REIGN (Game Demo)

Lewis Brown

CST-451 Capstone Project Final Architecture & Design

Grand Canyon University

Instructor: Professor Reha

Revision: 2.0

Date: 11/13/19

**ABSTRACT**

Reign is a round based hack and slash video game where the Player will fight rounds of enemies leading up to their inevitable demise. The player will defeat enemies and can pick up weapons and other shields. The player will have the ability to traverse the map, attack enemies, and block enemies attack. Once a round starts multiple enemies will come after the player and once that player has defeated all enemies on the map the round ends and the next round will start. At the start of round 1 the player will start off with a weak sword and a weak shield. In later rounds the enemies will spawn with better weapons and shields. If a player defeats one of those enemies the player will be able to pick up those weapons and use on other enemies. The player will also be able to hold multiple weapons. The player has multiple ways of avoiding attacks such as dodging, running, and blocking attacks.

.

|  |
| --- |
| History and Signoff Sheet |

**Change Record**

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Revision Notes** |
| 11/13/19 | Lewis Brown | Revised UML class Diagram |
| 11/13/19 | Lewis Brown | Created/added sequence diagram |
| 11/13/19 | Lewis Brown | Revised physical diagram |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

**Integrated Instructor Feedback into Project Documentation**

Yes  No

**TABLE OF CONTENTS**

Design Overview 4

Detailed High-Level Solution Design 5

Detailed Technical Design 6

Appendix A – Technical Issue and Risk Log 7

Appendix B – References 8

Appendix C – External Resources 9

Design Introduction

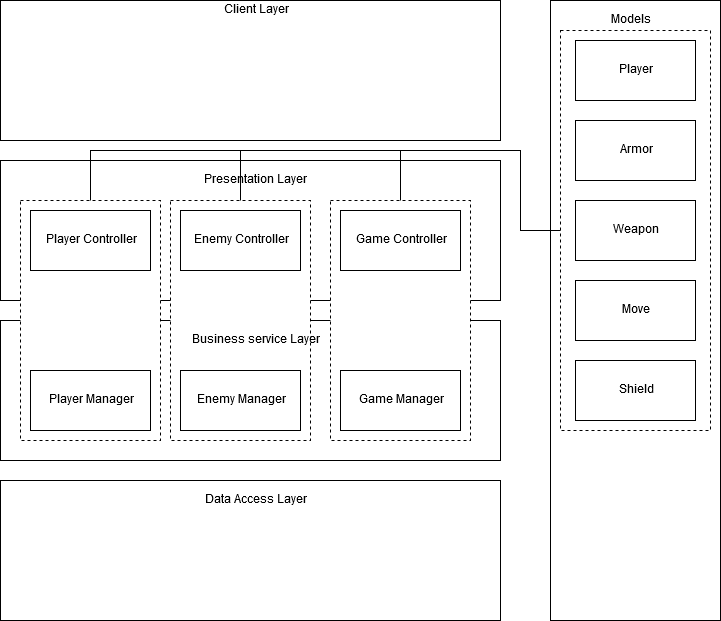
For the game Reign the user will start at the main menu only given two option which are new game and quite game. If the user selects quite game the application will close. If the user selects new game the game will go the loading screen where the game engine is loading all the assets into memory. After the users character spawns on the map with a low-level sword, a low-level shield, and no armor. While that happens the engine runs the scripts telling the game what round it is (round 1), where the enemies are spawning, how many enemies need to be defeated, detects if a player or enemy is hit, and detects if all enemies are defeated. Unity will also run the scripts for the player controls such as if the user presses W the character will move forward, S will move the character backward, D moves the character to the right, and A will move the character left. The user can hold down Shift and press and WASD key and the character will run in that direction. The user can press Space bar to jump as a tactic to avoid being hit. The user can also use Ctrl to crouch as another tactic of avoiding hits. The user can obtain a new weapon, shield, or armor by looking at it and pressing the E key. The user can only obtain a new weapon, armor, or shield by first defeating an enemy that has one of those items. If a player hits an enemy and that enemy blocks the attack no health will be taken away but their armor durability will go down. If a player attacks an enemy and that enemy does not Block both the armor durability and health will go down. The same will happen to the user’s character. The user can also use RMB (right mouse button) to block an attack. The user can press escape at anytime to pause the game. In the pause menu there are three options: resume game, controls, and quite game. If the user chooses resume game the game will resume as normal. If the player chooses controls from the pause menu the list of all controls will pop up. If the user chooses quite game from the pause menu then it will go back to the main menu. If the users character is defeated in game then the game over menu will appear with two options which are new game and quite. Selecting new game from this menu will start the game over with round 1. If quite game is selected for the game over screen then it will go back to the main menu screen.

