‘OSRS Clone’

CIS 1512

Software Engineering

Software Requirements Specification

Software Project Management & Planning

Professor Hadi Nasser

‘OSRS Clone’ Team:

Kyle

Malik

Mitchell

Laith

Table of Contents

1.0 Introduction

1.1 Goals and Objectives

The goal of our project was to develop an Old School Runescape (OSRS) inspired game. At the beginning of development, we were unsure the extent to which we would be able to recreate the game. So, we started by focusing on the basics: having a moveable character, a world to move around in, a basic combat system, a character-progression system, and interactions with non-playable characters. As we got further into development, we decided to focus more on the foundational elements of the game, and less on the “next-steps” such as having a playable questline and NPC interaction. This is consistent with our original scope of the project in our previous documentation, in which we specified that we only have one programmer in the group, so we planned to ‘flesh out’ the game as much as possible while keeping our goals and scope realistic.

1.2 Agile Methods

A blend of agile methods were applied in the development process. Sprints were used to deliver increments by functional requirements, with each sprint being aimed to deliver one. Incremental planning was applied by thinking of functional requirements as user stories, though we chose to use longer sprints, so each increment was not intended to contain multiple user stories unless it was convenient. Continuous integration was used via a development tool called Unity, which allowed us to test and run the build immediately after implementing a feature.So as to not overcomplicate an already complex project, we used simple design to ensure we stuck to the plan.

1.3 Functional Requirements

- The player will move by clicking.
- The player will be able to interact with the world (e.g. enemies) by clicking.
- There will be an in-game camera that displays to the user what is happening in the game.
- Implementation of a basic combat system.
- The ability for the user to have their character sprint.
- Health bar will automatically regenerate while out of combat.
- The player will be able to level up their character by gaining experience points through gameplay.
- In-game display that allows players to see health bars and character level.
- Damage numbers that pop up on the character and enemies in combat.
- A minimap that shows the player location from a birds-eye view.

- A text tutorial with instructions on how to play the game.
- An application to start the game.

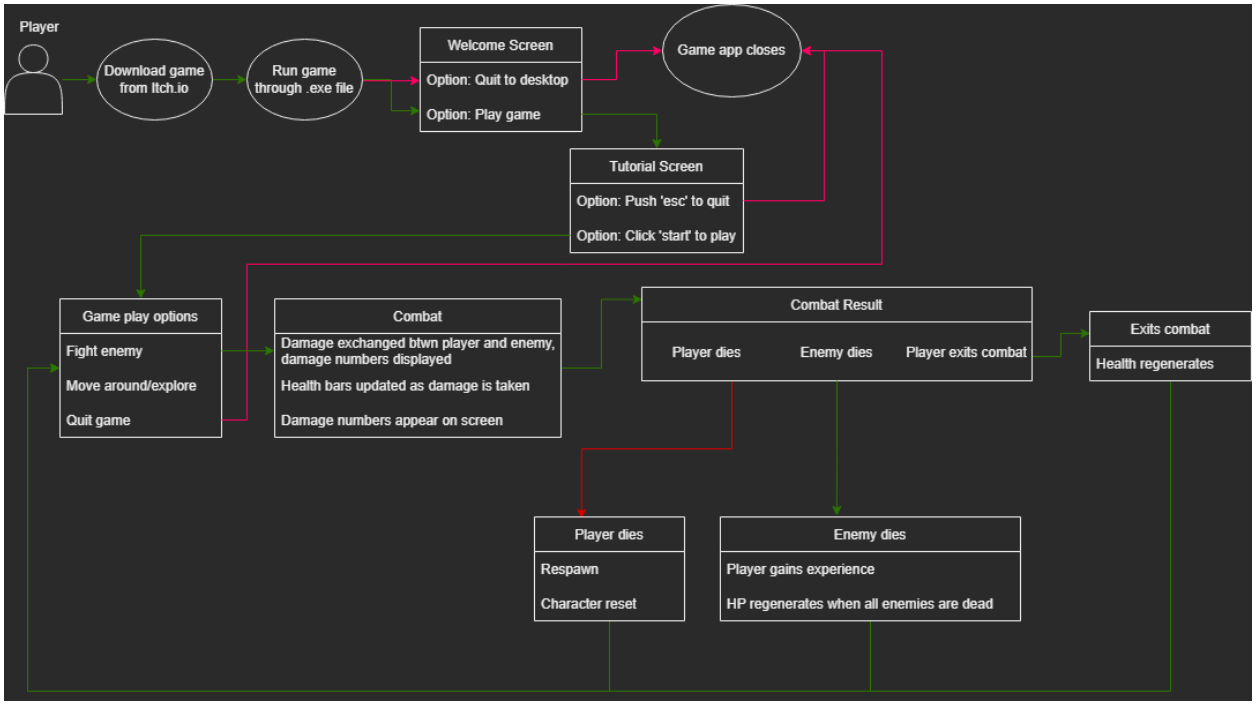
1.4 Non-Functional Requirements

- The game should run without crashing.
- There should be little to no input or visual lag. If any lag exists, it should not notably impact user experience.
- Navigating the menu system should be intuitive.
- The game should be able to be built upon after the completion of the class project.
- The game will be available to download from itch.io

1.5 Inverse Requirements

- This game will not contain a full campaign.
- This game will not have a multiplayer component.

