Final Group Deliverable

for

Ironhold

Version 1.0

Prepared by Ryan Floyd and Orion Mendes

Goonsquad Games

Finalized on: 11/24/2021

Table of Contents

Problem/project description

The problem/project that we undertook this semester was to make a third-person fighting game. This game has gameplay mechanics focused on engaging player versus computer combat that highlights the software engineering process. The game was developed on the Unity game engine using C# to program the scripts and was made from the ground up by a team of 5, 3 being from the Software Engineering class and two others assisting with various portions including art and sound. The goal was to create an engaging game that would be displayed at the Florida Polytechnic Game Expo and would be released to the public for free on itch.io.

Software Development Life Cycle

In order to make this game we decided on using the Prototype model of the SDLC to make our game as efficiently as possible. Throughout the life cycle of the game's development, we created various prototypes to test our designs and to revise our designs so that they may be better in the end. Due to the volatile nature of game development and its rather unorthodox software engineering process, this model was the best option for our group as it gave us a chance to see what was working for the game and what was not. It was also a model that is followed in Industry as large game development companies also utilize prototypes to develop their games.

Project Scope

We aimed to be realistic with our project scope for this game. We originally began with a small 3-man team but gradually added more members to the project to share the workload and help with the development process of the game. From there we revised our original requirements for the game once we gained an understanding of how much time we could dedicate to this project. We aimed specifically to develop a multi-level experience for the game with each level being different in style while maintaining the same core game mechanics and gameplay.

Project Team Organization and Contributions

Ryan Floyd: Project Manager, Game Designer, UI scripter, and Art Assistant – 35%

Orion Mendes: Gameplay Programmer, Unity Specialist, Animation, and Lighting – 45%

Alexander Vasta: 0%

Non-Software Engineering Members

Keagan MacDonald: Lead Artist and Assistant Game Designer -10%

Andrew Reimer: Music Lead and Sound Engineer – 10%

Overall Description

Product Perspective

This product is somewhat of an entirely new self-contained project. The reasoning for the project being "somewhat" self-contained is due to the fact that our group is building off the idea of a previous project that had similar goals but was not as advanced or planned out as this project. We are viewing that previous project as a pseudo prototype, or rather as a proof of concept.

Product Functions

- Solid Gameplay
- Skill System
- Combat
- Inventory

User Classes and Characteristics

Mostly all users that use this product will be gamers. we are aiming for ages above 13+. Only technical expertise is the ability to understand modern gaming schemes and elements and how to unzip a file.

Operating Environment

Our software will be operating on the Windows 10 Operating system. We are building off of the Unity Engine, a free game engine that allows us to efficiently develop for Windows platforms free of charge. In addition, all of our physics simulations and game systems will be handled by the engine. The final product will be distributed through itch.io which is an online game distribution site.

Design and Implementation Constraints

As of now, the only constraint is time and developer skill level. all the tools needed to make this application are in use and no other tool is needed as of now or for future development. the major limiting factor is our skill level in developing the app which makes development time longer.

User Documentation

With this game, we will be creating a basic tutorial on how to download the game from itch.io and the overall goal of the game. This will include the game objective as well as the button input maps so players are aware of the controls. The game can be played on a gamepad and keyboard and mouse so there will be controls schemes for both posted for the user to choose from.

Assumptions and Dependencies

Seeing as how we are developing in the Unity Engine, we run the risk of running into limitations with the engine or even bugs that have yet to have been fixed with the engine. This could possibly hinder our development time, or it could lead to the final product being faulty. We do not have access to the source code of the engine to fix any issues that we run into so we need to assume that engine issues are a possibility. Luckily, the Unity Engine is well maintained and is well supported by its developers and the community to help with any issues. We also need to ensure that any platform that we wish to expand to can support the Unity Engine to allow our product to run properly.

External Interface Requirements

User Interfaces

Our game will consist of user interfaces, not in the normal way that a normal program would but will have user-operated interfaces so that they may interact with the game and its systems. We will have a start menu that will include the options to start a new game, a help tab to give the players their main goal in the game, and give them information on the games control scheme, and a credits tab that will give credit to the developers of the game. We will also have in-game user interfaces that will dynamically update based on the game. This will include the player stats UI which will include the player's health, time, score, and ability cooldown.