Detailed High-Level Solution Design

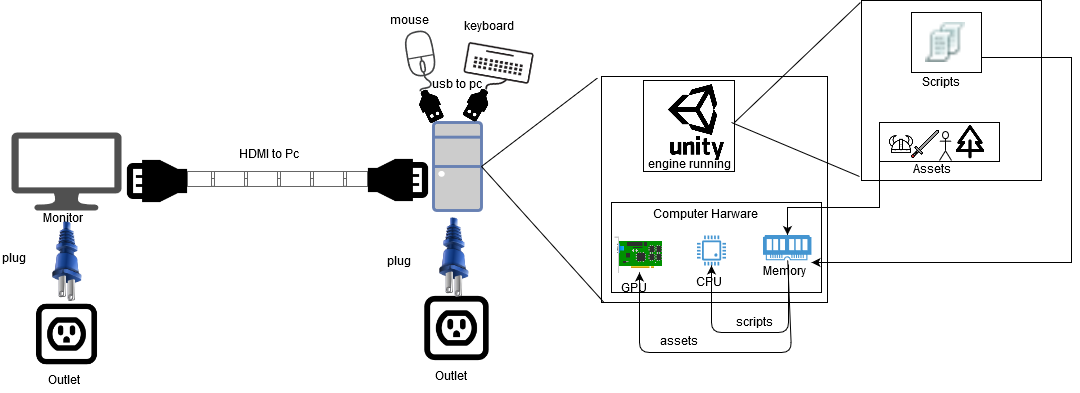
|  |  |  |
| --- | --- | --- |
| Proof of Concepts | |  |
| **Description** | **Rational** | **Results** |
| 1. Unity – is a game engine that the video game will be running on. | Unity will keep the scripts running in the background as well as loading all the assets into memory | The game will be running smoothly with full assets as well as scripts going through animations |

|  |
| --- |
| Hardware and Software Technologies |
| 1 – Unity version: 2019.2.9f1 (64-bit) |
| 2 - Blender version: 2.8 (64-bit) |
| 3 – Visual Studio 2019 version: 16.3.8 (64-bit) |
| 4 - .C# version: 8.0 |
| 5 – OS: Windows 10 Home |
| 6-Hardware game will be running on: CPU: Intel core i7-8750H @2.20GHz (6 core), RAM: 16 GB DDR4,Graphics card: GTX 1050ti 4GB VRAM |

**Logical Solution Design:**



**Physical Solution Design:**

****

Detailed Technical Design

**General Technical Approach:**

The approach to making this game starts will all of the documentation. When all of the documentation is completed, then move to blender where all the assets will be created from scratch. Once all the assets are created, then move to Unity where all the menus will be created. Once all the menus are created all of the assets created in blender are then imported to Unity. Once all in game assets and menus are in unity the process of coding will begin by clicking on an asset, there is an option called behavior that when clicked will open up Visual studios and the process of adding scripts to the scene will begin.

The approach to the design will start with importing all the assets created in blender to unity. Then in unity adding all the assets to a scene. After added to a scene in visual studios all the behaviors are added to an asset such as for a weapon damage, weapon name, and its associated getters and setters are applied. The game controller and game manager will be created and implemented will all the games rules such as how many enemies there are, how many enemies are left, if all enemies are gone move to the next round, and detect whether or not someone is hit.