2.0 Usage Scenario



3.0 Software Interface Description

3.1 External machine and system interfaces

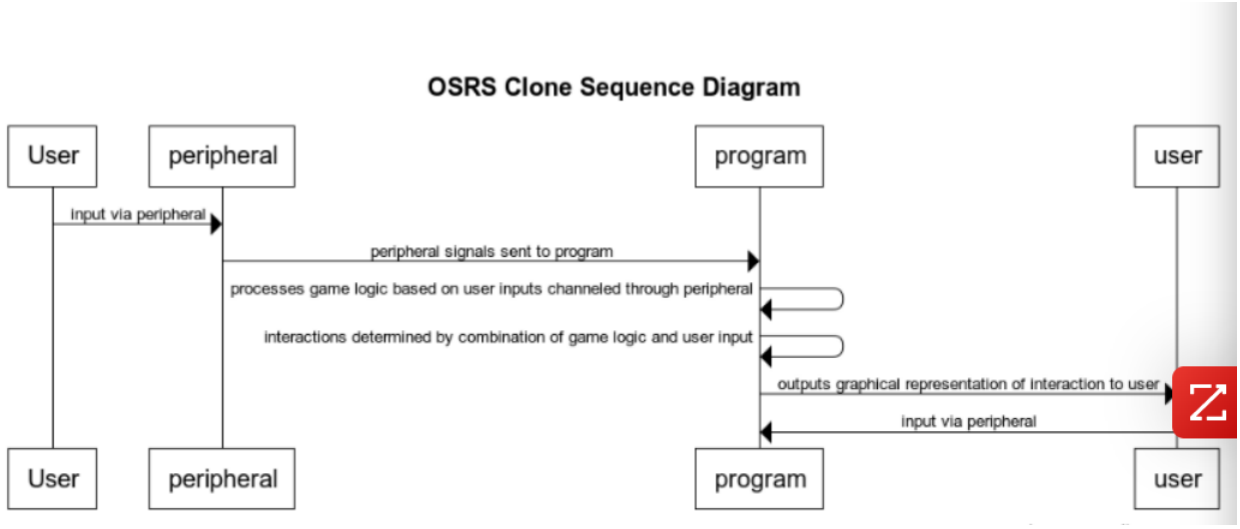
As a single player game, it will not have any direct interfaces to other machines or systems. However, the game will be available on www.itch.io for download, which requires the internet.

3.2 Human Interfaces

Multiple human interface components will be in use, primarily in the form of PC peripherals.

- A monitor will be in use to view the game world and GUI.
- A mouse and keyboard will be in use to register user input.
- Speakers, headphones, soundbar, etc. will be used to play the noises the game makes out to the player.

