Use Case Diagrams / Scenario and Class Diagrams

By: Gabe Bybee

GABE (PLAYER CONTROL)

- Player Control Feature:
  - Allows the player to use W,A,S,D keys to control the in-game character's movements.
    - W = Forward
    - A = Left
    - S = Down
    - D = Right
  - Ability to attack enemies with left mouse-click, Melee || Ranged.
    - Weapons will vary per stage.
  - Allow for character animations trigger via movement.
  - Allows for a health counter for the HUD to use for health-bar.
  - Allows room for different weapons for HUD to use for inventory management

PRIORITY: (**High)**

Without a character for the player to control, the game will be unplayable.

Without a movement mechanic, the game will not progress.

Without Health or Weapons there is no combat system.

COMPLEXITY: (**Low-Medium**)

Simple Unity commands exist for tracking input, and translating to player movement is relatively easy. Compared to other systems within the game, this will be one of the easier systems. May provide a challenge when presented with multiple weapons + animations to go with those weapons.

GABE (PLAYER CHARACTER)

- Player Character Feature:
  - Uses an "Active Character State" to determine key events progressed to throughout the game.
    - Will save at given checkpoints, such as arriving in a world, or completing a puzzle before a boss fight.
    - Will contain a list of Boolean number tags to track things such as "Entered World 1", "Collected Weapon 2", etc.
  - Provide the ability to save your progress with minimal progress loss.
  - Allow the player to load a saved file and resume where they last left off.

