Project Summary
Project Title: *Soul Keeper*

Team Members: Vince Curran, Kevin Vo, Logan Park

Description: A short, 3D ‘souls-like’ desktop game made in Unity and C#. It will feature a single dungeon for the player to fight through, with a boss at the end.

Project Requirements
Functional Capabilities: Class Selection, Combat System, Checkpoint System

Possible Capabilities: Save and Load between Games, Leveling Up, Stat Tracker, Achievements

Constraints: Playable on Windows, Mac, & in Browser, Single-Player Only

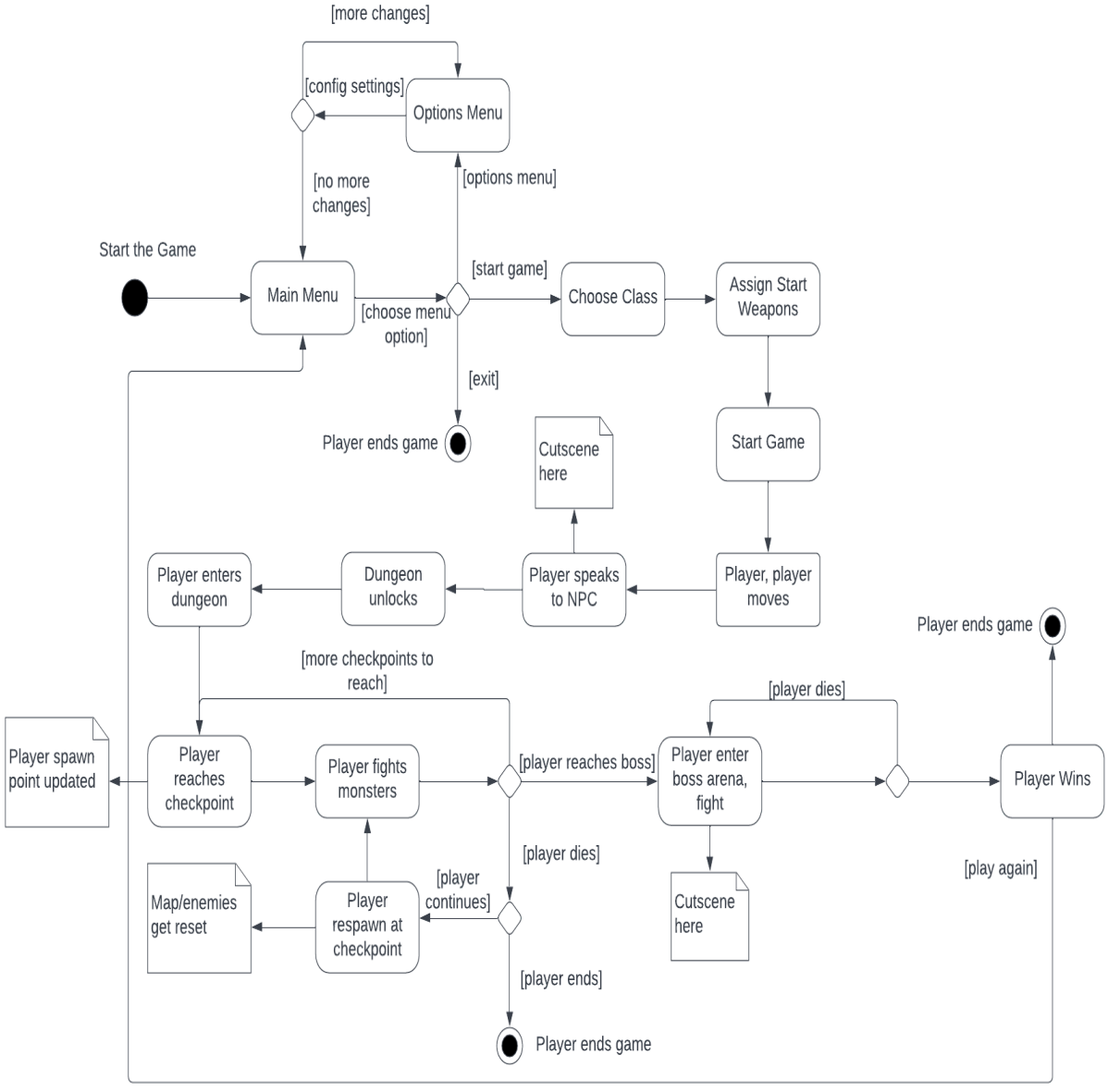
Non-Functional Characteristics: Low-Poly Art Style, High-Performance (avg. at least 60FPS on common machines)

