**LimeQuest**

Turn-Based RPG Framework

**V 0.8**

If you want to fix bugs, want to add improvements and/or features don’t hesitate to make changes.

If anything is unclear, contact [canturker00@gmail.com](mailto:canturker00@gmail.com).

(If this project made things easier for you, please let me know it!)

Content

Getting Started....................................................................................3

How to Play.........................................................................................3

PlayerManager.....................................................................................4

DangerZone..........................................................................................4

MovementController.............................................................................5

UIManager............................................................................................5

BattleManager......................................................................................6

EncounterData......................................................................................7

FighterData...........................................................................................7

MovesetData.........................................................................................8

SkillData...............................................................................................9

Effects.................................................................................................10

**Getting started**

* Create a new Unity3D project and move this project’s project files into the new project’s root folder **or** open this project on Unity Hub.
* Open *File>Build Settings* menu on Unity3D and drag&drop *Indoors* and *BattleScene* scenes into *Scenes in Build* field.
* Open the scene named *Indoors* and hit play.

IMPORTANT: This project was built in Unity3D version 2021.3.0f1. It’s recommended to use that version of Unity3D if you plan to use this template.

**How to Play**

* The example game has no objectives and just a playground for the features.
* Press F key on your keyboard to add predefined fighters to your player.
* The red field in the indoors area is a *random encounter field.* It will spawn the defined enemies if the player navigates on the field long enough.
* The battle scene starts after an encounter. You can use the fighter skills or change your fighter. You can only do one action per turn.

**PlayerManager**

Graphical user interface, text, application

Description automatically generated

PlayerManager is a singleton that holds the essential informations about the player such as,

**Fighter List:** The list of fighters that the player curently owns.

**Current Fighter:** The fighter that currently leads the list. It’s the fighter that will be called to the field when a battle starts.

**Datas (DEBUG ONLY):** Predefined fighter list contains the fighters that will be added to **Fighter List** when the player presses F in the indoors scene.

This class also handles operations like adding new fighters, loss checking, skill casting, fighter switching in battle.

**DangerZone**

Graphical user interface, application

Description automatically generated

This scripts handles the functionality of random encounter zones. Min travel and max travel values are the minimum and maximum travel the player has to make to encounter the enemies in the defined **Encounter Data** respectively.

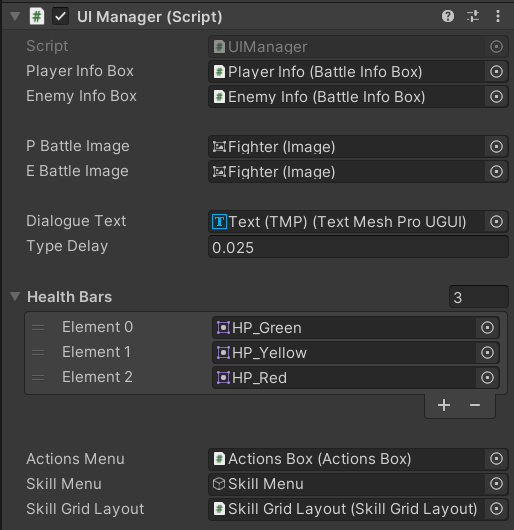
**MovementController**

Graphical user interface, text, application

Description automatically generated

This script handles the basic top-down movement of the player character.

**UIManager**



UIManager singleton component handles the UI functionality in BattleScene scene. This includes the HUD functions and TypeWriter for the dialogue box, turn management for the HUD.

**BattleManager**

A screenshot of a computer

Description automatically generated with medium confidence

BattleManager singleton script handles the main flow of the turn-based battle. It also handles the Enemy Logic (the simple ai for the enemies).

Functions:

**PlayAnim:** Plays the animations of the skills that are currently being used by the player or the enemy fighter.

**EndBattle:** Ends the battle and triggers a fade-out animation.

**GetTarget:** Returns the target that will be used in the skill casting algorithims.

**SetTurn:** Changes the turn.

**EncounterData**

**A picture containing graphical user interface

Description automatically generated**

This scriptableobject holds the encounter dataset that is going to be used in any random encounter zone.

Graphical user interface, application

Description automatically generated

**FighterData**

Graphical user interface, application

Description automatically generated

This scriptableobject holds the data for each individual Fighter character in the game.

Graphical user interface, application

Description automatically generated

**Ai Moveset** field holds the basic ai logic for the computer-controlled characters .................................................................................... (enemy encounters).

**Types** is an unused but planned functionality at the moment.

**MovesetData**

**Graphical user interface, application, Word

Description automatically generated**

**A screenshot of a computer

Description automatically generated with medium confidence**

MovesetData holds the moves that can be performed by ai characters. High, Medium and Low Moves take the name from the probabilities of happening of the actions.

**SkillData**

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Each SkillData holds the information for a skill. Each skill may have more than one Effect. **Skill Name** and **Skill Text** get showed on HUD. Target animator is the target’s animator that will play the named **Animation.**

**Max Use** is a planned feature that will hold an integer value and the skills won’t be able to used times more than that value.

**Effects**

Text

Description automatically generated with low confidence

Effects are the performances that can be casted by fighters. Effects are hardcoded because each effect’s functionality is different but effects can be assigned to multiple different **SkillData**.

Text

Description automatically generated

New effects can be created by implementing **Effect** abstract class. **Use()** functions must be overridden and implemented.

**Example effect**

Graphical user interface, application

Description automatically generated

**Halve HP** effect is a variant of ChangeAttribute effect. It’s **Change Type** is multiply (by half to halve) and the **Attribute** type is “H” for HP.