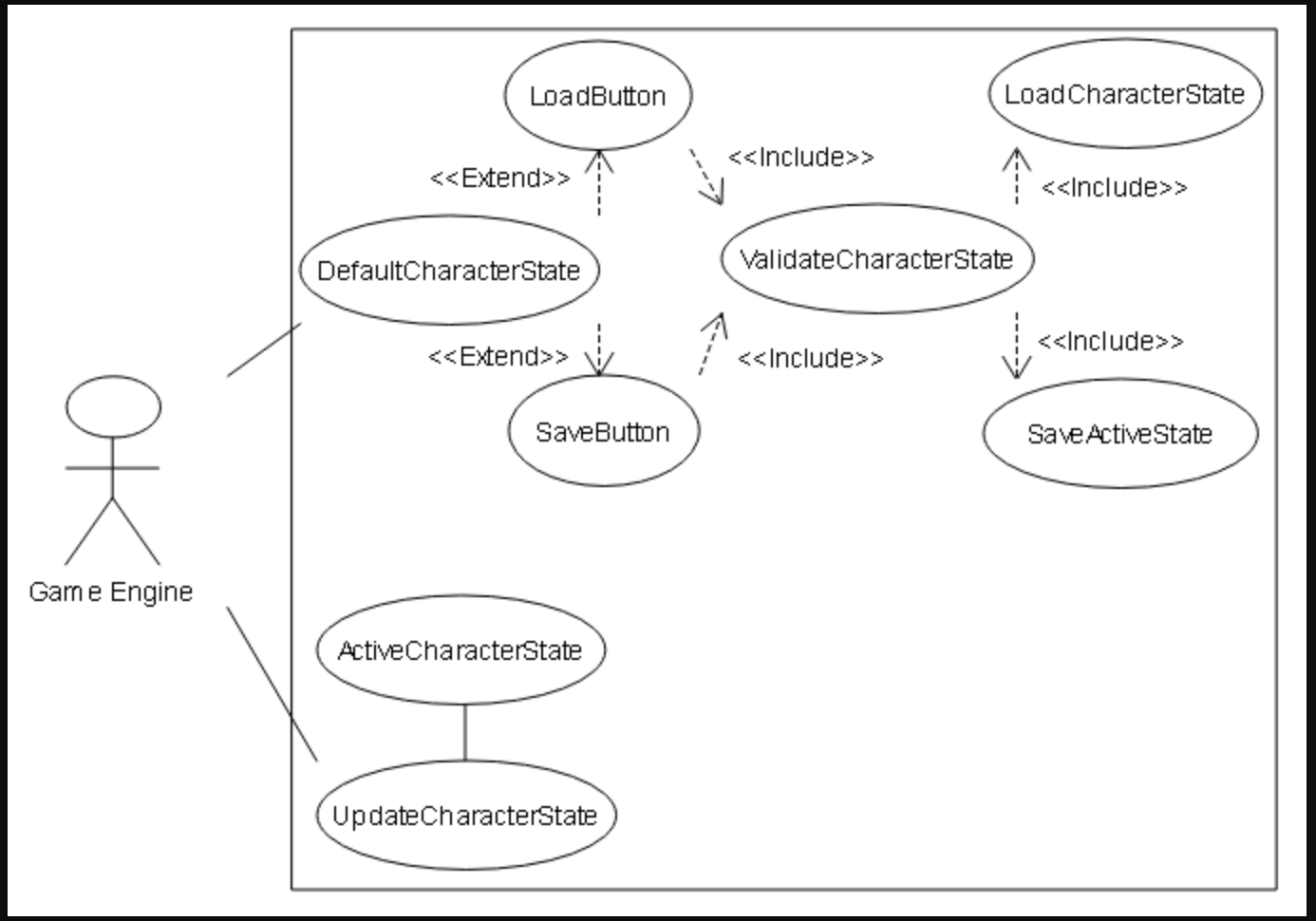
PRIORITY: (**Low)**

This Feature is unnecessary but will provide a quality-of-life feature that will make the game more "beginner friendly" by allowing them to not start completely over, especially on a loss. This feature was more thought of as a challenge to myself and something I would love to implement.

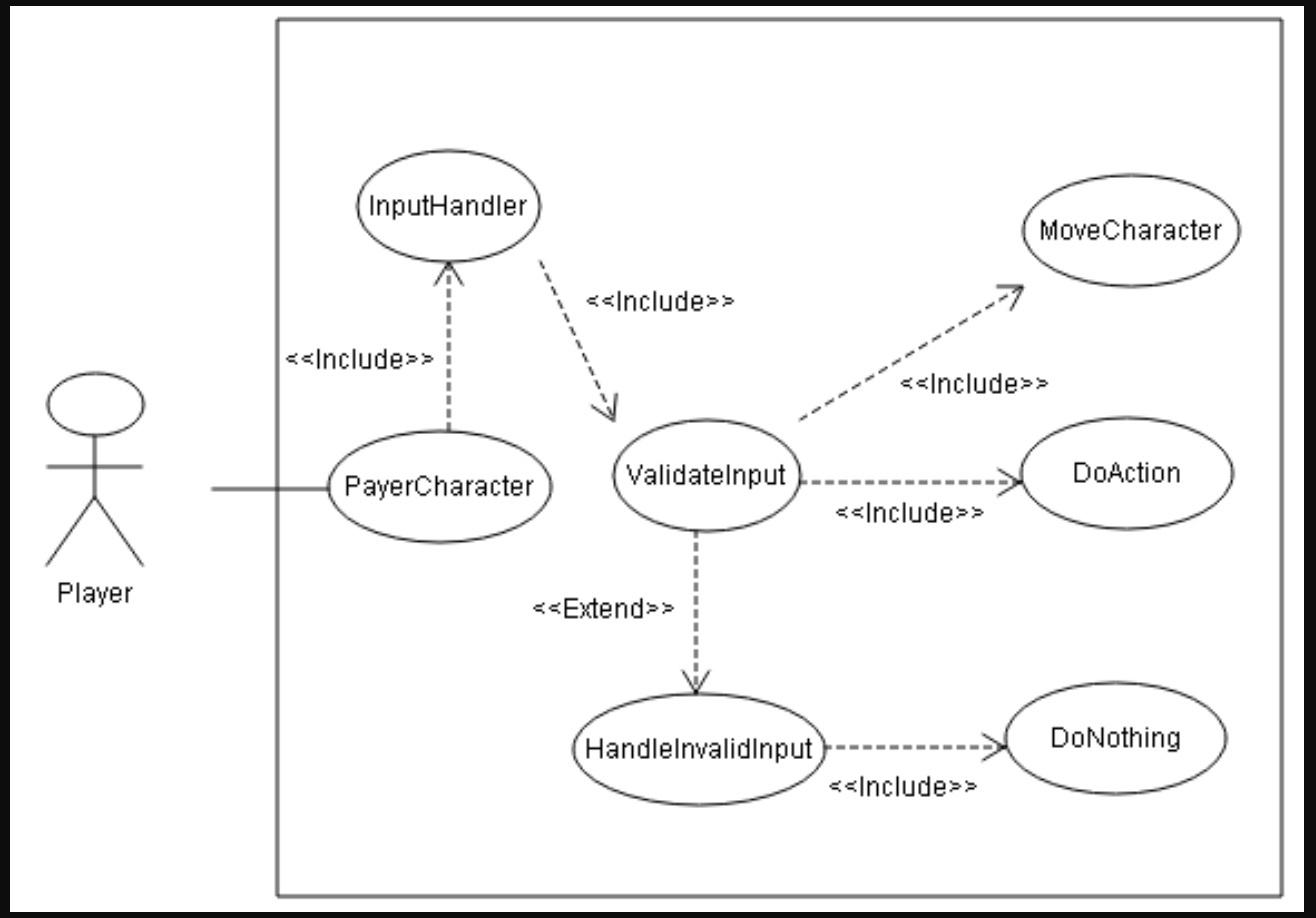
COMPLEXITY: (**High**)

