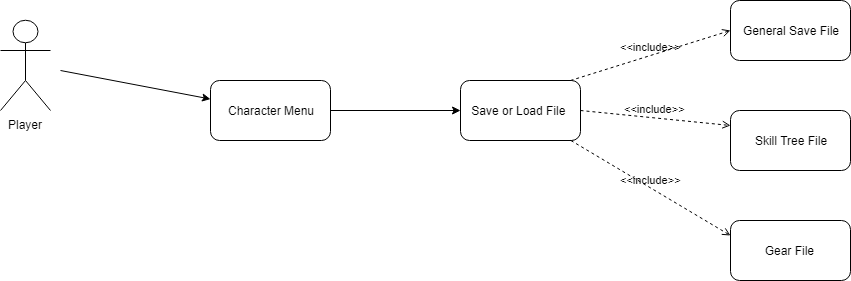
**User Character Menu Interaction Diagram** (Overview)



**User Fighting a Random Mob or Boss Diagram**

****

**User Saving or Loading a Game File Diagram**

****

### 3 Design Overview

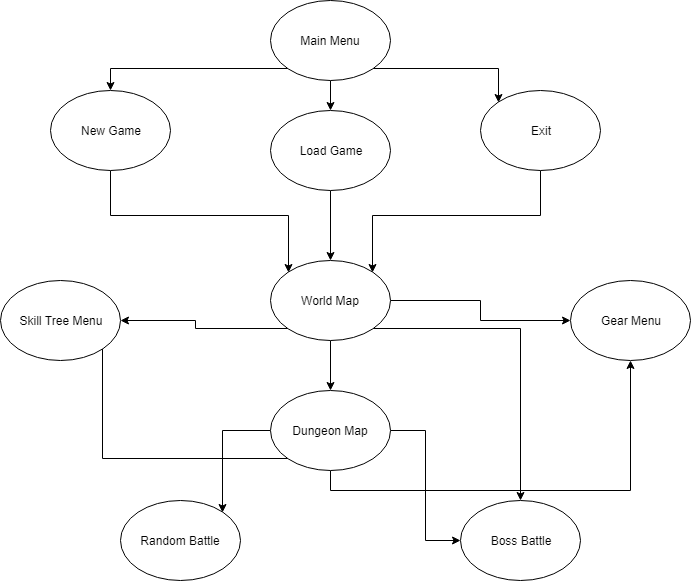
#### 3.1 Introduction

In this section, a breakdown of the system architecture will be provided to ensure that the structure of the game can be properly followed. This will allow the reader to properly link all the scenes / levels in their project. In Godot, each scene always has a single root node, and upon that root node, the developer adds child nodes to create game elements. Different nodes of the tree can have different functions, such as different level layout or different level components. When creating a project, one scene if first selected as a main scene, in which then the developer writes scripts to manage and show other scenes. For each node in the scene, the developer will be able to add a script. For instance, if the developer wishes to have a controllable character, they will add a script to a kinematic body element in which the script will define how the character is controlled.

#### 3.2 System Architecture

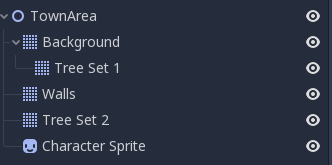
The following system architecture is followed for the Zalteros game. This hierarchy is built upon the scene concept explained in the introduction of the design overview. The structure of the scene flow is important to note as it will affect the overall flow of the game.

**Scene Flow Diagram**



##### 3.2.1 Creating an Example Scene

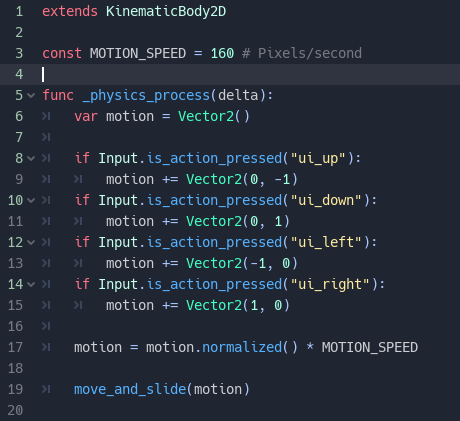
The following images shows the example of how one can build a simple scene. In the scene shown, we are first create a 2D node upon which we are going build all other scene components on. A background tileset is use to build out the grass or road sections of the map, walls tileset is used to construct walls upon which we will be adding physics later, and the two tree tilesets are used to add trees around our map. It is entirely possible to just add sprites onto a scene instead of using tilesets, however, using tilesets makes the process easier as we can essentially “paint” the scene with desired set.