Users and Tasks: Use Cases
- How many users?
    - 1 instance with 1 user per machine
- What tasks do they need to accomplish?
    - Be able to start the game
    - Be able to use basic game controls like movement, combat & interaction to play through the game
        - WASD to move
        - Mouse to look
        - Ctrl to crouch (hold)
        - Shift to sprint (hold)
        - Space to jump
        - Left Mouse to attack
        - Right mouse to secondary attack
        - Q to block/parry
        - H to heal
        - E to interact

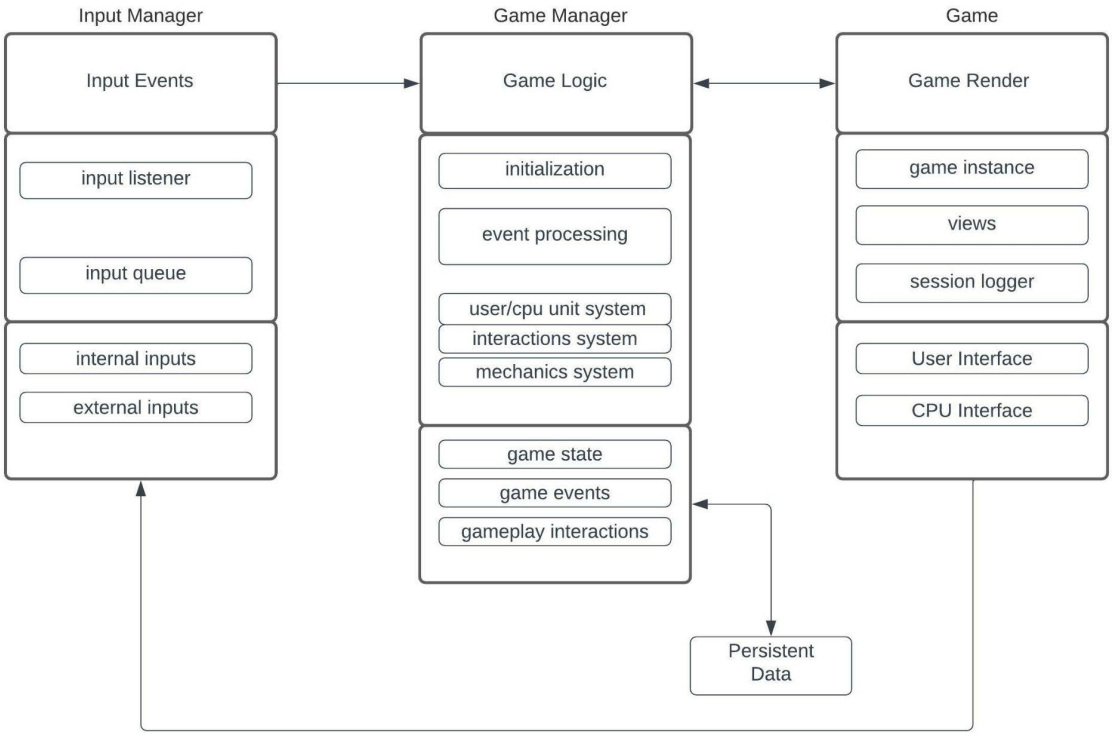
    - End the game by killing the boss or by choice
- Use Cases:
    - Combat
        - Moving, Mouse Moving, Crouch, Sprint, Jump, Attack, Parry, etc.
    - Start Menu
        - Clicking different options and getting responses from the programmed functions.

