Software Requirements and Design Document

For

Group 25

Version 2.0

Authors:
Nicholas Tidwell,
Keith Van Dyke
Bryce Hart

1. Overview

The system will be a 2D platform game with sprite graphics, audio, and the ability to control using a computer keyboard. The game will be part of the Platform Fighting genre where characters are stick figures and various weapons and power-ups can be utilized. The platform will be an arena that is unique and randomized upon each play. Players survive until their health reaches zero or they leave the bounds of the arena. In either case the other player wins. As in most fighting games, each battle will have a time limit, where if the time ends, it runs into Overtime. In Overtime, each player takes and deals significantly more damage. At the end of Overtime, whichever player has the least health loses. General settings for the game may be changed by the user. The game is started from the Main Menu. At the end of each battle, the choice is given to either play again or go back to the Main Menu.

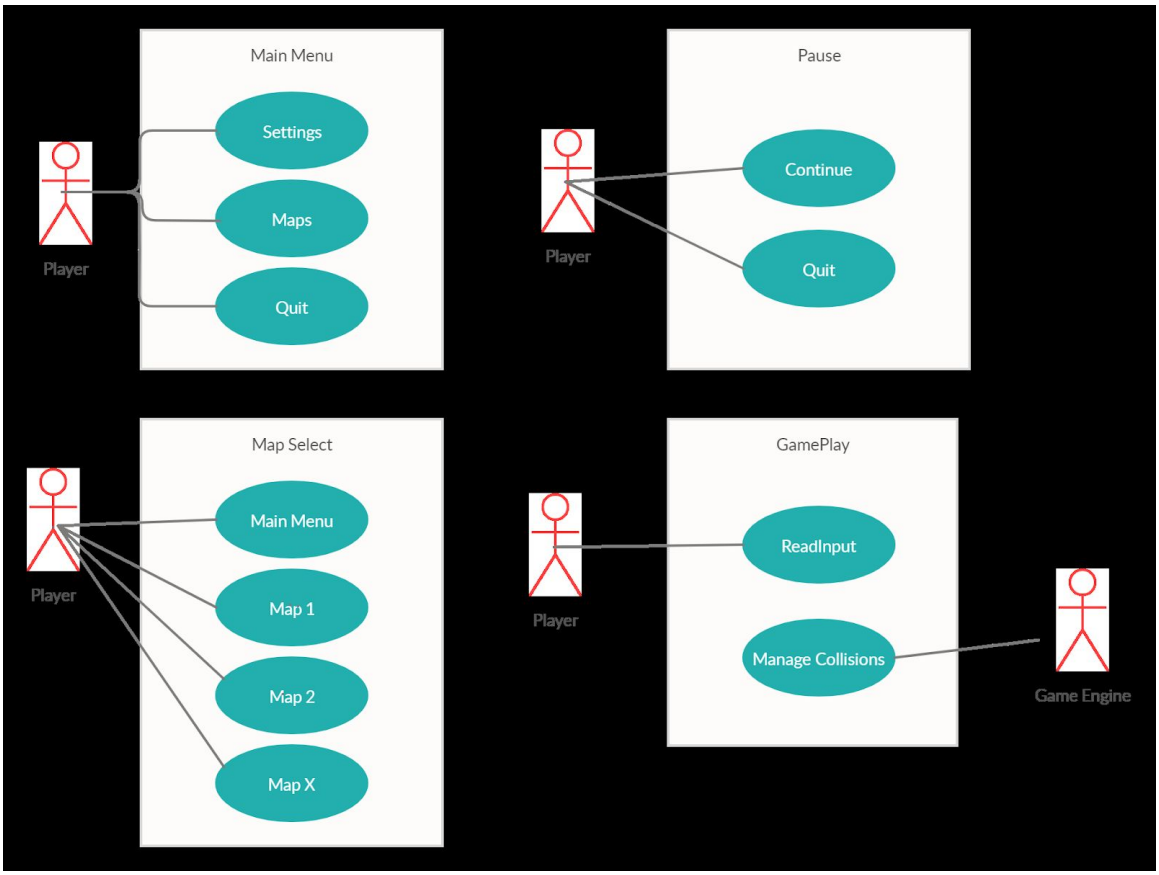
2. Functional Requirements

1. *High - Game shall be scripted in C#.*
2. *High - Game shall be developed in Unity2D.*
3. *Medium - Game shall contain graphics, audio, and keyboard controls.*
4. *Medium - Game shall allow strategy and creativity.*
5. *Low - Levels shall be unique and randomized.*
6. *Low - Game shall allow for consecutive plays.*
7. *Medium - Game shall contain settings.*
8. *Medium - Game shall show a HUD*
9. *Medium - Game shall include hazards that affect the player besides other players.*
10. *High - Game shall allow for players to move, jump, use weapons, and equip items.*

3. Non-functional Requirements

- *System shall have clean and minimal user interface*
- *UI shall be controlled by Keyboard (Mouse to aim if remote multiplayer is implemented)*
- *System shall run at minimum of 60fps;*
- *System shall require minimum processing power - (IE no more power than a mobile CPU)*
- *System graphics shall look clean and smooth.*
- *System code shall be peer reviewed.*

4. Use Case Diagram



Settings

- *Entry Condition: User Clicks on Settings button from Main Menu*
- *Normal Flow: User can tweak game settings while in menu*
- *Exit: User settings are updated and saved*

Maps

- *Entry Condition: User Clicks Maps from Main Menu*
- *Normal Flow: User is brought to Map Select Menu*

Quit

- *Entry Condition: User Clicks on Quit button from Main Menu or pause Menu*
- *Normal Flow: Game is closed and program stops*