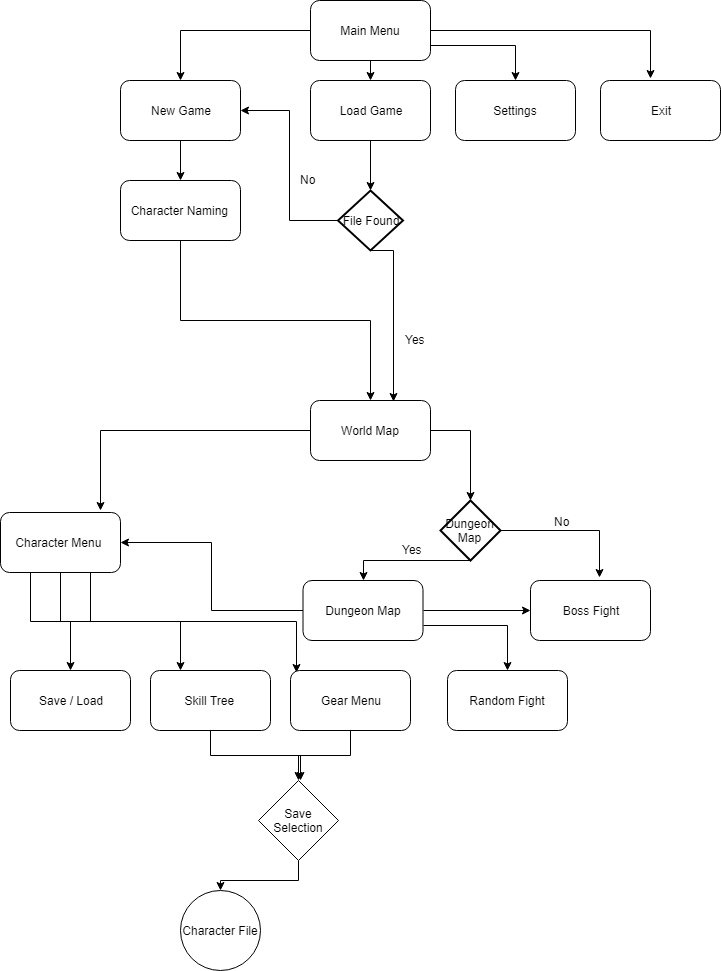
*(example node structure) (resultant scene)*

3.2.2 Writing an Example Script

In order to add motion to a sprite or an object, we click on the selected component to add script. An example of the character movement script is provided below. To control the speed of a sprite, we set the motion speed to a specific pixel per second value, and in our physics process function we specify which keys result in what output as well as the flow of the character and the motion.



