Group Deliverable III

for

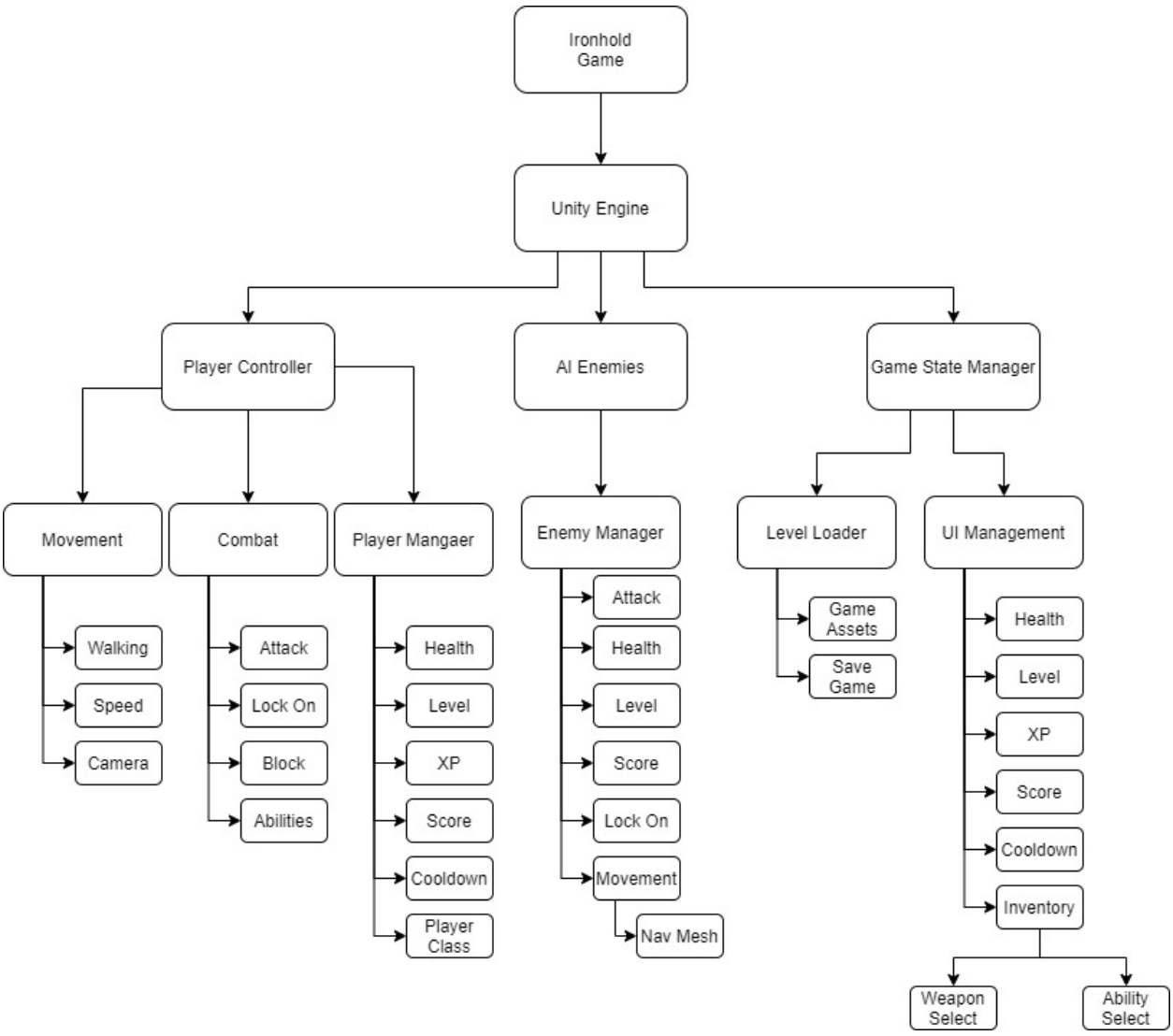
Ironhold

Version 1.0 approved

Prepared by Ryan Floyd and Orion Mendes

Goonsquad Games

11/05/2021



Functional Decomposition Diagram

Components of our System:

- Player Controller
    - The Player Controller is the aspect of our system that the end user will interact with directly through input on a mouse and keyboard to control the movement of the player and camera as well as to take part in combat.
    - There is also a portion of the player controller that runs in the background that maintains all of the stats the player currently has and will interact with other portions of the system.
- AI Enemy Controller

    - This component contains all of the scripts and pathing that each of our enemies will access. This includes the animations, enemy model, Artificial intelligence, combat mechanics, and other minor attributes.
- Game State Manager
    - The game state manager is a backend system that will keep track of all of the UI elements and scripts required to run the game and will allow us to monitor any issues that pop up during development.
    - The level loader controls which scene the game is on and will load levels accordingly once a level is completed.
    - The UI manager will manage all of the UI assets we have to create an informational layout that will help the user when playing the game, as well as to add functionality to our loot mechanic where you can change out the different weapons you wish to use.

Architectural Style:

While we do not know what architecture style the Unity engine is based on, our style that we are using to develop the game will be more of an object-oriented style due to each object having its own properties and having to return certain values to update the system. For example, we need to have colliders that return collision information for when the player or enemy gets hit, return health values to update the UI, and many other components.

Evaluation:

- Reusability
    - We plan on making this game and possible adding to it in the future to implement an online mode but this is trivial at this point and is not guaranteed. We are making some of our scripts so that we may use them in the future to help speed up development of other projects. Other than that, the game/system as a whole will not be reused.
- Reliability
    - When developing the game, we are aiming to make the game run as smoothly as possible with little to no failures. This comes from both the programming perspective of the game as well as design features that will ensure that you cannot break the game intentionally.
- Maintainability
    - Once the game is finalized and published, we plan to fix the game should any other issues pop up in the future or while people are playing our game. After this, we do not plan to maintain the game any further unless we move forward with our plan to implement online functionality.
- Testability
    - Seeing as how we are developing a game, we have an easier time testing the game as we have already made a working prototype that shows basic combat, player

movement, and AI. From there we are able to test the playability of the game during development.

- Performance
    - For the game itself we are aiming for the game to run on a modest laptop due to it having to be displayed and played at the Game Expo. We are aiming for the game to run at 60 fps and with little latency to have a smooth gaming experience.
- Portability
    - The game will be able to be downloaded as an executable from https://goonsquad.itch.io/ironhold and can run on most platforms that can run Unity
- Security
    - We have no need to be concerned with security right now. This will be a greater concern that we will consider if and when we move on to implementing online functionality.

Issues

When designing the game, we need to consider the scaling of the enemies, damage, and health among other features to ensure a fun and playable experience. Should just one component be off it can throw the whole game out of balance and cause it too not be enjoyable.

In addition, when programming the game, we need to consider the limitations of the engine so that we do not overload the system and cause a crash. We also have to be cognizant that all components that interconnect the various parts of our system work properly and will not cause any unforeseen issues. Other than those issues, we just need to focus on mitigating human error during the development process by testing as much as possible.

