Software Requirements Specification

Monkey Dungeon Maze

v0.2 indev

Prepared by Z. Andersen

UWT TCSS 360

Winter ‘24 Group 1

15 March 2024

Table of Contents

Revision History **2**
Introduction **3**
Purpose 3
Conventions 3
Audience 3
Summary Description **3**
Perspective 3
Features 3
User Classes 4
Operating Environment 4
Design Constraints 4
User Documentation 4
Assumptions and Dependencies 4
System Features **4**
Dungeon navigation 4
Description & Priority 4
Stimulus/Response 5
Function Requirements - FEAT1 5
Combat, mobs, and classes 5
Description & Priority 5
Stimulus/Response 5
Function Requirements - FEAT2 6
External Interface Requirements **6**
User Interface 6
Hardware + Software Interfaces 6
Further Requirements and To Do Items **6**

Revision History

Approved By	Date	Change Log	Version #
W.H. & L.A.	3/15	Updated conventions and made SRS match current game implementation.	1.1

Introduction

Purpose

Green Monkey Dungeon Maze is an infinitely replayable dungeon exploration game. Stages will be randomly generated for a unique experience in every playthrough with customizable levels of difficulty.

Conventions

Following standard .NET C# conventions.

Audience

This document is designed for developers and project managers to understand the overview and scope of the project. This document is best read in order top to bottom with special attention to overview sections.

Summary Description

Perspective

This is a self contained product designed in accordance with assignment requirements for TCSS 360 at UWT. This product is an entirely standalone game which required no external products to engage with.

Features

Users should have the ability to do all of the following

- Navigate and explore a randomly generated space themed dungeon
    - Have a unique playthrough experience with each iteration of the dungeon
    - Be able to concretely win by collecting required items and reaching the exit zone
- Play different player “classes” which offer unique gameplay
- Acquire items which can be used in the dungeon or are win condition requirements
- Engage in combat with hostile mobs
- Manage multiple playthroughs through save file management which can be performed within the software

User Classes

Users are expected to have a baseline knowledge of how to install software onto a personal computer and interact with normal devices such as keyboards, mice, and monitors. Users are not expected to have any advanced knowledge and will be shown how to play the game through the tutorial.

Operating Environment

This product is primarily targeting Windows 10+ PC and should be able to run on any modern hardware. The product is being developed with the expanded goal of working with Mac OS X and modern Linux distributions such as Ubuntu and OpenSuse in mind.

Design Constraints

Product is being developed in the C# language using the Monogame framework targeting OpenGL rendering. The software must make use of a SQLite database for data storage and retrieval.

User Documentation

Controls:
WASD: Player directions
Space: Shoot
Tilde: Debug Menu
Esc: Main menu
Numpad +: Zoom in
Numpad -: Zoom out

Keybinds can be changed through application data. This file can be found ~\AppData\Local\GameSpace\configs after the game has run once. Localization to Linux and MacOS is incomplete at this time.

Assumptions and Dependencies

This product relies on the .NET runtime version 6.0 as well as the Monogame framework implementation of OpenGL.

System Features

Dungeon navigation

Description & Priority

This is a top priority item and is the minimum viable product deliverable. Users must be able to have a player character deposited into a randomly generated dungeon stage which can then be explored. Users must be able to find and use items within the stage and then leave the dungeon.

Stimulus/Response

Users will be confronted with a character in an unknown dungeon space. We assume that from there users will be inclined to further explore. Users will need to avoid hazards which are placed randomly throughout the dungeon which should prompt further exploration.

Function Requirements - FEAT1

1. System which random generates dungeon stages with user understandable symbols for navigation
    a. Dungeon stages must be comprised of multiple rooms across a large enough space to create interesting and unique environments for player exploration
    b. Dungeon stages must have an exit location for victory condition.
    c. Dungeon stages must have randomly generated items including four win condition requirement items.
2. Entity system which can create Player entities that are controlled by the user via keyboard + mouse input.
    a. Entities must have the ability to track multiple statistics about themselves such as location, inventory, health, etc...
3. Item system which allows for generation of a set of useful items as well as four win condition requirement items which must be brought to the exit location.
    a. Items should have a concrete effect in game such as adjusting player stats, revealing parts of the map, or other useful interactions.
    b. The dungeon [FEAT1-1] must have the ability to have Items strewn about the stage.
    c. Entities [FEAT1-2] Must have the ability to carry an inventory of items for later use.

Combat, mobs, and classes.

Description & Priority

High priority. Entities must be able to engage in a real time combat system with one another using abilities and items. Players should be able to choose from multiple “classes” with unique gameplay abilities which may provide unique interactions in the combat system. The Entity system must be expanded to also create “mobs” which the player can fight, mobs should also have multiple classes with unique combat abilities to make combat interesting and unique for every encounter.

Stimulus/Response

Users will find mobs randomly throughout the dungeon stage, if a user is spotted by or deliberately attacks a mob then they will enter the turn based combat system.

Function Requirements - FEAT2

1. Entity system [FEAT1-2] must be expanded to create mobs with which the player can fight.
    a. Mobs & Players should have multiple classes with unique abilities.
        i. Classes should have unique abilities and statistics.
    b. Entities must have combat abilities to be used to affect other entities.
2. A combat system must be implemented wherein entities can engage in turn based play using abilities and items to defeat opponents.
    a. The system must be balanced for multiple levels of difficulty and to accommodate all player classes without making the game impossible to defeat.

External Interface Requirements

User Interface

The user interface must consistently display simple information about the status of the player such as health, number of key items held, and what items are held in the player’s inventory.

The user will primarily interface with the game from the map interface where the player can freely move through the 2D dungeon space and engage in combat via keyboard input; a default key mapping can be found in the User Documentation section. The user must also be able to navigate a menu structure via keyboard input to perform basic game functions such as save/load and viewing statistics.

Hardware + Software Interfaces

The software will require .NET Runtime version 6+ to operate. Any modern personal computer with an OpenGL compatible graphics interface should be able to run the program.

Further Requirements and To Do Items

The game requires a more robust UI system than is currently implemented in order to support dialogues and additional player interaction. The UI system needs to be made to support mouse input for ease of user interaction. The game currently lacks save/load functionality for a single playthrough but does track long term global statistics across multiple playthroughs via a persistent database.

Combat abilities and player classes are not yet implemented and would add spice to the game. Additionally only health items are available in game, but more are planned to add complexity to gameplay and especially combat.