**Activity Diagram**

****

#### 3.3 System Interfaces

##### 3.3.1 User Interfaces

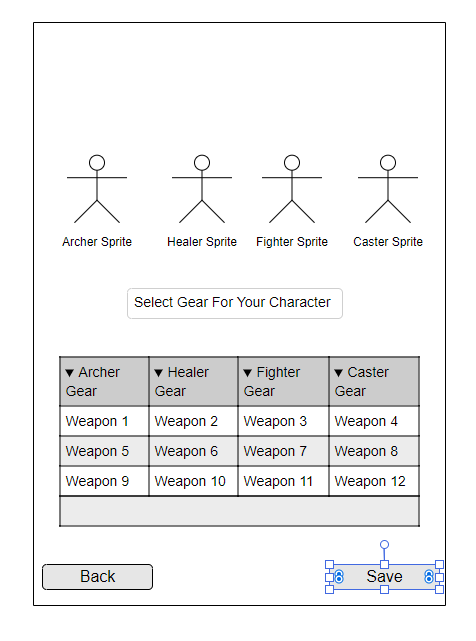
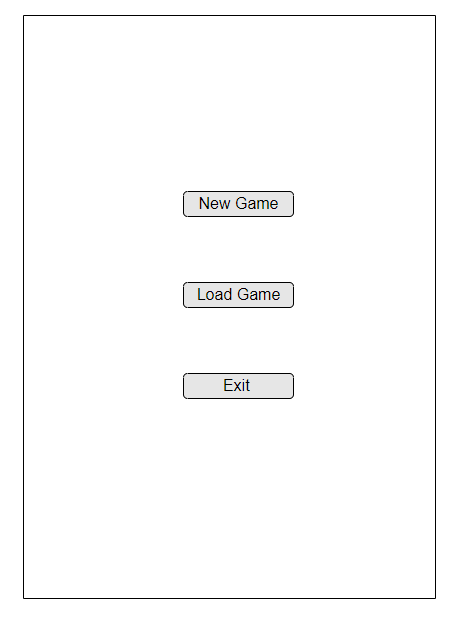
The user interface will allow the user to control their character, save or load files, and enter different levels. They will be able to control the skills of each player as well as selecting skills in the skill tree and assigning gear based on appropriate class type. First, the user will be presented with a main menu in which they will be able to start a new game, load an old save or exit the game. Upon selecting new game, the user will be met with a character naming screen. They will be able to set names for each of the four characters they will be playing as. After picking names and hitting “Continue”, the player will be first shown a cutscene. In any cutscene of the game, the user will be able to continue going through the story by hitting “Enter.” Once the first cutscene will be completed, the user will start in the village of their characters. The user will be able to explore said village or go to the main world map to enter other areas. Select map areas will be provided with a dungeon map in which the player will be able to walk around and encounter random mobs, as well as fight the main boss of the dungeon. In select areas where there isn’t a dungeon map, the user will directly enter a battle with a boss. The user will be provided with a set of defined controls (table below), which they will have the ability to modify in the settings menu.

|  |  |
| --- | --- |
| **Action** | **Controls** |
| Move Up | Arrow Up |
| Move Down | Arrow Down |
| Move Left | Arrow Left |
| Move Right | Arrow Right |
| Select | Either Enter or A |
| Back | S |
| Menu | Escape or D |

##### 3.3.2 Interface Design

The designs for non-map area are kept simple to make the user experience easier. Below

are two examples for the interface designs of the main menu and gear selection.



*(Sample Main Menu Layout) (Sample Gear Menu Layout) (Sample Dungeon Map Layout)*

#### 

*(Boss/Random Battle Reference Final Fantasy 4) (Character Menu Reference Final Fantasy 4)*

#### 3.4 Constraints and Assumptions

It is assumed that the disk on which the application is stored on has sufficient space for when the game will create new save and user data files. There are no external dependencies on which the application will rely on in order to successfully run. It will also be assumed that the computer which is used to run this application will have sufficient capabilities to run and render a 2D game.

#### 3.5 System Object Model and Collaboration Diagrams

### 4 Data Design

#### 4.1 File Storage

In order to save the user’s information and game session, the application will be writing and reading to a json file saved in the file system of the application. Using GDScript, when a user first saves their game, a new save file will be created for said user from the information in the current scene.

#### 4.2 File Storage Structure

Main Save File

|  |  |
| --- | --- |
| **Key** | **Value** |
| Id: | User id generated based on the time of first save. |
| AreasUnlocked: | Number of areas that the user has access to. |
| TimePlayed: | This will record the hours and minutes the player has spent in game. |
| SkillsUnlocked: | This will record total of skills unlocked for all playable characters. |
| SkillTreeFile: | When a new game is created, there will be a separate file in which the game will store / load the character skill tree. This will record the name of the file. |
| CharacterNames: | This will record the character names that the user has entered: |
| GearFile: | When a new game is create, there will be a separate file in which the game will store / load all the gear that the user’s characters have. This will record the name of the file. |
| BossesDefeated: | This will record the names of all the main bosses that the players has defeated so far. |
| StoryLine: | This will record at which point the user is in their story quests. |
| MobsDefeated: | This will record the total number of mobs defeated. |