Hardware Interfaces

Our game/project will not directly be handling the hardware interaction that is required for the game to run. This will be handled by the Unity Engine that we will be developing our game on. The engine was made beforehand by a third party and is freely distributed to allow developers to utilize its vast and fleshed-out systems to help increase project efficiency so that a custom game engine does not need to be created.

Software Interfaces

Our game will be interfacing with the Unity Engine and its subsequent systems to add functionality to our game. We created the scripts for the game using the C# library and the Unity C# plugin for the Visual Studio IDE. The Unity C# plugin has a library of functions developed specifically for the Unity Engine to allow for better functionality and will allow us to create the proper scripts to properly utilize the software.

System Features

Combat

Description and Priority

High priority-9

Combat of this game is the main feature of this game so it needs to be fast pace and fun. The combat will be like most third-person games. But the difference is the skills-based leveling system.

Stimulus/Response Sequences

Left Click - Attack clicking left click multiple times creates an attack chain

'E' input on keyboard - uses ability/skill to create interesting attack combos

Right Click - Blocks incoming attacks

Functional Requirements

- **Health System** - need to be able to add remove health of characters in the game
    - Time needed: Completed
- **User Input** - needed to be to accept user input
    - Time needed: Completed
- **Characters** - need characters for the game to have a function combat system
    - Time needed: Completed
- **Animations** - need animations for the characters to do so the combat is fluid.
    - Time needed: Completed

REQ-1: Attacking

REQ-2: Blocking

REQ-3: Skills

Other Nonfunctional Requirements

Performance Requirements

In order to make a smooth gameplay experience, we are targeting that the game will run at 60 frames per second. This is the general standard of modern games and at that frame refresh rate will allow us to adequately control the physics engine in Unity that is tied to the refresh rate.