- Pause Menu
    - This works similarly to the Start Menu
- Check Points
    - Once a player reaches a certain point in the game, they will trigger the check point interaction.

Activity Diagram



Architecture Diagram



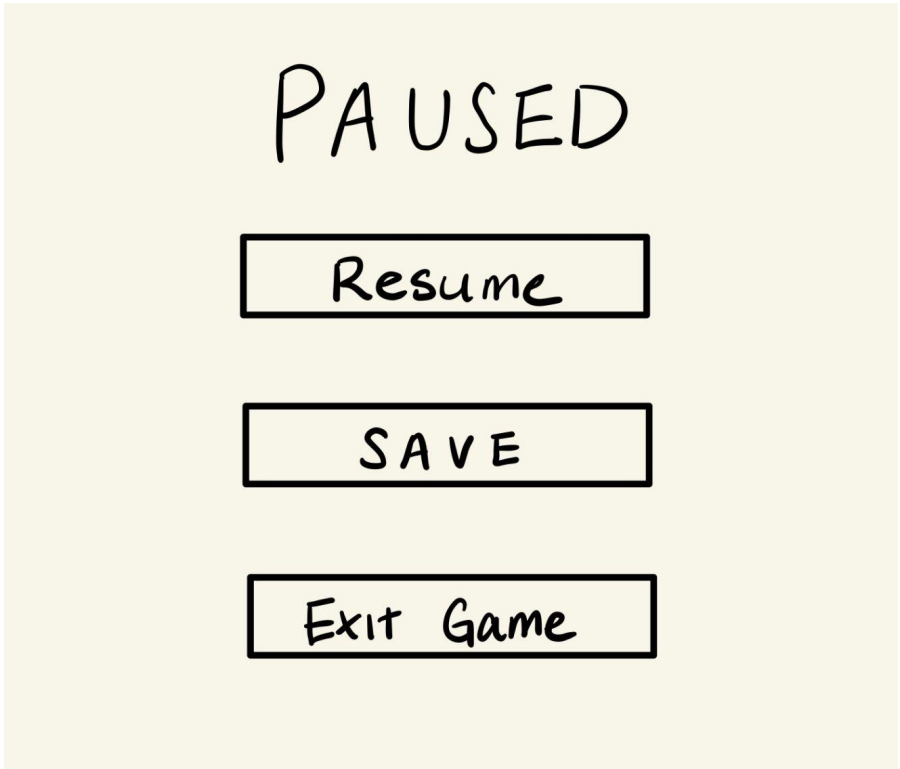
Data Storage

- Data will be kept in memory.
- If we implement a ‘Save and Load’ (Persistent Data) feature:
    - We will save the game state by serializing necessary data to a locally saved JSON file. It will then load by deserializing one of these JSONs. Users will have the option to save multiple games, choose where to save them, etc.