Skill Tree File:

|  |  |
| --- | --- |
| **Key** | **Value** |
| ArcherBranch: | This will record which skill tree branch is currently unlocked for archer. |
| HealerBranch: | This will record which skill tree branch is currently unlocked for healer. |
| FighterBranch: | This will record which skill tree branch is currently unlocked for fighter. |
| CasterBranch: | This will record which skill tree branch is currently unlocked for caster. |
| ArcherSkills: | This will record the list of currently unlocked skills for archer. |
| HealerSkills: | This will record the list of currently unlocked skills for healer. |
| FighterSkills: | This will record the list of currently unlocked skills for fighter. |
| CasterSkills: | This will record the list of currently unlocked skills for caster. |

Gear File:

|  |  |
| --- | --- |
| **Key** | **Value** |
| ArcherGear: | This will record the list of gear that the archer currently owns. |
| HealerGear: | This will record the list of gear that the healer currently owns. |
| FighterGear: | This will record the list of gear that the fighter currently owns. |
| CasterGear: | This will record the list of gear that the caster currently owns. |

### 5 Game Design

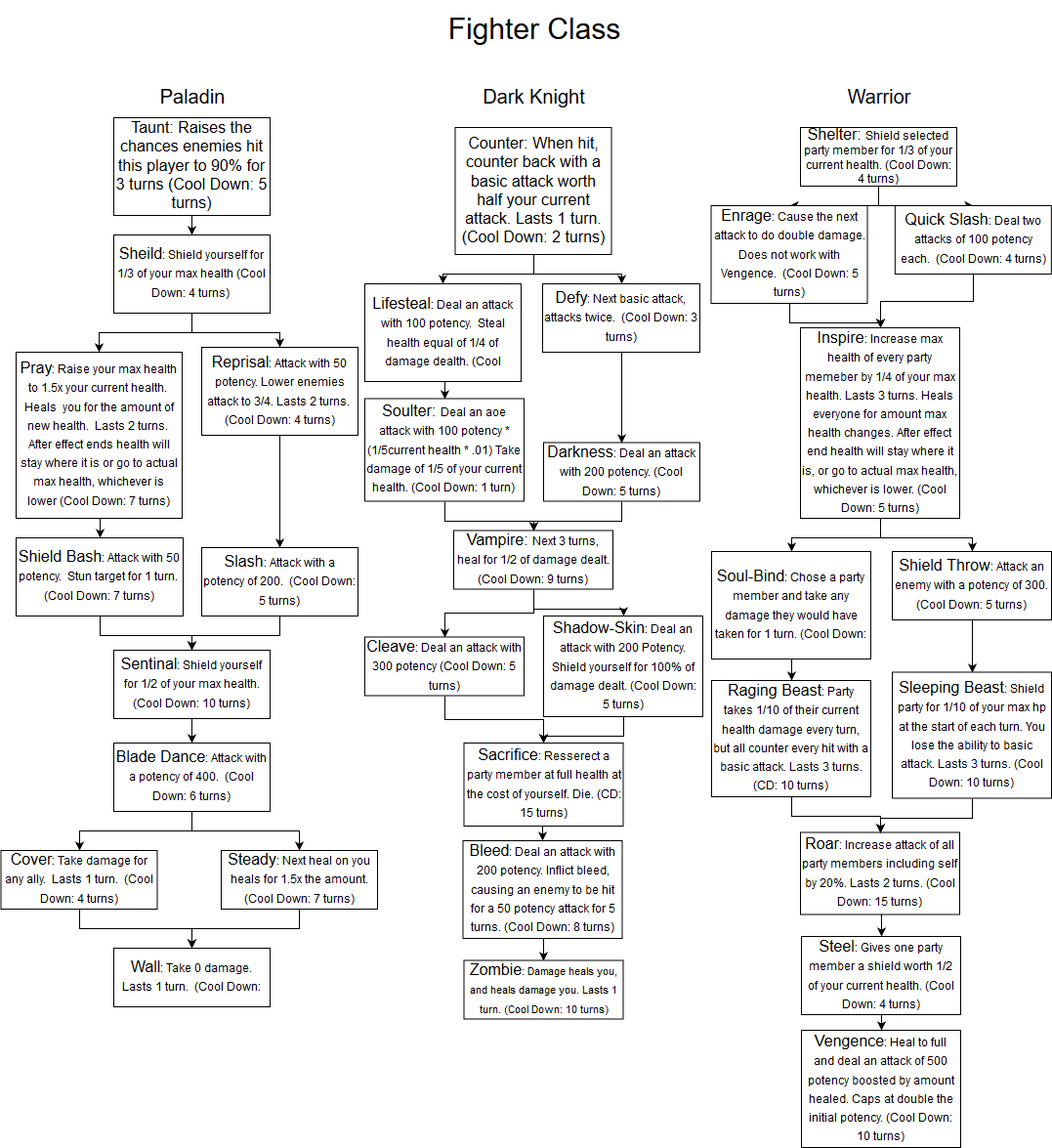
#### 5.1 Overview of Game Design

Zaletros is a turn-based RPG. The user will control a party that consists of four characters, each with a different class assigned to them. The user will be able to name each character individually and build each character specific ways. The characters will be able to learn skills by spending experience points and navigate through the skill trees each character has.

As the player walks around the world map, they will encounter random battles. These random battles can drop gear for each character to equip and improve the stats of that character. There are dungeons that exist around the map and boss battles can be in these dungeons, or on the world map itself. Random battles can also happen in dungeons. There will be a storyline that helps guide the user to where they should be heading, as well as unlock certain areas and bosses.

#### 5.2 Skill Trees

Each character will have it’s own skill tree, each consisting of three branches that will build that character in a certain direction based on the name of the branch. Each character will start the game with the first ability in each branch. To get more abilities, the user must spend experience points to unlock abilities. Abilities can only be unlocked if the ability before it has already been unlocked. As a character acquires more abilities, the amount of experience points needed to unlock the next one goes up. Below is an example of the skill tree for the character with the class Fighter.