Map Select

- *Entry Condition: User Clicks on Maps from Main Menu*
- *Normal Flow: User selects a map to load*
- *Alternate Flow: User selects to return back to menu.*

Pause

- *Entry Condition: User Pause Button within game*
- *Normal Flow: User Selects Continue and game is resumed*
- *Alternate Flow: User Quit and game is stopped.*

Gameplay – Read Input

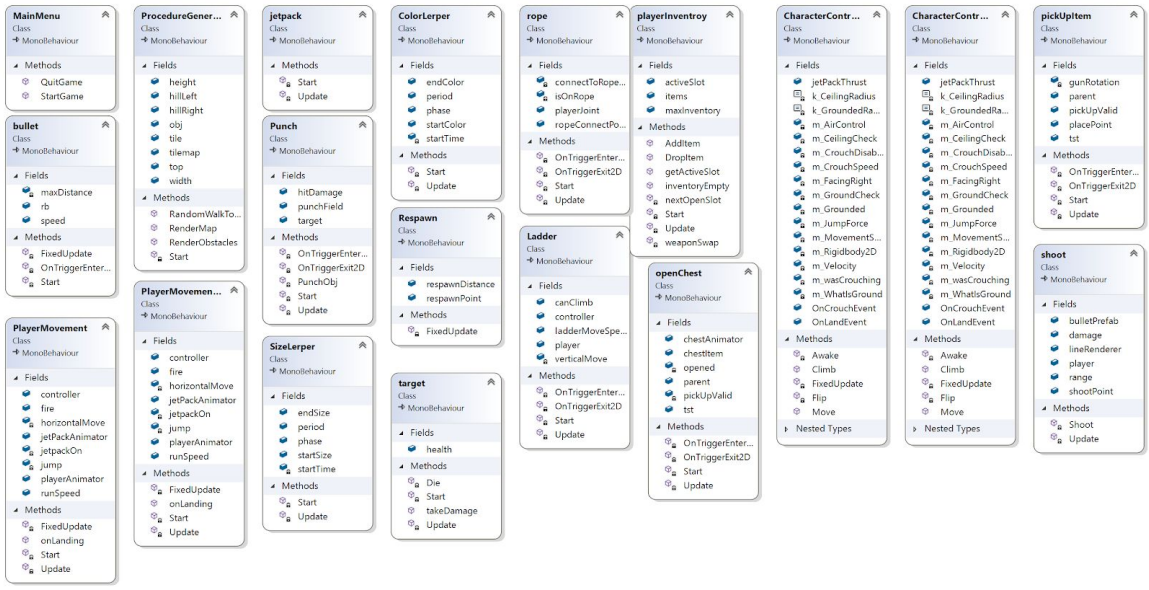
- *Entry Condition: User Selects Map and Game Engine is Started*
- *Normal Flow: User Types in Input*
- *Exit: Game Engine handles input*

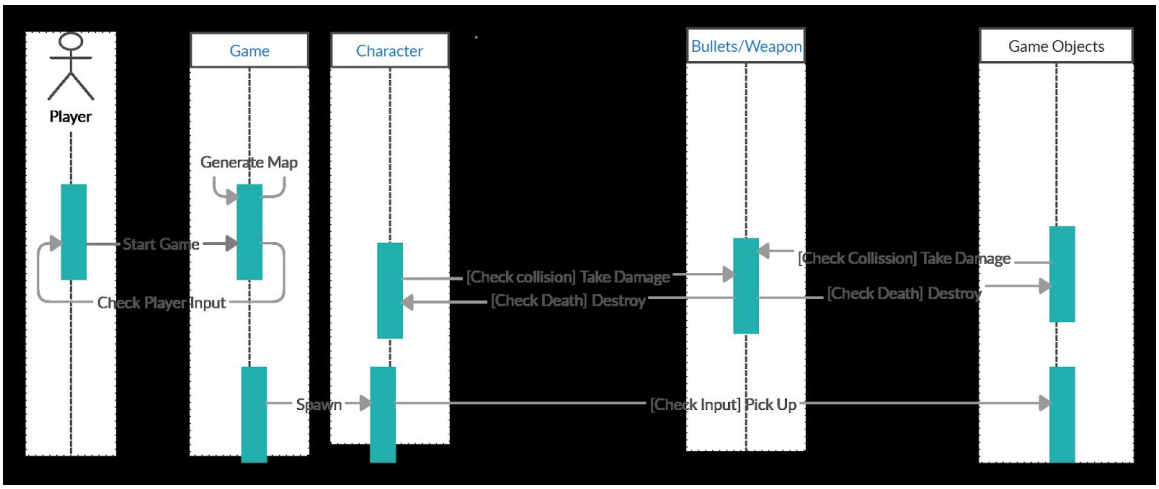
Gameplay – Manage Collisions

- *Entry Condition: Game Engine detects collision*
- *Normal Flow: Handle physics of collision-based encounter*

5. Class Diagram and/or Sequence Diagrams

All of our scripts derive from MonoBehavior which is “ The base class from which every Unity script derives. It offers some life cycle functions that are easier for you to develop your app and game. “ No script we wrote is derived from another so no relations are shown. Scripts are attached to gameobjects in Unity.





6. Operating Environment

Currently Developed in:

- *Unity 2019.2.3f1*
- *Unity 2019.2.8f1*

Tested on:

- *Toshiba Satellite Radius P55W - Windows 10*
- *Windows Surface 4 - Windows 10 Pro*
- *MacBook Pro (13-inch, 2017, Two Thunderbolt 3 ports)*
    - *macOS High Sierra Version 10.13.6*

7. Assumptions and Dependencies

Our game should be playable on any Desktop or Laptop and doesn't rely on any dependencies.