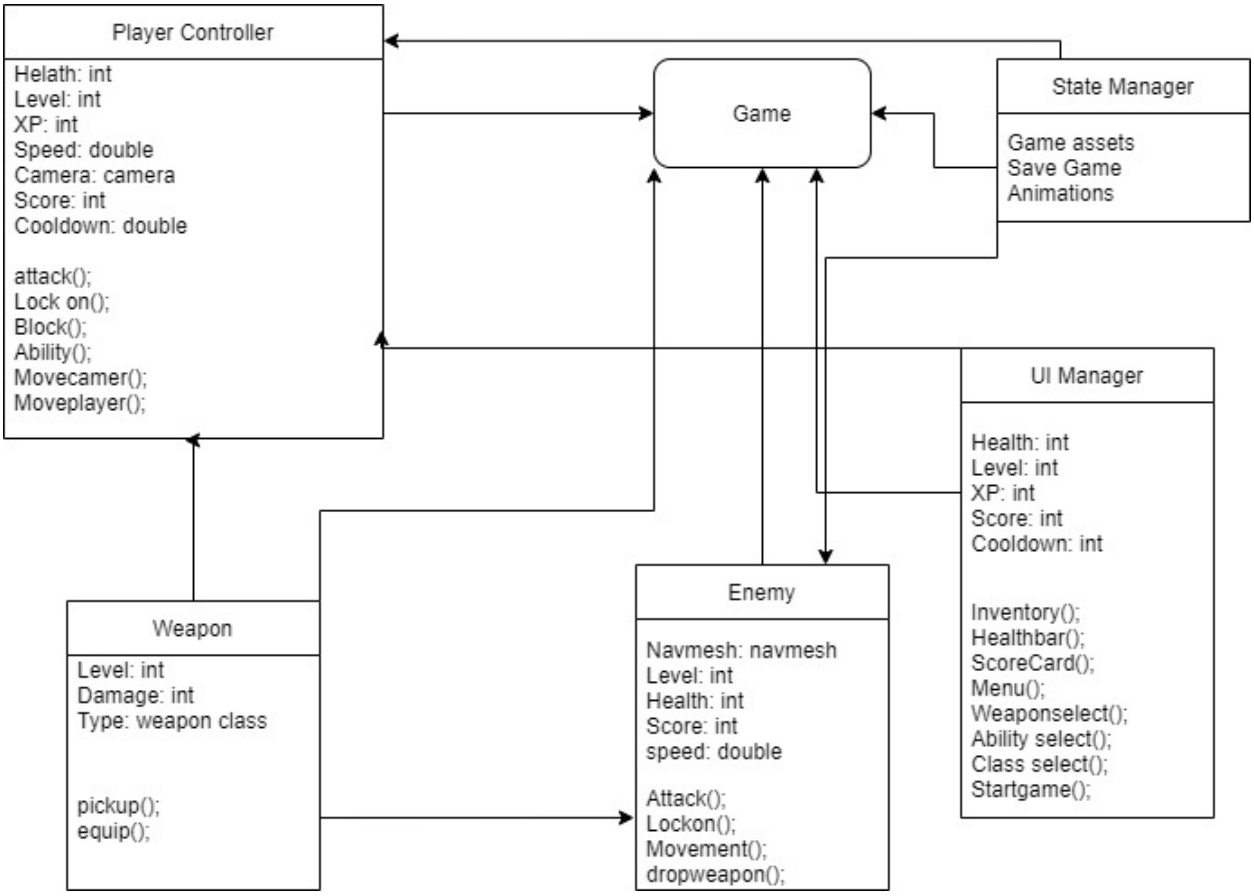
Prototypes

We have created a basic prototype that allows us to test out the basic player movement, combat mechanics, and the enemy AI. We believe that prototyping will play an important role in allowing us to utilize different design methods. We have already identified areas where we can improve from the original conception of our basic movement mechanics. Luckily, the Unity engine has the ability to compile and play the prototypes during development in real time so that allows us to test our game when we implement any new feature.

Technical Difficulties

Any technical difficulties we encounter will most likely be at the fault of us, the developers due to some inefficient code or any other sort of error. We may also run into some difficulty with the Unity engine but that is unlikely due to their vast support team and updates that fix known errors. If we do run into some technical difficulties we will be sure to take the necessary steps to diagnose and fix said difficulties.

UML Class Diagram



UML Message Sequence Chart

Due to the extreme interconnectivity of the game and its components a UML Message Sequence Chart is not a viable way to properly represent the data flow of the game and how the parts interact with each other as it would be too complex.

UML State chart Diagram