Meeting/brain storming session:

A colleague agreed to call and explained how scripting works in a game, a bit on MVC, as well as scripting with cycling animation and touched on a bit how to calculate damage in a game.

**Key Technical Design Decisions:**

The OS and Hardware listed under Hardware and Software Technologies is used for this demo because that is the hardware and OS the game will be made on and tested on.

Unity: 2019.2.9f1 (64- bit)

The Unity game engine is capable of rendering full 3D environments, handling complex animations, and uses C# as a programming language. The game engine unity was chosen because the engine was free, easy to use, has an asset store with tons of assets to choose from, and has dozens of online tutorials. Its purpose is to load all the assets such as weapons, shields, armor, characters, and environment into memory. This game engine also keeps the scripts running while playing the demo.

. This engine will be responsible for the following task:

* Rendering the 3d world the player will walk on
* Rendering the character that the player will play as
* Rendering the Enemies
* Rendering the equipment
* Collision of objects and player
* Handling all in game rules
* Putting assets into memory
* Keeping the scrips running while playing the demo

Visual studio 2019: 16.3.8 (64-bit)

Microsoft’s IDE that is able to create classes that are written in C#. It also supports integration to Unity. Visual studios and C# was chosen because unity recommended it and C# is almost required for unity. The purpose is to create the rules of the Demo as well as calculate battle damage and detect whether or not the play or the enemy is hit.

This IDE will be used to create these components:

- Will be used for going from round to round

- Used for combat such as hit points and damage

- Calculate damage

- Basic ai for the enemies

- Collision between two objects

- Calculate how many enemies are on the map alive/defeated

- Mapping controls

- Creating all in game rules

- moving from one animation to the next

Blender: 2.8 (64-bit)

Blender is a 3dmodeling software that does 3d modeling and animation. Blender is chosen for this game demo because it is free and has easy to use tools. Is used for this demo to make all the assets.

Blender will be used to complete these tasks:

- Walking animation

- Attacking animation

- Blocking animation

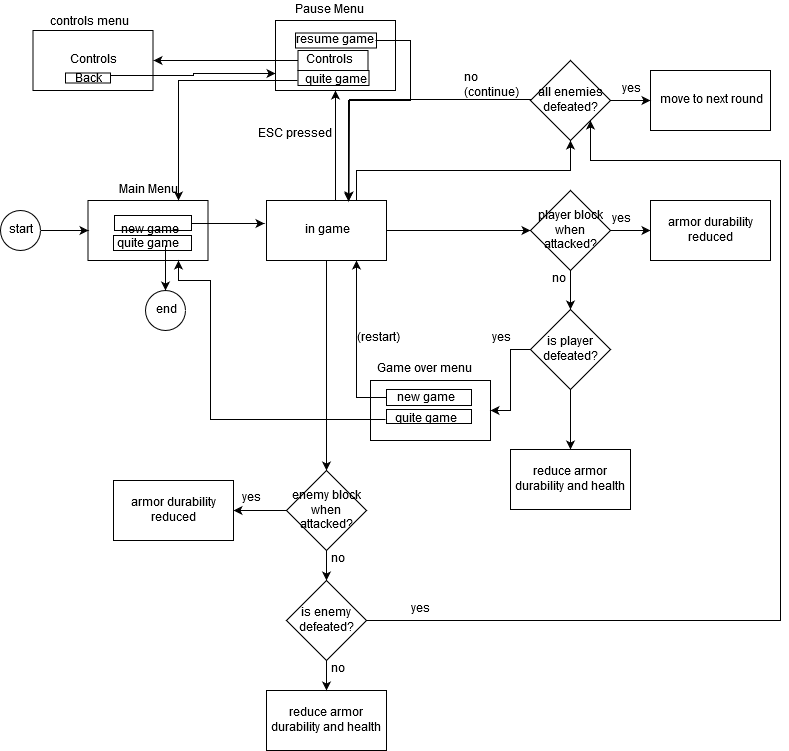
- 3d models of equipment such as sword, shield, and armor