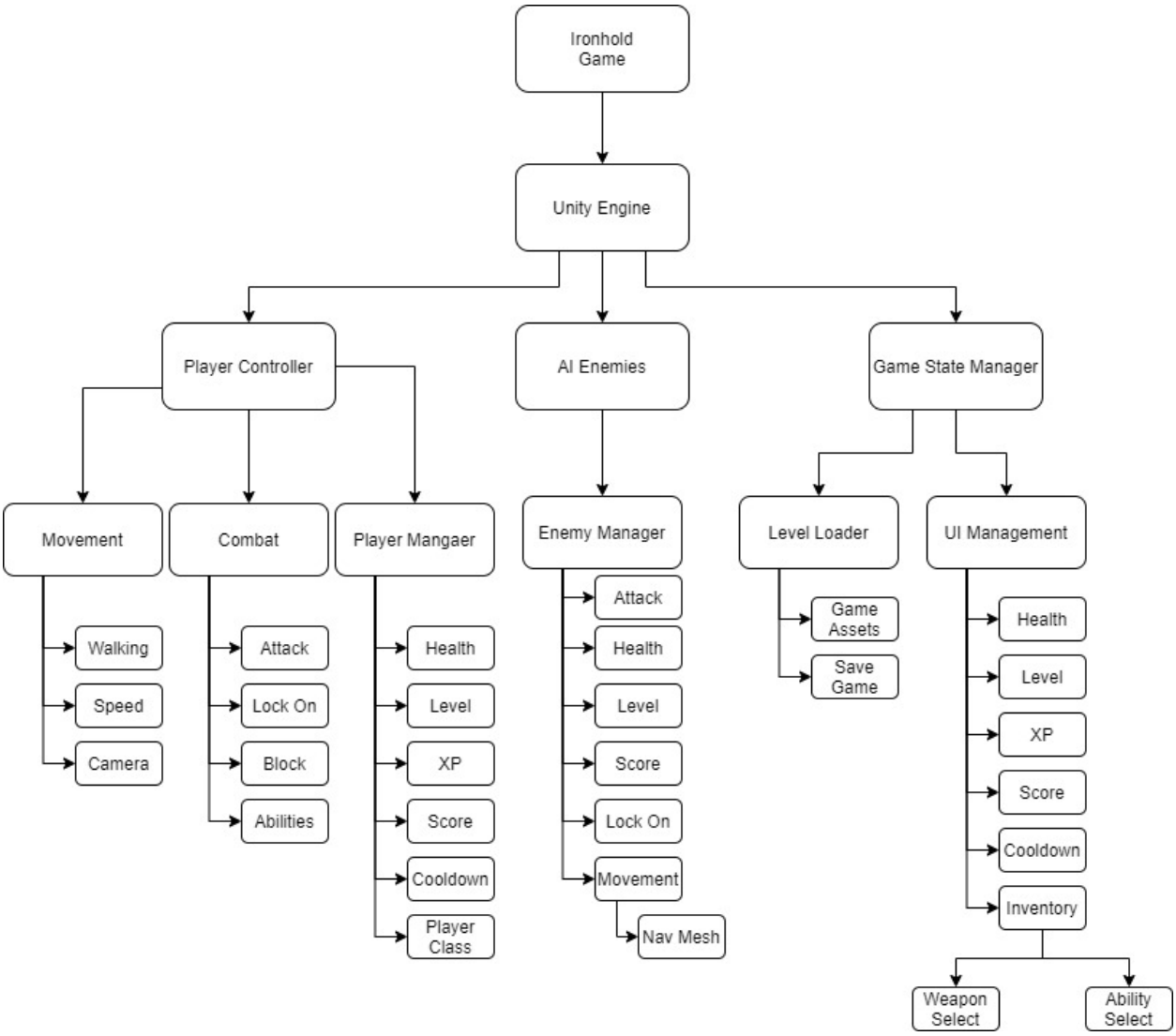
Safety Requirements

Our general safety requirements are in regards to the user's health and the health of the hardware being used. The Unity Engine has been built and optimized to run on practically any Windows computer so we are confident that our game will follow the general safety guidelines of the engine's creators to not intentionally damage the hardware of any individual. In addition to that, we always recommend to players that they take periodic breaks from the game to avoid eye strain, fatigue, sleep deprivation, or any other serious medical effects that come from sitting in one place for a long period of time. In addition, we will be making a notice on the game distribution site if we believe there are any hazards that would be classified as possible events in the game that may trigger an epileptic seizure to ensure the safety and preservation of our players.

Security Requirements

At this time, we are not aiming to add in any security requirements simply because we are not making this game connect to any network and no vital information is being created or stored. In the future, we will begin to consider security requirements based if online multiplayer is implemented where we would need to consider network security options.

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.

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 needed 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 needed to consider the limitations of the engine so that we do not overload the system and cause a crash. We also had 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 needed to focus on mitigating human error during the development process by testing as much as possible.