4.0 Behavioral Model/Data Model and Description

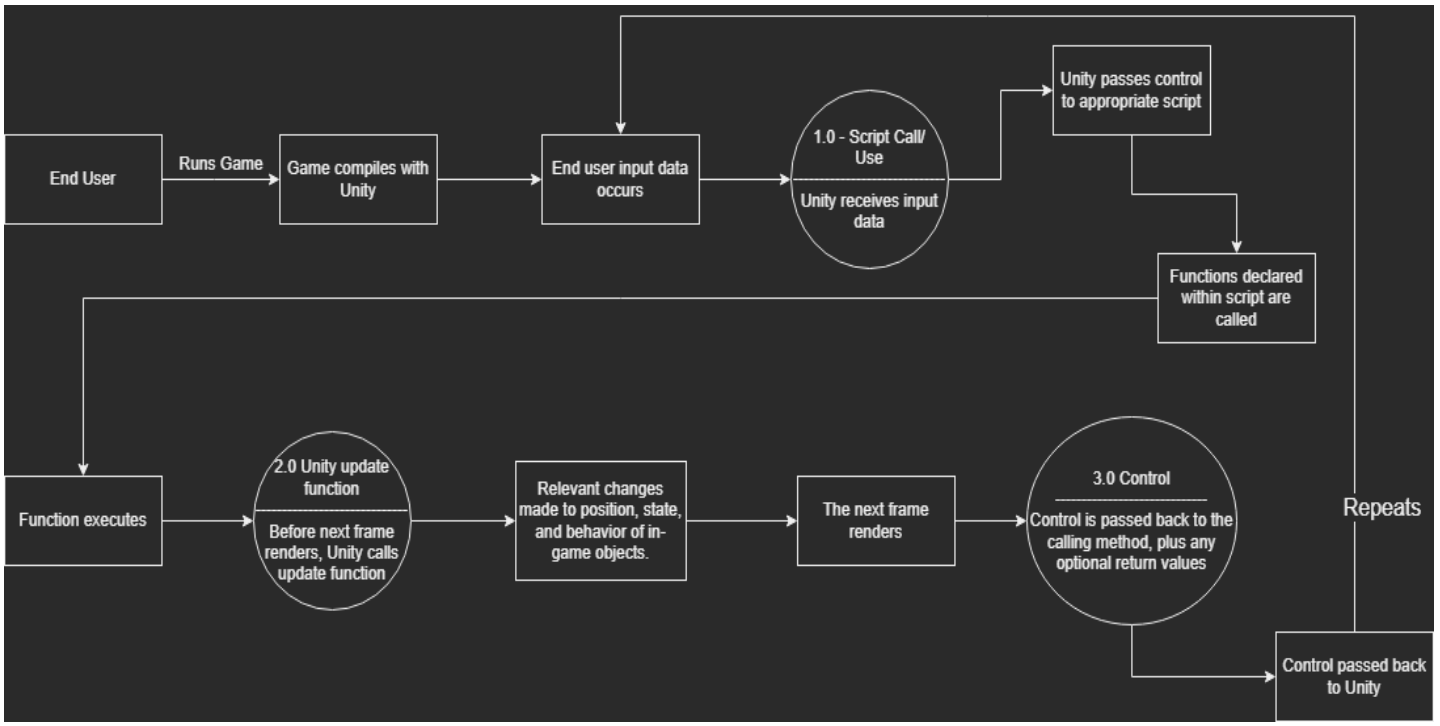


5.0 Data Design

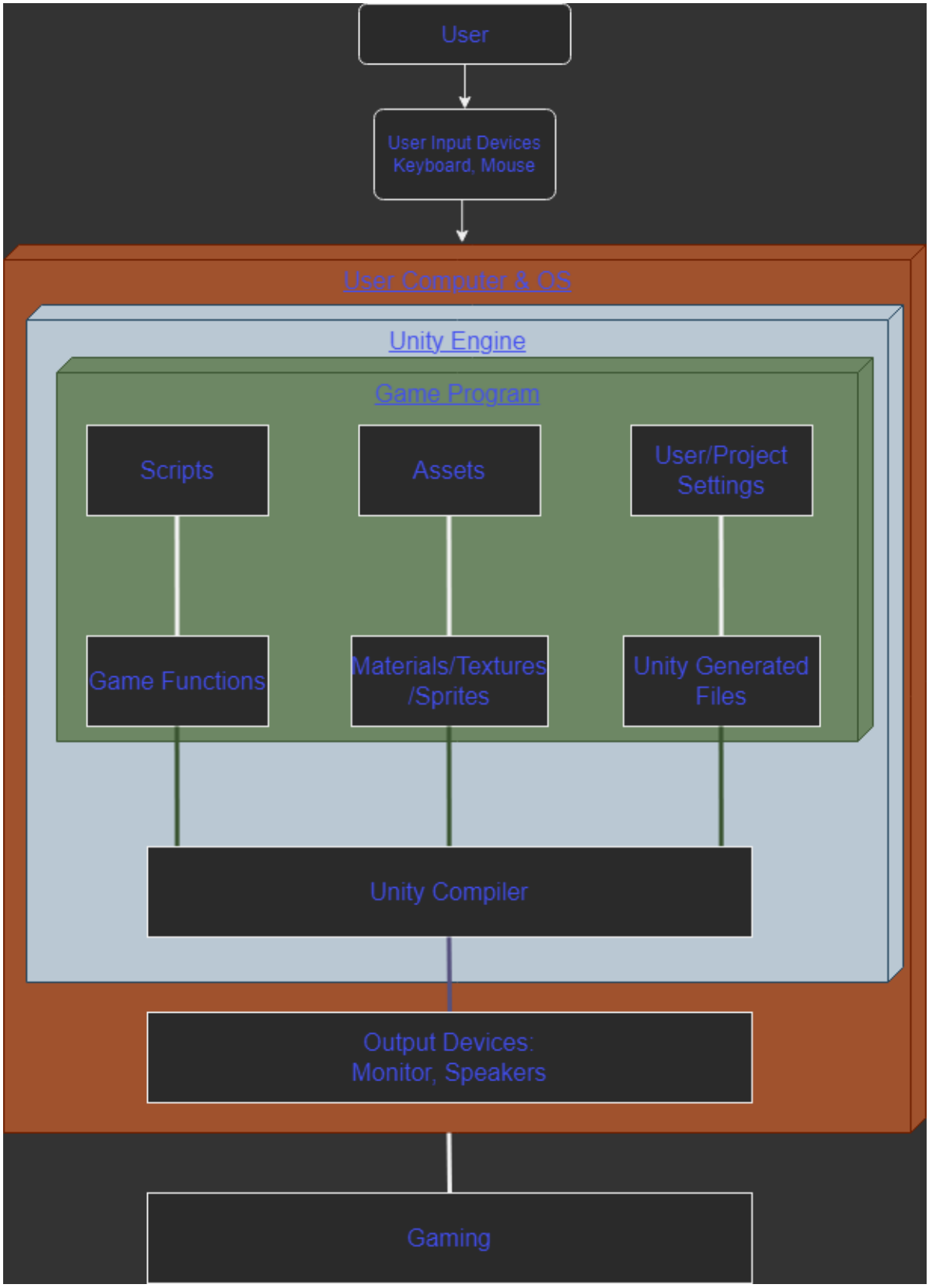
5.1 Data Structures

We are using arrays in our level-up system. The array is used to store how much experience the user needs per level, and how much accuracy they have per level. With this, there are also a few dictionaries and enumerations in our XP system. Our camera uses Vector3, which is a vector in UnityEngine that represents 3d vectors and points. The Vector3 stores camera coordinate positions. Our code does not use other data structures, but does include things the Unity engine uses to engage with data such as YAML and JSON.

5.2 Data Flow Diagram



6.0 Architectural and Component-Level Design



7.0 User Interface Design

There is limited UI in the game as of the project due date. It has a mini-map, sprinting and level indicators, and a couple menu options at the welcome screen. They are stylized as boxes with a gray background and white text with a font reminiscent of OSRS.

8.0 Restrictions, Limitations, and Constraints

The group had multiple limiting factors to the development process. Game development was inhibited by the group having only one programmer who was developing his first game. The overall group can help at certain points in the development process, such as by testing or providing feedback, but the overall group is not experienced which likely caused there to be issues that the group members are currently unaware of. The time restrictions on this project also played a role, and a section of the documentation will be dedicated to discussing what could be done if more time were available.

9.0 Validation and Verification

Test Case #	**Description**	**Input**	**Expected Output**	**Actual Output**	**Pass/Fail**
1	Program runs	Open game exe file	The game opens without error	same	pass
2	The user can move while clicking	Click within the game	The character moves to the inputted position	same	pass
3	User can interact with in-game objects by clicking	Click on an enemy (the only interactable object in this release)	The correct interaction occurs when clicking (e.g. attacking enemy)	same	pass
4	The in-game	Start and play the	The camera	same	pass

	camera will follow the character, displaying the game.	game and move around, rotating the camera to different angles, and moving the character and camera into different positions.	will follow the character and will respond to user input to adjust its position.		
5	Combat is engaged by the player after left-clicking an enemy	Attempting to engage combat with left-click and different inputs, such as right click.	Only left click will engage combat	same	pass
6	In combat, damage is exchanged	Engage in combat	The health bars going down will reflect damage is exchanged	same	pass
7	Damage numbers appear on the characters to show how much they took	Engage in combat and check if numbers appear and are reflected by accurate changes in health bar	The features should work	same	pass