- 3d rendering of the entire world the player can explore

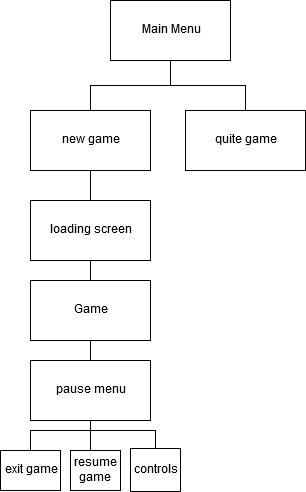
- 3d model of character the player plays as

- 3d model of Enemies

**Flow Charts/Process Flows:**



**Sitemap Diagram:**

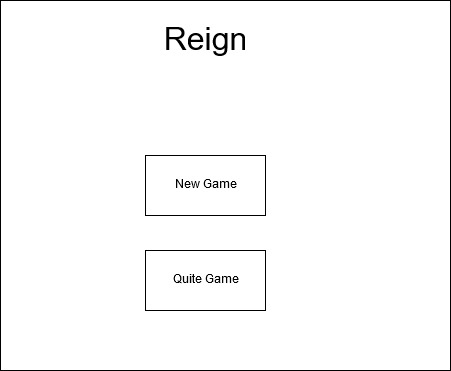


**User Interface Diagrams:**

Main Menu:

New game - starts the game

Exit game – closes the application

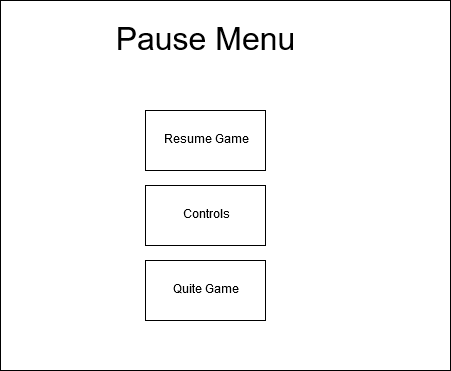


Pause menu:

Continue game – continues the game

Controls – brings up the in-game controls

Exit game – brings you back to the main menu



Controls Menu:

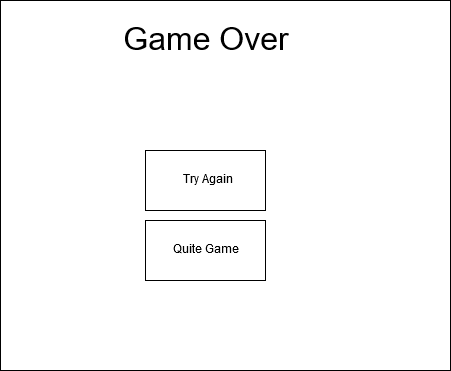
Back – brings you back to the pause menu



