Requirements and Analysis Document

Group 16

October 5, 2022

Contents

1 Introduction

1.1 The Background

In the beginning the Universe was created. This has made a lot of people very angry and been widely regarded as a bad move.

To solve this problem we have decided to create a game, but not just any game, the GREATEST GAME TO EVER EXIST!

This super awesome game, the one we call Feyrune (*Loud gasping!*) is best described as a medieval, fantasy themed, Pokémon(tm) ripoff.

1.2 Functionality

The application is a game with limited functionality. While it can't help you with the meaning of life, or even file your taxes, it can bring a smile to your face.=)

1.3 Stakeholders

The purpose of this application is to help the user to have fun on their free time,as it is a game. In the game, you take control of a character that has tamed a creature that can be used for battle.

1.4 Definitions, acronyms, and abbreviations

Feyrune:	The name of the application
DoD	Definition of Done

2 Requirements

2.1 User Stories

- **DONE** As an explorer i want to be able to to move around in the world so i can see new places
    + Change location of player character.
    + Take input from player.
    + Move player according to input.
    + Animation when walking.
    + Animation when standing still.

- **DONE** As an explorer i want to be able to transition between different maps, so that i'm not stuck in the same area
    + Update Map not to be static and use a map manager instead
    + Do the same thing as above for texture map
    + Create multiple maps
    + Update parser to use map transition data
    + Transition logically
   + Make TextureMapManager observe model MapManager for changing map

- **DONE** As a pacifist I want to be able to run from combat so my monsters don't get hurt
    + Run action in combat menu
    + Chance to end combat when running away

- **DONE** As a player i want to see my environment so I know where I am
    + Parse tile sets.
    + Implement map renderer class

- **DONE** As a player i want my monsters to be able to attack, so I can win against other monsters.
    + Create an attack handler witch handles attacks
    + Create a way for monsters to take damage

- **DONE** As a player i want the game to understand my inputs so i can play the game
    + Implement an input listener
    + Make input listener take input from keyboard
    + Connect Input to model

- **DONE** As a modder, i want to be able to parse and make my own Tiled maps
    + Load tile sets from Tiled file
    + Load tile information from Tiled file

- **DONE** as a player, I want to be able to fine a monster to attack
    + Encounters to trigger a new combat
    + Change scene to view combat

- **DONE** As a realist I want the world to limit my movements so I cant't walk through walls
    + Implement collision logic
    + Parse collision from a map file

- **DONE** As a player i want to be able to attack a monster with my monster because i want to be able to win
    + Design health interface
    + Implement health interface
    + implement hit enemy.

- As a collector I want to keep multiple monster with me so that I can use more then one
    + Implement somewhere to store multiple monsters
    + Design an interface for seeing your monsters
    + Pick one that is the current one

- As a pacifist, I want to befriend my enemies so I can have more friends
    + Design a befriend interface
    + Add befriending difficulty to monsters

+ Display befriending difficulty in combat

+ Add befriended monster to you monster party

- As a player, I want to be able to pick an attack so that I can strategically control my monster during combat

    + Design a "pick an attack"- interface

    + Implement visuals fo said interface

    + Create several (more then one) different attacks to choice from

- As a speedrunner

    + Design a befriend interface

    + Add befriending difficulty to monsters

    + Display befriending difficulty in combat

    + Add befriended monster to you monster party

2.2 Definition of Done

The project code was structured in such a way that when a person felt that they were done with a user story, they had to get it reviewed and be questioned about it by at least one other contributor. The purpose of this practice was not only to make sure that the user story was implemented fully, that can be done with tests, but to iron out code smells and to keep the code somewhat coherent and similar-looking throughout the project.

2.3 User interface
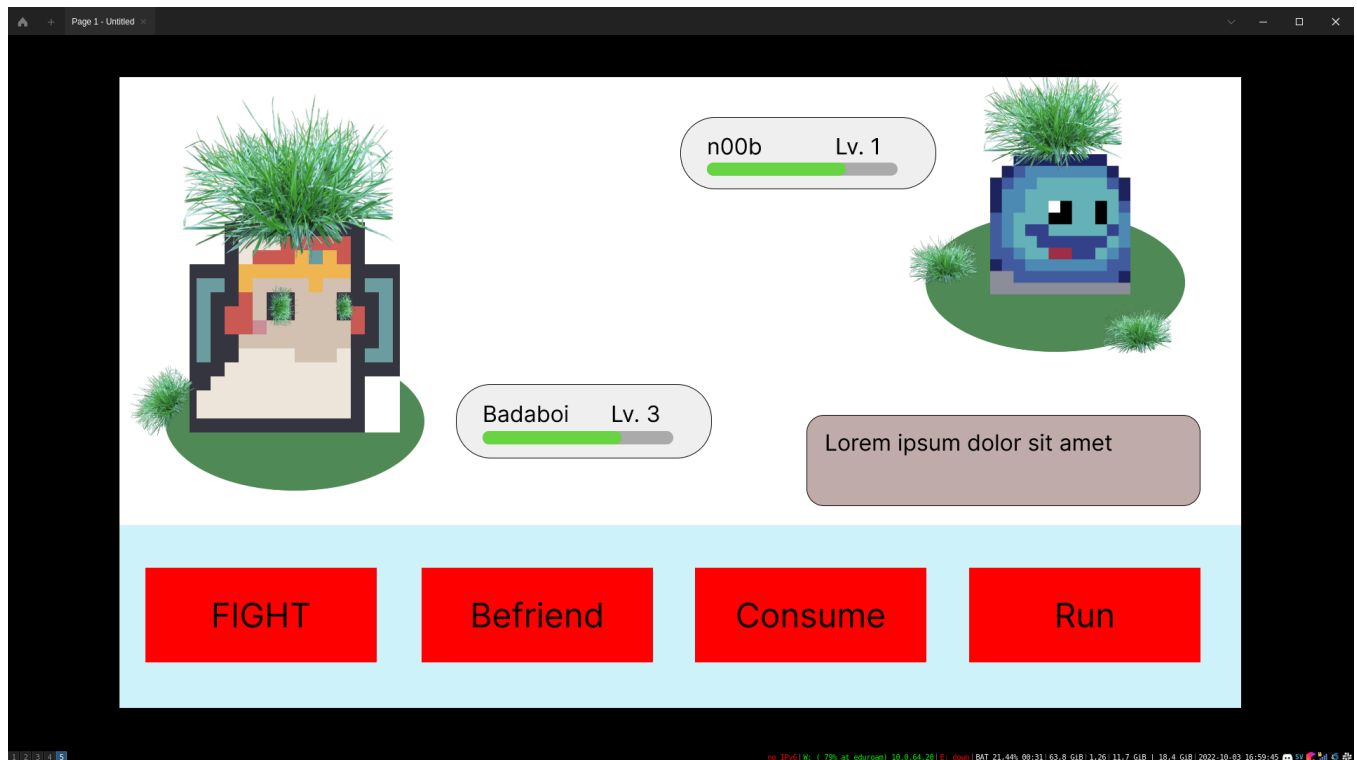
3 Domain model

3.1 Class responsibilities

4 References