Will require extensive research on other game's save features, how to write to a disk or file, and reading from the specified storage spot. At the time of starting this project, it feels as though this may be one of the more complicated systems to implement in our game.

GABE (PLAYER CONTROL USE CASE DIAGRAM)



GABE (PLAYER CHARACTER USE CASE DIAGRAM)



GABE (PLAYER CONTROL USE CASE SCENARIO)

1. Player Launches Game.

2. Player Starts The Game.

3. Player Uses W,A,S, or D to move the character.

4. Player Uses Left Click to perform equipped attack(s).
   4.1 Melee Weapon Attack
   4.2 Ranged Weapon Attack

5. As Player Explores:
   5.1 Weapons Are Collected
   5.2 Worlds Are Visited
   5.3 Puzzles Are Solved
   5.4 Bosses Are Defeated
   5.5 Obstacles Are Cleared

6. Character's Health Is Updated Based On Damage And Healing.

7. All Progress Is Tracked By Character System.

Preconditions:

Game is installed and running successfully.
Player has access to keyboard + mouse.

Exceptions:

No input detected.
Invalid key pressed(Ignored).
Game freezes during movement.

GABE (PLAYER CHARACTER USE CASE SCENARIO)

1. Player Arrives At A Checkpoint.
   1.1 Player Given A Button To Save Game.

2. System Captures Active Character State:
   2.1 Character Health
   2.2 Weapons Collected
   2.3 Puzzles Solved
   2.4 Obstacles Cleared
   2.5 Bosses Defeated
   2.6 Worlds Entered

3. System Writes Save Data To File.

4. Player Is Alerted A Save Has Successfully Taken Place.

5. On Menu Screen, Player Has Option To Load Game.

6. The Load Engine:
   6.1 Accesses Save File Location
   6.2 Loads The Saved Active Information Into Current Active Information