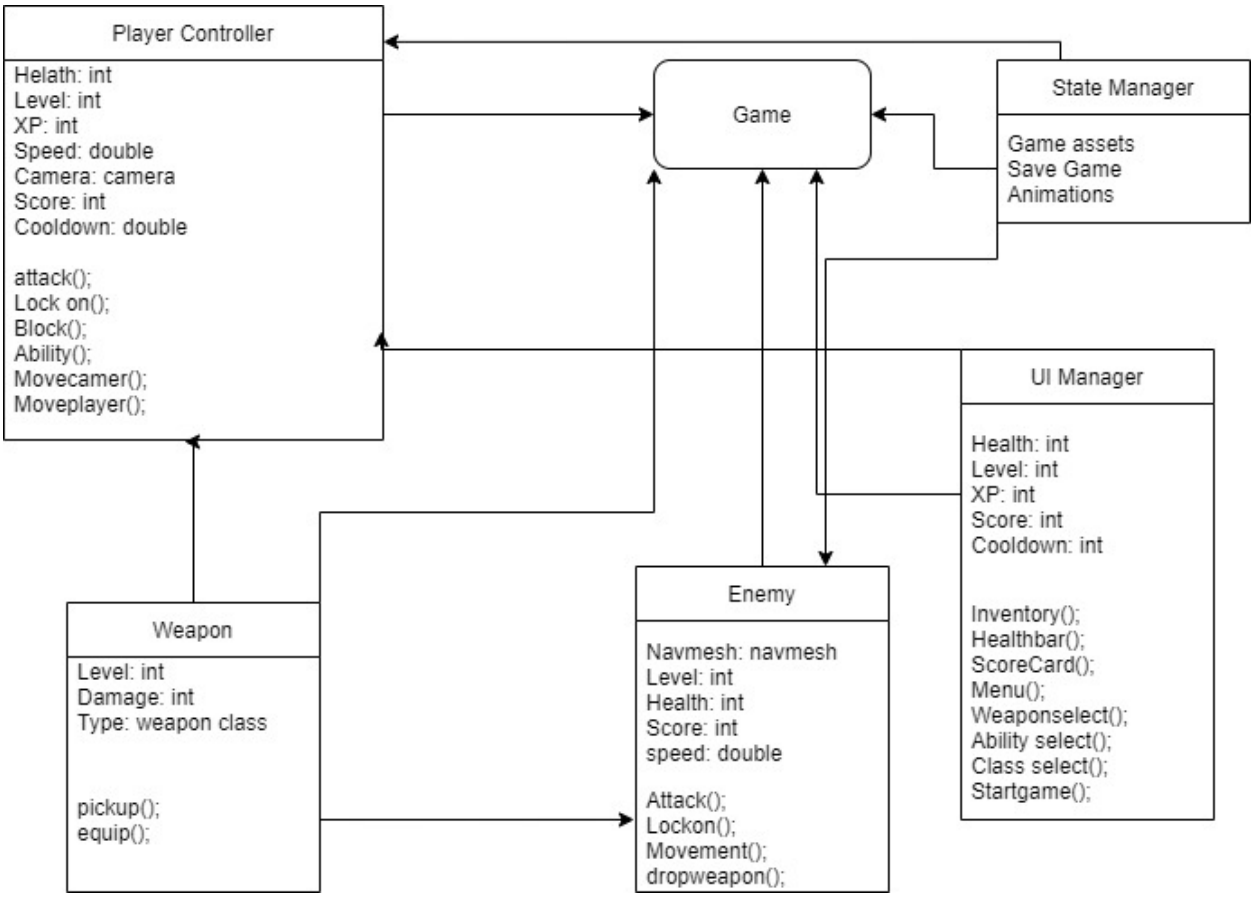
Prototypes

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. Using the prototype, we 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. From our first prototype we noticed a design issue where our combat worked but did not feel as good as we wanted and we decided to rework it and made it feel much better.