8	The user will gain EXP after ending combat	Engage in combat and see if the # of EXP points gained appears on the character after winning a fight.	The EXP number should appear	same	pass
9	The user will level up through experience points	Engage in combat to see if a level up occurs	The level up should occur when enough exp is gained	The player leveled up, and did not level up in cases where little exp was gained (e.g. 1 exp)	pass
10	Leaving combat	Leave combat after	Health will	same	pass

	will begin health regeneration	being damaged to check if health regenerates	regenerate out of combat, and not while in combat		
11	When the enemy reaches 0 health, they will die	Engage in combat and kill an enemy	All features will work as planned	Enemy died at 0 health, the player gained exp and leveled up as reflected by UI, combat end is signaled by health regeneration	pass
12	When the player reaches 0 health, the player should respawn	Engage in combat and die	The player will die, respawn, and the game will continue as normal	The player died, respawned, but the mini-map, sprint, and level indicators disappeared. Enemies no longer died when their HP reached 0. Damage numbers not displayed in combat	Failed -Issue resolved in update, passed
13	In-game display allows players to see relevant information	Open the game and check if the UI loads in various scenarios	The UI should work properly in all scenarios after last update	The UI continued to work properly after the update	pass
14	The mini-map shows the player	Open the game and check if the	The mini-map	The mini-map	pass

	location from a birds eye view	mini-map properly shows and tracks player location	should work properly	works properly	
15	A text tutorial will load after the welcome screen	Start a new game and see if the tutorial appears and can be passed by clicking start or play	The tutorial will work properly	same	pass
16	Does the game run on minimal hardware?	Open the game on a low quality computer and check for performance issues.	The game should run fine due to low complexity	The game ran at an acceptable quality (loaded quickly, gameplay worked, fps did not create lag or stuttering)	passed
17	When player health reaches a value of 0, does the *Health.cs* class work with the *Death.cs* class in order to execute the player death sequence?	Use Debug.Log();in the Death.cs class in order to verify that the onDeath(); function is called successfully from the Health.cs script	The player death sequence successfully executes (splash screen appears, UI elements are temporarily removed, player's model disappears from game)	The player death sequence successfully executes	pass
18	When the player's weapon has damage of 20, does the *Attack.cs* script successfully attack enemies	When a player sends an attack to the enemy, verify that: 1.The damage value of 20 is successfully	Enemy health decreases by 20 when a successful attack is sent out	same	pass