#### 5.3 Gear Design

There will be a total of 18 sets of gear for each character. One set will automatically be given at the start of the game to help the player proceed and will be labeled ‘starter’ gear. 9 sets will drop from bosses and be labeled as ‘epic’ gear, consisting of higher stat values then gear that had been dropping from random battles in the area. The other 8 sets will be dropped from random battles and be labeled as ‘normal’ gear. Gear will consist of two stats, attack and health. Health will add HP points to that character, and attack will change the attack of that character affecting how much damage they can do. Gear items will include weapon, helmet, chest, gloves, pants, and boots.

|  |
| --- |
| **Beginner Caster Hat**  Attack: 2  Health: 30 |

#### 5.4 Boss Design

There will be a total of 9 bosses for the player to defeat. Bosses will be spread out between two land masses, with the final boss on a floating island. Bosses on the second island will not be accessible until the player proceeds far enough into the story. 6 of these bosses will be locked behind story progression, and 3 the player can chose to kill once they can reach that area. Each boss can be killed multiple times and each one will had a different set of moves and pattern. Bosses may be inside of a dungeon, or out in the world map. There will be a confirmation message before a player enters any boss fight, allowing them to turn back and prepare. Below is an example of a boss, showing what the first boss fight may look like.

Boss: Forest Guardian

Location: Forest Dungeon  
Story Locked: Yes

Race: Living Tree

Health: 1000hp

Attack: 30

Abilities:

Harden: Gives itself a shield worth 200 hp. Uses at the start of the battle.

Smash: Attacks one player for 200 potency. Executes every turn that is a multiple of 4.

Basic Attack: Deal an attack to one player for 50 potency. Executes if nothing else is assigned for that turn.

#### 5.5 Map Design

The map will consist of a world map where the player will spend most of its time walking around to different points on the map. The map will have the character larger than many features to allow for the player to easily see where the character is. On the map will be points where the player can enter dungeons or boss fights. Dungeons will have their own map and also include a boss fight at the end. Below is an example of the player walking in a dungeon map.



#### 5.6 Character Design

The characters will be in the style of sprites, and be made using Aseprite. Every model including NPC’s, monsters, and bosses will be made as a sprite. The characters the player controls will have walking animations as well as animations when taking their turn in battle. Below is an example of what the character who is a caster will look like.