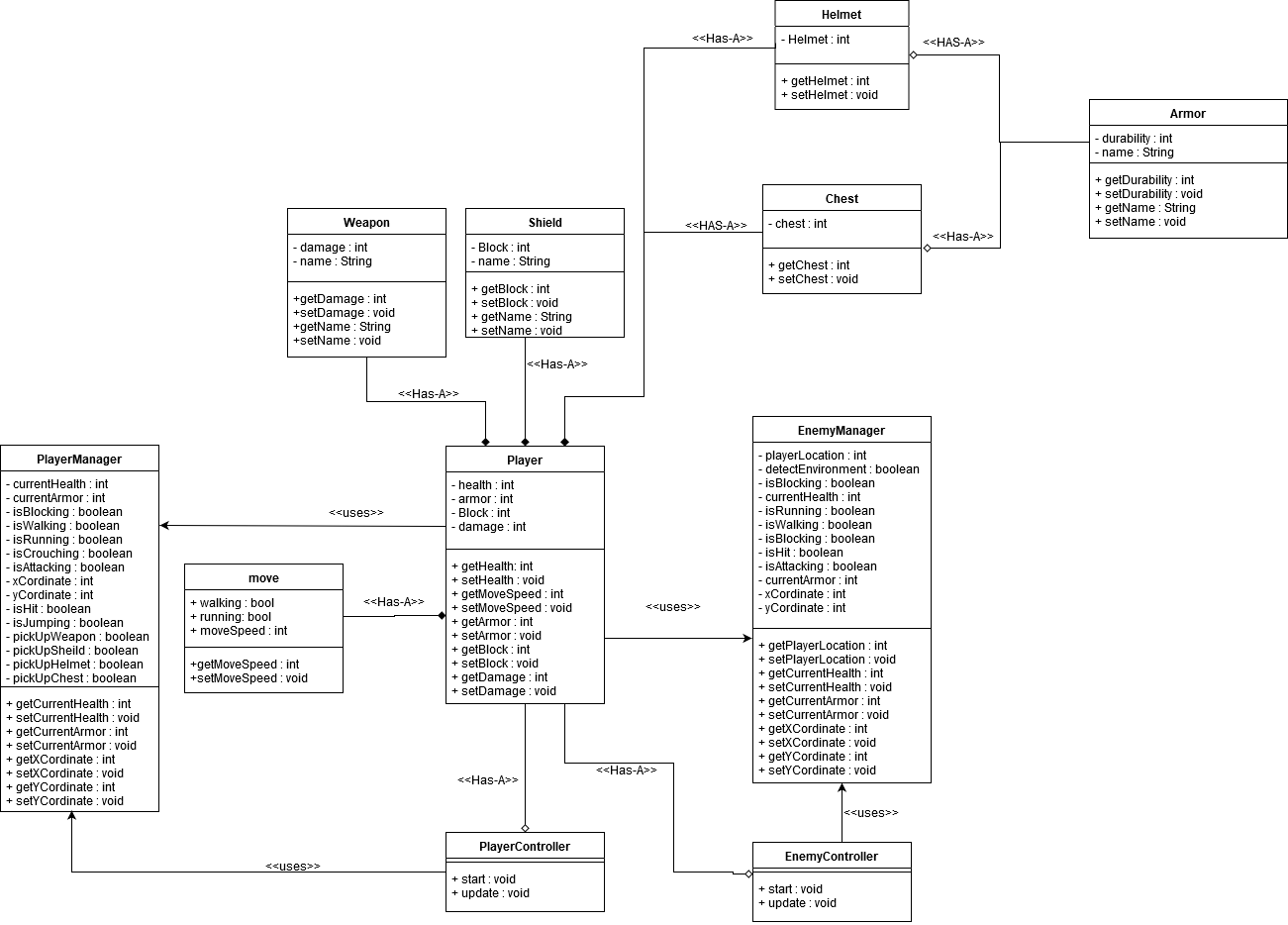
Game Over Menu:

Try Again? - starts the game over from the beginning

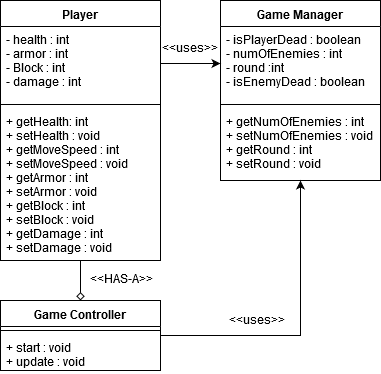
quite Game - takes you to the main menu



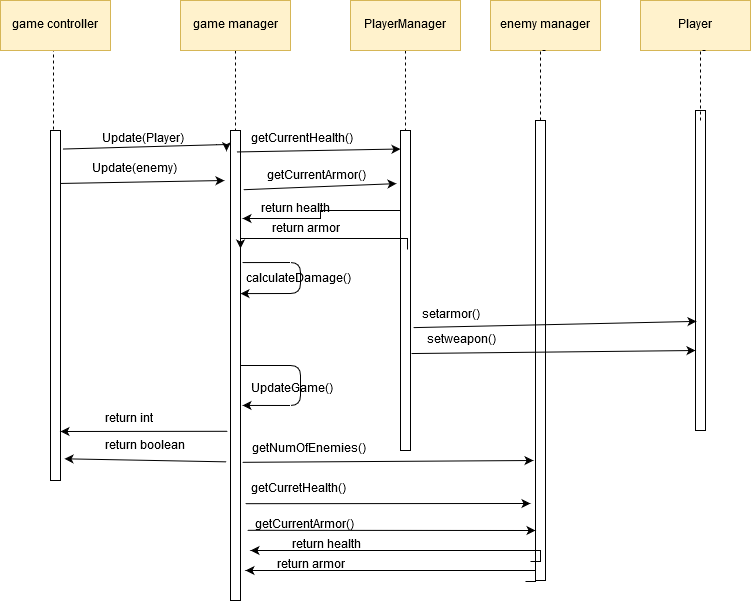
**UML Diagrams:**



The only reason these two are class diagrams are separate is because I did not save the Draw.io file I created the class diagram on.



**Sequence Diagram:**

****

**NFR’s (Security Design, etc.):**

.

**Performance**

* Frame Rates: Reign must run at a frame rate of at least 30 seconds, any lower the user can count the frames and will be annoyed because the game is slow
* Loading Times:Reign must have small loading times, the maximum loading time that will be allowed is about 1 minute.
* Running: Reign must be able to run for hours and still be perfectly stable, it should not go down in quality the longer a user plays.
* Response Time: Users actions in the game should have a rapid response time to their actions, if they click the LMB (left mouse button) to attack their character response has to be under 1 second.

**Usability**

* Navigation: The amount of button presses and clicks a user uses to start and stop the game should be very few with no more than 2 options in the main menu, 3 options in the pause menu, and 2 on the game over scene.
* Controls: The Keyboard controls that the user uses must be completely easy to use.

**Required Resources**

* Folder Size: The initial install folder should try and not be bigger than 8 gigabytes
* Ram Usage: The game shouldn’t never exceed the maximum of 4 gigs of ram

**Platform**

* Windows 10 Home: Reign will run the most efficiently on windows 10, tested with Minimum Specs, (16 gigs Ram, and 6core Intel i7 CPU) (should run on lower end pcs, but its only tested with those specs)

**Deployment**

* Installation: Reign must be easily installed with little to no user action required.
* Uninstalling: Reign must be easily uninstalled just in case someone wants it to be.

**Operational Support Design:**

Task Manager will be used for monitoring CPU usage and GPU usage and Memory usage. The ideal setting is to have as low CPU usage as possible and as much GPU usage as possible.

For a demo game a simple crash report log will do, telling the user that the game crashed because of incompatible hardware, High memory usage, or High CPU usage (may not crash on cpu usage but will slow the game down and can create graphical or mechanical issues)

Appendix A – Technical Issue and Risk Log

1. Use the template to identify and monitor project issues and risks.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Issues and Risk Log | | | | | | | | |
| **Issue or Risk** | **Description** | **Project Impact** | **Action Plan/Resolution** | **Owner** | **Importance** | **Date Entered** | **Date to Review** | **Date Resolved** |
| I/R | What is the issue or risk? | How will this impact scope, schedule & cost? | How do you intend to deal with this issue? | Who manages this issue? |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

Appendix B – References

*List all Project Documentation References*

*List all references using APA style*

Appendix C – External Resources

|  |  |
| --- | --- |
| **GIT URL:** | *The GIT URL (if applicable).* |
| **Hosting URL:** | *The Hosting URL (if applicable).* |