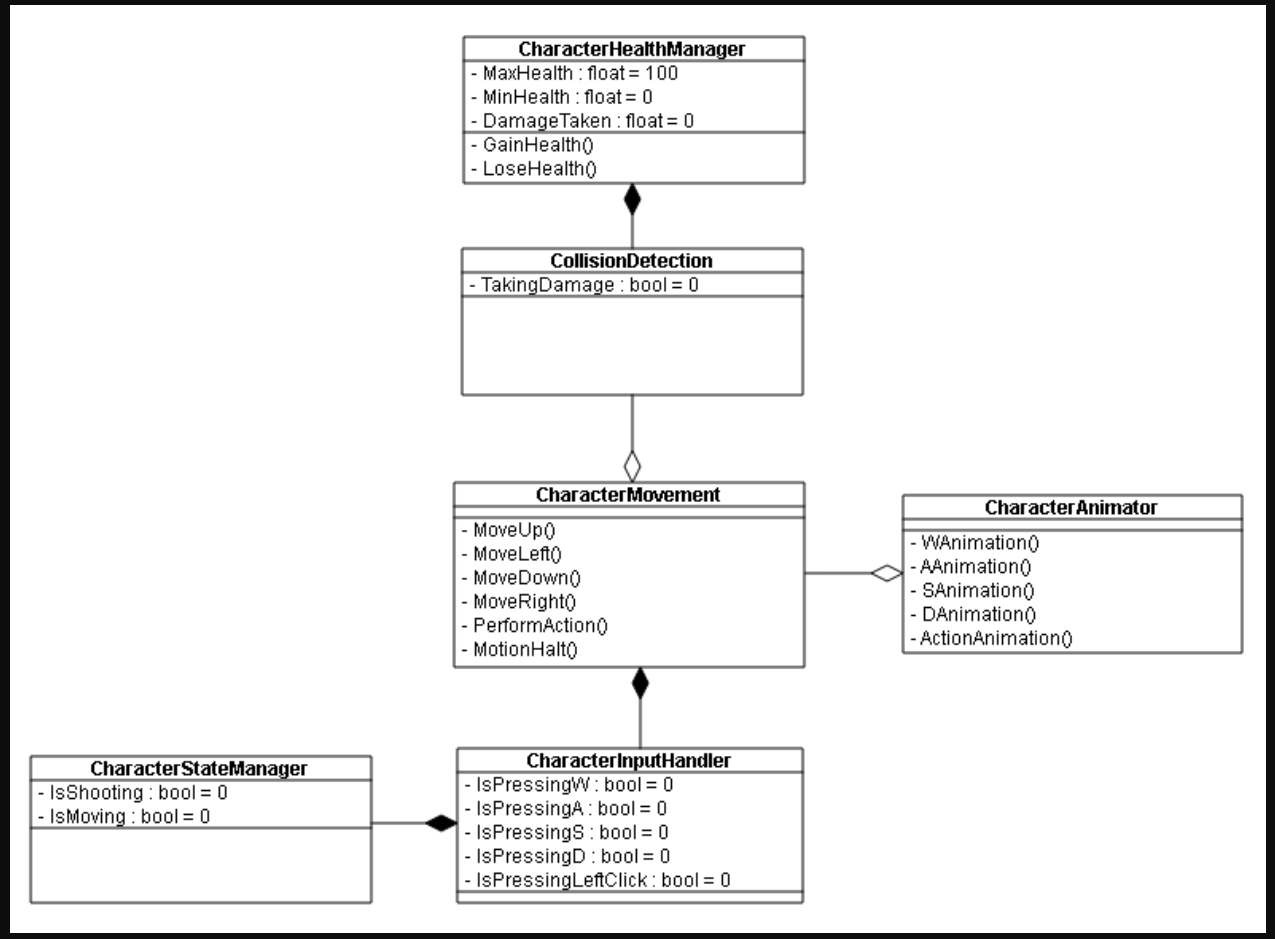
Preconditions:

Game is installed and running successfully.
Menu is accessible
Save system is enabled
Load system is enabled
Storage access is available to system

Exceptions:

If save fails, alerts player and retries save
Game crash during save, system attempts rollback

GABE (PLAYER CONTROL CLASS DIAGRAM)



GABE (PLAYER CHARACTER CLASS DIAGRAM)