	with 20 damage?	passed through to the enemy 2.The enemy takes 20 damage By using Debug.Log(); to print the values of the damage to the console, checking if they are the same and that the enemy has taken 20 damage.			
19	Does the *CamOrbit.cs* script successfully track the amount of rotation that the camera is doing while the game is running? (in degrees)	Using Debug.Log(); in the *CamOrbit.cs* script in order to check if the tracked rotation value matches the real rotation value in the inspector?	When the camera orbits the player by x degrees, the script reflects x degrees	same	pass
20	Do the XPParent.cs and child classes successfully calculate the desired accuracy values per level based on the function l(x)	Using Debug.Log(); in order to check if the array of desired accuracy values per level corresponds to the values of l(x)	When the player levels up, their accuracy is the value determined by XPParent.cs using l(x)	same	pass

9.1 Test Results Table

The above table reflects the results of black-box testing done by Laith and code testing done by Kyle. Without detailed knowledge of the program's code, Laith was able to test the program without bias as they would see the game through the lens of an end-user, rather than a programmer. A bug was found that was not found during Kyle's test-driven development process and was quickly resolved. Kyle's testing was done through unit tests of his scripts.Through the use of tools that check results objectively, like Debug.Log(), and expected versus actual outcomes, bias was able to be removed from

the test. Throughout development, Kyle would test code after implementing features by running the game and checking to see if things worked or changed.

9.2 Test Case Categories

Other forms of testing were used or coincided with the play-tests, such as:

- Functionality/Validation testing:
    - Make sure all criteria as specified in the use case diagram and functional requirements work properly.
        - User interface testing: Make sure all UI work properly and reflect the intended purpose.
    - Performance tests: Make sure the game runs without lag at a level sufficient for player enjoyment. Run the game on multiple computers.
    - Software/QA tests: Make sure there are little/zero bugs in the game that outright affect player enjoyment.
    - Black box testing: Tests can include play-testing for bugs, ensuring positive user experience, and general quality assurance. Play-testing can also be a part of the agile method of testing, where the user/customer is part of the development team. Feedback from play-tests was given to the coder of the group for improvement.
- White box testing: Programming members can review code and create test cases for it. Examples can include testing various aspects of the code to create errors or “break” and see how the software reacts.

9.3 Expected Software Response

At this stage in development, the software should run without issues. Due to the use of test-driven development, many errors were sorted out before the project reached its final stage. Therefore, the final tests should all pass. A bug was found, but was able to be resolved quickly and was the result of an oversight.

9.4 Performance bounds

No app crashes are to be expected unless the user lies outside of Unity System Requirements. Having a sufficient CPU, GPU, and enough RAM/storage needed for low end 3d gaming will also prevent unwanted crashes from occurrence.

10.0 Testing Issues

The game was able to be tested to check if its features are working in a manner suitable for the final deadline, however the thoroughness of the testing could be improved. For one, more advanced testing styles that more rigorously test the code were not used due

to issues with experience and workload. In the program's current state, running the game to check for errors is suitable, but if it were to be built upon later, issues could arise due to a lack of thorough analysis on the code architecture and the manner in which features were implemented.

11.0 Project Estimates

11.1 COCOMO

Software Project	a_b	b_b	c_b	d_b	Select
Organic	2.4	1.05	2.5	0.38	◉
Semi-detached	3.0	1.12	2.5	0.35	○
Embedded	3.6	1.20	2.5	0.32	○

Calculate Effort and Duration

Effort (E) = $a_b(KLOC)^{b}_{b}$ = 5.45 **Duration (D) = $c_b(E)^{d}_{b}$ =** 4.76

12.0 Risk Management

Risk Name	**Risk Description**	**Impact (High, Medium, Low)**	**Probability of occurrence (High, Medium, Low)**	**(Contingency) Plan of Execution**
Time constraints	Team members have their own schedule, limited time due to the project deadline.	High	Medium	Distributing workload accordingly, planning ahead, strong communication, motivating each other
Hardware malfunction	Unforeseen technological malfunctions	High	Low	It would depend on the hardware malfunction, but group members would have to use other computers if needed.