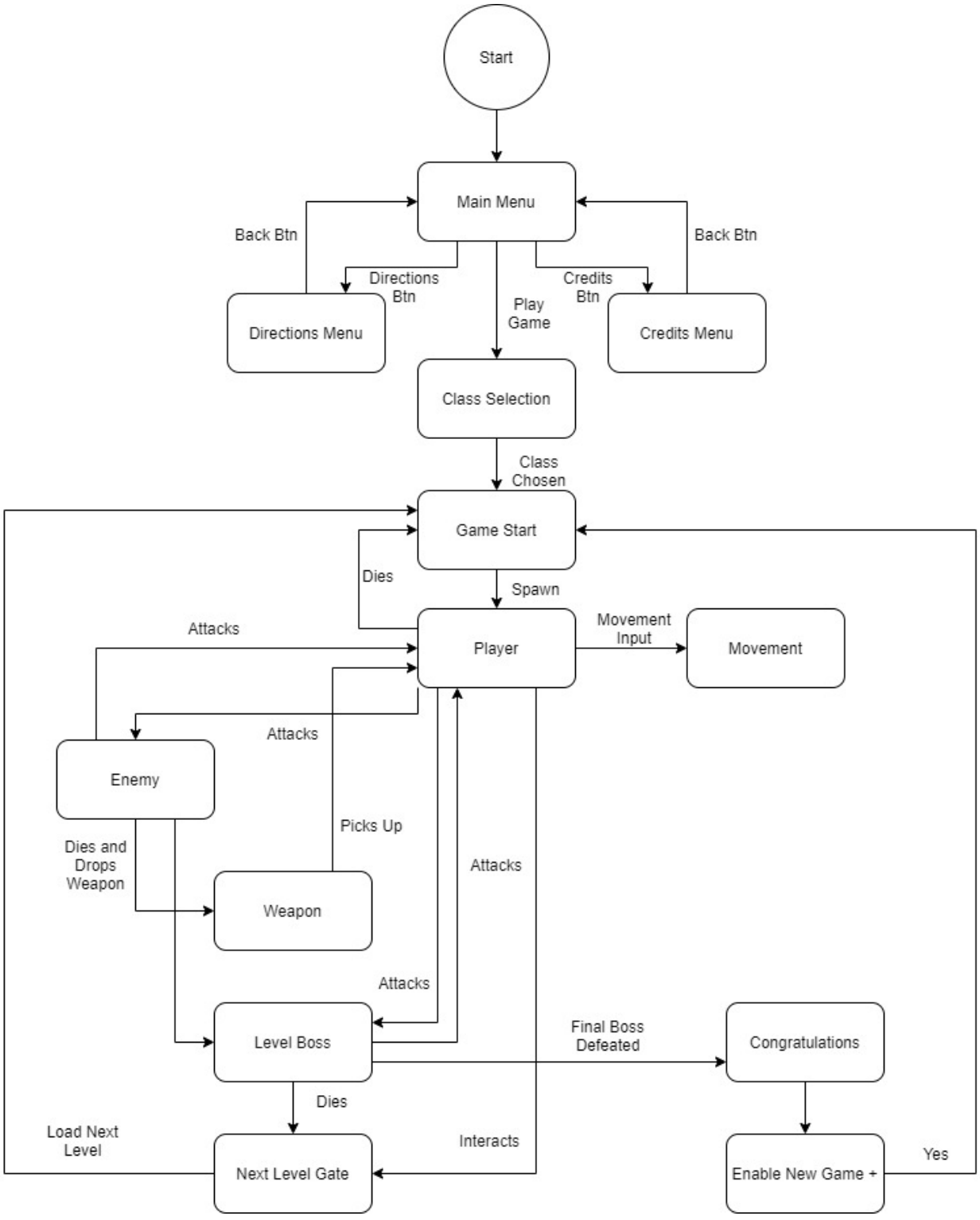
Technical Difficulties

During the development we ran into a few issues involving the wireframing and lighting of our Voxels models in the Unity engine. With a little tweaking we managed to fix this issue and have not had any other issues of note during development.

UML Class Diagram



UML State chart Diagram



Testing Report

Unit Testing Report

Within our development unit testing was done in a unique way thanks to the Unity engine and how we can compile code and run it in a live setting. Whenever we developed code for the game, let's say the player controller, we were able to upload it to the Unity engine and set a model to utilize the script and we were able to test the functionality of the script. While this type of testing leans more toward integration testing, this is the best way to test individual modules of code for their functionality.

Integration Testing

For the integration testing we set up the Unity environment to run with any combinations of code that we needed to test. For instance, to test the combat mechanics we needed to test the Enemy scripts, attacking scripts, player controller scripts, and any other scripts that we deemed necessary. Within this process we would verify whether the code worked or if we found any issues that needed to be addressed. This form of testing was most effective for us as we were able to identify a number of issues and fix them quickly during the development process. We did this with a number of combinations of interacting components and did so with an increasing number until the game was fully completed. Running the game environment live in Unity allowed us to effectively execute Integration tests.

System Testing

As of the writing of this document system testing could not be completed. However, we have outlined our process for system testing. This process involves finalizing the build within Unity and exporting the game to be an executable application that we will then test on the system that it was developed on. When we have verified that this test has been completed and we see no issues with the build, we will then upload the file to itch.io to be downloaded on to a separate system to test if the build is still stable and has no significant issues. From there we will consider the game live and playable for all people in which time it will be made public. Our criteria for a stable and good system test are little to no issues with the game and that it is playable on various types of systems with little hinderance.

Lessons Learned

Throughout our time developing this game we have all gained more experience and knowledge of the Unity game engine and what we can accomplish with it. In addition, we have gained experience in game design principles, C# programming skills, art design, and the software engineering life cycle of a game. The software engineering life cycle of making a game is more complex than we initially believed and has given us great insight to the complexity of larger games and what it takes to make a good game. One challenge that we all faced during the development of this project is the correct utilization of time management. It is difficult enough to have to develop a game in such a short amount of time but even more so when you are a full-time student with other various projects and responsibilities. With this experience, all of the members have gained valuable knowledge of how to use their time appropriately and effectively.