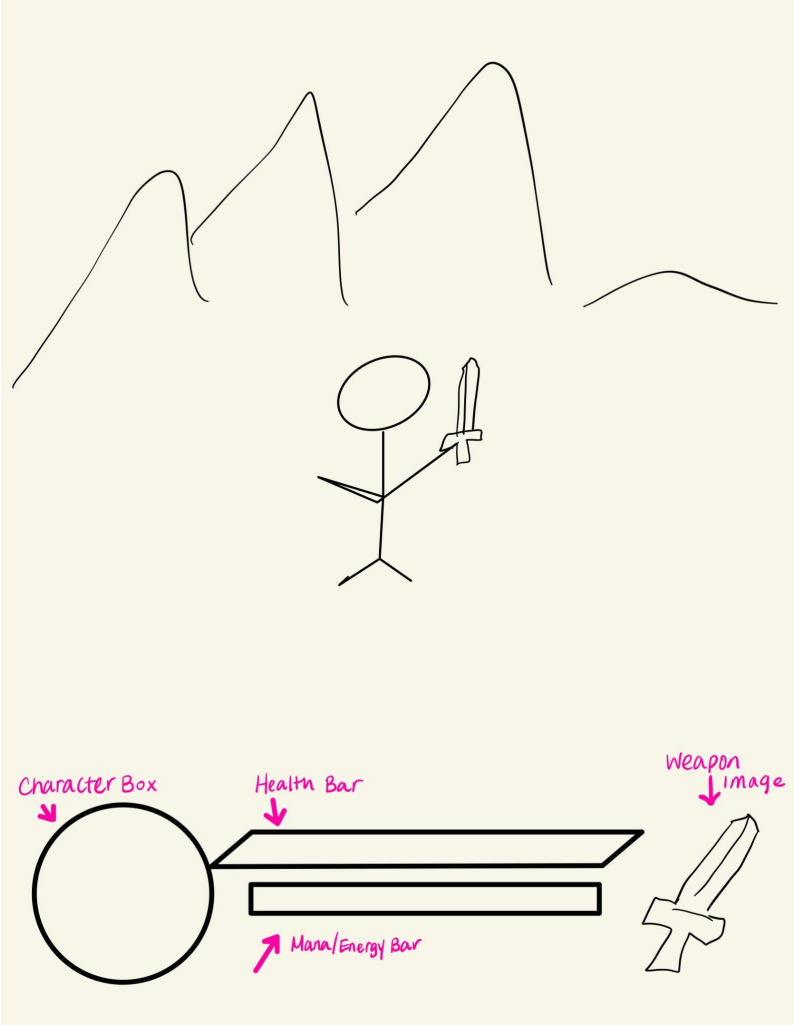
UI Mockup/Sketches

Main Menu:

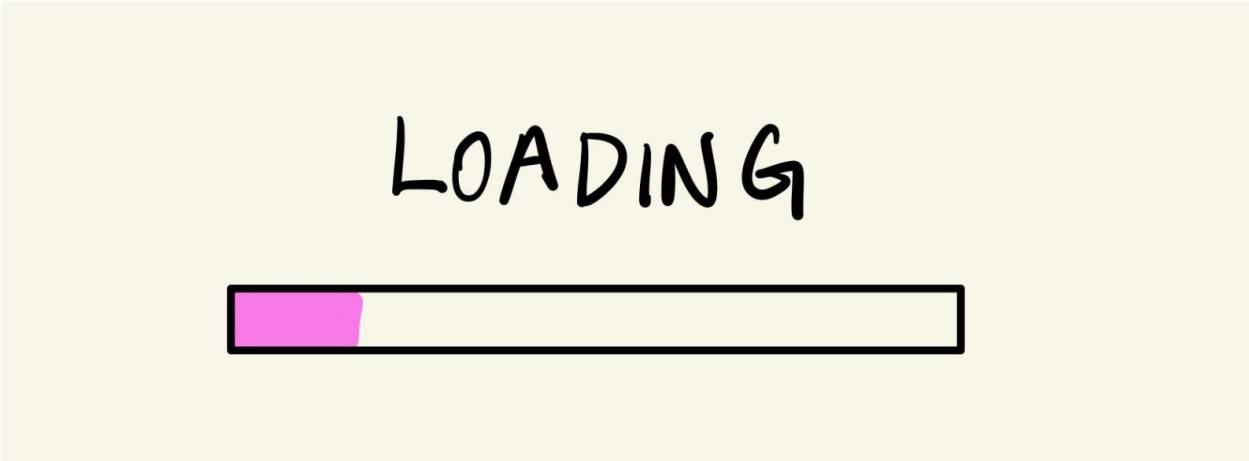
Pause Menu:



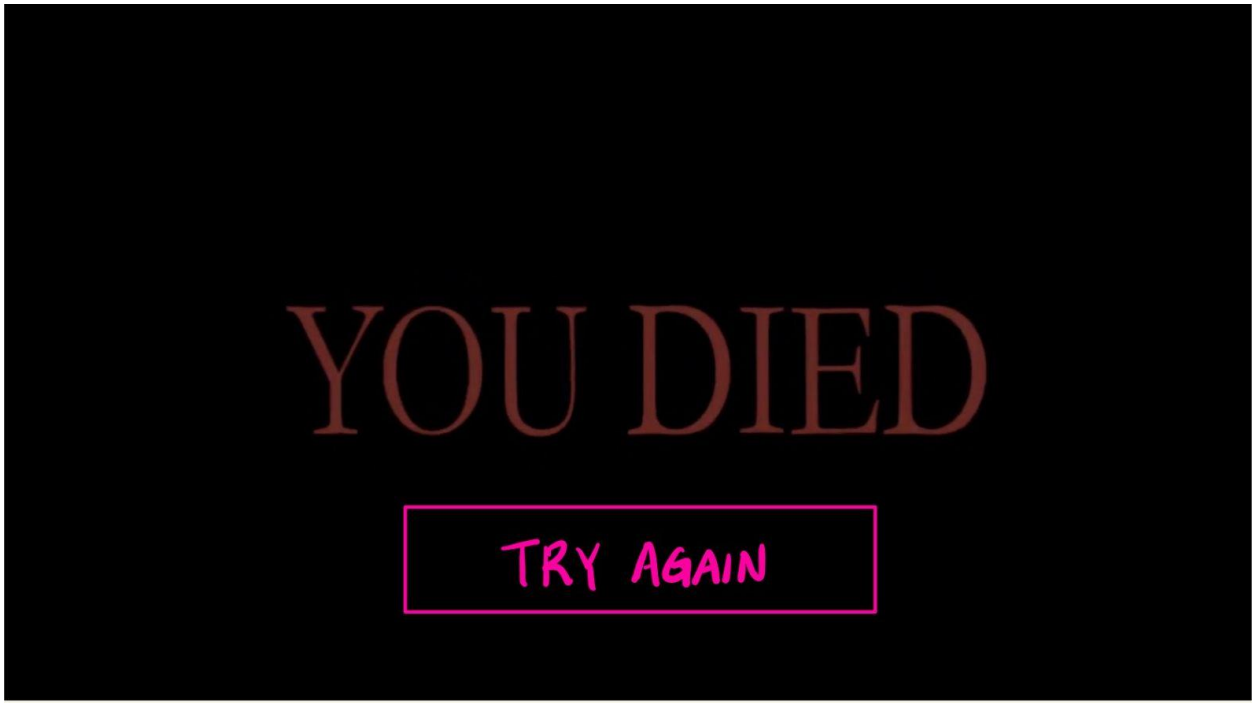
Main UI Overlay



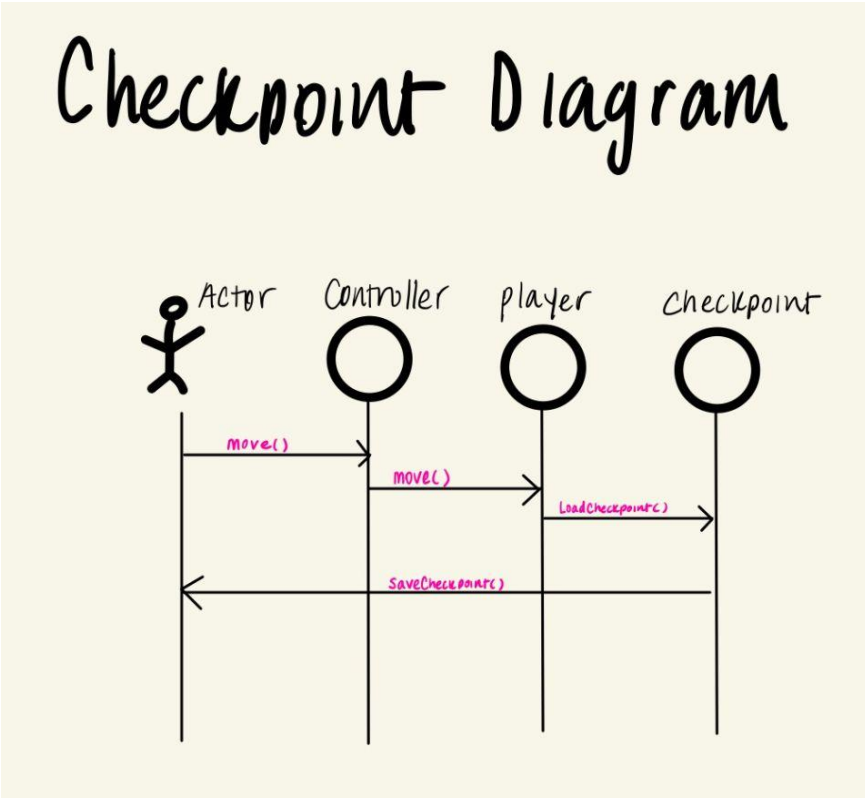
Loading Screen



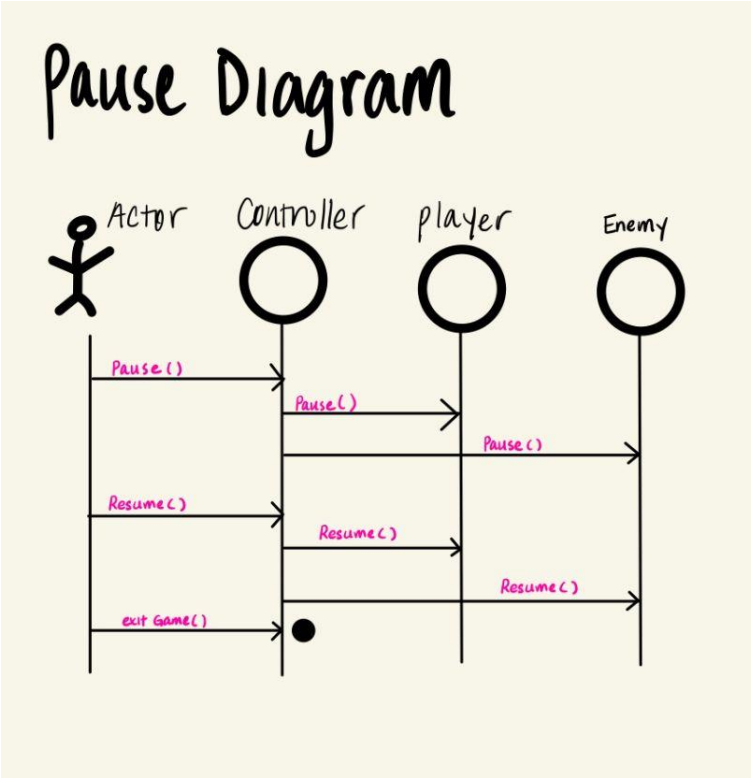
Death Screen



User Interactions/Sequence Diagram

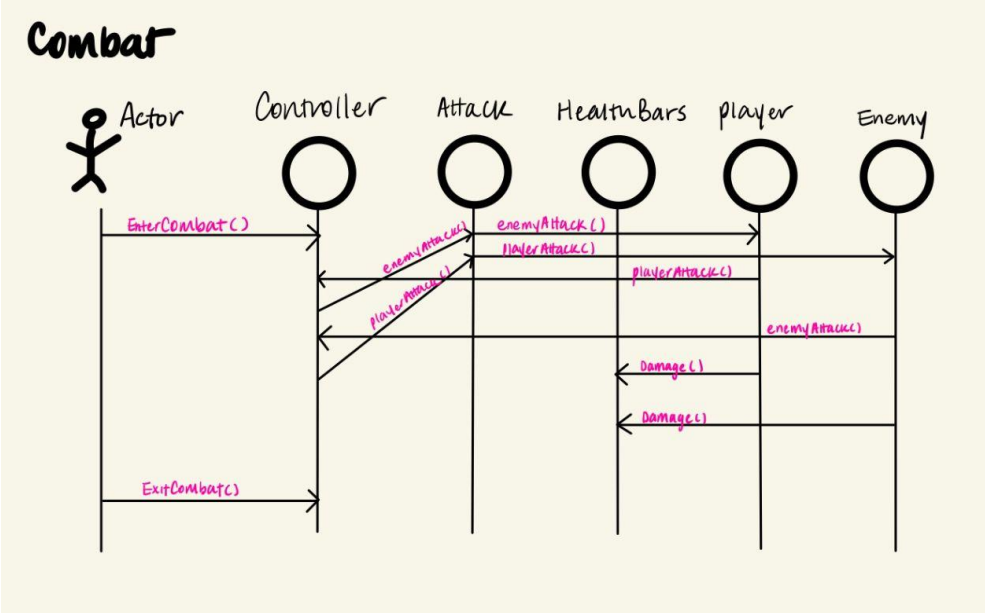


The user inputs the move functions and once the user enters the checkpoint, the loadCheckpoint() is activated, which saves the check point to the system.

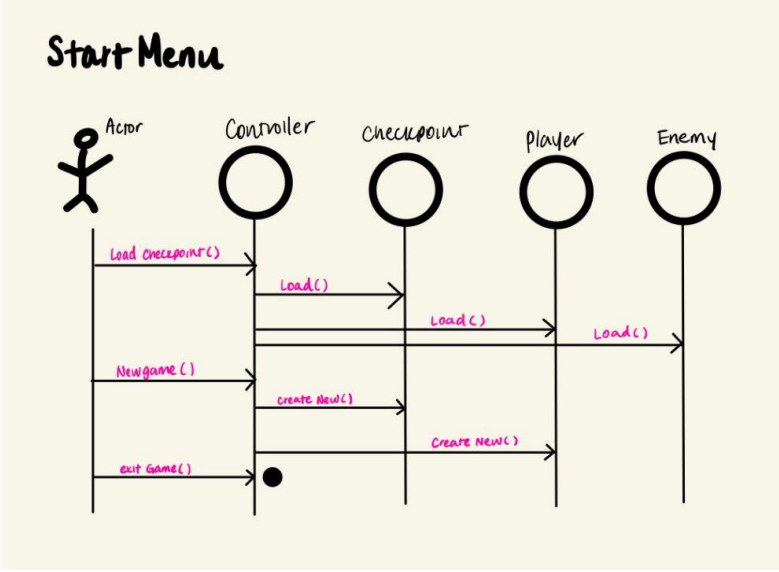


The user presses the button which pauses the game, and the pause function is passed to all objects that are interacting and

moving within the game. If the user chooses to resume the game, it will pass the resume function across all objects and if the user chooses to exit the game, it will simply exit out of the game.



The combat feature is determined by the combat functions that the user chooses to press. Once any combat action has been activated, the player will enter combat mode. Once this is activated, the enemy and player will be detecting which attacks have dealt damage and which attacks have not dealt any damage. Once the player exits combat, it will trigger the exitCombat(), which entails that the player is no longer in combat.



The start menu has a few basic options: Load Game, New Game, Exit Game. Depending on each of these interactions that the user chooses, the controller will determine which actions need to be delegated. If it is exitGame(), it will simply stop executing the application.

Class Diagram