Skill assessment	Not all members of the team can do the same tasks. Only some can program.	Medium	Low	Addressing each members' strengths & weaknesses and assigning them applicable tasks. High level of motivation from key members. Kyle is the one who really wanted to make a video game, and has been excellent and progressing in the development on his own, despite that it is mostly him working on the programming for now.
Specification delays	Unable to deliver a sprint containing a functional requirement on time. Can occur due to underestimation of task difficulty.	Medium	High	Being unable to deliver a functional requirement on schedule can heavily impact the future of the project. In this case, the schedule can be adjusted, and requirements can also be changed if needed.
Requirements change	Due to use of agile method, and the complexity of the project, it is likely that the requirements	High	Low	This is a low impact risk because we are adaptable and capable of conforming to whatever

	change throughout the development process.			changes are necessary to continue development. Our strong communication and planning skills will allow us to create a plan to overcome obstacles and implement changes when necessary.

13.0 Maintenance

13.1 Potential Enhancements

If we had more time to work on our project, we would focus on creating a more fully fleshed out game. We currently do not have any quests, and our game's lore is made to be more generalized, and is not even included in the playable game. In addition, we do not have many features that Old School Runescape is known for, such as skills or multiplayer, and we do not have audio. Many gamers feel nostalgic for OSRS because of the multiplayer aspect, the music, and grinding for their skills. To improve on these things, we would develop a more in-depth story and world history for our game, including planning out desired themes for the game to build a more engaging quest-based storyline around it. We would also implement any features we saw fit from OSRS as well as any of our own features that we would come up with in the development process. For instance, we had to take a step back from some features we wanted to add early in development, such as multiple weapon choices, and these core mechanics of gameplay would be definitely the first to be added in a future update.

In addition, the in-game art could use a lot of work. The current version of the game uses many placeholders or stock/free-use textures and materials. If given more time, these would be refined to a more unique and/or nostalgic style akin to OSRS. A more detailed menu system would be created, as well as the ability to adjust settings and save the game. The current game does not save, and settings cannot be adjusted.

13.2 Refactoring

The code for the project could be improved in numerous ways. Firstly, comments need to be added throughout the code. Currently, there are little comments being used, which can be a detriment to the project as the scale and complexity increases, especially if new people were to join as programmers.

This project taught us a lot of valuable lessons in software development. Firstly, at the beginning of the project, the exact scale and complexity was hard to be estimated, and so the initial plan was too ambitious for the time given and had to be scaled back. It is not better understood how to plan and to aim low with the goal of overshooting, allowing more time at the end for extra features and debugging.

The code does not suffer from ambiguous naming or duplicate code, and the coupling is very low. There are definitely a few scripts within our game that are knowingly over-engineered given the generosity of our scope originally. These scripts would/should be refactored to simplify them. This would especially improve performance on all types of machines, allowing room for the project to be scaled. Though these scripts generate no errors on the functionality of our game, it would be something to work on for future iterations. The code does not suffer from ambiguous naming or duplicate code, and the coupling is very low.

13.3 Lessons Learned

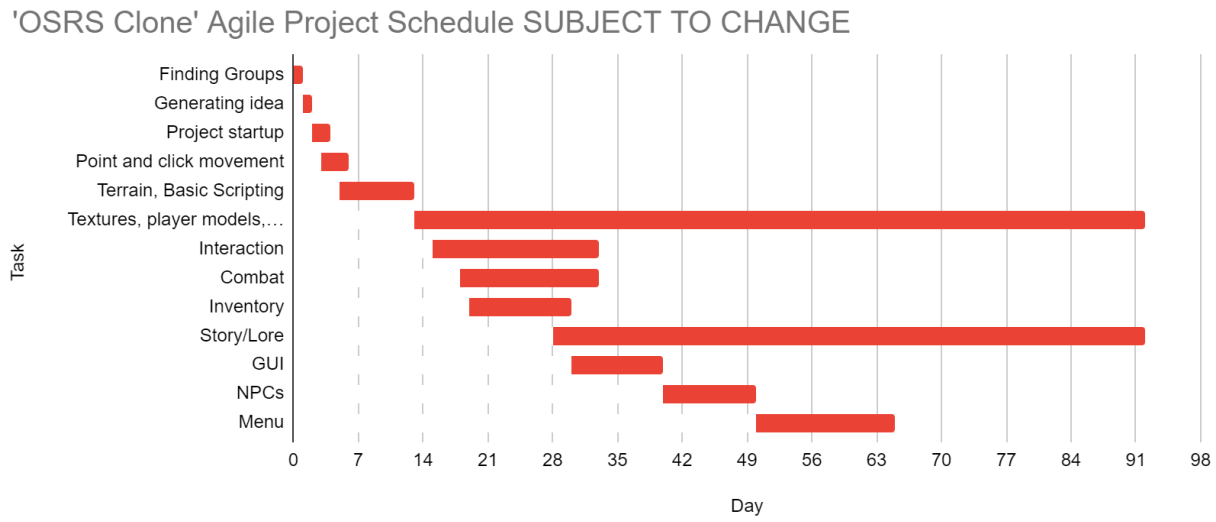
At a programming standpoint, we tried to keep this game modular, meaning that every script builds on top of one another without having dependency issues. We successfully accomplished this, and learned a lot about what is important to keep in separate scripts when making a larger piece of software. We tried to follow the SOLID principles of object-oriented design as closely as possible, and doing so we learned much of what is important in making single-use, single-purpose scripts for a larger software project.

