Dungeon Divers

Requirements Analysis Document

https://github.com/comp195/spring-2021-final-project-dungeon-diver

Ethan Lo (e_lo2@u.pacific.edu)

Last Revision: 3/7/21

System Description and Project Overview

Dungeon Diver is a new game being developed by Ethan. It is being developed in the Unity game engine. This document will discuss the requirements the game must have.

Non-Functional Requirements

- Performance
  - The game must not lag or crash
  - Input from the user must be fast
- Security
  - The game's code must not be easily hacked into
- Scalability
  - The game should not be that big in file size as the maps are randomly generated
- Maintainability
  - The game's code must be easily understood for maintenance, patches, and updates

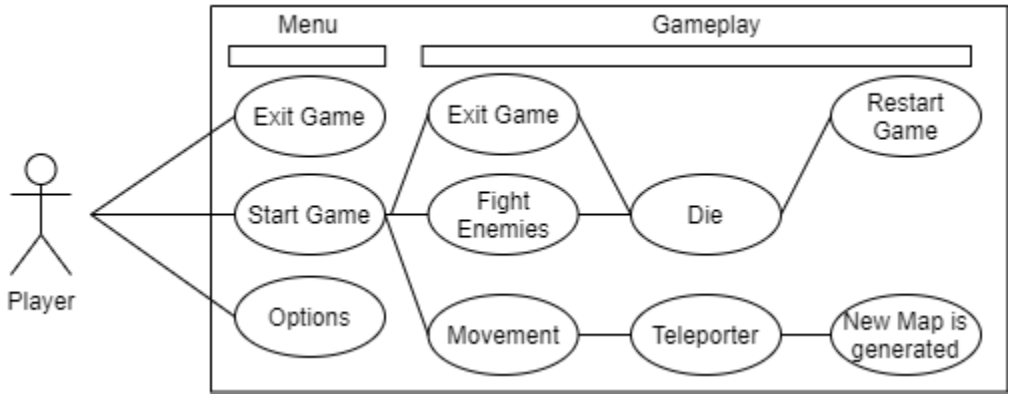
Functional Requirements

This section will list some features that the game will directly have in game:

- Options
  - Music volume
  - Mute sounds
- Randomized map generation
- Character Naming
  - When a character dies their "son" will take their place
- No saving game data

Use Case Diagram

The figure below is a use case diagram that illustrates the basic game loop. The menu has three options where you can start the game, exit the game, or select options in order to modify things such as volume level. The next section shown is game play, where you can see that you must search the map for a teleporter which will teleport you to another randomized map. If you die while fighting enemies, you will restart the game. If you want to quit, you will be treated as if you died and have to restart the game.



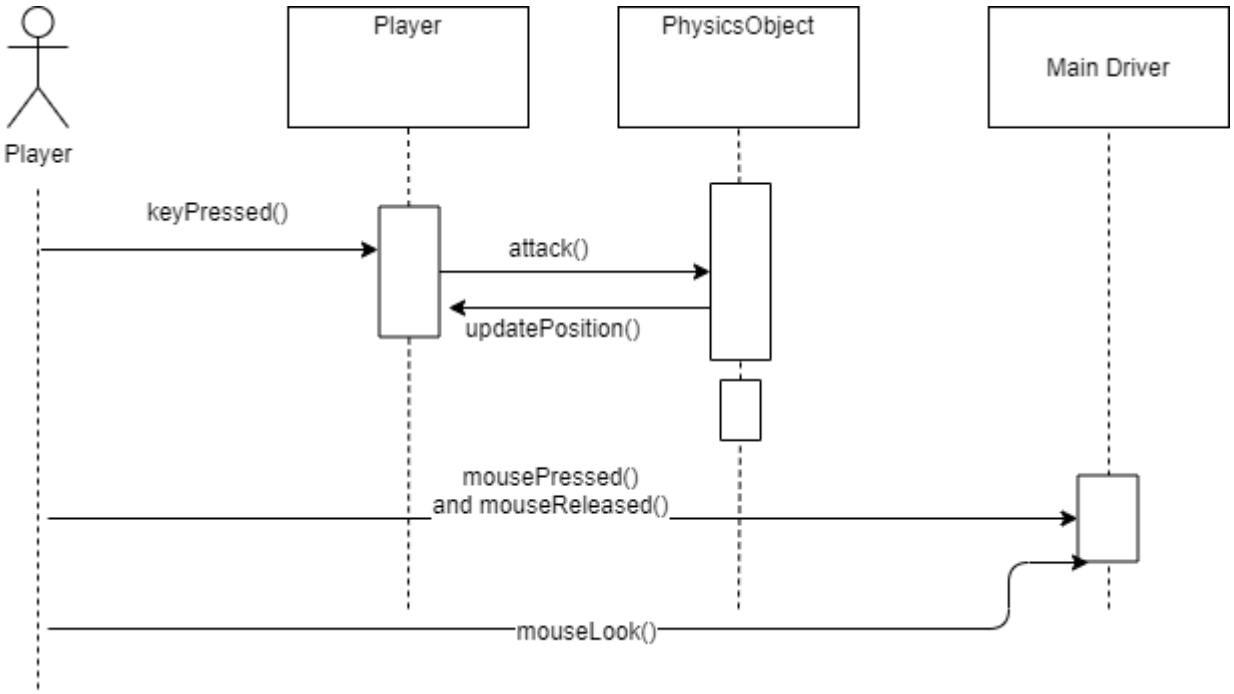
Use Case Descriptions

- Exit Game (Menu)
  - Condition: User is at the game's start menu
  - User uses the mouse to click and exit the game
- Options (Menu)
  - Condition: User is at the game's start menu
  - User uses the mouse to click and modify some game features
- Start Game (Menu)
  - Condition: User is at the game's start menu
  - User uses the mouse to click and start playing the game
- Movement (Gameplay)
  - Condition: User starts the game
  - User uses the WASD keys to move around the map. The user also uses the mouse to aim where to attack.
- Teleporter (Gameplay)
  - Condition: User moves around the map and discovers the teleporter. The user must also have discovered a key.
  - User must first find a key, then step onto the teleporter. Pressing a key, the user will be teleported into the next randomized map

- Fight Enemies (Gameplay)
    - Condition: User started the game and is exploring the map
    - The User uses the left mouse click to swing his weapon and attack the enemies
- Exit Game (Gameplay)
    - Condition: User started the game
    - The User wants to quit the game; progress is not saved; game will behave as if the player died
- Die (Gameplay)
    - Condition: User quit the game or User died while fighting enemies
    - User dies when they lose all their health and restarts the game
- Restart Game (Gameplay)
    - Condition: User dies or exits game
    - The User dies and progress is not saved. The game will restart from scratch, but the player will have access to new items from the last playthrough of the game.

System Sequence Diagrams

This interaction/sequence diagram below describes player movement and action. For example, you can see that if a specified key is pressed, the player will update its position and move on the map.



User Interface Design Sketches

The game must have an easy-to-understand user interface. It must not clutter the screen. The picture below is from another game that is similar to the intended user interface.

Here, you can see that the user interface is minimal, allowing the user to clearly see the screen. It also clearly transmits the information the user needs in order to play the game.

Glossary of Terms

- Random Map Generation
    - The game will not have pre-designed maps. It will use algorithms to make sure that the game is completely randomized in map design and gameplay.

References

N/A