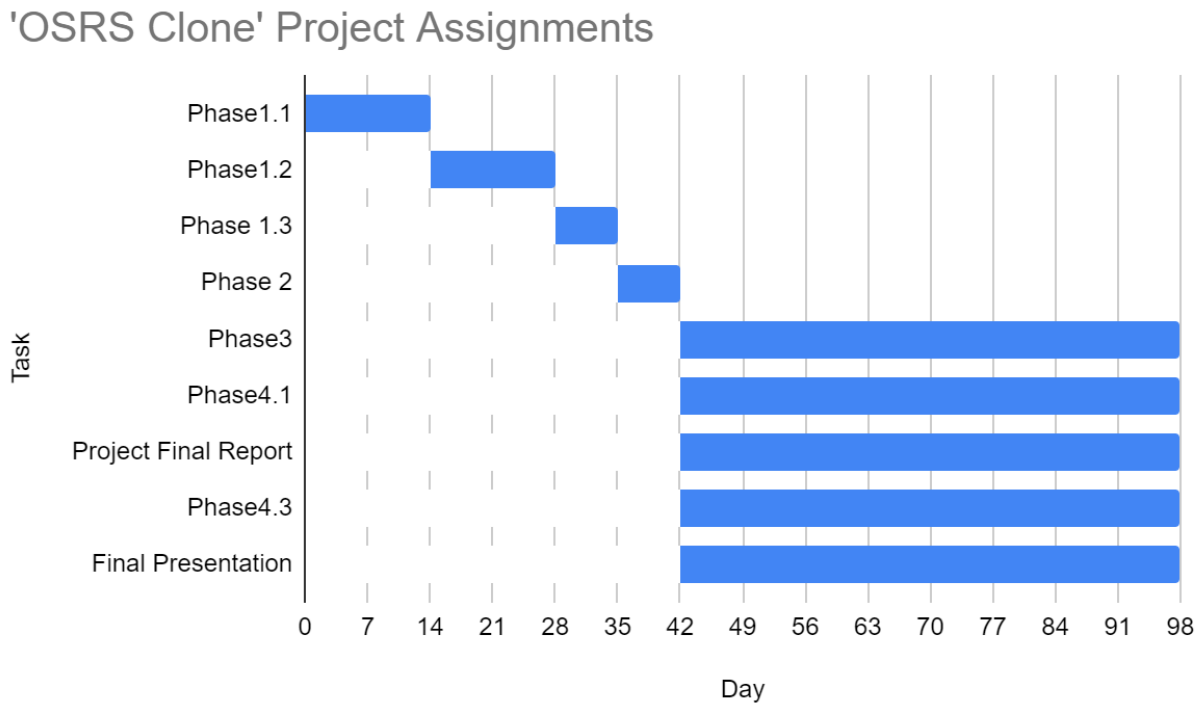
Going back, originally our scope for this project was much too big in terms of the relatively short time period (most games are worked on for years before they are released) we were given. We should have evaluated the amount of time that we had, and should have considered that game development is some of the most time-intensive work a software engineer can do.

Finally, all of the coding for the project was done by Kyle, so he didn't have to worry about anyone else reading over it. Because of this, many of the scripts used in the project lacked sufficient commenting that is necessary in a piece of software worked on by many people. While coding this project, Kyle intended to go into older scripts and

add comments so other people could understand the scripts, but this wasn't done and we learned that comments should be added while someone codes, not after. This issue didn't affect the outcome of our project, and most-likely improved the speed of development, but it would have been nice if all group members understood the purpose of the scripts written into the game.

14.0 Project Schedule





15.0 Staff Organization

Kyle

Kyle will be doing most of the programming for the game using the Unity game engine and c#. He is also helping with group communication and whatever problems are left outright. Kyle is also the subject matter expert.

Laith

Laith will be creating narrative elements, such as the story and world-building, as well as assisting with any non-programming aspects of the project. Laith will also function as a theoretical customer/end-user and help give feedback on sprints from that perspective.

Malik

Minimal experience with Java & python, will mostly be providing assistance with non in game graphical assets, and documentation. Malik will also function as a theoretical customer/end-user.

Mitchel

Mitch will be helping with graphic design for things such as the world map. Further, he is available to assist with programming, coordination, and documentation.

Business Owner/Stakeholder: Is the instructor, Professor Hadi.

16.0 Appendices

16.1 Configuration Management

We have Github for version control. We also use discord for communication and meetings. We use Google Docs to collaborate on documentation and keep project information stored. We have a meeting at least once per week, which functions as our daily standup.

16.2 Screenshots

16.2a) Example of Malik's graphic design in future versions. Also shows current minimap and UI.

16.2b) Malik icon concepts

16.2c) Malik weapon concepts

16.2d) Gameplay screenshot of combat

16.3 Minimal Hardware Requirements for the Developer

Below are the minimum requirements for the developer, which is the Unity Editor System Requirements.

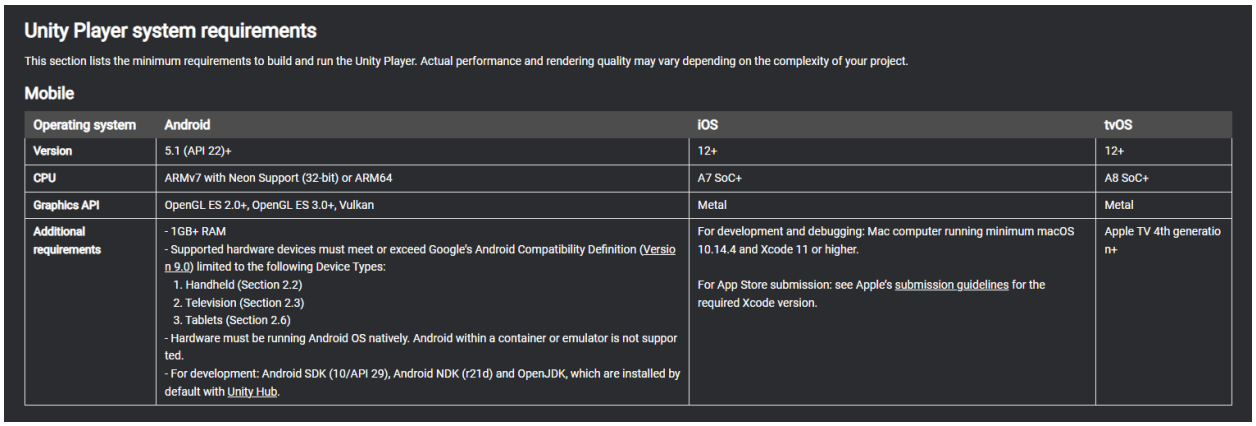
Unity Editor system requirements

This section lists the minimum requirements to run the Unity Editor. Actual performance and rendering quality may vary depending on the complexity of your project.

Minimum requirements	Windows	macOS	Linux (Support in Preview)
Operating system version	Windows 7 (SP1+) and Windows 10, 64-bit versions only.	High Sierra 10.13+	Ubuntu 16.04, Ubuntu 18.04, and CentOS 7
CPU	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support
Graphics API	DX10, DX11, and DX12-capable GPUs	Metal-capable Intel and AMD GPUs	OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs.
Additional requirements	Hardware vendor officially supported drivers	Apple officially supported drivers	Gnome desktop environment running on top of X11 windowing system, Nvidia official proprietary graphics driver or AMD Mesa graphics driver. Other configuration and user environment as provided stock with the supported distribution (Kernel, Compositor, etc.)
	For all operating systems, the Unity Editor is supported on workstations or laptop form factors, running without emulation, container or compatibility layer.		

16.4 Minimal Hardware Requirements for the User

Below are the minimum requirements for the user, which are the Unity Player system requirements.



Unity Player system requirements

This section lists the minimum requirements to build and run the Unity Player. Actual performance and rendering quality may vary depending on the complexity of your project.

Mobile

Operating system	Android	iOS	tvOS
Version	5.1 (API 22)+	12+	12+
CPU	ARMv7 with Neon Support (32-bit) or ARM64	A7 SoC+	A8 SoC+
Graphics API	OpenGL ES 2.0+, OpenGL ES 3.0+, Vulkan	Metal	Metal
Additional requirements	- 1GB+ RAM - Supported hardware devices must meet or exceed Google's Android Compatibility Definition (Version 9.0) limited to the following Device Types: 1. Handheld (Section 2.2) 2. Television (Section 2.3) 3. Tablets (Section 2.6) - Hardware must be running Android OS natively. Android within a container or emulator is not supported. - For development: Android SDK (10/API 29), Android NDK (r21d) and OpenJDK, which are installed by default with Unity Hub.	For development and debugging: Mac computer running minimum macOS 10.14.4 and Xcode 11 or higher. For App Store submission: see Apple's submission guidelines for the required Xcode version.	Apple TV 4th generation+

16.5 Packaging and Installation

Program can be downloaded from itch.io and does not need an installation. It can run when unzipped through the .exe file.

16.6 References/Supplementary Information

System requirements taken from Unity website:
https://docs.unity.cn/ru/2021.1/Manual/system-requirements.html

Data design diagram information received from Unity documentation:
https://docs.unity3d.com/Manual/EventFunctions.html

Link to download the game:
https://kylekrause3.itch.io/osrs-clone

Lower quality laptop used in testing:
https://support.hp.com/au-en/document/c06074694
Note - The laptop updated itself to Windows 11 but was not made for this OS, and has deteriorated in quality and speed since then, running very poorly on basic web searches.